Department of Defense
High Performance Computing Modernization Program

Proceedings of the High Performance Computing Modernization Program

Users Group Conference 2011

User Advocacy Group (UAG)
HPCMPO Outreach Team
www.hpc.mil

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2011		**00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Proceedings of The High Performance Computing Modernization Program Users Group Conference 2011	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **High Performance Computing Modernization Program Office (,10501 Furnace Road,Lorton,VA,22079**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Proceedings of the High Performance Computing Modernization Program,Portland, OR, June 20-23, 2011

14. ABSTRACT
The 2011 DoD High Performance Computer Modernization Program (HPCMP) Users Group Conference (UGC) was held June 21-23, 2011 in Portland, OR. This annual conference enables the Department of Defense (DoD) science and technology (S&T) and test and evaluation (T&E) communities to share their computational science and engineering results with their colleagues and other members of the DoD community. The conference brings together a community of scientists and engineers with common interests, encourages the sharing of techniques and methods, and allows the users to mix with the staff from the computer centers and HPCMP office. The users are able to gain an appreciation of the technical breadth of the entire community and interact with their scientific and engineering colleagues.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**672**	

Proceedings

2011 DoD High Performance Computing Modernzation Program Users Group Conference HPCMP UGC 2011

Portland, OR

June 20–23, 2011

Proceedings

2011 DoD High Performance Computing Modernzation Program Users Group Conference HPCMP UGC 2011

Portland, OR

June 20–23, 2011

The papers in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and, in the interests of timely dissemination, are published as presented and without change.

ISBN # 978-0-9839566-0-0
Additional copies may be ordered from:

HPCMPO
10501 Furnace Road
Suite 101
Lorton, VA 22079-2632.
ATTN: Outreach

Editorial production by Denise O'Donnell and Lisa Powell
Cover art production by Lisa Powell
Printed in the United Stated of America by Harris Lithographics, Inc.

High Performance Computing Modernization Program Office (HPCMP)

DoD High Performance Computing Modernization Program

Users Group Conference 2011

HPCMP UGC 2011

Table of Contents

3. Computational Chemistry and Materials Science (CCM)

7. Software Performance and Performance Modeling

8. Computational Methods

Editor's Preface

Proceedings of the 2011 DoD HPCMP Users Group Conference

The 2011 DoD High Performance Computer Modernization Program (HPCMP) Users Group Conference (UGC) was held June 21–23, 2011 in Portland, OR. This annual conference enables the Department of Defense (DoD) science and technology (S&T) and test and evaluation (T&E) communities to share their computational science and engineering results with their colleagues and other members of the DoD community. The conference brings together a community of scientists and engineers with common interests, encourages the sharing of techniques and methods, and allows the users to mix with the staff from the computer centers and HPCMP office. The users are able to gain an appreciation of the technical breadth of the entire community and interact with their scientific and engineering colleagues.

The UGC was organized by the Program's User Advocacy Group (UAG). That group was led by the UAG chair Stephen Ketcham (Army), and consisted of the technical program chair was Marshall McBride (Army); and the technical program co-chair was Soumya Patnaik (Air Force). The tutorials chair was Matthew Grismer (Air Force). Deborah Schwartz and the HPCMP staff (Leah Glick, Jakeya Morgan, Denise O'Donnell, and Lisa Powell) organized the conference arrangements. Larry Davis was the HPCMPO liaison and Cathy McDonald is the UAG Executive Secretary. Lisa Powell and Denise O'Donnell edited the UGC post-Proceedings.

The 2011 Conference opened on Tuesday with a welcome by Dr. Stephen Ketcham, the conference chair. Mr. Cray Henry then described the "State-of-the-HPCMP". The Program is undergoing a number of major changes, including transfer to the Army on October 1, 2011; a budget reduction of $30M for FY2012; and the closure of the Arctic Region Supercomputer Center (ARSC) and Software Protection Institute (SPI). The Program was reviewed by a panel of experts led by Institutes for Defense Analysis. The review concluded that the Program was doing a good job of providing computing resources to the DoD S&T and T&E communities, but needed to be more proactive in fostering the use of supercomputing by the DoD S&T and T&E communities. Three major Cray XE6 systems started operation since the last UGC (Air Force Research Laboratory (AFRL) Raptor with 43,712 cores, Engineering Research and Development Center (ERDC) Garnet with 20,244 cores, and ARSC with 11,648 cores—moved to ERDC before ARSC shut down). Three machines with a total of 15,664 cores were retired. In addition, five new Dedicated HPC Project Investments (DHPIs) were awarded for FY11/12. The Program has established a new allocation category aimed primarily at engineering applications and code development termed Dedicated Support Partitions, in which time is allocated to groups which need to emphasize minimizing the time to solution to meet design and code release deadlines. The Program also has established an HPC Enhanced User Environment (HEUE) with a utility server (~1,760 cores) and local storage at each DoD Supercomputing Resource Center (DSRC) to facilitate problem set-up, production runs, local storage of the results, and analysis of the problem results, including visualization by the remote user. To facilitate the use of the DSRC computers by remote users who often aren't allowed to install anything but browsers and Microsoft Office on their local computers, the DSRCs and Software Application Systems have launched a "portal" program to develop the capability for users to setup a problem, run it, and analyze and visualize the results on the DSRCs through a browser with no local client software. DREN (Defense Research and Engineering Network) continues to provide outstanding network support to the DoD S&T and T&E communities, and to a number of other DoD and federal agencies as well. The Program has established a single security enclave that encompasses all the DSRCs and DREN, which vastly simplifies and improves the connectivity of the DSRCs with the user communities. The Software Application Support (SAS) Group has started a new Institute led by the Army Research Laboratory for "Blast Protection for Platforms and Personnel". The Productivity, Enhancement, Technology Transfer and Training (PETTT) Program provided assistance to

the S&E and T&E user communities that enabled a significant number of users to accomplish their mission on the DSRCs. The Computational Research and Engineering Acquisition Tools and Environments (CREATE) program released eight new codes in FY2010 as part of an annual release cycle, and expects to release two additions codes for rapid conceptual design of ships and airplanes by the end of calendar year 2011 for a total of ten codes. A description of their code capabilities by CREATE team to the US Defense Industry at two meetings sponsored by the National Defense Industrial Association (NDIA) was met by an enthusiastic response, and interest by the US defense industry in using the codes to solve air vehicle, ship and RF antenna design problems. Cray was followed by Dr. Roger Strawn who described the progress that the (HPC Institute for Advanced Rotorcraft Modeling and Simulation) HI-ARMS Institute/CREATE Team had made developing the high-fidelity rotorcraft design and analysis code, Helios. Through the use of overset grids, high-order accuracy schemes and adaptive mesh refinement, Helios is able to provide the unique capability to accurately predict vortex shedding from rotor tips. This has been the "holy grail" of rotorcraft simulation. Helios is being used by a number of rotorcraft engineering groups in the government and industry.

Dr. Jeffery Holland, the Director of the Army Corps of Engineers' Engineering Research and Development Center, summarized the ERDC program with an address entitled: "From Discovery to Acquisition: Maintaining Technological Superiority for the Nation's Defense Through HCP." Jeff addressed the question on everyone's mind: "What would be the impact of the transition of the HPCMP from the Office of Secretary of Defense to the Army and ERDC?" Jeff stated that his guiding principle would be: "Do no harm" and "improve the ability of the DoD to use HPC to help meet its most important challenges". He described his early involvement in the HPCMP as a CTA (Computational Technology Area) leader for Environmental Quality Modeling and the long-standing modeling program at ERDC, both in civil works and in military missions. He then outlined a vision for how he would both enhance the effectiveness of the HPCMP for helping the DoD meet its mission, and for engaging the national HPC community as partners to address many of the computational challenges. Jeff was followed by Steven Wallach who described the "Brave New World" that HPC code developers and users face. Processor speed is no longer increasing due to the "power wall". Faced with the continued need to improve performance within this constraint, vendors are turning to massive parallelism and heterogeneous processor architectures for performance gains. The burden on code developers will be immense, but there is no choice.

There were five plenary talks on Wednesday. The keynote was given by Dr. Cynthia Dion-Schwartz , the former Director, Information Systems and Cyber Security in the Office of the Assistant Secretary of Defense, Research and Engineering (ASD(R&E)). The HPCMP was one of several organizations that reported to Dr. Dion-Schwartz before she moved to the NSF in September, 2011. She outlined her vision for the HPCMP, which was to enhance DoD S&T programs through the use of high performance computing. She described a number of concrete contributions resulting from the application of engineering design tools such as Kestrel and Helios, research codes in computational chemistry, armor design with blast codes, etc. She also stressed the import role of the DREN for both enhancing the ability of the DoD T&E community to exchange data for testing activities and the ability of the HPCMP customer community to gain high bandwidth access to the DSRCs. Dr. John Spyropoulos, Naval Air Systems Command (NAVAIR), described the work at NAVAIR modeling Exhaust Jet Noise. Accurate treatment of unsteady air flow is a key requirement, and physics-based models for accurate unsteady airflow are beginning to be sufficiently accurate that predictions for reduced noise systems are beginning to be credible for design studies and design decisions.

The first of three speakers on Wednesday afternoon, Dr. Wilfred R. Pinfold, Director Extreme Scale programs, Intel Corporation, described the challenges that face chip manufacturers who are working to improve the performance of computer chips. Intel is following the path of massive parallelism together with special purpose processors (e.g., GPGPUs—General-Purpose Graphics Processing Units). In spite of the challenges of changing a host of software applications to run on these new architectures, it appears that DoD code developers will need to address them. Dr. Joseph Gorski summarized the Navy's interest in and use of hydrodynamics analysis based on HPC in his talk entitled: "Impacting Naval Hydrodynamic Ship Design with HPC. The new

generation of Reynolds-Averaged Navier-Stokes codes is showing a lot of promise for predicting resistance and drag, and sea-keeping and seaway loads for both surface and underwater naval vessels. Joe's talk bridged that gap between the codes of the 1990s and the newer codes such as the CREATE software application for these issues (NavyFOAM). With the use of these codes, the Navy will be able to evaluate the performance of future naval systems before physical prototypes are available and evaluate many more cases than possible with physical prototypes and full-size systems. Dr. Thomas Gielda described his experiences at Whirlpool and several other companies applying physics-based models to engineering design. He described how he was able to build a capability at Whirlpool that enabled Whirlpool to use physics-based design tools to produce designs specifically tailored to meet consumer requirements on six continents. Whirlpool used these tools for designing refrigerators, stoves, outdoor grills, washing machines and dryers, and dishwashers and for specifying the manufacturing processes. Then he described the use of computational fluid dynamics and thermal convection and conduction, to develop vehicle designs that minimize heat transfer between the passenger compartment and the outside for new automobile designs. He described the convincing and compelling advantages that can result from the adoption of physics-based modeling for engineering design and analysis.

The meeting attendance was about the same as previous years' conferences (391 in 2011, 422 in 2010, 446 in 2009, and 451 in 2008). There were 125 technical talks presented and 71 presenters wrote papers—approximately the same as 2010.

The talks presented at the UGC provide a good overview of the unclassified computational science and engineering research and development in the DoD science and technology and test and evaluation community. Those talks that have been described in papers are written to be interesting and accessible to scientists and engineers who possess general scientific backgrounds. The papers cover much of the unclassified scientific research and engineering development technical areas that currently address computational techniques. Computational fluid dynamics (CFD) continues to be one of the major application areas, but computational chemistry is also strong. The general topics are listed in Table 1.

Table 1. Paper Categories in 2009, 2010, and 2011

Topic	2009	2010	2011
Computational Fluid Dynamics (CFD)	17	22	20
Multi-Physics (CFD, CSM, CCM, Plasma Physics...)	5	4	4
Computational Chemistry and Materials Science (CCM)	11	14	14
Climate/Weather/Ocean Modeling and Simulation (CWO)	7	7	3
Signal/Image Processing (SIP) and Sensors; Electronics, Networking, and Systems/C4ISR (ENS) and Testing	7	1	11
Computational Electromagnetics and Acoustics		8	2
Software and Hardware Infrastructure	4	8	10
Software Performance and Performance Modeling	8	5	3
Computational Methods	7	4	4
Total	**72***	**73**	**73**

*includes 6 papers on education

The format of conference allowed time for discussions among the attending scientists and engineers during presentations and during the breaks, at lunch, and after the day's presentations.

The HPCMP leadership, the proceedings editor Ms. Lisa Powell, and I gratefully acknowledge the help of all of the Reviewers—who are listed on a separate page—for helping us edit the papers.

—Douglass E. Post, Chief Scientist, DoD HPCMP

Conference Committee

HPCMP UGC 2011

General Chair

Stephen Ketcham, US Army

Technical Chair

Marshall McBride, US Army

Technical Co-Chair

Soumya Patnaik, US Air Force

Tutorials Chair

Matthew Grismer, US Air Force

Host

Deborah Schwartz, HPCMPO

HPCMPO Staff

Leah Glick, Jakeya Morgan, Denise O'Donnell, and Lisa Powell

Reviewers

HPCMP UGC 2011

Paul Antonik
Mark Barnell
Jerry Boatz
John Boles
stephen Bowman
Eddie Brooks
Doug Butler
Edward Byrd
Charles Cornwell
Mark Cowen
Larry Davis
John Dean
Carl Dyka
Datta Gaitonde
Raymond Gordnier
Steve Gorrell
Joseph Gorski
Andrew Greenwood
Matthew Grismer
Mark Hagenmaier
Aram Kevorkian
Michael List
Robert Littlefield
William Mattson
Robert Nichols
Michael Nusca
Steven Payne
Douglass Post
Cliff Rhoades
Rob Scott
Venkataraman Swaminathan
Ramakrishna Valisetty
Miguel Visbal
Bill Ward
Charles Welch
Michael White

HPCMP UGC 2011

1. Computational Fluid Dynamics (CFD)

Aeronautics

Computational Analysis for Air/Ship Integration: 2nd Year Report

Susan A. Polsky
US Naval Air Warfare Center, Aircraft Division (NAWCAD), Patuxent River, MD
susan.polsky@navy.mil

Abstract

This paper documents the accomplishments from the second year of a three-year Grand Challenge Project focusing on the application of computational fluid dynamics to predict coupled ship and aircraft aerodynamics. Unstructured chimera techniques were used to simulate the coupled ship and aircraft systems. Dynamic aircraft maneuvers were prescribed with the intention of building simulations with an auto-pilot-in-the-loop. All simulations were computed in a time-accurate fashion due to the unsteady nature of the flowfield, and used the commercial flow-solver Cobalt. Analyses for both vertical shipboard landings of the Joint Strike Fighter (JSF) and rotary-wing aircraft are discussed. Internal components of the JSF lift-fan were added to the model to increase the solution fidelity.

1. Introduction

While many aspects must be taken into consideration to ensure safe shipboard flight operations, a primary factor is evaluation of turbulent air-wake effects on aircraft performance and pilot workload. The air-wake is a product of wind passing over ship structures creating non-uniform, turbulent air flow. The US Navy conducts shipboard dynamic interface (DI) testing to evaluate ship air-wake effects on aircraft operations. These tests result in wind-over-deck (WOD) flight envelopes that prescribe in what wind conditions an aircraft can or cannot fly. The WOD flight envelopes are part of the operating procedures for all ship-based aircraft. Testing is required to generate WOD envelopes for each model of air vehicle operating from a given ship. The DI tests are performed at-sea, typically over the course of several weeks.

Application of computational fluid dynamics (CFD) methods to predict the turbulent ship air-wake has been studied in the past with considerable success[1–4]. This has resulted in the use of CFD as an analysis tool to "diagnose" air-wake structures that may impact air operations for both current and future ship designs. These diagnoses are accomplished by linking stored CFD-generated air-wake data with offline aircraft models, controlled by either a pilot model or some other autonomous controller. For this "one-way coupled" approach, the air-wake data is imposed on the aircraft model; however, the presence of the aircraft does not feed back into the air-wake data. While this approach has proven very useful, there are limitations to its applicability. When employing CFD data generated from a ship in isolation, the underlying assumption is that the presence of the aircraft will not affect the air-wake from the ship structures. In the case of a small uninhabited air vehicle (UAV), this is likely a valid assumption; however, for an aircraft that produces a large wake of its own (such as a helicopter), this assumption becomes less-and-less valid as the aircraft comes in closer proximity to ship structures. The interaction of the ship air-wake and the aircraft wake is generally referred to as ship/aircraft coupling. Aerodynamic coupling is a concern for both fixed-wing and rotary-wing shipboard operations.

As mentioned above, past research developed methods to accurately predict ship air-wake and laid the groundwork for prediction of coupled ship & aircraft predictions. Research executed in the 2006–2008 time-frame demonstrated the feasibility of modeling both stationary aircraft and aircraft with prescribed-motion immersed in ship air-wake. The aircraft types examined included fixed-wing (F-18) and rotary-wing (V-22, H-60). The present work builds upon past research in coupled ship/aircraft modeling through support from the Office of Naval Research "Coupled Aircraft Ship Simulation for Improved Acquisition" (CASSIA) program and the Joint Strike Fighter (JSF) program.

The goal of the CASSIA program is to understand the physical and numerical modeling deficiencies that prevent the application of current dynamic interface simulations for flight-envelope prediction. Analysis of an H-60 helicopter with a DDG (destroyer)-class ship was continued in the 2nd year (Figure 1). The motivation for the DDG/H-60 analysis is to understand where (in regards to proximity to a ship) aerodynamic coupling becomes important for rotary-wing vehicles. This knowledge, along with the coupled DDG/H-60 CFD data will be used to develop methods to account for aerodynamic coupling suitable for man-in-the-loop simulations (i.e., methods that run in real-time).

Figure 1. An H-60 on approach to DDG-Class ship

Analyses of the short takeoff vertical landing (STOVL) JSF on L-class US Navy ships (Figure 2) were also conducted. The JSF CFD analysis was required to prepare the ship for JSF testing. In particular, analyses were required to examine whether JSF outwash during vertical landings (VL) would damage critical (and costly) ship systems, either due to exposure to hot jet exhaust or due to the force of the jet exhaust. During an approach to a landing spot, the aircraft core nozzle passes near many large hull-mounted systems, such as radars and weaponry, and passes directly over the catwalk and flight deck. Although sub-scale experimental studies of the STOVL outwash field were conducted, these experiments were for zero ambient wind-speed with exhaust impingement on flat plates. It was recognized that the air-flow patterns would likely be affected by surrounding ship structures and prevailing wind-speed and direction. Therefore, the CFD analyses included the ship hull, island superstructure, and many of the larger ship systems in the area of concern (Figure 2). The CFD analyses were used to help determine whether steps should be taken to move or shield deck-edge equipment and replace it with instrumentation to gather data during flight-test at-sea. In the 2nd year of this project, short takeoff (STO) scenarios were also examined. In addition, the fidelity of the JSF model was significantly increased to include most of the internal lift-fan structure.

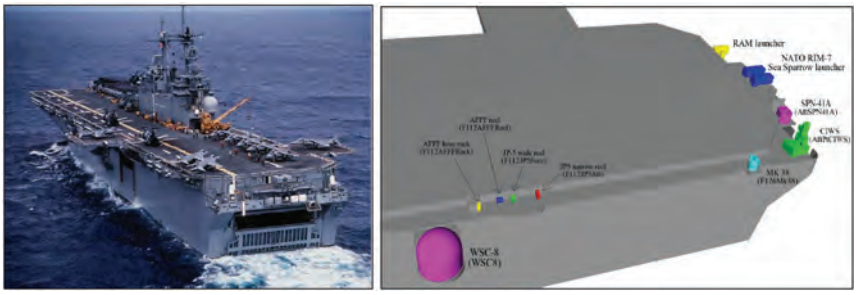


Figure 2. Left: A typical L-Class ship; Right: Stern- & port-mounted ship systems modeled in initial CFD model

2. Technical Approach

The commercial CFD solver Cobalt, which was initially developed by the US Air Force under a CHSSI project, was used for this work. The solver employs a finite-volume, cell-centered scheme with flux vector-splitting and unstructured, overset-grid technology for bodies in relative motion[5]. All cases were run with second-order spatial and temporal accuracy. Past work has established that employing a monotone large-eddy simulation (MILES) turbulence modeling approach is well-suited for the prediction of the large-scale turbulence found in ship air-wake flowfields. This approach was applied for the DDG/H-60 analysis discussed in this work. However, the MILES approach is not well-suited for JSF analysis due to the fine-scale turbulence which dominates the jet-flow. Previous to this work, a validation study was completed comparing core-nozzle-exit-flow to sub-scale experimental temperature and pressure measurements at various distances below the nozzle-exit. The results of this study indicated that applying a shear stress transport (SST) model was required to capture the core-nozzle exhaust-flow with required accuracy. This analysis approach was extended to prediction of the lift-fan-flow in the 2nd year and was found to accurately predict the thrust for the lift-fan. For coupled ship cases with no wind, using an SST model is not an issue. However, application of an SST model to aircraft/ship analyses with ambient wind presents a dilemma as it has been established that application of turbulence models to ship air-wake flows damps-out much of the higher-frequency turbulence[2]. While it is thought that this higher-frequency turbulence will not significantly affect the JSF outwash; more analysis is needed to confirm this.

The gridding and turbulence modeling approaches established by the core-nozzle-validation study were applied when modeling the complete JSF configuration including the core-nozzle, lift-fan and roll-nozzles. In the first year of this project, the lift-fan and roll-jets were modeled using user-prescribed boundary-conditions at the nozzle-exits. The boundary-conditions were derived from high-fidelity CFD solutions of the isolated nozzles provided by Lockheed Martin. Verification of the multiple nozzle-flow interaction against sub-scale experimental data using this configuration is complete. While the comparisons demonstrated that the CFD was in the "ballpark" of the sub-scale data, unsteady characteristics in the CFD analysis pointed to issues with application of the user-prescribed boundary-conditions for the lift-fan in particular. The internal geometry of the lift-fan was recently modeled using Cobalt and gave very good results for predicted thrust. As such, the lift-fan geometry has been added to the full-JSF model and test cases are currently underway.

Verification against full-scale shipboard data will take place in the 4th quarter of CY 2011. Shipboard and land-based testing will provide a wealth of verification data for the modeling approach applied here. Verification against full-scale data will be the major thrust of the 3rd year of this project.

Both the H-60 and JSF calculations employed overset grids. All grids were generated using the NASA GridTool/VGRID system. Grid sizes for the DDG/H-60 analysis were on the order of 11.4 million and 3 million cells, respectively. The rotors (main and tail) were modeled as actuator disks, and in some cases as blade elements. In the first year, grid sizes for the LHD/JSF outwash analysis were on the order of 29 million and 3 million cells, respectively. With the addition of the lift-fan internal geometry, the grid sizes for the JSF have grown to over 15 million cells. Because aircraft performance predictions were not required, a relatively simple fuselage model was used for the JSF aircraft. This allowed for construction of much smaller grids than would be required if the detailed surface-geometry was modeled. The JSF analysis modeled motion of the aircraft relative to the ship, while the H-60 analysis employed overset grids to integrate individual blade-segment aerodynamic information generated by an external blade-element model.

3. Progress to Date

3.1 LHD/JSF

As in the first year, all dynamic approach simulations were conducted with prescribed approach paths and included a 90° approach (shown in red in Figure 3) and a 45° approach and a short takeoff. Two landing spots were examined: spot 7 (forward spot shown in Figure 3) and spot 9. In the first year of the project, a generic LHD ship was used (Figure 3). However, in the 2nd-year, a ship surface model specific to LHD-1 was created and considerable detail added to the port-side deck-edge to gather data from STO analysis (Figure 4). A typical STO consists of three distinct phases: deck-roll, rotation, and fly-away. Currently, only the deck-roll portion is modeled because the core-nozzle and lift-fan conditions and orientations stay fairly constant. This is in contrast to both the rotation and fly-away portions where significant movement of both nozzles occurs. Modeling movements of the nozzles would require significant increase in complexity of the CFD model and is not currently planned.

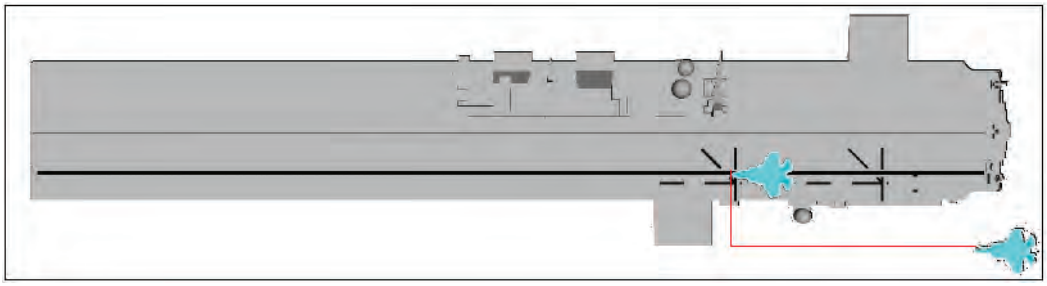


Figure 3. Example of a vertical landing approach path (red) on generic LHD ship model

The length of the deck-roll is illustrated in Figures 4 and 5. The motion is again prescribed and models the typical aircraft accelerations and speeds. To date, only zero atmospheric wind conditions have been modeled. Non-zero wind cases will be completed later this year. A snapshot from a zero-wind STO case is shown in Figure 6. The jet exhaust is visualized by surfaces of constant velocity.

Figure 4. Extent of deck-roll for short takeoff (dashed red line)

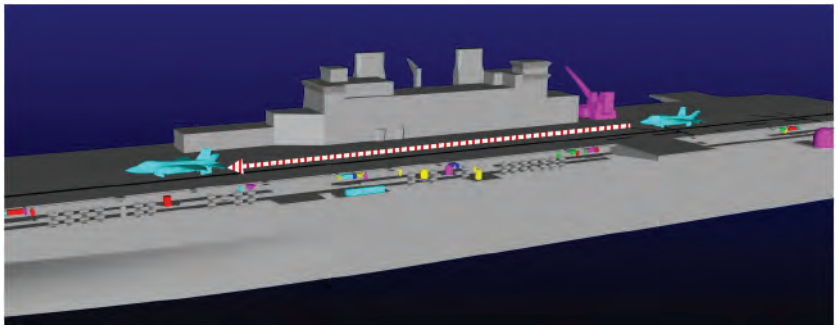


Figure 5. Perspective view of detailed deck-edge equipment modeled for STO analysis

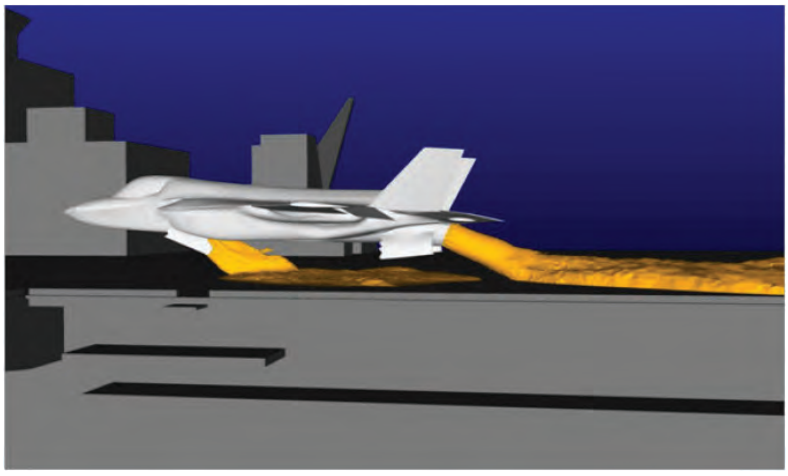


Figure 6. A snapshot-in-time of short take-off simulation. Jet exhaust and outwash visualized by velocity iso-surface.

JSF simulations were run on AFRL Hawk, MHPCC Mana, and ARL Harold. Calculations typically utilized 256 processors and required approximately 30,000–50,000 CPU-hours each, depending on the grid-fidelity and length of dynamic maneuver. Cases incorporating the internal geometry of the lift-fan are currently running. CPU requirements from these preliminary cases will be reported in the oral presentation.

3.2 DDG/H-60

In the first year, analysis of the coupled ship/helicopter flowfields used actuator disk models for the main- and tail-rotors of an H-60 helicopter. A total of 90 cases were completed with the helicopter in different hover positions around a DDG destroyer. This data is being used to create a correction methodology to incorporate first-order effects of ship/aircraft coupling into real-time simulations by modifying the ship-alone CFD air-wake datasets. Several approaches in this regard are under examination including the application of proper orthogonal decomposition (POD) analysis in conjunction with neural network algorithms.

In the 2nd year, emphasis has transitioned from actuator-disk models to incorporation of blade-element models. While Cobalt had a previously existing blade-element modeling capability, the code is being tailored to support interfacing with a blade-element model running as a separate executable outside of the Cobalt code. The motivation for this work is to enable

aircraft real-time simulation models used by the Manned Flight Simulator (MFS) at Patuxent River Naval Air Station to be "plugged" into Cobalt for coupled ship/aircraft CFD analysis. This allows models such as engine performance, landing gear response, actuator response, etc., to be incorporated into the analysis while the CFD code is used to predict the high-fidelity aerodynamics. A block diagram of the data passing from Cobalt (red box) to the external aircraft model (green box) is depicted in Figure 7. The blade-element model being linked with Cobalt is a generic "experimental" helicopter model called ExHel developed by the Air Vehicle Modeling and Simulation Branch at NAVAIR Patuxent River. The ExHel model was designed to run in a real-time, man-in-the-loop simulation.

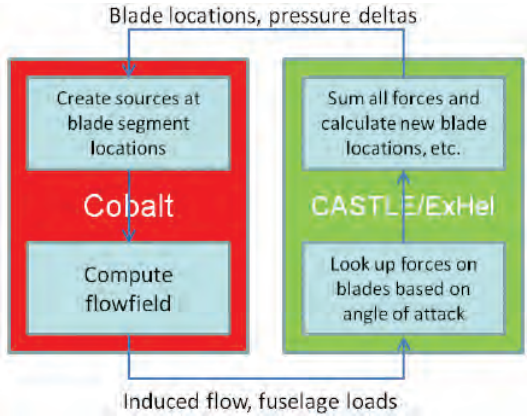


Figure 7. Block diagram showing Cobalt and CASTLE data paths

All aircraft simulation models at the MFS run within an operating system called CASTLE. As part of this project, CASTLE/ExHel were ported to an HPC system and successfully run in batch-mode. At the MFS, CASTLE communicates with outside models (such as CFD ship air-wake databases) through TCP/IP calls. However, due to security issues, using TCP/IP is not allowed on HPC systems. Therefore, data exchanges between Cobalt and CASTLE/ExHel will be accomplished using a file-in-file-out (FIFO) approach such that Cobalt will write a text file for CASTLE/ExHel input and vice-versa. Dependency on text files for data-transfer will, unfortunately, significantly hinder the execution and scalability performance of the analysis. It is hoped that a method to more-efficiently accomplish the data-transfer will be found in the future.

The ExHel blade-element model provides blade positions (including flap-angle and lead-lag positions) along with the aerodynamic variables for each blade-segment on each blade. Since the blade aerodynamic properties are provided by ExHel, it is not necessary to create body-fitted grids to model the blades within Cobalt. This significantly reduces required grid sizes. However, in order to ensure adequate grid quality always exists at the blade locations, it was decided to use overset (non-body-fitted) grids for each blade. With this approach, the background ship grid can be made fairly generic, avoiding the need to tailor a ship grid to support a particular aircraft maneuver. Examples of the blade-element grids are shown in Figure 8 along with the blue background grid. The blade locations provided by ExHel are shown as solid gray. The overset grids used for each of the four blades are shown in color (yellow, green cyan, red). Within each overset grid, cells on the blade segments do not change, ensuring consistent model-fidelity throughout the analysis.

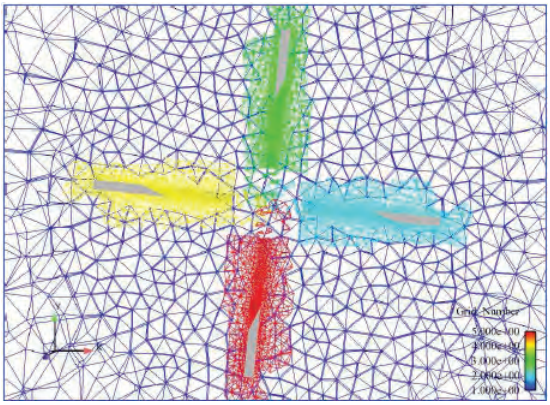


Figure 8. Blade-element overset grids (red, cyan, green, yellow) with blue background grid

Before tackling, getting Cobalt and CASTLE/ExHel to run in concert, a set of test-data was generated using ExHel running in its native environment and provided to the Cobalt developers as a text file. In this scenario, Cobalt does not talk back to ExHel, but simply reads a set of time history blade position and pressure-data and applies that data within the overset blade-grids. The intricacies associated with multiple coordinate transformations and interpolations make this step, which is still on-going, an essential part of the model development. Results from one of these preliminary calculations are shown in Figures 9 and 10. These figures depict iso-surface of vorticity colored by pressure at a very early time-step (Figure 9) and at a time-step ~5 seconds later (Figure 10) showing how the flowfield develops. Lessons-learned from this preliminary analysis are being used to better-define the data needed from ExHel and to establish a verification method for the CFD predictions.

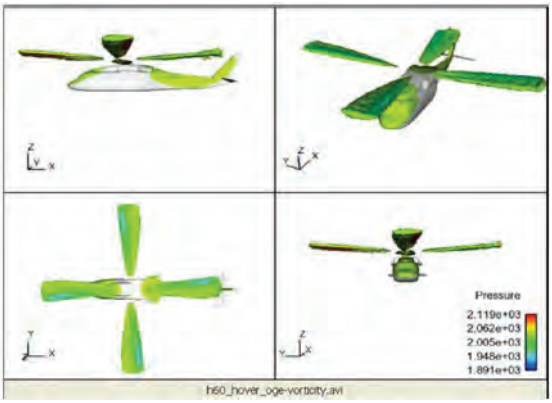


Figure 9. Iso-surface of vorticiy at an early time-step of Cobalt analysis using ExHel blade-element data

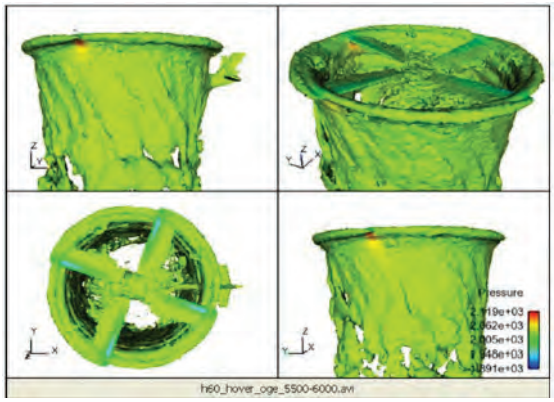


Figure10. Iso-surface of vorticiy at a later time-step of Cobalt analysis using ExHel blade-element data

Although the generic ExHel model is being used as the demonstration case for Cobalt/CASTLE integration, the ultimate goal is to use the integrated approach for a real air vehicle. A major difficulty in linking an aircraft like an H-60 in a non-real-time simulation is the need for a pilot to fly the simulation model. This requirement also exists for ExHel. While a pilot model (auto-pilot) has been developed to "fly" ExHel during a Cobalt run, it will never quite represent how an actual pilot would fly an aircraft. In the near-term, the pilot model will be programmed to maintain a stationary hover within a ship air-wake. Upon successful completion of hover analyses, the pilot model will be enhanced to include approach to the ship. A hover pilot model is complete and runs in its native CASTLE environment linked to offline CFD ship air-wake databases for DDG. The pilot model will be ported to an HPC system in the future.

The Fire Scout VTUAV provides a unique opportunity because the same autonomous control system that flies the aircraft can be integrated in the simulation, obviating the need for a pilot model. Because a pilot is not required in-the-loop, the simulation need not run in real-time. Development of a Fire Scout simulation model tailored to link with Cobalt is underway. A Fire Scout CFD fuselage surface model has been created, and initial calculation of fuselage drag has been calculated and compared to available wind-tunnel data. It is hoped that before the end of the 3rd year of this project, a demonstration case of Cobalt linked with a Fire Scout simulation model will be completed.

4. Significance to the DoD

Rotary-wing ship integration modeling and simulation research funded through the Office of Naval Research CASSIA program is paving the future in dynamic interface modeling and simulation. Results from this work will directly contribute to improving ship design, ship-based aircraft control system design, and flight-control design for piloted and autonomous air vehicles. Numerical algorithms and physical models developed through this program will be transitioned to the CREATE program to benefit the CFD community throughout the DoD.

Techniques and methodologies developed through this project will be directly applicable to the JSF program during developmental testing (DT), operational testing (OT), and fleet support to help ensure safe operation of the aircraft aboard ship. Successful application will require the development, validation, and application of CFD techniques related to coupled aircraft and ship air-wake prediction. Inherent in the development and validation of this technology are important science and technology areas such as vortex modeling, large-eddy simulation, grid-generation, parallel-code development, CFD boundary-condition development and temporal-accuracy issues. These developments will also benefit other DoD aircraft and ship programs, such as V-22, VTUAV, CVN-78, and DDG-1000.

The military advantages gained by exploiting HPC capability are related to design and testing. This analysis will enable ship integration of the JSF system in such a way that minimizes potential damage to ship structure and systems, and helps to ensure safety of personnel.

Acknowledgements

The author would like to thank: 1) the Office of Naval Research and the Joint Strike Fighter Program for their financial and professional support, 2) the High Performance Computing Modernization Program for their generous allocation of computer time, without which this work would not have been possible, and 3) Cobalt Solutions LLC for Figures 7–10.

References

1. Polsky, S. and Bruner, C.W., "Time-Accurate Computational Simulations of an LHA Ship Air-wake." *AIAA Paper 2000-4126*, 2000.

2. Polsky, S., "Computational Study of Unsteady Ship Wake." *AIAA Paper 2002-1002*, January 2002.

3. Polsky, S., "CFD Prediction of Air-wake Flowfields for Ship Experiencing Beam Winds." AIAA Paper 2003-3657, August 2003.

4. Woodson, S. and Ghee, T., "A Computational and Experimental Determination of the Air Flow Around the Landing Deck of a U.S. Navy Destroyer (DDG)", *AIAA Paper AIAA-2005-4958*, June 2005.

5. Strang, W.Z., Tomaro, R.F., and Grismer, M.J., "The Defining Methods of Cobalt60: A Parallel, Implicit, Unstructured Euler/Navier-Stokes Flow Solver", *AIAA-99-0786*, January 1999.

Computational Modeling of Geometrically Complex Modern Weapons Bays and Weapons Dispense at High Supersonic Speeds

Rudy Johnson
US Air Force Research Laboratory, Air Vehicles Directorate (AFRL/RB), Wright-Patterson AFB, OH
rudy.johnson@wpafb.af.mil

Chandrasekhar Kannepalli, Chris Chartrand, Roger Birkbeck, and Neeraj Sinha
Combustion Research and Flow Technology, Inc. (CRAFT Tech), Pipersville, PA
{kchandra, ccc, rbirkbeck, sinha} @craft-tech.com

Abstract

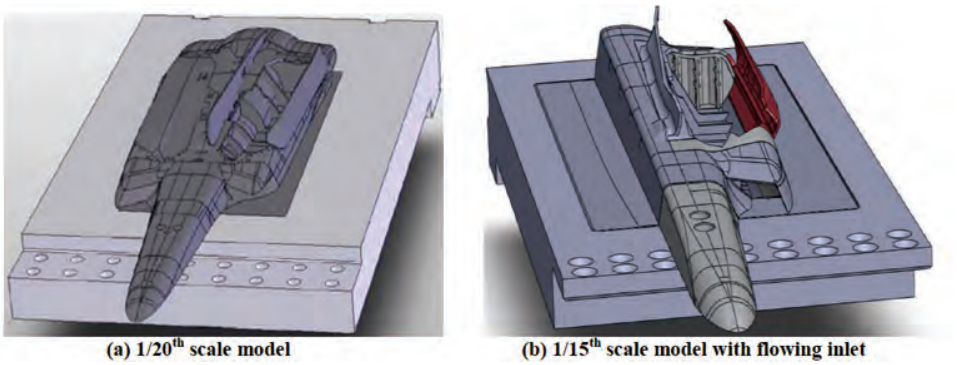
Computational modeling of the air flow in a geometrically-complex weapons bay, as well as the separation of weapons from a generic cavity model into a high-Mach (3.5–5.0) free stream, are summarized. Computational work on the geometrically-complex bay is focused on flow control strategies for the reduction of dynamic pressure loads. Results indicate significantly different acoustic characteristics for the bay when one versus two doors is open. When both bay doors are open, leading-edge mass blowing combined with aft-wall treatment is an effective approach to reduction of dynamic pressure loads. However, the effectiveness of this control strategy is reduced when only one bay door is open, due to the generation of strong resonant tones. Computational fluid dynamics (CFD) analysis has led to the development of novel, easily integratable "passive" control strategies involving "baffles" that have shown significant reduction in the tonal amplitude. The CFD simulations have enabled a good understanding of the complex flow field, led to the development of control strategies, and have guided the sub-scale wind-tunnel test program. Significance to the Department of Defense (DoD) is the integration of successful acoustic reduction concepts into realistic aircraft weapons bays.

CFD simulations of weapons separation from a generic bay at high-speed are being used to develop data acquisition techniques for sub-scale wind-tunnel testing in the high supersonic regime. The bays at these high-Mach numbers involve spatial and temporal flow field scales that are an order-of-magnitude smaller than those at Mach 1.5–2.0 regime, making the numerical simulations of such bays computationally more expensive. The availability of the high performance computing resources towards this end is gratefully acknowledged, while the significance to DoD is the improved understanding of the associated flow physics that will help in the better design of data acquisition techniques for store dispense at these high supersonic Mach numbers.

1. Introduction

This paper provides a sampling of the computational modeling accomplished during the third year of the high performance computing (HPC) Challenge Project "Weapons Bay Aeroacoustic Load Suppression and Separation Enhancement Simulations." Previous work included weapons bay acoustic modeling using the CRAFT CFD® software applied to a generic box cavity on a flat plate, and to weapons bay geometries representative of the F-22[1], and F-35[2] aircraft. The computational modeling has guided the design, development, and evaluation of weapons bay acoustic reduction techniques. Computational fluid dynamics (CFD) results were also used to support small-scale experiments conducted at the National Center for Physical Acoustics (NCPA) at the University of Mississippi, Oxford, Mississippi.

Since the previous User's Group Conference, the small-scale wind-tunnel model representing the F-35 weapons bay has been redesigned to allow flow through the aircraft inlet. In addition, the scale of the model was increased from 1/20th to 1/15th as a part of the redesign (Figure 1). CFD analysis was used to design the internal flow path to reduce inlet spillage that could contaminate the boundary-layer approaching the weapons bay. Two baseline aircraft configurations are examined. Selected flow control concepts are discussed and model results are presented with experimental data.



(a) 1/20th scale model (b) 1/15th scale model with flowing inlet

Figure 1. Small-scale weapons bay acoustics wind-tunnel model representative of the F-35

The unsteady flow character of the weapons bay can also have significant consequences for internally carried stores, either through damage to sensitive electronics or structural components. In addition, the unsteady store loads encountered can influence store separation trajectories[3]. Small-scale drop testing is a technique used to study store separation in a laboratory environment. The current research in this area attempts to synchronize the data collection to develop an understanding of the effect that the unsteady flow has on the store separation trajectory. Photogrammetric techniques are currently used to track the store motion, but a current project to develop small-scale telemetry packages for data acquisition during drop testing is underway. Computational modeling of a generic box cavity in a flat plate is accomplished with a store positioned in the opening of the cavity. The resulting store loads are used to establish an estimate of the design specifications for this telemetry package.

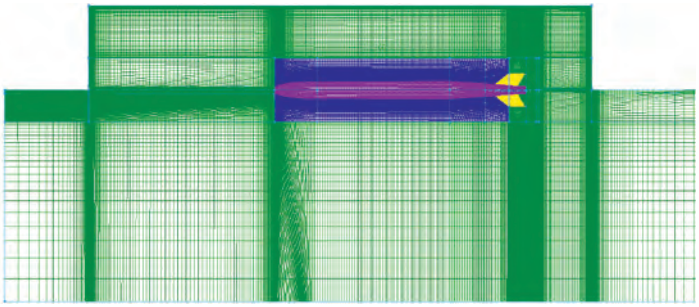


Figure 2. Computational mesh for a store located in the opening of THE L/D=6 generic bay

The remainder of the paper will first discuss the use of CFD and small-scale wind-tunnel testing to study the unsteady flow field in a weapons bay that is representative of the Joint Strike Fighter (JSF) aircraft at supersonic Mach number of 1.5. A description of the computational approach to collecting unsteady store loads in a generic cavity is given. Details of the computational methodology will be presented, followed by a discussion of the weapons bay behavior for two baseline configurations involving both bay doors open and one door closed. An analysis of one of the acoustic control strategies will be provided then a look into the store loads experienced in a Mach 3.5 cavity are presented. Finally, the conclusions and recommendations of these studies will be presented.

2. Modeling Approach

The process used for the acoustic modeling and the store loads are described. A short description of the CFD software completes this section.

2.1 Weapons Bay Acoustics

As indicated in Figure 1b, an approximate model of the F-35 bay has been designed that is floor-mounted. The original CAD model of the full aircraft was cropped along a plane parallel to the aircraft axis[2]. Care was taken to position the model relative to the wind-tunnel boundary-layer and to incorporate an inlet geometry with flow-through ducting to ingest the approaching flow; thus preventing it from spilling over the model and into the bay (an improvement over the previous 1/20th scale model, Figure 1a). CFD was used to carefully design the duct to insure venting at the back of the model,

downstream of the bay. The details of the upstream fore-body, the nose and the engine inlet are included to provide as realistic an inflow to the weapons bay as possible. Since the size of the model has been increased, the left-hand portion of the aircraft was removed to minimize blockage effects during tunnel operation. Geometric details of the complex internal structures of the bay, as well as the bay doors in two different configurations, are modeled (Figure 1b).

Since, modeling the entire flow field around the model in a single simulation is prohibitively expensive, a zonal approach involving three different domains is used. An initial Reynolds-averaged Navier-Stokes (RANS) simulation of the wind-tunnel nozzle and empty test section provides the inflow conditions to a second RANS simulation involving the fore-body model mounted on the test section floor (Figure 3). The latter fore-body simulation provides the inflow conditions to the Large-eddy simulations (LES) of the complex weapons bay. Figure 3a shows the computational extents of the domains for the three zonal simulations, while Figure 3b shows the Mach number contours on the mid-span plane passing through the center of the bay, demonstrating the continuity of the solutions across the different zonal simulations. The wind-tunnel simulation will not be discussed, but the fore-body simulation that provides the inflow boundary-conditions to the weapons bay LES on a plane upstream of the bay will briefly be discussed below.

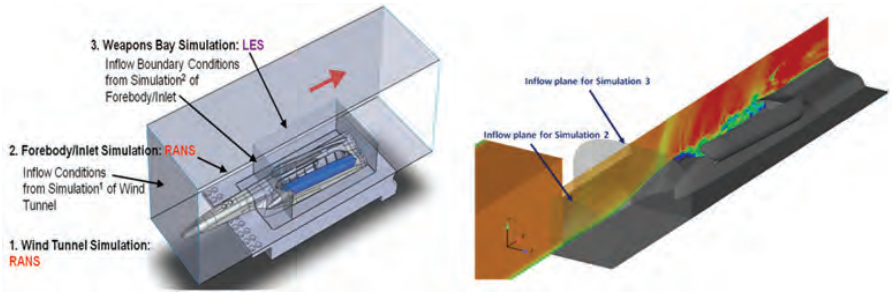
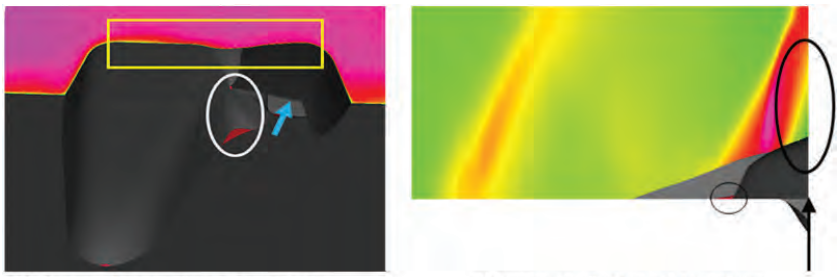


Figure 3. Zonal approach used for the weapons bay simulations

Figure 4a shows the fore-body geometry and the inlet along with iso-surfaces of separated flow regions (in red). Also shown on the far background are the mean U-velocity contours on the inflow/approach plane (that is just upstream of the weapons bay leading-edge) to the weapons bay simulation. Figure 4b shows the mean pressure contours and the iso-surfaces of separated flow zones in the mid-span plane of the bay. Clearly, the flow-through inlet allows the impinging flow to pass with minimal separated flow zones (that are very small); and hence does not result in any flow spillage into the bay. As a result, a nice, "clean" boundary-layer-type of flow approaches the bay, as can be seen from the U-velocity contours on the far background plane in Figure 4a. Further, the inlet only causes a weak shock structure with "quicker" recovery of the pressure to the nominal Q_∞ conditions upstream of the bay leading-edge. In an earlier paper, Johnson, et al.[2] showed that the modeling of the flow-through inlet is very important to assure a clean and uncorrupted flow entering the weapons bay, as would be the case for the prototype aircraft.



(a) Separated flow regions & u-velocity on approach plane of bay

(b) Pressure on the bay mid-span plane

Figure 4. Fore-body/inlet RANS simulation

2.2 Store Loads

The intention of this work is to develop an understanding of the magnitude and frequency content of store loads for store released from a weapons bay into a Mach 3.5 free stream. This data will be used to guide the selection of sensors (accelerometers & telemetry hardware) to capture the store loads during small-scale wind-tunnel drop testing. Since Mach 3.5 is beyond the typical flow regime where the CRAFT CFD® solver is typically run, an empty bay solution is generated

to work out the grid resolution and time-step requirements. The basic approach is the same as describe above; a RANS model of the blow down tunnel nozzle and empty test section is accomplished to provide the boundary-conditions for an LES simulation of the generic L/D=6 weapons bay, Figure 5. A vertical slice of the computational mesh along the centerline is shown in Figure 2, with the store model positioned in the opening of the cavity. The cavity mesh alone contains ~7.5 million points, while the total from the cavity and store is ~14.7 million.

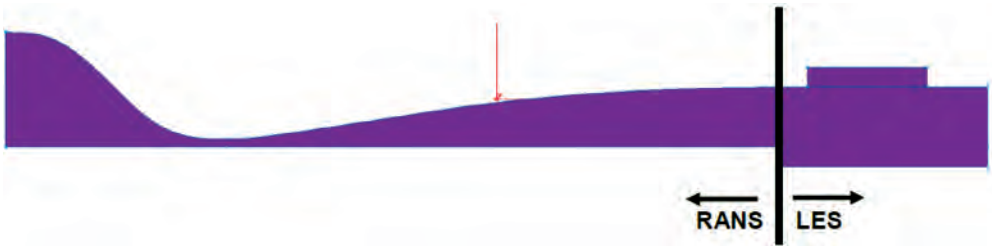


Figure 5. NCPA wind-tunnel nozzle & L/D=6 cavity mounted on the tunnel ceiling

The RANS simulation indicates the tunnel boundary-layer is ~1.45 in. thick as it approaches the bay. This will need to be controlled to be more representative of expected flight conditions, but for the current modeling effort this profile is used. The time-step required to model the unsteady flow is 2.0e-8 seconds, which is 5 to 6 times smaller than what was previously required for Mach 1.5 & 2.0 acoustic modeling for the same bay. The empty bay calculation shows that boundary-layer; and hence the shear-layer over the cavity are more stable than in lower Mach number flow fields due to the smaller, less coherent shear-layer structures. Similar observations were reported by Murray & Elloit[4].

2.3 CFD Software

All the numerical simulations in the present effort have been carried out using CRAFT Tech's CRAFT CFD® software[5]. CRAFT CFD® is a multi-block, structured flow solver. For unsteady LES, the spatial numerical scheme implemented uses a higher-order (fifth-order) upwind-biased reconstruction procedure[6,7] in conjunction with Roe's approximate Riemann solver. The fifth-order reconstruction scheme permits the resolution of waves using a minimal number of points, while also minimizing the numerical dispersion and dissipation errors. For problems dominated by the interaction of physical structures at scales only just larger than the grid scale, such a scheme is mandatory. The RANS models in CRAFT CFD® are based on a formulation of the k-ε turbulence model that is capable of modeling free-shear flows and near-wall flows within a unified framework[8]; while two sub-grid scale models, the Smagorinsky model, and a one-equation sub-grid scale kinetic energy model, are available for LES applications. In addition, a hybrid RANS-LES model[12] coupling the RANS k-ε turbulence models with the one-equation sub-grid scale kinetic energy model of LES is also available. The CRAFT CFD® code has been validated for various LES applications, including cavity flows[9–12]. In particular, it was rigorously validated (as part of a previously-funded US Air Force Research Laboratory (AFRL) program involving Separation Enhancement and Acoustic Reduction (SEAR)[15,16,12,13,14]) with wind-tunnel test data for the prediction of aero-acoustic loads in cavity flows at both subsonic and supersonic flow conditions. The time integration schemes in CRAFT CFD® include a four-stage, low-storage, second-order Runge-Kutta scheme, a directional ADI solver that is both computationally efficient and inexpensive in memory usage, and a fully-implicit, symmetric, Gauss-Seidel scheme. Sub-iterations are used to achieve second-order temporal accuracy. For efficient computation of large three-dimensional (3D) problems, a parallel framework based on distributed memory architectures has been implemented; wherein the computational domain is decomposed into sub-domains of equal size, and each of these sub-domains is solved for on different processors. Exchange of information across processors is implemented via the Message Passing Interface (MPI) libraries. The parallel efficiency of the code scales very well, and it runs successfully on several operating system platforms[1].

3. Results

3.1 Weapons Bay Acoustics

The simulations in this study were performed for a free stream Mach number, M_∞=1.5, and at a dynamic pressure, Q_∞=12.6 psi. Two baseline configurations are considered, both doors open (Figure 6a) and outboard door closed (Figure 6b). In addition, flow control strategies designed and tested for each configuration will be discussed.

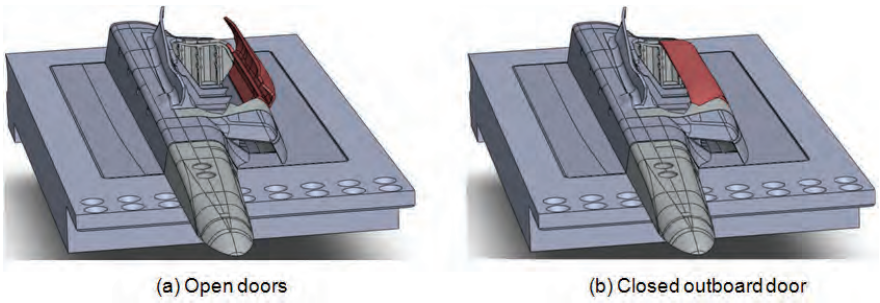


Figure 6. Bay door configurations analyzed

A very high-resolution, structured grid was developed for the wind-tunnel models that incorporated the external features (fore-body, engine inlet, fuselage undercarriage, weapons bay doors, etc.), as well as the dominant internal features, (the engine duct and a few bulkheads). Much of the grid generation effort was spent developing the computational mesh for the complex bay (approximately 10 million grid-points). Although the cavity was simplified significantly for the sub-scale testing, the geometry inside the cavity still remains complex from the perspective of making a structured grid that is sufficient to run a fifth-order spatial accurate solution via LES. The simulations presented here did not use any sub-grid scale model, and hence used a MILES approach. The simulations used the implicit ADI time integration scheme with three sub-iterations to minimize the linearization error on the implicit side. A constant physical time-step equivalent to 0.1 microseconds is used for advancing the flow solution in a time-accurate sense.

The flow field in the simulation is initialized with no-flow in the bay and the free stream conditions outside of the bay. The unsteady weapons bay flow field naturally evolves as the solution is advanced in time. At some point during the evolution, the turbulent unsteady flow will reach a statistical steady-state. Subsequently, snapshots or ensembles of the flow field solution are saved over a time period equivalent to 0.03s for statistical averaging. This corresponds to about 25 flow-through time periods, t_c (where $t_c=L/U_c$, L is the length of the cavity and $U_c \sim 0.57U_\infty$). Note that 0.57 is the κ value typically used in the Rossiter[17] or Heller-Bliss[18] empirical formula for evaluating the Rossiter or resonant modes/ frequencies associated with the flow over a cavity. Here, U_c (where $U_c=\kappa U_\infty$) is the mean convection speed of vortical structures/disturbances travelling downstream along the shear-layer above a cavity. Each simulation required about 200 CPU-hours with 128 processors on an IBM P5 machine.

3.1.1 Analysis of Baseline Configurations

The non-uniform "tooth-like" leading-edge for the weapons bay (Figure 7a) results in differential flow expansion into the bay. This sets up a strong stream wise vortex that entrains high-speed free stream fluid into the bay. The plots in Figures 7b and c of the iso-surface of the mean U-velocity (equivalent to 325m/s and corresponding to a local Mach number ~1.0), colored by the depth (of high-speed flow penetration) into the bay relative to the tip/edge of the bay show a "groove-like" structure that is an imprint of the above-discussed vortex. It is also representative of the mean shear-layer shape over the bay. Since the iso-surface level closely corresponds to the sonic Mach number, the flow above this surface can be said to be nominally supersonic, and that below it to be subsonic. For the open door configuration in Figure 7b, the groove-like structure is deeper and extends to a greater distance downstream in the bay; hence the vortical flow induced is stronger. For the closed door configuration (Figure 7c), the tooth-like structure on the leading-edge of the bay is partially closed by the door; hence the differential flow expansion into the bay is reduced, resulting in a weaker vortex. Consequently, the iso-surface "groove" is shallower, implying that the high-speed fluid entering the bay is reduced; hence the span wise curvature of the shear-layer is smaller. Note that in both of the baseline configurations; the flow predominantly heads straight down the bay with higher speed fluid along the inboard side.

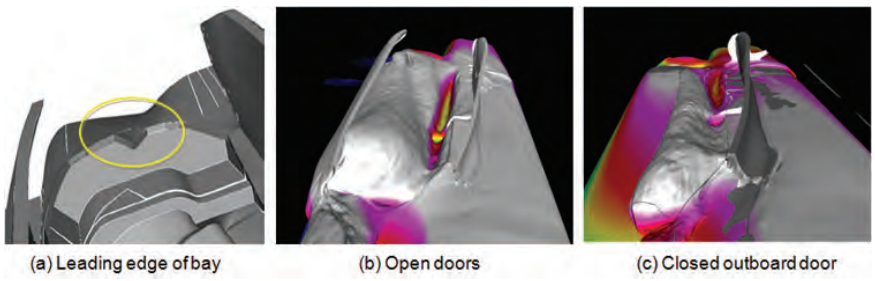


Figure 7. Iso-surface of U=325m/s colored with distance (depth) from bay edge

The mean-flow streamlines colored by the local speed in Figure 8 show that for the open door configuration, the main (high-speed) flow accelerates as it dips down into the bay. It forms a few re-circulation bubbles along the outboard side of the bay, but no distinct re-circulation pattern sets up that extends the entire length of the bay. The differential flow pattern due to the non-straight leading-edge is highlighted, and can be seen to be higher for the open doors configuration. This involves a larger turning of the flow from the entire leading-edge region to its straightening along the inboard side of the bay. For the closed door configuration, the main high-speed flow heads straight along the inboard side of the bay impinging on the inboard corner of the aft-wall. It then re-circulates along the outboard side, forming one big re-circulation zone communicating fluid from the aft-end to the front-end of the bay. The predominantly lighter colors on the inboard side and darker (blue) colors of the streamlines along the outboard side imply high-speed flow (in the downstream direction) and low-speed flow (in the upstream direction), respectively.

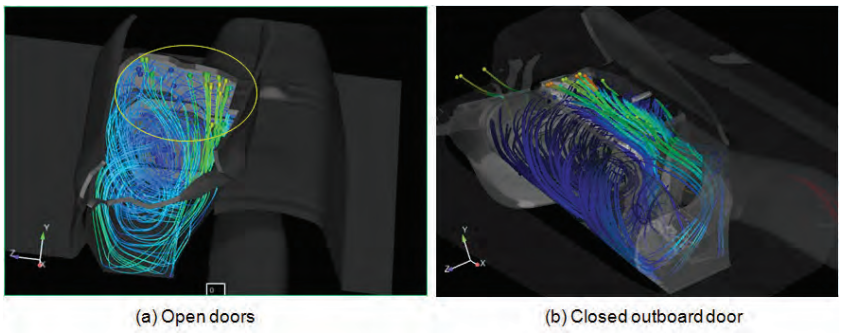


Figure 8. Mean-flow streamlines colored by the local speed

The OAPSL distribution is shown in Figure 9 for the two configurations. In general, the levels are higher for the closed door configuration over the entire bay (especially along the inboard side and on the aft-wall). This is rather a surprising result when one considers the fact that the surface area of flow affecting the bay has decreased (i.e., one-half of the bay is closed to the free stream flow), but the "aero-acoustic" imprint on the bay surfaces has increased. This is due to the classical Rossiter-like or longitudinal resonance mode facilitated by a large re-circulation region which extends the length of the bay with the outboard door closed. More details of the analysis is reported by Kannepalli, et al.[19].

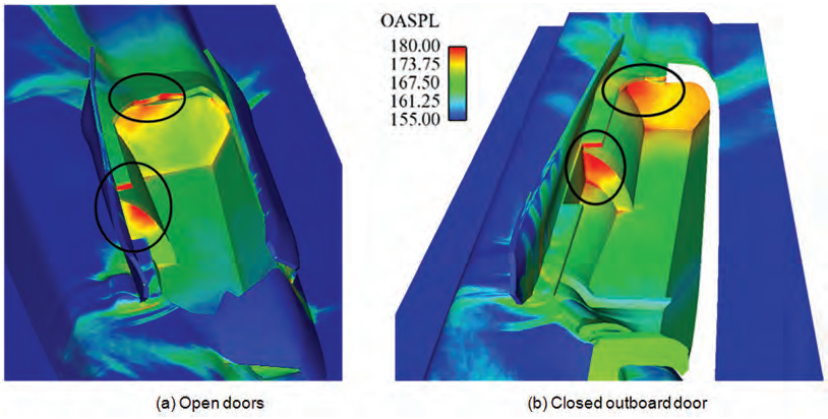


Figure 9. OASPL distribution on bay surfaces

Figure 10 shows a line plot of the OASPL distribution along the mid-span of the ceiling. The surprising result elaborated earlier, that the closed door configuration OASPLs are higher (~3–6 dBs) than the open doors configuration, is again evident in this figure. Also shown in the figure is the comparison with the data from the experiment[20], which is in excellent agreement (within 1–2 dBs). The closed door configuration displays a distinct double-inflexion distribution of the OASPL that is associated with the presence of resonant tones. The minimum at an x/L~0.7 has been seen for rectangular cavities too (with an L/D>4 and Mach 1.5) by Dix, et al.[21] and Arunajatesan, et al.[16]

Figure 11 shows the SPL spectra for the baseline outboard door closed configuration on the ceiling at x/L~0.95. From the CFD, there is a dominant tone of ~975 Hz and secondary tones at 480, 1,500, and 1,933 Hz. These frequencies compare to Rossiter modes R2 ~ 895 Hz, R1 ~384 Hz, R3 ~1,350, and R4 ~1,840 Hz, computed using the empirical formula with

constants (σ, κ) = (0.25, 0.57) and an L/D~6 (Heller-Bliss[18]). The dominant 2nd mode has a wavelength that is approximately equal to the length of the bay. Similar results of the 2nd mode dominance for supersonic Mach 1.5 rectangular bays have been reported by Dix, et al.[21] and Arunajatesan, et al.[16]. The entire weapons bay can be considered to be resonating with a dominant "Rossiter-like or longitudinal mode" of ~975 Hz with amplitude of ~25 dBs. ***This frequency corresponds to ~60 Hz for the full-scale bay, and with amplitude of ~25 dBs—it is of a serious concern from a structural standpoint.*** There is good agreement with the experiment with all the tones being captured very well. However, the tones reported in the experiment are at slightly lower frequencies. The experimental model has a few more internal details of the geometry inside the bay compared to the computational model.

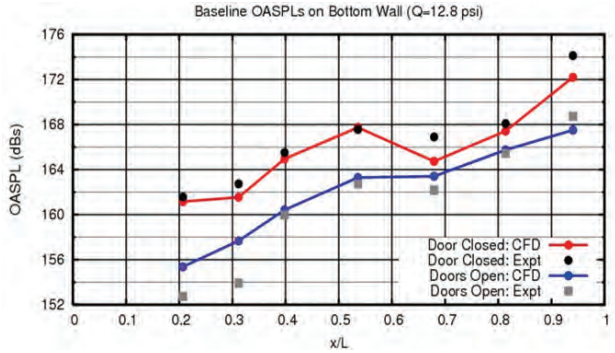


Figure 10. Bay ceiling OASPL

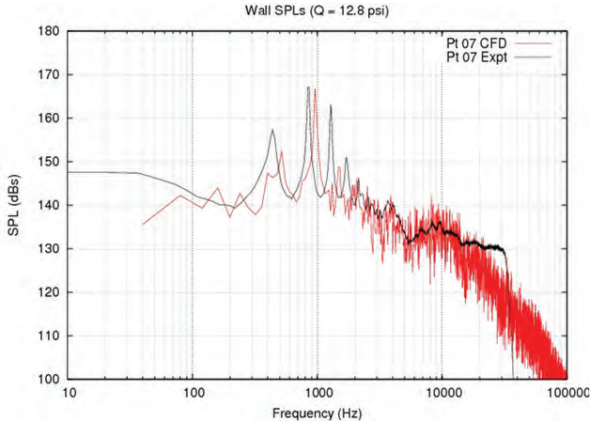


Figure 11. Computational and experimental spectra outboard door closed

3.1.2 Analysis of Control Strategies

Leading-Edge Mass Blowing (LEMB)

The motivation behind the use of LEMB is to help reduce the flow expansion into the bay and decrease the high-speed fluid entrainment into the bay. In addition, the slot jets would act like a "fluidic-spoiler" and also help stabilize and/or loft the shear-layer above the cavity. LEMB affords an adjustable concept that can be tuned (by increasing or decreasing the mass flow rate) for a wide range of operational flight envelope when compared to spoilers that are designed and work most efficiently for a given configuration. It was also earlier used in the SEAR program[13,15,14] with subsonic cavities, and Arunajatesan, et al.[16] demonstrated the effectiveness of this concept for a rectangular L/D=6 cavity at a free stream Mach 1.5.

LEMB involved blowing of sonic jets of air through the leading-edge slots, Figure 12a. The mass flow rates for the open doors and closed outboard door configurations are 3.6 lbs/s and 2.2 lbs/s, respectively, for the full-scale aircraft. Figure 12b summarizes the effect of LEMB on the OASPL distribution on the ceiling of the bay for both the configurations. The broadband levels in the open doors case decreased by 1–2 dBs in the aft-end of the bay, but the flow rate used exceeds 3 lbs/s. The LEMB was not effective at the mass flow rate studied for the closed outboard door configuration.

Other blowing rates with LEMB were examined at NCPA in the experiment in tandem with other control strategies involving pressure-relief-system (PRS) and aft-wall treatments such as porous liners that showed good reduction in the broadband noise levels[20,17].

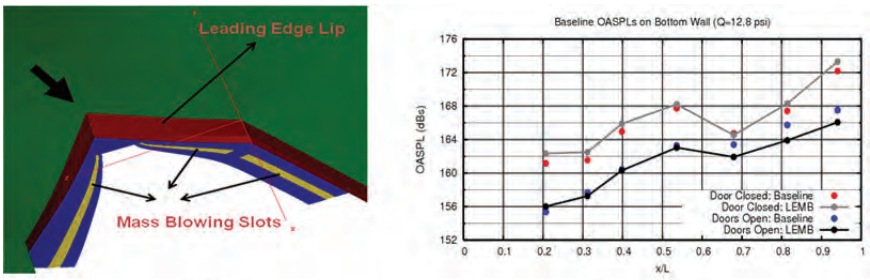


Figure 12. Illustration of LEMB slots, and (b) effect of LEMB on the OASPLs on the ceiling of the bay

Baffles

Baffles are physical partitions or walls positioned in the bay just like bulkheads, but are not required to provide structural integrity. These concepts were explored for the closed (outboard) door configuration motivated by the fact that cutting off the quiescent bay volume underneath the outboard door may be a good control strategy. These baffles could either be longitudinal, the green surface along the stream wise direction in Figure 13a that would physically partition the bay and block out the volume underneath the outboard door. They could also possibly be along the transverse/span wise direction (Figure 13b) and provide impedance to the acoustic disturbance/wave from freely moving upstream; i.e., destroying the acoustic feedback loop.

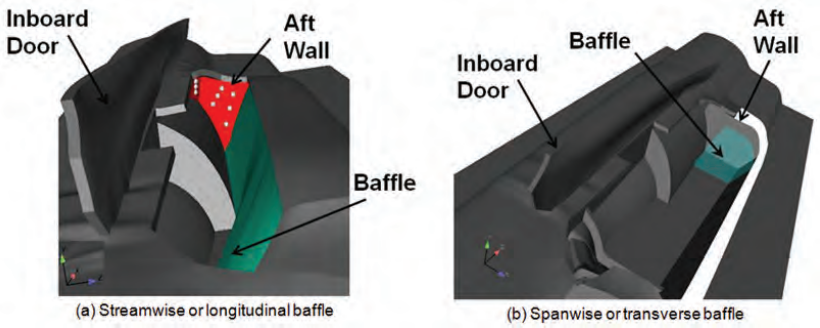


Figure 13. Illustration of (a) the stream wise baffle, and (b) the span wise baffle used as control concepts for the closed outboard door configuration

For brevity, only the span wise baffle results will be discussed here, see Kannepalli, et al.[19] for more details. This baffle was located at an x/L=0.87. It was designed to cut-off the re-circulating flow under the outboard door with a Mach <0.3 (as in Figure 14) from the baseline case involving the closed outboard door. The presence of this baffle lofts or deflects the flow (not shown) near the aft-wall region; hence the OASPLs on the aft-wall in Figure 15 are reduced by ~5 dBs over the entire wall (a 43% reduction in the fluctuating pressure component). In effect, it behaves like a "spoiler" for the aft-wall. It is very interesting to note that this effect is not only local to the aft-wall region, but also extends upstream, as can be seen in Figure 16, for the OASPL distribution on the ceiling. Except very close to the location of the baffle, the OASPLs over the entire ceiling of the bay are also lowered. This would imply that it has also hindered with the acoustic feedback loop responsible for resonance in the bay. The resulting SPL spectra indicate that the dominant tones shift to higher frequencies of 1,320 Hz and 1,754 Hz, respectively (that correspond to the 3rd and 4th modes seen in the baseline closed door configuration of the experiment). The overall noise levels; however, decrease as the amplitudes of these tones is also lower (only ~10–15 dBs).

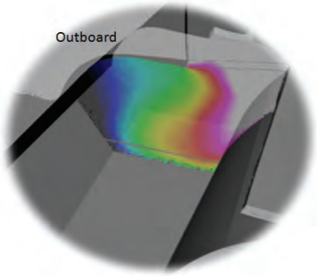


Figure 14. Mach number contours on a plane at x/L=0.87 in the outboard door closed configuration used to design the span wise baffle

It is important to note that there is some optimization necessary for this baffle based on its stream wise (x) location that shifts the dominant modes to higher frequencies and lowers the OASPLs. Also, conceivably more than one such baffle could be used.

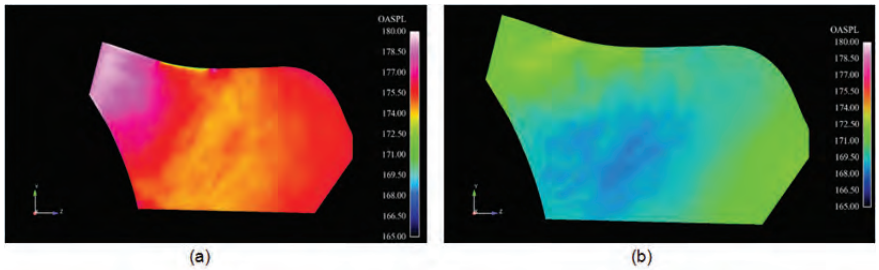


Figure 15. Aft-wall OASPL distribution; a) baseline door closed, and b) with the span wise baffle

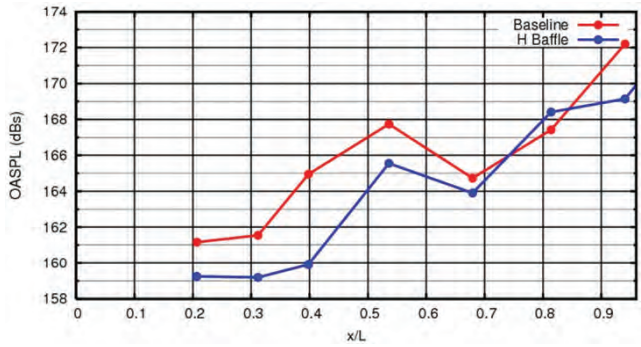


Figure 16. Comparison of the ceiling OASPLs

3.2 Store Loads

As mentioned earlier, an empty bay simulation and the case with the store submerged in the shear-layer in the opening of the cavity are modeled to generate unsteady time histories of the store aerodynamic loads. Once the simulations reached an approximate statistical steady-state data, was collected for 6 ms, equivalent to 6 cavity cycles based on the second Rossiter mode. This took approximately 250 CPU-hours on an IBM P5 using 108 processors for the case with the store in the shear-layer.

Looking at the mean stream wise (U) velocity contours in both the mid-span and mid-stream wise location indicates that the presence of the store acts to loft the high-speed flow away from the cavity, while increasing the growth of the shear-layer into the cavity, Figure 17. The store in this location also produces asymmetric flow within the cavity.

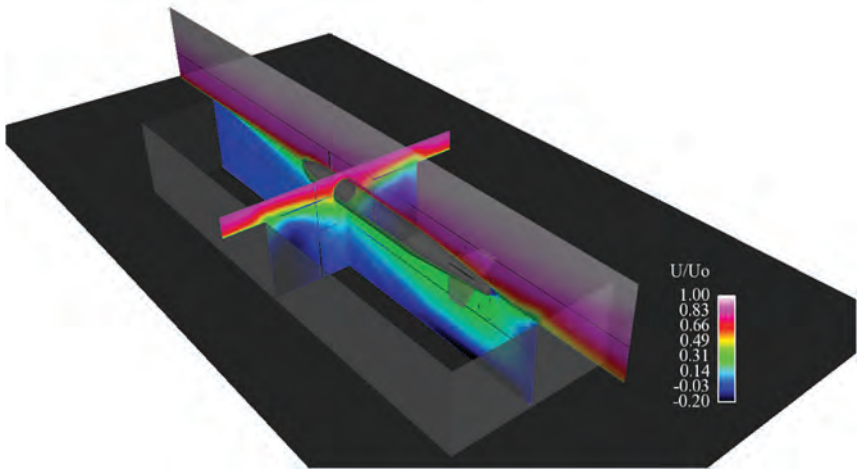


Figure 17. Mean U velocity contours with the store in the cavity opening

A quick comparison to the empty cavity indicates the change in the OASPL due to the presence of the store, Figure 18. Once again, the asymmetry in the solution is seen on the aft-wall for both cases. The free stream side of the store nose clearly takes a beating from the shear-layer as it diverts the high-speed flow away from the cavity. In general, the store causes slight reductions in the OASPL due to the blockage effects, which are consistent with what is reported in the literature[22].

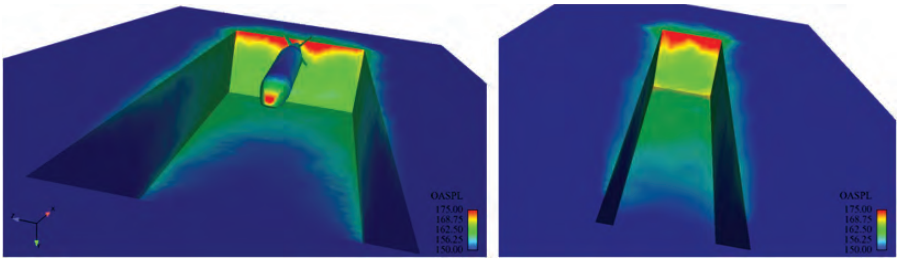


Figure 18. Surface contours of OASPL for the cavity with store and clean cavity

With a Mach 3.5 free stream velocity and the relatively thick shear-layer, the spectral character in the cavity is broadband. This is also reflected in the frequency spectra of the unsteady store loads, Figure 19. This indicates that the stream wise force component and rolling moment are relatively smaller in magnitude than the side and normal forces and the pitch and yawing moments; reflecting what is typically of concern during store separation analysis. The drop-off in the unsteady components beyond 10 KHz indicates limits on the data sampling and bandwidth requirements for the telemetry package. Further analysis to look at dynamic pressure scaling effects on the fluctuating loads, both frequency content and magnitude, is planned.

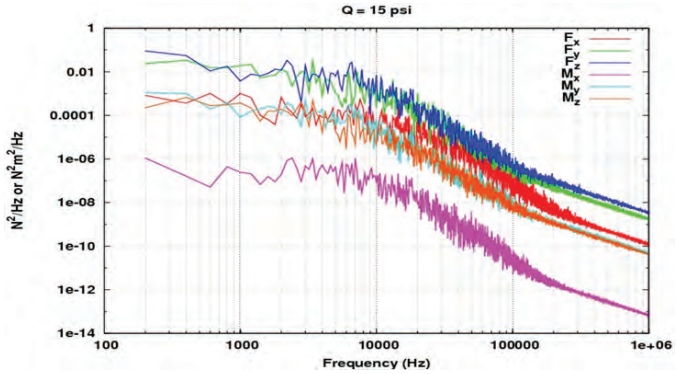


Figure 19. Frequency spectra of the unsteady forces/moments on the store

4. Conclusion

Computational analysis of a complex weapons bay geometry representative of that on the F-35 aircraft was done for a free stream Mach of 1.5. Baseline configurations with both bay doors open and with the outboard door closed were carried out. The dynamic pressure loads on the bay for the baseline open doors configuration showed broadband spectral behavior when compared to traditional rectangular cavities at the same Mach number. However, the closed door configuration shows a dramatic change in behavior with the generation of tones, as well as an increase in the OASPLs. The dominant tone present has a very high amplitude of ~20 dBsm and corresponds to a frequency ~60Hz, which is of serious concern from a structural stand point for the full-scale aircraft.

For the blowing rates studied, LEMB was not effective in suppressing the OASPLs in the bay for the outboard door closed configuration. However, a simple, passive control concept involving "baffles" has been shown to result in significant reduction in the tonal amplitude. Their principle mode of suppression is based on "physically" impeding or cutting-off of the feedback loop mechanism responsible for the setting up of resonant tones. The baffles hence satisfy both an easily "integratable" and working control concept over a wide operational range of the aircraft.

Modeling of a store placed in the opening of a generic L/D=6 cavity submerged in a Mach 3.5 free stream flow field was accomplished to help establish criteria for the development of a telemetry system suitable for small-scale store drop testing. Results indicate a frequency range of interest, and also provide information on the magnitude of the fluctuating components of the store loads. Further modeling to look at dynamic pressure effects on the fluctuating store loads components will be accomplished in the near future. Fully time-accurate simulations of representative drop tests are also planned.

Acknowledgments

The support of Al Piranian (JSF Program Office) and Jim Grove (AFRL/RBAI) and the computing time provided by the HPCMP Office for this effort are gratefully acknowledged.

References

1. Arunajatesan, S., Sinha, N., Stanek, M.J., Grove, J.E., and Johnson, R.A., "High Performance Computational Modeling of Unsteady Surface Loads in Complex Weapons Bays", *Proceedings of the DoD HPCMP Users Group Conference 2009*, San Diego, CA, June 15–18, 2009.

2. Johnson, R.A., Kannepalli, C., Arunajatesan, S., and Sinha, N., "Computational Modeling of Geometrically-Complex Weapons Bays", *Proceedings of the DoD HPCMP Users Group Conference 2010*, Schaumburg, IL, June 14–17, 2010.

3. Johnson, Rudy, Stanek, Michael, and Grove, James, "Store Separation Trajectory Deviations due to Unsteady Weapons Bay Aerodynamics", *AIAA-2008-188*, 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 2008.

4. Murray, R.C. and Elliot, G.S., "Characteristics of the Compressible Shear-Layer over a Cavity", *AIAA Journal*, Vol. 39, No. 5, pp. 846–856, May 2001.

5. Sinha, N., Hosangadi, A., and Dash, S.M., "The CRAFT NS code and preliminary applications to steady/unsteady reacting, multi-phase jet/plume flow field problems", *CPIA Pub. 568*, May 1991.

6. Rai, M.M. and Moin, P.," Direct numerical simulation of transition and turbulence in spatially-evolving boundary-layer", *Journal of Computational Physics*, Vol. 109, pp.162–192, 1993.

7. Kannepalli, C., Arunajatesan, S., and Dash, S.M., "RANS/LES Methodology for Supersonic Transverse Jet Interactions with Approach", *AIAA-2002-1139*.

8. Papp, J. L. and Dash, S. M., "Turbulence Model Unification and Assessment for High-Speed Aero-propulsive Flows", *AIAA Paper 2001-0880*, Jan. 2001.

9. Arunajatesan, S. and Sinha, N., "Large-Eddy Simulations of Supersonic Impinging Jet Flow Fields", *Paper No. AIAA-2002-4287*, 38th AIAA/ASME/SAE/ASEE Propulsion Conference & Exhibit, Indianapolis, IN, July 7–10, 2002.

10. Kannepalli, C., Arunajatesan, S., Calhoon, W.H., Jr., and Dash, S.M., "Large-Eddy Simulations of High-Speed Flows", *Paper No. HT-FED2004-56162*, 2004 Joint ASME-JSME Fluids Engineering Summer Conference, Charlotte, NC, July 11–15, 2004.

11. Calhoon, W.H., Jr., Kannepalli, C., and Dash, S.M., "LES Studies Of Scalar Fluctuations at High Convective Mach Numbers", Third AFOSR International Conference on Direct Numerical Simulation and Large Eddy Simulation (TAICDL), University of Texas at Arlington, Arlington, TX, USA, August 5–9, 2001.

12. Arunajatesan, S. and Sinha, N., "Hybrid RANS-LES Modeling for Cavity Aero-acoustics Predictions", *International Journal of Aeroacoustics*, Vol. 2, No. 1, pp. 65–91, 2003.

13. Arunajatesan, S., Shipman, J., Cavallo, P.A., Birkbeck, R, Sinha, N., Ukeiley, L.S. and Alvi, F.A., "Flow Control for Enhanced Store Separation", *AIAA Paper 2007-1239*.

14. Shipman, J.D., Arunajatesan, S., Cavallo, P.A., Sinha, N., Ukeiley, L. and Alvi, F., "Flow Control for Enhanced Store Separation", *Paper No. AIAA-2007-1236*, 45th Aerospace Sciences Meeting and Exhibit, Reno, NV, January 8–11, 2007.

15. Ukeiley, L.S., Sheehan, M. Coiffet, F., Alvi, F.S., Arunajatesan, S., and Jansen, B., "Control of Pressure Loads in Complex Cavity Configurations", *Paper No. AIAA-2007-1238*, 45th Aerospace Sciences Meeting and Exhibit, Reno, NV, January 8–11, 2007.

16. Arunajatesan, S., Kannepalli, C., Sinha, N., Sheehan, M., Shumway, G., "Suppression of Cavity Loads Using Leading-Edge Blowing Concepts", *AIAA Journal*, Vol. 47, No. 5, 2009.

17. Rossiter, J.E., "Wind-Tunnel Experiments on the Flow over Rectangular Cavities at Subsonic and Transonic Speeds", *Aeronautical Research Council, Reports & Memoranda No. 3438*, London, England, October 1964.

18. Heller, H.H. and Bliss, D.B., "The Physical Mechanism of Flow-Induced Pressure Fluctuations in Cavities and Concepts for Their Suppression", *AIAA Paper 75-491*, March 1975.

19. Kannepalli, C., Chartrand, C., Birkbeck, R., Sinha, N., and Murray, N.", Computational Modeling of Geometrically-Complex Weapons Bays", *AIAA Paper 2011-2774*, 17th AIAA/CEAS Aeroacoustics Conference, Portland, OR, 6–8 June 2011.

20. Murray, N. "Enhanced Acoustical Environment for Modern Weapons Bays", *Interim Report, AFRL SBIR Phase II Subcontract No. 09-C-3906/C390*, National Center for Physcial Acoustics, University of Mississippi.

21. Dix, R.E. and Dobson, T.W., Jr., "Database for Internal Store. Carriage and Jettison, Vol. 1", *AEDC-TR-90-23*, Arnold Eng. Develop. Center, November 1990.

22. Heller, H.H., Holmes, G., and Covert, E.E., "Flow-Induced Pressure Oscillations in Shallow Cavities", *AFFDL-TR-70-104*, DTIC Accession Number: AD880496, Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, OH, December 1970.

High-Fidelity LES of Asymmetric Supersonic Jets

Joseph W. Nichols, Frank E. Ham, and Sanjiva K. Lele
Center for Turbulence Research, Stanford University, Stanford, CA
{jwn, fham, lele}@stanford.edu

Yaser Khalighi
Cascade Technologies, Inc., Palo Alto, CA
khalighi@cascadetechnologies.com

John Spyropoulos
US Naval Air Systems Command (NAVAIR), Patuxent River, MD
john.spyropoulos@navy.mil

Abstract

This Capability Applications Project Phase-II (CAP-II) project is motivated by the need to reduce noise emitted from hot supersonic turbulent jets associated with tactical aircraft engine exhausts. The extreme levels of acoustic radiation emitted from these jets renders even the most advanced hearing protection worn by aircraft carrier deck crews insufficient, mandating the need to reduce jet noise at its source, through modifications to nozzle geometry, for example. The specific objective of this research is to assess the capability of high-fidelity large-eddy simulation (LES) for the prediction of far-field noise from hot supersonic turbulent jets from complex nozzles, to aid in the design of innovative noise-reducing nozzles. Using the CAP computer time, a series of five LES databases were generated for three different nozzle geometries, over a range of different operating conditions. The flow in and around each nozzle was solved on unstructured meshes containing up to 137 million control volumes. The massively parallel code demonstrated strong scalability up to 20,000 cores. A Ffowcs-Williams-Hawkings surface technique was applied to the LES databases to efficiently project the simulated near-field acoustics to the far-field. In the case of an under-expanded rectangular nozzle (precisely matching one fabricated for laboratory experiments), the downstream (peak) far-field acoustic radiation was found to be directed mostly along the rectangle's minor axis, with a 3dB relative reduction in the plane of the rectangle's major axis. The unheated rectangular jet was found to be strongly screeching; however, whereas the heated rectangular jet was not.

1. Introduction

This project is motivated by the growing need for innovative strategies to reduce the noise produced by high-performance supersonic military aircraft. Noise reduction techniques have a direct impact on the health and safety of aircraft carrier flight deck personnel. In FY2011, the US Department of Defense (DoD) is projected to spend more than $1 billion in compensatory hearing-loss claims alone. The exhaust noise of F/A-18E/F Super Hornets, EA-18G Growlers and other high-performance, tactical aircraft is consistently at the 150 dB level (NRAC, 2009). Advanced sound protection, including custom-molded earplugs and form-fitted cranials (see Figure 1), may provide flight deck crew up to 43 dB of protection; but even so, sound intensity levels remain well-above the 85 dB level at which sustained exposure begins to cause permanent hearing-loss. The conclusion of the NRAC panel was that the solution of the jet engine noise problem will require a multi-pronged approach, an important component of which is "reducing jet engine noise source, which requires a long-term research program." In addition to enhancing protection of aircraft carrier personnel, noise reduction techniques have the potential for considerable impact for regions where communities and military bases are in close proximity. In these increasingly common situations, deployment of effective sound-reduction technology would minimize civilian complaints and aid in compliance with noise regulations.

The aero-acoustic phenomenon of jet noise, intricately connected to the turbulence dynamics in the jet shear layers and in the region downstream of the jet potential core, remains a large component of the overall noise generated by supersonic aircraft, despite significant scientific investigation as to its cause. With ever-increasing modern computational capacity; however, Large-Eddy Simulation (LES) has begun to shed new light on this difficult scientific problem. Bodony and Lele (2008) provide a review of the state-of-the-art predictive quality of LES with respect to high-speed jet aero-acoustics. The extreme computer power afforded by the High Performance Computing Modernization Program's (HPCMP's) Capability

Applications Project (CAP) program helped to advance this state-of-the-art quality towards the simulation of practical noise reduction methodologies. In particular, current jet noise reduction techniques include the addition of tabs or chevrons to the nozzle rim (Liu, et al., 2009), alteration of the cross-sectional shape of the nozzle (Tam and Zaman, 2000), beveling of the nozzle exit (Viswanathan, et al., 2008), and the addition of an array of micro-jets around the nozzle perimeter. In each case, significant three-dimensional (3D) structure is added to the nozzle geometry, challenging traditional LES codes relying on a structured mesh. Unstructured LES; however, is ideally-suited to resolving complex geometry, although traditionally it has not been used for aero-acoustic calculations because of its formal 2nd-order accuracy. The compressible Navier-Stokes solver "CharLES" used in this project helps to remedy this problem by introducing numerical stencils based on higher-order polynomial expansions to minimize dissipation. In this project; therefore, our aim is to assess high-fidelity LES on unstructured meshes as a tool for the prediction, characterization, and control of supersonic jet noise from geometrically-complex nozzles over a range of operating conditions.

Figure 1. Aircraft carrier flight deck crew at takeoff (F/A-18E/F Superhornet). Full-cranial hearing protection is worn by all visible personnel. Jet blast deflector is visible at left.

2. Numerical Methodology

The "CharLES" code solves the spatially-filtered compressible Navier-Stokes equations on unstructured grids using a novel control volume-based finite volume method, where the flux is computed at each control volume face using a blend of a non-dissipative central flux and a dissipative upwind flux, i.e.,:

$$F=(1-\alpha)\, F_{central} + \alpha\, F_{upwind}$$

where $0 < \alpha < 1$ is a blending parameter. This blending approach is often the basis of implicit approaches to LES, where the blending parameter is selected as a global constant with a value large enough to provide all the necessary dissipation (and potentially quite a bit more). For example, in the turbulent jet literature, Tucker (2004) used this approach and reported that the smallest "usable" value of blending parameter was determined to be $\alpha=0.25$. The treatment is described in detail by Shur, et al. (2003). In later work, Xia, et al. (2009) reported that the minimum value of blending parameter was set to 0.1 "to avoid numerical instability." CharLES does not use the implicit LES approach—an explicit sub-grid scale model is used to model the effect of sub-grid scales. To minimize numerical dissipation relative to implicit LES approaches, the value α is allowed to vary spatially such that it can be set to zero in regions where the grid quality is good and the scheme based on the central flux is discretely stable and non-dissipative. In regions of less-than-perfect grid quality; however, the central scheme can introduce numerical instabilities that must be prevented from contaminating/destabilizing the solution by locally increasing α.

Because the supersonic jets from engine exhausts are many times not perfectly expanded, shocks form within the jet plume and are important flow features in the context of aero-acoustics. CharLES uses a hybrid Central-WENO scheme to simulate flows involving shocks. The scheme has three pieces:

1. A central scheme, described previously.
2. A scheme appropriate for computing a flux across a shock.
3. A hybrid switch, which detects where shocks are present in the flow, and activates the shock-appropriate scheme.

For the shock-appropriate scheme, CharLES uses a 3rd-order WENO method to perform reconstructions (Shi, et al., 2002), and the HLLC approximate Riemann solver to compute the flux (Harten, et al., 1983). The WENO method is fully unstructured and, as such, must consider a potentially large number of candidate stencils. In regions of nearly uniform orthogonal grids;, however, the number of candidate stencils reduces to two stencils per face-side (i.e., two for the left-side value and two for the right-side value at each face), substantially reducing the cost of the method over a large fraction of

the grid. The hybrid switch is based on the method developed originally by Hill and Pullin (2004), where the magnitudes of the smoothness parameters computed as part of the WENO reconstructions are compared to identify the presence of flow discontinuities.

To better quantify our understanding of the acoustic field produced by the rectangular jet, we assume the acoustic information captured by LES to be valid only in the well-resolved turbulence-containing regions and their immediate vicinity. We capture this acoustic information as it leaves the turbulence on a FWH surface tightly encompassing the turbulent regions (see Figure 2). This information is then propagated losslessly to the far-field through the solution of the FWH equation integrated over this surface. As long as the flow is irrotational outside of the FWH surface, we may neglect the computationally-expensive quadrupole volume integral present in this equation, yielding an efficient procedure for calculating the far-field noise. Note that the optimal position of the FWH surface is found by trading-off numerical resolution (better closer to the jet), and accuracy with respect to capturing the quadrupole-containing regions (better away from the jet).

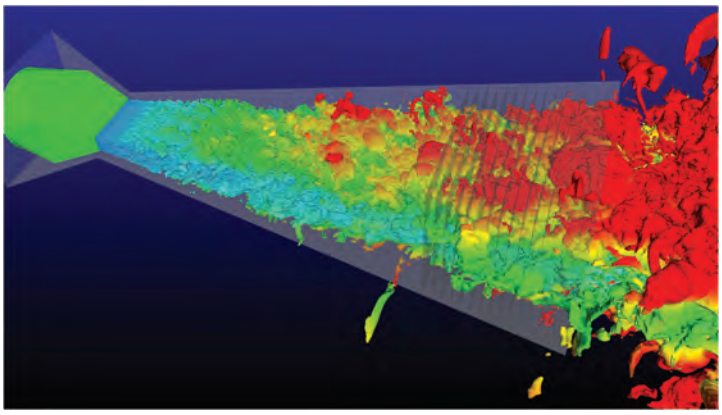


Figure 2. Unstructured LES applied to a complex nozzle geometry (left). Turbulence is visualized by an iso-surface of temperature, colored by a normalized distance away from the jet axis. The transparent surfaces represent a Ffowcs-Williams Hawking surface used for projecting sound information to the far-field.

Through a sensitivity study of FWH surface placement, we find that we may be rather aggressive in placing the FWH surface very close to turbulence-containing regions. Figure 2 shows a moderately aggressive FWH surface, designed for the isothermal rectangular jet. For visualization purposes, the three-dimensional iso-surface of temperature has been colored by distance away from the jet axis. As shown, this FWH surface does a fairly good job at avoiding most of the turbulence. The FWH integral equations require; however, a closed surface. This necessitates that the surface eventually passes through the main jet column. This challenge is mitigated through the technique of end-cap averaging where several (16 in the present case) closing end-cap surfaces are considered and the acoustic results are averaged, to statistically remove the effects of pseudo-sound produced by vortices convecting through any particular end-cap. For further details, Mendez, et al. (2009) presents a careful investigation of the effects of FWH surface placement and end-cap averaging for round supersonic jets.

3. Results

During HPCMP CAP Phase-II on the new Cray XE6 at US Army Engineer Research and Development Center (ERDC) [garnet], we generated a series of six datasets corresponding to snapshots of high-fidelity LES of supersonic jets, summarized in Table 1. Three of the runs simulated jets from round converging-diverging nozzles and three simulated jets from a rectangular nozzle of aspect ratio 4:1. In terms of numbers of control volumes, the unstructured mesh sizes ranged from 40 million to 137 million elements. The simulations were run on the full 20,000 cores of the Cray XE6 in increments of 24-hours, switching off between the other CAP Phase 2 users. Our code showed very favorable scalability, as will be discussed below. These databases reside in archival storage at ERDC and are currently continuing to be used for post-processing and analysis, probing the physics of jet noise production. In this paper, we will focus on the rectangular jet cases, in order to study the performance of our aero-acoustic methodology applied to complex geometry (see Figure 3). In particular, case R8 will be used to validate the simulation results against data collected in laboratory experiments performed at NASA Glenn Research Center using the same nozzle geometry and operating conditions (Frate and Bridges, 2011). After this first validation step we further assess the capability of our simulation to treat jets at a range of operating conditions,

including a heated case representative of a real propulsion configuration. At Mach 1.2, cases R1 and R2 will be used for this purpose, a comparison of which will be used to study the effect of heating on the overall far-field noise.

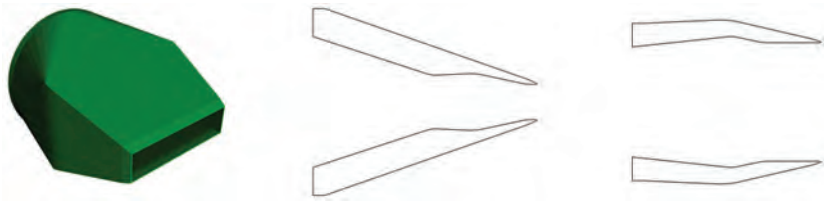


Figure 3. Rectangular nozzle geometry: (left) 3D view of the nozzle exterior; (middle) mid-plane cross section along minor axis; (right) mid-plane cross section along major axis

Table 1. Summary of high-fidelity LES datasets

Name	Shape	Mach	Expansion	Heating	Mesh size	Storage	Flow-through times
A1	Round	1.5	ideal	isotherm.	42M	8TB	100
A3	Round	1.5	over	heated	42M	8TB	170
R1	Rectangular	1.2	under	isotherm.	86M	15TB	170
R2	Rectangular	1.2	under	heated	86M	15TB	100
R8	Rectangular	1.4	under	Isotherm.	86M	15TB	100
SMC015	Round	1.4	ideal	isotherm.	137M	30TB	264

3.1 Supersonic Jet Flow from a Rectangular Nozzle

Thanks to the unstructured capability of the CharLES solver, the entire turbulent flow was simulated in and around each nozzle. In particular, the interior geometry of the rectangular nozzles involved a complicated sequence of contractions and expansions as shown in Figure 3. To examine the jet flow in further detail, Figures 4 and 5 show cross-sections of the supersonic jet produced by case R1. The streamwise development of the rectangular jet is visualized in Figure 4, which displays contours of instantaneous temperature on cross-sections normal to several different axial stations along the jet axis. Because this supersonic jet is under-expanded, a series of shock cells form in the jet column just downstream of the nozzle, as shown in Figure 5. We observe the shock cells to have a complex 3D structure, owing to the rectangular shape of the nozzle. Because of the high Reynolds number, healthy turbulence forms in the shear layers of the jet and spreads as the jet develops downstream. Note that the shear layers merge (or the potential core collapses) more quickly in the direction of the minor axis than in the direction of the major axis. This produces a region of intense intermittency in the z=0 plane just upstream of the collapse of the potential core as the center of the jet begins to be influenced by the shear layers in the positive and negative z-directions, before feeling any influence from the y-directions.

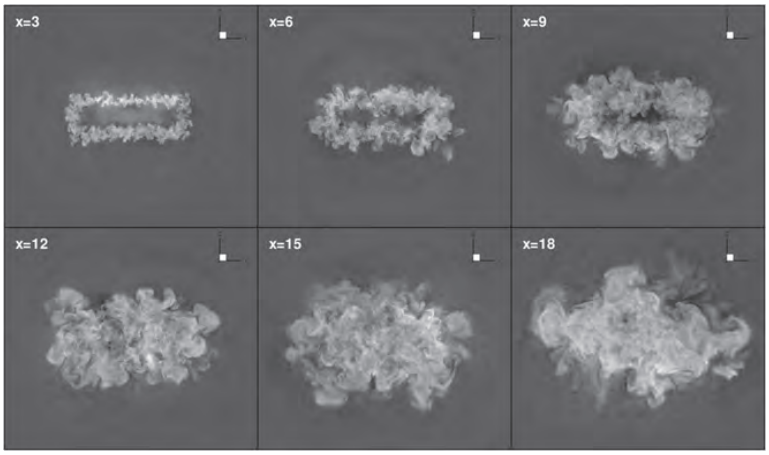


Figure 4. Instantaneous temperature contours on axial cross-sections from various streamwise stations, showing the spatial development of the turbulent mixing associated with the rectangular shear layers

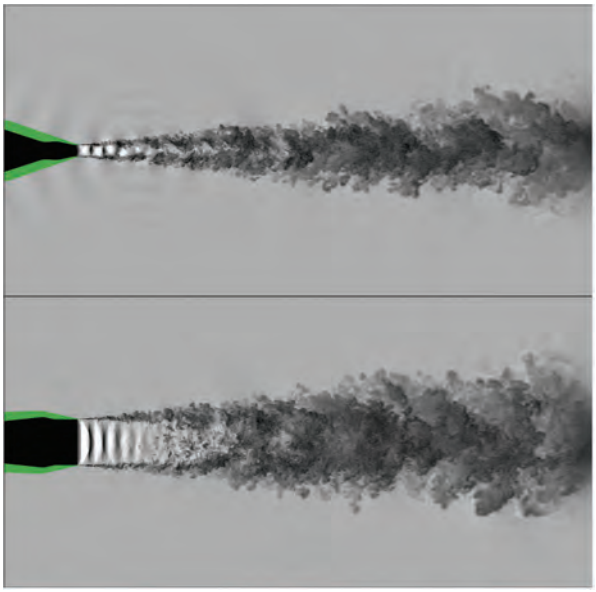


Figure 5.Instantaneous snapshots of the temperature of the nominally-isothermal supersonic rectangular jet, taken along mid-plane cross-sections y=0 (top) and z=0 (bottom). Black and white regions correspond to hot and cold regions, respectively.

3.2 Validation of Far-field Noise

Figure 6 provides a comparison of far-field sound spectra measured from an experimental investigation (Bridges, 2011, private communication) to spectra extracted from three LES simulations of the same geometry. The LES simulations are identical in every respect except for spatial resolution. In the figure, the columns show three different polar angles φ (as measured from the upstream jet axis), which from left to right are representative of upstream, lateral, and downstream acoustic radiation, respectively. The azimuthal angle θ=0° is aligned with the nozzle's minor axis (top row), and θ=90° with the nozzle's major axis (bottom row). In the laboratory experiment, the far-field sound was measured at a distance of 100 De away from the center of the nozzle exit plane. Likewise, in the simulations, the far-field sound was obtained by projecting acoustic information from the FWH surface to the same distance. Most importantly, the amplitudes of the resulting spectra were not scaled in any way. In figure 6, the sound pressure level (SPL) is plotted as a function of Strouhal number, defined as $St=f D_e/u_j$ where D_e is the nozzle equivalent diameter, u_j is the fully-expanded jet velocity, and f is the cyclic frequency. All of the spectra, including the laboratory experiment, were generated by summing the energy of the original signals in 256 logarithmically-spaced frequency bins over the range $0.02<St<9.5$. For the simulation spectra, all available symmetries of the nozzle geometry were employed to average the results. Note that 4-way symmetry is available to θ=30° and 60°, whereas results at θ=0° and 90° must rely on 2-way symmetry only. Therefore, the signal-to-noise ratio is a factor of two greater in the cases of θ=30° and 60°.

For φ=60° and θ=0° (upstream, minor axis), the experimental spectrum and all of LES spectra capture a strong screech tone at St=0.3. In agreement with the previous flow visualizations, the screech tone is directed mainly in the direction of the nozzle's minor axis, and it diminishes as θ increases from 0° to 90°. At φ=90°, the first harmonic of the screech tone (St=0.6) becomes dominant over the fundamental, but both still diminish as θ approaches 90°. The screech tone also appears in the downstream spectra, and is again directed mainly along the minor axis. At this polar angle; however, the fundamental tone has re-asserted itself as the dominant frequency. Overall, the unstructured LES/FWH methodology is able to accurately predict the frequency and, in most cases, the amplitude of the screech tones, giving us confidence that the simulation is accurately capturing the physical mechanisms responsible for this effect. A possible exception to this observation is that the simulation predicts a strong first harmonic tone for φ=90° and θ=60°, whereas the experiment suggests otherwise. It is interesting to note; however, that the experimental data at the surrounding azimuthal angles both suggest a dominant first harmonic and weaker fundamental, so that the θ=60° does not seem consistent. On the other hand, θ=60° is oriented such that it nearly passes through the corner of the aspect ratio 4:1 rectangular nozzle, so that corner effects may play a role.

In addition to the screech tones, Figure 6 shows that the simulations are also able to capture the shape and amplitude of the broadband spectra at most angles, although resolution effects become more apparent. For example, all of the simulations are able to capture the peak broadband acoustic radiation, which for this jet occurs close to polar angle φ=150°. At this important downstream angle, the shapes of the experimental spectra are also faithfully reproduced.

For φ=90°, the simulation spectra agree fairly well with the experimental spectra at intermediate and high frequencies, with the possible exception of θ=60°. The low-frequency noise tends to be slightly under-predicted by the simulations; however. The signal-to-noise ratio degrades in this region; however, owing to the relative sparsity of samples obtained at the low-end of the spectrum. Therefore, the low-frequency end of the spectrum always tends to suffer from lack of convergence, which may only be remedied by running the simulations for an even longer time. At the upstream angles (φ=60°), the low-frequency spectra again are slightly under-predicted, but overall agree more closely than for φ=90°. Finally, for both φ=60° and 90° at θ=60°, the high-frequency noise is over-predicted by 5–10dB. As before, this could owe to corner effects. Comparing only the experimental spectra at θ=30°, 60°, and 90°; however, we observe that the spectrum at θ=60° is consistently about 10dB quieter than the spectra at neighboring angles. The corresponding spectra from the simulation show no such dramatic reduction at this azimuthal angle.

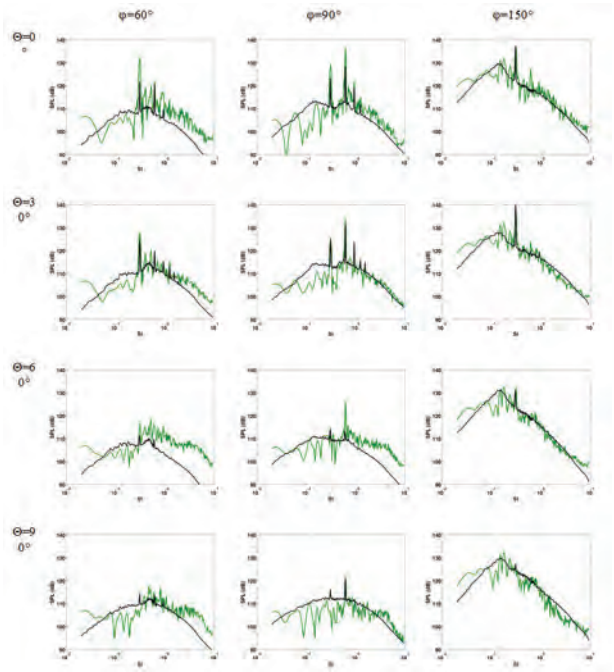


Figure 6. Comparison of laboratory (black) and simulation (green) far-field spectra measured at locations on a sphere of radius 100 De at different polar (φ) and azimuthal (θ) angles. The axes in each frame are scaled identically.

3.3 Effect of Heating on Far-field Noise

Using the FWH solver, we project the sound from both the isothermal and heated M=1.2 rectangular jets to a sphere of radius 100h (where h is the length of nozzle's narrow dimension) centered at the nozzle exit. Figure 7 is a spherical representation of the far-field directivity, where the radial position of the surface is proportional to the overall noise power levels. The far-field sound from the isothermal jet is composed of three lobes, with strongest components in the direction of the nozzle minor axis. The spectral content and azimuthal directivity of these lobes are analyzed in Figure 8. Lobe A is strongly tonal, and directed almost exclusively along the minor axis of the nozzle, and thus is interpreted as the fundamental screech tone of the nozzle. From animated flow visualizations (not shown), it is apparent that this screech tone is generated by lateral oscillations of the shock cells, which couple with a strong acoustic wave which propagates upstream along the nozzle's exterior surface, visible in the y=0 cross-section shown in Figure 5.

At 95 degrees to the upstream jet axis, lobe B is still directed along the minor axis of the nozzle, but contains a strong component at the first harmonic of the screech frequency, as well as the fundamental. For this reason, we interpret the lateral lobe B to be the first harmonic screech mode. The downstream acoustic radiation is somewhat less energetic than the extremely strong screech tones observed for this jet, but is more homogeneous in its azimuthal directivity. The sideline sound; however, is approximately 3dB less than the vertical, suggesting that a shaped far-field acoustic field is indeed possible from this geometry. Finally, we note from Figure 7 that heating completely eliminates the screech tones, as expected, and has the added benefit of directing the downstream jet noise a bit more in the vertical directions.

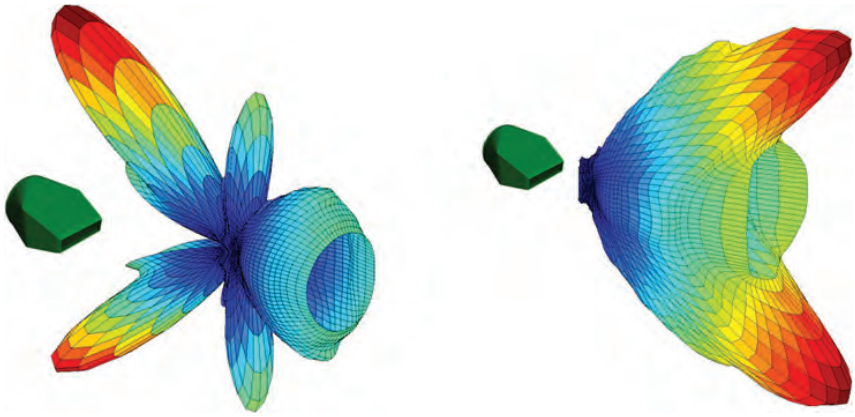


Figure 7. Spherical plots of far-field directivity, for the (a) isothermal and (b) heated rectangular supersonic jets. The nozzle orientation is shown for reference, but should be located at the origin. The radial location of the spherical surfaces is proportional to the far-field noise power levels.

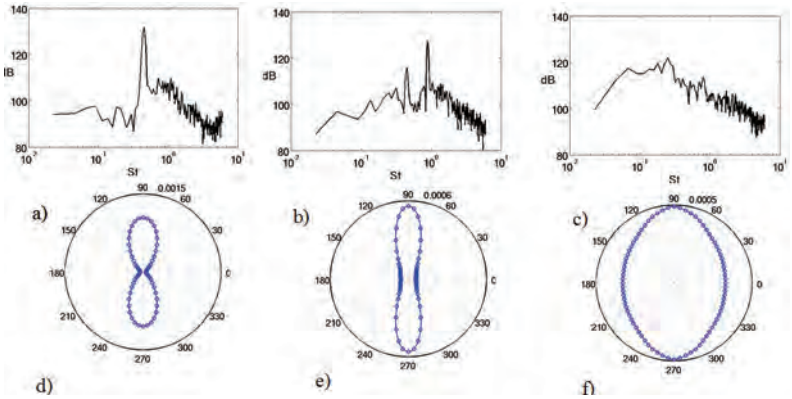


Figure 8. Spectral content (a-c) and azimuthal directivity (d-f) for the three lobes of the isothermal jet directivity. (a) phi=37.5, theta=90; (b) phi=95, theta=90; (c) phi=145; theta=90. The azimuthal directivities have corresponding angles: (d) phi=37.5; (e) phi=95; (f) phi=145.

4. Scalability Testing

The CharLES code is designed using Message Passing Interface (MPI) to function well in a massively-parallel, distributed memory environment. Because the numerical method is purely hyperbolic and fully explicit, the code requires relatively little communication at each time-step, as compared to other methodologies. Specifically, all-to-all communication is avoided so that the communication scales as p log p at worst, where p indicates the number of processors. This enables the code to scale well in massively-parallel environments. Figure 9 shows scaling statistics (speed-up vs. number of cores) for the 137M grid point problem computed during the Cray XE6 testing period, demonstrating that the CharLES code still exhibits strong scaling at the 20,000 core level. The algorithm efficiency, defined as the speed-up divided by the number of participating processors, remains at 96.4% for 19,584 cores compared to 100% at the 2,048 core baseline.

In addition to the scalability of the computation and communication, input/output (I/O) performance must also be taken into account. Specifically, as the CharLES code runs, data is collected by periodically writing out "snapshots" of 3D fields containing density, momentum, and energy. As discussed above, the snapshots are then post-processed in a variety of ways to extract turbulence and acoustic information. The number of time-steps taken between snapshots is referred to as the I/O interval. The number of cores also affects the I/O performance relative to the computational performance. During the CAP testing period, we measured the data-write rate for typical snapshot files (4GB each). As Figure 10(a) shows, the write rate decreases with the number of cores, which is to be expected, since as the number of processors becomes large, contention for a fixed number of disks in the Lustre File System increases. Because this trend is opposite to that for computational performance, there is a trade-off to be had between computational and I/O performance. For a mesh containing 100 million control volumes, Figure 10(b) shows the percentage of the total calculation time consumed by the I/O as a function of the

number of cores and the I/O interval. At the 4,096 core level, all of the I/O intervals considered yield an I/O time which is less than 10% of the total calculation time. At the 8,192 core level, this remains true for I/O intervals of 400 and 800. From our CAP calculations we found an I/O interval of 500 to be sufficient.

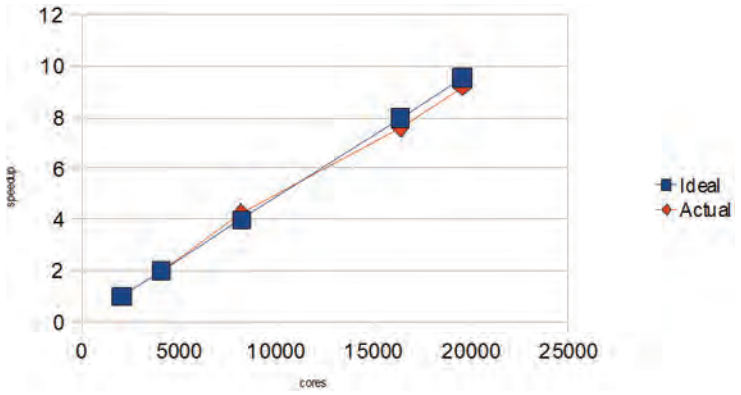


Figure 9. Scalability of CharLES code expressed as speedup vs. number of cores (up to 20,000)

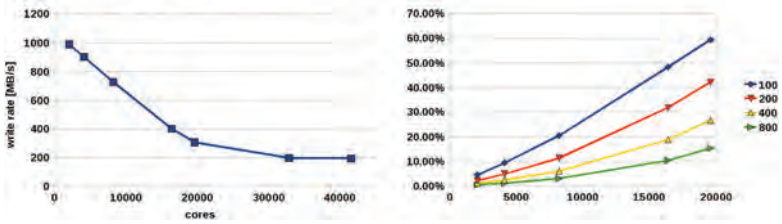


Figure 10.(a) Measured write rate vs. number of cores on the Cray XE6 (4GB file size) – the last two data points were measured during CAP Phase 1 testing on raptor at AFRL. (b) Percentage of total calculation time consumed by I/O as a function of I/O interval and number of cores.

5. Conclusion

HPCMP CAP-II results from three high-fidelity simulations of supersonic jets from rectangular nozzles were presented. Flow visualizations showed that a large range of turbulent scales were captured by the simulations, in addition to the shocks produced by the jet's under-expansion. The supersonic jet transitions from a rectangular cross-section to a circular cross-section as the turbulence develops downstream, with no axis-switching. For the isothermal jets, a strong flapping instability was observed, coupled at the same frequency to a strong screech mode, with peak amplitude in the upstream direction. Direct comparison to laboratory experiments showed that the computational aeroacoustic methodology was able to capture the screech tone and its harmonics with great accuracy. The broadband acoustic spectrum extracted from the simulation tracked well with those obtained in the laboratory in terms of shape and amplitude, with the possible exception of azimuthal angle θ=60° in the upstream and lateral directions. In the heated jet, the screech tones were eliminated, and increased the directivity of the peak downstream acoustic radiation along the minor axis of the nozzle. Finally, through the CAP program, we demonstrated that our computational infrastructure scales well at the 20,000 core and beyond.

Acknowledgments

This research was partially supported by NASA grant no. NNX07AC94A. Computational resources were provided by a DoD FY2011 Capability Applications Project at the ERDC supercomputing center.

References

Bodony, D.J. and Lele, S.K., "Current status of jet noise predictions using Large-Eddy Simulation", *AIAA Journal*, Vol. 46, No. 2, pp. 364–380, 2008.

Bridges, J.E., E-mail message, Acoustics branch, NASA Glenn Research Center, March 9, 2011.

Frate, F.C. and Bridges, J.E., "Extensible rectangular nozzle model system", *AIAA 2011-975*.

Harten, Lax, P.D., and van Leer, B., "On upstream differencing and Godunov-type schemes for hyperbolic conservation laws", *SIAM Review*, Vol. 25, pp. 35–61, 1983.

Hill, D.J. and Pullin, D.I., "Hybrid tuned center-difference-WENO method for large-eddy simulations in the presence of strong shocks", *Journal of Computational Physics*, Vol. 194, No. 2, pp. 435–450, 2004.

Liu, J., Kailasanath, K., Ramamurti, R., Munday, D., Gutmark, E., and Lohner, R., "Large-Eddy Simulations of a supersonic jet and its near-field acoustic properties", *AIAA Journal*, Vol. 47, No. 8, pp. 1849–1864, 2009.

Mendez, S., Shoeybi, M., Sharma, A., Lele, S.K., and Moin, P., "Post-processing of Large-Eddy Simulations for jet noise predictions", *Center for Turbulence Research Annual Research Briefs*, Stanford, CA, 2009.

Naval Research Advisory Committee Report on Jet Engine Noise Reduction, April, 2009.

Shi, J., Hu, C., and Shu, C.-W., "A technique of treating negative weights in WENO schemes", *Journal of Computational Physics*, Vol. 175, pp. 108–127, 2002.

Shur, M.L., Spalart, P.R., Strelets, M., and Travin, A.K., "Towards the prediction of noise from jet engines", International *Journal of Heat and Fluid Flow*, Vol. 24, pp. 551–561, 2003.

Tam, C.K.W. and Zaman, K.B.M.Q., "Subsonic jet noise from non-axisymmetric and tabbed nozzles", AIAA Journal, Vol. 38, No. 4, pp. 592–599, 2000.

Tucker, P.G., "Novel MILES computations for jet flows and noise", *International Journal of Heat and Fluid Flow*, Vol. 25, pp. 625–635, 2004.

Viswanathan, K., Shur, M., Spalart, P.R., and Stretlets, M., "Flow and noise predictions for single- and dual-stream beveled nozzles", *AIAA Journal*, Vol. 46, No. 3, pp. 601–626, 2008.

Xia, H., Tucker, P.G., and Eastwood, S., "Large-eddy simulations of chevron jet flows with noise predictions", International *Journal of Heat and Fluid Flow*, Vol. 30, pp. 1067–1079, 2009.

Highly Accurate Simulations of Flapping Wings for Micro Air Vehicle Applications

Raymond E. Gordnier Miguel R. Visbal, and Donald P. Rizzetta
US Air Force Research Laboratory, Air Vehicles Directorate, Computational Aerophysics Branch (AFRL/RB), Wright-Patterson AFB, OH
{raymond.gordnier, miguel.visbal, donald.rizzetta}@wpafb.af.mil

Abstract

In this paper, high-fidelity, multidisciplinary simulations are performed to elucidate the fundamental underlying physics associated with flexible and flapping flight for micro air vehicles (MAVs). Simulations for three separate canonical problems associated with flapping flight are presented. The first computations provide a description of the three-dimensional unsteady separation process induced by large-amplitude heaving oscillations of a low-aspect-ratio wing under low-Reynolds number conditions. The second problem investigates the impact of spanwise flexibility on the aerodynamic performance of a rectangular wing undergoing a pure plunging motion. Finally, the use of plasma-based actuation to modify the transitional flow and improve aerodynamic performance of a flapping-wing section at low-Reynolds numbers is explored. The flowfields are computed employing an extensively validated high-fidelity implicit large-eddy simulation (ILES) approach found to be effective for moderate-Reynolds number flows exhibiting mixed laminar, transitional and turbulent regions. For the flexible case, the flow solver is coupled with a structural solver that decomposes the equations of three-dimensional elasticity into cross-sectional, small deformation and spanwise, large-deformation analyses for slender wings. For the plasma control case, the flow solver was augmented by source terms used to represent plasma-induced body forces imparted by the actuator on the fluid. A simple phenomenological model provided the body force resulting from the electric field generated by the plasma.

1. Introduction

Micro air vehicles (MAVs) designed to operate in a complex urban environment will be required to have a number of unprecedented features to successfully carry out their mission requirements. These vehicles will need to be highly-agile in order to avoid the man-made obstacles present in their path or operate in the interior of buildings. They will be required to hover or perch in a location for extended periods of time to carry out proposed missions and energy harvesting tasks to remain on station. Operations will also be carried out in an environment where gusts and crosswinds with spatial sizes of the order or greater than the vehicle size and velocity will be present. These vehicles will need to be highly-adaptable to rapidly changing environmental conditions to prevent erratic and poorly-controlled flight.

The above demanding requirements have driven interest in developing MAVs dependent on flexible, flapping-wings. The flapping-wing is able to simultaneously provide both lift and thrust for the vehicle. The inherent flexibility of the structure may be used to adapt to unsteady aerodynamic loadings (gusts and crosswinds) and provide for very agile flight maneuvers. Flapping-wing MAVs also have the potential to effectively meet the requirements for both hovering and perching.

Improved understanding of the complicated, multidisciplinary physics critical to MAV design demands high-fidelity modeling capabilities. In the present paper a high-fidelity, multidisciplinary computational solver is employed to accomplish these types of challenging simulations. For the Navier-Stokes solver spatial derivatives are computed using high-order (up to sixth-order) compact schemes and a Pade-type low-pass filter is used to ensure stability. The high-order scheme is essential for the accuracy demanded by the transition process, whereas the discriminating low-pass filter operator provides regularization in turbulent flow regions in lieu of a standard sub-grid-scale (SGS) model. This provides a novel, unified approach to treat mixed laminar/transitional/turbulent flows termed implicit large-eddy simulation (ILES). This powerful fluid dynamics solver is coupled with a nonlinear finite element structural model that decomposes the equations of three-dimensional (3D) elasticity into linear cross-sectional and nonlinear spanwise structural analyses appropriate for slender

wings to address issues of wing flexibility. The solver is also coupled with a simple phenomenological model to represent a Dielectric-barrier-discharge (DBD) actuator to explore plasma-based flow control techniques to improve the aerodynamic performance of flapping-wing flight.

Computations for three canonical problems associated with flapping-wing flight for MAVs are presented in this paper. The first problem investigates the three-dimensional unsteady separation process induced by large-amplitude heaving oscillations of a low-aspect-ratio wing under low-Reynolds-number conditions. Calculations are performed for an aspect ratio two rectangular plate at a chord Reynolds number $Re_c=10^4$ and mean angle-of-attack $\alpha_o=8°$ plunging with reduced frequency $k=1.0$ and non-dimensional amplitude $h_o/c=0.25$. The complex unsteady 3D flow structure generated during dynamic stall of a low-aspect-ratio wing is elucidated. In the second problem, the impact of spanwise flexibility is investigated through computations of a rigid, moderately-flexible, and highly-flexible rectangular wings undergoing a pure plunging motion. A description of the complex interaction between the unsteady aerodynamics and the flexible wing structural dynamics is given. Connections between the results of this analysis and enhanced loads for the moderately flexible wing are made. Finally, the use of plasma-based actuation to enhance the aerodynamic performance of flapping a wing section at low-Reynolds number is investigated. Solutions are obtained for a combined periodic pitching-and-plunging motion about an 8-degree mean angle-of-attack at chord-based Reynolds numbers of 30,000. It is shown that plasma actuation can reduce drag by 80% over the flapping cycle, and increase the lift-to-drag ratio by a factor of five.

2. Computational Framework

2.1 Aerodynamic Solver

The computational approach employed for the aerodynamics in the present work solves the unsteady, three-dimensional Navier-Stokes equations using a well-validated and robust high-order solver[1–3] (FDL3DI). A number of key features of the FDL3DI code make it most suitable for the present computational challenge. All spatial derivatives are computed through high-order compact or Pade-type[4] difference methods. Schemes ranging from standard 2nd-order to highly-accurate 6th-order methods are possible. To enforce numerical stability, which can be compromised by mesh stretching, boundary conditions and non-linear phenomena, a higher-order, low-pass filter is utilized. This discriminating up to 10th-order low-pass filter preserves the overall high-order accuracy of the spatial discretization, while retaining stability for nonlinear applications. Careful attention to the treatment of the metric derivatives and the Geometric Conservation Law (GCL)[5] ensures higher-order accuracy on deforming and moving grids. The solver is embedded in a high-order overset-grid scheme, which is utilized to provide flexibility for modeling complex geometries. It also serves as a domain decomposition mechanism for application of the high-order approach on massively-parallel, high performance computing platforms.

2.2 Implicit Large-Eddy Simulation (ILES) Methodology

The ILES method to be used in the present computations was first proposed and investigated by Visbal, et al.[6] The underlying idea behind the approach is to capture, with high-accuracy, the resolved part of the turbulent scales, while providing for a smooth regularization procedure to dissipate energy at the represented, but poorly-resolved high wave numbers of the mesh. In the present computational procedure, the 6th-order compact difference scheme provides the high-accuracy, while the low-pass spatial filters provide the regularization of the unresolved scales. All this is accomplished with no additional sub-grid scale models as in traditional LES approaches. An attractive feature of this filtering ILES approach is that the governing equations and numerical procedure remain the same in all regions of the flow. In addition, the ILES method requires approximately half the computational resources of a standard dynamic Smagorinsky sub-grid scale LES model. This results in a scheme capable of capturing, with high-order accuracy, the resolved part of the turbulent scales in an extremely efficient and flexible manner.

2.3 Structural Solvers

For the three-dimensional, highly-flexible wing, the geometrically-nonlinear structural dynamics solution is based on an asymptotic approach to the equations governing the dynamics of a general 3D anisotropic slender solid.[7,8] It is implemented in the University of Michigan's Nonlinear Active Beam Solver (UM/NLABS). Assuming the presence of a small parameter (the inverse of the wing aspect ratio), allows for a multi-scale solution process, in which the problem is decomposed into separate cross-sectional (small-scale) and longitudinal (long-scale) analyses. The longitudinal problem solves for average measures of deformation of the reference line under given external excitations. The cross-sectional problem solves the local deformation for given values of the long-scale variables. Both problems are tightly-coupled and together provide an efficient approximation to the displacement field in the original 3D domain.

2.4 Aerodynamic/Structural Coupling

Coupling of the aerodynamics with the structural response occurs through the external loads and by the resulting deflection of the structure, which is returned to the aerodynamic grid. For the three-dimensional wing communication between the non-coincident aerodynamic and structural surface meshes is accomplished using a local bilinear interpolation procedure[9] for the physical quantities to be passed between the aerodynamic and structural solvers. Implicit coupling of the fluid and structural solvers is achieved by a global sub-iteration strategy which eliminates the temporal lag between the aerodynamic and structural models achieving a complete synchronization of the aerodynamic/structural equation set. Computations of the fluid dynamics and structural dynamics can be carried out in a parallel fashion, with grid deflections and loads being passed between the two solvers at the end of each sub-iteration.

2.5 Empirical Plasma Model

To describe the local forcing of a DBD device, the Navier-Stokes equations are augmented by a source term on the right-hand side of the equation. The source vector contains the terms pertinent to the DBD forcing, and may be derived from models incorporating various degrees of phenomenological and first-principles components.[10,11] The general development follows that described by Shyy, et al.[12] However, to factor uncertainties in the model, and to explore the sensitivity to different force distributions, additional parameters are introduced to permit variations in force orientation and strength.[10] Although this approach is empirical, it provides an attractive framework to explore plasma-based control of complex three-dimensional flows not amenable to a first-principles approach. The parameters describing the simulated body forces are the actuator strength, as well as the normal and streamwise dimensions of the plasma region. In the present work, a novel serpentine arrangement proposed by Roy and Wang[13] is employed. Details of the implementation of plasma model for this serpentine arrangement may be found in Reference 14.

3. Results

3.1 Aspect Ratio Two Plunging Wing

The first results described are for an aspect ratio two flat-plate wing subjected to a sinusoidal heaving motion with mean static angle-of-attack α=8°, Reynolds number Re_c=10^4, reduced frequency k=1.0 and plunging amplitude h_o/c=0.25. A detailed description of these computations, including validation with experimental measurements, can be found in Reference 15.

The computed instantaneous flow structure above the wing at several phases of the plunging motion is displayed in Figure 1 employing iso-surfaces of the Q-criterion.[16] The corresponding phase-averaged flow representation is also shown in Figure 2. At the top of the cycle (Figure 1, Φ=0°), there is a nascent leading-edge separation region on the wing upper-surface due to the presence of the sharp-edge. However, the flow is effectively attached over a significant portion of the wing. Near the trailing-edge, it is interesting to note the persistence of the vortex which was generated in the previous cycle. As the wing plunges downward, the increasing effective angle-of-attack results in the formation of a leading-edge, or dynamic-stall vortex system. During its initial stages (Φ≤45°), this LEV system is fairly uniform in the spanwise direction, except near the wing-tips where the vortex is pinned. By the time the wing reaches its maximum downward velocity (Φ=90°), the flowfield is already highly three-dimensional. There is strong axial flow in the LEV core towards the wing centerline, and the computed maximum instantaneous transverse velocity magnitude actually exceeds its freestream value. By Φ=135°, the leading-edge vortex system evolves into a Λ-shaped structure (Figure 1), as the vortex lifts away from the plate near the centerline while remaining pinned at the wing front corners. By this stage, small-scale transitional features have also emerged. By the end of the down-stroke, the Λ-shaped vortex has detached from the plate corners, as seen perhaps more clearly in the phase-averaged representation of Figure 2. Following its detachment, the vortex evolves into an arch-type structure which propagates downstream as the wing executes its up-stroke. Eventually, through what appears to be a vortex reconnection process, the arch-vortex detaches from the plate surface forming a ring-like vortical structure which is shed, as a new plunging cycle begins.

In summary, the process of dynamic-stall for the low-aspect-ratio plate is primarily characterized by the generation of a leading-edge vortex system, which is pinned at the front corners of the plate and exhibits intense axial flow toward the wing centerline during its initial stages of development. This vortex detaches from the corners and evolves into an arch-type vortex which convects along the plate and is eventually shed as a ring-like structure.

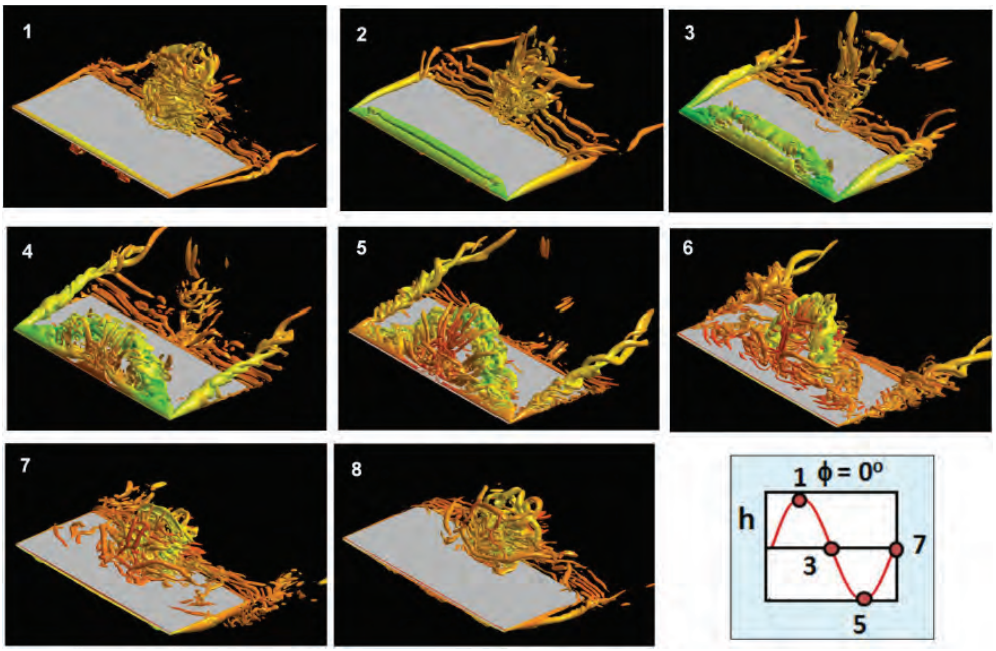


Figure 1. Iso-surface of instantaneous Q-criterion at selected phases of the plunging motion

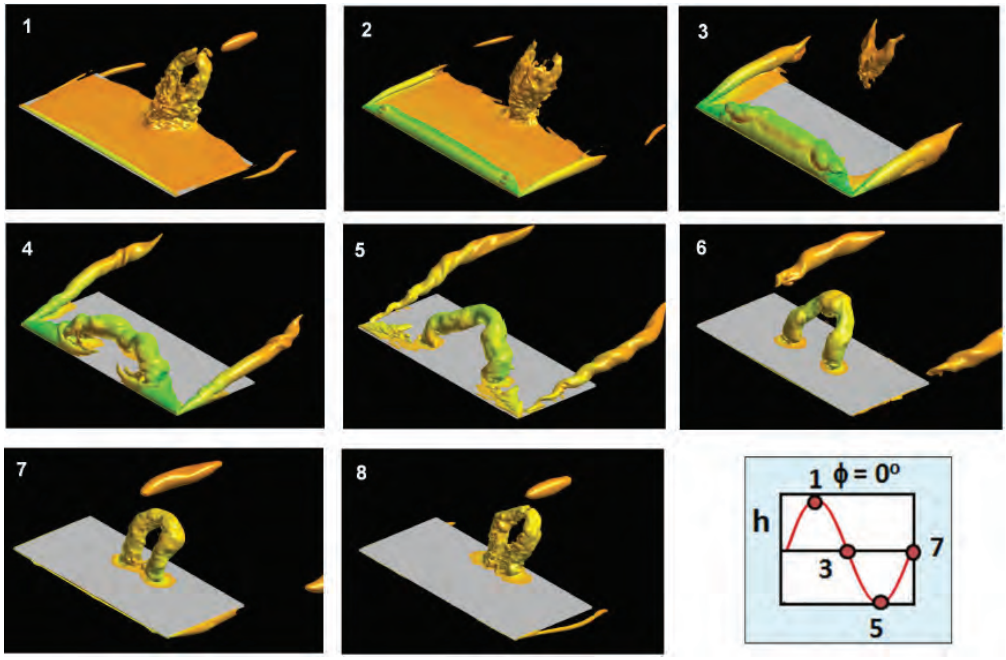


Figure 2. Iso-surface of phase-averaged total pressure at selected phases of the plunging motion

3.2 Effect of Spanwise Flexibility on a Plunging Wing

The second problem to be addressed is the simulation of a three-dimensional rectangular wing with a uniform NACA0012 cross-section oscillating in water in pure heave. Three wings of 0.3-m span and 0.1-m chord with increasing flexibility were investigated. The leading-edge at the wing-root was actuated by a prescribed oscillatory plunging motion $z=z_{root}\sin(2kt)$ where z_{root} is the non-dimensional heave amplitude and $k = \frac{\omega c}{2U_\infty}$ is the reduced frequency. The cases computed in the present work specified the following values for these parameters: k=1.82, period T=1.726, z_{root}=0.175 and Reynolds

number, Re=3.0×10^4. A more extensive discussion of these results and comparisons with experimental measurements may be found in Reference 17.

The flexible wing exhibits a significant deflection at the wing-tip due to the inherent flexibility of the structure, and this deflection lags the imposed plunging motion at the wing-root. This results in a spanwise variation in the deflection of the wing, as well as the development of a spanwise variation in the effective angle-of-attack which results from the velocity of the wing surface, $\alpha_{eff} = tan^{-1}\left(-\frac{dz}{dt}/U_\infty\right)$. Figure 3 demonstrates the impact of these effects on the global flowfield development on the upper-surface of the wing. In this figure iso-surfaces of vorticity magnitude are plotted with the iso-surfaces being colored by the pressure coefficient. Figures 3a-h correspond to equally-spaced points during the plunge cycle. Due to the relatively symmetric response between the down-stroke and the up-stroke of the wing in the computation, an understanding of the flow on the under-surface of the wing can be obtained by matching (a-d) for the top-side with (e-h) for the bottom-side in Figure 3 for the down-stroke and vice-versa for the up-stroke.

At the top of the plunge cycle, Figure 3a, only minor differences between the rigid and flexible wing flowfields are observed. For the rigid wing, distinct remnants of the previously-shed leading-edge vortex may still be seen near the trailing-edge. For the flexible wing only dispersed small-scale structures are seen, except outboard on the wing where a concentrated region of small-scale structures is observed. The presence of this feature will be discussed subsequently. The flexible wing exhibits higher pressure at the leading-edge, which results from the increasingly negative effective angle-of-attack towards the wing-tip.

During the downward plunging motion of the wing, the flowfield on the upper-surface is characterized by the development of two features, the leading-edge vortex and the tip vortex, Figures 3a-e. The rigid wing flowfield exhibits a fairly two-dimensional development, except in the region very near the tip where the leading-edge vortex is pinned to the wing-tip leading-edge. In contrast, the flexible wing flowfield shows notable spanwise variation and distinct differences from the rigid wing. The leading-edge vortex that develops increases in strength outboard on the wing, due to the increase in effective angle-of-attack. This results in lower values of pressure underneath this vortex than what is obtained on the rigid wing. The wing-tip vortices are also stronger for the flexible wing, Figures 3c-e. A corresponding low pressure band develops underneath the tip vortex.

During the down-stroke a laminar leading-edge vortex system emerges on the upper-surface, Figures 3b,c. As the wing moves from the mid-stroke to the bottom of the plunge motion, Figures 3c-e, this laminar vortex system undergoes flow transition with a fully-transitional flow present at the bottom of the down-stroke. A detailed description of this type of transition process for a typical wing section undergoing plunging motion may be found in Visbal.[18] This transition process is enhanced for the flexible wing, where the leading-edge vortex is strengthened due to the larger effective angle-of-attack on the outboard portion of the wing.

As the wing slows and reverses direction, the leading-edge vortex is shed and convects downstream, Figures 3 e-h. The vortex on the rigid wing tends to remain fairly intact and two-dimensional until it approaches the trailing-edge, Figures 3h, a. In contrast, the leading-edge vortex on the flexible wing tends to break apart and be much more three-dimensional in character. Outboard on the flexible wing at y≈2.22 a large region of turbulent flow exists at the terminus of the leading-edge vortex, Figure 3f. This region of turbulent flow persists in the leading-edge region during the full-up-stroke of the wing and is only convected downstream after the wing commences the down-stroke. This interesting behavior develops due to the large motion-induced negative pitch-down that occurs for the flexible wing in this region. During the portion of the wing up-stroke from points (e, t/T=0.5) to (h, t/T=0.83) the flexible wing undergoes a more severe pitch-down motion than the rigid wing from α_{eff}=24.5° to α_{eff}=−45°. This rapid pitch-down restricts the vorticity from propagating downstream. This feature is reminiscent of the flow behavior observed by Visbal[18] for high-frequency low-amplitude plunge oscillations of a wing section.

A distinct high-pressure zone is also noted on the flexible wing during the up-stroke, Figure 3f. This phenomenon arises predominantly from the large deceleration and reversal of direction of the wing. The markedly higher accelerations, as a result of the wing flexibility create greater non-circulatory effects. This increase in the non-circulatory behavior results in the development of these higher pressure regions. Equivalent lower pressures outboard on the under-surface, Figure 3b, are also observed.

The maximum deflection (Table 1), effective angle-of-attack and acceleration at the wing-tip do not vary significantly between the flexible and highly-flexible case. There is a significant lag between the wing-tip motion and the wing-root motion for the highly-flexible case; however. The wing-tip deflection is out of phase with the wing-root deflection for the majority of the plunge cycle. This leads to very large spanwise variations in effective angle-of-attack during the cycle. At the midpoint of the downward plunge, the effective angle-of-attack for the highly-flexible case exhibits a dramatic variation across the span from 32.5° to −38°.

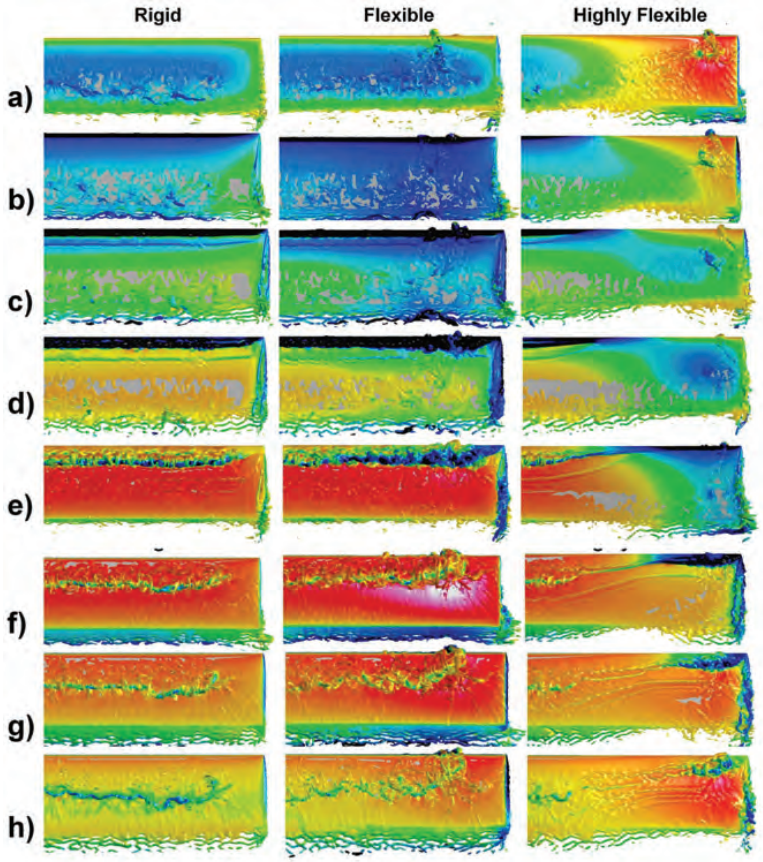


Figure 3. Comparison of flow structure (iso-surfaces of vorticity magnitude colored by pressure coefficient) on the wing upper-surface for the rigid, flexible and highly-flexible cases

Table 1. Maximum tip displacement and mean thrust data

		Z_{tip}/Z_{root}	Phase Angle	$\overline{CT}$
Rigid	Experiment			0.21
	Computation			0.195
Flexible	Experiment	1.64	-25.1	0.32
	Computation	1.59	-27.2	0.278
Highly Flexible	Experiment	1.76	-117.0	0.11
	Computation	1.73	-135.0	0.121

Starting at the top and proceeding through the stroke cycle for the highly-flexible case, Figure 3a-h, the development of the flowfield near the wing-root is similar to the rigid and flexible cases. Further outboard on the wing the flowfield development differs greatly due to the large lag in the tip motion. At the top of the down-stroke, Figure 3a, the wing-tip is located below the wing-root and is moving in an upward direction. The flowfield in the region of the wing-tip at this point in the cycle is similar to the flexible case flowfield between points f and g. Proceeding through the plunge motion with this phase offset the flow near the wing-tip develops similarly to the flow for the flexible wing, albeit with somewhat weaker flow features. This phase offset in flow development between the wing-root and wing-tip leads to portions of the cycle where the leading-edge vortex is developing on the upper-surface near the wing-root, and on the lower-surface near the wing-tip (see for instance Figure 3d). Substantial cross-flow gradients in pressure are also present due to the large phase lag introduced by the greater flexibility.

Another interesting feature observed in the highly-flexible case is the development of a series of small-scale vortical structures in the boundary-layer downstream of the leading-edge vortex system during the time period when the wing locally undergoes a downward motion, Figures 3d-h. These vortices form initially near the wing-root, and the development moves outboard on the wing as the plunge cycle progresses. In the case of the rigid and flexible wing, an initial small-scale vortex develops across the span, Figure 3d but this vortex rapidly breaks down with the onset of transition. In the highly-flexible case, a relief effect due to the large spanwise gradients tends to delay the onset of transition allowing this series of vortices to develop over a portion of the cycle. Eventually these vortices also succumb to transition.

The flowfield over the mid-portion of the wing develops quite differently from the rigid and flexible case. The development of the leading-edge vortex in this region is much weaker, smaller in size and further downstream. This results from the fact that the maximum effective angle-of-attack at the mid-span, $\alpha_{eff}\approx 20°$, for the highly-flexible case is less than that for the rigid, $\alpha_{eff}=32.5°$, and flexible $\alpha_{eff}\approx 38°$ response.

The experimental measurements of Heathcote[19] demonstrated that the flexible case produces an enhanced thrust coefficient, while the highly-flexible wing produces a very small amount of thrust. This effect is reproduced in these computations as seen in Figure 4a. The mean thrust generated increases from $\overline{CT} = 0.195$ for the rigid wing to $\overline{CT} = 0.278$ for the flexible wing, an increase of 42%. This is consistent with the experimentally-reported increase in thrust of 50%. The peak thrust generation and the corresponding maximum difference between the thrust for the rigid and flexible wing occurs over the mid-portion of the down-stroke and up-stroke. This corresponds to the portion of the motion where the formation of the leading-edge vortex occurs. As discussed previously, the moderate flexibility of the wing leads to the formation of a stronger leading-edge vortex which, in turn, leads to higher suction levels around the leading-edge of the airfoil; and therefore higher thrust.

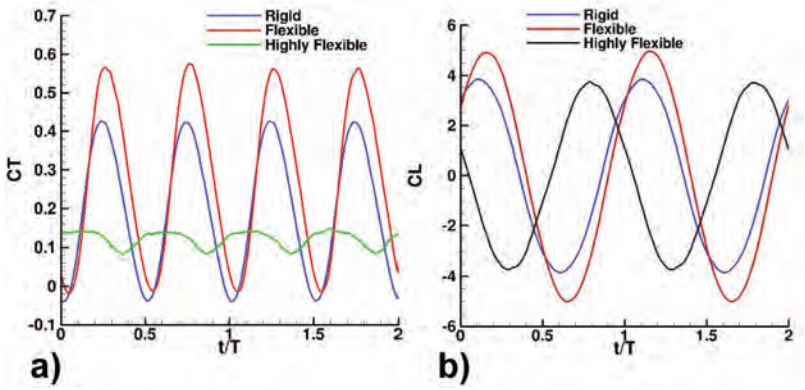


Figure 4. Effect of flexibility on generated a) thrust coefficient and b) lift coefficient

A striking drop in the thrust produced is noted for the highly-flexible wing due to the large lag in the wing-tip deflection, which considerably alters the flowfield development. An overall weaker leading-edge vortex development occurs, in particular over the middle portion of the wing where the maximum effective angle-of-attack is greatly reduced. This disruption of the flowfield and alteration of the vortical flow development leads to a loss in thrust production. These results suggest that an optimal amount of flexibility exists that would produce the most enhancement to thrust. This observation is consistent with the previous work of Heathcote, et al.[19] and Chimakurthi[20].

Enhanced lift is also obtained for the flexible wing. Figure 4b displays the time histories of the lift coefficient for the rigid, flexible and highly-flexible wings. The peaks in the lift-time histories correlate with the locations of maximum peak accelerations. The higher peak lift in the flexible case is consistent with the significantly larger peak accelerations. These greater accelerations give rise to stronger non-circulatory loads that arise from the previously-described high- and low-pressure zones. The additional flexibility introduced in the highly-flexible case results in maximum lift coefficients similar to the rigid case. This result again suggests that there is an optimum flexibility for lift production.

3.3 Plasma-Based Control for a Flapping Airfoil

In this third section, large-eddy simulations at *Re*=30,000 for the flapping motion of an SD7003 wing are described. Computations were performed to simulate the flapping movement considered in the experimental investigations of Baik, et al.[21] and Rival, et al.[22] This forced-motion consisted of combined pitching and plunging according to the following description in non-dimensional variables:

$$\alpha = \alpha_0 + \alpha_1 \sin(\omega_p t) \quad (1)$$

$$h = h_1 \cos(\omega_p t) \quad (2)$$

where α_0=8.0 deg, α_1=−8.42 deg, h_1=0.5, and ω_p=0.5. These experiments were constructed in order to determine whether thrust could be produced by the prescribed flapping motion. Pulsed plasma control with a duty cycle of 50% was applied to enhance the aerodynamic performance of the flapping-wing. Two values of the plasma scale parameter, D_c=50 and D_c=100, were specified for the computations to assess influence of the plasma force magnitude on control effectiveness. An extensive discussion of these computations can be found in Reference 14.

Planar contours of the phase-averaged streamwise velocity component for the control cases are compared to the baseline case (no control) in Figure 5. At all phase angles, plasma control is seen to reduce the vertical extent of the boundary-layer and to mitigate separation. This is most evident at ϕ=180, where the flapping motion produces the greatest disruption of the flow on the upper-airfoil surface. As expected, the higher value of D_c is somewhat more effective in controlling the flow. Corresponding contours of the phase-averaged spanwise component of vorticity are shown in Figure 6. The actuation pulsing cycle gives rise to vortical structures which can be observed at the edge of the boundary, most clearly at ϕ=270. When the angle-of-attack is sufficiently large during the flapping motion, a large leading-edge vortex is generated and dynamic-stall occurs. The leading-edge vortex results in formation of a counter-rotating vortex in the trailing-edge region, which is eventually shed and convects downstream into the wake. The trailing-edge vortex is quite visible for the baseline case at ϕ=180 in Figure 6. In the control situation for D_c=50, the strength and vertical extent of the leading-edge vortex is greatly diminished, and then so too is that of the trailing-edge vortex. When D_c=100, the leading-edge flow is controlled to such a degree that no trailing-edge vortex forms.

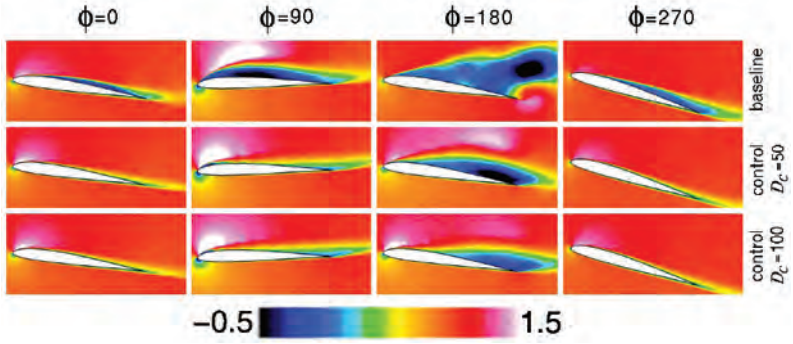


Figure 5. Phase-averaged streamwise velocity component for baseline and control cases with *Re*=30, 000

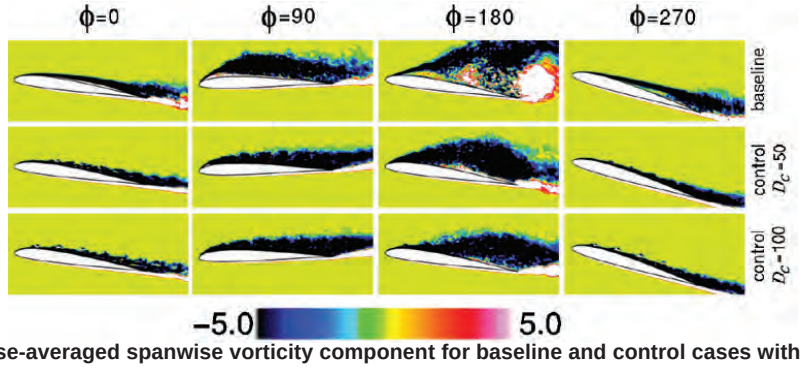


Figure 6. Phase-averaged spanwise vorticity component for baseline and control cases with *Re*=30, 000

An efficient way to compare the aerodynamic force coefficients is by constructing the phase-averaged coefficients for one cycle of the forcing, which was done in Figure 7. In frame a) of the figure, the forces are given as a function of the phase angle ϕ. It is observed that the drag is reduced considerably with the use of flow control. In addition, the time-mean lift has increased. Flow control has also had a pronounced effect on the moment coefficient, which indicates it may also be utilized to modify the stability and control of the airfoil. Representation of the forces as a function of the effective angle-of-attack α_e is provided by the phase diagram in frame b) of Figure 7. For the baseline case, maximum lift occurs near ϕ=130 following the formation of leading-edge vortex. The lift then decreases and the drag raises as the flow along the upper surface are completely separated.

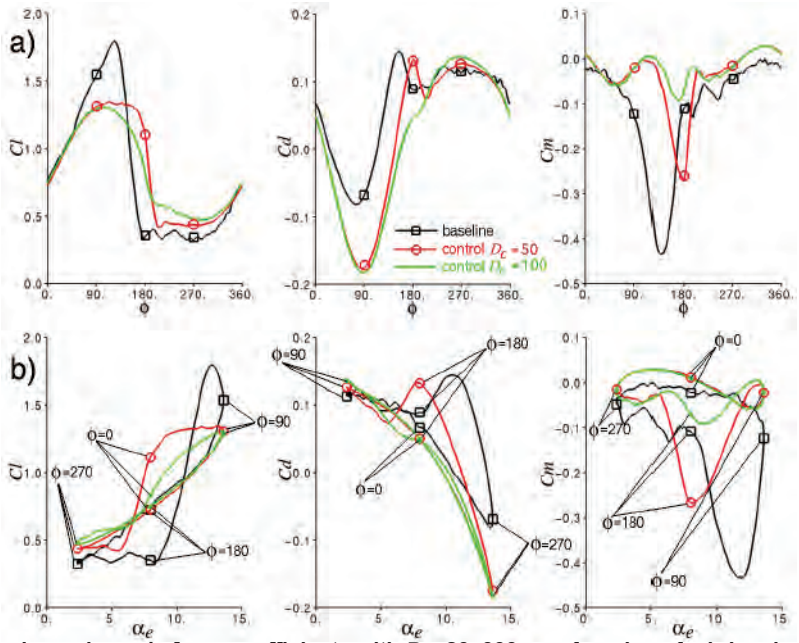


Figure 7. Phase-averaged aerodynamic force coefficients with *Re*=30, 000 as a function of: a) the phase angle ϕ and b) the effective angle-of-attack α_e

Time-mean values of the aerodynamic force coefficients for all cases are found in Table 2. For the baseline case, no thrust $\left(\overline{Cd} < 0\right)$ was generated. With the use of control, although thrust was not achieved, the time-mean drag was lowered by 64% for D_C=50 and by 80% for D_C=100. In addition, the time-mean lift-to-drag ratio $\left(\overline{Cl} / \overline{Cd}\right)$ increased by a factor of 2.9 for D_C=50, and 5.1 for D_C=100.

Table 2. Time-mean aerodynamic force coefficients

case	*Re*	$\overline{Cl}$	$\overline{Cd}$	$\overline{Cm}$	$\overline{Cl}$ / $\overline{Cd}$
baseline	30,000	0.8133	0.0612	-0.1092	13.29
control D_c = 50	30,000	0.8500	0.0219	-0.0421	38.81
control D_c = 100	30,000	0.8454	0.0125	-0.0228	67.63
baseline	60,000	0.8773	0.0462	-0.0993	18.99
control D_c = 50	60,000	0.8990	0.0194	-0.0534	46.34

4. Conclusion

An Implicit Large-Eddy Simulation (ILES) approach was applied to the prediction of transitional flow for canonical problems associated with flexible, flapping-wing MAVs. ILES is found to be particularly attractive for these types of low-Reynolds number flows that exhibit mixed laminar, transitional and turbulent regions, for which conventional high-Reynolds number analysis tools may not be adequate. The computational framework, FDL3DI (and its extension for aeroelastic problems FDL3DIAE), employed contains a robust, high-order Navier-Stokes solver which adapts overset grid technologies to treat the most complex geometries. This framework is highly-scalable and easily-portable to a variety of parallel computing platforms.

Computations for three separate canonical problems have been presented in this work. The first simulations were for a rectangular wing of aspect ratio two undergoing a pure heaving motion. The present high-fidelity simulations provide the most detailed representation to date of the unsteady 3D flow structure generated during the dynamic-stall of a low-aspect-ratio wing. This process is characterized by the generation of a leading-edge vortex system which is pinned at the front corners of the plate, and which exhibits intense transverse flow toward the wing centerline during its initial stages of development. This vortex detaches from the corners and evolves into an arch-type structure. The legs of the arch vortex move toward the wing centerline and reconnect, forming a ring-like structure which is shed as the next plunging cycle begins.

The second problem computed was a flexible, rectangular wing undergoing a pure plunging motion. The longitudinal flexibility of the wing gives rise to significant spanwise variations in the wing deflection, motion-induced effective angle-of-attack and wing acceleration. These effects produce a complex interaction between the vortex dynamics and the structural motion. The spanwise variation and increased strength of the leading-edge vortex system in the moderately-flexible case produces enhanced thrust. Increased non-circulatory loads, due to the larger values of acceleration achieved by the moderately flexible wing, result in higher peak lift. The positive effects for moderate flexibility are lost as flexibility is increased in the highly-flexible case. These results suggest that an optimum amount of flexibility exists for the case of a plunging wing, and is associated with wing motions where the wing-tip and wing-root motions are in phase over much of the plunge cycle.

The final problem investigated was the use of plasma-based control for enhanced aerodynamic performance for an airfoil undergoing a prescribed flapping motion. All cases showed improvement in flow quality and aerodynamic performance when control was applied. For values of the plasma scale parameter D_C=0, the time-mean drag could be reduced by over 60% and the lift-to-drag ratio increased by almost a factor-of- three. When a greater plasma force was exerted (D_C=100), the drag could be lowered by 80% and $\overline{Cl}/\overline{Cd}$ raised by a factor-of-five.

5. Significance to the DoD

This project harnesses the attributes of recently-developed high-fidelity, multidisciplinary codes to provide accurate, detailed simulations of flexible flapping-wing fliers. The high-order overset method employed in the present project provides a unique, scalable computational capability that has been specifically-created to address the simulation issues associated with highly-nonlinear, unsteady, transitional flows inherent to MAVs. Likewise, computational structural modules that treat large structural deflections and nontraditional structural materials have been developed and incorporated into the context of this computational framework. The computational techniques demonstrated in this project will provide MAV designers with unique tools and understanding to accelerate the development of unconventional, bio-mimetic MAV concepts.

Acknowledgments

This work was sponsored by AFOSR under a task monitored by Dr. Doug Smith and supported in part by a grant of HPC time from the DoD HPC Shared Resource Centers at the US Air Force Research Laboratory, Wright-Patterson AFB, OH and the US Army Corps of Engineers Research and Development Center at Vicksburg, MS.

References

1. Gaitonde, D. and Visbal, M., "High-Order Schemes for Navier-Stokes Equations: Algorithm and Implementation into FDL3DI", *Tech. Rep. AFRL-VA-WP-TR-1998-3060*, Air Force Research Laboratory, Wright-Patterson AFB, 1998.

2. Gaitonde, D. and Visbal, M., "Further Development of a Navier-Stokes Solution Procedure Based on Higher-Order Formulas", *AIAA Paper 99-0557*, January 1999.

3. Visbal, M. and Gaitonde, D., "High-Order Accurate Methods for Complex Unsteady Subsonic Flows", *AIAA Journal*, Vol. 37, No. 10, pp. 1231–1239, 1999.

4. Visbal, M.R. and Gaitonde, D.V., "On the Use of Higher-Order Finite-Difference Schemes on Curvilinear and Deforming Meshes", *Journal of Computational Physics*, Vol. 181, pp. 155–185, 2002.

5. Visbal, M. and Gordnier, R., "A High-Order Flow Solver for Deforming and Moving Meshes", *AIAA Paper 2000-2619*, June 2000.

6. Visbal, M.R., Morgan, P.E., and Rizzetta, D.P., "An Implicit LES Approach Based on High-Order Compact Differencing and Filtering Schemes (Invited)", *AIAA-2003-4098*, June 2003.

7. Palacios, R. and Cesnik, C.E.S., "A Geometrically-Nonlinear Theory of Active Composite Beams with Deformable Cross-Sections", *AIAA Journal*, Vol. 46, No. 2, pp. 439–450, 2008.

8. Palacios, R. and Cesnik, C.E.S., "Cross-Sectonal Analysis of Non-homogenous Anistropic Active Slender Structures", *AIAA Journal*, Vol. 43, No. 12, pp. 2624–2638, 2005.

9. Smith, M.J., Hodges, D., and Cesnik, C.E.S., "Evaluation of Computational Algorithms Suitable for Fluid-Structure Interactions", *Journal of Aircraft*, Vol. 37, No. 2, pp. 282–294, 2000.

10. Gaitonde, D., Visbal, M.R., and Roy, S., "Control of flow past a wing section with plasma-based body forces", *AIAA-2005-5302*, June 2005.

11. Gaitonde, D., Visbal, M.R., and Roy, S., "A coupled approach for plasma-based flow control simulations of wing sections", *AIAA-2006-1205*, January 2006.

12. Shyy, W., Jayaraman, B., and Anderson, A., "Modeling of glow-discharge-induced fluid dynamics", *Journal of Applied Physics*, Vol. 92, No. 6434, 2002.

13. Roy, S. and Wang, C.-C., "Bulk Flow Modifications with Horseshoe and Serpentine Plasma Actuators", *Journal of Physics D:Applied Physics*, Vol. 42, No. 3, pp. 1–5, February 2009.

14. Rizzetta, D.P. and Visbal, M.R., "Effect of Plasma-Based Control on Low-Reynolds Number Flapping Airfoil Performance", *AIAA-2011-735*, January 2011.

15. Visbal, M.R., "Three-Dimensional Flow Structure on a Heaving Low-Aspect-Ratio Wing", *AIAA-2011-219*, January 2011.

16. Jeong, J. and Hussain, F., "On the Identification of a Vortex", *Journal of Fluid Mechanics*, Vol. 285, pp. 69–94, February 1995.

17. Gordnier, R.E., Chimakurthi, S.K., Cesnik, C.E.S., and Attar, P.J., "High-Fidelity Aeroelastic Computations of a Flapping-Wing with Spanwise Flexibility", *AIAA-2011-570*, January 2011.

18. Visbal, M.R., "High-Fidelity Simulation of Transitional Flows Past a Plunging Airfoil", *AIAA Journal*, Vol. 47, No. 11, pp. 2685–2697, November 2009.

19. Heathcote, S., Wang, Z., and Gursul, I., "Effect of Spanwise Flexibility on Flapping-Wing Propulsion", *Journal of Fluids and Structures*, Vol. 24, No. 2, pp. 183–199, 2008.

20. Chimakurthi, S.K., *A Computational Aeroelastic Framework for Analyzing Flapping-Wings*, Doctoral Dissertation of the University of Michigan, 2009.

21. Baik, Y., Rausch, J.M., Bernal, L.P., and Ol, M.V., "Experimental Investigation of Pitching and Plunging Airfoils at Reynolds Number between 1.0×10^4 and 6×10^4", *AIAA-2009-4030*, June 2009.

22. "Unsteady Aerodynamics for Micro Air Vehicles", *RTO-TR-AVT-149*, North Atlantic Treaty Organization, Rhode St. Genese, Belgium, 2010.

Linux C-Shell Regression Testing for the SHAMRC CFD Code

Henry J. Happ
Applied Research Associates, Albuquerque, NM
hhapp@ara.com

Abstract

Applied Research Associates, Southwest Division uses, develops, and maintains SHAMRC for DTRA. SHAMRC is a 2D and 3D (serial and parallel) finite-difference computational fluid dynamics (CFD) code that models airblast by solving the Euler equations. SHAMRC is extremely efficient, and has been used to run multi-billion zone calculations on thousands of HPC processors to answer questions of high interest to the Warfighter.

Unlike other codes that release updated versions at most a few times per year, SHAMRC is under continuous modification, with changes made almost daily. To make sure changes for one capability do not affect adversely other capabilities (and thus frustrate SHAMRC users of high performance computing (HPC) resources), we have developed a set of Linux C-Shell regression testing scripts that are run daily. These scripts run approximately thirty different calculations that test all the important SHAMRC capabilities. The scripts trap extremely small differences in code results to allow the maintainer to evaluate quickly the code modifications that generated those differences. Adding another calculation is extremely easy. Detailed results are e-mailed automatically to the maintainer after the scripts run. These scripts can be made available to anyone who can use them.

1. Introduction

Applied Research Associates, Southwest Division uses, develops, and maintains SHAMRC (Second-order Hydrodynamic Automatic Mesh Refinement Code)[1] for the Defense Threat Reduction Agency (DTRA). SHAMRC is a two- and three-dimensional (2D/3D), inviscid, finite-difference, computational fluid dynamics (CFD) computer code. It has been supported and developed by DTRA over the past 35 years, and is used to solve a variety of airblast-related problems. These problems include high-explosive detonations, nuclear detonations, structure loading, thermal effects on airblast, cloud rise, conventional munitions blast and fragmentation, shock tube phenomenology, dust and debris dispersion, and atmospheric shock propagation. SHAMRC employs a number of physics-based and empirically-based models, including dust sweep-up, thermal heating, K-e turbulence, two-phase flow with massive-interactive particulates, non-equilibrium radiation diffusion, chemical kinetics, and thermobaric explosive models. SHAMRC runs in parallel, and is extremely scalable. Multiple billion-zone calculations have been run on thousands of Department of Defense (DoD) high performance computing (HPC) processors. SHAMRC is a government-owned, non-proprietary code under high-technology export control.

SHAMRC is a research code under continuous development, with changes occurring daily. It exists as a *metacode*. For each run, new source code is generated and compiled. The metacode status allows various code options (e.g., [2D, 3D], [Cartesian geometry, Cylindrical geometry], [Serial, Parallel]) to coexist within the same metacode, but the generated source code only contains the code with the desired options. This distinguishes SHAMRC from other well-known CFD codes where all options exist within a single executable. The customized aspect of each SHAMRC source code creates an extremely fast executable—for some similar calculations, SHAMRC can be more than an order-of-magnitude faster than similar codes.

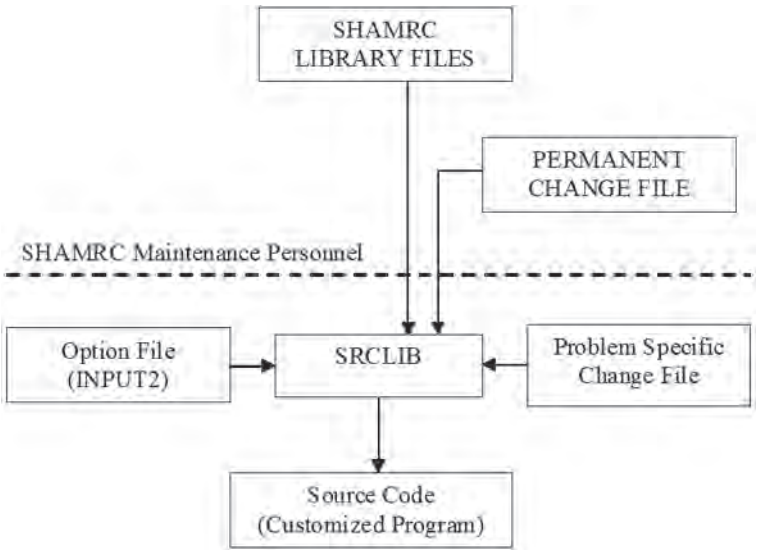
As is typical for CFD codes, a SHAMRC calculation is run in multiple steps. Table 1 summarizes the steps showing the executables that are run, and what they do. For the first step the user runs a setup program (called KEEL) that sets up the initial restart dump file with all the user-specified conditions, and the file that will contain sensor time histories. The propagation program (called SHARC) advances the calculation in time, generating restart dumps at user-specified intervals. During the course of the calculation, contour plots and histograms may be obtained via a plotting program (called PULL). Time history waveforms at specified sensor locations are created by program STAPLT.

Table 1. Summary of SHAMRC suite programs in a typical calculation

Executable Name	Function
KEEL	Set up the calculation, generate the initial restart file and initial time history file
SHARC	Advance the calculation in time, generating restart files at specified intervals and updating the time history file
PULL	While SHARC is running, and after it is finished, obtain contour and histogram plots of desired variables
STAPLT	While SHARC is running and after it is finished, get time histories of selected variables at selected locations.

SHAMRC is maintained by two primary developers/maintainers, Henry Happ and Joseph Crepeau. Others may develop code for SHAMRC, but all code that eventually becomes part of the SHAMRC suite is examined and approved by the maintainers. SHAMRC uses the SRCLIB[2] management system for version control. This system simplifies the management of large codes. For each version of SHAMRC, there is a suite of ASCII *metafiles* in SRCLIB format. (This is similar to codes written in C and C++ that have options surrounding sections of code.) These files are referred to as the SHAMRC library files and have read-only access permissions. Source code that is to be built for execution is processed through SRCLIB. Updates (new capabilities, bug-fixes) to the read-only metafiles are provided through a read-only Master Change File (called SHAMRC.CH) that is tested and maintained by Happ and Crepeau. This change file is applied to all code builds. The date of the last change to the Master Change File is part of the text of the file, and this date is written to the log file, so users always know which version of the Master Change File was used to create the source code.

Options for the SRCLIB program are generated by a program (PLANK) that reads the various program input files (for KEEL, SHARC, PULL, etc.) and generates an option file (INPUT2). All code builds also have a calculation-specific change file for each run. This provides an easy way to apply customized changes (additional prints, modified coefficients, etc.) for each run. The figure below illustrates this process.



The Master Change File is disseminated to SHAMRC users and to DoD HPC sites on a regular basis, and is made available to all users through an ARA SharePoint website. Intermediate updates are made whenever a code error is discovered and fixed. On approximately a yearly basis, local user changes that will become a permanent part of the code suite are consolidated into the Master Change File and a new version of the code is released. Once a new SHAMRC version is released, the previous version is still maintained – if bugs are found that apply to the previous version, the Master Change File associated with that version is updated. However, development is only done for the latest version. All SHAMRC versions are backward compatible—new versions can read input files and restart files from older versions.

As the Master Change File is updated, it is tested locally at ARA/SWD and is not released outside of ARA/SWD (to the DoD HPC sites or other users) until it has been thoroughly tested. Locally at SWD/ARA, an alpha version of the Master Change File is used for new capabilities that are in the initial development phase to make sure that local users are not affected adversely during their code build process.

This method of development allows for very fast response to problems—it is common for a bug to be reported and a fix developed and disseminated to all SHAMRC users *via* an updated Master Change File by the next day. SHAMRC users have the privilege of having "the latest version" almost continuously.

With code changes being made continuously, we needed a way to make sure that code changes have not introduced side effects, i.e., that other code capabilities have not been affected inadvertently. Sometimes a bug fix addresses the observed problem, but not other cases. We developed a regression-testing scheme that is easy to maintain and update, yet does a good job of detecting side-effects. We opted to develop our own scripts rather than using other open-source software because we wanted something we could modify easily and that would fit inside the existing SHAMRC framework.

2. Test Calculations

It is necessary to construct a suite of calculations that would form the database against which code changes would be tested. A clear set of calculations that must be included is the set of calculations in the Sample Calculations document[3]. This document is available to all SHAMRC users, and ARA must ensure that these calculations work as advertised. Then we developed additional calculations that would test the important, most-used capabilities of SHAMRC. To this end, we constructed a Code Coverage Matrix spreadsheet to assist in developing these calculations. A section of the spreadsheet is shown below. Down the left column are shorthand descriptions of SHAMRC capabilities, and across the top are the numerical labels of the calculations that include those capabilities. To date, over 120 separate capabilities are tested in 31 calculations. Each calculation is a standard SHAMRC calculation, with standard characteristics. As new capabilities are developed, a regression calculation is added to the regression suite to make sure that capability continues to work as development proceeds. If a user encounters an error that was not identified by the regression suite, a new regression calculation is added to the suite (or an existing calculation is modified) to cover the gap in the coverage matrix.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	***SHAMRC (6) Coverage Matrix***																
2	**SHARC Capability (Runs)**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
142	PLTFMT=1 (3D, Parallel only)								X	X					X		
143	PLTFMT=2					X											
144	PLTFMT=3																
145	POLYPR (2D)																
146	POLYPR (3D)																
147	PSDSTR=1																
148	PUTPART																
149	RAD=2																
150	RBOUND=0 (2D)													X			
151	RBOUND=0 (3D)		X														
152	RBOUND=1 (2D)			X	X						X						
153	RBOUND=1 (3D)							X	X	X					X		

For example, calculation 4 is a 2D serial calculation of a 45-degree shock on a ramp, exercising our "shore" (partial-fluid, partial-solid) capability with a feed-in square-wave left boundary condition and right and top transmissive boundary conditions. Calculation 8 is a 3D parallel calculation of non-ideal AF explosive AFX-757 surrounded by a steel case, exercising the non-ideal burn, massive-particle, and pseudo-material strength capabilities.

2.1 Relevant Calculation File Data

The setup program KEEL produces a file called *outkeel* that contains details about the setup, including mesh information, total mass and energy in the grid, option values, etc. Values are usually specified to at least 6 significant digits.

The propagation program SHARC generates a file called *outsharc* that reflects the state of the calculation grid at dump times, in particular, the total mass and energy (both internal and kinetic) in the grid. These values are reported to 15 significant digits. At each cycle, the value of the time-step is reported to 6 significant digits.

The program that post-processes the dump files (PULL) generates a file called *outpull* that gives statistics values (minima and maxima) for the contour plots and histograms generated. These values are reported to 5 significant figures.

The program that post-processes the time-history files (STAPLT) generates relevant data to 5 significant figures.

3. Scripts

The next step was to develop scripts to implement the testing of these changes. We chose the UNIX C Shell scripting language primarily on the basis of familiarity. The scripts have the following characteristics:

- Easy to maintain
- Easy to add new calculations to the database
- Ability to start from multiple spots within the scripts
- Well-documented internally
- Catch very slight differences while ignoring known differences

We set up a UNIX directory structure to facilitate the easy addition of a new calculation. There is a base directory named Regression with two subdirectories, *Baseline* and *Work*. The *Work* directory contains the scripts. Under the *Baseline* directory is a set of calculations, each calculation in its own directory and labeled with a number [1–31 currently]. To add a new calculation, one simply adds a new directory (named with the next consecutive integer) with the new calculation files to the *Baseline* directory—that is all.

The *Baseline* calculation directories contain the SHAMRC input files necessary to run all of the steps in a calculation. The script *baserun* runs all the baseline calculations (or a designated subset). This is done infrequently, only as necessary and when regression analysis has shown that the latest version of SHAMRC is producing correct results, and thus the new output files (usually with additional information or with slightly changed results from a tweaked algorithm) should now be the ones used for regression comparison.

3.1 Procedure

Every night each calculation in the *Baseline* directory is run automatically from problem setup to post-processing with the *regrun* script. This script copies the input files from each calculation in the *Baseline* directory to the *Work* directory, and runs through each step in the calculation process. The latest Master Change File is used to construct the source code at each step. The same scripts invoked by SHAMRC users to run each code step are invoked in the *regrun* script. The differences in the generated text output files between the baseline run and the latest run are noted using the UNIX *grep* utility. Known differences (detailed timing information) are ignored. All other differences (even a single character!) are placed into a file. If one or more of these differences are found, the run process for that calculation does not continue.

After all regression calculations in the *Baseline* directory are run, an e-mail is sent automatically (using the Linux *cron* utility) to one of the maintainers indicating the results of the runs. Each day the results shown in the e-mail message are examined. All differences are evaluated. If the new results are appropriate and expected, the new results replace the old results in the *Baseline* directory. This is usually done by running the *baserun* script. If differences need further evaluation, a differencing utility is invoked to highlight the differences between the baseline source code and the new source, and appropriate action is taken.

Below is a snippet from the e-mail message that is sent daily to the maintainer. It shows that calculation 5 ran successfully at all steps (KEEL, SHARC, PULL). Calculation 6 finished the SHARC calculation, but differences were found between the baseline results and these results. The script halts the calculation at this step and proceeds to the next calculation (7).

```
*** 5
===Constructing source code for KEEL===
===Done Making KEEL===
===Compiling and Running KEEL===
===Done Running KEEL===
=== Constructing source code for SHARC===
===Done Making SHARC===
===Compiling and Running SHARC===
===Done Running SHARC===
===Differencing SHARC output files===
=== Constructing source code for PULL===
===Done Making PULL===
===Compiling and Running PULL===
===Done Running PULL===
Time for directory 5 is 57 seconds.
```

```
*** 6
=== Constructing source code for KEEL===
===Done Making KEEL===
===Compiling and Running KEEL===
===Done Running KEEL===
=== Constructing source code for SHARC===
===Done Making SHARC===
===Compiling and Running SHARC===
===Done Running SHARC===
===Differencing SHARC output files===
Difference found in outsharc files.  See outsharc.diff

*** 7
=== Constructing source code for KEEL===
===Done Making KEEL===
. . .
```

3.2 Details

By default, the scripts will run all the calculations in the appropriate directory, but the user may specify a particular calculation. If a particular calculation is specified, the user may also specify a particular phase point of the calculation (KEEL, SHARC, PULL, STAPLT) to begin the script. This can save time when it is known that nothing has changed in, say, the setup part of the calculation.

Below is an outline of the steps followed in the *baserun* script:

1. Get the arguments from the command line, check them for validity.
2. Make sure PATH environment variables and regression directories have been set up.
3. Loop over all run directories in the *Baseline* directory. [Note: All run directories begin with a digit, as was seen in the coverage matrix above. So the C-Shell command to loop is "foreach d (([1-9]*)". Thus when a new directory to be tested is added, no update to the script is necessary]
   a. If not starting at *keel*, go to a specified label (*sharc*, *pull*, *staplt*).
   b. Run the same script the user runs in the process of creating the source, compiling the source, and running the calculation. Use the latest version of SHAMRC.CH to create the source.
      If an error is detected at any point (creating, compiling, running), note the error in a log file and go on to the next run directory.
   c. If the calculation runs to completion, go on to the next calculation in the series (*keel* → *sharc* → *pull* → *staplt*).

The steps for the *regrun* script are as follows:

4. Get the arguments from the command line, check them for validity.
5. Make sure PATH environment variables and regression directories have been set up.
6. Loop over all run directories in the *Baseline* directory.
   a. Copy all the input files from the run directory under the *Baseline* directory into an identically named run directory under the *Work* directory.
   b. Run KEEL.
      i. Run the same script the user runs in the process of creating the source, compiling the source, and running the calculation. Use the latest version of SHAMRC.CH to create the source.
      ii. Compare the appropriate output files from the KEEL run with those from the baseline run using the UNIX *diff* utility. This notes all differences between the files. Capture the output into file *outkeel*.diff.
      iii. Applying the UNIX *grep* utility against the *outkeel*.diff file, filter out known/expected differences, e.g, dates, times (local and CPU). If <u>any</u> differences are found, write a message and go on to the next calculation.
   c. If the KEEL run is successful, run SHARC, and follow the same steps as when KEEL is run.
   d. If the SHARC run is successful, run PULL and follow the same steps.
   e. If the PULL run is successful, run STAPLT.

The output files are set up so that even very small differences are detectable.

4. Conclusion

Since implementing the Regression procedure in 2006, it has proved invaluable in catching development errors quickly, preventing SHAMRC users from wasting valuable HPC resources running with flawed code that must then be fixed and rerun. Before the Regression procedure, code flaws would often stay hidden for a while, cropping up when the modification causing the problem was forgotten. Most of the errors turn out to be of the "side-effect" type, i.e., developed changes meant to correct or update one capability generate errors in other capabilities. Using this procedure has improved significantly turnaround time for DoD customers.

Acknowledgments

This work was funded by Applied Research Associates with Independent Research and Development funds. I thank my colleague Joseph Crepeau for his helpful suggestions.

References

1. Crepeau, Joseph, Hikida, Shuichi, and Needham, Charles E., *Second-Order Hydrodynamic Automatic Mesh Refinement Code: Volume I, Methodology*, January 2011.

2. Wittwer, Leon A., *SRCLIB, A Source Code Library Manager*, May 1995.

3. Crepeau, Joseph, et al., *SHAMRC Volume 3: Sample Calculations (Version 6)*, January 2011.

Mitigation of Optical Distortions in a Free Shear Layer using Feedback Flow Control

Jurgen Seidel, Casey Fagley, and Robert Decker
US Air Force Academy, USAF Academy, CO
{Jurgen.Seidel.ctr.de,Casey.Fagley.ctr,Robert.Decker.ctr}@usafa.edu

Abstract

The development of a flow-state database for the investigation of optical aberrations caused by density variations in a separated shear-layer behind a two-dimensional turret is discussed in this paper. Spatially and temporally well-resolved Delayed Detached-Eddy Simulations (DDES) using Cobalt were performed to investigate the flowfield in detail, both in its natural, unforced state and when periodic forcing is applied. Two different geometries of the forcing slot have been investigated and the results indicate that the presence of the slot, even when no actuation is applied, has an effect on the large, coherent structures in the shear layer due to a resonance phenomenon. The unforced and forced computational results are used to compile a flow-state database that will be the basis for the development of a Reduced-Order Model (ROM) and feedback control strategies to mitigate the optical aberrations caused by large coherent structures in the flow.

1. Introduction

An initially collimated beam passing through a medium is distorted by changes in the refractive index, n, in the medium. For air, the refractive index is primarily a function of the fluid density, described by the Gladstone-Dale relation,

$$n\,(x, y, z, t) = 1 + k_{GD}\rho\,(x, y, z, t), \tag{1}$$

where the proportionality constant is the Gladstone-Dale constant k_{GD}=2.289 · 10^{-4}m^3/kg and ρ is the fluid density.[1–3] A measure of the aberration of a beam is the optical path length,

$$OPL\,(x, y, z, t) = \int n\,(x, y, z, t)dl, \tag{2}$$

i.e., the path integral of the index of refraction along the beam. The wave front distortions ultimately result in reduced focus and beam intensity in the far-field. This is particularly detrimental for airborne optical systems, where density variations occur close to the beam aperture due to the motion of the aircraft. The distortions away from the ideal diffraction limit can greatly reduce the usefulness of an airborne optical system.

To highlight the wave-front distortions, the optical path difference,

$$OPD(\xi,\zeta,t;\eta) = OPL(\xi,\zeta,t;\eta) - (\overline{OPL(\xi,\zeta,t;\eta)}^{\xi,\zeta}, \tag{3}$$

is typically used, written in beam coordinates where the beam propagates in the η-direction and the spatial average is performed over the beam aperture in the ξ-ζ-plane. In addition, it is customary to remove the mean-wave front tip and tilt, which are the inclination of the mean-wave front with respect to the beam propagation direction.

Aero-optical aberrations can be categorized in two main groups. The first category, associated with fluctuations in the flowfield of the scale of the aperture, include bore-sight and tracking errors. These aberrations are amenable to correction by current adaptive optical systems, due to their relatively large length scales and concomitant slow time scales. The second category includes errors such as beam spreading, scintillation and reduction of resolution, contrast, etc., which are caused by the smaller scales in the flow, including the coherent motion in the flow close to the aperture and the coherent structures typically associated with turbulent flows. Aberrations caused by these unsteady flow structures cannot currently be controlled using optical systems and new approaches are necessary for their mitigation.

A relatively new approach to understanding and controlling the optical aberrations observed in airborne applications is to directly control the coherent structures in the flowfield over the aperture to minimize their strength. It has been shown that these coherent structures are responsible for the vast majority of the optical distortions in aero-optical systems.[2] If

these structures can be controlled, their aberrations could be reduced, significantly improving the performance of airborne optical systems.

Open-loop flow control, where small amplitude disturbances are introduced into the flow at sensitive locations, utilize the flow's inherent instabilities to amplify the disturbance input and achieve large global changes in the flowfield. Both passive and active flow control systems are possible. A well-known example of the former is the delay of separation on airfoils at large angles-of-attack by using vortex generators to introduce small vortices upstream of the separation point to keep the flow attached by energizing the boundary-layer. In active systems, energy is typically added or subtracted from the flow by active introduction of mass using, for example, micro jets.

The understanding gained from experiments and simulations of such open-loop flow control systems is critical to developing feedback flow control strategies. Feedback or closed-loop flow control systems contain, in addition to the actuators used in open-loop flow control, sensors and a controller that processes the sensor information and determines the actuator output; thereby tailoring the actuation to the instantaneous flow-state with the goal of achieving better control for a larger range of flow conditions, ideally at a smaller energy expenditure.

Previous research has shown that a flow-state database is required to design an efficient feedback flow control system. In particular, sensor placement, including both the number and location of the sensors, the development of the flow-state estimator, as well as the design of the controller heavily rely on accurate instantaneous information about the flowfield.[4] Such a flow-state database has to contain flowfield data that capture the essential physical processes to be controlled. In addition, the effect of actuation on the flowfield has to be included in the database, since feedback control, if effective, will alter the flowfield significantly. While the ultimate flow-state is unknown before a controller is employed, important insights into the dynamics of the flow can be gleaned from the flow's response to periodic forcing, especially from the transient behavior that occurs when the forcing is turned on or off.

With this database, a reduced-order model (ROM) of the flow dynamics can be constructed using state-of-the-art modeling techniques.[5,6] The main objective of this ROM is to capture the relevant flow dynamics with a small number of variables, reducing the enormous number of degrees-of-freedom in a typical simulation of experiment to a manageable set of data for controller development. In addition, this ROM is also used as the basis for sensor placement studies and development of flow-state estimation schemes.

In this paper, the simulations for the development of such a database of the three-dimensional shear-layer behind a notional two-dimensional turret are described. First, the numerical method is discussed, including the grid-generation. Second, some representative results of the simulations are shown. Finally, some conclusions are drawn and an outlook of future work is given.

2. Numerical Method

The computational fluid dynamics (CFD) simulations were performed using COBALT V4.4 and V5.0 from Cobalt Solutions, LLC, an unstructured finite-volume code developed for the solution of the compressible Navier-Stokes equations. The basic algorithm is described in Strang, et al., although substantial improvements have been made since then.[7] The numerical method is a cell-centered finite-volume approach applicable to arbitrary cell topologies. The spatial operator uses the exact Riemann Solver of Gottlieb and Groth, least-squares gradient calculations using QR factorization to provide second-order-accuracy in space, and TVD flux limiters to limit extremes at cell faces.[8] A point implicit method using analytic first-order Jacobians is used for advancement of the discretized system. For time-accurate computations, a second-order-accurate method with Newton sub-iterations is employed. Parallel performance is achieved using the ParMETIS domain decomposition library for optimal load balancing with a minimal surface interface between zones.[9] Message Passing Interface (MPI) for communication between processors, with parallel efficiencies above 95% on as many as 1,024 processors.[10]

Delayed Detached-Eddy Simulations (DDES) are a well-known hybrid technique first proposed by Spalart for prediction of turbulent flows at high-Reynolds numbers.[11] For natural applications of DES, Reynolds-averaged Navier-Stokes (RANS) is applied in the boundary-layer, while outside the boundary-layer in the separated region, Large-Eddy Simulation (LES) is used. Recent DES predictions of the flow around complex configurations (all using Cobalt) include the massively- separated flow around an F-15E at 65° angle-of-attack (this simulation was the first eddy-resolving simulation of flow around a full aircraft configuration), transonic shock-separated flow over an F/A-18E, and vortex breakdown on an F-18C.[12–14] DDES has also been used recently for aero-optics simulations.[16]

To build the database of flow-states that will be used to define the reduced-order model of the flowfield, unforced simulations were performed first. In a second step, open-loop active flow control (AFC), which was implemented using a blowing/suction boundary condition (see below), was studied and the data was added to the database.

For the computation of the optical path difference (OPD), the instantaneous density field data obtained from the CFD simulations was used to derive the corresponding index of refraction field and integrate it along the optical beam.

2.1 Computational Domain and Discretization

The geometry under consideration is a two-dimensional turret geometry investigated experimentally by Gordeyev, et al.[17] A sketch of the geometry is shown in Figure 1. The radius of the turret cylinder is R=2" and the wind-tunnel cross-section upstream is 4"×4" and 5.625"×4" downstream. The aperture size is 2"×4". While the look-back angle was adjustable in the experiments, it was fixed for the simulation at Θ=120°.

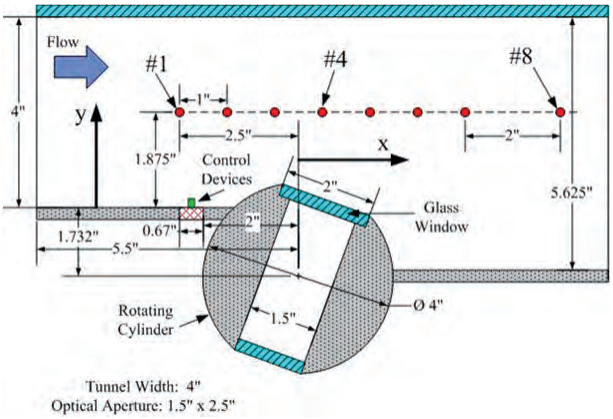


Figure 1. Sketch of turret geometry[17]

Due to the simplicity of the geometry, the CAD tools in the SimCenter software suite were used to generate the computational domain.[18] The basic computational domain is shown in Figure 2. The coordinate system is oriented such that the x-axis points downstream, the y-axis vertically up, and the z-axis is in the spanwise direction. The y- and z-domain size matched the wind-tunnel experiment, while the x-domain size was adjusted upstream to match the incoming boundary-layer characteristics (see results section) and downstream to allow the large flow structures to dissipate before reaching the outflow boundary. The final domain length upstream of the turret was L_1=0.3m and L_2=0.3m downstream.

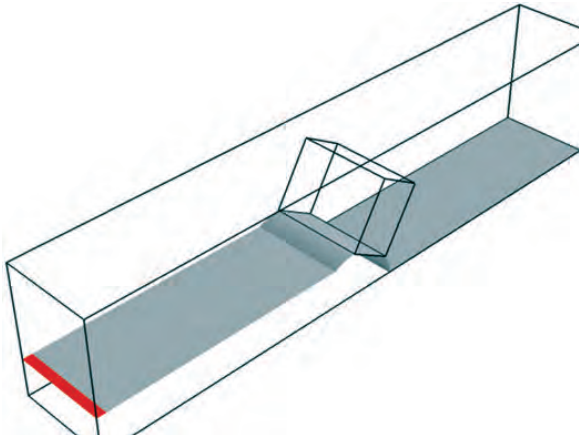


Figure 2. Computational geometry. The small box indicates the optical beam grid.

For the simulations with forcing, two different geometries with blowing/suction slots were created. For the geometry labeled 'Slot1' in the following, the slot was upstream of the aperture, oriented vertically upward. The 'Slot2' geometry featured a slot of the same size, located on the upstream-edge of the aperture and oriented normal to the aperture (60°measured against the positive x-axis). Details of both slot geometries are shown in Figure 3.

The grid-spacing was chosen such that the blowing and suction slot opening (b=1mm) could be resolved with approximately ten grid-points, resulting in a step-size of dx=0.1mm; the boundary-layer grid in the slot increases that resolution. A zoomed-view of the grid over the turret and in the downstream shear-layer is shown in Figure 4. Simulation results indicate that the average near-wall resolution is $y1^+ \cong 0.4$. All grids had approximately 38M cells.

A time-step study showed that $\Delta t = 1 \cdot 10^{-6}$s is sufficient to resolve the relevant turbulent scales, as well as to capture the vortex shedding in the separating shear-layer (see results).

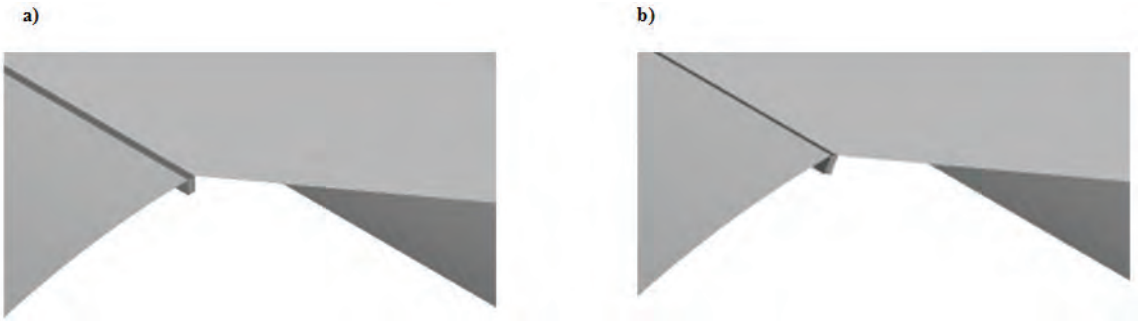


Figure 3. Slot geometry; a) Slot 1, vertical blowing/suction, b) Slot 2, blowing/suction normal to aperture

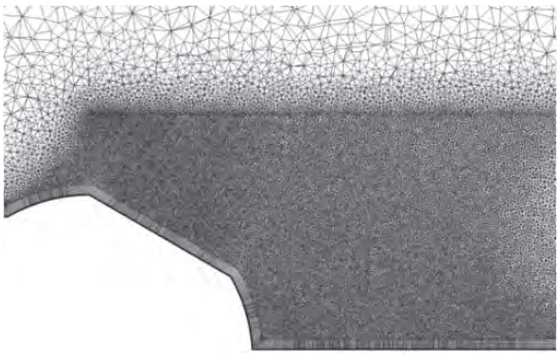


Figure 4. Grid for geometry without slot

To match the experimental setup as closely as possible, the walls (bottom, sides, and top) were modeled as a solid, no-slip wall. The inflow and outflow boundary conditions were prescribed as modified Riemann invariants, where the outflow pressure was lowered in order to maintain a prescribed inflow velocity. A small strip using a slip-wall boundary condition was used near the inflow (see red strip in Figure 2) to minimize interference between the solid wall and modified Riemann boundary conditions. Finally, blowing and suction mass flow rates could be prescribed on the bottom of the blowing and suction slot shown in Figure 3.

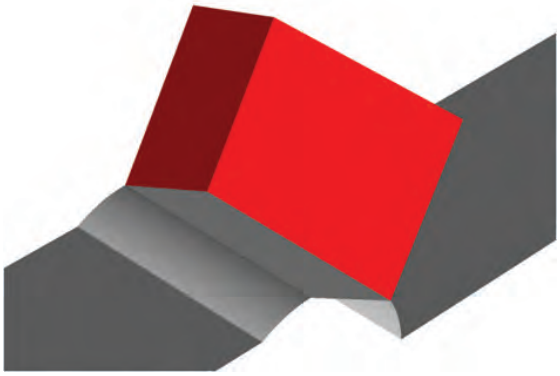


Figure 5. Optical beam grid on the aperture

For the computation of the optical path difference, a Cartesian grid was created above the aperture (see Figure 2 and Figure 5). With the η-direction of the beam grid normal to the aperture, the integration along the beam path, which is now aligned with the η-grid lines, is greatly facilitated. Cobalt's 'tap' feature is used to extract flowfield information at the grid-points in the beam grid. This ensures that the numerical interpolation onto the beam grid matches the discretization used in the Cobalt flow solver described above. The grid extends across the whole aperture, from $\xi/R=0$ to $\xi/R=1$, $\zeta/R=0$ to $\zeta/R=2$ (ξ is downstream, ζ spanwise along the aperture surface). A uniformly-spaced grid with $61 \times 81 \times 131$ points was used in the ξ-, η-, and ζ-directions, resulting in 597,861 points for the computation of the optical properties.

2.2 Computational Parameters

In order to match the experimental flow conditions, the inflow Mach number was maintained at M=0.4. Standard atmospheric sea-level pressure and temperature were assumed at the inflow, and the outflow pressure was lowered to maintain the inflow velocity. The free-stream velocity upstream of the turret is approximately U_0=140m/s. The resulting Reynolds number, based on the 'step-height', i.e., the elevation difference between the highest point on the turret (y_{max}=R) and the floor behind the turret is Re=470,000.

Compute times, including the data I/O, were measured at approximately 4.6 CPUh/iteration (grid size approx. 38M cells).

3. Results

3.1 Unforced Flow

3.1.1 Incoming Boundary-Layer Profile

To determine if the simulations achieve the same flow conditions as the experiment, the mean boundary-layer thickness at x/R=-1.25 upstream of the turret was measured and compared with the experiments. The results are summarized in Table 1. The good comparison indicates that the inflow length was chosen properly and that a fully-turbulent boundary-layer exists upstream of the turret. This is important to ensure that the separating shear-layer has matching characteristics, which determine the initial development of the shear over the aperture, and therefore the development of the optically-detrimental large-scale flow structures.

Table 1. Comparison of experimental and computational boundary-layer parameters

	Computational	Experimental	Difference
Thickness (δ)	5.08 mm	5.00 mm	1.6%
Momentum Thickness (δ*)	0.90 mm	0.81 mm	11.1%
Displacement Thickness (θ)	0.66 mm	0.69 mm	-4.3%

At the separation point, the boundary-layer thickness is δ=4mm, slightly thinner than upstream of the turret due to the favorable pressure gradient. With this boundary thickness, the natural shedding frequency can be estimated based on the Strouhal number $St_\theta=F_\theta\cdot\theta/U_0$=0.012 (based on the momentum thickness) to be $F_\theta\approx$4,300Hz.[19] Another important time- scale is the frequency of the turnover in the recirculation region, given by the Strouhal number of $St_H=F_H\cdot H/U_0$=0.185 (based on the step-height). For the current geometry, this yields a dimensional frequency of F_H=500Hz. In order to obtain statistically-stationary results, it is important to ensure that the flow in the recirculation is fully-developed before collecting data; this determines the overall number of computational time-steps.

3.1.2 No Forcing Slot

The unforced flow solution serves as a baseline case for comparison of the effect of forcing on the flow, and also as the initial condition for the open-loop forced simulations. Simulations were performed starting from a steady simulation of 1,000 iterations that established the main flow features. A total of 10,000 iterations were run for the unsteady simulation, which allowed for approximately ten eddy turnover-times when considering the time-scale of the recirculation region (see above). An instantaneous picture of the flow solution is shown in Figure 6. In the figure, the ramp geometry is shown in grey, while the structures in the flowfield are visualized using an iso-surface of $Q=3\cdot10^7 1/s^2$.[20] The Q-criterion has the advantage that vortical structures are identified without regard of mean-shear, which is particularly important in flows such as the shear-layer under investigation. The observed vortex shedding frequency is approximately F_0=4,000Hz, in very good agreement with the theoretical estimate given above. This result is significant in that it is a validation that using the DDES methodology does not significantly influence the naturally-occurring flow instability, which is also corroborated by the eddy-viscosity in the shear-layer (Figure 7).

The dominant coherent structures seen in Figure 6 just downstream of the separation point quickly break up into three-dimensional structures and S-shaped structures form, in accordance with earlier observations.[21] However, it is interesting to note that in the two-dimensional cut shown in Figure 7, the large coherent structures in the shear layer, identified by the higher eddy-viscosity ratio, persist far downstream of the separation point.

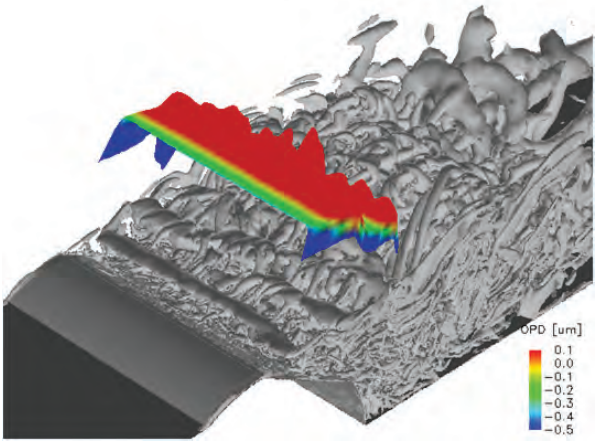


Figure 6. Unforced flow. Instantaneous flow structures and wave-front. Geometry without forcing slot.

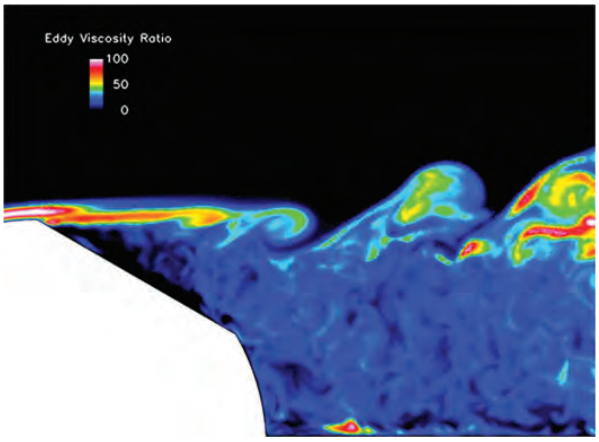


Figure 7. Eddy-viscosity ratio

Above the flow structures, the wave-front of a notional beam originating from the aperture surface is plotted, colored by OPD. The plot shows that the optical distortions grow in the streamwise direction. Comparing the wave-front with the structures in the flowfield shows that the 'valleys' in the wave-front coincide with the locations of the strongest structures in the shear-layer, consistent with the decreased density in the core of the structures, corroborating that the shear-layer structures are indeed responsible for the largest optical aberrations. It is also noteworthy that the side-wall boundary-layers show a large effect on the wave front.

3.1.3 Forcing Slot 1

Figure 8a shows the results from a simulation using the geometry with forcing slot 1. As for the results shown in Section 3.1.2, flow structures are visualized using a $Q=3\cdot10^7\ 1/s^2$ iso-surface and the optical wave-front is shown in color. Comparing Figure 8 with Figure 6 shows that the slot, even without active forcing, has a pronounced effect on the flow-field. The shear-layer structures are shed at a significantly higher frequency (approximately F_1=14kHz), and while pairing seems to occur, three-dimensional distortions of the initially very coherent structures seems to prevent distinct, large-scale structures. This is also evident in the wave-front, which only shows significant distortions toward the downstream end of the aperture.

3.1.4 Forcing Slot 2

Results for configuration forcing slot 2 are shown as the instantaneous $Q=3\cdot10^7\ 1/s^2$ iso-surface in Figure 8b). The results are very similar to configuration slot 1, suggesting that in both configurations the forcing slot acts as a passive control device due to internal resonance.

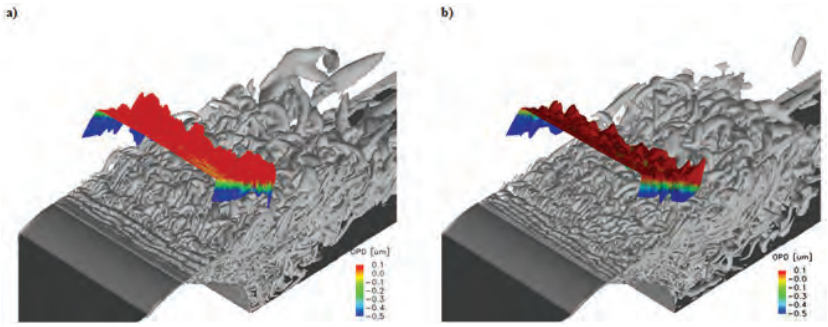


Figure 8. Unforced flow. Instantaneous flow structures and wave-front. a) Geometry with forcing slot 1, b) Geometry with forcing slot 2

3.2 Open-Loop Forcing

To investigate the effect of open-loop active forcing on the development of structures in the shear-layer over the optical aperture, forcing was introduced at a frequency of F_F=10kHz and an amplitude of A/U_0=0.1. Results are presented for both slot configurations.

3.2.1 Forcing Slot 1

When forcing at F_F=10kHz and an amplitude of A/U_0=0.1 is applied using slot 1, the initial flow development is markedly changed (Figure 9a). Close to the slot, a distinct, highly-coherent structure is visible in the figure. Comparing Figure 8a with Figure 9a also indicates that although the initial flow structures are more coherent, three-dimensional effects emerge more rapidly than in the unforced case and result in a highly three-dimensional flowfield with small- scale structures. While adding structures to the flow is not desired, it should be noted (see Equations 1 and 2) that the strength of the structure, indicated by its density change, as well as its size are important to determine its effect on the beam propagation. The structures introduced using the forcing parameters given above do not produce a significant effect on the optical beam propagating through the shear-layer.

3.2.2 Forcing Slot 2

Results of forcing at the default frequency and amplitude using slot 2 are shown in Figure 9b. The comparison between the results of simulations with the two forcing slot geometries does not show a significant difference, suggesting that either slot configuration could be used in an application of active flow control for this geometry.

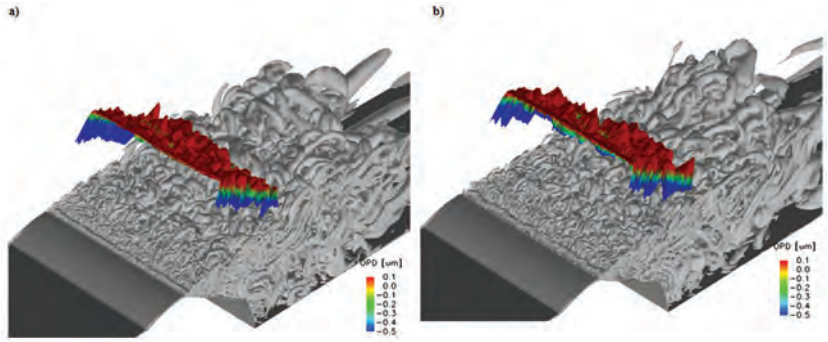


Figure 9. Instantaneous flow structures, forcing with F=10kHz, A/U_0=0.1; a) Slot 1 geometry, b) Slot 2 geometry

4. Conclusion

In this paper, an overview of the generation of a flow-state database for the development of feedback flow control strategies was presented. The flowfield under consideration is the shear-layer forming over the aperture of a two-dimensional turret looking back at an angle of 120°. Using High Performance Computing Modernization Program (HPCMP) resources under the Challenge Project C4L, Delayed Detached-Eddy Simulations using Cobalt were performed to study the unforced and open-loop forced flowfield using three different geometries, one without a forcing slot and two with different forcing

slot configurations. The results show that significant changes occur in the flowfield with the geometries that incorporated a forcing slot even when no forcing was applied. Due to the slot, vortex shedding at a high-frequency was observed, possibly due to a resonance mechanism of the flow with the slot. The shear-layer structures at this high-frequency are less coherent than the ones observed in the geometry without a slot. When forcing was applied at an amplitude of A/U_0=0.1 and a frequency of F_F=10kHz, the flow structures initially exhibited a marked increase in spanwise coherence, but also resulted in the earlier emergence of three-dimensional structures, which reduce coherence and limit the strength of the structures as measured by their density variations.

The effect of the large coherent structures in the shear-layer on a beam propagating through the flowfield was examined, and the optical path difference (OPD) was computed on a beam grid extracted from the simulation data using Cobalt's tap feature. For the geometries with a forcing slot, optical distortions were markedly reduced, even without active forcing, due to the higher shedding frequency and smaller flow structures. The results of these simulations confirm that the large, coherent structures in the shear-layer are responsible for the largest optical aberrations, and that even small changes in geometry, such as the inclusion of a forcing slot, can have a large effect on the flowfield and its effect on optical beam propagation. Forcing resulted in a further reduction of the optical distortions due to breakup of the coherent motion further upstream than in the unforced case.

The current results and further simulations at different frequencies and amplitudes will be used to compose a flow-state database for the development of a reduced-order model (ROM) of the flow. In turn, this model will be the basis for the design of feedback control strategies, including sensor placement, sensor-based flow-state estimation, and controller design. Feedback flow control holds the promise to play a key role in mitigating the optical distortions in the flow over an airborne optical platform.

Acknowledgements

This work was funded in part by the US Air Force Office of Scientific Research (AFOSR), Doug Smith, Program Manager. The Department of Defense (DoD) HPC Modernization Program supported this project by supplying supercomputer time under Challenge Project C4L at the DoD Supercomputing Resource Center at Maui and Alaska. We would like to thank Dr. Keith Bergeron and LtCol Charles Wisniewski, Directors, Modeling and Simulation Research Center, US Air Force Academy, for their support and for providing additional computing, storage, and visualization resources.

References

1. McMackin, L., Hugo, R.J., Pierson, R.E., and Truman, C.R., "High-speed optical tomography system for imaging dynamic transparent media", *Optics Express*, 1(11), pp. 302–311, 1997.

2. Jumper, E.J. and Fitzgerald, E.J., "Recent advances in aero-optics", *Prog. Aero. Sci.*, 37, pp. 299–339, 2001.

3. Settles, *Schlieren & Shadowgraph Techniques*, Springer, 2001.

4. Siegel, S., Seidel, J., Fagley, C., Luchtenburg, D., Cohen, K., and McLaughlin, T., "Low-dimensional modeling of a transient cylinder wake using double-proper orthogonal decomposition", *J. Fluid Mech.*, 610, pp. 1–42, 2008.

5.Fagley, C., Balas, M., Siegel, S., Seidel, J., and McLaughlin, T., "Reduced-Order Model of Cylinder Wake with Direct Adaptive Feedback Control", *AIAA Paper 2009-5856*, 2009.

6. Seidel, J., Siegel, S., McLaughlin, T., and Fagley, C., "Feedback Flow Control of a Shear-Layer for Aero-Optic Applications", *AIAA Paper 2010-0356*, 2010

7. Strang, W., Tomaro, R., and Grismer, M., "The defining methods of Cobalt60: A parallel, implicit, unstructured Euler/Navier-Stokes flow solver", *AIAA Paper 1999-0786*, 1999.

8. Gottlieb, J.J. and Groth, C.P.T., "Assessment of Riemann solvers for unsteady one-dimensional inviscid flows for perfect gases", *J. Comp. Phys.*, 78(2), p. 437–458, 1988.

9. Karypis, G., Schloegel, K, and Kumar, V., P*arMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering Library Version 1.0*, 1997.

10. Grismer, M.J., Strang, W.Z., Tomaro, R.F., and Witzemman, F.C, "Cobalt: A parallel, implicit, unstructured Euler/Navier-Stokes solver", *Advances in Engineering Software*, 29(3–6), pp. 365–373, 1998.

11. Spalart, P.R., "Detached-Eddy Simulation", *Ann. Rev. Fluid Mech.*, 41, pp. 181–20, 2009.

12. Forsythe, J.R., Squires, K.D., Wurtzler, K.E., and Spalart, P.R., "Detached-eddy simulation of the F-15E at high-alpha", *J. Aircraft*, 41(2), pp. 193–200, 2004.

13. Forsythe, J.R. and Woodson, S.H., "Unsteady CFD calculations of abrupt wing-stall using detached-eddy simulation", *AIAA Paper 2003-0594*, 2003.

14. Morton, S.A., Steenman, M.B., Cummings, R.M., and Forsythe, J.R., "DES grid resolution issues for vortical flows on a delta wing and an F-18C", *AIAA Paper 2003-1103*, 2003.

15. Morton, S.A., Cummings, R.M., and Kholodar, D.B., "High-resolution turbulence treatment of F/A-18 tail buffet", *AIAA Paper 2004-1676*, 2004.

16. Seidel, J., Fagley, C., and Decker, R., "Mitigation of Optical Distortions in a Free Shear-Layer using Feedback Flow Control", D*oD HPC User Group Conference,* 2010

17. Gordeyev, S., Cress, J.A., Smith, A., and Jumper, E.J., "Improvement in Optical Environment over Turrets with Flat Window Using Passive Flow Control," *AIAA Paper 2010-4492*, 2010

18.Marcum, D. and Weatherill, N., "Unstructured grid-generation using iterative point insertion and local reconnection", *AIAA J.*, 33(9), pp. 1619–1625, 1995.

19. Hasan, M.A.Z., "The flow over a backward-facing step under controlled perturbation: laminar separation", *J. Fluid Mech.*, 238, pp. 73–96, 1992

20. Jeong, J. and Hussain, F., "On the identification of a vortex", *J. Fluid Mech.*, 285, pp. 69–94, 1995.

21. Oster, D. and Wygnanski, I., "The forced mixing-layer between parallel streams", *J. Fluid Mech.*, 123, pp. 91–130, 1982.

Numerical Investigation of Three-Dimensional External Flow Separation

R. Jacobi, A. Gross, and H.F. Fasel
Department of Aerospace and Mechanical Engineering, University of Arizona, Tucson, AZ
{rjacobi, agross, faselh}@email.arizona.edu

Abstract

Flow separation is mostly undesirable, as it negatively affects hydrodynamic performance. In addition, separation is often associated with unsteadiness which is caused by large coherent structures that are a consequence of hydrodynamic instability mechanisms of the mean flow. Despite the fact that most of the past research has focused on two-dimensional separation, for practical applications flow separation is almost always three-dimensional. The three-dimensional character of separation is particularly relevant when low-aspect ratio geometries are considered. We are employing direct numerical simulations for investigating the highly-complex flow physics of three-dimensional laminar separation bubbles. By subjecting three-dimensional laminar separation bubbles on a flat plate to different low- and high-amplitude pulse disturbances, we are investigating the hydrodynamic instability mechanisms of three-dimensional separation in boundary layers. We are also investigating laminar three-dimensional separation from axi-symmetric bodies at angle-of-attack. A geometry that was derived by shortening the Defense Advanced Research Projects Agency (DARPA) bare hull Suboff model features a very thin steady laminar separation bubble and two pronounced counter-rotating vortices on the leeward side. When the fore-body shape is replaced by a hemisphere, a large laminar separation bubble is obtained at the nose of the model that is shedding strongly.

1. Introduction

Separation for Navy-relevant geometries (submarines, torpedoes, fins, low-aspect ratio lifting or control surfaces) is always three-dimensional (3D) and associated with considerable unsteadiness. Coherent structures that are the consequence of hydrodynamic instabilities, and which originate in the separated shear layer influence separation and reattachment, especially the size and shape of the separated region, which, as a consequence, affects the drag and lift characteristics. An improved understanding of the flow physics governing 3D separation in general, and the dynamics of such structures in particular, is desirable as this may lead to novel devices or strategies that may help prevent or control separation.

In our research we are employing both direct numerical simulations (DNS) and water tunnel experiments. We are investigating laminar 3D separation bubbles on a flat plate. The separated region on the flat plate is induced by the close proximity of an axi-symmetric displacement body. Because separation is generated on a flat plate, flow curvature effects are deliberately excluded. Insight into the hydrodynamic instability mechanisms can be obtained when instabilities are excited directly through the introduction of controlled disturbances. Compared to the experiments, this can be accomplished relatively easily in numerical simulations because location and type of the disturbances can be specified exactly.

In parallel, we are also investigating laminar separation bubbles on axi-symmetric bodies that are related to geometries used in practical applications. The Defense Advanced Research Projects Agency (DARPA) bare hull submarine geometry [Groves, et al., 1989] was shortened to fit into our water tunnel and allow for large angles-of- attack. Preliminary results from both water tunnel experiments and DNS show a very shallow separation bubble that is not shedding. Since our primary interest is in laminar 3D separation, we decided to use a different fore-body shape. We chose a hemisphere fore-body because earlier research indicated considerable flow separation for this particular geometry [Bippes, 1986; Wang and Hsieh, 1992].

In this paper, we first discuss the computational methods employed. We then show results for the flat plate model geometry. A brief summary of the results for the two axi-symmetric bodies follows. Finally, a short summary and conclusions are provided.

2. Computational Method

We employ a two-pronged computational approach: i) An incompressible finite-difference code is used for simulating 3D separation on a flat plate, and ii) A compressible finite-volume code is employed for simulating the flow over axi-symmetric bodies at incidence.

The higher-order accurate incompressible finite-difference code employed for the flat plate model geometry simulations was developed in our laboratory [Meitz and Fasel, 2000]. Derivatives in the streamwise and wall-normal directions are discretized with fourth-order-accurate compact differences. A pseudo-spectral method is employed in the spanwise direction and a fourth-order-accurate Runge-Kutta scheme is employed for time-integration. A hybrid parallelization approach, using both Message Passing Interface (MPI) and shared memory parallel programming (OpenMP) was adopted to make optimal use of the available supercomputers.

For simulations of the axi-symmetric bodies we are employing a higher-order-accurate compressible finite-volume code that was also developed in our laboratory [Gross and Fasel, 2008]. The Navier-Stokes equations are discretized with a ninth-order-accurate upwind scheme for the convective terms and a fourth-order-accurate discretization for the viscous terms. An implicit second-order-accurate time-integration allows for Courant-Friedrichs-Lewy (CFL) numbers in the order of 1,000. The code was parallelized using MPI.

3. Three-Dimensional Separation on a Flat Plate

Three-dimensional flow separation on a flat plate is generated by placing a 3D axisymmetric displacement body at a distance, h, from the flat plate (Figure 1). The cone-shaped displacement body has a hemispherical front end of diameter, d, and boundary layer suction was employed in the rear part of the body to prevent flow separation from the displacement body (For more details regarding experiments see Kremheller and Fasel, 2010). The cone opening angle is 20deg. The local boundary layer displacement thickness at separation can be varied by changing the downstream distance, s, of the body from the flat plate leading edge or by varying the free-stream velocity, and thus the Reynolds number. The results shown in this paper are for h=0.1m, s=0.2m, d=0.1m.

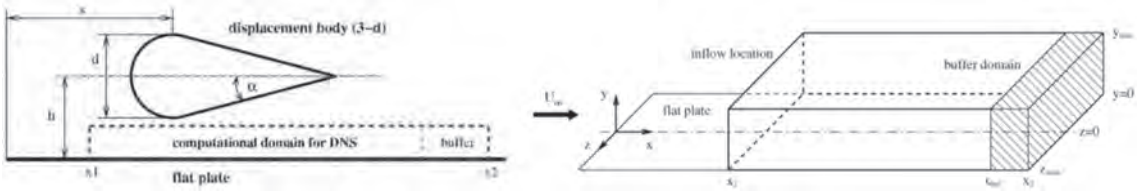


Figure 1. Experimental set-up and computational domain for 3D separation bubble simulations

As shown in Figure 1, the computational domain for our DNS is located below the displacement body. At the inflow boundary, steady velocity and vorticity profiles are imposed. At the free-stream boundary, y_{max}, the flow is assumed to be irrotational and all vorticity components and their derivatives are set to zero. A Dirichlet boundary condition is applied for the wall-normal velocity, which is obtained from a potential flow solution for the flow over the displacement body. The potential flow solution is mirrored in the wall-normal direction to account for the presence of the flat plate and in the spanwise direction for simulating the tunnel side walls. At the wall, y=0, no-slip and no-penetration conditions are imposed. To prevent reflections at the outflow boundary at x_2, the flow is "re-laminarized" in a buffer domain starting at x_{buf}. Inside the buffer domain the disturbance vorticity components are gradually ramped down to zero. The dimensions of the computational domain were 0.768m×0.048m×0.64m in the streamwise, wall-normal, and spanwise direction, respectively. The computational grid for the stability simulations with a steady baseflow had 769×129 grid points in the streamwise and wall-normal direction. For the simulations presented in this paper full-Fourier transforms with 321 modes (513 collocation points) were used in the spanwise direction.

Using DNS we carried out stability investigations for steady separation bubbles with a Reynolds number based on freestream velocity and displacement body diameter of 7,500. Starting from the steady flowfield (Figure 2) as initial condition, or baseflow, we introduced controlled perturbations through a spanwise blowing-and-suction slot that was located at the upstream end of the adverse pressure gradient region at x=0.245m. We used a single sine pulse disturbance in time to perturb a wide range of frequencies and streamwise wave numbers. The velocity distribution over the slot width (i.e., in the streamwise direction) was modeled by a monopole. The maximum amplitude of the wall-normal velocity was $10^{-8}U_\infty$ for the low-amplitude forcing and $10^{-2}U_\infty$ for the high-amplitude forcing. Disturbances were introduced for different spanwise

modes to excite oblique (or asymmetric) disturbance waves with varying wave angles (relative to the downstream direction), where a higher spanwise wave number, β, implies greater obliqueness. Figure 3 shows instantaneous visualizations of the perturbations for different spanwise wave numbers, β. In addition to these simulations where only one oblique disturbance wave was forced we also performed simulations where we perturbed a broad range of spanwise modes, in particular the 0^{th} spanwise or two-dimensional (2D) mode and the higher modes 1–40.

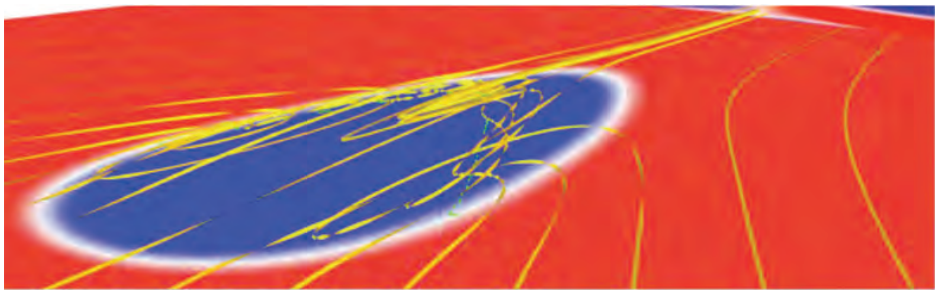


Figure 2. Perspective view of steady separation bubble at Re=7,500. Shown are 3D streamlines and color contours of the spanwise vorticity component, where blue indicates reverse flow.

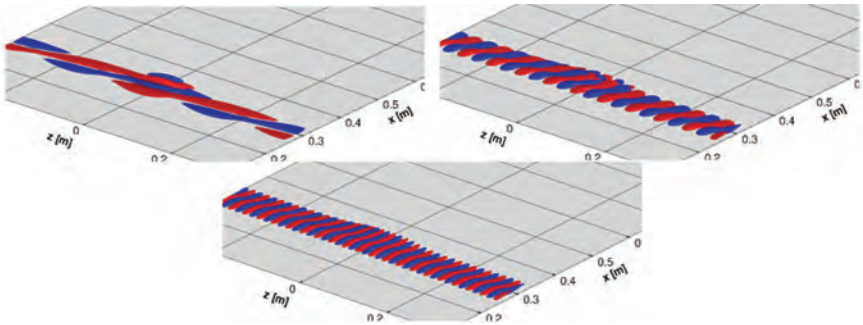


Figure 3. Visualization of oblique disturbance waves near blowing-and-suction slot. The spanwise wave number was β=2 (left), 10 (center), and 20 (right).

Figures 4–6 visualize the temporal and spatial development of the wave packets resulting from low-amplitude disturbance waves with spanwise wave numbers of 2, 10, and 20 respectively. Shown are iso-surfaces of the streamwise disturbance velocity u'. The physical structure of the low-amplitude wave packets indicates the absence of nonlinear interactions. For the more oblique disturbances, the wave packets have two clearly distinct parts. The part directly corresponding to the perturbed spanwise wave number decays in the downstream direction, while the part of the wave packet resulting from the modulation of the disturbance wave by the separated shear layer grows in the downstream direction. For the most oblique disturbance wave (Figure 6), the amplification of the disturbances and thus the resulting wave packet is very weak. With increasing downstream distance the general structure of the wave packets becomes less dependent on the wave angle of the initial perturbations and overall the wave packets become very symmetric. Another interesting observation is that although only one set of oblique waves was generated by the forcing, all wave packets contain pairs of oblique waves due to the modification by the baseflow. The evolution of the wave packet resulting from the broad wave number (β=0−40) low-amplitude forcing shown in Figure 7 is very similar to the wave packet observed for the least oblique disturbance waves, indicating that the most amplified disturbances have low spanwise wave numbers (or small wave angles).

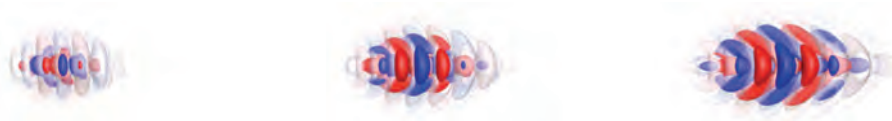


Figure 4. Downstream evolution of wave packet resulting from low-amplitude forcing of oblique mode with β=2. Shown are iso-surfaces of the streamwise disturbance velocity $|u'|=5*10^{-8}$ and $5*10^{-9}$ at t=5.76s (left), 7.68s (center), and 9.6s (right).

Figure 5. Downstream evolution of wave packet resulting from low-amplitude forcing of oblique mode with β=10. Shown are iso-surfaces of the streamwise disturbance velocity $|u'|=10^{-8}$ and 10^{-9} at t=5.76s (left), 7.68s (center), and 9.6s (right).

Figure 6. Downstream evolution of wave packet resulting from low-amplitude forcing of oblique mode with β=20. Shown are iso-surfaces of the streamwise disturbance velocity $|u'|=2*10^{-9}$ and $5*10^{-10}$ at t=5.76s (left), 7.68s (center), and 9.6s (right).

Figure 7. Downstream evolution of wave packet resulting from low-amplitude forcing of oblique modes with β=0-40. Shown are iso-surfaces of the streamwise disturbance velocity $|u'|=5*10^{-7}$ and 10^{-7} at t=5.76s (left), 7.68s (center), and 9.6s (right).

The wave packets resulting from the high-amplitude forcing are visualized in Figures 8–12. Similarity of the physical structure of the high-amplitude and low-amplitude wave packets is seen only for β=20. We attribute this to the weak amplification of very oblique disturbance waves. For all other spanwise wave numbers, the structure of the high-amplitude wave packets suggests non-linear interactions. For the less oblique disturbance waves the non-linear interactions occur downstream of the reverse flow region of the baseflow. Nevertheless, the wave packet resulting from the broad disturbance spectrum exhibits signs of non-linear interactions upstream of the reattachment location of the separation bubble (Figure 11). This is likely due to non-linear interactions between the various disturbance waves introduced simultaneously through the blowing-and-suction slot. The high-amplitude wave packets exhibit greater asymmetry than the corresponding low-amplitude wave packets, possibly due to stronger interactions of the disturbance waves with the baseflow. The evolution of the wave packet resulting from the high-amplitude broad spectrum is shown in greater detail in Figure 12. The wave packet displays a strong spanwise and streamwise modulation throughout its entire evolution. In contrast to the low-amplitude case, the wave packet contains a tail of highly oblique waves. Because no similarly pronounced tail is seen for simulations where only one spanwise wave number was forced, we speculate that the tail must be the result of nonlinear interactions and not of a linear amplification of highly oblique disturbances per se. In its last stages, some of the high-amplitude wave packets exhibit grid-mesh oscillations, indicating insufficient spanwise resolution.

Figure 8. Downstream evolution of wave packet resulting from high-amplitude forcing of oblique mode with β=2. Shown are iso-surfaces of the streamwise disturbance velocity $|u'|=5*10^{-2}$ and $5*10^{-3}$ at t=5.76s (left), 7.68s (center), and 9.6s (right).

Figure 9. Downstream evolution of wave packet resulting from high-amplitude of forcing oblique mode with β=10. Shown are iso-surfaces of the streamwise disturbance velocity $|u'|=10^{-2}$ and 10^{-3} at t=5.76s (left), 7.68s (center), and 9.6s (right).

Figure 10. Downstream evolution of wave packet resulting from high-amplitude forcing of oblique mode with β=20. Shown are iso-surfaces of the streamwise disturbance velocity $|u'|=2*10^{-3}$ and $2*10^{-4}$ at t=5.76s (left), 7.68s (center), and 9.6s (right).

Figure 11. Downstream evolution of wave packet resulting from high-amplitude forcing of oblique modes with β=0-40. Shown are iso-surfaces of the streamwise disturbance velocity $|u'|=5*10^{-2}$ and $5*10^{-3}$ at t=5.76s (left), 7.68s (center), and 9.6s (right).

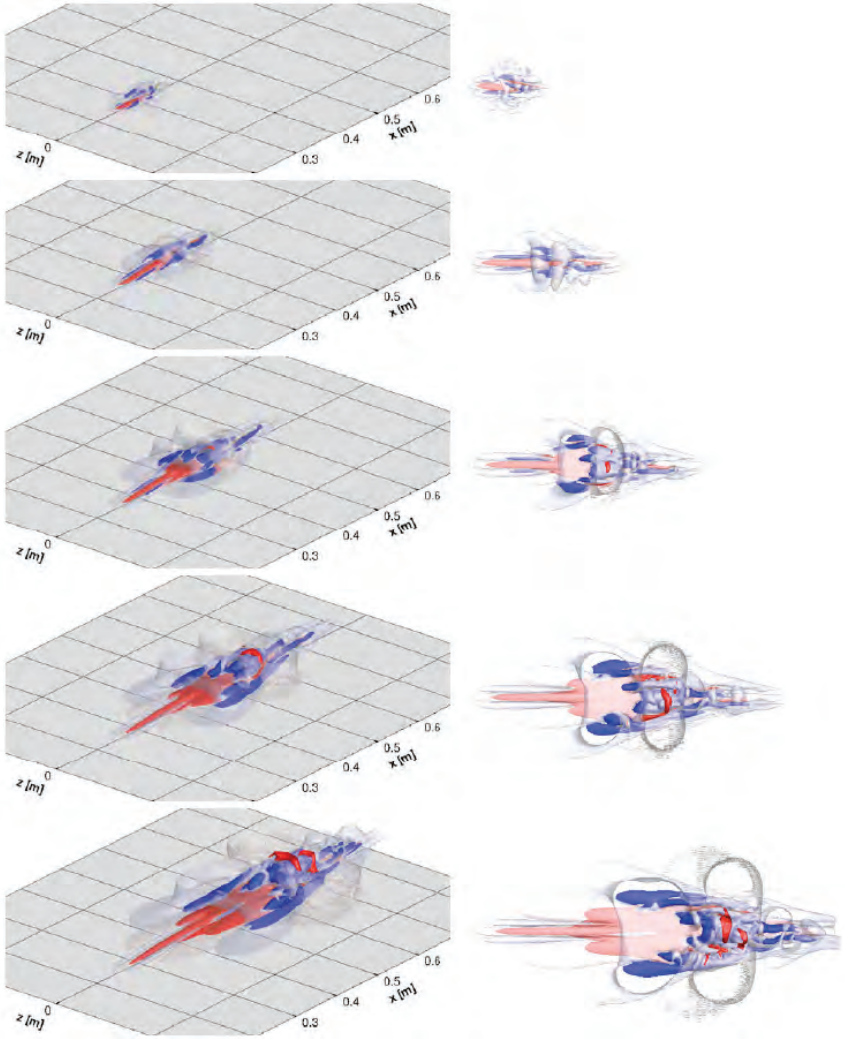


Figure 12. Downstream evolution of wave packet resulting from high-amplitude forcing of oblique modes with β=0−40. Shown are iso-surfaces of the streamwise disturbance velocity $|u'|=10^{-1}$ and 10^{-2}; perspective view (left) and top view (right).

4. Separation on Three-Dimensional Bodies

Direct numerical simulations were carried out for a shortened version of the DARPA bare hull Suboff model [Groves, et al., 1989] which has a fore-body and stern shape that is representative for submarines. We also investigated a body with identical length where the original nose section was replaced by a hemisphere. All length scales were made dimensionless with the body diameter. The body diameter and length were 1 and 4.81, respectively. A Poisson grid-generator was employed to generate an O-grid that was then extruded in the azimuthal direction. The outer (free-stream) boundary of the computational domain was located 50 body diameters from the center of the body. Finally, H-grids were inserted at the front and rear to remove the grid singularities associated with the axi-symmetric grid. Table 1 provides the block grid resolutions. Coarse grid simulations were carried out for both geometries. In addition, a fine-grid simulation was carried out for the hemisphere fore body geometry. A visual representation of the coarse grid is provided in Figure 13. Since a compressible code was employed, a reference Mach number had to be provided. To avoid compressibility effects in the strongly accelerated nose region, a low-reference Mach number of 0.1 was chosen. To prevent reflections of acoustic waves a characteristics-based non-reflecting boundary condition was applied at the free-stream boundary. Symmetry conditions were enforced in the spanwise direction. Simulations were carried out for Reynolds numbers based on body diameter of 10,000 and 20,000. The angle-of-attack was 30deg.

Table 1. Block grid resolution (number of cells in downstream, wall-normal, and azimuthal direction) and total number of cells

	Coarse grid	Fine grid
Block 1	186×120×32	386×200×64
Block 2	8×120×16	32×200×16
Block 3	8×120×16	32×200×16
total	744,960	5,145,600

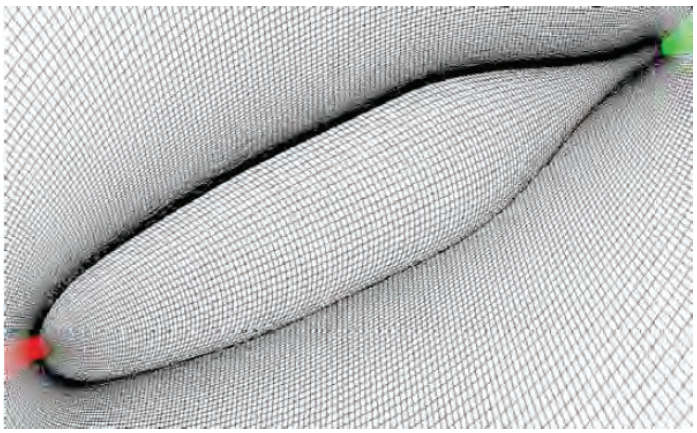


Figure 13. Coarse computational grid for Suboff geometry. Surface and symmetry plane grid.

Instantaneous flow visualizations obtained from coarse grid simulations for the Suboff geometry are shown in Figure 14. Iso-surfaces of the Q vortex identification criterion by Hunt et al., 1988, indicate areas where rotation dominates strain. The flowfields for the two Reynolds numbers are qualitatively similar. A shallow laminar separation bubble that is not shedding is located on the leeward side of the fore body. Another characteristic feature is two counter-rotating streamwise vortices on the leeward side which appear to counteract separation by transporting high-momentum fluid towards the wall. Finally, flow separation and wake unsteadiness can be observed near the stern. The same flow topology was also identified in water tunnel experiments at the Hydrodynamics Laboratory at the University of Arizona. Dye flow visualizations for Re=10,000 and a=30deg are shown in Figure 15. The dye is seen to spiral around the cores of the streamwise vortices.

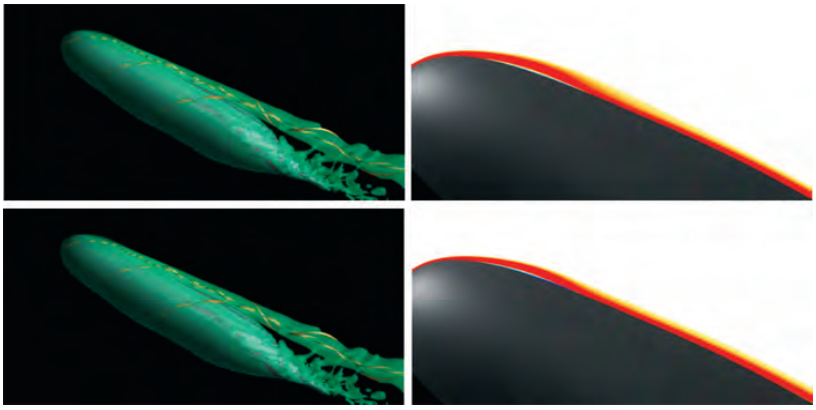


Figure 14. Suboff body at 30deg angle-of-attack. Iso-surfaces of Q=2 and streamlines (left) and contour lines of spanwise vorticity, -20<ωz<20, in symmetry plane (right) for Re=10,000 (top) and 20,000 (bottom)

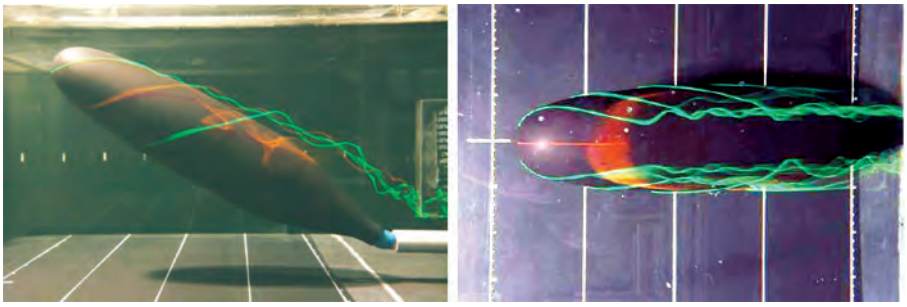


Figure 15. Suboff body at Re=10,000 and 30deg angle-of-attack. Water tunnel dye flow visualizations.

The focus of our research is on unsteady 3D separation. Since the laminar separation bubble on the fore body of the Suboff geometry is small and not shedding, we changed the fore body shape to a hemisphere cylinder. Earlier research for this configuration has shown larger separation bubbles and flow unsteadiness [e.g., Bippes, 1986; Wang and Hsieh, 1992]. The flowfield is visualized in Figure 16. As expected, much larger separation bubbles can be observed on the leeward side of the fore body. The bubble is shedding for both Re=10,000 and Re=20,000. As for the Suboff geometry at Re=10,000, two counter-rotating streamwise vortices are situated on the leeward side. For Re=20,000 the bubble is shedding more violently and the streamwise vortices appear to break up. Other arguments are possible. For example, one may argue that because of the reduced coherence of the streamwise vortices, separation is suppressed less and, therefore, the separation bubble on the fore body is shedding more strongly. Or one may argue that cross-flow instability of the flow around the body results in co-rotating vortices along the length of the body that weaken the streamwise vortices. Further research is needed to obtain deeper insight into the flow physics.

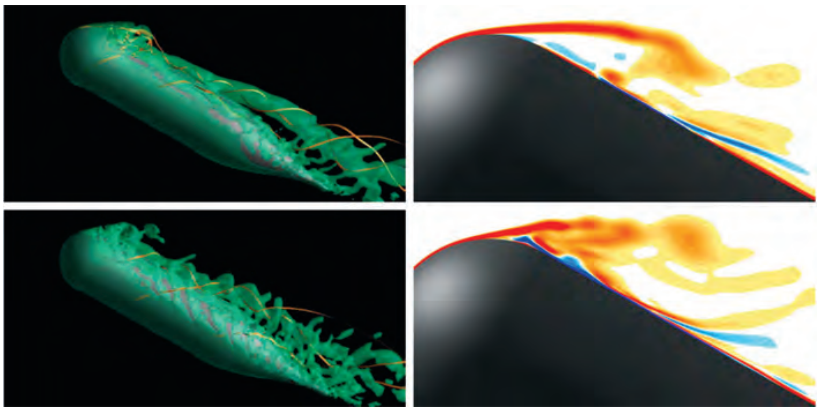


Figure 16. Hemisphere body at 30deg angle-of-attack. Iso-surfaces of Q=2 and streamlines (left) and contour lines of spanwise vorticity, -20<ωz<20, in symmetry plane (right) for Re=10,000 (top) and 20,000 (bottom).

We decided to increase the grid resolution for the Re=10,000 case. A comparison of the high- (Figure 17) and low-resolution (Figure 16, top) results reveals significant differences. Considerably more unsteadiness is seen in the bubble itself and downstream of the bubble for the higher resolution simulation. The simulations shown in Figures 14 and 16 were of an exploratory nature. Based on these exploratory simulations and water tunnel experiments, we will identify cases with significant laminar separation. Some of these cases will be investigated in great detail in the water tunnel and also in the simulations. Much higher grid resolution (such as shown in Figure 17) will be afforded for these future simulations. The combination of experiments and simulations will provide detailed insight into the flow physics of laminar 3D separation bubbles.

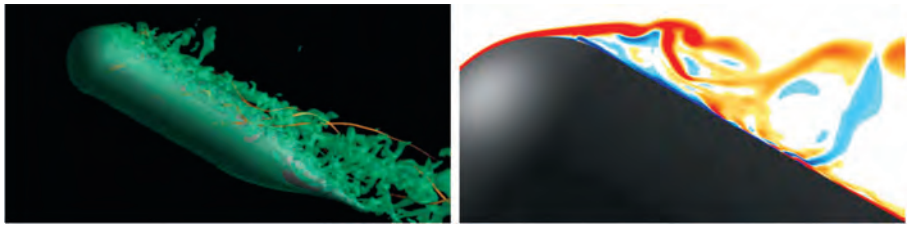


Figure 17. Hemisphere body at Re=10,000 and 30deg angle-of-attack. Iso-surfaces of Q=2 and streamlines (left) and contour lines of spanwise vorticity, -20<ωz<20, in symmetry plane (right). Fine-grid simulation.

5. Conclusion

We performed stability investigations of steady separation bubbles on a flat plate using direct numerical simulations, where we studied the flow response to linear and nonlinear oblique pulse disturbances. The resulting wave packets were found to be strongly modified by the separated shear layer of the baseflow, exhibiting a physical structure that did not correspond directly to the introduced disturbances. Disturbances with small wave angles were found to experience the strongest amplification. This is in agreement with earlier results that showed strong amplification for 2D disturbances [Gross, et al., 2010]. However, further investigations need to be undertaken to understand the instability mechanisms present in 3D separation bubbles.

We also carried out direct numerical simulations for two axi-symmetric geometries at 30deg angle-of-attack and for Reynolds numbers based on diameter of Re=10,000 and 20,000. A shortened DARPA Suboff bare hull geometry shows a very thin steady laminar separation bubble on the fore body and pronounced streamwise vortices on the leeward side. However, since our interest is mainly in 3D laminar separation, we then switched to a different geometry with a hemisphere fore body. In accordance with earlier experimental and numerical research, this geometry shows a very pronounced separation bubble that is shedding strongly. For Re=20,000 the coherence of the streamwise vortices on the leeward side appears diminished and the fore body separation is exacerbated compared to Re=10,000 case. Future simulations with increased resolution will allow for in-detail investigations of the hydrodynamic instability mechanisms responsible for the bubble shedding and the loss of coherence of the streamwise vortices.

6. Significance to DoD

Flow separation deteriorates hydrodynamic performance, increases structural loads, and may result in an undesirable acoustic signature. For Navy applications separation is mostly three-dimensional and unsteady. Our simulations of generic 3D separation bubbles advance the understanding of 3D separation, which paves the way for improved vehicle designs and the development of separation control devices that will increase hydrodynamic loading and reduce drag and acoustic signature.

Systems Used

Typical simulations were computed on up to 1,024 processors on the IBM P6 at the Navy DoD Supercomputing Resource Center (DSRC) and on the Cray XE6 and Cray XT5 at the Arctic Region Supercomputing Center (ARSC).

Acknowledgments

This research was supported by the Office of Naval Research (ONR) under grant number N00014-10-1-0404 with Dr. Ronald Joslin serving as program manager. Compute time was provided through Challenge Project ONRDC07692C4Y.

References

Bippes, H., "Experimental Investigation of Topological Structures in Three-dimensional Separated Flow", *Boundary-Layer Separation IUTAM Symposium*, London, 1986.

Groves, N.C., Huang, T.T., and Chang, M.S., "Geometric Characteristics of DARPA Suboff Models", *Report DTRC/SHD-1298-01*, David Taylor Research Center, Bethesda, MD, 1989.

Gross, A. and Fasel, H., "High-Order-Accurate Numerical Method for Complex Flows", *AIAA J.*, Vol. 46, No. 1, pp. 204–214, 2008.

Gross, A., Jacobi, R., and Fasel, H.F., "Investigation of Three-Dimensional Internal and External Flow Separation", *HPCMP Users Group Conference 2010*, Chicago, IL, 2010.

Hunt, J.C.R., Wray, A.A., and Moin, P., "Eddies, stream, and convergence zones in turbulent flows", *Report CTR-S88*, Center For Turbulence Research, Stanford, CA, 1988.

Kremheller, A. and Fasel, H.F., "Water Tunnel Experiments on Three Dimensional Separation Bubbles on a Flat Plate", AIAA-2010-4738, *40th AIAA Fluid Dynamics Conference and Exhibit*, Chicago, IL, 2010.

Wang, K.C. and Hsieh, T., "Separation Patterns and Flow Structures About a Hemisphere-Cylinder at High Incidences", *AIAA-92-2712*, 1992.

Meitz, H.L. and Fasel, H., "A Compact-Difference Scheme for the Navier-Stokes Equations in Vorticity-Velocity Formulation", *J. Comp. Phys.*, Vol. 157, pp. 371–403, 2000.

Numerical Investigations of Boundary-Layer Separation and Separation Control for the NACA 64(3)-618 Airfoil

A. Gross, W. Balzer, and H.F. Fasel
Department of Aerospace and Mechanical Engineering, University of Arizona, Tucson, AZ
{agross, wbalzer, faselh}@email.arizona.edu

Abstract

Flow separation from lifting surfaces deteriorates aerodynamic performance and is typically undesirable during nominal operation. For most practical Reynolds numbers, separation of laminar flow leads to transition. For moderate angles-of-attack, transition then results in reattachment. Experiments show that free-stream turbulence affects the laminar boundary layer upstream of transition as well as the separated flow. In general, free-stream turbulence accelerates transition and thus delays separation and/or favors reattachment. We are employing computational fluid dynamics for investigating the fundamental mechanisms of separation and transition for lifting surfaces. Using highly-resolved direct numerical simulations, we are investigating fundamental aspects of separation and transition in the presence of free-stream turbulence for canonical separation bubbles. Simulations with active separation control using a blowing and suction slot upstream of separation indicate that the primary physical mechanism remains unchanged for moderate free-stream turbulence levels. As direct numerical simulations become unaffordable for higher Reynolds numbers, we are also evaluating hybrid turbulence models. For a chord Reynolds number of 322,000, hybrid simulations based on a one-equation model are compared with measurements and data from an "under-resolved" direct numerical simulation.

1. Introduction

Flow separation from airfoils reduces lift and increases drag. While this may be desirable during certain flight segments, in general it has to be avoided because it negatively affects airplane performance (range, fuel burn, etc.). Separation can occur when large laminar runs over an airfoil (wing) are desired in order to minimize the skin-friction drag and, therefore, increase range and/or speed; or when airfoils that were originally designed for large Reynolds numbers are employed for off-design low Reynolds number applications. In such situations, boundary-layer separation can precede transition and, as a consequence, transition may occur in the separated region. Size and nature of the separated region itself affect the circulation and, thereby, separation itself.

Laminar airfoils are widely used in aerospace because of their greatly improved aerodynamic performance compared to older more conventional airfoils. By maintaining laminar flow over a large portion of the airfoil, the skin friction drag can be reduced. Hydrodynamic instability of the boundary layers, especially when associated with laminar separation, makes the laminar flow highly susceptive to free-stream turbulence (FST) and roughness. Experiments have shown that for sufficiently high FST levels transition to turbulence is accelerated. Clear air turbulence which is characterized by large length scales can reach high turbulence kinetic energy levels, and thus has to be considered when investigating separation and separation control.

For many Air Force applications, successful control of boundary-layer separation for lifting surfaces could lead to significant performance gains. This is especially true for unmanned aerial vehicles (UAVs) which have become increasingly important for military operations. Due to small wing dimensions and low air speeds, UAVs often operate within a Reynolds number flight regime for which a strong interaction exists between separation and transition. An improved understanding of the relevant physical mechanisms, especially in the context of FST, is required for an effective control.

We are employing computational fluid dynamics (CFD) for investigating the physics underlying laminar separation. This is a highly complex phenomenon as both unsteady separation and transition mechanisms are at work interactively. In particular, we are employing high-resolution direct numerical simulations (DNS) for investigating canonical separation bubbles on a flat-plate. For these simulations we accurately model FST as would be seen, for example, in a wind tunnel. Free-stream turbulence "seeds" disturbances in the flow which influence transition. Free-stream turbulence is often neglected in CFD, but is always a factor in real-world applications.

Direct numerical simulations become increasingly more expensive for large Reynolds numbers. Unsteady Reynolds-averaged Navier-Stokes (URANS) introduces too much modeling, resulting in the suppression of most unsteady fluid motion. Large-eddy simulation (LES), on the other hand, suffers from a restrictive near-wall-grid resolution requirement that makes LES computationally expensive. Hybrid RANS/LES models blend between DNS, LES, and URANS to make optimum use of the available computational resources. In particular, RANS is typically employed near walls. We carried out hybrid simulations and a reference "coarse grid" DNS (implicit large-eddy simulation, ILES) for a modified NACA 64_3-618 airfoil at a chord Reynolds number of 322,000.

The modified NACA 64_3-618 airfoil was used for the AMT 200S SuperXimango motor glider. The cruise Reynolds number of the AMT 200S based on mean aerodynamic chord is about 3 million. We have several 1:5 scale models of the airplane that are used for scaled model flight research. The chord Reynolds number for the Froude-scaled cruise conditions is 322,000. Performance losses can be expected at this low Reynolds number since the NACA 64_3-618 was designed for Reynolds numbers in the order of 3 million [Heine, et al., 2008]. Using the same airfoil, we are also carrying out wind tunnel experiments for a comparable Reynolds number range [Mack, et al., 2008; Plogmann, et al., 2009]. The overlap between simulations, free-flight experiments, and wind tunnel experiments provides synergisms and allows for a cross-validation of the various approaches.

In this paper, the computational methods are reviewed first. Next, results from direct numerical simulations that were carried out to investigate the effect of free-stream turbulence on laminar separation and separation control are discussed. Then, some results from our high Reynolds number simulations are shown. Finally, a summary and conclusion is provided.

2. Computational Method

A two-pronged computational approach is employed. An incompressible finite-difference code is employed for the simulations where the effects of free-stream turbulence for a separation bubble generated on a flat-plate are investigated. A compressible finite-volume code is employed for the simulations of the entire airfoil geometry.

The higher-order accurate incompressible finite-difference code was developed in our laboratory [Meitz and Fasel, 2000]. Derivatives in the streamwise and wall-normal directions are discretized with fourth-order-accurate compact differences. A pseudo-spectral method is employed in the spanwise direction and a fourth-order-accurate Runge-Kutta scheme is employed for time-integration. A hybrid parallelization approach, using both Message Passing Interface (MPI) and shared memory parallel programming (OpenMP) was adopted to make optimal use of available supercomputers.

For simulations of the entire airfoil geometry, a higher-order-accurate compressible finite-volume code that was also developed in our laboratory [Gross and Fasel, 2008] is employed. The Navier-Stokes equations are discretized with a ninth-order-accurate upwind scheme for the convective terms, and a fourth-order-accurate discretization for the viscous terms. The turbulence model transport equations are discretized with a second-order-accurate upwind scheme for the convective terms, and a second-order-accurate discretization for the viscous terms. An implicit second-order-accurate time-integration allows for Courant-Friedrichs-Lewy (CFL) numbers in the order of 1,000. The code was parallelized using MPI.

3. The Effect of Free-stream Turbulence

For investigating the effect of free-stream turbulence (FST) on laminar separation and separation control, a simplified flow configuration was considered: a laminar separation bubble generated on a flat-plate by use of a displacement body (Figure 1). The general setup of these simulations was chosen such that a comparison to water-tunnel experiments currently carried out at the Hydrodynamics Laboratory at the University of Arizona will become feasible in the future. Note that the NACA 64_3-618 airfoil will be used as the displacement body in the experiments. For the DNS, wall-normal blowing and suction was applied at the free-stream boundary to model the effects of the favorable-to-adverse pressure gradient associated with the displacement body used by Gaster [1966]. Length scales were made dimensionless with a reference length of 1in, and velocities were made dimensionless with a reference velocity of 6.6m/s. From a numerical stand point, this approach also has the advantage that the highly efficient incompressible code can be employed to solve the Navier-Stokes equations on a Cartesian grid within a rectangular domain. The grid is shown in Figure 1. The number of grid points in the streamwise and wall-normal direction was 1,621×256. The spanwise domain extent was λ_z=2 and was resolved using 200 collocation points (129 Fourier modes). In order to check for grid convergence of the results, selected cases were also computed on a finer grid with 207 million grid points (2,161×321×300).

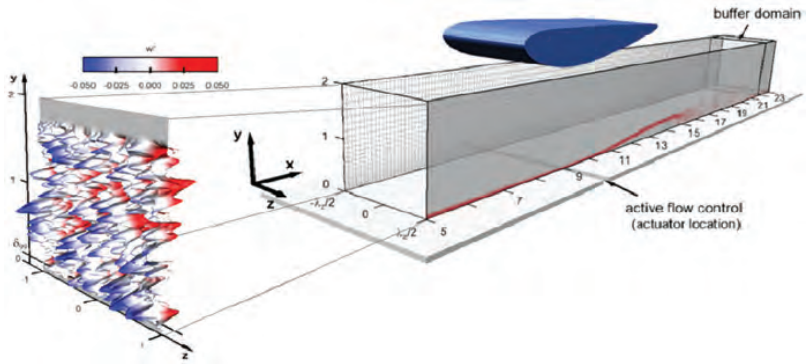


Figure 1. Simulation setup for flat-plate separation bubble simulations. The instantaneous elevated-surface plot of spanwise velocity (left) shows typical free-stream turbulence fluctuations that are introduced at the inflow boundary. Also shown is the computational grid (every 10th grid line is shown in both streamwise and wall-normal direction).

For the present study, the flow was subjected to FST, which was modeled by introducing disturbances at the inflow boundary of the domain (see Figure 1). The general method was based on an approach by Jacobs [2000]. Three different FST intensities were considered: 1) "quiet" flow with zero FST level (*Tu=0*, case *FST-0*), 2) flow with low FST level (*Tu=0.05%*, case *FST-0.05*), and 3) flow with elevated FST level (*Tu=2.5%*, case *FST-2.5*). While the FST intensity was varied, the turbulent integral length scale remained constant and was chosen as $L_{11}=5\delta_{1,0}$ where $\delta_{1,0}$ is the displacement thickness of the inflow boundary-layer profile. Note that changes of L_{11} are known to affect both the transition mechanism and the transition location. For the present study, such effects were; however, not investigated in more detail.

For flow control simulations, two-dimensional (2D) harmonic v-velocity disturbances were introduced across a narrow blowing and suction slot at the wall and upstream of separation. The forcing function can be written as $v_f\ (x,t)=v_{f,max}\ F_s(x)\ cos(2\pi f_f t)$. For the shape function, $F_s(x)$, a polynomial was employed which guarantees zero net volume flux through the disturbance slot at any time instant (Figure 2). For the forcing frequency, f_f, the natural vortex-shedding frequency of the uncontrolled separation bubble was used (case *UNC*). The maximum forcing amplitude was $v_{f,max}=0.05$, i.e., *5%* of the free-stream velocity.

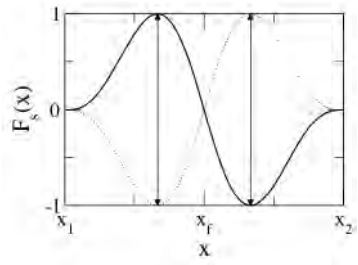


Figure 2. Exit velocity distribution for harmonic blowing and suction through a 2D slot

The response of the flow to the 2D harmonic forcing in a zero-FST environment (case *FST-0*) for the uncontrolled and controlled flow is demonstrated in Figure 3. The flow structures were identified using the λ_2 vortex criterion by Jeong & Hussain [1995]. Also shown are instantaneous, spanwise-averaged ω_z-vorticity contour lines. The visualizations indicate that the flow control very effectively reduces the size of the separation bubble. More important; however, is the observation that transition to turbulence can be significantly delayed. In fact, within the boundaries of the computational domain, the flow completely relaminarizes.

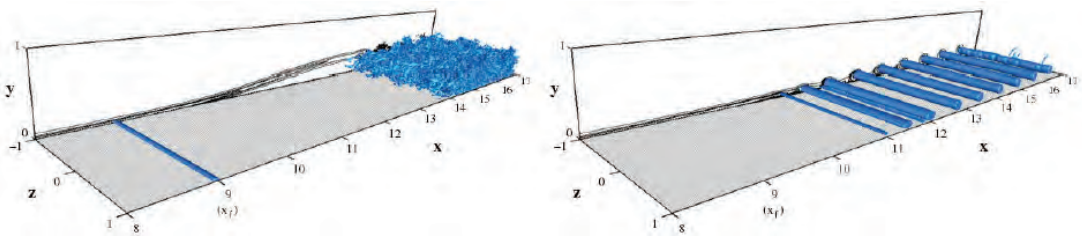


Figure 3. Instantaneous contour lines of spanwise vorticity $-20<\omega_z<20$ (sidewalls) and iso-surfaces of λ_z criterion ($\lambda_z=-12$). Left: case *UNC*; right: case *FST-0*.

Two fundamental flow control mechanisms that are responsible for the stunning effectiveness of 2D harmonic excitation were identified. The first mechanism is directly associated with the primary shear-layer instability. Inside the separated shear-layer, the periodic disturbances introduced at the 2D forcing slot were found to be exponentially amplified, leading to the formation of strong spanwise vortices. The disturbance amplification agrees very well with linear stability theory (LST), which is illustrated in Figure 4 (*left*), by plotting the downstream development of the u-velocity disturbance amplitude. Secondly, it was observed that while for the uncontrolled flow (case *UNC*) three-dimensional (3D) temporally-growing disturbances exist that ultimately lead to break-down to turbulence, for the controlled flow the development of such temporally-growing modes is suppressed. To demonstrate this, a small-amplitude pulse disturbance was introduced at $x=x_f$ for case *FST-0*. Then the downstream and temporal evolution of the ensuing wave packet was monitored. Figure 4 (*right*) demonstrates that a stationary observer located inside the separated flow region would first experience a strong disturbance increase associated with the incoming wave packet. Later in time; however, the disturbances decay exponentially with a decay rate of approximately $\sigma=-0.65$. This shows that the controlled flow is secondary absolutely stable, which explains why the flow does not transition to turbulence, even when computing for a very long time.

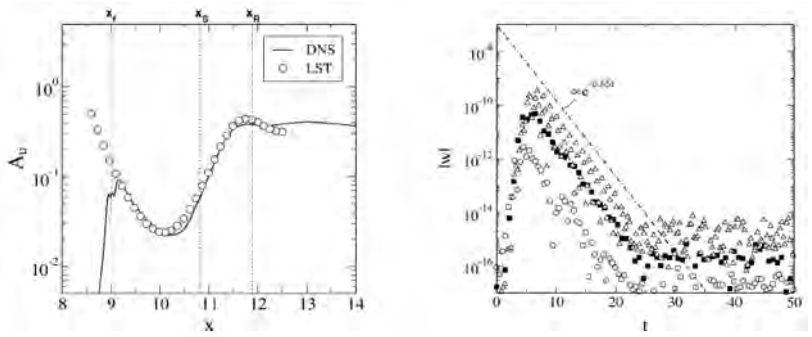


Figure 4. Left: Downstream development of the u-velocity disturbance introduced at the forcing location, x_f. Locations of separation, x_S, and reattachment, x_R, are indicated as well. Right: Temporal development of the w-velocity component associated with a small-amplitude pulse disturbance as seen by a stationary observer located at x=11.0 (○), x=11.5 (■), and x=12.0 (Δ).

Subsequently, the background disturbance levels were increased to *Tu=0.05%* and *Tu=2.5%* in order to investigate whether the stunning effectiveness of the 2D harmonic excitation and the resulting relaminarization could potentially also be observed in a more realistic flow environment. The response of the flow to the flow control under these conditions is visualized in Figure 5 for a time-instant corresponding to 15 forcing periods after the start of the 2D periodic excitation. Figure 5 shows spanwise-averaged ω_z-vorticity contour lines as well as λ_2 vortex structures, which are colored based on the local value of the streamwise vorticity component, ω_x. Note that only half of the spanwise domain extent is shown in Figure 5. From the close-up views in Figure 5, it can be seen that the 2D spanwise structures, which develop as a consequence of the harmonic forcing and the shear-layer instability, appear modulated in the spanwise direction. Note also that there seems to be a distinct spanwise wavelength of the modulation. The downstream amplification of the 3D disturbances associated with the modulation (secondary instability) ultimately initiates transition, and the flow becomes turbulent. This secondary instability was found to be convective in nature (see Balzer, 2011). These observations are in sharp contrast to case *FST-0* where transition was prevented as break-down to turbulence due to such a secondary convective instability mechanism did not occur.

By analyzing the stability characteristics of the controlled flows with non-zero FST levels it was found that even with the elevated FST levels (for case *FST-2.5*), the effectiveness of the flow control can still be attributed to the same two physical mechanisms identified earlier: 1) exploitation of the primary shear-layer instability, and 2) suppression of 3D temporally-growing disturbances (due to an absolute secondary instability) inside the separated shear-layer region. The first mechanism is again demonstrated by plotting the downstream development of the primary 2D u-velocity disturbance component (Figure 6, *left*). The second mechanism is demonstrated by plotting near-wall w-velocity signals that, observed by a stationary observer located inside the separated flow region after a low-amplitude pulse disturbance, had been introduced upstream of separation (Figure 6, *right*). The associated temporal decay rates for the 3D disturbances were approximately $\sigma=-0.3$ (case *FST-0.05*) and $\sigma=-0.1$ (case *FST-2.5*), respectively.

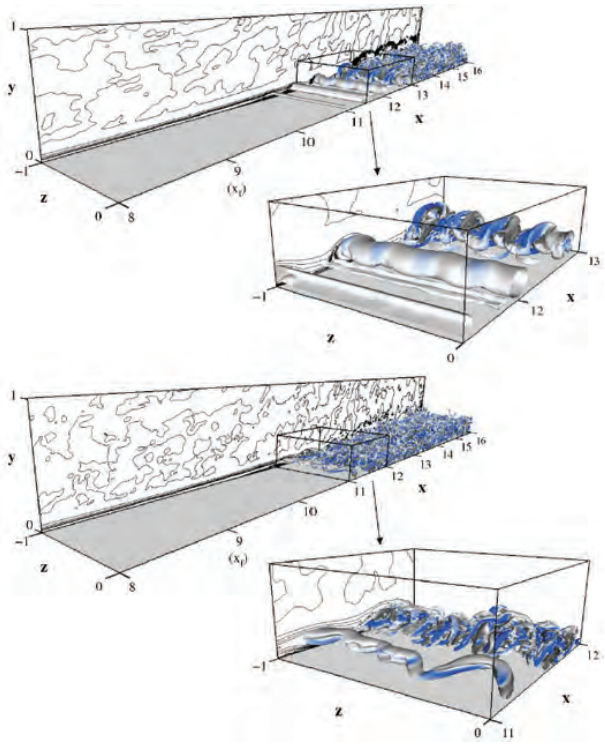


Figure 5. Instantaneous contour lines of spanwise vorticity $-20<\omega_z<20$ (sidewalls) and iso-surfaces of λ_z criterion ($\lambda_z=-12$). The color of the iso-surfaces is based on the local value of the streamwise vorticity (blue: $\omega_x>0$, gray: $\omega_x<0$). Top: case FST-0.05; bottom: case FST-2.5.

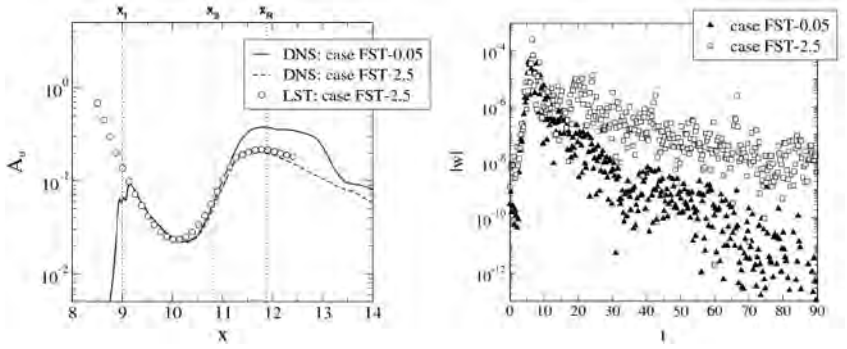


Figure 6. Results of a stability analysis for cases FST-0.05 and FST-2.5. Left: Downstream development of u-velocity disturbance introduced at the forcing location, x_f. Locations of separation, x_S, and reattachment, x_R, are indicated as well. Right: Temporal development of w-velocity component associated with a small-amplitude pulse disturbance as seen by a stationary observer located at x=12.0.

4. Simulations of Entire Airfoil

In addition to FST and surface roughness, the chord Reynolds number has a profound effect on separation and transition and the effectiveness of active flow control devices. We carried out simulations for a modified NACA 64_3-618 airfoil at a chord Reynolds number of 322,000. This is the scaled cruise Reynolds number based on mean aerodynamic chord for a 1:5 dynamically scaled model of the AMT 200S (SuperXimango) motor glider. Simulations are for an angle-of-attack of α=13.2deg. Experiments by Heine, at al. [2008] and Plogmann, et al. [2009] for the same Reynolds number showed stall onset for α>8deg. For α=13.2deg a laminar separation bubble near the leading edge with turbulent attached flow downstream and turbulent trailing edge separation was observed in the experiment. This angle-of-attack is therefore well-suited for investigating active control of turbulent separation, which is of high relevance for full-scale applications.

Considerable grid resolution is required for fully-resolved direct numerical simulations at Re=322,000. Hybrid Reynolds-averaged Navier-Stokes (RANS)/LES methods that blend between RANS and LES, according to the local grid resolution and flow properties, offer significant compute time savings. For example, near walls a model can revert to a pure RANS model which is advantageous as modern RANS models are well-suited for the computation of attached boundary layer flows. Away from the walls, a typical LES sub-grid stress model or a direct numerical simulation may be recovered. Ideally, a hybrid model would make the most efficient use of the available grid resolution. A one-equation hybrid model, based on renormalization group (RNG) theory by De Langhe, et al. [2008], was employed.

A Poisson grid generator to obtain O-grids was employed. The number of cells in the circumferential and wall-normal directions was 1,578×281 for the "coarse grid" DNS grid and 600×200 for the hybrid RANS/LES grid (Figure 7). The spanwise grid extent and the number of cells in the spanwise direction were 0.1c (where c is the chord length) and 128 for the DNS grid and 0.4c and 64 for the hybrid RANS/LES grid, respectively. A characteristics-based non-reflecting boundary condition was applied at the freestream boundary and periodicity was assumed in the spanwise direction. Both simulation strategies, DNS and hybrid RANS/LES, were employed for simulating the uncontrolled flow. We also investigated active flow control by wall-normal harmonic blowing through a spanwise slot. The slot was located at 0.5% chord and had a width of 1.4% chord. The maximum blowing amplitude was 10% of the free-stream velocity and the dimensionless frequency was 5.

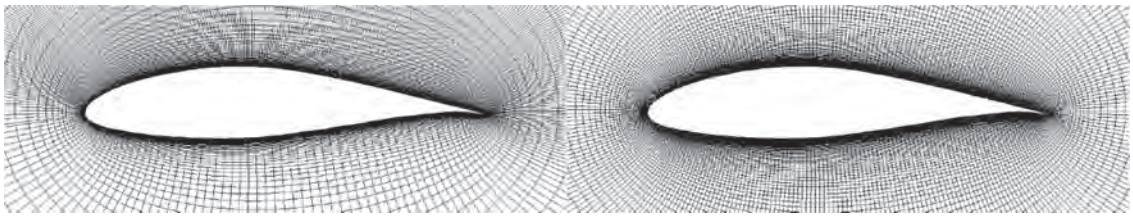


Figure 7. Computational grid for "coarse" DNS (left, every 4th line shown) and hybrid simulation (right, every 2nd line shown)

Figure 8 shows instantaneous iso-contours of the spanwise vorticity component, ωz, and the unresolved eddy viscosity, μ_{Tu}. For the uncontrolled flow, the turbulent boundary-layer separates shortly downstream of the maximum thickness location. For the controlled flow, the disturbances that are introduced at the leading edge by the actuator are large enough to result in a downstream modulation of the thickness of the turbulent boundary layer. However, the perturbations are not strong enough to facilitate a "roll-up" of the separated turbulent boundary-layer downstream of separation. Surprisingly with actuation the flow is seen to separate earlier. At this point, a convincing explanation as to why this happens is still lacking. Figure 8 also shows considerable unresolved eddy viscosity in the separated flow region. Small-scale turbulence is modeled and the structures that are convected upstream are part of the resolved turbulence.

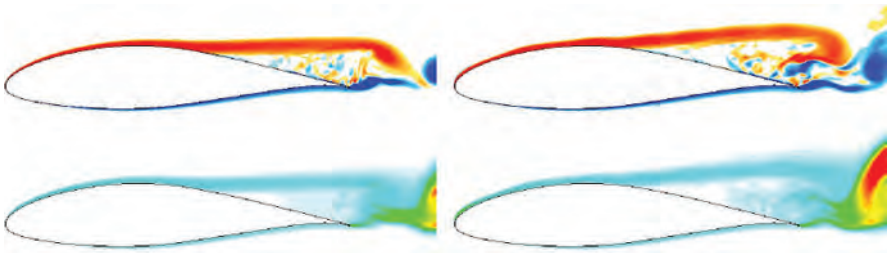


Figure 8. Hybrid RANS/LES simulations for Re=322,000. Instantaneous contour lines of spanwise vorticity, $-40<\omega_z<40$ (top) and unresolved eddy viscosity, $0<\mu_{Tu}<100$ (bottom) for uncontrolled flow (left) and controlled flow (right).

Instantaneous flow visualizations obtained from the hybrid simulations are provided in Figure 9. Iso-surfaces of the Q vortex identification criterion by Hunt [1988] indicate areas where rotation dominates strain. The boundary-layer is seen to separate from the suction surface near the maximum thickness location. Following the initial shear-layer "roll-up" downstream of the trailing edge, energy is quickly transferred into smaller scale motion. The small-scale structures in the recirculation region underneath the separated boundary-layer can be interpreted as coherent structures within a turbulent flow. The smallest dissipating scales are modeled in the hybrid RANS/LES approach. Figure 10 provides instantaneous flow visualization obtained from the DNS data. A small laminar separation bubble is generated shortly downstream of the leading edge. The separated boundary-layer transitions quickly, resulting in turbulent reattachment. The flow downstream

of the bubble is fully turbulent and shows little coherence. Turbulent separation is observed downstream of the maximum thickness location. The general flow topology is in good agreement with the experiment (Heine, et al. [2008], unpublished data).

Figure 9. Hybrid RANS/LES simulations for Re=322,000. Instantaneous iso-surfaces of vortex identification criterion, Q=1. Uncontrolled flow (left) and controlled flow (right).

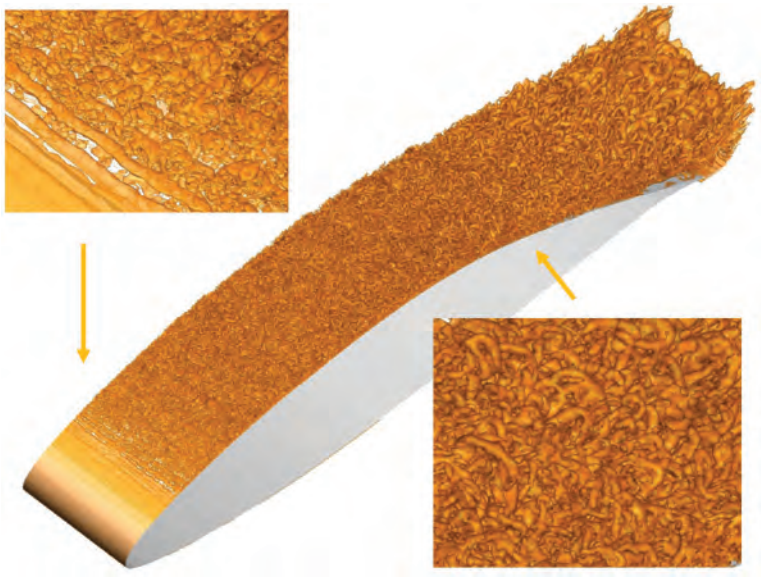


Figure 10. DNS for Re=322,000. Instantaneous iso-surfaces of vortex identification criterion, Q=100.

A comparison of the wall pressure and skin friction coefficient distributions with Xfoil predictions [Drela, 1989] and wind tunnel measurements [Mack, et al., 2008; Plogmann, et al., 2009] is shown in Figure 11. Also included are results from an earlier hybrid RANS/LES simulation with a filter-based RANS (FBR) turbulence model [Gross, et al., 2010]. The wall-pressure distributions extracted from the hybrid simulations all match the measured data fairly well. A comparison of the skin friction distributions; however, shows that all simulations, except for the DNS, fail to capture the laminar separation bubble near the leading edge. This bubble is predicted by XFoil and was also observed in the experiments. The one-equation hybrid turbulence model does not capture transition, but predicts turbulent flow starting from the leading edge which suppresses separation. For the angle-of-attack considered here, this flow scenario is in fairly good agreement with the experiment because the laminar separation bubble is located very close to the leading edge. For cases with a considerable laminar run, the transition point would have to be set (i.e., based on the e^N method) or the hybrid simulation strategy would have to be based on a more elaborate turbulence model that captures transition. When considering the lift curve and drag polar (Figure 11), it can be seen that the computed lift and drag for the previous hybrid simulation and the controlled case are in better agreement with the measurements than the results for the uncontrolled flow based on the RNG model. The DNS result was not included because the predicted lift is too high. This can likely be attributed to a poor convergence of the time average. As stated in the introduction, DNS of high-Reynolds number flows are barely affordable and good time averages are very expensive.

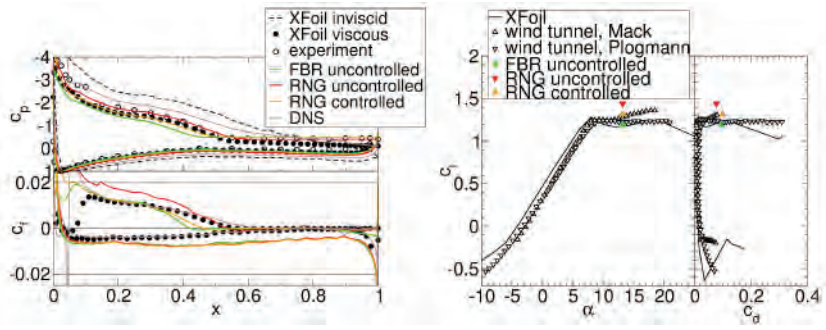


Figure 11. Wall-pressure and skin friction coefficient (left) and lift curve and drag polar (right)

5. Conclusion

High-resolution direct numerical simulations (DNS) of a canonical separation bubble on a flat-plate were carried out. Harmonic blowing and suction through a spanwise slot was shown to be very effective in eliminating laminar separation. This was attributed to shear-layer instability, which amplifies the disturbance input. Another stunning observation is that the onset of secondary instabilities and thus transition onset can be delayed for certain forcing frequencies. The separation bubble was then subjected to isotropic free-stream turbulence (FST). Although the transition delay could no longer be accomplished entirely by the primary physical mechanism, the exploitation of shear-layer instability was found to be the same and the control remained equally effective. This is an important finding, especially when considering that FST strongly affects laminar separation. Clearly, "clean" simulations where the FST intensity is zero are not always realistic when separation and transition are investigated.

"Under-resolved" DNS simulations (ILES) and hybrid RANS/LES simulations of a modified NACA 64_3-618 airfoil at Re=322,000 and α=13.2deg were also carried out. Direct numerical simulations at this Reynolds number are barely affordable and hybrid simulations are a possible alternative. A one-equation hybrid model for simulations with and without flow control was tested. The agreement with experimental data and with the reference DNS was fair, but leaves room for improvement. A considerable amount of further validation is needed to determine the limits of the hybrid RANS/LES model and its failure modes. In summary, both the inclusion of free-stream turbulence and the development of hybrid RANS/LES models allows for simulations of flows that are closer to practical applications.

6. Significance to DoD

The role of Unmanned Aerial Vehicles (UAVs) in military operations has become crucial in the combined effort to unman the front lines. Small UAV dimensions or high flight altitudes for larger flight vehicles result in low operating Reynolds numbers, which can lead to laminar separation. Laminar separation reduces the usable lift and increases the drag. Therefore, laminar separation negatively affects performance (e.g., range, loitering time, payload). In worst case scenarios, separation can lead to complete stall and loss of the vehicle. The necessary tools for investigating separation control strategies for lifting surfaces of small and large UAVs, as well as manned aircraft, with the goal of greatly improving aerodynamic performance are being developed. How "real-world effects" such as free-stream turbulence and surface roughness are affecting the operation of such devices is also being investigated.

Acknowledgments

This work was supported by the US Air Force Office of Scientific Research (AFOSR) under grant number FA9550-09-1-0214, with Dr. Douglas Smith serving as Program Manager. Compute time provided by Challenge Project AFOSR13312C4A is highly appreciated.

References

Balzer, W., "Numerical Investigation of the role of free-stream turbulence on boundary-layer separation and separation control," *Ph.D. thesis*, The University of Arizona, 2011.

De Langhe, C., Bigda, J., Lodefier, K., and Dick, E., "One-equation RG hybrid RANS/LES computation of a turbulent impinging jet," *Journal of Turbulence*, Vol. 9, No. 16, pp. 1–19, 2008.

Drela, M., "XFoil: An Analysis and Design System for Low Reynolds number airfoils," *Lecture Notes in Engineering, Proceedings of the Conference Notre Dame*, IN, Vol. 54, 1989.

Gaster, M., "The structure and behavior of laminar separation bubbles," *AGARD CP 4*, pp. 813–854, 1966.

Gross, A., and Fasel, H., "High-Order Accurate Numerical Method for Complex Flows," *AIAA J.*, Vol. 46, No. 1, pp. 204–214, 2008.

Gross, A., and Fasel, H.F., "Numerical Investigation of Separation for Airfoils at Low Reynolds Numbers," *AIAA*, 4736, 2010.

Heine, B., Mack, S., Kurz, A., Gross, A., and Fasel, H.F., "Aerodynamic Scaling of General Aviation Airfoil for Low Reynolds Number Application," *AIAA*, 4410, 2008.

Hunt, J.C.R., Wray, A.A., and Moin, P., "Eddies, stream, and convergence zones in turbulent flows," *Report CTRS88*, Center For Turbulence Research, Stanford, CA, 1988.

Jacobs, R.G., "Bypass Transition Phenomena studied by Computer Simulation," *Ph.D. thesis*, Stanford University, 2000.

Jeong, J. and Hussain, F., "On the Identification of a Vortex," *J. Fluid Mech.*, Vol. 285, pp. 69–94, 1995.

Mack, S., Brehm, C., Heine, B., Kurz, A., and Fasel, H., "Experimental Investigation of Separation and Separation Control on a Laminar Airfoil," *AIAA*, 3766, 2008.

Meitz, H.L. and Fasel, H., "A Compact-Difference Scheme for the Navier-Stokes Equations in Vorticity-Velocity Formulation," *J. Comp. Phys.*, Vol. 57, No. 1, pp. 371–403, 2000.

Plogmann, B., Mack, S., and Fasel, H.F., "Experimental Investigation of Open- and Closed-Loop Control for Airfoil under Low Reynolds Number Conditions," *AIAA*, 4282, 2009.

Numerical Simulation of Transition in Hypersonic Boundary Layers

Jayahar Sivasubramanian, Andreas Laible, and Hermann Fasel
Department of Aerospace and Mechanical Engineering, University of Arizona, Tucson, AZ
{jayahar, alaible, faselh}@email.arizona.edu

Abstract

Boundary-layer stability and transition are studied for three different cone geometries at Mach 6 using Direct Numerical Simulations (DNS). The three geometries are: i) the 7° straight-cone investigated in the Boeing/Air Force Office of Scientific Research Mach 6 Quiet Tunnel at Purdue University, and ii) the 5° flared-cone investigated in the NASA Langley Test Chamber Facility and (iii) its non-flared counterpart. For the 7° straight-cone, a "natural" transition scenario was modeled and investigated using wave packets. The main objective of this research is to determine which nonlinear mechanisms are dominant in a broad-band disturbance environment and then to perform controlled transition simulations of these mechanisms. To this end, fundamental resonance was identified as a dominant nonlinear mechanism and highly-resolved controlled fundamental breakdown simulations were performed. We then performed a comparison of the base flow, i.e., the undisturbed laminar flow, and the linear stability regime for the 5° flared-cone and its non-flared counterpart. As for the straight-cone, by considering the most amplified axi-symmetric wave as primary wave, a parameter study regarding the secondary instability of oblique waves with the fundamental frequency was conducted for the flared-cone. The case that exhibits the highest secondary growth rate is then investigated in a highly-resolved DNS. The present study can be seen as a step towards answering the question to what extent the transition process is altered by the cone flare.

1. Introduction

Laminar-Turbulent transition in hypersonic boundary layers is a major unresolved topic in fluid dynamics. Even after many years of research, crucial aspects of the transition physics are still in the dark. The lack of a basic understanding of high-speed transition, and as a consequence a lack of reliable transition prediction methods, hinders practical applications such as reusable launch vehicles, high-speed interceptors, and hypersonic cruise vehicles (Lin, et al., 1984). The increased heat loads (caused by turbulent mixing) on the structure of the flight vehicles represent the main difficulties in the design and safe operation of high-speed vehicles. Appropriate measures to guard against the heat transfer due to aero-thermal loads are expensive, and/or result in significant weight penalties. Accurate estimation of the transition location is of vital importance because only then can the aero-thermal loads and surface temperatures be adequately predicted. In addition to surface heating, transition to turbulence also has a significant effect on the aerodynamic performance of high-speed flight vehicles, as the skin friction for turbulent boundary layers is considerably higher than for the laminar boundary-layer.

Hypersonic transition research suffers from the fact that reliable experiments are very difficult and expensive. Wind tunnel noise level, short run-times and controlled disturbance generation are only a few of the difficulties faced in experiments. Therefore, relatively few reliable experimental studies of the growth of instability-waves have been conducted. On the other hand, Direct Numerical Simulations (DNS) of transition in high-speed boundary-layers require high grid resolution and large computation time. As a consequence, very little is known about the late stages of transition and in particular, about the final breakdown to turbulence. However, it is vital to investigate the final breakdown in order to identify which disturbance scenarios lead to turbulent flow and which do not. Therefore, accurate and reliable fully-resolved DNS of high-speed boundary-layers are essential for advancing our understanding of high-speed transition and for development of reliable transition prediction models.

In addition, the physics of high-speed boundary-layer transition are much more complex than for low speeds. From linear stability theory (Mack, 1969), it is known that multiple instability modes exist for high-speed boundary-layer flows, in contrast to only one mode (Tollmien-Schlichting [TS]) for the incompressible case. In addition to the so-called first mode in supersonic boundary-layers, which is equivalent to the TS-mode in incompressible boundary layers, higher modes exist for supersonic boundary layers that result from an inviscid instability mechanism. According to Linear Stability Theory (LST), the most unstable higher modes are two-dimensional unlike oblique first modes. Also from linear stability theory,

it is known that the first mode is dominant (higher amplification rates) for low supersonic Mach numbers while for Mach numbers above 4 the second mode is dominant (most amplified). Due to the difficulties in carrying out experiments (and "controlled" experiments, in particular) and due to the existence of multiple instability modes, the role and importance of the various instability modes in a realistic transition process are not understood at all. Of course, when amplitudes of the various instability modes reach high enough levels, nonlinear interactions of these modes can occur. As a consequence, the transition process in high-speed boundary-layers is highly non-unique, which means that slight changes in the environment or vehicle geometry may significantly alter the transition process.

Cones with circular cross-sections represent a useful prototypical geometry for investigating boundary-layer stability and transition at hypersonic speed. In the present work, we are employing DNS to study the laminar-turbulent transition process for three conical geometries at Mach 6. The first geometry is a 7° circular straight cone tested in the Boeing/Air Force Office of Scientific Research (AFOSR) Mach 6 Quiet Tunnel at Purdue University (Schneider, 2009). The two main goals for the straight cone investigations are to determine which nonlinear mechanisms are dominant in a broad-band, "natural" disturbance environment and then to perform controlled transition simulations of these mechanisms. Towards this end, several highly-resolved DNS were performed. In these simulations, in order to model a "natural" transition scenario, the boundary-layer is forced by a short duration (localized) blowing and suction pulse. The pulse disturbance will generate a wave packet, which contains a wide-range of disturbance frequencies and wave numbers.

The other two geometries are the flared cone investigated by Lachowicz, et al. (1996) in the NASA Langley Nozzle Test Chamber Facility and its non-flared (or straight) counterpart. Although originally researchers considered the flared-cone geometry for investigating the effects of adverse pressure gradient, the flared-cone became recently popular for stability and transition experiments in hypersonic quiet wind tunnels, like the Boeing/AFOSR Mach 6 Quiet Tunnel (BAM6QT) at Purdue University (Schneider, et al., 2003), for a different reason. In hypersonic quiet tunnels, the background disturbances are at such a low level that natural (unforced) transition will not lead to a turbulent boundary-layer on the straight- (non-flared) cone geometry, which is typically limited in length and base diameter by the dimensions of the test section. For a given length and base diameter, the flared-cone provides a more unstable flow (i.e., earlier transition onset) when compared to its "non-flared" counterpart. The question is left open whether significantly changing the geometry can provide meaningful data for the straight-cone geometry or if the cone-flare ultimately leads to a different route to turbulence. The simulations presented in this paper can be seen as a step towards addressing this issue.

2. Computational Method

For our simulations we apply a two-step computational approach as shown in Figure 1. In step 1, we simulate the entire cone geometry including the shock wave emanating from the nose-tip of the cone. For this task, we have available an in-house-developed research code based on the compressible Navier-Stokes equations (Gross and Fasel, 2010). This finite volume code has the option between several different numerical schemes. Among these is a second-order TVD scheme which was used to obtain the basic state, i.e., the undisturbed laminar flow.

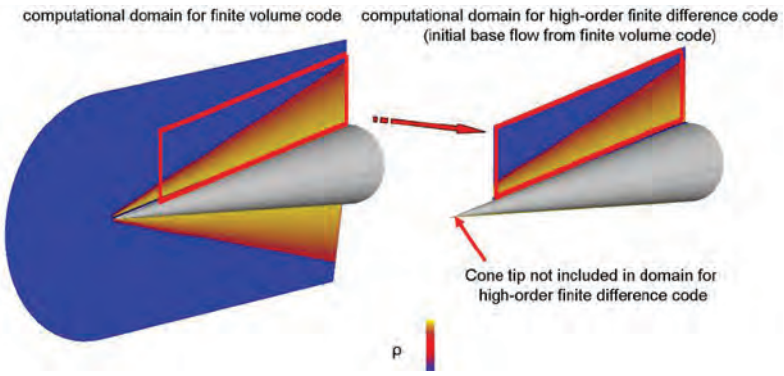


Figure 1. Schematic of simulation strategy used for our stability and transition investigations. Axi-symmetric flowfield (showing isocontours of density) over circular cone calculated by the finite volume code (left) and computational domain for the high-order-accurate stability and transition simulations (right).

In step 2, we use the flowfield obtained from the finite volume code as initial and boundary conditions for the high-order-accurate stability and transition simulations. Hereby, the computational domain extends only over a small part of the cone (see Figure 1) in order to channel all available computational resources into the region of interest. For these

DNS of transitional and turbulent high-speed flows, we employ a state-of-the-art in-house-developed high-order-accurate Navier-Stokes code (Laible, et al., 2008, 2009). The compressible Navier-Stokes equations are solved on either a conical coordinate system for (non-flared) cone geometries or on a cylindrical coordinate system with generalized coordinates in the streamwise and wall-normal directions for flared-cone geometries. Although non-flared cones can also be computed using the latter coordinate system, the conical coordinate system was implemented in order to eliminate unnecessary terms incorporated in the general transformation, and hence use the resources provided by the Department of Defense (DoD) High Performance Computing Modernization Program (HPCMP) most efficiently. The fluid is treated as an ideal gas and the viscosity is calculated using Sutherland's law with a correction for low-temperatures (Sivasubramanian and Fasel, 2011). For the time integration an explicit, low-storage, fourth-order Runge-Kutta scheme is adopted. The spatial discretization is based on high-order grid-centered upwind differences for the convective terms (Zhong, 1998) and high-order central differences for the viscous terms. The convective fluxes are split into upwind and downwind fluxes using the method proposed by van Leer (1982).

Using the Message Passing Interface (MPI), the finite difference code was parallelized for shared and distributed memory machines. The domain is decomposed in both downstream and wall-normal directions, such that the interface area, and hence the communication time, is minimized. The domain decomposition in the third-dimension would be more involved because of the pseudo-spectral discretization using Fourier modes. A schematic of the multi-processor parallelization is shown Figure 2 (left). Each sub-domain consists of internal points, which are advanced in time (dark shaded area), and ghost points, which are exchanged between the sub-domains after each time-step/sub-step (light shaded area). The ghost points allow the use of grid-centered finite difference stencils all the way to the boundaries of the sub-domains. The ghost point flow quantities are computed in the neighboring sub-domains. At the end of each Runge-Kutta sub-step, the ghost points are updated (first in the x-direction and then in the y-direction). The efficiency of the implemented parallelization algorithm for distributed memory machines has been tested, and it turned out to be high enough to justify the use of at least 512 processors and possibly more. In order to evaluate the performance of the parallel algorithm, we define the speed-up and the parallel efficiency, as $Sp=t(1)/t(n)$ and $\eta=Sp/n$, respectively, where t(n) is the elapsed time of the computation using n processors. The performance of the code on the Cray XT4 (US Army Engineer Research and Development Center's [ERDC] Jade) architecture is evaluated by computing 100 time-steps of two- and three-dimensional transition simulation at Mach 3.5. Tested are three different cases: a) two-dimensional (2D) simulation with 1,200×400 points, b) three-dimensional (3D) simulation with 1,200×400×129 grid points, and c) 3D simulation with 1,200×400×257 grid points. The resolutions of these cases are in the range of typical stability and transition simulations. The memory requirements for these cases are such that more than the tested number of processors is needed to fit the memory completely into the cache.

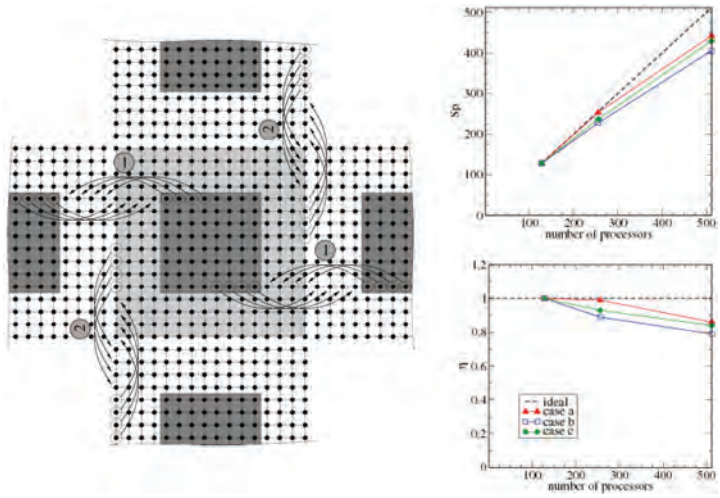


Figure 2. Schematic of ghost point exchange after each Runge-Kutta sub-step (left) and parallel speed-up, Sp, and parallel efficiency, η, for typical cone boundary-layer transition simulations on the Cray XT4 (Jade). Top-right: speed-up comparison between cases (a-c). Bottom-right: parallel efficiency comparison between cases (a-c).

The results of these test cases are shown in Figure 2 (right). Due to the memory constraints, not all cases could be run on a small number of processors. Therefore, the data points start at 128 processors. Figure 2 (right) shows excellent speed-up and parallel efficiency of all three test cases for up to 512 processors. Bottlenecks, which may limit the speed-up for

a large number of processors (≥1,024), are: i) the communication between processors, and ii) the overhead due to a small number of operations performed on the ghost points. The degree to which these bottlenecks affect the parallel efficiency is related to the ratio between the grid points and the ghost points of each sub-domain. Therefore, it can be expected that for larger number of grid points the speed-up and efficiency will be even better.

3. 7° Straight-Cone with Circular Cross-Section

Stability and transition of a hypersonic boundary-layer on a 7° sharp circular-cone at zero angles-of-attack are investigated using DNS. The computational setup follows the experiments in the Boeing/AFOSR Mach 6 Quiet Tunnel at Purdue University (Schneider, 2009). The cone model used in the Purdue experiments has a semi-vertex angle of 7°, and a cone length of L^*=0.517m. The nose radius of the cone is 0.05mm; and therefore, the cone can be considered a "sharp-cone". The approach flow has a Mach number of 6 and a unit Reynolds number of $11 \cdot 10^6 m^{-1}$. The stagnation temperature and pressure are 433K and 1,000KPa, respectively. The wall is set to be isothermal with 300°K.

3.1 Transition Initiated by a Wave Packet

The objective of these simulations is to investigate transition initiated by a broad disturbance spectrum in an attempt to model a natural transition scenario. A schematic of the cone model with the computational domain and the corresponding coordinate system is shown in Figure 3. Disturbances are introduced in a circular area on the cone surface by pulsing the wall-normal velocity using the streamwise and spanwise distributions as shown in Figure 4a. The time signal of the pulse is plotted in Figure 4b.

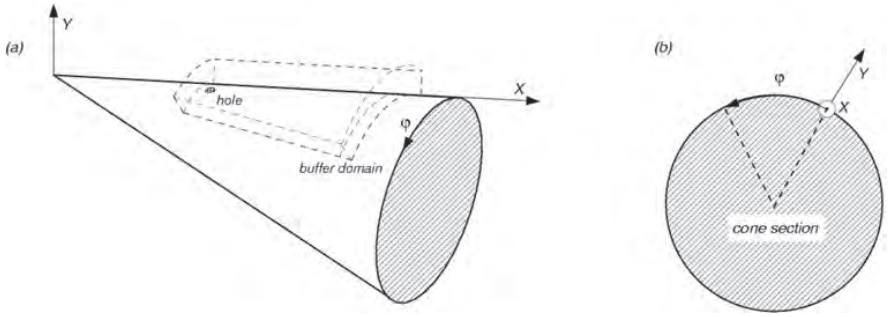


Figure 3. Cone model (a) with the computational domain and (b) cross-section

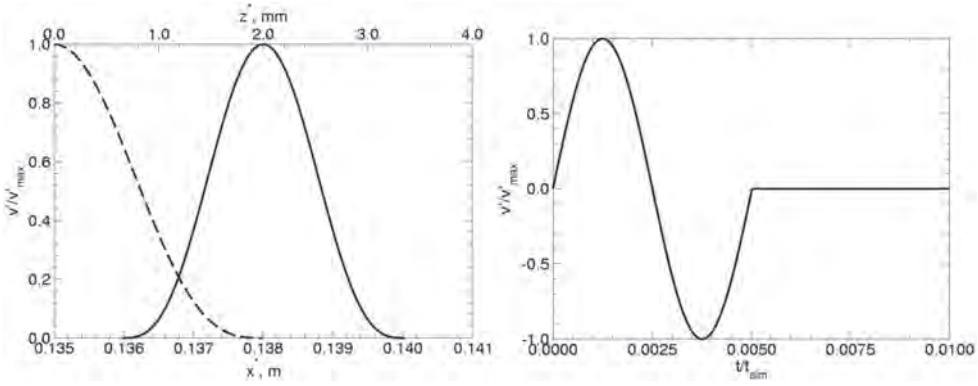


Figure 4. Wall-normal velocity at the wall (a) as a function of streamwise and spanwise direction (left) and (b) as a function of time (right)

We performed a low-amplitude pulse simulation to study the linear transition regime, and another simulation with a medium-amplitude pulse was performed to study the weakly nonlinear regime. When a short duration (localized) three-dimensional pulse disturbance is introduced in a boundary-layer, it develops into a three-dimensional wave packet, which propagates downstream. Snapshots of the wave packets generated by a low-amplitude pulse (0.001% of the freestream velocity) and a medium-amplitude pulse (0.5% of the freestream velocity) are compared in Figure 5. The dominant waves within the low-amplitude wave packet in Figure 5a are identified as the axi-symmetric second-mode disturbance waves.

This is expected, because according to linear stability theory, at this Mach number axi-symmetric waves experience higher streamwise amplification rates than oblique waves. In addition to the axi-symmetric waves, oblique waves can also be observed on the lateral sides of the packet. However, the amplitude of these oblique waves is small compared to the dominant axi-symmetric waves. These oblique waves are most likely first-mode waves which have lower frequencies than the second-mode waves. The medium-amplitude wave packet shown in Figure 5b appears to be altered by nonlinear effects. The axi-symmetric waves at the tail of the wave packet are strongly modulated in the azimuthal direction, and it appears as if the wave packet has three tails. This may be caused by nonlinear interaction of the axi-symmetric waves with oblique waves. In addition to that, changes can also be observed in the center of the weakly-nonlinear wave packet.

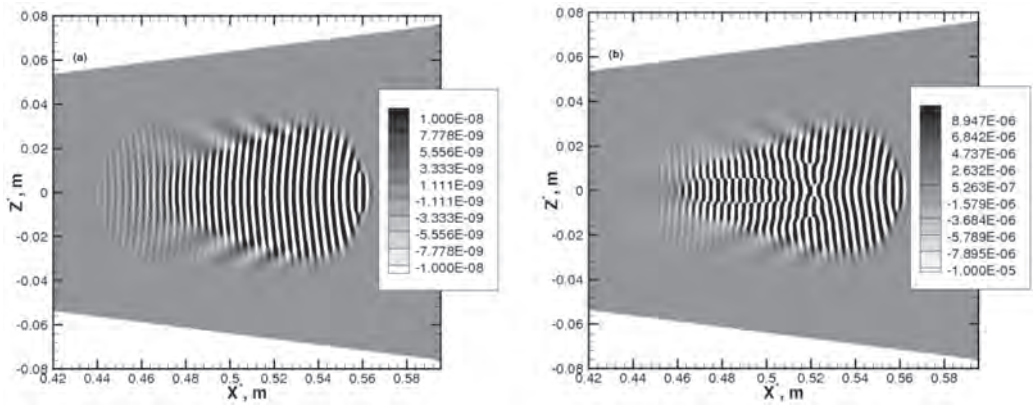


Figure 5. Snapshots of the (a) low-amplitude and (b) medium-amplitude wave packets. Shown are contours of wall-pressure disturbance on the unrolled cone surface.

The disturbance spectra obtained from Fourier transformation of the wall-pressure disturbance using 500 Fourier modes in time are shown in Figures 7 and 8 for the low- and medium-amplitude wave packet, respectively. The broad disturbance spectrum produced by the 3D pulse is shown in Figure 6. A strong peak can be seen in the spectrum, which corresponds to the time-period of pulse excitation. The streamwise development of the disturbance spectrum is shown in Figure 7 for the low-amplitude wave packet. As expected, the spectra exhibit a maximum at the azimuthal mode number k_c=0. The spectrum broadens in downstream direction, and a lower-frequency first-mode oblique disturbance with a peak at approximately k_c±30 also becomes visible. Overall, the peak amplitude remains fixed at k_c=0, but shifts to lower frequencies in the downstream direction.

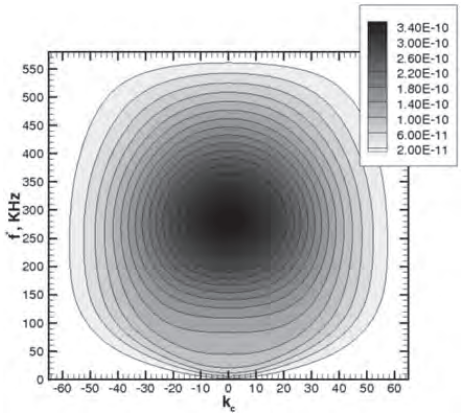


Figure 6. Disturbance spectrum in the azimuthal mode number – frequency plane at the forcing location (x*=0.138m) for the low-amplitude pulse simulation

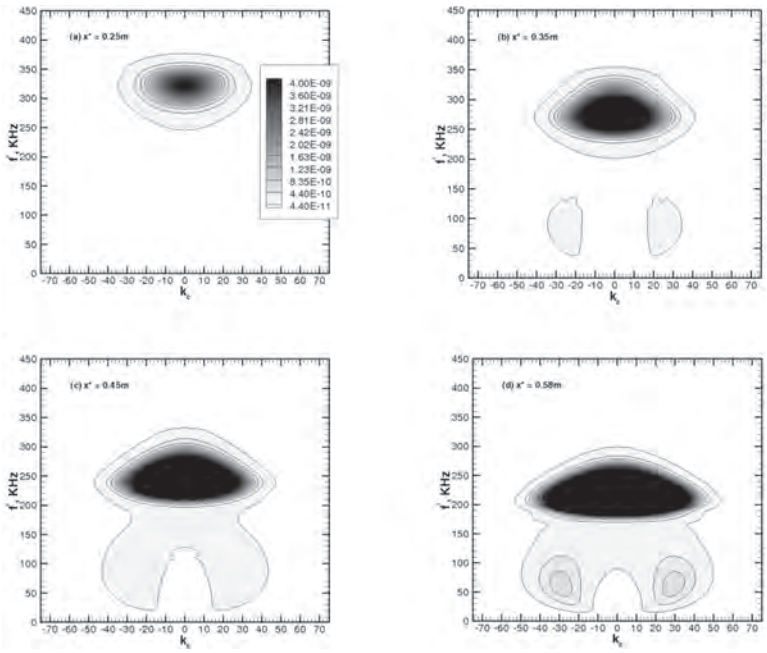


Figure 7. Disturbance spectrum in the azimuthal mode number – frequency plane for the simulation with low forcing amplitude obtained from wall-pressure disturbance at different streamwise positions. All the plots have the same contour levels.

In Figure 8, the wall-pressure disturbance spectra are shown for several streamwise positions for the medium-amplitude wave packet. The spectra obtained for x*=0.25m (Figure 8a) look very similar to those of the low-amplitude wave packet (Figure 7a), which confirms that initially the wave packet develops linearly. The disturbance spectrum for x*=0.35m indicates weak nonlinear interactions as the spectrum starts to broaden with respect to the azimuthal mode number range. For x*=0.45m the spectrum has spread further over higher azimuthal mode numbers, and develops additional frequency bands. This is an indication of nonlinear interactions between dominant axi-symmetric waves and oblique waves. In addition, lower-frequency first-mode oblique waves could also be identified along with other lower-frequency waves, which may have been generated nonlinearly. The spectrum for x*=0.58m in Figure 8d shows strong peaks at the fundamental frequency f^*≈200KHz for higher azimuthal wave numbers k_c≈30–150. This development may be an indication of a fundamental resonance mechanism. Higher-harmonics f^*≈400KHz of the high-amplitude frequency band f^*≈200KHz are also visible in Figure 8d. In addition, secondary-peaks are observed in Figure 8d at approximately half the frequency of the high-amplitude frequency band for azimuthal mode number k_c≈70, which would be an indication of a sub-harmonic resonance. However, the spectrum in Figure 8d indicates that fundamental resonance is much stronger than sub-harmonic resonance.

3.2 Fundamental Breakdown

The wave packet simulations have provided strong evidence of possible presence of fundamental and sub-harmonic resonance mechanisms. There are indications that fundamental resonance is much stronger than sub-harmonic resonance. Fundamental resonance or (K or Klebanoff type) breakdown is a breakdown initiated by a secondary instability mechanism governed by an axi-symmetric primary wave (mode (1,0)) that nonlinearly interacts with a symmetric pair of oblique waves (mode (1,±1)) at the same frequency and a steady vortex (0,1). The wave-angle of the secondary oblique wave-pair will strongly influence the strength of fundamental resonance. In the case of cone geometry this issue is even more complicated as the wave-angle of a disturbance wave changes in the downstream direction. Therefore, we performed a parameter study to determine the strongly-resonating oblique wave-pair.

A series of low-resolution simulations were performed for each azimuthal mode number ranging from 10 to 250. In these simulations, the axi-symmetric primary wave was forced at a high-amplitude and the secondary-oblique waves were forced at a very low-amplitude. It was determined that azimuthal wave number k_c=150 experienced the highest growth due to secondary-instability. Therefore a high-resolution fundamental breakdown simulation was performed with k_c=150 as the secondary-oblique wave. Instantaneous iso-surfaces of the Q-vortex identification criterion are shown in Figure 9. In these plots, flow structures that are typical for K-type breakdown were identified. The top view in Figure 9 shows the

development of L shaped vortices. These L vortices appear in an aligned pattern due to the same frequency of the primary axi-symmetric wave and the secondary-oblique wave. Downstream of this location, hairpin vortices could be seen on the tip of the L vortices. Eventually these structures will break down to small-scale structures as the flow starts to become turbulent.

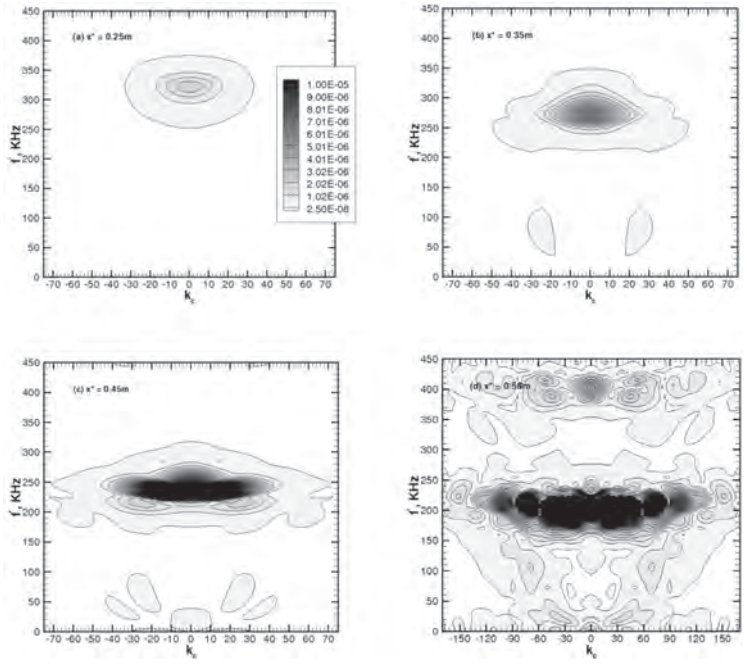


Figure 8. Disturbance spectrum in the azimuthal mode number – frequency plane for the simulation with medium forcing amplitude obtained from wall-pressure disturbance at different streamwise positions. All the plots have the same contour levels.

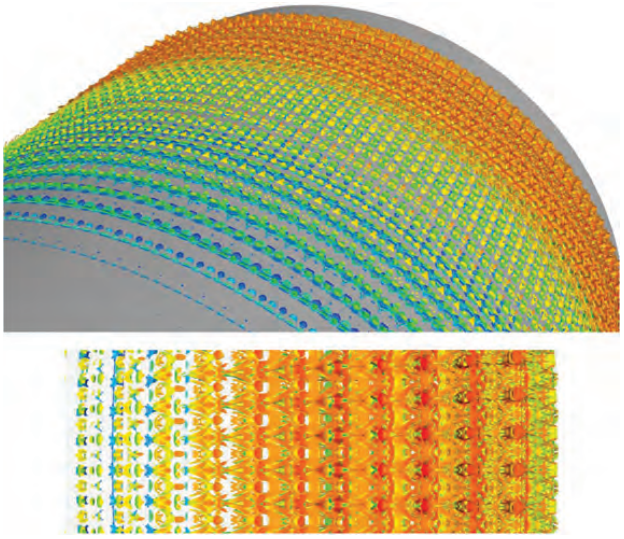


Figure 9. Fundamental breakdown with f*=210KHz and k_c=150. Iso-surfaces of the Q-vortex identification criterion (Q=10,000) colored by streamwise velocity. Top: perspective view, Bottom: close-up of the strongly nonlinear regime.

4. 5° Flared-Cone with Circular Cross-Section

The geometric dimensions of the cone model used in the experiments by Lachowicz (1996) are shown in Figure 10. The cone is divided into two equally long parts (each part: L=0.254m). The first part is a sharp (non-flared) cone with an opening half angle of 5°. The second part is flared with a flare-radius of r_{flare}=2.364m. The 5° non-flared cone is identical

to the flared-cone up to x*=0.254m, but continues without cone flare till x*=0.508m. The flow conditions of the NASA Langley Nozzle Test Chamber Facility are: the Mach number, M=6, the unit Reynolds number, $Re/L=9.2529\cdot10^{-6}m^{-1}$, and the free-stream temperature, T_{fs}=56.351K. Since the wind tunnel is a blow-down facility, the temperature boundary condition at the wall is set to be adiabatic for the basic state. For the simulation of disturbances; however, it is assumed that the heat conductivity of the cone model is sufficiently high, so that the cone surface temperature is not affected by high-frequency waves, and thus remains at its laminar value. Due to the lack of a better boundary condition, this condition is also prescribed for simulations deep into the transitional regime.

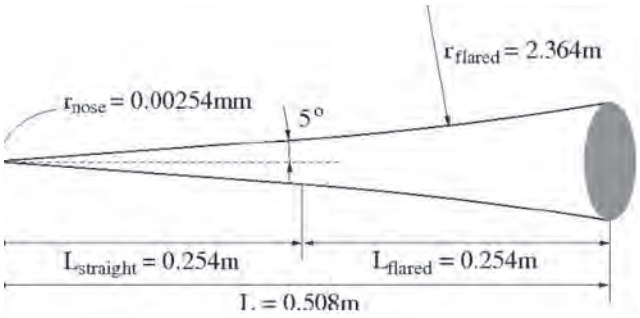


Figure 10. Flared-cone model used for the NASA experiments. Note that for this cone the first half is straight (non-flared), and only the second half is flared with a radius of r_{flared}=2.364m

Flow visualizations of the base flow, i.e., the laminar basic state without disturbances, showing iso-contours of pressure and streamlines for both cone geometries are displayed in Figure 11. The pressure contour levels indicate that the flared-cone incorporates a region of adverse pressure gradient. In particular, the pressure remains constant over the first part of the cone, but begins to rise at the streamwise position at which the flared region starts (x*=0.254m). Because exponentially growing disturbances initiate transition and may continue to be an important factor in the nonlinear stages, it is of vital importance to have a detailed knowledge of the linear regime for both geometries. Only with this knowledge, nonlinear transition simulations can be set up in a realistic and meaningful way. Therefore, the linear stability analysis poses a necessary prerequisite for the subsequently conducted transition simulations. In addition to the travelling wave instability typical for boundary-layer flows, the flared-cone is subject to another mode of instability caused by the streamwise curvature, i.e., the Görtler instability (Görtler, 1940). The coexistence of these two modes of instability, namely traveling waves and Görtler vortices, may considerably complicate the transition process. However, according to the stability analysis of Li, et al. (2010) the Görtler instability experiences significantly less growth over the entire domain than the travelling waves. Although possible interactions in the nonlinear transition zone between both modes cannot be ruled out, we focus on the travelling wave instability for the present paper.

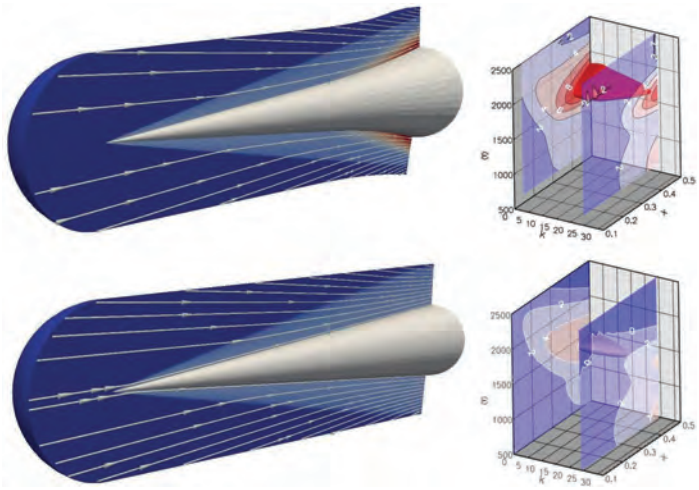


Figure 11. Iso-contours of pressure and streamlines (left) and 3D N-factor diagram in the frequency (ω), downstream direction (x), and azimuthal wave number (k) space (right) for NASA's flared-cone (top) and NASA's non-flared cone (bottom).

Several small-amplitude pulse/wave packet simulations were performed for both cone geometries. By appropriate post-processing, the relevant data regarding the linear regime was extracted from these simulations. Figure 1 1(top-right) shows the three-dimensional N-factor domain in the x-ω-k_c (downstream distance – frequency – azimuthal wave number) space for NASA's flared-cone. As expected for a boundary-layer with an edge Mach number larger than 4.5, the most amplified waves are axi-symmetric waves. For the flared-cone these waves reach N-factors higher than N=14. For waves with higher azimuthal wave numbers, the maximum N-factor that is reached at the end of the domain is significantly lower. This can be clearly seen by following the iso-surface of N=10 in Figure 11 (top-right) from k_c=0 to higher azimuthal wave numbers. In order to quantify the destabilizing effect of the cone flare, the N-factor domain for NASA's non-flared cone is also shown in Figure 11 (bottom-left). Two important observations can be made. First, the maximum N-factors at the downstream end of the domain are significantly lower for the non-flared cone; and second, the frequency (ω) of the most amplified waves stays approximately constant for the flared-cone, whereas the non-flared cone exhibits the typical shift to lower-frequencies with increasing downstream location. Because NASA's non-flared cone and NASA's flared-cone are identical for x<0.254, there are no differences between the diagrams in this region. A direct comparison of the linear stability regime between both geometries is shown in Figure 12, where the N-factor diagrams for the azimuthal wave number k_c=0 are overlaid.

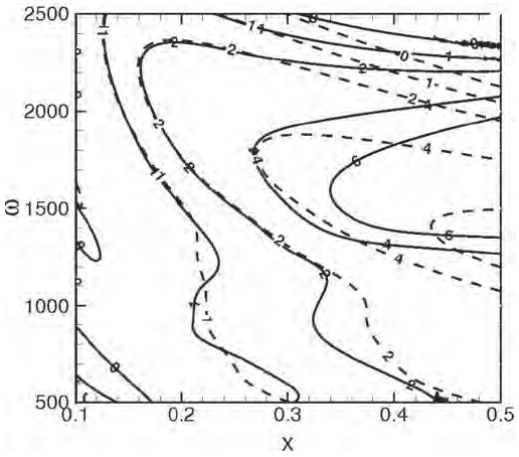


Figure 12. Comparison of iso-contours of constant N-factor for axi-symmetric waves (k_c=0) between NASA's flared-cone (—) and NASA's non-flared cone (- - -).

For NASA's flared-cone, Pruett and Chang (1998) performed a transition simulation with a set-up similar to a fundamental breakdown (K-type). They initiated transition by forcing an axi-symmetric wave (f^*=230kHz) and a pair of oblique waves (f^*=230kHz, k_c=15) with the same amplitude. Because the axi-symmetric wave is more amplified than the oblique waves, it has higher amplitude at transition onset; and thus the transition process can be considered to be of K-type. However, even relatively far into the transition zone the dominant flow structures are still two-dimensional, suggesting that transition progresses rather slow. In order to explore if there is a faster route to transition, i.e., a route with a shorter transition zone length, we performed the parameter study of secondary- (fundamental) instability plotted in Figure 13. For each azimuthal wave number we performed a simulation, where the axi-symmetric primary-wave was initiated at a high-amplitude level, and the secondary pair of oblique waves was initiated at a significantly-lower amplitude level. Once the primary wave reaches a certain threshold amplitude, the low-amplitude oblique wave experiences a growth due to secondary instability. Although the shallow pair of oblique waves for the case with wave number k_c=15 (Pruett and Chang, 1998) has a high growth-rate in the linear regime (blue line in Figure 13), the secondary growth-rate (black line in Figure 13) is rather low. Figure 13 suggests that a secondary-wave with an azimuthal wave number around k_c=48 exhibits the maximum amplification. Therefore, this case was chosen for a higher-resolved simulation. The set-up of this simulation can be considered to be a "classical" K-type breakdown, i.e., transition in initialized by a high-amplitude axi-symmetric wave and a small-amplitude oblique wave. The axi-symmetric wave is prescribed at the inflow with high-amplitude; whereas the oblique wave with azimuthal wave number k_c=48 is forced at low-amplitude. For this case, instantaneous iso-surfaces of the Q-vortex identification criterion are rendered in Figure 14. This flow visualization indicates the azimuthal modulation of the initially axi-symmetric wave and the subsequently emerging lambda structures typical for the K-type breakdown. In summary, our simulations suggest that the K-type breakdown may be a viable path to turbulence for NASA's flared-cone.

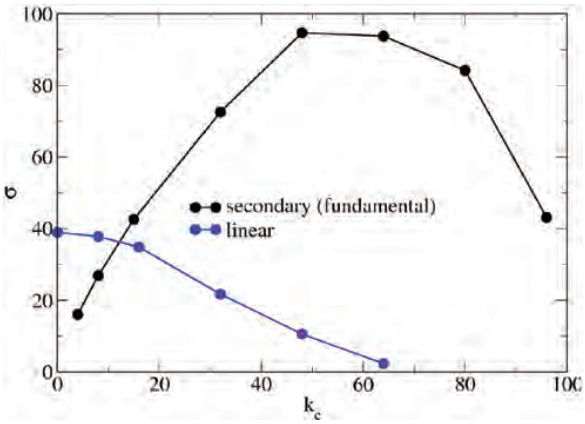


Figure 13. Growth rate, σ, of the secondary-wave, versus azimuthal wave number,k_c, at x*=0.47m of NASA's flared-cone. For reference the corresponding linear growth rate of the secondary pair of oblique waves is also shown.

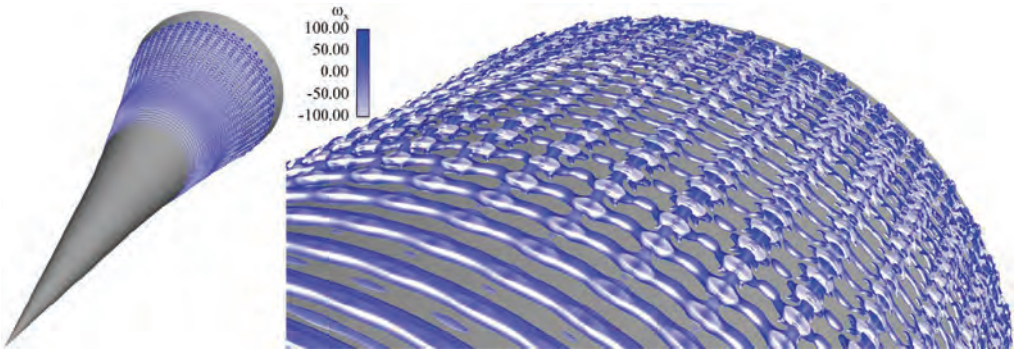


Figure 14. Fundamental Breakdown with f*=230kHz and k_c=48. Iso-surfaces of the Q-vortex identification criterion (Q=10,000) colored by streamwise vorticity, left: full geometry, right: close-up.

5. Conclusion

Direct numerical simulations were performed to investigate boundary-layer stability and transition for three different cone geometries at Mach 6. The first geometry was the 7° straight cone investigated in the Boeing/AFOSR Mach 6 Quiet Tunnel at Purdue University. For this geometry, we investigated transition initiated by a short-duration (localized) pulse as a model of a natural transition scenario. The pulse disturbance leads to three-dimensional wave packet, which travels in downstream direction. The dominant waves within the wave packet were identified as axi-symmetric second-mode waves. The response of the flow to the medium-amplitude pulse disturbance indicated the presence of a fundamental and sub-harmonic resonance mechanisms. However, the fundamental resonance is much stronger than sub-harmonic resonance. Therefore, we performed highly-resolved fundamental (K-type) breakdown simulations.

We also conducted a parameter study to identify the most dangerous (leading to the fastest transition) secondary-pair of oblique waves with the same frequency as the primary-wave for the 5° flared-cone. Using the findings from this study, we set up and performed highly-resolved breakdown simulations deep into the transitional regime. These simulations showed that the K-type breakdown is also a viable path to transition for the 5° flared-cone. Therefore, we can conclude that for a flared-cone, fundamental breakdown is still a viable route to transition in hypersonic boundary-layers. However, in order to answer the question if this is also the most dominant mechanism for both geometries (straight- and flared-cone), further investigations regarding different nonlinear processes are necessary.

6. Significance to DoD

Hypersonic transition research is vitally important for the success of many DoD programs, such as global reach bombers and space operations. Boundary-layer transition is associated with very-high aerodynamics heating and unsteady aerodynamic loads and; therefore, has an enormous impact on the performance of hypersonic flight vehicles. As a consequence of this, reliable transition prediction tools are desperately needed for the development and safe operation of

future high-speed flight vehicles. Our DNS of the late stages of transition are part of the research program of the National Center for Hypersonic Laminar-Turbulent transition (jointly funded by AFOSR and NASA) with the goal of advancing the understanding of high-speed transition and; hence, to improve existing transition prediction methods and to suggest strategies for transition control.

Systems Used

Typical simulations were computed on 512 processors on the Cray XT4 (Jade) and Cray XE6 (Garnet) at the US Army Engineer Research and Development Center (ERDC), and on up to 1,024 processors on the Cray XE6 (Chugach) at the Arctic Region Supercomputing Center (ARSC).

Acknowledgments

This work was funded by the Air Force Office of Scientific Research under grant FA9550-08-1-0211 with Dr. John Schmisseur serving as Program Manager, and the AFOSR/NASA National Center for Hypersonic Laminar-Turbulent Transition Research at the Texas A&M University. Computer time provided by the Arctic Region Supercomputing Center (ARSC) and the US Army Engineer Research and Development Center (ERDC) under challenge project AFOSR26292C4R is gratefully acknowledged.

References

Görtler, H., Über eine dreidimensionale Instabilität laminarer Grenzschichten an konkaven Wänden", *Ges. d. Wiss. Göttingen, Nachr. a. d. Math.*, Vol. Bd. 2, Nr.1, 1940, also NASA Report No. NASA-TM-1375, 1954.

Gross, A. and Fasel, H.F., "Numerical investigation of supersonic flow for axi-symmetric cones", *Mathematics and Computers in Simulation*, 81 (1), pp. 133–142, 2010.

Lachowicz, J., Chokani, N., and Wilkinson, S., "Hypersonic Boundary-Layer Stability Over a Flared Cone in a Quiet Tunnel", *AIAA Conf. 96-0782*, 1996.

Laible, A., Mayer, C., and Fasel, H., "Numerical Investigation of Supersonic Transition for a Circular Cone at Mach 3.5", *AIAA-2008-4397*, 2008.

Laible, A., Mayer, C., and Fasel, H., "Numerical Investigation of Transition for a Cone at Mach 3.5: Oblique Breakdown", *AIAA-2009-3557*, 2009.

Lin, T.C., Grabowsky, W.R., and Yelmgren, K.E., "The search for optimum configurations for re-entry vehicles", *Journal of Spacecraft and Rockets*, Vol. 21, No. 2, 1984, pp.142 – 149.

van Leer, B., "Flux--Vector Splitting for the Euler Equations", *International Conference on Numerical Methods in Fluid Dynamics*, Springer-Verlag, 170, pp. 507–512, 1982.

Li, F., Choudhari, M., Chang, C.-L., Wu, M., and Greene, P., "Development and Breakdown of Görtler Vortices in High-Speed Boundary-Layers", *AIAA Conf.*, 2010-705, 2010.

Mack, L.M., "Boundary-Layer Stability Theory", *Internal Document 900-277*, Jet Propulsion Laboratory, Pasadena, California, 1969.

Pruett, C. and Chang, C., "Direct Numerical Simulation of Hypersonic Boundary-Layer Flow on a Flared Cone", *Theoret. Comput. Fluid Dynamics*, 11, pp. 49-67, 1998.

Schneider, S., Skoch, C., Rufer, S., and Swanson, E., "Hypersonic Transition Research in the Boeing/AFOSR Mach-6 Quiet Tunnel", *AIAA Conf. 2003-3450*, 2003.

Schneider, S.P., Private Communication, 2009.

Sivasubramanian, J. and Fasel, H.F., "Transition Initiated by a Localized Disturbance in a Hypersonic Flat-Plate Boundary- Layer", *AIAA-2011-0374*, 2011.

Zhong, X., "High-Order Finite-Difference Schemes for Numerical Simulation of Hypersonic Boundary-Layer Transition", *J. Comp. Phys.*, 144, pp. 662–709, 1998.

Numerical Simulations of Unsteady Projectile Aerodynamics

Jubaraj Sahu and Karen R. Heavey
US Army Research Laboratory (ARL), Aberdeen Proving Ground, MD
{jubaraj.sahu, karen.heavey}@us.army.mil

Abstract

This paper describes the recent progress made in the development and application of a multidisciplinary computational capability for prediction of unsteady aerodynamics and flight dynamics of projectiles with and without control maneuvers. Advanced computational capabilities in computational fluid dynamics (CFD), rigid-body dynamics (RBD) and flight control system (FCS) have been successfully fully-coupled on high performance computing (HPC) platforms for "Virtual Fly-Outs" of munitions. The coupled CFD/RBD virtual fly-out technique has been applied to an unconventional V-tail finned projectile. Computed results show the projectile to develop coning motion in flight. For guided maneuvers, FCS capability has been integrated into the coupled CFD/RBD procedure. The resulting coupled CFD/RBD/FCS capability has been exercised on a canard-controlled projectile and has been demonstrated using a roll control example. An FCS design was implemented to command the projectile to a desired roll angle using a simple proportional derivative control law for the feedback loop. The effect of canard angle deflection on the aerodynamics and flight dynamics of the canard-controlled projectile can now be computed using the virtual fly-out method and the CFD/RBD/FCS software for different canard maneuvers, and can also be extended to include maneuvers due to jets, flaps, and other control mechanisms.

1. Introduction

Accurate determination of aerodynamics and flight dynamics is critical to the low-cost development of new affordable munitions. Recent advances made in high performance computing (HPC) and computational fluid dynamics (CFD) technologies have the potential for greatly reducing the design costs, while providing a more detailed understanding of the complex aerodynamics than the understanding achieved through experiments and actual test-firings. Three-dimensional (3D) steady and unsteady Navier-Stokes computational techniques have been used to predict aerodynamics of both spinning and fin-stabilized projectiles from subsonic to supersonic speeds.[1–4] Both steady-state and time-accurate CFD techniques are routinely used. Time-accurate or unsteady CFD modeling techniques have proven more challenging, but are increasingly being used for numerical prediction of unsteady projectile and missile aerodynamics.[7–13] In particular, time-accurate methods are used for dynamic derivatives such as the Magnus moment and pitch-damping moment of projectiles. Accurate determination of dynamic derivatives is critical, as they influence the dynamic stability of the projectiles. Fully time-accurate methods offer the greatest potential of providing accurate prediction of these dynamic derivatives. Algorithm and computing advances have also led to coupling of CFD codes to rigid-body dynamics (RBD) codes for the simulation of projectile free-flight motion in a time-accurate manner.[14]

As part of a Department of Defense (DoD) HPC Grand Challenge Project, the US Army Research Laboratory (ARL) has recently focused on the development and application of state-of-the-art coupled CFD and RBD algorithms for prediction of unsteady aerodynamics and flight dynamics of projectiles with and without flow control maneuvers.[14–18] Our objective is to exploit advanced CFD techniques and coupled methods on HPC platforms for design and analysis of unguided and guided projectiles. For maneuvering munitions, the effect of many weapon control mechanisms, such as canards,[19] deployable pins, pulsed flaps, and micro-jets on flight dynamics is critical to guided flight performance. Many of these mechanisms fall outside the range of conventional aerodynamic control, and accurate well-validated tools for prediction of aerodynamic loads are desired. These control mechanisms result in highly-complex, unsteady flow interactions and their accurate modeling during guided-flight with active control is a major challenge. Knowledge of the detailed unsteady aerodynamics of maneuvering precision weapons is rather limited. Multidisciplinary computations has the potential to provide detailed fluid dynamic understanding of the unsteady aerodynamics processes involving the maneuvering flight of modern guided weapon systems. Such knowledge cannot be obtained easily by any other means. A lot of progress has been made recently with the computational technology involving CFD and RBD for prediction of unsteady aerodynamics and

flight dynamics of projectiles without control maneuvers.[16,17] For simulation of control maneuvers, flight control system (FCS) has now been integrated into the coupled CFD/RBD procedure. The resulting CFD/RBD/FCS coupled capability can now be exploited to determine the unsteady aerodynamics and flight dynamics associated with conventional and new aerodynamic control technologies for maneuvering precision munitions.

Coupled CFD/RBD virtual fly-out technique has already been demonstrated and validated a finned projectile at a supersonic speed[14], a spinning projectile at subsonic speed[11], and another spinning projectile at transonic speeds[15]. Validation of the computed results from the virtual fly-out simulations was accomplished with comparison to free flight test data.[14,15] Ongoing research and coupling of FCS with CFD/RBD procedure now is beginning to extend the capability of the coupled technique further for simulation of control maneuvers. BoomFCS[19] has been integrated to provide that added FCS capability. This code has the flight control design capability that allows us to compute the unsteady aerodynamics and flight dynamics associated with guided maneuvers. The advanced CFD capability used in the coupled CFD/RBD/FCS procedure solves the Navier-Stokes equations, incorporates unsteady boundary conditions, special coupling procedure, and restart capability.

This paper describes the progress made in the development and application of the coupled CFD/RBD and CFD/RBD/FCS methods to projectile aerodynamics. The following sections describe the solution technique, coupled CFD/RB, CFD/RBD/FCS procedures and the computed results obtained for an unconventional V-tail projectile and a canard-controlled finned projectile.

2. Computational Methodology

2.1 CFD Flow Solver

The complete set of 3D time-dependent Navier-Stokes equations[20] is solved in a time-accurate manner for simulations of unsteady flowfields associated with both spinning and finned projectiles during flight. The 3D time-dependent Reynolds-averaged Navier-Stokes (RANS) equations are solved using the finite volume method[20,21]:

$$\frac{\partial}{\partial t}\int_V \mathbf{W}dV + \oint\left[\mathbf{F}-\mathbf{G}\right]\cdot dA = \int_V \mathbf{H}dV \tag{1}$$

where **W** is the vector of conservative variables, **F** and **G** are the inviscid and viscous flux vectors, respectively, **H** is the vector of source terms, V is the cell volume, and A is the surface area of the cell face.

Second-order discretization was used for the flow variables and the turbulent viscosity equations. Two-equation k-ε[22] turbulence models were used for the computation of turbulent flows. The RANS/Large-Eddy Simulation (LES) methodology is based on the solution of transport equations for the unresolved turbulence kinetic energy and its dissipation rate and incorporates anisotropy and low-Reynolds number damping effects in both LES and RANS modes. Dual time-stepping was used to achieve the desired time-accuracy. The virtual fly-out technique using coupled CFD and RBD procedures that is of interest here uses dual time-stepping.

2.2 Coupled CFD/RBD Technique

The flow solver allows the computational mesh to actually be moved by assigning a grid velocity to each mesh point. This general capability can be tailored for many specific situations. For example, the grid-point velocities can be specified to correspond to a spinning projectile. In this case, the grid speeds are assigned as if the grid is attached to the projectile and spinning with it. Similarly, to account for rigid-body dynamics, the grid-point velocities can be set as if the grid is attached to the rigid-body with six degrees-of-freedom (6DOF). For the RBD, the coupling refers to the interaction between the aerodynamic forces/moments and the dynamic response of the projectile/body to these forces and moments. The forces and moments are computed at every CFD time-step and transferred to a 6DOF module, which computes the body's response to the forces and moments. The response is converted into translational and rotational accelerations that are integrated to obtain translational and rotational velocities and integrated once more to obtain linear position and angular orientation. The 6DOF rigid-body dynamics module uses quaternions to define the angular orientations. However, these are easily translated into Euler angles. The grid-point locations and grid point velocities are set from the dynamic response.

Typically we begin with a computation performed in "steady-state mode" with the grid velocities prescribed to account only for the translational motion component of the complete set of initial conditions to be prescribed. At this stage, we also impose the angular orientations from the initial conditions and spin rate is added. Computations are performed with the spin in a time-accurate mode for a desired number of spin cycles. Converged solution from this step provides the initial

condition for the next step where a completely coupled computation is performed in time-accurate mode. Here, a complete set of initial conditions includes all translational and rotational velocity components along with initial position and angular orientations are used and accounted for.

2.3 BoomFCS Description

BoomFCS[19] is a RBD code that was specifically designed to predict the atmospheric flight mechanics of smart weapon systems. The key elements of the package are a dynamic model of the weapon in atmospheric flight and a dynamic model of the flight control system. The software automatically couples the air vehicle and its associated control system and subsequently allows a variety of analyses to be performed on the coupled system, including simulation of the time response of the system and impact dispersion statistics. It has been successfully applied to a number of projectiles, unmanned air vehicles, and other flight vehicles.

The rigid-projectile model utilizes six degrees-of-freedom to represent the inertial position and orientation of the projectile body. The three translation degrees-of-freedom are the inertial position components of the projectile mass center. Either Euler angles or quaternion parameters can be selected as the rotation degrees-of-freedom. All projectile models include sophisticated body aerodynamic models, which include steady and unsteady aerodynamic terms. The PRODAS aerodynamic expansion[23] is utilized for body aerodynamics.

BoomFCS has been designed to simulate smart projectile systems. Control of a projectile can be achieved in many different ways. For example, an extended range projectile might be guided with canards. To control the projectile, the canard pitch angle is changed in-flight depending on the location and orientation of the projectile. In a different application, the projectile might be controlled by pulse jets that get activated during flight. Not only are many different control mechanisms used to control projectiles, many different strategies are employed to guide and control a projectile. The control system strategy is conveyed through the control law. The control law stipulates a set of operations that are performed on sensor data to determine how the controls should be changed in flight. The code uses an open-structure flight control system modeling architecture that enables flight control system models to be built directly from block diagram information. The open-structure flight control system consists of a set of basic building blocks called flight control system modeling elements. Through program input data, the user selects and arranges flight control system modeling elements to mimic the physical arrangement which is desired to simulate. The software automatically couples all control system elements together. Any physical parameter of the projectile model can be dynamically controlled. With this arrangement, virtually any weapon flight control system can be modeled in detail.

2.4 Implementation of FCS into CFD/RBD Coupled Procedure

For simulations of controlled flights of projectiles BoomFCS[19] has been implemented into the coupled CFD/RBD procedures described in Section 2.2. The added element of this coupling is the flight control system or the guidance, navigation, and control aspect. An interface was created and developed for easy transfer of both the rigid-body dynamics state variables and the control system variables of interest between the flow solver and BoomFCS code.

The RBD coupling which refers to the interaction between the aerodynamic forces/moments and the dynamic response of the projectile/body to these forces and moments remains the same as before. The aerodynamic forces and moments are computed at every time-step in CFD part and transferred to BoomFCS, which does both RBD and FCS simulations. The FCS simulation provides as output the flight control variables based on a given FCS design. The FCS design allows for modeling of both controlled and prescribed motions. The output of RBD state and the control variables are transferred to the CFD flow solver, which then computes the aerodynamic forces and moments subject to these RBD state and control variables. For example, for a canard-controlled projectile, the output of the FCS variables would be the canard deflection angles. As the canards are deflected with the FCS-generated canard deflection angles, the flow solver must take into account the canard motion. Similarly, a set of different FCS variables could be used for a jet-controlled maneuver. As this FCS coupling matures, more and more control variables can be used for simulations of a variety of controlled flights.

2.5 Parallel Computational Aspects

The flow solver or the code used in the challenge project can be run in parallel on a wide variety of hardware platforms and parallel computers, including those from Silicon Graphics and PC workstation clusters. The numerical method used is scalable on newer computers such as the Linux Cluster and the SGI Altrix. It works on different communications libraries including MPI, PVM, and other libraries. The method incorporates programming enhancements such as dynamic memory allocation and highly- optimized cache management. Inter-CPU communications are included at the fine-grid level as well

as all the multi-grid levels. This strategy ensures a high degree of robustness in the flow solver, and is independent of the number of CPUs being employed.

The mesh files that are needed in the computations are generated only once. The same mesh files can be used for single-CPU or any multi-CPU runs. The same is true of the restart files. For multi-CPU runs, a domain-decomposition file is needed; again this file is obtained only once at the beginning of the run. The domain decomposition file defines the association between cell numbers and CPUs. Domain decomposition was done using the METIS tool developed at the University of Minnesota.

All present numerical simulations were performed on the 1,100-node, 4,400-core, Linux Networx Advanced Technology Cluster (MJM), and the newer 10,752-core SGI Altix ICE 820 System (Harold) at the US Army Research Laboratory (ARL) Department of Defense Supercomputing Resource Center (DSRC). Typically, the jobs were run with either 128 or 256 processors on these high-computing machines available at the ARL DSRC. The number of processors used for each run was chosen such that about 150,000 cells were partitioned on each processor. Very good performance of the flow solver is achieved on many modern platforms. The parallel performance of the flow solver on both parallel machines, MJM, and Harold, is shown in Figures 1 and 2. Three different mesh sizes were used in the benchmark runs for scalability of the code. It should be noted that these benchmark runs were done on these systems in a real multi-users/multi-jobs environment and not on a dedicated machine. Figure 1 shows the parallel performance of the code on the MJM Linux cluster (3.06GHz). As shown here, very good parallel performance has been observed up to 256 processors on this Linux cluster. More scatter in the data was observed with different mesh sizes as the number of processors increased. The parallel performance of the code on Harold is shown in Figure 2; again, excellent performance is obtained on this machine for number of processors up to 256 and reasonable speed-up up to 512 processors. Parallel performance results obtained with different meshes were similar, and not found to be sensitive to the size of the mesh.

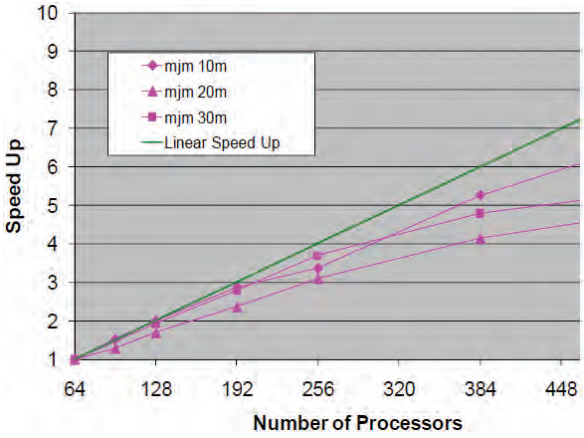


Figure 1. Parallel speed-ups on a Linux cluster (MJM)

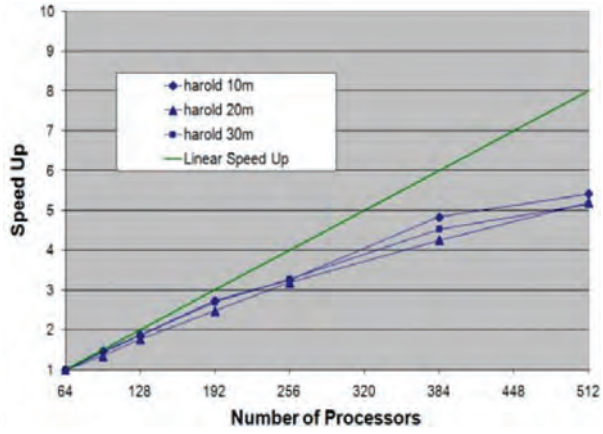


Figure 2. Parallel speed-ups on a SGI Altrix (Harold)

3. Results

The time-accurate coupled virtual fly-out approach described earlier has been used to generate aerodynamics and flight dynamics of a number of projectile configurations. Results obtained for some of these cases are presented next.

3.1 Virtual Fly-Out Simulations of an unconventional V-Tail Finned Projectile

Time-accurate numerical computations were performed using coupled CFD/RBD methods to predict the aerodynamics and flight dynamics of a new unconventional V-tail finned projectile. Research is currently ongoing at ARL to develop new non-rolling projectile configurations that can provide good platforms for easier control maneuvers. Some initial flight-tests have shown such projectile configurations to develop roll and coning motion. Our research efforts are directed at understanding this roll issue. Coupled virtual fly-out computational technique is being applied to simulate the projectile flight dynamics. Computed solutions have been obtained for this unconventional projectile at low subsonic speed, Mach=0.2 for various initial conditions.

The projectile modeled in this study is an ogive-cylinder body with two V-tail fins located at the back end of the projectile (see Figure 3). The length of the projectile is 190 mm and the diameter is 26.4 mm. The computational mesh for the projectile is a multi-block structured grid consisting of about 6 million cells. This mesh also included modeling of the base region of the projectile. Figure 4 shows the surface pressure distribution on the projectile at angle-of-attack at Mach=0.2 at a given instant in time during the virtual fly-out simulation. One can clearly see the asymmetry in the surface pressures on the projectile base. The flowfield and aerodynamic forces and moments acting on the projectile are expected to continually change as the projectile flies down-range.

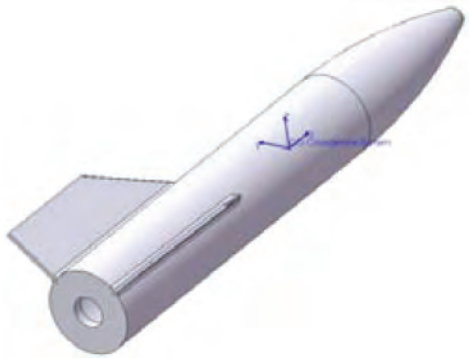


Figure 3. Computational model of the V-tail projectile

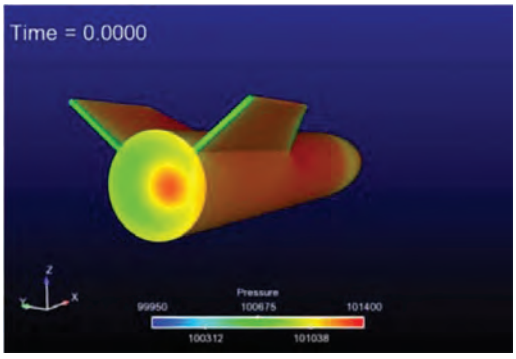


Figure 4. Computed surface pressure contours

The first step in the virtual fly-out simulation was to obtain the steady state results for this projectile at a given initial subsonic velocity. Virtual fly-out simulations were carried out using the time-accurate coupled CFD/RBD technique described earlier. Different sets of initial conditions were used where the roll-rate (p) was set to zero, but the pitch-rate (q) and the yaw-rate (r) were varied. The roll-rate p is set to zero; also set to zero are the lateral velocity (v) and the vertical velocity (w) components. The orientation and the position of the projectile of course change from one instant in time to another as the projectile flies down-range. Computed results produced by the virtual fly-out simulations for a representative

case are shown in Figures 5 and 6. Here the pitch-rate is set to 3 rad/s and the yaw-rate was set to 4 rad/s. Figure 5(a) shows the angle-of-attack and the side-slip angle as a function of time. Both of these angles start out at zero and begin to oscillate quickly. After 0.5 s in time, both angle-of-attack and the side-slip angle are seen to oscillate between +8° and -8°. The projectile is found to undergo coning motion which continues to persist in time. The corresponding total angle-of-attack is shown in Figure 5(b) as a function of time. As seen here, the total angle-of-attack increases in the beginning up to almost +10° at 0.2 s and then oscillates with a small amplitude with a mean angle-of-attack of 8°. Figure 6 shows a motion plot where angle-of-attack is plotted against the side-slip angle. It clearly shows the coning motion, the path the nose of the projectile takes as it moves down-range in time. This is similar to the coning motion observed in flight-tests of this projectile.

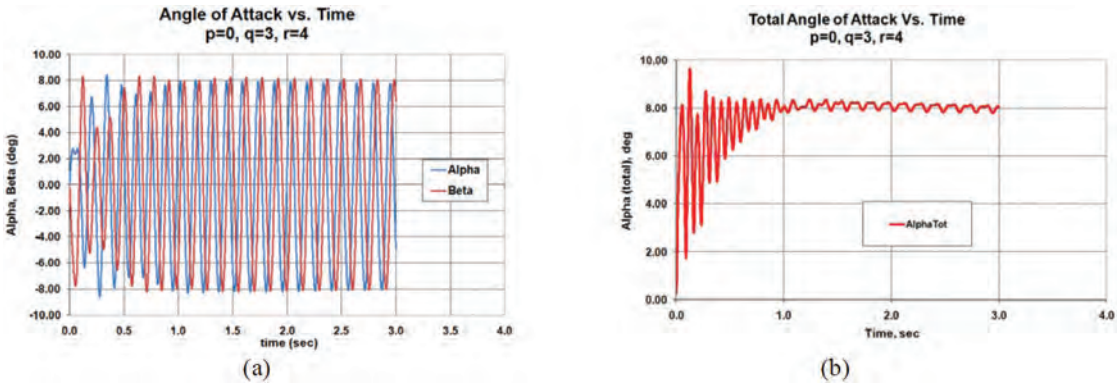


Figure 5. Angle-of-attack and side-slip (a) and total angle-of-attack (b) as a function of time

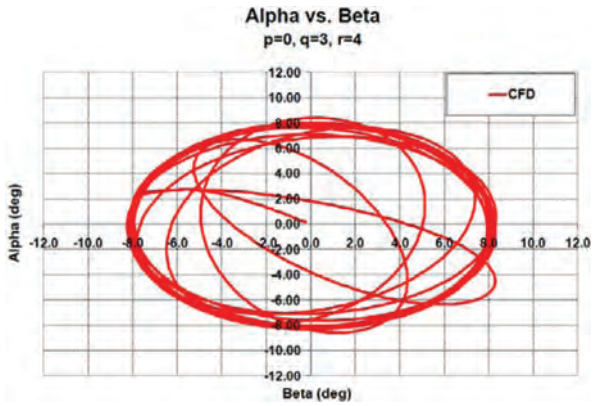


Figure 6. Motion plot, angle-of-attack vs. side-slip angle

As pointed out earlier, virtual fly-out computations were performed for this projectile at different pitch- and yaw-rates. Figure 7a shows the computed Euler angles as a function of time for the case with a given yaw-rate, but with the pitch-rate set to zero. As seen in this figure, the Euler roll angle, which initially was zero, begins to change immediately. The projectile starts rolling and the simulation here shows approximately four cycles of rolling motion. This rolling motion is undesirable. Computed results from another coupled virtual fly-out simulation for a given pitch-rate where the yaw-rate is set to zero are shown in Figure 7b. As can be seen here, the Euler roll angle in this case remains practically zero and the projectile doesn't experience rolling motion for the entire flight (3 s in time). These results shown in Figure 7 clearly identified the yaw-rate to be the cause of the rolling motion and not the pitch-rate. Based on these results and findings, new design configurations are being explored and efforts are currently underway to develop improved designs with desirable flight dynamic characteristics.

One of our main objectives in this challenge project is to develop and apply coupled CFD/RBD/FCS techniques for physics-based simulations of guided maneuvers. BoomFCS is being used for these coupled virtual fly-out simulation cases an example of which is described next.

3.2 Canard-Controlled Projectile in Transonic Flight

Of interest is the control of trajectory of projectiles using maneuver control mechanisms such as canards, jets, and flaps. In the present study, our application of interest is a canard-controlled projectile. Projectile control is provided using canards. The canards are deflected to generate control forces required to maneuver the projectile in roll, pitch, or a combination of both. BoomFCS, described earlier, is used to execute prescribed dynamic motion or a guided maneuver during the simulation runs. Coupled CFD/RBD/FCS method is used to predict the unsteady aerodynamics and the flight dynamics of the projectile during the canard-control maneuvers.

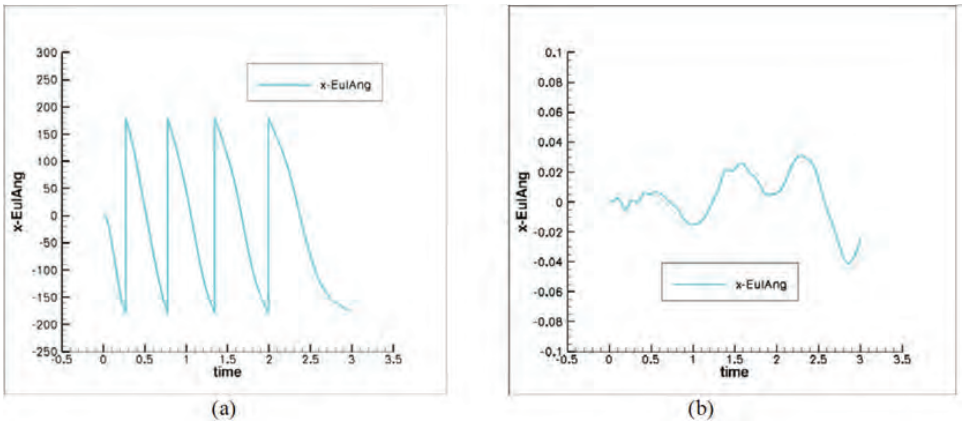


Figure 7. Time-history of Euler roll angle, (a) q=0, r=1 and (b) q=1, r=0

The canard-controlled projectile considered in this study has two canards and six fins (Figure 8). The control maneuver is achieved by two canards located in the nose section of the projectile. All six fins are located at the aft-end of the projectile. An unstructured grid is generated for the projectile body with the fins, which forms as the background mesh. Unstructured grids are generated about each canard separately. The two canard grids are then overset with the background projectile mesh to a Chimera overlapped mesh for the canard-controlled projectile. The total number of grid-points in this case is approximately 6.3 million. This Chimera procedure[10] requires proper transfer of information between the background mesh and the canard meshes. However, the advantage is that the individual grids are generated only once, and the Chimera procedure can then be applied repeatedly as required during the canard motion. There is no need to generate the meshes at each time-step during the canard movement.

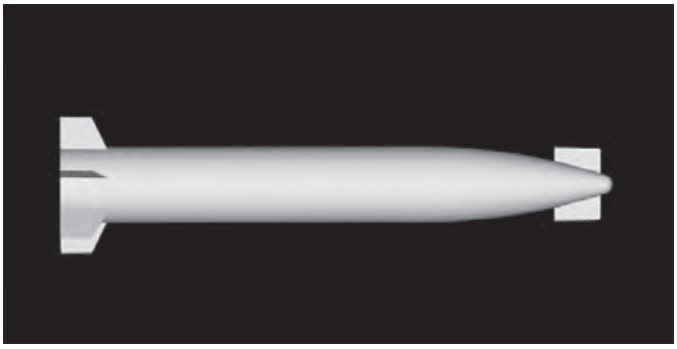


Figure 8. Canard-controlled finned projectile geometry

The first step in the coupled virtual fly-out simulation is the generation of a steady-state solution for the projectile moving forward at a velocity which corresponds to Mach=0.9. A simple test case considered here is the roll control of the canard-controlled projectile. The projectile is flying in a straight level flight at zero degree angle-of-attack. In addition, the projectile is not spinning; so, the initial spin rate is set to zero. A flight control system, FCS design has been implemented to command the projectile to a desired roll angle of 45°. A simple proportional derivative control law is used for the feedback loop. The maximum canard deflection angle was set to 5°. An initial coupled CFD/RBD/FCS calculation is performed to demonstrate the capability of the coupled technique using the roll control example. Figure 9 shows the computed surface

pressure contours on the nose section of the canard-controlled projectile at an initial speed, Mach=0.9 and α=0° shortly after canards started moving. The canard deflection makes the flow in the nose section asymmetric and provides a differential force. For the roll control case, canards on each side are deflected in opposite directions and the projectile starts to roll. Figure 10 shows the canard locations at the end of the simulation when the roll angle reaches the commanded value of 45°. Initially, the canards were located in the horizontal plane. Figure 11a shows the computed Euler roll angle as function of time during the roll control maneuver. As seen here, the commanded roll angle is reached after about 0.18 sec in time. The corresponding canard deflection of one of the canards is shown in Figure 11b. With the FCS on, the canard starts deflecting and reaches its minimum (-5°) very quickly and stays constant at that value for most of the flight time, and then drops to almost zero towards the end of the simulation very quickly again. These results correspond to just one FCS design. The maximum canard deflection angle, as well as control law parameters, can be adjusted and varied to obtain desired roll control characteristics.

Currently, work is in progress to perform additional coupled CFD/RBD/FCS simulations for different parameters. A variety of pitch and roll maneuvers can be implemented. In addition, both unguided and guided maneuvering cases can be simulated using the same coupled FCS method available in BoomFCS. For unguided maneuvers, the canards are deflected and execute prescribed dynamic motion during the simulation runs. For guided maneuvers, control law with feedback loop can be used, as has been demonstrated here with the roll control example.

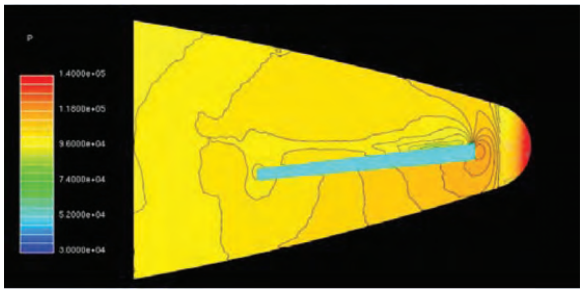


Figure 9. Computed surface pressure contours on the nose, M=0.9, α=0°

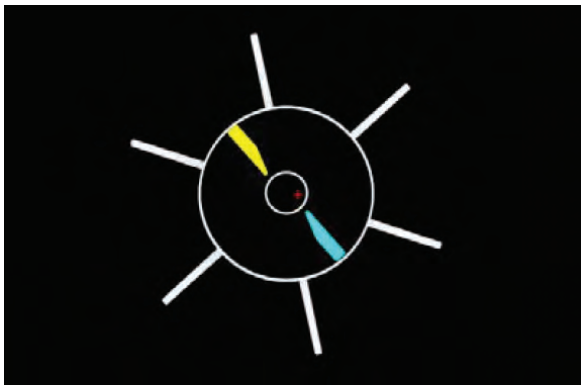


Figure 10. Orientation of canards at the end of simulation

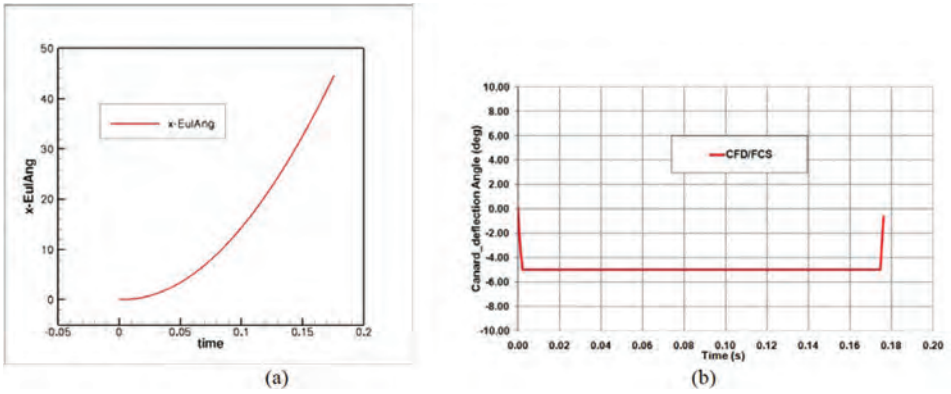


Figure 11. Roll angle (left) and Canard deflection angle (right) as a function of time

4. Concluding Remarks

This paper describes the development and application of a multidisciplinary capability for prediction of unsteady aerodynamics and flight dynamics of maneuvering projectiles. Advanced computational capabilities in computational fluid dynamics (CFD), rigid-body dynamics (RBD) and flight control system (FCS) have been successfully fully coupled on high performance computing platforms for physics-based "virtual fly-outs" of munitions. These coupled virtual fly-out simulations required time-accurate Navier-Stokes computations, and were performed using an advanced scalable unstructured flow solver on highly-parallel Linux and SGI Clusters.

Several coupled CFD/RBD virtual fly-out simulations were performed on an unconventional V-tail projectile at a subsonic speed for different initial pitch- and yaw-rates. Computed results clearly have identified the yaw-rate to be the cause for the non-spinning projectile to roll. In addition, the computed results show the projectile to develop coning motion similar to that observed in flight-test. The capability of the coupled CFD/RBD procedure was extended with implementation of an FCS capability. The implementation of FCS design into coupled CFD/RBD virtual fly-out simulations is a significant advancement, and was demonstrated with the roll control of a canard-controlled projectile with control law and feedback loop. The effect of canard control maneuvers on the aerodynamics and flight dynamics of canard-controlled projectiles can now be determined using the coupled CFD/RBD/FCS procedure. This coupled procedure can also be extended for simulation of control maneuvers due to jets, flaps, and other flow control mechanisms. Current and future research efforts will help determine the potential of these techniques for providing the actual time-dependent response, and the resulting unsteady aerodynamics and flight dynamics of maneuvering projectiles for both guided and unguided maneuvers.

Acknowledgments

The authors would like to thank the Department of Defense High Performance Computing Modernization Program (HPCMP) for sponsoring this work as a part of a Grand Challenge Project, and for providing critically-needed HPC resources for successful accomplishment of the work. The author also wishes to thank Prof. Mark Costello of Georgia Institute of Technology and Dr. Sukumar Chakravarthy of the Metacomp Technologies and for their technical assistance with the coupling between CFD/RBD/FCS methods. The scientific visualization and the computational support of the ARL DSRC are greatly appreciated.

References

1. Sahu, J., "Numerical Computations of Transonic Critical Aerodynamic Behavior", *AIAA Journal*, Vol. 28, No. 5, pp. 807–816, May 1990.

2. Sturek, W., C. Nietubicz, J. Sahu, and P. Weinacht, "Applications of Computational Fluid Dynamics to the Aerodynamics of Army Projectiles", *Journal of Spacecraft and Rockets*, Vol. 31, No. 2, pp. 186–199, 1994.

3. Sahu, J., D. Pressel, K.R. Heavey, and C.J. Nietubicz, "Parallel Application of a Navier-Stokes Solver for Projectile Aerodynamics", *Proceedings of Parallel CFD'97 Meeting*, Manchester, England, May 1997.

4. Weinacht, P., "Navier-Stokes Prediction of the Individual Components of the Pitch-Damping Sum", *Journal of Spacecraft and Rockets*, Vol. 35, No. 5, pp. 598–605, 1998.

5. Sahu, J., H.L. Edge, S. Dinavahi, and B. Soni, "Progress on Unsteady Aerodynamics of Maneuvering Munitions", *Users Group Meeting Proceedings*, Albuquerque, NM, June 2000.

6. Sahu, J., "Unsteady Numerical Simulations of Subsonic Flow over a Projectile with Jet Interaction", *AIAA Paper no. 2003-1352*, 41st Aerospace Sciences Meeting, Reno, NV, 6–9 Jan 2003.

7. Silton, S., "Navier-Stokes Computations for a Spinning Projectile from Subsonic to Supersonic Speeds", *Journal of Spacecraft and Rockets*, Vol. 42, No. 2, pp. 223–231, 2005.

8. Sahu, J., "Unsteady CFD Modeling of Aerodynamic Flow Control over a Spinning Body with Synthetic Jet", *AIAA Paper 2004-0747*, Reno, NV, 5–8 January 2004.

9. DeSpirito, J. and P. Plostins, "CFD Prediction of M910 Projectile Aerodynamics: Unsteady Wake Effect on Magnus Moment", *AIAA-2007-6580*, Aug. 2007

10. Sahu, J. and C.J. Nietubicz, "Application of Chimera Technique to Projectiles in Relative Motion", *Journal of Spacecraft & Rockets*, Sept–Oct 1995.

11. Sahu, J. and K.R. Heavey, "Unsteady CFD Modeling of Micro-Adaptive Flow Control for an Axi-symmetric Body", *International Journal of Computational Fluid Dynamics*, Vol. 5, April–May 2006.

12. Stalnaker, J.F. and M.A. Robinson, "Computations of Stability Derivatives of Spinning Missiles Using Unstructured Cartesian Meshes", *AIAA-2002-2802*, June 2002.

13. Sahu, J. and K.R. Heavey, "Time-Accurate Computations for Rapid Generation of Missile Aerodynamics", *AIAA Atmospheric Flight Mechanics Meeting*, Toronto, Canada, 2–6 Aug 2010.

14. Sahu, J., "Time-Accurate Numerical Prediction of Free-Flight Aerodynamics of a Finned Projectile", *AIAA-2005-5817*, AIAA Atmospheric Flight Mechanics Conference, San Francisco, CA, 2005.

15. Sahu, J., "Computations of Unsteady Aerodynamics of a Spinning Body at Transonic Speeds", *AIAA paper No. 2009-3852*, 27th Applied Aerodynamics Conference, San Antonio, TX, 22–25 June 2009.

16. Sahu, J., M. Costello, J. Stahl, J. DeSpirito, and K.R. Heavey, "Numerical Computations of Unsteady Aerodynamics of Maneuvering Projectiles", *DoD Users Group Conference Proceedings*, San Diego, CA, June 16–18, 2009.

17. Sahu, J. and K.R. Heavey, "Progress in Simulations of Unsteady Projectile Aerodynamics", *DoD Users Group Conference Proceedings*, Schaumburg, IL, June 2010.

18. Costello M., S. Gatto, and J.Sahu, "Using CFD/RBD Results to Generate Aerodynamic Models for Projectile Flight Simulation", *AIAA Atmospheric Flight Mechanics Conference*, Hilton Head, SC, 2007.

19. Costello, M., "Extended Range of a Gun-Launched Smart Projectile Using Controllable Canards", *Shock and Vibration*, Vol. 8, No. 3–4, pp. 203–213, 2001.

20. Peroomian, O., S. Chakravarthy, S. Palaniswamy, and U. Goldberg, "Convergence Acceleration for Unified-Grid Formulation Using Preconditioned Implicit Relaxation", *AIAA Paper 98-0116*, 1998.

21. Peroomian, O. and S. Chakravarthy, "A `Grid-Transparent' Methodology for CFD", *AIAA Paper 97-0724*, Jan. 1997.

22. Goldberg, U., O. Peroomian, and S. Chakravarthy, "A Wall-Distance-Free K-E Model With Enhanced Near-Wall Treatment", *ASME Journal of Fluids Engineering*, Vol. 120, 457–462, 1998

23. Arrow Tech Associates, *ARFDAS Technical Manual*, South Burlington, VT, 2001.

Qubit Lattice (QLG) Simulations for a T > 0 Superfluid

George Vahala and Bo Zhang
Department of Physics, College of William & Mary, Williamsburg, VA
{gvahala, bozhang1131}@gmail.com

Linda Vahala
Department of Electrical & Computer Engineering, Old Dominion University, Norfolk, VA
lvahala@odu.edu

Min Soe
Department of Mathematics and Physical Sciences, Rogers State University, Claremore, OK
msoe.rsu@gmail.com

Sean Ziegeler
HPTi, Vienna, VA
sean.ziegeler@nrlssc.navy.mil

Abstract

The qubit representation of Yepez for a T > 0 superfluid is examined numerically using a unitary representation that scales almost perfectly to all available processors (216,000 cores). Since the algorithm is unitary, it is directly encodable on quantum computers. At the macroscopic level, the finite-temperature superfluid has been described by a 2-fluid theory: an inviscid Eulerian fluid interacting with a viscous Navier-Stokes fluid. At the microscopic level, the 4-states of the 2-qubits are coupled together by unitary transformations: 2 of these states describe the superfluid ground-state and 2 of these states describe the Bogoliubov quasi-particles. Simulations are presented that show possible mode-locking of the BEC and Bogoliubov states for sufficiently-large coupling.

1. Introduction

We shall utilize the recent theory of Yepez [2011] to consider the effects of finite-temperature on a superfluid. At the macroscopic level, one has the well-known two-fluid theory (Tisza, 1947; Landau, 1941): a superfluid at a quantum ground-state with no viscosity and no entropy - showing some similarities to a classical inviscid Eulerian fluid - interacting with a normal fluid with viscosity and entropy and behaving much like a classical viscous Navier-Stokes fluid. At zero temperature, the normal fluid density is zero and one just has the superfluid, while at the transition temperature the superfluid density is zero and one just has the normal fluid. At zero temperature, we require 2 qubits/lattice site and consider that we are dealing with a Bose-Einstein condensate (BEC) state so that the N-body wave-function collapses to product of N 1-body wave-functions (Pethick & Smith, 2002). Since the BEC state is a weakly-coupled state, there is a weak nonlinear interaction between the bosonic states whose dynamics are given by the Gross-Pitaevskii equation (in dimensionless form):

$$i\frac{\partial \psi}{\partial t} = -\nabla^2\psi + a\left(g|\psi|^2 - 1\right)$$

g denotes the strength of this s-wave coupling while a is a rescaling parameter to resolve structure within the so-called coherence length. Under the Madelung transformation:

$$\psi = \sqrt{\rho}\exp(i\theta) \qquad \mathbf{v} = 2\ \nabla\ \theta$$

the Gross-Pitaevskii equation takes a form reminiscent of a classical inviscid Eulerian barotopic fluid

$$\frac{\partial \rho}{\partial t} + \nabla\bullet(\rho\mathbf{v}) = 0 \quad , \quad \rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\bullet\nabla\mathbf{v}\right) = -2\rho\nabla\left(g\rho - \frac{\nabla^2\sqrt{\rho}}{\sqrt{\rho}}\right)$$

The very unusual pressure-like term of the form $\nabla^2\sqrt{\rho}/\sqrt{\rho}$ has marked effects on the vortex dynamics of the BEC state. For example, it permits vortex reconnection without viscosity and vortex quantization—both features not seen in classical inviscid fluids.

2. Yepez's Unitary Qubit Theory for Bogoliubov States

We now briefly summarize the recent unitary qubit theory of Yepez [2011] to describe the coupling between the Bose-Einstein (BEC) zero-temperature field and the Bogoliubov quasi-particles comprising the normal fluid part for temperature states T > 0. Again only 2 qubits per lattice site are required. The basis for this Hilbert space is just the:

$$|00\rangle, |01\rangle, |10\rangle, |11\rangle \tag{1}$$

The ground BEC state is viewed as a Cooper pair forming a bound Bosonic molecule, with:

$$|\Psi_{\parallel}(\mathbf{x})\rangle = \Psi_{\parallel L}|00\rangle + \Psi_{\parallel R}|11\rangle \tag{2}$$

while the Bogoliubov quasi-particle state is a locally-entangled particle-hole pair with:

$$|\Phi_{\perp}(\mathbf{x})\rangle = \Phi_{\perp L}|01\rangle + \Phi_{\perp R}|10\rangle \tag{3}$$

In spinor representation, the 4-state:

$$|\psi(\mathbf{x})\rangle = \begin{pmatrix} \Psi_{\parallel L}(\mathbf{x}) \\ 0 \\ 0 \\ \Psi_{\parallel R}(\mathbf{x}) \end{pmatrix} + \begin{pmatrix} 0 \\ \Phi_{\perp L}(\mathbf{x}) \\ \Phi_{\perp R}(\mathbf{x}) \\ 0 \end{pmatrix} = |\Psi_{\parallel}(\mathbf{x})\rangle + |\Phi_{\perp}(\mathbf{x})\rangle \tag{4}$$

For the finite-temperature system the scalar-field:

$$\varphi(\mathbf{x}) = \Psi_{\parallel L}(\mathbf{x}) + \Psi_{\parallel R}(\mathbf{x}) + \Phi_{\perp L}(\mathbf{x}) + \Phi_{\perp R}(\mathbf{x}) \tag{5}$$

and the coupling between the BEC and normal fluid can be characterized by the unitary transformation:

$$|\psi'(\mathbf{x})\rangle = e^{-i\gamma}\begin{pmatrix} \cos\gamma & 0 & i\sin\gamma & 0 \\ 0 & \cos\gamma & 0 & i\sin\gamma \\ i\sin\gamma & 0 & \cos\gamma & 0 \\ 0 & i\sin\gamma & 0 & \cos\gamma \end{pmatrix}\begin{pmatrix} \Psi_{\parallel L}(\mathbf{x}) \\ \Phi_{\perp L}(\mathbf{x}) \\ \Phi_{\perp R}(\mathbf{x}) \\ \Psi_{\parallel R}(\mathbf{x}) \end{pmatrix} \tag{6}$$

with conservation of scalar-wave function density $|\varphi(x)|^2 = |\varphi'(x)|^2$. γ is real-valued and characterizes the strength of the BEC-Bogoliubov quasi-particle coupling.

The quantum algorithm requires 4 amplitudes at each lattice point and the strictly unitary qubit representation takes the form (Yepez, 2011):

$$\begin{pmatrix} \Psi'_{\parallel L}(\mathbf{x}) \\ \Phi'_{\perp L}(\mathbf{x}) \\ \Phi'_{\perp R}(\mathbf{x}) \\ \Psi'_{\parallel R}(\mathbf{x}) \end{pmatrix} = U^{1,4}U^{2,3}\begin{pmatrix} 0 & 0 & 0 & e^{-i\varepsilon^2\Omega_{\parallel}} \\ 0 & 0 & e^{-i\varepsilon^2\Omega_{\perp}} & 0 \\ 0 & e^{-i\varepsilon^2\Omega_{\perp}} & 0 & 0 \\ e^{-i\varepsilon^2\Omega_{\parallel}} & 0 & 0 & 0 \end{pmatrix} Exp\left[-i\varepsilon^2 g|\varphi_v|^2 N_{\text{int}}^{quasi}\right].$$

$$e^{-i\gamma}\begin{pmatrix} \cos\gamma & 0 & i\sin\gamma & 0 \\ 0 & \cos\gamma & 0 & i\sin\gamma \\ i\sin\gamma & 0 & \cos\gamma & 0 \\ 0 & i\sin\gamma & 0 & \cos\gamma \end{pmatrix}\begin{pmatrix} \Psi_{\parallel L}(\mathbf{x}) \\ \Phi_{\perp L}(\mathbf{x}) \\ \Phi_{\perp R}(\mathbf{x}) \\ \Psi_{\parallel R}(\mathbf{x}) \end{pmatrix} \tag{7}$$

where $\varphi_v(\mathbf{x}) = \Psi_{\|L}(\mathbf{x}) + \Psi_{\|R}(\mathbf{x})$, $\Omega_{\|}(\mathbf{x}) = g\,|\varphi_v|^2 - \mu$, $\Omega_{\perp}(\mathbf{x}) = 2g\,|\varphi_v|^2 - \mu$, and the Hermitian generator for the Bogoliubov quasi-particles:

$$N_{\text{int}}^{quasi} = \frac{i}{2|\varphi_v|^2}\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -\left(\varphi_v^2 - \varphi_v^{2}*\right) & \left(\varphi_v^2 + \varphi_v^{2}*\right) & 0 \\ 0 & -\left(\varphi_v^2 + \varphi_v^{2}*\right) & \left(\varphi_v^2 - \varphi_v^{2}*\right) & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{8}$$

Finally, the unitary matrices $U^{1,4}$ and $U^{1,4}$ are the 2-qubit collide-stream operator sequence that we have been using in our previous studies on qubit representation of quantum turbulence (Yepez, 2009; Yepez, et al., 2009; Vahala, 2010). In particular $U^{1,4}$ act on the BEC amplitudes $\Psi_{\|L}(\mathbf{x})$, $\Psi_{\|R}(\mathbf{x})$ with:

$$U^{1,4} = \hat{I}_{x0}^2\ \hat{I}_{y1}^2\ \hat{I}_{z0}^2\ \hat{I}_{z1}^2\ \hat{I}_{y0}^2\ \hat{I}_{x1}^2 \tag{9}$$

where $\hat{I}_{x0} = S_{-\Delta x,0}\ C\ S_{\Delta x,0}$. C. C is the unitary $\sqrt{SWAP}$ collision operator that locally-entangles the two BEC amplitudes:

$$C = \begin{pmatrix} \frac{1-i}{2} & \frac{1+i}{2} \\ \frac{1+i}{2} & \frac{1-i}{2} \end{pmatrix} \tag{10}$$

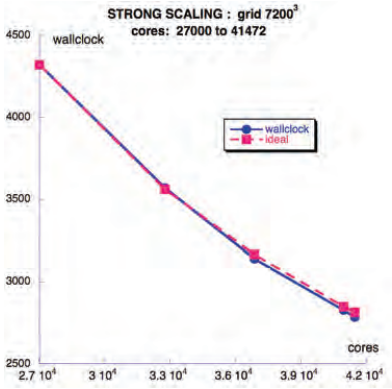
while the streaming unitary operator $S_{-\Delta x,0}$ moves the entangled post-collision amplitude $\Psi_{\|L}$ from $\mathbf{x} \rightarrow \mathbf{x} - \Delta x$. Similarly, $S_{\Delta x,1}$ moves $\Psi_{\|R}$ from $x \rightarrow x - \Delta x$. The Yepez qubit mapping, Equation 6, will result in the following coupled equations of motion for the spinor-fields:

$$\begin{aligned} i\,\sigma_x \frac{\partial \Psi_{\|}}{\partial t} &= \left(-\frac{1}{2}\nabla^2 + \Omega_{\|}\right)\Psi_{\|} + \gamma\ \Phi_{\perp} \\ i\,\sigma_x \frac{\partial \Phi_{\perp}}{\partial t} &= \left(-\frac{1}{2}\nabla^2 + \Omega_{\perp}\right)\Phi_{\perp} - i\frac{g|\varphi_v|^2}{2}\sigma_z\ e^{-2i\theta\,\sigma_x}\Phi_{\perp} + \gamma\ \Psi_{\|} \end{aligned} \tag{11}$$

where $\psi_v = \sqrt{\rho}\exp(i\theta)$.

3. Scaling of Bogoliubov Qubit Lattice Code BdG with Cores

We have performed scaling on *Raptor*, the Cray XE6 at the US Air Force Research Laboratory (AFRL) to the maximum number of cores available. In strong-scaling, we keep the grid fixed (in this case 7,200) and increased the number of cores from 27,000 to 41,472. In weak-scaling, the grids are increased with the number of cores, so that ideally the wall-clock should remain constant with the same amount of computations done per core. Figure 1 indicates that we are achieving excellent scaling, both in strong- and weak-scaling. The weak-scaling was performed for 200 time-steps.



Weak Scaling on Raptor, to 42875 cores

grid	cores	wallclock
2048^3	512	5209.1
4096^3	4096	5210.0
6144^3	13824	5226.6
8192^3	32768	5207.7
8960^3	42875	5246.1

Figure 1. Strong- and Weak-scaling of the BdG code on *Raptor*, Cray XE6. The solid blue curve is the code's wall-clock timing while the dashed red curve is ideal timing. The ideal wall-clock time for weak-scaling is 5209.1 s.

Strong-scaling runs were also performed on the Cray XT5/*Jaguarpf* to 216,000 cores and on Cray XE6/*Hopper II* to 150,000 cores with timings that suggest super-linear scaling.

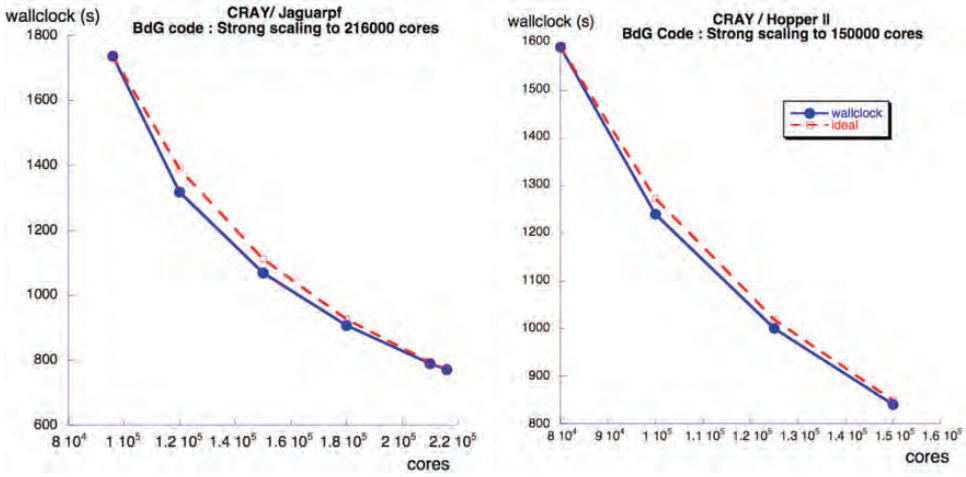


Figure 2. Strong scaling of the BdG code on the Cray XT5/*Jaguarpf* to 216,000 cores and on the Cray XE6/*Hopper II* to 150,000 cores. The solid blue curve is the code's wall-clock timing while the dashed red curve is ideal timing, showing super-linear scaling of the BdG code. Grid 8400³, 100 time-steps.

The weak-scaling results were also extremely good, tested up to 110,592 cores.

Table 1. The weak-scaling of the BdG code up to 46,656 cores on the Cray XT5 and up to 110,592 cores on the Cray XE6. In weak-scaling, the number of cores is increased in proportion to the increase in grid³ so that ideal timing would yield a constant wall-clock for each run. The weak-scaling is excellent.

(a) Weak scaling on CRAY XT5

grid	cores	wallclock
600^3	216	261
1200^3	1728	261.8
2400^3	13824	263.9
3600^3	46656	268.4

(b) Weak scaling on CRAY XE6

grid	cores	wallclock
600^3	216	209.2
1200^3	1728	212.7
2400^3	13824	212.1
3600^3	46656	212.7
4000^3	64000	213.3
4800^3	110592	214.0

4. Simulation of BdG States

We now consider some preliminary results of our simulations in the coupling of the BEC ground-state to the Bogoliubov quasi-particle $T > 0$ state. If the parameter $\gamma = 0$ in Equation 11, we uncouple the Bogoliubov state from the evolution of the BEC wave function, so that the evolution equations are:

$$i\,\sigma_x \frac{\partial \Psi_{\|}^{UN}}{\partial t} = \left(-\frac{1}{2}\nabla^2 + \Omega_{\|}^{UN} \right) \Psi_{\|}^{UN} \quad , \quad \varphi_v^{UN} = \Psi_{\|L}^{UN} + \Psi_{\|R}^{UN}$$

$$i\,\sigma_x \frac{\partial \Phi_{\perp}^{UN}}{\partial t} = \left(-\frac{1}{2}\nabla^2 + \Omega_{\perp}^{UN} \right) \Phi_{\perp}^{UN} - i\frac{g\left|\varphi_v^{UN}\right|^2}{2}\sigma_z\, e^{-2i\theta\,\sigma_x} \Phi_{\perp}^{UN} \tag{12}$$

As initial conditions, we again choose the Pade asymptotic straight-line vortices (Berloff, 2004) for the uncoupled GP-equation. In Figure 3, we show 9 such non-overlapping vorticies.

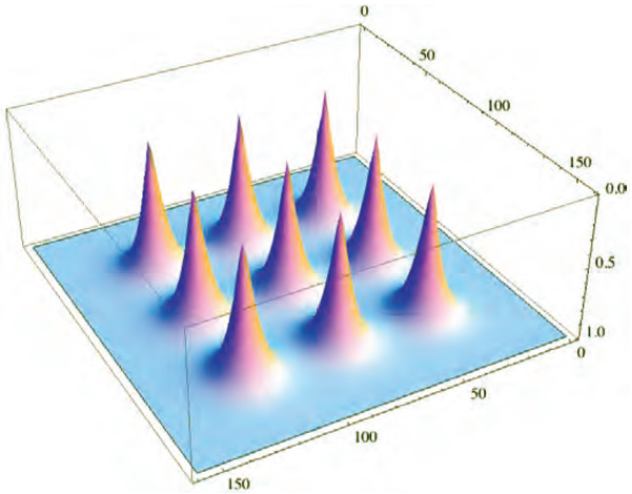


Figure 3. Initial straight- line vortices, with $|\varphi_v| \rightarrow 0$ at the vortex core. $|\varphi_v|$ is inverted and normalized to 1 asymptotically.

For the initial state of the Bogoliubov quasi-particles, we choose simple oscillations with varying periods. For example one choice we consider has its $|\Phi_{\perp R} + \Phi_{\perp L}|$ of the form shown in Figure 4.

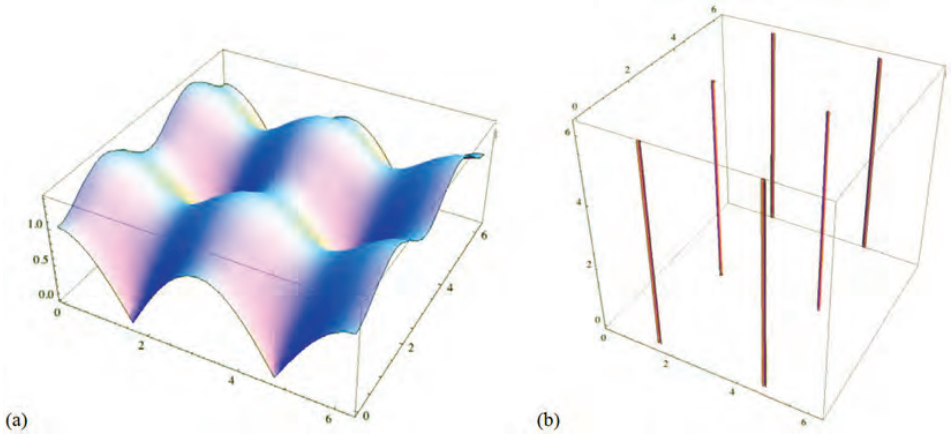


Figure 4. Initial Bogoliubov state $|\Phi_{\perp R} + \Phi_{\perp L}|$(a) as a function of (x,y), (b) the iso-surfaces near the core where $|\Phi_{\perp R} + \Phi_{\perp L}| \rightarrow 0$

For very weak interaction between the BEC and Bogoliubov states, there is no perceivable change in $|\Phi_{\perp R} + \Phi_{\perp L}|$ from its initial state. For stronger interactions we plot the effect of the Bogoliubov states on the ground-state BEC wave-function. At t=100, for the uncoupled systems (γ=0) there are only very small Kelvin waves excited along the vortex cores (purple surfaces in Figure 5), while for the coupled systems ($\gamma \neq 0$) there are a significant amount of Kelvin waves excited (green surfaces in Figure 5).

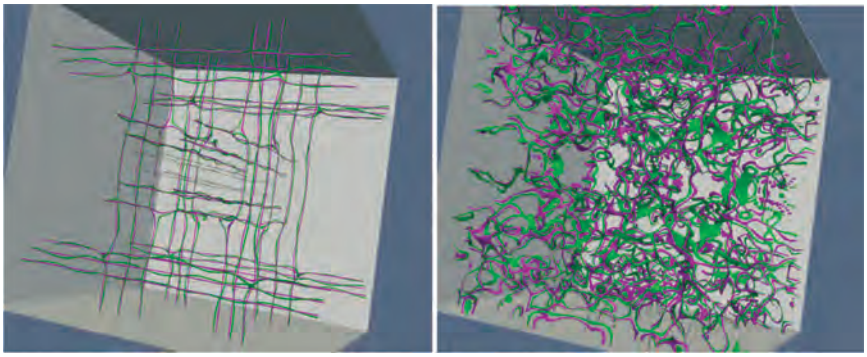


Figure 5. Effect of weak Bogoliubov states on the BEC ground-state wave-function vortex core iso-surfaces at t=100 and t=2,100. The evolution of the uncoupled BEC ground-state is shown in purple, while the coupled BEC-Bogoliubov system has vortex iso-surfaces shown in green.

We now consider a series of vortex iso-surface snapshots from a large-grid run ($2{,}048^3$), and contrast the effect of coupled ($\gamma \neq 0$) vs. uncoupled systems ($\gamma = 0$) on an initial set of 144 BEC quantum vortex cores.

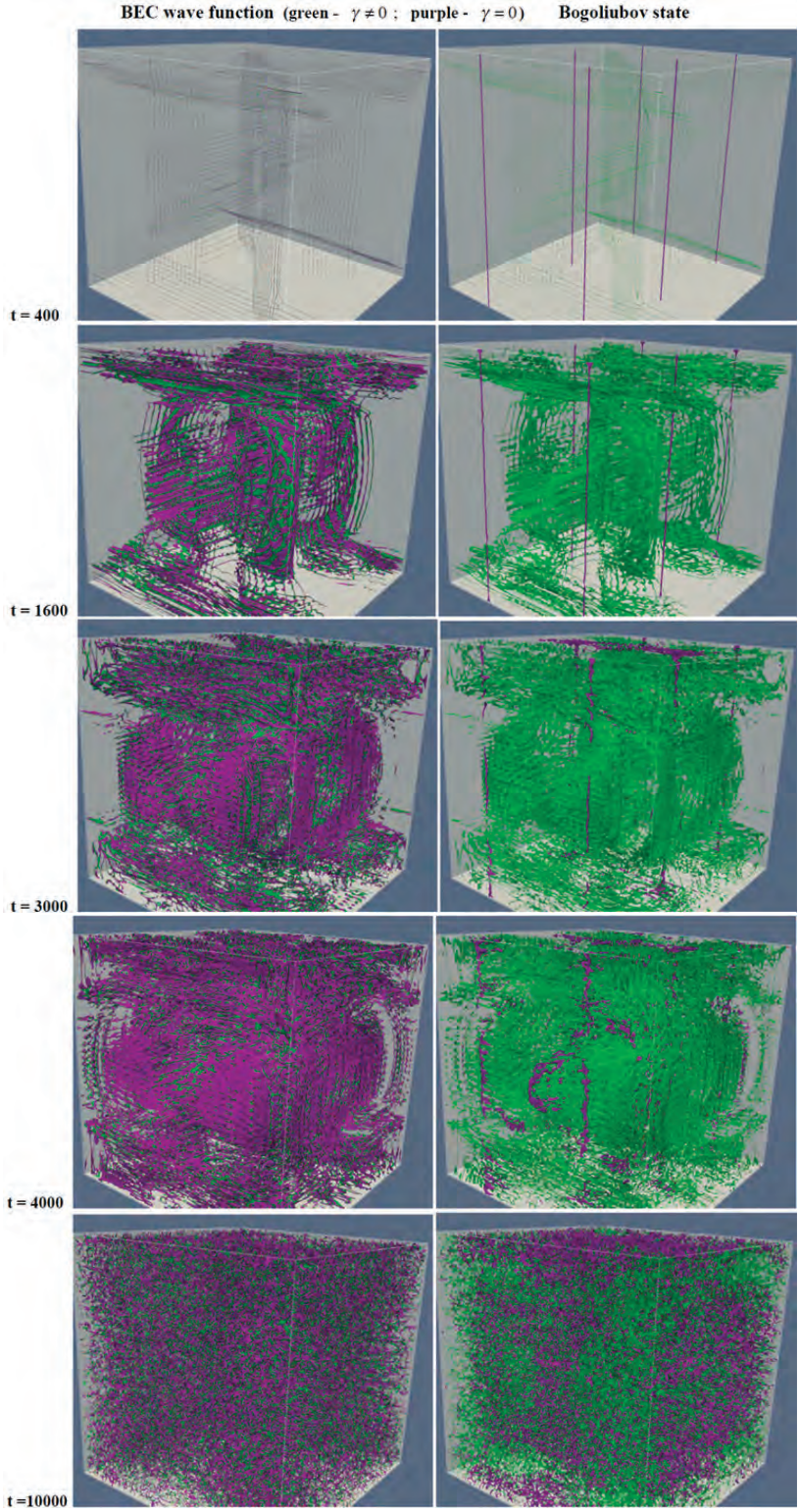


Figure 6. The evolution of the vortex core iso-surfaces for (left-panel of figures) the BEC ground-state and (right-panel of figures) the Bogoliubov state. The GREEN contours are for the coupled BdG model, Equation 10, and the PURPLE contours for the uncoupled system, Equation 11. The major differences appear in the Bogoliubov states. There also seems to be mode-locking for the coupled BdG system between the ground-state and the perturbed-state wave-functions.

At t=400, we notice that the initial uncoupled Bogoliubov state (purple) has not been affected in its evolution from its initial condition (c.f., Figure 4b). However, the coupled Bogoliubov state (green), appears to becoming mode-locked to the initial BEC ground-state wave-function. By t=3,000, the uncoupled Bogoliubov state (purple), is showing significant deviation from its initial state because of the effect of the (uncoupled) BEC ground-state wave-function in the evolution of the Bogoliubov state. This is readily seen in Equation 11 with the appearance of the terms $\Phi_{\perp}^{UN}$, φ_{v}^{UN} . There is a rapid onset of quantum turbulence.

5. Conclusion

We have continued our study of qubit unitary algorithms and have now extended them to include the physics of the interaction between the BEC ground-state and the Bogoliubov excited-state quasi-particles. These algorithms not only scale to all available processors (tested to 216,000 cores) on current-day classical supercomputers, but can be run on quantum computers as they become available. Building up such an arsenal of codes that will run on quantum computers is of great significance to Department of Defense (DoD).

Acknowledgments

This work was funded by the Air Force Office of Scientific Research (AFOSR) (Fariba Fahroo, Program Manager). The DoD HPC Modernization Program supported this project by supplying supercomputer time under the Capabilities Applications Program (CAP) on the Cray XE6 Garnet at the DoD Supercomputer Resource Center at ERDC. We are particularly grateful to the various staff members at ERDC for their 24/7 readiness (over the Christmas period) when things ran afoul with an untested racehorse. Thanks to Bob Alter, David Dumas, and the Cray Team.

References

Berloff, N., "Pade approximations of solitary-wave solutions of the Gross-Pitaevskii equation", *J. Phys. A*, 35, pp. 1617–1632, 2004.

Landau, L., "Theory of the superfluidity of helium II", *Phys. Rev.*, 60, pp. 356–358, 1941.

Pethick, C.J. and H. Smith, *Boise-Einstein Condensation in Dilute Gases*, Cambridge Univ. Press, 2008.

Tisza, L., "The theory of liquid helium", *Phys. Rev.*, 72, pp. 838–854, 1947.

Vahala, G., J. Yepez, L. Vahala, M. Soe, B. Zhang, and S. Ziegeler, "Poincare recurrence and intermittent loss of quantum Kelvin wave cascades in quantum turbulence", arXiv:1011.6334v1, 2010.

Yepez, J., "Quantum informational dynamics of a finite-temperature Fermi condensate", 2011 (to be published).

Yepez, J., G. Vahala, L. Vahala, and M. Soe, "Superfluid Turbulence from Quantum Kelvin Waves to Classical Kolmogorov Cascades", *Phys. Rev. Lett.*, 103, 084301, 2009; *Phys. Rev. Lett.*, 105, 129402, 2010.

Yepez, J., G. Vahala, and L. Vahala, "Quantum algorithm for Bose-Einstein condensate quantum fluid dynamics", *SPIE Quant. Info and Computat VII*, 7342, 73420M, 2009; arXiv:0905.0886, 2009.

Simulation of Rotorcraft Vortical Wakes Using an Adaptive Dual-Mesh Computational Paradigm

Nathan Hariharan and Michael Steffen
US Naval Air Systems Command (NAVAIR), Patuxent River, MD
nathan.hariharan.ctr@hpcmo.hpc.mil, mikesteffen@amewas.com

Mark Potsdam and Andy Wissink
US Army/AFDD, Ames Research Center, Moffett Field, CA
{mark.potsdam, andrew.m.wissink}@us.army.mil

Abstract

This paper describes the efforts to model vorticity laden flowfields using the CREATE-AV Helios platform. Helios employs a dual-mesh paradigm: near body unstructured grids, and high-order accurate off-body Cartesian grids, and information exchange is facilitated by an automated, implicit hole cutting method. Further, an automatic mesh refinement capability is employed to refine regions of intense vorticity. The simulation of the helical wake of a model rotor in hover is considered in this study. The ease of use, efficiency, and power of the Helios dual-mesh paradigm is demonstrated through high-fidelity solutions for the aforementioned unsteady, vortical field.

1. Introduction

Vortical wakes introduce important aerodynamic phenomena in certain classes of aerospace vehicles. Rotary-wing vehicles, in particular, fly in their own wake and experience numerous aerodynamic effects, affecting vehicle handling qualities, vibration, and noise. The wake of the vehicles also complicates near-ground operations, from shipboard landings to "brownout" conditions in desert flight. Fixed-wing aircraft also experience problems such as tail buffet[1] from tip vortices emanating from the nose and swept wing in high angle-of-attack fighter jets. The availability of accurate and efficient computational models to better predict vortex wake behavior could help to minimize the onset of these sometimes disastrous phenomena.

High-fidelity Reynolds-averaged Navier-Stokes (RANS) computational fluid dynamics (CFD) methods have demonstrated the ability to give accurate predictions of aerodynamic loads, but their ability to predict the wake is often limited by numerical dissipation. This can be mitigated by using very fine grids in the wake, but this quickly exhausts available computational resources. For problems where key solution features, like tip vortices, occur only in localized regions of the computational domain, spatial adaptive mesh refinement (AMR) can be an effective tool. AMR involves automatically refining and coarsening the grid locally to resolve important flow features. By focusing memory usage and computational effort in these localized regions, a highly-resolved solution may be obtained much more efficiently than a globally-refined grid of equal resolution.

The use of AMR has been studied extensively for wakes of hovering rotors. Strawn and Barth[2] first demonstrated the concept using AMR in an unstructured Euler solver. Potsdam[3] also applied unstructured AMR to wind turbine wake predictions. Deitz, et al.[4] introduced an overset-grid based approach that moved tubular curvilinear grids to align with the tip vortices. Hariharan[5] used Cartesian overset vortex-grids to analyze tip vortex from wings. Meakin[6] proposed a Cartesian-based AMR scheme within the Overflow[7] code. This approach was recently extended by Holst and Pulliam[8,9]. The aforementioned efforts all adopted AMR techniques targeting steady-state solutions; a solution is computed on an initial mesh, the mesh is reconstructed to adapt to features in the solution, then the simulation is run again on the new mesh. This steady-AMR approach is useful for isolated rotors in hover conditions, but many of the more complex problems in rotorcraft require time-dependent moving body capabilities. An AMR scheme that can resolve unsteady effects like rotor-fuselage interactions or rotor vehicles in forward flight or maneuver conditions requires an unsteady-AMR approach, for which the grid is adapted continually in a time dependent manner throughout the simulation.

Another important strategy for CFD-based wake resolution is the use of high-order numerics. High-order schemes are effective because they resolve features like tip vortices with much fewer points across the vortex core, so it is possible to

achieve better resolution of the vortex wake with a coarser mesh. The benefits of high-order schemes for rotor wakes have been shown by Hariharan[4,10], Sankar[11], Yeshala[12], and Duque, et al.[13]. Cartesian grids offer the most computationally efficient approach to high-order accuracy. Unlike Discontinuous Galerkin schemes on unstructured grids, which can be as much as an order-of-magnitude more expensive than standard second order schemes, finite-difference-based high-order schemes on Cartesian grids are only marginally more expensive than a second-order scheme.

2. CREATE-AV Helios Computing Platform

Helios is a new computational platform under the CREATE-AV umbrella targeted towards aero-mechanics simulations. Details of the underlying platform design, validational cases, and targeted applications can be found in Reference 14. At the heart of the Helios platform is an innovative dual-mesh paradigm with an unstructured mesh solution in the near-body, and Cartesian meshes in the off-body and using automated overset information exchange as a means of communication. The unstructured near-body solver facilitates ease of grid generation for complex body shapes, and the solution is computed using the second-order accurate flow solver NSU3D[15]. The off-body Cartesian grid generation is automatically done using Structured Adaptive Mesh Refinement Application Infrastructure (SAMRAI)[16]. Cartesian grid infrastructure, and employs an efficient fifth-order spatially accurate solver, SAMARC[17].

The Helios platform deploys overset hole cutting using an implicit methodology[18] that does not require any user interference. The overset connectivity is handled by PUNDIT[19], and the entire process supports parallel/distributed computation. Figure 1.1 illustrates the various components of the Helios platform. Further, the SAMRAI infrastructure supports the ability automatically to perform Cartesian refinement and adapt to geometric and flow features. The CREATE-AV release of Helios (named "Whitney") does not support the automated refinement functionality, but the 2011 release ("Shasta") will officially support the feature.

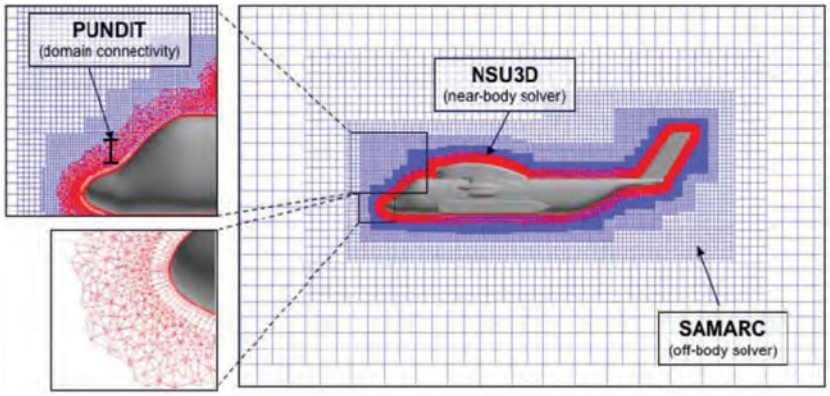


Figure 1.1. Dual-mesh paradigm used in the Helios platform with unstructured near-body grids to capture geometric features and boundary layer near the body surface, and block-structured Cartesian grids to capture far-field flow features

In this paper, the application of the Helios dual-mesh paradigm—with automated adaption—is explored for resolving the vorticity-laden flowfield of a rotor in hover.

3. Vortex-Wake of a Rotor in Hover

This study was focused upon the physics and numerics of computing the vortical wake of a rotor in hover. The Caradonna-Tung[22] rotor-blade—a two-bladed rotor model tested in 1981—was used to explore several aspects of predicting hover using Helios. The blade planform consists of untwisted, non-tapered NACA0012 sections with an aspect ratio of 6. An unstructured blade grid was generated using AFLR3 volume grid generator.

To initiate the calculations the blade grid was set in a prescribed Cartesian grid setting. The finest grid level was initiated at 9-grid-cells per chord (referred to as Level-5 grid). The background Cartesian grid was intentionally solved employing a second-order spatially-accurate method. The Helios code was run in the rotational-mode. The run collective pitch was set at 8 degrees, and the tip Mach number was 0.44. The blade grid had 3.3 million grid points, and the coarse, Level-5 off-body grid had 2.2 million grid points. Figure 3.1 shows vorticity contours across a chordwise cross-section in the middle of the middle of the blade. The tip vortex originates in the blade-grid, and transfers to the Cartesian grid. The large amounts of diffusion associated with the second-order scheme and coarser Cartesian grid causes the vortex system to settle-down

to a ring-state within a single-revolution. As the computation proceeds, the solution arrives at a numerically stable vortex structure as seen in Figure 3.2. The presence of the cumulative vortex ring so close the rotor-blade results in much larger induced flow, and hence a much lower predicted thrust coefficient of 0.003118 (experimental thrust coefficient=0.0046).

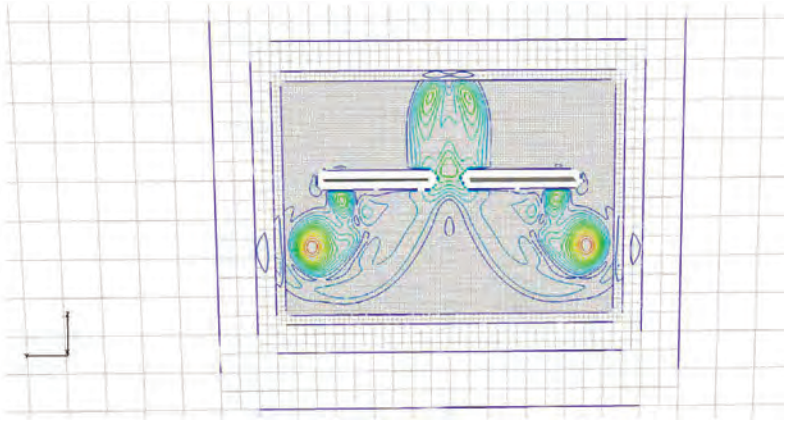


Figure 3.1. Vorticity magnitude contours;Cartesian: Level-5, 2nd-order

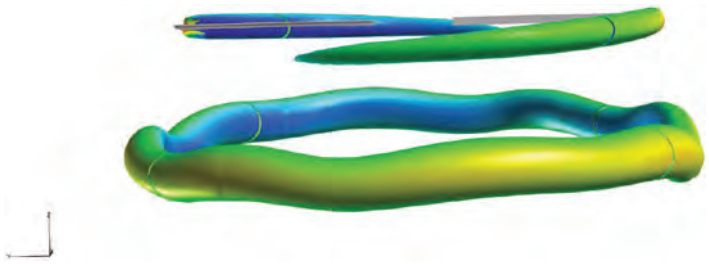


Figure 3.2. Iso-vorticity contours colored by z-velocity; Cartesian: Level-5, 2nd-order

The background Cartesian grid finest-level resolution was doubled to 18-grid-cells per chord (referred to as Level-6 grid). The off-body Cartesian grid count was 22.5 million. Figure 3.3(a, b) shows similar vorticity magnitude contours, and vorticity iso-surfaces (colored by z-velocity) with the computational order of accuracy in the Cartesian grid still being retained at the second-order level. The wake system evolves below the vortex further down before being consumed by the ring-vortex. Figure 3.4 shows on vorticity iso-surfaces focusing on the structure of the wake-sheet. The helical wake sheet system that descends faster than the tip-vortex system is seen to be well captured.

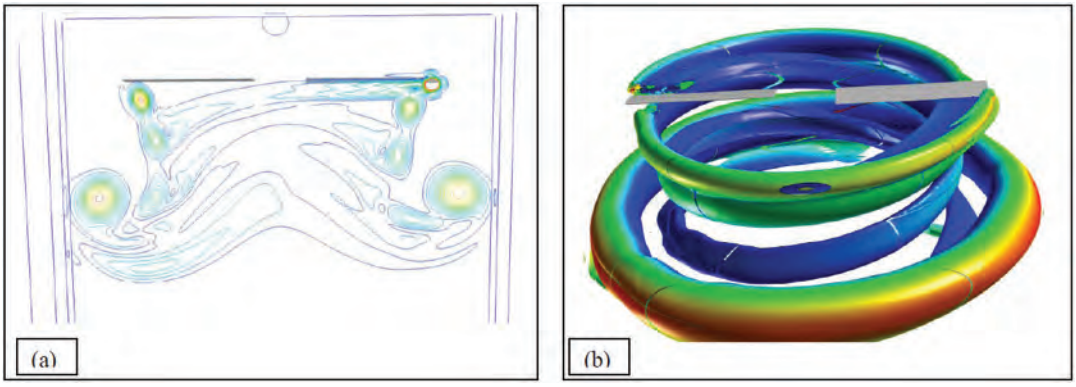


Figure 3.3. Vorticity magnitude and Iso-vorticity contours colored by z-velocity; Cartesian: Level-6, 2nd-order

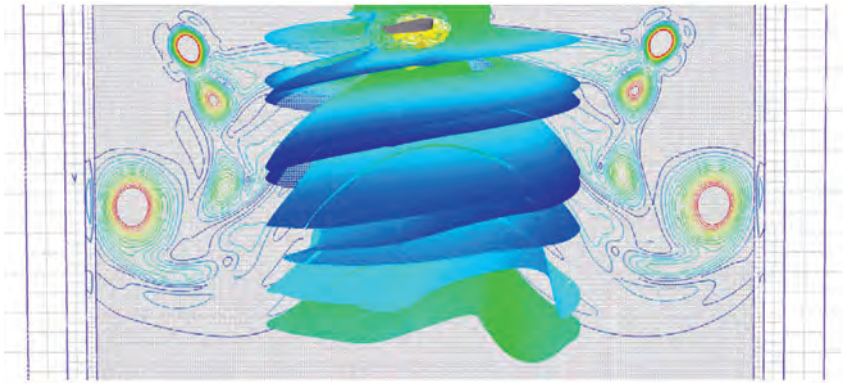


Figure 3.4. Vorticity magnitude and Iso-vorticity contours colored by z-velocity, showing wake-vortex sheet structure; Cartesian: Level-6, 2nd-order

Figure 3.5 shows wake vorticity contours when the computational order of accuracy of the Cartesian grid was increased to 5th. The wake vortex system is sharply defined with the vortex-ring structure pushed further down, but persisting in the eventual converged solution. Figure 3.6 (a,b) show two different views of helical tip vortex structure using iso-surfaces in the wake (colored by z-velocity). The helical system convects down due to its own self-induced velocity, but eventually rolls up into a toroidal vortex ring. Figure 3.7(a) shows vorticity iso-surfaces of both the computed tip-vortex, and the wake-sheet systems. Figure 3.7(b) shows the well-known idealized schematic of a rotor-wake. The agreement between figures 3.7(a) and 3.7(b) is striking. The computed thrust coefficient converged to a value of 0.0052, approximately 10% higher than the experimental value.

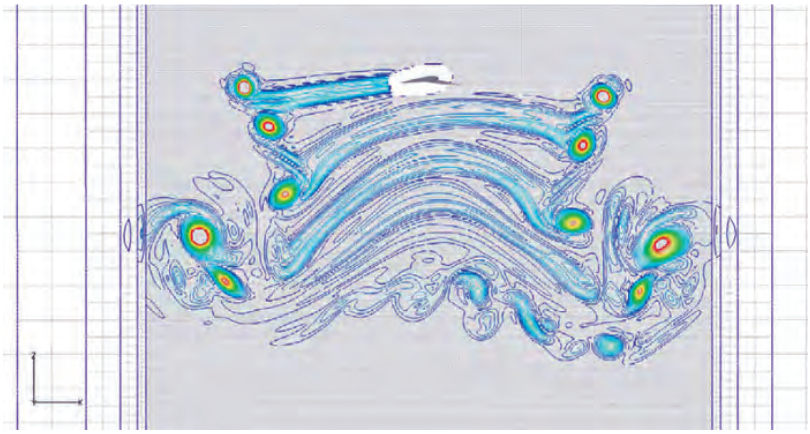


Figure 3.5. Vorticity magnitude contours in the wake, showing wake-vortex sheet structure; Cartesian: Level-6, 5th-order

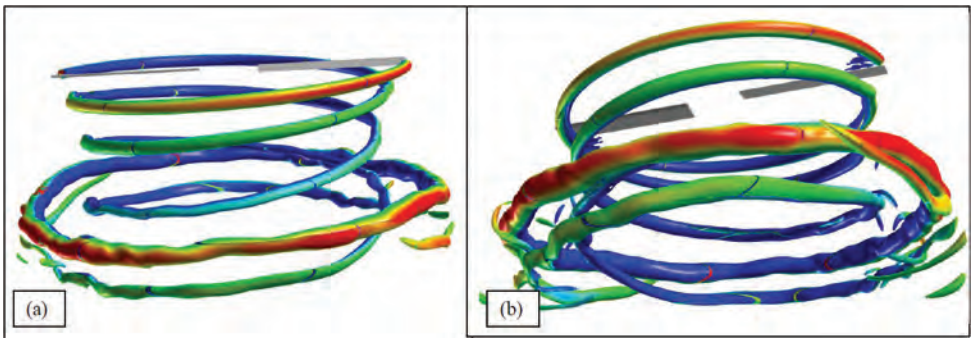


Figure 3.6. Vorticity Iso-vorticity contours colored by z-velocity, showing tip vortex structure; Cartesian: Level-6, 5th-order

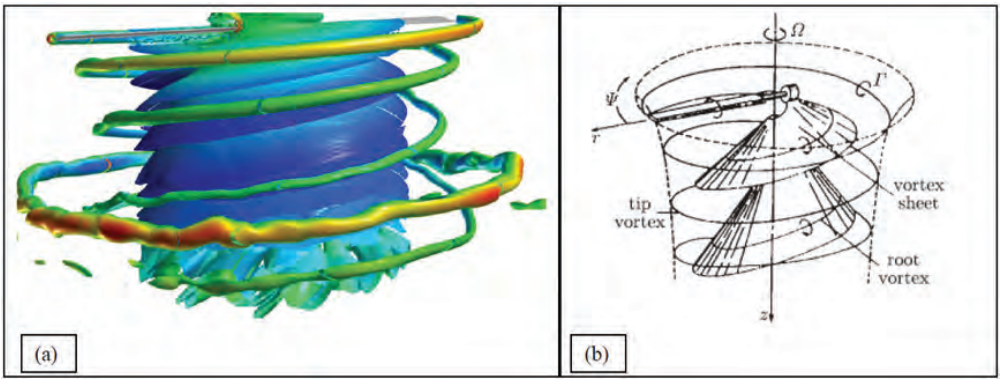


Figure 3.7. Vorticity Iso-vorticity contours colored by z-velocity, showing the helical wake structure comprising of the tip vortex, and the vortex sheet structure; Cartesian: Level-6, 5th-order

There are several possible reasons on why the thrust comes out higher: i) accuracy of the near-body solution, ii) effect of center-body, and iii) far-field conditions (wind-tunnel vs. free-stream), etc. We explored some of these issues, starting with the near-body solution. In the Helios dual-mesh paradigm, the near-body solution is computed using NSU3D, a second-order spatially accurate unstructured solver. Figure 3.8(a) shows a streamwise cross-section of the baseline near-body unstructured grid (R0: 3.3 million grid points). A grid refinement tool developed by the CREATE-AV Kestrel team was used to refine the tip region of the rotor-blade grid. Figure 3.8(b) shows the new blade grid (R1: 8.3 million grid points), refined to double the grid density in the tip region. The resulting solution improves the resolution of the tip vortex formation over the blade-tip. Figure 3.9(a, b) compares the tangential peak-to-peak velocity across the tip-vortex at a streamwise station near the blade trailing edge, between the Level-0, and Level-1 grid solutions. The peak-to-peak tangential velocity magnitude from the Level-1 simulation is approximately 30% more than the Level-0 value. Figure 3.10 compares resulting wake vorticity in the Cartesian grid between the two simulations (R0: colored by z-velocity, R1: black grid surface), and no significant difference in the wake location is noted. The computed thrust coefficient from the R1 simulation converged to a value of 0.0050, trending towards the experimental value.

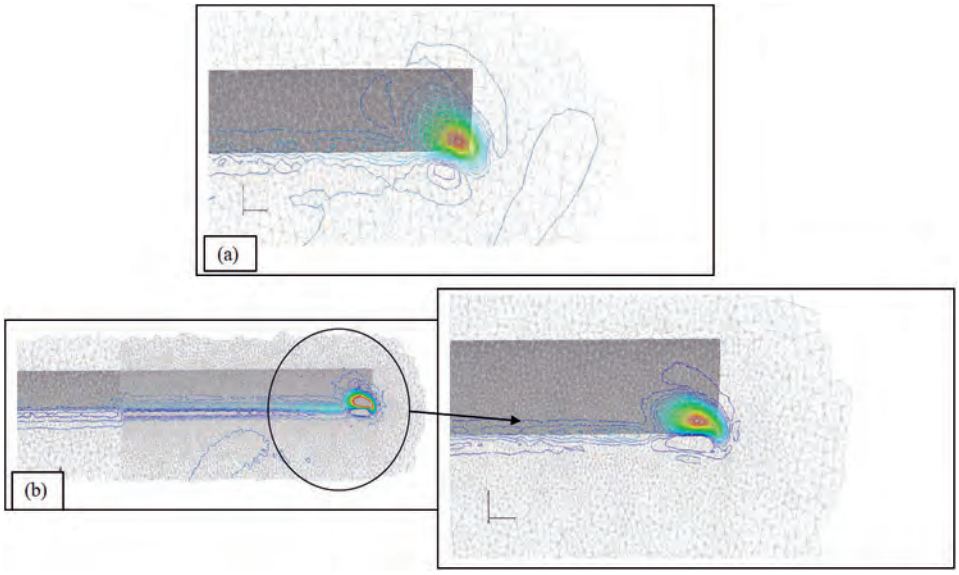


Figure 3.8. Vorticity contours in a streamwise plane off the blade trailing edge. Solution in the near-body grid, 2nd-order accuracy: (a) Level-0 grid (b) Level-1 refined grid.

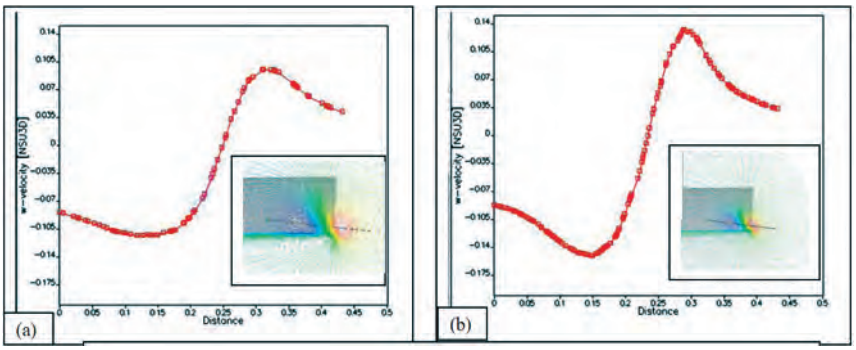


Figure 3.9. Tangential velocity profile across the tip vortex in the NBE grid (inset: w-velocity contours in a streamwise plane off the blade trailing edge). Solution in the near-body grid, 2nd-order accuracy: (a) Level-0 grid (b) Level-1 refined grid.

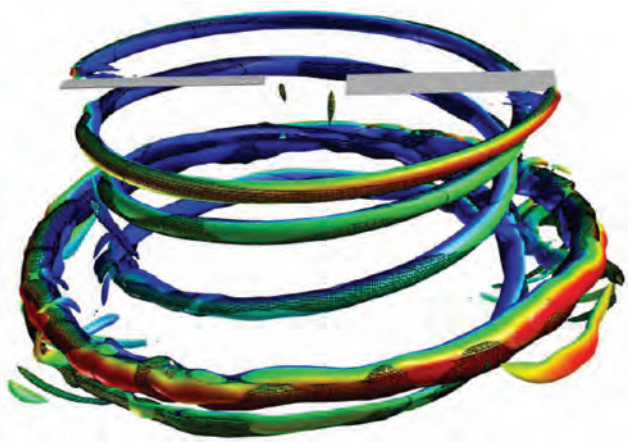


Figure 3.10. Vorticity iso-surface contours. Cartesian solutions from NBE-R0 (colored by z-velocity), and NBE-R1 (black grid lines) grids overlaid on top of each other.

Concurrently, the study started focusing on automated mesh refinement (AMR) of the Cartesian solution. In the AMR method, a maximum adaption cell size, and a minimum vorticity threshold to adapt to are prescribed. Figure 3.11 shows vorticity contours, and the resulting grid for a vorticity threshold of 0.005, and maximum grid level of 6 (18 cells per chord). The vorticity threshold is small enough to include refinement in the entire wake. Figure 3.12 compares the wake vortex peak-to-peak strength at several azimuthal angles (Ψ=30, 210, 390, 570 degrees). The number of nodes the peak-to-peak variation is capture is plotted for each of the azimuthal angles. The strength drops off from Ψ=30 degrees where the vortex is captured with ~9 cells till the point it spreads to ~12 cells across (~Ψ=210 degrees), and thereafter maintains the resolution. This is in line with expectations for a fifth-order spatially accurate scheme.

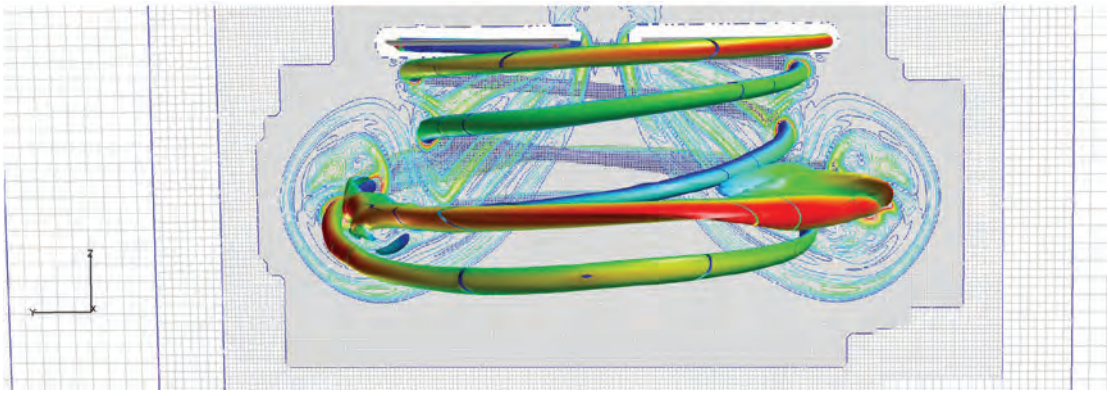


Figure 3.11. Vorticity iso-surface contours colored by z-velocity. Automated Cartesian Mesh Refinement applied (Maximum Level-6), vorticity threshold=0.005.

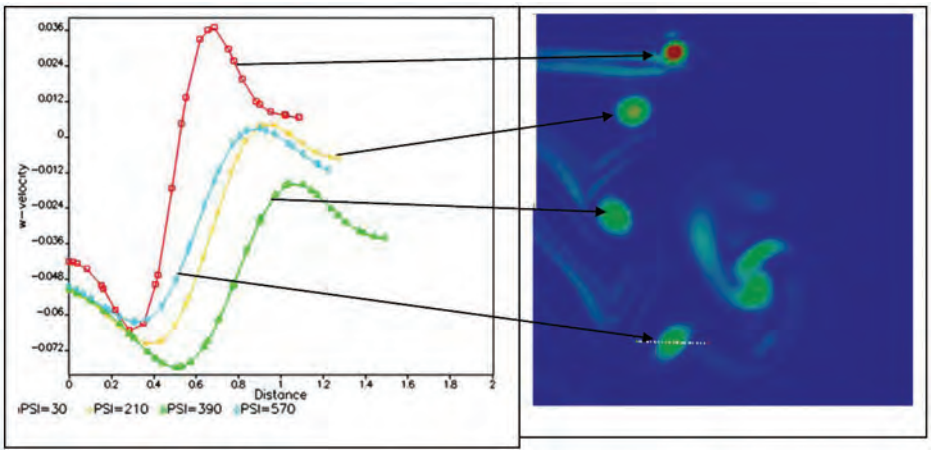


Figure 3.12. Tangential velocity profile across the wake tip vortex at several wake-age stations

It is instructive to compare the current simulation from an earlier, 7th-order spatially accurate, overset grid simulation by Hariharan and Ekaterinaris[23] using the NSHOCS code. Figure 3.13(a) shows a near-body C-grid, and off-body clustered Cartesian grid. The solution was run in a fully transient mode, for the same flow conditions, and both near and off-body grids were computed using a ENO-based 7th-order accurate scheme. Figure 3.13(b) shows the wake vortex structure, and the simulation predicts a similar vortex ring as seen in the current Helios simulations. Figure 3.14 shows tangential peak-to-peak variations across various tip-vortex azimuthal locations. The peak-to-peak values compare similar to the Helios wake structure in Figure 3.12. However, the computed thrust coefficient in the Hariharan and Ekaterinaris[23] simulation was reported at a value of 0.00462—very close to the experimental thrust coefficient. The only substantial difference between the NSHOCS simulation and the current Helios simulation is the computational accuracy of the near-body computation. Coupled with the fact that the R1 Helios solution trended in the right direction compared to the R0 Helios solution, it is quite suggestive that the near-body flow solution accuracy (especially the creation of the tip vortex and its evolution over the wing) plays a large role in closing the last 10% gap in computed vs. experimental loads. In Helios, this increase in near-body tip-flowfield accuracy could be achieved either by further near-body tip-grid refinement, or adding a higher-order structured/strand grid near-body solution option—both of which are in the works for future releases.

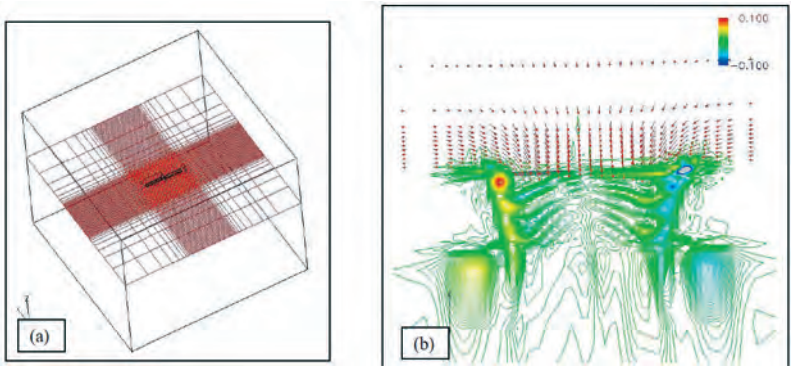


Figure 3.13. Fully high-ordered structured overset simulation of the C-T rotor. Near-body C-grids, Cartesian background grid. Seventh-order spatial accuracy (Hariharan and Ekaterinaris[23]).

Thus far, all the rotor-wake simulations presented in this section utilized the steady hover formulation, which uses a fixed grid with rotational source terms applied to the equations solved on the grid, and is usually accepted as a good approximation for isolated hover predictions. However, problems involving rotor-fuselage interactions, multiple rotors, or helicopters in forward flight which are ultimately of interest to helicopter engineers require an inertial formulation with moving grids capable of simulating bodies in relative motion. A useful validation is an investigation into whether the two formulations—steady hover vs. inertial hover—give the same answers. Figure 3.15(a) shows the tip vortex wake structure from an inertial simulation (R0 near-body, L6-Level Cartesian, 5th-order accuracy in the Cartesian), and it exhibits the helical structure feeding a persistent vortex ring similar to the rotational solution. Figure 3.15(b) plots the tip-vortex iso-surfaces for the inertial (colored by z-velocity) and the earlier rotational solution (grayscale). The vortex-ring stands off at

a slightly different z-location, and the helical vortex structure z-locations vary a bit after the wake azimuthal angle exceeds 360 degrees. However, the net thrust coefficient levels for the inertial simulation converge to a value of 0.00518, close to the rotational mode predictions. Further, a tear-drop shaped center-body was inserted into the simulation. Figure 3.16 shows the tip vortex of the resulting system. No major changes were observed either in the vortex structure or the resulting integrated thrust.

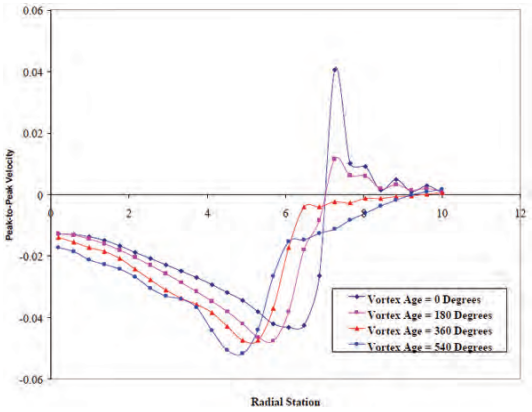


Figure 3.14. Fully high-ordered structured overset simulation of the C-T rotor. Near-body C-grids, Cartesian background grid. Seventh-order spatial accuracy (Hariharan and Ekaterinaris[23]).

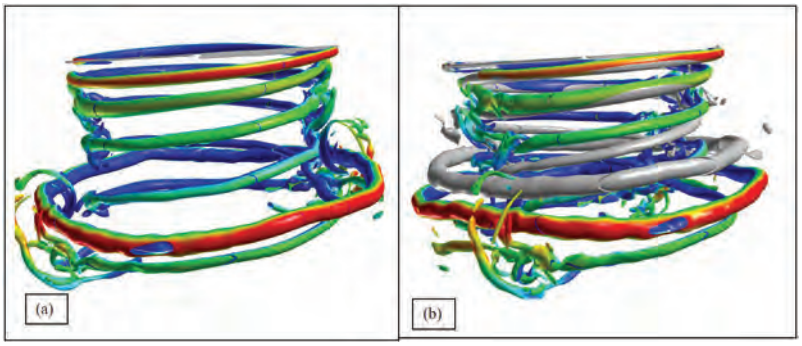


Figure 3.15. Vorticity magnitude iso-surface colored by z-velocity. Level-6: (a) Inertial, (b) Inertial (colored by z-velocity) vs. Rotational (grey).

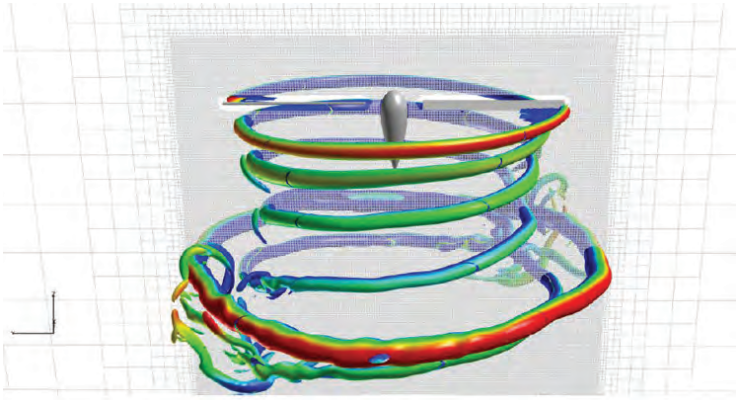


Figure 3.16. Vorticity magnitude iso-surface colored by z-velocity. Level-6: Inertial run with a tear-drop.

In all the simulations cited so far, the rolled-up vortex ring sits beneath the rotor-blade within 2–3 revolutions. It was conjectured that this was caused by positioning the far-field boundaries relatively close to the blade. Figure 3.17 shows a modified simulation in which the far-field was extended further out, and the transition between successively coarser Cartesian grids was made smoother. The vortex-ring structure was blown down further. Figure 3.18 shows the vorticity

contours for a similar setting, but now employing AMR to track the tip vortex rather than fixed refinement. The tracking was based upon vorticity, and seems to capture fine wake features. In either of these simulations, the integrated loads did not change much, coming out at a value of CT=0.0052, still approximately 10% higher.

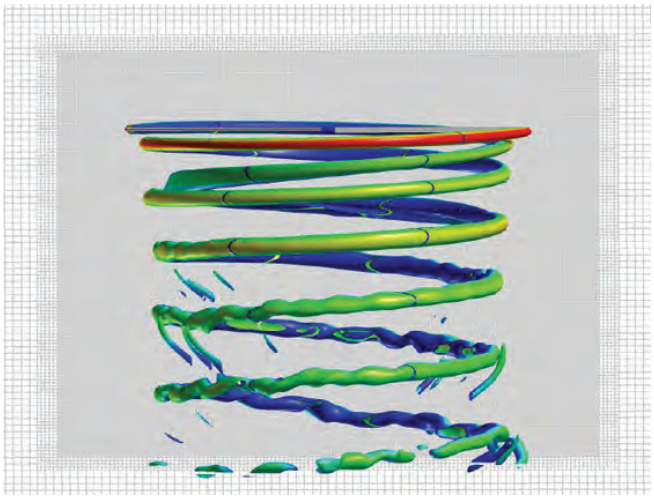


Figure 3.17. Vorticity magnitude iso-surface colored by z-velocity. Level-6: Far-field-boundaries extended out further.

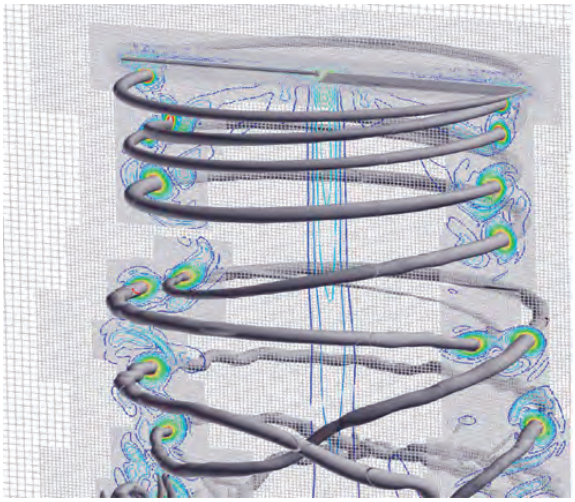


Figure 3.18. Vorticity magnitude iso-surface colored by z-velocity. AMR resolution of the wake.

As a final attempt, the near-body unstructured grid was redone from scratch, ensuring much higher tip density (a total of approximately 5 million grid points), shown in Figure 3.19. Figure 3.20 shows the vorticity contours and is similar to the earlier results. However, the integrated CT was computed at a value of 0.0048, less than 5% off from the experimental results. Figure 3.21 plots peak-to-peak tangential velocity variation across the tip vortex just off the trailing edge, similar to Figure 3.9. The peak strength is captured tighter compared to the earlier simulation (Figure 3.9), and it correlates with better integrated load predictions. The tip refined simulation in Figure 3.9b, and the current simulation had approximately the same number of grid points across the vortex core. However, it seems that the quality of grid cells obtained by splitting cells has an effect on the captured vortex fidelity.

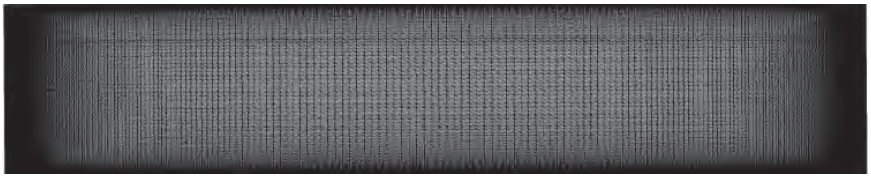


Figure 3.19. Tip refined near body surface grid

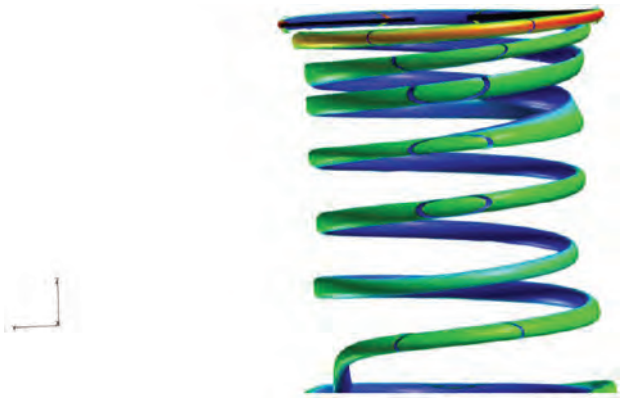


Figure 3.20. Vorticity magnitude iso-surface colored by z-velocity. Near-body refined grid.

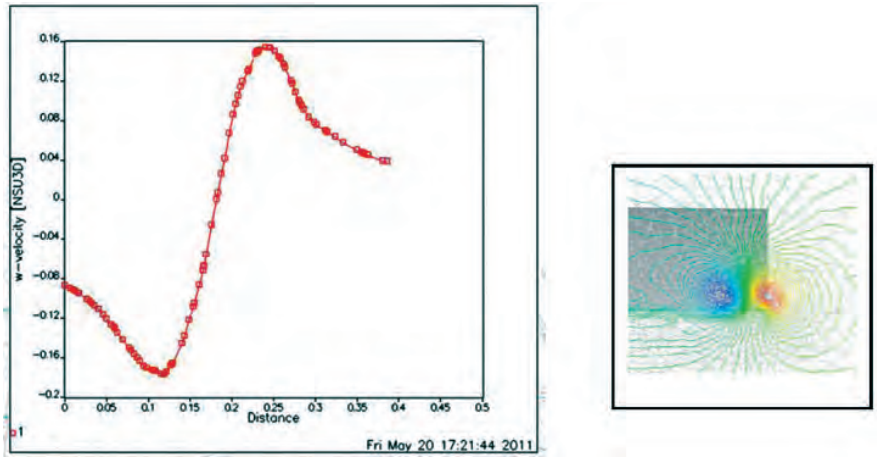


Figure 3.21. Peak-to-peak tangential velocity profile across the tip vortex near the blade TE

4. Conclusions & Recommendations

This paper describes the efforts to model vorticity-laden flowfields using the CREATE-AV Helios platform. Helios employs a dual-mesh paradigm: near-body unstructured grids, and high-order accurate off-body Cartesian grids, and information exchange is facilitated by an automated implicit hole cutting method. Further, an automatic mesh refinement capability within Helios was employed to refine regions of intense vorticity. The helical wake of a rotor in hover was simulated. The ease of use, efficiency, and power of the Helios dual-mesh paradigm was demonstrated through high fidelity solutions for the aforementioned unsteady, vortical fields.

The geometric complexities of a full-up aircraft with all the associated pylons, and missiles, and pods require flexible and efficient CFD simulation tools that utilize unstructured meshes. One of the charters of CREATE-AV is to propagate the use of Computational-Based Engineering (CBE) to non-CFD experts, and therefore the process utilized to arrive at the solution cannot be too dependent on fine-tuned expertise of the user. Hence, the right process to arrive at a good-quality engineering solution becomes equally important.

A combination of Helios capabilities, i) to compute 5th-order spatial accuracy in the Cartesian grids, and ii) to automatically refine the Cartesian grid to vortical flow features, enabled accurate convection of the vortex structures. A central advantage in this approach is the ability of the method to automatically adapt the background Cartesian grid with changing flight conditions without having to go back and regenerate the near-body unstructured grid. Such an ability to preserve vortical fluctuations over multiple-aircraft lengths will also be very useful to study interactional aerodynamics of multi-aircrafts in flight (i.e., mid-air refueling, close-proximity flights, impact of engine jet-exhaust, etc.).

The off-body Cartesian solver in Helios resolved the helical tip-vortex and wake sheet of the hover field adequately. The Cartesian grid resolution needs to be of the order of ~12 points across the vortex core for propagating the tip-vortex without noticeable dissipation. AMR needs to be further explored to refine just around the helical wake structure. For a blunt tipped blade such as the Caradonna-Tung model rotor, the near-body second-order solver resolution needs further refinement in order to resolve the high tangential and axial gradients across the tip vortex as the vortex forms over the wing. This lack of adequate grid resolution near the blade tip to capture the vortex formation has a definite bearing on the final predicted thrust.

Acknowledgements

Material presented in this paper is a product of the CREATE-AV Element of the Computational Research and Engineering for Acquisition Tools and Environments (CREATE) Program sponsored by the US Department of Defense HPC Modernization Program. Dr. Robert Meakin is the Program Manager for CREATE-AV.

References

1. Morton, S., R. Cummings, and D. Kholodar, "High Resolution Turbulence Treatment of F/A-18 Tail Buffet", *AIAA-2004-1676*, 45th AIAA Structures, Structural Dynamics and Materials Conference, Palm Springs, CA, 2004.

2. Strawn, R.C. and T.J. Barth, "A finite-volume Euler solver for computing rotary-wing aerodynamics on unstructured meshes", *Journal of the American Helicopter Society*, Vol. 38, pp. 61–67, 1993.

3. Potsdam, M. and D.J. Mavriplis, "Unstructured Mesh CFD Aerodynamic Analysis of the NREL Phase VI Rotor", *AIAA-2009-1221*, 47th AIAA Aerosciences Meeting, Orlando, FL, January 2009.

4. Dietz, M., E. Karmer, and S. Wagner, "Tip Vortex Conservation on a Main Rotor in Slow Descent Flight Using Vortex-Adapted Chimera Grids", *AIAA-2006-3478*, 24th AIAA Applied Aerodynamics Conference, San Francisco, CA, June 2006.

5. Hariharan, N., "Rotary-Wing Wake Capturing: High Order Schemes Towards Minimizing Numerical Vortex Dissipation", *AIAA Journal of Aircraft*, Vol. 39, No. 5, pp. 822–830, 2002.

6. Meakin, R.L., "Automatic Off-body Grid Generation for Domains of Arbitrary Size", *AIAA-2001-2536*, 15th AIAA Computational Fluid Dynamics Conference, Anaheim, CA, June 2001.

7. Buning, P.G., et al., *OVERFLOW User's Manual, Version 1.8*, NASA Langley Research Center, 1998.

8. Holst, T. and T. Pulliam, "Overset Solution Adaptive Grid Approach Applied to Hovering Rotorcraft Flows", *AIAA-2009-3519*, 27th AIAA Applied Aerodynamics Conference, San Antonio, TX, June 2009.

9. Holst, T. and T. Pulliam, "Optimization of Overset Solution Adaptive Grids for Hovering Rotorcraft Flows", 2010 AHS Specialists Meeting on Aeromechanics, San Francisco, CA, January 2010.

10. Hariharan, N., and L. Sankar, "High-Order Essentially Non-oscillatory Schemes for Rotary-Wing Wake Computations", *Journal of Aircraft*, Vol. 41, No. 2, pp. 258–267, 2004.

11. Sankar, L., N. Yeshala, and N. Hariharan, "Application of Spatially High Order Adaptive Methods for Unsteady Flow over Rotary Wing Configurations", *Paper No. 21-1*, International Forum on Rotorcraft Multidisciplinary Technology, American Helicopter Society Specialists Meeting, Seoul, Korea, October 2007.

12. Yeshala, N., A.T. Egolf, R. Vasilescu, and L. Sankar, "Application of Higher-Order Spatially Accurate Schemes to Rotors in Hover", *AIAA Paper No. 2006-2818*, 24th AIAA Applied Aerodynamics Conference, San Francisco, CA, June 2006.

13. Duque, E.L., et al., "Revolutionary Physics-based Design Tools for Quiet Helicopters", *AIAA Paper 2006-1068*, 44th AIAA Aerospace Sciences Meeting, Reno, NV, January 2006.

14. Sankaran, V., et al., "Application of the Helios Computational Platform to Rotorcraft Flowfields", *AIAA-2010-1230*, 48th AIAA Aerospace Science Meeting, Orlando, FL, January 2010.

15. Mavriplis, D.J. and V. Venkatakrishnan, "A Unified Multigrid Solver for the Navier-Stokes Equations on Mixed Element Meshes", *International Journal for Computational Fluid Dynamics*, Vol. 8, pp. 247–263, 1997.

16. Hornung, R.D., A.M. Wissink, and S.R. Kohn, "Managing Complex Data and Geometry in Parallel Structured AMR Applications," *Engineering with Computers*, Vol. 22, No. 3–4, pp. 181–195, December 2006

17. Wissink, A.M., S.J. Kamkar, J. Sitaraman, and V. Sankaran, "Cartesian Adaptive Mesh Refinement for Rotorcraft Wake Resolution", 28th Applied Aerodynamics Conference, June 2010.

18. Lee, Y. and J. Baeder, "Implicit Hole Cutting – A New Approach to Overset Connectivity", *AIAA-2003-4128*, 16th AIAA CFD Conference, Orlando, FL, June 2003.

19. Sitaraman, J., M. Floros, A.M. Wissink, and M. Potsdam, "Parallel Unsteady Overset Mesh Methodology for Unsteady Flow Computations Using Overlapping and Adaptive Cartesian Grids", *Journal of Computational Physics*, Volume 229, Issue 12, pp. 4703–4723, June 2010.

20. Hariharan, N., A. Wissink, J. Hunt, and V. Sankaran, "A Dual-Mesh Simulation Strategy for Improved AV-8B Aft-Fuselage Buffet Load Prediction", *AIAA 2010-1234*, 48th AIAA Aerospace Sciences Meeting and Exposition, Orlando, FL, January 2010.

21. Kamkar, S., et al., "Automated Grid Refinement Using Feature Detection," *AIAA 2009-1496*, 47th AIAA Aerospace Sciences Meet, Orlando, FL, January 2010.

22. Caradonna, F.X. and C. Tung, "Experimental and Analytical Studies of a Model Helicopter Rotor in Hover", *NASA TM 81232*, 1981.

23. Hariharan, N. and Ekaterinaris, J., et al., "An Evaluation of High Order Spatial Accuracy Algorithms for Modeling Fixed and Rotary Wing Tip Regions," AHS Aerodynamics, Acoustics, and Evaluation Technical Specialists Meeting, San Francisco, CA, January 2002.

Use of Detailed Flow Computations in the Design of Propulsion Systems for Small UAV's

Surya P.G. Dinavahi
Lockheed Martin Company, Aberdeen Proving Ground, MD
surya.dinavahi@us.army.mil

Rajneesh Singh
US Army Research Laboratory (ARL), Vehicle Technology Directorate, Aberdeen Proving Ground, MD
rajneesh.singh@us.army.mil

Abstract

In this paper, we look at the feasibility of using physics-based models to represent the rotor of a ducted rotor in hover. A straight duct with zero expansion and an elliptic inlet is chosen for our study. Reynolds-averaged Navier-Stokes (RANS) equations with two-equation realizable k-ε turbulence model is used to compute the flow through the duct. The commercial software CFD++ is used in our study. Blade element theory is used to model the rotor. The detailed and the model computations are carried out on structured and unstructured grids. The two contributions to the thrust—one from the rotor itself and the other due to the duct, are computed. In addition, we also computed the flow past the open rotor to obtain the thrust due to the rotor itself. It is observed that the physics-based model over-predicts both the thrust components. However, this level of accuracy should be sufficient during the initial stages of a design study where the relative merits of several designs are compared.

1. Introduction

Of late, there has been an increased interest in the development of rotors for small unmanned aerial vehicles (UAVs). The performance of a rotor can be augmented by enclosing the rotor in a shroud or a duct. The shroud provides additional benefits of noise abatement and protection for both the rotor blades and the operator. These characteristics make a ducted-rotor a good choice as the propulsion device or the platform for an aerial vehicle system.

Aerodynamics of the ducted-rotor system has been studied for several years. An experimental investigation[1] of the effects of various system parameters was conducted for ducted-rotors of micro air vehicle (MAV) scales. A high-fidelity computational fluid dynamics (CFD) simulation[2] for a ducted-rotor has been carried out using a compressible Reynolds-averaged Navier-Stokes (RANS) solver to investigate the aerodynamics of a micro-scale shrouded-rotor configuration in hover.

The Vehicle Technology Directorate at ARL Aberdeen Proving Ground is currently conducting studies on ducted-rotor systems. There are two general approaches to develop performance prediction tools for a ducted-rotor. One of them is a physics-based method to reduce the turnaround time. Some of the physics-based rotor models used are: the classical pressure jump across the rotor plane, pressure jump and swirl corresponding to the rate of rotation of the rotor, and an actuator disc model using blade element theory. The basic idea is to leverage the understanding of the flow-physics to make judicious assumptions and approximations to predict performance with an acceptable level of accuracy at a speed that is an order-of-magnitude faster than the traditional CFD simulations. An example of such an approach is in Reference 3. In this paper, a blade element theory-based method was explored to compute the thrust of a ducted-rotor system. Another approach is to use statistical analysis on the results generated by a method with fewer approximations at the cost of significantly larger computational time and resources. This approach utilizes design of experiments techniques to sample the design space and other statistical analysis methods to develop surrogate models of the actual physical process. An example of such an approach is seen in Reference 4. In this paper, the effect of varying several geometric parameters of the duct on the rotor performance is investigated.

Our earlier study[4], examined the relative effects of several design parameters using surrogate models for the integrated propulsion system. In this study, physics-based models were used to derive the surrogate models for a UAV in hover. One of the design cases examined in the surrogate model work is selected for further study. Our goal in the current study is

to compute detailed flow past the duct-rotor combination to validate the above physics-based models. Further, we aim to capture the flow characteristics caused by the tip vortices and the flow at the lip of the inlet. In our study, we are using the commercial-off-the-shelf software GridPro[5] for generating structured grids, ANSA[6], and AFLR3[7] for generating unstructured grids and the flow solver CFD++[8].

Vehicle Technology Directorate (VTD) has several initiatives related to the ducted-rotor systems. Ducted- rotor systems are being considered as the device to provide the ability to hover and perform short/vertical takeoff and landing (SVTOL) for unmanned aerial vehicles. High-fidelity unsteady modeling of rotor requires huge amount of computational resources. Physics-based models of rotor are used in CFD simulations to reduce the turnaround time for design optimization studies. Such models can reduce the simulation time by more than an order-of-magnitude. The current study helps to validate the physics- based models.

2. Geometry, Flow Parameters, and the Computational Domain

There are several duct geometric, rotor geometric, and rotor operating parameters that influence the aerodynamic performance of a ducted-rotor system. In this study, four parameters associated with the duct shape are considered. Figure 1 shows a schematic of the ducted-rotor describing the four parameters. The inlet shape is defined with two variables, the semi-major and semi-minor axes of the ellipse, R1 (0.15m) and R2 (0.075m), respectively. The diffuser is parameterized by the duct length R3 (0.3m) and the expansion length R4 (0.0m). The rotor diameter is represented by D (0.3m). The zero expansion length implies that we are considering a straight duct without expansion.

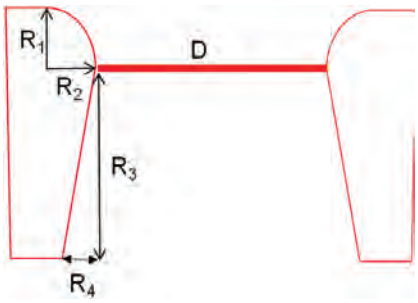


Figure 1. Ducted-Rotor showing geometric parameters

The rotor in consideration is a 4-bladed propeller whose airfoil section is a NACA0015. The collective pitch angle is 20 degrees. The rate of rotation of the propeller is 8,000 rpm, giving a tip Mach number of 0.3 and a tip Reynolds number of 1.3×10^6. The flow domain is enclosed by a cylinder of radius 3D (0.9m), extends upstream to 3D (0.9m) and downstream to 8D (2.4m).The computational domain shown in Figure 2 is enclosed by the green, red and amber surfaces. The surface in yellow is that of the duct.

3. Grid Generation

We used two different grid types, structured and unstructured, in our study. The structured grids are generated using the software package GridPro, and unstructured grids are generated using a combination of ANSA and AFLR3.

The computational grid extends all the way to the outer boundaries as shown in Figure 2.

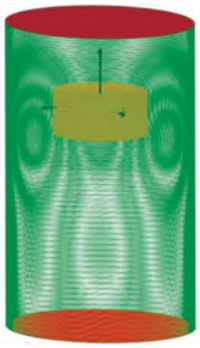


Figure 2. Computational Domain

Parts of the structured grid are shown in Figures 3 and 4. The rotor is enclosed in a computational disc, and this grid is patched with the larger domain grid above and below, as shown in Figure 4. Information across these two grids is passed via interpolation. This arrangement allows us to perform both steady and unsteady calculations as necessary. In the case of unsteady computations, the grid enclosing the rotor is rotated with the rotation rate of the rotor. The grid is very fine near the solid boundaries to capture the viscous boundary-layer. The first grid point is 2×10^{-7} m from the wall.

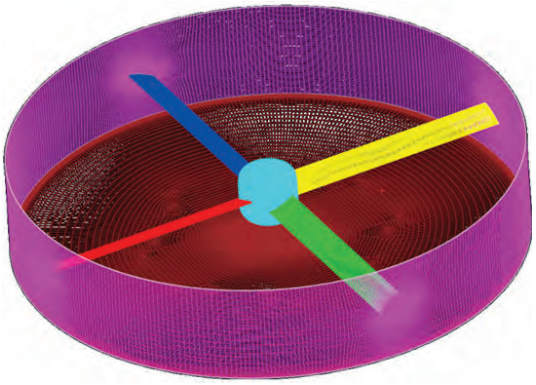


Figure 3. Computational domain enclosing the rotor

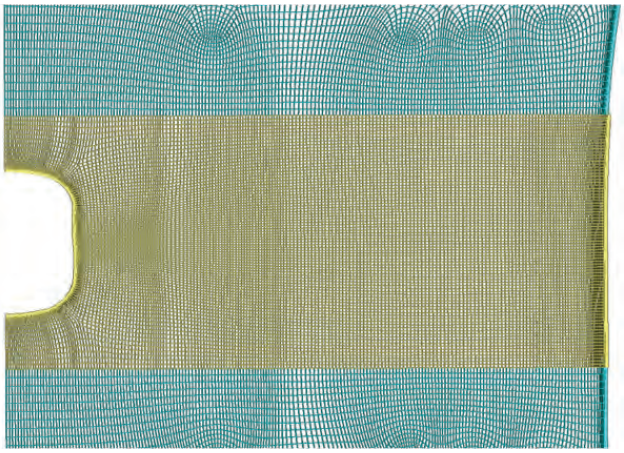


Figure 4. Structured patched grid

A portion of the unstructured grid is shown in Figure 5. The unstructured surface grid is created using the software ANSA and the volume grid is created with AFLR3. Near the walls, prism cells are created with a near-wall spacing of $y^+=1$. The unstructured grid is created for two different cases—one for the open rotor and one with the duct. The total number of cells generated for each case is listed in Table 1. We have a total of five different grids, and they are labeled as shown in the table for ease of reference subsequently.

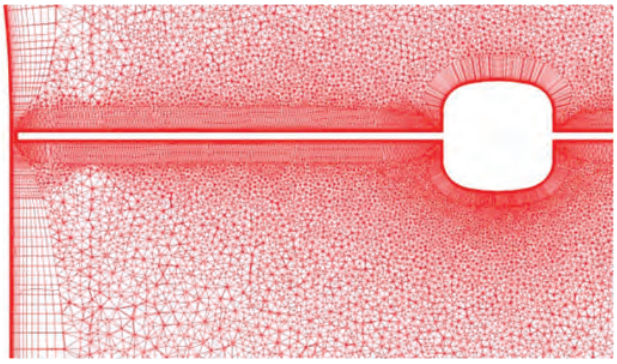


Figure 5. Unstructured Grid

Table 1. Number of cells for each case

Grid	Type	Description	Number of Cells
grid1	Structured	Duct w/o rotor	320,000
grid2	Unstructured	Duct w/o rotor	720,000
grid3	Unstructured	Open Rotor	18,000,000
grid4	Structured	Ducted Rotor	42,000,000
grid5	Unstructured	Ducted Rotor	29,000,000

4. Physics-Based Models

In the physics-based models, blade element theory is used to represent the rotor, but the Reynolds-averaged Navier-Stokes (RANS) calculations are performed in the entire computational domain. The structured grid (grid1) and the unstructured grid (grid2) are used to compute the flow and the thrust components using CFD++. The two-equation realizable k-ε model is used in all our computations. The results are tabulated in Table 2. We can see that the thrust components from the structured and the unstructured grids match closely. Even though the thrust component from the duct is mainly due to inviscid phenomenon, the type of turbulence model, the free-stream turbulence level, and the initial ambient turbulence level influence the flow characteristics. The flow might either stay laminar or re-laminarize if the turbulence level is not high enough. One other problem that might arise in the case of k-ε turbulence model is that when the upstream boundary is far away and if there is no ambient flow, like in the case of hover, the turbulence imposed at the upstream boundary will dissipate by the time it reaches the rotor giving rise to laminar flow. Without the mean flow, the turbulence will dissipate because of the nature of the k-ε equations. To circumvent this problem, we imposed an inactive region upstream of the rotor for the production terms in the turbulence model.

Table 2. Thrust computed from physics-based models

Grid	Structured (grid1)	Unstructured (grid2)
Turbulence Model	k-ε	k-ε
Thrust due to Duct (N)	14.71	14.56
Rotor Thrust (N)	19.20	19.30

5. Detailed Flow Computations

5.1 Open-Rotor

Open-rotor computations were done with the unstructured grid (grid 3, Table 1) generated using ANSA and AFLR3. Prism cells were generated near the viscous surfaces with the first grid spacing small enough to resolve up to a y^+ of 1. The 2-equation realizable k-ε turbulence model was used in these calculations. The convergence history of the rotor thrust is shown in Figure 6. The rotor thrust converges to a mean value of 22N. The iso-surface colored by pressure is shown in Figure 7. This plot shows how the tip vortices persist up to a quarter of the rotor revolution, whereas those that are shed along the trailing-edge dissipate quickly. The grid resolution is not fine enough to resolve the tip vortices to longer downstream distances.

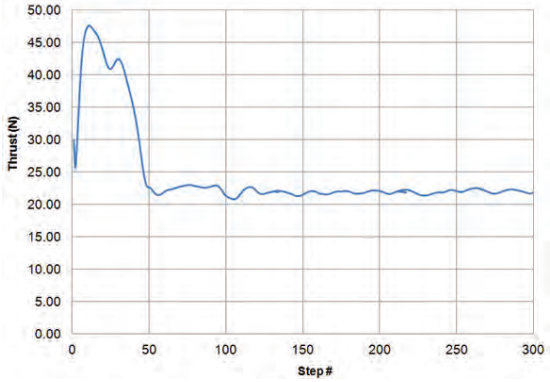


Figure 6. Open-rotor thrust obtained from K-epsilon models

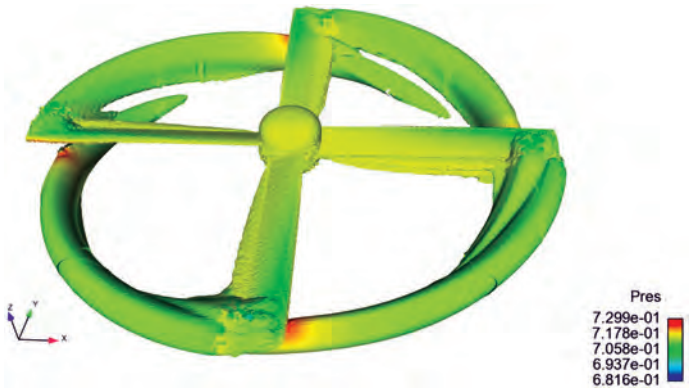


Figure 7. Iso-surface of vorticity colored by pressure. Results for open rotor case.

5.2 Ducted-Rotor

The ducted-rotor computations were conducted using structured (grid4) and unstructured (grid5) grids as shown in Table 1. In both cases, we used the k-epsilon turbulence model. In Figure 8, the vorticity iso-surface colored by pressure is shown. These results are those from using the structured grid (grid4). Although the near body vorticity is well captured, the vorticity downstream is dissipated. One of the reasons is because the grid is coarsened gradually downstream of the duct.

Figure 8. Iso-surface of vorticity colored by pressure

In Figure 9, we compare the vertical velocity contours from the rotor model and the rotor computations. The contours upstream of the rotor plane and the duct compare well in both the computations. However, there is considerable difference in the contours downstream of the rotor and the duct. In the detailed computations, the vorticity is well captured; whereas we don't see any vorticity in the physics-based model calculations. We see upward velocity in the wake of the rotor hub from the detailed computations; whereas we see no upward velocity in the model calculations. The rotor hub is not modeled in the physics-based model calculations. We observe increased downward velocities Figure 9, in the wake of the physics-based model. One of the reasons for this could be that the entire momentum is imparted to the fluid in the downward direction at the rotor plane; whereas in the detailed model, the fluid passing through the rotor is imparted both rotational and downward momentum. We also observe increased downward velocity near the exit plane in the model calculations. This could be because of the way the boundary conditions at the exit plane are applied. This needs to be further examined. However, this doesn't affect the computed thrust due to the rotor and duct, which are of interest in the current study. The thrust is divided into two components, one due to the duct and the other due to the rotor itself. The computed results are tabulated in Table 3.

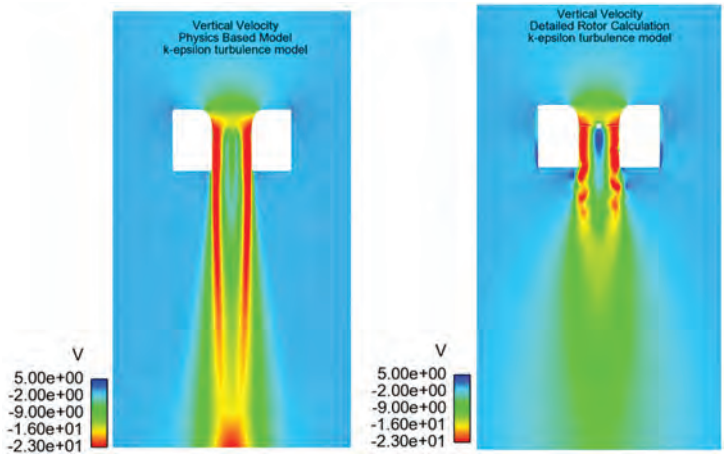


Figure 9. Comparison of vertical velocity contours

Table 3. Thrust from Ducted-Rotor Cases from detailed computations and physics-based model

	Thrust (N) Duct	Thrust (N) Rotor
Structured grid1 (Blade element theory model)	14.71	19.2
Unstructured grid2 (Blade element theory model)	14.56	19.3
Structured grid4 (Detailed computations)	12.7	16.9
Unstructured grid5 (Detailed computations)	12.3	17.3

The computed thrusts from the physics-based model due to the duct from grid1and grid2 calculations are 14.71N and 14.56N, respectively. As we can see, these results from different grids are quite close for the physics-based model calculations. The computed thrust from the detailed calculations due to the duct from grid4 (12.7N) and grid5 (12.3N) are also quite close.

The computed rotor thrusts from the physics-based model from grid1 and grid2 calculations are 19.2N and 19.3N, respectively. The computed rotor thrust from grid4 computations is 16.9N; whereas that from grid5 is 17.3N.

We see that the physics-based model over-predicts the rotor thrust by 13.7% in the structured grid computations, and by 15.5% in the unstructured grid computations. The physics-based model over-predicts the contribution from the duct by 12% for the structured grid case, and by 10.4% for the unstructured grid case.

The computations were performed on ARL DSRC computer systems MJM and Harold. MJM has 1,100 two-processor socket dual-core compute nodes, each with 8GB of memory. The processors on MJM operate at a frequency of 3GHz, resulting in a peak floating-point rate of 12.0 gigaflops per core. The physics-based model takes 200 iterations to converge on the structured grid (grid1) with 320,000 cells. It takes about 1,800 on 4 cores or about 2 CPU hours on MJM. The detailed computations take over 1,000 iterations on the structured grid (grid4) with 42 million cells. It takes over 10 hours on 128 cores or 1,280 CPU hours on MJM.

6. Conclusions and Observations

We have computed the thrust components due to the ducted-rotor using a physics-based model for two different grids—grid1 and grid2. We have carried out two detailed computations for the ducted-rotor, again using structured grid (grid4) and unstructured grid (grid5). We have also computed the detailed flow computations for the open-rotor using the unstructured grid (grid3). We have used the 2-equation realizable k-e model in all our calculations. For this particular design, the physics-based model consistently over-predicts both the rotor thrust, as well as the thrust due to the duct in the range of 10–15%. This level of accuracy might be sufficient during the initial design studies, since we are more concerned with the trends associated with changes in the design variables rather than obtaining very accurate numbers. Detailed calculations

could be conducted over a smaller range after the initial stages. Our current study included only one case from the several considered in Reference 4. We need to conduct several such studies at different design envelopes to give us confidence in the models. As we can see, the savings in the computational time is enormous, 2 CPU hours versus 1,200 CPU hours, when we use physics-based models as opposed to the detailed computations.

Acknowledgments

Surya Dinavahi acknowledges the support of Computational and Information Sciences Directorate (CISD) and the Vehicle Technology Directorate via the DoD contract number GS04T08DBC0020. The CFD computations were carried out on ARL DSRC computer systems MJM and Harold.

References

1. Pereira, J. and I. Chopra, "Hover Tests of Micro Aerial Vehicle-Scale Shrouded-Rotors, Part I: Performance Characteristics", J*ournal of the American Helicopter Society*, Vol. 54, (1), January 2009.

2. Lakshminarayan, V K. and J.D. Baeder, "Computational investigation of micro-scale shrouded-rotor aerodynamics in hover", *AHS International Specialists' Meeting on Aeromechanics*, AHS International, 2010.

3. Singh, R., M. Floros, B. Roget, and J. Sitaraman, "Ducted-rotor performance predictions using Momentum theory and CFD", *27th Army Science Conference*, Orlando, FL, 2010.

4 Singh, R. and S. Dinavahi, "Shape Optimization of a Ducted-Rotor System for Aerodynamic Performance", *49th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Orlando, FL, 4–7 January 2011.

5. *GridPro*, Program Development Company.

6. *ANSA Software – CAE Pre-processing Tool*, BETA CAE Systems S.A.

7. *AFLR3: Unstructured Grid-Generation Software*, Sim Center, Mississippi State University.

8. *CFD++ Computational Fluid Dynamics Software*, Metacomp Technologies.

Work in Progress: Vortex Detection and Visualization for Design of Micro Air Vehicles and Turbomachinery

Rhonda J. Vickery and Hugh Thornburg
High Performance Technologies, Inc. (HPTi), Wright-Patterson AFB, OH
{rvickery, hthornburg}@hpti.com

Thomas Wischgoll, Chris Koehler, Haibo Dong, Matthew Pickett, and Neal Eikenberry
Wright State University, Dayton, OH
{thomas.wischgoll, koehler.11, haibo.dong}@wright.edu, acrolith_@hotmail.com, neal62689@gmail.com

Lance Harris
Central State University, Wilberforce, OH
l.harris313@gmail.com

Richard Snyder and Darius Sanders
US Air Force Research Laboratory (AFRL), Wright-Patterson AFB, OH
{richard.snyder darius.sanders}@wpafb.af.mil

Randall Hand
Lockheed Martin IS&GS, Vicksburg, MS
randall.e.hand@usace.army.mil

Abstract

Vortex detection and visualization is an important technique for computational fluid dynamics (CFD) modelers and analysts. Since vortices are often not just local phenomena, algorithms for detecting the vortex core can be expanded by the use of streamline placement and termination methodologies to appropriately visualize the vortex. We are enhancing an existing VCDetect software tool for vortex detection to include new algorithms applicable to small-scale micro air vehicles (MAVs), improve the interface, and integrate it with Department of Defense (DoD) Visualization Toolkit (VTK)-based production tools. The code is being updated to include the latest parallelization features for efficient usage. The current VCDetect VTK-based code was developed with an XML file-based interface, and was partially parallelized with multi-threading. The code was tested with a few common examples and some turbo-machinery data test cases. In this work we are integrating newly developed visualization and vortex detection algorithms. An improved interface is added to allow the tool to be used both interactively and to easily set up multiple batch runs. Since this interface is expected to run on the user's local desktop, we are securing the communications using a previously developed direct through ssh methodology. We are also further parallelizing the code using the Compute Unified Device Architecture (CUDA), and we are applying it to a new problem domain in the area of micro air vehicle (MAV) design. Finally this code will be made available to more DoD users by coordinating with the High Performance Computing Modernization Program's (HPCMP) Data Analysis and Assessment Centers (DAACs) to include it in the Computational Science Environment (CSE) suite of production tools and libraries. We describe the extent of our work to date, and provide information on our path forward to completion of this project by 31 August 2011.

1. Introduction

This work enhances an existing VCDetect software tool for vortex detection to include new algorithms applicable to small-scale micro air vehicles (MAVs), improve the interface, and integrate it with Department of Defense (DoD) VTK-based production tools. The code is being updated to include optimizations for execution on heterogeneous CPU/GPU architectures. We are applying it to a new problem domain in the area of MAV design, and are integrating newly developed visualization and vortex detection algorithms as described herein. The code will be made available to DoD users by coordinating with the High Performance Computing Modernization Program's (HPCMP) Data Analysis and Assessment

Centers (DAACs) to include it in the Computational Science Environment (CSE) production tool suite. The next section describes our methodology, including our motivation for the problem domain and application design. We also include some preliminary results in our analysis of dragonfly data described in (Koehler, et al., 2011). Finally we close with some conclusions and ideas for future work.

2. Background

In many computational fluid dynamics (CFD) applications, we are interested in studying the vortices within the simulated flow to gain insight into the data at hand. This is also the case when investigating the wings of insects to derive better wing designs for MAVs that are more optimal with respect to flight characteristics as well as energy efficiency. In this particular case, the most important features of the flowfield are the leading edge vortices (LEVs) on each insect wing. The wide variation of experimental results surrounding research into how insects use the LEV to generate lift suggests that there is a lack of understanding about how it forms and behaves as the insect's wings flap. With the most recent deformable wing simulations, it is also important to study the relationship between how the wings deform and how the LEV is produced. The trailing edge vortex (TEV) is also potentially utilized by insects for certain maneuvers, such as rotation, takeoff and landing. Hence, flow visualizations that highlight the relationship between deformable wing kinematics and vortex (LEV and TEV) production are required.

When visualizing the flow to study the deformable wing of an insect, the focus should be on both the LEV and TEV. Hence, the visualization algorithm has to extract these features from the flow data and then visualize the flow properties in their vicinity. Typically, stream lines or stream surfaces are used to depict such flow properties. Due to the high complexity of the flow in the vicinity of the vortices as a result of the highly turbulent flow characteristics in those areas, stream lines are often chosen over stream surfaces since the latter tend to overlap on themselves significantly in such complex flows. Even with stream lines, special care has to be taken in seeding those stream lines and terminating them in such a way that ensures a visualization that is easy to comprehend by the user, yet emphasizes all necessary features of the flow. This is one of the major advantages of the proposed approach over existing techniques. Automatic seeding strategies are utilized based on the flow features, i.e., the leading edge vortex and the trailing edge vortex, and stream line termination keeps overlapping within the visualization at a minimum, while properly visualizing the important characteristics of the flow.

3. Methodology

In the initial stages of this project, we planned to integrate recent work in the area of vortex detection and visualization into a tool that could be transitioned into the DoD HPCMP production environment. This involved enhancing a Visualization Toolkit (VTK)-based tool developed with an XML file-based interface that was partially parallelized with multi-threading (van der Zwaag, et al., 2009). The code was tested with a few common examples and some turbomachinery data test cases. This interface is useful for running multiple test cases in the batch environment, but becomes tedious to use when setting up a large number of test cases covering many parameter options. Although VCDetect had two vortex detection algorithms, an eigenvector method developed by Sujidi and Haimes and the Parallel Vectors method developed by Roth and Peikert, often the vortices of interest are difficult to discern, and other vortex methods were needed (Jiang, et al., 2004). The Lambda2 method developed by Jeong and Hussain had been implemented and integrated with a visualization technique known as "vortex tubes", and the code was obtained to be implemented with VCDetect (Hand, et al., 2006). Additionally, a new streamline placement visualization technique has been recently developed in conjunction with recent research into the deformable dragonfly wing kinematics which provided the data for this work, including photogrammetric setup and parametric surfaces used to digitally reconstruct a useful dragonfly model (Koehler, et al., 2011). Integrating these new features into an easy-to-use tool for vortex detection is the thrust of this work.

A high-fidelity in-house direct numerical simulation tool was used to simulate the flow induced by the flapping dragonfly wings. It is based on an existing Navier-Stokes immersed-boundary solver (Mittal, et al., 2008), which is capable of simulating flows with complex moving boundaries, including solid and membranous objects, on stationary Cartesian grids (Dong, et al., 2006). The solver employs a non-dissipative, 2^{nd}-order, central-difference scheme, which is critical for accurately predicting vortices.

Using global metrics such as vorticity or helicity for streamline placement can be computationally-intense when dealing with large vector fields and many time-steps. Koehler, et al., 2011 developed an interactive streamline seeding algorithm which alleviates this problem by placing seed points based on the evolution of objects immersed in the flowfield, rather than metrics taken from the flowfield itself. An example using this method is shown in Figure 1. Streamlines are particularly effective at visualizing vortices in insect flight and micro air vehicle simulation data due to the easily identifiable helix

shape. However, streamline seed placement is challenging in highly-unsteady three-dimensional (3D) flapping flight data. Uniform coverage of the entire flow domain produces very busy results.

Figure 1. Streamlines colored by vorticity magnitude were generated using the dragonfly wings leading edge as seed points

Experiments have been executed with both seed generation and rendering techniques to more accurately display the key vortices without having them be occluded by less important features of the flow (Koehler, et al., 2011). This is particularly important in order to study formation and movement of the LEV, as well as the shedding of the wing-tip vortices and TEVs. Initial seed placement experiments were done based on the vorticity magnitude in the area surrounding the wing leading edges. Targeting specific vorticity ranges allows the visual capturing of different portions of the vortices that form on the leading edge of each wing during the upstroke and downstroke. Experiments were also done with semi-transparent streamlines. Making streamlines whose seed points are above a certain vorticity magnitude more transparent allows users to visualize vortices with less self-occlusion of the streamlines. Figure 2 shows the result of one of the prototypes where this criterion was applied.

We recently started transitioning the VCDetect tool into a ParaView plug-in that can be deployed as part of the CSE (Squillacote, 2008; Renteria and Mark, 2008). One advantage of using this approach is the ability to implement a standard interface with only a few lines of XML. ParaView also offers a large number of readers, including the Tecplot and CGNS readers, needed for this project.

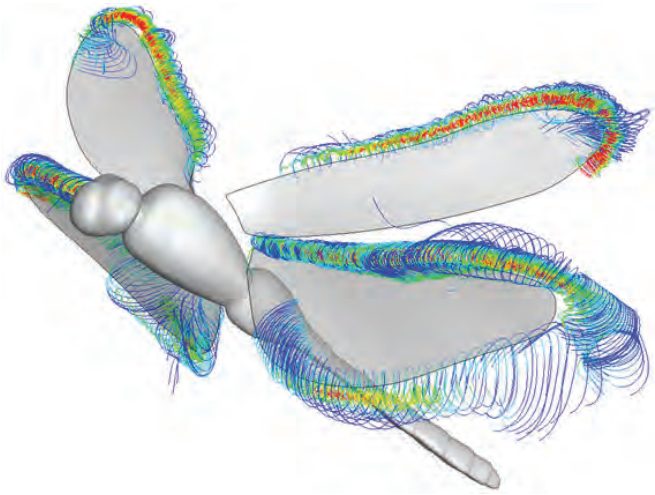


Figure 2. Streamlines distributed based on vorticity criteria

We have applied the VCDetect tool to automatically find corelines, and core surfaces that show the vortex locations and structure for the underside of a dragonfly wing (see Figures 3a and b). We have also used ParaView to create streamlines using the corelines as a starting point to highlight vortices which could be overlooked during initial analysis (Figure 4). These results may help provide additional insight into the flow characteristics, and therefore MAV design that mimics this type of flight.

Figure 3. Automatically generated: (a) vortex corelines, (b) surfaces

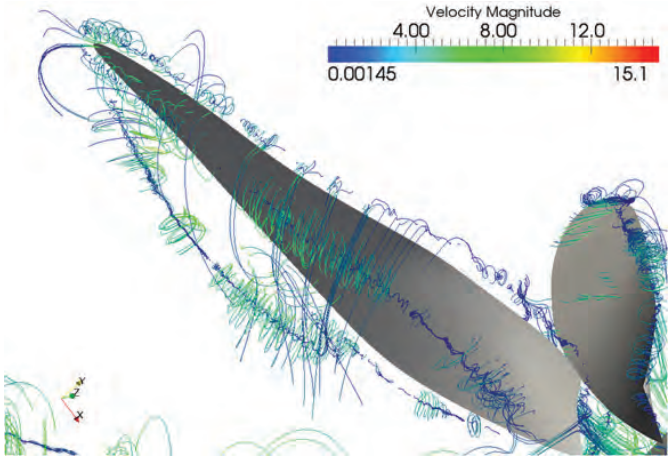


Figure 4. ParaView-generated streamlines using corelines as a starting point colored by velocity magnitude

4. Summary and Future Work

Currently we are in the process of integrating several features into a VCDetect plugin for ParaView to be included in the CSE. We have also applied this technology to turbomachinery data, and expect that it will be applicable to other CFD datasets.

In recent years, the architectural design of software has changed to include the proliferation of GPU's for computation. This provides exciting opportunities to gain significant speedup in applications with data parallelism. The previously developed VCDetect code, though multi-threaded, did not take advantage of this new approach. As shown in Figure 3, VCDetect locates vortex cores and generates surfaces defining the vortices of interest. Currently this takes 10 minutes or more per test case on a high-end workstation with 8 CPUs, and so becomes very time-consuming when multiple parameters and vortex algorithms need to be applied. In this project we have several workstations with NVIDIA Tesla cards for development, as well as access to a new GPU-based state-of-the-art mini-supercomputer. The availability of such hardware gives us the development platform needed to further parallelize the code for heterogeneous CPU/GPU architectures. We plan to compare the results of our optimized code with our previous implementation. Although using CUDA or OpenCL are logical choices for this, there are other parallel streaming technologies that we are also considering (Vo, et al., 2010).

We take advantage of the expertise of a multi-disciplinary team to provide value to all phases of the project. We are working jointly with our DoD partners to prioritize the desired features through a living requirements document in order to maximize our time and resources. We are using the Agile software development approach to iteratively deliver and refine the software to keep on track with evolving customer requirements while receiving valuable feedback for improvement.

This approach maximizes team productivity and allows for modification during the project lifetime. The only real technical risk is inherent in the field of vortex detection; a chosen algorithm may not find the vortices of interest in the particular data set. Hence we mitigate that risk by including multiple algorithms in an infrastructure that makes it easy to add others. Since our code is VTK based, adding it to other DoD production visualization suites is straightforward.

Acknowledgments

This work was funded by the HPCMP under the User Productivity, Enhancement, Technology Transfer and Training (PETTT) preplanned effort PP-ACE-KY02-013-P3, GSA contract GS04T09DBC0017.

References

Dong, H., R. Mittal, and F.M. Najjar, "Wake topology and hydrodynamic performance of low-aspect-ratio flapping foils", *Journal of Fluid Mechanics*, Vol. 566, pp. 309–343, November 2006.

Dong, H., C. Koehler, Z. Liang, H. Wan, and Z. Gaston, "An Integrated Analysis of a Dragonfly in Free Flight", *AIAA 2010-4390*, Chicago, IL, 28 June–1 July 2010.

Hand, R. and P. Adams, "Visualization of Time-Dependant, Quasi-Streamwise Vortex Tubes in a Bubble-Laden Turbulent Boundary-Layer over a Flat Plate", *Proceedings of the 2006 DoD High Performance Computing Modernization Program Users Group Conference*, 2006.

Jiang, M., R. Machiraju, and D. Thompson, "Detection and Visualization of Vortices", *Visualization Handbook*, C. Johnson and C. Hansen, editors, Academic Press, 2004.

Koehler, C., T. Wischgoll, H. Dong, and Z. Gaston, "Vortex Visualization in Ultra-Low Reynolds Number Insect Flight", *IEEE Transactions on Visualization and Computer Graphics* (under review).

Mittal, R., H. Dong, M. Bozkurttas, F.M. Najjar, A. Varga,s and A. von Loebbecke, "A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries", *Journal of Computational Physics*, vol. 227, pp. 4825–4852, May 2008.

Renteria, J.C. and E.R. Mark, "The Computational Science Environment (CSE)", *Proceedings of the 2008 DoD High Performance Computing Modernization Program Users Group Conference*, pp. 382–386, 2008.

Roth, M. and R. Peikert, "A Higher-Order Method for Finding Vortex Core Lines", *IEEE Visualization '98*, pp. 143–150, October 1998.

Schroeder, W., K. Martin, and B. Lorensen, *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, 4th Edition, Kitware, Inc., 2006.

Squillacote, A.H., *The ParaView Guide, Version 3*, Kitware, Inc., 2008.

Sujudi, R. and R. Haimes, "Identification of Swirling Flow in 3D Vector Fields", *AIAA 12th Computational Fluid Dynamics Conference*, pp. 95–1715, June 1995.

Van der Zwaag, J., R.J. Vickery, and R. Moorhead, "Vortex Detection through the Visualization Toolkit", *Proceedings of the 2009 DoD High Performance Computing Modernization Program Users Group Conference*, pp. 395–397, 2009.

Vo, H., D. Osmari, B. Summa, J. Comba, V. Pascucci, and C. Silva, "Streaming-Enabled Parallel Dataflow Architecture for Multicore Systems", *IEEE/VGTC EuroVis 2010* (to appear).

HPCMP UGC 2011

1. Computational Fluid Dynamics (CFD)

Gas Turbine and Propulsion

Application of CFD in Development of Large-Scale Scramjets

Mark A. Hagenmaier and Dean R. Eklund
US Air Force Research Laboratory, Aerospace Propulsion Division (AFRL/RZAS), Wright-Patterson AFB, OH
{mark.hagenmaier, dean.eklund}@wpafb.af.mil

John A. Boles and Ryan M. Milligan
Taitech Inc., Beavercreek, OH
{john.boles.ctr, ryan.milligan.ctr}@wpafb.af.mil

Abstract

Computational fluid dynamics (CFD) was used in coordination with ground-testing to develop the small-scale scramjets tested in the X-43A and X-51A programs. As we move toward larger scramjets, the role of CFD in the development approach will likely change. Ground-test facilities are not currently available, and would be prohibitively expensive, to fully-test large-scale scramjet-powered vehicles such as those envisioned for access-to-space missions. The technical goal of the present work is to explore two techniques that could change the development approach to be applicable to the development of large-scale scramjets. The first technique uses conventional RANS CFD in a design mode to develop wind-tunnel hardware that would allow large scramjet combustors to be tested at appropriate flight-like conditions. The second technique is to use higher-fidelity CFD to more accurately model the flow in the engine.

1. Introduction

Supersonic combustion ramjets (scramjets) provide efficient propulsion for flight at speeds greater than 4 times the speed-of-sound (Mach 4). In 2004, a scramjet developed by NASA (called the X-43A) demonstrated accelerating flight at Mach numbers of 7 and 10 using hydrogen fuel (McClinton, 2006). In 2010, a scramjet developed by the Air Force (called the X-51A) demonstrated accelerating flight at a Mach number of approximately 5, using a more conventional hydrocarbon jet fuel (88th Air Base Wing Public Affairs, 2011). Each of these demonstration vehicles was fairly small compared to the future systems envisioned for hypersonic propulsion. The development approach used to progress small-scale scramjet technology from the wind-tunnel into flight vehicles has involved a coordinated use of combustor ground-tests (referred to as direct-connect testing), full-engine ground-tests (referred to as free jet testing), and computational fluid dynamics (CFD) analysis of these physical experiments using solution procedures based on the Reynolds-averaged Navier-Stokes (RANS) equations. Comparison of the results from these techniques led to confidence in the engine performance and operability, which enabled flight-tests of X-43A and X-51A.

For larger scramjets, the development approach used for small scramjets becomes invalid. Ground-test facilities are not currently available, and would be prohibitively expensive, to fully-test large-scale scramjet-powered vehicles. The technical goal of the present work is to establish two techniques that could change the development approach to be applicable to the development of large-scale scramjets. The first technique uses conventional RANS CFD in a design mode to develop wind-tunnel hardware that would allow scramjet combustors to be tested at appropriate flight-like conditions. The second technique is to use higher-fidelity CFD to more accurately model the flow in the engine.

2. Use of RANS CFD for Wind-Tunnel Hardware Design

The first technique will allow for large-scale scramjet combustors to be tested in existing test facilities that have typically been used only for full-engine tests. In other words, this approach will turn existing free jet test facilities into direct-connect test facilities. This effort involves a large number of RANS simulations in order to properly design the wind-tunnel hardware that will provide a realistic simulation of the inflow conditions expected in flight. The large number of simulations is driven by the fact that this is an inverse design problem. In the first stage of the design, the target flow is determined, as will be shown in Section 2.2.1. The traditional approach, which does not adequately match the flow features of the target flow, is discussed in Section 2.2.2. In the later stages of the research, a series of calculations on a large number of geometric configurations is performed which leads to the final design that will be discussed in Section 2.2.3.

This work has resulted in the aerodynamic design of wind-tunnel hardware for use in a small-scale test facility. In the Fall of 2011, a test of the distortion-generation device in Research Cell 18 at AFRL/RZA will provide validation data for this design effort. The remainder of section 2 is based on the results of Hagenmaier, et.al. (2011).

2.1 Computational Tools

RANS CFD calculations have been performed with CFD++ from Metacomp Technologies. The code has a finite-volume numerical framework with Riemann solvers for accurate representation of supersonic flows. Shock discontinuities were handled using the HLLC Riemann solver with a continuous flux limiter. The realizable k-ε turbulence model was used to represent the apparent turbulent stresses. The realizability criterion accounts for known physical properties of the stress tensor by bounding the magnitude of the tensor components, which improves accuracy and stability. Multi-grid acceleration was used to provide accelerated convergence for steady flows.

Structured grids were utilized for all configurations. The grids for the flight inlet, direct-connect nozzles and distortion generation hardware used a wall-spacing for solve-to-wall turbulence models (i.e., y+ near 1), and spacing in the core flow of approximately 1 mm. The grids for the isolator used a wall-spacing for a wall-function turbulence model (i.e., y+ near 20), and spacing in the core flow of approximately 1 mm.

2.2 Development of Direct-Connect Distortion Generation Hardware

The present work applies a similar approach to that used in Gruber, et.al. (2006) to create a representative distortion for AFRL/RZA direct-connect combustor hardware in Research Cell 18 (RC18), which has the capability to support advanced optical diagnostics in the isolator section. The cross-section of RC18 is 38.1 mm×101.6 mm, and a facility nozzle exists which provides uniform flow with a throat Mach number of approximately 2.18. Figure 1 shows a diagram of the flow structure in a planar compression inlet. An expansion fan is generated by the turning of the body-wall at the throat (called the shoulder), but has been excluded in this diagram. In each of the flow regions upstream of the engine throat (noted as 0, A, B, C, and D) the flow properties are nearly constant. In a traditional direct-connect study, a facility nozzle is used which matches the average condition at the engine throat (noted as "throat" in the figure). In the present work, we aim to design hardware that matches these average conditions as well as the shock- and boundary-layer properties at the throat. All three testing modes are investigated in the current study using RANS CFD modeling tools, and the two direct-connect modes will be explored in a future experimental study in RC18.

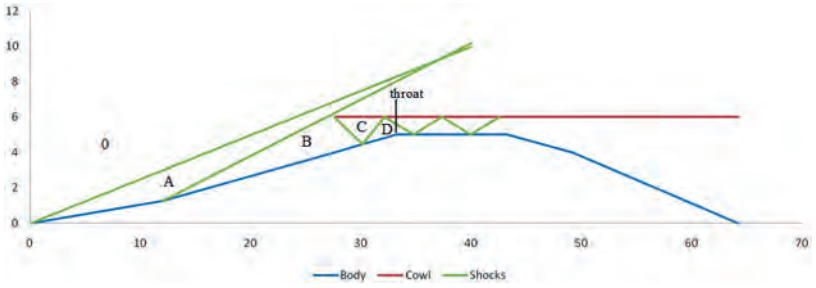


Figure 1. Planar compression scramjet flow structure

2.2.1 Flight Mode

A generic inlet is considered for this study, which employs a single 6 degree fore-body turning angle, and a cowl that turns the flow back to the axial direction. This is obviously not an optimized inlet, but the cowl turning angle, which dictates the strength of the internal shocks that create the flow distortion, is representative of planar compression scramjets. The Mach number and angle-of-attack of the flight inlet were varied such that the average Mach number at the throat was 2.18, to be consistent with an existing facility nozzle available for RC18. For the present study, a flight condition of Mach 4, dynamic pressure of 95.7 kPa (2,000 psf), and an angle-of-attack of 9.2 degrees provided the desired throat Mach number of 2.18. Figure 2(a) shows the Mach distribution on the center plane, while the Figure 3 shows the average Mach for each streamwise location.

2.2.2 Non-Distorted Direct-Connect Mode

An existing RC18 facility nozzle, which produces near-uniform flow, was analyzed for consistency with the flight inlet. The nozzle area ratio is approximately 2.0. At conditions consistent with Mach 4 flight at a dynamic pressure of 95.7 kPa

(2000 psf), the average Mach number at the engine throat is 2.18. This throat condition is consistent with the generic flight inlet. Figure 2(b) shows the Mach distribution on the center plane for the non-distorted direct-connect mode.

Figure 3 shows that the average Mach is consistent with the flight inlet from the engine throat through the end of the isolator.

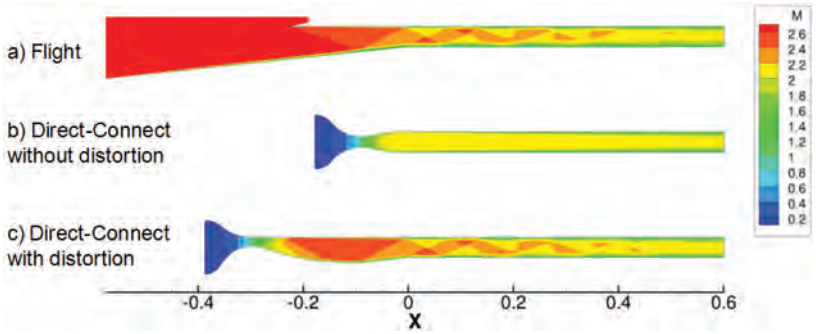


Figure 2. Center plane Mach for flight and direct-connect conditions

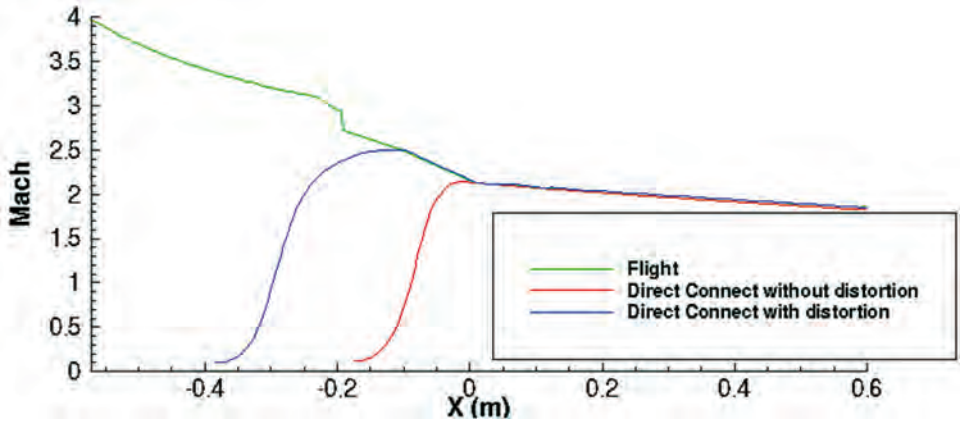


Figure 3. Average Mach for flight and direct-connect conditions

2.2.3 Distorted Direct-Connect Mode

The design of the distortion generator hardware proceeded from the CFD results for the flight inlet. The procedure developed in Reference 1 results in a new facility nozzle that generates the flow properties in region C (see Figure 1), where the flow is parallel to the cowl. For the generic inlet at the matching flight condition, the Mach number in region C is approximately 2.50, and the shock that separates region C from region D is generated 94 mm upstream of the engine throat. That location defines the start of the compression region of the distortion generation device. A method-of-characteristic procedure with boundary-layer corrections was used to define a nozzle that provides the Mach 2.50 conditions. Figure 4 shows a schematic of the distortion generator configuration. The facility nozzle that generates Mach 2.50 flow and the compression section from the shock reflection point 94 mm upstream of the throat will likely be constructed as a single-unit for the planned experimental program.

Figure 2(c) shows the Mach distribution on the center plane for the distorted direct-connect mode, and show similar shock structures as the flight inlet.

Figure 3 shows that the average Mach are consistent with the flight inlet from the shock reflection point 94 mm upstream of the engine throat through the end of the isolator.

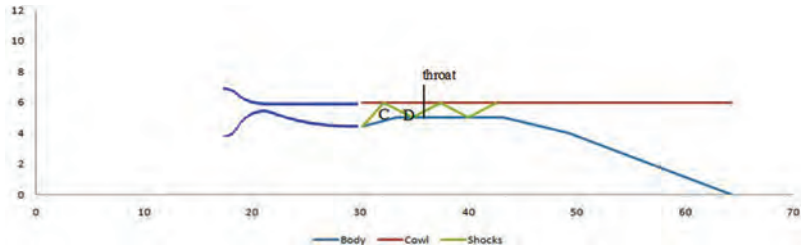


Figure 4. Schematic of distortion generator configuration

Comparisons of body centerline wall pressure of the three testing modes are included in Figure 5. Excellent agreement between the flight and distorted direct-connect modes is seen, showing a similar oscillating pressure caused by the reflections of the inlet shock and shoulder expansion. As expected, the non-distorted direct-connect pressures show a smoothly increasing pressure caused by the supersonic deceleration of the flow due to wall friction.

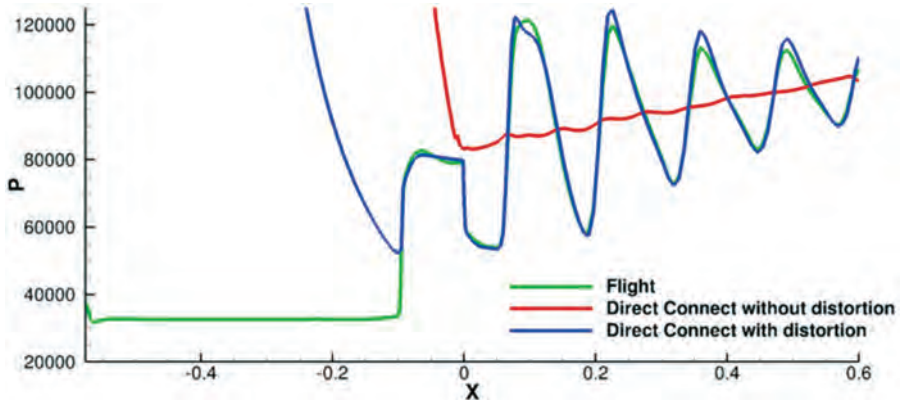


Figure 5. Body-side centerline wall pressure comparison

The preceding data shows how well the distorted direct-connect mode matches the average conditions and shock structure of the flight mode. However, isolator performance is also related to boundary-layer parameters such as momentum thickness and wall-shear stress. Comparison of the wall-shear stress on the body centerline is shown in Figure 6. It is seen that the distorted direct-connect mode more closely matches the wall-shear stress expected in flight.

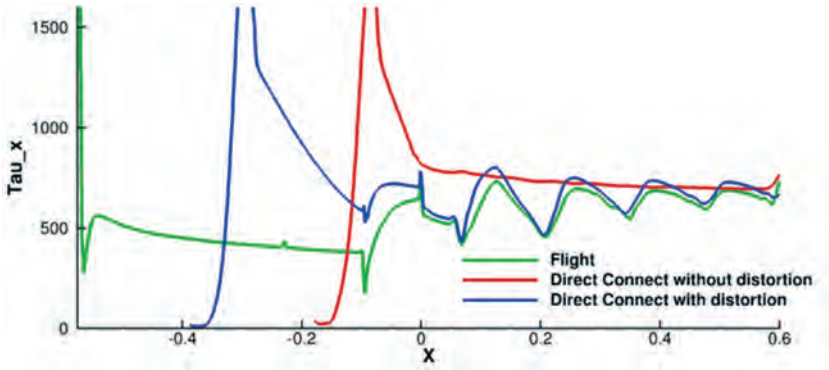


Figure 6. Body-side centerline shear stress comparison

Figure 7 shows Mach number in a plane just downstream of the throat. The direct-connect with distortion mode captures the general shape of the flow distortion that is seen in-flight. Note; however, that the corner boundary-layers on the cowl side are much thicker in the distorted direct-connect mode than in-flight. A closer look at the profiles on the centerline at the same plane is shown in Figure 8. The slope of the Mach profiles at the wall indicates that the wall-shear is similar for the distortion mode and the flight mode, but the overall boundary-layer thickness and shape are still somewhat different.

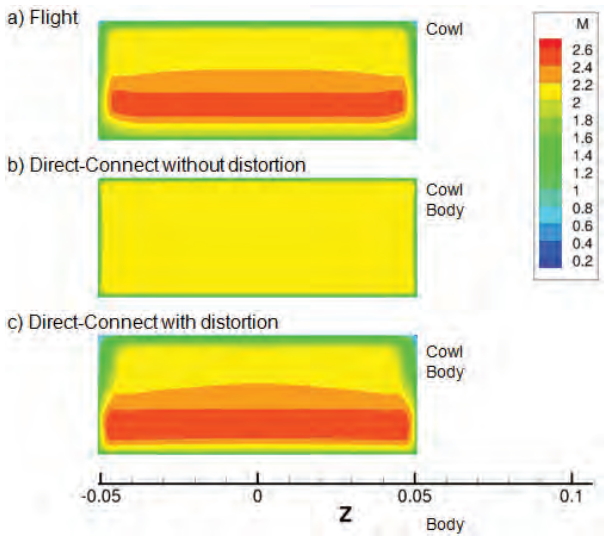


Figure 7. Mach number distribution in plane just after throat

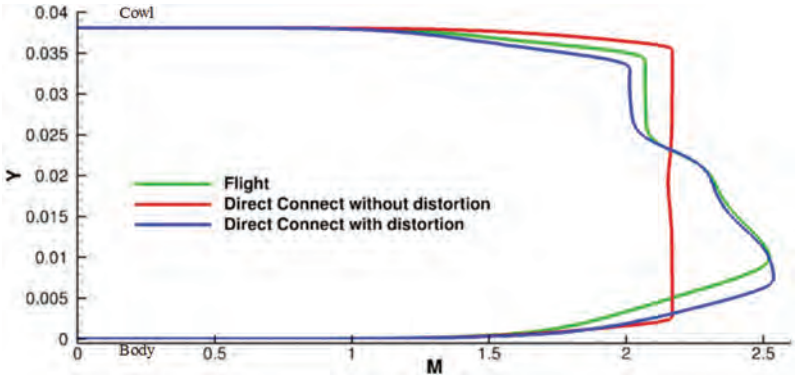


Figure 8. Mach distribution on center plane just after throat

3. Use of Hybrid RANS/LES CFD for Understanding Flow Phenomena

The second technique to change the development approach is to use higher-fidelity CFD to more accurately model the flow in the engine. Hybrid approaches that use RANS models near the viscous surfaces and Large-Eddy Simulation (LES) models away from these surfaces have been shown to be a reasonable compromise between the high-cost and fidelity of full-LES simulations and the low-cost and fidelity of conventional-RANS simulations. This effort involves a smaller number of simulations that will be focused on developing a better understanding of the complex flow interactions in unit problems relevant for scramjet flows. Progress has been made in the areas of scramjet isolator dynamics, non-intrusive fuel injectors, and flame-holding devices.

3.1 Computational Tools

REACT-MB, a research code developed at North Carolina State University by Prof. Jack Edwards and co-workers (Boles, et al., 2011) has been used for this portion of the research. REACT-MB allows for a hybrid solution approach using the RANS formulation near walls and a large-eddy simulation (LES) approach away from the walls. The hybrid LES/RANS model used in this investigation is based on Menter's k−ε/k−ω baseline (BSL) model and only involves modifications to the eddy-viscosity description. The eddy-viscosity is represented via a blending function between the RANS value and a sub-grid value based on the Smagorinsky model. The blending function is based on the ratio of the wall distance to a modeled form of the Taylor micro-scale, to force the average LES to RANS transition position for equilibrium boundary-layers to occur at the point where the wake law starts to deviate from the log law.

A planar relaxation sub-iteration procedure coupled with a Crank-Nicholson time discretization is used to advance the governing equations at second-order temporal accuracy. Inviscid fluxes are calculated using Edwards' low-diffusion

flux-splitting upwind scheme (LDFSS) (Edwards, 1997). These fluxes are extended to second-order accuracy, using the Piecewise Parabolic Method (PPM) (Collela and Woodward, 1984). PPM begins with a fourth-order central difference approximation (on uniform meshes). PPM preserves monotonicity by resetting the left and the right states in order to add dissipation where necessary. This improves the ability of the method to capture shocks, but reduces its ability to capture small vortical structures directly. A Ducros switch is implemented based on the work of Gieseking (2010) to retain the fourth-order scheme away from shocks. This implementation of a Ducros switch into the PPM solver will be referred to as the LD-PPM method henceforth.

3.2 Improved Understanding of Scramjet Isolator Flow Physics

A hybrid LES / RANS simulation of a back-pressured 2″×6″×24″ rectangular, constant-area isolator has been performed. The computational simulation was performed with conditions to match an experimental case where the shock train was observed to be in the middle of the isolator. The experimental isolator was 36″ long, so the computational exit boundary was set so as to match the static pressure at x=24″ in the experiment. A snapshot of Mach number contours at the center plane can be seen in Figure 9. This figure highlights the ability of REACTMB to capture the turbulent structure of the boundary-layers entering the shock region. Figure 10 shows a shock front tracking plot for 21.4 ms of computation. It can be seen that the shock train has a large movement downstream the first 15 ms, but that there are also higher frequency oscillations. These higher frequency oscillations have larger amplitude at the corners. The larger separated region associated with the corners causes these larger oscillations. Similar observations have been made in related experimental work. This simulation is incredibly expensive at 136 million cells, and a 0.15 μs time-step. Typically, it takes approximately 30 hours on 512 cores (15,360 CPU-hours) to advance the solution 1 ms of physical time.

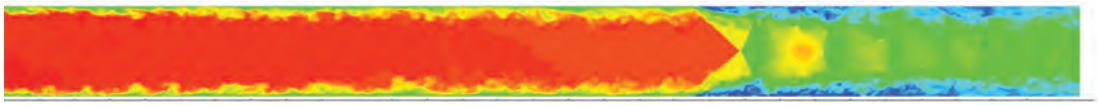


Figure 9. Centerplane Mach number

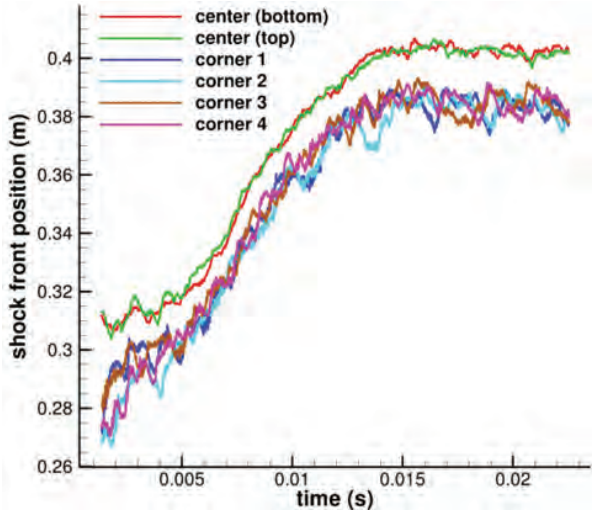


Figure 10. Transient location of shock front

3.3 Improved Understanding of Scramjet Injection Flow Physics

Hybrid LES/RANS simulations have been performed for sonic normal injection into a 5″×6″ rectangular duct with Mach 2 cross-flow. The computational grid consists of 19.9 million cells with constant streamwise and transverse spacing at approximately 19 cells per boundary-layer thickness in each direction except upstream of the injector and on the edges of the domain in the spanwise direction, where the spacing is stretched to approximately 15 cells per boundary-layer thickness. Wall normal spacing was also 19 cells per boundary-layer thickness outside of the boundary-layer, while clustering to the wall within the boundary-layer.

For these flows, RANS simulations typically under-predict the mixing of the jet in both the near-field and far-field. Hybrid LES/RANS simulation show moderate improvements in the far-field mixing, as seen in Figure 11, and moderate improvements in the near-field mixing, as shown in Figure 12. The LD-PPM approach shows an improvement in mixing in the near-field beyond the standard PPM. However, in the far-field, the standard PPM shows better agreement with the experimental measurements. Further investigations are underway.

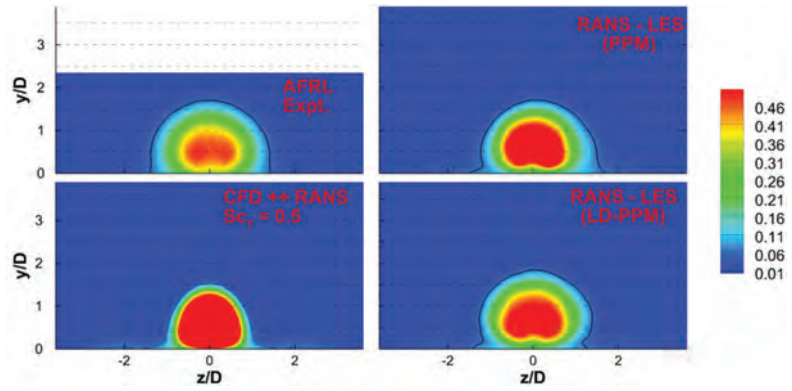


Figure 11. Injectant fraction 5 diameters downstream

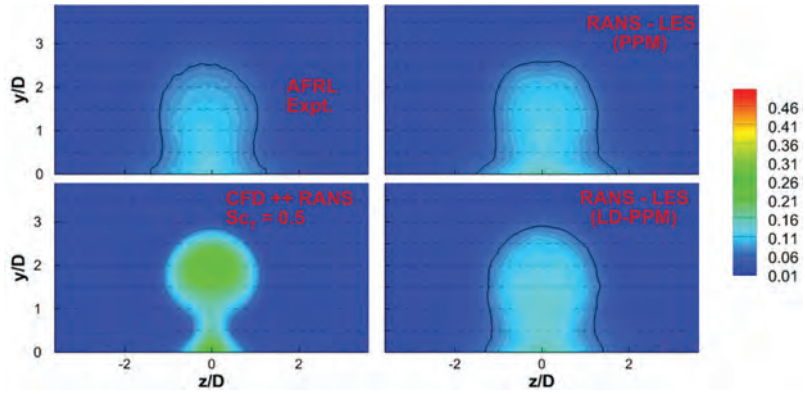


Figure 12. Injectant fraction 25 diameters downstream

Figure 13 shows the injectant mass fraction on the center-plane predicted by the LD-PPM approach in the top-half of the figure and by the PPM approach in the bottom-half. This confirms the ability of the LD-PPM to retain more small-scale turbulent structure due to its reduction in numeric dissipation when compared to the PPM technique, which allows for coarser grids than the standard PPM (Boles, et al., 2011). Similar observations can be made by comparison of the velocity tensor discriminant images shown in Figure 14.

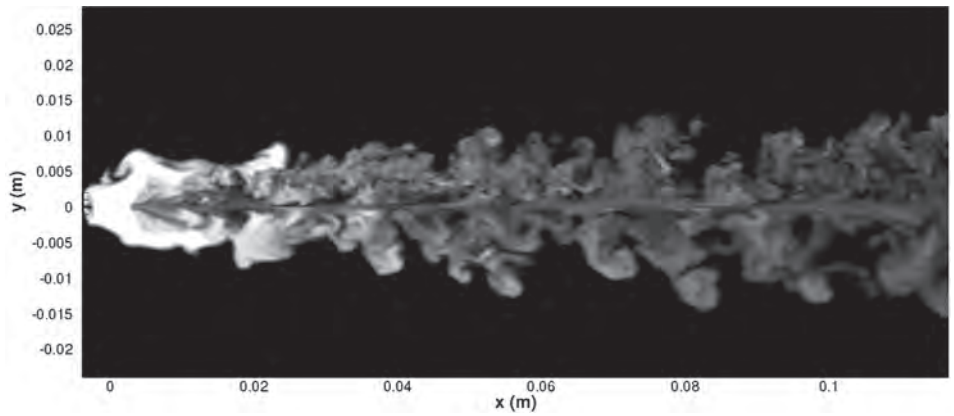


Figure 13. Comparison of turbulent structures captured by LD-PPM (top) and PPM (bottom)

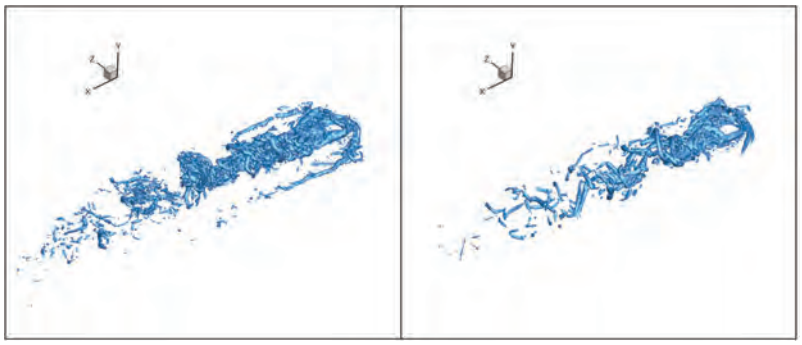


Figure 14. Velocity Tensor Discriminant for LD-PPM (left) and PPM (right)

4. Conclusion

Computational tools are being applied in two ways to aid in the development of large-scale scramjets. In the first case, conventional RANS CFD is being used in a design mode to develop wind-tunnel hardware that would allow large scramjet combustors to be tested at appropriate flight-like conditions. This first technique utilizes a relatively small amount of CPU resources for each simulation, but involves a large number of simulations. In the second case, higher-fidelity CFD using a hybrid RANS/LES approach is being used to more accurately model the flow in the engine. In this second case, a large amount of CPU resources are used for each simulation, so significantly fewer simulations are performed.

Acknowledgements

This work was funded by the US Air Force Office of Scientific Research (AFOSR) (Julian Tishkoff, Program Manager). The DoD HPC Modernization Program supported this project by supplying supercomputer time under the Computing Challenge Project C4S. Specifically, most calculations were performed at the DoD Supercomputing Resource Center at the US Army Research Laboratory, on the SGI Altix ICE 8200 named Harold. We would like to thank the staff at the Consolidated Customer Assistance Center (CCAC) who provided assistance in optimal use of these resources. Also, we would like to thank Cam Carter and Steven Lin at the Air Force Research Laboratory for their contributions to the research presented.

References

88 Air Base Wing Public Affairs, http://www.af.mil/information/factsheets/factsheet.asp?id=17986, 2011.

Boles, J., Milligan, R., Hagenmaier, M., and Edwards, J., "Hybrid Large-Eddy Simulation/Reynolds-Averaged Navier-Stokes Simulations of Sonic Injection into Mach 2 Crossflow", *AIAA-2011-769*, 2011.

Colella, P. and Woodward, P.R., "The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations", *Journal of Computational Physics*, Vol. 54, pp. 174–201, 1984.

Ducros, F., Ferrand, V., Nicoud, F., Weber, C., Darracq, D., Gacherieu, C., and Poinsot, T., "Large-Eddy Simulation of the Shock/Turbulence Interaction", *Journal of Computational Physics*, Vol. 152, No. 2, pp. 517–549, July 1999.

Edwards, J.R., "A Low-Diffusion Flux-Splitting Scheme for Navier-Stokes Calculations", *Computers & Fluids*, Vol. 26, No. 6, pp. 635–659, 1997.

Gieseking, D.A., Choi, J.-I., Edwards, J.R., and Hassan, H.A., "Simulation of Shock/Boundary-Layer Interaction Using Improved LES/RANS Models", *AIAA Paper 2010–111*, Orlando, FL, January 2010.

Gruber, M., Hagenmaier, M., and Mathur, T., "Simulating Inlet Distortion Effects in a Direct-Connect Scramjet Combustor", *AIAA-2006-4680*, 2006.

Hagenmaier, M., Eklund, D., and Milligan, R., "Improved Simulation of Inflow Distortion for Direct-Connect Scramjet Combustor", *AIAA-2011-233, 2011*.

McClinton, C., "X-43 – Scramjet Power Breaks the Hypersonic Barrier: Dryden Lectureship in Research for 2006", *AIAA-2006-1, 2006*.

Distortion Pattern Analysis for Diffuser-Fan Interaction

Michael List
US Air Force Research Laboratory, Propulsion Directorate (AFRL/RZTF), Wright-Patterson AFB, OH
michael.list@wpafb.af.mil

Abstract

Modern aircraft engines are subject to flowfield non-uniformities due to maneuvers, boundary-layer formation, separations, and secondary flows. Distortion which reaches the fan-face reduces stall margin and affects fan performance as tangential non-uniformities develop in the turbomachinery. The full distortion pattern may be composed of total pressure, total temperature, and flow angle (swirl) distortions. Quantification of the impact of complex distortion patterns, including both total pressure and swirl components, has become increasingly more important for engine design. To that end, the Compressor Aero Research Laboratory at Wright-Patterson Air Force Base is pursuing experimental and numerical testing of a coupled diffuser-fan system.

Simulation of an isolated modern serpentine diffuser has been conducted in order to quantify the content of distortion patterns produced by the specific class of diffuser. Post-processing of the diffuser simulations has resulted in an understanding of frequencies in the resulting distortion pattern. A spatial Fourier decomposition in the tangential direction of the total pressure field showed the dominant distortion frequencies. The impact of a distortion pattern depends upon the time rotor passages spend in the distortion, and the magnitude of the distortion. The high-magnitude 1/rev thru 5/rev content is most likely to propagate into the turbomachinery and cause performance detriment, whereas the higher frequency content will likely dissipate before reaching the fan-face. Presence of these multiple-per-rev frequencies has not been investigated in traditional analysis or experiment, and represents a significant departure from previous work conducted by the Fans and Compressors Branch.

It is hypothesized that understanding the frequency content of the distortion patterns present at a fan-face will allow modification of fan design parameters to provide distortion tolerance for the specific frequencies which pose the most dramatic tangential changes in the flowfield. This will allow future aircraft engines to operate more uniformly and without loss of stall margin under distorted conditions.

1. Introduction

The Compressor Aero Research Laboratory (CARL) has begun an investigation to experimentally and numerically quantify the flowfield in a closely-coupled diffuser-fan system, and to contrast the results with those obtained through traditional isolated component quantification. The experimental effort will move through logical stages: the fan and diffuser will each be tested in isolation; the fan will be tested with traditional total-pressure-only distortion screens; the coupled system will be analyzed. The numerical simulation of the first two items will be discussed in this paper.

Existing methods for quantifying total pressure distortion consider spatial variation of total pressure at the Aerodynamic Interface Plane (AIP), the location where the engine and airframe meet. Descriptors have been identified for intensity, extent, and multiple-per-rev content. Basic descriptions of time-averaged content plus the effect of unsteady variations have been quantified. This experience typically culminates in production of screens used to create worst-case patterns for steady engine tests. Results correlate with loss of performance and operation for inlets with little curvature. Recent publications have defined additional parameters which incorporate swirl distortion in light of a number of shortfalls of more traditional methods.

1.1 Operational Issues

Operational issues in a variety of aircraft spurred increased interest in swirl distortion. Flow angularity at the AIP in many inlet-fan systems is related to separation in curved ducts. These effects have been difficult to catch in ground test and

typically were found only at great expense and time. Many have been traced to flow angularity at the AIP.

In early Boeing 727 designs, the center inlet caused compressor stall if throttled to full-power below 60 kts in a crosswind that was greater than 10 kts. Similar limitations befell the Lockheed L1011 or TriStar. The Concorde number 4 engine was limited to 88% speed at takeoff due to vortices coming off the wing and intake sidewall. These were counter to the rotation of the engine and caused reduced operating range. Other examples include the Tornado[13,14], the Tomahawk missile[10,13], and the F-111[11].

1.2 Geometry

The Parametric Blade Study (PBS) was an effort during the 1980's which developed a number of high-performance transonic rotors. These were sequentially-numbered and were designed by various entities and tested by CARL. Rotor 4[7] was designed to be a state-of-the-art swept fan with diffusion control at the tip section. Its advanced design makes it still relevant today. More information on the design of the rotor is given in Table 1. The complete stage is being simulated with the spinner cone, providing a flowfield from the AIP location to the exit instrumentation plane. The stage efficiency during test was 88.5% and pressure ratio was 1.92.

Table 1. Rotor 4 design parameters

Design Parameter	Value
Number of Blades	20
Tip Clearance	0.010 inches
Rotational Speed	20,222 RPM
Flow Rate	60.77 lbm/s
Pressure Rise	1.980
Efficiency	93.1

A compact, serpentine diffuser was developed for the experimental study to generate a total pressure and swirl distortion pattern representative of those entering a fan-face in modern designs. The serpentine shape causes a pair of vortices to form at Top Dead Center (TDC), a thickened boundary-layer at Bottom Dead Center (BDC), and a vortex on each side of the AIP. These flow features result in a range of flow angularity in sectors large enough to impact the turbomachinery performance.

2. Methodology

2.1 Diffuser

The isolated diffuser was simulated using Fluent, StarCCM+, and Overflow at two conditions which bound the operation of the fan in terms of corrected flow. These correspond to a throat Mach number of 0.5 at the lowest flow condition (pre-stall at a low fan-speed) and Mach 0.7 at the high flow condition (choke at a high fan-speed). The simulations were conducted steady-state with at least second-order-accuracy in space, using appropriate upwind schemes. The k-ω SST turbulence model was used for its superior performance in diffusing and separated flows. The flow features which resulted at the AIP were consistent between the three solvers with modest variations attributable to the varying meshes.

2.2 Turbomachinery

The isolated fan stage was simulated using TURBO, a 3D unsteady Reynolds-averaged Navier-Stokes (URANS) solver[1–3]. It has been shown in many instances to be appropriate for multi-stage transonic turbomachinery analysis[4,5,8,9] and also analysis including inlet pressure distortion[6,15]. The turbulence model is a Center for Modeling of Turbulence and Transition (CMOTT) $k-\varepsilon$ model specialized for turbomachinery flows[12,16]. Each blade row is solved in the rotating reference frame and information is passed at each time-step with interpolations being performed to allow time-accurate, multi-stage interactions.

The inlet boundary condition was an isentropic inlet with total pressure, total temperature, and radial and tangential flow angles specified. In order to accommodate a two-dimensional (2D) pattern, the boundary was modified to store and interpolate from an input file any of the four boundary condition quantities. For all simulations, the total temperature was set to the constant total temperature used in the experimental facility for computing corrected flow. The inlet radial and tangential flow angles were set to zero. In doing so, swirl and radial distortion that formed in the diffuser simulations were ignored. This was intended to allow comparison of previous methods used in simulating distortion transfer. It also

allowed differentiation of the impact of total pressure and swirl distortion upon the performance of the stage. The exit boundary condition held an exit-corrected mass flow by adjusting local back pressure to set velocity vector magnitudes. This methodology has been used in both APNASA and TURBO.

The simulations discussed in this paper utilized the Cray XE6 architecture. Machines at the US Army Engineer Research and Development Center (ERDC) DoD Supercomputing Resource Center (DSRC) [Garnet] and the US Air Force Research Laboratory (AFRL) DSRC [Raptor] were used. Each turbomachinery simulation required 1,882 processors and 130 wall-hours (approximately 250,000 CPH) to reach convergence. Nine such points have been completed to date for a total of more than 2,000,000 CPH.

3. Results

This section describes the results of the isolated diffuser and isolated fan simulations. The diffuser was simulated to have the incoming flow conditions that will be provided in the CARL facility. The steady-state exit flowfield of the diffuser was extracted. The total pressure field was used as the inlet of the compressor. The compressor results discussed represent a single unsteady snapshot that is representative of the flowfield.

3.1 Diffuser Exit Analysis

Figure 1 shows the total pressure field computed for the diffuser for the two conditions at the AIP. Consistent flow features are present—at bottom dead-center (BDC) there is a thick boundary layer, at top dead-center (TDC) there is a pair of counter-rotating vortices, and on each side is an additional vortex. Aft of the diffuser recurve, flow separation at TDC results in the roll-up of two counter-rotating vortices. These vortices are the primary source of total pressure deficit at the AIP. On the sides, the additional vortices are formed as the diffuser contracts from a throat which is asymmetric top-to-bottom. This asymmetry induces a pressure differential which allows the vortex generation. Presence of the thickened boundary layer at BDC requires no explanation. The vortices in Figure 1(a) are asymmetric. It is theorized that this is a stability point for the low flow rate steady-state simulations. This result is repeatable, and is present in the Fluent and Overflow simulations.

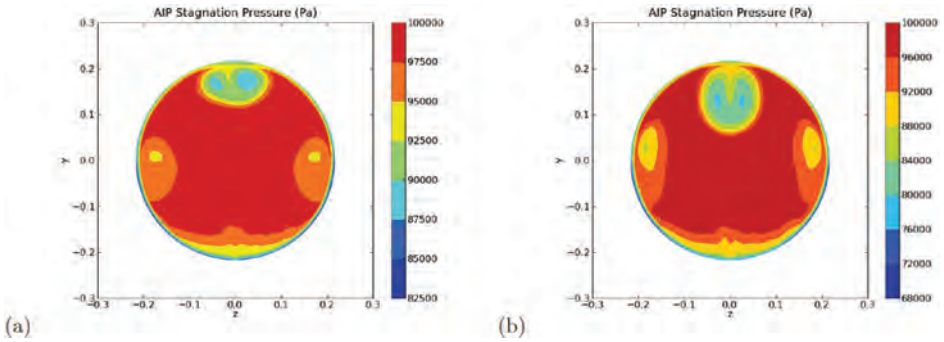


Figure 1. Total pressure distortion pattern at the diffuser exit for throat Mach number (a) 0.5 (b) and 0.7

The AIP patterns can be decomposed into per-rev components via a Discrete Fourier Transform (DFT) in order to better understand causes and effects related to the diffuser. Python's SciPy package was used to perform a DFT upon several radial lines taken from the exit of the steady-state diffuser total pressure pattern. Figure 2 shows total pressure magnitude versus tangential spatial mode for twenty-one radii at the exit of the diffuser. The spatial modes correspond to per-rev distortion components. Generation of these components can be tied to the secondary flow formation in the diffuser, and the result of particular components can be negative aeromechanic response. The 0 mode corresponds to the area-averaged total pressure and has been thresholded. The 1/rev thru 5/rev distortions can be seen over a range of radii to have large magnitude. While not negligible, the 6/rev thru 15/rev have less contribution. Figure 3 shows the decomposed total pressure patterns first thru sixth modes for visual comparison.

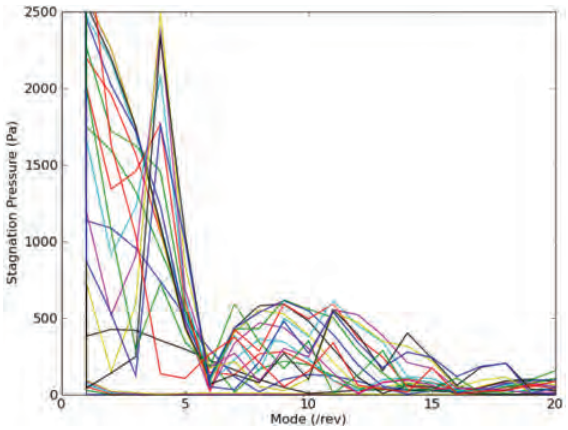


Figure 2. Magnitude versus mode for twenty-one radii across the diffuser exit

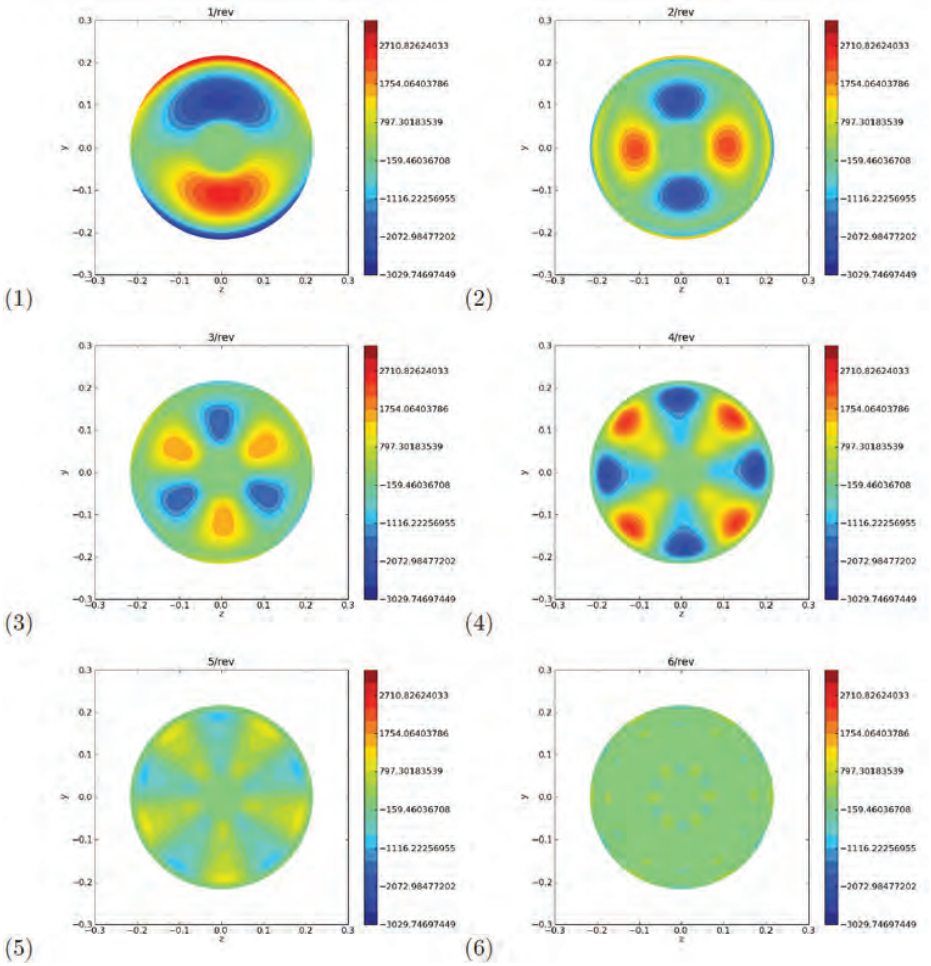


Figure 3. Total pressure distortion pattern per-rev components for the 1–6 modes

3.2 Turbomachinery Simulations

A computational compressor map is shown in Figure 4. The inclusion of the distortion patterns caused a reduction in the flow rates achieved by the fan stage. The points shown for the distorted inflow were in choke, and are labeled as `FA_M0.7_0` for the area-averaged distortion pattern and `FA_M0.7_FULL` for the full distortion pattern. The trend (though the study is not complete) was for the flow rate and pressure rise to decrease, shifting the speedline down and left. This result has been seen by numerous researchers.

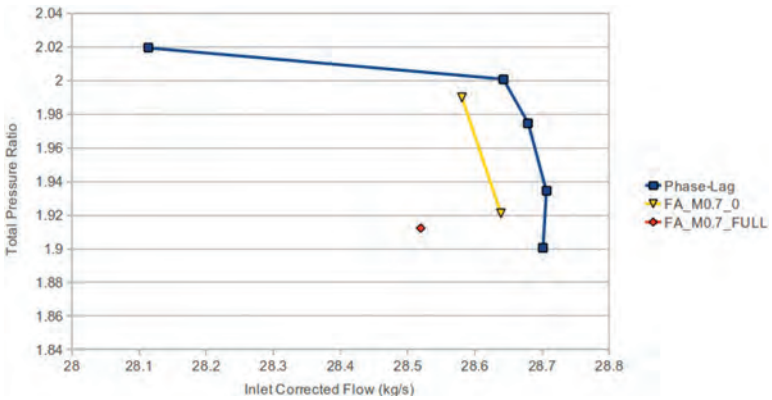


Figure 4. Computational compressor map

A plane was extracted at a radius of .155m, roughly mid-span of the stator. The results shown in Figures 5 thru 8 represent a preliminary effort to quantify the impact of total pressure distortion with the pattern from the diffuser. The contours shown are from an unsteady snapshot. Figures 5 and 6 show the contours of absolute total pressure and total temperature, respectively. For each of these figures, the distorted flow is at the top and the clean inflow is at the bottom. TDC is located on the left and right sides, and BDC is located at the center of the image. Flow direction is downward and the rotation direction for the rotor is to the right.

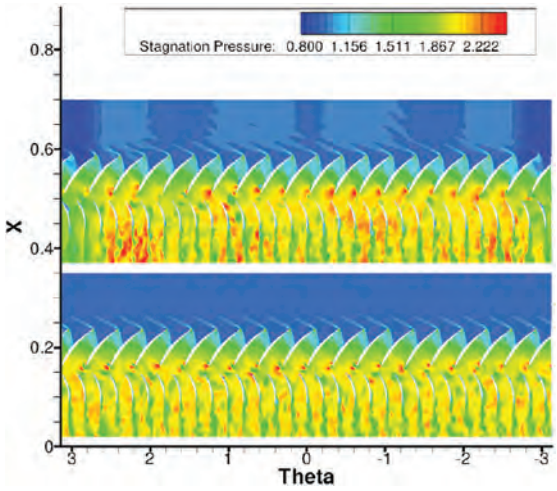


Figure 5. Absolute total pressure contours for distorted and clean inlet flows

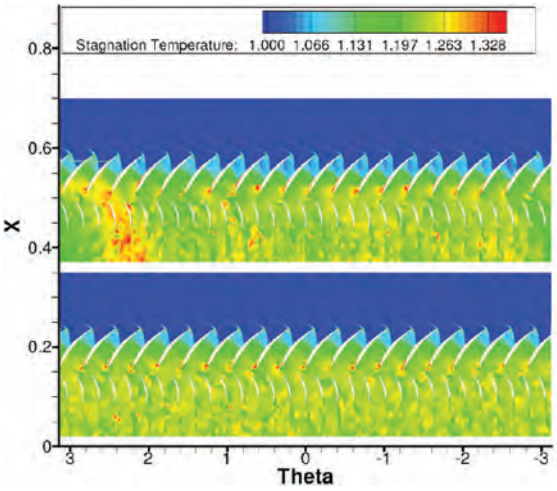


Figure 6. Absolute total temperature contours for distorted and clean inlet flows

The total temperature distortion that was generated by the change in compressor work is apparent in Figure 6. This was generated as a result of the uneven work performed by the rotor passing through the vortex system at TDC.

Static pressure is compared in Figure 7. Of most note is the overall decrease of static pressure from the clean inlet to distorted inlet case. The change in static pressure results in a change of tangential velocity. As the tangential velocity and axial velocity change, phase angle swirl distortion is formed. This follows the swirl formation discussed by Yao, et al.[15].

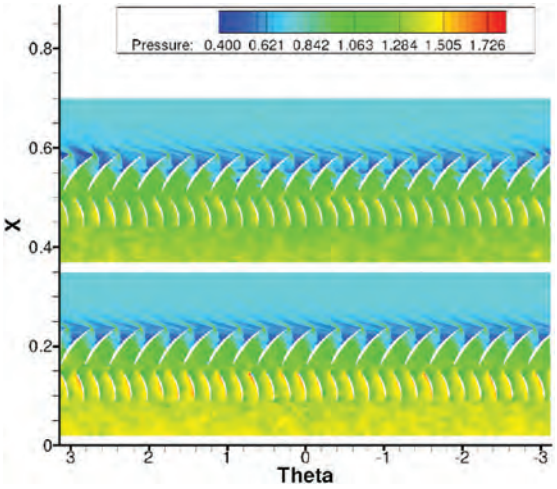


Figure 7. Static pressure contours for distorted and clean inlet flows

The contours of swirl angle, defined as tan^{-1} (C_θ/C_x), are shown in Figure 8. It can be observed that at TDC there was an increase in the magnitude of the swirl angle in the rotors as they passed through the distortion caused by the vortex pair. Also, the unsteady incidence on the rotor leading edge is apparent. This incidence change directly resulted in changes in expansion over the suction side to the passage shock, which was displaced as part of the process described by Yao[15]. Aft of the passage shock the swirl became more negative (to the right in an absolute sense). This is directly related to the work of the compressor in the particular rotor passage, and resulted in the increase in total temperature seen in Figure 6.

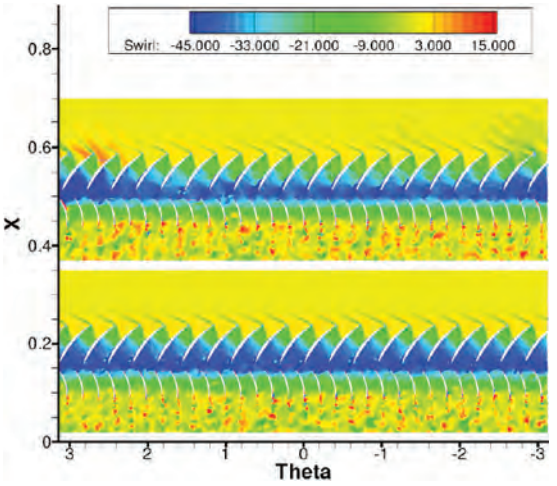


Figure 8. Swirl contours for distorted and clean inlet flows (magnitude is in degrees)

Figure 9(a) shows the circumferential variation of total pressure at the selected radius just downstream of the inlet. The clean inlet displayed a saw-tooth pattern as the upstream traveling bow shock decayed and interacted with the inlet boundary. The same observation could be made for the distorted inlet, which showed some perturbations, though a bit less clearly.

Figure 9(b) similarly shows the circumferential variation of total pressure near the exit plane. A shift can be observed in the distorted simulation of the total pressure deficit near top dead-center ($\theta=\pm\pi$). This shift was in the direction of the rotation.

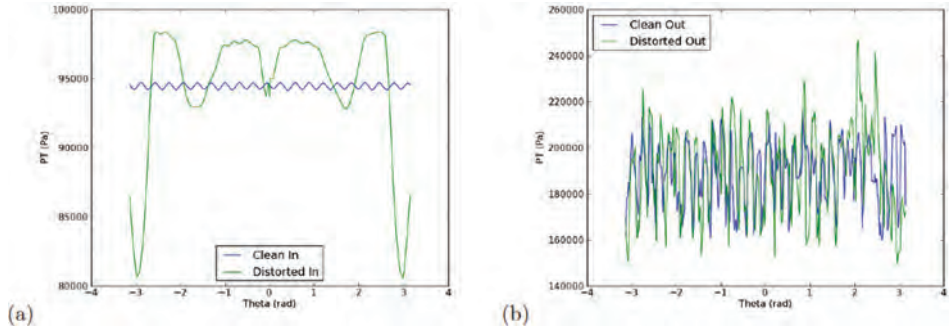


Figure 9. Total pressure signal around the annulus at the (a) inlet and (b) exit

The incoming and outgoing per-rev content has been determined through use of the spatial Fourier transform in the tangential direction. Figure 10(a) shows the inlet magnitude versus frequency for total pressure for both the clean inlet and distorted inlet. The clean inlet has only the 0/rev component. The inlet 0/rev components for the clean and distorted simulations are within 0.07% indicating that the boundary conditions held the inlet properly. Figure 10(b) shows the exit distortion component information. The exit 31/rev frequency (which matches the stator count) was higher for distorted flow indicating larger wake deficits.

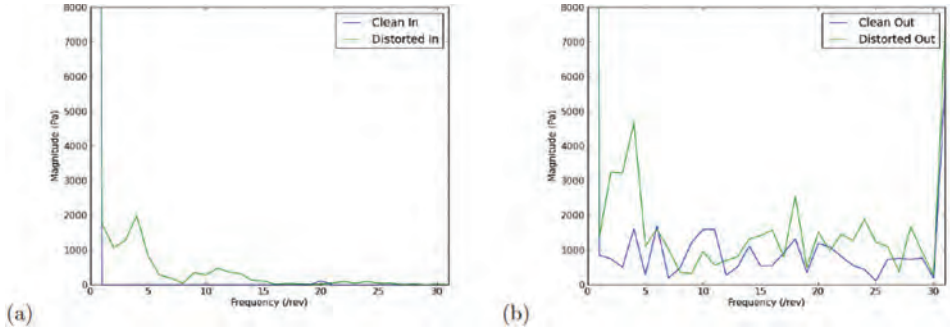


Figure 10. Frequency versus magnitude for the (a) inlet and (b) exit planes

Comparing the inlet signal content to the exit, as in Figure 11, amplification in magnitude for several of the frequencies can be seen. There was an observed amplification of 2/rev–4/rev components, which matched qualitatively previous findings by Gorrell[6]. None of the incoming frequencies were attenuated by the single-compressor stage.

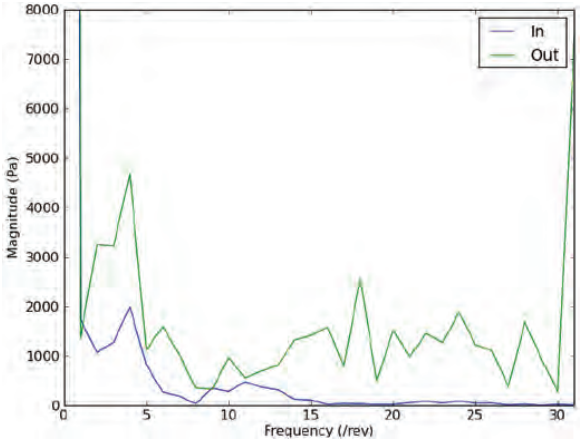


Figure 11. Frequency versus magnitude comparison for the inlet and exit of the distorted simulation

4. Conclusion

Simulation of an isolated diffuser representative of a modern, compact serpentine inlet system has been completed. The exit total pressure flowfield was extracted and used as the inlet for a single-stage transonic fan. The fan developed total temperature and swirl distortions which matched the physical phenomena seen in previous investigations. These solutions provide initializations for coupled system simulations and also allow analysis of the impact of various components of a distortion pattern. It is expected that a complete machine composed of several stages which ingests and consumes a distortion pattern would show a decrease in magnitude of several of the per-rev components.

Acknowledgments

The authors would like to thank the High Performance Computing and Modernization Program Office for supporting this work through computing resources, dedicated support staff, and generous allocations for both traditional and challenge projects. The authors would also like to thank the US Air Force Research Laboratory DoD Supercomputing Resource Center for support, assistance, and guidance.

References

1. Chen, J.P. and J.W. Barter, "Comparison of Time-Accurate Calculations for the Unsteady Interaction in Turbomachinery Stage", *ASME Paper 98-GT-3292*, 1998.

2. Chen, J.P. and W.R. Briley, "A Parallel Flow Solver for Unsteady Multiple-Blade Row Turbomachinery Simulations", *ASME Paper GT2001-0348*, 2001.

3. Chen, J.P., M.L. Celestina, and J.J. Adamczyk, "A new procedure for simulating unsteady flows through turbomachinery blade passages", *ASME Paper no. 94-GT-151*, ASME Turbo Expo 1994, 1994.

4. Gorrell, S.E., A. van de Wall, and F. Tsung, "Application of Time-Accurate CFD in Order to Account for Blade-Row Interactions and Distortion Transfer in the Design of High-Performance Military Fans and Compressors", *Proceedings of the HPCMP Users Group Conference 2005*, Nashville, TN, June 27–30, 2005.

5. Gorrell, S.E., T.H. Okiishi, and W.W. Copenhaver, "Stator-Rotor Interactions in a Transonic Compressor, Part 1: Effect of Blade-Row Spacing on Performance", *ASME Journal of Turbomachinery*, 125, pp. 328–335, 2003.

6. Gorrell, S.E., J. Yao, and A.R. Wadia, "High Fidelity URANS Analysis of Swirl Generation and Fan Response to Inlet Distortion", *Paper No. AIAA-2008-4985*, 44rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference, July 2008.

7. Law, C.H. and S.L. Puterbaugh, "Parametric Blade Study Test Report Rotor Configuration Number 4", AFWAL-TR-88-2110, Air Force Wright Aeronautics Labs, Nov 1988.

8. List, M.G., S.E. Gorrell, and M.G. Turner, "Investigation of Loss-Generation in an Embedded Transonic Fan Stage at Several Gaps Using High-Fidelity, Time-Accurate CFD", *Journal of Turbomachinery*, 132(011014), January 2010.

9. List, M.G., S.E. Gorrell, M.G. Turner, and J.A. Nimersheim, "High-fidelity modeling of blade-row interaction in a transonic compressor", *AIAA Paper No. 2007-5045*, 43rd AIAA/SAE/ASME Joint Propulsion Conference, Cincinnati, OH, 2007

10. Ludwig, G., "Tomahawk Engine/Inlet Compatibility Study for F107-WR-400/402 Engines", *Williams International Report CMEP 5003-2025*, October 1989.

11. Richey, G.K., *F-111 Systems Engineering Case Study*, Center for Systems Engineering at the Air Force Institute of Technology, March 2005.

12. Shih, T.H. and A.T. Hsu, "An improved $k-\epsilon$ model for near-wall turbulence", *Technical Report*, October 1991.

13. Society of Automotive Engineers, "A Methodology for Assessing Inlet Swirl Distortion", *Aerospace Information Report AIR5686*, 2010.

14. Stocks, C.P. and N.C. Bissinger, "The Design and Development of the Tornado Engine Air Intake", *AGARD CP-301, Paper 10*, May 1981.

15. Yao, J., S.E. Gorrell, and A.R. Wadia, "A Time-Accurate CFD Analysis of Inlet Distortion-Induced Swirl in Multistage Fans", *Paper No. AIAA-2007-5059*, 43rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference, July 2007.

16. Zhu, J. and T.H. Shih, "CMOTT Turbulence Module for NPARC", *Contract Report NASA CR 204143*, NASA Glenn Research Center, Lewis Field, OH, 2000.

Numerical Parametric Analysis for a Supersonic Combustor

Faure J. Malo-Molina
US Air Force Research Laboratory, Air Vehicles Directorate (AFRL-RBAC), Wright-Patterson AFB, OH
faure.malo-molina@wpafb.af.mil

Daniel Risha
US Air Force Research Laboratory, Propulsion Directorate (AFRL-RZAC), Wright-Patterson AFB, OH
daniel.risha@wpafb.af.mil

Houshang B. Ebrahimi
Flow Modeling and Simulations, LLC, Carson City, NV
houshang.ebrahimi.ctr@wpafb.af.mil

Datta V. Gaitonde
Ohio State University, Columbus, OH
gaitonde.3@osu.edu

Abstract

Substantial progress has been made with resources available under a Challenge Project in understanding some of the fundamental phenomena that contribute to the harsh environment encountered in scramjet flow paths. These analyses have been integrally connected to the HIFiRE-2 test program, for which extensive ground-test data is now available. The work of prior years is advanced by considering both the internal and external flow path. For the first, we augment our previous research on optimal injector placement to investigate the effect of fueling flow-rate. To this end, fuel is injected upstream of the cavity at a 30-degree angle to the freestream to generate different equivalence ratios (Φ). The code employed for this analysis, HEAT3D, is highly-scalable and has previously been employed to solve a variety of supersonic and subsonic combustion problems. Combustion is modeled using a finite-rate, ethylene-air chemistry model. As Φ is increased, the flow successively becomes more reacting and ash-back appears i.e., the flame moves upstream of the flame holder. The augmented fuel aids stabilization of the cavity shear-layer, but ultimately thermal choking results at Φ=1.0 or higher. The results provide insight into the stability of the flight vehicle under conditions when the shroud is discarded at high-speed.

1. Introduction

This Challenge Project exploits massively-parallel computational capabilities of the Department of Defense High Performance Computing Modernization Program (DoD HPCMP) to provide a significant computational element to an international program known as HIFiRE, administered by the US Air Force and Australian Defense Science and Technology Organization (DSTO), with NASA support. The recent pioneering success of the X-51 flight test has triggered substantial interest in making such flight routine. To realize this vision; however, there is a critical need to generate enabling insight into the physics of the accompanying fluid, combustion and structural interactions. The HIFiRE program provides a unique opportunity to obtain such understanding because of its emphasis on scientific exploration, rather than on engineering demonstration. This project leverages high-fidelity simulations to extract as much insight as possible from extraordinarily difficult and expensive ground- as well, as flight-tests. We seek to provide a clearer understanding of the manner in which the fluid and combustion dynamics interact to adversely impact thermal load, drag, inlet distortion, combustion, and thus thrust generation. Upon successful conclusion, the results of this effort will guide vehicle evolution to mitigate or manage each of these complex factors that presently preclude routine sustained hypersonic flight.

The hypersonic vehicle design is depicted in Figure 1.

The HIFiRE-II inlet consists of an inward-turning inlet coupled to a rectangular cross-section combustor. The configuration exhibits overall vertical and horizontal symmetry: the inlet conditions the air from its on-design flight to yield temperatures at auto-ignition level at the combustor entrance. A simplified dual-split nozzle is designed to exhaust the flow downstream of the payload away from the flight vehicle. Figure 1 shows the analyzed geometry of the HIFiRE-II payload as follows: 1) the metallic yellow delineates the outer surfaces of the un-shrouded inlet with its slightly blunt lips, 2) the interior walls of the inlet are colored green, 3) the internal inlet section and isolator region are in yellow, 4) the burner section is colored orange, and finally, 5) the nozzle is portrayed in red.

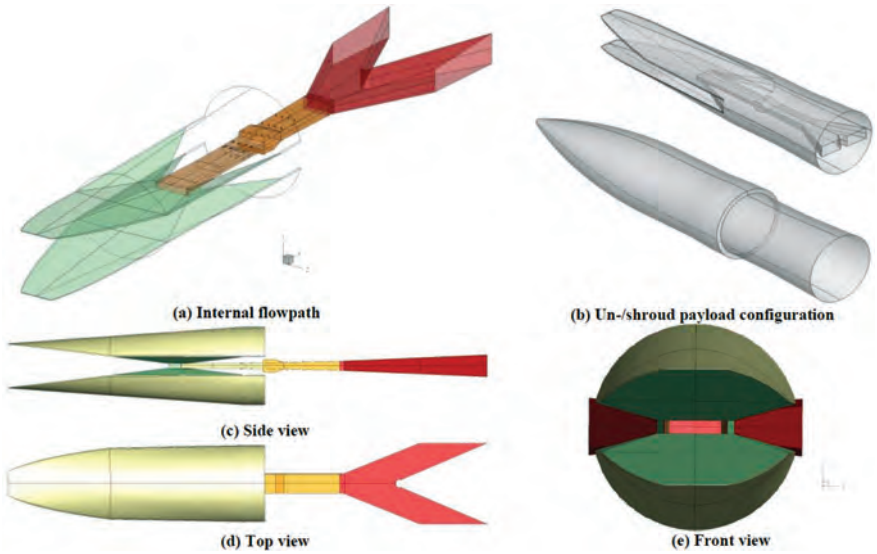


Figure 1. HiFire-II fore-body/payload configuration (not at scale)

One of the major challenges for the scramjet designers is to properly mix the high-speed air with the fuel to induce efficient and rapid combustion, since the residence time in the device is of the order of milliseconds. An additional complication is that even in streamline-traced inlets, as previously presented by the authors, flowfield distortions in the inlet have a substantial and complex impact on the combustor performance. Entropy gains which result in pressure losses can also engender gradients that can, under certain conditions, enhance mixing, and thus combustor performance.[1,2] Thus, two principal fluid dynamic issues that must be examined are: 1) the three-dimensional interplay between viscous and inviscid interactions including shock-boundary layer and shock-shock interactions in the inlet and corners of internal passages, and 2) the effects due to supersonic reactive flow mixture of the surrogate fuel to air in the combustor.[3,4]

In previous efforts, we have extensively explored the broad phenomena encountered in high-speed propulsion devices under different conditions. Uniform and non-reacting assumptions in supersonic combustors resulted in lesser mixing efficiency overall. These were believed to be caused by a poorer penetration and a weaker vortex driving in the cavity, most likely owed to distortions upstream into the inlets. In fully-coupled geometries, shock/boundary-layer interactions encountered in the inlet resulted in a slight increase in the mixing efficiency. However, higher distortion upstream and larger shear separation also increased the tendency to un-start the engine. In the present effort, we consider a ground test situation, with uniform flow entering the combustor region. Distortions associated with the inlet are ignored, and the focus is on complex mixing phenomena, non-equilibrium transfer of turbulence energy, and interactions between turbulence and chemical kinetics that may impact both the chemical reactions and turbulence field downstream of the system.

2. Theoretical/Computational Model

As above, the Reynolds-averaged Navier-Stokes (RANS) approach with k-ω Wilcox closure is adopted because of its lower computational requirements.[5] However, because the model is based on semi-empirical considerations and calibrated for very simple situations, the technique must be evaluated carefully.[6] Previous efforts have indicated that for the class of flows under current examination, the RANS approach successfully captures the key trends, though details of vortex shear-layer interactions in the vicinity of the cavity are not adequately reproduced.[7] The code employed in this portion of the research is HEAT3D, which has previously been employed to solve a variety of supersonic and subsonic combustion problems. Extensive validation of HEAT3D treating fully-coupled, three-dimensional flows with finite-rate chemistry has been reported in previous studies.[8,9] The code uses an implicit time integration and strongly conservative finite-volume formulation. Combustion is modeled using a finite-rate, Ethelyne chemistry model and Gordon/McBride thermodynamic curve fits. Fourteen gaseous chemical species are considered (C_2H_4, C_2H_2, CO_2, CO, OH, O_2, O, H_2, H, H_2O, NO, N, and N_2), with 20 chemical reactions applicable to a reduced reaction mechanism for reactants and products for Ethylene fuel and air oxidizer (see Table 1). Although this chemical model has yet to be validated for the ground-testing efforts and under these exact conditions, the parametric analysis reveals important trends to be considered when fueling. Current research is underway to validate and verify this chemical mechanism with a few experimental tests to be presented in the third year of this Challenge Project.

Table 1. Chemical mechanism for: fuel (Ethylene+Methane) and air mixture

Reaction 1:	$C_2H_4 \leftrightarrow C_2H_2 + H_2$	Reaction 11:	$H + OH + M \leftrightarrow H_2O + M$
Reaction 2:	$C_2H2 + O_2 \leftrightarrow 2CO + H_2$	Reaction 12:	$H + O + M \leftrightarrow OH + M$
Reaction 3:	$CO + O + M \leftrightarrow CO_2 + M$	Reaction 13:	$2O + M \leftrightarrow O_2 + M$
Reaction 4:	$CO + O_2 \leftrightarrow CO_2 + O$	Reaction 14:	$N + NO \leftrightarrow N_2 + O$
Reaction 5:	$CO + OH \leftrightarrow CO_2 + H$	Reaction 15:	$N + O_2 \leftrightarrow NO + O$
Reaction 6:	$OH + H_2 \leftrightarrow H_2O + H$	Reaction 16:	$N + OH \leftrightarrow NO + H$
Reaction 7:	$O + OH \leftrightarrow O_2 + H$	Reaction 17:	$H_2 + O_2 \leftrightarrow 2OH$
Reaction 8:	$O + H_2 \leftrightarrow OH + H$	Reaction 18:	$2H + H_2 \leftrightarrow 2H_2$
Reaction 9:	$2OH \leftrightarrow O + H_2O$	Reaction 19:	$2H + H_2O \leftrightarrow H_2 + H_2O$
Reaction10:	$2H + M \leftrightarrow H_2 + M$	Reaction 20:	$2H + CO_2 \leftrightarrow H_2 + CO_2$

The selected gaseous fuel injection conditions correspond to several equivalence ratios with a fuel temperature of 300K, and the density and pressure were varied to obtain the different equivalence ratios. The main flow corresponds to a Mach number of 2.2, static temperature and pressure equal to 900K and 58.7 kPa, respectively.

3. Results

3.1 HIFiRE-2 Direct Combustor Configuration and Computational Grid

The configuration is based on the HIFiRE-2 type combustor configuration, with a duct and cavity-based flame holder to facilitate a study of the fueling strategy, mixing, ignition and combustion dynamics. To this end, we consider several simulations to explore the effect of different parameters. Both frozen and chemically-reacting flows (with Ethylene fuel) are examined. The open cavity employed as a flame holder has a downstream ramp inclined at 30 degrees. Fuel is injected upstream of the cavity at a 30-degree angle to the freestream, to generate different equivalence ratios. Several calculations are now presented, with the goal of examining the effect of distortion associated with the inlet and combustor fueling in HIFiRE-II (see geometry in Figure 1): 1) injection with frozen chemical assumptions and 2) finite-rate reacting conditions. Different fueling mixtures were assumed for injection at the 1st row; these are classified in Table 2, below.

Table 2. Numerical methodologies/assumptions for cases

Cases	Φ_{fuel}	Chem. Assumption
1	1.00	Frozen
2	0.40	Finite-rate
3	0.55	Finite-rate
4	1.00	Finite-rate
5	1.20	Finite-rate

Figure 2 shows the structured multi-zone grid structure with a subset of the mesh density for clarity in the combustor region. The overall length of the inlet segment is 1.7 m, where the nozzle section is located. The mesh considered herein consists of about 12 million grid-points. The points are distributed to resolve the flow near the walls, in the cavity and around the fuel injectors. These structured-body conforming meshes are employed with emphasis on ease of generation, and limited to just ¼ of the geometry. The leading-edge of the configuration forms a coordinate surface. The mesh is clustered towards the outer-walls and around the injectors to capture the proper flow-physics and their interactions (shock-to-shock/shear-layer). The area of the average grid-sized cell is about $8\times10^{-6}m^2$. For the on-design cases, symmetry boundary conditions are employed at the vertical and horizontal planes (thus only one-fourth of the configuration is simulated). On the outside far-field (not shown herein), a fixed uniform inflow condition is specified. The outflow boundary condition at the combustor exit is modeled using first-order extrapolation. No-slip adiabatic wall conditions are applied on all solid walls.

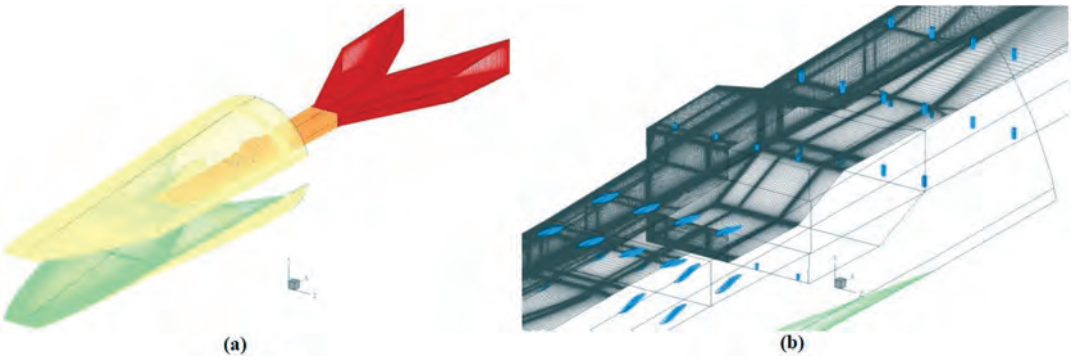


Figure 2. Computational grid: a) integrated HIFiRE-II vehicle, and b) 1/2 of the combustor configuration

3.2 Frozen Chemistry

The first case considered is the Φ=1.0 case, under frozen conditions. Figure 3a and b show the frozen solutions for the Mach number and mass fraction of fuel (C_2H_4). The vertical planes are positioned at the center of the injector at z=0.5 and 1.5 inches from the vertical symmetry plane. The five cross-sectional planes in Figure 3b are positioned at x=10, 14, 18, 22, and 26 inches downstream from the entrance. The Mach number contours show the primary shock interaction and shear layer destabilization produced by the strong injection (see Figure 3a). The shock system produced by the injection interaction with the isolator flow is readily discernable. The frozen assumption shows a peculiar effect on the shear-layer as it goes across the cavity. The flow is non-uniform and yields a high-expansion in the cavity region, followed by a contraction (high-pressure zone) at the center of the duct and near the end of the ramp. Downstream of the cavity, the flow expands once more to supersonic speed, forming another shock train with multiple interactions, further increasing the distortion of the cross-sectional profile.

In that same figure, some points were taken at the injection interface to trace streamlines of fuel added into the combustor. The vectors of these streamlines followed a very three-dimensional pattern and follow different composite trajectories. Since the injected fuel is assumed to be frozen (not-reacting), the traces expand downstream and continue mixing with the inflow air up to the exit plane (see Figure 3b for the iso-surfaces of fuel of the highest concentrations). This effect is highly influenced by the cavity and the side walls on the vertical and horizontal directions, respectively. Similarly, the shear-layer is pushed further away from the top-wall on the right side-wall (looking downstream) than the section near the symmetry plane.

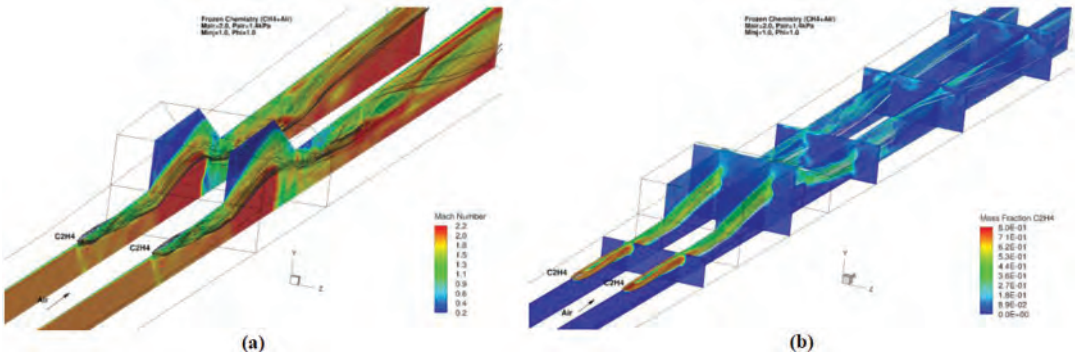


Figure 3. Contours and fuel streamlines for a) Mach and (b) fuel-mass fraction iso-surfaces (Case 1)

To further understand the dynamics, Figure 4a and b shows the swirl and vorticity magnitude, respectively. The swirl contours shed light on the complex circulation patterns in the cavity with upstream injection and its transverse irregularity caused by the wall. As shown in this figure, the multiple-eddy regions and scales can augment mixing and penetration of fuel downstream (Figure 4a). Moreover, the size of circulation inside the cavity and the strength of the vortices are influenced by several parameters: inflow condition, injection momentum ratios, cavity L/D ratio among others.[10] Figure 4b with the vorticity magnitude shows the shear-layer and the extent of turbulent contributions as they interact with the injectors, and disrupt the duct flow further downstream of the cavity due to a high-pressure interaction after the ramp.

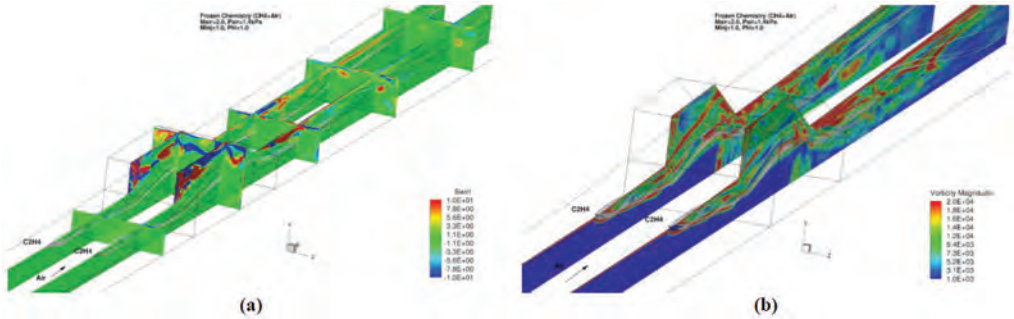


Figure 4. Contours and fuel streamlines for (a) Swirl and (b) Vorticity magnitude (Case 1)

3.3. Finite-Rate Chemistry

Figure 5 shows the Mach number and pressure contours for Case 2. Most of the differences from the frozen case are associated with the higher temperatures generated. Unlike the frozen assumption (Case 1), the solution is steadier and yields a small contraction around the cavity region, due to chemical reactions augmenting the temperature of the flow. In addition, the injection effects are not as apparent at the symmetry plane as in the frozen case. A separation region is generated just upstream of the injection slot by the intersection of the boundary-layer and the oblique shock-wave. Since the intensity of the shock-wave is much stronger than that of the expansion-wave, after the interaction of these two waves (top and bottom), a high-pressure region is formed on the top-wall of the combustor, which is just ahead of the divergent part of the combustor. Initially, when reactions start later from the injection and fuel is ignited, the separation zones upstream of the injection slot grow to be larger than those under the cold-flow (frozen) condition.[11] The pressure and streamlines calculated from both ports seem more symmetrical. The Mach and temperature contours show the wall-effects on the downstream cross-sectional planes. In this case, reactions give rise to pressure past the cavity, rather than a geometrical effect of expansion at the step, followed by a contraction at the ramp, as in the frozen case.

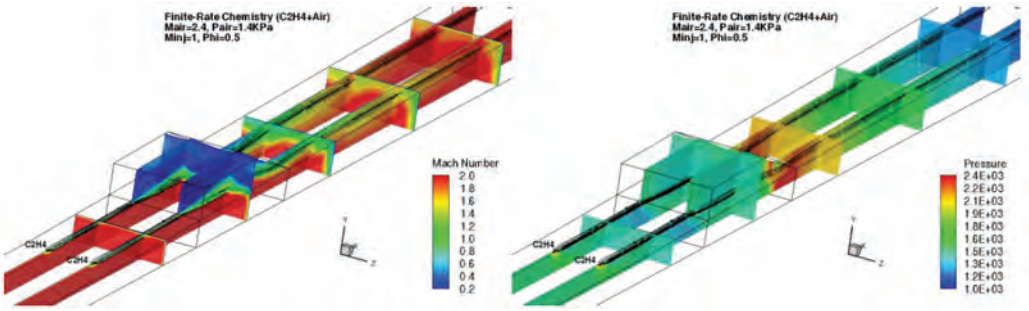


Figure 5. Mach number (left) and Pressure (psf, right) contours at vertical planes intercepting injectors and five cross-sections downstream (Case 2, finite-rate assumption)

Figure 6a and b shows the swirl and vorticity magnitude levels at five cross-sectional planes positioned at x=10, 14, 18, 22, and 26 inches downstream from the entrance. Figure 6c shows the water-mass fraction concentration iso-surfaces and contours of five downstream stations at x=10, 12, 14, 16, and 18 inches. Neither the swirl nor the vorticity magnitude reflects such a complex flow pattern as in the frozen case. Although the cavity helps mix the fuel and most reactions do take place within that region (see Figure 6c for concentration of water produced and Figure 5 for temperature), chemical reactions seem to lessen the circulation intensity inside the cavity and influence the vorticity produced by the shear-layer downstream as the flow expands towards the outflow.

The integrated Φ (equivalent fuel-air ratio) for Case 3 is 0.55, and the volume-averaged temperature inside the cavity is relatively higher than Case 2 over a larger zone (above 3,000K). The gases inside the cavity are insufficiently mixed. In this case, the injected fuel flow aligns with the structure of the main re-circulation zone and initially fills the bottom of the cavity. Consequently, the combustion zone is mainly located near the cavity floor. The unburned hydrocarbon is assumed to be ethylene and no further reaction will take place for CO; there is insufficient oxygen to consume unburned hydrocarbons and CO locally for conditions with and without cavity fueling. In other words, gas mixtures within the cavity flame holder are already relatively rich in the present flow path, in spite of the fact that combustion products constitute the majority of the gas mixture.

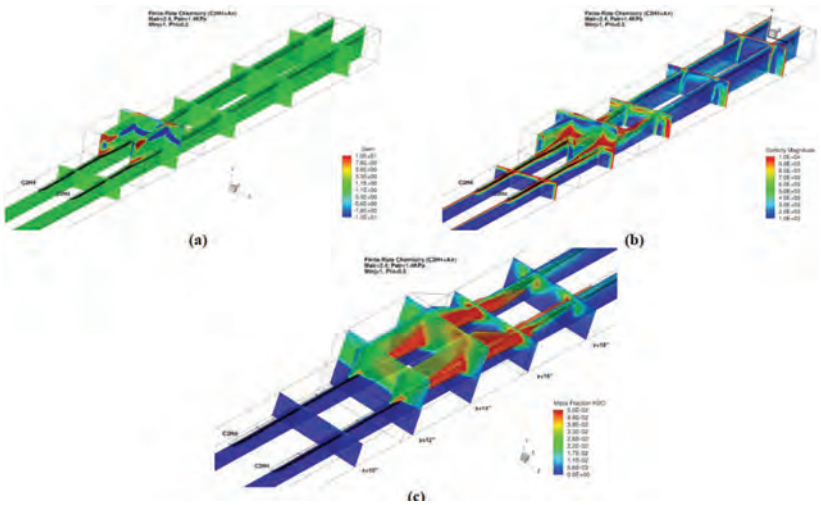


Figure 6. Resulting: a) Swirl, b) Vorticity magnitude and c) mass fraction of formed water concentrated near the cavity contours at vertical planes intercepting injectors and five cross-sections downstream (Case 2, finite-rate assumption)

Figure 7, as in Figure 5, shows a higher velocity flow in the main-stream region, which results in poorer fuel penetration. This effect pushes the fuel closer to the walls downstream from the cavity's ramp. The pressure variations due to shock interactions are slightly higher in this case, especially near the core region, where shock interactions exist (see Figure 7). The slightly higher fuel-air ratio seems to yield a more stable boundary-layer with lesser interactions (compare Mach number contours for Cases 5 and 6, Figure 5, and Figure 7). As shown in previous cases, the shear-layer on this configuration is pushed away from the vicinity of the step where the 30o injectors at the bottom of the cavity are situated.[12] This causes a circulation region ahead of the jet and near the back wall, opening a low-speed pocket that allows the fuel to flash back toward the combustor's entrance. The rapid and more abundant reactions within the shear-layer increase temperature in the cavity and near the wall. The increase of fuel also enhances a larger pressure region from the thermal expansion.

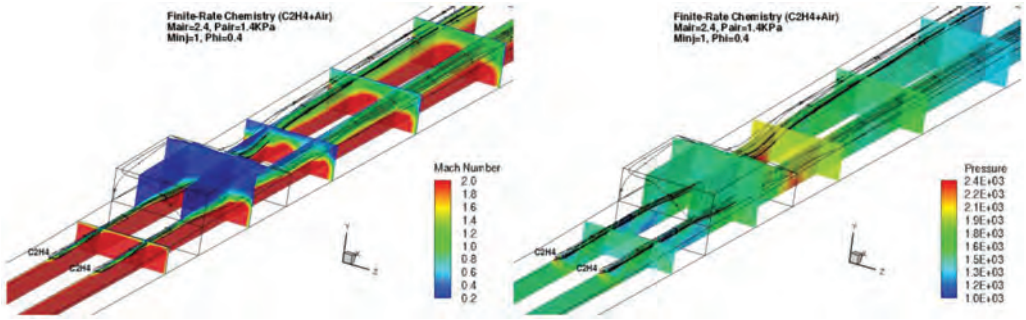


Figure 7. Mach number (left) and Pressure (psf, right) contours at vertical planes intercepting injectors and five cross-sections downstream (Case 3)

Figure 8 shows the Mach number and pressure for Case 4 (Φ=1.0). The flow experiences a sharp reduction in Mach number downstream of injection. Relative to previous cases, the temperature is higher downstream of the cavity and in the cavity as well. For this case, unburned fuel is observed further downstream of the injectors relative to the previously discussed Φ values. In the cavity, all fuel appears to be burnt very well, while downstream of the cavity (at the exit of the combustor), most of the fuel is consumed.

The final case has a Φ=1.2 and the average temperature inside the entire system is much higher than the two previous cases (Cases 2 and 3), since twice as much fuel is being burned overall. The gas mixtures within the cavity flame holder are rich in the present case. The Mach number contours show the flow to be thermally-shocked, with a strong shock perpendicular to the flow (see top-left part of Figure 9). This is caused by the high combustion pressure from the larger amount of fuel reacting (see top-right corner of Figure 9). Unlike previous reacting cases, Case 5 shows that a much slower subsonic speed is predominant in the flow, which results in poorer overall performance. This effect pushes the fuel closer to the walls downstream from the cavity's ramp. The temperature and pressure variations due to shock interactions are slightly higher in this case, especially near the core-region, where shock interactions exist. The slightly higher fuel-air ratio seems

to yield a more stable boundary-layer with lesser interactions. This causes a circulation region ahead of the jet and near the back wall, opening a low-speed pocket that allows the fuel to flash back toward the combustor's entrance. The rapid and more abundant reactions within the shear-layer increase temperature in the cavity and near the wall. The increase of fuel also enhances a larger pressure region from the thermal expansion.

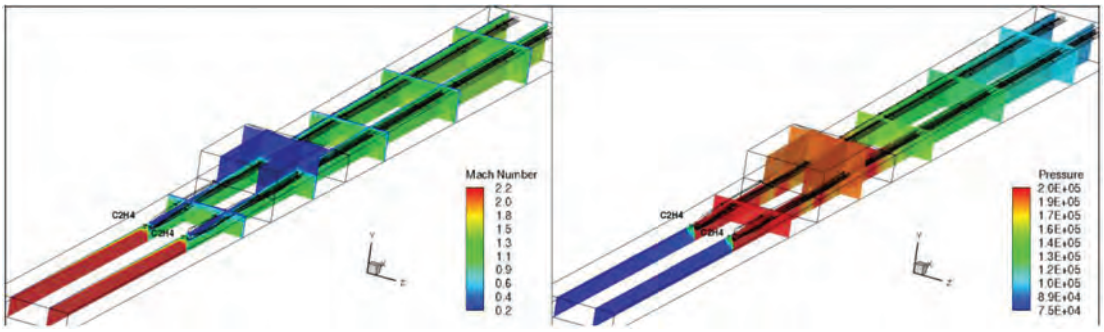


Figure 8. Mach number (left) and pressure (psf, right) contours at vertical planes intercepting injectors and five cross-sections downstream (Case 4)

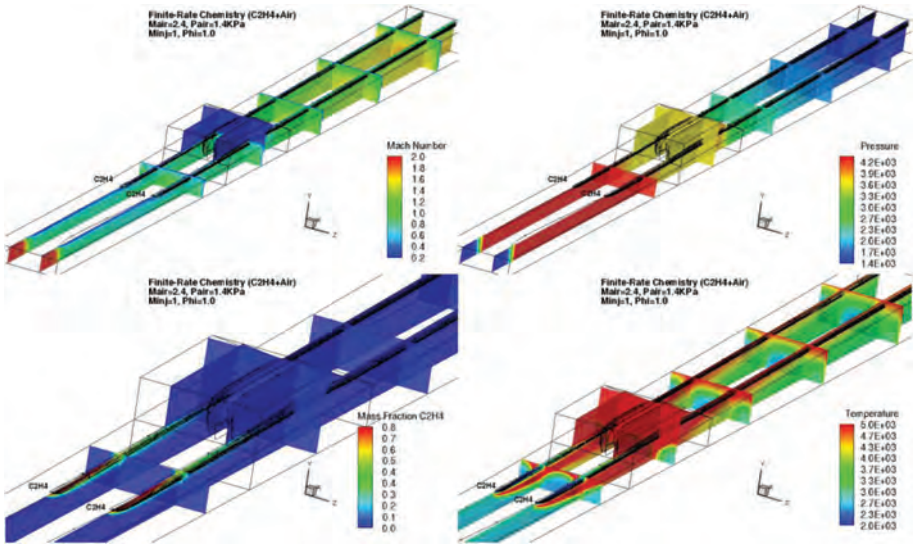


Figure 9. Mach number (top-left), pressure (psf, top-right), fuel mass fraction (bottom-left) and temperature (R, bottom-right) contours at vertical planes intercepting injectors and five cross-sections downstream (Case 5, finite-rate assumption)

To further understand the dynamics, Figure 10a and b shows the swirl and vorticity magnitude. The swirl contours shed light on the complex circulation patterns in the cavity with upstream injection and its transverse irregularity caused by the wall. As shown in Figure 10a, the multiple-eddy regions and scales can augment mixing and penetration of fuel downstream. Moreover, the size of circulation inside the cavity and the strength of the vortices are influenced by several parameters: inflow condition, injection momentum ratios, cavity L/D ratio among others. Figure 10b with the vorticity magnitude shows the shear-layer and the extent of turbulent contributions as they interact with the injectors, and disrupt the duct flow further downstream of the cavity due to a high-pressure interaction after the ramp.

Generalizing, when we compared all the cases, the boundary-layer separates from the upstream lip and reattaches downstream. As the boundary-layer separates from the leading-edge of the cavity, a free shear-layer forms. Depending upon the pressure inside the cavity, the shear-layer deflects upwards or downwards producing a compression. For these cases, a compression wave appears at the leading-edge of the cavity. Unlike the frozen case (Case 1), the finite-rate calculation (Case 4) exhibits a high-compression wave in a planar form or normal disk, which appears upstream of the cavity just after the injection location. This is a result of the rapid reactions within the shear-layer and the high-combustion pressure, which induce the reacting flow to move upstream and produce significant increases in temperature. This strong shock wave can be seen in Mach number and pressure contours.

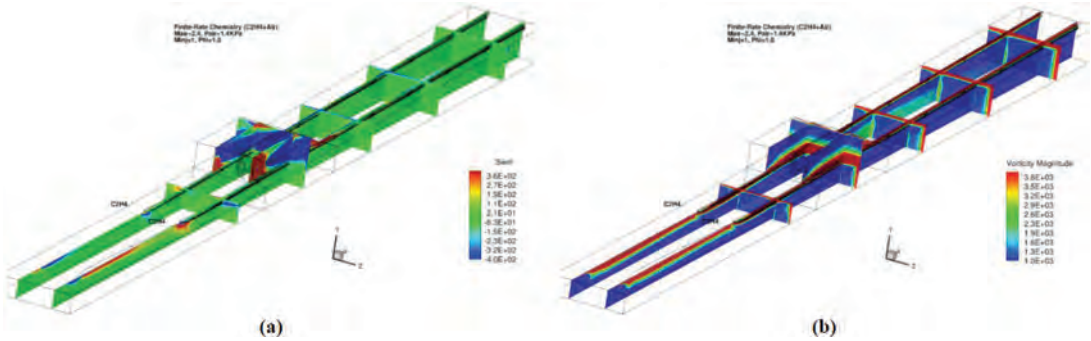


Figure 10. Contours and fuel streamlines for (a) Swirl and (b) Vorticity magnitude (Case 4)

The water-mass fraction and temperature line across the exit profile compare the reacting and non-reacting cases with a uniform inflow condition. The results obtained for the Φ=1.0, 0.5, and 0.4 cases at the exit of the combustor are shown in Figure 11. Comparison of these cases reveals that the trend is very similar for all the reacting cases at different Φ levels. As Φ level increases, the average temperature increases due to larger amounts of fuel reacting with air. Higher values of Φ yield smaller gradients or variations. Line and contour plots of Mach number, temperature and water-mass fraction at the exit of the combustor show that the flowfield is not uniform and the gradient of fluctuations is sizable (Figure 11).

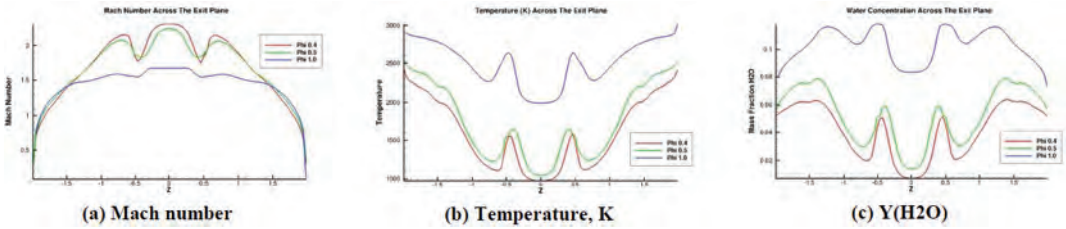


Figure 11. Line profile across the exit plane of the combustor for Cases 2–4

Figure 12 compares the upper-wall pressures in the streamwise direction and along the injector positioned closest to the symmetry plane for all cases. As shown for the higher added fuel, the Φ=1.2 case displays a high-pressure region near the inflow. This effect shows that the configuration is thermally-choked. This high-pressure compression disk arises upstream of the combustor due to very rapid reactions within the core flow. The line pressure for Φ=1 shows very similar trends as Φ=1.2, and a tendency towards instability may be postulated upstream of the cavity region for these higher values of Φ. Lower values of Φ do not show any sign of a high-compression region upstream of the injection due to lower combustion pressure. The pressure lines show similar developments near the step, where the pressure decreases slightly, followed by an increase due to the ramp. Past the cavity, the flow-pressure settles down due to the increase of area ratio towards the exit-boundary. The low Φ reacting cases (Cases 2 and 3) illustrate very similar wall-pressure trends compared to the frozen case (Case 1), with the exception of the cavity zone where the pressure is augmented due to the chemical reactions. The frozen case shows the largest pressure fluctuations due to multiple shock interactions along the total length of the combustor.

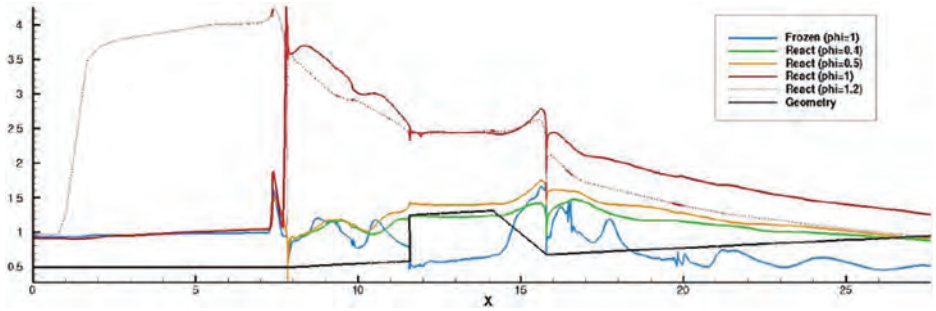


Figure 12. Pressure ratio at the walls center along the injection port plane (z=0.5 in), Cases 1–5

The results for this analysis were limited to the injection on the first row. Further research will help understand the influence of other injection rows and/or multiple location ports simultaneously. Also, additional injection sequencing and strategies must be investigated parametrically, and then fully-coupled to its rectangular inlet to achieve required efficiencies, stable combustors and larger thrust-to-drag ratios. The present effort represents a preliminary analysis of the chosen configuration. Future work will explore other injection strategies, as well as the effect of the actual fuel to be employed (an Ethelyne-Methane mixture). In addition, the distortion associated with the inlet, which is ignored in the present ground-test experiment, as well as modifications to the cavity structure, will be examined.

In a separate effort to explore predictive capability for scramjet flow paths, twelve different experimental cases were used to validate two chemistry models with the commercial CFD++ code. In particular, the goal is to overcome challenges pertaining to RANS steady-state, high-speed chemically-reacting flow. The evaluation is performed using experimental data for two different direct-connect 1x-axi-symmetric scramjet combustor flow path configurations at various fueling schedules for two different flight conditions. Each case was simulated using the variable turbulent Prandtl and Schmidt number capability in CFD++, constant turbulent Schmidt and Prandtl numbers (0.5 and 0.9) previously established in similar cases, 17 species reduced chemistry model, and a 22 species chemistry model.[13]

Figure 13a through d compare solutions for four of the baseline cases used as a starting point in the validation process. Each figure shows a comparison of the experimental data and the four simulations as a function of the streamwise wall static-pressure. It is evident that the results are dependent on the conditions. For example, in Case BN (Figure 13b), the sharp rise at x ~50 does not appear in the calculation. The cause for this is presently unclear and under investigation. Possible reasons include displaced development of coherent structures and neglect of unsteadiness effects. The next step is to analyze solution sensitivities in terms of turbulent Prandtl and Schmidt numbers, turbulence models, chemistry model effects, and numerical procedures.

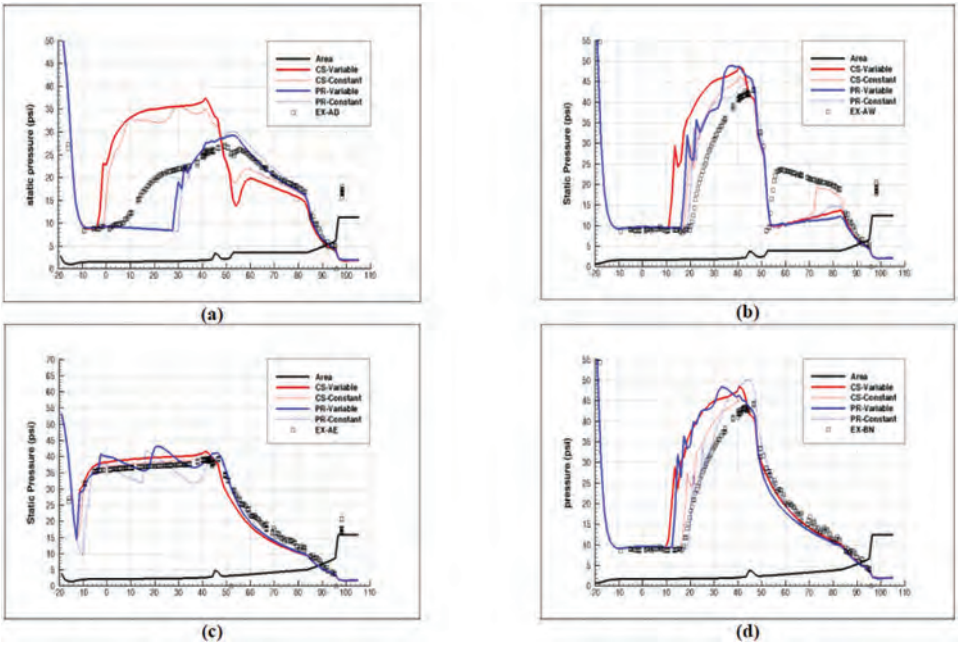


Figure 13. Static-pressure comparisons for (a) Case AE, (b) Case BN, (c) Case AD and (d) Case AW

As noted earlier, the flight test article will be launched with a shroud covering the inlet-combustor configuration. At about Mach 5, the shroud is discarded to expose the vehicle and initiate the scramjet test. The aerodynamic loads during this complex maneuver have the potential to introduce stability issues. High-fidelity simulations offer a unique opportunity to explore the details of the flow, and thus avoid technological surprises that could be catastrophic to the test program. All simulations were conducted using the Reynolds-average Navier-Stokes (RANS) option in the computational code (CFD++). The difference between drag with and without shroud is shown in Figure 14. The abscissa varies the dynamic pressure range (1000lbf, 2000lbf, and 3000lbf) at three different Mach numbers (6, 7, and 8) as shown. It is evident (Figure 14a) that the drag is not influenced significantly by the presence or absence of the shroud, with changes between 2% and 8% for

Mach 8, for all values of dynamic-pressure. Figure 14b exhibits the pressure versus streamwise distance for the Mach 6 case at all values of dynamic-pressure. The results clearly indicate the rise in surface-pressure with dynamic-pressure (as anticipated). The pressure varies only moderately over the length of the vehicle, however.

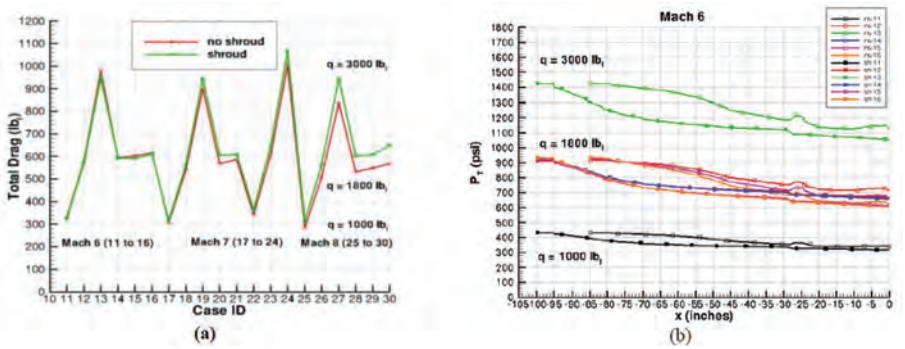


Figure 14. Computed total drag for shroud and Inlet for Mach Numbers equal to 6, 7, and 8 and (b) total pressure downstream of the shroud at different q values

4. Conclusion

Several different concurrent efforts have been pursued during the second year of the Challenge Project. These encompass both the internal flow path, characterized by complex viscous/inviscid interactions, combustion and nozzle effects, as well as the external flowfield, which imposes the drag and moments. The effects of the fuel injection and chemical reaction kinetics, assuming frozen versus finite-rate have been delineated for a test article in a direct-connect ground-test scenario. Reactions not only affect the shear-layer and development downstream, but also create larger separations upstream towards the inlet. Higher fuel flow-rates resulted in larger potential towards instability and likelihood of catastrophic un-start by displaced reactions upstream of the step which generate a more adverse pressure gradient. The capacity of current methods to provide accurate analyses was further examined with the code CFD++. Models which maintain the position of the combustion downstream of the isolator yield results that are in reasonable agreement with theory.

This is a promising development that suggests the importance of high performance computing is likely to play a substantially synergistic role in design evolution. Results on the external flow have characterized the loads anticipated with and without the presence of the protective shroud; these indicate that the effect is minimal. However, the dynamic pressure has a major effect on overall load. In the next year, research will focus on optimization of fuel injection rates by considering multiple location ports simultaneously. Also, the temporal injection sequencing strategy will be parametrically-varied and fully-coupled to the inlet, to account for the distortion.

5. Significance to DoD

The recent success of the X-51 program has demonstrated the feasibility of scramjet-operated flight. To make such flight routine; however, it is essential that the complex physical phenomena that exist in the flow path be understood and controlled. The joint USAF-DSTO HIFiRE program is designed to realize these objectives by acquiring data for discovery of fundamental physics issues. Recent advances in computational techniques, scalable software and large systems with massive numbers of processors have enabled CFD techniques to become a coequal partner with ground- and flight-test. The present project follows this paradigm by considering an ongoing ground- and flight-test program. During the past year, we have examined the effect of increasing fueling in a cavity-based flame holding scramjet flow path. It is shown that improper mixing combined with shock/combustion interactions triggers the potential for catastrophic un-start. A validation study has also been performed to ensure that the results represent reality. Finally, an estimate has been obtained of the drag that will be encountered in a planned test. These results reduce risk while enhancing payoff. By simulating HIFiRE configurations undergoing current testing, this Challenge Project has direct influence on evolution and analyses of current programs.

Acknowledgements

This work was supported by the Air Force Office of Scientific Research under a task monitored by J. Schmisseur and F. Fahroo. Computational resources were provided by the DoD Shared Resource Centers at AFRL and ERDC.

Systems Used

ASC: SGI Altix-4700 (Hawk), Cray-XE6 (Raptor, currently activated), and ERDC: SGI-ICE (Diamond)

Personnel

F.J. Malo-Molina, D.V. Gaitonde, H.B. Ebrahimi, and D. Risha

Computational Technology Area

Computational Fluid Dynamics, 100%.

References

1. Goyne, C.P., McDaniel, J.C., Quagliaroli, T.M., Krauss, R.H., and Day, S.W., "Dual-Mode Combustion of Hydrogen in a Mach 5 Continuous-Flow Facility", *Journal of Propulsion and Power*, Vol. 17, No. 6, pp. 1313–1318, 2001.

2. Ebrahimi, H.B., "Numerical Simulation of Transient Jet-Interaction Phenomenology in a Supersonic Freestream", *AIAA Journal of Spacecraft and Rockets*, Vol. 37, No. 6, pp. 713–719, November and December 2000.

3. Gruber, M., Jackson, K., Jackson, T., and Mathur, T., "Investigations of Shock Trains in Rectangular Ducts", JANNAF 35th Joint APS Meeting, *Paper Number 4D-03*, December 1998.

4. Mathur, T., Lin, K.-C., Kennedy, P.J., Gruber, M.R., Donbar, J.M., Donaldson, W.A., Jackson, T.A., Smith, C.R., and Billig, F.S., "Liquid JP-7 Combustion in a Scramjet Combustor", *36th JANNAF Combustion Subcommittee/Airbreathing Propulsion Subcommittee Meeting*, October 1999.

5. Wilcox, D., "Turbulence modeling - An Overview", *AIAA 2001-0724*, 2001.

6. Ebrahimi, H., "An Overview of Computational Fluid Dynamics for Application to Advanced Propulsion Systems", *AIAA Paper 2004-2370*, 2004.

7. Ebrahimi, H., Gaitonde, D., and Malo-Molina, F., "Parametric Study of 3-D Hydrocarbon Scramjet Engine with Cavity", *AIAA Paper 2007-0645*, January 2007.

8. Ebrahimi, H.B. "Numerical Investigation of Jet Interaction in a Supersonic Freestream", *AIAA Journal of Spacecraft and Rockets*, Vol. 45, No. 1, pp. 642–650, 2008.

9. Ebrahimi, H.B. "Validation Database for Propulsion Computational Fluid Dynamics", *AIAA Journal of Spacecraft and Rockets*, Vol. 34, No. 5, pp. 642–650, September 1997.

10. Malo-Molina, F.J., Gaitonde, D.V., and Ebrahimi, H.B., "High-Fidelity Flowpath Analysis of a Supersonic Combustor Coupled to Innovative Inward-Turning Inlets", *48th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition (AIAA-2010-412)*, Orlando, FL, January 5–8 2010.

11. Malo-Molina, F.J., Gaitonde, D.V., and Ebrahimi, H.B., *44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit (AIAA-2008-4577)*, Hartford, CT, 20–23 July 2008.

12. Malo-Molina, F.J., Gaitonde, D.V., and Ebrahimi, H.B., "High-Fidelity Flowpath Analysis of a Supersonic Combustor Coupled to Innovative Inward-Turning Inlets", *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition (AIAA-2009-131)*, Orlando, FL, January 5–8 2009.

13. Liu, J., Tam, C.J., Lu, T., and Law, C.K., "Simulations of Cavity-Stabilized Flames in Supersonic Flows Using Reduced-Chemical Kinetic Mechanism", *AIAA-2006-4862*.

HPCMP UGC 2011

1. Computational Fluid Dynamics (CFD)

CFD for Ships

Computational Naval Ship Hydrodynamics

Kyle A. Brucker, Thomas O'Shea, Kristine L. Chevalier Beale, and Douglas G. Dommermuth
Naval Hydrodynamics Division, Science Applications International Corporation (SAIC), San Diego, CA
{kyle.a.brucker, thomas.t.o'shea, kristine.lc.beale, douglas.g.dommermuth}@saic.com

Kelli Hendrickson, Gabriel Weymouth, and Dick K.P. Yue
Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA
{khendrk, weymouth, yue}@mit.edu

John Levesque
Cray Supercomputing Center of Excellence, Chief Technology Office, Cray, Inc., Seattle, WA
levesque@cray.com

Kevin D. George, Richard I. Walters, and Michael M. Stephens
US Army Engineer Research and Development Center (ERDC), Unclassified Data Analysis and Assessment Center (UDAAC), Vicksburg, MS
{kevin.d.george, richard.i.walters, michael.m.stephens}@usace.army.mil

Abstract

The primary purpose of our research efforts is to improve naval design and to evaluate various threats. Our current research efforts leverage high performance computing (HPC) resources to perform high-resolution numerical simulations with hundreds of millions to tens of billions of unknowns to study wave breaking behind a transom stern (Drazen, et al., 2010; Weymouth, et al., 2010), wave-impact loading (Fullerton, et al., 2010), the generation of spray by high-speed planing craft (Fu, et al., 2010), air entrainment by plunging breaking waves (Brucker, et al., 2010), internal-wave generation (Rottman, et al., 2010), wave diffraction (Ratcliffe, 2008), and storm seas (Dommermuth, et al., 2010). This paper focuses on the flow around a submarine on the ocean surface, the air entrainment and free-surface turbulence in the flow behind a transom-stern, the internal-wave generation by a submerged body, and the lift and drag acting on a high-speed planing craft. Two codes, Numerical Flow Analysis (NFA) and Boundary Data Immersion Method (BDIM), are used in these studies. Both codes are Cartesian-based Large-Eddy Simulation (LES) formulations, and use either Volume of Fluid (VOF) (NFA) or conservative Volume of Fluid (cVOF) (BDIM) treatments to track the free-surface interface. Comparisons of numerical predictions to laboratory measurements are in good agreement.

1. Introduction

The critical remaining challenges in modern naval ship design and analysis are accounting for the effects of breaking waves, spray, and air entrainment on the performance of surface ships, and to evaluate threats to naval combatants. Moreover, as the Navy considers novel hull concepts and extreme operating environments and conditions, the design process will require extrapolation from current design databases and thus will rely heavily on computational tools. Based on these challenges and requirements, the first objective of this research effort is to validate numerical methods that are capable of simulating complex interface physics and subsurface flows such as: i) spray-sheet and jet formation; ii) air entrainment and bubble generation; iii) strong free-surface turbulence interactions with large-amplitude breaking waves; and iv) nonlinear evolution of internal waves. The second objective of this research effort is to provide design support for assessing the performance of and threats to the next generation of naval vessels undergoing extreme loads and motions while operating at high speeds. The results from this research effort will improve our capability of simulating and understanding how these complex flow phenomena affect the design and analysis of Navy ships.

This paper focuses on the results of the third year of this effort towards meeting these two objectives. In this year, we have focused on the flow around a submarine on the ocean surface, the air entrainment and free-surface turbulence in the flow behind a transom stern, the forces acting on a high-speed planing craft, and the internal-wave generation by a submerged body by performing systematic studies of these problems using our core applications and validating them against available experimental data. Section 2 provides details and references for these core applications. Section 3 highlights our work on a submarine on the ocean surface with a special emphasis on recent performance upgrades as part of a separate Capabilities Applications Project (CAP) effort. Section 4 discusses transom-stern flows with some comparisons between experiments and simulations in the rooster-tail region where there is significant entrainment of air. Section 5 compares drag predictions to measurements due to internal-wave generation for a submerged sphere. Section 6 compares lift and drag predictions to measurements for a high-speed planing craft. Comparisons between numerical simulations and linear theory are also shown. Section 7 is a summary of our primary results and their significance to the Department of Defense (DoD).

2. Numerical Approaches

Two core applications, Numerical Flow Analysis (NFA) and Boundary Data Immersion Method (BDIM), are used in this study. Our numerical simulations are validated by comparisons to field and laboratory experiments. NFA is a Cartesian-based LES code with Volume of Fluid (VOF) interface capturing developed at SAIC, and the BDIM code is a conservative VOF (cVOF) interface capturing method coupled with a immersed boundary method developed at MIT. NFA has recently been further optimized for high performance computing (HPC) platforms. Improvements to communication as well as recent hardware upgrades and new computer availability have enabled simulations with order 25 billion grid cells, significantly improving resolution of small-scale physics (see Brucker, et al., 2011). Details of the NFA and BDIM formulations are available respectively in Dommermuth, et al. (2008), Hendrickson and Yue (2008), and Weymouth and Yue (2010, 2011).

For Level-Set LES (LS-LES) and BDIM, the performance of the multigrid solver is key to scalability of the overall numerical capabilities because most of the computational time is spent in this portion of the algorithm solving the Poisson equation for the pressure. Recent scaling analysis on the Cray XT5® (Cray, Inc.) (NAVY/Einstein) of the BDIM code using the multigrid solver to solve two different problems is shown in Figure 1, which shows number of iterations to converged solution per second (wall-time) as a function of number of cores. For this test, a straight line would represent perfect scaling. For both problems, a scaling analysis was performed using a fixed 64×64×64 block size with increasing numbers of cores. The two problems considered were for the solution of the pressure field of a water drop of fixed radius at different Weber numbers. Depending on the strength of surface tension (smaller Weber number, We), the test problem becomes more challenging as it involves an extremely strong and narrow source term in the system of equations. Test 1 is a severe case of We=1, and Test 2 is a moderate case of We=1,000. For both cases, near perfect scaling is achieved for up to 1,000 cores. Since BDIM is utilized for problems on the order of 10^8 cells (e.g., more than 1,000 cores in this test), the performance of the multigrid solver is satisfactory.

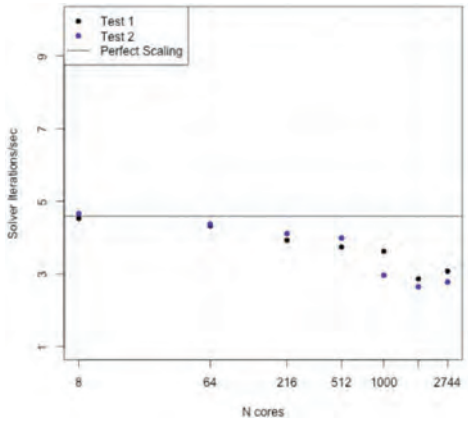
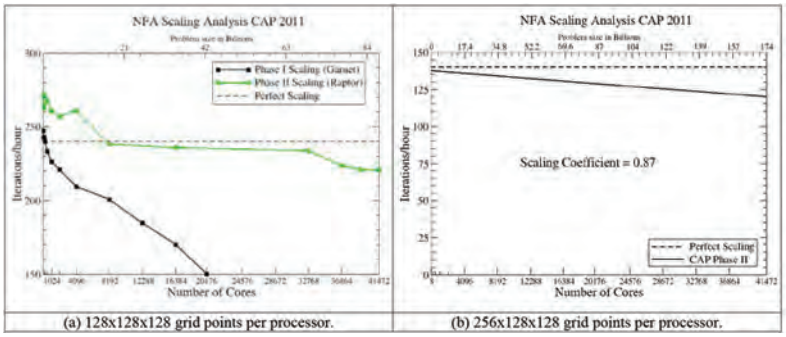


Figure 1. Scaling analysis of multigrid solver in BDIM

Advances made by John Levesque and the Cray Supercomputing Center of Excellence to the data communication routines significantly improved the scaling of NFA. The performance upgrades were made possible by the award of a CAP. Details of the CAP award are available in Brucker, et al. (2011). By replacing the Remote Memory Access Message

Passing Interface (RMA MPI) operations with non-blocking send/receive pairs, the scaling improved significantly at large core counts. This is evidenced by the difference in the black and green lines in Figure 2a. The black line is the scaling on Garnet (Cray XE6®) during Phase I of the CAP, whereas the green line is the scaling during Phase II of the CAP program on Raptor (Cray XE6®). In its current state, NFA is able to run on tens of thousands of processors. Figure 2a shows how many iterations (time-steps) per wall-clock hour can be achieved with different numbers of processors with each processor using a block of 128×128×128 grid points. The case with 1,024 blocks (processors) has 2.15 billion grid points, whereas the case with eight blocks (processors) has only 16.7 million grid points. The difference in the number of grid points is a factor of 128, while the number of iterations per hour only decreases by 10%. This scaling bodes well for NFA to run on even larger numbers of processors. Figure 2b shows the results for 256×128×128 blocks. Using blocks of this size requires 1.9GB of memory per core (near the 2GB per core limit hardware limit). The far right data point at 41,472 cores demonstrates the full ability of Raptor in that it uses nearly all the available memory and all of the available cores concurrently to simulate a complex free-surface flow. This simulation had 174 billion grid cells. The weak scaling coefficient of 0.87 bodes well for NFA to scale out to hundreds of thousands of cores and to break the one trillion grid-cell barrier. Levesque and Wagenbreth (2010) provide details of the techniques that were used to optimize NFA's performance.

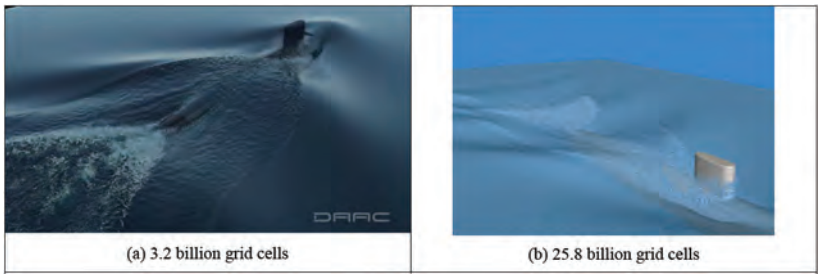


(a) 128x128x128 grid points per processor. (b) 256x128x128 grid points per processor.

Figure 2. NFA Scaling analysis

3. Flow Around a Surfaced Submarine

The CAP also provided the resources to complete two production-level simulations and a scaling study of the flow around a generic submarine moving with constant forward speed on the free surface. Key results are highlighted here, and additional details are available in Brucker, et al. (2011). The numerical simulations illustrate the ability of NFA to predict the generation of spray, the entrainment of air, and the breaking of waves. The simulations were performed at two different grid resolutions corresponding to 3.2 and 25.8 billion grid cells. The 3.2 billion-cell simulation ran for 32,000 time-steps using 12,288 Cray XE6 (Cray, Inc.) central processing units (CPUs) and took 45 wall-clock hours. The 3.2 billion-cell job generated 22Tb of data and 4.2Tb was archived for future study. The 25.8 billion-cell simulation ran for 48,000 time-steps using 24,576 Cray XE6 CPUs and took 225 wall-clock hours. The 25.8 billion-cell job generated 243Tb of data and 23.5Tb was archived for future study. Figure 3 illustrates the effects of free-surface turbulence, air entrainment, spray formation, and breaking waves. The ability to model these physical phenomena is important to the Navy for threat evaluation. We note that the 3.2 billion-cell simulation could be ray traced, but the 25.8 billion-cell simulation could not be ray traced because the number of facets exceeded the capabilities of the rendering software.



(a) 3.2 billion grid cells (b) 25.8 billion grid cells

Figure 3. NFA simulation of a surfaced submarine

4. Air Entrainment and Free-Surface Turbulence of Transom Sterns

NFA predictions of a transom-stern flow are compared to laboratory measurements that have been performed at the US Naval Surface Warfare Center, Carderock Division (see Wyatt, et al., 2008; Drazen, et al., 2010). The three-dimensional numerical simulations use 2,688×1,024×384=1,056,964,608 grid points which are distributed over 1,008 processors on the SGI® Altix® ICE (Silicon Graphics International) at the US Army Engineering Research and Development Center (ERDC) (Diamond) and run for 30,000 time-steps corresponding to 7.5 ship lengths. Each case requires about 90 hours of wall-clock time and 100,000 CPU hours.

The results from two cases are presented here. The first (Case 1) at 7 knots is a partially wetted transom condition. The second (Case 2) at 8 knots is a dry transom condition. It should be noted here that the physics of the flow change between these two conditions and NFA predicts these two conditions accordingly. Figure 4 compares a snapshot of the experiments on the left to NFA predictions of the transom-stern flow on the right. The transom is partially wet for 7 knots in the upper portion of the figure and fully dry for 8 knots in the lower portion.

Figure 5 shows a centerplane cut behind the transom with NFA predictions of the void fraction compared to measurements. The void fraction is the percentage of entrained air per unit volume. The top row of plots shows NFA predictions. The bottom row of plots shows NFA predictions with void-fraction measurements inserted. The plots are from 2.13m aft of the transom to the transom and from 0.6m above the mean waterline to -0.72m below. Predictions are in good agreement with measurements. An unsteady multi-phase shear layer forms in the rooster-tail region. The air is primarily entrained at toe of the spilling region and degasses over the top of the rooster tail. For the 7-knot case, the toe moves forward and wets the transom with foam. For the 8-knot case, the toe is slightly aft of the transom. For both cases, the primary entrainment of air occurs aft of the transom. The shear layer is thicker and longer for the 7-knot case than for the 8-knot case. The difference in thicknesses and amount of air entrainment affects the temporal and spatial structure of the shear layer. The agreement between experimental measurements and NFA predictions is excellent for a very complex physical phenomenon.

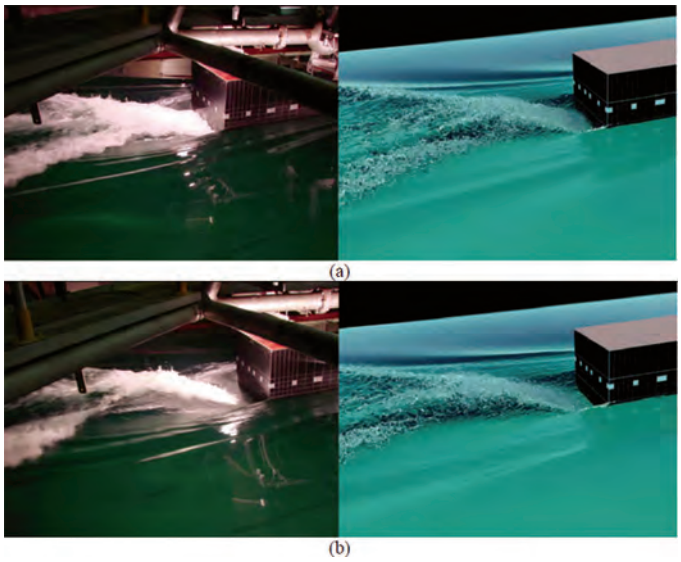


Figure 4. Perspective views of the transom region. Experimental measurements are plotted on the left and NFA predictions are plotted on the right; (a) 7 knots and (b) 8 knots.

BDIM predictions of a canonical transom-stern flow at draft Froude number of 3 were performed on the Cray XT6 at the NAVY DoD Supercomputing Resource Center (NAVY DSRC/Einstein). The three-dimensional numerical simulations use just over 100,000,000 grid points distributed over 768 processors. The simulation ran for 35,000 time-steps corresponding to over 200 draft lengths. Each case required approximately 95 hours of wall-clock time and 73,000 CPU hours. The highest resolution present in the simulation was 0.01562 drafts. The significant length of simulation time provided a large averaging window of data that was quasi-steady in nature. Figure 6 shows the average isosurface of the one-half void fraction (an estimate of the interface) for a subset of the domain. Through averaging, distinctive features can be identified such as the convergent corner waves off of the stern, the "rooster-tail" region behind where the waves converge, and the divergent waves where the waves begin to travel away from the centerline of the vessel.

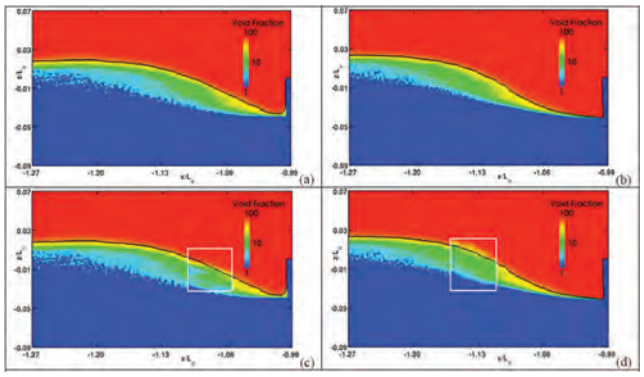


Figure 5. Void fraction predictions compared to experimental measurements on centerplane. (a) NFA, 7 knots; (b) NFA, 8 knots; (c) NFA with void fraction measurements inserted, 7 knots; and (d) NFA with void fraction measurements inserted, 8 knots. Transom is on right edge of plots. Black lines denote the free surface. White framing denotes measurements. The numerical results have been time-averaged over the last 10,000 time-steps of the numerical simulation.

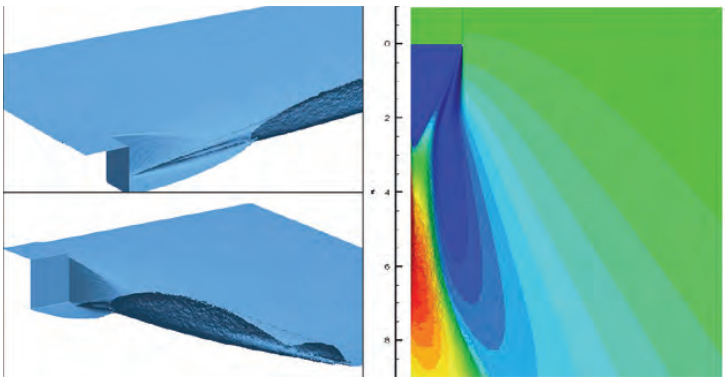


Figure 6. Average interface for a canonical transom stern operating at draft Froude number of 3 from data generated with BDIM. Top left: convergent corner waves off of the corner of the stern. Bottom left: convergent corner waves and divergent waves. Right: birds-eye view of average isosurface. Contours represent vertical distance.

Figure 7 shows the root mean square (RMS) values of the one-half void fraction overlaid onto the average interface. The intensity of the contours is relegated to regions of significant spray and air entrainment as well as potentially white water regions.

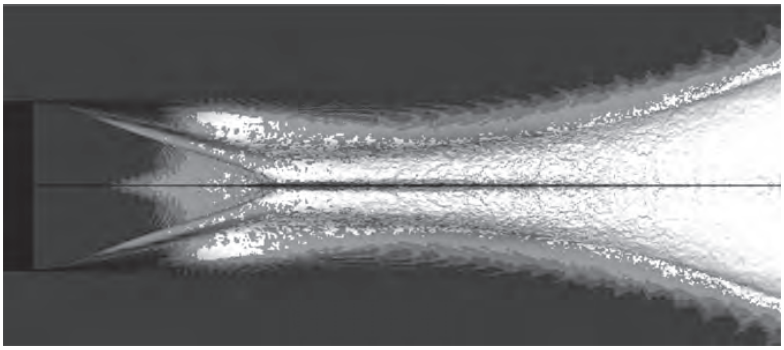


Figure 7. RMS fluctuations of the one-half void fraction of a canonical transom stern operating at draft Froude number 3 from data generated with BDIM

Of particular interest is to quantify the entrainment of air in the wake of the stern. Numerically, it is challenging to identify the regions of void fraction that are entrained. Using a state-of-the-art "blob" detection algorithm, connected regions of air and water can be identified and then determined if they are entrained. Figure 8a shows the isosurfaces of the average void fraction of entrained air behind this canonical stern. The maximum, average void fraction of entrained air for

this case (red) is 0.16. The RMS fluctuations of this entrained void fraction is shown in Figure 8b, where there persists a strong core of high RMS fluctuations (0.12) in the wake from the rooster tail region to the divergent waves.

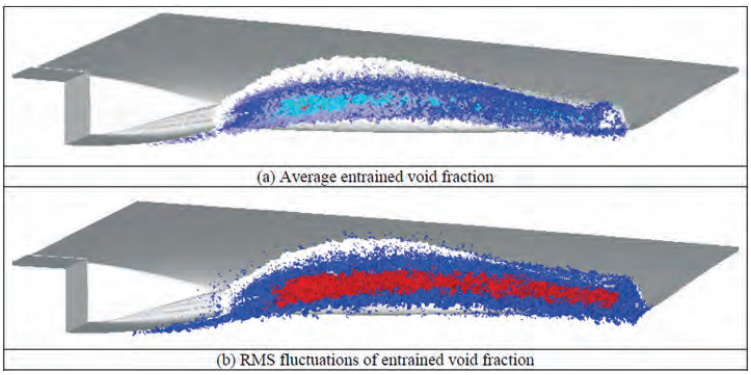


Figure 8. Average entrained void fraction and RMS fluctuations in the wake of a canonical transom stern operating at draft Froude number 3 from data generated with BDIM

5. Generation of Internal Waves

The computer code NFA, Dommermuth, et al. (2008), which was originally designed to provide turnkey capabilities to simulate the free-surface flow around ships, has been extended to simulate stratified sub-surface flows. NFA simulations of flow over a sphere in a stratified fluid are discussed. Rottman, et al. (2010) provides additional details of the simulations.

The drag coefficient of a sphere, is defined as $C_D=\mathcal{D}/(2\rho U^2\mathcal{A})$, where $\mathcal{D}$ is the drag force, ρ is the water density, U is velocity, and $\mathcal{A}$ is the project frontal area of the body. The stratification of the fluid can cause body-generated waves and significant drag at low Froude numbers ($F_r \leq 1.5$), while at moderate Froude numbers ($1.5 \leq F_r \leq 3$) the potential energy gained sweeping over the top and bottom of the sphere aids in pressure recovery in the lee and a small drag reduction. Following, Lofquist and Purtell (1984) the change in drag due to stratification, ΔC_D, is defined as: $\Delta C_D(F_r)=C_D(F_r)-C_D(F_r=\infty)$, where $F_r=\infty$ corresponds to a non-stratified fluid. Figure 9 shows the change in the drag coefficient, ΔC_D, as a function of the internal Froude number, $F_r=U/(DN)$, where D is the diameter and N is the Brunt–Väisälä frequency. NFA simulations at F_r=0.1, 0.2, 0.25, 0.35, 0.5, 0.75, 1.0, 1.5, 2.0 are compared with the theories of Greenslade (2000), Voisin (2007), and Gorodtsov and Teodorovich (1982) along with the experiments of Lofquist and Purtell (1984), Vosper, Castero, Snyder and Mobbs (1999), Shishkina (1996), and Mason (1977). In summary, Figure 9 shows that NFA is able to correctly predict the change in drag for both low and moderate Froude numbers, and predicts the transition from low to high Froude number regimes that occurs between Fr≈0.7 and Fr≈1.0.

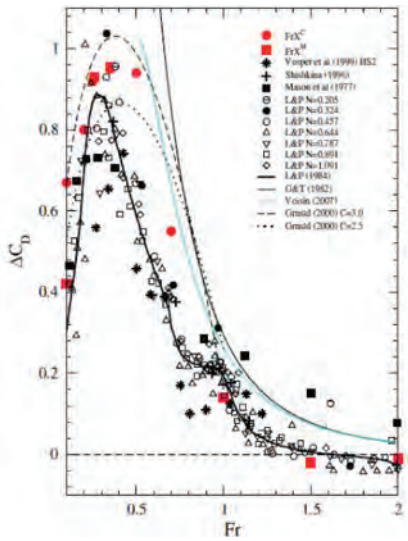


Figure 9. Change in the drag coefficient as a function of internal Froude number. In the legend, FrX^C and FrX^M denote cases

Figure 10 shows contour plots of the longitudinal velocity (u) for Fr=1. The NFA result is shown on the left and the linearized theory on the right for z/D=2 above the sphere. The two results are very close, with some small differences appearing downstream for x/D=−5 near y=0. At this relatively low Froude number, the body-generated waves dominate over the wake-generated waves.

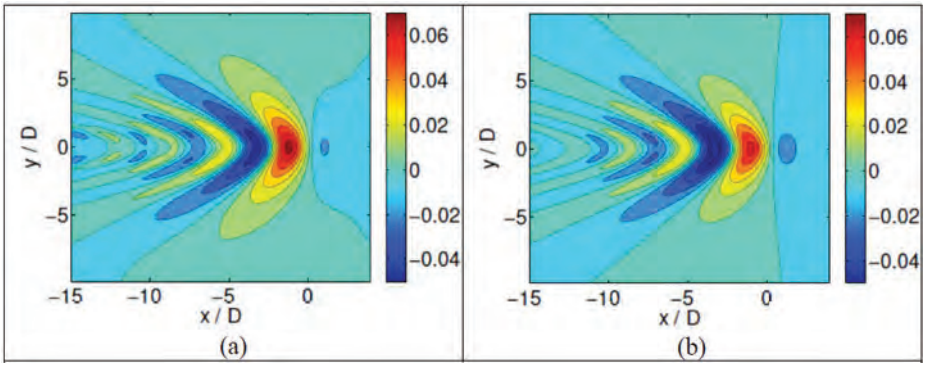


Figure 10. A comparison for Fr=1 from the numerical simulations (a) and from linear theory (b), on the (x, y) plane located at z=2D. The plotted variable u/U, the x-component of the velocity of the wavefield.

6. High-speed Planing Craft

Numerical simulations of the flowfield around a planing craft with 13-degree trim tabs have been performed utilizing the NFA code. Fu, et al. (2008) and Fu, et al. (2010) provide specific details of the simulations. The more recent simulations that are reported in this paper include a new treatment for the convective terms in the Navier-Stokes equations and a new model of the hull boundary layer. The new treatments significantly improve the predictions of forces acting on the planing model as quantified by comparisons to laboratory measurements.

Length scales are normalized by L_o=3.301m, the length between perpendiculars. U_o, the ship speed, normalizes velocity scales. Pressures are normalized by $\rho(U_o)^2$, where ρ is the density of water. Three model-scale speeds are considered, corresponding to 8.91m/s (slow), 11.88m/s (medium), and 14.38m/s (fast). Figure 11 shows NFA predictions of lift and drag to measurements. The numerical simulations accelerate the flow from rest to full speed. The agreement between numerical predictions and experimental measurements is very good. The numerical results at the beginning only include the new treatment for the convective terms, and then toward the end of the simulations, the new treatment for the hull boundary layer is added. The new results improve over those reported in Fu et al. (2010).

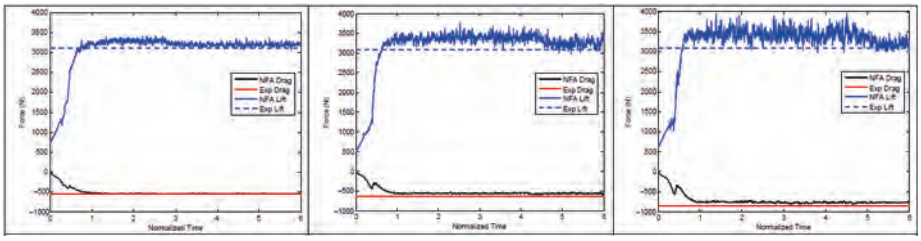


Figure 11. Experimental measurements compared to NFA predictions for lift and drag; (a) 8.91 m/s (b) 11.88 m/s (c) 14.38 m/s

Figure 12 shows NFA predictions of normalized pressures acting on the bottom of the planing hull. The results are shown after the planing boat has traveled six ship lengths. The highest concentrations of pressures are along the spray root near the bow and on the trim tabs. The concentrations in pressure that occur in the spray root are due to chines in the hull. Note that the color legend for the normalized pressure is saturated to emphasize the high-pressure regions.

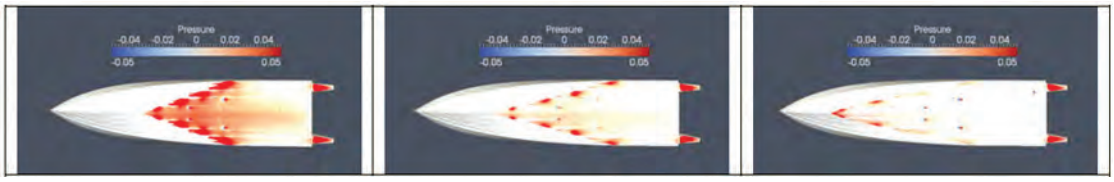


Figure 12. NFA predictions of normalized pressure on bottom; (a) 8.91 m/s (b) 11.88 m/s (c) 14.38 m/s

7. Conclusion

This third year of this research effort has addressed many scientific and technical challenges in computational ship hydrodynamics. Through this carefully validated computational effort, we have studied the flow around a generic submarine on the ocean surface, the air entrainment and free-surface turbulence in the flow behind a transom stern, the forces acting on a high-speed planing craft, and the internal-wave generation by a submerged body. Through a separate but coordinated CAP effort, NFA has successfully simulated over 174 billion grid cells and the scaling tests indicate that NFA will break the one-trillion grid-cell barrier. For the transom stern, using both NFA and BDIM, we identified regions of strong air entrainment and the intensity of the RMS values, which will be critical in modeling mixed-phase turbulence and air entrainment. The improved simulations of high-speed planing craft illustrate our capability to accurately calculate shiploads at extreme speeds. The simulations of internal waves generated by a submerged body illustrate the capability to correctly calculate loads and predict transitional Froude numbers. As this is the final year of this Challenge Project effort, we have put forth a new proposal that will further research the scientific and technical challenges in computational ship hydrodynamics as well as address the prediction of extreme operational environments encountered by naval vessels.

8. Significance to DoD

The complex interactions of naval combatants with the ocean environment and the resulting impact on the performance of and threats to naval combatants are some of the main challenges remaining in modern naval ship design. Recent software developments and hardware advances provided by the HPC program have enabled computational ship hydrodynamics, bringing us closer to meeting this challenge.

Acknowledgements

The Office of Naval Research and the Naval Surface Warfare Center, Carderock Division supports this research. The Program Managers are Patrick Purtell, Ph.D., Steven Russell, Ph.D., Ronald Joslin, Ph.D., and Thomas Fu, Ph.D. This work is supported in part by a grant of computer time from the DoD High Performance Computing Modernization Program (http://www.hpcmo.hpc.mil/). The numerical simulations have been performed on the SGI Altix ICE and the Cray XE6 at the US Army Engineering Research and Development Center (ERDC), on the Cray XE6 at the Air Force Research Laboratory (AFRL), and on the Cray XT6 at the Navy DoD Supercomputer Resource Center (NAVY). Animations of NFA simulations are available at http://www.youtube.com/waveanimations.

References

Adams, P., George, K., Stephens, M., Brucker, K.A., O'Shea, T.T., and Dommermuth, D.G., "A numerical simulation of a plunging breaking wave," *Physics of Fluids*, Vol. 22, 2010.

Brucker, K.A., O'Shea, T.T., Dommermuth, D.G., Levesque, J., George, K.D., Walters, R.I., and Stephens, M.M., "Numerical Flow Analysis," *Proceedings of the DoD High Performance Computing Modernization Program, 21st Users Group Conference*, Portland, OR, USA, 2011.

Brucker, K.A., O'Shea, T.T., and Dommermuth, D. G., "Three-Dimensional Simulations of Deep-Water Breaking Waves," *Proceeding of the 28th Symposium on Naval Hydrodynamics*, Pasadena, CA, USA, 2010.

Dommermuth, D.G., O'Shea, T.T., Brucker, K.A., and Wyatt, D.C., "A numerical formulation for simulating free-surface hydrodynamics," *Proceeding of the 27th Symposium on Naval Hydrodynamics*, Seoul, Korea, 2008.

Dommermuth, D.G., Fu, T.C., Brucker, K.A., O'Shea, T.T., and Wyatt, D.C., "Numerical prediction of a seaway," *Proceeding of the 28th Symposium on Naval Hydrodynamics*, Pasadena, CA, USA, 2010.

Drazen, D.A., Fullerton, A.M., Fu, T.C., Beale, K.L. ., O'Shea, T.T., Brucker, K.A., Wyatt, D.C., Bhushan, S., Carrica, P.M., and Stern, F. "Comparisons of model-scale experimental measurements and computational predictions for the transom wave of a large-scale transom model", *Proceeding of the 28th Symposium on Naval Hydrodynamics*, Pasadena, CA, USA, 2010.

Hendrickson, K. and Yue, D.K.-P., "Models for simulating breaking waves in computational ship hydrodynamics," *Proceeding of the 27th Symposium on Naval Hydrodynamics*, Seoul, Korea, 2008.

Fu, T.C., Dommermuth, D.G., O'Shea, T.T., and Ratcliffe, T., "A comparison of experimental measurements and computational predictions of a deep-v planing hull," *Proceeding of the 28th Symposium on Naval Hydrodynamics*, Pasadena, CA, USA, 2010.

Fu, T.C., Fullerton, A., Brewton, S., Brucker, K., and Dommermuth, D., "An experimental and computational study of breaking-wave impact forces," *Proceeding of the 27th Symposium on Naval Hydrodynamics*, Seoul, Korea, 2008.

Fullerton, A.M., Fu, T.C., Brewton, S., O'Shea, T.T., Brucker, K.A., and Dommermuth, D.G., "A comparison of measured and predicted wave-impact pressures from breaking and non-breaking waves," *Proceeding of the 28th Symposium on Naval Hydrodynamics*, Pasadena, CA, USA, 2010.

Greenslade, M.D., "Drag on a sphere moving horizontally in a stratified fluid," *J. Fluid Mech.*, Vol. 418, pp. 339–350, 2000.

Levesque, J.M. and Wagenbreth, G., *High Performance Computing: Programming and Applications*, Chapman & Hall/CRC Computational Science, 2010.

Lofquist, K.E.B. and Purtell, P., "Drag on a sphere movinghorizontally through a stratified liquid," *J. Fluid Mech.*, Vol. 148, pp. 271–284, 1984.

Mason, P.J., "Forces on spheres moving horizontally in rotating stratified fluid," *Astrophys. Fluid Dyn.*, Vol. 8, pp. 137–154, 1977.

Ratcliffe, T., Minnick, L., O'Shea, T., Fu, T., Russell, L., and Dommermuth, D., "An integrated experimental and computational investigation into the dynamic loads and free-surface wave-field perturbations induced by head-sea regular waves on a 1/8.25 scale-model of the R/V Athena," *Proceeding of the 27th Symposium on Naval Hydrodynamics*, Seoul, Korea, 2008.

Rottman, J.W., Brucker, K.A., Dommermuth, D.G., and Broutman, D., "Parameterization of the internal wave field generated by a submarine and its turbulent wake in a uniformly stratified fluid," *Proceeding of the 28th Symposium on Naval Hydrodynamics*, Pasadena, CA, USA, 2010.

Shishkina, O.D., "Comparisons of the drag coefficients of bodies moving in liquids with various stratification profiles," *Fluid Dynamics*, Vol. 31, pp. 484–489, 1996.

Voisin, B., "Lee waves from a sphere in a stratified flow," *J. Fluid Mech.*, Vol. 574, pp. 273–315, 2007.

Vosper, S.B., Castero, I.P., Snyder, W.H., and Mobbs, S.D., "Experimental study of strongly stratified flows past three-dimensional orography," *J. Fluid Mech.*, Vol. 390, pp. 223–239, 1999.

Weymouth, G.D. and Yue, D.K.-P., "Conservative volume of fluid method for free-interface simulations on Cartesian grids," *Journal of Computational Physics*, Vol. 229 (8), 2010.

Weymouth, G.D. and Yue, D.K.-P., "Boundary data immersion method for cartesian-grid simulations of fluid-body interaction problems," *Journal of Computational Physics*, 2011 (in press).

Weymouth, G.D., Hendrickson, K., Banerjee, S. and Yue, D.K.-P., "Bubble source modeling using macro-scale two-phase flow simulations," *Proceeding of the 28th Symposium on Naval Hydrodynamics*, Pasadena, CA, USA, 2010.

Wyatt, D., Fu, T., Taylor, G., Terrill, E., Xing, T., Bhusan, S., O'Shea, T., and Dommermuth, D., "A comparison of full-scale experimental measurements and computational predictions of the transom-stern wave of the R/V Athena I," *Proceeding of the 27th Symposium on Naval Hydrodynamics*, Seoul, Korea, 2008.

Wyatt, D.C., Bhushan, S., Brucker, K.A., Carrica, P., Chevalier, K., Dommermuth, D.G., Fu, T.C., O'Shea, T.T., and Xing, T., "Comparisons of model-scale experimental measurements and computational predictions for the transom wave of a large-scale transom model," *Proceeding of the 28th Symposium on Naval Hydrodynamics*, Pasadena, CA, USA, 2010.

Hydrodynamic Shape Optimization of Delft Catamaran

Wesley Wilson and Joseph Gorski
US Naval Surface Warfare Center, Carderock Division (NSWCCD), West Bethesda, MD
{wesley.m.wilson, joseph.j.gorski}@navy.mil

Mani Kandasamy, Tomohiro Takai, Wei He, and Fred Stern
The University of Iowa, IIHR – Hydroscience & Engineering, Iowa City, IA
mkandasa@engineering.uiowa.edu, frederick-stern@uiowa.edu

Yusuke Tahara
National Maritime Marine Institute (NMRI) Japan
tahara@marine.osakafu-u.ac.jp

Abstract

This effort is focused on hydrodynamic shape optimization with application to naval vehicles. At present, this is mainly applied to US Navy surface combatant hull forms and related systems; however, this approach could also be applied to other marine applications. The primary goal is to demonstrate the use of high-fidelity, physics-based tools to develop improved designs for naval vehicles and components. The present work focuses on the application of Simulation-Based Design (SBD) for the resistance optimization of the Delft catamaran for a single speed. Ongoing work related to optimization of the Joint High-Speed SeaLift (JHSS) ship concept has been used for verification and validation, and testing of the SBD process. Design optimization in both cases has been performed for both the parent hull form and the waterjet inlets.

1. Introduction

Navy ships of the future may be radically different from those already in the fleet in order to meet emerging missions and related operational requirements. This transformation can already be seen with the DDG-1000 tumblehome hull form. One difficulty with designing such new concepts is the lack of experience from which to draw from when performing design studies.

These issues point to a need for greater fidelity, robustness, and ease-of-use in the tools used in early stage ship design. The Computational Research and Engineering Acquisition Tools and Environments (CREATE) program attempts to address this in its plan to develop and deploy sets of computational engineering design and analysis tools. It is expected that advances in computers will allow for highly-accurate design and analyses studies that can be carried out during the early phases of the design process. In order to evaluate candidate designs and explore the design space more thoroughly, shape optimization is a key component of the CREATE Ships Hydrodynamic Program. This Program will attempt to use fast parameterized codes to bound the design space, and then more accurate Reynolds-averaged Navier-Stokes (RANS) codes to better define the geometry and performance of the specified hull forms. The potential for hydrodynamic shape optimization has previously been demonstrated for a SWATH hull form (Stern, et al., 2007) and for a catamaran in a related effort (Campana, et al., 2006). The tools are basically in place for performing hydrodynamic shape optimization of a hull form, but a significant computational effort is needed to demonstrate this capability for hull forms currently of interest to the Navy, and validate the prediction capability. The major goal of this effort is to utilize the increasing experience gained in using these methods to assess shape optimization techniques and how they might impact design for current and future naval ships.

In the current paper design optimization is carried out for both the hull form and the waterjet inlet, with resistance/powering performance as the single objective function for both JHSS (at Fr=0.34) and Delft catamaran (at Fr=0.5). For the waterjet inlet design, this implies the optimization of the inlet efficiency. Cavitation and non-uniformity of velocity distribution at the inlet plane are not considered in the current study. Since viscous effects involving boundary-layer

ingestion play a significant part in the waterjet inlet performance, variable fidelity approaches which integrate potential flow methods with RANS through metamodels cannot be used reliably for waterjet inlet design optimization. However, the variable fidelity approach can be used for optimization of the bare hull with the incorporation of waterjet-induced forces and moments through a variable physics approach by using the integral force/moment CFD waterjet model (Kandasamy, et al., 2010). Tahara, et al. (2011) use this variable fidelity/physics approach in a concurrent and complementary paper for the multi-objective design optimization of resistance and sea-keeping of the bare hull JHSS and Delft catamaran without inclusion of the waterjet duct flow. The current study employs just the variable physics method for the Delft catamaran, where the initial optimization of the bare hull is conducted with the waterjet model and the selected optimized bare hull is appended with the waterjet and the waterjet inlet is optimized.

As a pre-requisite to the optimization, the method used here for the waterjet simulation has been validated with experimental data for both JHSS (Takai, et al., 2011) and the Delft catamaran (Kandasamy, et al., 2011). The approach uses RANS with a single-phase level set method for the free-surface prediction, with an actuator disk model to replicate the effects of the impeller, and two degrees-of-motion were allowed for dynamic sinkage and trim predictions.

The Simulation-Based Design (SBD) toolkit used here is a product of the long-term collaboration between IIHR, INSEAN and NMRI research groups. The SBD toolbox provides high-fidelity/low-fidelity flow solvers, multiple optimization algorithms, and different geometry/grid modification methods (Campana, et al., 2009). Previous versions of the toolbox have been successfully used in the optimization of high-speed mono-hull (Campana, et al., 2006; Tahara, et al., 2008a) and multi-hull (Tahara, et al., 2008b; Peri, et al., 2010) displacement ships and foil-assisted semi-planing catamaran ferries (Kandasamy, et al., 2009). Currently the SBD toolbox is being used to optimize the Delft catamaran, and the optimized design will be validated with model testing.

Additional efforts explore combining multiple-fidelity analysis tools for early stage design (Wilson, et al., 2010). In this approach, a matrix of perturbed hull forms is generated and assessed using different analysis tools. The decoupled nature of the optimization, geometry modification, and solution methods is very attractive as it allows for any tool to be used in the analysis. This capability is planned to be implemented in the CREATE Integrated Hydrodynamics Design Environment (IHDE) as part of the CREATE-Ships program. The results of these studies, along with other sensitivity studies are used to populate the design space used in the optimization.

2. SBD Methodology

The SBD methodology comprises three main parts: the optimizer, the geometry modification methods and the flow solver. The flow solver sends the evaluated objective function for a certain set of design variables to an optimizer, which searches for the minimum value under the general mathematical framework of a Non-Linear Programming problem, and continually updates the design variables. Geometry modeling methods provide the link between the two by deforming the hull shape based on the updated design variables.

2.1 Optimizer

The particle swarm optimizer (PSO) was chosen for the current design optimization because recent studies (Takai, 2010) using analytical function-based search space indicated that it converges faster to the global minimum for single-objective optimization problems compared to evolutionary global optimization methods. The PSO strategy simulates the social behavior of a set of particles which share information among them while exploring the design variables' space. In the basic PSO method, each particle has its own memory (parameter 1) to remember the best places that it has visited, whereas the swarm has a global memory (parameter 2) to remember the best place ever visited. Parameters 1 and 2 are called the cognitive and social parameters, respectively. The cognitive parameter indicates how much confidence the particle has in itself, while the social parameter indicates how much confidence it has in the swarm. A random parameter (parameter 3) is applied to the cognitive and social parameters. Each particle has an adaptable velocity based on these parameters to move itself across the design space. According to these principles, each particle investigates the search space analyzing its own travel experience and that of the other members of the swarm. Inertia weights (parameter 4) control the impact of the previous velocities on the current velocity, and hence regulates the trade-off between global (wide-ranging) and local (nearby) exploration of the swarm. Fine-tuning of these parameters is crucial for the optimization process, and the final solution and the calculation time are strictly linked to the parameter's setting (Campana, et al., 2009).

2.2 Geometry Modification Methods

There exist a large number of approaches for geometry modification: simple morphing techniques, B-spline control-point modifications, CAD systems, and Free-Form Deformation (FFD) techniques. The morphing method was chosen for the current optimization due to the ease of transition from the sensitivity studies to optimization. The best geometries obtained from the sensitivity studies are simply transitioned to the optimizer. The design space is thus directly defined by the initial geometries, which will be blended with weighting coefficients during optimization. The number of design variables required is one less than the number of geometries used for the morphing, e.g., two design variables are used for three-hull-form blending as shown in Equation 1.

$$P_{new} = w_1 P_1 + w_2 P_2 + w_3 P_3 \tag{1}$$

where

$$w_1 = x_1$$
$$w_2 = (1 - x_1) \cdot x_2$$
$$w_3 = (1 - x_1) \cdot (1 - x_2)$$

so that $w_1+w_2+w_3=1$. P_{new} denotes the grid points of the morphed grid, P_1, P_2, and P_3 are the grid points for the three initial designs, and w_1, w_2, and w_3 are the corresponding weighting coefficients. x_1 and x_2 are the design variables that determine the weighting factors.

The initial geometries for the sensitivity studies can be generated using any technique: FFD, B-spline modification, or direct modification of the grid using any grid-generation software. The only stipulation is that all the initial grids need to have the same topology and grid size. Another benefit of the morphing method is that the geometric constraints are automatically satisfied during optimization, as long as the initial designs satisfy the geometric constraints.

2.3 Flow Solver

The RANS solver, CFDShip-Iowa (Carrica, et al., 2010) is used with the blended k-ε/k-ω turbulence model and a single-phase level set method to predict the free surface. A second-order upwind scheme is used to discretize the convective terms of momentum equations for RANS. A pressure-implicit split-operator (PISO) algorithm is used to enforce mass conservation on the collocated grids. The pressure Poisson equation is solved using the PETSc toolkit. All the other systems are solved using an alternating direction implicit (ADI) method. For a high-performance parallel computing, a MPI-based domain decomposition approach is used, where each decomposed block is mapped to one processor. The code SUGGAR runs as a separate process from the flow solver to compute interpolation coefficients for the overset grids, and communicates with a motion controller (six degrees-of-freedom [6DOF]) within CFDShip-Iowa at every time-step. The software USURP is used to compute area and forces on the surface overlapped regions. In addition, a simplified body force model is used for waterjet-propelled simulation to prescribe axi-symmetric body force with axial and tangential components. The propeller model requires thrust, torque, and advance coefficients as input, and provides the torque and thrust forces. These forces appear as a body force term in the momentum equations for the fluid inside the propeller disk. The location of the propeller is defined in the static condition of the ship, and moves according to the ship motions.

3. Verification and Validation

Detailed verification and validation (V&V) analysis was carried out as a pre-requisite for a RANS-based global optimization of intake duct shape (Kandasamy, et al., 2009). A joint high-speed sea-lift design (JHSS), which is a very large (970ft) high-speed ship concept operating at a transit speed of at least 36 knots using four axial flow waterjets, is selected for the initial geometry on current study and subsequent optimization. V&V studies are performed on both bare hull (BH) and waterjet (WJ)-appended design with corresponding experimental fluid dynamics (EFD) data from the 1/34 scale model testing (Jessup, et al., 2008). Figure 1 shows the grid topologies with domain and boundary conditions for BH and WJ designs, respectively. The grid sizes are tabulated in Table 1. The predicted curves of resistance and dynamic motions (sinkage and trim) over a speed range of 18–42 knots agree well with EFD data (see Figure 2).

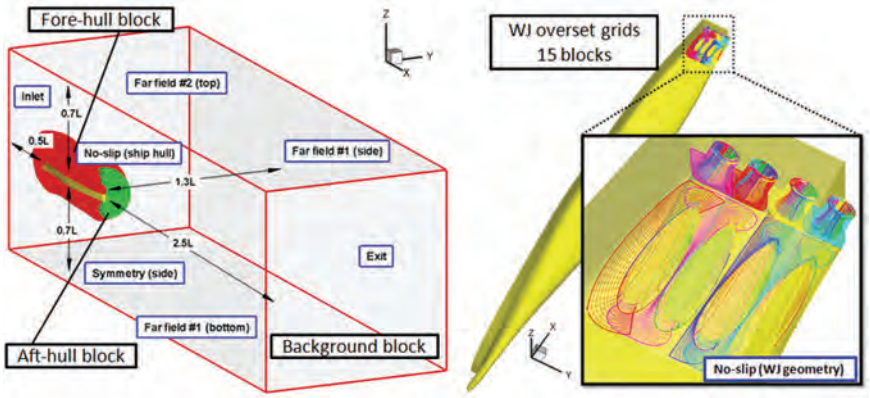


Figure 1. JHSS WJ Model: (a) domain and boundary conditions shown with BH grid, (b) overset grid design

Table 1. Grids information used for verification study of JHSS BH and WJ simulations

#	Description	Total grid points	Y+	#	Description	Total grid points	Y+
1	Finer BH	28,657,743	0.75	1W	Fine WJ	13,077,181	1.13
2	Fine BH	10,150,475	1.13	2W	Medium WJ	6,550,622	1.65
3	Medium BH	3,623,916	1.65	3W	Coarse WJ	4,214,081	2.53
4	Coarse BH	1,278,135	2.53				

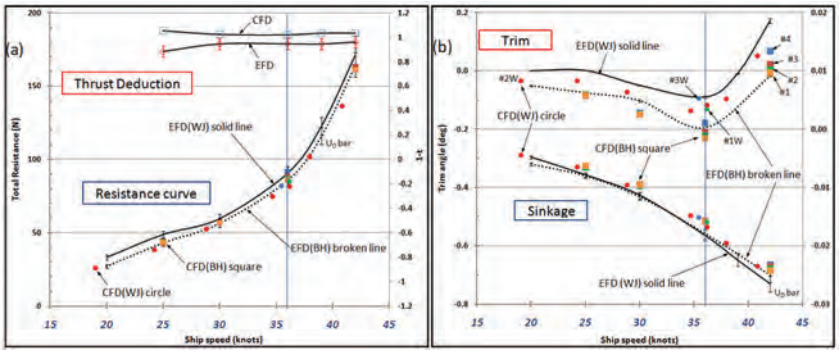


Figure 2. Comparison of force and moment coefficients with EFD data for JHSS bare hull and waterjet-powered: (a) total resistance and thrust deduction, (b) dynamic sinkage and trim

The waterjet-appended JHSS has four waterjet ducts side-by-side, and a half-domain grid utilized 15 overset blocks for the two waterjet ducts. Evaluation of the results during the validation phase showed that the extensive use of overset grids inside the duct leads to a defect in mass conservation due to interpolation errors (Takai, et al., 2011). An accurate flow-rate measurement is important since the estimation of power is dependent upon velocity cubed, and thrust by the velocity squared. Based on this observation, the Delft catamaran waterjet duct was discretized using a single structured grid, which overlaps with the hull grid at the inlet and the nozzle exit (Figure 3). A symmetry boundary condition was used, and body-fitted "O" type grids are generated around the port-side ship hull geometry. A rectangular background grid is used with clustered grid near the free-surface to resolve the wave-field. A cylindrical refinement block was generated immediately following the nozzle exit to better resolve the exiting jet.

Details of the solution domain, grid, and boundary conditions for the JHSS and Delft catamaran can be found in Takai, et al. (2011) and Kandasamy, et al. (2011), respectively. Figure 4(a) and 4(b) show the volume flow solutions for the original waterjet-appended JHSS and the Delft catamaran. Note that for the Delft catamaran, the waterjet inflow is greatly dependent on the hull contour, whereas the JHSS has a flat bottom and a straightforward inlet capture area. Hence, for the delft catamaran, it is imperative that the design optimization of the hull form and the waterjet inlet be coupled.

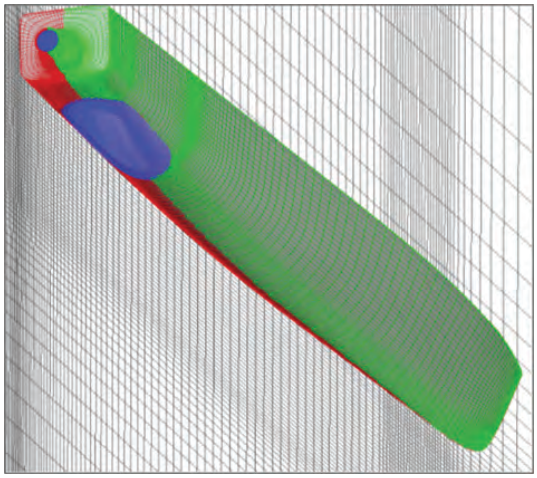


Figure 3. Overset grid assembly for Delft catamaran demi-hull with waterjet

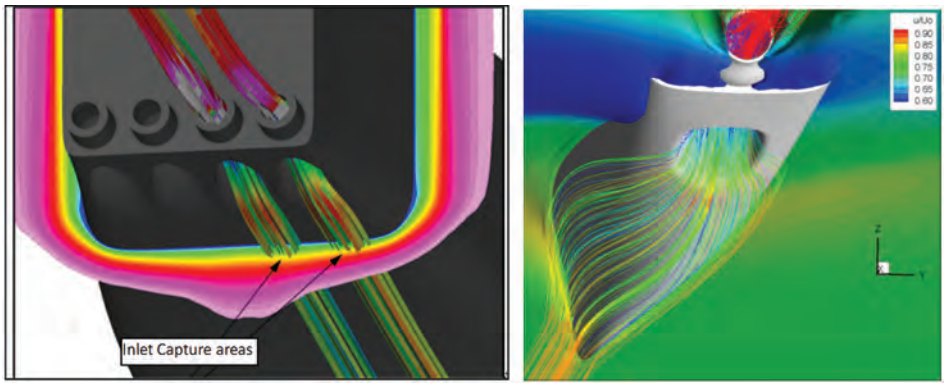


Figure 4. RANS solutions: (a) JHSS with WJ at Fr=0.34, (b) Delft catamaran with WJ at Fr=0.53

4. Sensitivity Studies to Define Feasible Design Space

Initial sensitivity studies were conducted on the hull and waterjet inlet to determine a feasible design space. The results of the sensitivity studies are the initial hull designs to be used in the hull blending approach in the optimizer.

4.1 JHSS BH Bow Shape Sensitivity

Prior to the waterjet inlet optimization studies, a simpler optimization study was carried out for the JHSS BH bow shape. Initial geometries were investigated by the direct movement of hull surface points using B-Spline representations. Figure 5 shows the blending function (design variables) and initial designs with hull surface pressure. The best design from reduction of the bow width (P2) gave a 6.7% reduction, and the best design from variation of the bow depth (P3) gave a 3.5% reduction in resistance. The original geometry (P1), P2 and P3 constitute the initial designs defining the search space with two design variables.

4.2 JHSS WJ Inlet Sensitivity

Two types of geometry modification were performed for the JHSS waterjet inlet, one on the upper duct Gaussian curvature and one on the cut water region, both using B-splines (Figure 6). A sensitivity analysis was carried out for WJ intake duct shape for the JHSS hull configuration at Fr=0.34. The control points for the deformation are put on the upper curvature and lip shape of the inlet duct (see Figure 6). Three control points are assigned along the boundary of the geometrical constraint region, and a B-Spline function is used to interpolate the points between the control points. The self-propelled simulations were performed with the same thrust and torque, but the self-propulsion point (SPP) is different for each case, since the resistance is changed as a result of the shape deformation.

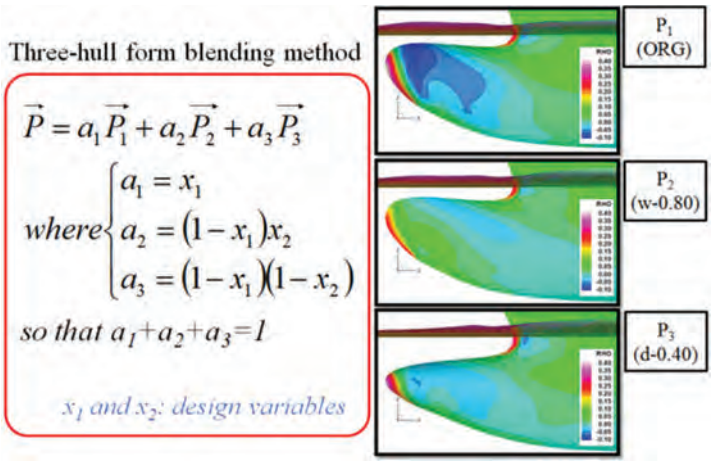


Figure 5. Blending functions (design variables) and initial designs used in optimization of bare hull JHSS bow shape

About 35 different shapes were investigated and the results are shown in Figure 7, where speed indicated the SPP. An improved design is demonstrated by a geometry that results in faster speed with the same thrust, where the frictional resistance component increases while the pressure component decreases. The reduction of pressure resistance is greater than the increase in frictional resistance; thus, the total resistance is reduced and leads to a decreased SPP. The predicted optimum shape shows 0.23% increase in speed and 1.3% reduction in total drag. The better hull shapes were found when using both types of modification; thus the two geometries will be selected for global optimization with the blending method.

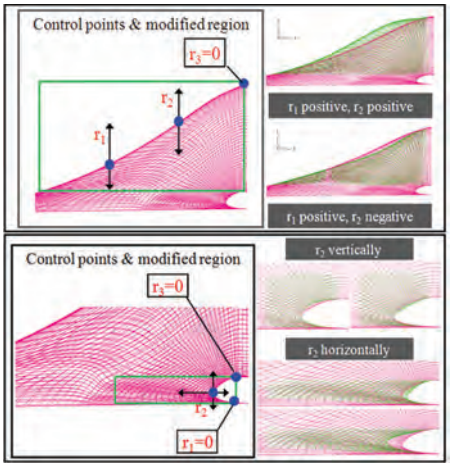


Figure 6. Two types of modification for WJ inlet duct shape, Left: Type 1 (duct curvature), Right: Type 2 (lip shape)

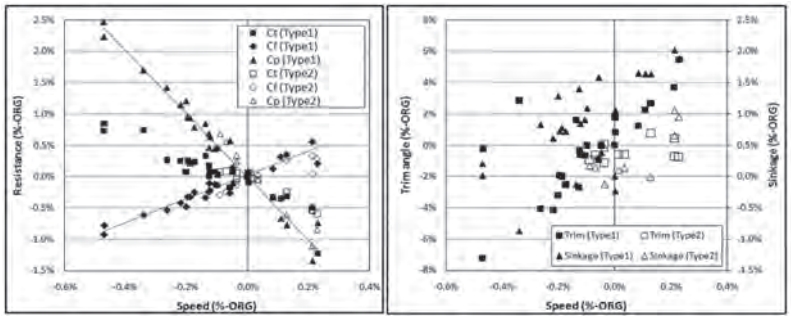


Figure 7. Results of sensitivity analysis for WJ inlet modeling, Left: Resistance vs. Speed, Right: Motions vs. Speed

A completely different approach of combining the waterjet inlets in order to decrease the overall surface area of the inlets was also investigated (Figure 8). Not only is the surface area reduced, the inlet capture areas are combined allowing

for more boundary-layer ingestion. Results showed ~7% decrease in total resistance; however, there are other issues to be considered, such as structural integrity and effect of cross flow during maneuvering, before a combined inlet design can be deemed feasible.

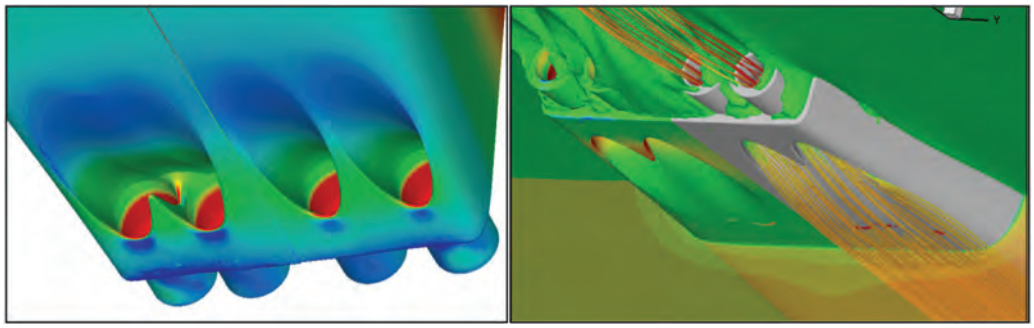


Figure 8. Merged inlet design for JHSS WJ (Left: geometry; Right: predicted streamlines through the duct and nozzle with iso-surface of free-surface position).

4.3 Delft Catamaran Hull Shape Sensitivity

For the Delft catamaran, six different hull shapes, including the original, were selected for morphing (Figure 9). DC1, DC2, and DC3 were created using B-spline modification of the original hull shape. DC1 was created by asymmetric deformation of the bow. DC2 and DC3 were created by increasing the slenderness ratio of the demi-hulls within limits imposed by the diameter of the waterjet impeller. DC4 and DC5 were created by asymmetric free-form deformation of the hulls. The modified geometries showed ~1–2% reduction in resistance compared to the original.

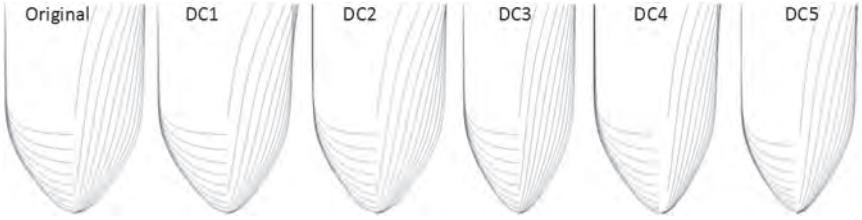


Figure 9. Hull lines of initial bare hull Delft catamaran demi-hull selections used for blending in optimization.

4.4 Delft Catamaran Waterjet Inlet Sensitivity

Similar to the JHSS inlet sensitivity studies, two types of geometry modification were performed for the DC waterjet inlet, one on the upper duct Gaussian curvature, and one on the cut water region, both using B-splines (Figure 10). The modification of the upper curvature provided a geometry with ~2% reduction in total resistance and the cut water modification provided a geometry with ~1% reduction in total resistance.

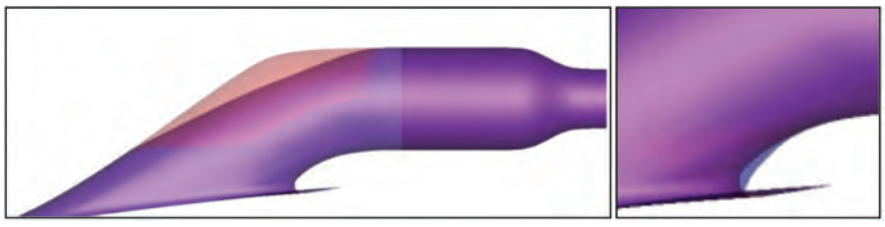


Figure 10. Selected initial Delft catamaran WJ duct geometries for morphing

5. Optimization Studies

The initial sensitivity studies were performed in order to define the initial feasible design space for the optimizations. In this section, the results of the optimization process are presented. As the Delft catamaran optimization is the primary focus of this paper, the JHSS waterjet inlet optimization is not presented here. Further details can be found in Kandasamy, et al. (2011).

5.1 JHSS Hull Optimization

The optimization problem is defined in Table 2. The optimization was carried out for a single objective function, i.e., minimize resistance at Fr=0.34, with constraints on the length, beam and displacement. During sensitivity studies, the solver experienced divergence for extreme deformations of the bow, and hence the design variables were constrained within limits 0<x1, x2<1. However, even with the constraints on the design variables, the PSO initially allows the particles to go out of bounds, and then the particles implode to explore the entire bounded region.

Testing of the PSO on analytical functions with two design variables showed that 4 particles were sufficient to find the global minimum. Based on this, two separate optimization simulations were conducted for the JHSS bow form, first with 4, and next with 6 particles. The topological map of the objective function in the explored region of the search space, obtained by interpolating the particle objective function values, is shown in Figure 11(a).

Table 2. Problem definition and constraints for JHSS bow shape optimization

Objective Function	Geometric Constraints	Design Var. Constraints
Minimize: F1 = RT	Waterline length LWL,	0 < x1,x2 < 1
Target Speed: Fr=0.34	Maximum beam B(max) &	
Sea condition: calm	Draft T = constant	
	Displacement D = constant	

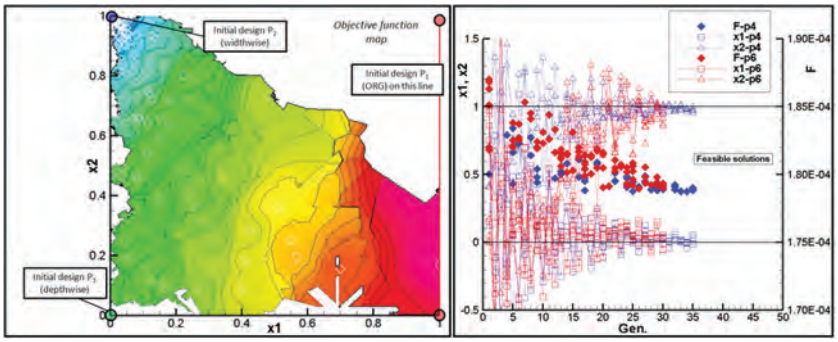


Figure 11. JHSS BH bow optimization: (a) map of objective function, (b) solution convergence history of design variables for initial population of 4 and 6 swarm particles

The particles converge at (x1, x2)=(0, 1), indicating the best design within the search space is the initial design P2. Both PSO approaches, i.e., with 4 and 6 particles, converge at the same location after 30 iterations, as seen in Figure 11(b). The total number of simulations for the 4-particle and 6-particle cases are 120 and 180, respectively. Tahara et al. (2011) conducted the concurrent and complimentary multi-objective genetic algorithm optimization for resistance and sea-keeping using the same initial geometries, with the same morphing approach and the same solver, thus exploring the same design space. They found the minimal value at (x1, x2)=(0.35, 1) after evaluating 400 points in the map. This lies in the unexplored region of the PSO topology map, indicating that the particles got stuck at a local minima during PSO optimization.

5.2 Delft Catamaran Hull Shape Optimization

The DC bare hull optimization utilized five design variables. The topology map is not shown, since it is difficult to visualize a 5-dimensional topology map. An eighteen-particle swarm approach was used. Figure 12(a) shows the convergence of the swarm for the 5 design variables, and Figure 12(b) shows the convergence of the swarm to the minimal objective function. The final optimized hull form shows a 3% decrease in resistance compared to the original hull form. Tahara, et al. (2011) used just three of the six geometries used here for morphing, for their multi-objective GA optimization and obtained a reduction of ~4%, indicating that the PSO for the DC may also have converged at a local minima.

A separate design optimization was carried out by replacing one of the initial candidate hull designs with one generated using an alternative approach. This was done by utilizing the SHAPE optimization framework, developed by SAIC (Kuhn, et al., 2007) and using Das Boot (Wyatt, 2000) to evaluate the objective function. The SHAPE code determines changes to a baseline hull shape that produce improvements to some user-defined metric, and bound by a set of local and generic constraints that are also prescribed by the user. The optimized hull shape is determined by examining how perturbations

to the baseline hull shape change the evaluation of the objective function. The optimization routine is completely separate from the objective function evaluations. In this way, it is possible to utilize computationally-intensive tools to perform the evaluations, and build up a database that reflects the derivatives of the objective function for each of the perturbed hull shapes. The optimization routine itself, which employs linear programming, can then be done very quickly using the pre-generated database of derivatives. This also has the advantage of allowing the user to perform a variety of different design studies in a very short time; for example, changing the design constraints and assessing a new optimum design based on those constraints. In this instance, the design determined by the SHAPE code, when confirmed using the Das Boot solver, showed a reduction in the total resistance of approximately 3.5%. When the same design was confirmed using the CFDShip-Iowa solver, the reduction in total resistance was shown to be approximately 3%.

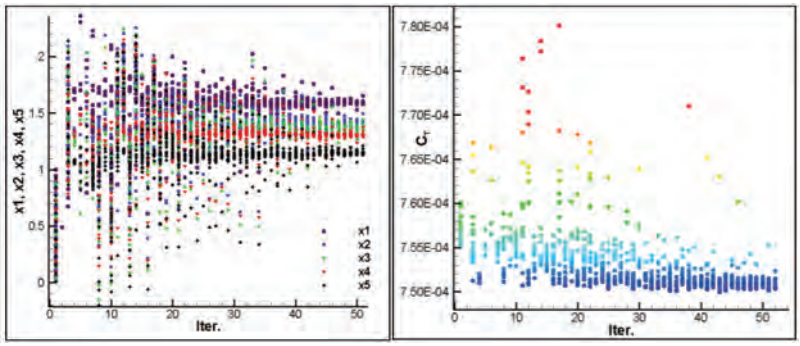


Figure 12. Convergence of (a) design variables and (b) objective function for DC bare hull optimization

Applying this new initial hull design to the optimization process resulted in a somewhat improved optimum design, but with some issues that need to be investigated. Examination of these results is ongoing to determine how the convergence of the PSO influences the optimum solution. But this has shown success in applying a mutli-fidelity approach by utilizing a faster potential flow method to influence the initial design space used in the optimizer.

6. Summary

This project is aimed at assessing the use of different hydrodynamic tools in hull shape optimization and in a larger ship design process. Simulation-based design optimization is performed on two waterjet-appended ships, the JHSS and the Delft catamaran. There is a strong coupling between the Delft catamaran hull shape and the inlet capture area for the waterjet. A variable physics SBD method has been employed for the Delft catamaran, in which the initial optimization of the bare hull is conducted with the integral force/moment CFD waterjet model, and then the selected optimized bare hull is appended with the waterjet and the waterjet inlet is optimized. The bow shape optimization and waterjet inlet optimization of the JHSS were done independent of each other, since the bow shape modification does not affect the local flow near the waterjet inlet. Also, the CFD waterjet model was not used for JHSS since the waterjet-induced forces and moments had negligible effect on the sinkage and trim of the hull.

The particle swarm optimizer is used for the optimization. The RANS code CFDShip-Iowa is used for the flow solver. A morphing-type geometry modification scheme is used where the initial geometries define the design search space. The initial geometries are obtained through geometry modification sensitivity studies to find a feasible search space.

The overall optimization methodology gave 6.7% resistance reduction for JHSS bow shape and 3% resistance reduction for the Delft catamaran hull optimization. An alternative approach to the hull form sensitivity studies for the Delft catamaran hull optimization has also been examined by using the SHAPE optimization framework with the Das Boot potential flow solver. This has shown a significant influence on the feasible design space and leads to optimum designs that show a 6% reduction in the objective function. Issues concerning the convergence of the particle swarm optimizer are ongoing. The effect of combining the adjacent inlets of the JHSS waterjets was also studied, and it showed a 7% decrease in overall resistance, warranting further studies. Work for the Delft catamaran waterjet inlet optimization is also in progress.

Comparison with concurrent and complementary multi-objective optimization results of the bare hull JHSS and Delft catamaran (Tahara, et al., 2011) revealed that at current settings, the PSO tends to get stuck in local minima, whereas GA does not. This issue needs to be addressed by re-running the PSO JHSS bow form optimization with a larger number of swarm particles, and fine-tuning the cognitive, social, random, and inertia parameters to find the best parameters.

With the ongoing development of technology and software related to hull form analysis and optimization, it is intended that these tools may provide more efficient solutions and improved ship designs in the future.

Acknowledgments

The authors are grateful to the US Department of Defense High Performance Computing Modernization Program (HPCMP) that provided the computer resources in support of this Challenge Project. Portions of this work are sponsored by the US Office of Naval Research through research grants N00014-08-0491, under the administration of Dr. Ki-Han Kim.

References

Campana, E.F., Peri, D., Tahara, Y., Kandasamy, M., Stern, F., Cary, C., Hoffman, R., Gorski, J., and Kennell, C., "Simulation-Based Design of Fast Multi-hull Ships", *Proc. 26th Symposium on Naval Hydrodynamics*, Rome, Italy, Sept. 2006.

Campana, E.F., Peri, D., Tahara, Y., and Stern, F., "Shape Optimization in Ship Hydrodynamics using Computational Fluid Dynamics", *Computer Methods in Applied Mechanics and Engineering*, 196, pp. 634–651, 2006.

Campana, E.F., Peri, D., Tahara, Y., Kandasamy, M., and Stern, F. "Numerical Optimization Methods for ship Hydrodynamic Design", T*rans. SNAME Annual Meeting*, Providence, RI, USA, 2009.

Carrica, P., Huang, J., Noack, R., Kaushik, D., Smith, B., and Stern, F., "Large-Scale DES Computations of the Forward-Speed Diffraction and Pitch-and-heave Problems for a Surface Combatant", *Computers and Fluids*, 39 (7), pp. 1095–1111, 2010.

Jessup, S., Donnelly, M., Fry, D., Cusanelli, D., and Wilson, M., "Performance Analysis of a Four Waterjet Propulsion System for a Large Sealift Ship", *Proc. 27th symposium on Naval hydrodynamics*, Seoul, Korea, 2008.

Kandasamy, M., Ooi, S.K., Carrica, P., and Stern, F., "Integral force/moment water-jet model for CFD simulations", *Journal of Fluids Engineering*, Vol. 132, pp. 101103–10, 2010.

Kandasamy, M., Georgiev, S., Milanov, E., and Stern, F., "Numerical and experimental evaluation of waterjet-propelled Delft catamarans", *11th International Conference on Fast Sea Transportation (FAST)*, Honolulu, HI, 2011 (submitted).

Kandasamy, M., Ooi, S.K., Carrica, P., Stern, F., Campana, E., Peri, D., Osborne, P., Cote, J., Macdonald, N., and de Waal, N., "URANS-based optimization of a high-speed foil-assisted semi-planing catamaran for low-wake", *Proc. 10th International Conference Fast Sea Transportation*, Athens, Greece, 2009.

Kuhn, J., Chevalier, K., Schlageter, E., Scragg, C., and Wyatt, D., "The Use of Linear Programming and Basis Functions for Hull-Form Optimization", *9th International Conference on Numerical Ship Hydrodynamics*, Ann Arbor, MI, August 5–8, 2007.

Peri, D., Campana, E.F., Tahara, Y., Kandasamy, M. and Stern, F., "New developments in Simulation-Based Design with application to High-Speed Waterjet Ship Design", *Proc. 28th Symposium on Naval Hydrodynamics*, Pasadena, CA, USA, 2010.

Stern, F., Carrica, P.M., Kandasamy, M., Gorski, J., O'Dea, J., Hughes, M., Miller, R., Hendrix, D., Kring, D., Milewski, W., Hoffman, R., and Cary, C., "Computational hydrodynamic tools for high-speed transports", *Transactions SNAME*, Vol 114, pp. 55–81, 2007.

Tahara, Y., Hino, T., Kandasamy, M., He, W., and Stern, F., "CFD-based multi-objective optimization of waterjet-propelled high- speed ships", *11th International Conference on Fast Sea Transportation (FAST)*, Honolulu, Hawaii, 2011 (submitted).

Tahara, Y, Peri, D, Campana, E.F., and Stern, F., "Computational fluid dynamics-Based multi-objective optimization of a surface combatant", *J. Marine Science and Technology*, 13(2), pp. 95–116, 2008a.

Tahara, Y., Peri, D., Campana, E.F., and Stern, F., "Single and Multi-objective Design Optimization of a Fast Multihull Ship: numerical and experimental results", *Proc. 27th Symposium on Naval Hydrodynamics*, Seoul, Korea, 2008b.

Takai, T., "Simulation-based design for high-speed sea-lift with waterjets by high-fidelity URANS approach", *Master's Thesis,* The University of Iowa, 2010.

Takai, T., Kandasamy, M., and Stern, F., "Verification and validation study of URANS simulations for axial waterjet-propelled large high-speed ship", *Journal of Marine Science and Technology*, 2011 (submitted).

Wilson, W., Gorski, J., Kandasamy, M., Takai, T., Stern, F., and Tahara, Y., "Hydrodynamic Shape Optimization for Naval Vehicles", *HPCMP Users Group Conference*, San Diego, CA, 2009.

Wyatt, D.C., "Development and Assessment of a Nonlinear Wave Prediction Methodology for Surface Vessels", *Journal of Ship Research*, Vol. 44, No. 2, pp. 96–107, 2000.

Numerical Flow Analysis Capability Applications Project 2010

Kyle A. Brucker, Thomas T. O'Shea, and Douglas G. Dommermuth
Naval Hydrodynamics Division, Science Applications International Corporation (SAIC), San Diego, CA
{kyle.a.brucker, thomas.t.o'shea, douglas.g.dommermuth}@saic.com

John M. Levesque
Cray, Inc., Seattle, WA
{levesque}@cray.com

Kevin D. George, Richard I. Walters, and Michael M. Stephens
US Army Engineer Research and Development Center (ERDC), Unclassified Data Analysis and Assessment Center (UDAAC), Vicksburg, MS
{kevin.d.george, richard.i.walters, michael.m.stephens}@usace.army.mil

Abstract

The primary aim of this paper is to provide an account of our experience using large core counts on the new Cray® XE6 (Cray, Inc.) platform to simulate complex free-surface flow around a body using Numerical Flow Analysis (NFA). The 2010 Capability Applications Project (CAP) program provided the resources to complete two scaling studies and two production-level simulations of the flow around a submarine moving with constant forward-speed on the free surface. The simulations were performed at two different grid resolutions corresponding to 3.2 and 25.8 billion grid cells. The 3.2 billion-cell simulation ran for 32,000 time-steps using 12,288 Cray XE6 cores and took 45 wall-clock hours. The 3.2 billion-cell job generated 22 terabytes of data and 4.2 terabytes were archived for future study. The 25.8 billion-cell simulation ran for 48,000 time-steps using 24,576 Cray XE6 cores and took 225 wall-clock hours. The 25.8 billion-cell job generated 243 terabytes of data and 23.5 terabytes were archived for future study. Results from the two submarine simulations presented here include visualizations of the free surface, which illustrates the ability of NFA to predict the generation of spray, the entrainment of air, and the breaking of waves. Issues regarding the visualization of the large data sets are identified and discussed. Results from the scaling studies show that a two-sided, non-blocking, send/receive nearest-neighbor communication model is superior to a one-sided remote memory access (RMA) nearest-neighbor communication model. The one-sided RMA model fails to scale at large core counts because collective communication operations are required. A five-percent increase in the performance (measured in iterations per hour) can be gained with a custom Message Passing Interface (MPI) process layout that accounts for the physical location of the nodes, for core counts greater than 16,384. The file output shows substandard performance between 512 and 4,096 cores. The difference between executables generated with the Cray and The Portland Group (PGI®) FORTRAN compilers is found to be between 10–15%, with the PGI compiler being faster.

1. Introduction

The ability to predict the complex flow around a naval combatant is very useful for the design and analysis of Navy ships. The ability to model breaking waves, spray formation, air entrainment, and free-surface turbulence is also important to global warming studies. In terms of computational science, we need to be able to develop elliptic solvers that scale well as the number of processor cores increase to hundreds of thousands. Elliptic solvers are more difficult to implement on parallel computers than solvers associated with hyperbolic and ordinary differential equations. The goal of this project is to illustrate the efficient application of a complex elliptic solver to a practical problem of relevance to the Department of Defense (DoD). In addition, the flow visualization of very large datasets is very challenging. The rendering of the iso-surfaces and volumetric data provided a good test of current flow visualization capabilities. Readers interested in

other projects involving Numerical Flow Analysis (NFA) are referred to an accompanying paper titled "Computational Naval Ship Hydrodynamics" (Brucker, et al., 2011). The format of this paper is as follows: Section 2 provides details and references for the NFA code, Section 3 describes the parameters for the two production simulations, Section 4 provides a visual comparison of the results of the simulations and a discussion of issues identified with the visualization tools, and Section 5 discusses the scaling of the code, and modifications necessary to improve the scaling, followed by a summary of the results and their significance to the DoD.

2. Numerical Approach

The Numerical Flow Analysis (NFA) computer code is employed in this study. NFA solves the Navier-Stokes equations utilizing a cut-cell, Cartesian-grid formulation with interface-capturing to model the unsteady flow of air and water around moving bodies. The interface-capturing of the free surface uses a second-order-accurate, volume-of-fluid technique. The cut-cell method is used along with the near-body model of Rottman, et al. (2010) to satisfy a partial-slip condition on the hull. NFA uses an implicit subgrid-scale model that is built into the treatment of the convective terms in the momentum equations (Rottman, et al., 2010). A panelized surface representation of the ship hull (body) is all that is required as input in terms of body geometry. A domain decomposition is used to distribute portions of the grid over a large number of processors. The algorithm is implemented on parallel computers using FORTRAN 2003 and the Message Passing Interface (MPI). The interested reader is referred to Dommermuth, et al. (2007); O'Shea, et al. (2008); and Brucker, et al. (2010) for a detailed description of the numerical algorithm and of its implementation on distributed memory high performance computing (HPC) platforms.

3. Large Production Simulations

The hull geometry of a generic submarine is shown in Figure 1 (left frame). In all the simulations, the bow of the submarine is fixed at x=0 and the flow is from positive x. Inflow and outflow boundary conditions are used in the streamwise (x) direction, and free-slip conditions are used in the span-wise (y) and cross-stream (z) directions. The inflow condition is a free-stream current, and the outflow condition is a one-dimensional, non-reflective, Orlanski-type boundary condition. The Froude number is $Fr=U(gL)^{-1/2}=0.28$, where U and L are, respectively, the speed and length of the submarine, and g is the acceleration of gravity. The extents of the computational domain, shown in Figure 1 (right frame), in the streamwise (x), span-wise (y), and cross-stream (z) directions are respectively [-2.5L, 1.0 L], [-1.0 L, 1.0 L], and [-0.75 L, 0.50 L]. Two computational domains and time-steps are employed, and are herein referred to as coarse (**c**) and fine (**f**). The number of grid points [nx, ny, nz] are [4,096, 1,536, 512]$^{\mathbf{c}}$ and [8,192, 3,072, 1,024]$^{\mathbf{f}}$. The coarse grid has 3.2 billon-cells, while the fine grid has 25.8 billion-cells. The grid is stretched with nearly uniform spacing around the submarine where the grid spacing is [0.00032L, 0.00028L, 0.00025L]$^{\mathbf{c}}$ and [0.00016L, 0.00014L, 0.000125L]$^{\mathbf{f}}$. The maximum grid spacing far away from the ship along the Cartesian axes is [0.025L, 0.014L, 0.018L]$^{\mathbf{c}}$ and [0.0129L, 0.007L, 0.009L]$^{\mathbf{f}}$. Note for the fine grid the dimensional resolution near the hull is 0.5 inches. The time-steps for the coarse and fine simulations are respectively $\Delta t=0.000125^{\mathbf{c}}$, and $\Delta t=0.0000625^{\mathbf{f}}$. The grid points are distributed in 128×64×64 blocks over 12,288 cores for the coarse simulation; and in the fine simulation, the grid points are distributed in 128×128×64 blocks over 24,576 cores. All simulations have been run on the Cray® (Cray, Inc.) XE6 platforms located at the US Army Engineering Research and Development Center (ERDC), and at the US Air Force Research Laboratory (AFRL). The coarse simulations require 11.25 wall-clock hours per boat length, T ($T=L/U$), while the fine simulations require 75 wall-clock hours per boat length. The iso-surfaces are output every five time-steps for the coarse simulation and every 10 time-steps for the fine simulation. The entire data field (three velocity components, pressure, and volume fraction) is output every 500 time-steps for the coarse grid and every 1,000 time-steps for the fine grid to permit analysis of turbulence, air entrainment, and spray formation. Each save of the entire data field requires 160 gigabytes of storage for the coarse resolution and 1.25 terabytes of storage for the fine simulation.

4. Visualizations

Figure 2 illustrates the effects of free-surface turbulence (yellow rectangle), spray formation (white circle), spilling breaking waves (red triangle), and air entrainment along the side and back of the hull. Figure 3 shows a comparison between the 3.2 billion-cell (left column) and 25.8 billion-cell (right column) simulations, at times 1.0, 1.5, 2.0, 2.5, and 3.0 L/U, from top to bottom. Note the much finer detail in the spray around the sail and in the turbulence on the free surface of the wake. For the higher resolution case, the water wets the hull differently, the air entrainment and spray formation are greater, and the turbulent roughening is greater. Analysis of the volumetric data including spectra and void fractions is ongoing.

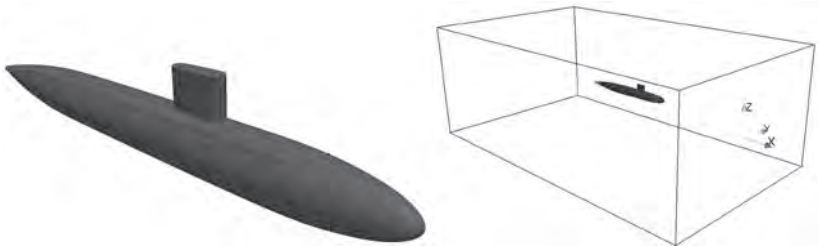


Figure 1. Model of generic submarine zoomed in view of hull on the left and computational domain on the right

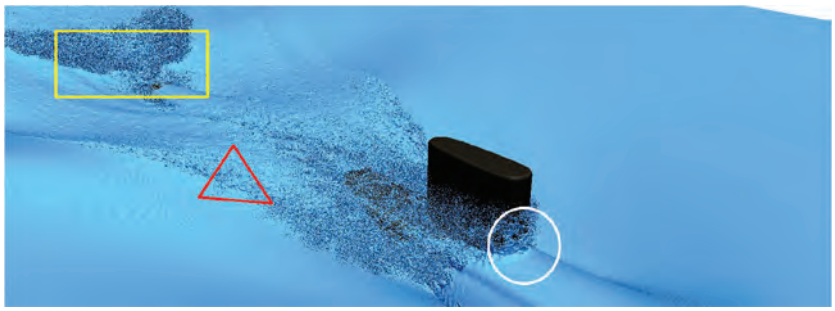


Figure 2. Visualization from the 25.8 billion-cell NFA simulation of a surfaced submarine. The white circle highlights the run-up on the sail and the formation of spray. The red triangle highlights a region of wave breaking and air entrainment. The yellow rectangle highlights the wake region where there is spilling wave breaking, turbulent roughening, and air entrainment.

The visualizations shown in Figures 2 and 3 were generated by NFA. The NFA has a marching cubes iso-surface algorithm built in so that data can be extracted from the simulation while it is running. The iso-surface routine in NFA also has a parallel writer so that the iso-surface files can be written efficiently. The files are written such that they are compatible with visualization software based on the Visualization Tool Kit (VTK). The parallel write routine was based on the binary LEGACY_VTK format, which requires that binary files be written with big-endian data. The ability to specify the data type of 'external_32', i.e., big-endian, to MPI_FILE_SET_VIEW was removed from the Cray MPI implementation[1], and hence NFA's parallel iso-surface writer was re-written to use the XML_VTP format, which supports little-endian data. The LEGACY_VTK to XML_VTP transition has the added advantage of an easy transition to parallel visualization, i.e., the pvtp format.

The 3.2 billion cell simulation has on the order of 25 million triangles per iso-surface, which can be handled by ParaView™ (Kit, Inc.) for visualization and analysis. With some down-sampling, the 25 million triangle iso-surfaces can be rendered by the Data Analysis and Assessment Center (DACC) with commercial ray-tracing software. However, the 25.8 billion-cell simulations have on the order of 150 million triangles per iso-surface, which causes problems for ParaView and cannot be ray-traced because the number of facets exceeds the capabilities of the rendering software (3D Studio Max, Autodesk Inc.'s 3ds Max®). ParaView 3.4, 3.6.2, and 3.8.1 all have trouble with the vtp file format when the number of triangles exceeds around 120 million. It appears that 32 bit integers are used in the vtp reader to represent offsets of the points, connectivity and offsets of data arrays. In particular, the offset of the offsets array, underlined in Figure 4, is suspect. When the number of triangles approaches 120 million, ParaView gives the runtime error shown in Figure 4. Figure 4 also provides the xml header from a file which causes the ParaView runtime error. As an aside, a custom-built version of Paraview 3.6.2 on a CentOS 4.7 workstation is able to read the vtp files with over approximately 120 million triangles. This issue has been reported to public.kitware.com as a bug report 11772 (see http://public.kitware.com/Bug/view.php?id=11772).

5. Scaling Analysis

Scalability quantifies a program's ability to utilize an increasing number of processing elements (cores) effectively. Typically, when the scalability of parallel codes is discussed, it is strong scaling that is being referenced. Strong scaling

[1]"The following MPI-2.2 features are not supported: canceling of MPI send requests, dynamic process management, and 'external32' data representation." From the Cray intro_mpi(3) document available at: http://docs.cray.com.

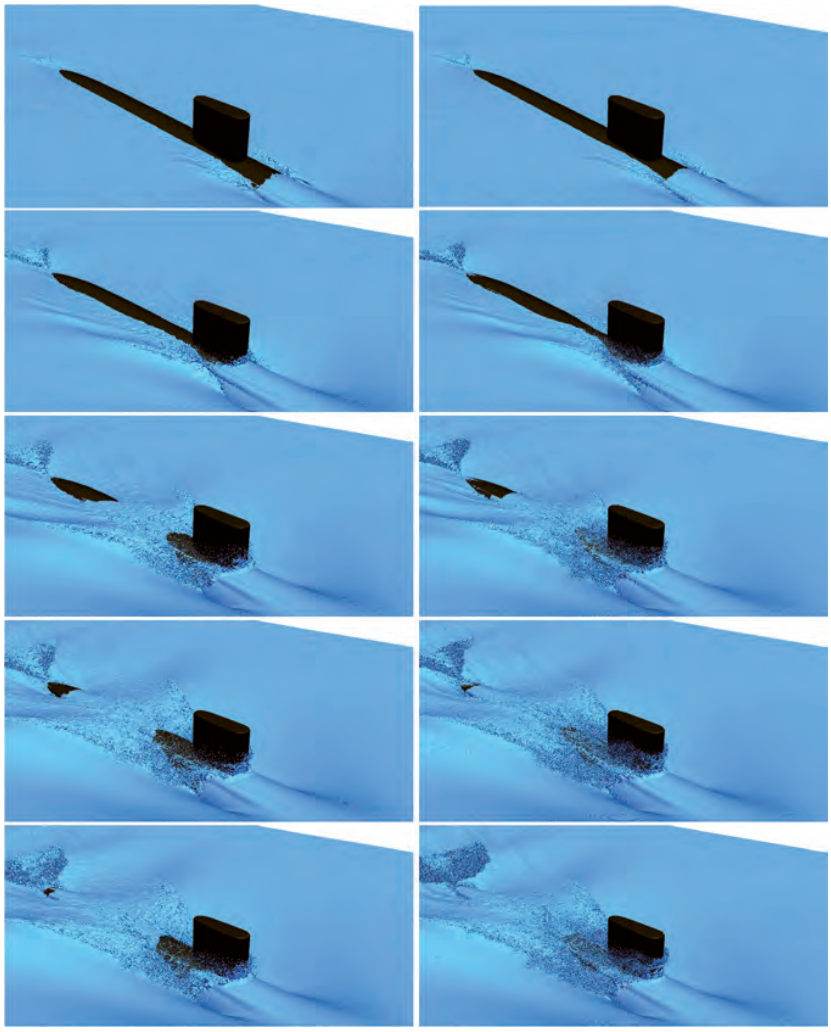


Figure 3. Comparison of the 50% iso-surface of the volume fraction from the 3.2 billion cell (left column) and 25.8 billion cell (right column) simulations at time 1.0, 1.5, 2.0, 2.5, and 3.0 *L*/*U* (from top to bottom). Note the differences in the run-up on the sail, the breakup of the flow around the sail, and in the turbulence on the free-surface in the wake.

PARAVIEW RUN TIME ERROR:

Error: In /Users/partyd/Kitware/ParaView-3.8/src/VTK/IO/vtkXMLUnstructuredDataReader.cxx, line 589 vtkXMLPolyDataReader (0x10a504c00): Cannot read cell offsets from Polys in piece 0 because the "offsets" array is not long enough.

XML_VTP HEADER:

```
<?xml version="1.0"?>
  <VTKFile type="PolyData" version="0.1" byte_order="LittleEndian">
    <PolyData>
      <Piece NumberOfPoints=" 61342156" NumberOfVerts="0" NumberOfLines="0"
  NumberOfStrips="0" NumberOfPolys=" 118579532">
        <Points>
          <DataArray type="Float32" NumberOfComponents="3" format="appended" offset="0"/>
        </Points>
        <Polys>
          <DataArray type="Int32" Name="connectivity" format="appended" offset=" 736105876"/>
            <DataArray type="Int32" Name="offsets" format="appended" offset=" 2159060264"/>
          </Polys>
        </Piece>
      </PolyData>
  <AppendedData encoding="raw">
```

Figure 4. Header of a vtp file which breaks binary distributions of ParaView (3.4, 3.6.2, 3.8.1) for Mac® OS X (Apple, Inc.) and CentOS 5. When the offset of the offsets array, underlined and in red, becomes larger than the size of a 32 bit integer, ParaView gives the run time error shown at the top of this figure. Note: 2159060264 > 2147483648 (2^{31}).

means given a fixed overall problem size, how the time-to-solution varies with the number of cores. For programs that are memory-bound, the problem size per core is limited by the available memory, and the overall problem size is too big to run on a single core. In this scenario, scaling means how the time-to-solution varies with a core count with a fixed system size per core; hence, when one doubles the number of cores, one also doubles the problem size. This type of scaling analysis is referred to as weak scaling, and strictly speaking, considers how efficiently machine resources are being used as a function of the problem size. Linear scaling is achieved if the run time stays constant while the workload is increased in direct proportion to the number of processors. Scott and Xie (1997) define the weak scaling efficiency, $E(N,P)$, as:

$$E(N,P) = \frac{T(n,1)}{T(N,P)}, \tag{1}$$

where n is the problem size per core, N is the total problem size, P is the number of cores, $T(n,1)$ the execution time on a single core, and $T(N, P)$ the execution time on P cores. Here, instead of $T(n,1)$, $T(n,8)$ is used in the efficiency calculations to correct for the initial overhead of nearest-neighbor communication, since with n=8 there are two processors in the x, y, and z domain decompositions and, hence, nearest-neighbor communication occurs in all three directions. The rationale behind this choice is as follows: A simulation with P=1 would not be sufficient to investigate the physics of interest, and the primary purpose of this study is to investigate the scaling of NFA at large core counts; hence, a discussion of the scaling at small core counts is tangent to the objective and omitted. Both the modified definition of weak scaling efficiency and iterations per hour are used in the subsequent discussion on the scaling of the NFA code. To test the scaling of NFA, the number of grid points was systematically increased up to 87 billion grid cells. The numerical simulations were distributed over an array of processors while fixing the size of each sub-domain on each core to n=128^3 grid points. The NFA simulations were performed for 201 time-steps, with full fields written every 100 time-steps. Here it is noted that the weak scaling was computed with and without file output, and Figure 5 does not include file output, which is discussed separately in Figure 7(a). The left frames of Figures 5 and 8 show the weak scaling efficiency for core counts between 8 and 1,024, and the right frames show the weak scaling efficiency for core counts between 1,024 and 41,472.

Recent advances made by John Levesque and the Cray Supercomputing Center of Excellence to the data communication routines significantly improved the scaling of NFA. By replacing the one-sided remote memory access (RMA) MPI operations with non-blocking send/receive pairs, the scaling improved significantly at large core counts. This is evidenced by the difference in the line with filled squares and the line with filled circles in Figure 5. The case with 41,472 cores has 87 billion grid points, whereas the case with eight cores has only 16.7 million grid points. The difference in the number of grid points is a factor of 5,184, while the number of iterations per hour only decreases by 15 percent. In its current state, NFA is able to run on tens of thousands of processors, and the scaling bodes well for NFA to run efficiently on even larger numbers of cores. For jobs that used more than 16,384 cores, a custom node placement algorithm (black diamond line shown in Figure 5), improved the efficiency by 5 percent. However, for smaller core counts, the better performance was achieved with the default value of MPICH_RANK_REORDER. Figure 6 shows the code to generate a custom node layout for core counts greater than 16,384.

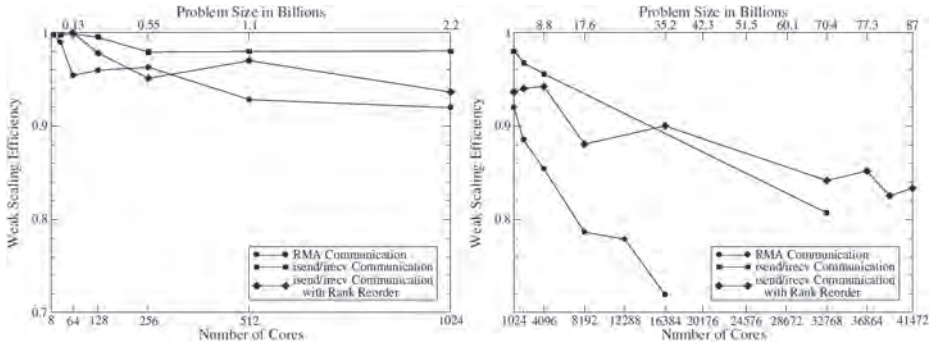


Figure 5. Weak scaling efficiency, $E(N, P)$ defined in Equation 1, of Numerical Flow Analysis (NFA) code compiled with Cray FORTRAN on Cray XE6 platform. File output time not included.

```
setenv MPICH_RANK_REORDER_METHOD 3  #add to job file (tcsh)
export MPICH_RANK_REORDER_METHOD=3 #add to job file (bash)

program mpi_task_reorder
!For Cray XE6, with 16 cores per node
integer,parameter                    :: l4=4
integer(l4)                          :: nx, ny, nz, total_cores, ncores
integer(l4)                          :: nx1, ny1, nz1, i, ii, iii, j, jj, k, kk, l
integer(l4),allocatable,dimension(:) :: mpi_task

read(5,*) nx,ny,nz,ncores
allocate(mpi_task(1:nx*ny*nz*ncores))

nx1 = 4
ny1 = 2
nz1 = 2
iii = 0
do k = 1,nz,nz1
 do j = 1,ny,ny1
  do i = 1,nx,nx1
   do kk = k,k+nz1-1
    do jj = j,j+ny1-1
     do ii = i,i+nx1-1
      iii = iii + 1
      mpi_task(iii) = (ii-1) + (jj-1) * nx + (kk-1) * nx*ny
     enddo
    enddo
   enddo
  enddo
 enddo
enddo
open(unit=2001,file='MPICH_RANK_ORDER')
write(2001,*) (mpi_task(l),',',l=1,nx*ny*nz)
close(2001)
deallocate(mpi_task)
stop
end program mpi_task_reorder
```

Figure 6. Code to generate custom mpi task layout for large core counts on Cray XE6. Note MPICH_RANK_REORDER file must be placed in the run directory, and environment variable MPICH_RANK_REORDER needs to be set to 3 in the job file.

Figure 7(a) shows the scaling results with (fill circles, black line) and without (filled squares, green line) file output. Note that the largest job with 87 billion grid cells wrote 3.24 terabytes of data every 100 time-steps. The file output performance degraded more rapidly than the overall code. The ERDC recommends[2] setting the Lustre file stripping to file size in gigabytes divided by 16. Based on the results of the Phase I scaling on Garnet, in which the output directories were stripped according to the ERDC recommendations, the file output took over 50 percent of the run time at 20,176 cores (not shown). For the Phase II scaling, the stripe count was increased. The stripe counts used in the Phase II scaling are given in Table 1, along with the individual file size and total data size, for a problem size per core of n=128×128×128 run for 201 iterations with full field data saved every 100 iterations. The large drop in the performance of the file output between 256 and 8,192 cores requires further investigation. Figure 7(b) shows the average write-speed in gigabytes per second, calculated by dividing the total amount of data written in gigabytes by the total time taken to write the data. There is a plateau in the write-speed of 1 gigabyte per second, which starts between 512 and 1,024 and ends between 4,096 and 8,192. Details of the high-speed interconnects, such as the maximum bandwidth and available bandwidth between a cabinet and object storage server, as well as details of the Lustre® (Oracle America, Inc.) file system such as how the object storage targets (OSTs) are laid out on the object storage servers (OSSs), need to be considered in the future to fully understand why the data throughput rate is plateauing.

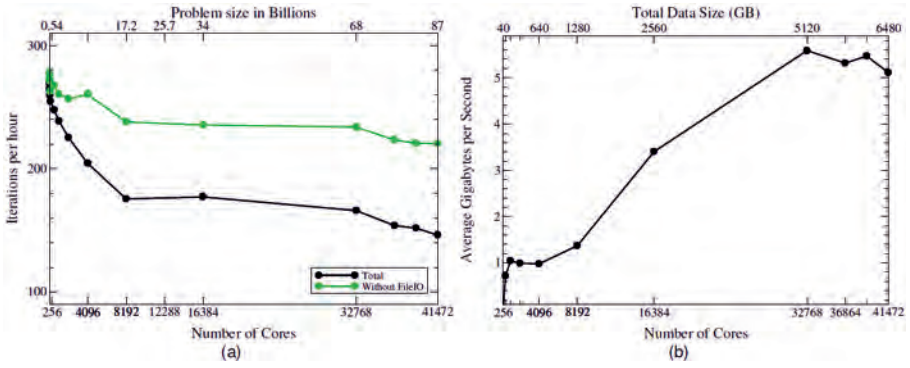


Figure 7. (a) Iterations per hour for Numerical Flow Analysis (NFA) code compiled with PGI FORTRAN on Cray XE6 platform, with and without file output. (b) Average gigabytes per second data write-speed.

[2] http://www.erdc.hpc.mil/documentation/Tips_Tricks/lustrePerformance

Table 1. Parameters for NFA scaling simulations for fixed problem size per core of *n*=128×128×128

Cores	Size in Billions	File Size (GB)	Total Data Size (GB)	stripe count ERDC rec.	stripe count used
1	0.002	0.015625	0.15625	0.009765625	4
2	0.004	0.03125	0.3125	0.001953125	4
4	0.008	0.0625	0.625	0.00390625	4
8	0.017	0.125	1.25	0.0078125	4
16	0.034	0.25	2.5	0.015625	4
32	0.067	0.5	5	0.03125	4
64	0.134	1	10	0.0625	4
128	0.268	2	20	0.125	4
256	0.537	4	40	0.25	8
512	1.074	8	80	0.5	8
1024	2.147	16	160	1	8
2048	4.295	32	320	2	8
4096	8.590	64	640	4	8
8192	17.180	128	1280	8	16
16384	34.360	256	2560	16	32
32768	68.719	512	5120	32	32
36864	77.309	576	5760	36	64
39168	82.141	612	6120	38.25	64

Figure 8 shows the difference between the PGI FORTRAN (blue triangles) and Cray FORTRAN (black diamonds) compilers. It should be noted here that version 10 of the PGI compiler has an optimization switch, -Msmartalloc, that improves the execution time by 15 percent. The authors are unaware of a similar switch for the Cray compiler. The compile time of the PGI compiler was also significantly faster than the Cray. However, some relief is possible by using 'make –j 16', with a makefile that has all of the dependencies properly listed. It is also worth noting here that NFA does a significant amount of memory allocation and deallocation as different portions of the code are executed, and hence the performance gains realized by compiling with –Msmartalloc may not be realized for codes that do not dynamically reallocate memory.

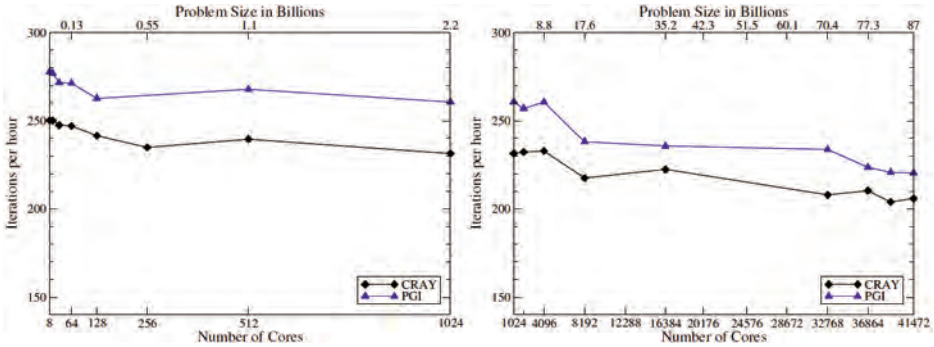


Figure 8. Iterations per hour for Numerical Flow Analysis (NFA) code compiled with Cray FORTRAN and PGI FORTRAN on Cray XE6 platform. Note file IO time is not included.

6. Conclusion

To the author's knowledge, the 25.8 billion cell simulation is the largest fluids simulation of a complex, incompressible free-surface flow with a body ever completed. The results of the scaling tests show that the NFA can be compiled and utilize tens of thousands of cores on the Cray XE6 platform for a typical production-level simulation, with good weak-scaling

efficiency. Figure 9(a) shows the results of the improvements that were made to the NFA code based on the participation in the 2010 Capability Applications Project (CAP) program. The significant improvement to the scaling is a result of changing from an RMA MPI communication model to a two-sided non-blocking send/receive communication model. Figure 9(b) shows the scaling results of simulations with n=256×128×128 grid cells per core. These simulations require 1.9 gigabytes of memory per core (near the 2-gigabyte per core hardware limit). The far-right data point, in Figure 9(b), 41,472 cores, demonstrates the full ability of the AFRL's Cray XE6, Raptor, in that it uses nearly all the available memory and all of the available cores concurrently to simulate a complex free-surface flow with 174 billion grid cells. There are fewer iterations per hour because there is more work with n=256×128×128 versus n=128×128×128. The weak-scaling coefficient of 0.87 bodes well for NFA to scale out to hundreds of thousands of cores and to one trillion or more grid-cells.

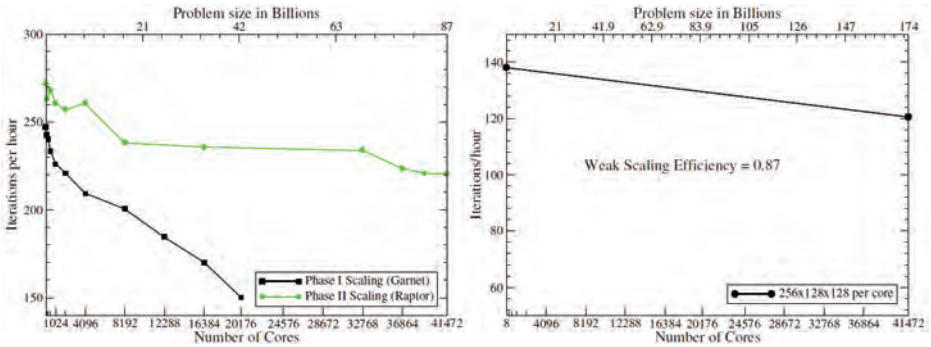


Figure 9. Iterations per hour, not including file IO achieved with Numerical Flow Analysis (NFA) code compiled with PGI FORTRAN on the Cray XE6 platform. (a) Phase I and Phase II 2010 CAP scaling, with 128×128×128 grid points per core. (b) Iterations per hour achieved with 256×128×128 grid points per core (maximum based on 2GB/core available memory).

7. Significance to DoD

The complex interactions of naval combatants with the ocean environment and the resulting impact on the performance of and threats to naval combatants are some of the main challenges remaining in modern naval hydrodynamics. Recent software developments and hardware advances provided by the Department of Defense (DoD) High Performance Computing Modernization Program (HPCMP) have enabled the development of a full-scale numerical wave tank, bringing us closer to meeting these challenges. To put it into perspective, consider the following: An NFA simulation using 41,472 cores on a Cray XE6 would allow for a simulation with 174 billion grid cells. One hundred seventy-four billion grid cells would allow for a dimensional grid spacing of around 0.25 inches over the entire length of a full-scale ship (several hundred feet), which is approaching the resolution needed to model spray.

The ability to efficiently solve a complex elliptic problem is also an important problem in computational science. The results of our scaling tests show that the multi-grid solver used by NFA scales out to the size of the largest DoD machine.

The ability to extract data while the simulation is running in a parallel fashion has been added. This allows for much higher temporal resolution for quantities of interest while reducing the amount of data output and post-processing time. Advancing these techniques will reduce the strain on the mass storage and archival storage systems and improve the quality of both the visualizations and scientific/engineering data.

Acknowledgements

The authors would like to acknowledge Dan Redig for his insight and input on the file output analysis. His help with interpreting the results in terms of the performance of Lustre file systems and with high-speed interconnects greatly improved our understanding of the complex issues involved. Collaboration is ongoing to develop strategies to analyze and optimize the Numerical Flow Analysis file output for the parallel file systems, such as Lustre, found in high performance computing environments.

This work is supported in part by a 2010 CAP grant of computer time from the DoD HPCMP (http://www.hpcmo.hpc.mil/), and in part by SAIC research and development funding. The numerical simulations have been performed on the Cray XE6 platforms located at the ERDC and the AFRL. Animations of NFA simulations are available at http://www.youtube.com/waveanimations.

References

Brucker, K.A., O'Shea, T.T., Beale, K.L.C., Dommermuth, D.G., Hendrickson, K., Weymouth, G., Yue, D.K.P., Levesque, J., George, K.D., Walters, R.I., and Stephens, M.M., "Computational Naval Ship Hydrodynamics," *Proceedings of the DoD High Performance Computing Modernization Program 21st Users Group Conference*, Portland, OR, USA, 2011.

Brucker, K.A., O'Shea, T.T., and Dommermuth, D.G., "Numerical simulations of breaking waves – weak spilling to strong plunging," *Proceedings of the 28th Symposium on Naval Hydrodynamics*, Pasadena, CA, USA, 2010.

Dommermuth, D.G., O'Shea, T.T., Wyatt, D.C., Ratcliffe, T., Weymouth, G.D., Hendrikson, K.L., Yue, D.K., Sussman, M., Adams, P., and Valenciano, M., "An application of Cartesian-grid and volume-of-fluid methods to numerical ship hydrodynamics," *Proceedings of the 9th International Conference on Numerical Ship Hydrodynamics*, Ann Arbor, MI, USA, 2007.

Levesque, J.M. and Wagenbreth, G., *High performance computing: programming and applications*, Chapman & Hall/CRC Computational Science, 2010.

O'Shea, T.T., Brucker, K.A., Dommermuth, D.G., and Wyatt, D.C., "A numerical formulation for simulating free-surface hydrodynamics," *Proceedings of the 27th Symposium on Naval Hydrodynamics*, Seoul, Korea, 2008.

Rottman, J.W., Brucker, K.A., Dommermuth, D.G., and Broutman, D., "Parameterization of the internal wave field generated by a submarine and its turbulent wake in a uniformly stratified fluid," *Proceedings of the 28th Symposium on Naval Hydrodynamics*, Pasadena, CA, USA, 2010.

Scott, L.R. and Xie, D., "The parallel u-cycle multigrid method," *Proceedings of the 8th Copper Mountain Conference on Multigrid Methods*, Copper Mountain, CO, USA, 1997.

HPCMP UGC 2011

2. Multi-Physics (CFD, CSM, CCM, Plasma Physics...)

Blade-Vortex Interaction and Rotor Wake Prediction using the *Helios* Flow Solver

Buvaneswari Jayaraman and Arsenio Dimanlig
Science and Technology Corporation, US Army/ AFDD, Moffett Field, CA
{buvana.jayaraman, arsenio.dimanlig}@us.army.mil

Andrew M. Wissink, Joon Lim, and Mark Potsdam
US Army/AFDD, Moffett Field, CA
{andrew.m.wissink, joon.lim, mark.potsdam}@us.army.mil

Abstract

In this paper, we present the validation of the multi-disciplinary rotorcraft simulation code Helios for its ability to predict the blade-vortex interactions and the rotor wake in the descending flight. Helios uses a dual-mesh paradigm with unstructured mesh near the body, and Cartesian mesh in the off-body. Cartesian-based unsteady adaptive mesh refinement (AMR) was applied in the off-body flow-field. Helios predictions compare favorably with measured data and current state-of-the art codes like OVERFLOW2/CAMRAD II. Modeling these complex flows is computationally expensive. Routine calculations with coarse grids require 12 hours per rotor revolution on 256 cores, and fine grids require substantially more time and processors. However AMR is shown to offer significant advantages for computing high-fidelity vortical rotorcraft flow-field compared with fixed refinement meshes. Helios software modules and the integrated software are found to be scalable and efficient across different DoD compute platforms.

1. Introduction

Blade-vortex interaction (BVI) is a significant source of annoying and intrusive noise generated by helicopter rotors. This phenomenon is one of the distinctive features of helicopter rotors, with the noise source becoming especially intrusive during low-speed descending and maneuvering flights where the rotor wake is blown back into the rotor plane. The high noise levels produced by BVI loading may prohibit rotorcraft from achieving wide acceptance for civil applications. Being able to accurately predict BVI with computational models is therefore of interest to designers of rotary-wing aircraft. Research efforts have been undertaken to better understand BVI loading and noise and to provide a comprehensive database for the development of predictive computational models. The HART II test is one of the most widely-used comprehensive databases, providing high-quality measured data for researchers in this discipline[11,15]. This test was conducted in the German-Dutch Wind tunnel (DNW) using a 40% Mach-scaled Bo105 model rotor in descending flight. It offers a wealth of useful measurements including 3C-PIV (Particle Image Velocimetry) wake measurement, SPR (Stereo Pattern Recognition) deflection measurement, noise measurement, and blade pressures. These data are a key resource for investigating the interaction of rotor dynamics, aerodynamics, and rotor wakes.

The HPC Institute for Advanced Rotorcraft Modeling and Simulation (HI-ARMS) and the Computational Research and Engineering Acquisition Tools and Environments-Air Vehicles (CREATE-AV) program sponsored by the Department of Defense (DoD) High Performance Computing Modernization Program (HPCMP) have the goal of developing production-quality rotorcraft simulation software for use by the acquisition community to assess, with high-fidelity tools, the flight characteristics of both fielded vehicles and those in early conceptual design stages. One of the software products being developed within these programs is the rotary-wing simulation code *Helios* (**Heli**copter **O**verset **S**imulations)[6]. Rotorcraft computations are inherently multidisciplinary, requiring the solution of moving-body aerodynamics coupled with structural dynamics for rotorcraft blade deformations, vehicle flight dynamics, and controls. *Helios* integrates the collection of separate physics packages that model the aerodynamics, structural dynamics, and vehicle flight controls through a Python-based Software Integration Infrastructure (SIF). The Python-based integration facilitates extensibility. Interfaces are constructed in the component's native FORTRAN/C/C++ and translate to Python for exchange with other modules. By maintaining a defined data layout at the python level, all modules can access and share a common set of data with any data-type translations occurring at the interface level. This readily allows substitution of alternative modules, for instance to replace an existing model with one that shows better scalability or higher fidelity, or the addition of new modules, such as adding new physics or simulation capabilities.

Computational research efforts have been undertaken in the past to better assess the ability of rotorcraft computational fluid dynamics (CFD) codes to predict the BVI-loading for the HART II rotor configuration and have been found promising[2–4]. But these simulations required large dense-wake grids to resolve the tip vortices that lead to the onset of the BVI. Even with such expensive grids, the wake was still quite under-resolved. Application of dual-mesh with AMR was investigated in earlier studies for the TRAM rotor (1/4 scale model of V-22) in hover[13] and the UH-60A rotor in forward flight[10]. These studies found that the resolution of the near-body grid was critical to resolve the rotor airloads, while AMR was most effective in resolving the wakes downstream of the rotor. The goal of this challenge project is to evaluate the ability of the Army's new *Helios* computational modeling tool to predict BVI and rotor wakes for the HART II rotor and fuselage configuration in forward flight. We investigate the use of high-order Cartesian Adaptive Mesh Refinement (AMR) in *Helios* to see if it can better resolve BVI-induced tip vortices and consequently improve prediction of the BVI noise signatures at a reduced cost.

The outline of the paper is as follows. In Section 2, we present more details of the *Helios* software including adaptive mesh refinement. In Section 3, we present correlations between the computed and measured airloads, rotor wake data using the fixed off-body mesh for the rotor-fuselage configuration, and adaptive mesh refinement (AMR) for the isolated rotor configuration. In addition, we present some discussion on the performance of *Helios*. Concluding remarks are presented in the final section.

2. *Helios* Technical Approach

Helios employs an innovative dual-mesh paradigm that utilizes unstructured meshes in the near-body for ease of mesh generation, and Cartesian meshes in the off-body for better accuracy and efficiency[8,12]. Figure 1 shows an example of such a system. The two mesh types overlap each other with data exchange between the two systems being managed by a domain connectivity formulation. The unstructured meshes are body-conforming and are comprised of a mix of tetrahedra, prisms and pyramids. The Cartesian meshes are managed by a block-structured mesh system, which has the ability to conform to the geometry and solution features. The interpolation of fringe data from one mesh system to the other is managed by a domain connectivity module called PUNDIT[9]. PUNDIT utilizes implicit hole-cutting and is completely automated and scalable, which is especially important for dynamic mesh problems, wherein the near-body unstructured mesh may move with the body (for instance, in the case of the rotor meshes), while the background Cartesian mesh is stationary, but may change due to adaptation to geometry and/or solution features.

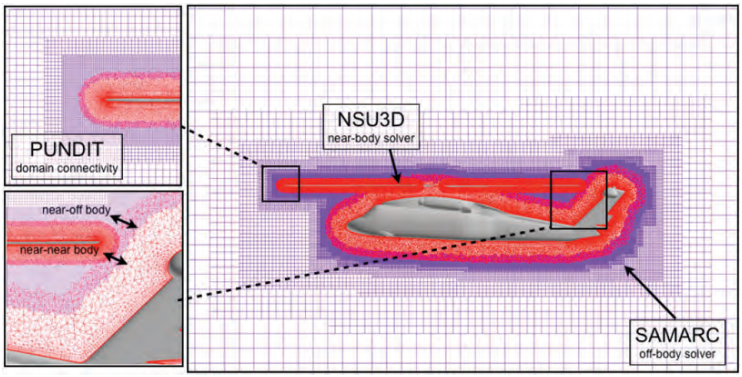


Figure 1. Dual-mesh CFD approach used by *Helios*

Helios is a multi-disciplinary computational platform that includes software components responsible for near-body (NSU3D) and off-body (SAMARC), domain connectivity (PUNDIT), rotorcraft comprehensive analysis (RCAS/CAMRAD), mesh motion and deformation (MMM), a fluid-structure interface module (RFSI) and a fluid-flight-dynamics interface (FFDI) module. All these components are interfaced together using a flexible and light-weight Python-based infrastructure called the Software Integration Framework (SIF). Figure 2 shows a schematic of SIF. In order to preserve modularity, all data transfer between components occurs through SIF and no component is allowed to "talk" directly to another component. The parallel execution of SIF is accomplished using by pyMPI.

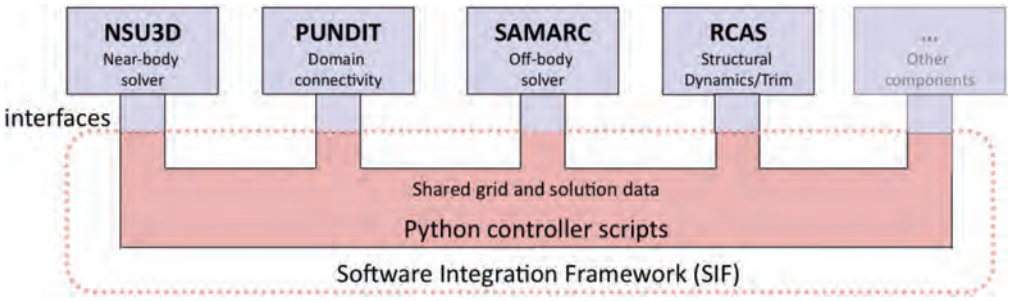


Figure 2. Python-based infrastructure in *Helios*

2.1 Adaptive Mesh Refinement

Adaptive mesh refinement is performed by the off-body solver SAMARC[14] to resolve the rotor wake. A Cartesian block-structured off-body grid system is automatically generated, refined, and de-refined around vortical features like tip vortices. The grid system is constructed using a Berger-Colella[1] approach in which a hierarchy of nested grid levels is constructed from coarsest to finest. The coarsest level defines the physical extent of the computational domain. Each finer level is formed by selecting cells on the coarser level, and then clustering the marked cells together to form the regions that will constitute the new finer level (Figure 3).

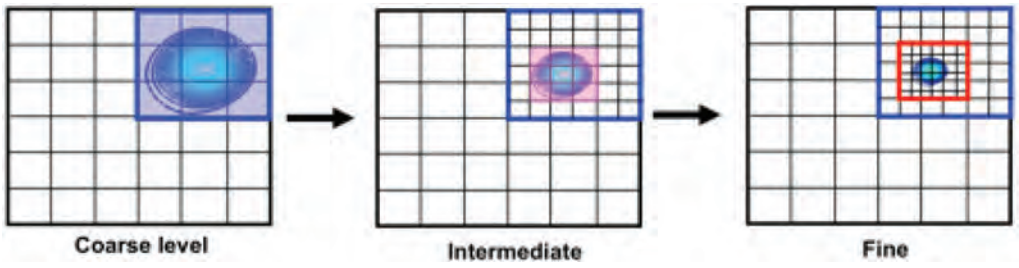


Figure 3. Block-structured AMR approach in *Helios*

The off-body grid is adapted frequently in forward-flight cases, generally every few degrees in azimuth, in order to allow the finer regions of the off-body mesh to keep up with the moving blades and tip vortices. Each adapt cycle incurs four operations: 1) mark regions of high-vorticity, 2) cluster those regions into grid blocks that constitute a finer level, 3) load-balance the blocks across parallel processors, and 4) transfer data from the pre-adapted grid system to the new adapted grid system, interpolating data from coarser meshes where necessary. These steps can incur significant overhead if not done efficiently on the parallel computer system.

The off-body grid is refined to target regions of high-vorticity using a local peak-vorticity, and also to match spacing of the near-body grid system. That is, it is necessary to retain similar resolution between the unstructured near-body and Cartesian off-body grid systems at the near-and off-body interface to avoid numerical errors. Consequently, on top of refining to solution features like tip vortices, the adaptive scheme also refines the off-body grid in the vicinity of the moving near-body grid to ensure commensurate resolution between the two grid systems where they overlap.

Further details of the adaptive Cartesian scheme and the implementation of the high-order solution algorithms on the adapted grid system are available in References 13 and 14.

2.2 Run-Time Environment

Helios modules are written in a variety of languages such as FORTRAN-77, FORTRAN-90, C++, Python, and wxPython. Moreover, *Helios* installation relies upon a number of other packages and tools such as HDF5, numpy, swig, matplotlib, and so on. Requiring the user to install all these packages as a pre-requisite to installing *Helios* is not only tedious but also time-consuming, and presents difficulties from the point-of-view of support and maintenance. To tackle such issues, *Helios* uses a customized build and run-time environment called `ptoolsrte`, developed by ParaTools, Inc.[7] This package bundles all the freely-available software packages that Helios relies on—viz., Python, numpy, swig, pyMPI, wxpython, matplotlib, etc. The primary goal is to avoid the burden of the end-user having to manually install these packages themselves. A secondary goal is to ease product support, because the development team can test a particular

configuration of the toolset and not have to consider additional complexities such as behavior with different versions of Python, for instance. The package is a hybrid source and binary distribution that provides most of the tools pre-built in a binary form, and is available on supported DSRC clusters.

3. Results and Discussions

The HART II rotor[11] is a 40% Mach and dynamically-scaled model of Bo105 hinge-less rotor, with 2m radius and 0.121m chord, operated at 1041 RPM (Ω=109*rad*/*sec*). To simulate a descending flight, the rotor had a shaft tilt of 5.3° aft with a thrust level (CT/σ) of 0.056 at an advance ratio of 0.15. After wind tunnel wall corrections, the corrected shaft angle in the computation model was 4.5° aft. The current study focuses on the HART II baseline case using the blade motion file, generated by coupling OVERFLOW2 structured grid CFD code and CAMRAD II at US Army AFDD. The FFDI module in *Helios* reads the prescribed deformation files. Figure 4b shows the computational grid for the HART II geometry (Figure 4a). The fuselage grid is fixed and the blade meshes move and deform during the simulation.

Figure 4b shows a cross-section of the *Helios* dual-mesh for the HART II rotor-fuselage configuration using a fixed off-body mesh. The near-body mesh is a mixed element unstructured grid (3M nodes), and it extends to about one chord length from the surface before intersecting with the Cartesian off-body mesh. The off-body domain extends approximately 80 chord lengths in each direction. Six levels of grids are used in the off-body with 35.2M non-overlapping nodes. The finest off-body grid had a spacing of 10% chord.

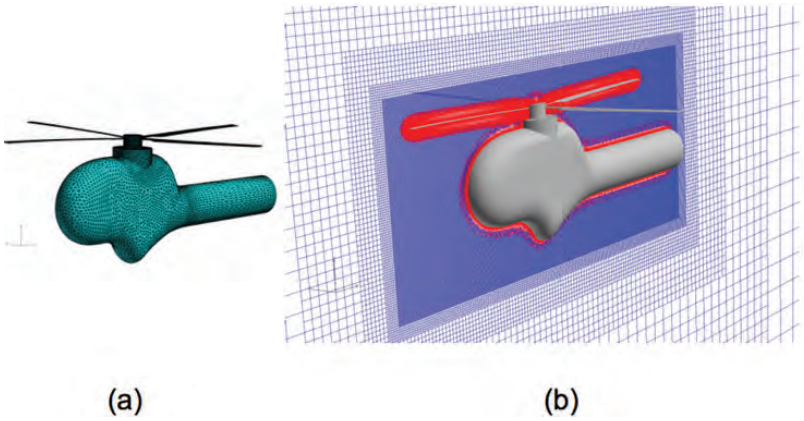


Figure 4. (a) HART II rotor-fuselage surface grid, (b) *Helios* dual-mesh: unstructured near-body mesh and Cartesian off-body mesh

A 0.1° time-step is used for time marching with 25 sub-iterations per time-step for the near-body, and 2 sub-steps per time-step for the off-body. Four revolutions were needed to get a fully-converged solution using 32 nodes (256 cores) on Harold (SGI Altix cluster at the US Army Research Laboratory [ARL]). *Helios* C_nM^2 prediction at r/R=0.87 is compared to the measured data in Figure 5a. As shown in the figure, the *Helios* load predictions for the rotor-fuselage are in good comparison with the measured data and OVERFLOW 2 results. BVI loading is captured well in both the advancing and the retreating side. Figure 5b shows that the isolated rotor case inaccurately predicts the BVI on the advancing side. Figure 6 shows the iso-surfaces of Q criterion for the rotor-fuselage case, colored by vorticity magnitude with red indicating high-strength. The rotor wake iso-surface shows significant details of discrete tip vortices traveling from the retreating side to the advancing side. Detailed comparison of the wake positions with the PIV measurement is part of the ongoing work.

3.1 Application of Adaptive Mesh Refinement

The adaptive mesh refinement (AMR) capability in *Helios* is tested using the isolated rotor configuration. AMR is applied to the off-body Cartesian grids in an effort to better resolve the wake solution (Figure 7a). Refinement is targeted to vorticity through the following means; 1) specify a user-defined threshold vorticity ω_{thresh} in the input file, 2) compute local vorticity at all grid-cells in the off-body at each adapt step, 3) mark grid-cells whose local vorticity exceeds the threshold value for refinement, and 4) cluster the selected cells into blocks that are refined to form a new finer level. Figure 7b shows the perspective view of rotor wake with AMR. AMR preserves the vorticity well downstream, much farther than the fixed-grid and displays very clear, strong vortex filaments without diffusion at the location of twice the rotor radius from the hub center.

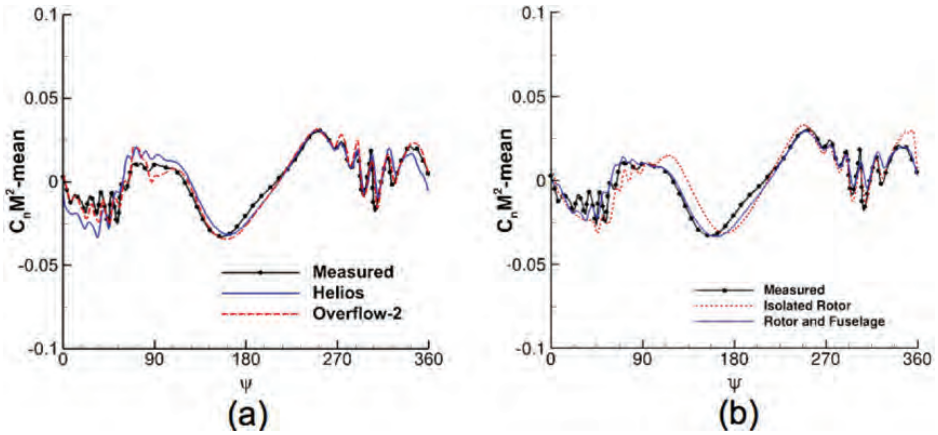


Figure 5. C_nM^2 predictions at *rlR*=0.87, (a) comparison of *Helios* and OVERFLOW 2 predictions for the rotor-fuselage configuration, (b) comparison of Helios rotor-fuselage and isolated rotor configurations

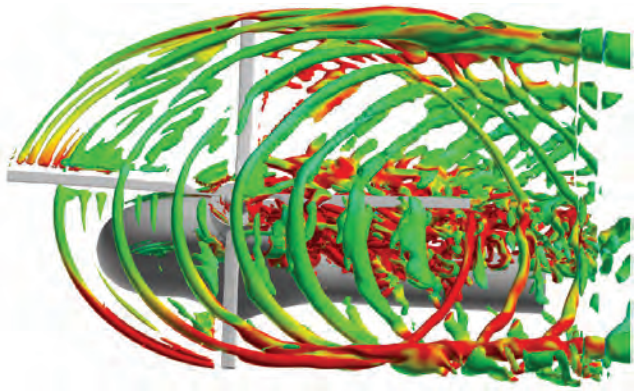


Figure 6. Q iso-surfaces for the HART II rotor-fuselage configuration

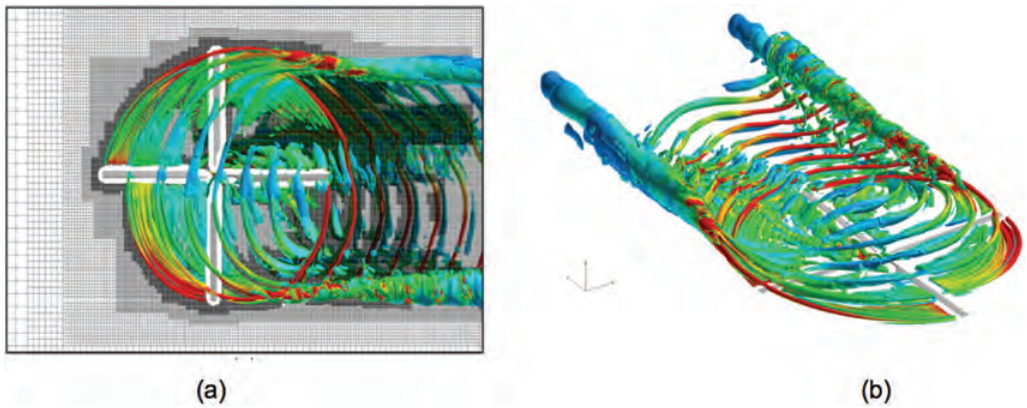


Figure 7. (a) Wake solution with AMR for isolated rotor configuration, (b) perspective view of AMR. Iso-surface of q-criterion=1e-4, colored by non-dimensional vorticity magnitude (blue low to red high, 0.0-2.6). Every other mesh point is shown.

No effort was needed to restrict refinement to regions near the blade, and the off-body code adapts to any region where it detects vorticity exceeding the user-specified threshold value. Likewise, no effort was made to prescribe a minimum refinement level. Although the solution is started from the L6-fixed-grid, the fixed-grid is not maintained after AMR is turned on. The AMR scheme allows the grid to be de-refined below the L6 level if the detected vorticity is weaker than the threshold value. Thus, the vorticity threshold value plays a very important role in resolving the wake, and thus selecting the appropriate threshold involves user's experiences to some degree. For a detailed report on the validation of *Helios* for the isolated rotor configuration refer to Lim, et al.[5]

Figure 8a shows the comparison of off-body grid size. The computation cost per time-step remains roughly constant for the fixed off-body calculations but with AMR the cost varies. The size of the AMR grid immediately grows when AMR is turned on, then increases almost-linearly in size as it tracks vortices far downstream. AMR grid-count surpasses the fixed-count after one and a half revolutions as it resolves the wake downstream (8a). The number of off-body grid-points increased from 15M in the fixed-case to 87M when using AMR to track the wake downstream for 2 rotor revs. However, despite this nearly 6X increase in grid-points in the off-body wake-grid, the overall computational cost grew by only 30% (8b). This demonstrates the extraordinary efficiency of the high-order calculations on the Cartesian grid system, and also the efficiency of the dynamic AMR process.

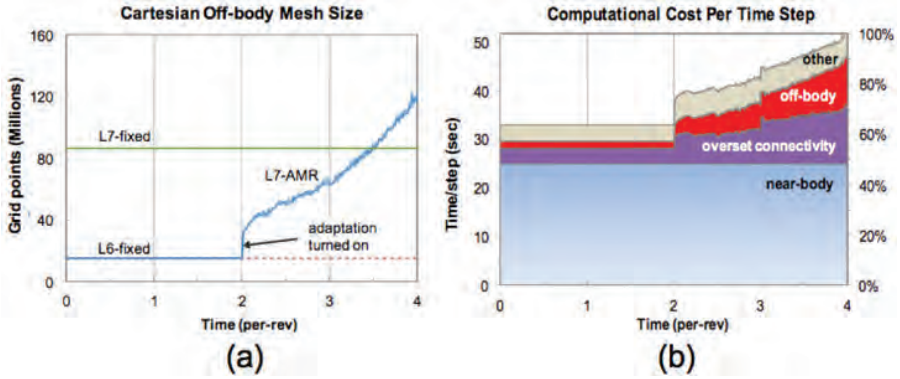


Figure 8. (a) Off-body grid count, (b) break down of computation time with AMR

Figure 9a shows the vortex core profile at one of the measurement locations (position 17). With the current off-body grid resolution of 10% chord (Figure 9a), there is only one grid-point inside the vortex core. The isolated rotor computations with AMR presented in this paper uses 5% chord spacing in the off-body which gives two nodes across the vortex core, and is still insufficient to capture the vortex. Prior research efforts indicate that about 10 points are required across the core of the vortex for a good resolution of the vortex. Figure 9b shows an estimate of the number of nodes across the vortex versus number of grid-points needed for a fixed and adaptively refined off-body mesh. Based on this estimate, an off-body spacing of 0.6% chord, results in 16 nodes across the core. To meet this requirement using a fixed off-body grid 20 billion nodes are required. Such a problem size is generally intractable today. Even if the computation could be completed using thousands of processors of an HPC system, the visualization and analysis software is not available to handle datasets of this size. But with the use of the Helios AMR capability, the number of grid-points reduces to 3 billion since grid-points are added only in the area of interest, resulting in a significant reduction in the computation time and effort required to get the same level of accuracy.

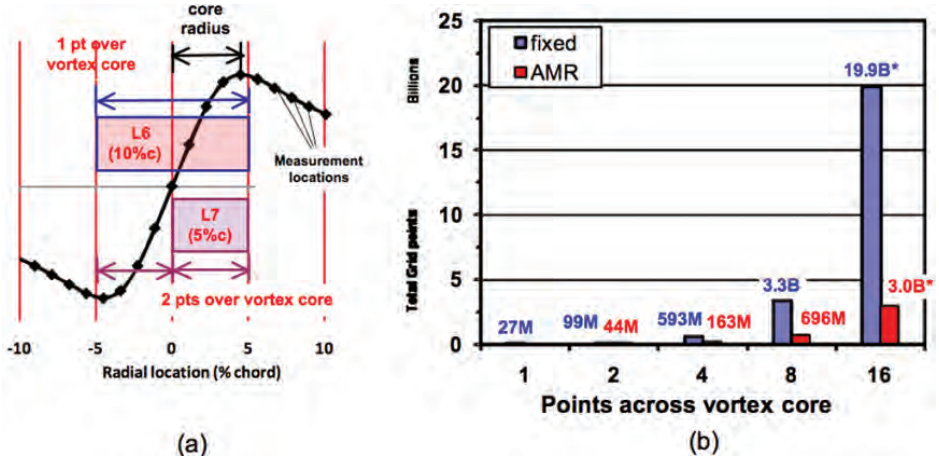


Figure 9. (a) Vortex core profile at one of the measurement location, (b) estimated off-body grid size

3.2 *Helios* Performance Analysis

In a multi-component code like *Helios*, poor performance or scalability of one of the modules can impede the performance of the integrated software. To understand the performance characteristics of the various modules in Helios and the performance of their integrated execution, the load-balance module in *Helios* automatically keeps track of the time spent in different modules, and reports the time at user-specified frequency. Figure 10a shows the breakdown of the computation time per time-step in the different modules of *Helios* for the HART II calculation run on 32 nodes (256 cores) of the Mana system. The largest amount of time was spent in the near-body solver, next was the domain connectivity, then the off-body solver. Figure 10b shows the comparison of the wall-clock time using 32 nodes (256 cores) on the Dell cluster Mana at Maui and the SGI Altix cluster Harold at ARL. The time spent in the different modules is comparable on both clusters, showing that the performance of *Helios* is maintained across platforms. Harold is found to be slightly faster than Mana. The domain connectivity solver PUNDIT takes slightly longer on Harold, and we are currently investigating the reason for the difference.

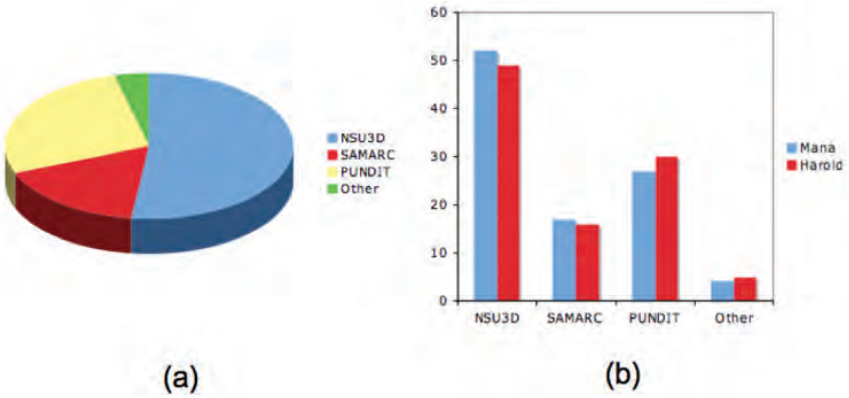


Figure 10. (a) Cost per time-step on Mana for different modules in Helios, (b) comparison of performance on Mana (Dell cluster at Maui DSRC) and Harold (SGI cluster at ARL DSRC)

4. Concluding Remarks

This paper presents the results from the validation of Helios for a descending flight using the HART II database, utilizing the innovative dual-mesh paradigm with unstructured grids in the near-body and structured Cartesian mesh in the off-body. Computed results with the rotor-fuselage configuration exercises all the key modules of *Helios*, and demonstrates its ability to handle multiple interacting components with ease. The C_nM^2 predictions are in good agreement with the measured data and the presence of the fuselage improved the prediction on the BVI loading on the advancing side. The adaptive mesh refinement (AMR) capability was successfully demonstrated. AMR automatically detects and refines to the wake features and preserves voriticity well downstream. Also, AMR is found to be a promising tool to accurately simulate complex wake structures with significantly less grid-points and computational effort compared to the fixed-grids. Finally the comparison of *Helios* performance on Harold and Mana systems shows comparable performance in the different *Helios* components on the different platforms. Most of the time is spent in the unstructured near-body solver, so future work will target improved efficiency in this portion of the solver.

Acknowledgments

Material presented in this paper is a product of the CREATE-AV Element of the CREATE Program sponsored by the US Department of Defense HPC Modernization Program Office. This work was conducted at the High Performance Computing Institute for Advanced Rotorcraft Modeling and Simulation (HI-ARMS). The authors gratefully acknowledge the contributions of other members of the Helios Team, in particular Dr. Venke Sankaran, Dr. Anubhav Datta, Dr. Jay Sitaraman, and Dr. Roger Strawn. The authors would also like to acknowledge the support of HART II partners from AFDD, DLR, DNW, NASA, and ONERA.

References

1. Berger, M.J. and P. Colella, "Local adaptive mesh refinement for shock hydrodynamics", *J. Comp. Phys.*, 82, pp. 65–84, 1989.

2. Biedron, R.T. and E.M. Lee-Rausch, "Rotor airloads prediction using unstructured meshes and loose CFD/CSD coupling", *AIAA Paper 2008-7341*, 26th AIAA Applied Aerodynamics Conference, Honolulu, HI, August 18–21, 2008.

3. Lim, J.W., T.A. Nygaard, R. Strawn, and M. Potsdam, "Blade-vortex interaction airloads prediction using coupled computational fluid and structural dynamics", *Journal of the American Helicopter Society*, Vol. 52(4), pp. 318–328, October 2007.

4. Lim, J.W. and R. Strawn, "Computational modeling of HART II blade-vortex interaction loading and wake system", *Journal of Aircraft*, Vol. 45, No. 3, pp. 923–933, May–June 2008.

5. Lim, J.W., A.M. Wissink, B. Jayaraman, and A. Dimanlig, "Application of adaptive mesh refinement technique in *Helios* to blade-vortex interaction loading and rotor wakes", *American Helicopter Society 67th Annual Forum Proceedings*, Virginia Beach, VA, May 3–5, 2011.

6. Sankaran, V., A.M. Wissink, A. Datta, J. Sitaraman, B. Jayaraman, M. Potsdam, S. Kamkar, A. Katz, D.J. Mavriplis, H. Saberi, B. Roget, and R. Strawn, "Overview of the *Helios* V2.0 computational platform for rotorcraft simulations", *AIAA Paper*, 49th Aerospace Sciences Meeting, Orlando, FL, January 2011.

7. Shende, S., "An infrastructure for deploying multi-language CFD application, Final report", *PET CD-KY8-SP1*, https://okc.erdc.hpc.mil, 2009.

8. Sitaraman, J., A. Katz, B. Jayaraman, A. Wissink, and V. Sankaran, "Evaluation of a multi-solver paradigm for CFD using unstructured and structured adaptive cartesian grids", *AIAA-2008-0660*, 46th AIAA Aerosciences Conference, Reno NV, Jan 2008.

9. Sitaraman, J., M. Floros, A. M. Wissink, and M. Potsdam, "Parallel unsteady overset mesh methodology for a multi-solver paradigm with adaptive cartesian grids", *AIAA-2008-7117*, 26th AIAA Applied Aerodynamics Conference, Honolulu, HI, January 2008.

10. Sitaraman, J., M. Potsdam, B. Jayaraman, A. Datta, A. M. Wissink, D. J. Mavriplis, H. Saberi",Rotor loads prediction using Helios: A multi-Solver framework for rotorcraft CFD/CSD analysis", *Proceedings of the 49th AIAA Aerospace Sciences Meeting and Exhibit*, Orlando, FL, AIAA Paper 2011-1123, January 4–7, 2011.

11. van der Wall, B.G., "2nd HHC aeroacoustics rotor test (HART II) Part I: Test documentation", *DLR IB 111-2003/19*, Braunschweig, Germany, November 2003.

12. Wissink, A.M., J. Sitaraman, V. Sankaran, D.J. Mavriplis, and T.H. Pulliam, "A multi-code Python-based infrastucture for overset CFD with adaptive cartesian grids", *AIAA-2008-0927*, 46th AIAA Aerosciences Conference, Reno, NV, January 2008.

13. Wissink, A.M., M. Potsdam, V. Sankaran, J. Sitaraman, Z. Yang, and D.J. Mavriplis, "A coupled unstructured-adaptive cartesian CFD approach for hover prediction", *American Helicopter Society 66th Annual Forum Proceedings*, Phoenix, AZ, May 11–16, 2010.

14. Wissink. A.M., S. Kamkar, T.H. Pulliam, J. Sitaraman, and V. Sankaran, "Cartesian adaptive mesh refinement for rotorcraft wake resolution", *AIAA Paper 2010-4554*, 28th AIAA Applied Aerodynamics Conference, Chicago, IL, June 2010.

15. Yu, Y.H., C. Tung, B.G. van der Wall, H. Pausder, C. Burley, T. Brooks, P. Beaumier, Y. Delrieux, E. Mercker, and K. Pengel, "The HART-II Test: Rotor wakes and aeroacoustics with Higher-Harmonic Pitch Control (HHC) inputs-The Joint German/French/Dutch/US Project", *American Helicopter Society 58th Annual Forum Proceedings*, Montreal, Canada, June 11–13, 2002.

Finite Element Modeling of Fasteners under Static and Dynamic Loading

Kevin Behan, Emily Guzas, Jeffrey Milburn, and Stacy Moss
US Naval Undersea Warfare Center, Division Newport (NUWCDIVNPT), Newport, RI
{kevin.behan, emily.guzas, jeffrey.milburn, stacy.moss}@navy.mil

Abstract

The Naval Undersea Warfare Center has funded a project to investigate the accuracy of various fastener models used to represent actual shipboard bolted-connections within an analytical shock survivability assessment. The ultimate goal within this project is to develop finite element fastener representations that are not only computationally-efficient, but also accurate. A significant task within this effort involved the development of highly-detailed finite element models of bolted-connections under various load configurations. Accordingly, high-resolution fastener models were developed and incorporated into simulations of four bolted-connection test arrangements: static shear, static tension, dynamic shear, and dynamic tension. These simulation results are validated against experimental data from physical testing of each configuration. Future research will focus on exploring simplified finite element fastener representations and comparing these against the high-resolution results.

1. Introduction

The analytical assessment of shock survivability of naval systems and components typically involves developing complex, detailed finite element models of individual components, sub-assemblies, and the systems themselves. Components of these systems often include bolted-connections, which are necessarily incorporated into the finite element models. However, due to computational constraints, bolted-connections are often modeled with simplified fastener representations rather than high-resolution models. Thus, finite element-based shock survivability assessments become partially dependent on the accuracy of these fastener representations. A first-step towards improving the accuracy of simplified fastener representations involves calibration of numerical models of fasteners to experimental data of bolted-connections under a variety of loading configurations. To aid in this calibration, the current research effort focused on developing highly-detailed finite element models of fastener arrangements as tested in actual physical experiments, and comparing simulation results against experimental data from the fasteners tested under static and dynamic shear and tensile loading. The intent here is to provide a benchmark for future research on the evaluation of simplified finite element fastener representations.

Other work regarding finite element modeling of bolted-connections with validation through experiment has focused on single-lap, single-bolt composite joints (McCarthy, et al., 2005), and steel beam-column connections consisting of single plate, shear-tab connections that were fillet welded to steel columns and bolted to the shear tabs using high-strength bolts (Khandelwal, et al., 2008). In particular, Khandelwal, et al. (2008) and Khandelwal, El-Tawil, and Sadek (2009), described the development and calibration of connection macro-models (combinations of beam and discrete spring elements) capable of capturing the dynamic response behavior of various steel building frame connections, where these macro-models were then used to examine the collapse resistance of planar ten-story steel moment frames. A similar approach was applied to naval structures subjected to dynamic shock loading could prove extremely beneficial for future shock survivability assessments performed using finite element modeling techniques.

2. Experimental Background

In order to provide experimental data highlighting fastener behavior under a variety of load types, a series of experiments was carried out involving a fastener subjected to different static and dynamic shear and tensile loading configurations (Behan, et al., 2010). These tests formed the basis for the numerical modeling that was the focus of the current research, and data from these tests were used for validation of the numerical results presented later in this paper. The testing focused on bolts made of K-Monel K500, a corrosion-resistant nickel alloy commonly used in wetted naval applications. Fasteners in two lengths, 8.89 cm (3.5 in) for shear tests, and 15.24 cm (6 in) for tensile tests, were manufactured according to MIL-

S-1222G. These bolts consisted of K500 material with properties as defined by QQ-N-286. All fasteners had hex heads, were 6.35 mm (0.25 in) in diameter, and included a standard 1.9 cm (0.75 in) length of 20UNC-2A thread.

The mechanical properties of the K500 fastener material were determined via quasi-static tensile testing of K500 bolts in accordance with ASTM E8 and ASTM F606. The mechanical and physical properties of the K500 fastener material are given in Table 1. Figure 1 shows the average stress-strain curve for the K500 material as determined from the tensile test data. Due to some discrepancies in the test data, it was necessary to piece together the strain data and displacement data to develop a single stress-strain curve for the material. This aggregate curve was developed by using the strain data until just past the material yield (when the measured strain values started decreasing in a physically unrealistic manner), after which the displacement data were used to calculate the strains corresponding to the remainder of the stress data points.

Table 1. K500 physical properties (strength properties from tests by Thielsch Engineering, Inc. on 12 October 2010)

Density (g/cm³)	Elastic Modulus (GPa)	Poisson's Ratio	Yield Strength, 0.2% Offset (MPa)	Ultimate Strength (MPa)	Elongation in 4D (%)
8.435	178.2	0.32	729.5	1123.8	25.8

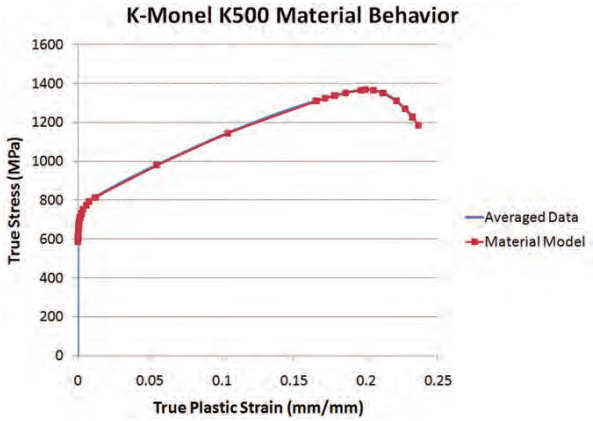


Figure 1. Constitutive behavior of K500 material

In the main series of static and dynamic fastener experiments, a test fastener in a typical bolted-connection was subjected to shear and tensile loading, as applied independently using a specially-designed test fixture. Hardware for the experiment included an A36 steel base fixture, various A36 steel-added mass and interface plates, high-strength steel guide rods, and the K500 shear and tensile fastener specimens. The fasteners were tested to failure under quasi-static loading on an Instron 4206 machine, and under dynamic loading on a MIL-S-901 Lightweight Shock Machine (LWSM).

For the static tensile tests, the tensile force was applied, by way of an adapter, to a tensile plate, as shown in Figure 2(a). The fastener connected the tensile plate to the base fixture which, in turn, was secured to the Instron machine. For the static shear tests, the test fixture was rotated 90 degrees from the static tension setup. The load was applied to the top center of the shear plate, which was free to move in the vertical direction along two lubricated guide rods, as per Figure 2(b).

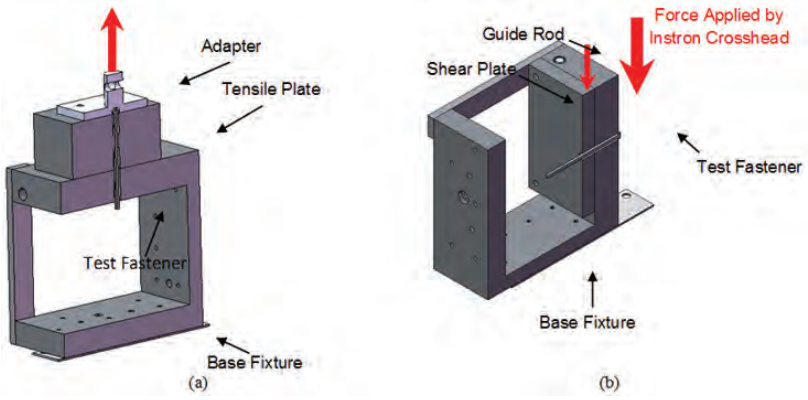


Figure 2. Section view of (a) static tension test; (b) static shear test

Two dynamic experimental test configurations were investigated on a Lightweight Shock Machine (LWSM); one focusing on tensile loading and the other on shear. For the dynamic tension tests, the bracket was mounted in the vertical plane of the LWSM interface plate so that the vertical hammer produced accelerations in the directions indicated in Figure 3(a). For the dynamic shear tests, the test fixture was rotated 90 degrees from the dynamic tension setup, and the test fixture was mounted in the vertical plane on the LWSM shelf plate so that the vertical hammer produced accelerations in the directions indicated in Figure 3(b).

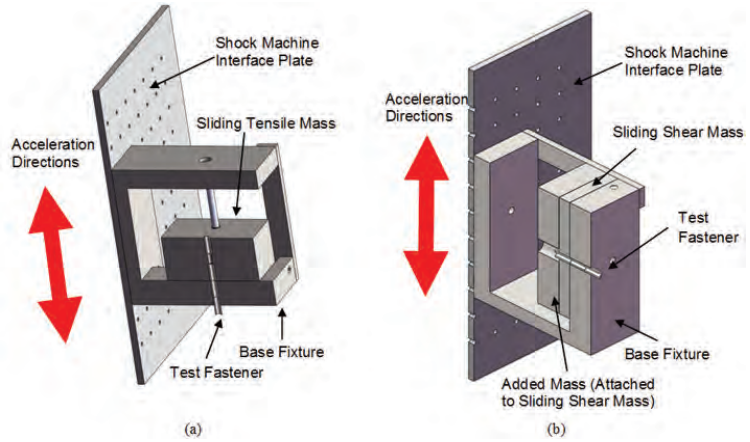


Figure 3. Section view of (a) dynamic tension test; (b) dynamic shear test

3. Numerical Modeling

Finite element models of the previously described test hardware were developed, using ABAQUS, version 6-9.1. All simulations were carried out in ABAQUS Standard and ABAQUS/Explicit using double-precision. C3D8 and C3D6 elements were used to model the shear and tensile fasteners and their associated fixtures. Due to potential problems with accuracy, C3D6 elements were used sparingly, only in regions where irregular geometry warranted their use (Selamet and Garlock, 2010). The numerical modeling focused on the subset of experiments in which the fastener specimens included a hollow bore, 3.9 mm (0.155 in) in diameter, for instrumentation, but which did not include the instrumentation itself. All of these tests included a static preload to 2/3 of the yield stress for K500.

Physical observations of the static tension experiments indicated that the fasteners all failed in the threaded region, and so care was taken to address the effect of the threads in these models. Based on the results from some preliminary studies, the threaded region was modeled using the correct profile for 20UNC-2A thread, but without the helical angle of the thread pattern—this approach captured the effect of the reduced area in the threads, without adding the complexity of producing a well-behaved mesh along a helically-varying thread profile. Additional preliminary analyses indicated that it was necessary to explicitly model the counter-bore at the top of the hollow bore of the tensile fastener in order to improve accuracy of results. Figures 4(a) and 4(b) show details of the tension test fastener mesh, which included 484,500 degrees-of-freedom. The fixture, tensile plate, nut, and washer were modeled using a total of 274,200 degrees-of-freedom.

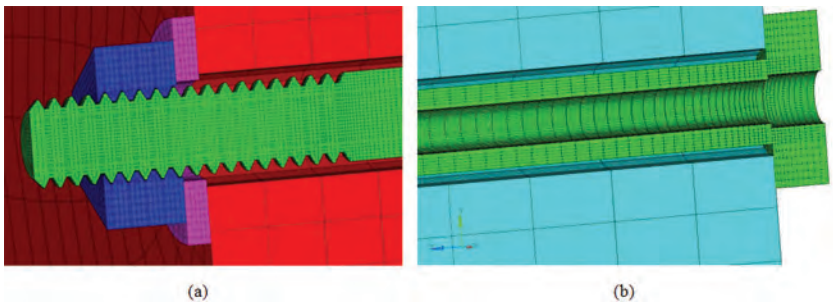


Figure 4. Section views of static tension finite element model, details of (a) threaded end; and (b) bore and counter-bore at head

In the static tension models, contact between the fixture and tensile plate, fixture and bolt, tensile plate and bolt, washer and fixture, washer and nut, and bolt threads and nut was modeled using finite sliding contact. Displacements were

incrementally applied to the top face of the tensile plate, and the load-deflection response was calculated using an arc length method. The static tension simulations were performed on four cores of a local high performance computing (HPC) cluster, and required approximately 5.1 hours of wall-clock time to fully trace the nonlinear equilibrium load-displacement path to a maximum displacement of 6.35 mm (0.25 in). Each core of the local HPC cluster was associated with a 2.67GHz Intel Xeon(R) processor and 32GB RAM.

For the static shear tests, the test fasteners were machined down 0.127 mm (0.005 in) after fabrication, producing a region 3.8 mm (0.15 in) long of reduced cross-sectional area in the vicinity of the shear plane, which was located at the interface between the sliding shear plate and the base fixture. As would be expected, in all of the experiments the test fasteners failed in the reduced-area region of the bolt near this interface. Figure 5 includes the mesh detail of the test fastener at the shear plane for the static shear test arrangement. Since the fasteners did not fail experimentally in the threads, the threads were not explicitly included in these models, but rather the threaded region was represented using a reduced cross-sectional diameter equal to the average pitch diameter of the threads over the threaded length of the bolt. The fastener was modeled with 319,500 degrees-of-freedom and the other components with 734,400 degrees-of-freedom.

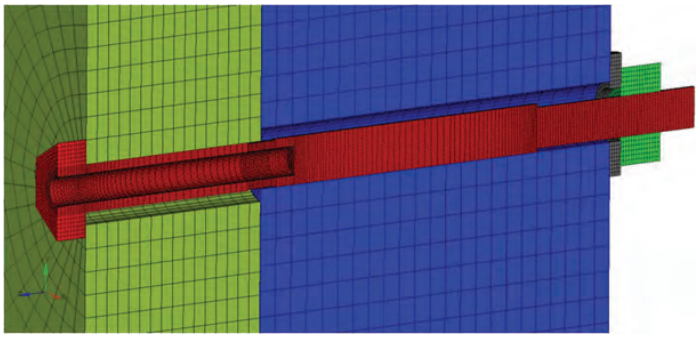


Figure 5. Section view of finite element model, static shear test

Finite sliding contact was included between the guide rods and shear plate, fixture and shear plate, shear plate and bolt, fixture and bolt, washer and fixture, and washer and nut. The nut and bolt were constrained to move together. Displacements were incrementally applied to the top and bottom faces of the shear plate, and the nonlinear load-deflection response was obtained using an arc length algorithm. As with the static tension analyses, the static shear simulations were carried out on four cores of a local HPC cluster. However, due to the amount of contact and complexity of failure, the static shear analyses required more run-time than the static tension simulations, using approximately 41.8 hours of wall-clock time to fully trace the nonlinear equilibrium path to a maximum displacement of 2.79 mm (0.11 in).

The simulations of the dynamic tension and shear tests were performed on the SGI Altix ICE (Diamond) HPC cluster at the US Army Engineer Research and Development Center (ERDC). The dynamic models were driven with velocity time histories in the three coordinate directions, where these time histories had been obtained from tri-axial accelerometer data recorded on the LWSM interface plate, which was connected to the base fixture, during the dynamic tension and shear experiments. Schematics of the models are shown in Figures 6(a) and 6(b).

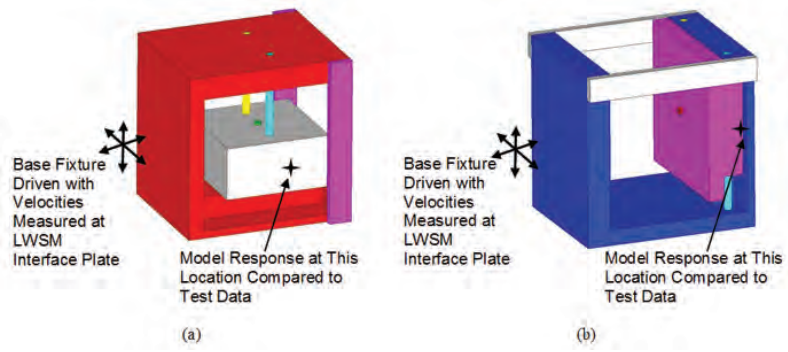


Figure 6. Schematic of the model for (a) dynamic tension; and (b) dynamic shear

Since the specimens of the K500 material were only tested under quasi-static conditions in order to characterize the behavior of the material, the rate-dependent behavior of the fastener material actually used in the physical experimental

tests was unknown. Thus, a dynamic increase factor of 30% was globally applied to the stress values of the K500 quasi-static stress-strain curve in order to estimate the K500 constitutive behavior at a strain rate of 1000/s.

For the dynamic tension models, the threaded region was again modeled using the correct profile for a 20UNC-2A thread, but without the helically-varying thread profile. Finite sliding contact was included between the guide rods and tensile sliding mass, fixture and sliding mass, fixture and bolt, sliding mass and bolt, washer and fixture, washer and nut, and bolt threads and nut. Contact between components was enforced using a penalty-based algorithm, as suggested by other research (Selamet and Garlock, 2010). The fastener was modeled with 439,000 degrees-of- freedom, and the fixture and other associated test components with 581,400 degrees-of-freedom. The dynamic tension simulations were performed on 32 cores of the ERDC Diamond cluster, and required about 72 hours of wall- clock time to produce 170 ms of response.

For the dynamic shear models, the threaded region was again represented using a circular cross-section with a reduced diameter equal to the average pitch diameter of the 20UNC-2A thread. Finite sliding contact, enforced by way of a penalty-based algorithm, was included between the guide rods and shear sliding mass, fixture and shear plate, shear plate and bolt, fixture and bolt, washer and fixture, and washer and nut. The nut and bolt were constrained to move together. The extra sliding mass, which was attached to the main sliding shear plate in the experiments, was modeled by way of lumped masses attached to the shear plate at each of the four bolted locations.

The fracture associated with the shear failure of the bolt was modeled using a strain-to-failure criterion (*SHEAR FAILURE), whereby elements were deleted from the model when shear strains reached a given level, prescribed herein as 0.24 mm/mm. As explained later in this paper, future work will focus on investigating more sophisticated damage criteria.

The fastener model included 292,600 degrees-of-freedom, and the fixture and other components consisted of 767,000 degrees-of-freedom. The dynamic shear simulations were performed on 32 cores of the ERDC Diamond cluster, and required about 72 hours of wall-clock time to produce 20 ms of response, considerably longer than the dynamic tension models, again because of the complexity of the contact and the material failure involved in this problem.

4. Results and Discussion

Both qualitative and quantitative comparisons were made between numerical models and the physical experiments for the different load configurations tested. For the static tests, the qualitative comparisons consisted of observations regarding the location and progression of failure in the fastener specimens, and the quantitative comparisons were made between load-displacement data as measured experimentally and numerically.

In the static tension tests, the fasteners experienced high levels of stress in the reduced-area region of the hollow bore (slight necking) and in the threads, where the bolts ultimately failed. These same regions of high-stress are also seen in the results of the finite element simulations, as shown in Figure 7. The stress contours here are given in units of pounds per square inch, spanning from 0 psi to 212,300 psi (1,464 MPa), in increments of 8,750 psi (60.3 MPa).

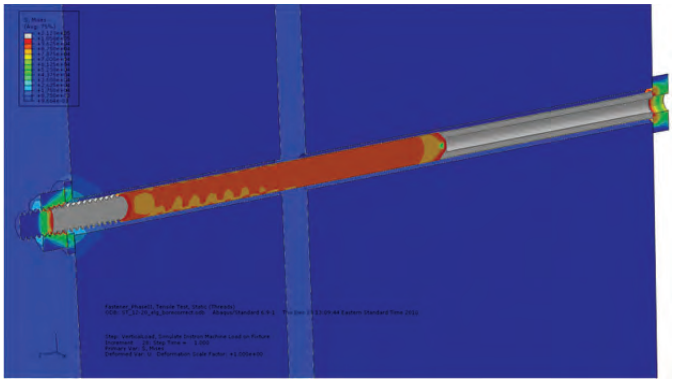


Figure 7. Contours of von Mises stresses at failure, static tension model

The load-displacement data from the static tension experiments compare extremely well to the corresponding load-displacement data from the finite element simulations, as shown in Figure 8. The experimental load levels and displacements were measured at the Instron machine crosshead, and were compared to the vertical nodal displacements applied to the nodes on the top face of the tensile plate and the corresponding forces developed there. The tensile plate was attached to the Instron crosshead by way of an adapter, as seen in Figure 2(a).

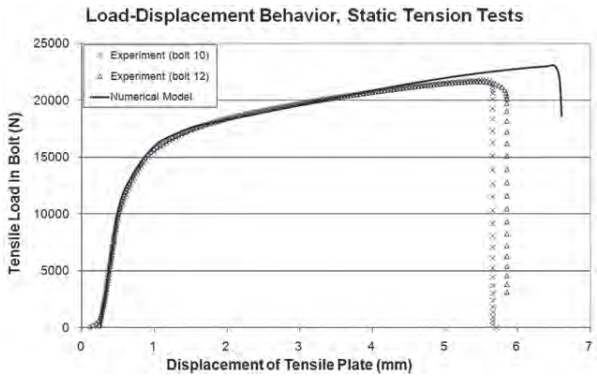


Figure 8. Comparison with experimental load-displacement data, static tension test

For the static shear experiments, the fasteners experienced the highest levels of stress near the shear interface, where the bolts ultimately failed. This is also observed in the numerical results, as seen in Figure 9. Again, the stress contours are shown in units of pounds per square inch, spanning from 0 psi to 296,900 psi (2,047 MPa), in increments of 8,750 psi (60.3 MPa). While the maximum stress level shown here exceeds the tested strength of K500, it is important to note that no damage criterion was used to model failure of the material in this model.

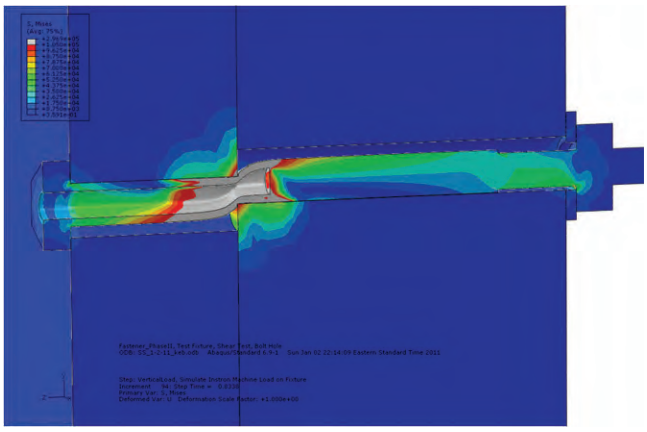


Figure 9. Contours of von Mises stresses at failure, static shear model

A comparison of the load-displacement data from the static shear experiments and the finite element simulation results is depicted in Figure 10. The experimental load levels and displacements at the Instron machine crosshead were compared to the vertical nodal displacements applied to the finite element nodes located at the top and bottom faces of the shear plate and corresponding forces developed at those locations. The Instron crosshead was in direct contact with the top of the shear plate, and a schematic of the setup is pictured in Figure 2(b).

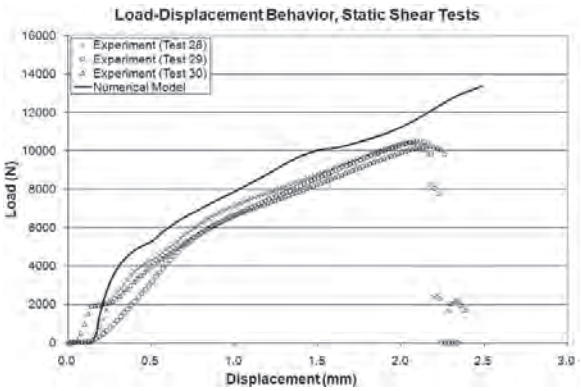


Figure 10. Comparison with experimental load-displacement data, static shear test

In the dynamic tension tests, the fasteners experienced the highest levels of deformation in the threads, where the bolts ultimately failed. This same trend in deformation is also seen in the finite element simulation results. Figure 11 depicts the contours of equivalent plastic strain at a time 14 ms into the analysis, where the contours span from 0 to 0.093 mm/mm in increments of 0.0104 mm/mm.

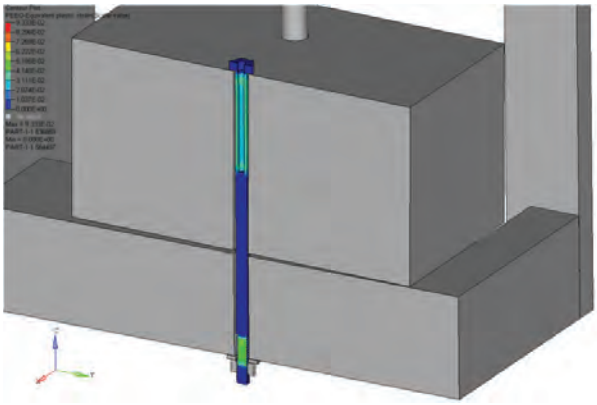


Figure 11. Contours of equivalent plastic strain, dynamic tension model

The velocity data that were experimentally recorded at the tensile sliding mass, at the location indicated in Figure 6(a), compare well with the corresponding data from the finite element simulations. The experimental and numerical results are plotted together in Figure 12.

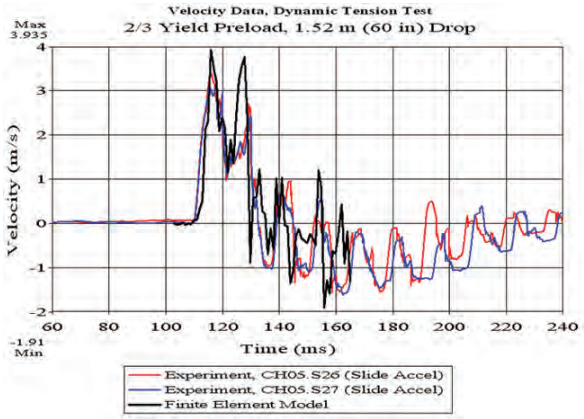


Figure 12. Comparison with experimental velocity data, dynamic tension test (starts at 100 ms)

In the dynamic shear tests, the fasteners experienced the highest amounts of plastic deformation near the interface between the shear sliding mass and the base fixture, eventually shearing apart here. The same concentration of fastener deformation near the shear plane is seen in the numerical results, as shown in Figure 13 at a time 10 ms into the analysis. The contours of equivalent plastic strain range from 0 to 0.097 mm/mm, and are shown in increments of 0.0107 mm/mm.

The velocity data experimentally recorded at the tensile sliding mass, as per Figure 6(b), is plotted together with the finite element simulation results in Figure 14. The discrepancy in peak velocity indicates that the finite element model under-predicts the strength of the fasteners for the dynamic shear configuration as modeled here. A number of factors could have contributed to this under-prediction of fastener strength, including problems with material failure prediction under dynamic loading, contact algorithms, or model loading.

Future work should be carried out to improve the accuracy of the detailed finite element representation of the dynamic shear test. The model employed for the above dynamic shear results uses a simple shear strain-to-failure criterion (*SHEAR FAILURE), whereby elements are eroded upon achieving this prescribed maximum strain level. However, more sophisticated damage models exist (e.g., *DAMAGE INITIATION /*DAMAGE EVOLUTION), and these should be investigated in order to improve model accuracy. Additionally, due to a lack of dynamic material data for K500, the authors chose to estimate the constitutive behavior at high strain rates by applying a dynamic increase factor of 30% to the stress values of the K500 quasi-static stress-strain curve. This approach is a rough estimate of dynamic material properties at best,

and although the method seems to have produced accurate numerical results for the dynamic tension case, the effect of rate-dependent material properties on both dynamic tension and dynamic shear tests should be investigated. Rate-dependent material behavior may have a small effect, or perhaps a significant effect, in these two dynamic load configurations, and it is important to investigate this further. Additionally, future work should focus on examining the contact algorithms used in this model, as well as ensuring that the loading used to drive the model is physically representative of the mechanism by which the dynamic loads from the Lightweight Shock Machine are actually transferred to the fastener test fixture.

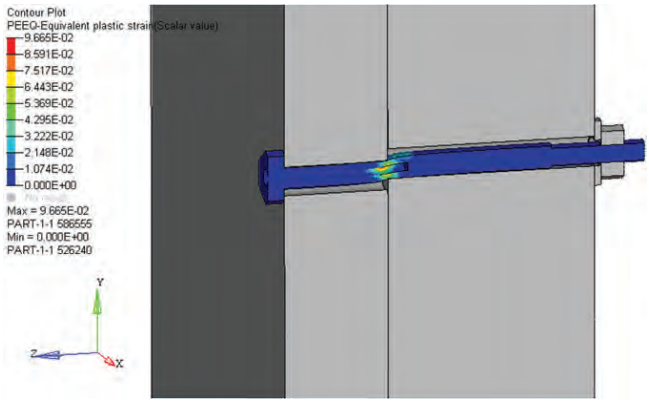


Figure 13. Contours of equivalent plastic strain, dynamic shear model

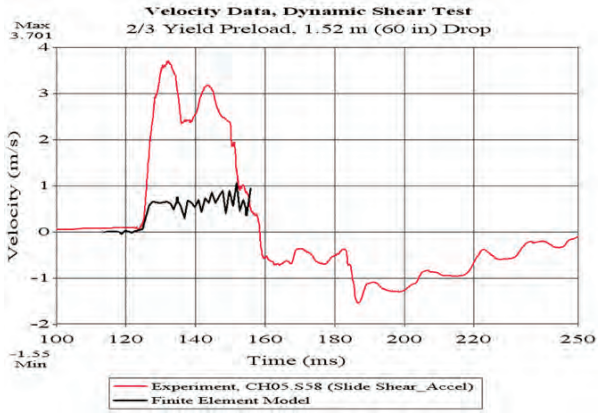


Figure 14. Comparison with experimental velocity data, dynamic shear test (starts at 115 ms)

5. Conclusion

In this paper, an approach is presented for high-resolution finite element modeling of K-Monel K500 fasteners in static and dynamic tension and shear configurations. Modeling approaches for each load type varied, with special attention taken to explicitly model the fastener threads for the tensile loading arrangements, a step that was not necessary for the shear tests. The best results for the static cases were achieved using displacement-based control, rather than load-based control, and finite sliding contact without automatic stabilization. Results for static tension, static shear, and dynamic tension simulations compare well with experimental data. However, further work is necessary in order to accurately capture the fastener behavior under dynamic shear loading.

Acknowledgments

This work was supported by internal basic and applied research funding (NUWCDIVNPT). The DoD HPC Modernization Program supported this project by supplying supercomputer time at the US Army Engineer Research and Development Center. We would like to acknowledge Pierre Corriveau, Tony Ruffa, Dawn Vaillancourt, and the entire Chief Technology Office at NUWCDIVNPT.

References

Behan, K.E, C. Pinero Pabon, E.L. Guzas, J.R. Milburn, and S.M. Moss, "Results of the Static and Dynamic Testing for the Improved Fastener Modeling: Phase II Testing", *NUWC-NPT-TM2010/100*, Nov. 2010 (UNCLASSIFIED).

Khandelwal, K., S. El-Tawil, S.K. Kunnath, and H.S. Lew, "Macromodel-Based Simulation of Progressive Collapse: Steel Frame Structures", *Journal of Structural Engineering*, 134, 7, pp. 1070–1078, 2008 (UNCLASSIFIED).

Khandelwal, K., S. El-Tawil, and F. Sadek, "Progressive collapse of seismically-designed steel-braced frames", *Journal of Constructional Steel Research*, 65, pp. 699–708, 2009 (UNCLASSIFIED).

McCarthy, M.A., C.T. McCarthy, V.P. Taylor, and W.F. Stanley, "Three-dimensional finite element analysis of single-bolt, single-lap composite bolted joints: part I—model development and validation", *Composite Structures*, 71, pp. 140–158, 2005 (UNCLASSIFIED).

Oberg, E., F.D. Jones, and H.L. Horton, *Machinery's Handbook*, 23rd ed., Industrial Press Inc., New York, 1988 (UNCLASSIFIED).

Selamet, S. and M.E. Garlock, "Guidelines for modeling three-dimensional structural connection models using finite element methods", *ECCS 2010 International Symposium 'Steel Structures: Culture and Sustainability*, Istanbul, Turkey, 2010 (UNCLASSIFIED).

Fluid-Structure Reaction Study of Transonic Flow Characteristics Associated with Limit Cycle Oscillations

Crystal L. Pasiliao
US Air Force Research Lab, Munitions Directorate, Flight Vehicles Integration Branch (AFRL/RWAV), Eglin AFB, FL
crystal.pasiliao@eglin.af.mil

Abstract

There exists a significant need for detailed understanding of the physical mechanisms involved in limit cycle oscillations (LCO) that can lead to a unified theory and analysis methodology. This work aims for a more thorough comprehension of the nature of the nonlinear aerodynamic effects for transonic LCO mechanisms, providing a significant building block in the understanding of the overall aeroelastic effects during the LCO. Examination of a true fluid-structure interaction (FSI) case (flexible structure coupled with computational fluid dynamics), is considered quasi-incrementally since this capability does not yet exist. We begin by performing fluid-structure reaction (FSR) simulations, examining the flow-field during rigid-body pitch-and-roll oscillations, simulating the torsional and bending nature of an LCO. Next, we execute FSR simulations during prescribed aeroelastic modes. Characterization of the flow-field is accomplished temporally via traditional flow visualization techniques, Lissajous, and wavelet analyses. Lissajous analyses provide key insight into the highly non-sinusoidal tracking of surface pressure with respect to aircraft motion. Additionally, wavelets are vital in identifying localized frequency differences at any point in time. Finally, once FSI codes are capable of accurately modeling a true LCO mechanism, these techniques will be crucial in identifying and characterizing the underlying physics that would be otherwise missed by frequency-domain-based techniques that are currently relied upon. Integration of fluid and structure, necessity of small time-steps, and complex weapon shapes drive up the complexity of the simulations and the required resources, extending the simulation times and post-processing requirements. The US Air Force requirement for numerous, simultaneous and quick-reaction solutions for a wide variety of stores and aircraft can only be accomplished through application of parallel high performance computing resources that meet the significant computational and memory demands associated with the certification computational environment.

1. Introduction

1.1 Limit Cycle Oscillation (LCO)

Aero-structural interaction (ASI) phenomena (i.e., flutter, limit cycle oscillations (LCOs), and buffet) are encountered on numerous flexible airframes. Of interest for this work, LCO is a sustained non-divergent periodic motion experienced by aircraft with configurations which often include stores.[1] The external stores consist of any object carried, such as a missile, bomb, fuel tank, engine, or pod. Safety and reliability are the primary concerns with regard to LCO. For example, while experiencing LCO, a pilot may have difficulty reading cockpit gauges and the heads-up display (HUD), and an unexpected LCO condition can lead to premature termination of the mission. There are also concerns regarding the effects of LCO on the store and Unmanned Air Vehicle (UAV) airframes. These issues include such things as potential degradation in reliability of components, whether or not stores can be safely released during LCO, the possible effects of LCO on target acquisition for smart weapons, and the effects of LCO on weapon accuracy for unguided weapons. Therefore, the prediction of LCO behavior of aircrafts is critical to mission safety and effectiveness, and the accurate determination of LCO onset speed and amplitude often requires flight-test verification of linear flutter models. The increasing stores certification demands for new stores and configurations, the associated flight test costs, and the accompanying risk to pilots and reliability of components all necessitate the development of tools to enhance LCO through modeling of the aerodynamic and structural sources of LCO nonlinearities.

Flutter, an instability caused by the aerodynamic forces coupling with the structural dynamics, and LCO are related phenomena. This association is due in part to the accuracy with which linear flutter models predict LCO frequencies and modal mechanisms.[2] However, since the characteristics of LCO motion are a result of unknown nonlinear effects, flutter models do not accurately predict LCO onset speed and amplitude. Current engineering knowledge and theories are not sufficient to provide an analytical means for direct prediction of LCO. The current approach relies heavily on historical experience and interpretation of traditional flutter analyses and flight tests, as they may correlate to the expected LCO characteristics for the configuration of concern.

1.2 Fluid-Structure Build-up Approach

There exists a significant need for a detailed understanding of the physical mechanisms involved in LCO that can lead to a unified theory and analysis methodology that is capable of predicting LCO responses. Many in the aeroelasticity community believe that LCO is indeed classical flutter at the onset, based on the strong similarity of the measured LCO deflection characteristics to linear flutter analysis computed deflections. However, the mechanism that bounds the oscillations still eludes the research community. Preliminary findings suggest that transonic aerodynamics is a potential bounding mechanism for typical LCO. Cunningham developed a semi-empirical method for LCO prediction based on his observation of "shock-induced trailing-edge separation" (SITES) during LCO wind tunnel testing.[2–4] His work established the foundation that LCO is predominantly an aerodynamic phenomenon. Once the relevant flow-field physics regarding the LCO ASI mechanism are understood, airframes can either be designed such that they avoid exciting the mechanism, or so that they capitalize on the mechanism and extract energy from the flow. However, since the characteristics of LCO motion are a result of unknown nonlinear effects, flutter models do not accurately predict LCO onset speed and amplitude. Current engineering knowledge and theories are not sufficient to provide an analytical means for direct prediction of LCO. The current approach relies heavily on historical experience and interpretation of traditional flutter analyses and flight tests as they may correlate to the expected LCO characteristics for the configuration of concern.

The present work aims to identify fundamental flow-field features, such as pressure gradients, flow separations, and shock interactions, which may be contributing to the bounding of the LCO mechanism, using computational fluid dynamics (CFD) analyses at known transonic LCO conditions observed in flight tests. Examination of a true fluid-structure interaction (FSI) LCO case (flexible structure coupled with CFD) is considered quasi-incrementally since this capability does not yet exist in the flutter community. The first step is to perform fluid-structure reaction (FSR) simulations, examining the flow-field via CFD analyses on the F-16 during prescribed rigid-body pitch-and-roll oscillations, simulating the torsional and bending nature of an LCO mechanism.[10] Three configurations have been examined so far: clean-wing aircraft without tip-missile launchers[5], clean-wing aircraft with tip launchers[6], wing-only with tip launchers[7], and aircraft with under-wing stores.[11] The next steps are to perform FSR simulations of the flow-field during prescribed aeroelastic modes, and then graduate to fully-coupled FSI simulations. Through this build-up fluid-structure approach, insight is to be gained into the characteristics of the flow-field during LCO conditions in order to assess any possible influences on the LCO mechanism. It is hypothesized that these flow features interact with the structure in a way that limits the magnitude of an LCO mechanism, effectively bounding the divergent flutter.

Additionally, temporal data analysis techniques will be applied to the resulting time-accurate CFD data.[8–9] Traditional flow visualization techniques will be utilized in order to capture the overall flow-field features of interest, such as surface pressure variations, Mach=1 boundary and vorticity iso-surfaces, and flow separation. Lissajous analysis will be applied to the data in order to isolate the phase relationships in regions of interest identified via the flow visualization. Wavelet analysis is also applied at these regions in order to accurately capture the nature of the frequency content of the mechanism.

1.3 Analysis Methodology

The chosen validation case-study chosen for this work is the F-16C fighter aircraft, tail number 88-0441, due to the available flight test data for comparison.[23] This aircraft is a Block 40 F-16 modified for flutter testing. The aircraft carries a symmetric load of: wing-tip launchers at stations 1 and 9; under-wing launchers, pylon-wing interfaces, and AIM-9P missiles at stations 2 and 8; pylon-wing interfaces and air-to-ground missiles at stations 3 and 7; pylon-wing interfaces and 370-gallon empty fuel tanks at stations 4 and 6. Flight conditions chosen for comparison are Mach numbers between 0.70 and 0.95 in increments of 0.05. Response amplitude and frequency data is available for accelerometers at stations 1 through 4. In level flight, measurable oscillations are present from 0.70 to 0.95 Mach. At the 10,000-, 5,000-, and 2,000-ft test altitudes, the LCO amplitude increases somewhat exponentially with Mach number from initial onset up to 0.90 Mach. Above 0.90 Mach at all altitudes, the LCO amplitudes show a trend of leveling off, and remained relatively constant. For elevated load factor flight, measurable oscillations are present from 0.75 to 0.95 Mach. At all three test altitudes,

the oscillation amplitudes increase almost linearly from 0.75 to 0.85 Mach, and show a significant increase in amplitude at 0.90 Mach. At 0.95 Mach for the 10,000- and 5,000-ft test altitudes, the LCO amplitude is lower than at 0.90 Mach. Elevated load factor flight is not investigated at 0.95 Mach, 2,000-ft. Overall, the dynamic aeroelastic characteristics of this configuration are well-behaved and can be categorized as typical LCO. The instability response is anti-symmetric and shows frequency variations from 7.9 to 8.2Hz, with the lowest frequencies seen at 0.90 Mach for all test altitudes.

For the aeroelastic simulations, one or more predicted flutter modes will be prescribed using Kestrel and AVUS. To determine the theoretical flutter-sensitive mode and its stability characteristics, linear flutter analyses are accomplished using a nominal worst-case aerodynamic condition with sea-level air density at 0.90 Mach number. The flutter analysis results for the typical LCO flight-tested configuration are presented in Figure 1 A) and B) and show a flutter speed of 493 knots true airspeed (KTAS) at 8.45Hz for the outboard wing torsion mode (Figure 1 C), which is coupled with the aft wing bending mode (bending mode (Figure 1 D). The damping curve for the flutter mode shows a moderately steep slope and crosses the 1% damping level at 540 KTAS. The flutter analyses indicate the flutter critical mode to be moderately sensitive to changes in airspeed, and thus show good correlation to the flight-test data for velocity sensitivity, but there is no consistent correlation to the rate of oscillation amplitude increase. The predicted flutter frequency shows good correlation to the flight-test frequency, which ranges from 7.9 to 8.2 Hz.

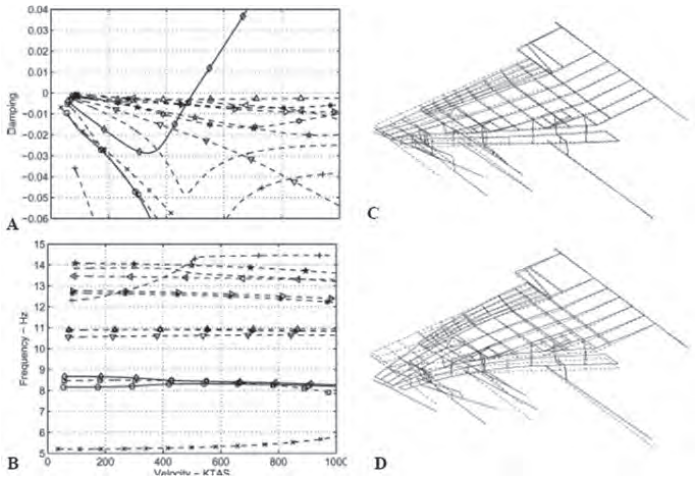


Figure 1. Flutter analyses at M_∞=0.9, sea level, V_f=493.6 KTAS, and f_f=8.45Hz: A) Damping vs. Velocity and B) Frequency vs. Velocity plots; C) Linear flutter mechanism unstable mode, anti-symmetric out-board wing torsion f_n=8.67Hz; and D) Linear flutter mechanism coupled mode, anti-symmetric aft wing bending f_n=8.16Hz, (right wing from full-span model)[23]

2. Analysis Tools

2.1 Flow Solvers

Rigid-body computations are performed using Cobalt, a commercial cell-centered, finite volume CFD code. It solves the unsteady, three-dimensional, compressible RANS equations on hybrid unstructured grids. It is fundamentally based on Godunov's[13] first-order-accurate, exact Riemann solver. Second-order spatial accuracy is obtained through a Least Squares Reconstruction. A Newton sub-iteration method is used in the solution of the system of equations to improve time accuracy of the point-implicit method. Arbitrary Lagrangian Eulerian (ALE) formulation is used to perform grid movement, where the grid is neither stationary nor follows the fluid motion but is reoriented without being deformed. The coordinate values change, but the relative positions between the grid points are unchanged. As a result, terms such as cell volume and face area remain constant and equal to the values in the original grid. The motion can include both translation and rotation. Complex motions can be defined by specifying arbitrary rotations and displacements of the grid in a motion file. This file then forms part of the required input deck. The hybrid RANS/Large-Eddy Simulation (LES) Delayed Detached-Eddy Simulation (DDES)[14] turbulence model based on the one-equation Spalart-Allmaras (SA)[15] is used for all calculations with and without Rotation and Curvature corrections (SARC).

Aeroelastic computations are performed using Kestrel and AERO suites. Kestrel is a cell-centered, finite volume CFD code which solves the unsteady, three-dimensional, compressible Reynolds-averaged Navier-Stokes (RANS) equations on hybrid unstructured grids of arbitrary cell topology. The flow solver used in Kestrel is Air Vehicles Unstructured Solver

(AVUS), formally known as $Cobalt_{60}$[12], an ancestor code of commercial Cobalt. AVUS is developed and maintained at the Air Force Research Laboratory at Wright-Patterson Air Force Base, OH. Arbitrary Lagrangian Eulerian (ALE) formulation is used to perform grid movement, where the grid is neither stationary nor follows the fluid motion but is reoriented without being deformed. The coordinate values change, but the relative positions between the grid points are unchanged. As a result, terms such as cell volume and face area remain constant and equal to the values in the original grid. The motion can include both translation and rotation. Complex motions can be defined by specifying arbitrary rotations and displacements of the grid in a motion file. This file then forms part of the required input deck. Adaptive mesh refinement (AMR) techniques are also being incorporated into Kestrel, and will be utilized as they become available. Kestrel is a research code being developed as part of the Department of Defense High Performance Computing Modernization (DoD HPCMP) Computational Research and Engineering Acquisition Tools and Environments (CREATE) Program[16,17]. The Kestrel software product is a modularized multidisciplinary fixed-wing virtual aircraft simulation tool incorporating aerodynamics, structural dynamics, kinematics, and kinetics. The first version of Kestrel is targeted to subsonic, transonic, and supersonic flight conditions with three major capabilities of a single static grid simulation, a single-grid rigid-body motion simulation, and a deforming single-grid aeroelastic simulation.[16] This initial capability couples a modal structural model with the core flow solver component, resulting in an ability to perform time-marching simulations of elastic bodies. This basic capability serves as an incremental build-up to full elastic aircraft analyses, which will be included in Kestrel v2.0. The structural model component includes the necessary fluid-structure interaction capabilities to properly transfer computed surface loads down to the underlying structural nodes, as well as determine an updated surface mesh due to the structural response. A separate mesh deformation component is tasked with deforming the fluid mesh based on the updated surface mesh position.[19] Morton[18] has shown that aeroelastic Kestrel results of the AGARD 445.6 wing have compared favorably with experiment.[20] Due to the limited FSI flutter cases available for validation, these results provide a confidence metric for the Kestrel code's ability to capture the relevant flow-field physics for aeroelastic phenomena.

The AERO suite of codes consists of AERO-F for the fluids simulations and AERO-S for structural simulations.[21] AERO-F is a three-dimensional, domain decomposition-based Euler/Navier-Stokes solver whose development started at the Center for Aerospace Structures at the University of Colorado at Boulder under the name AERO-F3D to address high-performance compressible flow, fluid-structure interaction, and aeroelastic computations. AERO-F operates on unstructured meshes that can combine tetrahedra, prisms, pyramids, and hexahedra. These meshes can move and/or deform in a prescribed manner (forced- body oscillations), or be driven via interaction with the structural solver, AERO-S. For this reason, the governing equations are formulated in AERO-F in an arbitrary Lagrangian Eulerian (ALE) setting. Large mesh motions are handled by a co-rotational approach which separates rigid and deformational components of the identified/specified motion of the surface of the obstacle. For turbulent flow computations, it offers one- and two-equation turbulence models, static and dynamic LES and Variational Multi-Scale LES (VMS-LES), as well as Detached-Eddy Simulation (DES) methods, with or without a wall function. The spatial discretization adopted by AERO-F blends the finite volume and finite element methods. More specifically, this semi-discretization combines a second-order-accurate Roe or HLLE upwind scheme for the convective fluxes, and a Galerkin centered approximation of the viscous fluxes. It can achieve fifth-order spatial dissipation error and sixth-order spatial dispersion error; and therefore, fifth-order spatial accuracy. Time integration in AERO-F can be performed with first- and second-order implicit, and first-, second-, and fourth-order explicit algorithms that satisfy their respective discrete geometric conservation laws (DGCLs). AERO-S is a finite element modal and dynamic nonlinear capable solver. The AERO suite is also suitable for distributed computing on supercomputers utilizing thousands of processors. The AERO suite has also been exercised with the aeroelastic AGARD 445.6 wing and compared favorably.[22]

2.2 Computational Models

For the pitch oscillation cases examined, a half-span viscous, rigid, full-scale model of the F-16 is used for the tip-launcher-only case and a full-span model is used for the store-configuration case; both modeling the fore-body bump, diverter, ventral fin, fuselage gun port, and leading edge antennae. Grid8, the tip-launcher-only configuration, consists of 3.18 million points (9.27 million cells) and Grid6679, the SDB-store-configuration, consists of 7.16 million points (19.03 million cells). The engine duct is modeled and meshed up to the engine face for both cases. For the aeroelastic cases examined, a half/full-span viscous, rigid, full-scale model of the F-16 wing, modeling the leading-edge antennae and tip launcher, is used. This grid, referred to hereafter as Wing1, consists of 2.25 million points (5.87 million cells). The boundary conditions for all computations are symmetry, adiabatic solid wall for the surface of the aircraft and the engine duct, and modified Riemann invariants for the far-field boundaries. A source boundary condition based on Riemann invariants is used to create an inflow condition at the engine exhaust. A sink boundary condition is used at the engine face to model the

corrected engine mass flow. The finite element model used for this is comprised of a dynamically representative model of the structure with 8,532 degrees-of-freedom representing the aircraft structure, under-wing stores, pylons, and launchers.

3. Preliminary Results

3.1 Flow Visualization

We see in previous analyses[8] that the pitching motion of the wing, emulating the torsional component of LCO, is the dominant mechanism for an LCO. Therefore, the FSR flow-field break-down is extended by performing forced, rigid-body pitch oscillation for a symmetric F-16 Grid6679 Small Diameter Bomb (SDB) store configuration with empty tip launchers, under-wing AIM-9 missiles, a single SDB on BRU-61 at each under-wing weapon station, and fuel tanks. This same grid and configuration have been previously examined by Dean.[24] This typical LCO configuration is highly sensitive to LCO, resulting in terminate level LCO at and above Mach=0.9, as seen in Figure 2. An 8Hz ±0.5° rigid-body pitch oscillation is examined at trim angle-of-attack (AOA_i) of 1.4° with the DDES-SA turbulence model at the same flow conditions (Mach=0.9, 5,000 feet) for comparison to the tip-launcher Grid8 case (trim AOA_i=1.34° and DDES-SARC turbulence model) previously examined.[10]

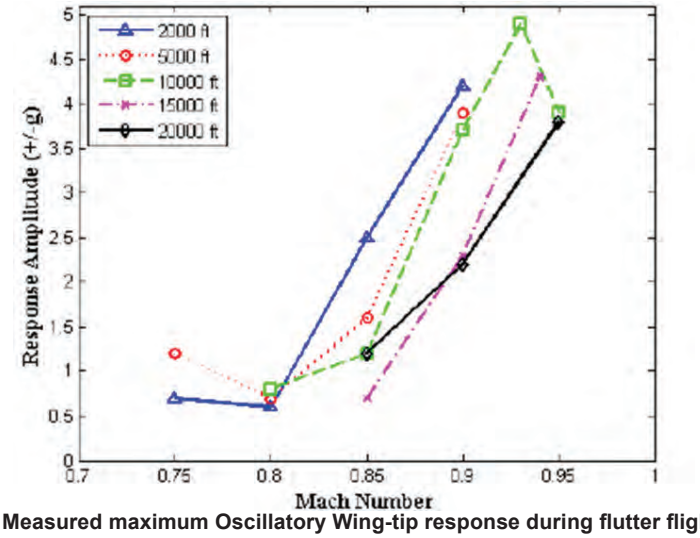


Figure 2. Measured maximum Oscillatory Wing-tip response during flutter flight testing

Figures 3 and 4 illustrate a sequence of images visualizing the flow computed for 8Hz±0.5° sinusoidal pitching motion depicting instantaneous Mach=1 boundary and vorticity magnitude iso-surfaces colored by pressure, respectively, at the upward stroke (between pitch angles of 135° and 180°) of the sinusoidal pitching cycle. The left set of images, display the tip-launcher-only Grid8 case A (upper) and B (lower) surface results, and the SDB-store-configuration Grid6679 comparisons are presented on the right for the C (upper) and D (lower) surfaces. The angle-of-attack is shown as a function of time in the top left-hand corner of the figures. Additionally, BL164 (91% span location) is indicated along the leading-edge (LE) of each of the images for reference purposes.

It can be seen from Figures 3 and 4 that the addition of the under-wing stores significantly influences the nature of the Mach=1 boundary and vorticity magnitude iso-surfaces. It is observed from animation of the Mach=1 iso-surface shown in Figure 3, that the trends between the tip-launcher-only and SDB-store-configuration cases are similar. For example, the iso-surfaces are at their maximum sizes at the top of the oscillation, and the aft extent of the upper-surface Mach=1 iso-surface maintains a mostly constant position on the aft portion of the wing, parallel to the trailing-edge (TE). Differences are also seen in Figure 3, as expected due to the presence of the under-wing stores, such as the extension of the forward upper surface shock to the LE (as though the presence of the stores sucks the shock forward) and the existence of shocks on the lower surface encompassing the stores. The upper surface profiles of the shocks are also different for the SDB case, manifesting in a much larger/stronger shock. Variations also emerge upon animation of the Mach=1 iso-surface aft of the shock on the inboard portion of the upper wing surface, where apparent separation of the shock is evident. For both the SDB-store-configuration and tip-launcher-only cases, this separation appears to occur in the vicinity of the strake-vortex. For the tip-launcher-only case, the separation is more significant and also happens near the wing-tip.

Upon animation of Figure 3C and D, the SDB case reveals that as the aircraft pitches nose-down, the Mach=1 iso-surface wraps around the wingtip significantly at the LE, extending about one-third the length of the wing in the chord-wise direction. This surface then engulfs the stores along the bottom surface of the wing, maintaining a fairly constant position with respect to the TE, as was seen on the upper surface. Upon animation of the tip-launcher-only case in Figure 3A and B, it is shown that as the aircraft pitches nose-down, the Mach=1 iso-surface is inhibited from wrapping around the wingtip at the LE by the tip launcher, but wraps around the tip-launcher about half-way down the tip in the chord-wise direction.

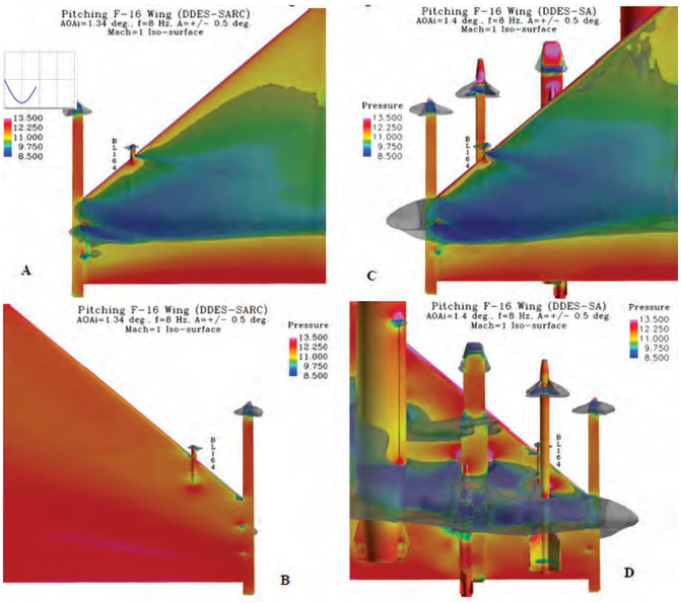


Figure 3. F-16, colored by pressure, 8Hz sinusoidal pitch motion with instantaneous Mach=1 boundary iso-surface: Grid8 half-span, tip-launcher Case A) Upper and B) Lower Surfaces; and Grid6679 full-span, SDB Case C) Upper and D) Lower Surfaces

The most obvious influence of the under-wing stores on the shock iso-surface is revealed upon animation of Figure 3, near the SDB where there is "bursting" of the shock, which is evidence of massive separation coming off of the pylon and store. There is an oscillatory shock feature on the under-wing AIM-9 missile forward of, but attached to, the primary shock where the shock begins to wrap around the missile during the pitch-down cycle, and recede toward the SDB during the pitch-up portion of the cycle. It appears that the presence of the forward missile fins on the AIM-9 create vortices that "punch holes" in the Mach iso-surface. We also see a significant ballooning of the secondary shock surface on the under-wing weapon pylon that grows with pitch-down motion, attaching to the fuel tanks, and recedes with the pitch-up motion.

From animation of the vorticity iso-surfaces for both configurations seen in Figure 4, vortices are seen rotating along the tip-launcher, correspondingly to the pitching up and down of the aircraft. Rotation in the LE antenna vortex is also apparent as the aircraft is pitching up and down. The tip and LE antenna vortices share the same direction of rotation on the upper surface, but the motion is contra on the lower surface for the tip-launcher-only case. The LE antenna vortex diminishes on the upper surface as the aircraft pitches down, and grows as it pitches up. Additionally, there is significant activity from the strake vortex on the inboard portion of the wing. This feature may be contributing to the inboard separation of the shock seen previously in animation of the Mach=1 iso-surfaces. From animation of the lower wing surface for the SDB case in Figure 4D, vortices are seen rotating along the AIM-9 and weapon pylon, corresponding to the pitching up and down of the aircraft, and share the same rotational direction as the tip vortex. These vortices grow as the aircraft pitches down, and recede when it pitches up. Additionally, the vortices originating on the forward fins of the AIM-9 trail all the way to the aft fins and interact with the separate vortices originating on the aft fins. Alternatively, the vortices on and near the SDB are much more unsteady in nature; their motion does not track with the pitching of the wing, and they maintain a more constant size. There are also regions of vorticity manifesting along the shock transition regions, which may be evidence of shock-induced separation, and could play an important role in the LCO mechanism.

The instantaneous C_p measurements plotted against non-dimensional chord are plotted in Figure 5 at 91% span on the left wing (looking aft-forward) for one developed cycle of oscillation. The 91% span location (Butt Line, BL, 164) is chosen as the region of interest due to its vicinity to the under-wing missile launcher location. Results for the Grid6679 SDB case are illustrated in: C (upper) and D (lower) surfaces; and the comparison Grid8 tip-launcher case results are shown

in: A (upper) and B (lower) surfaces. Each numbered line (1–9) corresponds to the time-accurate oscillation cycle angle, in degrees, during the sinusoidal pitching cycle, indicated by the corresponding cycle angle color given in the legend in the upper right-hand corner.

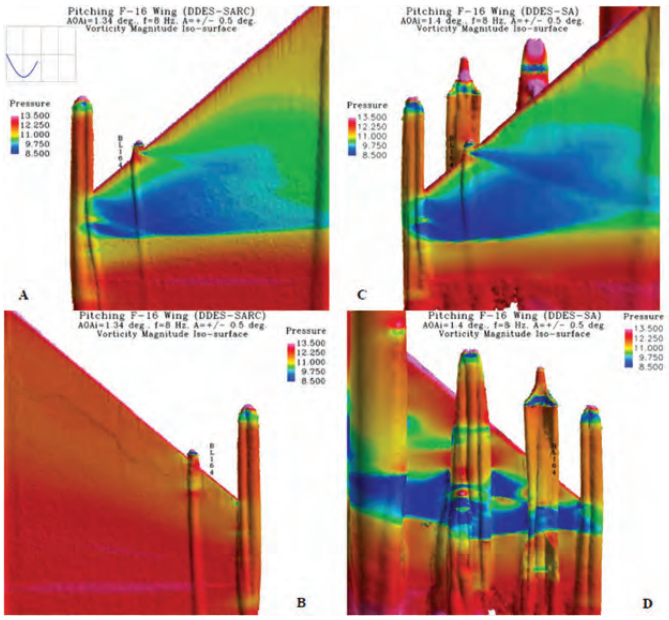


Figure 4. F-16 in 8Hz sinusoidal pitching motion with instantaneous vorticity magnitude iso-surface, colored by Pressure: Grid8 half-span, tip-launcher A) Upper and B) Lower Surfaces; and Grid6679 full-span, SDB Case C) Upper and D) Lower Surfaces

For the Grid 8 tip-launcher-only case in Figure 5A and B, the C_p on both the upper and lower surface overlap for 0° and 360° cycle angles, but do not for 180°. From a quasi-static perspective, it can be expected that the coefficients would be the same for all of these cycle angles since they are at the same absolute AOA. However, for the SDB case in 5C, the 0° and 360° cycle angles overlap on the upper surface until they split at about 72% chord and 0° pairs with 45° and 360° pairs with 315°. These pairings indicate the inability of the flow-field to "keep up" with the motion of the structure, indicating a lag in the flow. Both cases show the same suction loss feature for cycle angles 225°, 270°, and 315°, corresponding to the ascension, peak, and descent of the positive peak of the pitch oscillation, in the 65–75% chord region on the upper surface. However, on the upper surface, there is an aft shift in the shock location and an overall larger magnitude of the shock for the SDB case due to the presence of the stores. Upon examining Figure 5, and animating the C_p for all iterations on the upper surface, a rapid suction build-up is revealed at the 50–70% chord location for increasing AOA. As AOA decreases, there is a rapid loss of suction which progresses from the aft portion of the wing, then forward. The motion of this loss resembles that of a double-hinged door slamming shut; the first "hinge" near 70% chord and the second near 60% chord. This is an indication of the hysteresis behavior in the shock structure. Significant pressure differences occur on the lower surface for both cases in the 0–10% chord region, with the lower surface having more suction than the upper surface, and resulting in a loss of lift. For the tip-launcher-only case, this is indicative of the aerodynamic influence of the LE antenna. For the SDB case, the presence of the under-wing missile and associated hardware seem to decrease the suction in this region, likely due to the under-wing shock. For the SDB case in Figure 5D, we see evidence of the under-wing shock in the 10–50% chord region caused by the presence of the stores. This results in another region on the wing between 25–45% chord where there is a loss of lift due to the increase in suction on the lower surface. In the 70% chord area, the upper and lower surface pressures are equal during the upward stroke of the pitch cycle.

3.2 Lissajous Analysis

Figure 6A and B show Lissajous plots ($-C_p$ vs. local displacement from the surface of the airfoil at a chord vs. span location) for one cycle of pitching 8Hz ±0.5° oscillation. A low-pass filter is applied to the data at 100Hz to filter out the noise due to turbulence effects. The red circles indicate the starting point of the pitching cycle, and the black arrows indicate the direction of rotation of the Lissajous. Figure 6A is a Lissajous figure for the Grid8 tip-launcher case on the upper surface at 91% span vs. 65% chord location, which corresponds to the shock recovery region as seen in Figure 5A.

From A we see that in this region, a figure-eight shape forms due to the continuous phase variation caused by the oscillatory behavior in the shock transition region. The figure-eight plot is not a harmonic, as is normally associated with figure-eights, but the result of the phase changing sign; from a lead to a lag, or vice-versa. The shape of the figure-eight provides insight into phase/time lag. A fat top or bottom shows where C_p is lagging behind the motion; alternatively, a narrow top or bottom indicates where the C_p is leading the motion, trying to catch up. The transition across the shock is indicated by the change in lift (i.e., the change in value of C_p shown by position of starting red circle). An increase in $-C_p$ is ahead of shock, whereas decrease in $-C_p$ is behind shock. Figure 6B is a Lissajous figure for the Grid6679 SDB case on the upper surface at 91% span vs. 70% chord location, which corresponds to the shock recovery region as seen in Figure 5C. In this region, we again see a figure-eight shape in B forming due to the continuous phase variation caused by the oscillatory behavior in the shock transition region. However, this Lissajous is not as smooth and symmetric as the tip-launcher-only case.

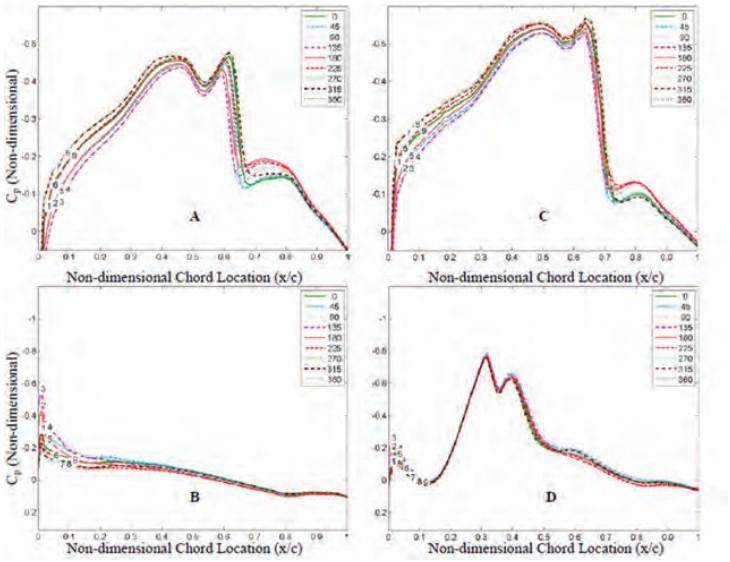


Figure 5. Instantaneous Cp measurements at 91% span for 8Hz pitch Grid8 half-span, tip-launcher Case A) Upper and B) Lower Surfaces and Grid6679 full-span, SDB Case C) Upper and D) Lower Surfaces: Lines 1–9 correspond to pitch cycle angle (°)

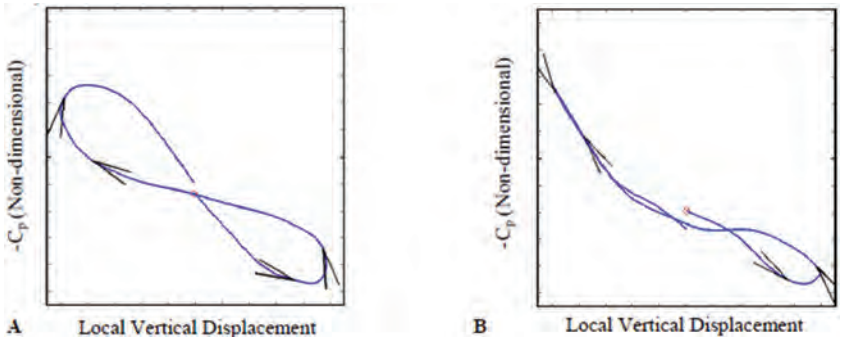


Figure 6. Lissajous of 8Hz pitch oscillation for: A) Grid8 tip-launcher case on upper surface at 91% span, 65% chord; B) Grid6679 SDB case on upper surface at 91% span, 70% chord

3.3 Wavelet Analysis

Wavelet analysis is applied to the same points of interest on the wing from Figure 6 and shown correspondingly for the upper surfaces in Figure 7 and 8, in order to gain further insight into the temporal nature of the results. In each set of figures, A gives the time history of the C_p variation, B shows the fast Fourier transform (FFT) of the data, and C illustrates the wavelet transform view of frequency vs. time.

Figure 7 contains the 8Hz Grid8 tip-launcher-only results on the upper surface at 91% span vs. 65% chord which corresponds to the shock recovery region as seen in Figure 5A. The time history plot in Figure 7A reveals a saw-tooth

shaped response in upper wing surface C_p. The FFT plot in B shows a prominent peak at the input frequency of 8Hz with smaller peaks occurring at harmonics, and assumes a periodic response. The wavelet plot in C reveals non-periodic features above the input frequency that are not seen in the FFT, such as higher frequency energy concentrations on the end of the cycle occurring when the C_p drops dramatically, such as at time=0.25 sec and 0.38 sec. These are non-symmetric responses compared to the times when the C_p is building at time=0.17, 0.3, and 0.43 sec.

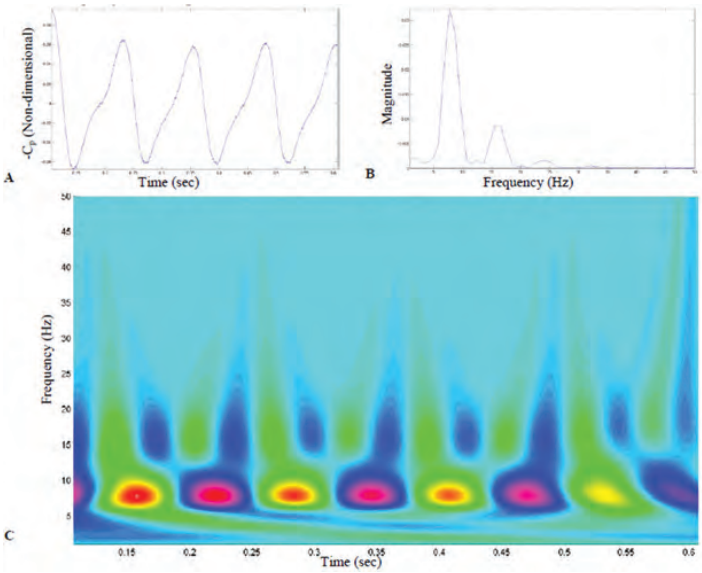


Figure 7. Temporal analysis of upper surface C_p for Grid8 8Hz pitch oscillation at 91% span, 65% chord: A) time history, B) fast Fourier transform, and C) wavelet frequency vs. time plots

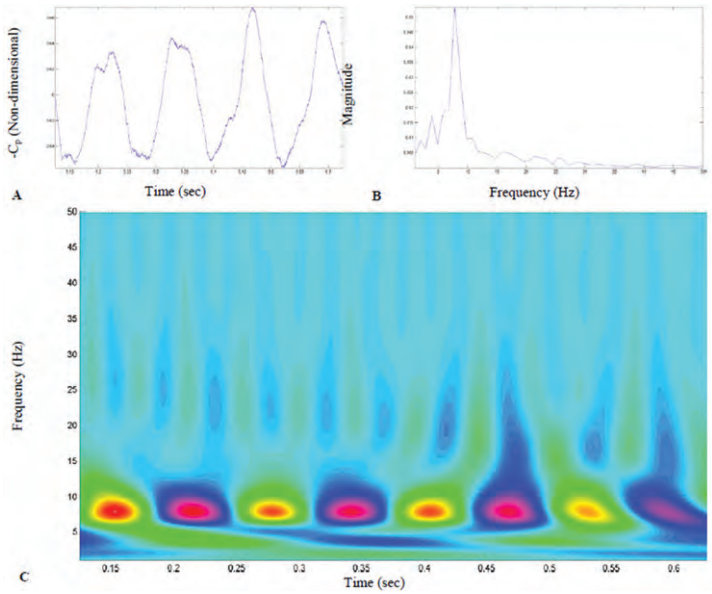


Figure 8. Temporal analysis of upper surface C_p for Grid6679 8Hz pitch oscillation at 91% span, 70% chord: A) time history, B) fast Fourier transform, and C) wavelet frequency vs. time plots

Figure 8 contains the 8Hz Grid6679 SDB results on the upper surface at 91% span vs. 70% chord and which corresponds to the shock recovery region as seen in Figure 5C. The time history plot in Figure 8 A) reveals a wide double-peak response in upper wing surface C_p for the first couple of cycles, and then the saw-tooth shape appears to come in for the next two cycles. The FFT plot in B) shows a prominent peak at the input frequency of 8Hz with much smaller peaks occurring at harmonics, and assumes a periodic response. The wavelet plot in C reveals non-periodic features above the input frequency that are not seen in the FFT. The blue regions at time=0.19, 0.24, 0.32, and 0.36 sec correspond to the double-peak regions

and consist of a range of frequencies. We also see similar distributions for the trough region at time=0.27 and 0.42 sec, consisting of a range of frequencies. The energy concentration then changes at time=0.47 sec with a one, clear frequency contributing.

4. Conclusion

The fundamental goal of this work is to gain a more thorough comprehension of the nature of the nonlinear aerodynamic effects for transonic limit cycle oscillation (LCO) mechanisms, providing a significant building block in the understanding of the overall aeroelastic effects during the LCO. The main benefits of successfully understanding the fundamental physics driving the mechanism are: 1) early discovery of complex aerodynamic phenomena that are typically only discovered in flight-testing, and 2) rapid clearance of aircraft envelopes with new store combinations when aeroelastic phenomena are not predicted. The chosen computational approach for solving these complex aircraft/weapons configurations is an unstructured grid, Navier-Stokes CFD solver with RANS and DES turbulence treatments, finite element or modal structural models, and computational grid deformation strategies. This high-fidelity approach will provide the ability to "virtually fly" missions before actual tests, as well as, provide a "measuring stick" for developing faster, lower fidelity approaches suitable for screening hundreds of cases before analyzing the few most critical cases.

Preliminary unsteady fluid-structure reaction (FSR) CFD solutions of a viscous, rigid, full-scale F-16 SDB configuration were computed and analyzed for prescribed rigid-body pitch oscillations, emulating the torsional component of an LCO motion. Interesting flow features were seen in the flow upon examining the surface pressure coefficient, C_p, Mach=1 iso-surface, vorticity magnitude iso-surface, Lissajous, and wavelet plots; motivating further examination of a true aeroelastic motion. It is believed that these flow-field features should develop and interact differently for a true aeroelastic motion, leading to varying pressure distributions, shock locations and strengths, and vortical structures on the wing; and resulting in added time and flow inertia lags. Features such as these could be participating in bounding the LCO mechanism. The rigid-body results also showed that temporal analysis techniques are critical in properly characterizing the nature of LCO flow-field physics. Lissajous figures provide key insight into the highly non-sinusoidal tracking of the C_p with respect to aircraft motion. Additionally, the wavelet analysis is a key component in identifying the localized frequency differences at any point in time. Temporal analysis techniques such as these are vital in uncovering the non-periodic behavior in the response. Once the state of FSI codes is capable of accurately modeling a true LCO mechanism, these techniques will be crucial in identifying the underlying physics that would be otherwise missed by frequency-domain-based techniques that are currently relied upon.

Acknowledgments

This work was partially funded by the US Air Force Office of Scientific Research (AFOSR) (Douglas Smith, Program Manager). The DoD HPC Modernization Program supported this project by supplying supercomputer time under the Computing Challenge Project C4P. This time was made available at the DoD Supercomputing Resource Centers at the US Army Research Laboratory (harold), the US Army Engineer Research and Development Center (diamond and jade), and the Maui High Performance Computing Center (mana). We would like to thank the US Air Force Flight Test Center, Air Force SEEK EAGLE Office, and the CREATE-AV team for their support.

References

1. Norton, W.J., "Limit Cycle Oscillation and Flight Flutter Testing", *21st Annual Symposium Proceedings*, pp. 3.4-1–12, Lancaster CA, 1990, Society of Flight Test Engineers.

2. Bunton, R.W. and Denegri, C.M., "Limit Cycle Oscillation Characteristics of Fighter Aircraft", *Journal of Aircraft*, Vol. 37, No. 5, pp. 916–918, 2000.

3. Cunningham, Jr., A.M., "A Generic Nonlinear Aeroelastic Method with Semi-empirical Nonlinear Unsteady Aerodynamics-Volume 1: Technical Discussion", *Report AFRL-VA-WP-TR-1999-3014*, Air Force Research Laboratory, February 1999.

4. Cunningham, Jr., A.M., "The Role of Nonlinear Aerodynamics in Fluid-Structure Interaction", *AIAA Paper 98-2423*, June 1998.

5. Cunningham, Jr., A.M. and Meijer, J.J., "Semi-Empirical Unsteady Aerodynamics for Modeling Aircraft Limit Cycle Oscillations and Other Nonlinear Aeroelastic Problems", *Proceedings of the International Forum on Aeroelasticity and Structural Dynamics*, Royal Aeronautical Society, London, Vol. 2, pp. 74.1–74.14, 1995.

6. Pasiliao, C.L. and Dubben, J.A., "Analysis of CFD Pressure Coefficients on the F-16 Wing Associated with Limit Cycle Oscillations", *AIAA Paper 2008-0409*, January 2008.

7. Pasiliao, C.L. and Dubben, J.A., "Determination of Flow-Field Characteristics of F-16 Wing Associated with Limit Cycle Oscillations Using CFD", *Proceedings of the NATO RTO Symposium AVT-152 on Limit-Cycle Oscillations and other Amplitude-Limited,* Self-Excited Vibrations, Norway, May 2008.

8. Pasiliao, C.L. and Dubben, J.A., "CFD-Based Determination of Transonic Flow-Field Characteristics of F-16 Wing Associated with LCO", *AIAA-2009-1084*, January 2009.

9. Pasiliao, C.L., Temporal Analysis of Transonic Flow-Field Characteristics Associated with Limit Cycle Oscillations", *PhD thesis*, Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL, May 2009.

10. Pasiliao, C.L. and Dubben, J.A., "Fluid-Structure Reaction Study of Flow-Field Physics Associated with Limit Cycle Oscillations", *IFASD Paper 2009-136,* International Forum on Aeroelasticity and Structural Dynamics, Seattle, WA, June 2009.

11. Pasiliao, C.L. and Dubben, J.A., "Fluid-Structure Reaction Study Roadmap for an F-16 Wing Undergoing Prescribed Limit Cycle Oscillations", *Proceedings of the 2010 ITEA Aircraft/Stores Compatibility Symposium*, Fort Walton Beach, FL, April 2010.

12. Pasiliao, C.L. and Dubben, J.A., "Fluid-Structure Reaction Study of an F-16 Limit Cycle Oscillation Case with Stores", *AIAA 2010-2633*, 51st AIAA SDM Conference, Orlando, FL, April 2010.

13. Strang, W.Z., Tomaro, R.F., and Grismer, M.J., "The defining methods of Cobalt60: A parallel, implicit, unstructured Euler/Navier Stokes Flow solver", *AIAA Paper 1999-0786*, January 1999.

14. Godunov, S.K., "A finite difference method for the computation of discontinuous solutions of the equations of fluid dynamics", *Matematicheskii Sbornik*, (47), pp. 357–393, 1959.

15. Spalart, P.R., Jou, W.H., Strelets, M., and Allmaras, S.R., "Comments on the feasibility of LES for wings, and on a hybrid RANS/LES approach", *Proceedings of the First AFOSR International Conference on DNS/LES*, Greyden Press, August 1997.

16. Shur, M.L., Strelets, M.K., Travin, A.K., and Spalart, P.R., "Turbulence modeling in rotating and curved channels: Assessing the Spalart-Shur correction", *AIAA Journal*, 38(5), May 2000.

17. Morton, S.A., McDaniel, D.R., Sears, D.R., Tillman, B., and Tuckey, T.R., "Kestrel – a fixed-wing virtual aircraft product of the CREATE program", *AIAA Paper 2009-338*, January 2009.

18. Morton, S.A., McDaniel, D.R., Sears, D.R., Tillman, B., and Tuckey, T.R., "Kestrel v2 - 6DOF and control surface additions to a CREATE simulation tool", *AIAA Paper 2010-511*, January 2010.

19. Morton, S.A., McDaniel, D.R., Sears, D.R., Tillman, B., and Tuckey, T.R., "Rigid, maneuvering, and aeroelastic results for Kestrel - a CREATE simulation tool", *AIAA Paper 2010-1233*, January 2010.

20. McDaniel, D.R. and Morton, S.A., "Efficient mesh deformation for computational stability and control analyses on unstructured viscous meshes", *AIAA Paper 2009-1363*, January 2009.

21. Yates, C.E., "AGARD standard aeroelastic configurations for dynamic response. Candidate configuration I - Wing 445.6", *Technical report, NASA Technical Memorandum*, Hampton, VA, 1987.

22. Geuzaine, P., Brown, G., Harris, C. and Farhat, C., "Aeroelastic dynamic analysis of a full F-16 configuration for various flight conditions", *AIAA Journal*, 41:363-371, 2003.

23. Lesoinne, M. and Farhat, C., "Higher-order subiteration-free staggered algorithm for nonlinear transient aeroelastic problems", *AIAA Journal*, 36(9), pp. 1754–1756, 1998.

24. Denegri, Jr., C.M., Dubben, J.A., and Maxwell, D.L., "In-Flight Wing Deformation Characteristics During Limit Cycle Oscillations", *Journal of Aircraft*, Vol. 42, No. 2, pp. 500–508, 2005.

25. Dean, J.P., Clifton, J.D., Bodkin, D.J., Morton, S.A., and McDaniel, D.R., "Determining the Applicability and Effectiveness of Current CFD Methods in Store Certification Activities", *AIAA Paper 2010-1231*, 48th AIAA Aerospace Sciences Meeting, January, 2010.

The Role of Simulation-Based Design in Systems Engineering

Thomas P. Gielda
Caitin, Inc., Fremont, CA
tgielda@caitin.com

Abstract

Simulation-based design has become a key enabler to fully utilize the capabilities of systems engineering. This paper will present a historical perspective of how simulation was first used to: 1) evaluate complex physics and designs, 2) assess the impact of key design parameters on customer requirements, and 3) employ simulation to drive systems engineering from platform requirements to critical parameters.

This study will begin with utilizing full three-dimensional Navier-Stokes simulations of complete rocket systems. In these simulations, we ask the basic performance question: will the rocket fly in stable flight? We then look at simulation in the development of automotive systems; here we look at the integration of smart CAD models into the simulation process to allow for the rapid turnaround of design iterations. The last case study will look at how simulation has impacted the implementation of systems engineering in the white goods industry. In this study, we will explore how simulation has been used to perform virtual product development and examine how these tools better enable systems engineering. We will look at product functional performance as well as the requirements placed upon the design to survive the trip through the supply chain. We will comment on the value of bi-directional traceability of customer/system requirements to key design features.

1. Introduction

Simulation-based design is an outstanding methodology to develop deep understanding of how a platform (rocket, automobile, or washing machine) functions in all aspects of usage.

This methodology can provide detailed information of system, subsystem, and component interactions. Identifying these interactions early in the design process greatly increases the probability of success when launching a new product. Failure to identify these interactions can, at best, result in program delays; at worst they will be the cause of catastrophic failures.

Utilization of analysis tools that simulate the complete platform are critical. In this paper, we will present three case studies, which will include:

1. Utilization of computation fluid dynamics to predict the flight characteristics of a single-stage-to-orbit rocket configuration (Gielda, et al.[1])
2. Application of multidisciplinary Computer-Aided Engineering tools to develop an automobile that will adequately cool the interior for 1/5th the traditional power consumption (Gielda, et al.[2])
3. Application of simulation-based design to optimize the performance of a household refrigerator (Gielda, et al.[3])

Each test case will build in complexity and utilize increasingly complex physical models to understand system interactions.

2. Case Studies

Case Study 1: Single-Stage-to-Orbit Rocket

In 1991, McDonnell Douglas was awarded a research contract by the Strategic Defense Initiative Organization (SDIO) to develop a single-stage-to-orbit demonstrator. The DC-X vehicle was to rise vertically to a specified altitude, transition to horizontal flight and achieve the pitch-up to land vertical. This vehicle utilized computation fluid dynamics (CFD) to perform analysis of all aspects of the flight envelope. In addition, CFD was employed to down-select the vehicle architecture. In the early phase of the program there were two competing architectures:

1. A traditional conical body with either embedded or external bell nozzles. (Pratt and Whitney RL-10, Figure 1)
2. A plug nozzle configuration with an aero-spike propulsion system (Aerojet, Figure 2).

Figures 1 and 2 depict wireframes of the vehicle topologies.

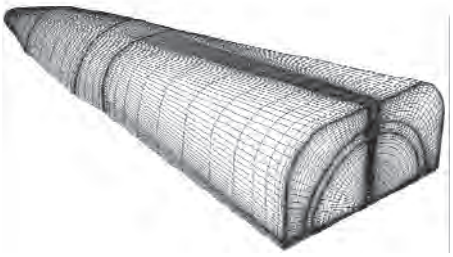


Figure 1. Traditional conical body with embedded bell nozzle rocket

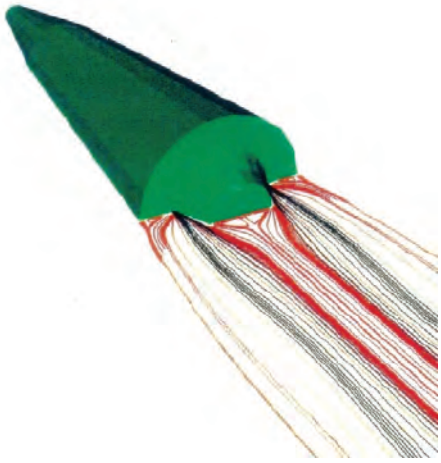


Figure 2. Conical body with aero-spike engine

A zonal three-dimensional (3D), finite-rate chemically-reacting CFD code was utilized to simulate the flight loads of the vehicle on its trajectory. Figure 3 depicts the Mach contours around the DC-X during the pitch-up maneuver. The angles-of-attack shown are approximately 2, 45, 90, and 1,800 degrees. The vehicle decelerates from mach 0.5 to 0.3 during the maneuver.

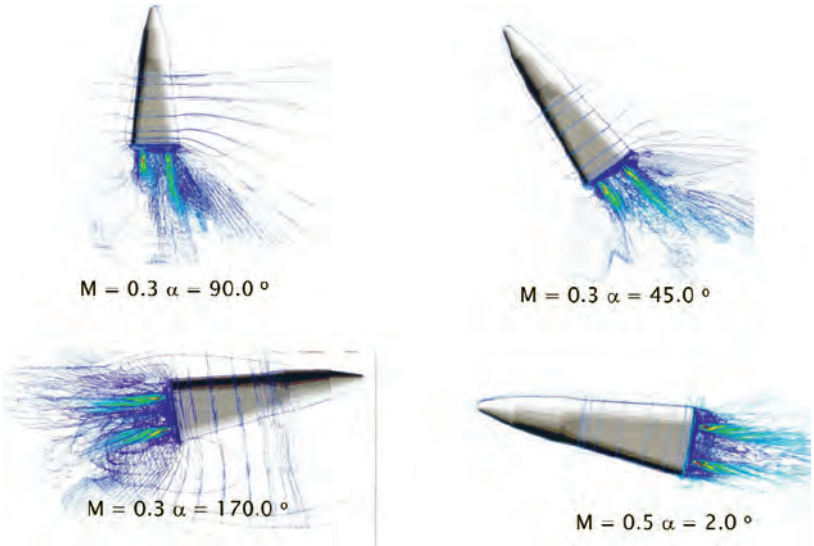


Figure 3. Mach contours about flight vehicle at various angles-of-attack during a pitch-up maneuver

Figure 4 depicts the streamline patterns off the propulsion stream for two different plug nozzle configurations.

Figure 4. Plug nozzle plume studies, 0.0 degree angle-of-attack, free stream Mach number 0.32

The results of the above simulations were utilized for more than generating colored fluid dynamics pictures. Since this vehicle was the test article, the normal, axial, and moment coefficients were predicted using CFD. Furthermore the reaction control jet system of the vehicle was sized based on CFD results. A failure in adequately predicting these loads would result in uncontrolled tumbling when the vehicle pitched-up from horizontal to vertical flight.

Figure 5 depicts some of the force and base heat flux results that were developed in this study.

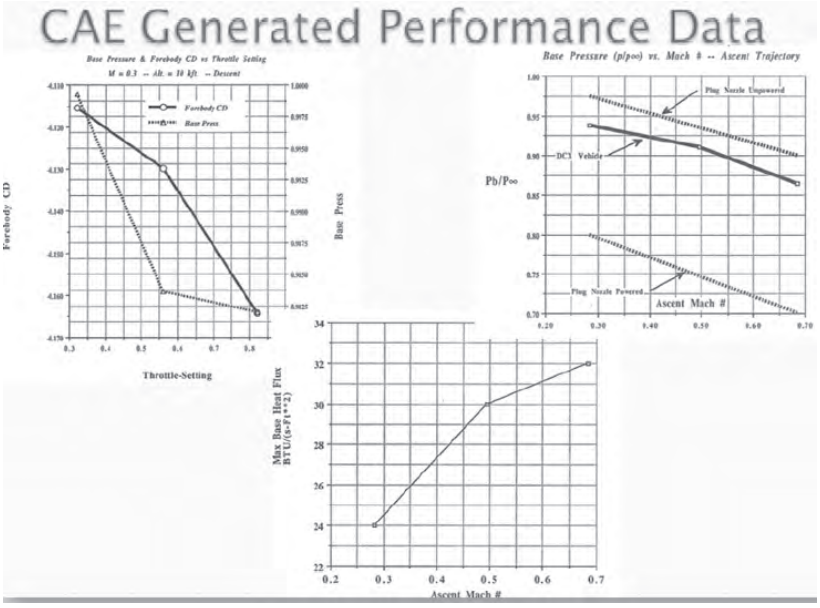


Figure 5. Drag, base pressure, and base heat flux for the DC-X flight vehicle

Results of the detailed analyses drove the vehicle architecture. The DC-X flight vehicle configuration is shown in Figure 1. The traditional engine selection was chosen due to the flight performance predictions. It is interesting to note that the comparison off the predicted force and moment coefficients and the flight telemetry data were within 15%.

Case Study 2: Energy-Efficient Vehicle

In 1999, Ford Motor Company developed an energy-efficient vehicle as part of the Partnership for the Next-Generation Vehicle (PNGV) program. The objective for this program was to develop a climate control system for the vehicle that ran on 1.0 kW. Note this represented an 80% reduction in power consumption over a traditional vapor compression system. In order to achieve these aggressive performance targets, Ford utilized its CAE tools to rapidly evaluate and optimize the vehicle thermal performance. The capabilities of these tools include:

1. Parametric geometry for the total vehicle
2. Automated mesh-generation for CFD meshing (All Tet meshing)
3. Time-accurate transient analysis, including radiation and conjugate heat transfer

4. Passenger thermal comfort model to predict how long it will take for a passenger to achieve thermal neutrality (Hosni, et al.[4])

In order to perform these types of analysis quickly it was essential to develop smart parametric models that allowed us to go from CAD to analysis seamlessly. In the past, it would require us 4–6 weeks to build an analytical model. Through the use of intelligent CAD and auto meshing, we could build a vehicle model in 2–3 days and then make subsequent changes in less than 24 hours. Our goal was to change a design and evaluate that change on performance in less than 24 hours. Figure 6 depicts the parametric model geometry of the vehicle exterior, engine compartment, and interior.

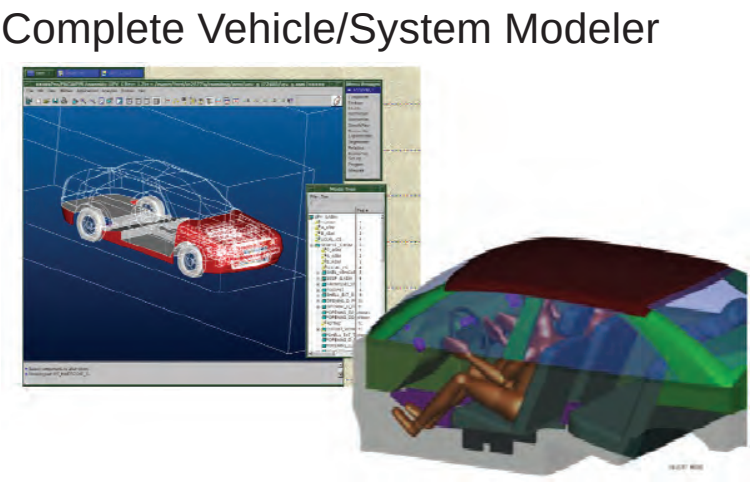


Figure 6. Examples of parametric models

Our method of approach to design this energy-efficient vehicle was to apply simulation-based design to first determine where all of the energy was going into and out of the vehicle. The results of these studies were very surprising. Figure 7 depicts an energy audit, conducted via simulation that identifies where heat is either leaving or entering the vehicle (winter or summer operation).

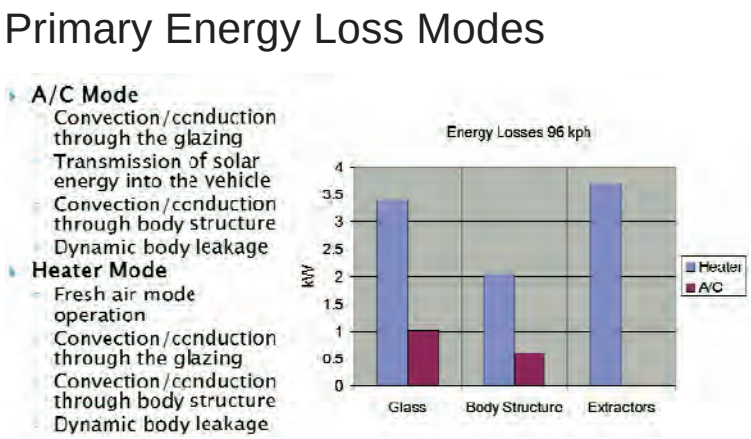


Figure 7. Results of energy audit for both summer and winter operation

From Figure 7, the results show that during summer operations the following conclusions can be made:

1. Primary heat pick-up is through convection and conduction through the glass system
2. Heat pick-up through the body structure is 50% of the convection through the glass
3. Solar gain, while significant, is much smaller than convection/conduction through the glass (Note that when the vehicle is parked, the glass is the primary heat pick-up contributor.)

Results of the study were surprising, in that we had spent significant research dollars to reduce our solar loads by the addition of films or glass coatings. These films would reduce the solar gain from approximately 5,000 W to 250 W (250 W

savings). This savings was ¼ of the heat gain we saw from convection/conduction through the glass. This result was, in effect, a case study on why one needs to perform total vehicle simulation of the vehicle.

The results of the cold weather simulations were even more astounding. We were losing 3.5 kW of heat through the glass, 2kW through the body, and over 3.5 kW went through the vehicle extractors. Note extractors must be used when the vehicle is run in "fresh-air mode" to reduce the interior vehicle pressure. It is also a safety concern: the vehicle must incorporate fresh-air in order to eliminate glass fogging on the interior.

Once the above results were studied, we went on a campaign to reduce the thermal losses in the vehicle and lower the thermal mass. This was accomplished by:

1. Incorporating dual-pane insulated glass for all windows except the front windshield (Safety Issue)
2. Increasing the R value of the body structure by employing gas-filled insulation
3. Installing infrared spectrum filter films on the windshield and backlight
4. Install a greatly-reduced-capacity air conditioning system (Small-scroll compressor and air handling unit)

Figure 8 shows the completed PNGV vehicle.

Figure 8. Completed PNGV vehicle

The completed PNGV vehicle was over 500 kg lighter than the base vehicle, ran on 1.116 kW power (in AC mode), and heated and cooled better than the baseline vehicle. The results of the heating and cooling tests are shown in Figures 9 and 10, respectively.

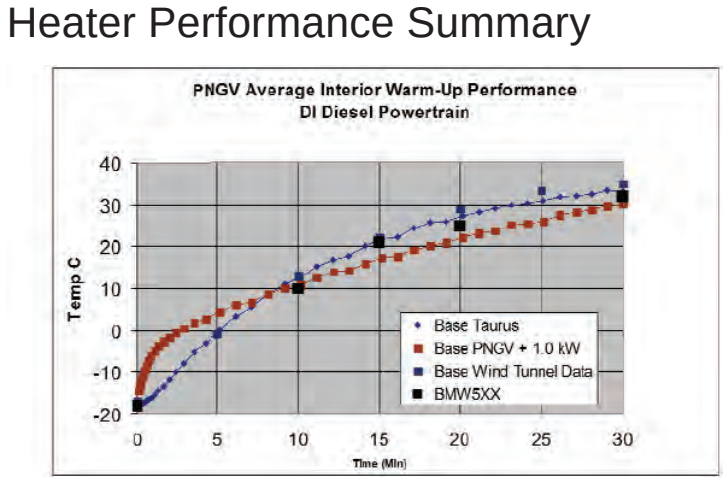


Figure 9. Results of heater testing (-18 C ambient)

Please note that testing was performed with an additional 1kW heater to make up the loss in engine heat due to the direct injection diesel engine. The PNGV vehicle warmed up faster than the baseline Taurus and a competitive BMW 5 Series sedan.

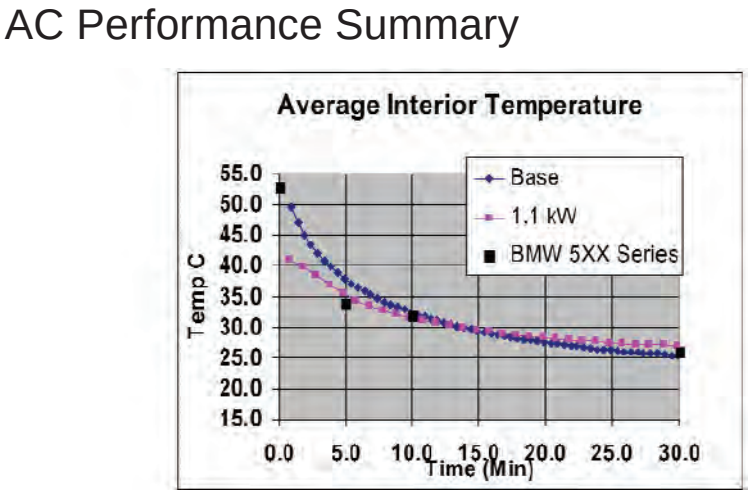


Figure 10. Results of summer testing (43 C ambient)

The reduced thermal mass, improved insulation, and infrared-reducing films results in a lower soak- temperature. (The vehicle was soaked under a simulated solar-load for 1 hour.) In addition, the PNGV achieved a faster pull-down than the baseline vehicle and a competitor BMW 5 Series sedan.

Case Study 3: Impact of Simulation-Based Design in the White Goods Industry

In 2005, Whirlpool Corporation initiated a strategic effort to implement systems engineering in its design process. This journey was necessary to reduce the product design cycle time in the journey. Simulation-based design was a key enabler to the implementation of systems engineering. Through simulation, Whirlpool was able to rapidly evaluate the impact of design changes on the platform performance, and more importantly, develop the bi-directional causal relationships from the design-critical characteristics to the platform requirements.

It is important to note that the platform requirements are not only related to performance; these requirements include, but are not limited to:

1. Business: Margin and volumes
2. Safety: Government codes, etc.
3. Manufacturing: Process capability, etc.
4. Environmental: Toxic materials, factory emissions, etc.
5. Performance: Product meets customer usage specifications
6. Supply Chain: Product survives trip from manufacturing site to customer's home
7. Societal: Product meets societal norms … global warming gases, etc.
8. End-of-Life: Product disposal

At Whirlpool, we utilized CAE to assess impact of design changes on requirements 1, 2, 4, 5, and 6. The traditional systems V diagram is shown in Figure 11.

In the systems engineering methodology, the platform requirements are then cascaded to the system and sub-system. Once the platform requirements are cascaded to the system and sub-system, platform simulation models are constructed to assess whether the system and sub-system designs met those requirements. NOTE: THE ANALYTICAL MODEL INCLUDES THE COMPLETE PLATFORM.

Once the system and sub-system requirements have been defined, these requirements are then cascaded to the components. At this point, another round of simulation is performed at the system/sub-system level. In addition, the overall platform model is then rerun to identify that component's/system's interactions.

Figure 12 depicts an example of the types of analyses that would be done at the component, system, and platform levels. These analyses were performed to assess design validity. Figure 12 shows results for thermal analyses of the refrigeration plant, evaporator performance, structural rigidity of the body structure, and the performance of the packaging material and the refrigerator structure during a typical clamping process. (Clamping: forklift truck clamping the product in the factory.)

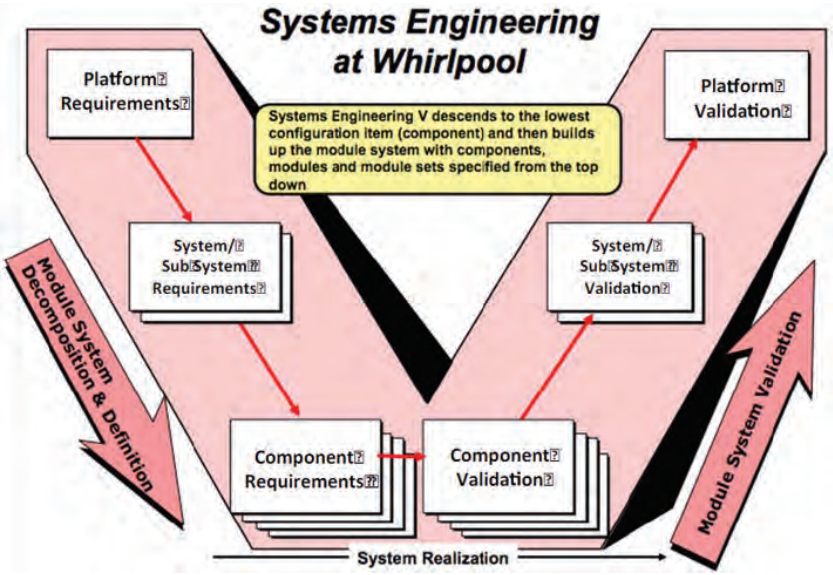


Figure 11. Traditional systems engineering V diagram

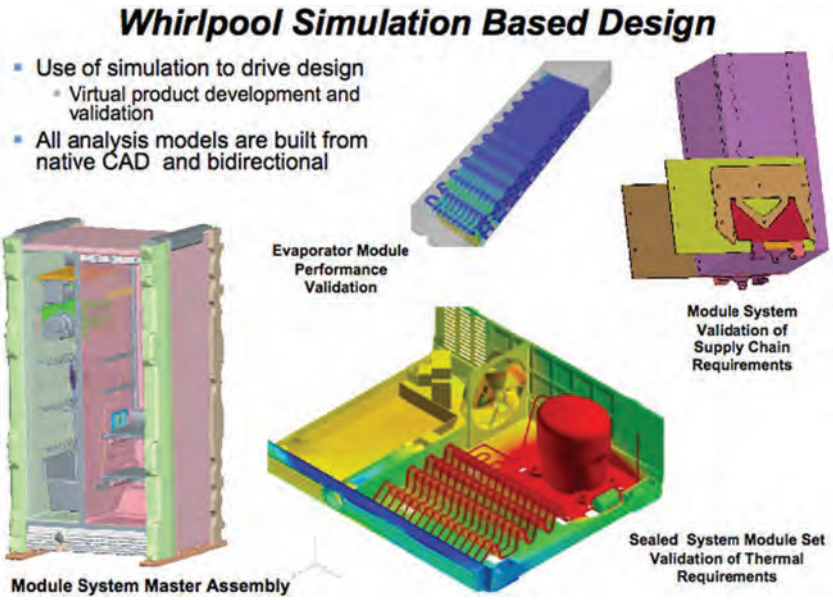


Figure 12. Example of component, system, platform analyses done during design validation

Once the platform requirements have been cascaded down to the component level and the design has been validated through CAE to meet those requirements, the first prototypes are then built. The prototypes are validated at the component, system, and platform level. Our goal is to perform many design churns on the left-side of the V diagram and build as few prototypes as possible. Only in this manner can the design cycle-time be reduced.

The primary benefit of performing simulation-based design and systems engineering is to establish the bi-directional causal relationships between the design-critical characteristics and the platform requirements. Establishing this relationship is critical to avoid being "surprised" in the development. Failure to identify these linkages usually results in catastrophic failure or product recall.

Figure 13 depicts a simplified example of requirements traceability.

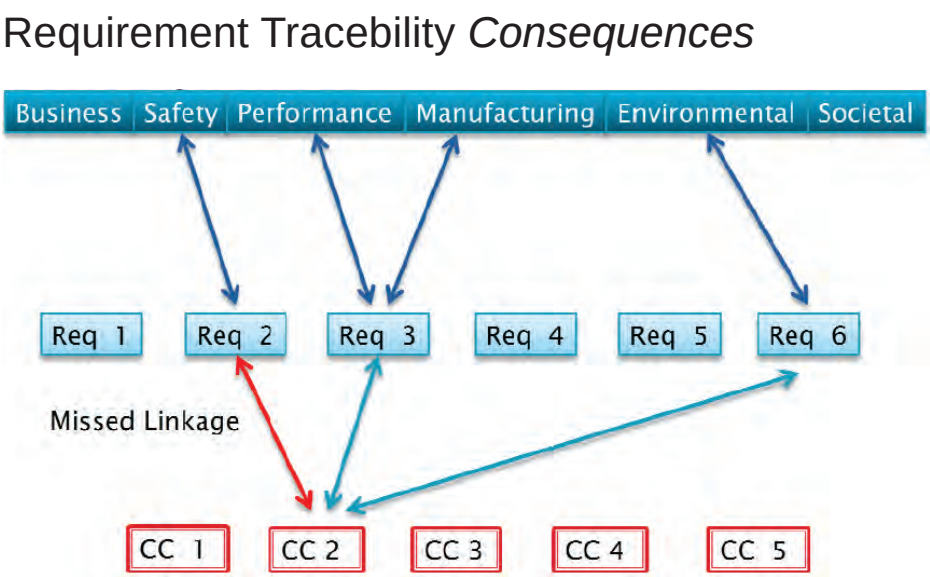


Figure 13. Example of requirements traceability to critical characteristics

From Figure 13, we can see that several platform requirements are connected to critical characteristic (CC) #2. CC2 is linked to Safety, Performance, and Manufacturing requirements via System requirements 2 and 3. However, there is a missed linkage of CC2 to the Safety requirement 2. Missing this linkage can result in product liability and hazard to the end-users. Simulation is an excellent way to determine these linkages on the platform level. These linkages cannot be established without robust simulation models.

4. Summary

We have demonstrated how CAE has grown from a tool used exclusively by Ph.D.s to being completely integrated into the product design cycle. This level of integration would not be possible without the advances made in CAD/CAE integration, auto meshing, and improvements in the solver technology of analysis tools. However, future work must be done to better integrate simulation results into existing requirements management software. To date, there does not appear to be a viable tool to manage the requirements traceability. In our time at Whirlpool, we had to grow our own software to manage this process.

References

1. Gielda, T.P., "STO Propulsion Installed Performance Predictions and Test", *International Journal of Computational Fluid Dynamics*, Volume 2, Issue 2, pp. 83–110, 1994.

2. Gielda T.P., "Systems Engineering a Case Study: Vehicle Thermal Management", NAFEMS 1999 World Congress, Newport, R.I. "Invited Lecture".

3. Gielda, T.P. "Simulation-Based Design in the Appliance Industry", Plenary speaker at the 2004 ASME Fluids in Engineering Conference, Charlotte, NC, July 2004.

4. Guan, Y., M. Hosni, B.W. Jones, and T.P. Gielda, "Investigation of human thermal comfort under highly-transient and non-uniform conditions for automobile applications- Part 2: Thermal Sensation Modeling", *ASHRAE Transactions*, 109, Part 2, 2003.

HPCMP UGC 2011

3. Computational Chemistry and Materials Science (CCM)

Energetic Materials

Construction of Accurate Reactive Potentials for Large Scale Molecular Dynamics Simulations of Materials Under Extreme Conditions

Edward F.C. Byrd and Neil Scott Weingarten
US Army Research Laboratory (ARL/RDRL-WML-B), Aberdeen Proving Ground, MD
{edward.fc.byrd, scott.weingarten}@us.army.mil

Abstract

Addressing the critical needs in the Department of Defense's (DoD's) Joint Vision 2020 to allow the US Armed Forces to fight and win by creating a dominant force across the full spectrum of military operations, the DoD has invested significant experimental and theoretical resources in the development of advanced materials designed to survive and optimally perform in the extreme conditions found in combat environments. In order to rapidly assess new emerging materials, this requires a fundamental understanding, at the atomistic level, of the physical and chemical properties of these materials. For large-scale atomistic descriptions of these materials, this requires the use of force-field models (due to their low computational cost). However, the effectiveness of these models is completely reliant on the accuracy of the force-field. A major limitation in the development of accurate, transferable force-fields is the inability of the general user to generate force-fields without requiring the input of an expert developer for the entire process.

Our goal is to demonstrate the ability of the Multiple-Objective Evolution Strategy (MOES) software to develop a reactive model of an explosive that will allow predictions of chemical reaction rates and mechanisms for thermodynamic conditions relevant to understanding energetic materials initiation and detonation phenomena, directly supporting the research efforts at the US Army Research Laboratory. Determination of reactive force-field (specifically ReaxFF) parameters for carbon, oxygen, nitrogen and hydrogen through the use of the MOES software will be used in large scale molecular dynamics simulations of a conventional explosive (RDX).

We are in the process of generating hundreds of potential candidate force fields using the evolutionary strategies utilized in the MOES software, with the force-fields closing in on the optimal solutions. Once we have generated these optimal solutions, we will further test the resultant force-fields using molecular dynamics calculations.

1. Introduction

Many materials of interest to the Department of Defense (DoD) are subjected to extreme conditions resulting from shock and thermal loading. In response, the materials often experience radical physical and chemical changes (e.g., material failure, phase transitions, and reactive chemistry) that influence their performance and survivability under these extreme conditions. Thus, it is imperative that a fundamental understanding of the properties and behavior of these materials be gained, in order to design and develop optimally performing weaponry or protective equipment. Such basic knowledge can be obtained through molecular dynamics (MD) simulations. While the method of MD is straightforward and, in principle, will provide a complete fundamental description of dynamic response of a material to insult, the quality of the description is completely dependent upon the assumed model of the interatomic interactions within the system (the force-field). An inaccurate force-field used in a simulation will produce spurious results. Conversely, an accurate force-field will provide an accurate representation of the behavior of the system. While it is not difficult to develop reasonably accurate force-fields for simple materials or for materials for which the MD simulations will explore a small volume of phase space (e.g., room condition simulations of elastic properties), these force-fields cannot correctly describe the complex chemical and physical behavior of materials subjected to shock or thermal loading. Therefore, more elaborate force-fields are required that can capture such complexities as the making and breaking of chemical bonds. The development of such "reactive" force-fields is not trivial, as evidenced by the dearth of reactive force-fields reported in the literature. Without such force-fields, the modeling of materials of interest to the DoD will be limited to low-energy, low-strain conditions, and will not be able to address high-energy phenomenon such as rapid heating, detonation, shear events or any condition where chemistry can occur.

ReaxFF is the most well-developed general reactive force-field for use in classical MD simulations of condensed phase material. The goal of the developers of ReaxFF was to create a reactive force-field that could be used to describe any chemical system composed of any element in the periodic table. The ReaxFF formalism is based on a bond-order/bond-distance relationship, where contributions to the sigma, pi and double-pi bond order terms are computed from the interatomic distances. The bond order is subsequently used to compute various partial energy contributions to the overall system energy, as given by Equation 1.

$$E_{sys} = E_{bond} + E_{lp} + E_{over} + E_{under} + E_{val} + E_{pen} + E_{coa} + E_{C2} + E_{tors} + E_{conj} + E_{Hbond} + E_{VdW} + E_{Coulomb} \quad (1)$$

The partial energy terms in Equation 1 account for: the bond energy, the lone-pair energy, corrections for atom over-coordination and under-coordination, the valence angle energy (including a penalty function E_{pen}), the angle conjugation energy, the C_2 correction energy, the torsion angle energy, the torsion conjugation energy, the hydrogen bond energy, the van der Waals energy and the Coulomb energy. Within these energy terms are a large number of adjustable parameters arranged of seven main types: general parameters, single-atom terms, general two-particle terms, heterogeneous two-particle terms, three-body terms, four-body terms, and hydrogen bond terms (see Figure 1 for a graphical representation). ReaxFF is parameterized using a training set of relevant structural and reaction path data that encapsulates the various environments each atom in the chemical system will experience. As it is becoming increasingly popular for use, ReaxFF consequently requires frequent reparameterization to accommodate the new chemical systems and materials that are being explored. To date, ReaxFF has been parameterized for use in molecular simulations of hydrocarbons (van Duin, 2001), nitramine and peroxide-based explosives (Strachan, 2003; Strachan, 2005; van Duin, 2005), silicon/silicon oxides (Buehler, 2006; van Duin, 2003; Chenoweth, 2005), aluminum/aluminum oxides (Q. Zhang, 2004; Q. Zhang 2005), aluminum hydride (Ojwang, 2009), transition metals (Nielson, 2005; Ludwig, 2006), metal oxide catalysts (Goddard, 2006), B-N-H systems (S. Han, 2005), alkali metal and alkaline earth systems (S.S. Han, 2005; Cheung, 2005), ferroelectrics (Goddard, 2002), hydrazines (L. Zhang, 2009), disordered ceramics (Chenoweth, 2009), antimony trihalides (Kua, 2009), biopolymers (Salmon, 2009), and shocked systems (Zybin, 2006; Zybin, 2006; Oleynik, 2006; L. Zhang, 2009). The formalism has been used in extremely large-scale computations (Nakano, 2007; Nakano, 2008; Nomura, 2007; Nomura, 2007). While ReaxFF is a breakthrough accomplishment, its complexity requires an exceptionally large degree of expertise in reparameterization, as evidenced by the observation in the literature that most of the reparameterization is done by the ReaxFF developers themselves. This unusually high hurdle to reparameterization (i.e., dependence on the academic developers) limits the number of potential DoD applications to those systems and materials for which ReaxFF has previously been parameterized. As an example, we cite an existing Challenge Project awarded to the original ReaxFF developers that uses ReaxFF to explore condensed phase chemistry of the CHNO explosive PETN (FY09–FY10 *Reactive Molecular Dynamics Simulations of Shock-induced Chemistry and Sensitivity of Energetic Materials for Insensitive Munitions*). Similar work was also performed under Capability Applications Project (CAP) 2720 titled "*Chemistry and Physics of Munitions Under Extreme Conditions: Atomistic Simulations of Shock-Induced Chemistry, Detonation and Sensitivity of Energetic Materials: Pentaerythritol Tetranitrate (PETN)*". Unfortunately, the version of ReaxFF used in the Challenge Projects and CAPs was specifically modified to represent PETN only and is not transferable or amenable for use for other chemical systems. Testing of the PETN ReaxFF force-field produced a completely erroneous result in its description of another CHNO explosive (RDX). Therefore, it is usually necessary to re-parameterize a new force-field for each chemical compound of DoD interest. Being reliant on the original developers to reparameterize ReaxFF for each potential DoD compound would be overly restrictive in the number and types of materials that DoD researchers could investigate.

In order to enable a non-expert user to reparameterize ReaxFF (or any other force-field) without reliance on the original force-field developers, the Multiscale Reactive Modeling of Insensitive Munitions Software Applications Institute (MSRM) [administered by US Army Research Laboratory (ARL) and Armament Research, Development and Engineering Center (ARDEC), under the High Performance Computing Modernization Program (HPCMP) support] created the first massively-parallel Multiple-Objective Evolution Strategies (MOES) program for fitting force-field parameters (including those of ReaxFF). While the software was first applied to CHNO explosives, MOES was written to address any chemical system. Thus the impact of the MOES software extends far beyond the scope of MSRM goals; it could potentially impact any material science project within the DoD involving condensed phase chemistry. Initial applications of MOES parameterizations of ReaxFF have produced promising results for RDX; however, these applications were constrained to allow variance of only a small subset of ReaxFF parameters. Further descriptions of the initial parameterization will be discussed in the Progress to Date section below.

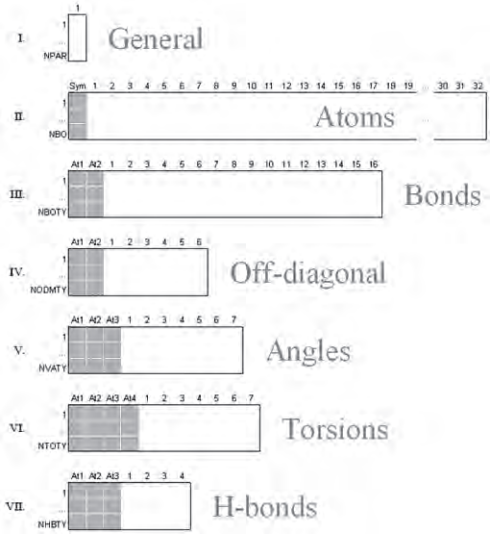


Figure 1. The seven blocks of the ReaxFF force field

2. Computational Approach

In order to generate accurate reactive force-field potentials for use in large-scale molecular dynamics calculations without relying on the original methods developers, we employ the use of the recently developed MOES software package. The generation of a reactive force-field requires substantial testing and optimization of multiple potential force-field candidates, with Challenge-level HPC resources necessary to properly search phase space. MOES employs a sophisticated self-adaptive evolutionary algorithm known as Evolution Strategies (Bäck, 1993; Bäck, 1996; Schwefel, 1995). The physical parameters to be optimized are augmented with a set of "strategy parameters" that include standard deviations (and covariances, if sufficient computational addressing space is available) which control the mutation of the physical parameters. These strategy parameters evolve along with the physical parameters of the force-field, and alter the manner in which the algorithm searches through the space of physical parameters, thus allowing the algorithm to adapt itself to each particular problem being optimized. The standard deviations play the role of the temperature in simulated annealing, except that in Evolution Strategies each degree-of-freedom has its own temperature and determines its own annealing schedule on-the-fly. The standard deviations approach zero as the algorithm concentrates the evolving population of parameter sets on a particular solution. Evolution Strategies has been employed extensively to solve engineering problems, but its use in scientific computations has been more limited; see, for example, Bäck (1995). The typical evolution strategies algorithm we employ can be outlined as follows:

1. Initialize parent solutions and their associated strategy parameters.
2. Mutate the parameters and their associated strategy parameters of a random parent solution to form an offspring solution.
3. Evaluate the objective function(s) for all the solutions in the population and through data envelopment analysis, assign a fitness value to each solution.
4. Perform selection; that is, choose new parent solutions for the next generation.
5. Go to step 2.

In multiple objective optimizations, we try to minimize several objectives (i.e., sums of errors) simultaneously (Fonseca, 1997). The concept of a Pareto frontier can be employed to decide which solutions offer the best balance of minimizing each of the objectives. Solutions on the Pareto frontier can be identified by the fact that in moving along the frontier (embedded in a space having the dimension of the number of objectives), we can make one objective smaller only at the expense of making some other objective(s) larger. Multiple objective optimization has been a subject of much interest in the evolutionary literature, and perhaps the best current algorithm is the Strength Pareto Evolutionary Algorithm (SPEA) of Zitzler and Thiele (Zitzler, 1999; Zitzler, 2001). We have compared MOES to the SPEA for a number of standardized multiple-objective test problems (Zitzler, 2000) and the MOES algorithm performs equally well.

We employ Data Envelopment Analysis (DEA) to compute a Pareto "efficiency" (Cooper, 2000) as a measure of fitness in our Evolution Strategies algorithm. DEA is a sophisticated application of linear programming (Chvátal, 1980). By solving one linear program for each parameter set in the evolving populations of solutions, we can identify those solutions

that lie on the Pareto frontier and assign a numerical score (efficiency) to each solution based on its L1 distance from the Pareto frontier. There has only been very limited use of DEA in evolutionary algorithms previously (Yun, 1999). However, our combination of a very flexible self-adaptive evolutionary algorithm that computes a multiple-objective Pareto fitness using linear programming methods is unique. An example of a Pareto frontier for a two-dimensional test case can be seen in Figure 2.

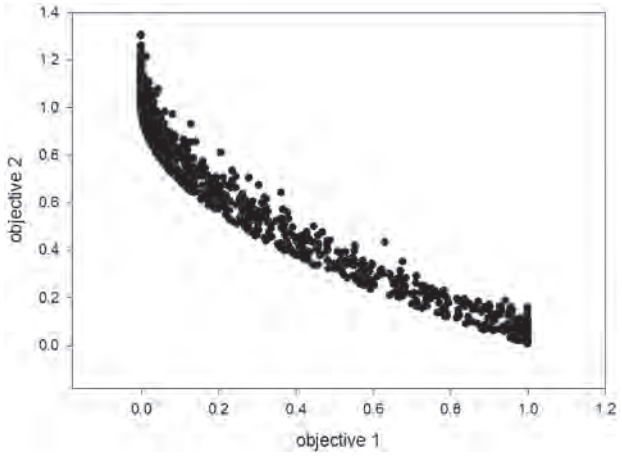


Figure 2. Solutions converging to a Pareto frontier for a two-dimensional test case

As a Pareto frontier provides a choice as to which objective(s) to satisfy at an arbitrarily level of accuracy by selecting the "proper" solutions lying on the frontier, there is no mathematical way to distinguish between the solutions lying on the Pareto frontier. We must therefore subject our generated force fields to test conditions not included in the fitting routine in order to determine force fields suitable to address problems of interest to the DoD. Upon the generation of candidate solutions using MOES (i.e., parameter sets for ReaxFF), subsequent assessment of the quality of each candidate force field will proceed as follows: an isothermal-isostress molecular dynamics (NsT-MD) simulation of a supercell of crystalline RDX at room conditions will be performed using the LAMMPS (Plimpton, 1995) suite of molecular dynamics simulation software and ReaxFF using MOES-generated parameters. Thermodynamic and crystallographic properties will be calculated and accumulated for averaging over the duration of the trajectory integration, during which atomic configurations will be recorded at periodic intervals. From these, thermally-averaged molecular orientational information (i.e., center-of-mass fractional coordinates and Euler angles) for each molecule will be generated. The Euler angles describe the orientations of molecules within the unit cells, and will be determined using a rotation matrix that transforms the principal axes of inertia of each molecule to the space-fixed coordinate system of the molecule (i.e., the *x*, *y*, and *z* Cartesian axes). This will allow direct comparison with experimental analogs to assess the quality of the force field at ambient conditions. Additionally, molecular structural parameters, such as bond lengths, bond angles, and ring conformation, for each of the eight symmetry-equivalent molecules in the RDX unit cell will be averaged and compared with experiment.

3. Results and Discussion

3.1 Previous Work

After augmenting the original training set with highly accurate quantum mechanical information of CHNO explosives generated in previous and current Challenge Projects (FY06–FY10 *First-principles Predictions of Crystal Structure of Energetic Materials – Army Research Office* - Krzysztof Szalewicz), ReaxFF was re-parameterized using MOES constrained to adjust a small subset of the parameters that govern long-range interactions as a proof-of-principle series of calculations. A final DEA analysis was performed on the MOES evolutions and produced nearly 440 viable, Pareto-efficient solutions. To test the quality of each Pareto-efficient force-field, molecular dynamics simulations were conducted to obtain the equilibrium crystal structure. The MD simulations were conducted on a 2×3×3 supercell of RDX for a total of 45 ps. The simulation was equilibrated for 10 ps within the NVE and NVT ensembles at 50K, followed by 5 ps within the NPT ensemble with isotropic expansion at 1 atm. Statistics were then collected and averaged over the final 30 ps to obtain the equilibrium lattice constants within the anisotropic NPT ensemble. Figure 3 shows the MD simulations of RDX using the re-parameterized ReaxFF, illustrating an outstanding agreement with experiment in terms of crystallographic parameters. The initial force-field predicted a density of the molecular crystal that was in error by greater than 12%; the new force-field

optimized by MOES predicts the density to 0.4%. Unfortunately, the molecular structural parameters were not in equally good agreement due to the constrained parameterization. These initial calculations provided the impetus for the current effort, where a much larger set of the ReaxFF parameters (more than 500 parameters for the current optimization versus 50 parameters for the initial optimization) are being varied.

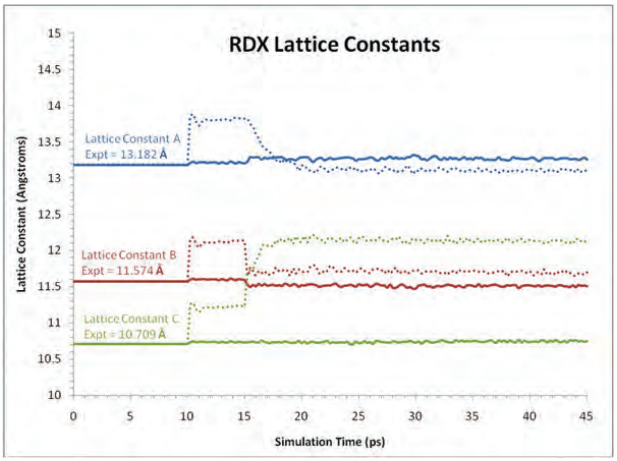


Figure 3. The equilibrium lattice constants obtained from a 45 ps MD simulation. The dotted lines represent the initial force-field and the solid lines represent the optimized force-field obtained from MOES.

3.2 Progress to Date

We are in the midst of generating multiple evolutions of potential candidate force-fields. However, until we have completed a converged evolution, it is of little value to test any force-field currently in the population of elite force-fields, as it may be replaced at any point in the future. We expect that our simulations will be finished before the end of the fiscal year, and that we will soon have some significant results. Following the successful completion and evaluation of our large-scale evolutions, we will perform further testing to determine how well the generated force- fields model reactive chemistry.

4. Conclusion

These calculations represent the largest employment of an evolutionary strategies-based methodology to generate ReaxFF force-field parameters to date. Hundreds of potential candidate force-fields have been generated using the evolutionary strategies utilized in the MOES software package, with the current force-fields closing in on the optimal Pareto frontier solutions. Once a final converged set of optimal solutions have been generated, further testing will be performed on the resultant force-fields using molecular dynamics calculations.

Acknowledgements

The DoD HPC Modernization Program supported this project by supplying supercomputer time under the Computing Challenge Project C4W. This time was made available at the DoD Supercomputing Resource Centers at the Army Research Laboratory SGI Altix, Harold, the US Air Force Research Laboratory SGI Altix, Hawk, and the Maui High Performance Computing Center Dell PowerEdge, Mana. We would like to thank PETTT contractors James V. Lill, at the US Air Force Research Laboratory, and James Larentzos, at the US Army Research Laboratory, for their invaluable aid in maintaining, updating and modifying the MOES and LAMMPS software packages.

References

Bäck, T. and H.-P. Schwefel, “An Overview of Evolutionary Algorithms for Parameter Optimization”, *Evolutionary Computation*, 1(1), pp. 1–23, 1993.

Bäck, T. and M. Schütz, “Evolution Strategies for Mixed-Integer Optimization of Optical Multilayers”, *Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Computation*, J.R. McDonnell, R.G. Reynolds, and D.B. Fogel, Eds., The MIT Press, Cambridge MA, pp. 33–51, 1995.

Bäck, T., "Evolutionary Algorithms", *Theory and Practice*, Oxford University Press, New York, NY, 1996.

Buehler, M.J., A.C.T. van Duin, and W.A. Goddard III, "Multiparadigm Modeling of Dynamical Crack Propagation in Silicon Using a Reactive Force Field", *Physical Review Letters*, 96, p. 095505, 2006.

Chenoweth, K., S. Cheung, A.C.T. van Duin, E.M. Kober, and W.A. Goddard III, "Simulations on the thermal decompositions of a poly(dimethylsiloxane) polymer using the ReaxFF reactive force field", *Journal of the American Chemical Society*, 127,

pp. 7192–7202, 2005.

Chenoweth, K., A.C.T van Duin, and W.A. Goddard III, "The ReaxFF Monte Carlo Reactive Dynamics Method for Predicting Atomistic Structures of Disordered Ceramics: Application to the Mo_3VO_x Catalyst", *Angewandte Chemie*, 48, p. 7630, 2009.

Cheung, S., W. Deng, A.C.T. van Duin, and W.A. Goddard III, "ReaxFF(MgH) reactive force field for magnesium hydride systems", *Journal of Physical Chemistry A*, 109, p. 851, 2005.

Chvátal, V., Linear Programming, W.H. Freeman, New York, NY, 1980.

Cooper, W.W., L.M. Seiford, and K. Tone, *Data Envelopment Analysis*, Kluwer, Boston, MA, 2000.

Fonseca, C.M., and P.J. Fleming, "An Overview of Evolutionary Algorithms in Multiobjective Optimization", *Evolutionary Computation*, S. Forrest, Ed., Morgan Kaufmann, San Francisco, CA, 3(1), pp. 1–16. 1997.

Goddard III, W.A., O. Zhang, M. Uludogan, A. Strachan, and T. Cagin, "The ReaxFF polarizable reactive force fields for molecular dynamics simulations of ferroelectrics", *American Institute of Physics Conference Proceeding*, 626, p. 45, 2002.

Goddard III, W.A., A.C.T. van Duin, K. Chenoweth, M.J. Cheng, S. Pudar, J. Oxgaard, B. Merinov, Y.H. Jang, and P. Persson, "Development of the ReaxFF Reactive Force Field for Mechanistic Studies of Catalytic Selective Oxidation processes on BiMoO~~x~~", *Topics in Catalysis*, 38, p. 93, 2006.

Han, S., J.K. Kang, H. M. Lee, A.C.T. van Duin, and W.A. Goddard III, "The theoretical study on interaction of hydrogen with single-walled boron nitride nanotubes. I. The reactive force field $ReaxFF_{HBN}$ development", *Journal of Chemical Physics*, 123, p. 114703, 2005.

Han, S.S., A.C.T. van Duin, W.A. Goddard III, and H.M. Lee, "Optimization and application of lithium parameters for the reactive force field, ReaxFF", *Journal of Physical Chemistry A*, 109 (20), pp. 4575–4582, 2005.

Kua, J., R.C. Daly, K.M. Tomlin, A.C.T. van Duin, T.B. Brill, R.W. Beal, and A.L. Rheingold, "Self-assembly of $SbCl_3$ and 1,4 dioxane: A cubic structure connected by very weak bonds", *Journal of Physical Chemistry A*, 113, p. 11443, 2009.

Ludwig, J., D.G. Vlachos, A.C.T. van Duin, and W.A. Goddard III, "Dynamics of the Dissociation of Hydrogen on Stepped Platinum Surfaces Using ReaxFF", *Journal of Physical Chemistry B*, 110, p. 4274, 2006.

Nakano, A., R.K. Kalia, K. Nomura, A. Sharma, P. Vashishta, F. Shimojo, A.C.T. van Duin, W.A. Goddard III, R. Biswas, and D. Srivastava, "A divide-and-conquer/cellular-decomposition framework for million-to-billion atom simulations of chemical reactions", *Computational Materials Science*, 38, pp. 642–652, 2007.

Nakano, A., R.K. Kalia, K. Nomura, A. Sharma, P. Vashishta, F. Shimojo, A.C.T. van Duin, W.A. Goddard, III, R. Biswas, D. Srivastava, and L.H. Yang, "De novo ultrascale atomistic simulations on high-end parallel supercomputers", *International Journal of High Performance Computing Applications*, 22, pp. 113–128, 2008.

Nielson, K.D., A.C.T. van Duin, J. Oxgaard, W. Deng, and W.A. Goddard III, "Development of the ReaxFF reactive force field for describing transition metal catalyzed reactions, with application to the initial stages of the catalytic formation of carbon nanotubes", *Journal of Physical Chemistry A*, 109, p. 493, 2005.

Nomura, K, R.K. Kalia, A. Nakano, and P. Vashishta, "Reactive nanojets: Nanostructure-enhanced chemical reactions in a defected energetic crystal", *Applied Physics Letters*, 91, p. 183109, 2007.

Nomura, K, R.K. Kalia, A. Nakano, P. Vashishta, A.C.T. van Duin, and W.A. Goddard III, "Dynamic Transition in the Structure of an Energetic Crystal during Chemical Reactions at Shock Front Prior to Detonation", *Physical Review Letters*, 99, p. 148303, 2007.

Plimpton, S., "Fast Parallel Algorithms for Short-Range Molecular Dynamics", *Journal of Computational Physics*, 117, pp. 1–19, 1995. (http://lammps.sandia.gov)

Ojwang, J.G.O., R.A. van Santen, G.J. Kramer, A.C.T van Duin, and W.A. Goddard III, "Parameterization of a reactive force field for aluminum hydride", *Journal of Chemical Physics*, 131, p. 044501, 2009.

Oleynik, I.I., M. Conroy, S.V. Zybin, and C.T. White, "Energetic Materials at High Compression: First-Principles Density Functional Theory Studies", *Proceedings of 13th International Detonation Symposium*, Norfolk, VA, July 23–28, 2006, IDS104.

Salmon, E., A.C.T van Duin, F. Lorant, P-M. Marquaire, and W.A. Goddard III, "Thermal decomposition process in algaenan race L. Part 2: Molecular dynamics simulations using the ReaxFF reactive force field", *Organic Geochemistry*, 40, p. 416, 2009.

Schwefel, H.-P. and G. Rudolph, "Contemporary Evolution Strategies", *Advances in Artificial Life: Proceedings Third International Conference on Artificial Life*, Granada, Spain, F. Morán et al, Eds., Springer, Berlin, pp. 893–907, 1995.

Strachan, A., A.C.T. van Duin, D. Chakraborty, S. Dasgupta, and W.A. Goddard III, "Shock waves in high-energy materials: The initial chemical events in nitramine RDX", *Physical Review Letters*, 91, p. 098301, 2003.

Strachan, A., E.M. Kober, A.C.T. van Duin, J. Oxgaard, and W.A. Goddard III, "Thermal decomposition of RDX from reactive molecular dynamics", *Journal of Chemical Physics*, 122, p. 054502, 2005.

van Duin, A.C.T., S. Dasgupta, F. Lorant, and W.A. Goddard III, "ReaxFF: A Reactive Force Field for Hydrocarbons", *Journal of Physical Chemistry A*, 105(41), pp. 9396–9409, 2001.

van Duin, A.C.T., A. Strachan, S. Stewman, Q. Zhang, and W.A. Goddard III, "ReaxFF SiO Reactive Force Field for Silicon and Silicon Oxide Systems", *Journal of Physical Chemistry A*, 107, pp. 3803–3811, 2003.

van Duin, A.C.T., Y. Zeiri, F. Dubnikova, R. Kosloff, and W.A. Goddard III, "Atomistic-Scale Simulations of the Initial Chemical Events in the Thermal Initiation of Triacetonetriperoxide", *Journal of the American Chemical Society*, 127, p. 11053, 2005.

Yun, Y.B., H. Nakayama, T. Tanino, and M. Arakawa, "A Multi-Objective Optimization Method Combining Generalized Data Envelopment Analysis and Genetic Algorithms", *IEEE, SMC on CD-ROM Proceeding of IEEE*, SME'99, 1999.

Zhang, L., S.V. Zybin, A.C.T. van Duin, S. Dasgupta, W.A. Goddard III, and E.M. Kober, "Carbon Cluster Formation during Thermal Decomposition of Octahydro-1,3,5,7-tetranitro-1,3,5,7-tetrazocine and 1,3,5-Triamino-2,4,6-trinitrobenzene High Explosives from ReaxFF Reactive Molecular Dynamics Simulations", *Journal of Physical Chemistry A*, 113, pp. 10619–10640, 2009.

Zhang, L., A.C.T van Duin, S.V. Zybin, and W.A. Goddard, III, "Thermal Decomposition of Hydrazines from Reactive Dynamics Using the ReaxFF Reactive Force Field", *Journal of Physical Chemistry B*, 113, pp. 10770–10778, 2009.

Zhang, Q., T. Cagin, A.C.T. van Duin, W.A. Goddard III, Y. Qi, and L.G. Hector, "Adhesion and nonwetting-wetting transition in the Al/α-Al_2O_3 interface", *Physical Review B*, 69, p. 045423, 2004.

Zhang, Q., Y. Qi, L.G. Hector, T. Cagin, and W.A. Goddard III, "Atomic simulations of kinetic friction and its velocity dependence at Al/Al and α-Al_2O_3/α-Al_2O_3 interfaces", *Physical Review B*, 72, p. 045406, 2005.

Zitzler, E., and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach", *IEEE Transactions on Evolutionary Computing*, 3(4), pp. 257–274, 1999. (http://www.tik.ee.ethz.ch/~zitzler/)

Zitzler, E., K. Deb, and L. Thiele, "Comparison of Multiple Objective Evolutionary Algorithms: Empirical Results", *Evolutionary Computation*, 8(2), pp. 173–195, 2000.

Zitzler, E., M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm", TIK Institut für Informatik und Kommunikationsnetze, Zürich, *TIK Report 103*, 2001. (http://www.tik.ee.ethz.ch/~zitzler/)

Zybin, S.V., L. Zhang, A.C.T. van Duin, and W.A. Goddard III, "Analysis of shock decomposition and sensitivity of energetic materials with ReaxFF molecular dynamic" *Proceedings of 13th International Detonation Symposium*, Norfolk, VA, July 23–28, 2006. (IDS163)

Zybin, S.V., V.V. Zhakhovskii, E.M. Bringa, S.I. Abarzhi, and B. Remington, "Molecular dynamics simulations of the Richtmyer-Meshkov instability in shock loaded solids", *American Institute of Physics Conference Proceeding*, 845, pp. 437–440, 2006.

Design of Energetic Ionic Liquids

Jerry A. Boatz
US Air Force Research Laboratory, Space and Missile Propulsion Division (AFRL/RZSP), Edwards AFB, CA
jerry.boatz@edwards.af.mil

Gregory A. Voth
University of Chicago, Chicago, IL
gavoth@uchicago.edu

Mark S. Gordon
Iowa State University, Ames, IA
mark@si.msg.chem.iastate.edu

Sharon Hammes-Schiffer
Pennsylvania State University, University Park, PA
shs@psu.edu

Abstract

An essential need of the US Air Force is the discovery, development, and fielding of new, energetic materials for advanced chemical propulsion in space and missile applications. Some of the key factors driving the requirement for new chemical propellants include: (a) improved performance in terms of increased specific impulse and density, (b) reduced sensitivity to external stimuli such as impact, friction, shock, and electrostatic discharge, and (c) mitigation of environmental and toxicological hazards (and the resulting costs) associated with currently used propellants.

A class of compounds that can potentially meet these requirements is known as ionic liquids (ILs), which are chemical salts with unusually low melting points. The physical and chemical properties of ILs render them useful for many purposes, most notably as environmentally-benign ("green") solvents/reaction media but also as catalysts, electrolytes, etc. From a Department of Defense (DoD) perspective, ILs are being explored as new propellants, explosives, and munitions. The US Air Force, in particular, is interested in ILs as potential replacements for currently used monopropellants such as hydrazine—which is carcinogenic, highly-toxic, and has relatively modest performance characteristics. In contrast, many ILs have superior densities and specific impulses, as well as significantly reduced sensitivity and toxicity characteristics. Furthermore, their properties can be carefully tuned via the choice of the component ions.

The overall objective of the Design of Energetic Ionic Liquids Challenge Project is to address several key technical issues and challenges associated with the characterization, design, and development of ILs as new monopropellants. Among these, for example, are a fundamental understanding of the (in)stability of ILs, the intrinsic nature of the short- and long-range structure and interactions between the component ions, and identification of the key steps in the initial stages of decomposition and combustion. A hierarchy of computational approaches is employed, including atomistic, high-level quantum chemical methods applied to individual ions and ion clusters, condensed phase atomistic molecular dynamics simulations utilizing polarizable force fields, and mesoscale-level simulations of bulk ionic liquids based upon multiscale coarse graining techniques.

1. Introduction

The design of new high-energy density materials, which are more efficient, reliable, and environmentally-benign than existing rocket propellants, is a high Department of Defense (DoD) priority. The focus of this effort has been on the development of new propellants and energetic additives, including highly strained hydrocarbons, poly-nitrogen compounds, and advanced monopropellants. Some of the issues that must be addressed in theoretical efforts to design new energetic materials include an assessment of their energy content, their thermodynamic and kinetic stability, and the design of new synthetic routes to proposed new compounds that have not yet been synthesized.

A specific area of interest to the DoD is the discovery of a suitable replacement for hydrazine, a widely-used monopropellant for low-thrust propulsion applications such as orbital maneuvering and satellite station-keeping. The desire to replace hydrazine is motivated by several factors. Perhaps the most severe limitation of hydrazine is its carcinogenic

nature and extreme respiratory and dermatological toxicity, with correspondingly large costs associated with controlling these environmental and toxicological hazards. Furthermore, the performance of hydrazine as a monopropellant is rather modest, due to its relatively low density and specific impulse compared to a prototypical ionic monopropellant salt such as 4-amino-1,2,4-triazolium dinitramide. The replacement of hydrazine with more energetic, less hazardous energetic monopropellants is clearly needed.

A specific type of energetic material of current interest is derived from a broad class of compounds known generically as ionic liquids (ILs), which are chemical salts with unusually low melting points; e.g., below 100° C. The general interest in ILs has focused mainly on their use as environmentally-benign ("green") solvents for a wide range of chemical reactions. Some of the properties of ILs that make them attractive as solvents include their low vapor pressure, large liquid ranges, and thermal stability. The interest in ILs as new monopropellants stems from several factors. For example, the properties of ILs, including their energy content, can be "tuned" through a judicious choice of component ions and their substituent. Furthermore, the virtually nonexistent vapor pressure of ILs greatly reduces the environmental and toxicological hazards due to respiratory and dermatological toxicity. Finally, the densities of ILs generally are significantly greater than those of conventional liquid monopropellants such as hydrazine.

Although there have been extensive experimental studies of chemical reactions in ILs, little has been done in the area of characterization of the fundamental chemical and physical properties of ILs. In particular, one of the most pressing needs in the broader area of IL development, and particularly in the design of energetic ILs, is the application of robust theoretical methods for the reliable prediction of IL heats of formation, synthesis routes, phase transitions, ion conformations, thermal stabilities, densities, and viscosities. The focus of this study is on the characterization, design, and synthesis of the next generation of monopropellants for rocket propulsion applications.

2. Computational Methods

An integrated approach utilizing multiple computational methods is used to predict and characterize the intrinsic and bulk properties of energetic ionic liquids. At the molecular level, highly-accurate electronic structure methods are used to predict the fundamental properties of the ionic liquid components, including molecular structures, charge delocalization, heats of formation, and proton transfer reaction pathways and barriers. Geometries, electronic structures, and properties (including heats of formation) of the component ions are predicted using second-order perturbation theory (MP2, also known as MBPT(2)[1]), density functional theory (DFT)[2], coupled cluster theory (CCSD(T)[3]), and the "Gaussian-N" (GN)[4] methods. The Nuclear-Electronic Orbital (NEO)[5] approach is used for capturing the quantum dynamical effects of hydrogen bonding and proton transfer. The Fragment Molecular Orbital (FMO) method[6], which decomposes a large molecular system (e.g., a cluster, protein, liquid, zeolite, etc.) into small subunits (fragments) that are designed to both retain the high accuracy of the chosen quantum mechanical level of theory while greatly reducing the demands on computational time and resources, is used in studies of ion clusters. In addition, the complex spectrum of ionic liquid physical properties requires utilization of atomistic molecular dynamics and coarse-grained condensed phase simulations in order to obtain reliable predictions of many key bulk properties. Although the entire spectrum of electronic structure methods described above has been utilized in this challenge project, only a subset of the results, obtained using MP2, are reported here.

GAMESS[7], the primary quantum chemistry code used in this study, is highly-scalable. For example, the fixed-size parallel efficiency of a large MP2 gradient calculation (930 atomic orbitals) SGI Altix ICE system at the US Army Engineer Research and Development Center (ERDC) DoD Supercomputing Resource Center (DSRC) on 4,096 cores is 86%, relative to the same calculation on 128 cores. The condensed phase molecular dynamics (MD) simulations were performed using the scalable LAMMPS classical MD code, which delivers a scaled-size parallel efficiency of approximately 90% on 8,192 cores on a Cray XT3 (see http://lammps.sandia.gov/bench.html).

3. Results and Discussion

Ionic Clusters: Since the processes of ignition and combustion of ILs take place in the gas phase, it is vitally important to determine the nature of ILs in the vapor phase. For example, do the species present in the vapor phase consist of individual ions, single ion pairs, neutrals, larger clusters of ions or neutrals, or a mix of these? Furthermore, the species present in the vapor phase are relevant to experimentally measured enthalpies of vaporization, which is a key property in determining the performance of ILs as chemical propellants.

Ionic liquids generally have negligible vapor pressures; nonetheless, it has been shown that some ILs can be distilled in vacuum with minimal thermal degradation. Under reduced pressure, certain ionic liquids have demonstrated volatility and they are thought to vaporize as intact cation-anion ion pairs. Recent experiments[8] have detected the products of reacting

ions such as NO^+, NH_4^+, NO_3^-, and O_2^- with vaporized aprotic ionic liquids in order to confirm the presence of the intact ion pair. To aid in the interpretation and confirmation of these and other similar experimental results, electronic structure calculations have been performed to predict the free energies of the reactions involving the ions NO^+, NH_4^+, NO_3^-, and O_2^- with the 1-ethyl-3-methylimidazolium *bis*(trifluoromethylsulfonyl)imide ([emim$^+$][NTf_2^-]) and n-butylmethylimidazolium dicyanamide (([bmim$^+$][dca$^-$]) ion pairs, at the MP2/6-311++G(d,p)[9] level of theory. Multiple reaction channels were considered, including proton transfer, charge transfer, fragmentation, ion addition, and ion exchange pathways.

For example, when [emim$^+$][NTf_2^-] is reacted with NH_4^+, the product cations detected experimentally include [emim$^+$], the ion addition cluster [emim$^+$][NTf_2-][NH_4^+], and the secondary condensation product [emim$^+$][NTf_2-][emim$^+$]. The MP2/6-311++G(d,p) computed reaction enthalpies and free energies summarized in Table 1 indicate that formation of [emim+] is more likely to occur via a combined proton transfer plus fragmentation process (reactions 3 and 4, both of which are exergonic) rather than by charge transfer (reaction 5, which is endergonic.) Formation of [emim+][NTf_2-][NH_4^+] and [emim+][NTf_2-][emim$^+$] are both predicted to be energetically favorable (reactions 6 and 12, respectively), which is consistent with the experimental detection of these species.

Table 1. Enthalpies and free energies of reaction of the RTIL $EMIM^+NTf_2^-$ with various ions at 298K (MP2/6-311++G(d,p)). The EMIM carbene species in reaction 10 is denoted as EMIM.

	Reaction	ΔH (298K) (kcal/mol)	ΔG (298K) (kcal/mol)
1)	$EMIM^+NTf_2^- + NO^+ \rightarrow EMIM^+ + NTf_2 + NO$	3.6	-11.0
2)	$EMIM^+NTf_2^- + NO^+ \rightarrow EMIM^+ + NO^+NTf_2^-$	-32.8	-35.9
3)	$EMIM^+NTf_2^- + NH_4^+ \rightarrow EMIM^+ + HNTf_2 + NH_3$	-3.3	-17.4
4)	$EMIM^+NTf_2^- + NH_4^+ \rightarrow EMIM^+ + NH_3HNTf_2$	-19.8	-25.6
5)	$EMIM^+NTf_2^- + NH_4^+ \rightarrow EMIM^+ + NTf_2 + NH_4$	117.5	103.0
6)	$EMIM^+NTf_2^- + NH_4^+ \rightarrow EMIM^+NTf_2^-NH_4^+$	-47.7	-40.8
7)	$EMIM^+NTf_2^- + NO_3^- \rightarrow NO_3^-EMIM^+NTf_2^-$	-36.3	-28.0
8)	$EMIM^+NTf_2^- + O_2^- \rightarrow NTf_2^- + EMIM{:}HO_2$	8.9	3.9
9)	$EMIM^+NTf_2^- + O_2^- \rightarrow NTf_2^- + EMIM^+O_2^-$	13.4	10.3
10)	$EMIM^+NTf_2^- + O_2^- \rightarrow NTf_2^- + EMIM{:} + HO_2$	27.1	13.0
11)	$EMIM^+NTf_2^- + O_2^- \rightarrow NTf_2^- + EMIM + O_2$	24.7	12.1
12)	$EMIM^+NTf_2^- + EMIM^+ \rightarrow EMIM^+NTf_2^-EMIM^+$	-32.4	-21.2

1,2,4-Triazolium dinitramide ion clusters: Previous studies of ionic liquids[10] have focused on the decomposition of ion pairs, providing insight into the chemistry of their ignition as high-energy fuels. The focus of this effort is to extend the previous work to examine the effects of water molecules on the stability of ion pairs and larger clusters. The addition of small concentrations of water in some energetic ionic liquid propellant formulations is used to tune their properties, so it is important to characterize and understand the chemistry of water-IL mixtures.

The structures of the 1,2,4-triazolium dinitramide ion pair in the presence of zero, one, and two water molecules were computed at the MP2/6-31++G(d,p)[9] level of theory. In the absence of water, the ionic form spontaneously undergoes proton transfer to form a hydrogen-bonded complex between the neutral 1,2,4-triazole and dinitramine molecules, as shown in Figure 1. An ion pair in the presence of a single water molecule undergoes a double proton transfer across the intervening water molecule, as illustrated in Figure 2. Addition of a second water molecule results in the possibility of preserving the ionic form of the dimer or forming a neutral complex via water-mediated double proton transfer, with the outcome governed by the initial placement of the water molecules. The most stable form is the ionic structure shown in Figure 3, with other ionic structures (not shown) higher in energy by 0.5–0.7 kcal/mol. The neutral forms are less stable than the ionic structure in Figure 3 by 2–19 kcal/mol.

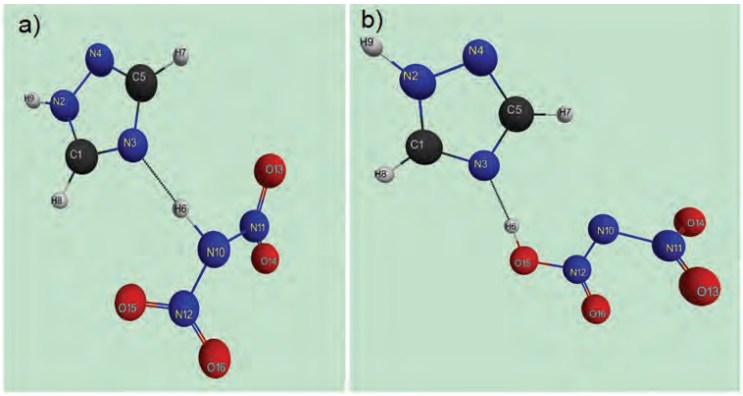


Figure 1. MP2/6-31++G(d,p) geometries of neutral 1,2,4-triazole dinitramine complexes

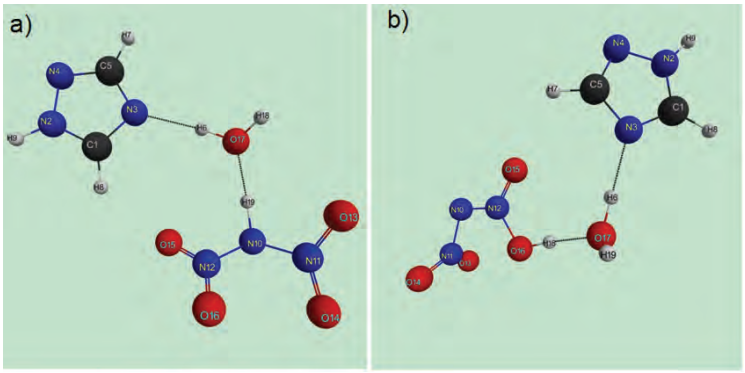


Figure 2. MP2/6-31++G(d,p) geometries of neutral 1,2,4-triazole dinitramine complexes with one water molecule

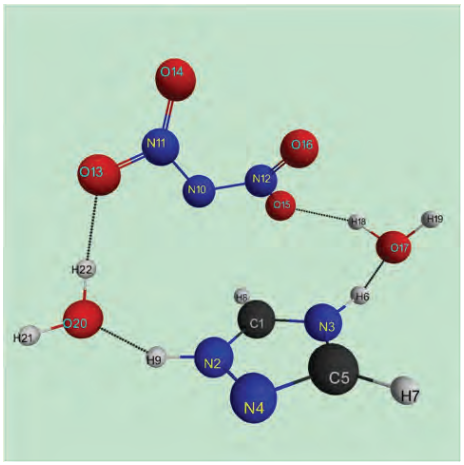


Figure 3. MP2/6-31++G(d,p) geometry of the most stable 1,2,4-triazolium dinitramide ion pair with two water molecules

Ionic liquids/water mixtures: The properties of ionic liquids can be "tuned" using any of several methods, such as adding water, varying the length of the alkyl side chains on the cation, and by choosing the anion. The role of water in imidazolium-based ionic liquids/water mixtures was investigated through large-scale molecular dynamics (MD) simulations utilizing the AMBER force field for the ionic liquid components and the SPC model for the water molecules.[11] The relation between water content and ionic liquids tail aggregation[12] was revealed by the MD simulation of 1-octyl-3-methylimidazolium ($OMIM^+$) BF_4^-/water mixtures.[11] Nearly all the tail groups aggregated into a few large domains below a water mole fraction (X_w) of ~0.8. At water mole fractions between 0.8 and 0.9, the tail groups broke into many small domains (Figure 4a shows a snapshot of the tail aggregation at X_w=0.875). Large aggregates were easily seen and micelle-

like structures were also observed. Figure 4b focuses on a roughly cylindrical aggregate taken from the same distribution as shown in Figure 4a. The tail groups are located inside the cylinder, while the head groups (only the nitrogen atoms are shown) lie outside the cylinder.

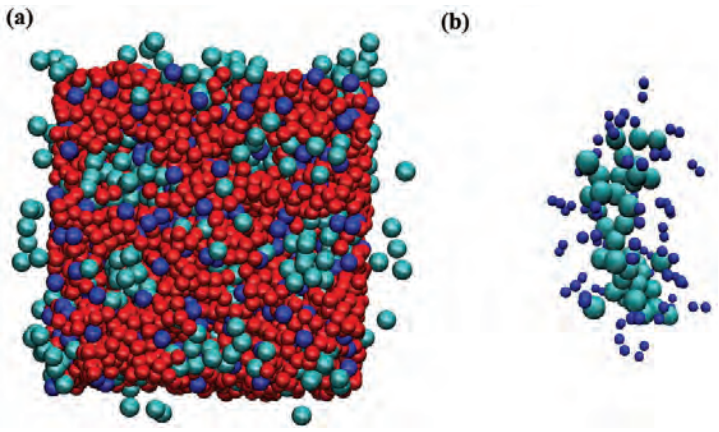


Figure 4. The distribution of tail groups at water mole fraction X_w=0.875 in one [OMIM+]BF_4^-/water mixture. The red spheres represent water molecules, the cyan spheres are tail groups, and the blue spheres are head groups. In panel (a) tail groups, head groups and waters are all shown, and Panel (b) zooms in on one of the roughly cylindrical tail group structures; the blue spheres represent nitrogen atoms.

It was found that the local water structure strongly correlated with the choice of anions, but was not notably affected by differing cations.[11] At low water concentrations, changing the anions from BF_4^- to Cl^- resulted in a substantially reduced water coordination number, as shown in Figure 5a. This result demonstrated that the interactions between anions and water molecules significantly affected the water structure. At high water concentrations, the lower number of anions in the solution resulted in the water's structure being less affected. Further insight was obtained from studying the distribution of water cluster sizes (i.e., the number of molecules in a cluster), as shown in Figure 5b. At X_w =0.20 water clusters containing 2 to 5 water molecules existed in 1-butyl-3-methylimidazolium ($BMIM^+$) BF_4^- and [$OMIM^+$]BF_4^-/water mixtures. In [$OMIM^+$]Cl^-/water mixtures at X_w=0.20, only those clusters with fewer than 4 water molecules could be seen. At a water concentration of 0.5, water clusters containing 8 or more water molecules formed in the [$BMIM^+$]BF_4^- and [$OMIM^+$]BF_4^-/water mixtures, demonstrating the existence of local water networks. The water clusters in the [$OMIM^+$]Cl^-/water mixture exhibited a tendency to increase in size when the water concentration changed from 0.2 to 0.5. However, their sizes remained much smaller than those in the [$BMIM^+$]BF_4^- and [$OMIM^+$]BF_4^-/water mixtures, which was found to be related to electrostatic interactions.

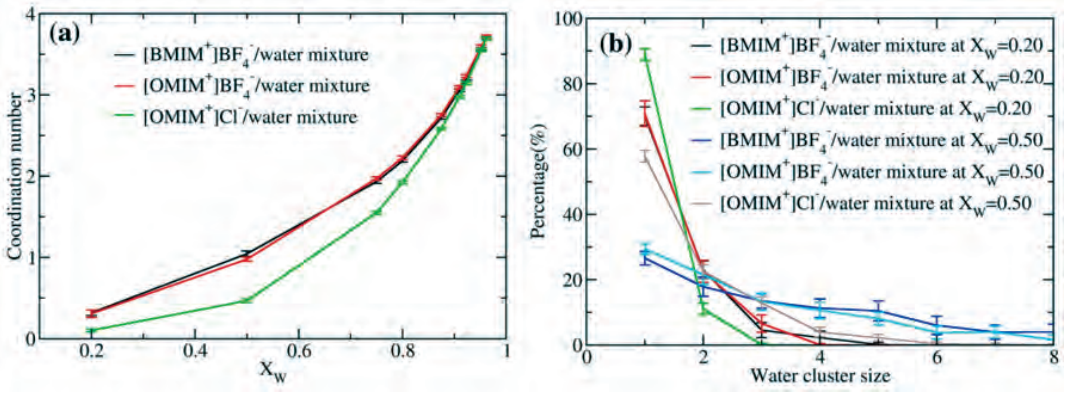


Figure 5. (a) The coordination number of waters in [[$OMIM^+$]BF_4^- and [$OMIM^+$]Cl^-/water mixtures, where X_w represents the water mole fraction; (b) Water cluster size distributions in [$OMIM^+$]BF_4^- and [$OMIM^+$]Cl^-/water mixtures at water mole fractions X_w=0.2 and 0.5

4. Summary and Conclusion

Free energies of the reactions between the ions NO^+, NH_4^+, NO_3^-, and O_2 and the 1-ethyl-3-methylimidazolium bis-trifluoromethylsulfonylimide ([emim$^+$][NTf$_2^-$]) ion pair determined by *ab initio* quantum mechanical calculations indicate that ion exchange or ion addition are energetically more favorable than charge transfer processes. The observed reaction products, the formation of which is consistent with the corresponding predicted reaction free energies, confirm the presence of single ([emim$^+$][NTf$_2^-$]) ion pairs in the gas phase.

In the triazolium dinitramide ion dimer, the presence of water increases the molecular distances between the ions, which could decrease the energy density of the liquid. Also, if a proton transfer is key to the ignition reaction, the protons would need to move across the water molecules, as opposed to direct transfer from ion to ion. This intermediate step could hinder ignition kinetically via a second energy barrier.

Large-scale MD simulation was performed in order to reveal the role of water in "tuning" the properties of ionic liquid/water mixtures. The tail domain of [OMIM$^+$] exhibited interesting structural changes (such as micelle, cylinder and network) corresponding to the changes in water concentrations. It was also found that there was a strong correlation between the local water structure and the choice of anion, while differing cations had an insignificant influence on the local water distribution.

Acknowledgements

The authors gratefully acknowledge grants of computer time at the five Department of Defense Supercomputing Resource Centers: the US Air Force Research Laboratory, the US Army Research Laboratory, the US Army Engineer Research and Development Center, NAVY, and Maui.

References

Møller, C. and M.S. Plesset, *Phys. Rev.*, 46, 618, 1934; J.A. Pople, J.S. Binkley, and R. Seeger, *Int. J. Quantum Chem.*, S10, 1, 1976; M.J. Frisch, M. Head-Gordon, and J.A. Pople, *Chem. Phys. Lett.*, 166, 275, 1990

R.J. Bartlett and D.M. Silver, *Int. J. Quantum Chem. Symp.*, 9, 1927, 1975; Y. Zhao and D.G. Truhlar, *Theor. Chem. Accounts*, 120, pp. 215–241, 2008.

2. Becke, A.D., *J. Chem. Phys.*, 98, 5648, 1993; P.J. Stephens, F.J. Devlin, C.F. Chablowski, and M.J. Frisch, *J. Phys. Chem.*, 98, 11623, 1994; R.H. Hertwig and W. Koch, *Chem. Phys. Lett.*, 268, 345,1997; S.H. Vosko, L. Wilk, and M. Nusair, Can. *J. Phys.*, 58, 1200, 1980.

3. Purvis III, G.D. and R.J. Bartlett, *J. Chem. Phys.*, 76, 1910, 1982; K. Raghavachari, G.W. Trucks, J.A. Pople, and M. Head-Gordon, *Chem. Phys. Lett.*, 157, 479, 1989.

4. See, for example, L.A. Curtiss, K. Raghavachari, G.W. Trucks, and J.A. Pople, *J. Chem. Phys.*, 94, 7221, 1991.

5. See, for example, C. Swalina, M.V. Pak, and S. Hammes-Schiffer, *J. Chem. Phys.*, 123, 14303, 2005.

6. Kitaura, K., T. Sawai, T. Asada, T. Nakano, and M. Uebayasi, Chem. Phys. Lett., 312, 319, 1999; K. Kitaura, E. Ikeo, T. Asada, T. Nakano, and M. Uebayasi, *Chem. Phys. Lett.*, 313, 701, 1999; T. Nakano, T. Kaminuma, T. Sato, Y. Akiyama, M. Uebayasi, and K. Kitaura, *Chem. Phys. Lett.*, 318, 614, 2000; K. Kitaura, S.-I. Sugiki, T. Nakano, Y. Komeiji, and M. Uebayasi, *Chem. Phys. Lett.*, 336, 163, 2001; Y. Inadomi, T. Nakano, K. Kitaura, and U. Nagashima, *Chem. Phys. Lett.*, 364, 139, 2002; T. Nakano, T. Kaminuma, T. Sato, K. Fukuzawa, Y. Akiyama, M. Uebayasi, and K. Kitaura, *Chem. Phys. Lett.*, 351, 475, 2002; D.G. Fedorov, and K. Kitaura, *J. Phys. Chem. A*, 111, 6904, 2007.

7. Schmidt, M.W., K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S. Su, T.L. Windus, M. Dupuis, and J.A. Montgomery, J. Comput. Chem., 14, pp. 1347–1363, 1993; M.S. Gordon and M.W. Schmidt, *Theory and Applications of Computational Chemistry: the first forty years*, C.E. Dykstra, G. Frenking, K.S. Kim, G.E. Scuseria (editors), Elsevier, Amsterdam, pp. 1167–1189, 2005.

8. Chambreau, S.D., J.A. Boatz, G.L. Vaghjiani, J.F. Friedman, N. Eyet, and A.A. Viggiano, *J. Phys. Chem. Lett.* (in press).

9. Ditchfield, R., W.J. Hehre, and J.A. Pople, *J. Chem. Phys.*, 54, 724, 1971; W.J. Hehre, R. Ditchfield, and J.A. Pople, J. *Chem. Phys.*, 1972, 56, pp. 2257–2261, 1972; M.M. Francl, W.J.P ietro, W.J. Hehre, J.S. Binkley, M.S. Gordon, D.J. DeFrees, and J.A. Pople, *J. Chem. Phys.*, 77, pp. 3654–3665, 1982; P.C. Hariharan and J.A. Pople, *Theoret. Chim. Acta*, pp. 28, 213–222, 1973; T. Clark, J. Chandrasekhar, G.W. Spitznagel, and P. von R. Schleyer, *J. Comput. Chem.*, 4, pp. 294–301, 1983.

10. Schmidt, M.W., M.S. Gordon, and J.A. Boatz, *J. Phys. Chem. A*, 109, 7285, 2005; D.D. Zorn, J.A. Boatz, and M.S. Gordon, *J. Phys. Chem. B*, 110, 11110, 2006; I.S.O. Pimienta, S. Elzey, J.A. Boatz, and M.S. Gordon, *J. Phys. Chem. A*, 111, 691, 2007.

11. Feng, S. and G.A. Voth, *Fluid Phase Equilibr.*, 294,148, 2010.

12. Wang, Y. and G.A. Voth, *J. Am. Chem. Soc.*, 127, 12192, 2005.

Energetic Materials: From Potential Energy Surfaces to Crystal Structures

Wojciech Cencek, Fazle Rob, and Krzysztof Szalewicz
Department of Physics and Astronomy, University of Delaware, Newark, DE
{cencek, frob, szalewic}@udel.edu

Rafał Podeszwa
Institute of Chemistry, University of Silesia, Szkolna 9, 40-006 Katowice, Poland, and Department of Physics and Astronomy, University of Delaware, Newark, DE
poszwa@tiger.chem.uw.edu.pl

Betsy M. Rice and DeCarlos Taylor
US Army Research Laboratory (ARL), Aberdeen Proving Ground, MD
{betsy.rice, decarlos.taylor}@us.army.mil

Abstract

Energetic materials have been the subject of intensive experimental studies in national laboratories, and an accurate prediction of their bulk properties is of great importance. Molecular simulations based on empirical potentials (force-fields) tend to be unreliable when applied to properties different from those used to develop these potentials. First-principle methods generate bias-free force-fields, but pose huge computational challenges, since most energetic materials contain fairly large organic molecules that form weakly bound molecular crystals with dominant dispersion interactions.

Generation of sufficiently accurate first-principle potentials became possible only a few years ago, with an advent of novel approaches combining high-accuracy with computational efficiency, such as symmetry-adapted perturbation theory based on density functional theory [SAPT(DFT)] and dispersion-less density functional theory plus dispersion [dlDF+D].

SAPT(DFT) was used to calculate the complete potential energy surface (PES) of the FOX-7 (1,1-diamino-2,2-dinitroethene) dimer, which was then applied in a molecular dynamics study of the solid. The predicted ambient state crystal structure and its thermal/pressure response are in a very good agreement with experiment, despite the fact that no experimental data was used to make these predictions (except for the monomer geometry).

HMX (octahydro-1,3,5,7-tetranitro-1,3,5,7-tetrazocane) contains molecules twice as large as FOX-7 (56 atoms in a dimer), which makes its accurate treatment much more challenging. Results of HMX PES calculations using the dlDF+D method are described.

Finally, we present pilot calculations for CL-12 (octanitrobenzidine), the largest energetic compound (84 atoms in a dimer) studied so far with accurate first-principle methods. Timings and results from both SAPT(DFT) and dlDF+D are compared and perspectives for similar quality studies of still larger systems, including biomolecules, are analyzed.

1. Introduction

Reliable potential energy surfaces (PES) computed from first-principles date back to early 1990s, but until recently calculations of such surfaces remained a niche restricted to dimers with only a few atoms. Although first-principles PESs possess many obvious advantages over empirical PESs, such as an unbiased character, possibility of systematic improvements by increasing the level of theory, etc., the calculations for larger molecules could not be performed due to their prohibitive costs. The costs of any *ab initio* quantum mechanical electronic structure calculations scale very steeply with system size N (the size can be approximately measured by the number of atoms or the number of electrons). In particular, the methods with high predictive power, such as the coupled cluster method with single-, double-, and non-iterated triple-excitations [CCSD(T)], or symmetry-adapted perturbation theory (SAPT), scale as N^7. Thus, the continuous tremendous progress in the computer hardware gets offset very quickly. The ubiquitous density functional theory (DFT) scales much better, as N^4, but lacks the ability to predict correctly the crucial component of intermolecular interactions, the dispersion energy. This weakness of the DFT method is of particular disadvantage in the case of molecular crystals,

where the dispersion energy is often the dominant effect. One example of interest for defense purposes is the large group of energetic compounds (propellants and explosives), usually medium-size (a dozen or so atoms) and large (many dozens of atoms) organic molecules. The availability of high-accuracy predictions of their thermo-physical and chemical properties is highly desirable and would dramatically reduce the development costs.

In the last decade, novel approaches appeared, combining the low cost of DFT with the high-accuracy of more sophisticated methods. Symmetry-adapted perturbation theory with the density-functional theory description of monomers[1–7], abbreviated as SAPT(DFT) or DFT-SAPT, is similar to the regular SAPT approach, but represents the monomers using the Kohn-Sham (KS) DFT approach[8] instead of the much more expensive coupled cluster theory. The crucial dispersion interaction is calculated from the so-called frequency-dependent density susceptibilities (FDDS) (i.e., polarization propagators) of the monomers[4,5], which in turn can be obtained from a coupled KS (time-dependent DFT) calculation. Another key observation was the fact that in order for the SAPT(DFT) predictions to be correct, the density functional used to obtain the KS orbitals of the monomers must be asymptotically corrected. In terms of the computational efficiency, one of the deciding breakthroughs in the development of SAPT(DFT) was the implementation[9–11] of the density-fitting technique[12], which improved the scaling from N^6 to N^5 (N^4 if a nonhybrid DFT method is used and some usually small SAPT corrections neglected), as well as greatly reduced the timing pre-factors and memory requirements. One of the most spectacular successes of SAPT(DFT) was a correct first-principle prediction of the crystal structure of RDX (1,3,5-trinitroperhydro-1,3,5-triazine), an important energetic compound[13–15] with the dimer containing 42 atoms. In the past year, we have introduced several significant improvements to the SAPT(DFT) suite of computer codes, aimed at making large-scale computations more efficient. In particular, we interfaced the perturbation theory codes with the Orca[16] front-end, one of the fastest available DFT packages. As a result, the calculation of the KS orbitals of the monomers represents now an almost negligible part of the whole calculation (see below). Another important factor in choosing this code was the fact that Orca includes the option of using asymptotically-corrected density functionals, necessary for achieving accurate SAPT(DFT) results, through the so-called gradient-regulated asymptotic correction (GRAC)[17]. Although not identical with the Fermi-Amaldi-Tozer-Handy correction[18] used so far in the published work, it gives results of about the same quality and has some formal advantages. We have also developed a new kernel integral code, independent of the DFT front-end. The kernel integrals are necessary to obtain the FDDS and, consequently, the dispersion energy. Since the accuracy of the kernel integrals is much less sensitive to the size of the numerical integration grid than the accuracy of the KS orbitals, it is now possible to use smaller grids for the kernel calculations without any loss of accuracy, but with significant gains in efficiency of calculations.

Another new hybrid method introduced in our group is based on the observation that DFT should be able to quite accurately predict all the components of the interaction energies but the dispersion energy. Consequently, a "dispersion-less" density functional (dlDF) has been developed[19] to reproduce the dispersion-less interaction energies, i.e., the interaction energies with the dispersion components subtracted. The parameters of dlDF were optimized on a training set containing only nine common dimers at various inter-monomer separations. The interaction energies were obtained using the CCSD(T) method, while the dispersion energies were represented by the sum of the second-order dispersion and exchange dispersion energies, $E_{\mathrm{disp}}^{(2)} + E_{\mathrm{exch-disp}}^{(2)}$, calculated with SAPT(DFT). The interaction energies produced by dlDF calculations must, of course, be supplemented with dispersion energies obtained by some other method, forming a hybrid approach known as dlDF+D. Up to some dimer size, one can use the dispersion energies computed using SAPT(DFT). For larger systems, the authors of Reference 19 developed a simple damped asymptotic dispersion function, in the form of a sum of pair (atom-atom) contributions. Its parameters were optimized to reproduce the sum $E_{\mathrm{disp}}^{(2)} + E_{\mathrm{exch-disp}}^{(2)}$ on a training set. The dispersion function was significantly improved in Reference 20 by introducing different parameter values for hydrogen atoms depending on their connectivity and by enlarging the training set. This form of the dlDF+D approach was used in the present work.

2. FOX-7 Calculations

The SAPT(DFT) method was used to calculate the complete PES of the FOX-7 (1,1-diamino-2,2-dinitroethene) dimer. This PES was used in bulk simulations, resulting in a much improved agreement with experiment, compared to previously used potentials. The complete results have been published elsewhere[21]. Here we report an analysis of the PES for radial cross-sections corresponding to several nearest-neighbors in the crystal. Figure 1 presents four molecules belonging to a single-unit cell, which we will refer to as UL, UR, LL, and LR (upper-left, etc.). Note that the molecules in the FOX-7 crystal are chiral, the left ones are mirror images of the right ones. Therefore, two potential energy surfaces had to be computed: RR and RS.

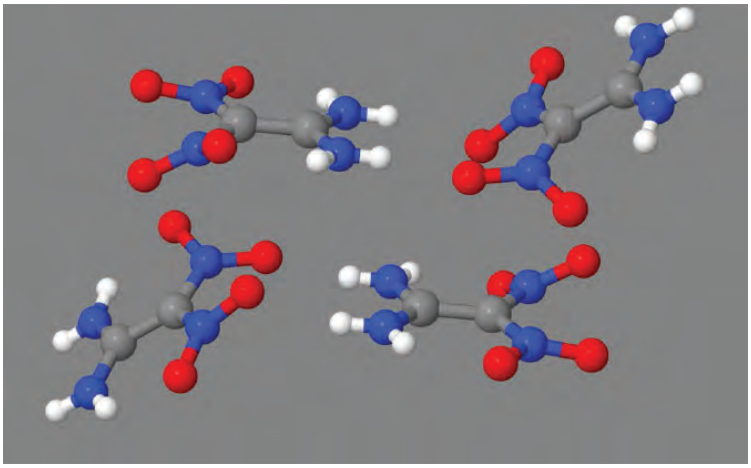


Figure 1. FOX-7 molecules of a single-unit cell. Here, as well as in the other pictures, the following color scheme is used: grey=carbon, blue=nitrogen, red=oxygen, white=hydrogen.

Figure 2 shows the plots of the interaction energies of three different monomer pairs (in kcal/mol) as functions of their center-of-mass distances, for the same relative orientations as present in the crystal. Note that, contrary to the HMX case discussed later, here the plots are calculated from a global analytical function obtained as a fit to a large number of calculated points[21]. The physical picture; however, is very similar in both cases: the actual locations of the monomer pairs are very close to the minima on the radial cross-sections, which means that two-body monomer interactions determine the intermolecular separations in the crystal with high accuracy.

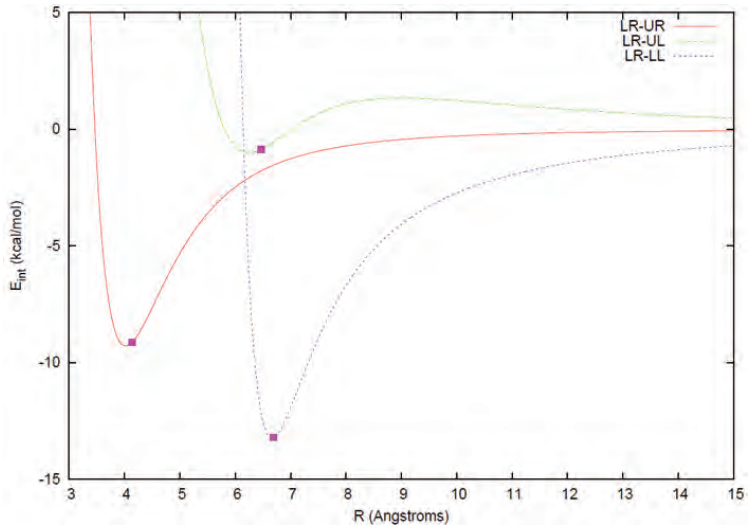


Figure 2. The interaction energies of various FOX-7 dimer configurations as functions of the center-of-mass distance, *R*, obtained from an analytical fit to SAPT(DFT) energies. The squares correspond to the monomer pairs occurring in the crystal. UL, UR, LL, and LR stand for the upper-left etc. molecule in Figure 1.

One should emphasize that none of the three crystal dimer configurations discussed here includes the global minima on the dimer surfaces RR and RS. In both cases, the global minima correspond to stacked configurations which do not appear in the crystal, see Figures 5 and 6 in Reference 21. The stacked RR1 configuration exhibits the largest magnitude of interaction energy amounting to 16.1 kcal/mol, whereas the RS1 value is 15.3 kcal/mol[21]. The LR-LL configuration from the crystal is fairly similar to the second minimum RS2 with the interaction energy of −13.5 kcal/mol[21]. Similarly, the crystal configuration LR-UR is reminiscent of the minimum RR2 (but is about 4 kcal/mol shallower). Despite the fact that the crystal is not built from dimers on the PES minima, the crystal packing, minimization, and molecular dynamics simulations techniques used in Reference 21 correctly predict the crystal structure.

One characteristic property of the FOX-7 crystal is that its pressure response is quite anisotropic, with the highest compression along the b axis. This property is well-rationalized by the curves in Figure 2. The LR-LL interaction,

controlling in the first approximation the compression along the *c* axis, has a much steeper repulsive wall than the LR-UR interaction responsible for the *b* axis compression.

3. HMX Potential Energy Surface

The HMX (octahydro-1,3,5,7-tetranitro-1,3,5,7-tetrazocane) molecule is an eight-membered analogue of the six-membered RDX ring and its dimer has 56 atoms. The bulk properties of HMX crystals were the subject of numerous atomistic simulation papers[22–37]. In a recent study[38], we reported results of a pilot HMX dimer study for a single radial cross-section of the PES for a range of inter-monomer separations. The dlDF+D method and the correlation-consistent augmented double-zeta Dunning basis set (aug-cc-pVDZ, 1,064 basis functions)[39] were used. Subsequently, we have used the same methodology to develop the complete PES of HMX. Of particular interest are results for various dimer configurations occurring in the crystal, which we report in this work.

The form known as β-HMX, stable at ambient conditions, crystallizes in the monoclinic space group $P2_1/n$ and contains molecules in the C_i-symmetry conformation. The unit cell dimensions are: a=6.5209 Å, b=10.7610 Å, c=7.3062 Å, β=102.058°[40]. The lattice is composed of two types of alternating layers (sub-lattices) stretching along the ac plane, depicted in Figures 3 and 4, related to each other by the 2_1 screw rotation along the *b* direction. We will refer to the four molecules in Figure 3 as UL, UR, LL, and LR (upper-left etc.). When superimposed, the two layers are separated by *b*/2 and the molecules of layer 2 are located above the centers of the parallelograms formed by molecules of layer 1. We will refer to the molecule located directly above the center of the parallelogram formed by UL, UR, LL, and LR as C. With this notation, the closest pair of molecules within the whole lattice is UL-UR separated by the a unit cell dimension of 6.52 Å. It is followed by UL-C (6.92 Å), UL-LL (7.31 Å), and UR-C (7.61 Å). All these pairs are characterized by different mutual orientations. Obviously, the inclusion of such pairs in the calculated data set improves the description of the crystal structure predictions by a PES. However, it is also very interesting to analyze the dependence of the dimer interaction energy on the intermolecular separation with frozen mutual orientations. We calculated, therefore, the dlDF+D energies at the following additional separations: 6.00, 6.25, 6.75, 7.70, and 8.00 Å for the UL-UR orientation, 6.65, 7.15, 7.50, and 8.00 Å for the UL-C orientation, 6.80, 7.05, 7.55, and 8.00 Å for the UL-LL orientation, and 7.10, 7.35, and 8.00 Å for the UR-C orientation. Figure 5 contains the calculated points together with interpolating spline curves. Similarly as in the case of FOX-7, it is seen that in three cases the actual separations observed in the crystal correspond to the minimum interaction energies, and in the fourth case (the UL-C pair) the crystal separation is very close to the minimum. This means that the dimer interaction energy alone is the deciding factor in the β-HMX crystal packing and strongly suggests that the three- and higher-body effects are rather small. The lowest interaction energy displayed in Figure 5, equal to −13.7 kcal/mol, is achieved at the closest-pair configuration present in the crystal. Note that this value is much lower than −7.0 kcal/mol previously claimed by us[38] to be the energy at the global minimum. This results from the fact that our computer code for the empirical potential of Sewell[23] contained a mistake and, therefore, the optimization of the dimer geometry performed using this potential did not find the true global minimum (the fifth and seventh columns of Table 2 in Reference 38 are also incorrect). This error is inconsequential for the essential part of Reference 38, which consisted of testing and comparing the performance of the dlDF+D and SAPT(DFT) methods, so that any configuration of the HMX dimer could be chosen. The analytical fit of the complete PES of the HMX dimer will be published in a subsequent paper.

Table 1. The interaction energies (in kcal/mol) of the CL-12 dimer from the SAPT(DFT) calculations in def2-SVP

energy contribution	value
electrostatics	−7.16
exchange	17.86
induction	−8.61
dispersion	−12.57
SAPT(DFT) total	−10.48
dlDF+D, def2-SVP	−9.71
dlDF+D, aug-cc-pVDZ	−8.72

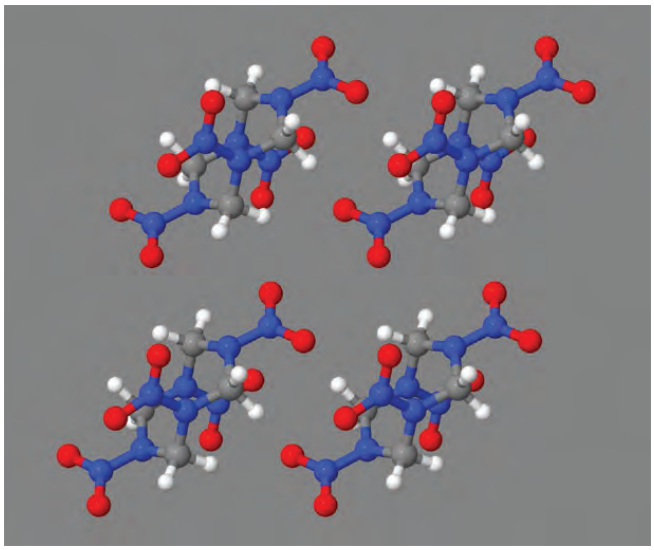


Figure 3. First layer of the *β*-HMX crystal lattice (viewed along the *b* direction). Each of the four molecules is located in the middle of the longest edge (*b*=10.761 Å) of the unit cell.

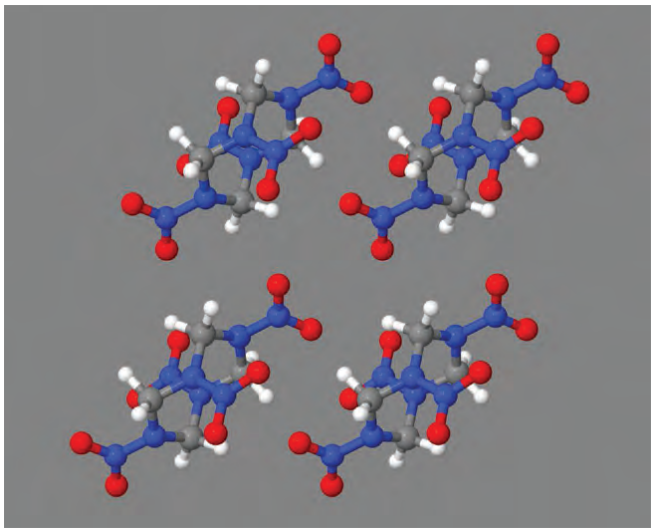


Figure 4. Second layer of the *β*-HMX crystal lattice (viewed along the *b* direction). The four molecules are located in the middles of the *ac* plane sides of the four adjacent unit cells. This layer is related to that depicted in Figure 3 by the 2_1 screw axis symmetry.

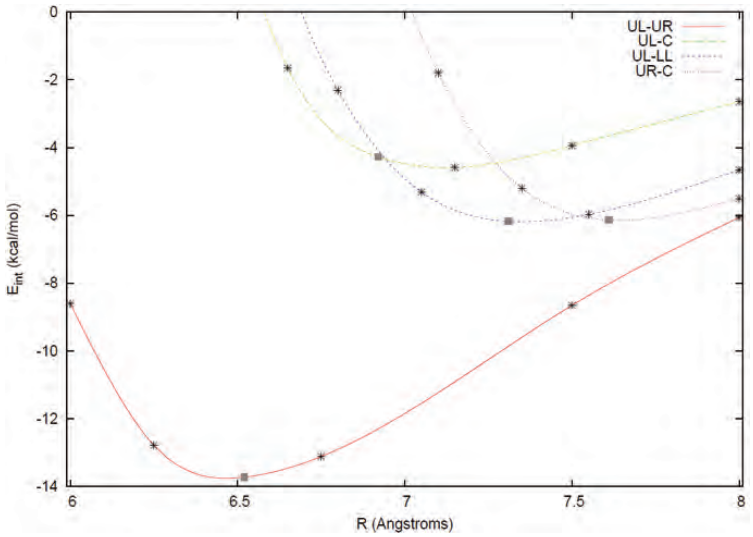


Figure 5. The dIDF+D interaction energies of various HMX dimer configurations as functions of the center-of-mass distance, *R*. The squares correspond to the four closest pairs occurring in the crystal. The stars correspond to configurations with the same mutual monomer orientations as the former, but different inter-monomer separations. See text for the description of the dimer configurations.

Table 2. Timings of the SAPT(DFT) and dIDF+D CL-12 dimer calculation on a single core of the 2.2GHZ Opteron processor. The dIDF+D data include all three components (one dimer and two monomer runs in the dimer-centered basis set).

step	CPU time (hours)
DFT for monomers	4.8
kernel integrals	11.6
auxiliary basis integrals	0.5
integral transformation	27.2
induction	11.6
dispersion (incl. propagators)	165.6
1st-order electrostatics + exchange	0.3
SAPT(DFT) total	221.6
dIDF+D, def2-SVP	13.4
dIDF+D, aug-cc-pVDZ	450.0

4. CL-12 Dimer Calculations

CL-12 (octanitrobenzidine) is a relatively new energetic compound[41] with molecules containing eight nitro groups attached to the benzidine (4,4'-diaminobiphenyl) skeleton. The molecule size (42 atoms) makes it a much more computationally demanding system than any energetic compounds studied before with first-principles methods. Moreover, the molecule contains only a few hydrogen atoms, so its size measured in the number of electrons is even relatively larger.

In the first step, we optimized the monomer geometry using the B3LYP density functional in the 6-31G** basis set. At this geometry, the (vertical) ionization potential (IP) was calculated using B3LYP and a larger, 6-311G** basis set, by subtracting the cation and neutral molecule energies. The obtained result, IP=9.442 eV, was used in the GRAC expression in the subsequent SAPT(DFT) calculations. Since there are, to our knowledge, no studies of the CL-12 dimer in the literature, we selected a random dimer configuration with the center-of-mass distance of 6.72 Å (see Figure 6).

In the SAPT(DFT) calculations, we employed the def2-SVP basis set of Weigend and Ahlrichs[42]. We used the so-called "monomer-centered-plus basis set" (MC$^+$BS) approach, i.e., expanded each monomer's KS molecular orbitals in a basis set consisting of all the functions centered on this monomer, and some of the functions centered on the other monomer.

This is a well-established technique in interaction energy calculations, notable for a good ratio of accuracy to basis set size ("mid-bond" functions, located between both monomers, are also often used in this approach). The core and valence functions (*s* for hydrogen, *s* and *p* for other atoms) of the second monomer were used in the expansion, except for those that were located more than 6 Å from all the atoms of the first monomer. This resulted in a basis set of 808 atomic orbitals for each molecule. In the monomer DFT calculations, the PBE0 functional with GRAC was used. For the subsequent density-fitting SAPT(DFT) calculations, we chose the SVP-type auxiliary basis set optimized by Weigend, et al.[43], centered on both monomers and containing 3,760 orbitals.

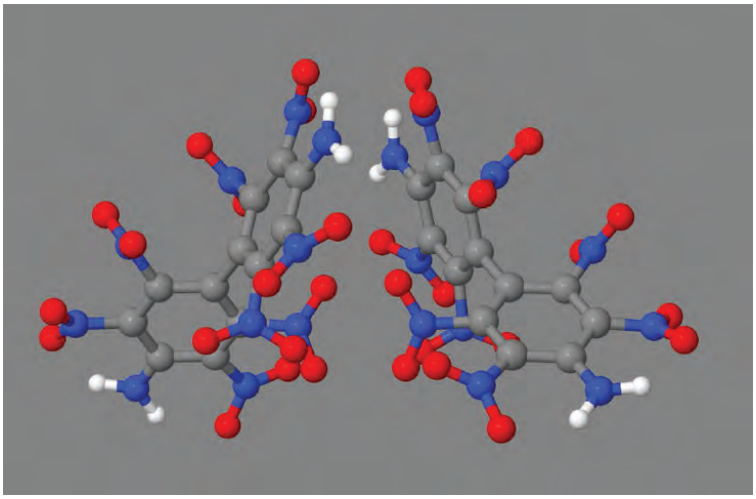


Figure 6. The CL-12 dimer configuration used in the benchmark calculation. The center-of-mass distance between the monomers is 6.72 Å.

The SAPT(DFT) results are summarized in Table 1. For comparison, we also calculated the interaction energy of the same dimer configuration with the dlDF+D method, using both def2-SVP (1,104 orbitals in the full dimer-centered basis set) and the larger, aug-cc-pVDZ basis (1,820 orbitals). Note that, in these pilot SAPT(DFT) calculations, two comparatively small contributions (exchange-induction and exchange-dispersion) were neglected. Table 2 presents timings obtained on a single core of the 2.2GHZ Opteron processor. As it can be seen, the dispersion calculation (including the dispersion propagators for both monomers) dominates the SAPT(DFT) execution time, reaching in this example 75% of the total time. For the same basis set, dlDF+D is much faster than SAPT(DFT) (by a factor of 16 in the case of def2-SVP) but it is expected to provide, in general, a lower level of accuracy. The reason for a dramatic jump of dlDF+D execution time when switching from the def2-SVP to the aug-cc-pVDZ basis set (34 times with a 1.7 times increase of the basis set) is not clear to us at this time (most likely the exchange energy density-fitting procedures are ineffective for the diffuse basis functions present in the aug-cc-pVDZ basis set).

All parts of the calculations listed in Table 2 can benefit from parallel processing. The DFT part of SAPT(DFT) and the whole dlDF calculation (the dispersion portion of the dlDF+D method requires negligible time) scale almost linearly on dozen or even hundreds of processors, depending on the third-party DFT code used. The other modules use BLAS calls and can be run in parallel (up to about 16 cores) if linked with a multithreaded library.

5. Conclusion

Recent progress in the efficiency of our SAPT(DFT) software suite made it possible to increase the range of investigated systems. The 84-atom CL-12 dimer is the largest energetic compound ever calculated at this level of theory. With the computation time of about 10 days (on a single core) per dimer configuration, it is feasible to obtain the complete first-principle PES of this system for use in molecular simulations. Even larger systems can be now routinely investigated using the super-molecular dlDF+D method. However, when possible, SAPT(DFT) should be used for applications requiring particularly high accuracy. Further progress in the SAPT(DFT) applicability to very large molecules requires the optimization of the most time-consuming propagator calculation, preferably by reducing its current N^5 scaling with the system size. We are investigating several ways of achieving this goal, such as calculating the propagators in the atomic (localized) basis sets or density-fitting sets to utilize the sparsity of the resulting matrices.

Acknowledgments

This work was supported by an Army Research Office grant. The DoD HPC Modernization Program supported this project by supplying supercomputer time under the Challenge Project C3W. The calculations were performed on IBM Cluster 1600 system Babbage at the Navy DoD Supercomputing Resource Center (DSRC) and on the Linux Networx Advanced Technology Cluster MJM at the Army Research Laboratory DSRC.

References

1. Misquitta, A.J. and K. Szalewicz, *Chem. Phys. Lett.*, 357, 301, 2002.
2. Hesselmann, A. and G. Jansen, *Chem. Phys. Lett.*, 357, 464, 2002.
3. Hesselmann, A. and G. Jansen, *Chem. Phys. Lett.*, 362, 319, 2002.
4. Misquitta, A.J., B. Jeziorski, and K. Szalewicz, *Phys. Rev. Lett.*, 91, 033201, 2003.
5. Hesselmann, A. and G. Jansen, *Chem. Phys. Lett.*, 367, 778, 2003.
6. Misquitta, A.J. and K. Szalewicz, *J. Chem. Phys.*, 122, 214109, 2005.
7. Misquitta, A.J., R. Podeszwa, B. Jeziorski, and K. Szalewicz, *J. Chem. Phys.*, 123, 214103, 2005.
8. Williams, H.L. and C.F. Chabalowski, *J. Phys. Chem. A*, 105, 646, 2001.
9. Hesselmann, A., G. Jansen, and M. Schütz, *J. Chem. Phys.*, 122, 014103, 2005.
10. Bukowski, R., R. Podeszwa, and K. Szalewicz, *Chem. Phys. Lett.*, 414, 111, 2005.
11. Podeszwa, R., R. Bukowski, and K. Szalewicz, *J. Chem. Theory Comput.*, 2, 400, 2006.
12. Dunlap, B.I., J.W.D. Connolly, and J.R. Sabin, *J. Chem. Phys.*, 71, 4993, 1979.
13. Podeszwa, R., B.M. Rice, and K. Szalewicz, *Phys. Chem. Chem. Phys.*, 9, 5561, 2007.
14. Podeszwa, R., B.M. Rice, and K. Szalewicz, *Phys. Rev. Lett.*, 101, 115503, 2008.
15. Podeszwa, R., B.M. Rice, and K. Szalewicz, *Phys. Chem. Chem. Phys.*, 11, 5512, 2009.
16. Neese, F., *ORCA*, An Ab Initio, DFT and Semiempirical electronic structure package, with contributions from U. Becker, D. Ganyushin, A. Hansen, D. Liakos, C. Kollmar, S. Kossmann, T. Petrenko, C. Reimann, C. Riplinger, K. Sivalingam, B. Wezisla, and F. Wennmohs.
17. Grüning, M., O.V. Gritsenko, S.J.A. van Gisbergen, and E.J. Baerends, *J. Chem. Phys.*, 114, 652, 2001.
18. Tozer, D.J. and N.C. Handy, *J. Chem. Phys.*, 109, 10180, 1998.
19. Pernal, K., R. Podeszwa, K. Patkowski, and K. Szalewicz, *Phys. Rev. Lett.*, 103, 263201, 2009.
20. Podeszwa, R., K. Pernal, K. Patkowski, and K. Szalewicz, J. *Phys. Chem. Lett.*, 1, 550, 2010.
21. Taylor, D.E., F. Rob, B.M. Rice, R. Podeszwa, and K. Szalewicz, *Phys. Chem. Chem. Phys.*, 13,16629, 2011.
22. Dzyabchenko, A.V., T.S. Pivina, and E.A. Arnautova, *J. Mol. Struct.*, 67, 378, 1996.
23. Sewell, T.D., *J. Appl. Phys.*, 83, 4142, 1998.
24. Sorescu, D.C., B.M. Rice, and D.L. Thompson, *J. Phys. Chem. B*, 102, 6692, 1998.
25. Sorescu, D.C., B.M. Rice, and D.L. Thompson, *J. Phys. Chem. B*, 103, 6783, 1999.
26. Lewis, J.P., T.D. Sewell, R.B. Evans, and G.A. Voth, *J. Phys. Chem. B*, 104, 1009, 2000.
27. Bedrov, D., G.D. Smith, and T.D. Sewell, *J. Chem. Phys.*, 112, 7203, 2000.
28. Bedrov, D., G.D. Smith, and T.D. Sewell, *Chem. Phys. Lett.*, 324, 64, 2000.
29. Bedrov, D., C. Ayyagari, G.D. Smith, T.D. Sewell, R. Menikoff, and J.M. Zaug, *J. Comput.-Aided Mater. Des.*, 8, 77, 2002.
30. Sewell, T.D., R. Menikoff, D. Bedrov, and G.D. Smith, *J. Chem. Phys.*, 119, 7417, 2003.
31. Zerilli, F.J. and M.M. Kuklja, *J. Phys. Chem. A*, 110, 5173, 2006.
32. Byrd, E.F.C. and B.M. Rice, *J. Phys. Chem. C*, 111, 2787, 2007.
33. Xiao, J.J., H. Huang, J.S. Li, H. Zhang, W. Zhu, and H.M. Xiao, *Acta Chim. Sinica*, 65, 1746, 2007.
34. Lian, D., L.Y. Lu, D.Q. Wei, Q.M. Zhang, Z.Z. Gong, and Y.X. Guo, *Chin. Phys. Lett.*, 25, 899, 2008.
35. Conroy, M.W., I.I. Oleynik II, S.V. Zybin, and C.T. White, *J. Appl. Phys.*, 104, 053506, 2008.
36. Zhu, W.H., X.W. Zhang, T. Wei, and H. Xiao, *Theor. Chem. Acc.*, 124, 179, 2009.
37. Duan, X.H., C.X. Wei, Y.G. Liu, and C.H. Pei, *J. Hazard. Mater.*, 174, 175, 2010.

38. Cencek, W., F. Rob, K. Szalewicz, R. Podeszwa, B.M. Rice, and D.C. Taylor, *HPCMP User Group Conference Proceedings*, Schaumburg, Il, p. 208, 2010.

39. Kendall, R.A., T.H. Dunning Jr., and R.J. Harrison, *J. Chem. Phys.*, 96, 6796, 1992.

40. Zhurova, E.A., V.V. Zhurov, and A.A. Pinkerton, *J. Am. Chem. Soc.*, 129, 13887, 2007.

41. Nielsen, A.T., R.L. Atkins, and W.P. Norris, United States Patent No. 4 935 544, June 19, 1990.

42. Weigend, F. and R. Ahlrichs, *Phys. Chem. Chem. Phys.*, 7, 3297, 2005.

43. Weigend, F., M. Häser, H. Patzelt, and R. Ahlrichs, *Chem. Phys. Lett.*, 294, 143, 1998.

Environmental Fate and Transport of Energetic Materials: Alternative Compounds and Alternative Soils

Margaret Hurley
US Army Research Laboratory, Aberdeen Proving Ground, MD
margaret.hurley@us.army.mil

Abstract

This project strives to improve our understanding of the behavior of energetic materials (EM) in soil and water resources to facilitate development of improved remediation strategies. Periodic density functional calculations are used to study the adsorption of energetic materials on soil mineral models. Earlier work focused on the adsorption of conventional energetic materials on a model kaolinite surface, and utilized periodic Density Functional Theory (DFT) both with and without dispersion corrections. In light of the lack of firm data characterizing soil sorption of novel energetic materials, we study the compound metronidazole as a feasible chemical simulant. In addition, we extend previous work on the adsorption of trinitrotoluene (TNT) to the (001) pyrophyllite surface, an alternative compound within the clay mineral group and a complement to the kaolinite surface used in previous models.

1. Introduction

The question of environmental remediation of firing ranges, storage depots, and munitions plants in the US and abroad is of international interest [Poulin, 2009; Spiegel, 2005; Clausen, 2005], and is intimately connected to details of the interactions of energetic materials with the soil medium. The behavior of RDX, or Hexahydro-1,3,5-trinitro-1,3,5-triazine (CAS 121-82-4), in various soil types has been a topic of considerable interest experimentally and computationally due both to its wide usage in both civilian and military applications and its persistence in the environment [Talmage, 1999]. While newer energetic compounds are emerging [Steinhauser and Klapötke, 2008] which are posited to possess a minimal environmental footprint, little data is available to confirm their degree of "greenness" or to clarify their behavior in the environment.

Accordingly, computational studies have been undertaken to probe the details of the interaction of energetic materials with soil models [Hurley and Paul, 2009] and with water [Hurley, 2010], as well as to predict breakdown products of novel energetic compounds [Paul, Irikura, and Hurley, 2009]. These studies have used a variety of computational methods from ab initio molecular dynamics to the novel Isopotential Searching algorithms [Irikura, 2006]. Additionally this work ha demonstrated some of the advantages and drawbacks of the new dispersion corrected atom centered potentials (DCACPS) [Lin, 2007] as applied to the adsorption of energetic materials (EM) (RDX, TNT, TNB) on the kaolinite (001) surface as a model soil [Paul and Hurley, 2011; Hurley and Paul, 2011]. The current focus is to apply these methods to the analysis of alternative energetic compounds and more varied soil models.

Unfortunately, very little detailed information is available on the soil sorption properties of novel energetic materials. However, within the literature there exist chemically similar compounds which are used in the pharmaceutical industry, compounds whose environmental presence is beginning to assert itself. For example, the sorption and mobility of metronidazole (2-(2-methyl-5-nitro-1*H*-imidazol-1-yl) ethanol CAS 443-48-1) in soil has been studied in sufficient detail to provide a useful basis of comparison [Rabolle 2000]. Metronidazole (Figure 1) is a nitroimidazole based antibiotic, antiprotozoal, and amebicide, which has therapeutic applications both human and veterinary. The Hazardous Substances Data Bank [HSDB] has flagged metronidazole as having a very high mobility in soil based upon an *estimated* soil sorption coefficient (Koc) of 23. Measured values of Koc for metronidazole are somewhat higher and range between 38 and 56 [Rabolle, 2000]. For comparison, the reported Koc for RDX covers a higher but overlapping range (42 to 167) [Spanggord, 1980] and is listed as having moderate to high soil mobility.

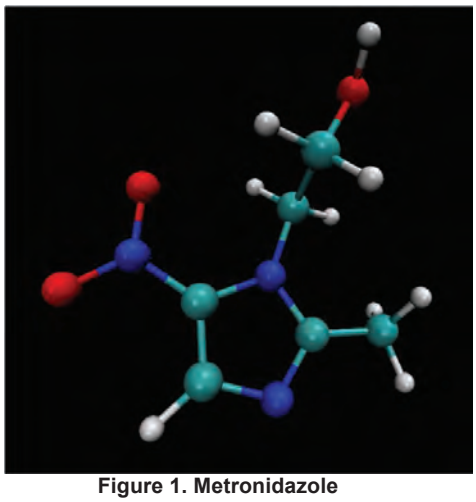


Figure 1. Metronidazole

We use periodic density functional theory (DFT) to analyze the binding energy of metronidazole on the (001) surface of kaolinite for the purpose of comparison with previous work studying the binding of water, TNT, TNB, and RDX on this model soil mineral surface.

In addition to the kaolinite model, it was decided to include models of alternative soil minerals within the clay group, notably pyrophyllite. Pyrophyllite is a dioctahedral 2:1 phyllosilicate with the formula $AlSi_2O_5OH$. It consists of an aluminum-containing octahedral sheet sandwiched between two Si-containing tetrahedral sheets. This is shown in Figure 2, where labeling mirrors that of Jordan, et al. [Voora, 2011]. Pyrophyllite has a number of uses, including as a component of ceramics, as a refractory material, as a filler material (e.g., rubber), and it is believed to play an important role in geological processes as a pressure-transfer medium. For these reasons it has received a great deal of experimental and computational interest within the last few years. Binding of TNT to the ideal (001) pyrophyllite surface was studied through the use of periodic DFT, for comparison with previous results of binding to the kaolinite surface.

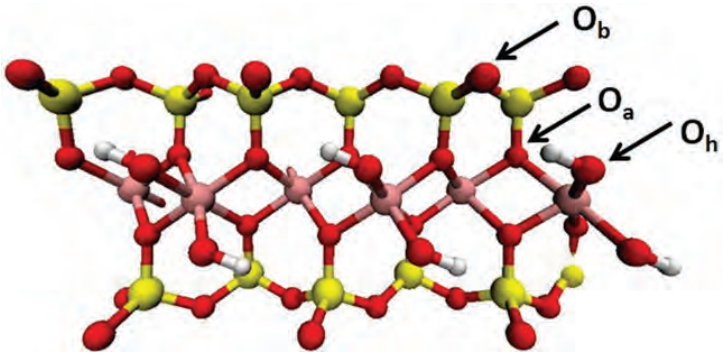


Figure 2. Pyrophyllite model. The color scheme is O(red), Si(yellow), Al(pink), and H(white). Oxygen labeling follows that of Jordan.

2. Methodology

A structural model for pyrophyllite was built based upon structural models of Tsipursky and Drits [Tsipursky and Drits, 1984]. The computational cell was built containing 240 atoms ($Al_{24}Si_{48}O_{120}(OH)_{24}$) in a 3×2 slab with lattice parameters *a*=15.4872 Å, *b*=17.9592 Å, *c*=25.0 Å, and β=93.641°. This corresponded to approximately 18 Å of vacuum. The single unit cell parameters therefore corresponded to *a*=5.1624 Å, *b*=8.9796. This is in close agreement with the experimental values of *a*=5.160(2)Å and *b*=8.966(3)Å [Lee and Guggenheim, 1981], as well as recent theoretical results of 5.15 and 8.98 [Voora, 2011] Periodic Density Functional Theory (DFT) calculations were performed with the CPMD code using norm-conserving Troullier-Martins pseudopotentials. The exchange-correlation functionals used were PBE and BLYP, as these functionals were successfully used for previous work studying the binding of NACs on kaolinite. In the BLYP calculations, dispersion corrected atom-centered pseudopotentials (DCACPs) were used for H, C, N, and O as noted in

the results. DCACPs were not used for the PBE calculations, as our previous work indicated that these corrections had a greater impact with results obtained with the BLYP functional. The charge density cutoff was set to 340Ry, and the plane-wave kinetic energy cutoff to 85 Ry, again as in previous work. Γ-point sampling was used due to the relatively large cell size. Geometry optimizations were performed with a wavefunction gradient convergence criterion of 1×10^{-7} and an ionic force gradient convergence criterion of 5×10^{-4}. The cell was fixed and no constraints were applied to any atoic coordinates. Additional work (Paul and Hurley, to be published) has been performed to investigate four different binding sites on the (001) pyrophyllite surface. Here all optimizations are begun with the TNT adsorbate centered directly above the ditrigonal siloxane cavity, as this was seen to be energetically most favorable, both in preliminary work, and in the related kaolinite tetrahedral surface.

The kaolinite model used has been described in detail elsewhere [Paul and Hurley, 2011] [Hurley and Paul, 2011], and is roughly comparable in size (a=5.1476 Å, b=8.9386 Å, c=14.390 Å) to the pyrophyllite model. The octahedral surface was chosen for this binding study due to the important role in binding played by the surface hydroxyl groups. We wished to probe the interplay of this system with the metronidazole hydroxyl moiety for comparison with water binding, as previously investigated. All other simulation details were as described above. Visualization and analysis for all simulations was performed with VMD [Humphrey, 1996].

3. Results and Conclusions

Previous work within this project [Kaul and Hurley, 2011] has demonstrated the utility of the dispersion-corrected potentials (DCACPs) of Lin, et al. [Lin, et al.] in studying EMs in model soil. While the dispersion correction had a large effect on the adsorbate-surface interaction, the effect on the structure of the soil model itself was minor. A recent study by Voora, et al. [2011] applied the DFT-D2 dispersion correction method of Grimme, et al. to bulk pyrophyllite and substituted montmorillionites, and also found the effect of the dispersion correction on the pyrophyllite structure was relatively minor. Preparatory to carrying out the binding studies, the (001) pyrophyllite surface was optimized alone with the PBE and BLYP functionals, as well as with the BLYP functional with the dispersion correction (DCACPs). Table 1 below gives average bond lengths for the model (where the labeling follows that of Figure 2). These results are well in accordance both with previous theory, and with experiment. We note that agreement is excellent with both functionals, and that the dispersion correction does not markedly affect the pyrophyllite surface structure. It is also interesting to note that there is little to no surface relaxation in this system relative to the bulk, undoubtedly due to the compact sheet-like structure.

Table 1. Average bond lengths of optimized (001) pyrophyllite model

	PBE (current work)	BLYP (current work)	BLYP+DCACP (current work)	Expt [Lee, 1981]	DFT PBE [Voora, 2011]	DFT-D2 [Voora, 2011]	CLAYFF [Larentzos, 2007]	DFT PW91 [Larentzos, 2007]
HOh	0.971	0.971	0.971	0.978	0.970	0.971	1.03	0.97
SiOb	1.625	1.627	1.630	1.615	1.628	1.627	1.56	1.62
SiOa	1.650	1.656	1.656	1.614	1.652	1.650	1.63	1.63
AlOh	1.900	1.902	1.903	1.897	1.906	1.898	1.98	1.89
AlOa	1.935	1.938	1.939	1.927	1.941	1.935	1.98	1.92

Once the surface was optimized, the TNT molecule was added to the model and further optimization performed with the BLYP functional, both with and without the DCACP. The resulting structure is shown in Figure 3, both from above the surface (Figure 3a) and from the side (Figure 3b). Note that only the top layer of the model is shown in Figure 3a to improve visual clarity.

As can be seen in Figure 3, in the final optimized configuration the TNT remains oriented symmetrically within the cavity (Figure 3a), although it is slightly tilted with respect to the surface normal (Figure 3b). Here again the DCACP is seen to change the adsorption process quantitatively but not qualitatively. The distance from the TNT ring carbons to the nearest surface oxygen vary from 4.04 Å to 4.45 Å for the system studied at BLYP without DCACP. With the addition of the DCACP these distances are shortened to 3.27 Å and 3.74 Å. This is in excellent agreement with the vertical distance of 4.0 Å (BLYP) and 3.3 Å (BLYP+DCACP) found in our analogous TNB/tetrahedral kaolinite surface work [Paul and Hurley 2011]. With use of the PBE functional, these distances ranged from 3.50 Å to 3.94 Å. This is somewhat shorter than the 4.0 Å vertical distance found with PBE for the analogous TNB/tetrahedral kaolinite surface. Also, for the purposes

of comparison with previous theory, we note the work of Alzate et al [Alzate, 2006], who studied the binding of TNT to a siloxane cluster at MP2 and B3LYP with a variety of basis sets (6-31G to 6-311+G(d)) , who also note the importance of dispersion in this binding reaction. While the siloxane cluster was posited as a model for the tetrahedral surface of kaolinite, it is sufficiently truncated for application to the pyrophyllite surface as well. The vertical distance between TNT and the siloxane surface in the Alzate work was calculated to be 3.5 Å.

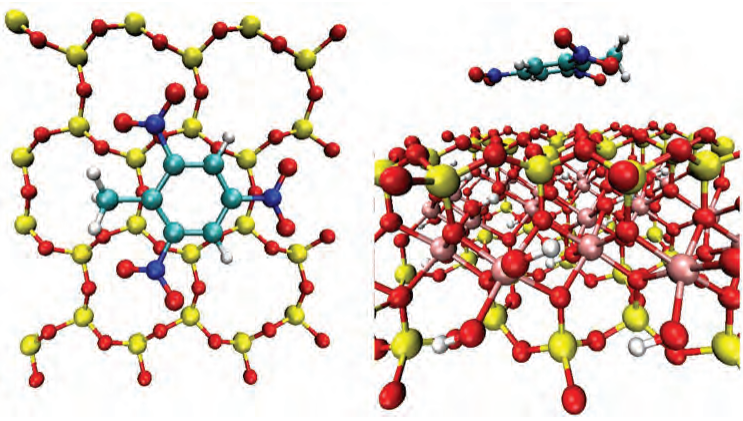


Figure 3. TNT adsorbed to (001) pyrophyllite surface, top (3a) and side (3b) view. The color scheme is O(red), Si(yellow), Al(pink), H(white), and C(cyan). Only the surface layer is visualized in 3a for clarity.

Binding energies for the TNT/pyrophyllite surface follow a similar trend. With the PBE functional, a binding energy of -3.17 kJ/mol is found. When the BLYP functional is used without the DCACP, binding is not energetically favorable with a ΔE of +7.07 kJ/mol. Inclusion of the dispersion correction lowers this dramatically to -26.09kJ/mol. This is markedly similar, both qualitatively and quantitatively, to the binding of TNT to the tetrahedral surface of kaolinite [Paul and Hurley, 2011], where the binding energy was found to be -6.1 kJ/mol (PBE), +3.1 kJ/mol (BLYP), and -30.0 kJ/mol (BLYP+DCACP). These results further confirm the large effect of dispersion on binding in these systems. The most analogous abbreviated model of Alzate, et al. [Alzate, 2006] found a binding energy of -28 kJ/mol at the B3LYP//HF/6-31G level, which decreased to -10 kJ/mol after correcting for BSSE.

In contrast to TNT, the binding of metronidazole to the (001) octahedral kaolinite surface involves more explicit interplay of functional groups, as is to be expected for this overtly hydroxylated surface. The kaolinite model utilized has been described in detail elsewhere [Hurley and Paul, 2011]. The BLYP functional with DCACP was used for this work, as this combination was found to be most efficacious in previous studies of the binding of water [Hurley and Paul, 2011] and RDX [Paul and Hurley, 2011]. The optimized structure is shown in Figure 4, where again the top-view image shows only the surface layer for visual clarity.

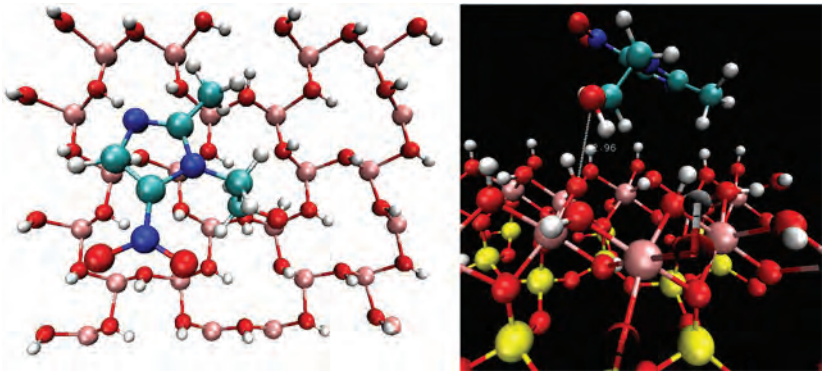


Figure 4. Top (4a) and side (4b) view of the binding of metronidazole to the (001) kaolinite tetrahedral surface. The color scheme is O(red), Si(yellow), Al(pink), H(white), and C(cyan). The metronidazole oxygen—kaolinite surface oxygen distance is noted as 2.96 Å. Only the surface layer is visualized in 4a for clarity.

The binding proceeds as expected primarily through a metronidazole hydroxyl group/kaolinite hydroxyl group interaction. The oxygen-oxygen distance for this complex is 2.96 Å, denoting a relatively weak hydrogen bond. The kaolinite binding site of interest is the surface hydroxyl, rather than the inner hydroxyl group previously studied for the binding of H_2O (where the oxygen-oxygen distance of the optimized complex was 2.70 Å).

Very little previous theoretical work has been performed on metronidazole. Ramalho and Taft [Ramalho, 2005] published a study which focused on the effects of solvation on NMR and UV parameters, utilizing G03 [G03] at the B3LYP/6-31G* level, as well as performing *ab initio* molecular dynamics of metronidazole in a water box with the CPMD program and the BP86 functional. The basic structure of metronidazole obtained in the current work is in very reasonable agreement with this previous theory, as well as with experiment [as cited within Ramalho]. Calculated atomic distances around the imidazole ring vary by less than 0.01 Å, regardless of theoretical method used (B3LYP in G03 as per Ramalho or BP86 in CPMD as in Ramalho and PBE in CPMD the current work), regardless of the solvation state (gas phase or PCM in Ramalho, explicit water in Ramalho, or gas phase in the current work), and regardless of absorption state (surface-complexed or not, in the current work). Unfortunately the most pertinent structural feature for our immediate model is the hydroxyl group OH distance, which is not quantified in the previous work [Ramalho, 2005]. We note that in the current work, the OH distance is 0.974 Å in the unbound system. Upon binding, this distance is marginally longer at 0.979 Å.

The binding energy of metronidazole on the (001) octahedral kaolinite surface is predicted to be -45.4 kJ/mol. This is slightly weaker than the octahedral surface binding result for both RDX (-60.9 kJ/mol [Paul and Hurley, 2011] and for water (-52.0 kJ/mol [Hurley and Paul, 2011]). While there are many factors involved in soil sorption, and the soil mineral component represents only a portion of this, it is interesting to note that the small size and weaker binding of metronidazole to the kaolinite (relative to the RDX/kaolinite system) are in accordance with the higher soil mobility of metronidazole which has been measured experimentally[Rabolle, 2000].

Periodic DFT has been used to successfully delineate the atomistic level interactions which contribute to the environmental behavior of existing EMs and novel stimulants. Dispersion corrections have been successfully applied to relatively weak binding systems (e.g., TNT on pyrophyllite), and the chemistry of the soil model (tetrahedral face of pyrophyllite vs octahedral face of kaolinite) has been seen to play a role in these interactions. This provides a partial explanation for the wide variance on measured soil sorption values depending on the nature of the soil sample studied. Another factor, of course, is the organic carbon content, which has been studied separately within the course of this project and will be reported elsewhere.

4. Significance to DoD

This project strives to improve our understanding of the environmental behavior of EM in soils and water resources. Energetic materials such as HMX, RDX, and TNT are present in nontrivial concentrations at military installations and have received a considerable amount of public scrutiny. A thorough understanding of how these EM interact with clay surfaces and water is expected to ultimately lead to improved remediation strategies. Furthermore, by determining how to effectively characterize these complex systems with QC calculations, the DoD will be better prepared to characterize future replacement EM with well-established QC methods.

Acknowledgments

The author wishes to thank the US Army Environmental Quality Technology Program for financial support and Kristian Paul of DuPont for useful discussion and preliminary models.

References

Alzate, L.F, C.M. Ramos, N.M. Hernandez, S.P. Hernandez, and N. Mina, "The vibrational spectroscopic signature of TNT in clay minerals", *Vibrational Spectroscopy*, 42, pp. 357–368, 2006.

Clausen, J.L., "Range Assessment Lessons Learned", *Federal Facilities Environmental Journal*, Summer, pp. 49–62, 2005.

CPMD, http://www.cpmd.org/, Copyright IBM Corp. 1990–2008, Copyright MPI fur Festkorperforschung Stuttgart 1997–2001, version 3.13.2.

Frisch, M.J., Trucks, G.W., Schlegel, H.B., Scuseria, G.E., Robb, M.A., Cheeseman, J.R., Montgomery, Jr., J.A., Vreven, T., Kudin, K.N., Burant, J.C., Millam, J.M., Iyengar, S.S., Tomasi, J., Barone, V., Mennucci, B., Cossi, M., Scalmani, G., Rega, N., Petersson, G.A., Nakatsuji, H., Hada, M., Ehara, M., Toyota, K., Fukuda, R., Hasegawa, J., Ishida, M., Nakajima, T., Honda, Y., Kitao, O., Nakai, H., Klene, M., Li, X., Knox, J. E., Hratchian, H P., Cross, J.B., Bakken, V., Adamo, C., Jaramillo, J., Gomperts, R., Stratmann, R.E., Yazyev, O., Austin, A.J., Cammi, R., Pomelli, C., Ochterski, J.W., Ayala, P.Y., Morokuma, K., Voth, G.A., Salvador, P., Dannenberg, J.J.,

Zakrzewski, V.G., Dapprich, S., Daniels, A.D., Strain, M.C., Farkas, O., Malick, D.K., Rabuck, A.D., Raghavachari, K., Foresman, J.B., Ortiz, J.V., Cui, Q., Baboul, A.G., Clifford, S., Cioslowski, J., Stefanov, B.B., Liu, G., Liashenko, A., Piskorz, P., Komaromi, I., Martin, R.L., Fox, D.J., Keith, T., Al-Laham, M.A., Peng, C.Y., Nanayakkara, A., Challacombe, M., Gill, P.M.W., Johnson, B., Chen, W., Wong, M.W., Gonzalez, C., and Pople, J.A., *Gaussian 03*, Gaussian, Inc., Wallingford, CT, 2004.

Hazardous Substances Data Bank [Internet], National Library of Medicine (US), Bethesda, MD, last revision date: 2011 Feb 4, cited 2011 May 12, Metronidazole; Hazardous Substances Databank Number: 443-48-1, available from http://toxnet.nlm.nih.gov/cgi-bin/sis/htmlgen?HSDB.

Humphrey, W., A. Dalke, and K. Schulten, "VMD - Visual Molecular Dynamics", *J. Molec. Graphics*, 14, pp. 33–38, 1996.

Hurley, M.M., "Environmental Fate and Transport of Energetic Materials: the Aqueous Environment", *Proceedings of the DoD HPC Users Group Conference 2010*, Chicago, IL, 2010.

Hurley, M.M. and K.W. Paul, "Environmental Fate and Transport of Energetic Materials", *Proceedings of the DoD HPC Users Group Conference 2009*, San Diego, CA, 2009.

Hurley, M.M. and K.W. Paul, "Adsorption of Water on the (001) Surfaces of Kaolinite: A Periodic Density Functional Theory Study Incorporating Dispersion-Corrected Atom-Centered Pseudopotentials", *Phys Chem Chem Phys*, 2011 (submitted).

Irikura, K.K. and R.D. Johnson III, "Is NO3 Formed during the Decomposition of Nitramine Explosives?" *J. Phys. Chem A*, 110, pp. 13974–13978, 2006.

Larentzos, J.P., J.A. Greathouse, and R.T. Cygan, "An ab Initio abd Classical Molecular Dynamics Investigation of the Structural and Vibrational Properties of Talc and Pyrophyllite", *J. Phys. Chem C*, 111, pp. 12752–12759, 2007.

Lee, J.H. and S. Guggenheim, "Single crystal X-ray refinement of pyrophyllite-1Tc", *American Mineralogist*, 66, pp. 350–357, 1981.

Lin, I.C., et al., "Library of dispersion-corrected atom-centered potentials for generalized gradient approximation functionals: Elements H, C, N, O, He, Ne, Ar, and Kr", *Physical Review B*, 75, pp. 205131–5, 2007.

Paul, K.W. and M.M. Hurley, "Adsorption of Energetic Materials on the (001) Surfaces of Kaolinite: A Periodic Density Functional Theory Study Incorporating Dispersion-Corrected Atom-Centered Pseudopotentials", *Phys Chem Chem Phys*., 2011 (submitted)

Paul, K.W, M.M. Hurley, and K.K. Irikura, "Unimolecular Decomposition of 5-Aminotetrazole and its Tautomer 5-Iminotetrazole: New Insight from Isopotential Searching", *J. Phys. Chem A*, 11, pp. 2482-2490, 2009.

Poulin, I., Nadeau, G., and A. Gagnon, "Development of a remediation strategy for surface soils contaminated with energetic materials by thermal processes: Phases 1, 2 and 3", *DRDC Valcartier TR 2009-150*.

Rabolle, M. and N.H. Spliid, "Sorption and mobility of metronidazole, olaquindox, oxytetracycline, and tylosin in soil", *Chemosphere*, 40, pp. 715–722, 2000.

Ramalho, T. and C.A. Taft, "Thermal and solvent effects on the NMR and UV parameters of some bioreductive drugs", *Journal of Chemical Physics*, 123, 054319, 2005.

Spanggord, R.J., T. Mill, T.-W. Chou, W.R. Mabey, and J.H. Smith, "Environmental Fate Studies on Certain Munition Wastewater Constituents – Laboratory Studies", *Document no. AD A099256*, U.S. Army Medical Research and Development Command, Ft. Detrick, MD, 1980.

Spiegel, K., et al., "Residues of Explosives in Groundwater Leached from Soils at a Military Site in Eastern Germany", *Communications in Soil Science and Plant Analysis*, 36, pp. 133–153, 2005.

Steinhauser, G. and T.M. Klapotke, "Green' Pyrotechnics: A Chemists' Challenge", *Angewandte Chemie Int. Ed.*, 47, pp. 2–20, 2008.

Talmage, S.S., D.M. Opresko, C.J. Maxwell, C.J.E. Welsh, F.M. Cretella, P.H. Reno, and F.B. Daniel, "Nitroaromatic Munitions Compounds: Environmental Effects and Screening Values", *Review of Environmental Contamination and Toxicology*, 161, pp. 1–156, 1999.

Tsipursky, S.I. and V.A. Drits, "The distribution of octahedral cations in the 2:1 layers of dioctahedral smectites studied by oblique-texture electron diffraction", *Clay Minerals*, 19, pp. 177–193, 1984.

Voora, K.V., W.A Al-Saidi, and K.D. Jordan, "Density Functional Theory Study of Pyrophyllite and M-Montmorillonites (M=Li, Na, K, Mg, and Ca); Role of Dispersion Interactions", *J. Phys. Chem. A*, Articles ASAP Web Publication, April 26, 2011.

Modeling the Combustion Chamber Dynamics of Two Selectable-Thrust Rocket Motor Concepts

Michael J. Nusca, Chiung-Chu Chen, and Michael J. McQuaid
US Army Research Laboratory, Weapons and Materials Research Directorate (ARL/WMRD), Aberdeen Proving Ground, MD
{michael.j.nusca, chiungchu.chen, michael.j.mcquaid}@us.army.mil)

Abstract

Challenge Project C4C is dedicated to developing and applying high performance computing capabilities to accelerate the development of hypergolic and hybrid rocket motor concepts. Computational fluid dynamics is employed to model chemically-reacting flows within motor combustion chambers, and computational chemistry is employed to characterize propellant physical and reactive properties. Recent accomplishments are presented and discussed.

1. Introduction

The US Army is developing hypergolic bipropellant (liquids and gels) and hybrid (liquids and solids) rocket motor concepts for tactical missile propulsion systems[1–3]. Hypergolic propulsion systems employ a pumpable (i.e., liquid or gel) fuel and an oxidizer that ignites spontaneously when mixed at ambient temperatures and pressures, and hybrid propulsion systems employ a pumpable oxidizer with a solid fuel grain. Both systems have many potential advantages over the solid-propellant-based rocket motors currently employed to propel tactical missiles. Among them is an active thrust control, which increases targeting options and range, also increasing a tactical missile's lethality while reducing the vulnerability of its launch platform. The fuel and oxidizer are stored separately in such systems, and are thus inherently insensitive to a variety of stimuli that can produce catastrophic events in solid-propellant-fueled rocket motors. There are, however, a number of technical issues that must be addressed before these technologies can be fielded.

The liquid bipropellant hypergolic propulsion system concept that the Army is developing for tactical missile applications is referred to as the Impinging Stream Vortex Engine (ISVE)[1,2]. In the ISVE, the fuel and oxidizer injection ports are located in the radial wall of the combustion chamber, and they are oriented so that initially the fluids flow tangentially to it. This configuration forces the propellants to mix in a highly-turbulent region between the injector orifices and the wall, promoting both an increase in the mixing path length and a lower wall-temperature. Facilitating the design of smaller, lighter propulsion systems, these benefits have been observed in experimental tests of the concept conducted by the US Army Aviation and Missile Research, Development, and Engineering Center (AMRDEC). However, the complexity of the interplay between design variables and engine performance make engine optimization based solely on experimental testing problematic. Propellant physical properties and reactivity being among the variables with a significant influence on engine design and performance, the selection and development of a fuel-oxidizer (bipropellant) combination to use in the application has been an important and integral aspect of the ISVE development effort. Until seven years ago, the combination employed in AMRDEC's test program was monomethylhydrazine-red fuming nitric acid (MMH-RFNA)[4]. (RFNA is composed primarily of nitric acid [HNO_3] and nitrogen dioxide [NO_2].) However, the superior performance attributes of MMH became outweighed by the risk its carcinogenic potential poses to human health, causing the Army to screen alternatives. The two classes of compounds considered to have the best prospects for meeting desired performance objectives while having risks to human health and the environment that are acceptable are saturated, alkylamines and ethanamine azides[5]. Candidates that are representative of these two classes are, respectively, N,N,N',N'-tetramethylethylene-1,2-diamine (TMEDA) and 2-azido-N,N-dimethylethanamine (DMAZ), and a TMEDA-DMAZ blend referred to as TEDMAZ is the current candidate of choice.

In addition to the ISVE, AMRDEC is developing a novel hybrid rocket motor concept employing RFNA as an oxidizer with a fuel grain formulated with hydroxyl-terminated polybutadiene (HTPB, $C_{7.1102}H_{10.813}N_{0.1071}O_{0.1375}$)[3]. Certain attributes this combination contains include hypergolicity sufficient for an engine to shutdown and reliably relit in-flight following a pyrotechnic-ignited boost phase.

To gain insight into the relationships between design parameters and performance of AMRDEC's hypergolic and hybrid rocket motor concepts, modeling and simulation of combustion chamber flow-physics and gas dynamics (via computational fluid dynamics, CFD) and propellant physical properties and reactivity (via computational chemistry and material science, CCM) are being performed with resources provided through the Challenge Project C4C. Presented herein are overviews of the CFD and CCM technical approaches and a brief summary of results achieved over the past year.

2. CFD Modeling

Modeling of the ISVE's combustion chamber fluid dynamics is being conducted with the ARL-NSRG3 code[6–9]. Designed to simulate unsteady, multi-component, chemically-reacting flows in various gas-dynamic systems, the latest upgrades to the code for the ISVE application include: 1) the addition of a sub-model for the simulation of injection schedules enabled by AMRDEC's Pneumatically-Actuated Bipropellant Valve (PABV) configuration for the ISVE[2,6] and 2) a reduced TEDMAZ-RFNA chemical kinetics mechanism.

The control of engine throttle is an essential aspect of the AMRDEC program to utilize non-hydrazine-based bipropellants such as TEDMAZ in the ISVE. In combination with RFNA, these fuels have demonstrated longer ignition delays than hydrazine or MMH. Since a delay in ignition allows propellant to accumulate in the combustion chamber, a high-amplitude pressure transient with the potential to destroy the motor can be generated. Referred to as a "hard-start," this situation can be avoided by limiting the injection rate at start-up and by reducing the number of operating injection pairs. A previous hardware option termed SLAMMITT successfully demonstrated the mitigation of TEDMAZ-210 ignition delay over-pressures[8,9] in the ISVE, but this valve design was only tested in heavy-weight, thick-walled engines. In an effort to transition to flight-weight engines, the AMRDEC developed the PABV configuration[2]. This new valve system is not part of the propellant manifold, and actually decouples the manifold from the injectors. Figure 1 shows schematics of the PABV engine designed to deliver 800 lb_f of thrust at full-throttle. This engine has a dedicated PABV unit for each of the three pairs of bipropellant injectors, and the units operate independently. By varying which pair (or pairs) of the injectors are opened, the engine will operate at the corresponding thrust-level.

Figure 1. Isometric and cross-section views of the heavy-walled ISVE with PABV[2]

A recent PABV engine test included four pulses separated by about 4.5 seconds of engine dwell-time[2]. During each pulse, the three pairs of injectors opened separately and then closed together. The average thrust and ISP measured during the four pulses were 878 to 888 lb_f and 244 to 251 lb_f-s/lb_m, respectively. It is interesting to note that the measured ignition delays for the four pulses (13, 9, 8, and 8 ms) actually shortened through the course of the engine test. Engine-wall temperature data was collected during the test using two thermocouples installed within the faceplate at the domed-head-end of the engine, and the three thermocouples installed below the engine chamber-wall in the same plane as propellant injectors. The data shows that the engine-wall temperatures are not reduced while the engine is dwelling between pulses; rather engine temperatures continue to rise, albeit well-within manageable levels.

For the CFD simulation of the PABV engine-test, fuel/oxidizer mixture conditions were taken from representative values near the impinging injectors. The actual values of bipropellant flow rate (lb/s) were extracted from propellant tank mass measurements made continually during the test firing and then included in the model. In this manner, the numerical simulation was allowed to "run" as close as possible to the actual test firing of the engine. The bipropellants used in this engine simulation are TEDMAZ-RFNA. The "full" chemical kinetics mechanism, which consists of 455 species and 1,930 reactions[8], was reduced to produce a mechanism of 90 species and 111 reactions. Computed pressure histories for three locations in the engine along with engine-thrust results are plotted in Figure 2. The "MID" location is in the dome about halfway toward the top, the "EDGE" location is near the point where the radial chamber-wall joins the domed-head, and the "WALL" location is along the radial chamber-wall between two pairs of injectors. The computed thrust-levels of 900 and

995 lb_f are similar to the recorded values of 880 and 881 lb_f for the first two pulses. This concurrence between the measured and computed results demonstrates the ability of the model to simulate reignition of the bipropellants.

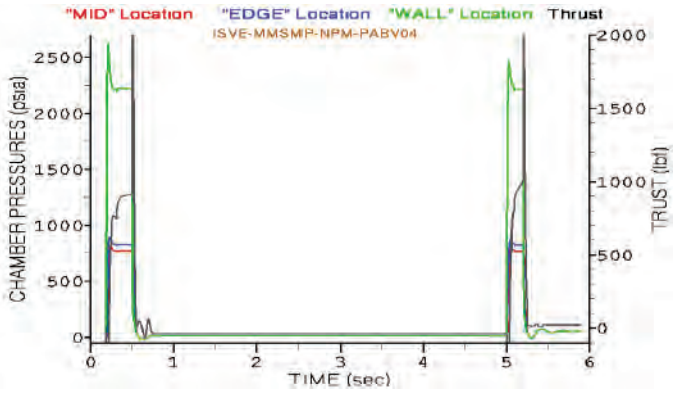


Figure 2. Computed chamber pressure and thrust profiles during the first two throttle events of ISVE PABV test 04

Figures 3 through 6 show selected CFD results of the flow-field within the ISVE for a given azimuthal plane; note that these figures are not drawn to scale. The times given for these figures can be referenced to Figure 2. Figure 3 shows the computed color OH species mass fraction contours (blue to red : 0 to 0.25) for a time of full-thrust during the first pulse (0.46 sec), and then the time (0.57 sec) just after shutdown of the first pulse (0.50 sec). Figure 4 shows the computed color OH mass fraction contours (same scale as Figure 3) for a time of low-thrust during the initiation of the second pulse (5.0 sec), and then a time of full-thrust during the second pulse (5.1 sec). As is the case for most other reaction product species, the OH (hydroxyl radical) concentrations remain at low-levels within the chamber and nozzle during the 4.5 seconds between engine pulses. A more dramatic demonstration of these trends is shown in Figures 5 and 6, which displays the computed color contours of H mass-fraction (blue to red : 0. to 0.7) for the times corresponding to Figures 3 and 4. Hydrogen atom concentrations are high in the domed-head–end of the engine during the full-thrust condition at 0.46 seconds and remain significant even after the first pulse has ended. The start of the second pulse at 5.0 seconds generates renewed levels of hydrogen atoms that increase when the second pulse is at full-thrust.

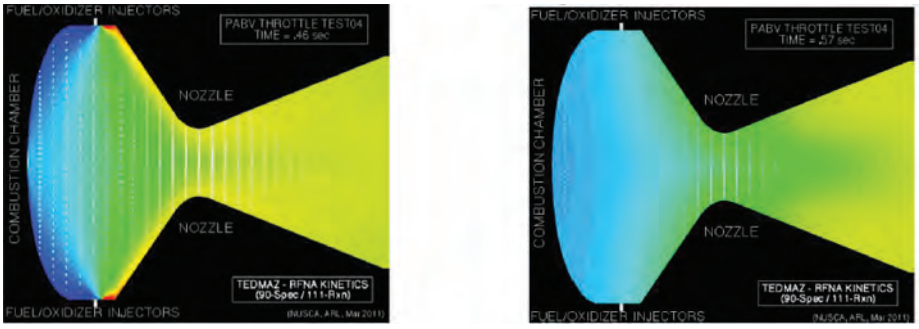


Figure 3. Computed color OH mass-fraction contours (blue to red: 0 to 0.25) and selected velocity vectors for the PABV-04 test conditions: (left) thrusting at 0.46 sec; (right) shutdown at 0.57 sec

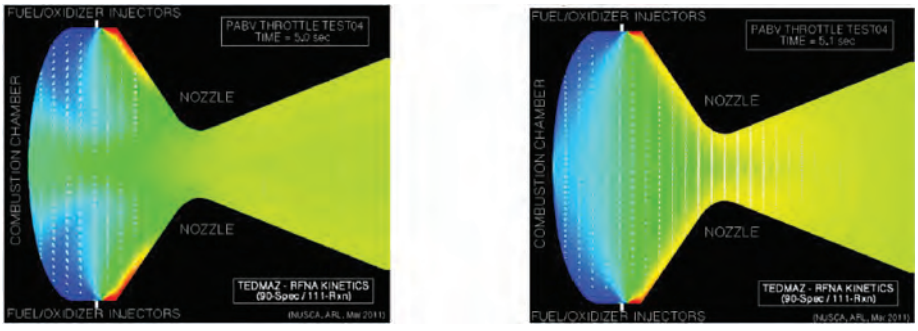


Figure 4. Computed color OH mass-fraction contours (blue to red: 0 to 0.25) and selected velocity vectors for the PABV-04 test conditions: (left) re-ignition at 5.0 sec; (right) thrusting at 5.1 sec

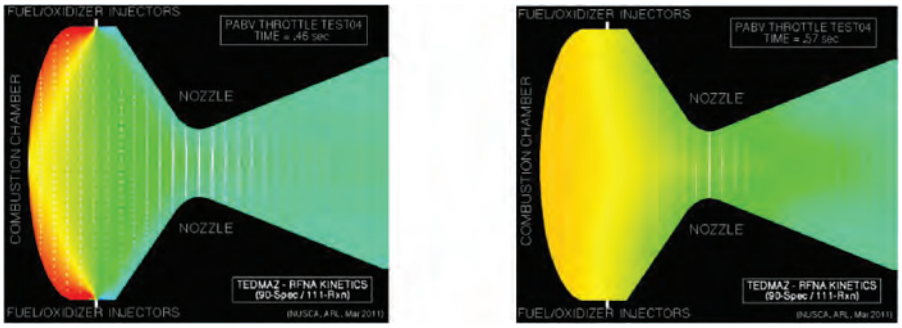


Figure 5. Computed color H mass-fraction contours (blue to red: 0 to 0.7) and selected velocity vectors for the PABV-04 test conditions: (left) thrusting at 0.46 sec; (right) shutdown at 0.57 sec

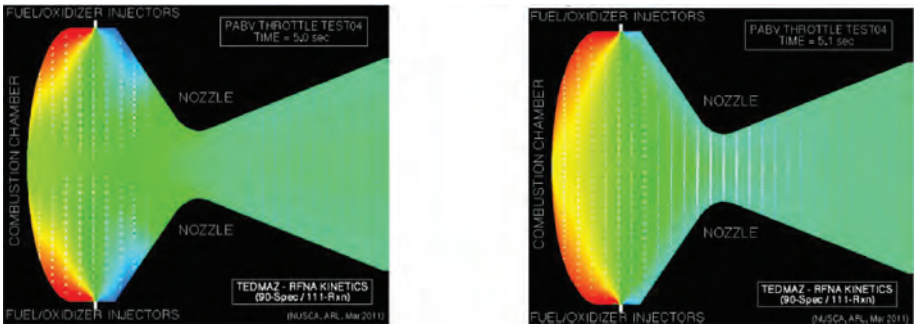


Figure 6. Computed color H mass-fraction contours (blue to red: 0 to 0.7) and selected velocity vectors for the PABV-04 test conditions: (left) re-ignition at 5.0 sec; (right) thrusting at 5.1 sec

This series of simulations for the first two pulses of PABV Test No. 4 demonstrate that even though engine pressure levels drop significantly between pulses, the concentrations of bipropellant combustion (such as OH and H) remain at significant levels in the engine. Between pulses, when convective heat transfer is insignificant within the engine chamber, these combustion gases continue to radiate heat to the engine-walls, especially in the well-stirred, domed-head-end of the chamber. Figure 7 shows the computed results of gas temperature near the engine-wall (domed head-end on the centerline, recall Figure 1) as well as computed temperature recessed below this wall location. Heat conduction analysis was utilized to compute the material temperature below the surface using known thermal properties of the wall (INCONEL-600 with thermal conductivity from 15 to 21 W/m-K over temperatures from 70 F to 800 F). The gas temperature near the wall rapidly rises to about 845 F during the full-thrust period of each pulse, but drops rapidly to about 180F (not included in the figures) between pulses. The wall-temperature slowly rises to about 620 F during the first pulse, drops slightly between pulses, but quickly rises to 720 F during the second pulse. There is less relaxation for the wall-temperature due to the presence of radiation heat input and the ability of the metal to retain heat. The measured data was given in Reference 5, and although measured wall-temperatures did not exceed 300 F during the four pulses, the trend was similar to those generated by the simulation.

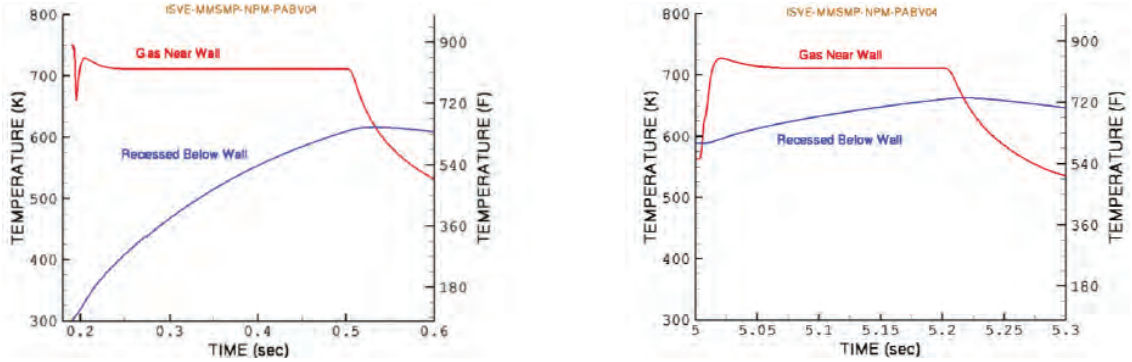


Figure 7. Computed gas temperature near and recessed below the engine-wall during the first throttle-event (left) and the second throttle-event (right) corresponding to ISVE PABV test 04

A second objective of this project was to extend the computational investigation of the AMRDEC hypergolic hybrid demonstration rocket engine to the tactical hybrid rocket engine, as displayed in Figure 8 and discussed by Wingard, Thompson, and Simmons[3]. The HTPB fuel grain in the tactical motor is approximately 7-inches in length and 4.5-inches in diameter. This highly-configured "roll-up" grain design featured 25 ridges and the surface area of the resulting grain was 250 square inches. Oxidizer is injected into the engine through one pair of injector orifices located in the plane shown in the Figure 8, aimed at the surface of the grain. The orifices are not explicitly represented in the simulation; the oxidizer is injected into the grid-cell adjacent to the wall at these locations. For purposes of this initial simulation, the igniter used in the tests[3] has been ignored and the fuel grain is assumed to be warm (273°C, 546K), but not at the postulated ignition temperature for HTPB of 547°C (820K). The fuel grain regresses at the measured rate of approximately 0.0496 in/s (0.1260 cm/s)[3] and the computational mesh in the model is self-adaptive to the regressing fuel grain surface. The burning fuel is assumed to emit butadiene (C4H6) gas into the computational cells adjacent to the surface at a computed blowing velocity. Numerous tests were performed using the 7-inch fuel grain (as well as some tests using a 10.5-inch grain) and are reported in Reference 3. For numerical analysis, the present effort is focused on test HY-NLOS-19 with chamber pressure results reproduced in Figure 8. In this test, the 7-inch motor demonstrated over 2,500 psi and peak thrust of over 700 lb (the small "hump" recorded at the end of the pressure-time curve is a result of a purging procedure).

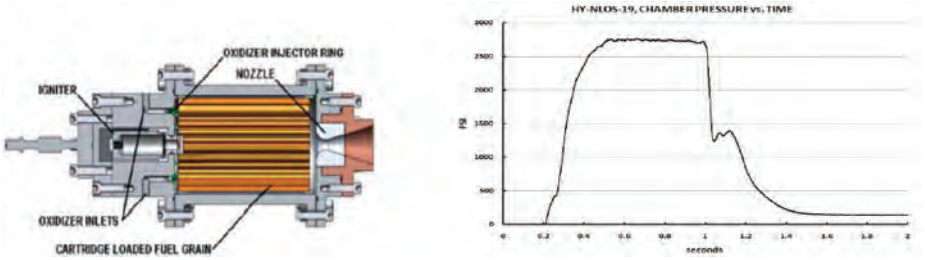


Figure 8. Schematic of the AMRDEC Hypergolic Hybrid Tactical Rocket Engine (left) and typical pressure-time data for test number 19 (right)[3]

Figure 9 shows the computed grain thickness (average over the grain length) and the engine pressure as a function of time for the 0.8 seconds of engine operation. The model assumes the measured regression-rate for the fuel grain, and it was also run for the case of regression-rate accelerated by a factor of sixty. The computed reduction in average grain thickness is consistent with the regression-rate, and the computed pressure level is quite similar to that observed in the test. In the case of an accelerated regression-rate, the engine reaches full-pressurization just before the grain burns out. Also, because the fuel grain regression-rate is directly proportional to the oxidizer flow-rate, engine shutdown stops the changes in average thickness. The slower depressurization of the engine evident in the computed results (Figure 9) as compared to the measurements (Figure 8) could indicate that the model produces a higher-rate of continuous fuel surface heating and chemical reaction of pyrolysis gases after shutdown.

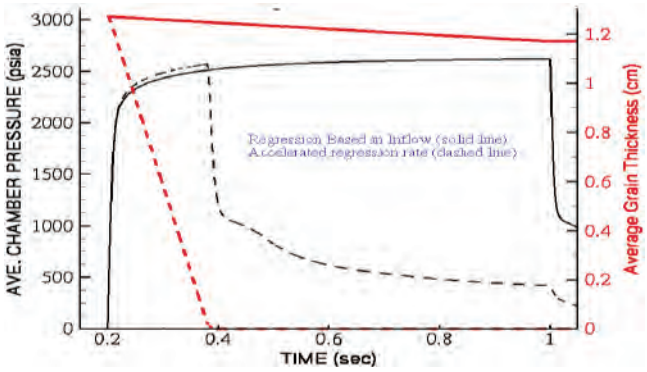


Figure 9. Computed chamber pressure (average) and HTPB grain thickness for regression-rate determined using measured RFNA inflow and using an accelerated (60x) regression-rate

Additional computational results (i.e., gas molecular weight, pressure, temperature) for this engine configuration are given in Reference 7. An interesting aspect of the computed engine conditions that can be explored using the model is the production of solid phase particles from the burning and regressing fuel grain surface. The best manner in which to

view these results is by accelerating the HTPB burn-rate by a factor of about sixty, as evidenced by the previous testing. Figures 10 and 11 show the porosity with the solid limit of zero colored red and the void limit of unity colored blue; the presence of the solid propellant (red in color) is easily recognized, along with locations in the flow-field where particles are accumulating. Figure 10 shows that as the fuel grain begins to regress, a large number of particles accumulate at the forward-end of the motor (i.e., green color near and below the oxidizer injectors) while a smaller number of particles are entrained into the flow toward the nozzle (i.e., light blue "clouds" of particles). The predominant "red" color along the motor centerline indicates a large number of particles that have been forced and trapped in the low-radial velocity-field. However, these particles that become resident along the centerline continue to burn and regress during axial movements through the nozzle. Although not readily apparent in the figures, solid particle collision with the nozzle throat surface is frequent in the simulation. Figure 11 highlights the development of a low-velocity boundary-layer of particles near the burning fuel grain surface that grows thicker over time (i.e., note the absence of significant velocity vectors in the light-blue region adjacent to the fuel surface).

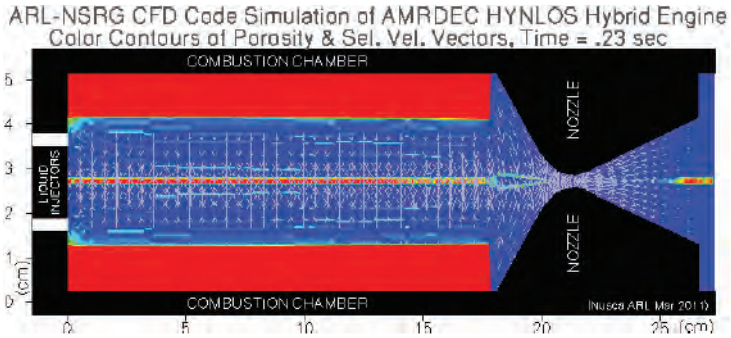


Figure 10. Computed color porosity contours and selected velocity vectors for the hypergolic hybrid engine showing a single azimuthal plane at a time of 0.23 seconds using an accelerated (60x) regression-rate

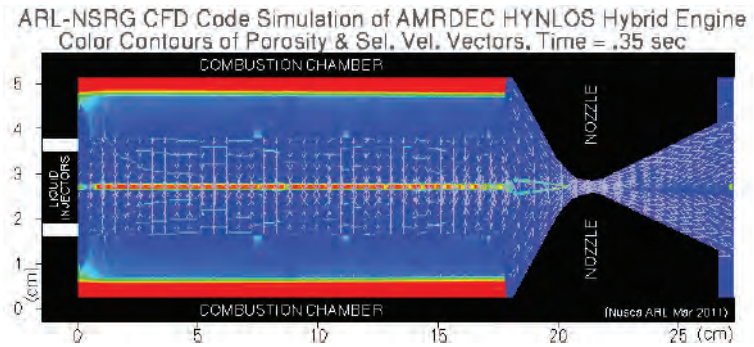


Figure 11. Computed color porosity contours and selected velocity vectors for the hypergolic hybrid engine showing a single azimuthal plane at a time of 0.35 seconds with an accelerated (60x) regression-rate

3. CCM Modeling

Hydroxyl-terminated polybutadiene (HTPB) is widely-used as a polymeric binder/fuel in composite solid-rocket-propellant and hybrid-rocket motor-fuel grains, thus the reaction kinetics of its combustion in air is pertinent to a wide-variety of operational scenarios, and has been experimentally studied. The effort summarized here was undertaken to develop a finite-rate chemical kinetics mechanism that can be employed to model the process. A previously-developed mechanism for modeling the thermal decomposition of R45M was employed as the starting point for the study[11]. Thermochemical kinetic properties and reaction path parameters were determined by *ab initio* and density functional theory quantum chemistry methods. Modeled reaction-steps included: 1) abstraction of H-atoms from an HTPB proxy (6-ethenyl-2,8,12,16-octadectetraene) by O_2, OH , O and H, 2) bimolecular reactions between daughter radicals and O_2, 3) unimolecular dissociations of HTPB-O_2 adducts, and 4) H, O, and OH reactions with primary products. Details of the computational methods are provided elsewhere[12]. Representative examples of reaction characterizations are presented and discussed here.

There are two types of reactions responsible for initiating the combustion of HTPB in the air: abstraction of H-atoms and unimolecular decomposition. At low-temperatures, H-atom abstraction reactions are the dominant mechanism. The

abstraction of H-atoms from HTPB can take place at secondary allyl [$\mathbf{C}_{(a)}$ and $\mathbf{C}_{(b)}$], tertiary allyl [$C_{(c)}$] or secondary alkyl [$\mathbf{C}_{(d)}$] C-atoms (see Figure 12). To establish rate expressions for abstraction from each C-atom type, mechanisms of abstraction for analogous C-atoms in smaller hydrocarbons were modeled. The species employed for this purpose included propane, (E)-2- pentene, (Z)-2- pentene, and 3-methyl-1-butene. Transition- state structures for reactions involving H-atom abstraction from the secondary alkyl C-atom of propane, the secondary allyl C-atom of butene and the tertiary allyl C-atom of 3-methyl-1-butene are shown in Figure 13. G3//B1K- and G3MP2//B1K-calculated reaction enthalpies and activation energies for various products are listed in Table 1. Rate constants for the reactions based on the G3//BIK results are also provided in Table 1. Figure 14 compares computed vs. measured rates[13–17] as a function of temperature for $C_3H_8 + O_2 \rightarrow$ iso-$C_3H_7 + HO_2$ and $C_3H_6 + O_2 \rightarrow C_3H_5 + HO_2$. Good agreement is observed, providing a measure of validation for this computational approach.

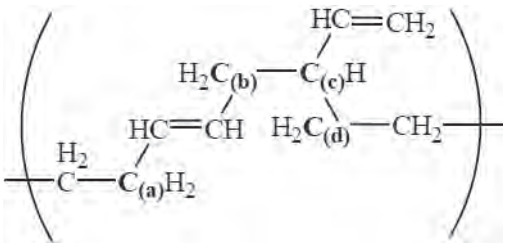


Figure 12. Different C-atom types in HTPB

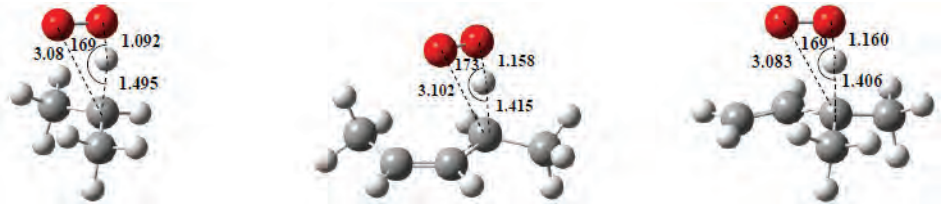


Figure 13. Structures of transition states for O_2's abstraction of an H-atom from the secondary alkyl C-atom of propane, the secondary allyl C-atom of pentene, and the tertiary allyl C-atom of 3-methyl-1-butene

Table 1. Calculated reaction enthalpies and activation energies (in kcal/mol at 298K) for the abstraction of H-atoms from various hydrocarbons by O_2

Producta (+HO2)	Type	ΔH_{rxn}	$\Delta H^{\neq}$		A^b	n	E_a^c
			G3MP2	G3			
CC•C	sec-alkyl	49.3	44.24	43.87	1.68E+07	1.85132	43.95
ECC•VC	sec allyl	34.5	36.48	35.72	5.69E+01	3.28819	34.72
ZCC•VC	sec allyl	33.5	35.96	35.16	1.41E+01	3.46819	34.08
$C_2C•V$	tert-allyl	33.2	36.63	35.89	5.70E+02	3.03767	35.00
CC•VCCIVCC	sec allyl	36.2	37.16	36.30[d]			
CCVCCI•VCC	tert-allyl	35.4	38.09	37.20[d]			

[a] CC•C [$CH_3C•HCH3$]; ECC•VC [(E)-$CH_3C•HCH=CHCH_3$]; ZCC•VC [(Z)-$CH_3C•HCH=CHCH_3$]; $C_2C•V$ [$(CH_3)_2C•CH=CH_2$]; CC•VCCIVCC [$CH_3C•HCH=CHCH_2CH(CH=CH_2)CH_2CH_3$]; CCVCCI•VCC [$CH_3CH_2CH=CHCH_2C•(CH=CH_2)CH_2CH_3$]
[b] $mol^{-1}\ cm^{-3}\ s^{-1}$
[c] kcal/mol
[d] scaled to G3//B1K

Three different radicals were postulated as products of H-atom abstraction from HTPB: two resonance-stabilized allyl radicals [$R_{sec\text{-allyl}}$=*sec*-(C•HCH=$CHCH_2$)- and $R_{tert\text{-allyl}}$=*tert*-(C • ($CH=CH_2$)CH_2)-], and a secondary-alkyl radical [$R_{sec\text{-alkyl}}$=-(CH($CH=CH_2$)C•H)-]. Figure 15 shows a potential energy diagram for some of the paths that may be followed upon the addition of O_2 to $CH_3CH=CHCCH•CH_2CH=CHCH_2$. As shown, the first step involves the formation of a $CH_3CH=CHCCH(OO•)CH_2CH=CHCH_2$* adduct. $CH_3CHCHCCH(OO•)CH_2CHCHCH_2$* can: 1) stabilize to $CH_3CHCHCCH(OO•)CH_2CHCHCH_2$ (via R1), 2) dissociate back to reactants (via R-1), 3) undergo concerted HO_2 elimination to produce $CH_3CH=CH-CH=CHCH=CHCH_3$ (via R2), 4) cyclize to form a six-membered cyclic peroxide radical, $CH_2•CH=CHy(CHOOCHCH_2-$

CH•)CH_3 (via R3), or 5) isomerize via an intramolecular H shift to form hydroperoxy ally radical, CH_2•CH=CHCH(OOH)CH_2CH=CH-CH_3 (via R4). The CH_2•CH=CHCH(OOH)CH_2CH=CH-CH_3 radical can undergo 1) dissociative three-member ring closure to form cyclic ether radicals with elimination of an OH radical (via R5) and 2) HO_2 elimination to form CH_3CH=CH-CH=CHCH=CHCH$_3$ (via R6).

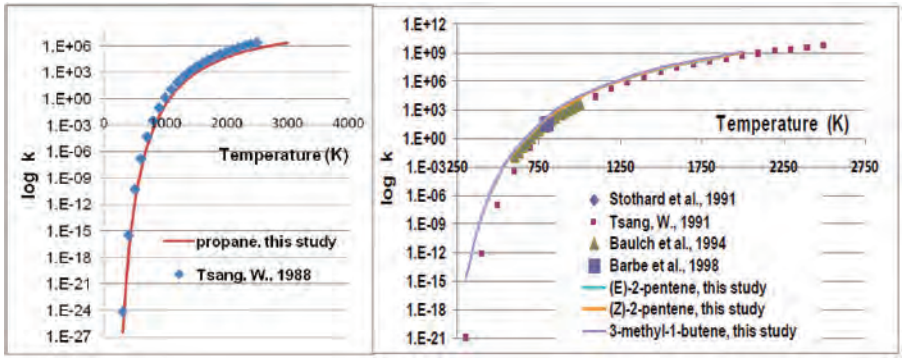


Figure 14. Rates calculated for H-atom abstraction reactions: (a) $C_3H_8 + O_2 \rightarrow$ iso-$C_3H_7 + HO_2$; (b) 2-pentene + $O_2 \rightarrow$ CH_3C•HCH=CHCH$_3$ + HO_2 and 3-methyl-1-butene + $O_2 \rightarrow (CH_3)_2$C•CH=CH_2+ HO_2

Figure 15 also shows paths for the reaction of CH_3CH=CHy(CHOOCHCH$_2$CH•)CH_3. They include: 1) cleavage of a O-O bond with concerted cyclization to form three-member cyclic ether oxy radical (via R7), 2) cleavage of an ally C-C bond in the ring and the formation of CH_3CH=CHCH•OOCH(CH_3)CHCH$_2$ (via R8), and 3) isomerization via a six-membered ring to an allyl cyclic peroxide radical (CH_2•CH=CHy(CHOOCHCH$_2$CH)CH$_3$) (via R9). CH_3CH=CHCH•OOCH(CH_3)CHCH$_2$ can rapidly decompose to CH_2CH=CHCHO and CH_2=CHCHO•CH_3.

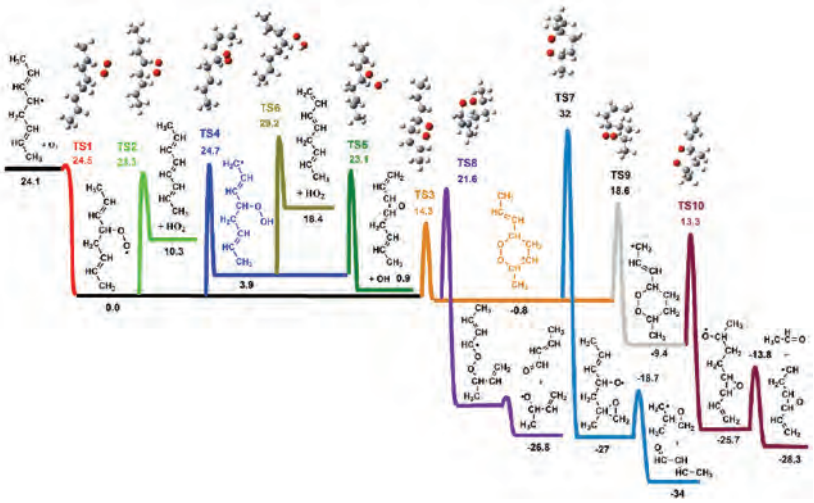


Figure 15. Some paths for the decomposition of CH_3CH=CHCCH•CH_2CH=CHCH$_2$ following the addition of O_2

As illustrated in Figure 16, the addition of a second oxygen molecule to the product of R3 is 35.5 kcal/mol exothermic; and thus ~10 kcal/mol more exothermic than the first oxygenation step. The key step for the second O_2 addition is isomerization via H-transfer from the allyl carbon atom to the oxygen of the peroxy radical. The barrier to this reaction is 27 kcal/mol below the entrance channel, and the path forms a highly-unstable intermediate with a radical center localized on the carbon atom attached to the peroxide in the cyclic ring. This radical is expected to dissociate with little or no barrier via the formation of a strong carbonyl double-bond in the ketohydroperoxide intermediate CH_3CH=CHC(=O)CH_2CH(OOH)CH(O•)CH_3.

Rate constants for the reactions discussed are summarized in Table 2, and Figure 17 illustrates the temperature-dependence of rates. Although the channel for CH_3CH=CHCH=CHCH=CHCH$_3$ formation (R2) is close in energy with the initial reactants, and higher by ~9 kcal/mol than the lowest energy channel (R3), R2 is still dominant at temperatures above 800K due to its high-A-factor [~1×10^{13}]. Figure 17 also shows that the R-1 is an important reaction path at moderate- to high-temperatures.

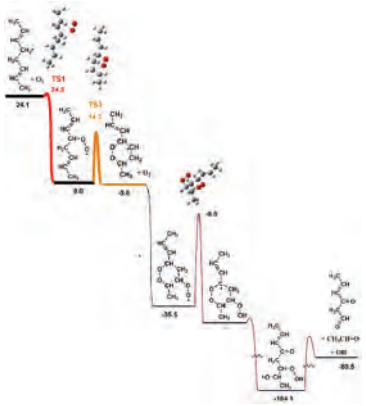


Figure 16. Potential energy diagrams for the addition of a 2nd O_2 to $H_3CH=CHCCH•CH_2CH=CHCH_2$

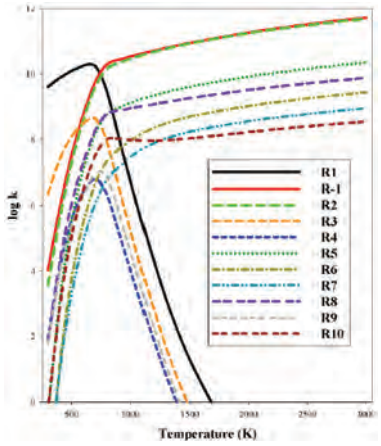


Figure 17. Reaction rates as a function of temperature for the $CH_3CH=CH$ $CH•CH_2CH=CHCH_2+O_2$ system

Table 2. Rate constants for selected reactions[a]

	Reaction	*A* (s^{-1} or cm^3/mol-s)	*n*	E_a (kcal/mol)
R1	$CH_3CH=CHCH•CH_2CH=CHCH_3 + O_2 \rightarrow CH_3CH=CHCH(OO•)\ CH_2CH=CHCH_3$	8.59×10^2	2.60834	-0.30
R-1	$CH_3CH=CHCH(OO•)\ CH_2CH=CHCH_3 \rightarrow CH_3CH=CHCH•CH_2CH=CHCH_3 + O_2$	1.21×10^{10}	0.89519	22.02
R2	$CH_3CH=CHCH(OO•)\ CH_2CH=CHCH_3 \rightarrow CH_3CH=CHCH=CHCH=CHCH_3 + HO_2$	4.33×10^{10}	0.79497	23.49
R3	$CH_3CH=CHCH(OO•)\ CH_2CH=CHCH_3 \rightarrow CH_3CH=CHy(CHOOCHCH_2CH•)CH_3$	6.74×10^7	0.15351	14.78
R4	$CH_3CH=CHCCH(OO•)\ CH_2CH=CHCH_2 \rightarrow CH_2•CH=CHCH(OOH)CH_2CH=CHCH_3$	7.49×10^9	0.70566	24.74
R5	$CH_2•CH=CHCH(OOH)CH_2CH=CHCH_3 \rightarrow CH_2=CHy(CHOCH)CH_2CH=CHCH_3 + OH$	3.17×10^{12}	0.47368	19.67
R6	$CH_2•CH=CHCH(OOH)CH_2CH=CHCH_3 \rightarrow CH_2=CHCH=CHCH_2CH=CHCH_3 + HO_2$	2.07×10^{13}	0.27438	25.87
R7	$CH_3CH=CHy(CHOOCHCH2CH•)CH_3 \rightarrow CH_3CH=CHCHO + CH_2•y(CHOCH_2)CH_3$	3.12×10^{12}	0.47314	33.21
R8	$CH_3CH=CHy(CHOOCHCH_2CH•)CH_3 \rightarrow CH_3CH=CHCHO + CH_2=CHCHO•CH_3$	4.62×10^{11}	0.48547	22.81

	Reaction	*A* (s^{-1} or cm^3/mol-s)	*n*	E_a (kcal/mol)
R9	$CH_3CH=CHy(CHOOCHCH_2CH\bullet)CH_3 \rightarrow CH_2\bullet CH=CHy(CHOOCHCH_2CH)CH_3$	3.04×10^{10}	0.36706	19.58
R10	$CH_2\bullet CH=CHy(CHOOCHCH_2CH)CH_3 \rightarrow CH_2=CHy(CHOCH)CH_2CH_2\bullet + CH_3CHO$	4.81×10^{12}	0.36297	23.24

[a]High-pressure-limit rate parameters. $k=A'T^n\exp(-E_a/RT)$. Calculated for T=300 to 2,000K.

Results such as those discussed were coupled with a set of reactions for small molecules to produce a chemical kinetics mechanism for HTPB oxidation in air. The major classes of elementary reaction-steps in the present mechanism include: 1) unimolecular fuel decomposition, 2) H-atom abstraction from the fuel, 3) alkenyl radical isomerization or decomposition, 4) alkenyl radical + O2 reactions, 5) alkdiene + OH + O2 reactions, 6) RO• decomposition, 7) R'CH=O decomposition, and 8) small-molecule reactions assembled for a GRI mechanism[18]. Figure 18 provides a diagram of the general sequence of important reactions that HTPB oxidation is expected to take. At present, the mechanism is composed of ~2,000 elementary reaction steps and involves ~600 species.

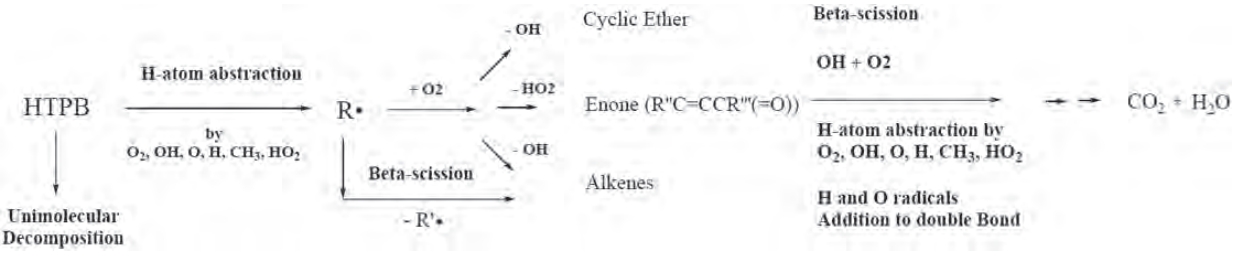


Figure 18. General sequence of reactions expected in HTPB oxidation ignition and combustion

4. Summary of Accomplishments

1. The successful application of CFD for the visualization of chemically-reacting flows within the ISVE's combustion chamber yields insights into the nature of the performance of an ISVE that employed the PABV concept and the hydrazine-alternative fuel TEDMAZ (with oxidizer RFNA). The CFD simulations reveal that combustion gases that remain in the chamber at significant levels between thrust modulation events (i.e., bipropellant injector open-close, and then eventual hypergolic reignition) contribute to the maintenance of high- chamber wall-temperatures. For the hybrid rocket engine (i.e., solid fuel, liquid oxidizer), CFD simulations reveal that hot solid particles vaporizing from the propellant grain can be a source of heat transfer to the nozzle-wall during extended engine operation.
2. To enable modeling of the dynamics of HTPB combustion in air, a multi-step, finite-rate chemical kinetics mechanism was developed for the decomposition and oxidation of 6-ethenyl-2,8,12,16-octadectetraene. H-atom abstraction reactions are expected to be involved in the initial step of the decomposition process, and three different types were modeled: two producing resonance-stabilized allyl radicals (Rallyl) and one producing a secondary-alkyl radical (Rsec-alkyl). At low-temperatures, the rate for Rsec-alkyl formation is much slower than the rates for Rallyl formation, and it is a factor of ~3 slower as temperatures reach 2,000K. The path with the lowes-t barrier leads to the formation of a six-membered cyclic peroxide radical, and is the dominant reaction at temperatures below 800K. Having a high-A-factor, direct HO2 elimination with olefin formation is the dominant path for the decomposition of R-OO adducts at temperatures above 800K. Results such as these were coupled with a set of reactions for small-molecules that were assembled for other mechanisms, leading to a mechanism that has ~2,000 elementary reaction-steps and involves ~600 species. Reduction of the mechanism for use in CFD modeling is planned.

Acknowledgements

The DoD HPC Modernization Office supported this project by supplying supercomputer time under the Computing Challenge Project C4C. The computer time was made available at the DoD Supercomputing Resource Centers at AFRL and NAVY. Mr. R.S. Michaels and Mr. N. Mathis (AMRDEC) provided helpful information concerning the ISVE, and Mr. D. Thompson (AMRDEC) and Mr. Z. Wingard (ARL) provided helpful information concerning the hybrid rocket motor development effort.

References

1. Michaels, R.S. and Wilson, B.F., "The Low L/D Vortex Engine for Gel Propulsion", *Proceedings of the 1995 JANNAF Gel Propulsion Technology Symposium, Chemical Propulsion Information Center, CPIA Publication. 627*, pp. 9–16, 1995.

2. Mathis, N.P., Salehi, D.M., Michaels, R.S., and Turner, T.W., "Evaluation of the Army's Throttleable PABV Vortex Engine", *Proceedings of the 57th JANNAF Propulsion Meeting, CPIAC Publication JPM-CD-09*, 2010.

3. Wingard, Z.K., Thompson, D.M., and Simmons, E.L., "Tactical Army Hybrid-Propulsion Development", *Proceedings of the 5th JANNAF Liquid Propulsion Subcommittee Meeting, Chemical-Propulsion Information Analysis Center, CPIAC Publication JSC CD-62*, 2010.

4. Anderson, W.R., McQuaid, M.J., Nusca, M.J., and Kotlar, A.J," A Detailed, Finite-Rate Chemical Kinetics Mechanism for Monomethylhydrazine-Red Fuming Nitric Acid, MMH-RFNA Systems", *ARL-TR-5088*, US Army Research Laboratory, Aberdeen Proving Ground, MD, 2010.

5. McQuaid, M.J.," Notional Hydrazine-Alternative Hypergols: Design Considerations, Computationally-Based Property Determinations, and Acquisition Possibilities", *ARL-TR-3694*, US Army Research Laboratory, Aberdeen Proving Ground, MD, 2006.

6. Nusca, M.J., Mathis, N.P., Michaels, and R.S., "Computational Study of Throttling and Heat Transfer for the Army's Impinging-Stream Vortex Engine", *Proceedings of the 44th JANNAF Combustion Subcommittee Meeting, Chemical Propulsion Information Analysis Center, CPIAC Publication*, April 2011 (in press).

7. Nusca, M.J., "Computational Fluid Dynamics Modeling of the Army's Hybrid Hypergolic Rocket Engine", *Proceedings of the 44th JANNAF Combustion Subcommittee Meeting, Chemical Propulsion Information Analysis Center, CPIAC Publication*, April 2011 (in press).

8. Chen, C.-C., Nusca, M.J., Kotlar, A.J., and McQuaid, M.J., "Combustion Chamber Fluid Dynamics and Hypergolic-Gel-Propellant Chemistry Simulations for Selectable Thrust Rocket Engines", *Proceedings of the DoD HPCMP Users Group Conference*, San Diego, CA, IEEE Computer Society, 2009.

9. Nusca, M.J., Chen C.-C., and McQuaid, M.J., "Modeling the Combustion Chamber Dynamics of Selectable-Thrust Rocket Motors", *Proceedings of the DoD HPCMP Users Group Conference*, Nashville, TN, IEEE Computer Society, 2010.

10. Chen, C.-C., Nusca, M.J., and McQuaid, M.J., Modeling Combustion Chamber Dynamics of Impinging-Stream Vortex-Engines Fueled With Hydrazine-Alternative Hypergols", *Proceedings of the 26th Army Science Conference*, Orlando, FL, 2008

11. Chen, C.-C. and McQuaid, M.J., "Thermochemical and Kinetic Studies of The Pyrolysis of Hydroxyl-Terminated Polybutadiene, HTPB", *Proceedings of the 25th JANNAF Propulsion Systems Hazards Subcommittee Meeting*, La Jolla, CA, 2009

12. Chen, C.-C. and McQuaid, M.J., "Thermochemistry and Kinetics Modeling of the Oxidation of Hydroxyl-Terminated Polybutadiene in Air", *Proceedings of the 44th JANNAF Combustion Subcommittee Meeting*, Arlington, VA, , 2011 (in press).

13. Tsang, W., "Chemical Kinetic Data Base for Combustion Chemistry. Part 3. Propane", *Journal of Physical and Chemical Reference Data*, 17, 1988.

14. Barbe, P., Baronnet, F., Martin, R., and Perrin, D., "Kinetics and Modeling of the Thermal Reaction of Propene at 800 K. Part III. Propene in the Presence of Small Amounts of Oxygen", *International Journal of Chemical Kinetics*, 30, 503, 1998.

15. Baulch, D.L., Cobos, C.J., Cox, R.A., Frank, P., Hayman, G., Just, T., Kerr, J.A., Murrells, T., Pilling, M.J., Troe, J., Walker, R.W., and Warnatz, J., "Evaluated Kinetic Data for Combustion Modeling. Supplement I", *Journal of Physical and Chemical Reference Data*, 23, 847, 1994.

16. Tsang, W., "Chemical Kinetic Data Base for Combustion Chemistry. Part V. Propene", J*ournal of Physical and Chemical Reference Data*, 20, 221, 1991.

17. Stothard, N.D. and Walker, R.W., "Determination of the Arrhenius Parameters for the Initiation Reaction $C_3H_6 + O_2 \rightarrow CH_2CHCH_2 + HO_2$", *Journal of the Chemical Society: Faraday Transactions*, 87, 241, 1991.

18. Smith, G.P., Golden, D.M., Frenklach, M., Moriarty, N.W., Eiteneer, B., Goldenberg, M., Bowman, C.T., Hanson, R.K., Song, S., Gardiner, W.C., Lissianski, V.V., and Qin, Z., http://www.me.berkeley.edu/gri_mech/.

Potential Energy Surface Mapping of Energetic Materials Using Coupled Cluster Theory

DeCarlos E. Taylor
US Army Research Laboratory (ARL), Aberdeen Proving Ground, MD
decarlos.taylor@us.army.mil

Abstract

1,1-diamino-2,2-dinitroethylene (FOX-7) has proven to be a promising energetic material due to its low shock sensitivity and high thermal stability, and has been the subject of several Department of Defense (DoD) theoretical investigations employing Hartree-Fock and density functional theory. However, it is well-known that many energetic crystals have a significant binding contribution resulting from dispersion, a phenomena that is not accurately described by computational methods that are not explicitly correlated. This paper details the use of a state-of-the-art quantum chemistry method, coupled cluster theory, to compute a six-dimensional, dimer potential energy surface of the FOX-7 energetic, which will be used to develop a pair potential for use in molecular dynamics simulations. In addition, a dimer potential energy function determined using symmetry-adapted perturbation theory is described, and molecular dynamics simulations using the fitted potential are presented.

1. Introduction

The US Army Research Laboratory (ARL) is the Army's corporate basic and applied research laboratory. Munitions research is a specialty of scientists within the Weapons and Materials Research Directorate (WMRD) of ARL; and the organization serves as the principal conduit for research and development of weapons and materials technologies for the US Army. Due to the inordinate time and pecuniary costs associated with development and testing of energetic materials (EM), a principle component of the WMRD mission is the design and performance characterization of insensitive munitions and high-energy density materials using computational methodologies. The suite of quantum chemistry (QC) and molecular dynamics (MD) simulation tools developed at WMRD, and in conjunction with Department of Defense (DoD) scientists operating across all military branches, has proven to be an invaluable resource in EM research via provision of properties such as crystal structures, lattice energies, densities, and thermodynamic data. The quality of the MD simulations for existing and/or proposed munitions is almost completely contingent upon the level of accuracy of the underlying QC methods used to generate the potential parameterization data. Consequently, the design- and performance-related munitions recommendations, which themselves are based on results of MD simulations, are also predicated upon the accuracy of the underlying QC method.

The development of accurate potentials for MD involves several steps:

1. Choice of functional form
2. Development of a reference data set
3. Tuning of the potential parameters with respect to the reference data
4. Use of the potential in MD simulations

The choice of functional form is typically dictated by the overall size of the MD simulation and the complexity of the simulated property. For very large simulation cells and non-reactive simulations, simple functional forms such as exponential-6, Lennard-Jones, or Morse potentials may be applied; however, complex structures and reaction chemistries often necessitate many-body formulations such as Tersoff[1], embedded atom[2], or Reax force-fields[3] (ReaxFF). Irrespective of the functional form, the performance of the potential is contingent upon the quality of the data used to parameterize the model. Often, experimental data, when available, is used to develop potential parameters. The disadvantage of this approach is that notional compounds, where experimental data does not yet exist or is very limited, cannot be studied; therefore, most reference data for potential parameterization is computed using quantum mechanics. As an example, the ReaxFF is a many-body potential developed by the Goddard group at Cal Tech, and has been widely used in the simulation

community[3]. ReaxFF was fitted to a large compendium of reference data computed using density functional theory (DFT). However, it is well known that DFT does not properly treat van der Waals interactions, a dominant contributor to the binding energy in some energetic crystals[4]. This can lead to erroneous results when applied to energetic formulations as exemplified in Figure 1, where the simulation cell dimension for cyclotrimethylene trinitramine (RDX) is shown to be in very poor agreement with experiment when the ReaxFF is used in ambient state MD simulations of the materia.

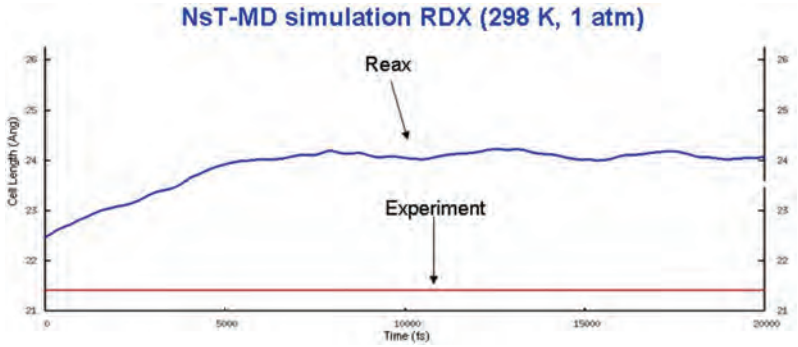


Figure 1. Time trace of the simulation cell dimension using the ReaxFF in an MD simulation of the RDX energetic. Blue line=ReaxFF, Red line=experiment.

Although the shortcomings of DFT are well-documented, it is by far the overwhelming method of choice in the modeling and simulation community, due to its relatively low computational scaling compared to explicitly-correlated wavefunction-based approaches. However, with the ever-increasing compute power available to DoD researchers, coupled with scalable software, ab initio wavefunction methods are becoming more viable options for the study of large energetics, which often contain over 20 heavy atoms and more than 100 correlated electrons per molecule. This Challenge Project, C4N, is a first step towards the application of the explicitly-correlated QM approach, coupled cluster theory, in the development of a force-field to be used in an MD simulation of an advanced energetic.

2. Numerical Methodology

Coupled cluster theory[5] is generally regarded as the gold standard of computational quantum chemistry methods. It provides accurate results, often rivaling experiment, for a wide variety of molecular properties including structures, energies, vibrational frequencies, etc. In coupled cluster theory, the ground-state wavefunction, Ψ_g is given by

$$|\Psi_g\rangle = e^T |\phi_o\rangle \tag{1}$$

where ϕ_o is a single Slater determinant (Hartree-Fock reference) and T is an excitation operator which creates single, double, up to n-tuply excited configurations from the reference determinant. T can be written as

$$T = T_1 + T_2 + \ldots = \sum_{i,a} t_i^a a^+ i + \frac{1}{2} \sum_{i,j,a,b} t_{ij}^{ab} a^+ i b^+ j + \ldots \tag{2}$$

where the standard creation/annihilation operators have been used, and summation indices (i,j….) (a,b….) represent the occupied and virtual space, respectively. A coupled cluster calculation reduces to a determination of the "t amplitudes" present in each level of excitation, and typically the excitations are treated completely through excitation levels 1 and 2 (the CCSD-level), and a partial inclusion of excitation level 3 is done via perturbative techniques in the (T) method[6].

The advantage of coupled cluster theory is the use of the exponential ansatz for the wavefunction. Due to the exponential parameterization of the excitation operator, CC theory is a more rapidly convergent series of approximations toward the exact solution of the time-independent Schroedinger equation than is the linear configuration interaction approach, for example. It can be shown that the exponential wave operator results in *products* of T operators; therefore, certain classes of excitations enter the CC equations in a non-linear way and effects from higher excitation manifolds are included in a convenient and compact manner[5]. For example, even when the CC equations are truncated at the single and double excitation level, *triple* excitation effects are included through terms such as T_1*T_2 (which would not be possible with a linear approach); however, the number of independent amplitudes to be determined remains the same as in the singles and doubles methods. (These products of terms also play a role in ensuring size extensivity of the CC approach.)

Due to the explicit inclusion of electron correlation, CC methods are capable of capturing the important dispersion interactions that dominate EM binding, unlike the current plethora of DFT functionals. However, due to the steep computational scaling of CC methods, their application to large EM molecules has thus far been limited. Given that the minimum acceptable level of CC theory, CCSD, has a 6th power dependence on the number of basis functions, CC approaches are limited to 10–20 atoms in routine application. Further, when the focus is potential parameterization, as is the case in this work, a large number of data points (1,000+) may be necessary, which makes application of CC methods even more intractable. However, armed with the newly developed ACES III[7] program system for CC computations combined with high performance computing resources available through a High Performance Computing Modernization Program Challenge award (Project C4N), potential function parameterization using CC theory is now a definite possibility.

3. Software

ACES III is a parallel implementation of the ACES II[8] program developed at the University of Florida by Bartlett and co-workers. ACES is generally regarded as the most efficient and robust suite of programs for performing CC theory calculations, and the parallelization of the program was achieved with HPCMP support (portfolio project CBD-03) and approved for public release in March of 2006. The ACES III program has shown efficient scaling across a wide range of computer architectures and problem sizes. Shown in Figure 2 is the wall-clock time per CCSD iteration for several representative systems including luciferin, sucrose, and a protonated water cluster. The luciferin calculation has 498 basis functions (N), and 46 correlated occupied orbitals (O), and was run on a Sun cluster using up to 256 processors. Sucrose (N=546, O=91), and water (N=657, O=84) were run on a Cray XT4 using 256 up to 4,096 processors. As shown in Figure 2, ACES III is scalable across several architectures and processor counts. Using a combination of HPC resources and the ACES III software program, an EM potential energy function parameterized with a large number of energy data points at the CC level is being developed under this Challenge Project.

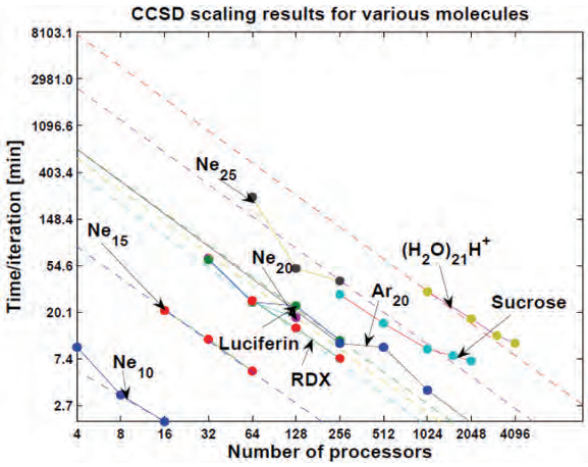


Figure 2. Scaling of ACES III. Solid lines connect the calculated points and dashed lines show the slope of ideal linear scaling

4. Results and Discussion

The 4 monomers within the FOX-7 unit cell are depicted in Figure 3 and each unique pair has been used in development of the potential. In order to fit an *inter*-molecular potential energy function, the interaction energy between FOX-7 dimers must be computed at a variety of orientations until an adequate parameterization is obtained.

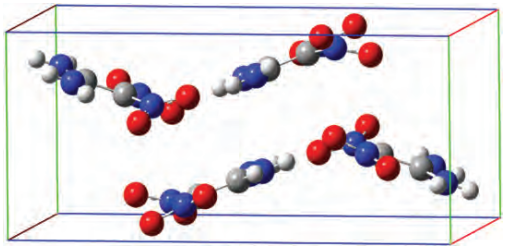


Figure 3. FOX-7 unit cell

In order to get an initial estimate of the total number of grid points required to saturate the parameter space, symmetry-adapted perturbation theory[9] (SAPT) has been used to develop a FOX-7 potential before initiation of the CC work. This was done to identify the most important data points for determination of the intermolecular potential so that the Challenge CPU hours received (7,000,000/FY10 and 4,000,000/FY11) would not be wasted on chemically-irrelevant dimer configurations at the CC level. SAPT is an intermolecular perturbation theory that provides accurate intermolecular interaction energies, and circumvents the basis set superposition error inherent in the usual super-molecular approach. Using an aug-cc-pvdz basis set, supplemented by a set of 3s3p2d2f mid-bond functions, SAPT intermolecular energy points were computed and fitted to an exponential-6 function:

$$V_{ij}(r) = \frac{q_i q_j}{r} + A_{ij} \exp\left(-\frac{r}{B_{ij}}\right) - \frac{C_{ij}}{r^6} \tag{3}$$

Atomic charges were determined by fitting to a grid of 59,841 asymptotic electrostatic energies computed using monomer multi-pole moments through L=7 and the remaining parameters were fitted to the SAPT interaction energies using fixed values of the charges. The charges yield a dipole moment value of 3.65 a.u., which compares well with the ab initio value of 3.73a.u.

The C_6 parameters were fitted to ab initio asymptotic dispersion+induction energies using the same center-of-mass separations as used in fitting of the charges. However, since long-range van der Waals interactions are critical components for energetic molecular crystals, the unique asymptotic interactions present in the FOX-7 crystal were fitted with **separate** potentials in order to maximize the accuracy of the fit. Specifically, the FOX-7 crystal is composed of a racemic mixture of chiral 1,1-diamino-2,2-dinitroethylene monomers (Figure 4); therefore, two types of interaction energies have to be determined. Structurally, the monomer consists of amine groups that are essentially coplanar with the C-C bond axis (out of plane twist of ≈0.5°); however, the nitro groups have significant torsions of ≈9° and ≈36° with respect to that plane. The nearest neighbor intra-molecular O…H hydrogen bond distance is 1.97 Å. In the following, each enantiomer will be referred to as either R or S following the Cahn-Ingold-Prelog notational system, where the R,S configuration is determined by the orientation of the nitro group with the 36° torsion with respect to the carbon-carbon bond axis (see Figure 4). The fitted asymptotic R-R and R-S electrostatic+dispersion interactions have root mean square errors of only 6.1×10^{-4} and 1.85×10^{-3} kcal/mol over the 59,841 data point grid.

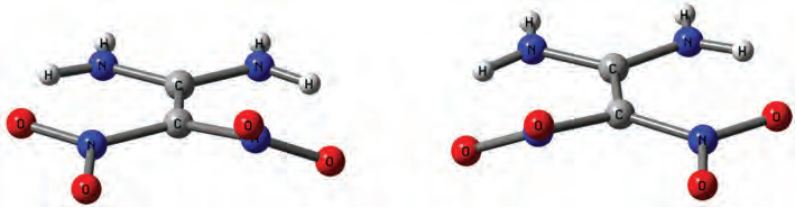


Figure 4. R (left) and S (right) enantiomers of FOX-7 monomer

Using the fixed set of charges and C_6 coefficients, the remaining parameters (*A* and *B* terms of Equation 3) were fitted to SAPT(DFT) dimer interaction energies sampling the repulsive wall, potential well, and dissociative tail typically covering center-of-mass separations of 4 to 15 Å and, similar to the asymptotic fitting strategy, a separate set of parameters was determined for the R-R and R-S interactions. The radial grid was sampled every 0.3 Å, and, at each separation, ±45° rotations of one monomer were taken about the three orthogonal coordinate axes. This resulted in a total of 1,008 *ab initio* data points with 491 and 517 data points corresponding to the R-R and R-S interactions, respectively. The parameters were fitted with the PIKAIA[10] genetic algorithm (GA) using a population of 100 individuals that evolved for 500 generations with fitness scoring determined by the magnitude of the root mean square deviation from the SAPT(DFT) interaction energy reference data set for each individual. The crossover probability during the evolutionary runs was 0.85, and the elitism reproduction scheme was used at all times. Each data point was assigned a weight given by $(E_i+E_o)^{-1}$ where E_o was chosen to be 11.0 kcal/mol for the R-R interactions and 14.0 kcal/mol for the R-S type, and these values were chosen by trial-and-error to maximize the quality of the fit and ensure that points in the important low-energy regions were weighted more heavily than those of positive energy. The root mean square deviation for all points with $E<0$ is 0.10 and 0.12 kcal/mol for the R-R and R-S interactions, respectively. The monomer geometry, fitted charges, and converged parameters of the fit are given in the Appendix.

Using this SAPT-based potential, constant pressure/temperature simulations using a fully flexible simulation cell have been performed within the rigid monomer approximation available in the DLPOLY[11] MD program. A 7×7×5 supercell consisting of 980 FOX-7 molecules was used with a potential cutoff of 21 Angstroms. The total simulation time was 75 picoseconds, with a 1 femtosecond time-step. Statistical averages were accumulated over the final 50 picoseconds of the simulation. Table 1 is a comparison of the simulated structure at 298K and 1 atm for the current SAPT potential and experiment.[12] The agreement between the simulated structure and the experimental data is excellent.

Table 1. Simulated unit cell geometry and volume compared to experiment. Lattice vectors a, b, c in Angstrom, α, β, γ in degrees, and density in g/cc. Values in parentheses are percent error with respect to experiment.

Method	a	b	c	α	β	γ	Density
SAPT	6.932 (0.13)	6.515 (0.83)	11.381 (0.58)	89.990(0.01)	90.310 (0.24)	89.970 (0.03)	1.914 (0.37)
Exp	6.941	6.569	11.315	90.000	90.550	90.000	1.907

The PV isotherm at 300K, resulting from the SAPT(DFT) potential and experiment[13] are shown in Figure 5. The isotherm predicted using the SAPT(DFT) potential is in excellent agreement with experiment over the entire range of pressures. These data were fitted to a third-order Birch-Murnaghan equation of state, and the theoretical bulk modulus and pressure derivative have values of 9.81 GPa and 20.2, respectively, which compare very well with the corresponding experimental values (9.6 ± 1.6 GPa and 20.8 ± 4.3).[13]

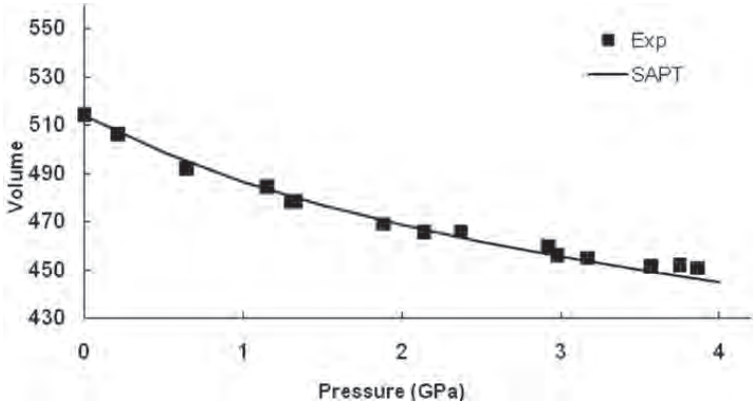


Figure 5. PV isotherm at 298K. Experimental data from Reference 13.

CC calculations are currently being computed for each of the configurations used to parameterize the SAPT potential. The interaction energies at the CC level are determined using the supermolecular approach with a counterpoise correction for the basis set superposition error i.e.,

$$E_{int} = E_{AB} - E_A - E_B$$

where the monomer energies E_A and E_B are computed in the full dimer-centered basis. As a result, the requisite number of computations for generation of the energy grid at the CC level is three-times that required for the equivalent SAPT grid and is a much larger computational effort; therefore, the CC data points are still being determined.

5. Conclusion

The availability of HPC resources, combined with scalable software, is presenting new capabilities to DoD researchers that have previously been deemed computationally infeasible. In years past, computing over 1,000 energy points with CC theory on 28 atoms in a sizable basis set was a veritable impossibility, due to poor computer performance and serial structure of the computer codes. However, with massively-parallel computers and newly implemented parallel quantum chemistry codes designed to take maximum advantage of current hardware, previously impossible work is now being completed and with ever-increasing processor speeds, the molecular size limit to which state-of-the-art quantum chemical methods can be applied will continue to increase.

6. Significance to DoD

This Challenge Project will augment the predictive capability of DoD computational scientists through provision of the best-available reference data for use in simulations and analysis of energetic formulations. Computation of the FOX-7 dimer interaction energies at the coupled cluster level, a method that exceeds the quantitative accuracy of both Hartree-Fock and density functional theory, will lead to development of better potentials that can be used reliably in molecular dynamics simulations that predict the structural and dynamical characteristics of FOX-7 at a variety of temperatures and pressures. Further, with proven success of the FOX-7 potential based on coupled cluster energetics, and using the vast computational resources available through the HPCMP Challenge Program, potentials can be developed for other existing energetics at the coupled cluster level. This will lead to a catalog of high-quality potentials spanning the range of energetic material formulations of current interest to the US military.

Acknowledgments

This work has been supported by the HPC Modernization Program through provision of 7 million CPU hours at the NAVY, ARSC, and ERDC DSRCs.

References

1. Tersoff, J., "Modeling solid-state chemistry: Interatomic potentials for multicomponent systems", *Physical Review B*, 54, pp. 5566–5568, 1989.

2. Daw, Murray S., and M.I. Baskes. "Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals", *Physical Review B*, 29, pp. 6443–6453, 1984.

3. Goddard III, W.A, Zhang, Q., Uludogan, M., Strachan, A., and T. Cagin, "The ReaxFF polarizable reactive force fields for molecular dynamics simulation of ferroelectrics", *Fundamental Physics of Ferroelectrics*, AIP Conference Proceedings No. 535, 2000.

4. Byrd, E.F.C., G.E. Scuseria, and C. Chabalowski. "An ab initio study of solid nitromethane, HMX, RDX, and CL20: Successes and failures of DFT", *Journal of Physical Chemistry B*, pp. 13100–13106, 2004.

5. Bartlett, R.J., "Coupled cluster theory: An overview of recent developments", *Modern Electronic Structure Theory*, Singapore, World Scientific Publishing Co., 1995.

6. Raghavachari, K., G.W. Trucks, J.A. Pople, and M. Head-Gordon, "A fifth order perturbation comparison of electron correlation theories", *Chemical Physics Letters*, pp. 479–483, 1989.

7. Lotrich,V., N. Flocke, M. Ponton, A. Yau, A. Perera, E. Deumen, and R.J. Bartlett. "Parallel implementation of electronic structure energy, gradient, and hessian calculations", *Journal of Chemical Physics*, pp. 194104-1–194104-15, 2008.

8. ACES II is a program product of the Quantum Theory Project, University of Florida. Authors: J.F. Stanton, J. Gauss, S.A. Perera, J.D. Watts, A.D. Yau, M. Nooijen, N. Oliphant, P.G. Szalay, W.J. Lauderdale, S.R. Gwaltney, S. Beck, A. Balkova, D.E. Bernholdt, K.K. Baeck, P. Rozyczko, H. Sekino, C. Huber, J. Pittner, W. Cencek, D. Taylor, and R.J. Bartlett, Integral packages included are VMOL (J. Almlöf and P.R. Taylor); VPROPS (P. Taylor); ABACUS (T. Helgaker, H.J. Aa. Jensen, P. Jørgensen, J. Olsen, and P.R. Taylor); HONDO/GAMESS (M.W. Schmidt, K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.J. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S. Su, T.L. Windus, M. Dupuis, J.A. Montgomery).

9. Podeszwa, R., R. Bukowski, and K. Szalewicz, "Density fitting method in symmetry-adapted perturbation theory based on Kohn-Sham description of monomers", *Journal of Chemical Theory and Computation*, pp. 400–412, 2006.

10. See the PIKAIA homepage at http://whitedwarf.org/parallel/.

11. http://www.cse.scitech.ac.uk/ccg/software/DL_POLY.

12. Bemm, U. and H. Ostmark, Acta Crystallogr., C54, 1998.

13. Peiris, S., C. Wong, and F. Zerilli, J. Chem. Phys., 120, 8060, 2004.

Appendix A

Table A-1. Coordinates (Angstroms) of the FOX-7 monomer (R enantiomer) used in potential fitting and simulation. The S configuration is generated by reflection of the atomic coordinates below through the xz plane.

Atom	X	Y	Z
C1	-0.02155	0.01502	0.02857
C2	0.53064	-0.45663	1.29036
N3	-1.38939	0.17793	-0.21164
N4	0.85035	0.2896	-1.06713
N5	-0.26101	-0.8488	2.26997
N6	1.84152	-0.54663	1.46426
O7	-1.78082	0.74484	-1.25755
O8	-2.20401	-0.19519	0.65844
O9	1.97937	0.74746	-0.82794
O10	0.49545	0.03538	-2.23025
H11	0.07243	-1.12864	3.0372
H12	-1.12705	-0.83961	2.1773
H13	2.14205	-0.83961	2.18861
H14	2.38338	-0.22212	0.91008

Table A-2. Atomic charges corresponding to atom labeling in Table A-1

Atom	CHARGE (a.u.)
C1	-0.4100270
C2	0.7362470
N3	0.6745610
N4	0.7220050
N5	-0.8923120
N6	-0.9515670
O7	-0.3605990
O8	-0.4836020
O9	-0.4675610
O10	-0.3729130
H11	0.4209090
H12	0.4655270
H13	0.4564320
H14	0.4629000

Table A-3. Potential parameters for R-R interactions

i	j	A_{ij} (kcal/mol)	B_{ij} (Ang)	C_{ij} (Ang^6 * kcal/mol)
C	C	764046.0121	0.2086	0.0801
C	N	72.8379	0.1816	19.0796
C	O	88075.9408	0.2529	250.2805
C	H	3879.6614	0.3185	204.8703
N	N	111518.8046	0.2775	822.6244
N	O	5658.5589	0.3329	645.8825
N	H	10158.7541	0.2491	0.2165
O	O	88046.0549	0.2425	129.6677
O	H	73731.6262	0.2196	278.7350
H	H	11.2533	0.7838	5.7440

Table A-4. Potential parameters for R-S interactions

i	j	A_{ij} (kcal/mol)	B_{ij} (Ang)	C_{ij} (Ang^6 * kcal/mol)
C	C	512900.0918	0.2501	0.0732
C	N	60.4168	0.1845	12.7313
C	O	67862.0867	0.2467	169.1749
C	H	2577.8179	0.2844	204.7925
N	N	80398.8032	0.2827	1116.5450
N	O	6696.2147	0.3711	903.9598
N	H	14899.0853	0.2263	0.3438
O	O	82609.7660	0.2525	243.1237
O	H	42344.2009	0.2104	252.6077
i	j	A_{ij} (kcal/mol)	B_{ij} (Ang)	C_{ij} (Ang^6 * kcal/mol)

HPCMP UGC 2011

3. Computational Chemistry and Materials Science (CCM)

Materials Science

A Scalable RVE-TFA Analysis of Woven Composites

Ramakrishna R. Valisetty
US Army Research Laboratory, Computational & Information Sciences Directorate (ARL/CISD), Aberdeen Proving Ground, MD
rama@arl.army.mil

A. Rajendran
Department of Mech. Engineering, University of Mississippi, University, MS
raj@olemiss.edu

Abstract

As new materials are developed for the Army's future combat needs spanning micro-to-macro scale lengths, a need has arisen for parallel computational codes that are capable of investigating the material systems at the multiple-length scales at which they are synthesized. Such codes permit modeling of local damage modes and their progression, and thus reliable predictions of the overall response for the composites. Although many multi-scale formulations are developed for the composites, such as homogenization, transformation field, Voronoi cell, etc., addressing many microstructure types, they are demonstrated mostly through stand-alone codes. Before being accepted for wider use in the Department of Defense (DoD) communities, the codes that implement these theories need to be made parallel and scalable and then integrated into the global codes. In this work, a representative volume element (RVE)-based transformation field analysis (TFA) model of a 3D woven composite was made parallel, and the scalability achieved in the computations is being reported. The RVE-TFA method works in two stages, first by calculating the stresses in the undamaged RVE, and in the next stage by correcting them for the locally spreading damage modes. The local stress corrector problem increases in size with the spreading local damage modes, and would benefit by having its computations made parallel. In the current effort, the computations in the RVE pre-processor, the part of the code used for generating the undamaged RVE stresses, are made parallel first. The stresses so generated are consolidated, and finally the computations in the RVE-TFA analysis are also made parallel. The paper will present some preliminary results on the role of the RVE mesh size and of the spreading damage modes in increasing the TFA problem size and how the current parallelization effort helps in reducing the overall solution times.

Keywords: multi-scale computing, high performance computing, RVE-TFA analyses, 3D woven composites

1. Introduction

While the soldier-protecting armor and vehicle structures are designed to survive under blast and projectile impacts, the global codes consider structures only in one length scale. But the material systems used in the armor structures often involve synthetically-engineered materials consisting of several length scales, e.g., fiber-reinforced composites, particulate metal matrix composites, ceramics, and metals. The design process used for the armor systems seeks to localize the impact events within the high-strength materials, such as ceramics or metal matrix composites, and to minimize the impact energy and momentum transfer into soldiers and their vehicles using resins and elastomers. When two such dissimilar materials are required to function together, they require yet a third type, in the form of woven composites or encapsulations, to hold them together giving shape and rigidity to the eventual armor structure. Composites used in such situations often carry weave features such as stitching in the three-dimensional (3D) direction, to provide resistance to de-lamination failures, but nevertheless introduce into the armor design process additional design features. When such an armor structure is under impact, the three constituent material systems interact, deform and fracture, hopefully as designed, in a controlled manner; while also undergoing myriad damage modes depending on how the overall impact affects the specific material constituents.

Other than vast experimental data sets, no procedures are currently available to designers that tell them why certain combinations of materials and geometry are better at providing superior impact resistance, i.e., how the overall impact is absorbed by the damage modes spreading locally in the material constituents of the armor structure. Experiments reveal the dominant damage modes in the armor constituents. Designers explain them from the traditional damage mechanics perspective of failures on principal stress planes, but have not transferred such knowledge into the computational structural codes. The currently available codes consider structures only in one length scale and are capable of modeling only the linear displacement-based stress resolution.

Porting the damage mechanics understanding of experimental databases into explicit time integration-based structural codes has not progressed, owing to the difficulties in modeling the involved damage modes across the multiple-length scales of the armor material systems. By using averaged properties and thus ignoring length scales, these codes often focus only on how the overall impact response is affected by the built-up nature of an armor configuration, but not on how the local damage mechanics is enabling such improvement. What is needed then within the codes is a method to explicitly model the individual damage modes, i.e., damage mechanics handled at least on two length scales: one the familiar global scale of the armor and the other the local scale of the armor constituents.

There are currently available many multi-scale models that are demonstrated to have excellent capability in modeling the local damage modes that develop within a micro-structure that is put together with a set of representative volume elements (RVE). Among these models, the ones developed for the structural impact problems can be placed in the following categories: homogenization (Terada, 1995), transformation field (Dvorak, 1992), Voronoi cell (Ghosh, et al., 1995), and eigen deformation-based reduced-order homogenization, (Oskay, 1997). These models were demonstrated in several applications: metal matrices (Chung, 1999), 3D woven composites, (Bahei-El-Din, et al., 2004), plane weave composites, (Oskay, 2007), etc. Although these applications demonstrated the superior modeling capability of the underlying multi-scale formulations, the demonstrations were made mostly using the stand-alone codes, and are rarely integrated into the global computational codes.

In the rare instances when such integration occurred, the integration was done in the form of user-defined material "UMAT" models to provide only the overall constitutive behavior to the global analysis. The impediment to analysis integration across all armor constituent length scales is evident when one considers the complexity of a simple armor configuration consisting of penetration resistant tiles, tile encasing fiber composites, and metal backing structures. It takes up to 1,000–100,000 finite-elements for modeling such a configuration in local length scales. The resulting local-level micro-structural problem with 3,000–300,000 degrees-of-freedom turns out to be as big as the global problem. Given this situation, researchers hard-code "specific" micro-level behaviors into the "UMAT" subroutines, thus simplifying the local material failure and fracture behaviors that are key to solving the detailed micro-level problem. What is not recognized is that the infrastructure available in the global finite-element codes, i.e., the numerical integrators of the equations of motion, the material constitutive and strength models, contacts, and element formulations, etc., all work equally well in the millimeter to micro-meter scale range that spans the armor material system constituents. By making the micro-level structural analyses as versatile as the global analyses by exploiting the finite-element infrastructure available in the global codes, the impact analyses' integration can be made possible across all armor constituent length scales.

Multi-scale analyses will gain wider use in the Department of Defense (DoD) design communities if they are demonstrated in the global codes. Several issues are to be addressed before such demonstration can take place. To begin with, for example, there is the issue of which formulation is numerically efficient. Several implementations of the homogenization method used displacement formulation for solving the micro-structural problem, while Voronoi cell and transfer field analyses used stress formulation to address directly the locally developing damage modes. While the mesh size controls the local problem size for the first type of formulation, the number of stresses active in the local damage modes control the local problem size in the second type of formulation. Even a hybrid stress-displacement formulation (Hu, 2008) was used for adapting the RVE mesh to the locally spreading deformation. There are additional issues which are more familiar: 1) material approximation in a micro-structure, e.g., fibers or fiber bundles, extant of interface regions, 2) selecting the convergence criteria for fixing local meshes, and 3) adapting the local mesh to the locally developing deformation. The convergence effect that follows from having the increasing micro-structure mesh sizes is investigated in terms of the progressive damage for a 3D woven composite by Valisetty, et al. (2010a). The global-local theory used in this work was the representative volume element (RVE)-based transformation field analysis (TFA) of Bahei-El-Din, et al. (2004). In Valisetty (2010b), the local solution matrix size is shown to depend not only on the local mesh size but also on the damage modes triggered in a micro-structure by the applied global strain increments. The size of the local stress corrector problem tracks the spreading of the local damage modes. This underpins the numerical advantage for the RVE-TFA methods.

Increasing the local mesh size increases the size of the local solution matrix. However, the micro-level computations can be made parallel and the total global-local multi-scale analyses can be made manageable. In this work, the RVE-TFA model of Bahei-El-Din, et al. (2004) was made parallel, and the scalability achieved in the computations is being reported. The RVE-TFA method works in two stages, first by calculating the stresses in the undamaged RVE, and in the next stage by correcting them for the locally spreading damage modes. In the current effort, the computations in the RVE pre-processor, the part of the code used for generating the undamaged RVE stresses, are made parallel first. The stresses so generated are consolidated, and finally the computations in the RVE-TFA analysis are also made parallel. The paper will present some preliminary results on the role of the RVE mesh size, and of the spreading damage modes in increasing the TFA problem size and how the current parallelization effort helps in reducing the overall solution times.

2. RVE-TFA Formulation

The RVE-TFA formulation is adopted in two stages. In the first stage, it is assumed that there is no damage in the RVE, and the local stresses are computed to be elastic. When the local stresses are approximated in such a way, they may not represent the on-going damage in the RVE micro-structure correctly. This situation is addressed in the second stage by adding certain self-equilibrating local stresses as corrections to the elastic stresses. The damage spreading in the RVE is considered in terms of some pre-selected damage modes, and each local stress component involved in those damage modes is corrected to a level that is dictated by some damage criteria. Corrections introduced in this manner to the initially approximated elastic stresses would disturb the local RVE equilibrium. Using a TFA in the RVE, the local equilibrium is restored, and the stress corrections are determined as linear aggregates of the self-equilibrating local stress, or transformation, fields.

The aforementioned process is repeated each time the overall strain on the RVE is incremented. Although computing the first approximation of the local stresses under the elasticity and zero damage assumptions is straightforward, the computations can be speeded up by having sets of stress and strain concentration factors available at the beginning of an RVE-TFA analysis. Similarly, for coming up with self-equilibrating stress fields for making the stress component corrections, it helps to have the necessary stress transformation factors readily available. The two sets of factors, stress concentration and the transformation, are determined with an RVE pre-processor code. As the number of elements in an RVE micro-structure increases, the computations in this code increase, requiring a parallel computation effort. As the overall strain increment on the RVE is increased, more damage modes spread into the more local elements. This increases the TFA problem size, requiring another parallel solution effort. The RVE-preprocessor computations and the TFA equations are presented in the following sections; the parallelization efforts are described next.

2.1 RVE-Preprocessor Computations

Following Hill (1965) and Dvorak (1992), and using the notation of boldface lower case for representing the (6×1) engineering stress/strain vectors, and boldface upper case letters for the corresponding stiffness/compliance matrices, parts of the local stresses, $\boldsymbol{\sigma}$, and strains, $\boldsymbol{\varepsilon}$, are related to their global counterparts, $\bar{\sigma}$ and $\bar{\boldsymbol{\varepsilon}}$, and the rest to a superposition of all the transformation fields arising in all the local elements, as:

$$\boldsymbol{\varepsilon}_r = \mathbf{A}_{rk}\bar{\boldsymbol{\varepsilon}}_k + \sum_{s=1}^{Q}\mathbf{D}_{rs}\mu_s, \boldsymbol{\sigma}_r = \mathbf{B}_{rk}\bar{\boldsymbol{\sigma}}_k + \sum_{s=1}^{Q}\mathbf{F}_{rs}\lambda_s, \qquad k = 1,2,,,6; r = 1, 2,\ldots,Q. \tag{1}$$

where **A** and **B** are the strain and stress concentration factors, **D** and **F** are the transformation strain and stress factors, and $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ are the transformation strains and stresses, respectively. Q is the number of local elements in the RVE mesh.

Both sets of the concentration factors and the transformation factors depend upon the geometry and properties of the RVE micro-structure, and can be determined from an elastic analysis of its mesh. The concentration factors are statically equivalent to the global stresses. For example, a k^{th} column, k=1,2,…,6, of the stress concentration factors matrix, $\mathbf{B}_{rk}$, r = 1,2,…,Q, is computed by giving a unit value to the k^{th} stress component of the 6×1 $\bar{\sigma}$, and 0 to the rest. A similar procedure is also used for computing the transformation stress factors. By realizing that a kth column, k=1,2,…,6, of $\mathbf{F}_{rs}$, r,s=1,2,…,Q, is a response in the local element V_r after a unit stress, $\boldsymbol{\lambda}_k$=1, is applied in the local element V_s, a total of 6Q RVE finite-element solutions are used to completely evaluate the transformation factors.

2.2 TFA Equations

The equations of the transformation stress, or TFA, problem are generated by considering how each of the 6Q local stress components is participating in the locally spreading damage modes. If a stress component in an RVE local element is found to be involved in a damage mode, then a correction to it is proposed. Since this correction upsets the local RVE equilibrium, a self-equilibrating stress-field in proportion to the correction is used. Although the RVE equilibrium is maintained, the local stresses in the other RVE elements are changed by this correction, further affecting the damage situation in those elements. The stress correction process is repeated to compute the local stresses in the presence of all possible damage modes caused under all local stress corrections. The transformation stresses are evaluated by setting the local stresses on the left-hand side of Equations 1 at values that can be sustained in the presence of progressing damage modes.

To present the TFA equations matrix, local stresses are considered in local coordinate systems with appropriate co-ordinate transformation matrices, $\mathbf{R}_\rho$. Referring to such local coordinate systems and using stress ratios, $\varphi_k^{(\rho)}$, which are current stresses divided by their elastic undamaged values, the equations for computing transformation stress are presented as (Bahei-El-Din, et al., 2004):

$$\sum_{\varsigma=1}^{Q^*} \mathbf{F}_{\rho\eta} diag(\boldsymbol{\alpha}_k)\boldsymbol{\lambda}_\eta = -\mathbf{I} - \mathbf{R}_\rho^{-1} diag(\boldsymbol{\phi}_k^{(\rho)})\mathbf{R}_\rho)\mathbf{B}_\rho \bar{\boldsymbol{\sigma}},\ \rho = 1,2,,,Q^*, \tag{2}$$

$$\alpha_k = \begin{cases} 1\ if\ \ 0 \le \varphi < 1 \\ 0\ if\ \varphi = 1 \end{cases}, \tag{3}$$

where diag(x) is a (6 Q^*×6 Q^*) matrix, and Q^* is the number of stress components that are being corrected for the spreading damage. The stress ratios, $\varphi_k^{(\rho)}$, k = 1,2,…,6, ρ=1,2,…, Q^*, $Q^* \le Q$, imply pre- and post-damage loading as follows: $\varphi_k^{(\rho)} = 1$ for a stress component that has yet not violated the onset of damage, $\varphi_k^{(\rho)} = 0$ indicates complete unloading, and $0 \le \varphi_k^{(\rho)} \le 1$ indicates progressing damage with partial property decay. If $\varphi_k^{(\rho)} < 1$ for all k=1,2,…,6 and for all Q* sub-volumes, then there are 6 Q^* equations in Equations 2 for evaluating the transformation fields $\boldsymbol{\lambda}_r$ which number to 6 Q^*. If some of the local stress components are unaffected by the underlying damage criteria, the corresponding stress ratios would equal unity. In such cases α_k=0, and the corresponding transformation stresses will not be needed and a corresponding number of equations will drop out from Equations 2. The equations available after this consideration are solved for the transformation stresses.

3. RVE-TFA Parallelization Efforts

The parallelization effort proceeded in two parts. The results from the RVE pre-processor parallelization effort are first described. These are followed by the results from the TFA solution parallelization effort. The results of the RVE-TFA parallelization effort are demonstrated for a 3D woven glass/epoxy composite that was first evaluated by Bahei-El-Din, et al. (2004). As the global strain on an RVE is incremented, more damage modes appear and spread into more local RVE elements. Capturing such damage growth requires more details in the RVE mesh. Figure 1 shows seven RVE meshes used in the study. Starting from top left and going right, these meshes have 150, 294, 490, 882, 1,458, 2,106, and 2,366 elements, respectively. The preponderance of local elements in the thickness direction was in view of the composite's application in armor impacts. The composite was made with large woven fiber bundles.

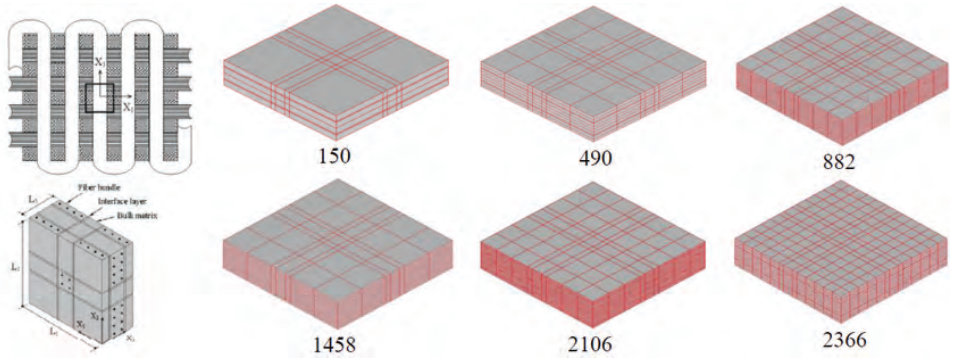


Figure 1. An RVE for a 3D composite and different finite-element meshes

3.1 RVE Pre-Processor Parallelization Effort

RVE-TFA-based analyses carry enormous initial inputs in the form of the stress concentration, $\mathbf{B}_r$, and the transformation stress factors, $\mathbf{F}_{rs}$. These inputs help to readily postulate the needed local stress corrections as per Equation 1. Among the RVE pre-processor computations, the effort required for computing $\mathbf{B}_r$ is relatively straightforward. It involves putting the RVE mesh under static equilibrium, and solving it for the local stresses as statically equivalent to the global stress components that are applied with a unit value, one at a time. Although six solutions are implied here, all it required was building the RVE stiffness matrix once, and using its inverse repeatedly. The same process is used for computing the transformation stress factors, $\mathbf{F}_{rs}$, but now there were 6Q self-equilibrating local stress systems that are determined as

statically equivalent to 6Q unit stress components, each of which are applied with a unit value, one at a time. Recognizing that the inverse of the RVE elastic stiffness problem can be re-used makes the parallelization effort required for the pre-processor computations simple. The task of computing the 6Q local stress systems was equally apportioned to the available compute processors. The processors were all given the whole RVE mesh, but were directed to focus only on computing the sub-blocks of the 6Q local stress systems that were assigned to them.

Among the sets of the $\mathbf{B}_r$ and $\mathbf{F}_{rs}$ factors, the Frs set of factors constitutes a voluminous set, so it is given attention in the pre-processor computations. As can be seen from the 3rd column in Table 1, the full counts of these sets of factors depend on Q, the number of the elements in the RVE mesh, and come in at 36Q2. Using the considerations of similarity, these counts were reduced by about 69%.

Table 1. Mesh size vs. size of the set of the Transformation Influence Factors

No. of RVE mesh elements	No. of RVE mesh nodes	No. of F_{rs} factors	Reduced no. F_{rs} of factors	% reduction in no. F_{rs} of factors
150	252	810,000	249,912	69.15
294	448	3,111,696	972,900	68.73
490	704	8,643,600	2,689,848	68.88
882	1216	28,005,264	8,869,896	68.33
1458	1900	76,527,504	23,748,408	68.97
2106	2700	159,668,496	49,854,888	68.78
2366	2940	201,526,416	62,798,400	68.84

The scalability achieved in making the $\mathbf{F}_{rs}$ computations parallel is shown in Figure 2 for the 294, 490, 882, 1,458, 2,106, and 2,366 element RVE meshes. The multi-processor runtimes were non-dimensionalized with respect to the corresponding serial, i.e., one processor, runtime. This figure shows that about 50% scalability was achieved for the present RVE pre-processor parallelization effort.

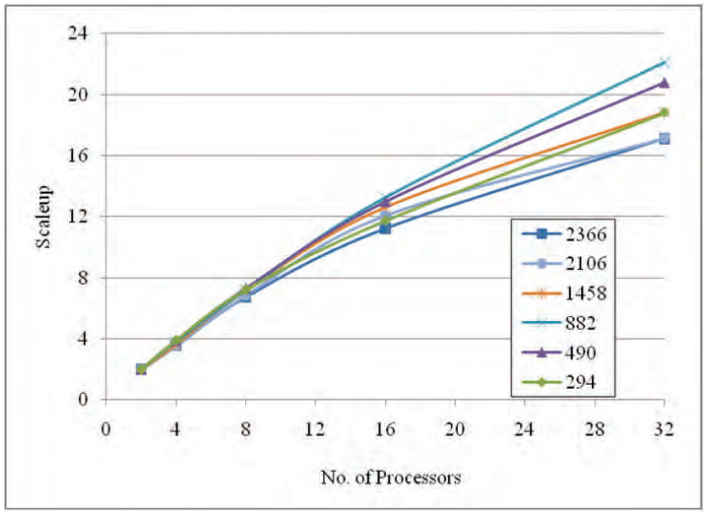


Figure 2. Scale-up achieved in the RVE pre-processor parallelization effort

The RVE stiffness matrix was assembled over all the RVE elements and was inverted. Therefore, there was a scope for putting in additional parallelization by using the traditional domain decomposition-based parallel stiffness matrix assembly and parallel matrix inversion methods, but this was not attempted because significant additional reduction in computational time was not expected from such an effort.

3.2 TFA Parallelization Effort

The parallelization effort required for solving the TFA problem proceeded on a different tack. When the stress components in the damaged RVE local elements are sought for corrections, the corrections are introduced as stress resizing ratios in Equation 2. The TFA problem matrix is matrix of the equations of such unknown ratios which are solved in this effort. Unlike the RVE stiffness matrix, mentioned in the previous section, the TFA matrix is not assembled over all the RVE elements, but over a set of damage causing stress components. As the global strain is incremented, and as more damage modes are spread into more local RVE elements and new damage modes are initiated, more local stress components turn up requiring stress corrections increasing the size of the TFA problem matrix. Since which local elements and which local stress components would cause the damage to spread cannot be predicted, the traditional domain decomposition-based parallel stiffness matrix assembly and matrix inversion methods cannot be used for obtaining the TFA solution. Also, not all TFA problems would benefit by having their micro-calculations made parallel. This happens when the TFA matrix sizes are too small because RVE mesh sizes are too small, or when applied global strain levels are just starting to cause local damage.

The size of the TFA problem depends on the nature and extent of the damage spreading in the RVE. Damage in RVE-TFA analyses is considered in the form of some pre-selected damage modes, e.g., fiber bundle splitting, fiber bundle sliding, bulk matrix failure, matrix interface failure, etc. By keeping track of the global strains applied to the RVE and monitoring the local stresses in the RVE elements, the spread of the damage modes is tracked. Even in simple global strain conditions, the present composite develops complex damage modes, e.g., see Figure 3, Valisetty (2010b). In the figure, the spread of the damage modes in the 1,458-element RVE mesh is shown as the global strain is incremented from uni-axial compression to uni-axial tension. Counting the participating local RVE element totals for each of the considered damage modes, the spread of the damage modes is shown as percentages of the total RVE elements.

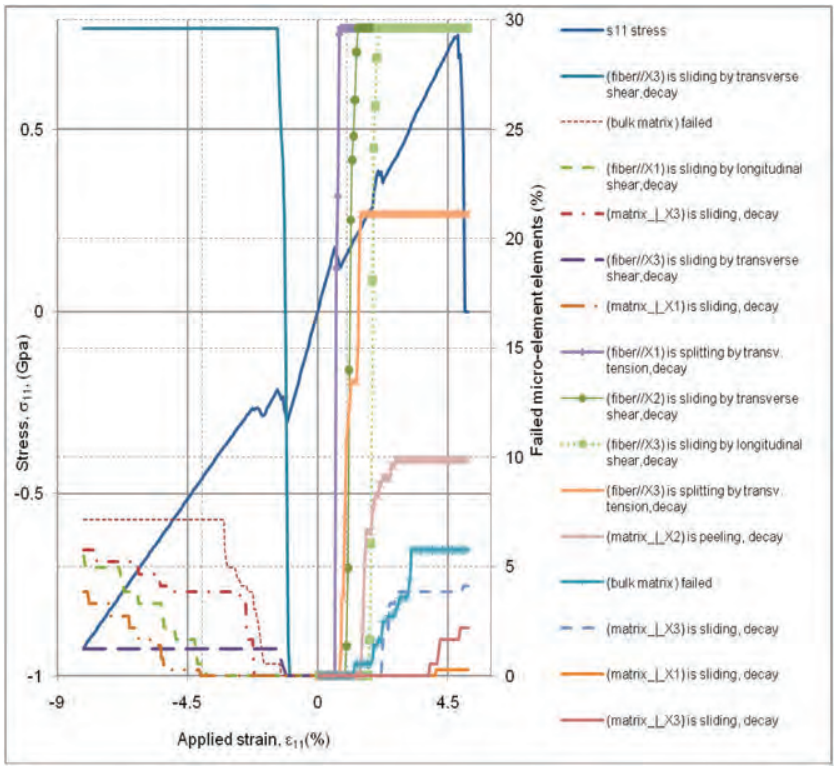


Figure 3. Participation of the local RVE elements in some selected damage modes

When an RVE mesh is refined with increasing numbers of the local elements, scope for refining the damage modeling increases. When this is coupled with the increasing global strains, more damage modes appear and spread into more local elements, increasing the numbers of the local stress components requiring correction. Thus the TFA problem matrix size changes from one global strain increment to the next. For a series of unit value applications for each of the six global strain components, the sizes of the TFA problem matrices are shown in Figure 4 against the RVE mesh sizes. The figure shows clearly that the TFA matrix size varies from being very small for smaller RVE meshes to being very big for larger RVE meshes subjected to larger global strain levels.

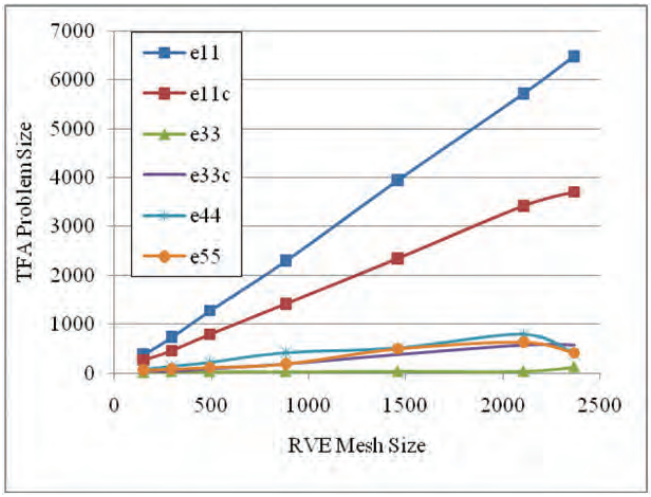


Figure 4. TFA problem size vs. RVE mesh size for the six global strain components

Now coming to the TFA parallelization effort, the TFA problem is solved only when there is local damage. Therefore, as the global strain is incremented on the RVE, the TFA problem is solved anew in the following three computational stages: 1) checking the local RVE elements for the spread of old damage modes and for the initiation of any new damage modes, 2) assembling the TFA matrix, and 3) solving the TFA matrix. Computational efforts required for the first two stages are found to be small for the RVE mesh sizes considered for the example composite. For increasing TFA mesh sizes, the computational effort involved in the third matrix solution stage is found to overwhelm the TFA computations. Accordingly, this stage of the computations was made parallel with the aid of subroutines from the SCALAPACK numerical library. All the available compute processors were directed to obtaining the solution of the TFA matrix.

The scalability achieved in the TFA parallelization effort is shown in Figure 5 for the 1,458, 2,106, and 2,366 element RVE meshes. The multi-processor runtimes were non-dimensionalized with respect to the corresponding serial, i.e., one processor, runtime. This figure shows that about 100% scalability was achieved for the 2 processor runs and about 66% for the 4 processor runs.

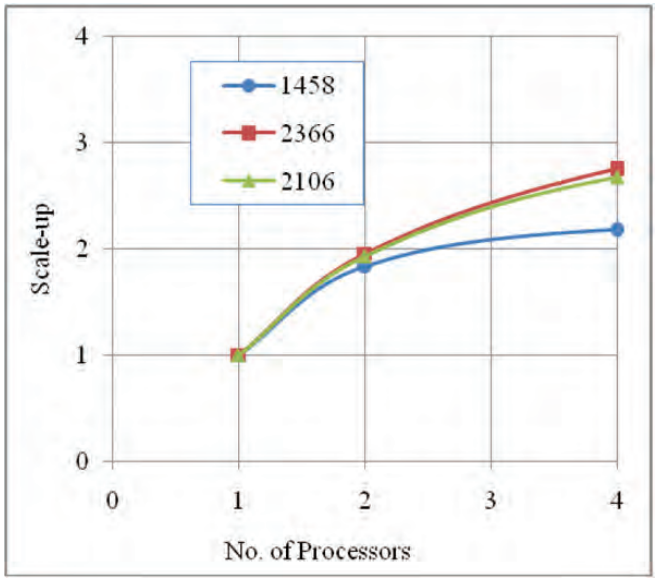


Figure 5. Scale-up achieved in the TFA parallelization effort

The results in Figure 5 show a drop-off in the achieved scale-up for the smaller RVE mesh sizes. Similarly, the scale-up is also shown to be dropping off for the higher compute processor runs. This is to be expected, but this is not a problem. This is because not many compute processors are expected to be available for micro-level computations anyway, i.e., in the presence of an on-going parallel global level analysis. Strategies for making both the global and local

level computations simultaneously parallel are not well developed, and the scale-up issues can be addressed only when processing requirements for both the global and local meshes are available. In the present work, the parallelization was implemented with a strategy that relied only on explicit communication between the co-working local processors. With the availability of multi-core compute processor nodes with shared memory programming methods, opportunities exist for introducing additional parallelization into the local RVE analyses; this has to be explored in the future.

4. Conclusion

In this work, the RVE-TFA model of Bahei-El-Din, et al. (2004) was made parallel, and the scalability achieved in the computations was reported. The computations in the RVE pre-processor, i.e., the part of the code used for generating the many statically equilibrating stress systems needed for generating the TFA problem matrix, were made parallel first. Robust scale-up was reported for this part of the parallelization effort for several small-to-large sized RVE meshes and for processors up to 32. Computations in checking for the spread of the damage modes in the RVE elements, assembling the TFA problem matrix, and in obtaining a solution for the TFA matrix were made parallel under the TFA parallelization effort. Results indicated that, for increasing TFA mesh sizes, the TFA matrix solution takes up most of the TFA computations, and the parallelization effort that was put in has potential for achieving scalability. It was; however, noted that for putting the reported TFA scalability in the context of what can reasonably be achieved in a simulation in which both global and local level analyses are simultaneously running in parallel, the processing requirements for both the global and local meshes are required. It was also noted that the currently available multi-core compute processor nodes and the shared memory programming methods have to be explored for introducing additional parallelization into the local RVE analyses.

Acknowledgements

This work was supported by a grant of computer time from the DoD High Performance Computing Modernization Program at the US Army Research Laboratory, Department of Defense Supercomputing Resource Center, Aberdeen Proving Ground, MD.

References

Bahei-El-Din, Y.A., A.M. Rajendran, and M.A. Zikry, "A micromechanical model for damage progression in woven composite systems", *International Journal of Solids and Structures*, Vol. 41, pp. 2307–2330, 2004.

Chung, P.W. and K.K. Tamma, "Woven Fabric Composites Developments in Engineering Bounds, Homogenization and Applications", *International Journal of Numerical Methods in Engineering*, Vol. 45, pp. 1757–1790, 1999.

Dvorak, G.J., "Transformation fields analysis of in-elastic composite materials", *Proceedings of the Royal Society*, Vol. A437, pp. 311–327, 1992.

Fish, J., W. Chen, and G. Nagai, "Non-local dispersive model for wave propagation in heterogeneous media. part 1: one-dimensional case", *International journal of Numerical Methods in Engineering*, Vol. 54, pp. 331–346, 2000.

Ghosh, S., K. Lee, and S. Moorthy, "Elasto-plastic analysis of arbitrary heterogeneous materials with the Voronoi cell finite-element method", *Computer Methods in Applied Mechanics and Engineering*, Vol. 121, pp. 373–409, 1995.

Hu, C. and S. Ghosh, "Locally enhanced Voronoi cell finite-element model (LEVCFEM) for simulating evolving fracture in ductile microstructures containing inclusions", *Int. Journal for Numerical Methods in Engineering*, Vol.76, pp. 1955–1992, 2008.

Oskay, C. and J. Fish, "Eigen deformation-based reduced-order homogenization for failure analysis of heterogeneous materials", *Computational Methods in Applied Mechanics and Engineering*, Vol. 196, pp. 1216–1243, 2007.

Terada, K. and N. Kikuchi, "Nonlinear homogenization method for practical applications, computational methods in micromechanics", *American Society of Mechanical Engineers*, AMD, Vol. 212, pp.1–16, 1995.

Valisetty, R., R. Namburu, A. Rajendran, and Y. Bahei-El-Din, "Scalable coupling of TFA and PARADYN analyses for impact modeling of 3-d woven composites", *Proceedings of the 2004 International Conference on Computational & Experimental Engineering & Sciences*, ISBN: 0-9657001-6-X2004, Tech Science Press, pp. 92–97, 2004.

Valisetty, R., A.M. Rajendran, and Y. Bahei-El-Din, "Scalable Computing of the Mesh Size Effect on Modeling Damage Mechanics in Woven Armor Composites", *65th Army Science Conference*, Orlando, FL, 2008.

Valisetty, R., A.M. Rajendran, and D. Grove, "Mesh effects in predictions of progressive damage in 3D woven composites", *CMES*, Vol.60, no. 1, pp. 41–71, 2010a.

Valisetty, R. and A. M. Rajendran, "Multi-Scale Modeling Capable Global Codes", *Proceedings of the 2010 International Conference on Computational & Experimental Engineering & Sciences*, Mar 28-Apr 1, Las Vegas, USA, 2010b.

The ScaLAPACK Home Page, www.netlib.org/scalapck/scalapack_home.html.

Ab-Initio Molecular Dynamics Simulations of Molten Ni-Based Superalloys

Christopher Woodward
US Air Force Research Laboratory, Materials and Manufacturing Directorate (AFRL/RX), Wright-Patterson AFB, OH
christopher.woodward@wpafb.af.mil

James Lill
High Performance Technologies Inc. (HPTi), Wright-Patterson AFB, OH
james.lill@wpafb.af.mil

Abstract

Convective instabilities responsible for misoriented grains (freckle defects) in directionally solidified turbine airfoils are produced by variations in liquid-metal density with composition and temperature across the solidification zone. Here, fundamental properties of molten Ni-based alloys, required for modeling these instabilities, are calculated using ab initio molecular dynamics simulations. Previous calculations on elemental, binary and ternary alloys produced liquid-phase molar volumes (V(c,T)) within 0.6–1.8% of available experimental data, and demonstrated that the simulations cells of 500 atoms were sufficient to converge properties of interest. In this work, density inversion, an increase in density with increasing temperature, is assessed in model Ni-Al-W and RENE-N4 alloys. Calculations are performed using a recently implemented constant pressure methodology (NPT) which enables efficient simulation of highly-complex alloys. Density inversion (~2%) is found for the Ni-Al-W alloys for target compositions and temperatures consistent with 0 and 0.5 solid-volume fractions in the melt. Recently published parameterizations of V(c,T) developed using binary experimental data from a narrow range of compositions do not predict a density inversion. The liquid-metal density for a Ni based superalloy, RENE-N4, is calculated at the liquidus and solidus temperatures expected in the solidification (mushy) zone. Multiple 500-atom instantiations of the superalloy melt produce densities well within expected numerical error. This illustrates that simulations of moderate spatial and temporal scales sample enough degrees-of-freedom to properly represent the configuration entropy found in a liquid-metal superalloy.

1. Introduction

Over the last decade, alloy developers have increased the concentration of refractory elements in single-crystal Ni-based superalloys in order to improve the balance of high-temperature properties. Unfortunately, higher concentrations of some of these elements are associated with the formation of defects during the directional solidification process. These so-called freckle defects are formed by enhancing density-driven convective instabilities in the solidification front (i.e., the mushy-zone), and consist of chains of small equiaxed grains. The high-angle grain boundaries and composition gradients associated with freckles are known to severely degrade turbine airfoil performance. A systems design of the processing-structure-properties relationship for these alloys requires a quantitative model for predicting processing regimes, which will allow defects-free commercial castings over the widest possible range of refractory metal compositions. Such casting defects form due to thermo-solutal convection in the mushy-zone, and current theory indicates such instabilities are likely when a critical Rayleigh- number is exceeded.[1–4] The Rayleigh-number (R) is a measure of the ratio of the buoyancy force to the retarding frictional force in the mushy-zone:

$$R = \left(\Delta\rho / \bar{\rho}\right) gKl / \alpha v \tag{1}$$

where l is an appropriate length scale, K is the average permeability, g gravitational acceleration, α is the thermal diffusivity, v is the kinematic viscosity of the fluid, and $\left(\Delta\rho / \bar{\rho}\right)$, the density contrast, is a measure of the density variation over the mushy-zone between the solid and liquid phases. The tendency towards defect formation is highly-sensitive to the permeability of the mushy-zone (e.g., the primary dendrite arm spacing), as well as the density contrast. Historically, models for the Rayleigh-number criterion for freckle-formation assume that the density gradients are aligned with both the solidification front and gravity. However, this assumption has become problematic as turbine airfoil geometries have become more complex with the introduction of cooling channels and other design features. Currently, most castings have significant heat losses through the mold that create horizontal thermal gradients and tilted solid-liquid interfaces.

Validating mathematical models for freckle-formation in specific alloy systems requires accurate assessment of the various parameters in Equation 1. The permeability and density gradients are probably the most poorly- documented quantities for superalloys. Values for the former quantities have recently been assessed using fluid-flow simulations employing realistic models for the mushy-zone topology obtained from three-dimensional (3D) reconstructions derived by serial sectioning.[5] The focus of these high performance computing (HPC) Challenge calculations is the development and validation of accurate models for $\Delta\rho$ by *ab initio* molecular dynamics (AIMD) simulations. Here, $\Delta\rho$ represents the mass-density difference between the hot liquid near the dendritic solid-liquid interface and the cooler melt at the top and side-walls of the casting. The mass-density difference originates from both the composition (c) and temperature (T) dependencies of the liquid-phase molar volume $V(c,T)$.

Recently, Mukai, et al.[6–8] developed a multi-component parameterization for $V(c,T)$ in superalloys based on extensive experimental measurements of the densities of binary liquid Ni alloys, and a few representative ternaries.[8] Intrinsic to this interaction scheme is the assumption that the partial molar volume of each solute species is independent of the compositions of the others. Therefore, effects arising from the interactions between different solute species atoms are ignored. Also, due to the limited availability of measured density data, several key elements, such as Re treated as parameters in fits to $V(c,T)$ data for multi-component superalloys. While Mukai's parameterizations are within a few percent of the measured values for the liquid-metal densities of several superalloys, we are unaware of experimental data for the composition and temperature-dependent densities in the multi-component superalloy RENE-N4. In the current work, we are using of state-of-the-art AIMD simulations as a means for testing the accuracy of the model for parameterizing $\rho(c,T)$ in this system. These calculations are also being used to independently test the accuracy of the Mukai parameters for elements where direct measurements of liquid density are unavailable.

2. Method

Ab-initio molecular dynamic simulations, based on density-functional theory (DFT), have been used to evaluate the properties of molten Ni-based alloys. Ionic positions were evolved in time using classical Nosé-Hoover dynamics[9,10], for both a fixed-number, volume, and temperature (NVT); and fixed-number, pressure and temperature (NPT) ensembles. The time evolution was based on inter-atomic forces (i.e., the Hellmann-Feynman forces) computed directly from DFT using the commercial DFT software VASP (Vienna Ab-Initio Simulation Package)[11–14], developed at the Institut für Materialphysik of the Univerität Wien. As we have shown previously, the VASP code scales well for systems with large numbers of atoms on parallel computers with high-bandwidth communications networks.[15] The underlying algorithms employed in VASP have been extensively tested and optimized for parallel processing with large numbers of processors.

Previously, we have used NVT dynamics to calculate the molar volumes of a variety of simple, binary and ternary metal alloys.[16] These include Ni, Al, Ni-X, and Ni-Al-X (X= W, Re, Ta) alloys at several compositions and temperatures. The DFT calculations made use of ultra-soft pseudo potentials[17,18] and the PW91 generalized-gradient approximation (GGA)[19] with a plane-wave basis cutoff energy of 260 eV. The simulations employ a single k-point (G), and time-steps (Δt) for the MD simulations were chosen as Δt=0.002 ps or Δt=0.003 ps, to ensure that the conserved energy in the Nosé-Hoover dynamics displayed a drift in time no larger than 1 meV/atom·ps. Analysis radial distribution functions suggest that the simulation system sizes are large enough to avoid the introducing of appreciable order in the liquid structures. Total simulation times ranged from 5–10ps, a time-interval long enough to obtain good statistical precision in calculated molar volumes and diffusion rates.[16]

In order to improve the efficiency of the calculations, the current version of VASP (release 4.5) was modified to integrate the molecular dynamics equations appropriate for fixed-number, pressure, and temperature ensemble using the ab initio computed forces and pressures. The implementation allows for Parrinello-Rahman dynamics using (deterministic) Nosé-Hoover chain thermostats for the atoms and the simulation cell vectors or Parrinello-Rahman dynamics using Langevin thermostats with stochastic forces and pressure tensors. The dynamics are constrained to eliminate center-of-mass motion and to impose irrotational deformations on the simulation cell, and a further option restricts deformations to volumetric (shape-preserving) fluctuations. Minimal changes were made to the VASP source code to include this new AIMD option and to accommodate the required input and output parameters.[16]

Current calculations are focused on known freckle-prone alloys, including model Ni-Al-W and the Ni-based superalloy RENE-N4. Assuming that other parameters for the Rayleigh criteria are constant across the mushy-zone, the signature for freckle formation is reduced to the condition when the liquid-metal at higher temperature has a greater density than at lower temperatures. Convective flow is then highly favored, as the high-temperature liquid at the top of the mushy zone displaces the lower temperature liquid at the bottom of the mushy-zone. It is precisely this instability that produces convective flow

during directional solidification. Density inversion is more complex than just a negative coefficient of thermal expansion, because the (equilibrium) composition of the liquid changes with temperature across the height of the mushy-zone. Also, there is some ambiguity about the relevant temperature-range sampled in the mushy-zone.

For the model Ni-Al-W alloy, we adopted a nominal composition of Ni-14Al-3W at. %, and evaluated the equilibrium liquid compositions and temperatures for 0 and 0.5 solid volume fraction using CalPhad (Pandat™) methods to estimate the Scheil solidification pathway.[17] The predicted liquid compositions at 1,711 and 1,720K are Ni82-Al15.6-W2.4 and Ni83-Al14-W3 at. %, respectively. For the 500 atom AIMD cells, this corresponds to $Ni_{410}Al_{78}W_{12}$ and $Ni_{415}Al_{70}W_{15}$ where the subscripts refer to the number of atoms of each element in the cell. For RENE-N4 we adopted the solidus and liquidus as the reference states. Using a similar Scheil model for RENE-N4, the solidus and liquidus temperatures were used to calculate the approximate liquid compositions at the bottom and top of the solidification zone.[18] The predicted compositions are Ni62.2-Cr11.0-Co5.2Mo1.3-W1.9-Al10.-Ti5.8-Ta2.3-Nb0.5 and Ni63.0-Cr11.2-Co7.6-Mo0.9-W1.9-Al9.2-Ti4.3-Ta1.6-Nb0.3 at 1,635K and 1,582K, respectively. These chemistries include very dilute Nb additions, which were removed from the calculations; the balance in composition being spread proportionally overall the remaining species. These compositions then yield the following number of atoms of each species in the 500-atom cell calculations: $Ni_{315}Cr_{56}Co_{38}Mo_5W_{10}Al_{46}Ti_{22}Ta_8$ and $Ni_{312}Cr_{55}Co_{26}Mo_6W_{10}Al_{50}Ti_{29}Ta_{12}$ for 1,635K and 1,582K, respectively. These chemistries were then used to construct 500 atom cells with random distributions of the appropriate elements. The different instantiations at each temperature were run in order to assess the influence of configurational entropy on the molar volume or alloy density.

The NPT density inversion calculations for model and Ni-based superalloys make use of PAW (Projector- Augmented Wave) and the PW91 generalized-gradient approximation (GGA).[19] This was found to produce a good representation of the liquid-metal superalloy.[16] Improvements in the electron-ion interactions require a larger cutoff energy for the plane-wave basis representing the electronic wave functions. However, the NPT dynamics allow derivation of the equilibrium volume in a single series of calculations, significantly reducing the required computational effort.

3. Results and Analysis

The main focus of these AIMD simulations is to demonstrate that such methods can reliably predict liquid metal densities to a level-of-accuracy that can inform models convective instabilities. Table 1 shows the calculated density at the 0 and 0.5 volume fraction temperatures for a nominal composition of Ni-14Al-3W at. %. This alloy was chosen because it has been demonstrated to produce freckle defects in conventional Bridgeman directional solidification. Here, the results are given as a density for the two solid-volume fractions. As shown in Table 1 the AIMD calculations predict a 2% increase in the density going from 1,711 to 1,720K, which suggests a relatively strong density inversion. Further, there is also a decrease in the molar volume with increasing temperature, from 7.782(5) to 7.767(8) cm3/mole, so density inversion is predicted even when the increase in elemental mass is not taken into account (i.e., the change in composition from the Scheil model favors an increase in density at the higher temperature). When applied to the model alloy, the liquid-metal Mukai parameterization produces a large density inversion at these temperatures and chemistries, see Table 1. The estimated molar volumes are 7.74(15) and 7.73(15) for 1,711 and 1,720K, respectively, which is within statistical error. So no density inversion would be expected based on just volumetric effects.

Table 1. Predicted liquid-metal densities for the Ni-Al-W model system. The numbers in parentheses represent the expected error on the last digit. For the AIMD results this is the estimated 95% confidence interval generated through standard propagation of error. For Mukai's parameterization, expected error is taken to be within 2.5%.[8]

		Composition	Density (g/cm³)	
V_f	T (K)	Atomic (/500)	AIMD	Mukai
0	1720	$Ni_{415}Al_{70}W_{15}$	7.46(2)	7.8(2)
0.5	1711	$Ni_{410}Al_{78}W_{12}$	7.29(1)	7.4(2)

The Ni-based superalloys have extremely complex alloy chemistries, and RENE-N4 is no exception. The first goal is to show that a relatively small population of atoms, in this case 500 atoms, can give a precise and reproducible density in molecular dynamics simulations running for less than ~10 ps. Table 2 shows the AIMD predicted densities for four different instantiations (at the solidus and liquidus temperatures) of a 500 atom simulation with runtimes less than 8 ps. For the eight cases considered, the AIMD densities are self-consistent and highly-reproducible. The results illustrate that even for very complex chemistries, liquid-metal densities can be accurately calculated using relatively small simulation cells.

Table 2. AIMD predicted densities (g/cm^3) for RENE-N4 at the Scheil model solidus and liquidus temperatures and compositions. Numbers in parentheses are the expected errors as defined in Table 1.

	Temperature	1,582K	1,625K
	Composition	$Ni_{315}Cr_{56}Co_{38}Mo_5$ $W_{10}Al_{46}Ti_{22}Ta_8$	$Ni_{312}Cr_{55}Co_{26}Mo_6$ $W_{10}Al_{50}T_{i29}Ta_{12}$
Density (g/cm^3)	AIMD-1	7.72(2)	7.51(2)
	AIMD-2	7.73(2)	7.51(2)
	AIMD-3	7.73(2)	7.51(2)
	AIMD-4	7.72(2)	7.51(2)
	Mukai[8]	7.3(2)	7.3(2)

The calculated densities for RENE-N4, shown in Table 2, do not predict a density inversion at these temperatures and compositions. Similar to the results for the model alloy, Mukai's parameterization produces numerically- equivalent densities for the liquidus and solidus states. Given the success in applying the 0–0.5 solid volume fraction reference states in the Ni-Al-W model system it is possible that the liquidus and solidus compositions of RENE-N4 are not representative of the relevant liquid-metal densities responsible for convective instability. Fortunately, the current results show that the new reference states can be accurately represented with several moderate-sized calculations. Calculations for 0.0–0.5 solid volume fractions in RENE-N4 are underway.

4. Summary

AIMD simulations have been performed to derive liquid-phase densities for model Ni-Al-W ternary alloy compositions designed to produce convective instabilities through a density inversion. Similarly, a freckle defect prone Ni-based superalloy, RENE-N4, was studied using four AIMD instantiations at the liquidus and solidus temperatures. The current AIMD calculations for the model ternary Ni-Al-W alloy predict a 2% density inversion, consistent with convective instabilities that produce freckle defects. For the RENE-N4 superalloy a series of moderately-sized simulation cells (i.e., 500 atoms) based on random distributions of the atomic species produced numerically-equivalent results. This suggests that even for highly-complex chemistries moderately sized AIMD simulations can produce reliable results. Finally, results for RENE-N4 indicate that reference chemistries and temperature based on 0.0–0.5 solid volume fraction may produce liquid-metal densities more relevant to convective instabilities.

Future work will involve more RENE-N4 AIMD calculations and the application of these methods to a larger set of Ni-based superalloys, particularly those that are known to form solidification defects. Also, in the coming year we will begin to apply methods for predicting the shear-viscosity in simple and binary alloys. The shear- viscosity is another parameter in the Rayleigh-number criteria (Equation 1) that is difficult to measure, and is not well-known for complex alloys. Finally, we will run alloy compositions for several other Ni-based superalloys where there are reliable, though limited, experimental measurements of liquid-metal densities.

Acknowledgements

This work was supported, in part, by a grant of computer time by the Department of Defense's High Performance Computing Modernization Program at the Aeronautical Systems Center, Major Shared Research Center on the SGI Altix-4700.

References

1. Sarazin, J.R. and A. Hellawell, *Met. Trans.*, 19A, 1861, 1988.

2. Pollock, T.M. and W.H. Murphy, *Met. Trans. A*, 27A, 1081, 1996.

3. Auburtin, P., S.L. Cockcroft, and A. Mitchell, *Superalloys 1996*, TMS, Warrendale, PA, pp. 443, 1996.

4. Beckermann, C., J.P. Gu, and W.J. Boettinger, *Met. Trans. A*, 31A, 2545, 2000.

5. Madison, J., J.E. Spowart, D.J. Rowenhorst, J. Fiefler, and T.M. Pollock, *Proceedings of the Eleventh International Symposium on Superalloys (The Minerals, Metals and Materials Society*, Warrendale, PA, pp. 881–888, 2008.

6. Mukai, K., L. Fang, Z. Li, and F. Xiao, *Mater. Trans.*, 45, 1754, 2004.

7. Mukai, K., Z.S. Li, and F. Liang, *Mater. Trans.*, 45, 2987, 2004.

8. Mukai, K., Z.S. Li, and K.C. Mills, *Met. Mater. Trans. B*, 36, 255, 2005.

9. Nosé, S., *J. Chem. Phys.*, 81, 511, 1984; *Mol. Phys.*, 52, 255, 1984.

10. Hoover, W.G., *Phys. Rev. A*, 31, 1695, 1985; 34, 2499, 1986.

11. Kresse, G. and J. Hafner, *Phys. Rev. B*, 47, 558, 1993.

12. Kresse, G. and J. Hafner, *Phys. Rev. B*, 49, 14251, 1994.

13. Kresse, G. and J. Furthmüller, *J. Computat. Mater. Sci.*, 6, 15, 1996.

14. Kresse, G. and J. Joubert, *Phys. Rev. B*, 59, 1758, 1999.

15. Asta, M., D.R. Trinkle, and C. Woodward, *Proceedings 2006 HPC-UGC*, IEEE Computing Society, Los Alamitos, CA, pp. 177–181, 2006.

16. Woodward, C., M. Asta, D.R. Trinkle, J. Lill, and, S. Angioletti-Uberti, *Jour. Applied Phys.*, Vol. 107, p. 113522, 2010.

17. Vanderbilt, D., *Phys. Rev. B*, 41, 7892, 1990.

18. Kresse, G. and J. Furthmüller, *Phys. Rev. B*, 54, 11169, 1996.

19. Perdew, J.P. and Y. Wang, *Phys. Rev. B*, 45, 13244, 1992.

Central Mode in Disordered $(Ba_{0.5}Sr_{0.5})TiO_3$ Solid Solutions from First Principles

S. Lisenkov and I. Ponomareva
Department of Physics, University of South Florida, Tampa, FL
{slisenk, iponomar}@usf.edu

L. Bellaiche
Department of Physics, University of Arkansas, Fayetteville, AR
laurent@uark.edu

Abstract

A first-principle-based approach is used to model static and dynamical properties of $(Ba_{0.5}Sr_{0.5})TiO_3$ disordered solid solution. Such a technologically-important system exhibits four different crystallographic states as it is cooled down. Interestingly, in addition to normal "soft" phonon modes whose existences can be derived from group theory, a novel dielectric peak occurring in the 10–30 cm^{-1} range is found in the simulations. This peak is the one denoted as the central mode in the literature, and is predicted to occur here over an unusually large temperature window (from 120 to 300K). Such prediction is of fundamental importance (by, e.g., revealing that all the different phase transitions that $(Ba_{0.5}Sr_{0.5})TiO_3$ undergo have an order-disorder character), but is also of technological relevance for the design of communication devices operating in the GHz–THz range.

1. Introduction

Ferroelectric (FE) and related materials are of high-importance for a variety of existing and potential device applications. Examples include piezoelectric transducers, actuators, non-volatile ferroelectric memories, and dielectrics for microelectronics and wireless communication. For some of these applications, the dynamical properties of these materials need to be determined and understood. For instance, some recent studies discovered that, just above the paraelectric-to-ferroelectric transition (Curie) temperature, $BaTiO_3$ has two (rather than one, as believed for more than 30 years) modes contributing to its dielectric response in the terahertz range[1,2]. The first peak occurs at higher frequency and is associated with the well-known soft phonon mode. The unexpected second peak is referred to as the central mode, is of the relaxation type, and is representative of a order-disorder character of the paraelectric-to-ferroelectric phase transition[2]. Note that, for an order-disorder phase transition, the paraelectric phase is non-polar "only" in a macroscopic sense since local electric dipoles—that average out to zero—exist in that state. On the other hand, in a displacive phase transition the paraelectric phase is both microscopically and macroscopically non-polar. Having different dielectric peaks can, of course, affect important properties of technological importance such as the dielectric tunability.

Furthermore, one has to realize that, for technological applications, disordered $(Ba_{1-x}Sr_x)TiO_3$ (BST) solid solutions are much more preferable than pure $BaTiO_3$. This is true, especially for compositions being near 50%, because the Curie temperature is just below room temperature (see Reference 3 and references therein). Interestingly, $(Ba_{0.5}Sr_{0.5})TiO_3$ is known to exhibit four phases as the temperature is cooled down: paraelectric cubic, FE tetragonal, FE orthorhombic, and FE rhombohedral (see Reference 3 and references therein). Based on the recent discovery of a central mode in $BaTiO_3$, it is legitimate to wonder if a central mode can also exist in $(Ba_{0.5}Sr_{0.5})TiO_3$ around the three transitions involving these four phases.

The main goal of this article is to answer this issue by using a state-of-the-art and newly-developed first-principles-based technique.

The article is organized as follows: Section 2 briefly describes the numerical method we use to predict properties of $(Ba_xSr_{1-x})TiO_3$ solid solutions, while Section 3 reports and discusses our results. Finally, Section 4 concludes the article.

2. Method

In the present work, we used an atomistic effective Hamiltonian scheme to mimic properties of $(Ba_xSr_{1-x})TiO_3$ materials[3]. The total energy of the system is given by:

$$E_{tot} = E_{self}\left(\{\mathbf{u}_i\}\right) + E_{dpl}\left(\{\mathbf{u}_i\}\right) + E_{short}\left(\{\mathbf{u}_i\}\right) + E_{elas}\left(\{\mathbf{v}_i\}\right), \eta_H + E_{\text{int}}\left(\{\mathbf{u}_i\},\{\mathbf{v}_i\},\eta_H\right) + E_{loc}\left(\{\mathbf{u}_i\},\{\mathbf{v}_i\},\{\sigma_i\},\{\eta_{loc,i}\}\right) + E_{elec}\left(\{\mathbf{u}_i\}\right)$$

where $\mathbf{u}_i$ is the local soft-mode at the site *i* of the supercell ($\mathbf{u}_i$ is directly proportional to the electrical dipole moment at site *i*) and $\mathbf{v}_i$ is a dimensionless vector related to the inhomogeneous strain around this site. η_H is the homogeneous strain tensor, and $\eta_{loc,i}$ characterizes the strain resulting from the difference in ionic size between Ba and Sr atoms[3]. $\{\sigma_i\}$ represents the alloy configuration, that is σ_i=+1 or −1 corresponds to the presence of a Ba or Sr atom, respectively, at site *i*. Here, the $\{\sigma_i\}$ are randomly distributed and kept frozen during the simulations to mimic a disordered solid solution. E_{tot} includes a local-mode self-energy, E_{self} (harmonic and unharmonic contributions); a long-range dipole-dipole interaction, E_{dpl}; a short-range interaction between local modes, E_{short}; an elastic energy, E_{elas}; an interaction between the local modes and strains, E_{int}; interactions related to alloying effects, E_{loc}; and an interaction of the local modes with an external electric field (if any), E_{elec}. This Hamiltonian correctly reproduces the complex sequence of phase transitions in BST alloys for a wide compositional range, and has been shown to yield predictions in good agreement with both experiment and first-principles calculations for various static and dynamical properties (see References 3, 4, 1, and 2).

In order to obtain dynamical properties for any temperature, the total energy of this effective Hamiltonian method is incorporated into a molecular dynamics (MD) technique[1]. Technically, we first run 10^5 MD-steps of NPT (isothermal-isobaric ensemble) simulations on a 16×16×16 supercell (20,480 atoms) to equilibrate the system at a chosen temperature and pressure. Then, the equilibration of the system within an NVE (micro-canonical) ensemble is performed through 10^5 MD-steps. Subsequent 4.3×10^6 NVE-steps are performed to obtain the time-resolved physical properties of the system.

To determine the complex dielectric response from MD simulations, we follow the approach of Reference 5; that is we compute the dielectric response, $\varepsilon(\nu)$, as:

$$\varepsilon(\nu) - 1 = \frac{1}{3\varepsilon_0 V k_B T}\left[< \mathbf{M}^2 > + i2\pi\nu \int_0^\infty dt e^{i2\pi\nu t} < \mathbf{M}(t)\cdot\mathbf{M}(0) > \right] \quad (1)$$

where "⟨⟩" denotes thermal averages, while *V*, *t*, and ν are the volume of the chosen supercell, the time and the frequency, respectively. $< \mathbf{M}2 >$ is the time-averaged square of the total dipole moment of the supercell and $< \mathbf{M}(t) \cdot \mathbf{M}(0) >$ is the dipole moment autocorrelation function[5,6]. Each peak appearing in the dielectric spectra is fitted by a classical harmonic oscillator[7,8]:

$$\varepsilon(\nu) = \frac{S}{\nu_r^2 - \nu^2 + i\nu\gamma} \quad (2)$$

where ν_r, γ, and *S* are the resonant frequency, damping constant and strength, respectively, of the oscillator.

3. Results and Discussion

Lets us consider a $(Ba_xSr_{1-x})TiO_3$ disordered solid solution with a Ba and Sr compositions of 50% each. Figure 1 shows the largest, middle and smallest cartesian coordinates (u_1, u_2, and u_3) of the supercell average of the local mode vectors in disordered $(Ba_{0.5}Sr_{0.5})TiO_3$ as a function of the temperature, as predicted by the effective Hamiltonian approach. This figure indicates a transition from cubic (C) paraelectric phase (for which u_1=u_2=u_3=0) to tetragonal (T) ferroelectric phase (for which u1is non-zero while u_2=u_3=0) at 235K ± 5K when cooling the system. Two other phase transitions occur when further cooling down the temperature: a transition from the FE tetragonal to FE orthorhombic (O) phase (for which u_2 now increases and becomes equal to u_1) at 180K ± 5K, and then a transition from the FE orthorhombic to a FE rhombohedral (R) phase (where all the coordinates of the supercell average of the local mode vectors are nonzero and equal to each other) at 155K ± 5K[3]. Such predictions agree rather well with the experimental values of 220–250K, 180K, and 140K, respectively[3].

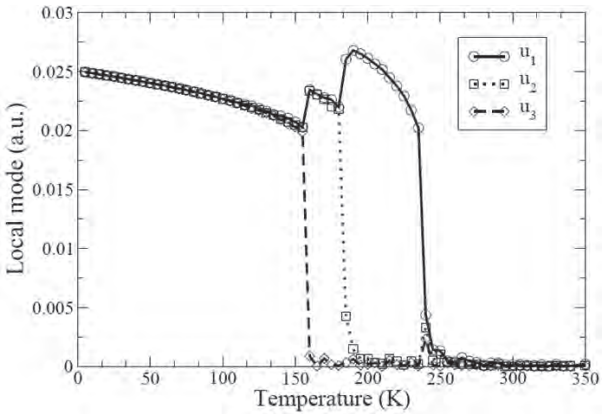


Figure 1. Largest, middle, and smallest average Cartesian coordinates, u_1, u_2, and u_3 of the local-mode vector as a function of temperature in the disordered $(Ba_{0.5}Sr_{0.5})TiO_3$ solid solution

Figures 2 (a and b) show the real and imaginary parts of the complex dielectric function, respectively, of $(Ba_{0.5}Sr_{0.5})TiO_3$ solid solution in the C phase at T=280K, as obtained from our MD simulations. Note that 280K is slightly above the paraelectric-to-ferroelectric phase transition. Our simulations clearly show the presence of two peaks in the dielectric spectra. The first one at ~75 cm^{-1} originates from the soft-mode (SM), while the second peak (at v_r ~30 cm^{-1}) corresponds to the central-mode (CM). Our results are in good agreement with infrared reflectivity measurements for BST systems that support the appearance of the CM for temperatures close to the paraelectric-to-ferroelectric phase transitions[9].

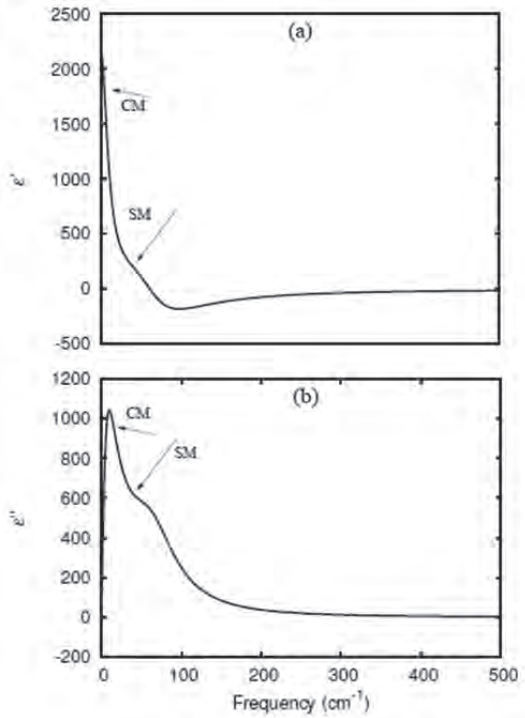


Figure 2. Real (a) and imaginary (b) part of the complex dielectric function of $(Ba_{0.5}Sr_{0.5})TiO_3$ as a function of frequency for T=280K. Arrows show the location of peaks originating from the soft-mode (SM) and the centra- mode (CM).

Figure 3 shows the dependence of the resonant frequencies v_r of all the dielectric peaks predicted by our MD simulations for the whole temperature range. One can see that, for temperatures higher than 300K, only one peak is observed in the dielectric response. This peak corresponds to the SM, implying that the CM response completely disappears for the temperatures larger by more than 65K from the Curie transition temperature. One can also notice that the resonant frequency of the SM decreases when the temperature is decreased down to 300K, as consistent with the term "soft" associated with the name of the corresponding phonon.

In the ferroelectric phases, the simulated dielectric response is anisotropic and dielectric tensor components along different crystallographic directions show very distinct spectra. For instance, in the tetragonal T phase (green symbols in Figure 3), the soft-mode splits into E and A_1 modes that correspond to dipolar oscillations perpendicular and parallel to the polarization, respectively. One striking feature displayed by Figure 3 is that the central-mode is present in the entire temperature region defining the FE tetragonal phase. Within this T state, the frequency of the A_1 mode strongly increases when the temperature is decreasing, which is consistent with the fact that the polarization gains in magnitude during this cooling process (see Figure 1). On the other hand, the resonant frequencies of the E and CM modes are slightly decreasing when decreasing the temperature in the T phase, as consistent with the fact that another phase transition towards the O state is going to occur at lower temperature. When crossing the tetragonal-orthorhombic boundary, the E and A_1 modes evolve into B_2, B_1, and another A_1 modes, while the resonant frequency of the CM (that also survives in the entire O phase region) is basically unaffected by such FE-to-FE transition. Finally, the transition from the orthorhombic-to-rhombohedral state generates E and A_1 modes, for which the resonant frequencies considerably increase as the temperature is decreased down to 0K. It is interesting to realize that the central-mode is still present in the R phase for temperature above 120K, and then disappears for lower temperature.

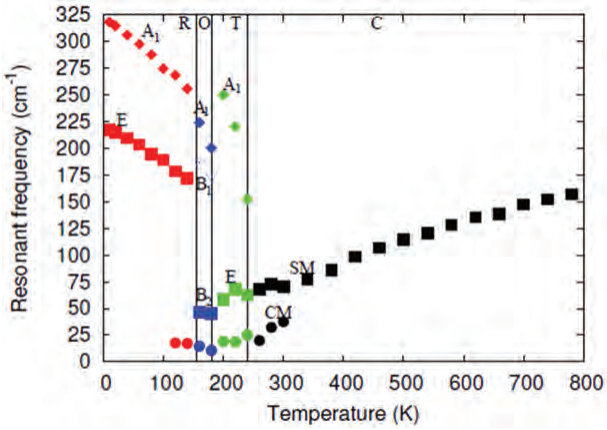


Figure 3. Temperature dependence of the resonant frequency of all the dielectric peaks predicted by the MD simulations for any temperature in the $(Ba_{0.5}Sr_{0.5})TiO_3$ solid solution. Paraelectric phase is labeled as *C*, ferroelectric tetragonal, orthorhombic, and rhombohedral phases are labeled as *T*, *O*, and *R*, respectively. Squares, circles, rhombus, and triangle symbols represent the SM, CM, A_1, B_1, E, and B_2 modes.

4. Conclusion

In conclusion, using HPCMP resources with Challenge Project C4F, we have simulated static and dynamical properties of $(Ba_{0.5}Sr_{0.5})TiO_3$ disordered solid solutions using a first-principle-based approach. The most striking discovery offered by our simulations is that the CM can exist not only over a large temperature window, namely from 120K to 300K, but also for all the different possible symmetries (i.e., C, T, O, and R). This strongly suggests that each of the three phase transitions that BST undergo as the temperature is varied exhibits an order-disorder character. We hope that our predictions will be soon experimentally-confirmed.

Acknowledgments

This work is supported by the Office of Naval Research Grants N00014-08-1-0915 and N00014-07-1-0825 (DURIP).

References

1. Ponomareva, I., Bellaiche L., Ostapchuk, T., Hlinka, J., and Petzelt, J., Terahertz dielectric response of cubic $BaTiO_3$", *Phys. Rev. B*, 77, 012102, 2008.

2. Hlinka J., Ostapchuk, T., Nuzhnyy, D., Petzelt, J., Kuzel, P., Kadlec, C., Vanek, P., Ponomareva, I., and Bellaiche, L., "Coexistence of the Phonon and Relaxation Soft-Modes in the Terahertz Dielectric Response of Tetragonal $BaTiO_3$", *Phys. Rev. Lett.*, 101, 167402, 2008.

3. Walizer L., Lisenkov, S., and Bellaiche, L., "Finite-temperature properties of $(Ba,Sr)TiO_3$ systems from atomistic simulations", *Phys. Rev. B*, 73, 144105, 2006.

4. Lisenkov, S. and Bellaiche, L., "Phase diagrams of $BaTiO_3/SrTiO_3$ super-lattices from first-principles", *Phys. Rev. B*, 76, 020102(R), 2007.

5. Caillol, J.M., Levesque, D., and Weis, J.J., "Theoretical calculation of ionic solution properties", *J. Chem. Phys.*, 85, 6645, 1986.

6. Rapaport ,D., *The Art of Molecular Dynamics Simulation*, Cambridge University Press, Cambridge, 2001.

7. Buixaderas, E., Kamba S., and Petzelt J., "Lattice Dynamics and Central-Mode Phenomena in the Dielectric Response of Ferroelectrics and Related Materials", *Ferroelectrics*, 308, 131, 2004.

8. Gervais, F., *Infrared and Millimeter Waves*, edited by K. Button, Academic, New York, Vol. 8, pp. 279339, 1983.

9. Ostapchuk, T., Petzelt, J., Hlinka, J., Bovtun, V., Kuzel, P., Ponomareva, I., Lisenkov, S., Bellaiche, L., Tkach ,A., and Vilarinho, P., "Broad-band dielectric spectroscopy and ferroelectric soft-mode response in the $Ba_{0.6}Sr_{0.4}TiO_3$ solid solution", *J. Phys. Condens. Matter*, 21, 474215, 2009.

Designing Tough Carbon Nanotube Fibers

Charles F. Cornwell and Charles R. Welch
US Army Engineer Research and Development Center (ERDC), Vicksburg, MS
{charles.f.cornwell, charles.r.welch}@usace.army.mil

Abstract

The superior mechanical properties of carbon nanotubes (CNTs) such as low-density, high-strength, and stiffness make them attractive for many structural applications. While the strength and stiffness of CNTs are extremely high, to-date, fibers of aligned CNTs have been found to be far weaker than the constituent CNTs. The intermolecular interactions between the CNTs in the fibers is governed by weak van der Waals forces resulting in slippage between CNTs that occurs at tensions well below the breaking strength of the CNTs. Both theoretical and experimental studies show that the strength and stiffness of parallel, aligned CNT fibers are increased by introducing chemical bonds, such as interstitial carbon-atom bonds, between the CNTs. Such bonds significantly increase load transfer between the CNTs and prevent them from slipping. Molecular dynamics simulations of the tensile response of such cross-linked CNT fibers reveal a sharp transition from ductile to brittle behavior, with increasing cross-link concentrations of the CNT fibers.

1. Introduction

The discovery of carbon nanotubes (CNTs) is credited to several groups and individuals. The most influential papers identifying the basic structure of CNTs are arguably the paper by Iijima (1991) and papers by Iijima and Ichihashi (1993) and by Bethune, et al. (1993). Monthioux and Kuznetsov (2006) provide a detailed discussion of the discoverers of CNTs.

The extreme characteristics of CNTs have been the subject of intense theoretical and technological interest in materials science since their discovery by Iijima (1991). Both theoretical and experimental studies have shown that the elastic modulus of a CNT is in the range of 1–2 TPa (Krishnan, et al., 1998). Treacy, et al. (1996) reported one of the first measured values for Young's modulus of isolated CNTs by measuring the amplitude of their intrinsic thermal vibrations using transmission electron microscopy, and found the average value of Young's modulus to be 1.8 TPa. Krishnan, et al. (1998) refined the technique used by Treacy, et al. (1996) and found the average value of Young's modulus to be 1.25 TPa. Haskins, et al. (2007) used Tight-Binding Molecular Dynamics simulations to examine the effects of defects on CNTs, and deduced values of Young's modulus and tensile strengths for (5,5) chiral CNTs of 0.95 to 1.15 TPa and 70 to 110 GPa, respectively, for the molecular defects considered.

While the strength and stiffness of CNTs can be extremely high, fibers composed of aligned CNTs are far weaker than the constituent CNTs (Qian, et al., 2003; Ericson, et al., 2004; Koziol, et al., 2007; Zhang, et al., 2008). Studies show that the bundles fail because the relatively weak van der Waals forces between molecules allow CNTs to slip past one another before the shear forces between molecules reach the intrinsic breaking strength of the CNTs (Ericson, et al., 2003; Walters, et al., 1999; Yu, et al., 2000a, 2000b).

This paper describes the use of chemical bonds as a mechanism for coupling the superior mechanical properties of the individual CNTs across length scales. The simulations were carried out using Sandia Laboratory's Large-scale Atomic/Molecular Massively-Parallel Simulator (LAMMPS) code (Plimpton, 1995). The potential energy of the system was calculated using the Adaptive Intermolecular Reactive Empirical Bond Order (AIREBO) Potential (Stuart, et al., 2000). Tersoff-type potentials such as AIREBO are reactive, meaning they allow bonds to form and break during the course of a simulation. They have been shown to be qualitatively good at modeling the mechanical properties of carbon-based materials (Yakobson, et al., 1996, 1997; Cornwell and Wille, 1997; Garg, et al., 1998). The computational cost of reactive potentials is relatively high compared with that of nonreactive potentials. Simulating large ensembles of CNTs containing millions of atoms requires high performance computers (HPC) and computer algorithms that scale efficiently on thousands of processors. LAMMPS is specifically designed to run on parallel computer systems, and has good scaling characteristics on a wide-range of HPC platforms (LAMMPS Molecular Dynamics Simulator, 2011). The calculations were performed

on the SGI Altix ICE (diamond) at the Department of Defense (DoD) Supercomputing Resource Center (DSRC) at the US Army Engineer Research and Development Center (ERDC), Vicksburg, MS.

This paper is a follow-up to an earlier paper designed to study the effects of bond density and CNT length distribution on fiber strength and stiffness (Cornwell and Welch, 2011). The results showed an increase in the strength and elastic modulus with increased CNT length and cross-link concentration. The results also revealed a transition from ductile to brittle behavior going from fibers constructed from the shorter CNTs and low cross-link concentrations to fibers constructed from longer CNTs and higher cross-link concentrations. This paper describes a more detailed study of the transition from ductile to brittle behavior as the cross-link concentrations are increased. The geometry, boundary and initial conditions, time step, and potential functions for these simulations are identical to the simulations performed in an earlier paper (Cornwell and Welch, 2011). These are described below for completeness.

2. Bundle Construction

Experimental results indicate that CNTs within a bundle have similar radii and are randomly distributed; (Thess, et al., 1996); i.e., there is no correlation between the length of a CNT and the axial distribution of the tubes, and the tubes likely have random azimuthal orientations. In our molcular dynamic simulations, the digital representations of the fibers were constructed using a random distribution of CNT lengths, with each CNT given a random rotation about its longitudinal axis between 0 and 2π. The longitudinal axes of the fibers were initially aligned parallel to the z-axis. The strands consisted of parallel (5,5) CNTs placed end-to-end, arranged parallel to one another with a gap of 3.33Å between CNTs in the strands (Figure 1). This gap corresponds to the equilibrium distance between the capped ends of the CNTs as determined by the potential function. Each bundle had 19 strands arranged in an HCP configuration (Figure 2).

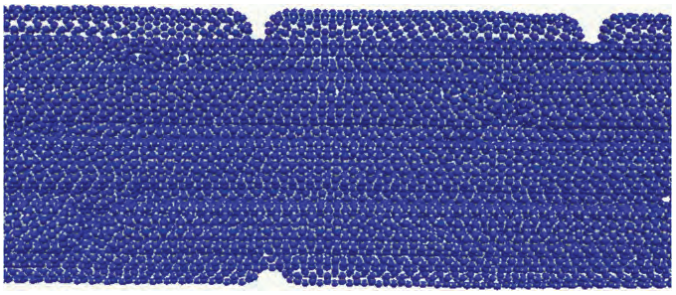


Figure 1. Side-view of a section of a CNT bundle with CNTs placed end-to-end over the length of the bundle

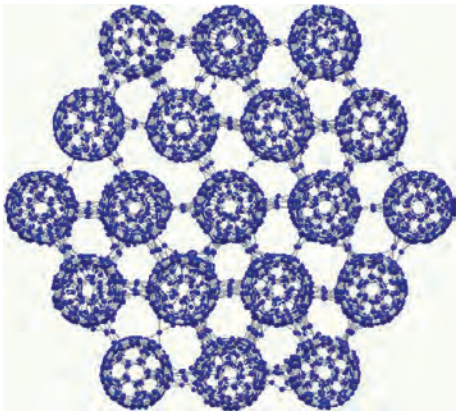


Figure 2. Cross section of bundle with cross-link atoms forming chemical bonds between the CNTs

The positions of the cross-link atoms were randomly selected from the volume of the bundle. A check was made to determine if the cross-link atom formed the correct number of nearest-neighbor bonds, and that it formed bonds between CNTs from two different strands. If it did not meet these criteria, it was rejected. The concentration of cross-link atoms is defined as the number of cross-link atoms divided by the number of atoms in the bundle expressed as a percentage.

The simulations had periodic boundary conditions in the three Cartesian dimensions. The system size in the x-y plane was large enough to prevent any interaction across the boundary for atoms in the bundle. A Berendsen thermostat was

applied to all the atoms to minimize the heat conduction problem pointed out in Berendsen, et al. (1984) and Mylvaganam and Zhang (2004). The simulations were run at a temperature of 300K.

The stress tensor was calculated using:

$$< S_{ij} >= \frac{\sum_{k}^{N} m_k v_{k_i} v_{k_j}}{V} + \frac{\sum_{k}^{N} r_{k_i} f_{k_j}}{V} \quad (1)$$

The first term of Equation 1 is the kinetic energy tensor, and the second term is the virial stress tensor. N is the number of atoms in the system, and the Cartesian coordinates are designated by i and j=x,y,z. The variables m_k, v_k, r_k, and f_k are the mass, velocity, position, and force, respectively, for atom k, and V is the volume of the bundle. Here, the volume of the bundle is calculated using the length of the bundle times its cross-section. The area of a regular hexagon that encloses the strands of the bundle is used to define the cross-sectional area of the fiber. Figure 2 shows the cross-section of a fiber with the cross-link atoms forming chemical bonds between the CNTs. The cross-section of the bundle is considered constant throughout the simulation. This assumption causes the stresses being reported to be the "engineering stress" as opposed to "true stress." True stress takes into account the change in cross-section as the fiber is elongated.

3. Simulation

Strain was induced in the fibers by increasing the length of the bundle in small increments. The length of the bundle was increased in steps by scaling the positions of the atoms by an amount that varied along the longitudinal axis of the fiber, depending on the coordinate of the atoms. Figure 3 shows the time evolution of the stress and strain for two strain increments. After each strain increment, the stress was then allowed to approach equilibrium. At time t=0, the strain was increased at a rate of 2.0 E8 (s^{-1}) for 10 picoseconds (ps), increasing the strain to 0.002 (fraction). The strain was then held constant to allow the stress in the fiber to equilibrate until t=150 ps, at which time the strain was increased at a rate of 2.0 E8 (s^{-1}) for 20 ps, increasing the strain by another 0.004, and again the strain was held constant for a period after each increase in strain to allow the stress to equilibrate.

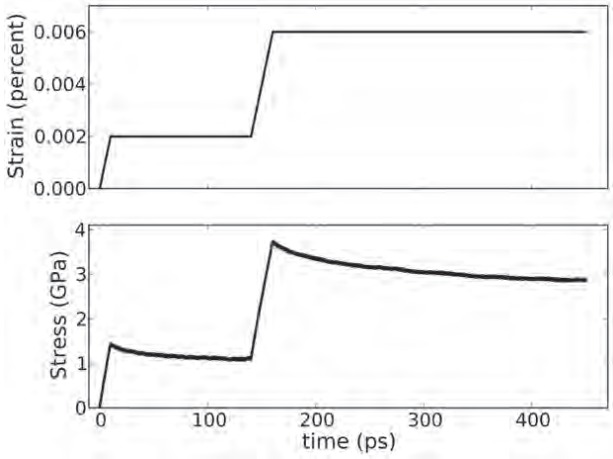


Figure 3. Time evolution for the stress and strain for one of the bundle runs. The strain is increased in increments of 0.004, and then allowed to equilibrate before the strain was increased. The stress and strain are recorded at the point at which the stress reaches equilibration.

The stress tensor was calculated by averaging Equation 1 over 20 ps. If the difference in the stress between two consecutive stress calculations was below a given threshold, the stress was considered at equilibrium and the process continued. This provided a feedback loop that effectively resulted in a variable strain rate, based on simulation conditions, that gave the bundle time to respond to bond breaking, defect formation and migration, slipping or failure of the CNTs, and the resulting structural changes that took place in the bundle from these processes.

Some variations were seen in the mechanical properties of the fibers because the fibers were constructed using CNTs with random orientations, random distribution of lengths, and a random distribution of cross-link atoms. Figure 4 shows the stress-strain curves for eight statistically-equivalent fibers. The fibers were 4,000 Å long and were constructed using

CNTs with a Gaussian distribution in length, with an average length of 1,000 Å, and a standard deviation in the tube length of 200 Å. All fibers had a cross-link concentration of about 0.225 percent.

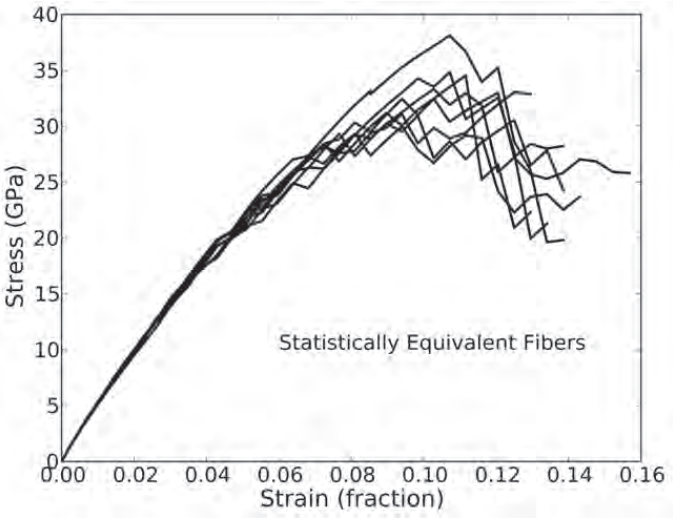


Figure 4. Stress-strain curve for eight statistically-equilvalent fibers. A different seed was used for the random number generator used to construct the CNTs with an average length of 1,000 Å and a standard deviation of 200 Å. A different seed was used to select the position of the cross-link atoms.

The elastic modulus was calculated over the linear portion of the stress-strain curves, hence is an initial or tangent modulus. The point of maximum stress was used to determine the tensile strength of the fibers. While there are some variations in the stress-strain curves in Figure 4, the simulations produce consistent and reproducible results for statistically-equivalent fibers. The average maximum stress for the eight simulations is 33.467 GPa, with a standard deviation of 2.495 GPa. This represents a standard deviation of 7.46 percent of the average maximum stress. The eight simulations had an average initial elastic modulus of 586.434 GPa with a standard deviation of 14.743 GPa, representing a standard deviation of 2.51 percent of the average initial modulus.

4. Results

A series of simulations were run to investigate the effect of CNT cross-link concentrations on the tensile response of parallel-aligned CNT fibers. Results of the stress-strain calculations are presented in Figure 5 in which the tensile response was predicted for fibers constructed from CNTs whose initial lengths was 4,000 Å, and with cross-link concentrations that ranged from 0.113 to 0.451 percent. The lowest response curve corresponds to the fiber with the lowest cross-link densities, and the curves increase in amplitude as the cross-link concentrations increase. The minimum elastic modulus and tensile strength were 582.83 GPa and 22.88 GPa, respectively, calculated for the bundle with a percentage of cross-links of 0.113. The maximum elastic modulus and tensile strength were 656.50 GPa and 53.58 GPa, respectively, calculated for the bundle with a cross-link density of 0.451.

The results of the stress-strain calculations presented in Figure 5 also reveal a sharp transition from ductile to brittle failure going from bundles with low to higher cross-link concentrations. All of the bundles with cross-link concentrations less than or equal to 0.226 percent showed ductile behavior. Bundles with cross-link concentrations greater than 0.226 percent showed brittle behavior. A detailed understanding of the bundle post-yield response is required to optimize the strength and elastic modulus of fibers, and to prevent brittle failure.

The results in Figure 6 show the time evolution of the stress and strain for two 4,000 Å fibers. The average length and standard deviation of the CNTs in both fibers are 1,000 Å and 200 Å, respectively. The cross-link densities in the two fibers differ, and were 0.226 and 0.281. The plot of the stress and strain data in Figure 6 is the time average of the stress and strain at 10-ps intervals.

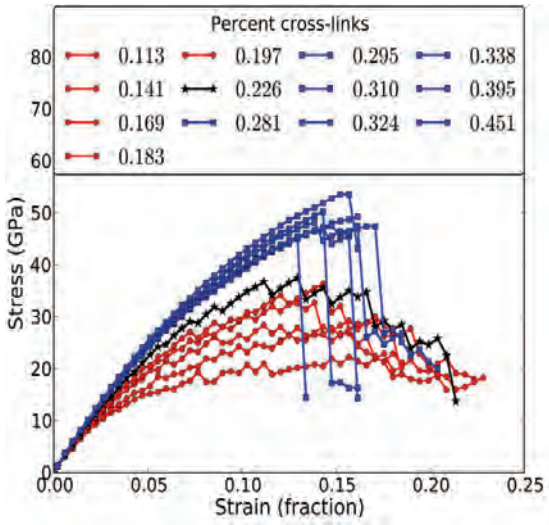


Figure 5. Stress-strain curves for a bundle 4,000 Å long, with the percentage of cross-links ranging from 0.113 to 0.451

In Figure 6, the bundle with a cross-link concentration of 0.281 percent reached a maximum stress of 47.49 GPa, at a strain of 0.143 in 2.20 ns. The fiber reaches a maximum stress of 44.08 GPa at 2.53 ns. At 2.67 ns and a strain of 0.162, the bundle began to fail. At that point, the average strain held constant, and the stress decreased to 14.30 GPa before reaching equilibrium at 5.46 ns. Recall that displacement boundary conditions were used in the simulations. While the average strain in the bundle was holding constant, inside the bundle, bonds were failing and the average internal stress was decreasing.

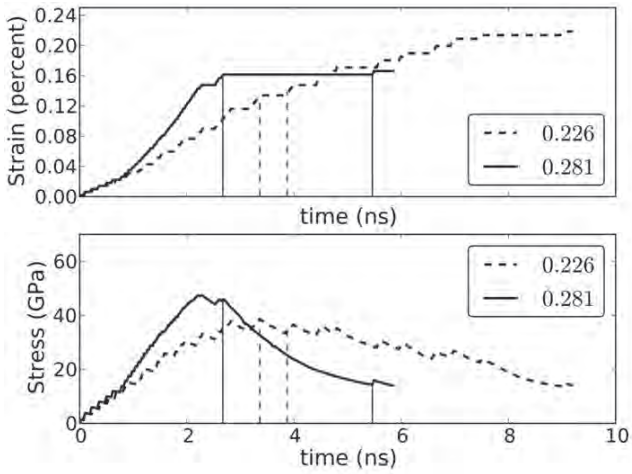


Figure 6. Time evolution of the stress and strain for a 4,000-Å bundle. The average length of the CNTs in the bundle is 1,000 Å, with a standard deviation of 200 Å. The initial percentages of cross-links in the fibers are 0.226 and 0.281.

Figure 7A shows the damaged section of the bundle at 2.67 ns prior to failure. Figure 7B shows the damaged section of the bundle once the stress stabilized at 5.46 ns . Considerable damage was done to the bundle before the stress stabilized. In a macroscopic analogy, this abrupt change in stress results in brittle failure.

By contrast, the response of the fiber with cross-link density of 0.226 in Figure 6 reached a maximum stress of 38.57 GPa, at a strain of 0.134 in 3.36 ns. At that point, the bundle began to fail, and the strain was held constant for 0.51 ns before the bundle stress stabilized at a stress of 34.67 GPa. Figures 8A and 8B show the damaged section of the bundle at the start of failure and after the stress stabilizes. It can be seen that some necking took place before the fiber reached its maximum stress (Figure 8A). The stress continued to decrease with increasing strain, but the bundle did not break, and was able to support stress up to a strain of 0.218, where the simulation was terminated. In a macroscopic analogy, the gradual decrease in stress with increased strain is equivalent to ductile behavior.

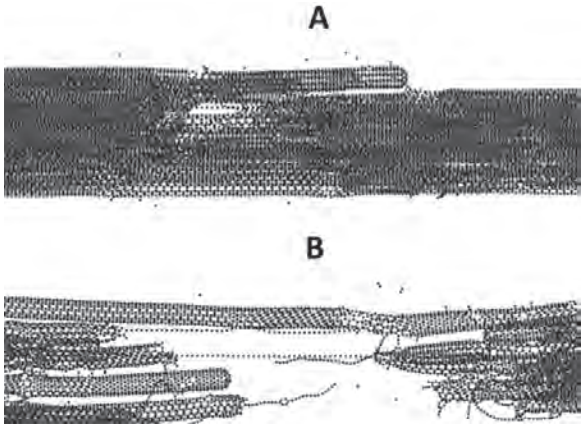


Figure 7. Bundle with 0.281 percent cross-link atoms. Fibers with higher cross-link concentrations tend to achieve higher maximum stress, but produce substantial damage to the bundle during failure. As a result, the failure tends to be brittle.

The simulations indicate that failure in all fibers begins when bond breaking, defect formation, and slipping or failure of the CNTs occur in the fibers. This results in a release of stress and causes structural changes in the fiber. If the resulting structure of the fiber is able to support the residual stress, the stress stabilizes. Otherwise, the fiber will fail again, and the process is repeated until the stress stabilizes or the fiber breaks.

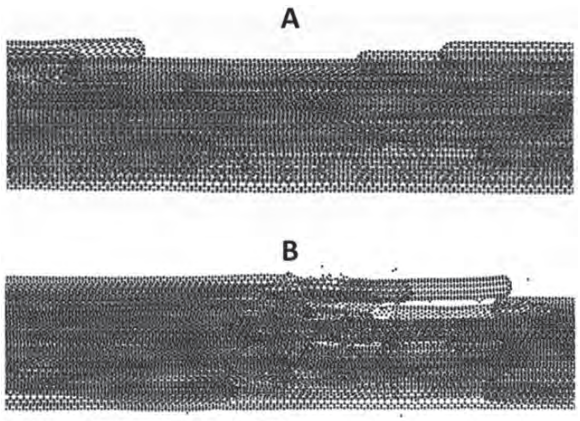


Figure 8. Bundle with 0.226 percent cross-link atoms. Fibers with lower cross-link concentrations tend to achieve lower maximum stress, but produce less damage to the bundle during failure. As a result, the failure tends to be more ductile.

Generally, the fibers with lower concentrations of cross-links fail at the cross-links. This tends to produce less damage to the bundle, and allows the bundle structure to stabilize to the point at which it can support the residual stress. On the other hand, the CNTs tend to fail in fibers with higher concentrations of cross-links. CNT failures release considerable energy, damaging the bundle and making it less-likely that the resulting structure will be able to support the residual stress in the fiber.

These analyses considered fibers composed only of one chirality of CNTs, that is, (5,5) chiral CNTs. An earlier analyses presented in Haskins, et al. (2007) used Tight Binding Molecular Dynamics simulations, and considered the effect that defects and chirality have on individual CNT tensile response. The authors found that (10,5) chiral CNTs exhibited ductile-type behavior with the carbon-carbon bonds failing spirally around the CNT, while (5,5) chiral CNTs displayed brittle behavior with carbon-carbon bond failures occurring within a plane perpendicular to the CNT axis. It may be that choice of CNT chirality can also be used to affect ductile or brittle behavior of cross-linked CNT fibers, and it is likely that defects within the CNTs can also affect cross-linked fiber behavior.

5. Conclusion

The molecular dynamics simulations presented considered fibers composed of chiral (5,5) CNTs of varied lengths and interstitial carbon atom cross-link densities. The simulations show that careful control of the cross-link density and distribution will produce either fibers with high tensile strength and high modulus that exhibit brittle behavior, or fibers with lower strength and lower modulus that exhibit ductile behavior. The simulations reveal a sharp transition from ductile to brittle behavior with all fibers having cross-link concentrations less than or equal to 0.226 percent showing ductile behavior, while fibers with cross-link concentrations greater than 0.226 percent showing brittle behavior.

Based on these analyses and the analyses presented in Haskins, et al. (2007), we speculate that choice of CNT chirality can also be used to affect the brittle or ductile response of cross-linked CNT fibers, and that defects within CNTs will affect cross-linked CNT fiber response.

Acknowledgments

This study gratefully acknowledges funding support from the US Army Engineer Research and Development Center Directed Research Program "Nanoscale Studies of Polycrystalline Materials with Emphasis on Ceramic Synthesis", and the HPC Challenge Project "Molecular Dynamics Simulations To Underpin the Design and Development of High-Performance Carbon-Nanotube-Based Filaments, Membranes, and Coatings" and allocation of computer time from the Department of Defense High Performance Computing Modernization Program. Permission was granted by the Chief of Engineers to publish this information.

References

Berendsen, H.J.C., J.P.M. Postma, W.F. van Gunsteren, A. DiNola, and J.R. Haak, "Molecular dynamics with coupling to an external bath", J*ournal of Chemical Physics*, 81, pp. 3684–3690, 1984.

Bethune, D., C. Kiang, M.D. Vries, G. Gorman, R. Savoy, J. Vazquez, and R. Beyers, "Cobalt-catalysed growth of carbon nanotubes with single-atomic-layer walls", *Nature*, 363, pp. 605–607, 1993.

Cornwell C.F., and L.T. Wille, "Elastic properties of single-walled carbon nanotubes in compression", *Solid State Communications*, 101, pp. 555–558, 1997.

Cornwell, C.F., and C.R. Welch, "Very-high-strength (60-GPa) carbon nanotube fiber design based on molecular dynamics simulations", *Journal of Chemical Physics*, 134, pp. 204798–204708, 2011.

Ericson, L.M., H. Fan, H. Peng, V.A. Davis, W. Zhou, J. Sulpizio, Y. Wang, R. Booker, J. Vavro, C. Guthy, A.N.G. Parra-Vasquez, M.J. Kim, S. Ramesh, R.K. Saini, C. Kittrel, G. Lavin, H. Schmidt, W.W. Adams, W.E. Billps, M. Pasquali, W.F. Hwang, R.H. Hauge, J.E. Fischer, R.E. Smalley, "Macroscopic, neat, single-walled carbon nanotube fibers", *Science*, 305, pp. 1447–1450, 2004.

Garg, A., J. Han, and S.B. Sinnott, "Interactions of carbon-nanotubule proximal probe tips with diamond and graphene", *Physical Review Letters*, 81, pp. 2260–2263, 1998.

Haskins, R. W., R.S. Maier, R.M. Ebeling, C.P. Marsh, D.L. Majure, A.J. Bednar, C.R. Welch, B.C. Barker, and D.T. Wu, "Tight-binding molecular dynamics study of the role of defects on carbon nanotube moduli and failure", *Journal of Chemical Physics*, 127, pp. 074708–074718, 2007.

Iijima, S., "Helical microtubules of graphitic carbon", *Nature*, 354, pp. 56–58, 1991.

Iijima, S. and T. Ichihashi, "Single-shell carbon nanotubes of 1-nm diameter", *Nature* 363, pp. 603–605, 1993.

Koziol, K., J. Vilatela, A. Moisala, M. Motta, P. Cunniff, M. Sennett, and A. Windle, "High-performance carbon nanotube fiber", *Science*, 318, pp. 1892–1895, 2007.

Krishnan, A., E. Dujardin, T.W. Ebbesen, P.N. Yianilos, and M.M.J. Treacy, "Young's modulus of single-walled nanotubes", *Physical Review B*, 58, pp. 14013–14019, 1998.

LAMMPS Molecular Dynamics Simulator, http://lammps.sandia.gov, Sandia National Laboratories, Albuquerque, NM (accessed January 25, 2011).

Monthioux, M. and V L. Kuznetsov, "Who should be given the credit for the discovery of carbon nanotubes?" *Carbon*, 44, pp. 1621–1623, 2006.

Mylvaganam, M. and L.C. Zhang, "Important issues in a molecular dynamics simulation for characterizing the mechanical properties of carbon nanotubes", *Carbon*, 42, pp. 2025–2032, 2004.

Plimpton, S.J., "Fast Parallel Algorithms for Short-Range Molecular Dynamics", *Journal of Computational Physics*, 117, pp. 1–42, 1995.

Qian, D., W.K. Liu, and R.S. Ruoff, “Load transfer mechanism in carbon nanotube ropes”, *Composites Science and Technology*, 63, pp. 1561–1569, 2003.

Stuart, J.S., A.B. Tutein, and J.A. Harrison, “A reactive potential for hydrocarbons with intermolecular interactions”, *Journal of Chemical Physics*, 112, pp. 6472–6486, 2000.

Thess, A., R. Lee, P. Nikolaev, H. Dai, P. Petit, J. Roert, C. Xu, Y.H. Lee, S.G. Kim, A.G. Rinzler, D.T. Colbert, G.E. Scuseria, D. Tomanek, J.E. Fischer, and R.E. Smalley, “Crystalline Ropes of Metallic Carbon Nanotubes”, *Science*, 273, pp. 483–487, 1996.

Treacy, M.M.J., T.W. Ebbesen, and J.M. Gibson, “Exceptionally-high Young’s modulus observed for individual carbon nanotubes”, *Nature*, 381, pp. 678–680, 1996.

Walters, D.A., L.M. Ericson, M.J. Casavant, J. Liu, D.T. Colbert, K.A. Smith, and R.E. Smalley, “Elastic strain of freely- suspended single-wall carbon nanotube ropes”, *Applied Physics Letters*, 74, pp. 3803–3805, 1999.

Yakobson, B.I., C.J. Brabec, and J. Bernholc, “Nanomechanics of carbon tubes: Instabilities beyond the linear response”, *Physical Review Letters*,. 76, pp. 2511–2514, 1996.

Yakobson, B.I., M.P. Campbell, C.J. Brabec, and J. Bernholc, “Tensile strength, atomistics of fracture, and c-chain unraveling in carbon nanotubes”, *Computational Materials Science*, 8, pp. 341–348, 1997.

Yu, M.-F., B.I. Yakobson, and R.S. Ruoff, “Controlled sliding and pullout of nested shells in individual multi-walled carbon nanotubes”, *Journal of Physical Chemistry B*, 104, pp. 8764–8767, 2000a.

Yu, M.-F., O. Lourie, M.J. Dyer, K. Moloni, T.F. Kelly, and R.S. Ruoff, “Strength and Breaking Mechanism of Multi-walled Carbon Nanotubes Under Tensile Load”, *Science*, 287, pp. 637–640, 2000b.

Zhang, S., L. Zhu, M.L. Minus, H.G. Chae, S. Jagannathan, C.-P. Wong, J. Kowalik, L.B. Roberson, and S. Kumar, “Solid-state spun fibers and yarns from 1-mm long carbon nanotube forests synthesized by water-assisted chemical vapor deposition”, *Journal of Materials Science*, 43, pp. 4356–4362, 2008.

Properties of High-Performance Capacitor Materials and Nanoscale Electronic Devices

J. Bernholc, V. Ranjan, C. Han, W. Lu, and M. Buongiorno Nardelli
Center for High Performance Simulation and Department of Physics, North Carolina State University, Raleigh, NC
{bernholc, chan3, mbnardelli}@ncsu.edu and luw@chips.ncsu.edu

Abstract

Recent advances in theoretical methods, combined with the advent of massively-parallel supercomputers allow one to reliably simulate the properties of complex materials and device structures from first-principles. We describe applications in two general areas: i) novel polymer composites for ultrahigh-density capacitors, necessary for pulsed-power applications, such as electric rail guns, power conditioning, and dense electronic circuitry; and ii) electronic properties of graphene, an emerging ultrahigh-speed electronic material. For capacitors, polyvinylidene fluoride (PVDF) with a small concentration of chlorotrifluoroethylene (CTFE) has been observed to store very-high energy, as compared to currently-used polymers. We have recently suggested that the ultra-high energy storage is due to an electric-field-induced phase transition from the non-polar α to the polar β phase. We are now investigating the kinetics of this transformation, focusing on the atomistic mechanism and energetic barriers for the transition. Turning to nano-electronic materials, we simulate the scanning tunneling microscope (STM) images of graphene nano-flakes and graphene on a hydrogenated Si substrate. The results show that the observed STM patterns are strongly influenced by the character of the edges, resulting in different super-structures depending on edge symmetries.

1. Introduction

In many energy storage applications, there is an urgent need for a high-power density material with both a large dielectric constant and a large breakdown field, so that the necessarily large amount of energy can be stored and repeatedly discharged without failure. Such materials do not exist currently, and have to be developed by combining current knowledge, exploratory experimental work, large-scale calculations, concepts derived from various microscopic and macroscopic models, and manufacturability considerations. For current pulsed high-power applications, two materials systems are being explored: i) very-high dielectric constant oxides that are; however, brittle and relatively heavy; and ii) polymers, which are light, flexible and can withstand high-electric-fields, but their dielectric constants are usually not large. A composite material, consisting of oxide particles embedded in a polymeric matrix, is also being considered, because it would avoid the brittleness of the bulk oxides, while still taking advantage of their large dielectric constants.

We describe our current progress in further developing a novel class of polyvinylidene fluoride (PVDF)-based ferroelectric polymers for high-power density applications. While pure PVDF has a relatively-low energy density, an admixture of a small amount of chlorotrifluoroethylene (CTFE) dramatically increases the achievable density.[1] We have shown[2] that this highly-non-linear increase in the energy density is due to the formation of disordered nano-domains with different copolymer concentrations, which undergo first-order non-polar to polar phase transitions with an increase of the applied field. However, in order to understand the nature of the transition and to optimize the material, one needs to map out the atomic transformations and the energy barriers. Previously, we have determined the initial stages of this process; namely the transformation from the non-polar α phase to the polar γ phase. In this work, we describe the significantly more complex transition from the γ phase to the highly-polar β phase.

Graphene is a highly-promising two-dimensional (2D) material for future electronic devices. It has exceptionally high in-plane mobility, which, however, is significantly affected by interactions with nearby layers and the substrate. Defects and adsorbates also have a major effect on its electronic properties. Graphene, being a planar material, is an ideal candidate for scanning tunneling microscopy (STM) imaging, which provides unparalleled insight into electronic properties near the Fermi level. For graphene, the most interesting effects in STM images are edge-state localization and electron interference

patterns near edges and defects. The STM observations can resolve the scattering properties of graphene edges, and help in understanding of the electronic configuration of the sample.

Recently, Lyding and co-workers used an *in situ* technique[3] to deposit graphene flakes onto highly-doped Si(100)2x1-H substrate and measure the STM tip-current without electrodes attached to the flake. The STM observations unraveled the properties of intrinsic edge-states of mono-layer graphene in the absence of graphitic stacking.

The most widely accepted STM theory of interference patterns on graphene was originally derived for small defects in the graphene plane, where the scattered and normal wave functions interfere around the defect.[4] There are two types of interference processes near the Fermi level, namely inter- and intra-valley scattering. Interference between two wave-functions from different K points gives a $\sqrt{3}\times\sqrt{3}R30^{\circ}$ pattern and its inverse—the honeycomb pattern. On the other hand, interference between wave functions of the same Dirac cone results in a long wave-length modulation of the charge density. Many experimental and theoretical studies have shown evidence that various defects introduce similar scattering processes. Among many of the observations, the edge-introduced interference pattern is of interest in this work: a straight armchair edge on a graphite surface is recognized as the source for the honeycomb super-structure, while a zigzag edge on the same substrate does not contribute additional super-structure to the graphite STM image. A slight mixture of zigzag and armchair edges leads to coexistence of the $\sqrt{3}\times\sqrt{3}R30^{\circ}$ and the honeycomb STM patterns.[5,6]

It turns out that vertices and polygonal shapes have dramatic effects on the STM patterns of graphene nano-flakes, due to wave-function interference in a relatively-small sample. In this communication, we report the results of *ab initio* investigations of the properties of electronic states close to nano-flake vertices and their effects on STM images. Our simulation results agree with the STM observations and also point out the effects of substrate on STM patterns, especially when scanning alters the substrate-graphene distance. This work is part of a larger study aimed at understanding and categorizing the patterns expected in unperturbed graphene samples, to help in characterization of prototype graphene structures and devices, and in distinguishing defect structures with electrically-active levels that would affect device performance.

2. Methods and Calculations

The band structure calculations are performed using PWscf[7] implementation of density-functional theory and ultra-soft pseudo-potentials. Depending on the system, the cutoffs are up to 35 Rydbergs and up to 32 k-points are used to ensure a high-level of convergence. All atoms are fully-relaxed. For polymer calculations, the Born effective charges Z_i^* for each of the ions *i* are obtained using density-functional perturbation theory.[8] Following Fu and Bellaiche[9] the term $-eZ_i^*E$ is added to the Hellmann-Feynman forces $\mathbf{f}_i$, where *e* is the elementary charge. The atomic structure is relaxed until the total force on each atom is close to zero, i.e., until $\mathbf{f}_i = -eZ_i^*E$.[9–11]

The graphene calculations are carried out using a massively-parallel real-space multi-grid implementation[12,13] of density-functional theory (DFT), employing the generalized gradient approximation of PBE form and ultra-soft pseudo-potentials. The nano-flake super-cells include vacuum regions of up to 12 Å in all directions to ensure negligible interactions between the periodic images. The grid-spacing is 0.17 Å, which is equivalent to an energy cut-off of 45 Ry in plane-wave representation. All the calculations are performed with Γ point Brillouin zone sampling. The super-cells include up to 1,200 carbon atoms.

STM patterns are calculated with Tersoff's formalism for constant-height STM images. For a small positive bias, the tunneling-current density $j(r)$ is proportional to local density of states (LDOS) at the tip position. Using the expression derived by Tersoff and Hamann,[14] the tunneling-current density $j(r)$ can be written as:

$$j(r,V) \propto \int_{E_f - eV}^{E_f} dE\rho(r,E)$$

where

$$\rho(r,E) \equiv \sum_{n,k} \left|\psi_{nk}(r)\right|^2 \delta(E_{nk} - E)$$

Here, $\rho(r, E)$ is the LDOS at the tip position r=(x, y, z) and $\psi_{nk}(r)$ is the eigen-state of the unperturbed surface with the corresponding energy $E_{n,k}$. Therefore, the simulated STM images crucially depend on the surface charge density of graphene. The STM image is obtained by integrating electron-density in the bias window at constant height above the graphene surface.

In order to enable direct comparison with published experimental results, we also simulate, for the first time, the derivative STM images using:

$$\tilde{j}(r,V) = \frac{dj(r,V)}{d\hat{n}}$$

where $\hat{n}$ indicates the STM slow-scan direction. Our results show that the computed space-resolved current is calculated with enough precision to obtain derivative images in good agreement with experimental data.

When calculating STM images for nano-flakes, we noted that changes of bond-lengths at the flake edges do not alter the simulated patterns. When studying large nano-flakes containing up to 1,200 atoms, we thus set all bond-lengths to 1.43 Å, the calculated bond-length in the middle of the flake. (The calculated bond-length for a graphene sheet is 1.42 Å.)

3. Results and Discussion

3.1 High Energy Density Polymers

Poly-Vinylidene Fluoride (PVDF) and its copolymers exist in two main phases, the non-polar α phase and the polar β phase. The existence of these phases provides an avenue for manipulating their electrical and electro-mechanical properties, and we have previously shown that the high-energy density in P(VDF-CTFE) is due to an electric-field-induced transformation between the non-polar α and the polar β phases. An important point is that while pure PVDF prefers the α phase at zero-field, the polar β phase becomes the preferred phase at high co-polymer concentration even at zero-field.[2] The transformation from the non-polar to polar phase is facilitated by the copolymer, which breaks the structure into nano-domains, which can separately switch to the polar phase as the field is increased. While this scenario provides an appropriate thermodynamic description, the governing issue is, of course, the kinetics, because the barriers for the transformation must be low enough to enable fast phase-switching at room temperature without significant hysteretic effects.

The structure of a single PVDF polymer chain is characterized by the ordering of the dihedral angles in the individual monomers: when all the dihedrals are equal (180°) the chain is in an all-trans conformation (TT) and all the monomers possess a permanent electric dipole of ~2 Debye. When the dihedral angles alternate between -57° and +57°, the chain is in a trans-gauche conformation (TGTG'). When the individual chains organize in a crystalline environment, they can form four different polymorphs, depending on the individual conformation of the chains and their relative orientation: i) anti-polar TGTG' (α-PVDF), ii) polar TGTG' (γ-PVDF), iii) anti-polar TT (δ-PVDF), and iv) polar TT (β-PVDF).

Table 1. The results of the first-principles calculations for structural parameters, total energy difference from the α-PVDF structure, and Berry-phase-based polarization values

Type	A (Bohr)	b (Bohr)	c (Bohr)	Volume (Bohr^3)	Total energy (meV)	Polarization (C/m^2)
α	18.27	8.81	9.56	1532.24	0	0
γ	20.52	8.81	9.37	1702.08	408	0.09
β	16.16	9.75	9.07	1433.82	181	0.19

In the past, we have extensively investigated the phase diagram of PVDF and its copolymers as a function of the applied electric-field and CTFE concentration.[2] In particular, we have shown that ordered phases of P(VDF-CTFE) with well-defined concentrations exhibit sharp phase transformations between non-polar (α) and polar (β) phases. However, the existence of these transformations cannot explain the strongly-increasing displacement-field and the large variation of the dielectric constant that are observed in experiments, and eventually lead to high-energy density. Moreover, a direct transformation from α to β-PVDF, although thermodynamically feasible, would still require un-physically-large electric-fields. Only by assuming a disordered multi-domain structure, the distribution of concentrations will lead to a range of polar transitions that would result in high-energy density. This assumption, albeit successful in interpreting the data, falls short of providing a complete microscopic explanation of the mechanism of high-energy density storage in PVDF and its co-polymers.

In the bulk phases, the different polymorphs of PVDF are related to each other by non-trivial symmetry operations: the rotational manifold of α-PVDF, where one of the polymer chains is rotated by 180°, yields γ-PVDF; and β-PVDF is a torsional manifold of γ-PVDF, so that a clear topological path is available to the system to transform from one phase to the other. As a consequence, PVDF may provide an adaptable response to electric-fields produced by a conformational selection

of an appropriate functional structure from a large ensemble of topological states. However, the necessary condition for this to happen is that the manifold of torsions and inter-chain rotations must be thermodynamically- and kinetically-accessible at relatively-low temperatures. The existence of a low- barrier transition path would allow for a non-linear dielectric response; and, consequently could explain the high- energy density observed in these materials. Confirming this involves identification of a transition path and calculation of the energy barrier between the end-phases.

In the previous year, we have investigated the α→γ transformation pathway, which could serve as a topologically-accessible intermediate phase for the transformation to the β phase. In this case, one PVDF chain rotates about its axis to change inter-chain arrangement from anti-polar to polar. The activation energy of only ~50 meV is sufficiently small for this transformation to proceed at room temperature. However, the transition from the γ to β phase is significantly more complicated, and involves changing the TGTG' configuration to an all-trans (TT). We generated intermediate structures by changing alternatively dihedral angles in steps of ±Δθ (Δθ=15°) from their values in the γ configuration (±60°) to the final β geometry (180°). The intermediate structures thus created are relaxed under the constraint that the dihedral angles remain fixed to their assigned values. As we can see in Figure 1, the energy barrier is still a relatively-low 100 meV per carbon atom, enabling the transformation at room temperature. The next steps in this investigation are to examine the influence of copolymers on the transformation mechanism and activation barrier. The ability of tuning the barrier by suitable admixtures would add much-needed flexibility for optimizing the power-density and release-speed for high-speed power applications.

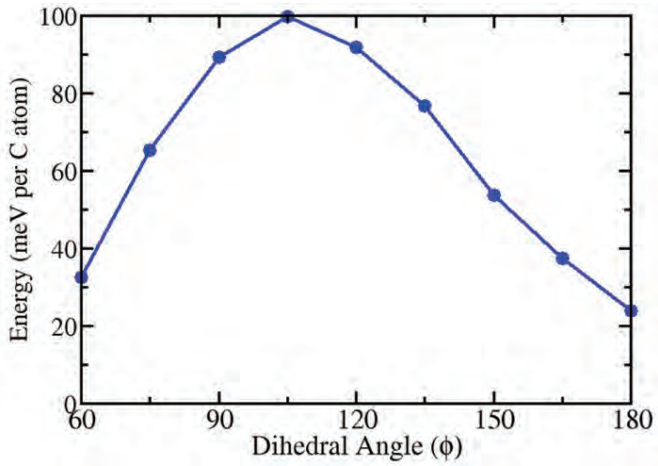


Figure 1. Energetics of the γ to β phase transformation for pure PVDF as a function of the dihedral angle

3.2 Simulation of STM Images of Graphene Nano-flakes

In this section, we discuss the STM patterns of graphene nano-flakes that emerge from DFT simulations, and compare them to experimental results. In Figure 2, we show STM and derivative-STM images of two kinds of flakes, both terminated with zigzag edges. The normal STM images, shown in the left panel, are of two hexagonal graphene samples, an elongated hexagon and an equilateral hexagon. The corresponding current derivative images, with the slow-scan direction being horizontal, are shown in the right panel.

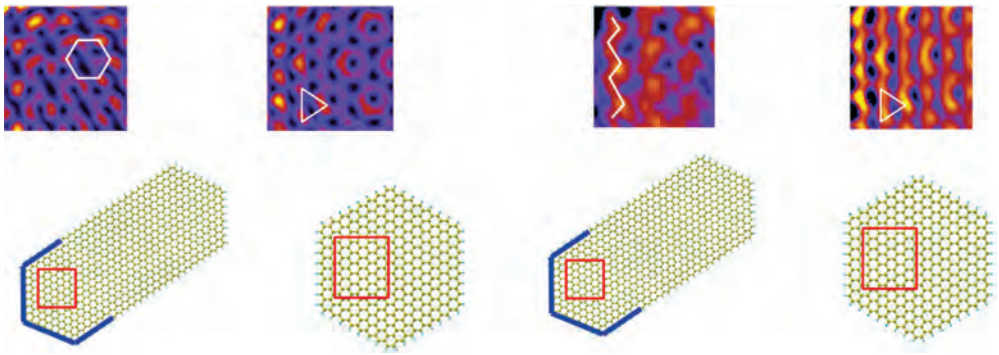


Figure 2. Simulated-STM (left panel) and derivative-STM (right panel) images of elongated and equilateral graphene flakes. The derivative image of the elongated flake is in good agreement with the corresponding experimental observations.

In each case, the quasi-particle wave-function scattering at each edge is critical to the observed STM image. In the equilateral hexagonal sample, the scattering of the wave-functions from the close-packed six vertices result in a (1×1) pattern. In the elongated sample with three 120° vertices near each other, a hexagonal superstructure appears, which is similar to that observed in experiment on a similarly-shaped sample.[3] These experimental and computational observations suggest that the zigzag edge is not only an intra-valley scatterer[15], but with specific vertex angles it can introduce inter-valley scattering.

We now consider the effects of substrate on the STM image of graphene, using Si(100)2x1-H monohydride substrate as a paradigm. We use a super-cell with 38.6×27.2×32.2 Å dimensions. To preserve the periodicity of the graphene lattice, the substrate bonds are slightly stretched, by 2.7% and 5.9% at the two super-cell boundaries. Γ point sampling is used, which for this size super-cell corresponds to a 12×12 k-point sampling in a graphene unit cell. We consider two heights above the substrate, ~3 Å, which is the computed equilibrium distance, and 2.5 Å, which occurs when graphene is depressed slightly by the STM tip.[16] In Figure 3, we compare STM images generated for these two cases with an image of a free-standing graphene. The first two images, for free-standing graphene and graphene at 3 Å above the substrate, are essentially identical in our voltage window of (-1 eV, 1 eV). However, when graphene is pushed toward the Si(100)2x1-H by 0.5 Å, in a voltage window of (-0.3eV, 0), a honeycomb super-structure is visible in the STM image. This super-structure suggests a substrate-induced inter-valley scattering. In fact, the honeycomb pattern has a translational symmetry around 3.67 Å, which is almost the same as that of the dimers of the Si(100)2x1-H substrate. The observation of a substrate-induced pattern and semi-transparency of the graphene sheet is similar to that observed experimentally for slightly depressed graphene on GaAs and InAs surfaces.[16]

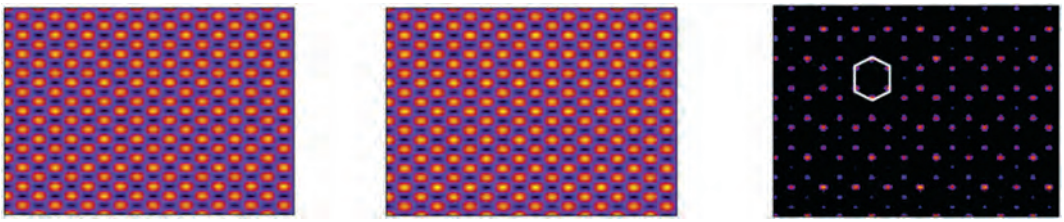


Figure 3. Simulated STM images of (a) free-standing graphene, (b) graphene on Si(100)2x1-H at the equilibrium distance, and (c) graphene on Si(100)2x1-H depressed by 0.5 Å

References

1. Chu, B., X. Zhou, K. Ren, B. Neese, M. Lin, Q. Wang, F. Bauer, and Q.M. Zhang, "A Dielectric Polymer with High-Electric-Energy-Density and Fast Discharge-Speed", *Science*, 313, 334, 2006.

2. Ranjan, V., L. Yu, M. Buongiorno Nardelli, and J. Bernholc, "Phase Equilibrium in High-Energy-Density PVDF-Based Polymers", *Phys. Rev. Lett.*, 99, 047801, 2007.

3. Ritter, K.A. and J.W. Lyding, "The influence of edge structure on the electronic properties of graphene quantum dots and nano-ribbons", *Nat. Mater.*, 8, pp. 235–242, 2009.

4. Mizes, H.A. and J.S. Foster, "Long-Range Electronic Perturbations Caused by Defects Using Scanning Tunneling Microscopy", *Science*, 244, pp. 559–562, 1989.

5. Niimi, Y., et al., "Scanning tunneling microscopy and spectroscopy of the electronic local density-of-states of graphite surfaces near mono-atomic-step edges", *Phys. Rev. B*, 73, 085421, 2006.

6. Kobayashi, Y., K.-ichi Fukui, and T. Enoki, "Edge-state on hydrogen-terminated graphite edges investigated by scanning tunneling microscopy", *Phys. Rev. B*, 73, 2006.

7. Giannozzi. P. et al., http://www.quantum-espresso.org.

8. Baroni, S., S. de Gironcoli, A. Dal Corso, and P. Giannozzi, *Rev. Mod. Phys.*, 73, 515, 2001.

9. Fu, H. and L. Bellaiche, *Phys. Rev. Lett.*, 91, 057601, 2003.

10. Sai, N., K.M. Rabe, and D. Vanderbilt, *Phys. Rev. B*, 66, 104108, 2002.

11. Antons, A., J.B. Neaton, K.M. Rabe, and D. Vanderbilt, *Phys. Rev. B*, 71, 024102, 2005.

12. Briggs, E.L., D.J. Sullivan, and J. Bernholc, *Phys. Rev. B*, 54, 14362, 1996.

13. Hodak, M., S. Wang, W. Lu, and J. Bernholc, *Phys. Rev. B*, 76, 085108, 2007.

14. Tersoff, J. and D.R. Hamann, "Theory of the scanning tunneling microscope", *Phys. Rev. B*, 31, 805, 1985.

15. Sakai, K.-ichi, K. Takai, K.-ichi Fukui, T. Nakanishi, and T. Enoki, “Honeycomb super-periodic pattern and its fine-structure near the armchair edge of graphene observed by low-temperature scanning tunneling microscopy”, *Phys. Rev. B*, 81, 235417, 2010.

16. He, K.T., J.C. Koepke, S. Barraza-Lopez, and J.W. Lyding, “Separation-Dependent Electronic Transparency of Mono-layer Graphene Membranes on III−V Semiconductor Substrates”, *Nano Letters*, 10, pp. 3446–3452, 2010.

HPCMP UGC 2011

3. Computational Chemistry and Materials Science (CCM)

Computational Biology

A Bioinformatics Platform for Rapid Detection of Bacterial and Viral Sample Components through NextGen Sequencing Technologies

Shijie Yao, Greg Donarum, and Alvin Liem
OptiMetrics, Inc., Abingdon, MD
{syao, gdonarum, aliem}@omi.com

David L. Hirschberg, Komal Jain, and Craig Street
The Center for Infection and Immunity, Columbia University, New York, NY
{david.hirschberg, kj2230, cs2673}@columbia.edu

Tom Slezak
Lawrence Livermore National Laboratory, Livermore, LA
slezak@llnl.gov

Henry S. Gibbons and C. Nicole Rosenzweig
US Army Edgewood Chemical Biological Center (ECBC), Edgewood, MD
{henry.gibbons, nicole.rosenzweig}@us.army.mil

Abstract

In today's next-generation sequencing environment, high-throughput sequencing technologies have allowed scientists to gather unprecedented amounts of data at astonishing rates. Making sense of these large volumes of data pose new and significant challenges to both computational and system biologists. The development of applied mathematical and computational tools leading to dynamic bioinformatics methods and pipelines can open the door for multiple applications of this critical, actionable knowledge. Scientific research ranging from host biomarker discovery, pathogen identification and characterization, and bio-surveillance depend on bioinformatics platforms to elucidate discovery and provide a robust pipeline of chemical, biological, radiological, and nuclear (CBRN) capabilities and products. This paper describes an ongoing Department of Defense (DoD) project to design and implement a bioinformatics platform for the rapid identification of bacterial and viral sample components through NextGen sequencing data analysis. The overall architecture design and detailed analytical capabilities of the platform are described by presenting its application on selected data sets. The analytical process was developed by integrating both in-house and open-source analytical tools into one single system available on the DoD network.

1. Introduction

The advent of new DNA sequencing technologies such as Roche 454 pyro-sequencing (Margulies, 2005), ABI SOLid (Shendure, 2005), Illumina (Bentley, 2008), or next-generation (NetGen) sequencing, has made the act of DNA sequencing a high-throughput operation. By applying NetGen sequencing technologies, analysis on environmental and clinical samples can quickly generate DNA sequence data that number in the order of millions of reads. This abundant data can quickly provide the identity of the organisms in question while describing population structures in the source samples. These advanced sequencing technologies, when coupled with well-designed post-sequencing analytical capabilities, can readily provide actionable information for threat characterization, and have the ability to segue into an adaptive response capability to include applications such as medical diagnostics, antibiotic resistance determination, bio-surveillance, attribution, host-agent dynamic signatures, and therapeutic target capabilities.

In 2006, the Department of Defense (DoD)/Defense Threat Reduction Agency's (DTRA) initiated the Transformational Medical Technologies Initiative (now TMT) to fully-exploit the advanced science and technology innovation necessary to successfully counter advanced bio-terror threats, including genetically-engineered, intracellular bacterial pathogens and hemorrhagic fevers. Edgewood Chemical Biological Center (ECBC) has supported the TMT mission since FY09, serving as both the primary DoD lab for bacterial sequencing as well as the bioinformatics development team.

In support of the TMT effort, we have constructed a bioinformatics platform to provide the analytical capability of analyzing DNA sequencing data and identifying the bacterial and viral sample components existing therein. The system is designed to include client-server architecture, providing a web-based graphical user interface (GUI) front-end for sequence

data submission with sample tracking capabilities. The act of submitting a sequence initiates a quality assurance pipeline; including data integrity check and database update, in addition to uploading the sequence and associated files. The system is designed so that the act of data submission immediately initiates the back-end analytical pipeline performing user-defined and default analysis on the data.

2. Methods

The bioinformatics platform was designed to integrate in-house and open-source analytical tools into one single analytical system. The design allows users to log onto the system remotely, upload sequence data and other supporting data to the platform's repository subsystem, and starts an analytical pipeline analysis session on the uploaded sequence data. The platform is composed of a web application providing user login and sequence/metadata uploading capability, a set of stand-alone analytical pipelines to analyze genomic data files, a DNA sequence repository database for the DoD DNA sequences and metadata, and a number of Basic Local Alignment Search Tool (BLAST) databases facilitating the sequence analysis. As the DNA sequence data files tends to have large sizes, the sequence repository database is designed to have a well-defined disk-based file structure for the raw DNA sequence data storage, and store only minimal information, include references to the disk files, in a relational database.

The web application is built with a Liferay portal server running on top of an Apache Tomcat 6.0 application server. In-house portal java components are developed to fulfill the platform's front-end user interface functionalities. The back-end analytical modules and utilities are written in PERL and java, with open-source programs (e.g., Apache MySQL, NCBI [National Center for Biotechnology Information] blast+) and third-party applications (e.g., GS FLX Newbler) wrapped with PERL code. Each functional module is wrapped within individual PERL code, and can be independently executed with well-defined input/output conventions. The analytical capability of the bioinformatics platform is summarized in Figure 1. It composes three major logical components: 1) optional DNA sequence preprocessing processes; 2) an iterative nearest-neighbor identification process; and 3) genomic sequence threat assessment process.

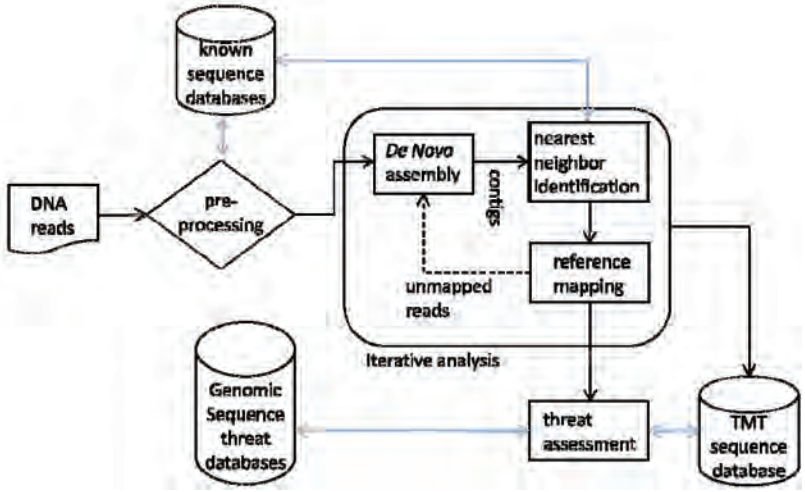


Figure 1. Summary of the analytical capability of the bioinformatics platform

2.1 Sequence Quality Adjustment Pre-Processing

We expect several sequencing platforms to be supported by this analytical capability by the end of CY11. However, the primary sequencers used for detection of unusual or novel organisms are the Roche 454 Titanium whole genome sequencing instrument. By default, all sequencing data input is summarized into a table with read-length and quality statistics. Should read-quality be a concern, all the reads can be reduced according to user-defined cutoffs for sequence-length and quality score. The raw sequences with length shorter than the cut-off length are masked and not utilized in down-stream analysis. The quality scores of the longer sequences are examined, and end-bases with quality scores lower than the cut-off value is trimmed from the original sequence. If the after-trim length is shorter than the length of the cut-off value, the sequence is also masked. A pair of fasta and qual files is produced containing the high-quality sequences for *de novo* assembly and additional analysis.

2.2 Host Sequence Identification Pre-Processing

A pre-processing procedure, developed by Columbia's Center for Infection and Immunity, is provided to mask low-value reads derived from host in complex samples. In this procedure, all the reads from the sample are compared (using NCBI blast program) against pre-prepared common virus host databases. The host databases include: *Anopheles gambiae* (mosquito), *Danio rerio* (zebra fish), *Gallus gallus* (chicken), *Homo sapiens* (human), *Homo sapiens rRNA* (human), *Homo sapiens chromosome* (human), *Mus_musculus* (rodent), *Sus scrofa* (pig), mitochondrion genome, and *Xenopus laevis* (frog). The sequences that match well to any of the host sequences are identified, masked, and not utilized in down-stream analysis.

2.3 Sequence Clustering Pre-Processing for Complex Samples

For complex genomic samples, the platform provides a generic taxonomy-based analysis to cluster bacterial and viral reads, as well as reads with no near-neighbors in the reference database for de novo assembly. The inclusion of reads with no near-neighbors improves the probability of identifying a novel or engineered organism.

All raw-reads are compared with NCBI nt database sequences using the NCBI blast program. Each read is assigned to a NCBI taxonomy ID according to the so-called lowest common ancestor (LCA) algorithm described by the MEGAN (Huson, 2007) program. The user can define the criteria (minimum-bits score, minimum-identity ratio, and top-percent of hits to use in LCA) to optimize analysis when required. The reads assigned to the bacterial and viral sub-tree of the NCBI taxonomy, together with reads lacking a near-neighbor assignment, are collected for downstream analysis.

2.4 Iterative Process to Identify Nearest-Neighbor from NCBI NT Database

This bioinformatics platform is designed to identify pathogens in complex samples. To fulfill this goal, an iterative process has been constructed to perform a subtractive approach in searching for multiple-pathogen or multiple-chromosomal elements in a single sample. In the first iteration, the genomic data uploaded to the system, or reads retained following the pre-processing manipulations, are assembled by the *de novo* assembler of GS Newbler program. The *de novo* assembly produces longer contiguous lengths (contigs) of genomic data. Then, the contigs produced by Newbler are compared with NCBI nt database to identify high-quality matches. All the contigs' assigned nearest-neighbors are collated and ranked, based on the number of times a near-neighbor ranked top-most. A second method of ranking is based on the number of occurrences a near-neighbor has with an E-value of 0.0. The top-most-ranked NCBI database record in the ranked list is selected as the nearest-neighbor for the iteration. Finally, all the reads used for *de novo* assembly are compared with the nearest-neighbor by means of reference mapping. Those reads that do not result in a pair-wise match to the nearest-neighbor sequence are collected and used as input for the next iteration. The iterative analysis allows multiple chromosomes, plasmids, or inserted genomic elements to be identified and reported to the user for directed, manual analysis.

2.5 TMT Sequence Archive

At the end of the iterative process, the assembled DNA sequence contigs are tagged with a unique sequence ID and archived in a sequences database. This sequence database can be compared against any new sequence data submission.

2.6 Genomic Sequence Threat Potential Assessment

Investigators at Lawrence Livermore National Laboratory (LLNL) have compiled a DNA sequence database for characterizing the threat potential of genomic sequences, placing particular focus on assembled draft genomics (Jonathan, 2008). ECBC has incorporated the LLNL sequence database into our bioinformatics platform by comparing our *de novo* contigs from the first iteration of analysis against this genomic database in search of high-quality matches.

2.7 Hardware and Queue System

The high-performance cluster hardware hosting this bioinformatics system is located at the US Air Force Research Laboratory (AFRL) in Dayton, OH. Support for the productions system is by RCM, access to the system provided through DTRA approval, development system supported by ECBC CIO. The HPC system is an Aspin Linux 64 bit CentOS 5.5 cluster composed of 12 compute nodes with additional 5 server nodes. Each compute node is equipped with 8 CPUs (cores), 250GB local disk space and 24GB ROM. The system is also equipped with a large shared Panasas parallel file system with disk space of 42TB. The OpenPBS torque/Maui queuing system is used for the analytical job submission and control.

3. Results and Discussion

To demonstrate the analytical process of this bioinformatics platform, as well as its ability of identifying the DNA origin for sequencing data, we present three analysis examples: 1) simulated sequence data-sets, 2) a laboratory-generated cultured bacterial data-set, and 3) a laboratory-generated complex viral data-set.

3.1 Application of the Bioinformatics Platform Analytical Capability on a Laboratory-Cultured Bacterial Data: DNA Extracted from Organism with Both Chromosome and Plasmid Representation

A laboratory-cultured *Yersinia pestis* strain Antiqua sample was sequenced, and the sequence data were processed by this pipeline (Table 1). The sequence data have 479,496 reads that resulted in 222 contigs. 124 of the 222 contigs have a high-quality match to the *Yersinia pestis* Antiqua sequence in the NCBI nt data-set. Over 92% of the original reads mapped to the *Yersinia pestis* Antiqua sequence well. The remaining 8% of the original reads were further assembled and the contigs were analyzed to be matching a variety of *Y. pestis* sequences in NCBI nt data-set. The results provided emphasize that the nearest-neighbors identified in each iteration do not necessarily follow the rank-order of the hit list in the first iteration. For example, the top-most-ranked sequence in iteration 4 was ranked 2nd in first-iteration.

Table 1. Pipeline processing results on cultured *Yersinia Pestis* sequencing data

Iteration	Reads Count	Contigs Count	Top Hit Contigs Count	Mapped Reads	Top Hit Record	rank in iteration				
						1	2	3	4	5
1	479496	222	124	441743	Yersinia pestis Antiqua, complete genome	1				
2	37753	13	4	12843	Yersinia pestis Antiqua plasmid pMT, complete sequence	4	1			
3	24910	8	2	23507	Yersinia pestis Z176003 plasmid pCD1, complete sequence	15	2	1		
4	1403	3	2	309	Yersinia pestis Z176003, complete genome	2	4	3	1	
5	1094	1	1	752	Yersinia pestis D182038 plasmid pPCP1, complete sequence	16	-	-	32	1
6	342	0								

3.2 Application of the Bioinformatics Platform Analytical Capability on Simulated Sequence Data: Simulation of Complex Matrices Analyzed Without Preprocessing

Three sequences from NCBI GenBank were selected to respectively represent the bacterial, viral and human nucleic acid components of complex data samples (Table 2). By using the MetaSim program (Richter, 2008), two simulated data-sets employing the Roche 454 error model were generated. Read counts and relative base/reads abundances are listed in Table 3. One data-set (data1) contains roughly equal amounts of DNA base/reads content of the three representative genomes, while the other (data2) having predominant human DNA sequences. Both data-sets contain 500,000 simulated-reads, and over 98% of reads have lengths in 200–300 bp range. The Roche 454 error model statistics for the two data-sets are listed in Table 4.

Table 2. DNA sequences selected from NCBI Genbank for simulation

Name	GI	Length	Description
BAC	49176966	5228663	Bacillus anthracis str. Sterne chromosome, complete genome
VIR	300872625	196959	Monkeypox virus strain Zaire 1979-005, complete genome
HUM	26 contigs	226719604	Homo sapiens chromosome 1 genomic contig, GRCh37.p2 reference primary assembly

Table 3. The MetaSim simulated data sets of 500,000 reads

	data1			data2		
Sequence	Copy number	reads	(base/reads) %	Copy number	reads	(base/reads) %
BAC	4350	167064	33.41%	40	4593	0.92%
VIR	11500	166462	33.29%	220	961	0.19%
HUM	100	166474	33.29%	100	494446	98.89%

Table 4. The 454 Error Model run statistics for the two simulated data-sets

454 Error Model Run Statistics:	data1	data2
Total Number of Flows:	258000000	258000000
Total Number of Negative Flows:	133138912 (51.6%)	136868032 (53.0%)
Total Number of one-mer Flows:	90737144 (35.2%)	87004099 (33.7%)
Total Number of two-mer Flows:	23478271 (9.1%)	22774870 (8.8%)
Generated:	500000 Reads	500000 Reads
Average Read Length:	356.72 Base Pairs	352.10 Base Pairs
Processed:	175391153 Base Pairs	172980897 Base Pairs
Generated:	178359557 Base Pairs	176050379 Base Pairs
Insertions:	4063681	4157127
Deletions:	1095277	1087645
Substitutions:	0	0

The pipeline analysis results on the two simulated data-sets are summarized in Table 5. For **data1** with equal DNA content of the three representative DNA sequences, the *de novo* assembly step of the pipeline (Figure 1) produced 412 contigs, with lengths range from 200 to 339,550 bp. Comparison of all the contigs with sequences in NCBI nt (bacterial/viral subset) in the nearest-neighbor identification step showed that 40 contigs matched best to the correct nearest-neighbor, BAC, in a completely blinded fashion. This reference was selected by the algorithm as the top-most hit. The VIR sequence was ranked at top 9th. Next, the reads were mapped against the BAC reference and all BAC reads in the original data, in addition to 26 HUM reads were mapped. The outstanding reads (unmapped, all from VIR and HUM components) were collected and subjected to the next iteration of data processing.

In the further iterations (iteration 2 through 5), the data-size was gradually reduced in iterations which selected human references, Stealth virus, and uncultured bacterial clones. Analysis is ongoing to determine the identification of uncultured bacterial clones as a near-neighbor in this simulated data-set. VIR was identified as the nearest-neighbor by the pipeline in iteration 6. Seven contigs best-matched with this sequence. Reference mapping resulted in the assignment of all the VIR reads, in addition to 14 HUM reads, to this nearest-neighbor. After this iteration, there are 142,949 outstanding (unmapped) reads, which are all originated from the HUM component.

Table 5. Pipeline processing results on the two simulated sequence data-sets

			Mapped %		
Iteration	(Topmost hit) [BAC/VIR sequences in top 100]	Mapped reads	BAC	VIR	HUM
1	(a); [VIR @ 2nd, no BAC]	116982 HUM	0.00%	0.00%	23.66%
2	VIR; [BAC @ 10th]	all VIR; 11 HUM	0.00%	100.00%	0.00%
3	(b); [no BAC]	10329 HUM	0.00%	na	2.09%
4	(c); [BAC @ 11th]	685 HUM	0.00%	na	0.14%
5	(d); [no BAC]	5834 HUM	0.00%	na	1.18%
6	(e); [BAC @ 6th]	349 HUM	0.00%	na	0.07%
7	(f); [no BAC]	4278 HUM	0.00%	na	0.87%

Iteration	(Topmost hit) [BAC/VIR sequences in top 100]	Mapped reads	Mapped %		
			BAC	VIR	HUM
8	(g); [BAC @ 4th]	66 HUM	0.00%	na	0.01%
9	(h); [no BAC]	8461 HUM	0.00%	na	0.98%
10	(i); [BAC @ 3rd]	141 HUM	0.00%	na	0.03%
11	(j); [no BAC]	3014 HUM	0.00%	na	0.61%
12	(k); [BAC @ 2nd)	65 HUM	0.00%	na	0.01%
13	(l); [no BAC]	3188 HUM	0.00%	na	0.64%
14	(BAC)	all BAC	100.00%	na	0.00%

(a) Homo sapiens BAC clone CH17-43A10 from chromosome 1, complete sequence
(b) Homo sapiens BAC clone CH17-118O6 from chromosome 1, complete sequence
(c) Human endogenous retrovirus H HERV-H/env59 proviral copy, clone 916F3
(d) Human DNA sequence from clone RP11-417J8 on chromosome 1 Contains three novel genes, two pseudogenes similar to part of sorting nexin associated golgi protein 1 (SNAG1) , a ankyrin repeat domain containing protein pseudogene and a novel (DKFZp434A171) pseudogene, complete sequence
(e) Human endogenous retrovirus HERV-K(II) DNA, complete sequence and flanking region
(f) Human DNA sequence from clone RP11-404H1 on chromosome 1, complete sequence
(g) Multiple sclerosis associated retrovirus polyprotein (pol) mRNA, partial cds
(h) Homo sapiens BAC clone CH17-366F13 from chromosome 1, complete sequence
(i) Staphylococcus capitis clone 3d human Alu/LINES1-directed PCR sequence
(j) Homo sapiens chromosome 1 clone RP11-153F1, complete sequence
(k) Uncultured bacterium clone LM0ABA3ZG12RM1 genomic sequence
(l) Homo sapiens BAC clone CH17-76O21 from chromosome 1, complete sequence

For a better representation of real-world complex sample data, we created a second simulated data-set (**data2**) that contained 99% HUM DNA (Table 3). This data-set was subjected the same pipeline analysis as **data1**, and the results are listed in the second half of Table 4. Because of the predominant human DNA reads in this data-set, the *de novo* assembly of the original simulated reads produced a large number of contigs (4,690) with low depth-of-coverage. Comparison of all the contigs with sequences in NCBI nt (bacterial/viral subset) ranked VIR sequence at 7th and BAC sequence at 43th in the first iteration. In further iterations, while the data-size was gradually reduced by reference mapping, both the VIR and BAC sequences were gradually moved up in the ranked hit list. The VIR sequence eventually became the top-most hit at iteration 4. At iteration 7, the nearest-neighbor identification step identified a human endogenous retrovirus sequence as the top hit, but none of the outstanding reads map to this sequence well, leading to the analytical pipeline's termination.

To minimize the data processing time, the nearest-neighbor identification step in the original pipeline design only used the reduced bacterial/viral subset of NCBI nt database as the target reference data-set. As the **data2** results demonstrated above, when human (or other eukaryotic) DNA is predominant in the reads data, the narrowed target reference data-set can also weaken the sensitivity of the analytical process in searching for bacterial and viral components of the sample. The BAC component in **data2** was not properly identified, due to the fact that the largest proportion of contigs is composed of HUM-reads. As a result, the hits from BAC component contigs were pushed down the ranked hit list. Applying the full-NCBI nt database on this test data in each of the iterations not only greatly increased the computational time required, but also pushed the BAC sequence further down the ranked hit list because of the additional large number of higher-ranked hits from contigs originated from the HUM reads. To recover the lowered sensitivity of the original pipeline design for complex sample data, a number of variations in the nearest-neighbor identification algorithm have been introduced into the pipeline implementation. One variation was to alternatively apply the narrowed- and the full-NCBI nt data-set in the odd/even iterative cycles. The results applying this variation to **data2** are listed in Table 6, with the shaded row iteration (odd number of iteration) using the full-NCBI nt data-set, while the reduced-NCBI nt data-set was used in the even number iterations. When these results are compared with Table 5, both the BAC and VIR sequences were moved up towards the top in the ranked list in fewer iterations. The VIR sequence was identified in iteration 2 instead of iteration 4. The BAC sequence was identified as a high-quality near-neighbor in iteration 6 in Table 6, while it was only listed as a possible near-neighbor during the iteration cycles in Table 5. The most significant achievement by this new algorithm was that, while Table 4 failed to identify the BAC component as a near-neighbor in the original reads data, the new algorithm succeeded in identifying the BAC sequence in 14 iterations (Table 6).

Table 6. Pipeline processing results on data2 with the NCBI nt db variation

Iteration	(Topmost hit) [BAC/VIR sequences in top 100]	Mapped reads	Mapped %		
			BAC	VIR	HUM
1	(a); [VIR @ 2nd, no BAC]	33566 HUM	0.00%	0.00%	6.79%
2	VIR; [BAC @ 10th]	83491 HUM	0.00%	0.00%	16.89%
3	(b); [no BAC]	all VIR; 11 HUM	0.00%	100.00%	0.00%
4	(c); [BAC @ 11th]	10281 HUM	0.00%	na	2.08%
5	(d); [no BAC]	685 HUM	0.00%	na	0.14%
6	(e); [BAC @ 6th]	5827 HUM	0.00%	na	1.18%
7	(f); [no BAC]	349 HUM	0.00%	na	0.07%
8	(g); [BAC @ 4th]	4276 HUM	0.00%	na	0.86%
9	(h); [no BAC]	66 HUM	0.00%	na	0.01%
10	(i); [BAC @ 3rd]	4849 HUM	0.00%	na	0.98%
11	(j); [no BAC]	141 HUM	0.00%	na	0.03%
12	(k); [BAC @ 2nd)	3013 HUM	0.00%	na	0.61%
13	(l); [no BAC]	65 HUM	0.00%	na	0.01%
14	(BAC)	3188 HUM	0.00%	na	0.64%

(a) Homo sapiens BAC clone CH17-43A10 from chromosome 1, complete sequence
(b) Homo sapiens BAC clone CH17-118O6 from chromosome 1, complete sequence
(c) Human endogenous retrovirus H HERV-H/env59 proviral copy, clone 916F3
(d) Human DNA sequence from clone RP11-417J8 on chromosome 1 Contains three novel genes, two pseudogenes similar to part of sorting nexin associated golgi protein 1 (SNAG1) , a ankyrin repeat domain containing protein pseudogene and a novel (DKFZp434A171) pseudogene, complete sequence
(e) Human endogenous retrovirus HERV-K(II) DNA, complete sequence and flanking region
(f) Human DNA sequence from clone RP11-404H1 on chromosome 1, complete sequence
(g) Multiple sclerosis associated retrovirus polyprotein (pol) mRNA, partial cds
(h) Homo sapiens BAC clone CH17-366F13 from chromosome 1, complete sequence
(i) Staphylococcus capitis clone 3d human Alu/LINES1-directed PCR sequence
(j) Homo sapiens chromosome 1 clone RP11-153F1, complete sequence
(k) Uncultured bacterium clone LM0ABA3ZG12RM1 genomic sequence
(l) Homo sapiens BAC clone CH17-76O21 from chromosome 1, complete sequence

3.3 Application of the Bioinformatics Platform Analytical Capability on a Viral Sample: Complex Matrix Analyzed With Pre-Processing

Viral sequence data containing host nucleic acid sequences often cause difficulties for Newbler's *de novo* assembly algorithm. It has been observed that the GS Newbler *de novo* assembler execution on this type of sequence data takes a very long time to complete, or fails due to an "out of memory" error. To address this potential problem, the pipeline provides an optional pre-processing, developed by Columbia's Center for Infection and Immunity, to compare all the reads with a set of host DNA sequences. When selected, this process masks the positively identified host DNA sequences in the input DNA sequence data, and reduces the computational burden of host DNA sequences entering into the iterative analysis.

A data-set was selected which did not successfully result in *de novo* assembly due to an "out of memory" error. The results of applying the host-identification pre-processing procedure to this sample data-set are listed in Tables 7 and 8. The pre-processing step identified the majority of the reads (54,438 reads, or 98.6%) to be from the host genome, leaving only 759 reads to analyze further (Table 7). The de novo assembly on the 759 viral reads produced 3 contigs, all matching well to Lujo virus (Table 8).

Table 7. Host-identification pre-processing effects on viral sequence data processing

sequences and results	original	After preprocessing	
		non-host	host
Reads	55227	759	54468
Bases	13789439	46347	13726989
Length min	40	33	40
Length max	587	508	587
pipeline results	*de novo* assembly by Newbler runs out of memory	Finished in 3 iterations, terminated because *de novo* assembly in 3rd iteration does not produce contigs	

Table 8. Pipeline processing results for non-host reads

Iteration	Reads (length)	Contigs (length)	Top Hit Contigs Count	Mapped Reads	Top Hit Record
1	759 (33–508)	3 (492–779)	2	709	Lujo virus segment S glycoprotein precursor and nucleocapsid protein genes, complete cds
2	90 (50–508)	1 (393)	1	21	Lujo virus segment L multifunctional matrix-like protein and large RNA-dependent RNA polymerase genes, complete cds

In another demonstration, when the host-identification methods were applied to the simulated data-set of **data2**, almost all of the human DNA sequences were successfully masked, the iterative analysis was completed in two cycles, and the VIR were properly identified in the first iteration. The outstanding 4,503 reads from BAC failed to produce any contigs. A separate *de novo* assembly calculation on the original 4,593 reads from BAC produced 3 contigs with a median depth of coverage of 3. This illustrates that the number of BAC-reads generated in this simulated data was right at the limits of sensitivity for Newbler. Improvements in sensitivity and accuracy continue to be pursued for these automated methods.

Table 9. Host-identification preprocessing effects on reads in data2

	Original reads			After preprocessing					
				Non-host reads			Host reads		
	BAC	VIR	HUM	BAC	VIR	HUM	BAC	VIR	HUM
	4593	961	494446	4503	955	118	90	6	494328
pipeline results	Failed to identify BAC without using the full NCBI nt by BLAST			Completed in 2 iterations, identified VIR in first iteration and leaving all the 4503 BAC reads outstanding. 4503 BAC reads failed to produce contigs.					

4. Conclusion

In today's next-generation sequencing environment, high-throughput sequencing technologies have allowed scientists to gather unprecedented amounts of data at astonishing rates. Making sense of these large volumes of data pose new and significant challenges to both computational and system biologists. The development of applied mathematical and computational tools leading to dynamic bioinformatics methods and pipelines can open the door for multiple applications of this critical, actionable knowledge. Technologies ranging from host biomarker discovery, pathogen identification and characterization, all the way to bio-surveillance, point-of-need diagnostics and therapeutics development depend on bioinformatics platforms to elucidate discovery and provide a robust pipeline of CBRN capabilities and products. This paper presents an overview of an ongoing effort for the design and implementation of a bioinformatics platform providing the capability of quickly analyzing DNA sequencing data and identifying the bacterial and viral components within a given sample. This application has been installed on the HPC facility of AFRL at Dayton, Ohio. Results presented were derived by applying the bioinformatics analytical capability on simulated as well as laboratory-generated sequenced data-sets, and demonstrates the developed system is capable of analyzing complex sequence samples while properly identifying the bacterial and viral components of the sample source. Also presented and provided were a number of known difficulties inherent in sequence data and/or data processing design. Solutions to address these difficulties were described.

Acknowledgements

This work was made possible by the Defense Threat Reduction Agency effort CB3576 to C.N.R and CB2847 to H.S.G and C.N.R. Conclusions and opinions presented here are those of the authors and are not the official policy of the US Army, ECBC, or the US Government. Information in this report is cleared for public release and distribution is unlimited. Thanks to Kelley Betts for her critical review of the paper, and Jessica Hill for her data preparation assistance.

References

Bentley, D. et al., "Accurate whole human genome sequencing using reversible terminator chemistry", *Nature*, 456, pp. 53–59, 2008.

Huson, Daniel H., et al., "MEGAN analysis of meta-genomic data", *Genome Res.*, 17(3), pp. 377–86, March 2007.

Jonathan, E.A., et al., "DNA signatures for detecting genetic engineering in bacteria", *Genome Biology*, 9(3):R56, 2008.

Margulies, M., et al., "Genome sequencing in micro-fabricated high-density picolitre reactors", *Nature*, 437, pp. 376–380, 2005.

Richter, D.C., et al., "MetaSim- A Sequencing Simulator for Genomics and Meta-genomics", *PLoS ONE*, 3(10): e3373. doi:10.1371/journal.pone.0003373, October 2008.

Shendure, J., et al., "Accurate multiplex polony sequencing of an evolved bacterial genome", *Science*, New York, NY, 309, pp. 1728–1732, 2005.

A Web-based High-Throughput Tool for Next-Generation Sequence Annotation

Kamal Kumar, Valmik Desai, Li Cheng, Maxim Khitrov, Deepak Grover, Ravi Vijaya Satya, Chenggang Yu, Nela Zavaljevski, and Jaques Reifman
Biotechnology HPC Software Applications Institute, Telemedicine and Advanced Technology Research Center, US Army Medical Research and Materiel Command, Fort Detrick, MD
{kamal, valmik, lcheng, mkhitrov, dgrover, rvijaya, cyu, nelaz, reifman}@bioanalysis.org

Abstract

The availability of a large number of genome sequences, resulting from inexpensive, high-throughput next-generation sequencing platforms, has created the need for an integrated, fully-automated, rapid, and high-throughput annotation capability that is also easy-to-use. Here, we present a web-based software application, Annotation of Genome Sequences (AGeS), which incorporates publicly-available and in-house-developed bioinformatics tools and databases, many of which are parallelized for high-throughput performance. The current version of AGeS provides annotations for bacterial genome sequences, and serves as a readily-accessible resource to Department of Defense (DoD) scientists for storing, annotating, and visualizing genomes of newly-sequenced pathogens of interest.

The AGeS system is composed of two major components. The first component is a web-based application that provides a graphical user interface for managing users' input genomes, submitting annotation jobs, and visualizing results. Sequence contigs are uploaded as a multi-FASTA input file and submitted for annotation, and the resulting annotations are visualized through GBrowse. The input genome sequences and the annotation results are stored in a secure, customized database. The second component is a high-throughput annotation pipeline for finding the genomic regions that code for proteins, RNAs, and other genomic elements through a Do-It-Yourself Annotation framework. The pipeline also functionally annotates the protein-coding regions using an in-house-developed high-throughput pipeline, the Pipeline for Protein Annotation. The annotation pipeline has been deployed on the Mana Linux cluster at the Maui High Performance Computing Center. The two components are connected together using the DoD user interface toolkit application programming interface.

The AGeS system was evaluated for scaling of its parallel execution and annotation performance. AGeS scaled with super-linear speedup for up to 128 processors, after which performance degraded. A 2.2-Mbp bacterial genome sequence can be annotated in ~1 hr using 128 processors. AGeS annotations of draft and complete genomes were compared with the original annotations from three different sources, and were found to be in general agreement with them.

1. Introduction

Access to inexpensive, high-throughput DNA sequencing technology has led to an explosion in the number of sequenced organisms and the volume of sequenced data[1]. To date, due to the so called "next-generation sequencing" technology, the genomes of >1,000 microbial pathogens and their near neighbors are available, and many more are being sequenced. A genome sequence provides valuable information in terms of genomic features, such as genes that code for proteins and RNAs, as well as the positions and numbers of tandem repeats. In addition, we can gain further insights by annotating the functions of the proteins that the genes code for. This valuable information, gleaned from the annotation of a newly sequenced complete genome, can help devise new strategies in diagnostics and forensics. Moreover, these annotations, coupled with comparative genomics, can enable novel approaches to identify vaccine candidates and potentially discover "universal" drug targets. For such downstream applications, the annotation of genomic sequences needs to be integrated, fully-automated, rapid, and high-throughput; and for such annotation capability to be truly effective, it should also be easy-to-use and readily available.

To address this need, we developed the Annotation of Genome Sequences (AGeS) software system, which was designed as a modular and flexible platform to facilitate the annotation, storage, and comparative analysis of sequenced genomes[2]. The AGeS system is composed of a Web-based application and a software pipeline. The Web-based application enables users to upload and store input contig sequences and the resulting annotation data in a central, customized database and users

can visualize the annotations via easy-to-use graphical user interfaces (GUIs). The visualization of annotated sequences is presented using the open-source genome browser GBrowse[3]. The integrated software pipeline analyzes contig sequences, and locates genomic regions that code for proteins, RNAs, and other genomic elements through a Do-It-Yourself Annotation (DIYA) framework[4] and Tandem Repeats Finder (TRF)[5]. The identified protein-coding regions are then functionally annotated using an in-house-developed high-throughput pipeline, the Pipeline for Protein Annotation (PIPA)[6]. All of these capabilities are available for bacterial genomes. Overall, AGeS provides the functionalities to: 1) store input sequences and annotated sequence data, 2) annotate completed and draft bacterial genomes in a fully-integrated and automated manner, 3) use high performance computing (HPC) for high-throughput annotation through efficient parallelization of the various publicly-available and in-house-developed bioinformatics resources, 4) visualize annotations using the familiar GBrowse[3] interface, and 5) download annotated genomes in GenBank[7] format.

Several software systems have recently been developed for high-quality, automated annotation of bacterial genomes. These include BASys[8], RAST[9], and Microbial Genomes Database Web resources[10] as well as annotation services provided by some of the large genomic annotation centers, such as the Annotation Engine at the J. Craig Venter Institute (http://www.jcvi.org/cms/research/projects/annotation-service/), the Genoscope's annotation service MicroScope[11], and the Microbial Annotation Pipeline at Integrated Microbial Genomes[12]. However, these systems or services do not provide integrated, fully-automated, rapid, high-throughput, and readily-available capability, and some of the important features, such as mapping to standard Gene Ontology (GO) annotation[13], are also missing. Although most annotation systems contain components that are based on publicly-available bioinformatics programs and databases, integration of these components into pipelines is not a trivial task for researchers without significant bioinformatics and computer science expertise. While recently published DIYA[4] and the Genome Reverse Compiler[14] provide integrated software packages for genome annotation, they do not enable the full use of parallel computing and lack fully-integrated and automated visualization of annotations.

2. Methods and Implementation

The AGeS system is composed of two main components: a Web-based application that provides user-friendly GUIs accessible via a standard Web browser; and a high-throughput software pipeline for the annotation of input genome sequences. Figure 1 shows the overall system architecture of the AGeS system. The AGeS Web application has been designed to control all aspects of the annotation process, i.e., input sequence management for uploading and manipulating genomic sequences, submitting annotation jobs to the AGeS annotation pipeline at an HPC cluster, storing input sequences as well as annotation results into a central relational database management system (RDBMS), and visualizing the annotations in the integrated GBrowse genome browser. For uploading the genome sequences, along with the required genus, species, and strain information, users have the option to upload the data pertinent to the minimum information about a genomic sequence (MIGS)[15]. Internally, the AGeS Web application uses a workflow manager module to guide the entire lifecycle of the annotation process, starting from the upload of an input sequence and ending with the visualization of the annotated sequences.

2.1 Web Application

The AGeS system is accessible at https://applications.bioanalysis.org/ages/, and is available to the Department of Defense (DoD) Supercomputing Resource Centers (DSRCs) users for genome sequence annotation using a standard Web browser. The AGeS Web application has been designed as a modular application for the easy integration of future sequence analysis modules, as they become available, and uses a workflow manager to invoke its modules. Resource-intensive annotation tools are run on the Mana Linux cluster at the Maui HPC Center (MHPCC), which is accessed by the Web application using the DoD User Interface Toolkit (UIT) application programming interface (API) (https://www.uit.hpc.mil/). UIT is a Web service-based API that provides secure access to DoD HPC resources. AGeS users are authenticated through the UIT API using their Kerberos credentials. The AGeS Web application provides GUIs for managing sequences, submitting annotation jobs to the HPC cluster, and visualizing and downloading the annotation results. Figure 2 shows a screenshot of the AGeS Web application, showing the sequence management GUI. When an annotation job is completed on the HPC end, the results are automatically transferred back to the Web server and stored into the central database for visualization and download. Upon completion of an annotation job, an e-mail is also sent automatically to the user.

The AGeS Web application was developed using standards-based technologies, which include Java (http://www.oracle.com/technetwork/java/), J2EE (http://www.oracle.com/technetwork/java/javaee/overview/), JavaServer Faces (JSF) (http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html), asynchronous JavaScript and XML (AJAX)[16],

ICEfaces (http://www.icefaces.org/), jBPM (http://www.jboss.org/jbpm/), and Apache ActiveMQ (http://activemq.apache.org/). The Web application mainly consists of server-side Java codes that use JSF- and AJAX-based APIs from ICEfaces. ICEfaces provides a rich set of user interface components, such as menus, buttons, etc., and generates updated views of Webpages without reloading the entire page. The workflow manager module has been implemented, within the Web application, using the jBPM workflow engine API for controlling the execution of various modules. The Web application uses an Apache ActiveMQ server for asynchronous message passing between the modules and the workflow engine. A PostgreSQL (http://www.postgresql.org/) RDBMS server is used to store users' input genome sequences, annotation results, and other job-related data. The Web application is deployed on an Apache Tomcat (http://tomcat.apache.org/) server, using a secure hypertext transfer protocol over a secure socket layer connection for encrypting all of the data flowing to and from the user's Web browser.

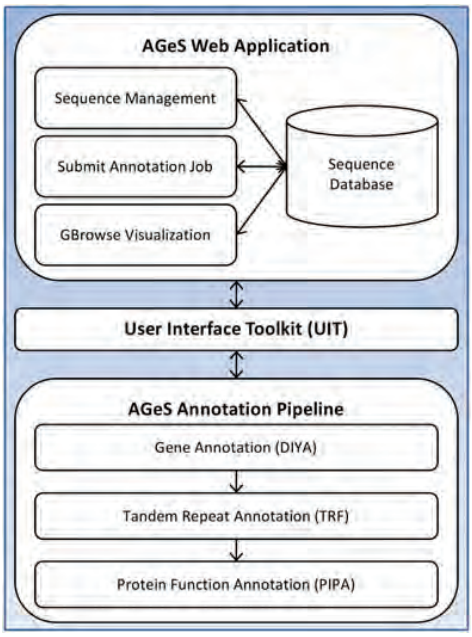


Figure 1. Overall system architecture for the Annotation of Genome Sequences (AGeS) system

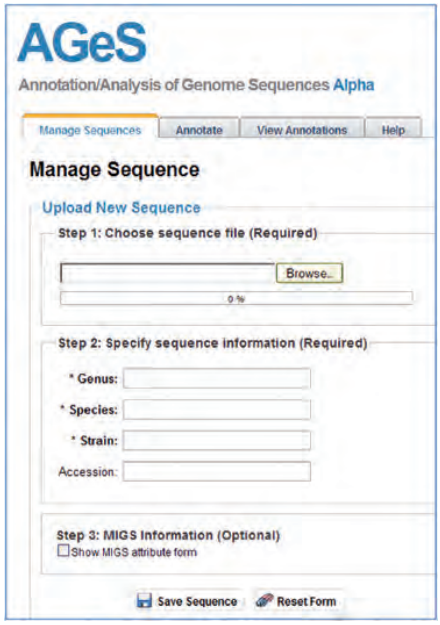


Figure 2. The AGeS graphical user interface used for sequence data management

2.2 Annotation Pipeline

As shown in Figure 1, the AGeS annotation pipeline is composed of three modules for gene, tandem repeats, and protein function annotations. The annotation pipeline takes assembled contiguous sequences, or contigs, as input in multi-FASTA format files generated by high-throughput, next-generation sequencing technologies (http://www.454.com/, http://www.illumina.com/, and http://www.appliedbiosystems.com/). First, a customized DIYA[4] framework is used to locate protein-coding genes using Glimmer[17] and RNA genes using RNAmmer[18] and tRNAscan-SE[19]. Within the DIYA framework, the system uses BLAST[20] searches to extract coding regions from the Glimmer predictions, and to infer gene products by transferring annotation from the best BLAST match. Next, the system finds tandem repeats in the pseudo-assembled sequence using TRF[5]. Outputs from the different DIYA component programs and TRF are post-processed and parsed to generate a file in GenBank format.

After annotation of the genomic regions is complete, the identified protein-coding regions are annotated using the high-throughput protein function annotation methods implemented in PIPA[6]. One of the most useful features of PIPA is that it exploits and consistently consolidates protein function information from disparate sources, including the in-house-developed CatFam enzyme profile database[21]. As an added benefit, the consolidated function predictions are given in GO terms, which is the de facto standard for protein annotation. The protein annotation results from PIPA are included in the GenBank file from the previous step, and are transferred back to the AGeS Web application for storage into the central database.

3. Results

AGeS provides the capability to annotate whole bacterial genomes, including both genomic features and protein functions. The annotation pipeline that has been deployed on the Mana Linux cluster at the MHPCC scales well and is suited for whole genome sequence annotation. In this section, we present the results of the parallel processing performance testing of AGeS as well as of the software validation experiments.

3.1 Parallel Performance

To assess the scalability of the parallelization of the annotation modules of the AGeS pipeline, we computed the speedup curve for the annotation of a typical bacterial genome (Figure 3). Speedup is defined as the ratio of the time taken by a program to run on *N* processors to the time taken to run the same program on a single processor, with an ideal speedup being linear, meaning that the speedup is directly proportional to the number of processors. AGeS achieves super-linear speedup for up to 128 processors, after which its performance declines. The super-linear speedup is attributed to faster processing achieved by fully using the processors' local memory, and the speedup decline beyond 128 processors is attributed to communication overhead. A 2.2-Mbp bacterial genome sequence (e.g., *Staphylococcus hominis* SK119, which is an opportunistic pathogen in patients with a compromised immune system) can be annotated in ~1 hr using 128 processors.

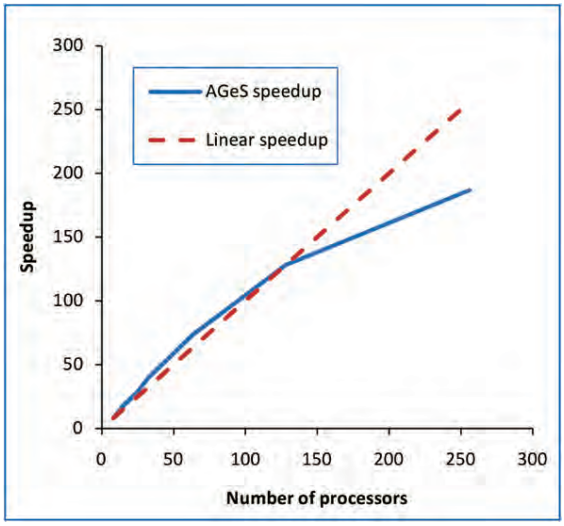


Figure 3. AGeS performance speedup as a function of the number of processors

3.2 Software Validation

We validated AGeS by comparing its annotations of bacterial genomes with annotations from three other sources. We evaluated two draft genomes, *Staphylococcus hominis* SK119 and *Staphylococcus aureus* subsp. *aureus* TCH60, and one completed genome, *Yersinia pestis* CO92. The *S. hominis* draft genome, sequenced by J. Craig Venter Institute (http://www.jcvi.org/cms/research/groups/microbial-environmental-genomics/), consists of 37 contigs, and the *S. aureus* draft genome, sequenced by the Human Genome Sequencing Center at Baylor College of Medicine (http://www.hgsc.bcm.tmc.edu/), consists of 65 contigs. Both of these draft genomes were sequenced using 454 pyrosequencing technology (http://www.454.com/). The complete *Y. pestis* genome was sequenced by the Wellcome Trust Sanger Institute (http://www.sanger.ac.uk/resources/downloads/bacteria/yersinia.html) using Sanger sequencing technology.

The annotations for these three genomes were retrieved from the corresponding sequencing centers, and their sequences were re-annotated using the AGeS system. Figure 4A shows a subset of the compared genomic features[2]. The total number of annotated genes for each of these genomes was compared with the original annotations provided by the corresponding centers. Each of the two compared annotation sources predicted similar numbers of genes. For *S. hominis* (Sh), we found that 1,753 (~78%) genes were identical across both predictions. Most of the remaining genes overlapped at the start or end positions, with only 0.2% of the predictions unique to AGeS (data not shown). For the *S. aureus* (Sa) genome, 2,037 (~77%) genes were identical, with only 1% of the predictions unique to AGeS (data not shown). For the *Y. pestis* (Yp) genome, 2,637 (>60%) genes were identical across the 2 annotations, and another ~30% had identical start or end positions (data not shown). Annotation comparisons indicated larger differences for the *Y. pestis* completed genome than for the two draft genomes. These differences could be attributed to the more extensive studies performed in this well-studied genome. A similar level of agreement was observed for other genomic features, such as CDSs, rRNAs, and tRNAs.

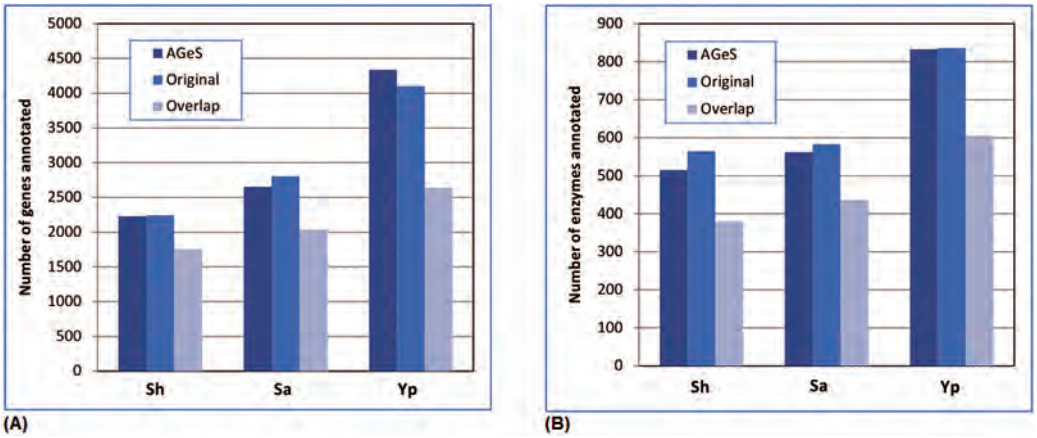


Figure 4. Comparison of gene annotations and enzyme function predictions between AGeS and the other three annotation systems for the three analyzed genomes, *Staphylococcus hominis* SK119 (Sh), *Staphylococcus aureus* subsp. *aureus* TCH60 (Sa), and *Yersinia pestis* CO92 (Yp). *A*: the number of genes predicted by the original annotation centers and AGeS, with the overlap corresponding to identical predictions. *B*: the number of enzymes predicted by the original annotation centers and AGeS, with the overlap corresponding to identical predictions.

We also compared the annotations of the enzyme functions predicted by the CatFam enzyme profile database with those provided by the other three annotation centers. Figure 4B shows the similar numbers of annotated enzymes for each of the three compared genomes[2]. For example, for the *S. hominis* (Sh) draft genome, CatFam assigned Enzyme Commission (EC) numbers for 515 genes, whereas the J. Craig Venter Institute assigned EC numbers to 565 genes, with 379 enzymes having identical EC number annotations. In general, our results indicate that the AGeS annotations are in agreement with the other evaluated methods both on the genomic and proteomic annotation levels.

4. Conclusion

The Web-based AGeS system described in this paper is a computationally-efficient and scalable system for high-throughput genome annotation of newly sequenced pathogens of military relevance and their near neighbors. The AGeS annotation pipeline is fully-parallelized and is currently operational at the Mana Linux cluster at the MHPCC, where we performed scalability tests and found that a 2.2-Mbp bacterial genome sequence can be annotated in ~1 hr using

128 processors. Validation results indicated that the AGeS system's annotations are in general agreement with the other evaluated methods, both on the genomic and proteomic annotation levels. Due to significant cost reductions afforded by the recently developed next-generation genome sequencing technologies, we expect that software applications such as AGeS will become vital for microbial comparative genomics studies.

Acknowledgements

This work was partially sponsored by the US DoD High Performance Computing Modernization Program, under the High Performance Computing Software Applications Institutes Initiative.

Disclaimer

The opinions and assertions contained herein are the private views of the authors, and are not to be construed as official or as reflecting the views of the US Army or of the US Department of Defense.

References

1. Hall, N., "Advanced sequencing technologies and their wider impact in microbiology", *The Journal of Experimental Biology*, 210(9), pp. 1518–1525, 2007.

2. Kumar, K., V. Desai, L. Cheng, M. Khitrov, D. Grover, R.V. Satya, C. Yu, N. Zavaljevski, and J. Reifman, "AGeS, a software system for microbial genome sequence annotation", *PLoS ONE,* 6(3), e17469, 2011.

3. Donlin, M.J., "Using the Generic Genome Browser (GBrowse)", *Current Protocols in Bioinformatics*, Chapter 9, pp. 9.9.1–25, 2009.

4. Stewart, A.C., B. Osborne, and T.D. Read, "DIYA, a bacterial annotation pipeline for any genomics lab", *Bioinformatics*, 25(7), pp. 962–963, 2009.

5. Benson, G., "Tandem repeats finder, a program to analyze DNA sequences", *Nucleic Acids Research*, 27(2), pp. 573–580, 1999.

6. Yu, C., N. Zavaljevski, V. Desai, S. Johnson, F.J. Stevens, and J. Reifman, "The development of PIPA, an integrated and automated pipeline for genome–wide protein function annotation", *BMC Bioinformatics*, 9, 52, 2008.

7. Benson, D.A., I. Karsch–Mizrachi, D.J. Lipman, J. Ostell, and E. W. Sayers, "GenBank", *Nucleic Acids Research*, 38(suppl 1), pp. D46–D51, 2010.

8. Van Domselaar, G.H., P. Stothard, S. Shrivastava, J.A. Cruz, A. Guo, X. Dong, P. Lu, D. Szafron, R. Greiner, and D.S. Wishart, "BASys, a web server for automated bacterial genome annotation", *Nucleic Acids Research*, 33(suppl 2), pp. W455–W459, 2005.

9. Aziz, R.K., D. Bartels, A.A. Best, M. DeJongh, T. Disz, R.A. Edwards, K. Formsma, S. Gerdes, E.M. Glass, M. Kubal, F. Meyer, G.J. Olsen, R. Olson, A.L. Osterman, R.A. Overbeek, L.K. McNeil, D. Paarmann, T. Paczian, B. Parrello, G.D. Pusch, C. Reich, R. Stevens, O. Vassieva, V. Vonstein, A. Wilke, and O. Zagnitko, "The RAST Server, rapid annotations using subsystems technology", *BMC Genomics*, 9, 75, 2008.

10. Uchiyama, I., T. Higuchi, and M. Kawai, "MBGD update 2010, toward a comprehensive resource for exploring microbial genome diversity", *Nucleic Acids Research*, 38(suppl 1), pp. D361–D365, 2010.

11. Vallenet, D., S. Engelen, D. Mornico, S. Cruveiller, L. Fleury, A. Lajus, Z. Rouy, D. Roche, G. Salvignol, C. Scarpelli, and C. Médigue, "MicroScope, a platform for microbial genome annotation and comparative genomics", *Database*, 2009, 2009.

12. Markowitz, V.M., I.–M. A. Chen, K. Palaniappan, K. Chu, E. Szeto, Y. Grechkin, A. Ratner, I. Anderson, A. Lykidis, K. Mavromatis, N.N. Ivanova, and N.C. Kyrpides, "The integrated microbial genomes system, an expanding comparative analysis resource", *Nucleic Acids Research*, 38(suppl 1), pp. D382–D390, 2010.

13. Ashburner, M., C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel–Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, and G. Sherlock, "Gene ontology, tool for the unification of biology", *Nature Genetics*, 25(1), pp. 25–29, 2000.

14. Warren, A. S. and J. C. Setubal, "The Genome Reverse Compiler, an explorative annotation tool", *BMC Bioinformatics*, 10, 35, 2009.

15. Field, D., G. Garrity, T. Gray, N. Morrison, J. Selengut, P. Sterk, T. Tatusova, N. Thomson, M.J. Allen, S.V. Angiuoli, M. Ashburner, N. Axelrod, S. Baldauf, S. Ballard, J. Boore, G. Cochrane, J. Cole, P. Dawyndt, P. De Vos, C. DePamphilis, R. Edwards, N. Faruque, R. Feldman, J. Gilbert, P. Gilna, F. O. Glockner, P. Goldstein, R. Guralnick, D. Haft, D. Hancock, H. Hermjakob, C. Hertz–Fowler, P. Hugenholtz, I. Joint, L. Kagan, M. Kane, J. Kennedy, G. Kowalchuk, R. Kottmann, E. Kolker, S. Kravitz, N. Kyrpides, J. Leebens–Mack, S.E. Lewis, K. Li, A.L. Lister, P. Lord, N. Maltsev, V. Markowitz, J. Martiny, B. Methe, I. Mizrachi, R. Moxon, K. Nelson, J. Parkhill, L. Proctor, O. White, S. A. Sansone, A. Spiers, R. Stevens, P. Swift, C. Taylor, Y. Tateno, A. Tett, S. Turner, D. Ussery, B. Vaughan, N. Ward, T. Whetzel, I. San Gil, G. Wilson, and A. Wipat, "The minimum information about a genome sequence (MIGS) specification", *Nature Biotechnology*, 26(5), pp. 541–547, 2008.

16. Paulson, L.D., “Building Rich Web Applications with Ajax”, *Computer*, 38(10), pp. 14–17, 2005.

17. Delcher, A.L., K.A. Bratke, E.C. Powers, and S.L. Salzberg, “Identifying bacterial genes and endosymbiont DNA with Glimmer”, *Bioinformatics*, 23(6), pp. 673–679, 2007.

18. Lagesen, K., P. Hallin, E.A. Rødland, H.–H. Stærfeldt, T. Rognes, and D.W. Ussery, “RNAmmer, consistent and rapid annotation of ribosomal RNA genes”, *Nucleic Acids Research*, 35(9), pp. 3100–3108, 2007.

19. Lowe, T.M. and S.R. Eddy, “tRNAscan–SE, a program for improved detection of transfer RNA genes in genomic sequence”, *Nucleic Acids Research*, 25(5), pp. 0955–0964, 1997.

20. Altschul, S.F., W. Gish, W. Miller, E.W. Myers, and D.J. Lipman, “Basic local alignment search tool”, *Journal of Molecular Biology*, 215(3), pp. 403–410, 1990.

21. Yu, C., N. Zavaljevski, V. Desai, and J. Reifman, “Genome–wide enzyme annotation with precision control, catalytic families (CatFam) databases”, *Proteins*, 74(2), pp. 449–460, 2009.

HPCMP UGC 2011

3. Computational Chemistry and Materials Science (CCM)

Computational Chemisty

The Generalized Gradient Approximation Applied to the Three-Dimensional Hubbard Model in a Trap: A CAP and Challenge Project

J.K. Freericks and K. Mikelsons
Department of Physics, Georgetown University, Washington, DC
{freericks, karlis}@physics.georgetown.edu

H.R. Krishnamurthy
Centre for Condensed Matter Theory, Department of Physics, Indian Institute of Science and Condensed Matter Theory Unit, Jawarharlal Nehru Centre for Advanced Scientific Research, Bangalore, India
hrkrish@physics.iisc.ernet.in

Abstract

We describe a generalization of the local density approximation for ultra-cold atomic systems, which takes into account the gradients of the Green's functions with respect to position when solving the local Dyson equation and, hence, is sensitive to the local variation of the system within the trap. We use this technique along with dynamical mean-field theory to simulate the Hubbard model data of the Swiss group, which measured the Mott transition in a three-dimensional optical lattice with ultra-cold ^{40}K atoms. We simulate the same size system as used in experiment, which includes about 8 million lattice sites and up to 300,000 atoms. We describe our computational algorithm in detail, along with modifications needed for the communications to make it work on the Cray XE6. We show strong scaling curves up to approximately 43,500 cores. The code performs remarkably well with perfect linear scaling up to the largest systems we ran on. The results that we present are a comparison of the dynamical mean-field theory results within the generalized gradient approximation to the strong-coupling perturbation theory, which is known to be accurate for high-temperature and large interaction strength. This work is part of the Defense Advanced Research Projects Agency (DARPA) optical lattice emulator program.

1. Introduction

The Department of Defense (DoD) is always interested in the newest materials which can outperform those that already exist. Currently, there is a dearth of design principles that can create a material with a specific set of desired properties. Instead, we usually follow an Edisonian approach, where we try many different kinds of materials and examine their properties to see if they are better than ones already discovered. While nearly all materials of interest that have been used in electronic devices, sensors, magnetic devices, etc. have been discovered in this fashion, we are now beginning to enter a new age where this might change. Conventional high performance computers allow for the computation of the materials properties of a wide-range of materials using techniques like density-functional theory, and there certainly have been a number of successful materials designed from first-principles, including electrodes for rechargeable lithium ion batteries, high-performing ferroelectrics for sonar applications, and so on. One class of materials, however, defies this approach because even these state-of-the-art techniques are not sufficient to properly describe their behavior. These materials are called strongly-correlated materials, and their complication comes from the fact that the electrons in them interact so strongly with each other that one needs to know what all other electrons are doing in order to determine how any particular electron will act. Only a very small percentage of all materials fall into this strongly-correlated electron class, but they display some of the most interesting behavior from high-temperature superconductivity, to ultra-strong permanent magnets, to materials with strongly renormalized energy scales like the heavy fermion materials. Because they display complex phenomena like metal-insulator transitions, complex magnetic ordering, and the so-called Kondo effect (marked by a loss of magnetism at low-temperature), it is believed that these materials may outperform more conventional materials in a number of different categories, and hence are a very fruitful area for new materials research.

The challenge, of course, is to be able to predict the properties of these materials from first-principles, which has become an extremely difficult problem. Some success has occurred by combining density-functional theory with dynamical mean-field theory, which is a new technique that can solve the quantum many-body problem exactly if one is in a universe

with a very large number of spatial dimensions (it turns out that three dimensions is close to this large-dimensional limit). Even so, there still remains a wide class of materials that have not yet yielded to a successful description with conventional computation, or the computational time is too long to use them for new materials design and discovery.

Enter Richard Feynman (Feynman, 1982) who suggested an alternative way to solve this problem by using an analog quantum computer. The basic idea is that nature itself is the most efficient computer for solving the quantum many-body problem. Hence, if we can set up a controlled system of quantum particles that obey the properties of the material we are trying to calculate properties for, then we just let the system evolve as a function of time and determine its properties by measurement. While this might sound like just what an experimental physicist would do in a laboratory, the key difference here is that Feynman was proposing creating an analog computer where we could rapidly "dial-in" the different materials properties and investigate a wide-class of materials over a short period of time. While some might argue that this would still be an Edisonian approach, it is a much more efficient Edisonian approach than trying to make a wide-class of new materials and measure all of their properties. We are not yet at the stage where such a Feynman analog computer is available, but we have started to construct simpler prototypes as part of the Defense Advanced Research Projects Agency (DARPA) program (Abo-Shaeer, 2010). One class of analog computers is made from ultra-cold neutral atoms placed in an optical lattice.

These optical lattice emulators are constructed by placing ultra-cold clouds of alkali atoms in the egg-carton-like potential energy surface created by retro-reflected laser beams in each spatial direction, which have a lattice spacing that is typically about 0.5 micron. This lattice spacing is about two hundred times larger than the lattice spacing in a material, which produces a huge advantage—we can directly image the atoms as a function-of-time, and see how they evolve due to their quantum-mechanical behavior. This would be the equivalent of having an ultrafast microscope which can image the electrons deep inside the bulk on a time scale short, compared to the time for an electron to move from one lattice site to the next. Unfortunately, we can only examine the simplest of models of strong electron correlation with these first-generation optical lattice emulators.

In order to verify that these optical lattice emulators are working properly, we need to also benchmark their accuracy. This approach is further complicated by the fact that these atomic systems involve clouds of atoms which are trapped in space at some particular location via a harmonic trap. The trap makes the system inhomogeneous; hence, one needs to use caution in comparing their behavior to those of quantum particles in a perfectly periodic lattice, which is needed to describe a real material. It is precisely the role of high performance computing to determine the properties of these inhomogeneous systems, and thereby verify and validate the analog optical lattice emulators. Since the exact problem is too difficult to solve, even with today's computers, we need to make approximations of one form or another. We try our best to do so in such a way that the approximations are faithful to the exact solutions and will yield interesting information about the behavior of the system.

Our numerical approach uses dynamical mean-field theory, which we already described as a formal technique to solve the many-body problem which becomes exact in the limit of an infinite number of spatial dimensions. While the approach is approximate in three dimensions, it is believed that the approximation is quite good for many systems. One huge advantage of this approach is that it is the only numerically-tractable method that can describe fermionic systems at low-temperature without facing the so-called sign problem, and is also efficient enough that it can model systems that are as large as the actual experimental systems being studied. We must generalize the original formulation of dynamical mean-field theory to inhomogeneous systems (called IDMFT) to take into account the effect of the trap. The approach works with objects called Green's functions which describe the amplitude and phase for creating a particle at one lattice site at time t_1 and removing the particle from another lattice site at time t_2. The Green's function can be parameterized by an object called the self-energy, which describes how the energy of the excitations is modified by the interactions between the particles, and how long-lived they are. In a conventional many-body problem, the self-energy is a function of both momentum and frequency. In an inhomogeneous system, momentum is no longer a good quantum number, so the momentum dependence would become a matrix-valued function of two momenta. Fortunately, in the IDMFT approach, the self-energy is actually independent of momenta, so we need only determine a (different) scalar function of frequency for each lattice site. The algorithm for solving IDMFT involves the following steps: 1) start with a guess for the self-energy on each lattice site; 2) calculate all the local Green's functions (for each frequency) by finding the diagonal of the inverse of a general complex and sparse matrix (whose dimension is the number of lattice sites) which includes the self-energy and trapping potential for each lattice site on the diagonal, and the connections between neighboring lattice sites on the off-diagonal elements; 3) extract the local effective medium by using the local Dyson's equation to remove the local self-energy from the local Green's function; 4) solve a quantum impurity problem in the extracted local effective medium for each lattice site to find the new local Green's function; 5) use the local Dyson's equation with the new local Green's function and the old effective medium to find the new self-energy; and 6) use the new self-energy in step (2) to continue the iterative algorithm. One repeats these

steps until the functions have converged, which typically takes from 10 to 1,000 iterations. The two numerically-intensive parts of the algorithm are finding the diagonal of the inverse of the large sparse matrix (step 2) and solving the local impurity problem on each lattice site (step 4). The former problem grows as N^3 with N the number of lattice sites, while the latter is also slow for complicated quantum models (like the Hubbard model), but grows linearly with the system size N. It turns out that finding the diagonal of the inverse of a few million by few million matrixes is not feasible with current computational resources which employ dense matrix algorithms. One can, instead use the sparsity of the matrix to really speed up the inversion and to reduce the storage requirements. We are currently working on developing such an algorithm, but it is not yet complete (Freericks, et al., 2010; Carrier, et al., 2011).

Instead, there is an alternative approach which is called the local density approximation (LDA). In this approach, we assume the system changes slowly from one lattice site to another, so we can replace the problem in the inhomogeneous system by a series of problems in homogeneous systems, but with a chemical potential for the homogeneous problem set by the global chemical potential of the inhomogeneous system minus the local potential due to the trap $\mu_{homog.}=\mu-V_i$. This technique requires us to essentially solve N independent homogeneous problems and then "stitch" them together into an approximation for the inhomogeneous system. This technique has long been used for these ultra-cold atomic systems in a trap, but it isn't obvious how accurate these techniques actually are. A systematic way to improve upon the LDA is needed. If the corrections to the LDA are small, then one can deduce that the LDA is a good approximation, and if the corrections are large, then it is not. In density-functional theory this same problem has been addressed many years ago, and it is now routine to include gradient corrections on top of the LDA. We can do the same thing here, and we similarly call this approach the generalized-gradient approximation (GGA) to make contact with the same ideas which arose from the density-functional theory community.

The derivation of the GGA equations is rather straightforward, but is somewhat lengthy and will be omitted here. We do describe; however, the basic ideas that are used. The key computational step which is difficult to achieve is the solution of the Dyson equation, which involves finding the diagonal of the inverse of an $N\times N$ matrix. The LDA approximation involves replacing the inhomogeneous problem, where along the diagonal, the local chemical potential (chemical potential minus trap potential) minus the local self-energy appears, by a homogeneous one, where every diagonal element is equal to the current diagonal element. By taking the difference of these two Dyson equations, we can find a formula for the gradient of the Green's function, which involves the difference of the local Green's function at one site and the local Green's functions at its nearest-neighbors. Solving this set of equations, where we discard the differences between the Green's function and the second-neighbors, yields a set of equations for the GGA (which is the correction of the LDA result by gradients of the Green's functions at the given site, and its nearest-neighbors). The modification of the IDMFT algorithm then rests solely with changes to step (2) of the above algorithm. For the LDA, we replace every diagonal by the diagonal element of the ith lattice site, and find the inverse by Fourier transformation. For the GGA, we need to solve a self-consistent equation which requires the LDA solutions at the given lattice site, at the nearest-neighbors, the nearest-neighbor Green's function (in the LDA) and the local GGA Green's function and self-energy. Because this step is embedded in the self-consistent solution of the IDMFT algorithm, we need to solve N impurity problems as well, at each iteration. For the GGA approach, it is the solution of these impurity problems which takes the most computational effort. For the full IDMFT approach, it is the computation of the diagonal of the inverse of the large sparse matrix that will likely be the most computationally-intensive step.

We also need to solve the impurity problem, which can be mapped onto an effective single-impurity Anderson model. No exact solution for this problem exists (when we have an arbitrary time-dependent source-field), but it can be solved with a number of different numerical techniques ranging from explicit perturbation theory, to quantum Monte Carlo (QMC) approaches, to the numerical renormalization group. We will use a QMC approach here, because the experiments are at high-temperature and the numerical renormalization group is best at lower-temperatures.

The QMC algorithm that we use is a weak-coupling version of the continuous-time quantum Monte Carlo algorithm (Rombouts, 1999; Rubtsov, et al., 2005). This approach uses stochastic sampling to evaluate Feynman diagrams in imaginary-time to solve the problem. The algorithm sums over a random collection of diagrams at different orders, and adjusts the order of the diagram based on importance sampling via a Metropolis strategy. The approach is most accurate at high-temperature and small interaction strength. As the temperature or the interaction is reduced, the average order of the calculation increases, and the integration range in imaginary-time also increases. This technique is currently believed to be the most accurate state-of-the-art approach for determining properties that can be found at finite-temperature using Green's functions evaluated along the imaginary-time axis (which includes the particle-density at each lattice site, the double-occupancy, the entropy-per-particle, and the order parameter if the system goes into an ordered anti-ferromagnetic phase). Details for how this algorithm is implemented can be found elsewhere (Jarrell, et al., 2008; Gull, et al., 2011).

We end this section by describing the model that is used, which is the Hubbard model in a trap (Hubbard, 1963). The Hubbard model includes three terms:

$$H = -\sum_{\langle i,j\rangle\sigma} t_{ij}\left(c_{i\sigma}^{+}c_{j\sigma} + c_{j\sigma}^{+}c_{i\sigma}\right) - \sum_{i}\left(\mu - V_i\right)c_{i\sigma}^{+}c_{i\sigma} + U\sum_{i} n_{i\uparrow}n_{i\downarrow} \quad (1)$$

Where $c_{i\sigma}^{+}$ and $c_{i\sigma}$ are fermionic creation and annihilation operators at site *i* which satisfy the canonical anti-commutation relations $\left\{c_{i\sigma}^{+}, c_{j\sigma'}\right\} = \delta_{ij}\delta_{\sigma\sigma'}$. The number operator is $n_{i\sigma} = c_{i\sigma}^{+}c_{i\sigma}$. The first term in Equation 1 is the fermion kinetic energy, which has an overall scale set by the hopping integral (*t*) and which involves a hopping of the fermions (of either spin) from one site to any of its nearest-neighbors. The second term is the local chemical potential, which is a site-energy given by the global chemical potential minus the trap potential at the given lattice site (which is taken to increase quadratically away from the origin, but with a different curvature along each of the different spatial axes, just like in experiment). The third term is the interaction term, which includes a Coulomb repulsion (*U*) multiplied by the double-occupancy, which is the number of up-spin particles multiplied by the number of down-spin particles at the lattice site. All of the parameters are taken directly from the experiment (Jördens, et al., 2008; Jördens, et al., 2010).

2. Scaling of the Numerical Algorithm

Since this project was selected for a Capability Applications Project (CAP) on the XE6 machines, we started by running scaling studies on the Cray XE6 machine at Engineering Research Development Center (ERDC) [garnet] which has approximately 22,000 cores. An initial porting of our code failed to run on the XE6. It turned out this was related to buffer issues for send and receive calls under Message Passing Interface (MPI). In the original code, we did not organize the communications to guarantee that they would be non-blocking. On most high-performance machines, such sloppiness is allowed because they receive and send buffers are large enough to wait for the expected calls and to receive them out of order. Not so on the XE6, so we were forced to engage in the more proper practice of ensuring we had non-blocking communications. Making these modifications was fairly straightforward to do, and once we had completed it, the code ran very well.

Before reporting our strong scaling analysis, we need to discuss a bit about how we chose the lattice sites in our system. We begin with an *M*×*M*×*M* cubic lattice (with M in the range from 100 to 250). We calculate the lattice potential at each lattice site, and include only those sites whose potential is smaller than the smallest value of the potential at the edge boundary of the cube. Then, using a fact that our system is symmetric with respect to mirror reflections along each axis, we can further reduce the number of sites for which we need to solve the QMC impurity problem, by solving it only for inequivalent lattice sites (all lattice sites must be used, however, in the Dyson equation). Since we cannot store the data for the Green's functions and self-energies on just one core, we have to distribute the memory. Our code is set up to distribute the memory on a number of cores which is equal to a multiple of two (because we use a binary bifurcation to assign lattice sites to a given core), or to the number of lattice sites divided by 2. In order to perform scaling studies, we need to vary the number of cores by increasing by a factor of 2 or by choosing them to equal *N*/2 (with N the number of inequivalent lattice sites now). Since these machines do not have the number of cores equal to a multiple of 2, we have to choose our lattice size appropriately to be able to map the same number of lattice sites to each core for the maximal number of cores, so that the system can equally distribute the labor for solving the QMC impurity problem (it turns out that the computational time is not the same, because it depends on the particle density at a given site, so there always is a spread in the computational time for different lattice sites and different runs). Using this approach, we can have an optimal way of testing the scaling of the code up to the maximal number of cores that we can run on.

We report results for the initial scaling analysis on garnet in Figure 1. This data is for the computational part of the code, not the input/output part, and the scaling studies only ran for a few iterations, rather than the number that would be used in a production run. One can see that the initial scaling is quite linear, and we fit the computational speed (inverse wall-clock time as measured by MPITIME) versus number of cores to a linear curve with nearly zero intercept for the lowest few runs. This curve is then extrapolated to the larger number of cores where we also run the code. One can see that the linear scaling starts to tail off for 16,384 slave cores, but then recovers for 20,000 cores. This result occurs because the lattice was set up so that we would have two impurity problems to solve on each core for the 20,000 core case, but then the 16,384 case will have some cores solving two and some solving three impurity problems, hence there is a degradation of the scaling for that case.

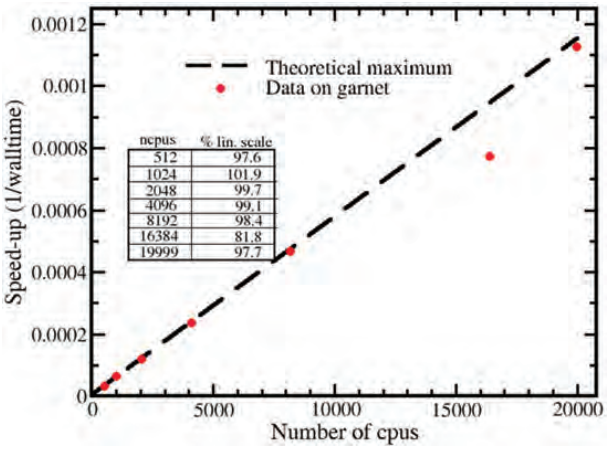


Figure 1. Strong scaling results for the algorithm on garnet (Cray XE6 with approximately 22,000 cores). We report the speed-up (inverse of the wall-time) for the different number of cores used in the scaling run. Note how there is a dip in the curve at 16,384 cores. This occurs because there are some cores with 3 impurity solvers required and some with 2. When we go to the commensurate limit of 2 solvers per core at 20,000 cores, we find the speed-up recovers, as expected. The inset is a table showing the percentage of linear scaling of the speed-up.

After completing the scaling analysis, we were selected for a CAP Phase II run on the Arctic Region Supercomputing Center (ARSC), machine chugach, which has about 11,000 cores. We ran our code on 8,192 slaves (plus one master) in a continuous fashion for approximately six weeks, generating a large portion of our data. We then performed a scaling analysis on the raptor machine at the US Air Force Research Laboratory (AFRL), which has about 44,000 cores. There we were able to run a scaling analysis with up to 43,500 cores, and we found that the system seems to show a super-linear scaling (see Figure 2)! Most likely this is just arising from some issues with run-to-run variations on the machine and to the possibility that our fit to a linear curve at small core counts is not perfectly accurate (although those points are quite linear, with a correlation coefficient of 0.99994). We compare the normalized speed-up curves on garnet and raptor in Figure 3. This needs to be done because the size of the problems we ran was different on the two machines. One can see that the behavior is quite similar on the two machines. Overall, we used about 10 million CPU-hours for the scaling runs and for the production runs on the machines during the CAP. We also use the same codes, running now on 4,097 cores for our Challenge Project.

In conclusion, we have shown excellent scaling behavior of this algorithm. This occurs primarily because the code has limited communications between nodes during a given iteration, and MPI can handle those communications very efficiently. In addition, the workload for the impurity solvers seems to divide well between the different cores as we increase the core count. We are currently working on understanding the differences in the workload and searching for means to balance the loads amongst the cores "on-the-fly" during the computation. It is likely that such an approach will require us to use a global addressing approach for the memory, which is available with either co-array FORTRAN or UPC.

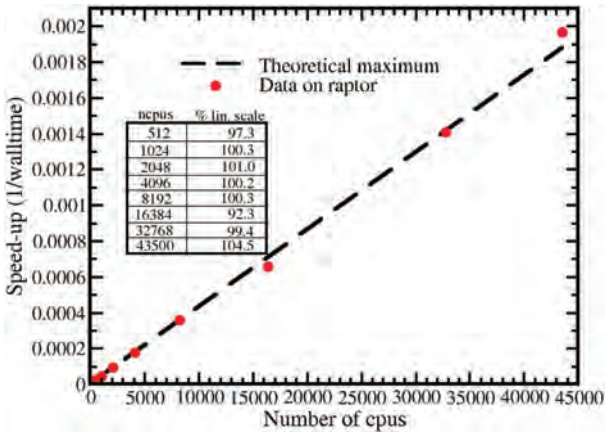


Figure 2. Same as Figure 1, except on the AFRL machine raptor, which has about 44,000 cores. Here, we surprisingly see super-linear scaling up the maximum size we could run on the machine.

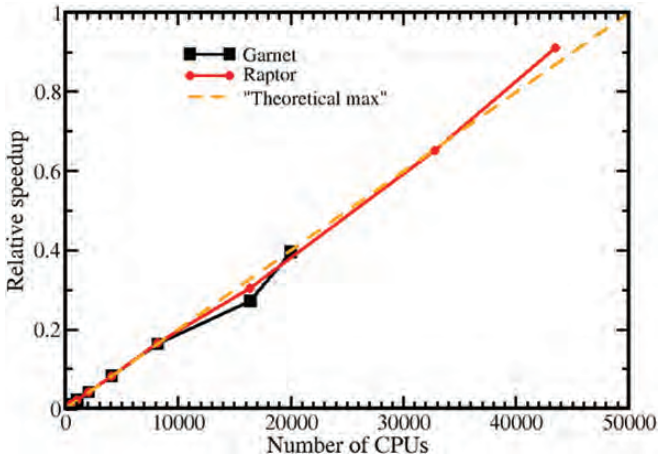


Figure 3. Comparison of the renormalized speed-ups on the two machines (we need to normalize to unit slope since the job sizes run on the different machines were different). One can immediately see the excellent scaling of the code which is showing no signs of stopping up to 43,500 cores.

3. Results

We begin by discussing the experimental data of the Swiss group (Jördens, 2010). They work with the fermionic isotope of K, ^{40}K, placed on an optical lattice. They cool their system down to an entropy per particle of about $1.3k_B$. After their experiment is finished, they drop the optical lattice potential and measure the entropy per particle again, and find it to be equal to $2.5k_B$. Hence, they bracket the entropy per particle of the system within the lattice to be between these two extremes. They use a clever method to detect the double-occupancy of the system. First, they perform an absorption image to count the total number of particles in the optical lattice. Next, they use a radio frequency pulse to transfer a K atom to a new hyperfine state if and only if two atoms sit at the same site, then drop the lattice and perform an absorption image to count the number of atoms in the new hyperfine state, which equals the number of double-occupancies. This gives them the total number of particles and the total number of double-occupancies. These two quantities allow one to see behavior that gives rise to the Mott transition.

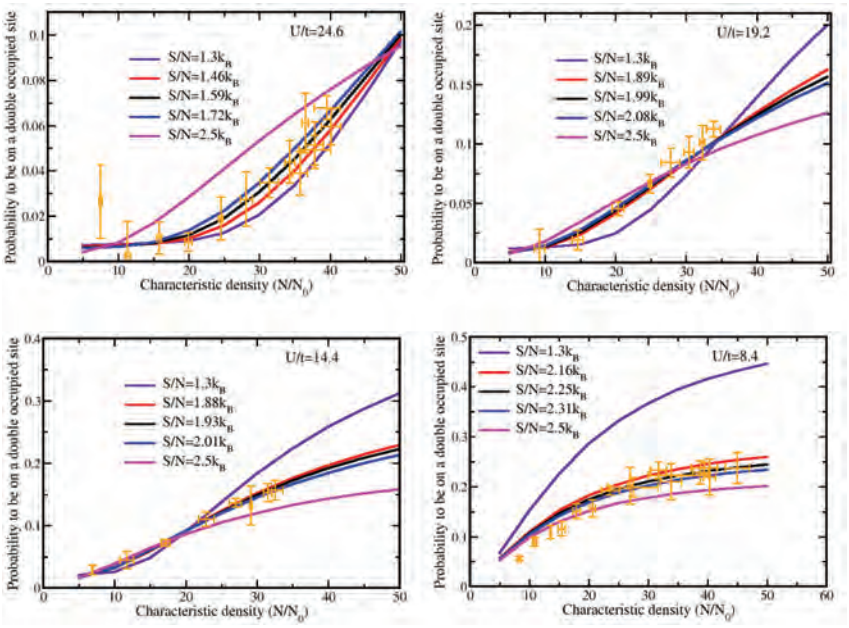


Figure 4. Fitting results for constant entropy curves of the double-occupancy versus particle number and the experimental data reported with error bars for the four different interaction strengths. Note how the fits are in general excellent, except for the lowest value of interaction strength (*U*/*t*=8.4), where the fit is poorer, especially at the lower densities, which also correspond to lower temperatures.

The way to see this is as follows: As we start filling fermions onto our lattice, we initially can fill them in the lowest available energy levels, which are typically spread throughout the lattice. As we keep filling in levels, the fermions begin to interact with each other as they become denser. If the repulsion term, U, is large enough, it will forbid two fermions of opposite spin from occupying the same lattice site at the same time (two fermions of the same spin are already forbidden due to the Pauli exclusion principle). Hence, the larger the interaction is, the lower the double-occupancy will be, until we have too many particles in our system that we have to increase the filling to be larger than one per site in the center of the trap. Then the double-occupancy will sharply increase as sites are filled with two fermions. This transition to the sharp increase in the double-occupancy will become smoother as the interaction is reduced. The ratio of the interaction to the hopping (U/t) is controlled by varying the depth of the lattice potential, which primarily controls the magnitude of the hopping t, and by tuning the interaction strength by using a Feshbach resonance to vary the two-particle scattering length, which changes U. Four values are considered in the experiments; i) U/t=24.6; ii) U/t=19.2; iii) U/t=14.4; and iv) U/t=8.4, where the lattice depth is fixed at 7 potassium recoil energies (t=174 Hz). The optical lattice is a simple cubic lattice in three dimensions with a trap that is harmonic, but with different curvatures along the three different axes, yielding constant energy shells which are ellipsoidal spheroids in shape (the trap frequencies are 49.4, 52.6, and 133 Hz).

Since the optical lattice is turned on slowly, it is believed to be a nearly adiabatic process, which implies that the entropy will be conserved. Of course it cannot be completely adiabatic; otherwise the final entropy per particle would be equal to the initial entropy. But, if we believe the adiabatic processes are governed by the speed that the lattice is turned on, then we expect the entropy of the fermions in the lattice to be the same for all cases with the same interaction strength, and hence this is the criterion we will use to fit the data. The data is plotted as the probability for an atom to be in a doubly-occupied state (two times the number of double-occupancies divided by the total number of particles) versus the total number of particles (plotted relative to the characteristic density N_0=7,393). In the experiment the number of particles varies from about 40,000 to 300,000, and the diameter of the cloud is always less than about 250 lattice sites.

The simplest way to solve this problem is to solve it in the high-temperature limit when U/t is large using what is called the strong-coupling expansion (Scarola, 2009). This produces a set of algebraic equations that need to be solved to determine the density of particles at each site as well as the double occupancy and the entropy per particle. This technique is only approximate, but is very efficient, requiring only a few hours on a single-processor machine for the total analysis. Our fitting procedure is slightly different than the one performed by the Swiss group, so we describe it in detail here. What we do is minimize the error for the double-occupancy versus the number of particles for a fixed-value of the entropy, taking into account the error bars of the double-occupancy in computing the goodness of fit. We compute the best-fit (black), the curves for the extremal values where the entropy per particle is fixed at the largest or smallest allowed values (purple and magenta), and then for the results where we shift the number of particles up or down (red and blue) by the amount of one standard deviation and recalculate the best-fit. Results are shown in Figure 4 for the four different values of the interaction strength. In general, the fits are excellent, but one can clearly see deviations for the lowest number of particles in the smallest interaction strength case U/t=8.4. Hence, we focus our efforts on trying to fix this discrepancy by performing quantum Monte Carlo simulation for this case.

We approximately solve the IDMFT problem using the QMC+GGA algorithm sketched above. Just like in experiment, we will determine the total number of particles, the total number of double-occupancies, and the entropy per particle. The former two results come straight out of the QMC solution for each lattice site, where the particle filling and the probability to be on a doubly-occupied site both are outputs of our impurity solver. The entropy per particle is more complicated to solve. We start at a high-temperature of about twenty-times the hopping, where the strong-coupling calculation will accurately determine the entropy for us. Then, we reduce the temperature, keeping the number of particles fixed. The entropy can then be calculated as an integral over the inverse of the temperature of the energy per particle, which can also be calculated directly by the QMC solver for each lattice site (Werner, 2005). Armed with this data, we can then recalculate the fit to the double-occupancy versus particle number plots at constant entropy. We have not yet completed this exercise, since we are still generating QMC data for some of the different experimental cases. But we can describe to what extent the IDMFT(QMC+GGA) approach differs from the strong-coupling approach, and also we can discuss the differences between the GGA and the LDA. We begin with the latter. We find that for nearly all cases we have examined that the LDA and the GGA agree very well with each other, which supports the notion that the LDA is very accurate when we are above the ordering transition temperature.

We also find that the strong-coupling approach is quite accurate until we start to get into the low-temperature regimes, where the IDMFT results clearly lie above the strong-coupling results for the entropy per particle. This is illustrated as a function of inverse temperature for two values of particle numbers: one with N=61,455 and one with N=202,780 in Figure 5. We see the deviation between the two entropies is more significant at the lower densities. In fact, we are probably reaching to temperatures where the IDMFT results themselves may no longer be so accurate, since they neglect the momentum

dependence of the self-energy which gives rise to quantum fluctuations which reduce the entropy per particle (the IDMFT cannot have the entropy reduced below ln 2 at half-filling until the system orders into an anti-ferromagnetic order, which we have not examined yet with our code; away from half-filling it can be reduced somewhat more). We expect that there is a range of temperature where the true entropy per particle lies in between the IDMFT and strong-coupling results.

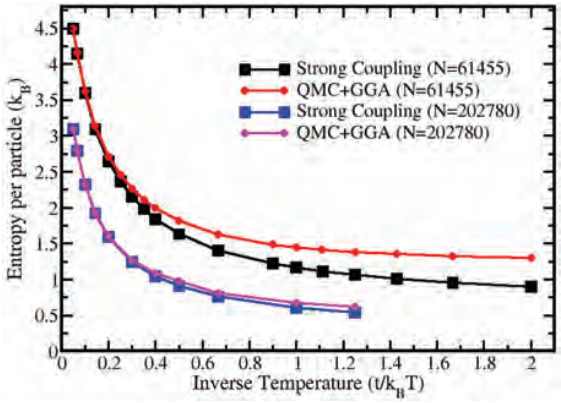


Figure 5. Entropy per particle versus inverse temperature for two different numbers of particles with *U*/*t*=8.4. Note how the entropy per particle deviates more from each other for the more dilute system. The deviations do become large enough to have a significant effect at the lower temperatures (toward the right of the plot).

We finally compare the radial particle density profiles to the radial double-occupancy profiles at two temperatures for the two different numbers of particles in Figure 6 (using the QMC_GGA algorithm). The low particle number case never achieves too high a density because the Fermi pressure makes the cloud size large enough that one never has any sites which have, on average two particles per site; hence the total number of double-occupancies is small. The higher particle number has an appreciable core region with two particles per site; hence the double-occupancy is sharply increased. The change in curvature of the density profile near unit density occurs due to the initial signatures of Mott-like physics entering at the lower temperatures (the interaction strength is too low for the system to be a true Mott insulator). The high-temperature results do not appear to have much, if any quantum degeneracy effects, while the lower temperature results certainly do.

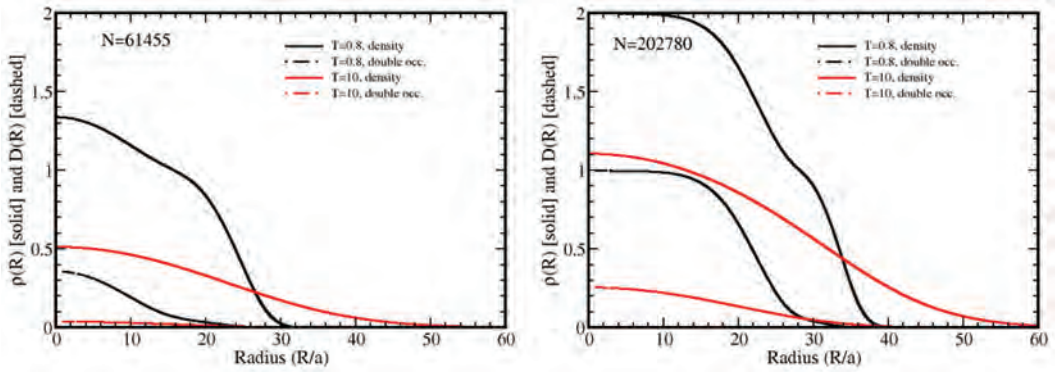


Figure 6. Radial density (solid) and double-occupancy (dashed) profiles for high (k_BT=10t, red) and low (k_BT=0.8t, black) temperatures for the two different cases of particle numbers shown in the previous figure (*U*/*t*=8.4). The change in curvature of the radial density near a filling of one at low-temperature arises from the beginning signals of the formation of a Mott-insulating state. These signatures only begin to be seen at the lower temperatures.

4. Significance to DoD

DARPA's interest in this problem is to ultimately build a materials science emulator out of ultra-cold atomic atoms. One can then hypothesize a particular material, program the emulator to simulate its properties, see if these properties are an improvement over currently-known materials, and then devise a way to make the new materials with these targeted properties. We are still far away from this goal, but have made much progress with being able to make simpler quantum many-body problem emulators in a variety of different platforms and for a variety of different models. These simpler emulators can still be benchmarked with conventional high performance computing, which we do here.

5. Conclusion

In this work, we have shown how one can go beyond the LDA to include gradient corrections. We find at high-temperature these gradient corrections are quite small. As the temperature is lowered, they become larger, and certainly can have some effect on the entropy per particle at moderate temperatures (the difference between the DMFT entropy and the strong-coupling entropy is much more significant). We expect both the LDA and GGA approximations to be less accurate once the system begins to display anti-ferromagnetic order, because the transition temperature for both approximate methods is closely-tied to the transition temperature in the bulk, while a full IDMFT solution would probably have a suppressed transition temperature due to inhomogeneity effects and proximity effects that are not present in the current generation of codes. We are currently working on an efficient implementation of the full IDMFT algorithm.

Acknowledgements

JKF acknowledges support from the Army Research Office Grant Number W911NF0710576, with funds from the DARPA OLE program (for the computations that are used to benchmark the Swiss experiment), and from the National Science Foundation under grant number OCI-0904597 (for development of the MPI algorithm). KM was supported by the AFOSR under the MURI program from grant number FA9559-09-1-0617. HRK acknowledges support from the DST (India). The collaboration between the US and India was supported by grant number JC-18-2009 of the Indo-US Science and Technology Forum. DoD HPC computer time was provided on Cray XE6 machines located at the Arctic Region Supercomputer Center (ARSC), the US Air Force Engineering and Research and Development Center (AFRL) and the US Army Engineering Research and Development Center (ERDC). This project was supported primarily by Challenge Project DARPA-C4J and a CAP in fiscal year 2010–2011. We thank the Esslinger group for providing us with their experimental data and for conversations with Niels Strohmaier and Leticia Tarruell that helped clarify the data.

References

Abo-Shaeer, J., website for the optical lattice emulator program, http://www.darpa.mil/dso/thrusts/physci/funphys/ole/index.htm, 2010.

Carrier, P., J.M. Tang, Y. Saad, and J.K. Freericks, "Lanczos-based low-rank correction method for solving the Dyson equation in inhomogeneous dynamical mean-field theory", *Physics Procedia*, 2011 (submitted); arxiv:1102.5738.

Feynman, R.P., "Simulating physics with computers", *International Journal of Theoretical Physics.*, 21, pp. 467–488, 1982.

Freericks, J.K., H.R. Krishnamurthy, P. Carrier, and Y. Saad, "Efficiently generalizing ultra-cold atomic simulations via inhomogeneous dynamical mean-field theory from two- to three-dimensions", *Proceedings of the HPCMP Users Group Conference 2010*, Shaumburg, IL, edited by D.E. Post, IEEE Computer Society, Los Alamitos, CA, 2010.

Gull, E., A.J. Millis, A.I. Lichtenstein, A.N. Rubtsov, M. Troyer, and P. Werner, "Continuous-time Monte Carlo methods for quantum impurity problems", *Rev. Mod. Phys.*, 83, pp. 349–404, 2011.

Hubbard, J., "Electron correlations in narrow energy bands", *Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences*, 276, pp. 238–257, 1963.

Jarrell, M., A. Macridin, K. Mikelsons, D.G.S.P. Doluweera, and J.E. Gubernatis, "The dynamical cluster approximation with quantum Monte Carlo cluster solvers", Lectures on the physics of strongly-correlated systems XII, *American Institute of Physics Conference Proceedings*, 1014, pp. 34–106, 2008.

Jördens, R,. N. Strohmaier, K. Günter, H. Moritz, and T. Esslinger, "A Mott insulator of fermionic atoms in an optical lattice", *Nature*, 455, pp. 204–207, 2008.

Jördens, R., L. Taruell, D. Greif, T. Uehlinger, N. Strohmaier, H. Moritz, T. Esslinger, L. De Leo, C. Kollath, A. Georges, V. Scarola, L. Pollet, E. Burovski, E. Kozik, and M. Troyer, "Quantitative determination of temperature in the approach to magnetic order of ultra-cold fermions in an optical lattice", *Physical Review Letters*, 104, p. 180401, 2010.

Rombouts, S.M.A., K. Heyde, and N. Jachowicz, "Quantum Monte Carlo method for fermions, free of discretization errors", *Physical Review Letters*, 82, pp. 4155–4159, 1999.

Rubtsov, A.N., V.V. Savkin, and A.I. Lichtenstein, "Continuous-time quantum Monte Carlo method for fermions", *Physical Review B*, 72, p. 035122, 2005.

Scarola, V.W., L. Pollet, J. Oitmaa, and M. Troyer, "Discerning incompressible and compressible phases of cold atoms in optical lattices", Physical Review Letters, 102, pp. 135302–1-4, 2009; Erratum, *Physical Review Letters*, 103, p. 189901(E), 2009.

Werner, F., O. Parcollet, A. Georges, and S. Hassan, "Interaction-induced adiabatic cooling and anti-ferromagnetism of cold atoms in optical lattices", *Physical Review Letters*, 95, p. 056401, 2005.

HPCMP UGC 2011

4. Climate/Weather/Ocean Modeling and Simulation (CWO)

Large-Scale Deterministic Predictions of Nonlinear Ocean Wave-Fields

Wenting Xiao, Yuming Liu, and Dick K.P. Yue
Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA
{wtxiao, yuming, yue}@mit.edu

Abstract

*The reliable description of spatial-temporal evolution of ocean surface waves in deep sea and littoral zones is essentially important to the new Navy ship design process and optimal naval operations. For a physics-based and phase-resolved prediction of ocean wave-field evolution, a direct simulation tool called SNOW (**s**imulation of **n**onlinear **o**cean **w**ave-fields) has been developed. SNOW is developed using a highly-efficient pseudo-spectral approach. It solves the primitive Euler equations, follows the time evolution of a large number (N) of wave modes, and accounts for their nonlinear interactions up to an arbitrary high-order (M). SNOW obtains exponential convergence and requires near-linear computational effort with respect to N and M. Moreover, SNOW achieves almost linear scalability on modern HPC platforms. The accuracy and reliability of SNOW in describing nonlinear ocean wave statistics is demonstrated through a quantitative comparison between model predictions and experimental observations. Routine large-scale SNOW simulations are performed to study the statistical properties of nonlinear directional ocean wave-fields. We confirm that linear theory significantly under-predicts the probability of large-wave events, especially for sea states with narrow spectra bandwidth and narrow directional spreading angle. In addition, the effect of finite-depth on the ocean wave-field evolution is examined. We find that in general, the non-Gaussian feature as well as the occurrence of large-wave events decreases as water depth reduces. Moreover, significant long-waves are generated by near-resonant wave-wave interactions in the evolution of nonlinear wave-fields in the near-shore area. The capability of obtaining detailed spatial-temporal description of the ocean wave-field will greatly increase the operational envelops and survivability of existing naval ships.*

1. Introduction

The long-term objective of this project is to apply large-scale phase-resolved nonlinear wave simulations with physics-based modeling of complex physical processes in upper-ocean surface boundary-layers to establish a new generation of tools for ocean wave prediction. In this year, with the use of large-scale phase-resolved wave-field simulations, we focused on the understanding of the nonlinear statistical characteristics of directional wave-fields and the dependence of wave statistics on physical processes involved in oceanic dynamics. In addition, for applications of Navy operation in littoral zones, the wave statistics and spectral evolution of nonlinear wave-fields in intermediate water depth are investigated.

The accurate prediction of nonlinear ocean surface wave-fields is of importance to the new Navy ship design process and optimal naval operations in deep-water and littoral zones. The task is challenging due to the complexity involved in the systems including nonlinear wave-wave interactions, wave-breaking and wind-forcing, and also the effects of variable currents and bottom-bathymetry. Until recently, phase-averaged models such as WAM and SWAN (for near-shore regions) are the only tools of practical predictions. These models are developed based on the phase-averaged energy-balance equation with the physical effects associated with nonlinear resonant wave-wave interactions, wind-forcing, and wave-breaking/bottom dissipations modeled as source terms. There has been much progress in global and regional wave predictions and satisfactory comparisons to field and laboratory measurements have been obtained in some cases (Komen, 1994). However, due to necessary simplifications and inherent assumptions, the predictions often fall outside of the error band of the observations (Komen, 1994). More importantly, these phase-averaged models provide predictions only of the statistical properties of ocean waves, but not detailed description of space-time variations of ocean surface which is essential for improving human safety and operation performance in severe seas by using (active) optimal control. With recent development of computational capabilities and fast numerical algorithms for phase-resolved simulations of nonlinear wave dynamics, the time and opportunity have arrived for the application of direct phase-resolved simulations for nonlinear ocean wave predictions. Based on an efficient high-order spectral method (HOS), we have established a powerful computational

capability for phase-resolved prediction of large-scale nonlinear ocean wave-field evolutions (Tsai & Yue, 1996). We call this direct-simulation-based wave prediction capability as SNOW (simulation of nonlinear ocean wave-fields). Unlike the phase-averaged model, SNOW obtains the detailed phase information of the wave-field during its nonlinear evolution, which enables the modeling of complex physics such as wave-breaking and wind- forcing in a more physics-based manner.

A proper description of the ocean wave statistics has always been essential for ship design and Navy operations. In practice, Tayfun's second-order theory (Tayfun, 1980) is generally used to estimate the wave height or wave crest distribution. Tayfun's formulation is based on the assumption that all free waves follow a Gaussian distribution. This theory considers only self-interactions of free waves, but neglects nonlinear interactions among different free waves. Tayfun's theory generally compares well with the field observations in mild seas. In severe seas with narrow-band wave spectra, the effect of modulational instability associated with higher-order wave-wave interactions (that are not considered by Tayfun's theory) can cause an exponential growth of the central wave mode (Janssen, 2003). In the study of the effect of modulational instability on ocean wave statistics, nonlinear wave-field evolution in a large domain over a long time is needed in order to obtain statistically important information. This requires the use of very large numbers of wave modes and time-steps in direct phase-resolved simulations that were limited by available computing resources in the past. Most of the studies are thus based on the wave envelope models developed with nonlinear Schrödinger equation (NLS) and modified nonlinear Schrödinger equation (MNLS) (Dysthe, 1976) assuming the wave spectra is narrow-banded and the modulation of the wave profile is slowly varying with time and space. Although considerable progress has been made in the past, the reliability and accuracy of these models has not been carefully justified. In particular, the application of NLS-type model to realistic ocean wave-fields is questionable because the inherent assumption of narrow bandwidth wave spectrum is often invalid. In this study, large-scale direct phase-resolved SNOW simulations of nonlinear wave-field evolutions are used to study the effect of modulation instability on nonlinear ocean wave statistics and to assess the accuracy and reliability of the existing models prediction.

For the application in littoral zones, the understanding and prediction of near-shore wave environment is of fundamental importance. For example, the presence of significant long waves in littoral regions can affect the dynamic stability of surface ships and is a major concern in the design of mooring system for floating marine facilities such as liquid natural gas (LNG) tanks. Moreover, an appropriate description of near-shore wave statistics is a necessary input in the motion analysis of surface vessels and floating marine structures. Despite its importance, the understanding of nonlinear near-shore wave environment is limited, and the tool capable of properly predicting its nonlinear evolution is still lacking. In this work, large-scale phase-resolved SNOW simulations are performed to investigate the dynamics and statistical characteristics of nonlinear ocean surface waves in littoral zones.

2. Numerical Methodology

The problem of nonlinear ocean surface wave-field evolution is considered under the assumption of potential flow. SNOW solves the primitive field equations with nonlinear kinematic and dynamic free-surface boundary conditions in the Zakharov form (Zakharov, 1968). SNOW solves these equations based on the use of a high-order spectral method (Dommermuth & Yue, 1987) that is capable of following nonlinear evolution of a large number of wave modes in the wave-field with a minimum computation requirement. The method includes nonlinear interactions of all wave components up to an arbitrary order (M) in wave steepness. The nonlinear wave interactions with variable currents and bottom-bathymetry are also considered in a direct way while the wave-breaking dissipation and wind input are accounted for using physics-based modeling.

With SNOW, the computational effort is almost linearly proportional to M and the large number of wave modes (N). Exponential convergence of the solution with M and N is also obtained. Such high-efficiency and accuracy makes SNOW an ideal approach for the computation of large-scale nonlinear ocean wave-field evolution.

At the initial stage of SNOW simulations of nonlinear wave-field evolution, the free-surface boundary conditions are smoothly transitioned in time from linear to nonlinear for minimizing any standing wave effect that results from using linear initial conditions (Dommermuth, 2000). In a typical SNOW simulation, this is done over five periods of the dominant wave mode.

Parallel implementation of SNOW is achieved by decomposing the spectral domain across processors using Message Passing Interface (MPI). SNOW utilizes transposes and on-processor fast-Fourier transforms to move between the physical and spectral domains. SNOW achieves almost linear scalability for a large number of processors O (2000), making it highly-scalable. Figure 1 shows the typical performance of SNOW as a function of processor number and problem size on the Cray XT3, Cray XT4, and Cray XT5 at the US Army Engineer Research and Development Center (ERDC). The

performance is measured as the number of time-steps of simulations achieved in five wall-clock minutes. In general, the near-linear scalability is achieved on both systems.

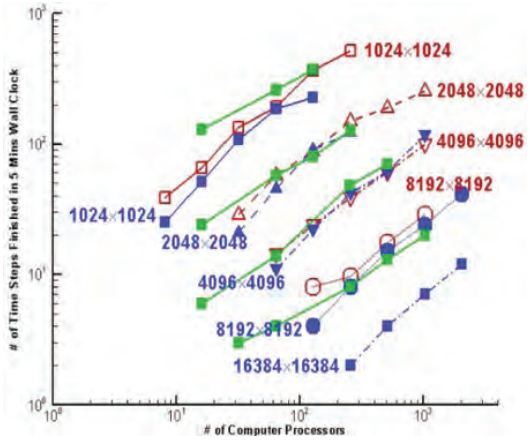


Figure 1. Scalability of typical SNOW simulations on Cray XT3 (blue), Cray XT4 (black), and Cray XT5 (green). Plotted is the number of time-steps of computations completed within 5 minutes of wall-clock time versus the number of processors used in the simulation for different problem scales ($N_x \times N_y$).

3. Results and Discussion

3.1 Variation of Skewness and Kurtosis

In this part, the variation of skewness and kurtosis of ocean surface elevation with evolution time is investigated. The predictions, obtained from the phase-resolved SNOW computation and MNLS model, are compared with available experimental observations.

The initial wave-field is specified from a JONSWAP frequency spectrum with random phase of each wave component uniformly distributed in $[0, 2\pi]$. A COS-square directional spreading function is used to describe the energy distribution over different directions. The key spectrum parameters are the peak enhancement coefficient γ, angular spreading angle Θ, Phillips parameter α, and peak wave-number k_p. One notes that γ describes the bandwidth of the spectrum with a smaller value of γ corresponding to a wider bandwidth; θ defines the crest length, and α is associated with the significant wave height of the wave-field.

Skewness of the surface elevation η is defined as $\langle\eta^3\rangle / \langle\eta^2\rangle^{3/2}$ and it measures the asymmetry of the free- surface elevation. The skewness of a Gaussian random variable is zero. If the second-order bound waves are considered, the value of skewness can be approximated by $3k_p\sigma$ for a narrow-banded wave-field (Longuet-Higgins, 1963), where σ is the standard deviation of the wave-field elevation. The results of skewness for the wave-fields with H_s=0.08 m, T_p=1 s, γ=6, and three different spreading angles θ=15, 31, and 72 degrees are shown in Figure 2. The predictions from SNOW simulations and by MNLS model are compared with the wave basin measurements of Onorato (2009). The results by SNOW and MNLS are obtained by the average over five realizations. To compare with the wave probe records in the experiment, the temporal variation of skewness obtained from SNOW and MNLS simulations is converted into spatial variation using $x=c_g t$ with c_g being the linear group velocity of the dominant wave component. The non-Gaussian behavior is observed for all three cases and the deviation is more significant for the case with smallest spreading angle (θ=15°). Compared to the experimental data, both SNOW and MNLS give a reasonably good prediction on the skewness. The second-order estimation for this case is 0.24 that is very close to both of the model predictions. This suggests that the departure from the linear Gaussian value of skewness is contributed mainly by the bound waves. It is observed that the skewness calculated from SNOW simulations quickly deviates from the Gaussian value due to nonlinear bound waves and stays around the same value through the evolution. This feature again suggests that the departure from the linear Gaussian value is contributed mainly by the bound waves, and is not sensitive to higher order nonlinearity that becomes important at longer time scale. This is because in the time scale considered, the dominant free-wave dynamics is third-order in elevation and their effect cancels for skewness.

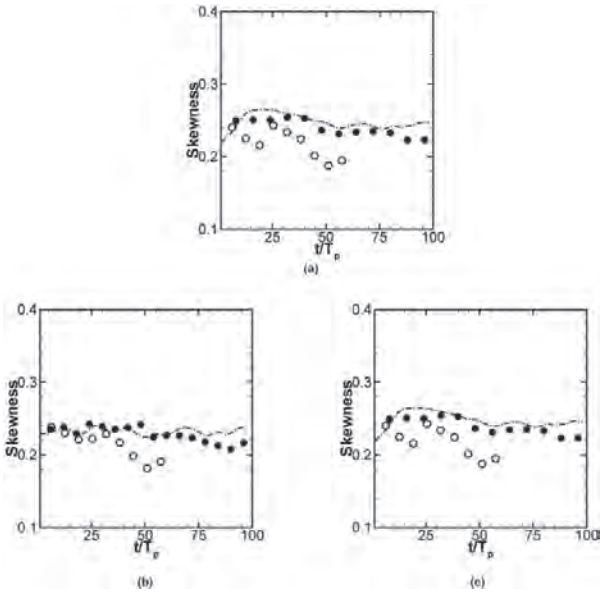


Figure 2. Variation of skewness over evolution time for wave-fields with H_s=0.08 m, T_p=1 s, γ=6, and (a) θ=15°; (b) θ=31°; and (c) θ=72°. The plotted are the results of experiment (○), SNOW (●), and MNLS (dash-dot line).

Kurtosis of surface elevation, defined as $\langle\eta^4\rangle/\langle\eta^2\rangle^2$, denotes the peakness of one distribution. The kurtosis for a Gaussian random variable is 3. The distribution of a random variable having a kurtosis value greater than 3 suggests a heavier tail than the normal distribution. The second-order estimation for kurtosis is $kurtosis=3+24(k_p\sigma)^2$. Figure 3 shows kurtosis variation for the same wave-fields described above: H_s=0.08 m, T_p=1 sec and γ=6 with spreading angles θ=5, 31, and 72 degrees. (The results by SNOW and MNLS are obtained by the average over five realizations). For all three cases, non-Gaussian values of kurtosis are observed. Both of the numerical models give a prediction of kurtosis much larger than the second-order theoretical estimation, which is 3.02, except for the case with largest spreading angle θ=72°. It is observed that as wave-fields evolve, the values of kurtosis deviate from the Gaussian value, and the discrepancy is more significant for the wave-field with small spreading angle θ=15° and less significant for the wave-field with large spreading angle θ=72°. For all the cases, SNOW simulations agree better with experiments compared with MNLS predictions.

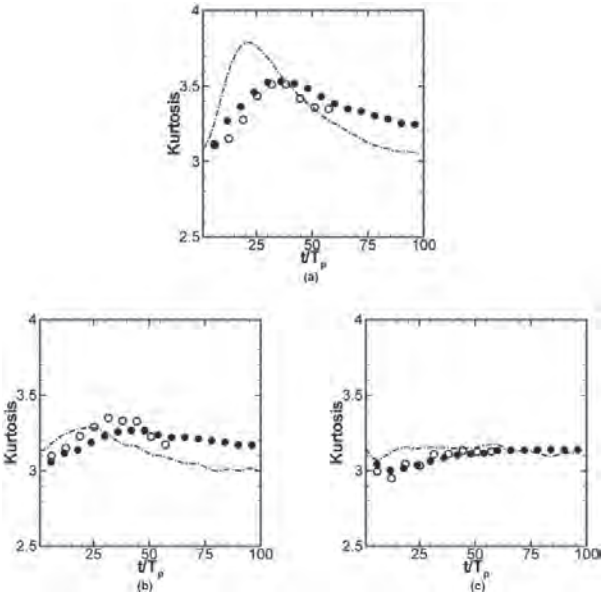


Figure 3. Variation of kurtosis over evolution time for wave-fields with H_s=0.08 m, T_p=1 sec, γ=6, and (a) θ=15°; (b) θ=31°; and (c) θ=72°. The plotted are the results of experiment (○), SNOW (●), and MNLS (dash-dot line).

3.2 Exceeding Probability of Wave Crest

The spatial variation of exceeding probability of wave crest, defined as local maximum point of free-surface elevation, for the case with spreading angle θ=15° is presented in Figure 4. The crest distribution calculated as the second-order upper-surface envelope from the MNLS model is also presented for the purpose of comparison. The linear Rayleigh theory and second-order theory derived under the narrow bandwidth assumption (Tayfun, 1980) is plotted as references. To compare with the wave probe records, again the conversion $x=c_g t$, with c_g being the linear group velocity of dominant wave component, is applied. It is shown in Figure 4, that the agreement between experimental observation and SNOW simulation is very good up to x/λ_p=28.7, while MNLS with second-order reconstruction gives good estimation up to x/λ_p=15.9 and underestimates the large-crest probability at locations far away from the wave-maker. The classic Rayleigh distribution significantly underestimates the crest distribution for this wave-field. The second-order theory gives a fairly good estimation on the crest distribution at early stage of the wave-field evolution (x/λ_p=3.1). However, the deviation of the second-order theory from the experimental data is seen to be significant at x/λ_p=15.9 when the higher-order nonlinear wave dynamics becomes important.

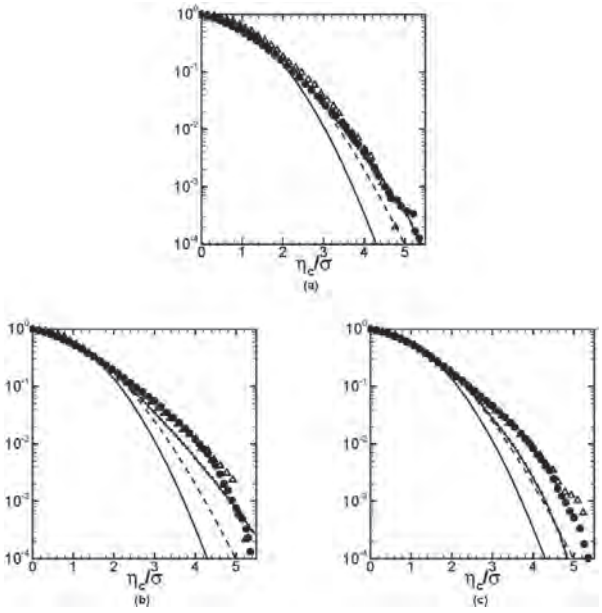


Figure 4. Exceeding probability of the wave crest for directional sea with spreading angle θ=15° at locations (a) x/λ_p=3.1, (b) x/λ_p=15.9, and (c) x/λ_p=28.7. The plotted are the results of experiments (Δ), SNOW (●), MNLS (solid line with dots), Rayleigh distribution (solid line), and the second-order theory (dash-line).

3.3 Effect of Finite Water Depth on the Wave Statistic

In this section, we apply large-scale direct simulations of nonlinear wave-field evolution to investigate the statistical characteristics of ocean waves in littoral regions. The objective is to provide realistic wave environment for the design and motion analysis of ships and marine structures. The focus of this study is on the understanding of the water depth effect upon nonlinear wave spectrum evolution, nonlinear wave statistics, and occurrence and characteristics of extreme waves. For all these simulations, the initial wave- fields are chosen to have the same significant wave height H_s=10*m*, peak period T_p=12*s*, and γ=3.3. We introduce a normalized water depth parameter $\mu \equiv k_p h$ where h is the water depth and k_p is the peak wave number of the wave spectrum. In this paper, we will limit our study to intermediate water depth, for which the so-called Ursell number (defined as $Ur=H_s k_p/2(k_p h^3)$) is small.

Figure 5 shows the time variation of kurtosis in nonlinear evolution of the wave fields at different water depths for two spreading angles θ=20° and 80°. For both spreading angles, the kurtosis decreases as water depth reduces. This is somewhat as expected since the kurtosis is largely influenced by the modulational instability that vanishes at shallow depth μ≤1.363. The reduction of kurtosis is much stronger for the wave-field with relatively smaller spreading angle. For the case of θ=20° and μ=1.07 (h=30 m), in particular, the kurtosis even becomes less than 3.0, the value for Gaussian random variables. This result is consistent with the predictions based on the Zakharov equation by Janssen & Onorato (2007). Figure 6 compares the maximum value of the kurtosis for different water depths and spreading angles. The maximum kurtosis shows a strong dependence on water depth and spreading angle.

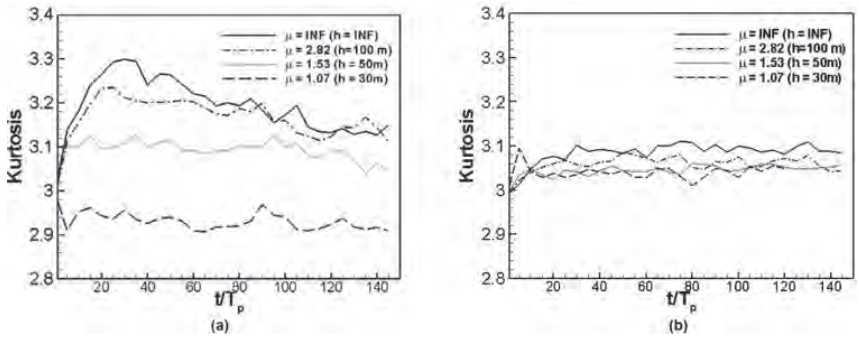


Figure 5. Time variation of the kurtosis in nonlinear wave-field evolution at various water depths: (a) θ=20°; (b) θ=80°

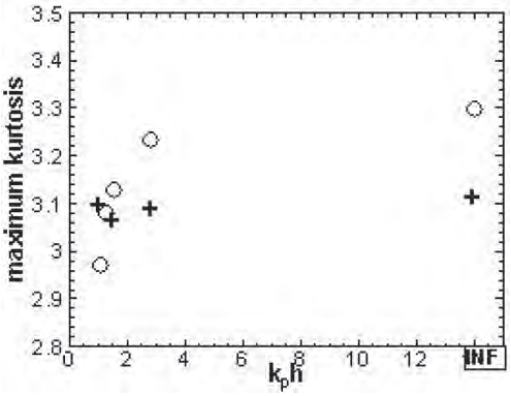


Figure 6. Maximum kurtosis as a function of water depth for the wave-fields with θ=20° (○) and θ=80° (+)

3.4 Effect of Water Depth on Spectrum Evolution

The omni-spectrum E(|k|) is defined as

$$E\left(|k|\right)=\int_{|k|} S\left(\vec{k}\right)d\vec{k}=\int_{-\pi}^{\pi} S\left(|k|,\theta\right)|k|\,d\theta \tag{1}$$

Figure 7 compares the omni-spectra of nonlinear wave-fields for θ=20° with different water depths at evolution time t/T_p=0, 50, and 100. At deep water, the peak of the spectrum apparently shifts to a lower wave number (or frequency) in the evolution. The downshift of the spectrum peak becomes weaker as water depth decreases. In the high wave number region, the spectrum shows a loss of wave energy, which becomes stronger for shallower depth. This is associated with the dissipation effect of wave breaking. As water depth decreases, SNOW results show the presence of a significant energy transfer to the low wave number region, leading to the generation of very long waves in near-shore areas. This is a result of near- resonant wave-wave interactions, which are stronger with shallower depth. Proper prediction of such long waves is of particular importance in the design of moorings and optimal operation of floating marine facilities in near-shore areas.

To better understand the transfer of energy to long waves by near resonant wave-wave interactions, we compare the wave spectra in the long-wave region obtained at t/T_p=50 with θ=20° and θ=80°at different water depths. The comparisons are shown in Figure 8. In general, the comparisons indicate that stronger energy transfer to long-waves is a resulted at shallower depth and with smaller spreading angle θ in accordance with the fact that in the limit of shallow water, waves are non-dispersive and second-order triad interactions become resonant.

3.5 Effect of Finite Depth on Crest Distribution

For nonlinear wave-fields with different water depths, we compare the exceeding probabilities of crest at t=30T_p for two spreading angles θ =20° and θ =80°. The results are shown in Figure 9. It is seen that the deviations of the SNOW results from the Rayleigh and the second-order Tayfun solutions are the most significant for the case with small spreading and deepest water depth. As water depth decreases, the exceeding probability of crest (obtained by SNOW) tends to match better with the second-order Tayfun solution. This suggests that the occurrence probability of extreme waves is somewhat

reduced by the finite water depth effect. For the wave-field with a wider spreading angle, it is seen that the second-order solution gives a quite satisfactory prediction of the exceeding probability of wave crest.

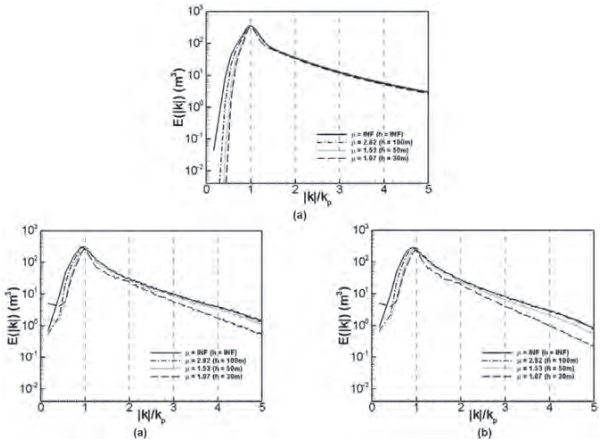


Figure 7. Evolution of the omni-spectra of nonlinear wave-fields with θ=20° at different water depths. The plotted are the result at the evolution time: (a) t=0T_p, (b) t=50T_p, and (c) t=100T_p.

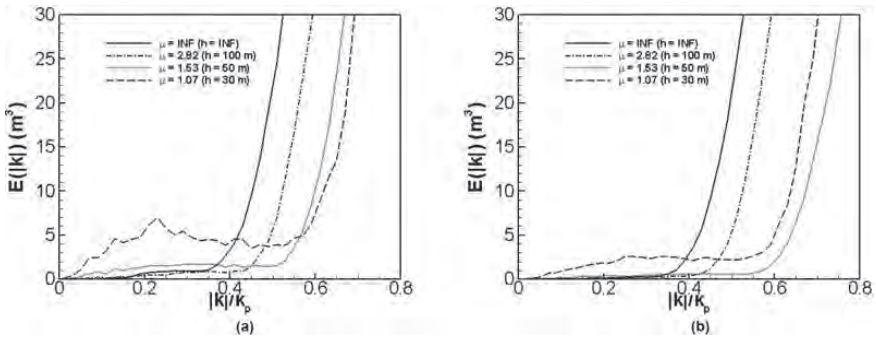


Figure 8. Wave spectra in the long-wave region at t=50Tp in different water depths. The plotted are the results for the wave-fields with spreading angle: (a) θ=20° and (b) θ=80°.

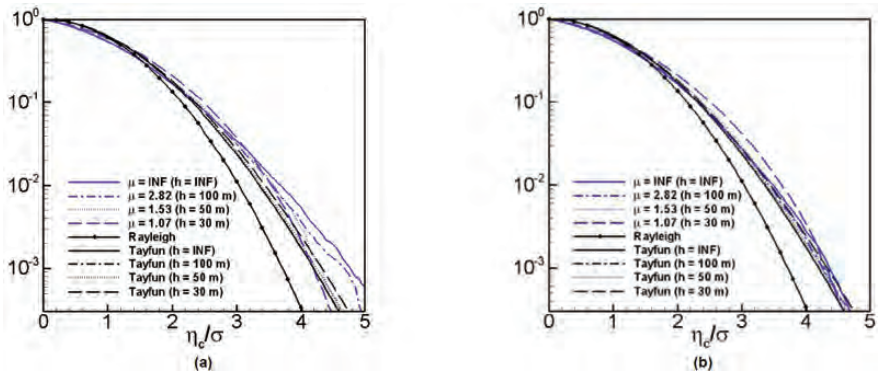


Figure 9. Exceeding probability distribution of wave crest at evolution time t=30T_p. The plotted are the results for the wave-fields with spreading angle: (a) θ=20° and (b) θ=80°.

4. Conclusion

A direct phase-resolved wave prediction tool, SNOW, is applied to simulate the nonlinear evolution of three-dimensional ocean-surface wave-fields under a variety of sea conditions. The ultimate objective is to establish SNOW as a new-generation ocean wave prediction tool in the near future. Based on the large-scale SNOW-simulated nonlinear wave-fields, the statistical properties such as skewness, kurtosis and exceeding probability of surface elevation can be computed.

In this work, we demonstrate the accuracy and reliability of SNOW in describing nonlinear wave-field statistics through a quantitative comparison with experimental observations. In addition, the prediction of wave statistics by the NLS-type models is also compared and discussed. It is shown that for wave-fields with small spreading angle (near long-crested), SNOW gives more accurate results while NLS-type model is accurate only up to a short time. For wave-fields with large spreading angle, SNOW and NLS-type models give similar good predictions on nonlinear ocean wave statistics.

In addition, SNOW is applied to study the characteristics of near-shore wave dynamics. The focus is on the effect of finite water depth upon nonlinear wave statistics and wave spectrum evolution. It is observed that the kurtosis of surface elevation decreases as water becomes shallower. Compared to deep water, the occurrence probability of rogue-waves is reduced in the littoral regions. Moreover, we find that significant long-waves are generated by near-resonant wave-wave interactions in the evolution of nonlinear wave-fields in the near-shore area.

5. Significance to DoD

Phase-resolved prediction of nonlinear ocean wave-fields by SNOW significantly increases the operational safety of naval ships and is of importance to the design of new Navy advanced ships. It will also provide a powerful framework for the assessment and improvement of phase-averaged wave-prediction models and greatly enhance the capability of ocean wave sensing deployment and data interpretation.

Acknowledgements

We gratefully acknowledge the support of the DoD High Performance Computing Modernization Program and Office of Naval Research (N000140810610 managed by Dr. Charles Linwood Vincent) for this study.

References

Dysthe, K., "Note on a modification to the nonlinear Schrödinger equation for application to deep water waves", *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 369(1736), pp. 105–114, 1979.

Dommermuth, D.G. and D.K.P. Yue, "A high-order spectral method for the study of nonlinear gravity waves", *J. Fluid Mech.*, 184, pp. 267–288, 1987.

Dommermuth, D.G., "The initialization of nonlinear waves using an adjustment scheme", *Wave Motion*, 32, pp. 307–317, 2000.

Janssen, P.A.E.M. and M. Onorato, "The intermediate water depth limit of the Zakharov equation and consequences for wave prediction", *J. Phys. Oceanogr.*, 37(10), pp. 2389–2400, 2007.

Janssen, P.A.E.M., "Nonlinear Four-Wave Interactions and Freak Waves", *J. Phys. Oceanogr.*, 33(4), pp. 863–884, 2003.

Komen, G.J., L. Cavaleri, M. Donelan, K. Hasselmann, S. Hasselmann, and P.A.E.M. Janssen, *Dynamics and Modeling of Ocean Waves*, Cambridge University Press, 1994.

Liu, Y. and D.K.P. Yue, "Large-scale phase-resolved simulations of ocean surface waves", *Oceanography in 2025: Proceedings of a Workshop*, 2009.

Longuet-Higgins, M.S., "The effect of non-linearities on statistical distributions in the theory of sea waves", *J. Fluid Mech.*, 17(03), pp. 459–480, 1963.

Onorato, M., L. Cavaleri, S. Fouques, O. Gramstad, P. Janssen, J. Monbaliu, A.R. Osborne, C. Pakozdi, M. Serio, and C.T. Stansberg, "Statistical properties of mechanically-generated surface gravity waves: a laboratory experiment in a 3D wave basin", *J. Fluid Mech.*, 627, pp. 235–257, 2009.

Tayfun, M.A., "Narrow-band nonlinear sea waves", *J. Geophys. Res.*, 85(C3), pp. 1548–1552, 1980.

Tsai, W. and D.K.P. Yue, "Computations of nonlinear free-surface flows", *Ann. Review of Fluid Mech.*, 28, pp, 249–78, 1996.

Yue, D.K.P., "Nonlinear Wave Environments for Ship Motion Analysis", *Proc. 27th Symp. on Naval Hydrodynamics*, Seoul, Korea, 2008.

Towards a Next-Generation Tropical Cyclone Track and Intensity Capability for the Navy

James Doyle, Carolyn Reynolds, Sue Chen, Yi Jin, Hao Jin, Chi-Sann Liou, Justin McLay, Jon Moskaitis, and James Ridout
US Naval Research Laboratory (NRL-MRY), Monterey, CA
{james.doyle, carolyn.reynolds, yi.jin, hao.jin, chi-sann.liou, justin.mclay, jon.moskaitis, james.ridout}@nrlmry.navy.mil

Richard Hodur
Science Applications International Corporation (SAIC), Monterey, CA
richard.hodur@nrlmry.navy.mil

Abstract

The capability to predict tropical cyclone track, intensity, and structure has been developed and extensively tested using the Navy's regional and global modeling systems. A tropical cyclone capability has been developed for the Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS-TC) and tested in real-time for multiple regions of the globe. The results show that COAMPS-TC was the most skillful dynamical model for tropical cyclone intensity forecasts in the W. Atlantic basin for the 2010 season, particularly for the critical 36-60 h forecast period. The model has superior intensity skill relative to the current Navy's operational regional tropical cyclone model, and comparable tropical cyclone track skill. For the global atmospheric ensemble prediction system, the impact of parameter variations on the Navy Operational Global Atmospheric Prediction System ensemble performance is examined. In the tropics, the parameter variations significantly increase ensemble spread, and significantly improved probabilistic prediction of low-level wind-speed. There are also small but significant improvements in the ensemble mean tropical cyclone track forecasts. Work is continuing on improving ensemble performance through inclusion of model uncertainty in ensemble design, and it is anticipated that these improvements will be transitioned to operations within the coming year.

1. Introduction

Tropical cyclones (general term that includes hurricanes in the Atlantic basin and typhoons in the Western Pacific) are among the deadliest and most costly natural hazards, and have the potential to substantially impact civilian and military lives and property, because they are often accompanied by severe winds, torrential rainfall, and flooding. On average, hurricanes and tropical storms account for over $100 billion of damage in the US from 1987 to 2006 (Schmidt, et al., 2009). The infamous Typhoon Cobra, also known as Halsey's Typhoon after Admiral William Halsey, struck the Navy's Pacific Fleet in December 1944 during World War II, resulting in the loss of 790 sailors and three destroyers.

Because of the large economic and societal impact, the need for more accurate forecasts of tropical cyclone position and intensity is in demand. Tropical cyclone (TC) track predictions have improved steadily over the past two decades, although every year there are examples of cases of significant forecast error in the TC position, particularly when a TC may be re-curving into the mid-latitudes. In contrast, there has been almost no progress in improving TC intensity and structure forecasts over the past two decades, in part because of inadequacies in the observing network and numerical weather prediction (NWP) models, and a general lack of understanding of the key processes that contribute to TC intensification and structure changes.

In this paper, we report on recent tropical cyclone predictions made using the Navy's next-generation regional deterministic and global ensemble tropical cyclone modeling systems. In addition to research simulations of previous seasons, these systems have been applied during real-time in the past several TC seasons using the Department of Defense (DoD) high performance computing (HPC) computational resources. Here, we briefly described the systems and summarize the results from an evaluation of the performance of these new TC model capabilities.

2. COAMPS-TC

The Navy's next generation tropical cyclone prediction system is based on the Coupled Ocean Atmosphere Mesoscale Prediction System (COAMPS-TC). A description of the original COAMPS model is provided by Hodur (1997).

The model uses a terrain-following sigma-height coordinate and the non-hydrostatic compressible equations of motion. A parameterization for microphysics is used, with prognostic equations for mixing ratios of cloud droplets, ice particles, rain, snow, graupel, and drizzle. The model also includes a short-wave and long-wave radiation processes, and a planetary boundary-layer parameterization with a 1.5-order turbulence closure. The tropical cyclone version includes the following capabilities: i) relocation of the first guess-field to the observed TC position for data assimilation purposed, ii) synthetic wind and mass observations of a bogus TC in the data assimilation cycle, iii) dissipative heating, and iv) moving grid meshes that track the TC center. The forecast system also has a capability to be coupled to ocean circulation (using the Navy Coastal Ocean Model or NCOM) and wave (Simulating Waves Nearshore or SWAN) models. For the real-time experiments conducted, three nested-grids with horizontal resolutions of 45 km, 15 km, and 5 km, respectively, were used. The 15 and 5 km grids move following the TC circulation, and the 45 km grid was held fixed. Two-way interactive nesting was used.

An example of a real-time forecast for the S. Hemisphere from several months ago is shown in Figure 1. The 84-h forecast of 10-m winds for typhoon Yasi initialized at 00 UTC 30 January 2011 is shown just prior to landfall in NE Australia, when it was a category 4 storm. The COAMPS-TC track forecast for Yasi was quite accurate, as shown in Figure 1. An example of a reasonably-accurate intensity forecast is shown in Figure 2 for the 06 UTC 30 January initialization time. Although Yasi was nearly a category 4 storm, COAMPS-TC did develop it to a category 3, and was the most intense of all of the dynamical models. Note that the current Navy limited-area model GFDN kept Yasi very weak and it failed to develop into a typhoon.

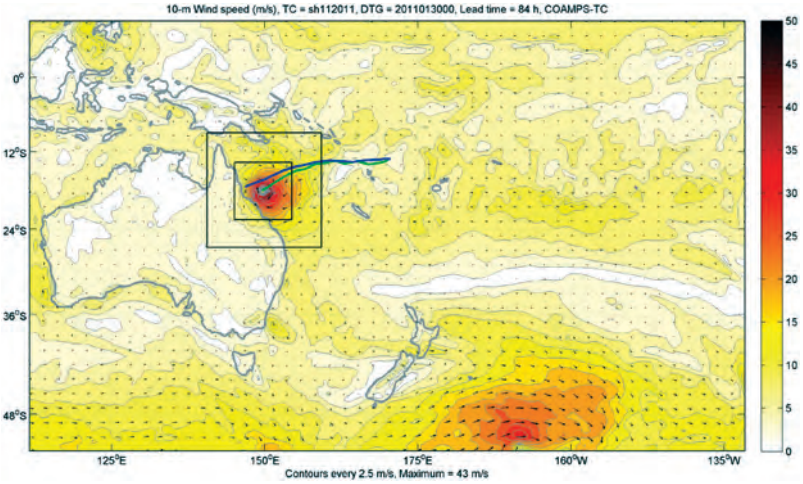


Figure 1. The 10-m wind forecast for the COAMPS-TC coarse mesh at the 84-h forecast time initialized at 00 UTC 30 January and performed in real-time. The moving 15 km and 5 km resolution grid meshes are shown by the boxes. The best track is the blue line and the green represents the COAMPS-TC forecast track.

A summary of the model intensity forecast skill for the 2010 Hurricane season in the W. Atlantic basin is shown in Figure 3. COAMPS-TC was the most skillful dynamical model for tropical cyclone intensity in the W. Atlantic basin for 2010. Note that COAMPS-TC intensity skill is the leading dynamical model for the critical 36–60-h forecast period. The performance of COAMPS-TC during this period is an improvement over the National Oceanic and Atmospheric Administration (NOAA) regional hurricane models: HWRF, experimental HWRF-X, and the GFDL models for the 2-day forecast period. These statistics show the cumulative result of many experiments conducted on the DoD HPC testing various aspects of COAMPS-TC prior to the 2010 season. Deficiencies were identified through thorough diagnosis of model experiments conducted prior to the 2010 season. These deficiencies include aspects of the data assimilation system, model initialization, and model representation of physical processes such as convection, and turbulence. A number of new advancements and improvements to the system were made that have enabled a more accurate initial state for the model, as well as improved representations of the physical processes. Additional improvements are currently being tested, with an emphasis on the microphysical parameterizations and the synthetic observations used to initialize the vortex.

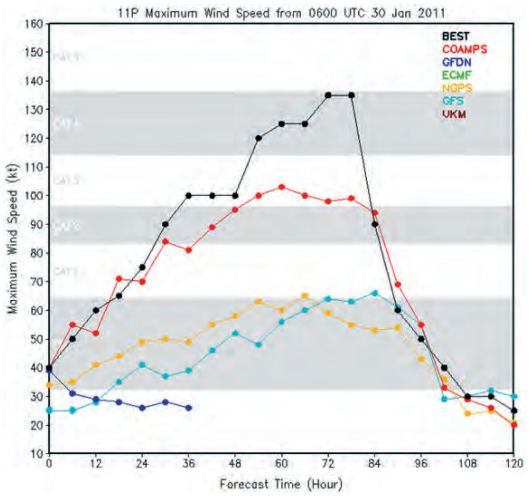


Figure 2. The maximum wind-speed forecast for the COAMPS-TC initialized at 06 UTC 30 January and performed in real-time. The best track is the black line and the red line corresponds to COAMPS-TC. Other models are shown as well.

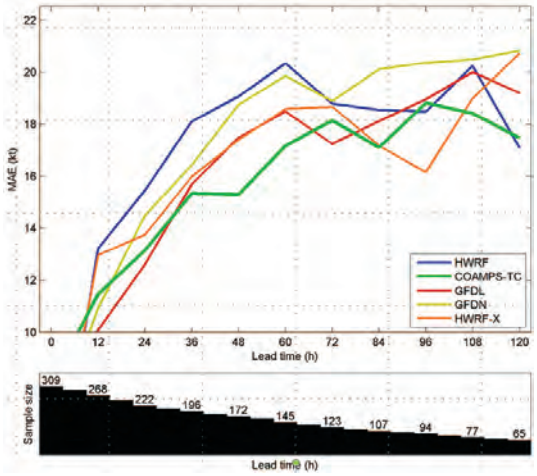


Figure 3. Summary of COAMPS-TC intensity mean-absolute-error (MAE) for the 2010 Atlantic Hurricane season and a comparison with HWRF, GFDL, GFDN, and HWRF-X results (see legend). The sample size is shown along the x axis.

A summary of the track forecast skill for the W. Pacific basin for 2010 is shown in Figure 4. COAMPS-TC exhibits comparable skill to the current Navy's regional tropical cyclone model, GFDN, with regard track error, although both limited area models trail the Navy NOGAPS and NOAA GFS global models, particularly after 72-h. The models, in general, display a systematic slow and to the right of track bias, when the track errors are decomposed in a relative track error sense (Figure 4, right panel). The track error has been demonstrated to be sensitivity to the manner in which the tropical cyclone vortex is initialized. We have been exploring alternate methods of generating synthetic observations to initialize COAMPS-TC. The current method makes use of typically 49 total synthetic observations that are constructed in radial rings extending out quite a far distance. The Joint Typhoon Warning Center (JTWC) or National Hurricane Center (NHC) warning message is used to specify the storm structure using the position and 34 and 50 kt wind radii in the quadrants. The first-guess cyclone is relocated in the analysis to provide an accurate position initialization. Based on careful verification and evaluation, the initial hurricane circulation appears to be larger in scale than the observed structure would indicate. To address this problem, a new method for generating synthetic observations has been developed. The current methodology places synthetic observations at fixed radial locations, out to 4°–6° away from the TC center. The new methodology dynamically places observations at the radius of maximum winds out to the radius of the 34-knot wind. The new synthetics represent the size and structure of the tropical cyclone more accurately relative to the current method. The statistical evaluation and impact of these synthetic observations on the track and intensity forecasts is currently under evaluation.

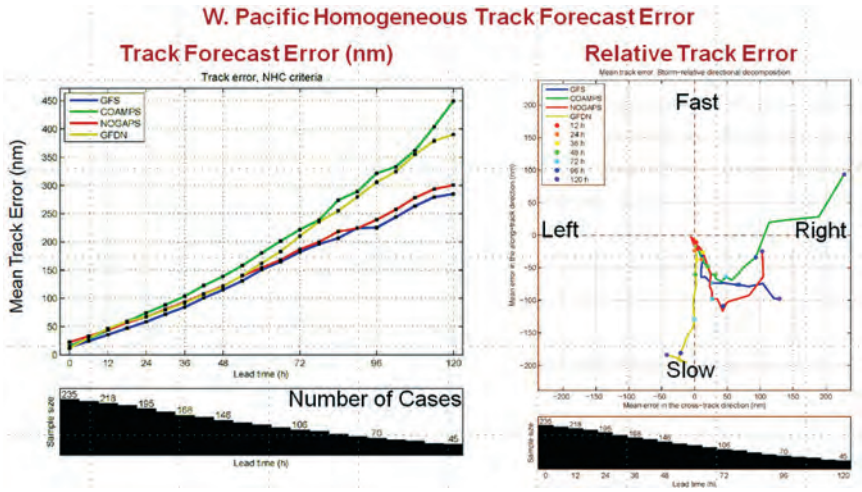


Figure 4. Summary of COAMPS-TC track skill (left) and relative track error (right) for the 2010 W. Pacific Hurricane season and a comparison with GFDN, NOGAPS, and GFS results (see legend). The sample size is shown along the x axis.

The forecast track skill has also been shown to be sensitive to the representation of the physical processes within the COAMPS-TC model. A new more sophisticated representation of the short-wave and long-wave radiation processes, which interact with clouds, has been implemented in COAMPS-TC. This new radiation parameterization results in modest improvements to the COAMPS-TC tropical cyclone track forecast skill. Additional physical parameterization developments that showed improvements to the tropical cyclone track forecasts include a new vertical mixing technique within model clouds, and a modified version of the cloud microphysical parameterization. These experiments underscore the sensitivity of tropical cyclone track prediction to not only the larger-scale flow (such as synoptic-scale troughs and ridges), but also the smaller-scale processes immediately around the tropical cyclone, which impact the larger-scale environment through secondary circulations.

3. Global Atmospheric Ensembles

Experiments were performed with the Navy Operational Global Atmospheric Prediction System (NOGAPS) ensemble prediction system (EPS) to examine how the inclusion of model uncertainty in ensemble design influences ensemble performance, including tropical cyclone (TC) track errors. The results shown are based on ensemble forecasts using NOGAPS (Peng, et al., 2004), the global spectral weather prediction model of the US Navy. The physical parameterizations include boundary-layer turbulence (Louis, et al., 1982), shallow and deep moist convection (Emanuel and Zivkovic-Rothman, 1999; Peng, et al., 2004), convective and stratiform clouds (Teixeira and Hogan, 2002), and solar and long-wave radiation (Harshvardhan, et al., 1987). All ensembles are run using the Ensemble Transform (ET) (McLay, et al., 2008) method with a 6-h cycling interval to produce the initial perturbations, with forecasts run at triangular truncation 119 (110 km) horizontal resolution and 30 levels (T119L30) with 33 members (32 perturbed members plus one member without initial perturbations). The control ensemble is run without any changes to the forecast model, and is referred to as the CTL (control) ensemble. The second ensemble is run with parameter variations only in the parameterization of the effects of sub-grid-scale deep convection, and is referred to as PAR1. The third ensemble, PAR2, is run with parameter variations in both the deep convection parameterization and the boundary-layer parameterization. All ensembles forecasts are run for 10 days starting at 00Z each day between 10 May and 12 September 2007.

One reason to include parameter variations is to enhance the diversity of model solutions through inclusion of some component of model uncertainty in ensemble design. The top panel of Figure 5 shows the ensemble spread (measured as the ensemble standard deviation about the ensemble mean) for the day-5 forecasts of 850-hPa wind-speed, averaged for the entire period, for the CTL ensemble. For the May through September time period considered here, the ensemble spread is greatest in the Southern Hemisphere mid-latitudes, and smallest in the tropics. The percent difference between the ensemble variance of PAR1 and CTL [100*(PAR1-CTL)/CTL] is shown in the middle panel of Figure 1. One can see that the increase in variance is greatest in the tropics, and is significant at the 95% level only in this region. The percent increase in variance is a maximum over the Bay of Bengal and, in general, is large in regions of strong tropical convection. This is expected, as PAR1 ensemble parameters are varied in the convective parameterization scheme only. The percent difference between PAR2 and CTL is shown in the bottom panel, and exhibits similar, somewhat larger, increases in variance than PAR1.

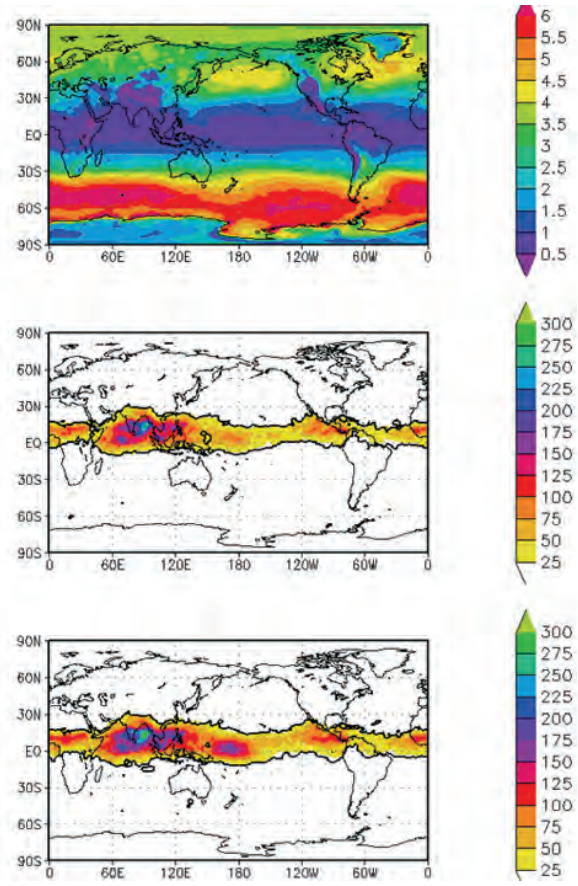


Figure 5. Ensemble standard deviation for 850-hPa wind-speed 5-day forecasts about the ensemble mean (m s^{-1}) for the control ensemble (top). Percent difference between the PAR1 ensemble and CTL ensemble (middle), and percent difference between PAR2 ensemble and CTL ensemble (bottom) for the 850-hPa wind-speed ensemble variance. Black contour indicates 95% confidence level.

The goal of accounting for model uncertain is to increase ensemble spread to be more consistent with forecast error, *and* to improve the ability of the ensemble to provide useful probabilities of events of interest, such as the probability that the low-level wind-speed in a particular location will exceed a particular threshold. The Brier scores (which may be interpreted as mean square errors of the ensemble-based probability of an event, Wilks, 2006) for 10-meter wind-speed in the tropics at two thresholds (5 m s^{-1} and 10 m s^{-1}) are shown in Figure 6. The statistical significance of the Brier score differences is determined using a moving block bootstrap technique as described in McLay and Reynolds (2009). While the Brier scores in the mid-latitudes (not shown) are basically not sensitive to the parameter variations, there is significant improvement (reduction) in the Brier scores in the tropics at both the 5 and 10-m s^{-1} thresholds. For both thresholds, PAR2 produces better (lower) Brier scores than PAR1, and both are better than the CTL. Specifically, for the 5-m s^{-1} threshold, both PAR1 and PAR2 are significantly better than CTL throughout the time period. PAR2 shows a statistically-significant, though very small, improvement over PAR1. For the 10-m s^{-1} threshold, PAR1 and PAR2 are significantly better than CTL through the first half of the forecast period, the improvement of PAR2 over PAR1 is significant through 96-h, but the gains are very small.

Previous work has shown that multi-model ensembles of tropical cyclone (TC) tracks may produce ensemble mean tracks of lower-error than the individual ensemble members (Goerss, 2000). Therefore, it is also plausible that a multi-parameter ensemble, to the extent that it represents different forecast model formulations, may also improve ensemble-mean TC tracks over a single-model ensemble. Figure 7 shows the ensemble mean TC track errors for all storms during the May 10 through September 12, 2007 period for which the ensemble is run. The comparison is homogenous and the number of storms in each sample is shown below the x axis. Track errors and statistical significance (accounting for serial correlation) are computed using the Automated Tropical Cyclone Forecast (ATCF) system (Sampson and Schrader, 2000). In addition to the results for the CTL, PAR1, and PAR2 ensemble means, the TC track error from the operational NOGAPS T239 deterministic forecast is also shown. While the differences are relatively small, there are some statistically-

significant improvements gained by adding parameter variations. In particular, PAR2 offers small but statistically-significant improvements over CTL at the 95% level at 24, 72, and 120 hours (and better at the 94% level at 96-h).

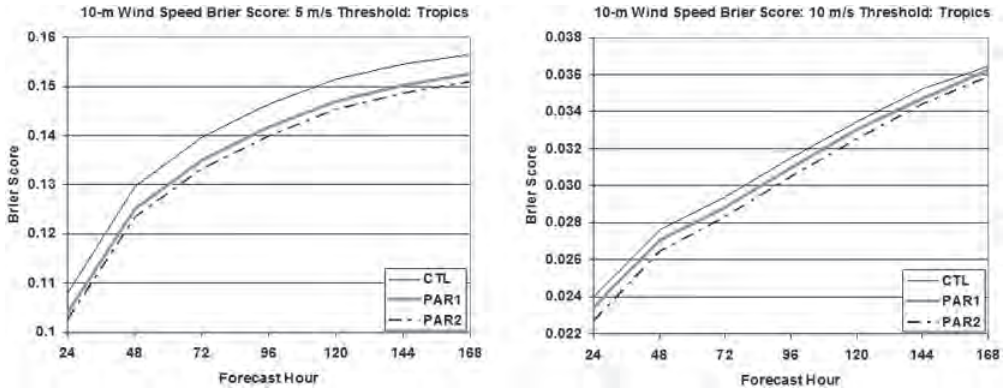


Figure 6. Brier score for 10-m wind-speed in the tropics (20S-20N) for CTL (thin black), PAR1 (thick grey) and PAR2 (thick dot dash) ensembles for (top) 5-m s^{-1} wind-speed threshold, and (bottom) 10 m s^{-1} wind-speed threshold. For the 5-m s^{-1} threshold, the improvements of PAR1 and PAR2 over CTL are statistically-significant at the 95% level for all forecast lengths. For the 10-m s^{-1} threshold, the improvements of PAR1 and PAR2 over CTL are statistically-significant at the 95% level out to 96-h.

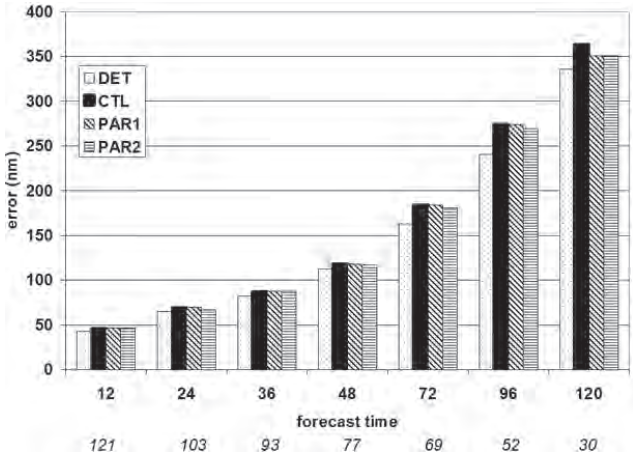


Figure 7. Homogenous NH TC track forecast error (nm), for CTL, PAR1, and PAR2 ensemble mean track as denoted in key. Also shown is the average forecast error of the T239L30 NOGAPS operational deterministic forecast (DET). The numbers of verifying forecasts are shown below the x axis. The improvements of PAR2 over CTL are statistically-significant at the 95% level at 24-, 72-, and 120-h.

4. Conclusion

In this paper, we have summarized recent progress in the development and testing of the Navy's next-generation tropical cyclone prediction system for the mesoscale, COAMPS-TC, and new capabilities for the Navy's global ensemble system using NOGAPS. An improved version of the COAMPS-TC has been developed in the past year and executed in real-time for multiple regions of the globe, including the W. Atlantic, W. Pacific, E. Pacific, C. Pacific, and S. Hemisphere basins. Evaluation of these real-time forecasts performed on the DoD HPC reveals that COAMPS-TC was the most skillful dynamical model for tropical cyclone intensity forecasts in the W. Atlantic basin for the 2010 season. The COAMPS-TC intensity skill is superior to other dynamical models for the 36–60-h forecast period. COAMPS-TC has comparable TC track skill as the current Navy's regional tropical cyclone model, GFDN; although both regional models are not as skillful as the Navy NOGAPS and the NOAA GFS global models after 3 days.

Future improvements to COAMPS-TC are currently being tested. These improvements include new methods to initialize the tropical cyclone vortex, improved representations of key physical process such as turbulence and clouds, and new methods to evaluate COAMPS-TC forecasts to better diagnose systematic tropical cyclone forecast errors. New

methods for data assimilation using Ensemble Kalman Filter and four-dimensional variational assimilation methods are currently being developed, and are expected to provide significant improvements to both track and intensity forecast skill for COAMPS-TC.

For the global ensemble system, it has been shown that significant improvement is obtainable in the NOGAPS ensemble performance in the tropics through the inclusion of parameter variations, which reflect components of model error. Ensemble performance was improved in the tropics under a variety of metrics, including a closer match between ensemble spread and ensemble means error, better probabilistic predictions of low-level wind-speed, and improved tropical cyclone track forecasts. Better performance was obtained when varying parameters in the parameterization of both deep convection and boundary-layer processes, instead of varying parameters in the deep convection parameterization only. Work is continuing on methods to account for model uncertainty in ensemble design, and it is anticipated that these improvements will be transitioned into operations within the next year.

Acknowledgements

We acknowledge support through the Office of Naval Research's Program Element 0601153N. We also appreciate support for computational resources through a grant of Department of Defense High Performance Computing time from the DoD Major Supercomputing Resource Centers at Stennis, MS and Wright-Patterson, OH. COAMPS® is a registered trademark of the US Naval Research Laboratory.

References

Emanuel, K.A. and M. Zivkovic-Rothman, "Development and evaluation of a convection scheme for use in climate models", *J. Atmos. Sci.*, 56, pp. 1766–1782, 1999.

Goerss, J.S., "Tropical cyclone track forecasts using an ensemble of dynamical models", *Mon. Wea. Rev.*, 128, 1187–1193, 2000.

Harshvardhan, R. Davies, D.A. Randall, and T.G. Corsetti, "A Fast Radiation Parameterization for Atmospheric Circulation Models", *J. Geophys. Res.*, 92, pp. 1009–1016, 1987.

Hodur, R.M., "The Naval Research Laboratory's Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS)", *Mon. Wea. Rev.*, 125, pp. 1414–1430, 1997.

Louis, J.F., M. Tiedtke, and J.F. Geleyn, "A short history of the operational PBL parameterization at ECMWF", *ECMWF Workshop on Planetary Boundary Parameterizations*, pp. 59–79, 1982.

McLay, J.G., C.H. Bishop, and C.A. Reynolds, "Evaluation of the ensemble transform analysis perturbation scheme at NRL", *Mon. Wea. Rev.*, 136, pp. 1093–1108, 2008.

McLay, J.G. and C.A. Reynolds, "Two alternative implementations of the ensemble-transform (ET) analysis-perturbation scheme: The ET with extended cycling intervals, and the ET without cycling", *Q. J. R. Meteorol. Soc.*, 135, pp. 1200–1213, 2009.

Peng, M.S., J.A. Ridout, and T.F. Hogan, "Recent Modifications of the Emanuel Convective Scheme in the Navy Operational Global Atmospheric Prediction System", *Mon. Wea. Rev.*, 132, pp. 1254–1268, 2004.

Sampson, C.R. and A.J. Schrader, "The Automated Tropical Cyclone Forecasting System (Version 3.2)", *Bull. Amer. Meteor. Soc.*, 81, pp. 1231–1240, 2000.

Schmidt, S., C. Kemfert, and P. Hoppe, "Tropical cyclone losses in the USA and the impact of climate change - A trend analysis based on data from a new approach to adjusting storm losses", *Environ. Impact Assess. Rev.*, 29, pp. 359–369, 2009.

Teixeira, J. and T.F. Hogan, "Boundary-layer clouds in a global atmospheric model, simple cloud-cover parameterizations", *J. Climate*, 15, pp. 1261–1276, 2002.

Wilks, D.S., *Statistical Methods in the Atmospheric Sciences*, 2d ed., Academic Press, 627 pp., 2006.

What is Required to Model the Global Ocean Circulation?

James G. Richman, Prasad G. Thoppil, Patrick J. Hogan, and Alan J. Wallcraft
US Naval Research Laboratory, Oceanography Division (NRL-SSC), Stennis Space Center, MS
{james.richman, prasad.thoppil, pat.hogan, alan.wallcraft}@nrlssc.navy.mil

Abstract

Simulating and forecasting the circulation of the global ocean is a difficult task. The circulation is forced at the surface by the atmosphere, but below the surface the circulation is dominated by internal nonlinear interactions. The kinetic energy in the ocean is dominated by mesoscale eddies, meanders and rings of the boundary currents with scales of 50 km to 500 km. Mesoscale eddies have a critical role in the dynamics of the ocean circulation with instabilities of the strong mean currents generating eddies in the upper-ocean, interactions between eddies transferring energy from the upper-ocean to the deep ocean, where eddies interact with bottom topography to generate abyssal mean-flows and eddies transferring momentum back to the mean currents. The present generation of high-resolution ocean circulation models, with horizontal resolution of ~1/10°, appears to be deficient in kinetic energy when compared with long-term observations. A series of near-twin experiments, using the global HYbrid Coordinate Ocean Model (HYCOM) with identical atmospheric forcing, but varying in horizontal resolution and assimilation of altimetric steric height anomalies, show significant improvement with a better representation of mesoscale eddies when compared to observations from surface drifters, geostrophic currents from satellite altimetry, subsurface floats and deep current meter moorings. For a 1/12.5° (~9 km) global model, the eddy kinetic energy (EKE) at the surface and in the abyss is low by ~21% and ~24%, respectively, compared to surface drifting buoys and deep current meters. Increasing the model resolution to 1/25° (~4.4 km) or injecting mesoscale eddies through the assimilation of surface observations in a 1/12.5° model increases the surface and the abyssal EKE to levels consistent with the observations. In these models, the surface (abyssal) EKE is increased by 23% (51%) for the higher resolution model and 15% (46%) for the data-assimilative model. The upper-ocean kinetic energy of the mean-flow in the data-assimilative hindcast is decreased even though the surface and abyssal EKE are increased.

1. Introduction

The eddy kinetic energy (EKE) in the upper-ocean, encompassed by mesoscale eddies, meanders and rings of the boundary currents [Stammer, 1997; Ferrari and Wunsch, 2009, 2010], is generated by instabilities of the mean flow and direct wind forcing. The present eddy-resolving global ocean general circulation models (OGCMs), running at ~1/10° horizontal grid resolutions, appear to underestimate EKE at the surface compared to observations, indicating that the mean circulation is not inertial enough to generate vigorous upper-ocean instabilities, which in turn is responsible for the generation of meanders and eddies. Maltrud and McClean [2005] noted that eddy energy in the western boundary currents and relatively quiescent regions of the global 1/10° Parallel Ocean Program OGCM is too weak, when compared to sea surface height altimetry data. The eddy energy in the abyssal ocean by is supplied by nonlinear interactions transferring energy from vertically-sheared baroclinic to depth-independent barotropic states [Rhines, 1979; Hurlburt and Hogan, 2008; Ferrari and Wunsch, 2009]. In the abyssal ocean, eddies interact with bottom topography to generate a strong eddy-driven mean circulation [Holland, 1978; Rhines, 1979].

Adequately representing mesoscale eddies and the energy and enstrophy cascades in ocean models is key to simulating the mean circulation with studies suggesting that horizontal resolutions around 1/10° are sufficient [Smith, et al., 2000; Hurlburt and Hogan, 2000; Oschlies, 2002; Maltrud and McClean, 2005]. However, at this resolution the models significantly underestimate the EKE in the abyssal ocean (depths greater than 3,000 m) [Scott, et al., 2010]. The eddy-driven mean abyssal circulation, which is constrained by the topography, can steer the mean pathways of the upper-ocean currents through a dynamical process known as upper-ocean - topographic coupling [Holland, 1978; Hogan and Hurlburt, 2000; Hurlburt, et al., 2008]. Recent model studies suggest that a strong eddy-driven mean abyssal circulation is sufficient to obtain a realistic Gulf Stream pathway and its separation from the western boundary [Hurlburt, et al., 2008]. Thus, a strong

abyssal circulation plays a critical role in the maintenance of mean circulation over the entire depth of the ocean, especially in regions dominated by intrinsic instability rather than atmospheric forcing. An intriguing question then becomes; how can we achieve a realistic representation of the abyssal ocean circulation in the OGCMs?

Resolution studies [Bryan, et al., 2007; Smith, et al., 2000; Hogan and Hurlburt, 2000; Oschlies, 2002] show that increasing the horizontal resolution for an OGCM generates a stronger mean-flow and thereby additional instabilities in the upper-ocean, which in turn can lead to a stronger eddy-driven abyssal circulation by vertically transferring the energy downward. Here we show that increasing the horizontal resolution from 1/12.5° to 1/25° yielded the most realistic representation of the ocean EKE from the surface to the abyss. Alternately, one can increase the EKE in the upper-ocean indirectly by injecting eddies via assimilating ocean surface observations in a 1/12.5° model. In this case, the additional EKE is generated by eddies introduced through data assimilation rather than intrinsic instability of the ocean dynamics. Our results indicate a significant increase in the abyssal ocean EKE when data assimilation is included. The major focus of attention in this paper is the comparative global ocean energetics from observations, employing four independent sets of observations representing the entire water column, and twin model experiments, only varying in horizontal resolution and data assimilation.

2. Model and Data

We present three model experiments differing only in horizontal resolution and data assimilation using the HYbrid Coordinate Ocean Model (HYCOM) [Bleck, 2002], which has 32 hybrid layers in the vertical and is forced with three-hourly, 0.5° Navy Operational Global Atmospheric Prediction System (NOGAPS) atmospheric fields. The models are a 1/12.5° (~9 km at the equator) and a 1/25° (~4.4 km at the equator) horizontal resolution non-assimilative models, denoted respectively as 1/12.5° FR and 1/25° FR and a 1/12.5° with data assimilation, denoted as 1/12.5° DA. The initial conditions for each experiment are spin-up from rest using the GDEM3 climatology for 10 years using climatalogical forcing. Thereafter, the model is forced with interannually varying NOGAPS atmospheric fields from 2003 to 2009. For the non-assimilative models, the analysis is performed for the five years 2005 to 2009. Only the two years, 2008–2009, are examined for the data-assimilative, where observations of satellite-derived sea surface height (SSH) and vertical profiles of temperature are incrementally inserted using a Multi-Variate Optimal Interpolation scheme [Cummings, 2005].

The modeled kinetic energy is compared to four independent sets of observations for three different dynamical regimes representing: 1) surface, 2) below the wind-driven mixed layer (150 m) and thermocline (1,000 m), and 3) abyssal ocean. At the surface, the instabilities of the mean-flow and direct-wind-forcing dominate, while quasi-geostrophy controls the energy below the mixed-layer and thermocline. We use kinetic energy estimates from surface drifter observations [Lumpkin and Pazos, 2007], satellite altimetry (150 m) [Ducet, et al., 2000], ARGO floats at 1,000 m [Lebedev, et al., 2007], and deep current moorings [Scott, et al., 2010] for model-data comparisons.

3. Results

At the surface, the highest EKE (>800 $cm^2\ s^{-2}$) are concentrated in the vicinity of the major current systems associated with the Gulf Stream and its continuation as the North Atlantic Current, the Loop Current in the Gulf of Mexico, the Brazil Current, the Kuroshio Current (off Japan), the equatorial current system and, in the southern hemisphere, the Antarctic Circumpolar Current (ACC), Agulhas Current (off southeast Africa), Eastern Australian Current, and Leeuwin Current near the western coast of Australia (Figure 1). The major circulation features, observed with the drifting buoys (Figure 1d), are reproduced by all models, as indicated by the high-spatial correlation (~0.8) between the model and observed EKE (Table 1). However, the global mean EKE differs significantly between the models and the observations, with 1/25° FR being the closest. Relative to the drifting buoys, the 1/12.5° FR model underestimates the global mean EKE by 21% (Table 1), implying weaker flow instabilities and fewer meanders. Increasing the model resolution increases the surface EKE by 23% from 343 $cm^2\ s^{-2}$ in the 1/12.5° FR to 423 $cm^2\ s^{-2}$ in 1/25° FR, which is 97% of the observed EKE (436 $cm^2\ s^{-2}$). The models are forced by identical atmospheric fields, which implies that the increase in EKE with resolution arises primarily from increased baroclinic and barotropic instability of the stronger mean-flow in the higher resolution model generating more meanders and eddies. Alternately, the increase in surface EKE could arise from a better representation of horizontal small-scale mixed-layer processes that are not resolved by the 1/12.5° model. Injecting eddies via assimilation of surface observations in the 1/12.5° DA experiment increases the surface EKE by 15% to 393 $cm^2\ s^{-2}$, which is 90% of the observed EKE. Thus, both the resolution and the data assimilation increase the overall global mean surface EKE by 23% and 15% respectively. There are; however, obvious model-data discrepancies of the spatial distribution of EKE in the simulations. For example, high-EKE, associated with the anti-cyclonic rings from the Agulhas Current (AC) at its

retroflection, occurs along a narrow path in the non-assimilative simulations, but is not observed by the drifters nor the data-assimilative hindcast. The non-assimilative models tend to shed AC rings at regular intervals between the Agulhas Bank and Agulhas Plateau, and the rings travel northwestward into the South Atlantic with little dissipation. In the non-assimilative models, the AC is unable to penetrate into the Agulhas Basin. The erroneously large EKE along the wrong pathway is a known artifact of the model (also appears in other global models as well) and may be of the result of a too-strong AC, although the exact cause remains to be determined. For the global EKE statistics presented in Table 1 we excluded this region (20°W-10°E, 40°-20°S). Exclusion of this region led to less than 5% changes in EKE at 150 m, and had a negligible contribution to the EKE at surface and depths below 1,000 m. Two other regions with significant departure from the observations occur off Java/Indonesia and in the Southeast Indian Ocean (seen only in 1/25° FR). Examination of altimeter-derived EKE in these regions shows similar patterns. Model-data differences in relatively quiescent regions (e.g., a bias toward lower-EKE in the southeast Pacific ~30°S) may be attributed, in part, to the shorter analysis period for the models (2005–2009) and to the atmospheric forcing. Compared to the drifting buoys the 1/25° FR model is too energetic in the high-EKE regions where intrinsic instability dominates the mesoscale eddies. In these highly-energetic regions, the estimated drifter EKE may be underestimated due to the tendency of the drifting buoys to accumulate in regions of weak flow.

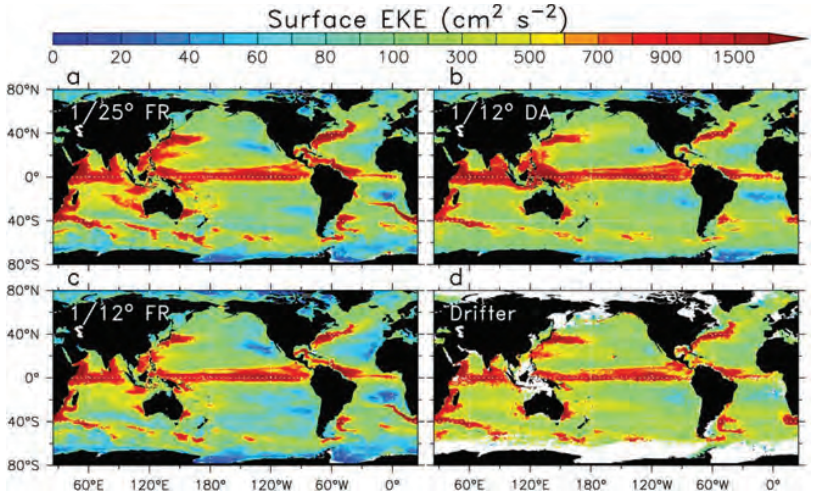


Figure 1. Surface-eddy kinetic energy (EKE in cm² s⁻²) from the three numerical experiments (a) 1/25° FR (2005–2009), (b) 1/12.5° DA (2008–2009), and (c) 1/12.5° FR (2005–2009) and (d) drifter observations encompassing the period 1983–2009. The surface drift observations are binned into 1°×1° grids using daily values and those grid-points with at least 100 observations are considered. The EKE is computed from the daily velocity fields using the equation (<u'²> + <v'²>)/2, where brackets indicate time means and primes denote deviations from the time-mean velocities, (u', v') = (u, v) - (<u>, <v>).

The kinetic energy of the mean flow (KEM) increases by 13% in the higher resolution simulation, from 171 cm² s⁻² in 1/12.5° FR to 193 cm² s⁻² in 1/25° FR. Associated with the stronger mean circulation is an increase in EKE. Over the entire ocean, a modest spatial correlation of ~0.67 occurs between the model mean kinetic energy and the drifting buoy mean kinetic energy (Table 1). Visual inspection of the maps of KEM (not shown) exhibit arrow bands of high KEM (>700 cm² s⁻²) coincident with the regions of high EKE indicating a stronger mean circulation with increased generation of meanders and eddies. The non-assimilative modeled KEM, however, is systematically higher than the drifter estimates by 21% and 42% respectively. The differences between the models and observations may arise from two possible sources; 1) drifting buoys tend to accumulate in regions of weak flow leading to a low bias in the mean-flow [Pasquero, et al., 2007] and 2) the analysis period for the models (2005–2009) is much shorter than the 22 years of buoy history. Injection of eddies by data assimilation increases the surface EKE, but the KEM is weaker in 1/12.5° DA compared to the non-assimilative simulations. Among the models, the spatial pattern of KEM in the 1/12.5° DA has a better correlation with the drifter observations, but a lower KEM of 123 cm² s compared to the observed 159 cm² s⁻².

Table 1. Observed and modeled eddy and mean kinetic energy. Abbreviations used: EKE, eddy kinetic energy; KEM, kinetic energy of mean-flow computed using the equation (<u2> + <v2>)/2, where brackets denote time means; FR, non-assimilative simulation; DA, data assimilative hindcast.

	Surface*		150 m[a]	1000 m[b]	Abyssal Ocean*		Abyssal Ocean[†]	
Model Expts.	EKE[c] ($cm^2\ s^{-2}$)	KEM ($cm^2\ s^{-2}$)	EKE[c] ($cm^2\ s^{-2}$)	EKE[c] ($cm^2\ s^{-2}$)	EKE ($cm^2\ s^{-2}$)	KEM ($cm^2\ s^{-2}$)	EKE ($cm^2\ s^{-2}$)	KEM ($cm^2\ s^{-2}$)
1/12.5° FR (2005–2009)	343 (0.81)‡	171 (0.69)	121 (0.64)	26.4 (0.77)	8.37	2.81	13.27 (0.71)	6.84 (0.51)
1/25° FR (2005–09)	423 (0.82)	193 (0.67)	181 (0.70)	37.9 (0.77)	12.61	4.44	18.28 (0.80)	8.54 (0.54)
1/12.5° DA (2008–09)	393 (0.77)	160 (0.67)	123 (0.79)	33.7 (0.71)	12.24	3.48	14.17 (0.80)	6.83 (0.33)
Obs.	436	135	159	27.5			17.73	8.21

*Mean over the global ocean (70°S-70°N)
[a]Mean over the global ocean (60°S-60°N) excluding the tropical ocean (5°S-5°N) where the assumption of geostrophy leads to potentially large errors
[b]Mean over the global ocean (70°S-70°N) using 3°×3° grid
[c]Due to the unrealistic Agulhas overshooting into the south Atlantic in the simulations, the region bounded by 20°W-10°E, 40°-20°S is excluded from mean.
[†]Mean values obtained at the 712 current meter mooring locations
[‡]The spatial correlation coefficient between the model and observed kinetic energy

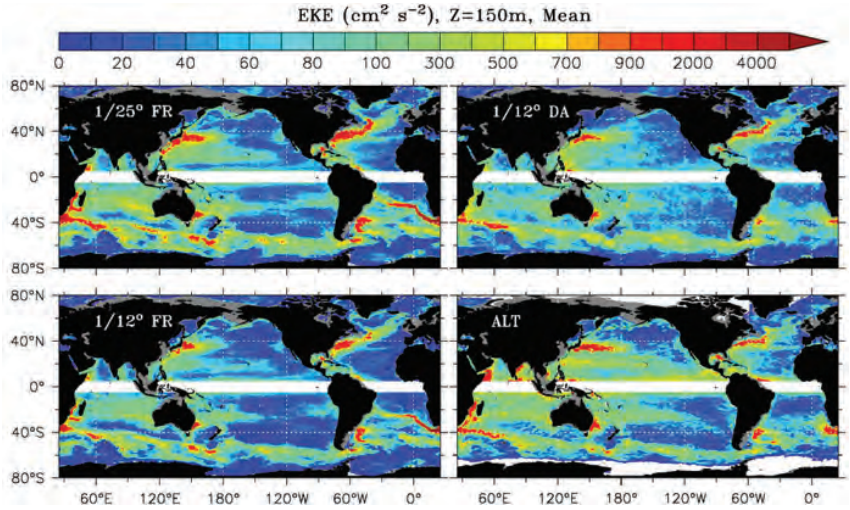


Figure 2. Eddy kinetic energy (EKE in $cm^2\ s^{-2}$) at 150m from the three numerical experiments (a) 1/25° FR (2005–2009), (b) 1/12.5° DA (2008–2009), and (c) 1/12.5° FR (2005–2009) and (d) geostrophic velocity estimates from AVISO sea surface height maps for the period 2005–2009. The band 5°N to 5°S is excluded from the EKE estimates.

Quasi-geostrophic flow dominates the EKE below the wind-driven mixed layer. In Table 1 and Figure 2, the EKE of the models at 150 m is compared with geostrophic velocity estimates from mapped satellite altimeter sea surface heights (SSH) [Ducet, et al., 2000]. At 150 m, which is below the wind-driven mixed layer, but still representative of the upper-ocean, the EKE of the 1/25° FR is the highest at 181 $cm^2\ s^{-2}$ exceeding the altimeter estimate by 14%. Both 1/12.5° FR and 1/12.5° DA have nearly the same EKE (122 $cm^2\ s^{-2}$) approximately 23% below the altimeter estimate (159 $cm^2\ s^{-2}$). The 1/12.5° models rapidly attenuate the EKE with depth which makes a quantitative comparison with the surface geostrophic velocity difficult. Among the models, 1/12.5° DA is highly correlated with the altimeter EKE with a correlation coefficient of 0.79. We expect the energy estimates from the SSH are affected by sampling artifacts. The mapped geostrophic velocity estimates will be lower than the true geostrophic velocity due to the removal of variability on times shorter than approximately 10 days and horizontal scales smaller than approximately 50 km.

Within the lower thermocline we use subsurface drift vectors at 1,000 m from the ARGO floats [Lebedev, et al., 2007] to examine the EKE. Again, the energy estimates from the ARGO float are subjected to sampling errors. The number of ARGO floats is much smaller than the number of surface drifters. For the past 5 years, approximately 3,000 ARGO floats return a position observation every 10 days. Thus, the sampling of the ARGO floats is coarser in space and time with a

shorter history than surface drifters. We have binned the drift vectors on a 3°×3° grid to get at least 100 observations in each grid box. Given the small sample size, we expect the ARGO EKE estimates to be biased low. At 1,000 m, the higher resolution and data assimilative model EKE exceed the 1/12.5° FR by 44% (26.4 to 37.9 $cm^2\ s^{-2}$) and 28% (26.4 to 33.7 $cm^2\ s^{-2}$) respectively, similar to the surface EKE (Table 1). The 1/12.5° FR underestimates the observed EKE by 4%. Eddies in the upper-ocean have a significant impact on the abyssal circulation (depths greater than 3,000 m), as abyssal currents can be generated by baroclinically-unstable flows via vertical transfer of eddy energy into the abyssal ocean. In the models, high-abyssal ocean EKE (80–300 $cm^2\ s^{-2}$) is confined to the regions below high-surface EKE such as western boundary currents and the ACC (Figure 2), a strong indicator of vertical transfer of EKE from the surface to the abyssal ocean. For the global abyssal ocean, the EKE increases by 51% from 8.4 $cm^2\ s^{-2}$ to 12.6 $cm^2\ s^{-2}$ when the resolution is doubled, and by 46% to 12.2 $cm^2\ s^{-2}$ with data assimilation (Table 1). In the 1/12.5° DA, the surface eddies introduced by the assimilation of sea surface height are driving a stronger eddy-driven abyssal circulation. The SSH anomalies are converted into synthetic profiles of temperature and salinity in the upper-ocean for assimilation. A comparison of model EKE with a global dataset of 712 quality-controlled moored current meter records [Scott, et al., 2010] indicates that the 1/25° FR has the most realistic representation of abyssal ocean EKE and the 1/12.5° FR underestimates the EKE by 24%. At these locations, the EKE increased from 13.3 $cm^2\ s^{-2}$ in 1/12.5° FR to 18.5 $cm^2\ s^{-2}$ in the 1/25° FR, comparable with the observed current meter measurements (17.5 $cm^2\ s^{-2}$, Table 1). When correlated with the current meter observations, both 1/25° FR and 1/12.5° DA EKE has higher correlation (~0.8) compared to 1/12.5° FR (0.71). For the mean global abyssal circulation, the KEM increases by 58% for 1/25° FR and 24% with data assimilation. However, at the current meter locations the KEM increases by 25% with the doubled resolution, but remains virtually unchanged with data assimilation.

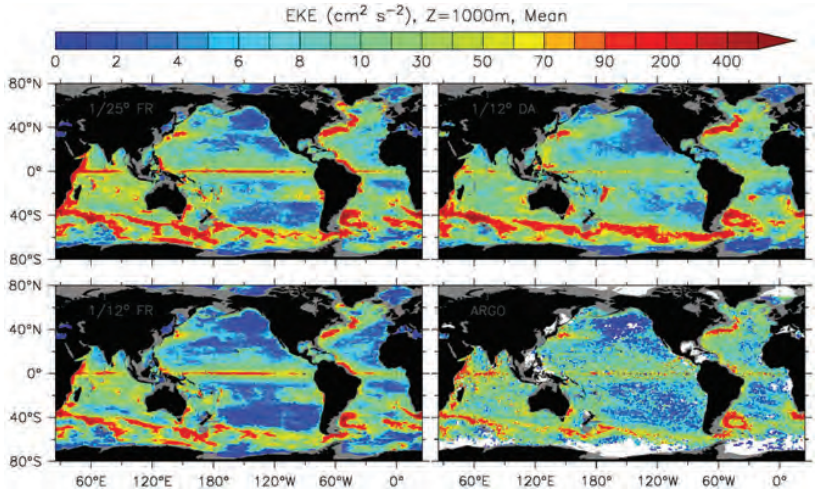


Figure 3. Eddy kinetic energy (EKE in $cm^2\ s^{-2}$) at 1,000m from the three numerical experiments (a) 1/25° FR (2005–2009), (b) 1/12.5° DA (2008–2009), and (c) 1/12.5° FR (2005–2009) and (d) ARGO 1,000m drift observations encompassing the period 1998–2009. The surface drift observations are binned into 3°×3° grids using daily values and those grid points with at least 100 observations are considered.

Noting that the abyssal EKE is greatest beneath the regions of high-surface EKE, we have extracted the Gulf Stream region (80°W-30°W, 10°N-60°N) for closer examination. The overall patterns of surface EKE in both 1/12.5° DA and 1/25° FR are similar to the drifter observations (Figures 3a–3d). In 1/12.5° FR, the simulated EKE along the North Atlantic Current between 55°W and 35°W is significantly underestimated (see box in Figures 3a–3d). In the abyssal ocean, it is clear from the superimposed current meter observations in Figures 3e–3g that both the double- resolution and data assimilative models have realistic EKE below the Gulf Stream. The 1/12.5° FR EKE is too low east of 60°W, typically by a factor of two and too weak farther to the west consistent with the weaker surface EKE.

4. Conclusion

The surface and abyssal circulation of the ocean are strongly coupled through the cascades that vertically redistribute the energy and vorticity throughout the entire water column. The surface kinetic energy of ocean circulation is dominated by eddy kinetic energy (EKE) associated with the instabilities of the mean-flow and direct wind forcing. The eddy-eddy interactions transfer energy, initially, from large scales towards the Rossby radius scale and vorticity towards small scales.

At scales near the Rossby radius, energy is transferred from the upper- ocean into the abyss (barotropification). The abyssal-eddies interact with topography and can transfer energy back to the mean-flow. The present generation of eddy-resolving global ocean circulation models at ~1/10° resolution appear to underestimate the EKE in both the upper and abyssal ocean. For example, the simulated EKE in a 1/12.5° FR accounts for only about 79% and 76% of the observations at the surface and at the abyssal ocean respectively. We show that doubling the model resolution (1/25°) to better resolve the nonlinear eddy cascades of energy and enstrophy or by injecting eddies via assimilating ocean surface observations improves the model energetics to be consistent with independent observations over most of the world's oceans. An increase in the surface EKE by 23% (15%) and the corresponding 51% (46%) increase in the abyssal EKE in the 1/25° FR (1/12.5° DA) model clearly demonstrates the need to improve representation of upper-ocean EKE as a prerequisite for strong eddy-driven abyssal circulation. The surprising result from these experiments is the impact of a modest increase in resolution on the energetic of the eddy-resolving OGCM. While the models examined clearly resolve the Rossby radius (dominant-eddy scale), the model dynamics require resolving the eddy-eddy interactions and the cascades of energy and enstrophy. Vorticity and enstrophy are dominated by smaller horizontal scales and benefit from the increased resolution.

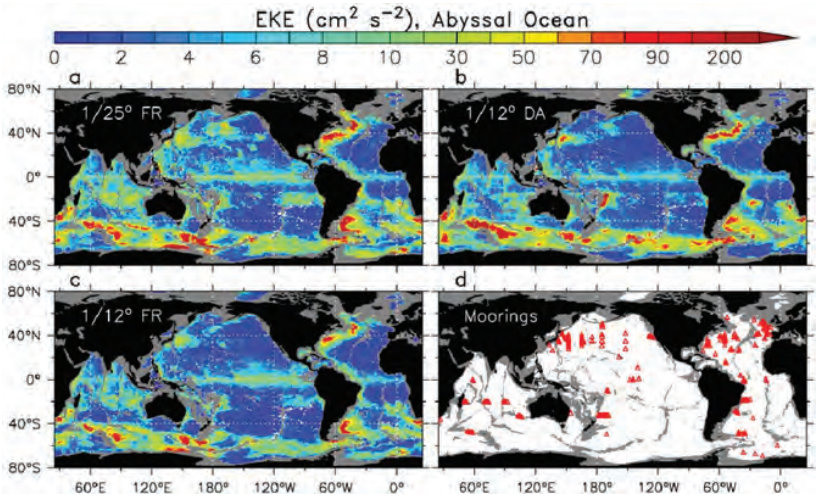


Figure 4. Abyssal ocean EKE (cm² s⁻²) averaged below 3,000 m from the three numerical experiments (a) 1/25° FR, (b) 1/1212.5° DA, (c) 1/1212.5° FR and (d) locations of the 712 deep current meter moorings used to validate the model kinetic energy [see Scott, et al., for details]. Moorings with a record at least 180 days are considered. Depths less than 3,000 m are masked grey.

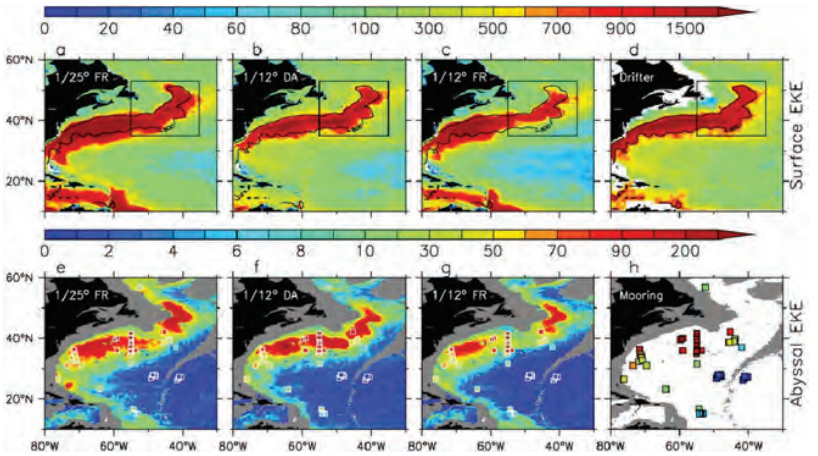


Figure 5. The EKE in the Gulf Stream region. Surface EKE (cm² s⁻²) for the three numerical experiments (a) 1/25° FR, (b) 1/12.5° DA, and (c) 1/12.5° FR with the 800 cm² s⁻² contour from the (d) drifter observations superimposed on the model surface EKE. Abyssal ocean EKE (cm² s⁻²) averaged below 3,000 m for the Gulf Stream region from the three numerical experiments (e) 1/25° FR, (f) 1/12.5° DA, and (g) 1/12.5° FR with superimposed EKE estimates from the (h) moored current meter observations shown as color-filled squares (same color shading for both models and the observations). Depths less than 3,000 m are masked grey.

Acknowledgements

This work was supported in part by a grant of computer time from the DoD High Performance Computing Modernization Program at the NAVY DoD Supercomputing Resource Center (DSRC). It was sponsored by the Office of Naval Research (ONR) through the NRL projects, Eddy-Resolving Global Ocean Prediction Including Tides and Ageostrophic Vorticity Dynamics of the Ocean. We thank Robert Scott of the Université de Bretagne Occidentale for providing the historical current meter data and E. Joseph Metzger, Ole Martin Smedstad, and Luis Zamudio (NRL) for making the global HYCOM simulations available.

References

Bleck, R., "An oceanic general circulation model framed in hybrid isopycnic-cartesian coordinates", *Ocean Modell.*, 4, pp. 55–88, 2002.

Bryan, F.O., M.W. Hecht, and R.D. Smith, "Resolution convergence and sensitivity studies with North Atlantic circulation models. Part I: The western boundary current system", *Ocean Modell.*, 16, pp. 141–159, 2007.

Cummings, J.A.," Operational multivariate ocean data assimilation", *Q. J. Roy. Meteorol. Soc.*, 131, pp. 3583–3604, 2005.

Ducet, N., P.-Y. Le Traon, and G. Reverdin, "Global high-resolution mapping of ocean circulation from TOPEX/Poseidon and ERS-1 and 2", *J. Geophys. Res.*, 105, pp. 19477–19498, 2000.

Ferrari, R. and C. Wunsch, "Ocean circulation, kinetic energy: reservoirs, sources and sinks", *Annu. Rev. Fluid Mech.*, 41, pp. 253–282, 2009.

Ferrari, R. and C. Wunsch, "The distribution of eddy kinetic and potential energies in the global ocean", *Tellus*, 62A, pp. 92–108, 2010.

Hogan, P.J. and H.E. Hurlburt, "Impact of upper-ocean - topographic coupling and isopycnal outcropping in Japan/East Sea models with 1/8° to 1/64° resolution", *J. Phys. Oceanogr.*, 30, pp. 2535–2561, 2000.

Holland, W.R., "The role of mesoscale eddies in the general circulation of the ocean - numerical experiments using a wind-driven quasi-geostrophic model", *J. Phys. Oceanogr.*, 8, pp. 363–392, 1978.

Hurlburt, H.E., and P.J. Hogan, "Impact of 1/8° to 1/64° resolution on Gulf Stream model-data comparisons in basin-scale subtropical Atlantic ocean models", *Dyn. Atmos. Oceans.*, 32, pp. 283–329, 2000.

Hurlburt, H.E. and P.J. Hogan, "The Gulf Stream pathway and the impacts of the eddy-driven abyssal circulation and the deep western boundary current", *Dyn. Atmos. Oceans.*, 45, pp. 71–101, 2008.

Hurlburt, H.E., E.J. Metzger, P.J. Hogan, C.E. Tilburg, and J.F. Shriver, "Steering of upper-ocean currents and fronts by the topographically-constrained abyssal circulation", *Dyn. Atmos. Oceans.*, 45, pp. 102–134, 2008.

Lebedev, K.V., H. Yoshinari, N.A. Maximenko, and P.W. Hacker, "Velocity data assessed from trajectories of Argo floats at parking level and at the sea surface", *IPRC Technical Note No. 4(2)*, Honolulu, HI, 16pp, 2007.

Lumpkin, R. and M. Pazos, "Measuring surface currents with Surface Velocity Program drifters: the instrument, its data, and some recent results", *Lagrangian Analysis and Prediction of Coastal and Ocean Dynamics*, A. Griffa, A.J. Mariano, A.D. Kirwan, T. Ozgokmen, and H.T. Rossby (editors), Cambridge Univ. Press, Cambridge, pp. 39–67, 2007.

Maltrud, M.E. and J.L. McClean, "An eddy-resolving global 1/10° ocean simulation", *Ocean Modell.*, 8, pp. 31–54, 2005.

Oschlies, A., "Improved representation of upper-ocean dynamics and mixed-layer depths in a model of the North Atlantic on switching from eddy-permitting to eddy-resolving grid resolution", *J. Phys. Oceanogr.*, 32, pp. 2277–2298, 2002.

Pasquero, C., A. Bracco, A. Provenzale, and J.B. Weiss, "Particle motion in a sea of eddies", *Lagrangian Analysis and Prediction of Coastal and Ocean Dynamics*, A. Griffa, A.J. Mariano, A.D. Kirwan, T. Ozgokmen, and H.T. Rossby (editors), Cambridge Univ. Press, Cambridge, pp. 89–118, 2007.

Rhines, P.B., "Geostrophic turbulence", *Ann. Rev. Fluid Mech.*, 11, pp. 401–440, 1979.

Scott, R.B., B.K. Arbic, E.P. Chassignet, A.C. Coward, M. Maltrud, W.J. Merryfield, A. Srinivasan, and A. Varghese, "Total kinetic energy in four global eddying ocean circulation models and over 5000 current meter records", *Ocean Modell.*, 32, pp. 157–169, 2010.

Smith, R.D., M. Maltrud, F.O. Bryan, and M.W. Hecht, "Numerical simulation of the North Atlantic Ocean at 1/10°", *J. Phys. Oceanogr.*, 30, pp. 1532–1561, 2000.

Stammer, D., "Global characteristics of ocean variability estimated from regional TOPEX/Poseidon altimeter measurements", *J. Phys. Oceanogr.*, 27, pp. 1743–1769, 1997.

HPCMP UGC 2011

5. Signal/Image Processing (SIP) and Sensors; Electronics, Networking, and Systems/C4ISR (ENS) and Testing

Signal/Image Processing (SIP)

General Purpose Computing on Graphics Processing Units: Decomposition Strategy

Henry Au, Gregory Lum, and Ronald Thompson
US Space and Naval Warfare Systems Center Pacific (SSC Pacific), Pearl City, HI
{henry.au, gregory.lum, ronald.thompson}@navy.mil

Abstract

This paper describes the decomposition strategy required when porting traditional C/C++ algorithms which run on CPU's to run on the parallel processing architectures that are found on Graphics Processing Units (GPUs). Parallel programming architectures, such as Compute Unified Device Architecture (CUDA), are explored with focus placed on code segmentation for GPU porting. The Tuning and Analysis Utilities (TAU) tool created by the University of Oregon is used to provide a top-level code breakdown to help with code decomposition. To provide an image processing context for algorithm decomposition and segmentation, the Adaptive Linear Filter (ALF) is used. Video from a Heavy Weapon Thermal Sight (HWTS) can be easily obstructed by fog, haze, smoke, or dust clouds resulting in poor visibility. In order to enhance the video, signal processing techniques such as the ALF are performed. However, algorithms such as the ALF require large amounts of data to be analyzed, thus limiting the picture quality and size of the video if real-time image processing is desired. By exploiting the computational power of GPUs to process the video, a speed-up of at least 100x is expected. This will enable the processing of higher resolution video in real-time to improve the Warfighter's ability to detect and identify objects in low-visibility conditions. The ALF performance results using the GPU are also reported. GPU bottlenecks and limitations are briefly discussed in this paper as well, as not every algorithm will benefit from execution on parallel processing architectures.

1. Introduction

Graphics Processing Units (GPUs) have been in use in one form or another to display information to users since the 1980's. GPUs continued to evolve from simple shape accelerators to more complex computations such as three-dimensional (3D) rendering. GPUs continued to become more and more robust in order to meet the demanding calculations normally reserved for the Central Processing Unit (CPU). However, only as recently as 2007 did General Purpose Computing on Graphics Processing Units (GPGPUs) become a viable option for high performance computing. This availability is due to NVIDIA's Compute Unified Device Architecture (CUDA). CUDA has provided a lot of the back-end coordination required for managing the hundreds of parallel cores found on their GPUs, allowing programmers to focus on exploiting this computational power. Another benefit of GPGPU is the ease with which GPUs can be added or upgraded to a standalone desktop machine at a cost of a few hundred dollars for increased performance.

Using the large number of cores available on a single GPU, a desktop computer or even a laptop can become a mobile high performance computing (HPC) device. This makes it ideal for military applications where mobility, package size, and energy requirements are important factors. Remote drones or unmanned aerial vehicles (UAVs) suddenly become possible applications. With a GPU installed on a UAV, data can be processed in near-real-time on the aircraft instead of post processing the data at a remote site. This will improve the accuracy of the data being sent to the Warfighter that relies on time-sensitive information.

The parallel nature of GPU data processing makes it an ideal tool to improve execution time of signal and image processing. Figure 1 shows how a set of data is segmented and processed in parallel using blocks and threads. A thread is a set of operations that processes data independent of order, thus allowing for parallel execution. Multiple threads create a block and multiple GPU cores process multiple blocks at the same time. With this architecture, it can be easily seen that additional cores results in more data being processed in parallel; thus, overall computational time is reduced. This makes it more efficient than a CPU that processes data sequentially. However, there are limitations associated with GPGPU due to the fundamental differences between CPU and GPU cores. The CPU core is much more robust and faster, enabling it to

handle a wider range of tasks when compared to a GPU core. However, since CPU processors have orders-of-magnitude fewer cores than GPUs, when dealing with highly parallel computations the GPU outperforms the CPU in floating point operations per second (FLOPS). As will be discussed later, GPGPUs do have fundamental limitations.

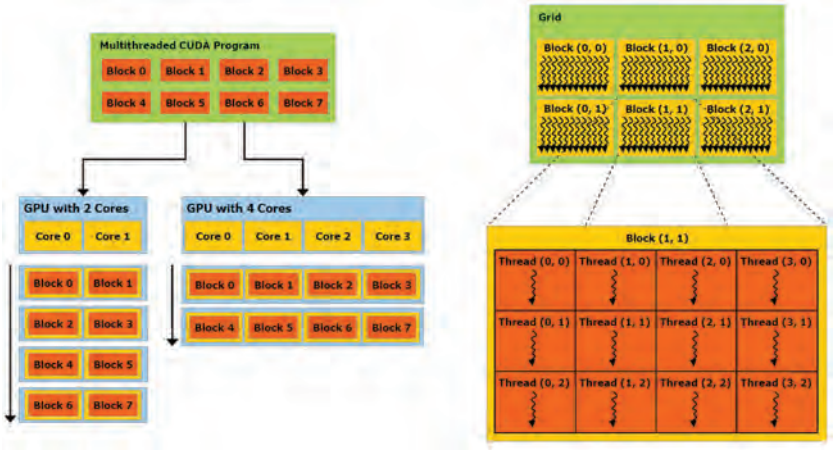


Figure 1. Parallel computation of data

Figure 2 shows the interaction between GPU, CPU, and memory. Because of the CPU's wide range of abilities, it is still the main controller and determines what the GPU computes. Data comes from the main memory to the onboard memory of the GPU. Here the CPU instructs how the GPU should process the data. When the GPU has finished processing the data it is then sent back to the main memory and this process continues for the entire application.

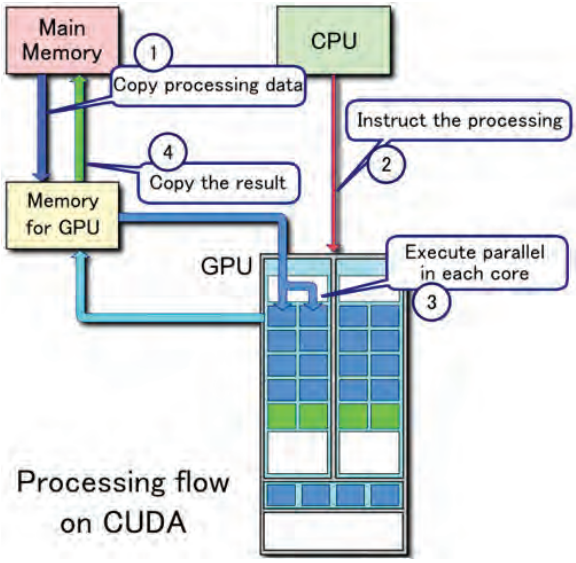


Figure 2. GPU, CPU, and memory interaction

Amdahl's and Gustafson's Laws represent a mathematical representation of the amount of computational speed-up possible when using a multi-processor system. They are listed together because it has been shown by Shi (1996) that they are, in fact, two ways of arriving at the same answer. Amdahl's Law (1996) predicts the speed-up to be:

$$\text{Speed up} = \frac{1}{\beta_a + \frac{1-\beta_a}{P}},$$

and Gustafson's Law (1996) predicts the speed-up to be:

$$\text{Speed up} = P - (P-1)\beta_g ,$$

The two equations are related by

$$\beta_a = \frac{1}{1 + \frac{(1-\beta_g)P}{\beta_g}}$$

Where,

P = Number of processors

β_a = Ratio of processor-scaled serial processing time to total processing time

β_g = Ratio of non-processor-scaled serial processing time to total processing time

These laws represent a performance upper bound that can be achieved by using a multi-core system. In order to show the performance increase through use of GPUs, the Adaptive Linear Filter (ALF) algorithm was selected as a test bed. It is ideally suited for parallel computations as it deals with large amounts of video data and functions which perform the same mathematical operations numerous times. In its current form, the ALF algorithm is able to run at near-real-time speeds. However it can only process a low-resolution monochrome video feed.

With a general understanding of how GPUs can reduce processing time, the rest of the paper will cover the decomposition and porting strategy. Also, the GPU and CPU execution times will be compared and performance bottlenecks discussed.

2. Methodology

In order to utilize the computational power of GPUs, a standalone desktop was built with the following hardware: an Intel Core i7, 8GB of DDR3 RAM, 7,200 RPM hard drive, 1,000W power supply, and two NVIDIA GTX 460 graphics cards (one for computer display and the other dedicated for processing). Software utilized included Windows 7 64-bit Pro, CUDA Toolkit 4.0 May 2011 release, University of Oregon's Tuning and Analysis Utilities (TAU) 2.20 timing software, and Microsoft Visual Studio 2008 Professional.

Figures 3 and 4 describe the CUDA-enabled NVIDIA GTX 460 graphics card used for parallel processing. It should be noted that NVIDIA offers much higher performance GPUs dedicated for general purpose computing, such as the Fermi-based GPGPUs. The GTX 460 was selected due to its lower price-point. Also, this card has been selected to be used as the baseline benchmark so that future progress using different NVIDIA graphics cards can be compared. It can be seen in Figure 4 that memory transfers between the CPU and GPU are costly due to the high latency of the GTX 460.

```
Device 0: "GeForce GTX 460"
  CUDA Driver Version / Runtime Version          4.0 / 4.0
  CUDA Capability Major/Minor version number:    2.1
  Total amount of global memory:                 993 MBytes (1041694720 bytes)
  ( 7) Multiprocessors x (48) CUDA Cores/MP:     336 CUDA Cores
  GPU Clock Speed:                               1.44 GHz
  Memory Clock rate:                             1800.00 Mhz
  Memory Bus Width:                              256-bit
  L2 Cache Size:                                 524288 bytes
  Max Texture Dimension Size (x,y,z)             1D=(65536), 2D=(65536,65535), 3D=(2048,2048,2048)
  Max Layered Texture Size (dim) x layers        1D=(16384) x 2048, 2D=(16384,16384) x 2048
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 32768
  Warp size:                                     32
  Maximum number of threads per block:           1024
  Maximum sizes of each dimension of a block:    1024 x 1024 x 64
  Maximum sizes of each dimension of a grid:     65535 x 65535 x 65535
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and execution:                 Yes with 1 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Concurrent kernel execution:                   Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support enabled:                No
  Device is using TCC driver mode:               No
  Device supports Unified Addressing (UVA):      No
  Device PCI Bus ID / PCI location ID:           2 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >
```

Figure 3. NVIDIA GTX 460 GPU specifications

```
Device 0: GeForce GTX 460
Quick Mode

Host to Device Bandwidth, 1 Device(s), Paged memory
  Transfer Size (Bytes)        Bandwidth(MB/s)
  33554432                     2890.2

Device to Host Bandwidth, 1 Device(s), Paged memory
  Transfer Size (Bytes)        Bandwidth(MB/s)
  33554432                     2981.0

Device to Device Bandwidth, 1 Device(s)
  Transfer Size (Bytes)        Bandwidth(MB/s)
  33554432                     62925.1
```

Figure 4. NVIDIA GTX 460 GPU bandwidth specifications

The ALF algorithm was profiled using TAU 2.20, with special consideration taken to ensure that functions which occur repeatedly, such as *for loops*, were profiled. The timing information from the ALF TAU profile was then used to develop a decomposition strategy which focused on portions of code which can become bottlenecks in the event of larger data sets being processed. It is important to keep in mind that profiling a system that currently operates at near-real-time through CPU processing may not reveal bottlenecks based only on timing. It is therefore important to understand the algorithm being profiled as well as the timing profile created using TAU. Figure 5 below depicts the ALF algorithm processing a 640×480 resolution image. The red circles below point out how the ALF algorithm has enhanced the original image so that dust created by the helicopter flying by is filtered.

Figure 5. Left ALF filtered image vs. right original image

The parallelizable code is then ported and another timing profile is conducted to determine the increased performance. As mentioned earlier, memory transfers between CPU and GPU have high latency. Thus, the overhead associated with the use of CUDA is also measured and evaluated.

3. Results and Discussion

3.1 Decomposition & Porting Strategy

The first phase of the decomposition strategy identified all *for loops*, which repeated multiple times and were self-contained (in the sense that there were no other calls to other functions within the *for loop*). Ideally, the computations within the *for loop* would be highly mathematical in nature since GPUs were designed with mathematical processing in mind. Figure 6 depicts the TAU profiling results. An image with resolution selected to be 640×480 pixels with 14 bit depth was profiled, and the results are presented on the left. A 14 bit 1,920×1,440 pixel image was also profiled, and those results are presented on the right. It can be seen that a timing profile of the algorithm operating under normal data loads may not reveal functions which are bottlenecks or worth porting. Take, for example, Function49 ALSF boxed in red shown in Figure 6. When processing the 640×480 resolution image, Function49 ALSF contributes a time of 0.412ms. When a 1,920×1,440 image is being processed, a run-time of 9ms results. Later it will be discussed why a function having a total execution time of 0.412ms would be ignored for porting.

A *for loop* with relatively large execution time using conditional statements operating on a 640×480 resolution image was identified using the TAU profile and ported to the GPU realm. The function is within PreFilter ALF, boxed red in Figure 6 above. Although not mathematically intensive, it is still very parallel in nature. The second phase of the decomposition strategy then used high-precision C++ timers to determine the execution time of the *for loop*. This timing benchmark helps provide the baseline for comparison to determine the performance gain when the function is ported for GPU execution. The results are presented in the following section.

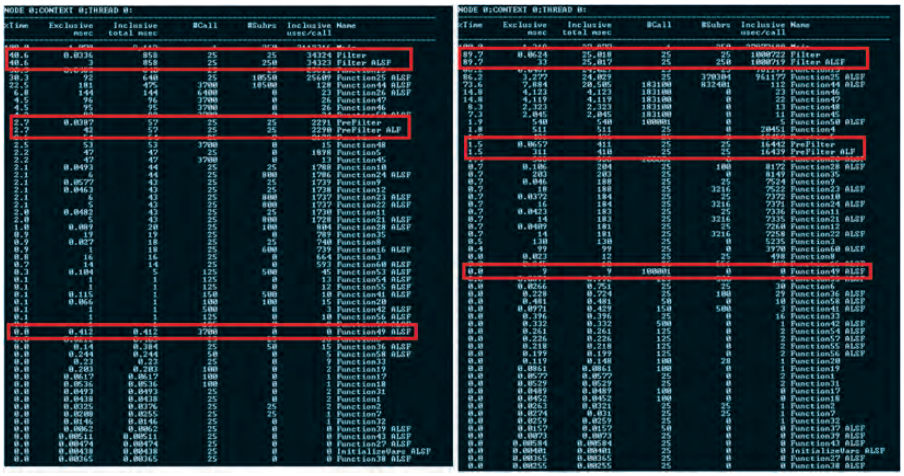


Figure 6. TAU profiling of ALF algorithm (Left: 640×480, Right: 1,920×1,440)

Having ported the *for loop* within PreFilter ALF, the overhead associated with the ported GPU call was measured and described in Table 1. Careful consideration should be taken to ensure that when porting conditional structures within *for loops* that no case exists in which divergence within the code could occur. If diverging code is not addressed and is executed on the GPU and a single thread becomes stuck, the entire program will wait for the single thread to finish execution, and thus enter a continual wait with no timeout.

Table 1. CUDA overhead associated with GPU ported PreFilter

Memory Size	cudaMalloc	cudaMemcpy (Host to Device)	cudaMemcpy (Device to Host)	CudaFree	~ Overhead
640x480x2Bytes (614KB)	0.048 ms	0.262 ms	0.279 ms	0.224 ms	0.813 ms/614 KB
1280x960x2Bytes (2.46MB)	0.066 ms	0.783 ms	0.893 ms	0.196 ms	1.938 ms/2.46 MB
1920x1440x2Bytes (5.53MB)	0.080ms	1.812 ms	2.002 ms	0.229 ms	4.123 ms/5.53 MB

The approximate overhead in processing time associated with the two image resolutions were used to determine other functions which would benefit from porting to GPU. If a portion of code is classified as being parallel in nature and profiled to have an execution time greater than the required overhead, it is highly likely that when ported to the GPU a performance increase will be achieved.

Based on the expected overhead of 0.8ms, one would immediately determine that porting Function49, seen in Figure 6, to the GPU would only cause a decrease in performance. However, this is not the case when Function49 operates on a larger data set since the overhead is 4ms but the execution time required is 9ms. Thus, it is very important to keep in mind the size of the data set being processed. Because higher resolution video processing is desired it would be beneficial to port Function49 to the GPU.

3.2 CPU vs. GPU Processing

Two versions of the PreFilter ALF *for loop* were coded for porting. One version utilized conditional operators and the other used a conditional structure. These two versions were compared to determine which type of operation requires a higher latency. As noted earlier, the best achieved GPU performance occurs when performing highly mathematical computations.

Figures 7 and 8 depict the results of the ported PreFilter ALF. It can be seen in Figure 7 that the execution of the ALF using the GPU results in less time required to perform the calculation.

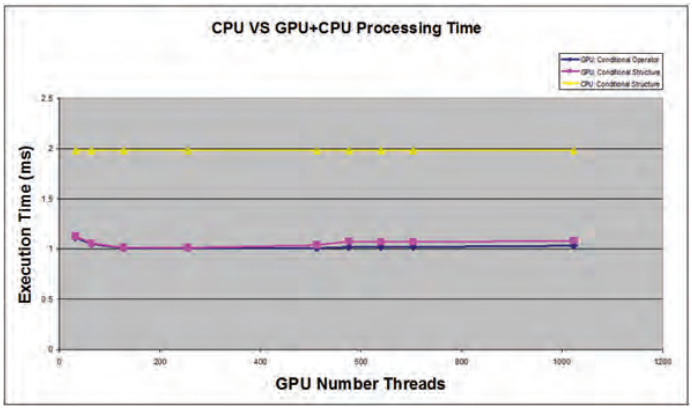


Figure 7. CPU and two versions of GPU+CPU ALF execution time for various number of GPU threads

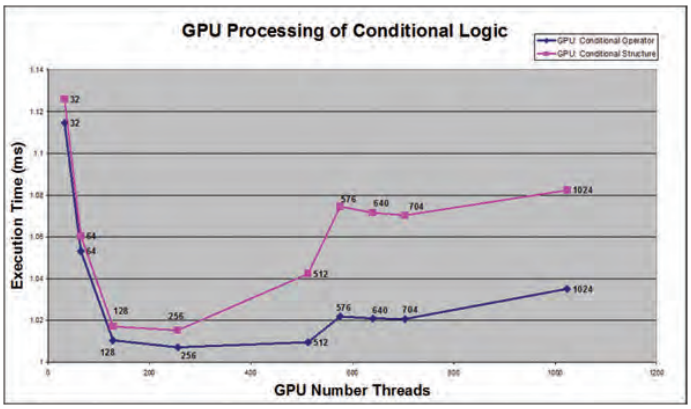


Figure 8. Two versions of GPU+CPU ALF execution time for various numbers of threads

Figure 8 is a close-up of the execution time required for the two versions of GPU ported code. It can be seen that the conditional operator provides slightly better performance, and that both GPUs are thread-dependent thus revealing another limitation of the GPU. The best performance is achieved when using the GTX 460 NVIDIA graphics card and 256 threads per block. This is a result of the available shared memory among each thread within a block. Figure 9 depicts the ALF algorithm processing in terms of frames per second (FPS). As expected, as the resolution increases the required processing time increases, thus decreasing the number of images processed per second. It can be seen that the overall ALF algorithm benefited with decreased processing time by porting a portion of the PreFilter ALF.

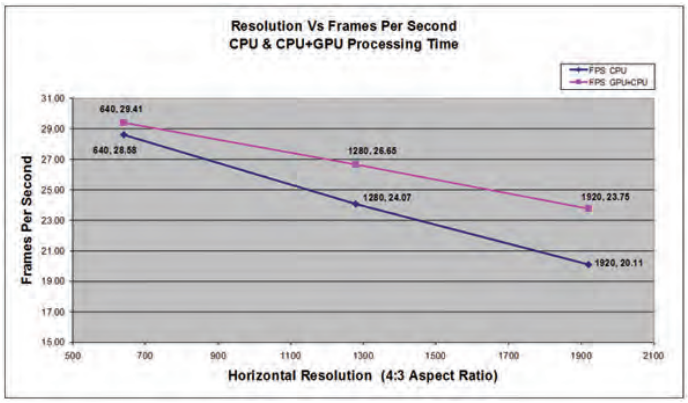


Figure 9. CPU ALF FPS & GPU+CPU ALF FPS at different image resolutions

Table 2 describes the speed-up of the ported portion of the PreFilter ALF function. It can be seen that the performance gain is quite linear, with an approximate speed-up of 2x achieved with the three different resolutions. However, one might quickly jump to the conclusion that a speed-up of 2x would be achieved when timing the ALF algorithm as a whole. But, as Table 3 describes, this is not the case. Due to the structure and nature of the ALF algorithm the PreFilter ALF is only a small portion of the total execution time. Thus the total execution time benefits from the decreased PreFilter ALF execution time, resulting in increased frames per second, but not the speed-up of 2x as might be initially concluded.

Table 2. Ported portion of ALF PreFilter execution time

Resolution	ALF Pre Filter: CPU (ms)	ALF Pre Filter: GPU+CPU (ms)	ALF Pre Filter: Speed-Up
640 × 480	1.894	1.007	1.88
1280 × 960	8.549	4.522	1.89
1920 × 1440	16.722	9.098	1.84

Table 3. Complete ALF algorithm execution time

Resolution	ALF: CPU (ms)	ALF: GPU+CPU (ms)	ALF: CPU (FPS)	ALF: GPU+CPU (FPS)	Speed-Up (FPS)
640 × 480	34.98396	34.00713	28.58	29.41	0.83
1280 × 960	41.54935	37.52175	24.07	26.65	2.58
1920 × 1440	49.72224	42.09779	20.11	23.75	3.64

4. Conclusion

This paper has demonstrated that GPGPUs can and will improve the speed at which large amounts of data can be processed. However, it has also been demonstrated that when requiring memory transfers between the CPU and GPU, and vice-versa, high-latency can greatly degrade performance. Thus, developing a decomposition strategy in conjunction with benchmarked CUDA overhead requirements will help develop a porting strategy to utilize the computational power available through GPUs. This paper has shown that a speed-up of ~2x is achieved for a ported portion of the ALF algorithm, and a desired ALF algorithm speed-up in FPS achieved. It can then be expected that as more functions are ported and benchmarked, the expected 100x speed increase will be achieved.

GPGPUs should not be thought of as the end-all technology to completely replace HPC centers. As described previously, the GPUs have shared thread memory limitations and high-latency associated with memory transfers. For specific applications GPUs can be a viable option alongside large HPC Centers. The GPU's size-to-performance makes it the ideal technology for a range of military applications such as UAV onboard data processing and mobile HPCs for forward operating bases.

Acknowledgements

This work has been funded by SPAWAR System Center Pacific Internal Applied Research (IAR) program (Dave Rees, Program Manager). We would like to thank Dave Buck for allowing us to use his ALF algorithm and Bill Morgart for his expertise in the algorithm. As well, if not for the support from our Department Manager, George McCarty, Code H Business Deputy, Neal Miyake, H5600 Division Head, Alan Umeda, H5603 Deputy for Business Development, Wesley Yamamoto and H56D0 Branch Head, and Justin Lee, we would not have had this great opportunity. Last, and definitely not least, we would like to thank Nick Kamin and Dr. Randy Shimabukuro for mentoring us throughout the course of this research.

References

Amdahl, G., "Validity of the single processor approach to achieving large scale computing capabilities", *AFIPS Spring Joint Computer Conference*, 1976.

CUDA C Best Practices Guide Ver 4.0, 5/2011.

Gustafson, J., "Reevaluating Amdahl's Law", *Communications of the ACM*, Vol. 31 Number 5, May 1988.

NVIDIA CUDA Programming Guide Ver 4.0, 5/6/2011.

Shi, Y., "Reevaluating Amdahl's Law and Gustafson's Law", Temple University, October 1996.

GPU Accelerators for Portable Radar Data Processing

C.T. Fallen, B.V.C. Bellamy, G.B. Newby, and B.J. Watkins
Arctic Region Supercomputing Center (ARSC), University of Alaska Fairbanks (UAF), Fairbanks, AK
{ctfallen, bvbellamy, gbnewby, bjwatkins}@alaska.edu

Abstract

Radar is an important data-intensive technology limited by computational capabilities in remote environments. Particularly, high-resolution visualization of data from the phased-array Modular UHF Ionosphere Radar (MUIR) located at the Department of Defense (DoD) High-frequency Active Auroral Research Program (HAARP) facility in Alaska is not fast enough to be used on-site during experiment campaigns. General-purpose supercomputers can reduce the time-to-solution for typical radar data processing tasks, but those resources are not accessible in many problem domains, including remote field sites, where data loses value with time. Portable special-purpose data processing systems equipped with field-programmable gate arrays (FPGAs) can achieve supercomputing-class performance. However, large-scale FPGA-accelerated systems are expensive.

Our objective is to process high-resolution MUIR data in near-real-time using a portable, low-cost workstation equipped with commodity graphics processing units (GPU)-accelerator hardware and freely available software. The time required to process the data is determined by the speed at which each of a large number of parallel calculations can be completed in sequence: array multiply, discrete Fourier transform, and maximum-find.

We achieved a ~90× speedup factor over single-threaded processing of actual MUIR experiment data using a Linux workstation from Penguin Computing, two GeForce GTX 480 accelerators, and an OpenCL implementation of the one-dimensional fast Fourier transform (FFT). The GPU performance speedup is more than an order-of-magnitude greater than that achieved with the system CPU and OpenMP or the Matlab Parallel Toolbox. Real-time high-resolution radar data products can be made accessible to DoD and affiliated researchers during HAARP experiment campaigns with an inexpensive GPU-accelerated workstation, allowing more efficient use of experiment time.

1. Introduction

1.1 Radar

The term *radar* refers to equipment and techniques used for the radio detection and ranging of objects, originally developed in the 1920s by E.V. Appleton and M.A.F. Barnette of Cambridge University for the purpose of detecting ionosphere layer heights. Major efforts to refine radar technology in the late 1930s were motivated by wartime needs to locate and track distant metallic objects such as aircraft. Modern uses of radar incorporate a variety of sophisticated techniques to transmit radio waves—electromagnetic radiation with frequency typically between 30 kHz and 300GHz—and then receive the radiation scattered by one or more targets. Radar is now used to remotely sense the shapes of objects, the speed and direction of atmospheric winds, the type and magnitude of atmospheric precipitation, ionosphere plasma density and temperature, and the distribution of aboveground and underground structures.

The Modular UHF Ionosphere Radar (MUIR) [Oyama, et al., 2006] is a phased-array radar at the Department of Defense (DoD) High-frequency Active Auroral Research Program (HAARP) facility in Alaska (Figure 1) that can use pulse modulation techniques for high-range resolution operation. MUIR is a diagnostic radar that detects strong ionosphere plasma waves driven by the HAARP Ionosphere Research Instrument (IRI), a powerful phased-array HF transmitter capable of producing ionosphere temperature and density irregularities.

Figure 1. (a) The HAARP ionosphere research instrument (IRI) is collocated with (b) the Modular UHF Ionosphere Radar (MUIR). A workstation at ARSC (b) equipped with two NVIDIA GTX 480 GPU accelerators (indicated with arrows) processed high resolution coded-long pulse data from MUIR.

In the *standard long-pulse mode*, MUIR transmits a 996 μs UHF pulse—corresponding to a range resolution of ~150 km—and records the received signal with a sample rate of 250 kHz. At this sample rate, the range resolution can nominally be improved to ~600 m by using a "coded long-pulse" [Sulzer, 1986] where the phase of the pulse is modulated with a specified pattern of 4 μs bits. Pulse modulation (or *pulse compression*) techniques were patented in the early 1950s as a method to increase radar sensitivity by lengthening the transmitted pulse without sacrificing range resolution [Cook, 1960]. However, the increased range resolution is attained at the cost of increased signal processing. Pulse compression techniques generally involve modulating either the frequency or the phase of the transmitted pulses, then correlating the modulation pattern with the received pulse before proceeding with spectral processing.

The correlation times of the target plasma waves are typically greater than the baud length but less than the time between successive radar pulses (the *inter pulse period*, or IPP), so processing the coded long pulse data essentially requires calculating the spectra of every lag self-product of each sampled pulse. For each ~50 sec of MUIR coded long-pulse operation (10 ms IPP and 1,100 complex-valued 600 m range-bin samples per pulse), the processing effort to generate typical range-time-intensity (RTI) images is roughly equivalent to performing 11 million 1,100-point complex-array multiplies and fast Fourier transforms (FFTs). This can be a prohibitive task at a remote or mobile facility with limited computational resources and data transfer rates. Consequently MUIR high range-resolution measurements are not typically available during HAARP experiment campaigns, slowing the pace of research and increasing the cost of results.

Space weather is relatively unpredictable and affects ionosphere experiment conditions, sometimes scuttling results (There are significant energy and labor costs associated with repeating experiments that each last approximately one hour.) Variable space weather occasionally contributes to remarkable observations. Figure 2 shows an example of MUIR coded long pulse measurements made during exceptionally bright artificial auroral airglow emissions. The radar-detectable plasma turbulence clearly corresponds to regions of intense airglow emissions. However, the high-resolution radar results from this experiment were not available until several weeks following the conclusion of the campaign due to the processing load. In this paper, we describe our initial efforts to use graphics processing units (GPU)-accelerator cards in workstation-scale machines to accelerate MUIR coded long-pulse data processing. We tested a Linux workstation equipped with two GPU accelerators and used the radar data plotted in Figure 2 as a test collection for measuring processing performance.

1.2 Graphics Processing Units

Graphics processing units (GPUs) are specialized processors invented in 1999 by NVIDIA[1]. GPUs were first used to perform calculations for rendering graphics, freeing the main central processing unit (CPU) for non-graphics calculations. Typical two-dimensional (2D) and three-dimensional (3D) graphics calculations compute intensive but parallelizable, involving transformations (e.g., translation, rotation, reflection, and shading) applied to many independent elements (pixels, vertices, segments, and fragments).

The release of general-purpose graphics processing units in combined software and hardware architectures such as the NVIDIA CUDA/OpenCL software application programming interface (API) allowed C or FORTRAN programmers to use massively parallel GPU hardware to speedup non-graphics calculations. Performing general-purpose GPU-accelerated calculations previously required specialized programming languages and detailed knowledge of the graphics hardware. Now, massively parallel computations using, for example, 480 compute cores of the GeForce GTX 480 (Figure 1) or the 448 compute cores of the Tesla M2050, are possible today on commodity desktop systems. Even ~1,000-core calculations are possible on desktop systems accommodating two cards, so systems that require no more than a standard household

[1]http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf

A/C power outlet can achieve peak performance surpassing 2 Teraflops. That is, the peak performance of the fastest supercomputer in 1998 or the 500th fastest supercomputer in 2005 can in 2011 fit in the trunk of a passenger car with room to spare[2].

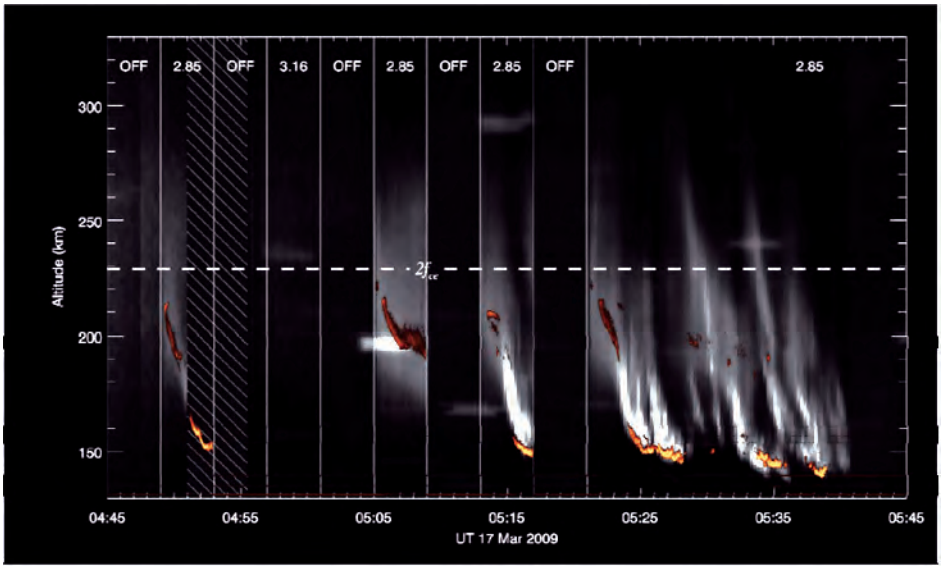


Figure 2. High-resolution MUIR coded long-pulse measurements of field-aligned HF-enhanced ion-line intensity (copper) by Fallen [2010] superimposed over optical measurements of field-aligned artificial airglow (white) by Pedersen, et al. [2010] during an ionosphere modification experiment at HAARP. Without processing, the radar signal would be spread over a range of 150 km.

1.3 Objective

Our objective is to process MUIR coded long-pulse data faster than real-time using a commodity workstation equipped with one or more GPU accelerators. Producing RTI images from coded long-pulse data is a data-parallel and compute-intensive radar processing algorithm, making it an excellent candidate for GPU acceleration. However, radar processing is also data intensive, so bandwidth and other memory restrictions mean that GPU performance may fall far short of the theoretical peak.

The MUIR coded long-pulse processing task consists of three major algorithmic steps applied to a complex-valued matrix representing ~50 sec of baseband radar data: array multiply, discrete Fourier transform, and peak-find. Figure 3 shows an example range-time-intensity (RTI) plot of radar power before and after processing. The image was constructed from one MUIR data file containing coded long-pulse measurements made during a typical HAARP experiment.

Our data-processing application, in single-threaded operation, currently requires approximately 1,200 sec of wall-clock time to process one ~50 sec (~40MB) data file. Multithreaded computation using OpenMP or the MATLAB Parallel Toolbox with a 4-core system results in linear speedup, cutting the processing time to 500 sec per file. Fallen [2010] reports near-real-time processing with multi-node computation using the MATLAB Distributed Computing Server and 32 processor cores on the Arctic Region Supercomputing Center's (ARSC's) *Midnight* system, a Sun cluster of nodes with Opteron processors.

2. Method

2.1 Algorithm

The MUIR coded long-pulse data is recorded in a sequence of HDF5-format[3] (.h5) files containing the single-precision matrix $\mathbf{X} \in \mathbb{C}^{1100\times5000}$ of baseband data and meta-information including the transmitted phase-code sequence. Each column of **X** represents a single received pulse and the rows correspond to 600 m range bins. The algorithm described below is

[2]http://www.top500.org/lists/2010/11/performance_development

[3]http://www.hdfgroup.org/HDF5/

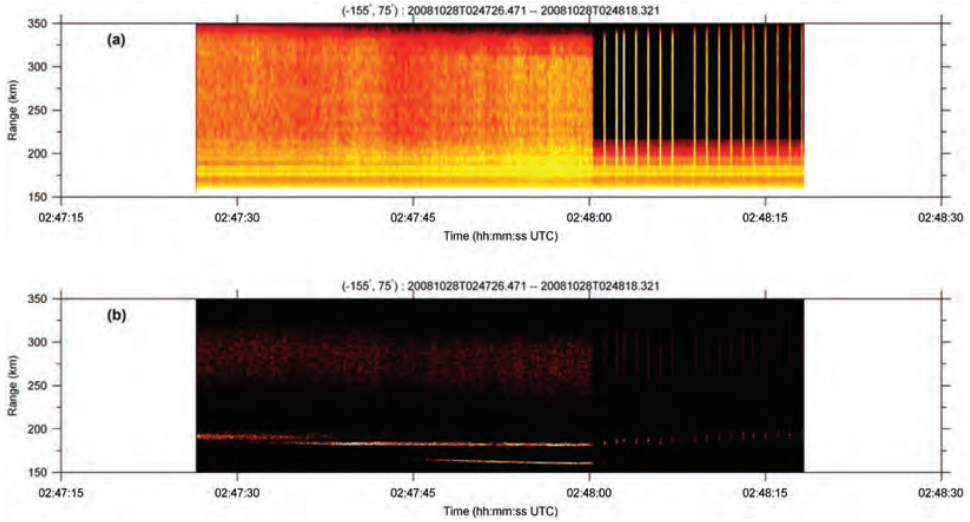


Figure 3. RTI image of one MUIR coded long-pulse data file recorded during a typical HAARP experiment, (a) before and (b) after processing. The HF pump switches from continuous to pulsed operation at 02:48 UTC.

expressed in terms of matrix-vector operations wherever possible, even at the expense of extra memory use or floating point operations, in order to easily leverage accelerated linear algebra libraries. Heuristically, the algorithm outlined by Sulzer [1986] contains three major steps applied to each range bin of each radar pulse: sample the adjacent range bins and multiply by the code, calculate the power spectral density, then analyze the calculated spectra or accumulate with previous pulses.

The sampled pulses first need to be "decoded" before the spectral density is calculated. During this step the signal scattered from the range-bin of interest is isolated from the signals scattered from nearby ranges (recall that the MUIR coded long-pulse is ~150 km wide). Let $\mathbf{C} \in \mathbb{Z}^{1100\times5000}$ be a *modulation matrix* whose columns correspond to those in $\mathbf{X}$ and encode the modulation pattern of each transmitted pulse. The entries are ±1 for 'transmitter on' with a sign change representing a 180° change in phase, 'transmitter off' is represented by 0. MUIR coded-long pulse mode uses a 996 μs pulse width and 4 μs baud length, so each column of $\mathbf{C}$ starts with 249 non-zero entries followed by 851 zeros. While the transmitter modulation can change from pulse to pulse, this is not typical so the columns of $\mathbf{C}$ are identical in this case. Then $\mathbf{C} = \mathbf{c1}^{1100\times5000}$ where $\mathbf{c}$ is an 1,100-element column vector specifying the phase modulation code. The nonzero entries are (reading left to right, top to bottom):

```
++-+--+-+--++-++--+++--+++-+----+---++-++++-++--++---+-----++-+-+++----+++----+++-
+--++---++--++---+++-+++++---+-+-------++-+++-----++--+++---+--+--++---+----++--++
-++++--+++-+++++---+-++++--++++-++++-----+++++--+---+--++++-+++++-+-++---+---+--+
```

To decode the signal scattered from range bin k (zero-index notation), the rows of $\mathbf{C}$ are shifted by pushing k rows of zeros onto $\mathbf{C}$ then taking the array (not matrix) product of the baseband and shifted code matrix. In block matrix notation, a modified permutation matrix $\mathbf{P}$ can be written

$$\mathbf{P}_k = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{matrix} k \\ 1100-k \end{matrix}$$

where $\mathbf{I}$ is an identity matrix with the specified number of rows, and 0 represents corresponding blocks of zeros. The matrix-array products that must be performed before calculating the discrete Fourier transform along each column are

$$(\mathbf{P}_k\mathbf{c1}^{1\times5000}) * \mathbf{X} \rightarrow \mathbf{Y}_k \qquad (1)$$

where * is the array or element-by-element product and the remaining multiplication operations are matrix products. For each data file, we used the GPU to populate the matrix in Equation 1 parentheses directly (without formally calculating the matrix product) and then to calculate the array product.

The rows of matrix $\mathbf{Y}_k$, residing in the GPU memory, represent the decoded baseband signal (time series) scattered and isolated from range bin k and the columns correspond to individual radar pulses. Next, the power spectral density at range

bin k is calculated for each radar pulse. That is, we calculate the discrete (fast) Fourier transform (FFT) of each column of the 1,100×5,000 complex matrix $\mathbf{Y}_k$. The FFT calculation on $\mathbf{Y}_k$ is essentially the matrix product

$$\boldsymbol{\Omega}_N \overset{\text{FFT}}{\times} \mathbf{Y}_k = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{-2\pi i/N} & \cdots & e^{-2\pi i(N-1)/N} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-2\pi i(N-1)/N} & \cdots & e^{-2\pi i(N-1)(N-1)/N} \end{bmatrix} \overset{\text{FFT}}{\times} \mathbf{Y}_k \rightarrow \tilde{\mathbf{Y}}_k \tag{2}$$

where the matrix multiplication $\overset{\text{FFT}}{\times}$ is optimized by using symmetric properties of the transformation matrix $\boldsymbol{\Omega}_N$. To increase the efficiency of the FFT calculation even further, the number of points used in the transform N=1,100 is reduced to the next smallest power of two 1,024=10^2. (Recall that the matrix $\mathbf{Y}_k$ is already padded with zeros.) We used a GPU-accelerated library function to calculate the FFT in Equation 2.

At this point in the calculation, the matrix $\mathbf{Y}_k$ can be deleted from the GPU memory and the matrix $\tilde{\mathbf{Y}}_k$ can be copied back to system memory if the entire complex spectrum from range bin k of each received pulse is to be stored for later analysis. This is a large amount of data to store for several hour-long experiments, so the complex range-resolved time-dependent spectrum is usually processed further then discarded. Since the goal of the benchmark calculation reported here is to produce a plot of radar intensity vs. range and time (RTI plot) similar to Figure 2, the largest frequency component power amplitude from each pulse (column of $\tilde{\mathbf{Y}}_k$) is extracted and then normalized by the mean power of each pulse:

$$\begin{aligned} \operatorname{Re}\tilde{\mathbf{Y}}_k * \operatorname{Re}\tilde{\mathbf{Y}}_k + \operatorname{Im}\tilde{\mathbf{Y}}_k * \operatorname{Im}\tilde{\mathbf{Y}}_k &\rightarrow \mathbf{T}_k \\ \max(\mathbf{T}_k)/\operatorname{mean}(\mathbf{T}_k) &\rightarrow \mathbf{z}_k \end{aligned} \tag{3}$$

The array multiply and matrix addition operations in Equation 3 are performed on the GPU, and the intermediate power spectra are stored on the GPU in the temporary matrix $\mathbf{T}_k$. Finally, array (element-by-element) division is applied to the intermediate row vectors containing the maximum and mean power from each pulse (from each column of $\mathbf{T}_k$). The max, mean, and array division functions are performed on the GPU and the final resulting row vector $\mathbf{z}_k$ containing the intensity of range k for each received radar pulse, is copied to system memory.

For each radar data file, GPU-accelerated calculations in Equations 1, 2, and 3 are completed in sequence for each range bin k, and the intensities stored in $\mathbf{z}_k$ are stacked to form a positive real matrix:

$$\begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_{1100} \end{bmatrix} \rightarrow \mathbf{Z}$$

The entries in matrix $\mathbf{Z}$ are assigned to a (logarithmic) color-scale and plotted on the RTI diagram to form a column approximately 50 sec wide as shown in Figure 2, where results from 71 files have been concatenated in time. Therefore, the visualization of radar data in Figure 2 requires the application of approximately 78,000 iterations of the algorithm described above to a 5,000-pulse data file. Figure 4 shows a diagram of the algorithm execution flow and parallelism across CPU or GPU threads. Section 2.2 below describes implementation details.

Typical filtering operations that may benefit from GPU acceleration have not been considered here. For instance, to eliminate ground clutter effects resulting from radar side-lobes, or to eliminate target features with a correlation time larger than the radar IPP, a two-pulse moving-window difference filter can be applied to the baseband matrix $\mathbf{X}$ before proceeding with calculation Equation 1. Similarly, applying a moving-window filter to integrate the power spectra in $\mathbf{T}_k$ or the peak power values in $\mathbf{Z}$ can help improve signal-to-noise ratio.

2.2 Implementation

We implemented the MUIR data processing application in C++. The sequence of steps (1) through (3) is easily parallelizable since the calculation for each RTI image pixel uses data only from the column containing that pixel and may be performed independently of the calculations for the remaining pixels. The CPU application is parallelized via OpenMP v3.0 (as distributed with GCC v4.5.2), and is accomplished by distributing the algorithm iterations corresponding to each range-bin row in the output RTI image. The GPU application is parallelized via OpenCL, achieved by expressing the calculations for a range-bin row as a sequence of OpenCL kernels, executed for each pixel (element of $\mathbf{X}$).

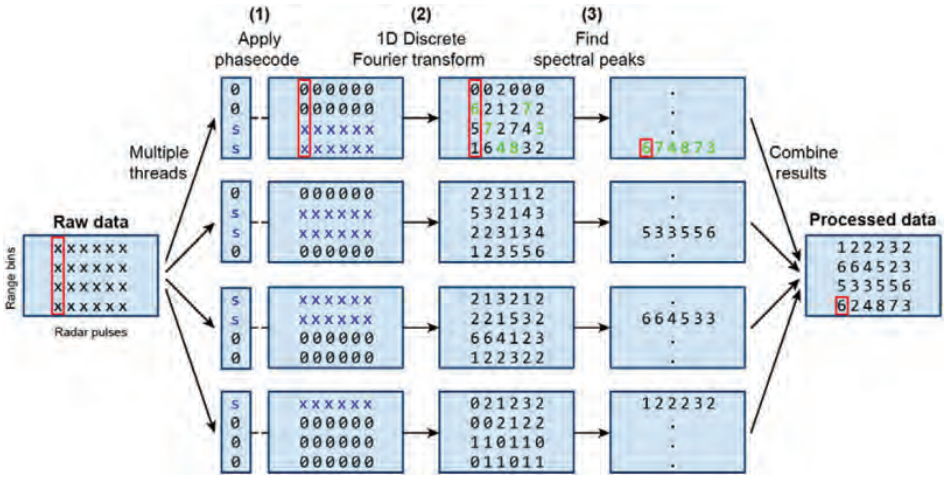


Figure 4. Execution flow and thread parallelism of the high-resolution range-time-intensity (RTI) processing algorithm for MUIR coded long pulse data. Each thread or GPU core operates independently on the columns of the raw data set and the results are collected to produce one row of the final RTI image.

OpenMP is a standard API available in C++ for writing shared-memory parallel applications and is well-suited for multicore architectures. We used the OpenMP *parallel for loop* directive to distribute the algorithm sequence in Section 2.1 to multiple threads; each thread calculated one 5,000-element row of the RTI output image for each MUIR data file. Ad-hoc code performed the phase-code multiplication and spectral peak-finding. The *fftw* v3.2.2 library was used to calculate the Fast Fourier Transform. An OpenMP *critical* construct is required to initialize *fftw* in each thread upon the completion of a row, negatively affecting multithreaded CPU performance. Timing of each algorithmic step was measured and recorded with the timer class from the Boost C++ Libraries v1.45.0.

OpenCL is an API for writing data-parallel applications that can take advantage of a heterogeneous collection of computational resources—such as CPUs and GPUs—installed in a workstation or compute-node platform. We adapted the Section 2.1 algorithm to the OpenCL programming model, which is well-suited for data-parallel tasks. Each of the algorithmic steps was implemented as one or more OpenCL kernels. The FFT kernel used in the experiment is based on the Apple *OpenCL_FFT v1.4* demonstration code[4].

After a data file is read into the host memory, the phase-code vector and complex-valued baseband data matrix are copied to the GPU device global memory. The sequence of kernels is then mapped to the individual elements of the baseband matrix and executed. The final algorithmic step is a reduction operation, and the results are stored in the GPU device global memory. After the kernels have executed, the processing program copies the image matrix from GPU global memory to the system memory, then writes the data to disk. When the application uses two GPUs, two input files are loaded simultaneously and processed in parallel. We are currently exploring the possibility of processing one file with two GPUs.

2.3 System

The TAU system is a single-user workstation made by Penguin Computing operating Red Hat Enterprise Linux Client release 5.6 (Tikanga). It contains a 64-bit 2.4GHz AMD quad-core Opteron CPU and 4GB of memory. TAU is equipped with two NVIDIA GeForce GTX 480 GPU cards. The GTX 480 contains 480 CUDA cores and 1.5GB of dedicated memory with a bandwidth of 177GB/sec. The theoretical peak performance of each GPU is 168 double-precision Gigaflops or 1,350 single-precision Gigaflops. NVIDIA driver version 260.19.36 was installed to allow for an OpenCL v1.0 context.

2.4 Experiment

We defined 71 HDF5-format data files containing CLP measurements (Figure 2) from the MUIR receiver to be a test collection of input data. Each file occupied 44MB on disk and contained a 1,100×5,000 complex-valued single-precision matrix of baseband radar data corresponding to **X** in Section 2.1. Each matrix represents approximately 50 sec of data. TAU executed several GPU-experiment runs and CPU-control runs on the data test collection. Single-precision experiments

[4]http://developer.apple.com/library/mac/#samplecode/OpenCL_FFT/

were executed on both systems using one or two GPU accelerators, for a total of two GPU-accelerated runs. A control experiment was executed using one CPU thread.

For CPU or single-GPU processing, the files were processed serially using OpenMP or OpenCL to divide the computation among the CPU or GPU cores. Two files were processed simultaneously during dual-GPU experiments. The processed data from each input file was saved to disk in .h5 format. Results were verified by visually comparing the resulting images to reference images produced by a separate program and by numerically comparing the respective CPU- and GPU-calculated output matrices.

Each experiment produced 71 output data files containing the radar RTI data along with diagnostic and timing information. The total time to read and process the 71-file test collection in each experiment was recorded with GNU *time* (version 1.7). Additionally, the time for each thread to complete each task for a given range gate and radar pulse was recorded with the *Timer* class from the Boost C++ Libraries[5].

3. Results

Our performance objective is to process MUIR coded long-pulse data in real-time, i.e., to load and process the one-hour test collection and then to save the RTI data in one hour of wall time or less. The MUIR receiver records the baseband data in single precision so we focused on single-precision floating-point calculation performance.

Speedup is measured relative to single-threaded CPU execution time. We verified accuracy of the GPU-accelerated calculations with visual comparisons of the GPU- and CPU-calculated RTI plots and also with numerical comparison of the GPU- and CPU-calculated output matrices. Both the CPU and GPU codes are under development, so additional optimizations are likely available, and may potentially affect the speedup estimates in either direction.

Since the sequence of processing tasks described in Section 2.1 may be parallelized over the radar pulses (columns of data matrix **X**) and range bins (rows of **X**), both OpenMP multithreaded computation and GPU-acceleration are expected to significantly reduce the time to solution. In addition to measuring total processing time, we measured the performance of each algorithm task to identify performance bottlenecks.

Figure 5 shows the wall-time elapsed while processing the one-hour coded long-pulse MUIR data test collection illustrated in Figure 2. The TAU workstation processes one-hour of radar data in 14 hours using single-threaded computation. Approximately 4 hours are required when using 4 threads with OpenMP (one thread per CPU core). Speedup over single-threaded performance increases sub-linearly with increasing CPU threads. However, even with linear speed-up, OpenMP multithreaded computation will require at least 14 CPU cores using the current software to achieve the real-time performance target, more than is typically available in current commercial workstations.

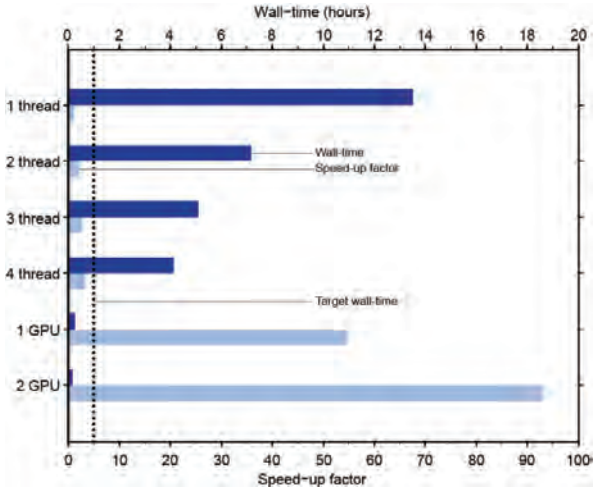


Figure 5. Wall-time (dark blue, top axis) required to process the radar data test collection illustrated in Figure 2 using either CPU or GPU processing. Real-time performance is achieved when the processing time falls below one hour. The speedup factor (light blue, bottom axis) is calculated over single-threaded performance on the CPU. Both measurements include file-system I/O time.

[5]http://www.boost.org/

GPU-accelerated computation meets and significantly exceeds the performance target. The GTX 480 processes the one hour radar data set in less than 15 minutes, corresponding to a speed-up factor of 55× over the measured single-threaded computation and ~16× over 4-thread performance that nominally utilizes the full capability of the CPU. Dual-threaded computation that distributes the processing to both GTX 480 cards installed in the TAU workstation results in a speedup factor of nearly 93×, completing the job in less than 9 minutes. Note that the wall-time and speedup factor measurements reported in Figure 5 include file input and output to disk.

The box plot in Figure 6 illustrates processing performance per thread isolated from file-system input/output (I/O) performance common to both CPU and GPU computations. Vertical red lines indicate the median time required—per thread—to process one range bin of a 5,000-pulse MUIR data file by completing the three tasks of the Section 2.1 algorithm. The GPU measurements include the time required to transfer the baseband radar data matrix to the GPU memory and also the time to transfer the processed row vector back to system memory.

GPU row-completion times exhibit far lower absolute and relative variability than the multithreaded CPU times, consistent with the synchronized nature of GPU calculations. The variability in the CPU times may be the result of inefficient use of the cache and implies that the CPU calculations may benefit somewhat from further optimization. Processing time per thread increases with the number of concurrent threads, accounting for the less than linear speedup observed in Figures 5 and 6.

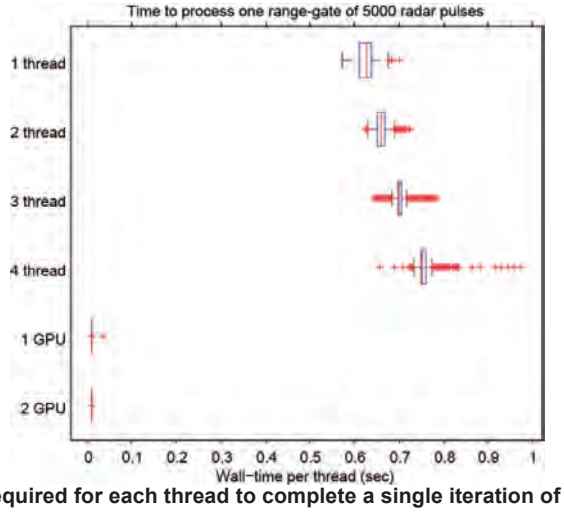


Figure 6. Distribution of wall-times required for each thread to complete a single iteration of the algorithm described in Section 2.1 applied to ~50 sec of radar data (5,000 pulses). The median wall-time is indicated by a vertical red line; the blue box indicates the 25th and 75th percentiles; black whiskers show the minimum and maximum values of typical measurements, outliers are plotted as red points.

Finally, it is worthwhile to briefly examine the execution time and relative speedup of each of the Section 2.1 algorithmic steps. Table 1 shows the time required per processing task per thread for each of the six experiments. Somewhat surprisingly, the phase-code (1) requires significantly more processing time on the CPU as the FFT (2), and we are currently investigating remedies. The CPU performance can likely be improved and will reduce the GPU relative speedup somewhat. The peak-find (3) on the CPU costs approximately one third of the time spent on the complex FFT.

Table 1. Mean wall-time required for each thread to complete the algorithmic tasks from Section 2.1 applied to ~50 sec of radar data (5,000 pulses)

	Phase code (1)		FFT (2)		Peak find (3)	
Experiment	Time (msec)	Speedup	Time (msec)	Speedup	Time (msec)	Speedup
1 thread	524.98	1.00	62.15	1.00	21.92	1.00
2 thread	559.45	1.88	63.54	1.96	22.04	1.99
3 thread	602.04	2.62	73.80	2.53	25.56	2.57
4 thread	647.06	3.25	77.29	3.22	31.78	2.76
1 GPU	0.17	3,087.37	0.90	69.28	8.71	2.52
2 GPU	0.17	6,175.82	0.89	139.42	8.71	5.03

Whereas the time to complete each processing step on the CPU was a fraction of the time spent on the previous step, the respective GPU-accelerated times, although lower the CPU times, increased by a similar fraction with each successive step. That is, the greatest speedup was observed for the phasecode array-multiply operation (possibly due to the unexpectedly poor CPU performance) and the least speedup was obtained for the peakfind operation, with the FFT performance falling somewhere in between. Some speedup differential between (1) and (3) is expected, given the large-scale synchronous computation architecture of GPU accelerators. Array multiply is work that can be evenly distributed to the compute cores, but a peak-finding operation is less efficient due to a load imbalance where some compute cores must traverse their entire input data set while others sit idle.

4. Conclusion and Discussion of DoD Relevance

We demonstrated that an inexpensive, portable workstation equipped with one or more GPU accelerator cards is capable of providing real-time high-resolution radar data products. This new capability has potential for immediate impact to DoD and affiliated researchers during HAARP experiment campaigns. Currently, high-resolution data products from phase-coded radar measurements are otherwise not available until days, if not weeks, following the conclusion of each campaign. MUIR coded long-pulse range-time-intensity data products described here—without access to HPC resources—previously required several hours to process each hour of radar data. The same data product can now be produced in less than 10 min using two off-the-shelf GPU hardware accelerators. Broader applications of this approach include use in portable radar systems as well as remote radar facilities. In particular, we are investigating GPU-accelerated applications for ionosphere incoherent scatter radar (ISR) data processing at remote ISR facilities. Accelerated high-resolution processing with GPU-equipped systems also creates opportunities for real-time adaptive control of radar systems.

Acknowledgements

Chris Fallen and Beau Bellamy thank Jeremiah Dabney and Rob Cermak for TAU system support; Don Bahls for PACMAN system support; and Oralee Nudson for assistance with editing this manuscript. Hardware and HPC resources were supported by a grant from the Arctic Region Supercomputing Center and the University of Alaska Fairbanks.

References

Cook, C.E., "Pulse compression-key to more efficient radar transmission", *Proceedings of the IRE*, 48(3), doi:10.1109/JRPROC.1960.287599, 1960.

Fallen, C.T., *Applications of a Time-Dependent Polar Ionosphere Model for Radio Modification Experiments*, Ph.D. thesis, Dep. of Phys., Univ. of Alaska Fairbanks, Fairbanks, AK, 2010.

Oyama, S., B.J. Watkins, F.T. Djuth, M.J. Kosch, P.A. Bernhardt, and C.J. Heinselman, "Persistent enhancement of the HF pump-induced plasma line measured with a UHF diagnostic radar at HAARP", *J. Geophys. Res.*, 111(A06309), doi:10.1029/2005JA011363, 2006.

Pedersen, T., B. Gustavsson, E. Mishin, E. Kendall, T. Mills, H.C. Carlson, and A.L. Snyder, "Creation of artificial ionospheric layers using high-power HF waves", *Geophys. Res. Lett.*, 37, doi:10.1029/2009gl041895, 2010.

Sulzer, M.P., "A radar technique for high range resolution incoherent scatter autocorrelation function measurements utilizing the full average power of klystron radars", *Radio Sci.*, 21(6), doi:10.1029/RS021i006p01033, 1986.

Introducing Tools for Defining and Operating on Distributed Matrices

Peter G. Raeth
High Performance Technologies, Inc. (HPTi), Wright-Patterson AFB, OH
praeth@hpti.com

Abstract

As the Department of Defense (DoD) prepares to take its last shared-memory high performance computing (HPC) off-line, it is essential for our users to be aware of tools that enable construction of and operation on distributed matrices. A typical trail followed by domain experts as they push the envelope of science is to start with theory. Once theory is brought to a meaningful milestone, a computational model is constructed. As the model reaches a suitable maturity, experimental data is fed to the model to verify correctness. Over time, the model's level of detail is increased. Gradually, the memory required by the model's matrices grows to the point where a given matrix occupies more memory than is available on a single HPC node. In the past, this problem has been mitigated by the existence within DoD HPC of Hawk, a shared-memory machine. However, that machine is scheduled to be taken off-line within the next two years. Therefore, models need to be re-engineered so that their matrices can be distributed across the memory of several computational nodes. Tools exist to facilitate this transition. This paper introduces three such tools (PETSc, Global Arrays Toolkit, and MATLAB). A calculation of the Kronecker Tensor Product demonstrates their use. User Productivity, Enhancement, Technology Transfer and Training (PETTT) provides personnel who are familiar with all three tools and who can assist transition projects.

1. Introduction

Hawk (http://www.afrl.hpc.mil/hardware/hawk.php) is the only shared-memory computer within Department of Defense's (DoD's) high performance computing (HPC) family. The funding for this machine extends beyond the typical four year life cycle, deep into FY12. After that, according to current plans, Hawk's funding is to be terminated. As always, the Technology Insertion (TI)-11/12 selection process will decide what new systems would best meet the largest percentage of documented requirements. The complete portfolio of systems is oriented toward meeting those requirements. For a Hawk-equivalent machine to be purchased, a defensible articulation of the fraction of DoD's workload that is dependent on large-scale shared memory going into the TI-11/12 process needs to be articulated. This would justify the relatively high cost-per-cycle for such a machine.

Assuming that a comparable shared-memory machine will not be purchased, it is important to consider alternative approaches to satisfying large-memory requirements. There is evidence to suggest that HPC nodes keep getting larger, with nodes having as many as 64 cores that share memory. This creates some relief for codes needing shared memory. However, for programs that would take advantage of petascale machines, it is still necessary to consider the matter of distributed memory. There are many tools for making best use of distributed memory. This paper reviews three of those tools: PETSc, Global Arrays Toolkit, and MATLAB with Parallel Computing Toolkit and Distributed Computing Engine. A calculation of the Kronecker Tensor Product illustrates the application of each tool.

We will start with a brief overview of the Kronecker Tensor Product. This will be followed by the design used to create the software. The author implemented the entire distributed Kronecker calculation separately via each of the three tools. Each tool provides its own unique support to the chosen design. A review of each tool's support for this calculation is given. Finally, results and closing thoughts are offered.

2. The Kronecker Tensor Product

As part of his post-doctoral research, Van Horn produced an excellent introduction to the Kronecker Tensor Product. He reports that, "[The Kronecker Tensor Product] can be used to represent block-recursive algorithms. These algorithms include the Fast Fourier Transform, the Hadamard Transform, bit reversal, bitonic sorting, and Strassen multiplication. By expressing these algorithms in tensor product form and manipulating them using the rules of tensor and permutation

algebra, implementations can be derived for uniprocessor as well as parallel and vector processor architectures." Besides these broad applications, I have found that User Productivity, Enhancement, Technology Transfer and Training (PETTT) customers use this calculation in radar signature modeling, image processing, and automatic target recognition. I have also seen this type of matrix used by algorithms that apply pre-conditioners to systems of equations. Because of the requirements expressed by PETTT customers, I chose the Kronecker Tensor Product as a way of illustrating the use of tools for implementing and operating on matrices that are distributed across the memories of discrete and independent computing nodes (distributed matrices).

The Kronecker Tensor Product is defined by K=kron(A,B)=A ! B. K is formed by overlaying sub-matrices onto an empty matrix. Sub-matrices are formed by multiplying an element of A by matrix B, ($S=A_{r,c}$ * B). If one assumes 0-based counting, the upper-left corner of this sub-matrix gets overlaid on K at $K_{r*rB,\ c*cB}$. Therefore, if A is of size (rA × cA) and B is of size (rB × cB) then K is of size (rA*rB × cA*cB).

Weisstein gives a general example in Figure 1.

Consider the matrix direct product of the 2×2 matrix A and the 3×2 matrix B (K=A ! B). The calculation's result is given by the following 6×4 matrix:

$$K = \begin{bmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{11}b_{31} & a_{11}b_{32} & a_{12}b_{31} & a_{12}b_{32} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \\ a_{21}b_{31} & a_{21}b_{32} & a_{22}b_{31} & a_{22}b_{32} \end{bmatrix}.$$

Figure 1. General example of a Kronecker Tensor Product

A specific example is given by Mathworks in Figure 2.

```
A  = 1    2        B  = 1    0        K  = 1    0    2    0
     3    4             0    1             0    0    1    2
                                           3    0    4    0
                                           0    3    0    4
```

Figure 2. Specific example of a Kronecker Tensor Product

Further depth on the Kronecker Tensor Product is provided by Graham, as well as Mathews and Sicuranza.

3. Designing the Distributed Calculation

Because of the nature of our customers' requirements, we make the following assumptions:

- Matrices A and K are large enough to require distribution across the memories of several independent nodes. These matrices are distributed across all nodes in the cluster.
- Matrix B is small enough that a copy can exist on each node.
- All matrices are dense.
- Matrix A has at least as many rows as there are nodes.

Matrix A is evenly distributed in such a way that entire rows exist on the available nodes. Matrix K's allocation is also by rows but the distribution is not necessarily even, as defined by the default distribution of the three tools. Blocks of K are manually allocated in such a way that the entire calculation involving B and a node's block of A's rows can be overlaid on sub-matrices of K that reside entirely on that same node. Thus, if A's rows rt through rw reside on a given node, K rows rt*rB through (rw*rB)+rB-1 reside on that same node. This allows for all sub-matrix operations to be carried out locally. There is no need for communication between nodes when calculating and overlaying K's various sub-matrices.

Each of the three tools provides unique approaches for carrying out the required sub-matrix calculations and overlaying those on a node's block of the K matrix. Essentially, in a very broad sense, each node receives its components of each

matrix involved in the calculation. Then, each node iterates through its local elements of matrix A. Matrix B is multiplied by each element of matrix A. Then, each of the resulting sub-matrices is overlaid on the appropriate portion of matrix K.

Adding to Weisstein's example, let's assume our network contains two nodes. The entire B matrix exists on each node. Node 1 has row 1 for matrix A and rows 1…3 for matrix K. Node 2 has row 2 for matrix A and rows 4..6 for matrix K. This allocation is illustrated in Figure 3.

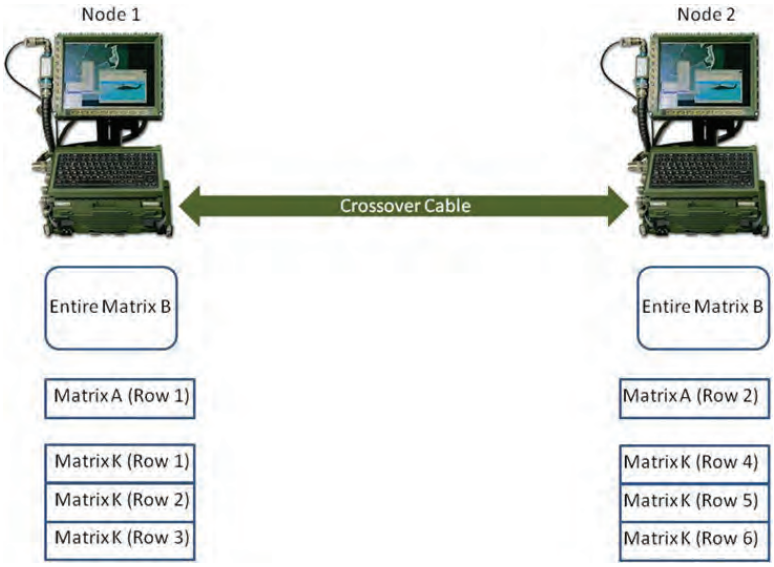


Figure 3. Example allocation of matrices for Kronecker calculation

The calculate/overlay process is illustrated in Figure 4. Each node iterates through its local elements of matrix A, multiplies matrix B by each element, and overlays the resulting sub-matrices onto matrix K. Each node's activities are independent of activities carried out by all other nodes. The supporting tool ensures the global integrity of a distributed matrix.

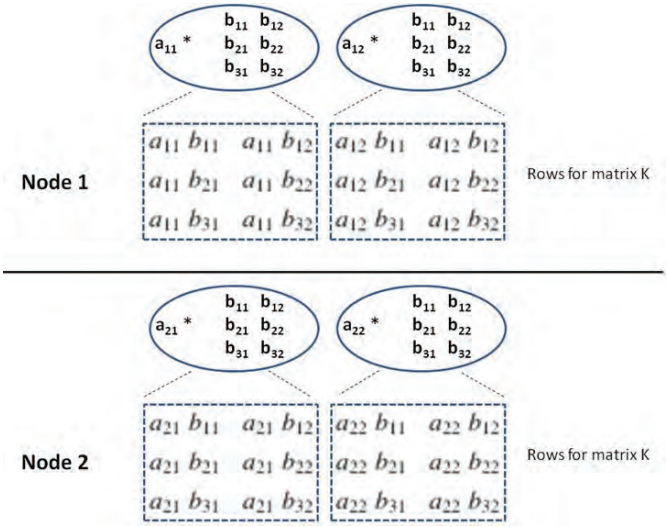


Figure 4. Example iterate/calculate/overlay process carried out in parallel by each node

4. Implementation via Specific Tools

Each tool (PETSc, Global Arrays Toolkit, and Matlab) has its own approach to facilitating a distributed Kronecker calculation. The following sections review the major aspects of how individual tools were employed. The distributed-matrix Kronecker Tensor Product was fully implemented on each of these tools.

Tests with each tool began with data ingest. Each file contained the number of rows, number of columns, and matrix data in row-major order. The A and B matrices were used to produce a new calculation of the K matrix. Calculated and groundtruth K matrices were compared to ensure they were the same.

4.1 PETSc (http://www.mcs.anl.gov/petsc/petsc-as)

According to the website, PETSc is a suite of data structures and routines for parallel solutions of scientific problems modeled by partial differential equations. The Message Passing Interface (MPI) standard defines the inter-process communications infrastructure. PETSc contains a layer that provides general support for distributed matrices. It is this part that we employed for the Kronecker calculation.

4.1.1 Data Ingest

For distributed matrices A and K_ground_truth, the supervisor process opens the file, reads the number of rows and columns for the matrix, and closes the file. The number of rows and columns are sent to all other processes. Each process participates in the creation of distributed matrices:

```
Mat matrix;
MatCreateMPIDense(PETSC_COMM_WORLD, PETSC_DECIDE, PETSC_DECIDE, Rowsize,
                  Colsize, PETSC_NULL, &matrix);
```

Each process queries for the component of the matrix it holds:

```
MatGetOwnershipRange(matrix, &myStartRow, &myEndRow);
```

Each process uses MPI to read its portion of the matrix. This process uses the range of rows held by the process to read each row:

```
PetscMalloc(Colsize * sizeof(PetscInt), &ColIndices);
for(index=0; index < Colsize; index++) ColIndices[index]=index;
input_values=(double*)malloc(Colsize * sizeof(double));
PetscMalloc(Colsize * sizeof(PetscScalar), &values);
MPI_File_open(MPI_COMM_WORLD, "filename", MPI_MODE_RDONLY, MPI_INFO_NULL, &file);
MPI_File_seek(file, (sizeof(long) * 2) +
              (sizeof(double) * myStartRowK * Colsize), MPI_SEEK_SET);
for(row=myStartRow; row < myEndRow; row++)
{
  MPI_File_read(file, input_values, Colsize, MPI_DOUBLE, &status);
  for(col=0; col < Colsize; col++)
    values[col]=(PetscScalar)input_values[col];
  MatSetValues(matrix, 1, &row, Colsize, ColIndices, values, INSERT_VALUES);
}
MPI_File_close(&file);
free(input_values);
PetscFree(values);
PetscFree(ColIndices);
```

Once the distributed matrix is populated, each process participates in ensuring a global view:

```
MatAssemblyBegin(Kron_Groundtruth, MAT_FINAL_ASSEMBLY);
MatAssemblyEnd(Kron_Groundtruth, MAT_FINAL_ASSEMBLY);
```

For the non-distributed B matrix, the supervisor process reads the matrix and passes it on to the other processes. Each process creates a PETSc matrix during that activity. The entire matrix creation and population portion of the program generates data for function call to the Kronecker calculation.

4.1.2 Calculating the Kronecker Tensor Product

A function entirely contains the activity each process performs during this calculation.

A local matrix is created to contain sub-matrices as they are created. he creation employs an initialization for the sake of completeness.

```
PetscScalar * Intermediate_values;
Mat Intermediate;
MatCreateSeqDense(PETSC_COMM_SELF, B_Rowsize, B_Colsize, Intermediate_values,
                  &Intermediate
```

```
for(i=0; i < B_Rowsize * B_Colsize; i++)
  Intermediate_values[i]=B_values[i];
MatAssemblyBegin(Intermediate, MAT_FINAL_ASSEMBLY);
MatAssemblyEnd(Intermediate, MAT_FINAL_ASSEMBLY);
```

The matrix to hold the Kronecker result is created next. Since the A and B matrices are inputs to this calculation, the size of the Kronecker matrix can be readily calculated as described above.

```
Mat Kron_Calculated;
MatCreateMPIDense(PETSC_COMM_WORLD, PETSC_DECIDE, PETSC_DECIDE,
                  K_Rowsize, K_Colsize, PETSC_NULL, &Kron_Calculated);
```

Each processor has to find out its local range of matrix rows:

```
MatGetOwnershipRange(A, &myStartRow_A, &myEndRow_A);
```

Finally, the actual calculation takes place. Each sub-matrix is overlaid on its appropriate place in the K matrix:

```
j=B_Rowsize * B_Colsize;
for(A_Row=myStartRow_A; A_Row < myEndRow_A; A_Row++)
   K_Row_Begin=A_Row * B_Rowsize;
   K_Row_End=K_Row_Begin + B_Rowsize;
   r=-1;
   for(row=K_Row_Begin; row < K_Row_End; row++)
   {
     r++;
     RowIndices_set[r]=row;
     }
     for(A_Col=0; A_Col < A_Colsize; A_Col++)
     {
        K_Col_Begin=A_Col * B_Colsize;
        K_Col_End=K_Col_Begin + B_Colsize;
        c=-1;
        for(col=K_Col_Begin; col < K_Col_End; col++)
      {
         c++;
         ColIndices_set[c]=col;
        }
         MatGetValues(A, 1, &A_Row, 1, &A_Col, &scalar);
         for(i=0; i < j; i++)
         Intermediate_values[i]=B_values[i] * scalar;
         MatSetValues(Kron_Calculated, B_Rowsize, RowIndices_set, B_Colsize,
                      ColIndices_set, Intermediate_values, INSERT_VALUES);
     }
}
MatAssemblyBegin(Kron_Calculated, MAT_FINAL_ASSEMBLY);
MatAssemblyEnd(Kron_Calculated, MAT_FINAL_ASSEMBLY);
```

Calculated and groundtruth result matrices are compared using the following statement:

```
MatEqual(Kron_Calculated, Kron_Groundtruth, &oolean);
```

4.2 Global Arrays Toolkit (http://www.emsl.pnl.gov/docs/global)

According to the website, the Global Arrays Toolkit provides a programming interface for distributed-memory computers. This interface allows a shared-memory view of matrices that physically exist on several nodes. Each process in a parallel program can asynchronously access blocks of physically-distributed dense multi-dimensional arrays, without the need for explicit cooperation across processes. Locality information for matrix data is available and direct access to the local portions of shared data is provided. It is very important to note that this particular tool physically stores its data as column-dominant single-dimension vectors. We access that data directly in some cases.

Contrary to popular notions, Global Arrays Toolkit is not a tool for chemists. While this toolkit is indeed used by computational-chemistry codes, the toolkit itself provides general-purpose support for distributed matrices. This support can be employed by codes in any domain that requires matrices whose memory exceeds that of a single cluster node.

4.2.1 Data Ingest

For distributed matrices A and K_ground_truth, the supervisor process opens the file, reads the number of rows and columns for the matrix, and closes the file. The number of rows and columns are sent to all other processes. Each process participates in the creation of distributed matrices:

```
/* Size of each matrix A dimension */
#define numDims 2
read_matlab_file_2d("A.bin", &A_numRow, &A_numCol, NULL);
dimension[0]=(int)A_numCol;
dimension[1]=(int)A_numRow;

/* Use default blocking for each matrix A dimension */
chunk[0]=A_numCol;
chunk[1]=1;

/* Establish matrix A, a regularly distributed array of type double */
A_handle=GA_Create_handle();
GA_Set_data(A_handle, numDims, dimension, C_DBL);
GA_Set_array_name(A_handle, "Array_A");
GA_Set_chunk(A_handle, chunk);
returnCode=GA_Allocate(A_handle);
if(!returnCode) GA_Error("Unable to create Global Array A\n", returnCode);
```

Each process queries for the component of the matrix it holds:

```
int lo_A[numDims];
int hi_A[numDims];
me=GA_Nodeid();
nproc=GA_Nnodes();
NGA_Distribution(A_handle, me, lo_A, hi_A);
```

Each process uses MPI to read its portion of the matrix. This process uses the range of rows held by the process to read each row:

```
int ld[numDims];
int localRow=-1;
void *ptr_local_A;
MPI_File mpi_file;
MPI_Status mpi_status;

NGA_Access(A_handle, lo_A, hi_A, &ptr_local_A, ld);
double *double_ptr_local_A=(double*)ptr_local_A;

MPI_File_open(MPI_COMM_WORLD, "A.bin", MPI_MODE_RDONLY, MPI_INFO_NULL, &mpi_file);
MPI_File_seek(mpi_file, (sizeof(long) * 2) + (sizeof(double) * lo_A[1] *
              A_numCol), MPI_SEEK_SET);

input_values=(double*)malloc(A_numCol * sizeof(double));
localRow=-1;
for(i=lo_A[1]; i <= hi_A[1]; i++)
{
  MPI_File_read(mpi_file, input_values, A_numCol, MPI_DOUBLE, &mpi_status);
  localRow++;
  for(j=0; j < A_numCol; j++)
  {
     index=(j * (hi_A[1] - lo_A[1] + 1)) + localRow;
     double_ptr_local_A[index]=input_values[j];
  }
}

MPI_File_close(&mpi_file);
free(input_values);
NGA_Release_update(A_handle, lo_A, hi_A);
```

For the non-distributed B matrix, the supervisor process reads the matrix and passes it on to the other processes. Each process allocates and populates its own copy of the matrix during that activity. The entire matrix creation and population portion of the program generates data for function call to the Kronecker calculation.

4.2.2 Calculating the Kronecker Tensor Product

A function entirely contains the activity each process performs during this calculation.

Matrix A uses the default and regular row distribution. Since we want the rows of K in physical residence on the same node as their associated rows of A, we employ a non-default irregular distribution.

```
int nblock[numDims]={1, nproc};
int mapc[nproc+1];
dimension[0]=(int)K_numCol;
dimension[1]=(int)K_numRow;
mapc[0]=0;
mapc[1]=0;
for(i=2; i <= nproc; i++)
{
  NGA_Distribution(A_handle, i-2, lo_A, hi_A);
  numRows=hi_A[1] - lo_A[1] + 1;
  mapc[i]=mapc[i-1] + (numRows * B_numRow);
}
```

The creation of the K matrix employs an initialization for the sake of completeness.

```
K_handle=GA_Create_handle();
GA_Set_data(K_handle, numDims, dimension, C_DBL);
GA_Set_array_name(K_handle, "Array_K");
GA_Set_irreg_distr(K_handle, mapc, nblock);
returnCode=GA_Allocate(K_handle);
value=-1.0;
GA_Fill(K_handle, (void*)(&value));
```

Finally, the actual calculation takes place. Each sub-matrix is overlaid on its appropriate place in the K matrix. Improving a bit on the approach taken with PETSc, no separately-allocated sub-matrices are employed.

```
A_localRow=hi_A[1] - lo_A[1] + 1;
A_localCol=hi_A[0] - lo_A[0] + 1;
K_localRow=hi_K[1] - lo_K[1] + 1;
K_localCol=hi_K[0] - lo_K[0] + 1;
for(A_col=0; A_col < A_localCol; A_col++)
{
  for(A_row=0; A_row < A_localRow; A_row++)
  {
    A_thisEle=(A_col * A_localRow) + A_row;
    K_col=A_col * B_numCol;
    for(j=0; j < B_numCol; j++)
    {
      K_row=A_row * B_numRow;
      for(i=0; i < B_numRow; i++)
      {
        K_thisEle=(K_col * K_localRow) + K_
                  row;
        double_ptr_local_K[K_thisEle]=
double_ptr_local_A[A_thisEle] * B[i][j];
        K_row++;
      }
      K_col++;
    }
  }
}

NGA_Release_update(A_handle, lo_A, hi_A);
NGA_Release_update(K_handle, lo_K, hi_K);
```

Calculated and groundtruth result matrices are compared using several statements since there is no function for directly comparing two equally-sized matrices.

```
GA_Elem_divide(K_groundtruth_handle, K_handle, K_groundtruth_handle);
value=GA_Ddot(K_groundtruth_handle, K_groundtruth_handle);
if(value == (double)(K_numRow*K_numCol)) printf("\nK matrices match\n");
else printf("\nK matrices do not match\n");
fflush(stdout);
```

4.3 MATLAB with Parallel Computing Toolbox and Distributed Computing Engine (http://www.mathworks.com/products/parallel-computing)

According to MATLAB's documentation, support is provided for distributed arrays and parallel functions that operate on those arrays. Distributed arrays can occupy memory across multiple workers running on multiple nodes. Parallel functions perform operations on the distributed data. The toolbox provides more than 150 parallel functions for operating on distributed arrays, including linear algebra routines based on ScaLAPACK. You can use the familiar operators for arrays in MATLAB to perform indexing, matrix multiplication, and transforms directly on distributed arrays. Not all Matlab functions operate on distributed arrays. Examples include kron() and sort().

4.3.1 Data Ingest

Each MATLAB process (lab) reads its portion of the matrix. In the case of matrix B, the entire matrix is read. Each process participates in the creation and population of distributed matrices by creating independent matrix components that are joined together to create the complete matrix.

```
% Read the A matrix. (Size=A1 x A2)
spmd
  filename='/usr/people/praeth/John/Challenge/A.bin';
  fid=fopen(filename, 'r');
  a1=fread(fid, [1 1], 'int64');
  a2=fread(fid, [1 1], 'int64');
  A_Partition=codistributor1d.defaultPartition(a1);
  if(labindex ~= 1)
    for i=1 : labindex - 1
      fread(fid, [A_Partition(i) a2], 'float64');
    end
  end
  AA=fread(fid, [A_Partition(labindex) a2], 'float64');
  fclose(fid);
  if(labindex == 1)
    A_StartRow=1;
    A_EndRow=A_Partition(labindex);
  else
    A_StartRow=1;
    for i=2 : labindex
      A_StartRow=A_StartRow + A_Partition(i - 1);
    end
    A_EndRow=A_StartRow + A_Partition(labindex) - 1;
  end
  K_StartRow=((A_StartRow - 1) * b1) + 1;
  K_EndRow  =(((A_EndRow - 1) * b1) + 1) + b1 - 1;
end

% Construct the distributed A matrix
spmd
  A=codistributed.build(AA, codistributor1d(1, A_Partition, [a1 a2]));
  A_Codistributor=getCodistributor(A);
end

% Free the components used to construct the A matrix
for i=1 : numLabs
```

```
    clear AA{i};
end
```

4.3.2 Calculating the Kronecker Tensor Product

The distributed Kronecker calculation is pretty simple in MATLAB since much of the manual work required in other tools is handled automatically. Local components are calculated using the same approach MATLAB takes for dense matrices as shown in its copyrighted kron.m code. These three lines are marked by %*.

```
% Calculate the Kronecker Tensor Product
spmd
   AA=getLocalPart(A);
   [ia,ib]=meshgrid(1:A_Partition(labindex),1:b1); %*
   [ja,jb]=meshgrid(1:a2,1:b2);                    %*
   KK=AA(ia,ja) .* B(ib,jb);                       %*
end

% Create a distributed version of the calculated K matrix.
spmd
  K=codistributed.build(KK, codistributor1d(1, K_Partition, [k1 k2]));
end

% Free components used to construct the calculated distributed K matrix
for i=1 : numLabs
    clear KK{i};
end
```

Calculated and groundtruth result matrices are compared using several statements since there is no function for directly comparing two equally-sized matrices.

```
match=sum(sum(K_Groundtruth == K));
if( match == (k1{1} * k2{1}) )
  disp('equal')
else
  disp('not equal')
end
```

At issue with the MATLAB implementation is that we are still learning how to distribute labs across multiple nodes while employing a pure-batch approach to running a job.

5. Results

As much as possible, the same approach to the Kronecker calculation was followed in its implementation by all three tools. To test these three implementations, Matlab was used to generate the A and B matrices via uniform-distribution random-number generation. These and the baseline Kronecker matrix occupied the entire memory of a Harold node (http://www.arl.hpc.mil/hardware/#harold). The resulting three matrices were output to disk (A, B, and K matrices). For each implementation, the A and B matrices were input and the Kronecker calculation was performed. Then the K matrix was read in and compared to the calculated matrix. In all three cases, the results were exactly the same. For PETSc and GA, matrices were distributed across nodes. For MATLAB, we still need to learn how to do that so the matrices were distributed across processes on the same node.

The author's opinion is that the Global Arrays Toolkit provides the best general-purpose support to distributed matrices. This is a personal impression due from an assessment of documentation and ease of programming. For problems requiring solvers, PETSc is the superior choice, although GA also provides some support in that domain. (Many people use PETSc and Global Arrays Toolkit within the same code.) If you work in MATLAB, we have found that trying to access PETSc and Global Arrays Toolkit distributed matrices are difficult at best. It is better to employ MATLAB's Parallel Computing Toolbox and Distributed Computing Engine(DCE), although one must consider DCE's learning curve.

6. Future Work

There are a number of steps we can still take, depending on the path followed by active users. The most important is embedding this capability into user code. Tests with increasingly-large matrices need to be conducted in a way that allows users to judge the accuracy of the application's results. The author would feel comfortable trusting the distributed Kronecker calculation once this next phase of testing is completed.

Other steps that may be useful in the future are:

- Supplement row-oriented allocations with column-oriented allocations. A more complex allocation scheme may be necessary as the A and K matrices increase in size. This would increase the number of nodes used and thus would, within overhead limits, reduce the amount of time required and increase the amount of memory available.
- Loosen the present dense matrix requirement so that any or all of the three matrices can be sparse.
- Allow matrix B to also be distributed.
- Allow matrices to be distributed across only some cluster nodes.
- Examine the basic code for opportunities to increase efficiency. For instance, explore the use of memcpy instead of index-by-index copying. Another might be components of vector indexes that do not have to be recalculated.

As user needs evolve, this additional work will be undertaken as needed.

7. Summary

By applying effort to re-engineering, users can transition from shared-memory to distributed-memory machines. This effort becomes unavoidable as access to sufficient shared-memory resources becomes unachievable. The re-engineering effort must include careful baseline testing that compares the re-engineered code's results with those achieved by the original code. Except for round-off in outer decimal places, the results should be the same. Any other differences need to be assessed to ensure that all transition errors are removed. "Warnings" also need to be removed since these obscure existing issues and can lead to unexpected results, especially as warnings accumulate.

DoD HPC users can contact the PETTT program for assistance on their projects. Contact a PETTT on-site at your location, send email to the author, or send email to pettt_requests@hpti.com.

Acknowledgements

Many thanks are due to Gary Kedziora for his superb comments, and for alerting me to Global Arrays Toolkit. Don Bahls alerted me to PETSc.

References

Graham, A., *Kronecker Products and Matrix Calculus*, Halsted Press, New York, NY, 1981.

Mathews, V.J. and Sicuranza, G.L., *Polynomial Signal Processing*, New York, NY, Wiley, 2000.

Mathworks, "Kronecker tensor product", http://www.mathworks.com/help/techdoc/math/f4-988203.html#f4-1892, 2011.

Van Horn, M.H., "An introduction to Kronecker (Tensor) Product factorization and implementations", North Carolina State University, http://www.mindspring.com/~mvanhorn, 1999.

Weisstein, Eric W., "Kronecker Product", *MathWorld--A Wolfram Web Resource*, http://mathworld.wolfram.com/KroneckerProduct.html, 2011.

Performance Comparison of an HPC ℓ_1-Optimization Algorithm for Compressed Sensing

Miguel Hernandez IV, Julio Olaya, Reinaldo Sanchez, Carlos Ramirez, Rodrigo Romero, Leticia Velazquez, and Miguel Argaez
The University of Texas at El Paso, El Paso, TX
{miguelher, leti, margaez, raromero2}@utep.edu, {rsanchezarias, caramirezvillamarin, jolaya}@miners.utep.edu

Abstract

In this paper, we present a performance comparison of the Path-Following Signal Recovery Algorithm (PFSR) using three different high performance computing approaches: The first is a serial version using the Basic Linear Algebra Subprograms (BLAS) optimized library; the second is a parallel version using the CUDA BLAS (CUBLAS) graphics processing unit (GPU) optimized library; and the last is a parallel GPU version using customized kernel functions. The goal of this comparison is to determine which particular version results in the most reduction of total execution time.
This work is supported in part by the US Army Research Laboratory through the US Army High Performance Computing Research Center, Cooperative Agreement W911NF-07-0027.

1. Path-Following Signal Recovery

1.1 Compressed Sensing

In recent years, a major interest in ℓ_1 regularization-based methods has arisen in the optimization and signal processing communities, producing several efficient algorithms for sparse-signal reconstruction (Candès, et al., 2006; Donoho and Huo, 2001). The motivation for ℓ_1 approaches come from the theory of compressed sensing (compressive sampling) pioneered by Emmanuel Candès, Justin Romberg, Terence Tao, and David Donoho. Compressed sensing answers the question: *How much information is necessary to accurately reconstruct a signal?* From (Candès, 2006; Donoho, 2006), we find that sparse or compressible signals can be accurately reconstructed from a very limited number of measurements. A signal $x \in \mathbb{R}^n$ can be recovered using information from a collection of m linear measurements $b_i=(a_i, x)$ for $i=1, \ldots, m$. In matrix notation, we can write this as $b=Ax$ where $A \in \mathbb{R}^{m\times n}$ and the vectors a_i are rows. We will assume that $m \ll n$, and the measurement matrix A has full rank.

1.2 Problem formulation

The basis pursuit ℓ_1-minimization problem is shown in Equation 1.

$$\begin{aligned} \min \quad & \|x\|_1 \\ \text{subject to} \quad & Ax = b \end{aligned} \tag{1}$$

A solution to Equation 1 is proposed in (Argáez, et al., 2011) that solves a sequence of problems of the form:

$$\begin{aligned} \min_x \quad & \sum_{i-1}^{n} \left(x_i^2 + \mu\right)^{1/2} \\ \text{s.t} \quad & Ax = b, \end{aligned} \tag{2}$$

This approach leads to a path-following method to find the solution x of the ℓ_1-minimization problem by solving a sequence of linear equality constrained multi-quadratic problems. A MATLAB implementation of the path-following algorithm can be found at: http://www.math.utep.edu/Student/cramirez/.

1.3. Algorithm

Algorithm 1 outlines the PFSR algorithm used to find a solution to the basis pursuit problem in Equation 1. For a complete discussion of the PFSR algorithm, see (Argáez, et al., 2011).

Algorithm 1 Path-Following Signal Recovery

Task: Find an approximate solution x to the problem
$\min_x ||x||_1$ subject to $Ax + v = b$ and $||v|| \leq \epsilon$

Parameters: matrix **A**, vector **b**

1: **Initialization:** Set σ, τ, μ, ϵ_1
2: **Initial approximate solution:** $x = A^T b$
3: **Outer Loop:** For j =1, …, *maxiter*
4: **Inner Loop:** Set $x\_ = x$
5: **Update weight matrix:** $D_\mu(x\_) = diag\left(x_-^2 + \mu\right)$
6: **Solve the fixed-point equation:**

$$\begin{bmatrix} D_\mu(x\_)^{-1/2} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}$$

7: **Check proximity to the central path:**
if $\frac{\|x - x\_\|}{1 + Vertx\|} \geq \sqrt{\mu}$ **go to** Step 4.
8: **Set:** $\tilde{x} = |x| \, w = A^T y, \tilde{z} = (\mathbb{1} - |w|)$
9: **Compute:** $error_{primal} = \frac{\|Ax - b\|}{1 + \|b\|}, gap = \frac{\tilde{x}^T \tilde{z}}{n}$
10: **Stopping criteria for the problem:**
if $error_{primal} > \epsilon$ or $gap > \epsilon_1$
Update $\mu = \min\{\sigma \, gap, \tau \, \mu\}$, **go to** step 3
else
Return solution x^*

It is important to note that Step 6 is reformulated and solved using a specially-designed Conjugate Gradient (CG) algorithm. Using this CG method, only matrix-vector multiplication operations are needed.

1.4 Execution Time Assessment

A sequence of synthetic problems is generated to examine how the problem size affects the execution time. A family $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ with n varying from 2^{12} to 2^{23} and $m = \frac{n}{4}$ m = are created. The matrix A is randomly generated with independent identically distributed entries. The vector b is constructed as $b=Ax^0+ v$, where x is the original signal and v is an additive noise vector. The sparsity of x^0 is controlled so that the total number of nonzero entries $\frac{5}{2^7} n$ with amplitude ±1. The vector v is set according to a Gaussian distribution with mean 0 and standard deviation.

2. Graphics Processing Unit (GPU) Background

2.1 GPU Architecture

The standard Tesla architecture has a set of stream multiprocessors (SMs) with several stream processors (SPs), or cores. This architecture has a multithread instruction unit, instruction cache, global memory, constant memory and shared memory. Each SP is a hardware multithread that has 512 lightweight threads, a scalar integer and a floating-point arithmetic unit. The SMs use single-instruction multiple-thread (SIMT) architecture to manage and schedule 32 threads in parallel, forming a group called warp (NVIDIA, 2010b).

2.2 Programming Model

The programming model is based on a single-program multiple-data (SPMD) approach in which the programmer writes a single program that the GPU executes using multiple threads in parallel. The programming model has three important features. The first is a hierarchy of thread groups which consist of three levels. A grid is a set of thread blocks that execute the kernel function. Each grid has block of threads, and each block consists of hundreds of threads. The

second feature is shared memory that allows the threads to share data. The last feature is the barrier synchronization that manages the execution of threads (Kirk and Hwu, 2010). Computer Unified Device Architecture (CUDA) from NVIDIA is a parallel program and software platform that comes with an extended C compiler, called CUDA C, which allows for the programming of the GPU with a high-level language (NVIDIA, 2010c).

3. Linear Algebra Libraries

3.1 BLAS

In the area of high-performance linear algebra, BLAS is an essential library fundamental to high performance computing (HPC), and has become the de facto standard (AMD, 2009). Classified into levels based on the degree-of-complexity, BLAS level 1 includes the dot product, level 2 includes matrix-vector multiplication, and level 3 includes matrix-matrix multiplication. The BLAS library was created to speed-up these operations (Dongarra, 1986).

3.2 CUBLAS

In recent years, high performance computing (HPC) on GPGPUs has become increasingly popular (Kestur, et al., 2010; Zhang, et al., 2009; Dang, et al., 2010; Yang, et al., 2010); consequently, the BLAS library has been ported to GPGPUs. CUBLAS is an implementation of BLAS on top of the NVIDIA CUDA runtime that provides access to the computational resources of the GPU (NVIDIA, 2010a).

The basic template by which applications use the CUBLAS library is to (see Figure 1):

- Allocate memory for matrix/vector objects in GPU memory space.
- Transfer matrix/vector data from CPU to GPU memory space.
- Call a sequence of CUBLAS functions for GPU execution.
- Transfer the results from GPU to CPU memory space.

CUBLAS provides helper functions for creating and destroying objects in the GPU, and for writing data to and retrieving data from these objects (NVIDIA, 2010a).

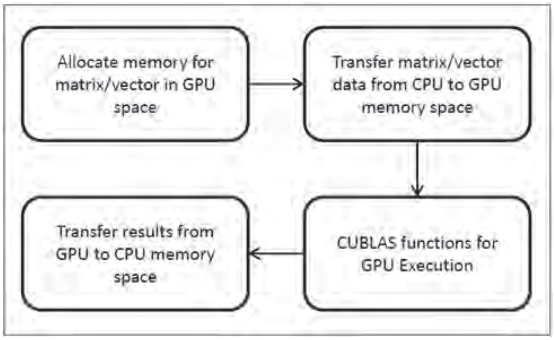


Figure 1. CUBLAS program flow

4. PFSR Algorithm Profile

Profiling allows the analyst to determine where a program spends its time during execution. This information identifies areas of a program that may be candidates for rewriting to make the program execute faster. GNU gprof was used to profile the PFSR algorithm (Fenlason and Stallman, 1998).

Table 1 shows the profile results with the column headers defined as:

- Function - Name of the function.
- Time - Percentage of the total time that was spent in the function, including time spent in subroutines called by the function.
- Called - The number of times the function was called.

The entries are sorted by time spent in the function and its subroutines. The preceding lines in the table describe the callers of a function, and the lines that follow describe the subroutines it calls. For example, focusing on the conjugate gradient function (CG), we see that CG was called by PFSR, and CG calls mv product. The program spent 99.2% of its time in the CG function and CG was called 30 times.

Table 1. PFSR Algorithm Profile

Function	Time (%)	Called
main	100	
PFSR	99.8	1
CG	99.2	30
mv product	86.5	1296

Listing 1: Naive Matrix-Vector Multiply

```
/* ************************************************
* Ax = y
* A : m x n matrix
* x : n x 1 vector
* y : m x 1 vector
************************************************** */
void mv_product (float *A, float *x
          float *y, int m, int n)
{
          int i, j ;
          for (i = 0; i < m; i++);
          {
               y [i] = 0;
               for (j = 0; j < n; j++);
               y [i] += A[1 * n + j] * x[j];
          }
}
```

Using the results shown in Table 1, mv product (which corresponds to matrix-vector product) was identified as the bottleneck function. The program spent 86.5% of its time performing matrix-vector products, and a total of 1,296 matrix-vector products were executed. Based on these findings, it was concluded that speeding-up the PFSR algorithm (and reducing total execution time) corresponded to speeding up the matrix-vector products.

To do this, the following approaches were considered:

1. BLAS matrix-vector product
2. CUBLAS matrix-vector product
3. Matrix-vector GPU kernel function

Because the PFSR algorithm is scalable in terms of matrix and vector dimension, we recognize that the most appropriate approach depends heavily on the dimensions of the matrix-vector elements. Our goal is to find the crossover point matrix dimensions that dictate the most appropriate approach, i.e., the point at which a GPU-based version should be used instead of the serial or BLAS versions.

5. Methodology

In this section, the BLAS, CUBLAS, and kernel approaches for matrix-vector multiplication are described.

5.1 BLAS

The BLAS implementation used in this experiment was the CBLAS version supported in the GNU Scientific Library (GSL) (Galassi, 2009). The conversion of the naive matrix-vector product shown in code listing 1 to BLAS sgemv is shown in code listing 2.

Listing 2: BLAS Matrix-Vector Multiply

```
/* ************************************************
* Ax = y
* A : m x n matrix
* x : n x 1 vector
* y : m x 1 vector
************************************************
void mv_product (float *A, float *x, float *y,
                 int m, int n)
{
    float alpha = 1.0;
    float beta = 0.0;
    int inc = 1;
    int floatSize = sizeof (float);
    // perform operation on the CPU
    /*
    * y = alpha * op (A) * x + beta * y
    * where op (A)=A or op (A)=transport (A)
    /*
    cblas_sgemv (CblasRowMajor, CblasNoTrans ,m, n,
                 alpha ,A, n , x , inc , beta , y , inc);
}
```

5.2 CUBLAS

The conversion of the naive matrix-vector product to CUBLAS sgemv is shown in code listing 3. Notice that the CUBLAS version deviates from the typical program flow shown in Figure 1, that is, within the mv product function, the matrix A is not transferred to GPU memory space prior to its use in sgemv. In the PFSR algorithm, the matrix A is an input value to the PFSR algorithm that never changes. We exploit this fact and write A to GPU memory once during main() and avoid transferring A to the GPU before each matrix-vector calculation.[1] Although not strictly necessary, we allow the pointer *A to remain in the function prototype to maintain a standard function signature across the three approaches. The pointer to GPU memory devPtrA is a global variable that points to the address of A in the GPU. It is this variable that is used during the matrix-vector multiplication.

5.3 Kernel

Similar to the CUBLAS method, the matrix-vector product kernel writes the matrix A to GPU memory once during main()[2]. The vector x is written to GPU memory, Ax computed, and y retrieved from the GPU during each matrix-vector operation. Since the row dimension of the matrix A is always a multiple of 32 in our experiment, 32 threads (1 warp) are used during the computation of y, allowing one thread to be used for each row of A. Code listing 4 shows the matrix-vector product kernel implementation[3].

6. Results

In this section, the performance results obtained using two different hardware architectures are discussed.

[1]The matrix A^T is also used by the PFSR algorithm. This matrix also never changes and is written once to GPU memory during main(). In those areas of the algorithm that require multiplication by A^T, a matrix-vector function is used specifically for A^T.

[2]Similar to the CUBLAS version, the matrix A^T is also written once to GPU memory during main() with a different matrix-vector kernel used for A^T.

[3]The kernel function is called by a function that conforms to the standard function signature in code listing 1.

6.1 OS X

The PFSR algorithm was executed on a MacBook Pro with a GeForce 9600 GT NVIDIA GPU card. The matrix was scaled up from 64×512 to 2,048×8,192, and the results are shown in Table 2. In addition, four different matrix-vector product implementations are shown. These results represent execution times averaged over 16 runs compiled with level 3 (-O3) optimization using gcc. The columns are defined as:

- log_2: $log_2(mn)$.
- Total Time: The total execution time for the PFSR algorithm.
- Time MV: Cumulative time spent performing matrix-vector products.
- % Total Time MV: (Time MV/Total Time) * 100

Listing 3: CUBLAS Matrix-Vector Multiply

```
/* ************************************************
* Ax = y
* A : m x n matrix
* x : n x 1 vector
* y : m x 1 vector
************************************************/
void mv_product (float *A, float *x float *y,
                 int m, int n)
{
    float alpha = 1.0;
    float beta = 0.0;
    int inc = 1;
    int floatSize = sizeof (float);
    // define a pointer to memory to be allocated
    // on the GPU
    float *devPtrx;
    float *devPtry;
    cublasStatus stat;
    stat=cublasAlloc (n, floatSize , ( void **)& devPtrx);
    stat=cublasAlloc (m, floatSize , ( void **)& devPtry);
    stat=cublasSetVector (m, floatSize , x, 1 , devPtrx , 1);
    stat=cublasSetVector (n, floatSize , y, 1 , devPtry , 1);
    // perform operation on the GPU
    /*
     * y = alpha * op (A) * x + beta * y
     * where op (A)=A or op (A)=transport (A)
     */
    cublasSgemv ( 'N' , m,n , alpha , devPtrA , m,
                  devPtrx , inc , beta , devPtry , inc);
    stat=cublasGetVector (m, floatSize , devPtry , inc ,
                          y , inc );
    cublasFree (devPtrx );
    cublasFree (devPtry );
}
```

Listing 4: Kernel Matrix-Vector Multiply

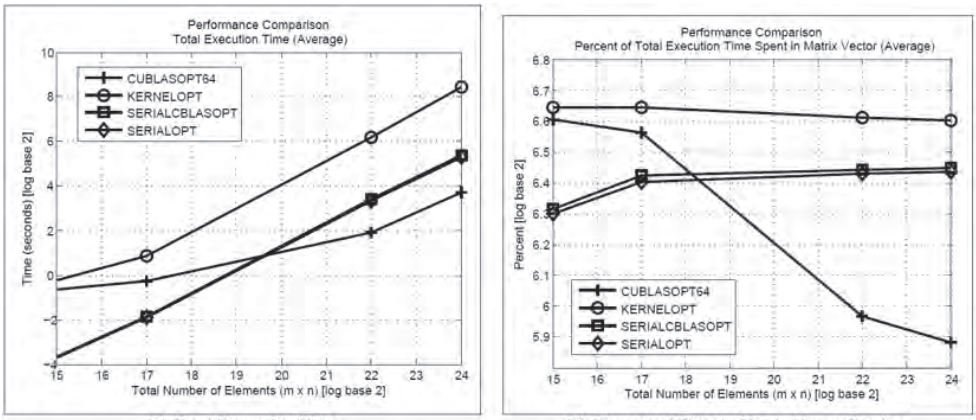
```
/* ************************************************
* Ax = y
* A : m x n matrix
* x : n x 1 vector
* y : m x 1 vector
************************************************
_ _global_ _  void mv_productKernal ( int  column ,
                                      float  *A,
                                      float  *x,
                                      float  *y)
{
     int tx = blockIdx. x * blockDim. x + threadIdz .x;
     C[tx] = 0.0f;
     for (int  i = 0;  i < column ; ++i)
      y [tx]  += A[tx*col+i];
}
```

Table 2. Performance Table (OS X)

Approach	m	n	log_2	Total Time (s)	Time MV (s)	% Total Time MV
CUBLAS	64	512	15	0.64	0.62	97.27
CUBLAS	128	1024	17	0.84	0.80	94.64
CUBLAS	1024	4096	22	3.73	2.34	62.59
CUBLAS	2048	8192	24	13.05	7.72	59.08
KERNEL	64	512	15	0.86	0.86	100.00
KERNEL	128	1024	17	1.82	1.82	100.00
KERNEL	1024	4096	22	72.18	70.55	97.74
KERNEL	2048	8192	24	345.46	338.39	97.95
CBLAS	64	512	15	0.08	0.06	79.63
CBLAS	128	1024	17	0.28	0.24	85.85
CBLAS	1024	4096	22	10.56	9.17	86.87
CBLAS	2048	8192	24	42.00	36.67	87.30
SERIAL	64	512	15	0.08	0.06	78.81
SERIAL	128	1024	17	0.27	0.23	84.61
SERIAL	1024	4096	22	10.00	8.62	86.17
SERIAL	2048	8192	24	39.78	34.45	86.59

Figure 2(a) shows a plot of total execution time vs. the total number of matrix elements. A crossover point exists between 2^{19} and 2^{20} matrix elements where the CUBLAS version begins to outperform all other versions, i.e., using the CUBLAS matrix-vector product results in faster total execution time.

Figure 2(b) shows a plot of the percent of total execution time spent performing matrix-vector products vs. the total number of matrix elements. A crossover point exists between 2^{18} and 2^{19} matrix elements where using the CUBLAS version begins to outperform all other versions.



(a) Total Execution Time (b) Percent of Time in Matrix-Vector Product

Figure 2. Average performance on OS X

6.2 Longhorn

The same tests performed on the MacBook Pro were repeated on the high performance computing resources at the Texas Advanced Computing Center (TACC). Specifically, the PFSR algorithm was executed on Longhorn's NVIDIA Quadro FX 5800 GPUs with the results shown in Table 3.

Table 3. Performance Table (Longhorn)

Approach	m	n	Log2	Total Time (s)	Time MV (s)	% Total Time MV
CUBLAS	64	512	15	0.81	0.80	98.13
CUBLAS	128	1024	17	0.85	0.83	97.34
CUBLAS	1024	4096	22	2.34	1.51	64.65
CUBLAS	2048	8192	24	5.64	2.45	43.41
CUBLAS	4096	16384	26	17.88	5.82	32.70
CUBLAS	8192	32768	28	67.58	20.40	30.33
KERNEL	64	512	15	1.34	1.34	100.00
KERNEL	128	1024	17	2.06	2.06	100.00
KERNEL	1024	4096	22	22.01	22.01	100.00
KERNEL	2048	8192	24	77.99	75.77	97.15
KERNEL	4096	16384	26	351.38	339.35	96.58
KERNEL	8192	32768	28	1659.27	1612.43	97.18
CBLAS	64	512	15	0.06	0.05	77.41
CBLAS	128	1024	17	0.23	0.19	82.82
CBLAS	1024	4096	22	8.01	7.21	90.05
CBLAS	2048	8192	24	31.53	28.77	91.27
CBLAS	4096	16384	26	125.33	114.97	91.73
CBLAS	8192	32768	28	499.15	459.47	92.05
SERIAL	64	512	15	0.08	0.06	73.21
SERIAL	128	1024	17	0.32	0.29	91.97
SERIAL	1024	4096	22	10.32	9.54	92.39
SERIAL	2048	8192	24	40.73	38.02	93.33
SERIAL	4096	16384	26	161.94	151.82	93.75
SERIAL	8192	32768	28	645.65	606.34	93.91

Figure 3(a) shows a plot of total execution time vs. the total number of matrix elements with a crossover point between 2^{19} and 2^{20} matrix elements. Figure 3(b) shows a plot of the percent of total execution time spent performing matrix-vector products vs. the total number of matrix elements. A crossover point exists between 2^{18} and 2^{19} matrix elements.

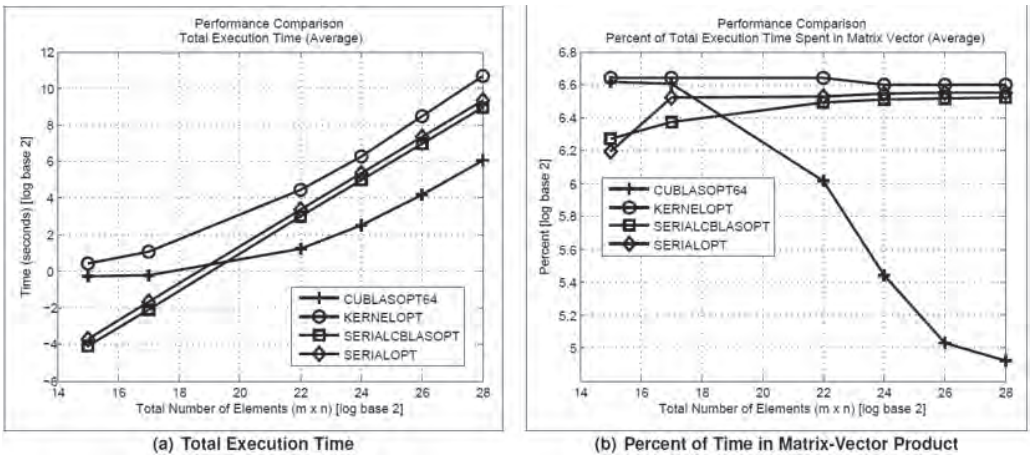


Figure 3. Average performance on Longhorn

7. Conclusion

A performance comparison the of Path-Following Signal Recovery Algorithm (PFSR) was presented using three different high-performance approaches: one using the BLAS optimized library; another using the CUBLAS optimized library; and a GPU version using customized kernel functions. We found that a crossover point exists that dictates the appropriate matrix-vector multiplication approach to use based on matrix and vector dimensions. Given these results, the best course of action is to integrate both the BLAS and CUBLAS versions into a single code-base and allow the program to choose the best approach to use based on the dimensions of the matrices and vectors. For those matrix-vector dimensions under the crossover point, the BLAS matrix multiplication version should be used. For those dimensions over the crossover point, the CUBLAS matrix multiplication version should be used.

Acknowledgements

The authors thank the Department of Mathematical Sciences, the Computational Science Program, Department of Computer Science, and the Cyber-Share Center. This research was performed at The University of Texas at El Paso and supported by the US Army Research Laboratory through the Army High Performance Computing Research Center, Cooperative Agreement W911NF-07-0027 and the National Science Foundation under Grants CREST HRD-0734825 and CNS-0923442.

References

AMD, "AMD core math library for graphic processors," *Advanced Micro Devices*, March 2009.

Argáez, M., C. Ramirez, and R. Sanchez, "An ℓ_1-algorithm for underdetermined systems and applications," *IEEE Proceedings of the North American Fuzzy Information Processing Society*, 2011 (to appear).

Candès, E., "Compressive sampling," *Proceedings of the International Congress of Mathematicians*, 2006.

Candès, E., J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly-incomplete frequency information," *IEEE Trans. Info. Theory*, 2006.

Dang, D.M., C.C. Christara, and K.R. Jackson, "A parallel implementation on GPUs of ADI finite- difference methods for parabolic PDEs with applications in finance," *Mathematical Finance*, 2010.

Dongarra, J.J., *An extended set of Fortran basic linear algebra subprograms*, Argonne National Laboratory, Mathematics and Computer Science Division, 1986.

Donoho, D., "Compressed sensing," *IEEE Trans. Inf. Theory*, 2006.

Donoho, D. and X. Huo, "Uncertainty principles and ideal atomic decomposition," *IEEE Trans. Info. Theory*, 2001.

Fenlason, J. and R. Stallman, *GNU gprof: The GNU profiler*, Free Software Foundation, 1998.

Galassi, M., *GNU Scientific Library Reference Manual*, Free Software Foundation, 2009.

Kestur, S., J.D. Davis, and O. William, "BLAS Comparison on FPGA, CPU and GPU," *IEEE Annual Symposium on VLSI*, 2010.

Kirk, D.B. and W.W. Hwu, *Programming Massively-Parallel Processors: A Hands-on Approach*, Morgan Kaufmann, Feb 2010.

NVIDIA, *CUDA CUBLAS Library*, NVIDIA, May 2010a.

NVIDIA, *CUDA C Best-Practices Guide*, NVIDIA, 2010b.

NVIDIA, *CUDA C Programming Guide*, NVIDIA, 2010c.

Yang, D., J. Sun, J. Lee, G. Liang, D.D. Jenkins, G.D. Peterson, and H. Li, "Performance comparison of cholesky decomposition on GPUs and FPGAs," *Symposium on Application Accelerators in High Performance Computing*, 2010.

Zhang, Z., Q. Miao, and Y. Wang, "CUDA-based jacobis iterative method," I*nternational Forum on Computer Science-Technology and Applications*, 2009.

Using a Heterogeneous Interactive HPC to Enhance Streaming Images

Scott Spetka
ITT Corp, Rome, NY and SUNYIT, Utica, NY
scott@cs.sunyit.edu

George Ramseyer
US Air Force Research Laboratory, Information Directorate (AFRL/RI), Rome, NY
george.ramseyer@rl.af.mil

Scot Tucker
ITT Corp, Rome, NY
scot.tucker@itt.com

Abstract

Developing interactive applications for heterogeneous high performance computers (HPCs) raises new issues that are not encountered for traditional HPC applications. It is important to understand how the interactive nature of these systems and differences in machine architectures complicate the development process. An experimental system for processing streaming-images using the Physically-Constrained Iterative Deconvolution (PCID) algorithm on the Department of Defense's (DoD's) largest interactive supercomputer provided valuable insights into these issues. This case study exposed two problems associated with interactivity: managing the allocation of HPC resources to process incoming image frames, and managing the input of a live data stream into PCID. The study showed that a Phoenix high-performance information management system could be the basis of a solution to these problems. Services implemented using Phoenix to simplify communications controlled the allocation of processing nodes, the flow of information within the HPC, and the external streaming inputs from outside of the HPC. Using Phoenix also allowed remote users to view the results and control the PCID execution. Adapting the PCID code to run in a Xeon/Cell-BE environment resulted in an improved understanding of implementation techniques and configuration issues. The experimental implementation demonstrated the potential of heterogeneous HPC systems to serve the growing needs of the DoD for interactive real-time data processing services for processing streaming data from remote sources.

1. Introduction

Interactive *high performance computing (HPC)* has been a goal of the US Air Force Research Laboratory's Information Directorate (AFRL/RI) for the past ten years (Ramseyer, 2001, 2002; Spetka, 2002). Using HPCs interactively can achieve real-time performance that is not possible for interactive time-sharing systems, which have been used for most interactive computing. Over the past decade, new approaches have been taken to submit jobs to HPCs as one would normally submit them to time-sharing systems, using a graphical user interface. Most HPC use over this period has been focused on increased throughput, to use processing resources as efficiently as possible and process a maximum number of jobs. However, our research goal is to minimize latency, even at the expense of some efficiency. For interactive computing to be effective, an expectation of rapid response must be fostered. The metrics used for judging success necessarily vary among applications. For example, in some cases, a video stream must be processed in near real-time for it to be used effectively in decision making.

Dedicated resources are needed to operate a production-interactive HPC system. The current batch resource allocation systems are designed with the expectation that most jobs submitted will be delayed until resources become available. This type of system allows processing to be pushed into timeframes when interactive users would not normally be using the system, such as in the middle of the night. This model is very effective in increasing the utilization of HPCs, but does not support interactive HPC. Interactive HPC applications can also drive HPC systems at a high-level of efficiency, for example when processing live video streams. Other interactive applications, which might take an existing video stream and reprocess it, would be less efficient, since dedicated HPC resources would be waiting for users to decide which existing stream to process next.

Interactive HPC use has become feasible due to dropping costs for HPC systems, but also due to improved power management. Early HPCs were few and expensive. Efficiency was very important to capitalize on the large investment in hardware. Power was not a concern, compared to hardware cost, especially since HPCs were few. Today, HPC hardware is less expensive, and can sit idle while consuming minimal power. Special purpose "green" systems, like AFRL Condor (Barnell, 2011), use power-efficient SONY Playstation 3's to implement a low-Watts/TeraFLOP system. For interactive computing, it is essential to configure a system with hardware resources that are sufficient to meet a maximum expected load. If the expected load is exceeded, the interactive experience and the ability to operate at near-real-time will degrade.

Geographically-distributed interactive HPC processing resources play a unique role in meeting mission requirements, along with mobile interactive HPCs. Processing sensor inputs in real-time is essential for air traffic control, disaster recovery, space situational awareness and many other applications. Sensors may provide imaging, audio and other data for analysis. Mobile HPC platforms reduce overall network traffic by processing data near the source, for example in a reconnaissance aircraft. Geographic distribution of HPCs provides increased local access to support applications with intense graphics, while supporting improved connectivity over a region. In addition, geographic distribution allows HPC users to use systems in time-zones that are not in peak power consumption periods.

Section 2 of this paper describes the system architecture that is being implemented to support interactive heterogeneous HPC use for streaming image enhancement. Section 3 presents further detail in the context of the Physically-Constrained Iterative Deconvolution (PCID) image enhancement code (Matson, 2007). Then, PCID is used in Section 4 to demonstrate system information flow and user interfaces needed to drive an end-to-end application. The paper continues to discuss issues for heterogeneous HPC developers in Section 5. The paper conclusion discusses the potential for interactive HPC to enhance productivity and creativity, lessons-learned and future research directions.

2. System Architecture

The system architecture shown in Figure 1 was developed to support the PCID streaming-image correction algorithm for Space Situational Awareness (SSA) as well as other SSA codes, like the PARallel Speckle rEconstruction Code (PARSEC). The PCID code has been optimized for HPC execution (Linderman, 2009). The architecture supports the allocation of HPC resources to interactive codes, and the management of inputs and results. The Phoenix information management system (Grant, 2009) supports all communication for these functions. Figure 1 shows the In-line Speckle Image Recovery Engine (INSPIRE) Service-Oriented Architecture (SOA), used to support SSA codes. The main components of the INSPIRE SOA are the Code Input Publisher, Clique Coordinator, Clique Master, and Result Subscriber. The INSPIRE SOA implements a framework that is adapted to each code that is implemented, to accommodate code-specific differences. Extensions to the SOA architecture to support PCID are noted in Figure 1.

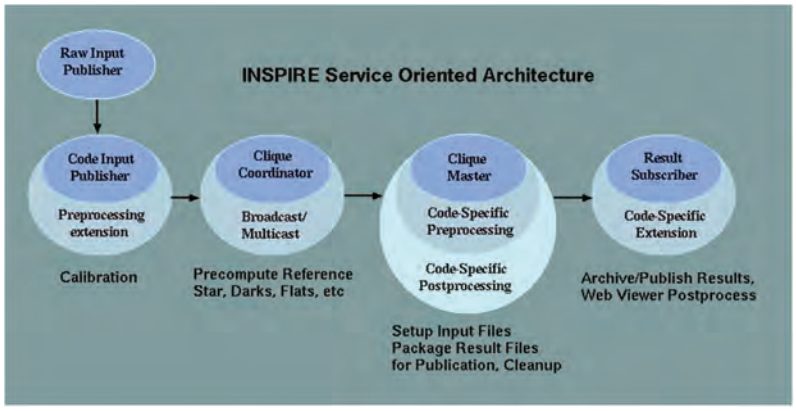


Figure 1. INSPIRE Service-Oriented Architecture

The Phoenix information management system is responsible for all of the communication between the INSPIRE component services (Spetka, 2010). Each code that is launched requires a unique "run_id" that is used to segregate traffic among the codes. Each instance of INSPIRE that is created adds the run_id to each publication and subscription. The run_id is assigned by a single program that is responsible for initiating execution. Each INSPIRE component can also be launched manually or scripted. In addition to supporting location independence for INSPIRE processes, Phoenix pub/sub communications simplifies the mechanism that allows multiple instances of services for flexibility, performance and

reliability (Yan, 2005; Spetka, 2008; Tucker, 2008). A script initiates the execution of the four INSPIRE services. The required inputs include the name of the code to execute, host files or batch id for reserved nodes, parameters needed to divide the nodes into cliques (groups of processors assigned for each parallel code execution), and code-specific parameters. The following sections describe each of the INSPIRE services and the parameters that are needed for their initialization.

The Code Input Publisher (CIP) receives inputs for an associated HPC code through a byte-stream interface. The inputs, which could be raw image frames or other raw data, can also include other inputs, like calibration data. Inputs are processed to create a Phoenix information object (IO) that is published. The IO includes basic code information, like code name and run_id, which are used to route the IO through to a processing clique. The CIP also encodes raw data into the payload, and inserts decoding information into the message metadata or into the payload along with the raw data. Metadata can also include time-stamps for performance benchmarking. The base CIP service implementation is extended by code-specific methods to adapt it to each code implemented. The CIP is essentially a filter that processes the raw inputs into a format that is expected by the HPC code. In most cases, this requires collecting sufficient data to process in an HPC execution. The pub/sub implementation provides flexibility for collecting inputs and for expressing execution requests. Inputs may come from multiple sensors, for example in an application that uses multiple sensors to locate the source of a signal. For reliability, code execution can be performed at multiple HPCs. Multiple execution requests can be initiated, without changing the CIP implementation, simply by starting additional Clique Coordinators, which subscribe to execution requests as described in the next section.

The Clique Coordinator (CC) provides scheduling services for execution requests for a supported code. The main function of the CC is to find available HPC nodes to service each execution request from a pool of allocated nodes which are dedicated to the CC upon execution. The currently implemented allocation policy uses a fixed clique size. A clique is a set of nodes that is sized to support a single execution of the code. The CC can easily be extended to support other allocation policies, like dynamically-sized cliques. This is useful because it allows the size of the input to vary. For example, in the current implementation of the CC, input ensembles (sets of raw frames) are fixed-size. But, for many image streams, ensemble sizes would naturally vary, due to object rotation, changing telescope settings, etc. For the PCID code, IOs are sent to the Clique Master (CM), which is responsible for managing the execution of the code on the assigned nodes. The CM is described in the next section. More complicated execution models are easily supported by the CC. For example, the PARSEC code requires preprocessing of calibration data, used during the code execution. The CC can broadcast ensembles with requests to process the calibration data to all CMs, allowing each of them to create a local copy for use in executing the PARSEC code. The CC is notified when each execution completes, to make the occupied clique available.

The Clique Master (CM) subscribes to its input execution requests from the CC. Each code may have its own unique set-up requirements. The CM performs set-up functions and initiates the code execution. The current CM implementation uses mpiexec to launch the code, but this can be easily adapted to accommodate other parallel execution systems. When the execution is completed, the CM usually publishes the results. In the case of PARSEC, instead of publishing the resulting calibration data, it is instead prepositioned in local storage. In some cases, multiple outputs may be loaded into the IO payload and described in the IO metadata. The CM cleans up the execution environment of temporary files, etc,. and notifies the CC that it is again available.

The Result Subscriber (RS) is usually started at the same time that the other INSPIRE components are initialized. The RS extends a simple implementation that mirrors the CM extensions for result publication. The results then have to be unpacked from the IO result payload and processed in some way. Each execution of the PCID code produces a single corrected image. For a streaming-image display, results are converted to the JPEG format and copied to a Web server, where they may be viewed. Additional result processing can be performed by multiple RS instances. Each result is sent to the publication stream, and then delivered according to its subscription. For example, an additional subscriber could produce an MPEG movie, while the live display proceeds. Archival of results is directly and transparently supported by Phoenix according to a simple parameter which is configured when the Phoenix connection is established. Any of the INSPIRE publications may be archived and accessed at a later time using the Phoenix query feature.

3. PCID/INSPIRE

The PCID implementation of the INSPIRE architecture is shown in Figure 2. A web interface controls each of the modules in the INSPIRE pipeline. A separate instance of INSPIRE is created for each of the simultaneously executing copies of the code. Running multiple copies of the same code can be useful for testing different sets of input parameters. Phoenix makes multiple executions transparent, since the Sensor Publisher, the Image Calibrator and the Ensemble Publisher interfaces are unchanged. Phoenix routes each raw sensor frame publication to each of the subscribers. The

Image Calibrator is defined as a service, using the template of the Ensemble Publisher, which is derived from the Code Input Publisher in Figure 1. Using this pattern, a general pipeline or a tree of processing steps may be constructed for any set of inputs. One example is the multiple-input sensor processing code mentioned above. Applications that collect data over wider geographic regions may merge data in pre-processing steps to create a tree structure to ultimately produce inputs for the HPC codes, which would be received by the Clique Coordinator for the code.

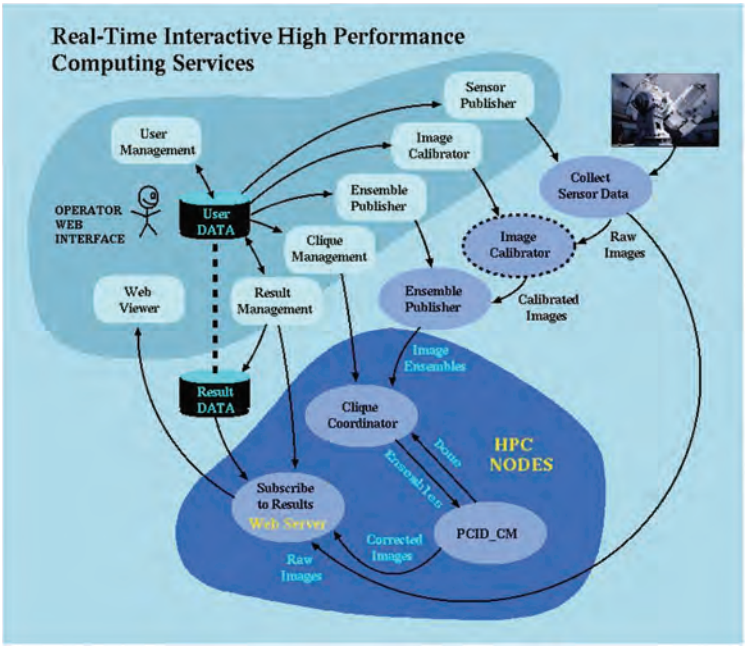


Figure 2. Interactive Processing Pipeline

The Ensemble Publisher operator can view the results as they are produced in real-time. The interface allows the operator to interact dynamically to adjust the input parameters, and thus improve result quality. The Clique Management web interface allows the operator to interact with the Clique Coordinator to dynamically add additional cliques, or remove cliques, to meet processing requirements. It also allows adding or deleting nodes from the node pool for a dynamic clique allocation policy. The Clique Coordinator provides feedback to the Web interface operator each time that an execution request cannot be processed because there is no clique available. These execution requests are dropped to avoid a growing backlog of processing and make it possible to keep up with real-time requirements. The inherent latency in the production of corrected images by PCID is caused by processing delays. For a typical PCID run, the Ensemble Publisher collects 200 raw frames into an ensemble each second. Since each ensemble takes about four seconds to process, a minimum of four cliques are needed to support a real-time flow. The latency for each PCID execution depends on its input parameters. If the latency for a single-run is five seconds, five cliques would be required to assure that a clique is available for processing when execution is requested for an incoming ensemble. Since execution times may vary, assigning additional cliques that may be rarely used is sometimes appropriate.

The Result Management interface can be used to start the Result Subscriber (RS). It is used to control result sharing and for deleting results. Archiving results is done by the Clique Master under the control of user inputs. The Result Management and User Management interfaces combine to assure that the user's view of results can be mapped to the actual results, and that the results are shared appropriately. Viewers can be configured to view a sampling of the raw images beside each corrected image, to demonstrate the improvement from processing. In our current implementation, the Ensemble Publisher, which handles all raw input images, makes the raw images available for display by copying them to a local web server. Another approach would be to have a separate subscriber to the raw image stream that would move some of the raw images to storage that is accessible to the web viewer.

4. Interactive HPC Information Flow

The INSPIRE architecture was designed to support flexible information flow, which maximized the benefit of the investment in expensive shared observatory resources. A goal of the design was to make data available at all stages of processing, which resulted in the configuration of processing pipelines to share both raw and processed data. In addition to allowing the shared viewing of processed results, the system allowed the sharing of the raw input stream and the partially-processed data. The pub/sub paradigm allowed multiple users to subscribe to the raw stream and process it independently, allowing several experiments with different codes to occur simultaneously. For example, two codes could operate from a single stream of raw sensor data, with each code having different calibration requirements. Also, image analysts who are seeking the optimum results with the best correction possible could initiate several processing streams for the same code with the same raw image stream, but vary input parameters for the code. The INSPIRE system can also implement a video production environment where multiple streams of corrected imagery are produced simultaneously, with the best output being selected for result publication at any given time. This is similar to the model used in televising sporting events, where the best image streams are selected from a variety of camera feeds.

Supporting a flexible interactive HPC information flow required the publication of "advertisements", so that others can become aware of data streams that can be viewed or processed. Figure 3 illustrates this approach, which is currently being implemented. A sensor publisher/advertiser is started when a stream of raw data is planned. The operator may select one of the previously collected datasets, corresponding to an orbit pass of an associated object, including the Hubble telescope, the Atlantis space shuttle, and the Seasat satellite, along with inputs to control the processing. These datasets were used to simulate inputs from a live sensor. In addition, the operator can select the time of publication. The raw sensor stream information is published periodically in the form of an advertisement. Authorized users that are interested in the raw stream will have used their Web interface to set up a subscription according to the stream characteristics, the name of telescope, and so forth. The advertised stream information will appear on their interface, as shown in Figure 3. The user may then start a process to create ensembles (collections of image frames) from the raw data by starting an ensemble publisher/advertiser code. Several users can simultaneously create ensemble publisher/subscribers to process the same data with different characteristics. A Clique Coordinator can then be started, based on the information associated with the advertisement for a planned ensemble stream, to process that stream according to selected input parameters.

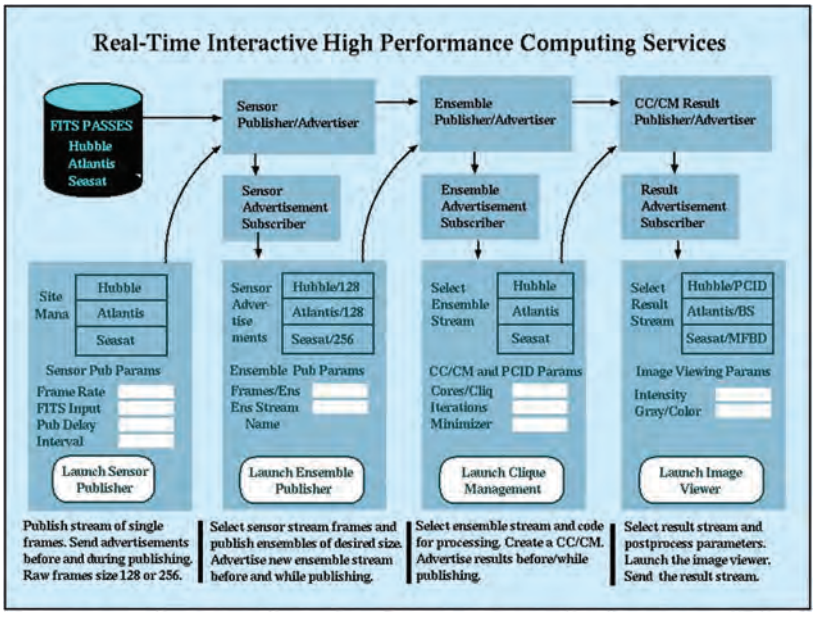


Figure 3. Interactive HPC Model

Advertisements allow raw and processed data streams to be associated with processes that perform the next step in the processing pipeline. The pub/sub paradigm provides the ability for all interested users to process and view data at any stage of processing. A web-enabled pub/sub interface makes it easy to discover interesting data streams through subscriptions

that specify associated characteristics. The flexible access to data streams and processing resources, supported by INSPIRE, will enable scientists to collaborate in real-time, and allows image analysts to direct their attention to the most relevant information sources.

5. Heterogeneous Implementation Issues

The PCID code has been ported to run on the Cell-BE processors in the SONY Playstation 3 (PS3) nodes of the Condor heterogeneous cluster. Each of Condor's Xeon head-nodes has 22 Playstations directly connected through 1 Gbit connections to accommodate the PS3 network speed. Condor head-nodes have two 10 Gbit connections to support the data flow from the PS3s. In adapting the PCID code to the PS3s, we encountered implementation and configuration issues due to the heterogeneous hardware as well as PCID-specific issues.

The Clique Coordinator and Clique Masters were configured to run on the Condor Xeon Nodes. PCID worker nodes are configured with a "master" process that distributes work to "worker" processes. Experiments were performed with the master PCID process on the Xeon nodes and with the master process on a PS3 node. Workers were assigned to PS3s. The distribution of PCID processes across Condor nodes was controlled by Message Passing Interface (MPI) commands and control files.

Additional configuration was required to support a heterogeneous Xeon/PS3 environment, to assure that proper tools were available, and to adapt the development environment. With respect to hardware heterogeneity, the Xeons are little-endian, while the PowerPC processor on the Cell-BE is big-endian, resulting in problems for code generation. The development tool-chain has to be built to cross-compile for the PS3s. Cross-compilation is highly recommended, as the Cell-BE PowerPC is very slow, compared to the Condor Xeon. Other tools, like the MPE libraries, were difficult to configure for our heterogeneous environment. Openmpi was the only implementation of MPI that supported heterogeneous codes by byte-swapping for big/little endian conversion. To take advantage of this feature, the code must be written to specify each data-type, rather than defining typed arrays as MPI_BYTE type, as is often found in MPI codes.

Architecture differences also resulted in implementation issues for the PCID code. The code is written to take advantage of the large amounts of memory (24GB) found on the Condor nodes. The PS3s have only 100MB, and some of that memory is used by the operating system. Available memory limited the processing for the PS3s to two image frames per node. On the Xeons, assigning at least four frames per node is optimal for execution, depending on specific input parameters. The CPU speed differential also affects the PCID code for the PS3 implementation. The PCID code has been optimized to minimize FFTs, since they were a large percentage of processing in the early stages of optimization. For the PS3s; however, this results in increased processing time on the PowerPC, which has a relative disadvantage to the Xeon. The PS3 advantage of faster FFT hardware contributes less performance improvement.

So far our results show the potential of the heterogeneous Xeon/PS3 Condor cluster to approach its theoretical performance gains within our interactive HPC framework. Early experiments have shown that performance optimization in this environment adds some complexity over homogeneous environments. It also shows that performance improvement at lower power consumption and cost/flop is possible with heterogeneous architectures.

6. Conclusion - Interactive Computing Potential

When interactive computing began appearing in universities in the early 1970s, it resulted in increased productivity. Time-sharing systems allowed engineers to execute their codes without waiting for an operator to introduce them to the system from a batch queue. Since then the UNIX "shell" has evolved to allow for the easy construction of "commands" for execution. Graphical user interface (GUI) interfaces began to evolve in the 1980s when bit-map graphics appeared, allowing users to issue commands by using a mouse along with a keyboard. Voice-driven interfaces provide another way to interact with a computer, telling it which programs to run and providing inputs.

The shift from batch to interactive led to new models for processing involving pipelining data between codes using the UNIX "pipe". Scripting languages like Bash, Awk, Perl, and Python allowed codes to be used together in creative new ways that further enhanced the power of interactive computing.

Web technologies have made it possible for users to interact with remote computers as if they were local. Also, HPCs are now proliferating, as they can be purchased by small companies and universities. Using HPCs for interactive computing can take advantage of web interfaces to support creative ways of working with codes. Users can even interact with multiple remote HPCs simultaneously. Tools are needed to efficiently deliver the data for processing and to disseminate the results. One of our goals in developing new approaches, based on the Phoenix information management system, is to assure that the results are rapidly available for all authorized users. Inputs for codes should also be available for scientists and information

analysts that need to process the inputs with their own parameters. The availability of raw and processed streams of data provides opportunities for scientists to experiment with their own codes, in combination with other codes that would not be accessible without an integrated communications environment.

The early efforts of AFRL in developing models for interactive HPC have evolved along with new techniques for information management to provide user interfaces and development tools that simplify HPC system development and access. This has allowed scientists and engineers to experiment with new techniques in their disciplines and to experiment with ways to interact with other disciplines. As a result, new systems are being developed that enhance the capabilities of analysts while encouraging innovation, to exploit existing technology and develop new technology.

Acknowledgements

Many thanks to the INSPIRE Team in Maui, Hawaii: Kathy Borelli, Adam Mallo, Jason Addison, Brad Farnsworth, Ron Viloria, Bruce Duncan, and Chris Sabol.

References

[Ramseyer, 2001] Ramseyer, G.O., S.E. Spetka, R.W. Linderman, and B.C. Romano, "Rapid C4I High Performance Computing for Hyperspectral Imaging Exploitation", 6th International Command and Control Research and Technology Symposium, U.S. Naval Academy, Annapolis, MD, June 19–21, 2001.

[Spetka, 2002] Spetka, S.E., G.O. Ramseyer, R.W. Linderman, and J. Gilmore, "The FrameWork: An Open-Architecture for Hyperspectral Image Exploitation", GOMAC 2002, The Government Microelectronics Applications ITAR Restricted Conference, Hyatt Regency Monterey and Naval Postgraduate School, Monterey, CA, March 11–14, 2002.

[Ramseyer, 2002] Ramseyer, G.O., S.E. Spetka, R.W. Linderman, and B.C. Romano, "The FrameWork: An Open-Architecture for Very-Large Image Exploitation", 2002 Command and Control Research and Technology Symposium, Naval Postgraduate School, Monterey, CA, June 11–13, 2002.

[Spetka 2002] Spetka, S.E., G.O. Ramseyer, R.W. Linderman, and M.J. Moore, "A Software FrameWork for HPEC System Development", Sixth Annual Workshop on High Performance Embedded Computing, MIT Lincoln Laboratory, September 24–26, 2002.

[Barnell, 2011] – Barnell, M., "DHPI: Integration, Development and Results of the 500 Tflop Heterogeneous Cluster (Condor)", *Proceedigs of the DoD HPCMP Users Group Conference*, 2011 (to appear).

[Matson, 2007] Matson, Charles L., Charles C. Beckner, Jr., Kathy Borelli, Tom Soo Hoo, Shiho You, Brandoch Calef, Maria Murphy, and Ron Viloria, "PCID and ASPIRE 2.0 – The Next Generation AMOS Image Processing Environment", Advanced Maui Optical and Space Surveillance Technologies Conference, Wailea Beach Marriott Resort, Maui, HI, 2007.

[Linderman, 2009] Linderman, R.W. , S.E. Spetka, S. Emeny, and D. Fitzgerald, "Enhancing Image Processing Performance for PCID in a Heterogeneous Network of Multi-code Processors", Advanced Maui Optical and Space Surveillance Technologies Conference, Wailea Beach Marriott Resort, Maui, HI, September 1–4, 2009.

[Grant, 2009] Grant, Rob, Vaughn Combs, Jim Hanna, Brian Lipa, and Jim Reilly, "Phoenix: SOA-based information management services", *Proc. SPIE*, Vol. 7350, 73500P (2009); doi:10.1117/12.817911, Defense Transformation and Net- Centric Systems 2009, Orlando, FL, 14 April 2009.

[Spetka, 2010] Spetka, S.E, S. Tucker, and G.O. Ramseyer, "Fawkes Information Management for Space Situational Awareness", Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS), Maui, HI, September 14–17, 2010.

[Yan, 2005] Yan, L.K., G.O. Ramseyer, R.W. Linderman, and E. Balster, "Video Teleconferencing on the Distributed Interactive HPC Testbed (DIHT)", 2005 DREN Networking & Security Conference, Charleston, SC, October 17–21, 2005.

[Spetka, 2008] Spetka, S.E., G.O. Ramseyer, and R.W. Linderman, "Fault-Tolerant Integrated Information Management Support for Physically-Constrained Iterative Deconvolution", Applied Imagery Pattern Recognition (AIPR) Workshop: Multiple-Image Information Extraction, Cosmos Club, Washington, DC, October 15–17, 2008.

[Tucker, 2008] Tucker, S., S.E. Spetka, G.O. Ramseyer, S. Emeny, D. Fitzgerald, and R.W. Linderman, "High-Performance Information Management for HPC Parallel Computing," *Proceedings of the DoD HPCMP Users Group Conference 2008*, pp. 409–412, 2008.

HPCMP UGC 2011

5. Signal/Image Processing (SIP) and Sensors; Electronics, Networking, and Systems/C4ISR (ENS) and Testing

Networking and Systems C4ISR (ENS)

Entropy-Based Transformation for Characterizing Heavy-Tailed Distributions in Network Traffic

Keesook J. Han, Virginia W. Ross, and Christopher Hall
US Air Force Research Laboratory/Information Directorate (AFRl/RI), Rome, NY
{keesook.han, virginia.ross, christopher.hall.ctr}@rl.af.mil

Abstract

Designing a reliable and real-time network monitoring service for network traffic analysis is an increasingly challenging task. Studying heavy-tailed distributions in network traffic is an important research topic in performance modeling and evaluation of computer network systems. Characterizing heavy-tailed network traffic plays a critical role in improving the Quality of Services. There are two types of heavy- tailed distributions, one-tailed and two-tailed distributions. Heavy-tailed distributions can be used to simulate network traffic. However, due to the difficulty of parameter estimation, heavy-tailed distributions are limited in their ability to characterize real-time network traffic. The Entropy-Based Heavy-Tailed Distribution Transformation (EHTDT) has been developed to convert the heavy-tailed distribution into a transformed probability distribution to characterize real-time network traffic analysis. In practice, the entropy distribution of the transformed probability distribution exhibits a linearity parameter that allows one to characterize real-time network traffic. Theoretical EHTDT and simple estimation approaches of parameters for heavy-tailed distributions are introduced in this paper. Moreover, rate-controlled eigen-based coding has been developed to obtain other parameters to extract the compact information from the EHTDT data. Some results of applying this method to real-time network traffic data are presented here.

1. Introduction

With communication systems growth and increased traffic, along with increasing reliance on network traffic to satisfy communication needs, networks require high-Quality of Services (QoS) for reliable communication. There has been a large increase in statistical analysis studies to characterize heavy-tailed network traffic traces for performance modeling and evaluation of computer network systems. Traffic traces have non-Gaussian with mixtures of heavy-tailed statistical distributions because network traffic has dynamic behavior. Therefore, various parameter estimations are required for characterizing network traffic.

Minimizing the computational complexity to estimate parameters is necessary for real-time network traffic monitoring and visualization. Efficient methods are required to effectively characterize dynamic and mixed traffic behaviors in large-scale network environments. This research developed a computationally feasible statistical approach to represent heavy-tailed distributions. These unconventional estimators are extracted to streamline calculations with a low-latency.

This paper organized as follows: Sections 2–4 present a brief survey of heavy-tailed distributions, data analysis, performance modeling and evaluation; Sections 5–6 describe the EHTDT and rate-controlled eigen-based coding with real-time traffic data experiments; and Section 7 describes the DoD contribution. Finally, Section 8 presents the conclusion.

2. Heavy-Tailed Distributions

Heavy-tailed distributions are probability distributions. Heavy-tails are not exponentially bounded, since their tails are heavier than the exponential distribution. A heavy-tailed distribution can have either one or two tails. Some examples of one-tailed and two-tailed distributions are as follows:

- One-tailed distributions include the Pareto, the Log-Normal, the Log-Gamma, the Lévy, the Burr, and the Weibull distributions with a shape parameter of less than 1.
- Two-tailed distributions include the Cauchy, the t-, and the skew log-normal cascade distributions.

In general, there are heavy-tailed distributions whose tails follow a power law with a low exponent, and there are mixtures of heavy-tailed distributions (Dasgupta, et al., 2005; Vempala and Wang, 2004).

3. Heavy-Tailed Network Traffic Data Analysis

Transmission control protocol (TCP) applications, such as HTTP and SMTP, show spatial and temporal invariances in network traffic (Dainotti, et al., 2006). Traffic is viewed at different levels ranging from the larger session, through mid-levels such as the packet, down to the smallest level, the byte. The main focus is on packet-level characterization, illustrating traffic flows in inter-packet-time and packet-size. Single-flows produce long-range-dependent (LRD) correlations and non-Gaussian marginal distributions (Sarvotham, et al., 2001). Traffic bursts typically come from a single high-volume connection. Dominating connections are called alpha traffic. Alpha traffic happens when large file transmissions are made over high-bandwidth links. If the alpha-traffic is removed, one is left with LRD-Gaussian beta-traffic (Sarvotham, et al.). Pareto's principle is the effect where a few traffic elements make a major contribution to a traffic analysis (Sarvotham, et al.). Traffic characterization is integral to queuing systems (Elleithy and Al-Suwaiyan, 2001). This characterization helps to specify important QoS parameters, including buffer-size and link-capacity, and helps predict the bandwidth requirements for congestion control. Discrete-time linear scale-invariant (LSI) systems formulated by Rao and Lee (2000) can synthesize self-similar data that have the same parameters as those estimated from real traffic. There are other heavy-tailed distribution applications, such as heavy-tailed distribution of cyber-risks (Maillart and Sornette, 2010; Baiardi, et al., 2009), efficient recording and retrieval of high-volume network traffic (Kornexl, et al., 2005), heavy-tailed modeling techniques (Resnick, 1997; Willinger, et al., 1998), and heavy-tailed network traffic characterization methods (Ashir, et al., 2001).

4. Heavy-Tailed Distribution Performance Modeling and Evaluation

Heavy-tailed distribution performance modeling and evaluation show various distinctive characteristics (Crovella, 2001). These distributions have infinite variance and relate to self-similar modeling (Sarvotham, et al., 2001; Rao and Lee, 2000; Willinger, et al., 1998; Willinger, et al., 1996). Various Internet traffic characterization techniques exist (Veres, et al., 2000; Rezaul and Grout, 2009; Crovella, et al., 1998). The self-similar model may have a range of time-scales. For example, individual hosts generate network traffic patterns; creating textured plots for source-destination packets and arrival times. Data-points are shown individually for further analysis. A host's network traffic is in distinguishable bursts before the source-destination split. Barford and Crovella (1998) reported characteristics of the Web workload for network and server performance. Crovella (2001) described two properties that capitalize on heavy-tailed distributions to improve computer systems' network performance. The first property is the conditional expectation, which depends on the declining hazard-rate. The second property is the mass-count disparity. Examples include load-balancing for distributed systems, scheduling Web servers, and Internet routing and switching.

5. Entropy-based Heavy-Tailed Distribution Transformation (EHTDT)

Network traffic characterization has been studied extensively, but an accurate characterization of network traffic still remains elusive due to the difficulty of parameter estimation. This section describes the transformation procedures to characterize network traffic data. A simple transformation process can predict large-scale network traffic behavior. In general, network traffic data have heavy-tailed distributions. Power-law distributions are widely-used for estimating packet interval times, as well as in other networking applications. For example, a power-law distribution is one-tailed and a Cauchy distribution is two-tailed. Two-tailed distributions were selected for deriving the generalized EHTDT.

5.1 Forward EHTDT

This subsection describes the transformation procedures, employing real-time network traffic data. Figure 1 depicts the simple procedures to extract two-tailed distribution information from raw network traffic datasets. Frequency patterns of network traffic are important features to control traffic throughput. Frequencies of network connections are shown at the bottom in Figure 1, and exhibit heavy tails. Plots of daily connections, the first-order difference along connections and histogram of heavy-tailed are shown. Note: diff(connections(t))=connections(t) - connections(t-1) where t>1. Frequencies of network connections are shown at the bottom in Figure 1, and exhibit heavy tails. This histogram approach is used to get a probability mass function for the experiments. Experimental results indicate that it is difficult to use these heavy-tailed distributions to characterize network traffic. For example, Figure 2 approximately shows a Cauchy distribution but other experimental data-sets can show mixtures of heavy-tailed distributions. Hence, the Forward EHTDT has been developed for fast network traffic characterization for mixtures of heavy-tailed distributions.

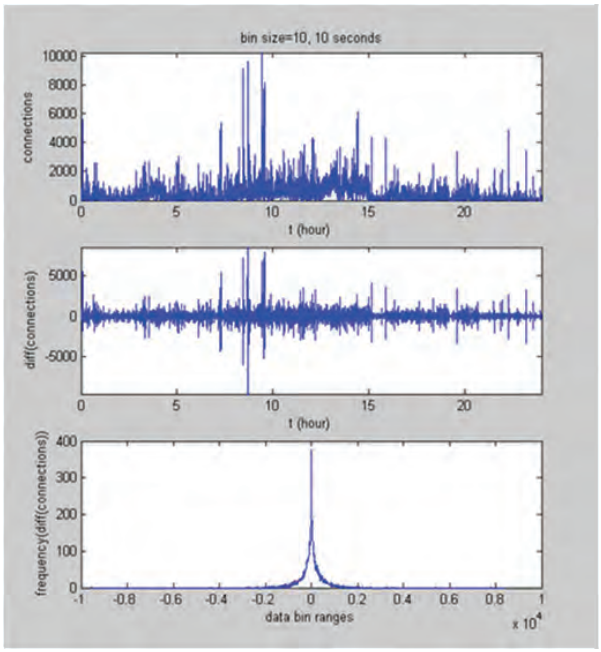


Figure 1. Network connections, first-order difference, and histogram

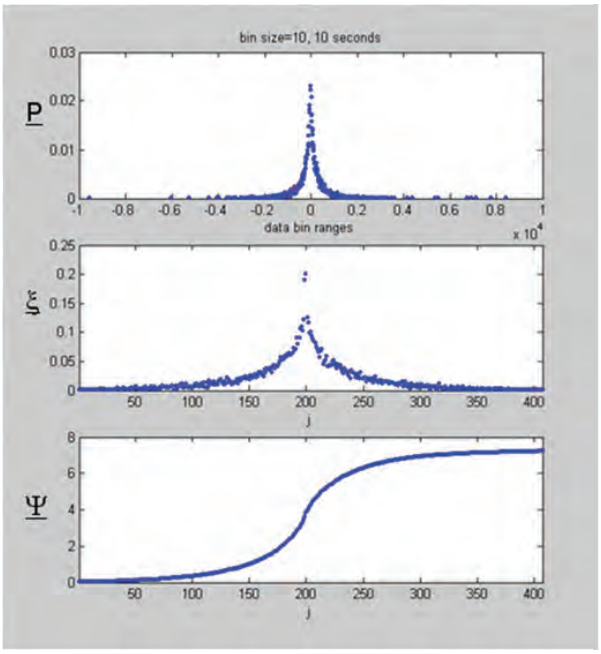


Figure 2. EHTDT processing using Probability Mass Function

A two-tailed Probability Mass Function is defined as a probability vector:

$$\hat{P} = (\hat{P}(1), \hat{P}(2), ..., \hat{P}(K-1), \hat{P}(K), \hat{P}(K+1), ..., \hat{P}(N)) \tag{1}$$

where $\hat{P}(K)$ is the unique maximum element of $\hat{P}$ and N is the maximum index number. Note: $\underline{P}$ is the same as $\hat{P}$ in Figure 2.

In statistics, measurements of statistical dispersion are important to describe quantitatively the degree of variation or dispersion of values in a population or in observed data-sets. Statistical dispersion methods use measurements of central tendency and location. Measurements of dispersion include standard deviation, range, distance standard deviation, mean-

difference, etc. These common measurements are not suitable to measure statistical dispersion for network traffic data. Hence, the new QoS measurements of statistical dispersion for heavy-tailed network traffic data were developed.

The normalization factor λ is defined by:

$$\lambda = \sum_{x=1}^{N} (1 - \hat{P}(x)) \tag{2}$$

Since heavy-tailed distributions lead to overdispersion, traditional dispersion measures of central tendency can be misleading. Current QoS measurement tools use maximum and minimum values, average, variance and standard deviation. These parameters provide insufficient information to measure statistical overdispersion. There is considerable literature on overdispersion parameter estimation techniques, and on quantification of the effects of overdispersion (Cameron and Trivedi, 1990; Czado, 2002). However, existing statistical measurements employed in these techniques are not practical for real-time QoS measurements. A novel rapid QoS measurement tool is proposed for estimating the statistical dispersion parameter λ that has previously been used to characterize dynamic heavy-tailed distribution behaviors of network traffic data.

Experimental results indicate that the parameter λ can accurately describe the spreading characteristic of heavy-tailed information. This research used the CAIDA Distributed Denial-of-Service (DDoS) Data (Source: http://www.caida.org/data/passive/ddos-20070804_dataset.xml) and selected the number of packets transferred over a network to demonstrate the EHTDT performance. An analysis of this DDoS attack data is presented by Han, Hodge, and Ross (2011). The probability mass function presents that the (DDoS) attack not only drastically increases the number of data-points, but also causes the data-points to increase in spread. The sudden increase in histogram data-points and data-spread drastically lower the probability of experiencing a specific change in packets transferred over the network. In Figure 3, the λ progression exhibits an increase in λ of approximately two orders-of-magnitude during the DDoS attack period, for all bin sizes except for the shortest time interval, 1ms.

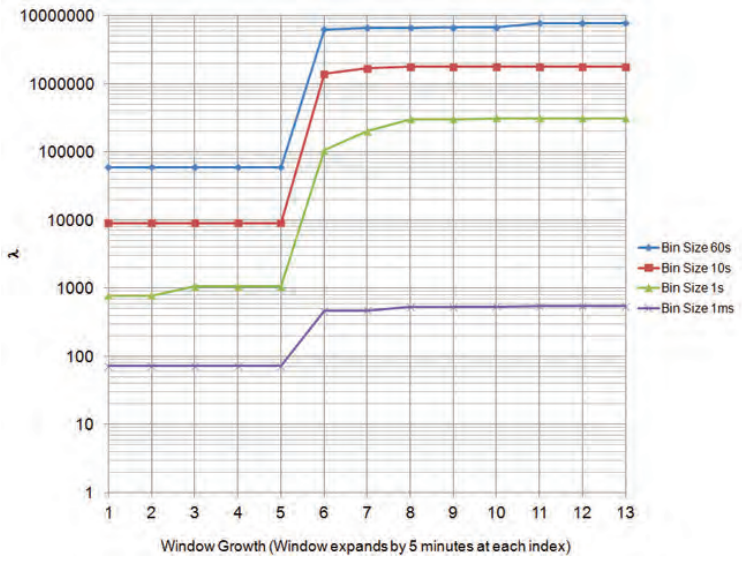


Figure 3. λ (Normalization Factor) Progression vs. a Growing Window

To characterize network traffic in a linearized form of cumulative distribution, the probability vector $\hat{P}$ is converted into a transformed probability vector P by the following two procedures:

The vector β is defined, such that:

$$\beta(x) = \frac{1 - \hat{P}(x)}{\lambda} \tag{3}$$

where x=1,2,…,N and λ is used for the normalization factor.

The Transformed Probability Mass Function P is defined as a transformed probability vector:

$$P = (\beta(K), \beta(K-1), \ldots, \beta(1), \beta(N), \beta(N-1), \ldots, \beta(K+1)) \tag{4}$$

where $P(1)=\beta(K)$ is the minimum element of P.

The Entropy-Based Heavy-Tailed Distribution ξ is defined by:

$$\xi(j) = -\mathrm{P}(j)\log_2 P(j) \tag{5}$$

where j=1,2,…,N.

The Cumulative Entropy-Based Heavy-Tailed Distribution Ψ is defined by:

$$\Psi(x) = \sum_{j=1}^{x} \xi(j) \tag{6}$$

where x=1,2,…, N.

Note that the last element of Ψ is the entropy of the transformed probability vector P

$$\Psi(N) = \sum_{x=1}^{N} \xi(x) = -\sum_{x=1}^{N} P(x)\log_2 P(x) = H(P) \tag{7}$$

The reason to transform probability as part of a regression analysis is to achieve linearity. In practice, the proposed transformation provides approximate linearity as shown in Figure 4.

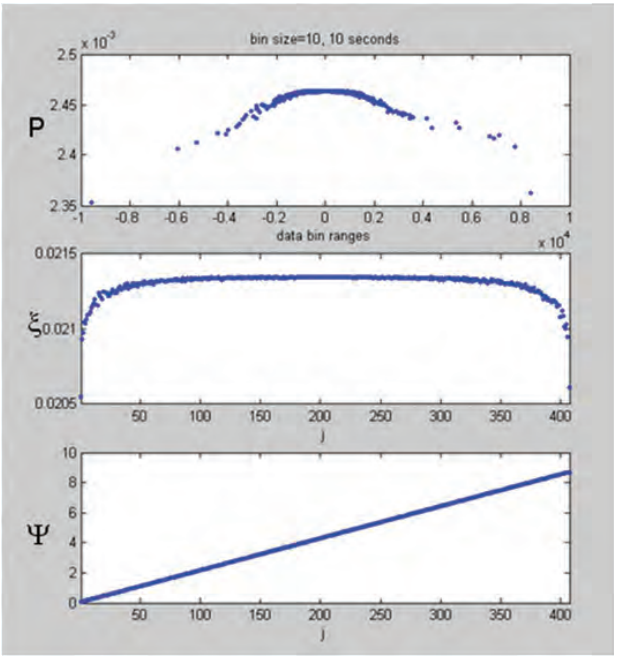


Figure 4. EHTDT processing using Transformed Probability Mass Function

Benefits of applying the Transformed Probability Mass Function P are as follows:

- Normal traffic ranges are suppressed and bursty traffic regions are exposed.
- It increases the flatness of the Entropy-Based Heavy-Tailed Distribution (ξ).
- It provides linearization of the Cumulative Entropy-Based Heavy-Tailed Distribution (Ψ).

For example, (ξ,ξ) and (Ψ,Ψ) are computed by Equations 5 and 6, respectively. Figure 2 and Figure 4 illustrate the effectiveness of linearization of Cumulative Entropy-Based Heavy-Tailed Distribution. Experimental results indicate that linearization can be only achieved by applying the Transformed Probability Mass Function (see Figure 2, nonlinear function Ψ and Figure 4, linear function Ψ).

Forward EHTDT provides linearization of the Cumulative Entropy-Based Heavy-Tailed Distribution. Simple linear regression can be used to fit a predictive linear function to observed data (Ψ). Other QoS measurements of statistical dispersion are the slope and x-y intercept of Ψ. This process is also useful for visualizing real-time network traffic. One of the principle concerns of the EHTDT is the capability of linearization. Experimental results show that applying the EHTDT

to traffic data produces a linear representation of network activity, and the derived parameters are capable of exhibiting changes in network activity. Both the CAIDA Distributed Denial-of-Service (DDoS) Data (Source: http://www.caida.org/data/passive/ddos-20070804_dataset.xml) and live-traffic data were analyzed. Both experimental results show that both traffic data-sets produce linear representations of network activities. Slopes of Cumulative Entropy-Based Heavy-Tailed Distribution can be simply measured for both stored- and live-traffic data. Figure 5 shows that the slopes of Cumulative Entropy-Based Heavy-Tailed Distribution are lowered by nearly two orders-of-magnitude during the DDoS attack period for all bin sizes except for the shortest time interval (1ms).

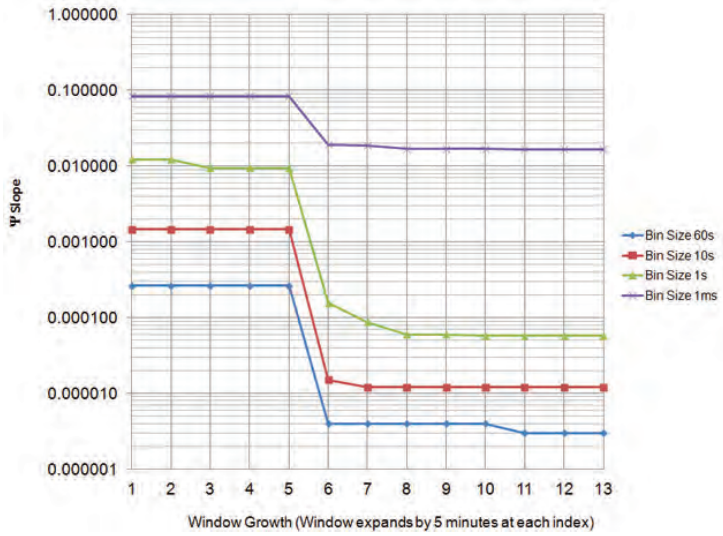


Figure 5. Ψ Slope vs. a Growing Window

5.2 Inverse EHTDT

The inverse EHTDT can be determined by a series of steps. The first-order differences of Ψ are used to determine ξ as follows:

$$\xi(x) = \Delta(\Psi) = \Psi(x) - \Psi(x-1) = -\mathrm{P}(x)\log_2 P(x) \tag{8}$$

where $\xi(1)$ iss a stored parameter and x=2,3,…,N.

It is emphasized that one can deal directly with P (or an estimate of P via an iteration technique), and then obtain the initial heavy-tailed probability vector $\hat{P}$ from P. Inverting (3), the probability vector β is obtained as:

$$P = (\beta(K), \beta(K-1),\ldots, \beta(1), \beta(N), \beta(N-1),\ldots, \beta(K+1)) \tag{9}$$

Then, $\hat{P}$ can be calculated via:

$$\hat{P}(x) = 1 - \lambda\beta(x) \tag{10}$$

where the normalization factor λ is a stored parameter and x=1,2,…,N.

5.3 EHTDT Decomposition of Linear and Nonlinear Functions

The Inverse Entropy-Based Heavy-Tailed Distribution Transform is used to enable processing for retrieving the Probability Mass Function. In practice, the Probability Mass Function is not necessary for perfect retrieval. Estimation procedures for the entropy-based probability distribution are introduced in this section.

The Entropy-Based Heavy-Tailed Distribution Ψ can be decomposed as:

$$\Psi = \Phi + \eta. \tag{11}$$

The sum of a linear function Φ and a nonlinear function η where Φ is taken as:

$$\Phi(x) = \xi(1) + \frac{\Psi(N) - \Psi(1)}{N-1}(x-1) \tag{12}$$

where x=1,2,…,N.

Φ will contain most of the energy of the heavy-tailed information; the linear distribution of the heavy-tailed approximation can be estimated with the entropy of the inverse distribution Ψ(N) and ξ(1). The nonlinear function η is then:

$$\eta = \Psi - \Phi. \tag{13}$$

The differential vector ε of nonlinear function η is given by:

$$\varepsilon(x) = \eta(x) - \eta(x-1) \tag{14}$$

where x=2,3,…,N.

Data reduction and feature selections are two reasons to compress the nonlinear function η. Fourier coefficients, wavelet coefficients, and principal components are commonly selected for features. In this approach, the principal component analysis (PCA) is applied to select features and analyze anomaly detection data sets. The estimated nonlinear function $\hat{\eta}$ can be determined with a few principal components.

The reconstructed function $\hat{\Psi}$ can be expressed as:

$$\hat{\Psi} = \Phi + \hat{\eta}. \tag{15}$$

6. Low-Bit-Rate Compression

There is multiple analysis techniques employed for low-bit-rate compression. First, ordinary metric principal component analysis (PCA) and Singular Value Decomposition (SVD), and Fast Eigen-Based Coding are described. Then, the Rate-Controlled Eigen-Based Coding for EHTDT is introduced. The proposed coding system will reduce the dimensionality of the data enormously and capture the effective feature structure.

6.1 Principal Component Analysis

Principal component analysis is a popular technique in multivariate analysis for representing data in a compact form by grouping data into one or more principal components, based on similar data parameters. There are various generalizations of PCA, such as multiple correspondence analysis (MCA), non-metric principal component analysis (NCA) and ordinary metric PCA. In correspondence analysis, the variables are linearly transformed to provide orthogonal solutions.

For a set of observed M-dimensional non-zero mean-observation vectors $\{f_1, f_2,\ldots, f_n\}$, let μ be an M-dimensional mean-vector of the observation vectors $f_1, f_2,\ldots, f_n$. M-dimensional zero mean-observation vectors are given by:

$$\phi_x = f_x - \mu \tag{16}$$

where x=1,2,…,n.

The covariance matrix $\mathbb{S}$ is computed as:

$$\mathbb{S} = \frac{1}{n}\sum_{x=1}^{n} \phi_x \phi_x^T. \tag{17}$$

The unique set of M orthonormal eigen-vectors of $\mathbb{S}$, $Q_M=[q_1, q_2,\ldots, q_M]$, and their associated eigen-values, $\lambda_1, \lambda_2,\ldots, \lambda_M$ are computed. Linear combinations of the first L eigen-vectors $Q_L=[q_1, q_2,\ldots, q_L]$ corresponding to the L largest eigen-values (e.g., $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_L$) span the space of the zero mean-observation vector to capture most of the relevant information in the input data. The projection of the vector ϕx onto the lines spanned by the orthonormal basis $Q_L=[q_1, q_2,\ldots, q_L]$ is given by the following operation:

$$p_x = Q_L^T \phi_x = (p_{1x}, p_{2x}, ..., p_{Lx})^T \tag{18}$$

where $1 \leq x \leq n$.

The elements of vector p_x are called the principal components. The *Mx1* principal component vector p_x contains compact information for f_x. The reconstructed vector $\hat{f}_x$ can be computed as:

$$\hat{f}_x = Q_L p_x + \mu \tag{19}$$

where $1 \leq x \leq n$.

6.2 Fast Eigen-Based Coding

The Singular-Value Decomposition (SVD)-based compression method is popular for compressing large data matrices. The fundamental concept of the SVD-based compression scheme is to use a smaller number of dimensions to approximate the original matrix. The SVD does not provide a computationally-efficient method of data compression. However, the importance of using the SVD for principal component analysis is that SVD provides the standardized versions of principal component scores. Component scores are useful for correspondence analysis. Fast Eigen-Based Coding is suitable for massive traffic data compression and analysis.

The *empirical covariance matrix R* of *M-dimensional* non-zero mean observation vectors $\{f_1, f_2,\ldots, f_n\}$ is defined for computational simplicity. The MxM empirical covariance matrix R is defined by:

$$R = \frac{1}{n}\sum_{x=1}^{n} f_x f_x^T. \tag{20}$$

Instead of using the ordinary covariance matrix $\mathbb{S}$ or the correlation matrix, the *empirical covariance matrix R* of the non-zero mean-observation vectors is defined for computational simplicity. The covariance matrix $\mathbb{S}$ provides that the first principal component corresponds to a line that passes through the mean, and minimizes the mean-square error of approximating the data. On the other hand, the empirical covariance matrix R of the non-zero mean-observation vectors provides an effective cluster separation for each streaming network traffic dataset. For anomaly detection, the *empirical covariance matrix R* minimizes the computational complexity and maximizes the detection rate.

The unique set of *M* orthonormal eigen-vectors is computed with the correlation matrix *R* and the corresponding *L* eigen-vectors to form the *MxL* eigen-vector matrix Q_L. The first *L* eigen-vectors are $Q_L=[q_1, q_2,\ldots, q_L]$ and their associated eigen-values are $\lambda_1\geq\lambda_2\geq\ldots\geq\lambda_L$.

The Fast Eigen-Based Coding pairs are given by:

$$P = Q_L^T X \tag{21}$$

and:

$$\hat{X} = Q_L P \tag{22}$$

where $L<M$, and $X=[f_1, f_2,\ldots, f_n]$ and $\tilde{X} = [\tilde{f}_1, \tilde{f}_2,\ldots, \tilde{f}_n]$.

Note that the columns of matrix $P=[p_1, p_2,\ldots, p_n]$ are the *L-dimensional* principal component vectors $p_1, p_2,\ldots, p_n$,and the columns of matrix $\tilde{X}$ are the reconstructed M-dimensional vectors $\tilde{f}_1, \tilde{f}_2,\ldots, \tilde{f}_n$. This coding scheme is simpler than the *Karhunen-Lòeve* transform and other principal component analysis techniques.

The classes of admissible transformations in SVD are different for different types of data. Admissible transformations should be found to minimize the appropriate loss function. It is common to calculate the principal components using a covariance matrix $\mathbb{S}$, because eigen-vectors of a covariance matrix may provide admissible transformations. There are other ways of computing principal components. In one method, eigen-vectors of a non-zero correlation matrix are used to compute principal components. Even though the loss function is not minimized in the coding scheme, the scheme does yield a reduction in the computational complexity and misclassification rate.

6.3 Rate-Controlled Eigen-Based Coding for Data Analysis

Suppose that the *Nx1* non-zero mean-observation vector ε is given by:

$$\varepsilon(x) = \eta(x) - \eta(x-1) \tag{23}$$

where $\xi(1)$ is a stored parameter and:

$x=\{2,\ldots,x1,(x1+1),\ldots,x2,\ldots,(x2+1),\ldots,x3,\ldots,(x3+1),\ldots,x4,(x4+1),\ldots,N\}$ (note: region A: 2,…,x1, region B: (x1+1),…,x2, region C: (x2+1),…,x3, region D:(x3+1),…,x4, region E: (x4+1),…,N), as shown in Figure 6.

Piece-wise regression analysis is appropriate to characterize network traffic. For region C, with x ranging between (x2+1),…,x3, ε(x) can be taken as zero for all practical purposes; this gives $f(x)$ equal to zero in region C (abnormal behavior region), and collect non-zero mean-observation vectors in region A (normal behavior region), B, D, and E (normal behavior region). However, region C data is the most important region to measure anomalies. Especially, sparse data analysis is required to obtain *tail characteristics* for feature extraction in region C, and Fast Eigen-Based Coding is good to apply in regions A, B, D, and E. Since region B and region D are close to zero, x2 and x3 can be adjusted to obtain the

fixed-length of *Mx1 non-zero mean-observation* vector *f* such as:

$$f = \{f_A, f_B, f_D, f_E\}. \quad (24)$$

Fast Eigen-Based Coding can be used to compress M-dimensional non-zero mean-observation vectors $\{f_1, f_2, \ldots, f_n\}$ to obtain compact network traffic information in A, B, D, and E regions for feature extraction.

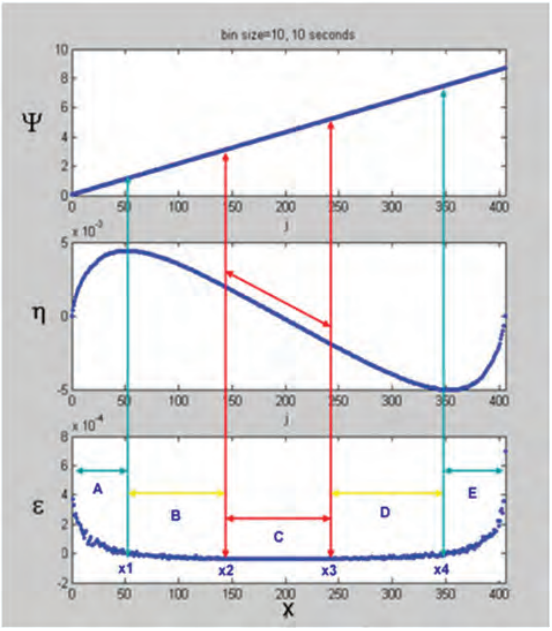


Figure 6. Rate-controlled eigen-based coding regions

Power-law distributions are widely used for estimating packet interval time as well, as in other networking applications; and ε-contaminated (Gaussian-mixture) distributions are useful to detect anomaly network traffic. The estimation of important tail characteristics is directly linked to the interpretation of the underlying network traffic. Since mixture distribution characteristics of heavy-tailed network traffic limit one's ability to estimate parameters of various heavy-tailed distributions, an efficient and practical parameter estimation technique has not yet been derived. For statistical anomaly detection, heavy-tailed probability distributions of network traffic data have been proposed to mitigate parameter estimation limitations. In this work, the EHTDT converts such a heavy-tailed distribution into a transformed probability distribution, which is more amenable for loss compression of network traffic data to obtain meaningful features.

7. Significance to the DoD

The Department of Defense has a strong need to monitor network traffic to protect their networks against threats. The ability to reliably monitor network traffic in real-time contributes significantly to this mission. Conventional heavy-tailed analysis and models are not appropriate for estimating parameters of dynamic and massive time-varying distributions. Moreover, such conventional approaches have not produced effective information metrics for mission-critical operations. This research took actual network traffic and employed the EHTDT transform to extract compact, usable results of network traffic trends. These results contribute significantly to this capability of real-time network traffic monitoring.

Statistical network traffic characterization for QoS measurement to control computer network systems is becoming more difficult with the increasing size and amounts of information handled by these systems. To address this concern, a transformation method for heavy-tailed probability distribution network traffic data has been developed. In this work, the EHTDT transform, using real-time and stored data, has converted such a heavy-tailed distribution into a transformed probability distribution more amenable to obtain QoS measurements. In experiments with real-time network traffic data, the transformed distribution was used to characterize the network traffic. A compact characterization of heavy-tailed network traffic data was achieved by the EHTDT transform and the rate-controlled eigen-based coding. The normanlization factor λ, linearized function Ψ, Ψ(N), $\xi(1)$, $\xi(N)$, sparsity of region C data, and compact information in regions A, B, D, and E are good features for QoS measurements. Future research will focus on temporal granularity and statistical

characteristics of other network traffic header information to analyze mixtures of heavy-tailed distributions. Intervals or bin sizes and sliding windows for real-time traffic characterization will be studied to obtain effective QoS measurement to control network throughput.

Acknowledgements

This material is based upon work supported by the US Air Force Office of Scientific Research in-house project No. 08RI03COR, the US Air Force Research Laboratory (AFRL) in-house Job Order Number 231GIH02. The DoD High Performance Computing Modernization Program supported this project by supplying supercomputer time under the Project 2646, at the AFRL Affiliated Research Center in Rome, NY.

References

Ashir, A., Suganuma, T., Kinoshita, T., Roy, T.K., Mansfield, G., and Shiratori, N., "Network traffic characterization and network information services—R&D on JGN", *Computer Communications*, 24, Issue 17, pp. 1734–1743, 2001.

Baiardi, F., Telmon, C., and Sgandurra, D., "Modeling and managing risk in billing infrastructures", Critical Infrastructure Protection III, *IFIP Advances in Information and Communication Technology*, 311, C. Palmer and S. Shenoi, Eds., Springer, Boston, pp. 51–64, 2009.

Barford, P. and Crovella, M., "Generating representative Web workloads for network and server performance evaluation", *ACM SIGMETRICS Performance Evaluation Review*, 26, Issue 1, pp. 151–160, 1998.

Cameron, A. and Trivedi, P., "Regression-based tests for overdispersion in the Poisson model", *Journal of Econometrics*, Vol. 46(3), pp. 347–364, 1990.

Czado, S. "Quantifying overdispersion effects in count regression", *Sonderforschungsbereich 386*, Paper 289, http://epub.ub.uni-muenchen.de/1667/1/paper_289.pdf, 2002.

Crovella, M., "Performace evaluation with heavy-tailed distributions", *Lecture Notes In Computer Science*, 2221 Springer-Verlag, London, pp. 1–10, 2001.

Crovella, M., Taqqu, M., and Bestavros, A., "Heavy-tailed probability distributions in the world-wide web", *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, R. Adler, R. Feldman, and M. Taqqu (editors), Birkhauser, Boston, pp. 3–25, 1998.

Dainotti, A., Pescapè, A., and Ventre, G., "A packet-level characterization of network traffic", *11th IEEE International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*, June 2006.

Dasgupta, A., Hopcroft, J., Kleinberg, J., and Sandler, M., "Learning mixtures of heavy-tailed distributions", *Proceedings of the IEEE 46th Annual Symposium on Foundations of Computer Science*, pp. 491–500, 2005.

Elleithy, K.M. and Al-Suwaiyan, A.S., "Network traffic characterization for high-speed networks supporting multimedia", *IEEE Proceedings of the 34th Annual Simulation Symposium*, p. 200, 2001.

Han, K., Hodge, M., and Ross, V., "Entropy-based heavy-tailed distribution transformation and visual analytics for monitoring massive network traffic", *Proceedings of SPIE 8019-13*, 2011.

Kornexl, S., Paxson, V., Dreger, H., Feldmann, A., and Sommer, R., "Building a time machine for efficient recording and retrieval of high-volume network traffic", *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, pp. 267–272, 2005.

Leland, W., Taqqu, M., Willinger, W., and Wilson, D., "On the self-similar nature of ethernet traffic", *IEEE/ACM Trans. Networking*, pp. 1–15, 1994.

Maillart, T. and Sornette, D., "Heavy-tailed distribution of cyber-risks", *European Physical Journal B*, 75, Issue 3, pp. 357–364, 2010.

Rao, R. and Lee, S., "Self-similar network traffic characterization through linear scale-invariant system models", *IEEE International Symposium on Personal Wireless Communications*, pp. 138–142, December 2000.

Resnick, S.I., "Heavy-tail modeling and teletraffic data", The Annals of Statistics, 25, No. 5, *Institute of Mathematical Statistics*, pp. 1805–1849, 1997.

Rezaul, K. and Grout, V., "An approach for characterising heavy-tailed internet traffic based on EDF statistics", *Intelligent Engineering Systems and Computational Cybernetics*, J.A. Tenreiro Machada, B. Patkai, and I.J. Rudas (editors), Springer, Netherlands, pp. 173–184, 2009.

Sarvotham, S., Riedi, R., and Baraniuk, R., "Connection-level analysis and modeling of network traffic", *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pp. 99–103, 2001.

Vempala S. and Wang, G., "A spectral algorithm for learning mixture models", *Journal of Computer and System Sciences (Archive)*, Special Issue on FOCS, 68, Issue 4, pp. 841–860, 2004.

Willinger, W., Paxson, V., and Taqqu, M., "Self-similarity and heavy-tails: structural modeling of network traffic", *A Practical Guide to Heavy-Tails: Statistical Techniques and Applications*, R. Adler, R. Feldman, and M. Taqqu (editors), Birkhauser, Boston. pp. 27–53, 1998.

Willinger, W., Taqqu, M., and Erramilli, A., "A Bibliographical guide to self-similar traffic and performance modeling for modern high-speed networks", *Stochastic Networks: Theory and Applications*, F.P. Kelly, S. Zachary, and I. Ziedins (editors), Clarendon Press, Oxford, 1996.

Veres, A., Kenesi, Z., Molnár, S., and. Vattay, G., "On the propagation of long-range dependence in the Internet", *ACM SIGCOMM Computer Communication Review Archive*, 30, Issue 4, pp. 243–254, 2000.

Implementing Autonomic Computing Methods to Improve Attack Resilience in Web Services

Weston P. Monceaux, Deland E. Evans, Keith N. Rappold, Cary D. Butler
US Army Engineer Research and Development Center (ERDC), Information Technology Laboratory, Vicksburg, MS
{weston.p.monceaux, deland.e.evans, keith.n.rappold, cary.d.butler}@usace.army.mil

Sherif Abdelwahed, Rajat Mehrotra
Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State, MS
sherif@ece.msstate.edu, rm651@msstate.edu

Abhishek Dubey
Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN
abhishek.dubey@vanderbilt.edu

Abstract

Over the last decade, web services have evolved from a fringe technology to a commonplace tool driving many information technology processes in both industry and the Department of Defense (DoD). Web services are used predominantly as a way for critical information systems to communicate in an automated fashion using standard protocols. Protecting these services from the types of attacks often aimed at web applications and other network services is an increasing concern. One method for protecting web services involves making them more resilient to attack using autonomic computing techniques. This paper presents our initial prototype for implementing autonomic computing methods to improve the resilience of web services. This initial prototype provides a platform for testing various aspects of autonomic computing techniques as they relate to the detection of software faults and malicious activity. The approach to threat detection used is a hybrid of standard open-source pattern-based tools in combination with a proposed anomaly-based detection engine. Intelligent, automated response to security-related events will increase the resilience of web services in hostile environments. This raised level of resilience will significantly improve the ability of DoD agencies to provide a high quality-of-service (QoS) to their customers.

1. Introduction

Web services provide a method for servers or processes to communicate in an automated fashion by sending XML messages over standard web protocols, such as HTTP, or more commonly, HTTPS. Ease-of-use and standardization caused web services to be used in ever-increasing numbers and critical functions than ever before. As web services became more pervasive in enterprise environments, the threat of attack also became a more serious issue. Many of these communication channels are trusted paths, and used for identification and authentication services. As the reliance of Department of Defense (DoD) agencies on web services is increasing, these services could greatly benefit from higher-levels of assurance and resilience.

2. Background

2.1 Web Services

Traditionally web protocols have been used primarily in human-to-computer applications such as web browsers. Over the past decade, these protocols have been refined and restricted in scope to offer web services, which enable computer-to-computer communication channels. These automated services also provide application programming interfaces (APIs) to help in the development of distributed applications. Currently, the web service landscape is divided predominantly

into SOAP (Simple Object Access Protocol) and REST (Representational State Transfer) architectures. While there are differences between these architectures from a web service developer and consumer perspective, our work applies equally to both as it analyzes data at the HTTP(S) level.

2.2 Autonomic Computing

Autonomic computing seeks to enable information systems to become more self-healing and resilient to abnormal operating conditions such as high utilization, software or hardware faults, and malicious attacks. This is analogous to the autonomic features of humans such as breathing, blood sugar regulation, wound healing, and innate reflexes. The four aspects of autonomic computer are Self-configuring, Self-healing, Self-optimizing and Self-protecting (Murch, 2004). Our research focus has been on building Self-protecting web services. We will use the framework in Figure 1 to implement our prototype.

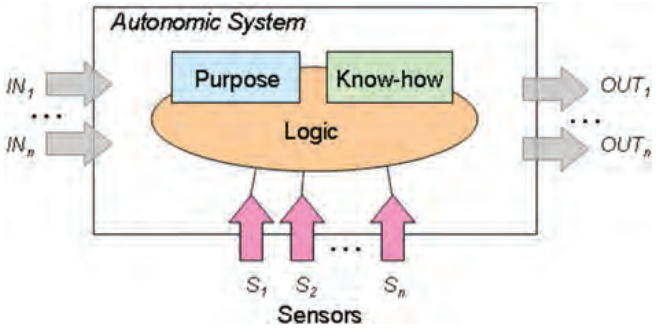


Figure 1. Autonomic computing (from Wikipedia contributors, 2011)

The architecture described in this paper is based, in part, upon the systematic distributed event-based (DEB) performance monitoring approach described in Dubey, et al. (2009) and Abdelwahed and Kandasamy (2006). This approach applies to distributed systems by measuring system variables (CPU and memory utilization), application variables (application queue size, queue waiting time, and service time), and performance variables (response time, throughput, and power consumption), and using these values in an autonomic fashion to maintain the multidimensional quality-of-service (QoS) requirements of the monitored system.

2.3 Anomaly Detection

Fault detection has been an area of research focus for quite some time. The specific application of fault detection to detecting intrusions in information system environments can be divided into two categories: pattern-based and anomaly-based. Traditional intrusion detection systems (IDS) utilize the pattern-based approach. This approach requires a priori knowledge of attack features in order to build a pattern recognition rule specific to each known attack. One of the obvious drawbacks is the tedious nature of examining each attack of interest and creating custom rules. The historical frequency of new attacks shows that a better means for recognizing new or unknown attacks is desperately needed. The anomaly-based approach uses statistical methods to better distinguish between normal and malicious activity. The sheer volume of data inspected makes intrusion detection a fundamentally-complex problem (Axelsson, 2000). The reduction of detection errors is the main focus in most enterprise computing environments.

2.4 Threat Response

Traditionally IDSs have been passive in nature and require human inspection and action to address ongoing threats. The delay between threat detection and threat response has introduced an unacceptable level of risk. The skill level and subsequent cost of staff needed to respond to incidences and threats have created a desire for accurate and automated tools.

3. Approach

3.1 Prototype Architecture

Our prototype consists of a self-contained Autonomic Protector. The Autonomic Protector, shown in Figure 2, acts as a front-end web proxy and IDS. It receives input from several data streams including intrusion detection alerts, application

logs, operating systems logs, and key system performance measurements. This data stream is analyzed to produce a threat condition ranking about the overall security posture or health of the protected web application platform. As shown in Figure 3, this ranking system from normal to highest includes the following states: Guarded, Elevated, High, and Severe.

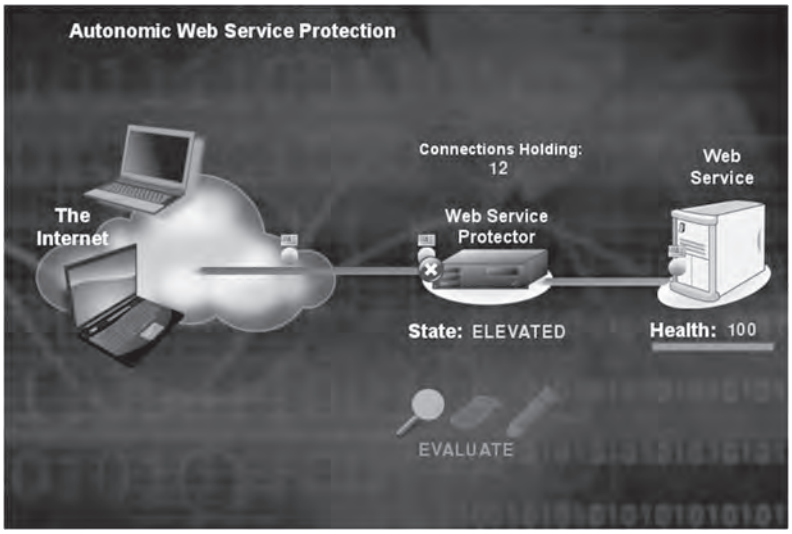


Figure 2. Autonomic web service protection

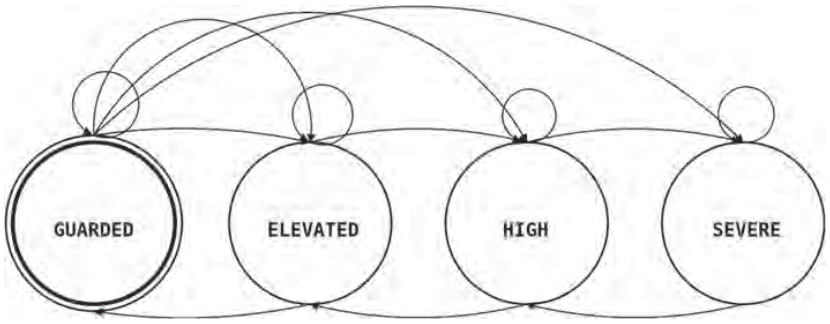


Figure 3. Threat condition state transition

The analysis of these data streams includes the use of classification algorithms to better predict malicious events. Once a prediction is made, effectors are then used to take an action that will improve the security posture of the web service in the face of ongoing threats and malicious activity. These actions include modifying the running state of the web application processes, updating existing application software, disabling needed but non-critical services, or rejecting traffic from known hostile sources.

3.2 Autonomic Protector

The Autonomic Protector prototype currently runs on the Ubuntu Linux operating system. It contains a valid instance of a Snort sensor and custom web proxy software. This allows all incoming web traffic to be inspected before being released to its final destination at the back-end web server. Encryption processing will be off-loaded to the Autonomic Protector as well. The Autonomic Protector is constantly attempting to return to a safe, reliable threat ranking. This constant effort is accomplished by continually cycling through evaluate, calculate, and update procedures. As new connections are received, classification algorithms are utilized to better evaluate possible threats and calculate the effectiveness of responses to those threats.

3.2.1 Sensors

Currently our prototype utilizes a local Snort sensor, web service application logs, web server operating system audit logs, and performance measurements of CPU utilization, file system usage, and network packet information such as hop

count, client/server response timing, and payload size. We will also inspect the total number of packets exchanged in client/server communications and the total amount of data exchanged. We fully anticipate adding sensors as more interesting data streams are enabled.

3.2.2 Effectors

The Autonomic Protector sends commands to Effectors to perform various configuration changes or changes in run-states (i.e., stop/start/restart). These effectors perform actions based on these commands to include denying/rejecting requests from known-hostile sources and allowing requests from known-trusted sources using white- and black-lists. Further research could include reconfiguring operating system variables related to network bandwidth utilization, disk space quotas, per process memory usage, and CPU prioritization. Experiments will certainly need to be conducted to verify the safety and effectiveness of using effectors.

3.2.3 Classification/Decision Algorithms

As the system makes use of existing pattern-based IDSs to highlight known attacks/problems, further classification of "normal" activity is needed to help minimize false-positive results. Therefore, one of the more interesting aspects of the development of this prototype is the classification algorithms used to classify events as normal or anomalous. Currently we are focusing on the following candidate algorithms:

a. Bayesian networks
b. Hidden-Markov models
c. Decision trees
d. Random Forests
e. Learning vector quantization
f. k-means clustering
g. Fuzzy c-means clustering
h. Quality Threshold (QT) clustering

Initially, the three clustering algorithms (k-means, fuzzy c-means, and quality threshold) will be used to classify incoming web service requests.

In statistics and machine learning, k-means clustering is a method of cluster analysis that aims to partition n observations into k clusters, in which each observation belongs to the cluster with the nearest mean. It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data as well as in the iterative refinement approach employed by both algorithms (Tan, 2005).

In fuzzy clustering, each point has a degree of belonging to clusters, as in fuzzy logic, rather than belonging completely to just one cluster. Thus, points on the edge of a cluster may be in the cluster to a lesser degree than points in the center of cluster (Tan, 2005).

QT (quality threshold) clustering (Heyer, et al., 1999) is an alternative method of partitioning data, invented for gene clustering. It requires more computing power than k-means, but does not require specifying the number of clusters *a priori*, and always returns the same result when run several times.

While all three clustering techniques could partition web service request data, the trade-offs in speed and accuracy will need to be tested to determine which is most appropriate.

4. Significance to DoD

The benefits of a successful research effort should be evident across public and private sectors. Department of Defense information systems will benefit greatly due to the heightened security risk and ever-present threat from a variety of sources. DoD networks and systems are constantly under attacks that range from outright attacks to advanced persistent threats.

5. Conclusion

The current research prototype will incorporate more sensor input. However, the major focus of future research activity will be choosing the most efficient and accurate classification algorithms. As the system is designed to analyze data and make decisions in real-time, the classification algorithms will need to be optimized to strike a balance between speed and accuracy. Further analysis will be done to determine the minimum and maximum numbers of variables required to be analyzed to produce acceptable performance and accuracy. Utilization of HPC resources will be pursued to facilitate the testing and performance analysis of these algorithms. Future research will also include networks of autonomic protectors

that share threat condition information to represent more accurately the real threat condition level needed to address ongoing or future risks properly.

Acknowledgements

This work was supported by the DoD High Performance Computing Modernization Program at the US Army Engineer Research and Development Center DoD Supercomputing Resource Center, Information Technology Laboratory, Vicksburg, MS.

References

Abdelwahed, S., and N. Kandasamy, "A control-based approach to autonomic performance management in computing systems", *Autonomic Computing: Concepts, Infrastructure, and Applications*, M. Parashar and S. Hariri (eds.), CRC Press, Boca Raton, FL, pp. 149–168, 2006.

Axelsson, S., "The base-rate fallacy and the difficulty of intrusion detection", *ACM Transactions on Information and System Security*, 3, 3, pp. 186–205, 2000.

Dubey, A., R. Mehrotra, S. Abdelwahed, and A.Tantawi, "Performance modeling of distributed multi-tier enterprise systems", *ACM SIGMETRICS Performance Evaluation Review*, 37, 2, pp. 9–11, 2009.

Heyer, L. J., S. Kruglyak, and S. Yooseph, "Exploring expression data: Identification and analysis of co-expressed genes", *Genome Research*, 9, 11. pp. 1106–1115, 1999.

Murch, R., *Autonomic Computing*, Prentice Hall, Upper Saddle River, New Jersey, pp. 10–15, 2004.

Tan, P., M. Steinbach, and V. Kumar, *Introduction to Data Mining*, (First Edition), Addison Wesley Longman, Boston, MA, pp. 490–493, 2005.

Wikipedia contributors, "Autonomic Computing", Wikipedia, The Free Encyclopedia, last modified April 27, 2011, http://en.wikipedia.org/w/index.php?title=Autonomic_Computing&oldid=426164376 (accessed May 19, 2011).

Large-Scale High-Fidelity Mobile Ad-hoc Network Emulation

David A. Richie
Brown Deer Technology, Forest Hill, MD
drichie@browndeertechnology.com

Brian J. Henz and Dale R. Shires
US Army Research Laboratory (ARL), Aberdeen Proving Ground, MD
{brian.j.henz, dale.shires}@us.army.mil

James A. Ross
High Performance Technologies, Inc., Aberdeen, MD
jross@hpti.com

Abstract

The HPCMP Mobile Network Modeling Institute (MNMI) is consolidating pre- and post-processing tools for mobile ad-hoc network simulation-emulation-experimentation cycles. This effort has resulted in the Network interdisciplinary Computational Environment (NiCE). We have leveraged the NiCE framework and developed a real-time terrain-dependent RF propagation path-loss model to provide data analysis tools, improve model fidelity, and increase visualization options for the Extendable Mobile Ad-hoc Network Emulator (EMANE) software used by the MNMI for network emulation. Accurate large-scale mobile ad-hoc network emulation requires high-fidelity RF propagation path-loss estimation in real-time, based on the dynamic position of each radio. Traditionally, this has been achieved through pre-processing of known network configurations. The nominal target emulation scenario of 5,000 radios imposes a significant high performance computing requirement. We have developed an implementation of the Longley-Rice Irregular Terrain Model (ITM), including dynamic path extraction, targeting hybrid multi-core (CPU)/many-core (GPU) compute nodes, capable of providing the necessary computational performance. The method has been extended to utilize multiple GPUs per compute node, and integrated with EMANE, to provide dynamic RF path-loss estimation in real-time. Initial results have been used to architect a recently selected High Performance Computing Modernization Program (HPCMP) DHPI (Dedicated HPC Project Investment) platform that is expected to utilize over 5,000 CPU-cores and 500 GPU co-processors.

1. Introduction

In mobile networks, access points can move and coverage may vary widely in a region. At times, the access points will cluster together, leaving parts of the map with sparse coverage and parts with compromised service, due to competition for available bandwidth as channel subscriptions become saturated. On the battlefield, relocating receivers to areas with better coverage may not be an option, and calls have been made for tools to optimize and plan prior to a mission as much as possible.

The Mobile Network Modeling Institute (MNMI) was established in FY2007 to exploit high performance computing (HPC) resources through the development of computational software; thus enabling the Department of Defense (DoD) to design, test, and optimize networks at sufficient levels of fidelity and with sufficient speed to understand the behaviors of network-centric warfare technologies in the full-range of conditions in which they will be deployed. Operational goals include the development of scalable computational modeling tools for simulations and emulations, the ability to understand a *priori* the performance of proposed radio waveforms in the field, and to optimize the network for US Army Warfighters. The focus here will be on the HPC developments for MANET emulation that use general-purpose graphics processing units (GPGPUs) for real-time RF path-loss predictions.

The scale and complexity of mobile ad hoc networks to the DoD is unique, and to the Army in particular as a mobile fighting force. The military is rapidly becoming a network-centric force, with substantial access to sensor-derived surveillance information, as well as increasingly complicated application layers running over many different devices. This introduces significant advantages to the Warfighter, but also brings in new dependencies and risks from a rapid change in configuration of the MANETs that provide network access across the battlefield.

Mobile networks must be understood at a series of levels, from radios to packet network to communication infrastructure and its resulting impact on the Warfighter. To make this difficult problem tractable, the MNMI has developed a four-pronged approach. The first is Mobile ad hoc Network (MANET) simulation, where large-scale HPC assets can be used to test and optimize large radio deployments. Second is MANET emulation, where researchers can investigate the performance of proposed radios high in the network stack (application layer) all the way to the lower physical layers. Third is MANET experimentation, where live and constructive exercises can capture real radio performance and test interoperability with real and virtual assets. This also results in data sets that later can be mined to fill data-gaps and verify models. Fourth, a system that ties together all of these aspects and brings in support for visualization and data analysis. The MNMI is addressing all of these topics, and a Dedicated HPC Project Investment (DHPI) for the MNMI will greatly enhance all four of these endeavors with a special ability to greatly augment efforts in MANET emulation. Throughout this MANET emulation effort the EMANE (Extendable Mobile Ad-hoc Network Emulator) software from DRS (formerly Cengen Labs) is used. This software is open-source and sponsored by NRL (US Naval Research Laboratory) and ARL (US Army Research Laboratory). EMANE can be obtained from DRS.[1]

2. Methodology

2.1 Large-Scale MANET Emulation

One aspect of the MNMI is focused on developing a framework for large-scale MANET emulations with a target of up to 5,000 emulated devices. This will allow for the research, development, and evaluation of network algorithms, applications and devices in a controlled environment. With this environment, it will be possible to use unmodified software applications and integrate virtual devices into live-experiments in order to augment the testing parameters and the perceived traffic by physical network devices in the field. This augmentation of live-experiments enhances the experience of testers, and increases the degrees-of-freedom that can be evaluated in an experiment. The achievement of these goals for large-scale MANET emulations and live-experiment integration requires the bridging of a number of technical gaps. One of the bridging technologies presented here is the development of real-time RF propagation computation.

2.2 Real-Time RF Path-Loss Computations

RF wave propagation models play an essential role in the planning, analysis and optimization of radio networks[2]. For instance, coverage and interference estimates of network configurations are based on field-strength predictions. Approaches for field-strength prediction can be divided into (semi-) empirical and ray-optical models. For example, the semi-empirical COST-Walfisch-Ikegami model[3] estimates the received power predominantly on the basis of frequency and distance to the transmitter. Ray-optical[4] approaches identify ray paths through the scene, based on wave-guiding effects like reflection and diffraction. Semi-empirical algorithms usually offer fast computation times, but suffer from inherently-low prediction quality. Ray-optical algorithms feature a higher prediction quality at the cost of higher computation-times. For MANET emulation integration, these algorithms must be computed in real-time for each of the propagation paths possible, with an $O(n^2)$ complexity for a flat network hierarchy.

GPUs have been identified as a solution to provide the raw floating-point performance required to compute the RF propagation path-loss in real-time. The GPU architecture is also ideally-suited for accelerating ray-tracing algorithms that are found in the ray-optical approaches for RF wave propagation modeling. These RF propagation computations must be tightly-coupled with the MANET emulation environment in order to provide real-time response to computation requests, where real-time here requires a 0.5 second response to maintain the integrity of the emulation for realistic mobilities. In addition to enabling real-time propagation modeling, a GPU-accelerated algorithm is the basis for adding higher-fidelity modeling capabilities such as foliage[5] effects.

2.3 Real-Time Propagation Path-Loss Estimation Using GPGPU Co-Processors

In order to meet the real-time requirements for RF propagation path-loss estimation suitable for network emulation, an implementation of the Longley-Rice[6] Irregular Terrain Model (ITM) was developed for GPU co-processors. The implementation is based on the open-source C implementation available from the US Department of Commerce[7]. The code development required significant re-factoring to employ algorithms suitable for the fine-grained parallelism of modern GPU architectures. A digital terrain extraction algorithm was also ported to GPUs to allow the entire propagation path-loss calculation to be performed on a GPU, with the GPS coordinates of each radio as an input, and the path-loss matrix as the output.

The overall algorithm is designed to calculate the point-to-point path-loss between transmitter/receiver pairs in parallel on one or more GPUs. The first step is to load the digital terrain data onto each GPU co-processor, which incurs a one-time cost prior to the network emulation. At intervals of roughly 1-second, the emulation software publishes the GPS coordinates of all radios to the path-loss server. These coordinates are used to generate a matrix of transmitter/receiver pairs that define the work to be performed in the path-loss calculation. Each point-to-point calculation is independent, allowing for a simple distribution of work across the computational resources. Each computational node can process a subset of the pairs independently, and publish partial path-loss matrices. For a given node, which may contain multiple GPUs, the transmitter/receiver pairs are loaded into stripes of size 1,024 and submitted to one of the GPUs to perform the path-loss computation. Multiple GPUs are utilized in a simple sequential order, and all data-transfer and execution is non-blocking on the host-side so that good parallel efficiency is achieved across multiple GPUs on a given compute node.

The ITM algorithm is a complex heuristic model for RF propagation path-loss estimation. The details of the path-loss computation are shown in Figure 1. The first step (READ_PATH kernel) is to extract the path data from the elevation map for each transmitter/receiver pair using the digital terrain data that was pre-loaded onto the GPU co-processor. The result is a discretised function reflecting the height profile along the path connecting the transmitter and receiver, corrected for the curvature of the Earth. The path data for each transmitter/receiver pair is stored in an array and passed to the next stage of the ITM algorithm. The array of height profiles is pre-processed with three kernels (PRO_QLRPS, QLRPF, QLRPF_BOTTOM) re-factored from the original ITM C code implementation. The results are multiple arrays of per-path quantities needed in the next stages of the computation.

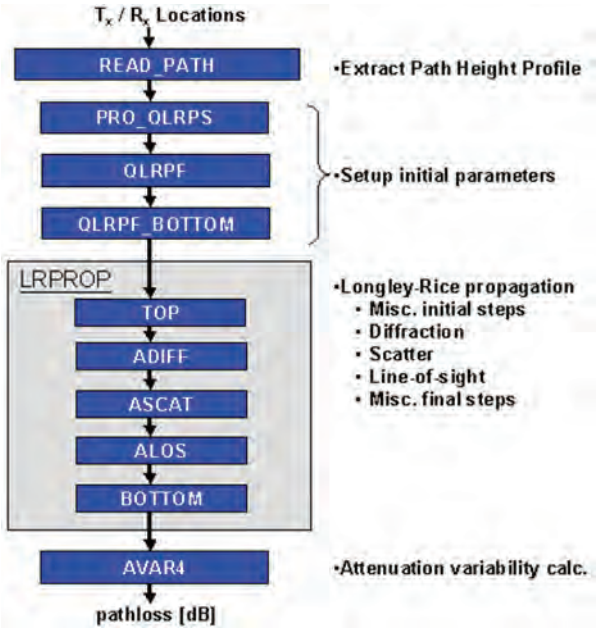


Figure 1. Data flow for the path extraction and ITM path-loss calculation algorithms. The algorithms are implemented using 10 OpenCL kernels for GPU-acceleration.

The core of the computation is the Longley-Rice propagation model (LRPROP kernels) that is broken down into sub-calculations for different physical effects impacting the path-loss. The kernels ADIFF, ASCAT, and ALOS implement models for diffraction, scatter, and line-of-sight attenuation, respectively. A final kernel implements the attenuation and variability computation that results in the final path-loss estimate for each transmitter/receiver pair. The result of the computation is a matrix of path-loss values generated in stripes across multiple GPUs per node, with each node responsible for a subset of the complete matrix of transmitter/receiver pairs. The path-loss server publishes these path-loss matrices back to the emulation software, allowing the latter to account for physical effects in determining link-strength between radios. The model assumes omnidirectional antennae with identical polarization.

Several challenges were encountered in the development of the GPU implementation. First, the C code, upon which the implementation is based, closely reflected the original FORTRAN implementation and contained many basic constructs in its design that were ill-suited to modern processors, requiring substantial reformulation. One of the challenges encountered in obtaining an efficient GPU implementation was the replacement of nested, conditional control-flow with predicated

execution inside inner-loops. Another issue encountered during development of the GPU implementation was the use of single-precision that required the reformulation of several algorithms to address the numerical stability of transcendental functions at small-angles. The port to GPU co-processors was enabled by the use of OpenCL™, which is an industry-standard programming applications programming interface (API) for parallel programming of heterogeneous computing platforms.[8] The use of OpenCL ensures portability across modern multi-core and many-core processors and specifically supports the use of GPUs from AMD and Nvidia. The ITM implementation makes use of the STDCL[9] interface, which greatly simplifies the use of OpenCL for complex HPC applications.

3. Results and Discussion

After several iterations of development, the current ITM implementation supports multiple GPUs, has been tested using AMD and Nvidia GPUs, and is capable of providing path-loss estimation results for 256 radios within the 1-second limit. This 1-second limit includes the 0.5-second GPU compute-time along with the overhead of the computation on the host, including calculation set-up time and the time needed to republish the results for network emulation. For 256 radios, the calculation can be performed in 0.44 sec using an Nvidia Tesla C2070[10] as compared with 0.82-sec using an AMD Radeon HD 6970.[11] More detailed performance results are shown in Figure 2 with benchmarks performed using a single graphics card in a simple Linux desktop platform. For each GPU, the results show the expected quadratic dependence on the number of radios.

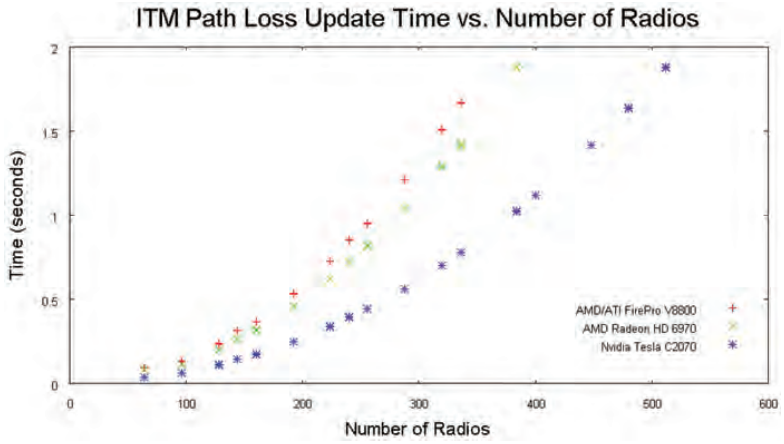


Figure 2. Benchmarks using a single GPU for the ITM path-loss update for various numbers of radios. A time of under 0.5 seconds is required for reliable real-time operation.

3.2 Dedicated HPC Project Investment

Through the High Performance Computing Modernization Program (HPCMP), a dedicated HPC project investment was awarded to the MNMI in order to reach this and other goals set forward by the MNMI. One of these goals is to emulate 5,000 radios in real-time. This emulation goal requires not only the computing power to host the virtual machines, but the capability to compute the RF path-loss in real-time so that these emulated radios can be interfaced with live-experiments or force-modeling simulations. The results from the optimized ITM code were used to project the resource requirements for performing the real-time RF propagation path-loss estimation. The DHPI system requirements were designed to include over 5,000 physical CPU cores and approximately 500 GPUs. This configuration gives the DHPI system a peak theoretical throughput approaching 1 PFLOPs. This would place it near the top 10 supercomputers in the world based on the November 2010 rankings.[12]

4. Conclusion and Future Work

Through this effort we have developed a framework that leverages many technologies allowing for the emulation of large MANETs, and which includes the use of GPUs for real-time RF propagation path-loss. The use of a common data file format such as NetDMF allows for the mining and visualization of data in a consistent manner with MANET experiments and simulations. At runtime, the use of HPC tools such as perceus and slurm for cluster management and job control, respectively, provides efficient use of our hardware resources and scalability to 1,000's of virtual machines and hosts. The

use of GPUs provides the computational capability required to compute the RF propagation path-loss ITM algorithm in real-time. With the installation of the DHPI hardware and this framework, we will be able to emulate 5,000 MANET nodes in real-time and provide critical data for application analysis and scenario development.

This framework allows the exploration of VoIP over MANET performance by varying numerous design parameters including physical layer radio models, network configuration (number of nodes and placement), topology, node movement, vocoder selection, multicast (conference calls), unicast (point-to-point calls), etc. We anticipate that this exploration will help lead to the design of an additional voice communication capability for the soldier on the battlefield.

5. Significance to DoD

The scale and complexity of mobile ad hoc networks to the DoD is unique, and to the Army in particular, as a mobile fighting force. The military is rapidly becoming a network-centric force, with substantial access to sensor-derived surveillance information, as well as an increasingly complicated application layer running over many different devices. This introduces significant advantages to the Warfighter, but also brings in new dependencies and new risks from the rapid change in configurations of the MANETs that provide network access across the battlefield.

Unfortunately, it is difficult to design and evaluate mobile ad hoc networks with sufficient fidelity and scale. The research plans and focus of the MNMI is addressing the primary issues associated with optimized MANET planning and execution. Without the MNMI, there would be inadequate research into characterizing MANETs for military use. The DoD's network-centric warfare transformation would be forced to rely on expensive testing and low-fidelity or small-scale simulations.

The MNMI encompassed two research areas that will benefit or potentially benefit from the development of a real-time RF path-loss calculation. The primary beneficiary is in the area of MANET emulation where the physical layer must be computed in real-time. MANET emulations incorporate virtual and real devices in a partially-virtual environment. The RF path-loss calculation creates the realism of the virtual environment by calculating path-loss due to terrain profiles, weather effects, etc. MANET simulation also requires quick turnaround in RF path-loss calculation, but with a much smaller number of point-to-point calculations rather than a single NxN matrix of calculations. MANET simulation will more likely benefit from high-fidelity models such as TLM and ray-tracing, where a single point-to-point calculation can take a measurable amount of time to complete. The final area of expected benefit is in radio testing using attenuators to model propagation in the physical layer. Real-time RF path-loss data is computed and used to program attenuators connected to the antennae port on a physical device. The attenuator then modifies the signal, making it weaker by the amount predicted and passing through the resulting EM wave to the receiving devices.

Acknowledgements

The authors would like to acknowledge the support received from the High Performance Computing Modernization Program Office (HPCMPO) under the Mobile Network Modeling Institute (MNMI) and the User Productivity Enhancement, Technology Transfer and Training (PETTT) program. The authors would also like to thank the DoD shared resource center (DSRC) at the Aberdeen Proving Ground, MD for support in the form of computer time.

References

1. DRS-Cengen, EMANE Website, http://labs.cengen.com/emane, accessed online 28-April-2011.

2. Rick, T. and R. Mathar, "Fast-edge diffraction-based radio wave propagation model for graphics hardware", *Antennas*, INICA '07, 2nd International ITG Conference on, pp. 15–19, 2007.

3. Damasso, E. (editor), *Digital Mobile Radio Towards Future Generation Systems*, Office for Official Publications of the European Communities, Luxembourg, 1999.

4. Bertoni, Henry L., *Radio Propagation for Modern Wireless Systems*, Prentice Hall Professional Technical Reference, 1999.

5. Wang, Feinian and K. Sarabandi, "A physics-based statistical model for wave propagation through foliage", *IEEE Transactions on Antennas and Propagation*, 55(3), pp. 958 –968, March 2007.

6. Longley, A.G. and P.L. Rice, "Prediction of Tropospheric Radio Transmission Loss over Irregular Terrain, A Computer Method-1968", Technical Report, Environmental Sciences Services Administration, Boulder, CO, 1998.

7. U.S. Department of Commerce, *Irregular Terrain Model (ITM) (Longley-Rice)*, http://flattop.its.bldrdoc.gov/itm.html, accessed online 26-April-2011.

8. OpenCL Overview, http://www.khronos.org/opencl, accessed online 28-April-2011.

9. STDCL is a simplified interface to OpenCL developed by Brown Deer Technology and available as part of the COPRTHR SDK under an open-source LGPLv3 license, http://www.browndeertechnology.com/coprthr stdcl.htm, accessed online 28-April-2011.

10. Benchmarks used the OpenCL implementation provided by the NVIDIA CUDA Toolkit v3.2.

11. Benchmarks used the OpenCL implementation provided by the AMD ATI Stream SDK v2.3.

12. TOP500 Org., Top 500 Supercomputer Sites, http://top500.org/, accessed online, 26-April-2011.

Persistent Surveillance Supercomputing using the LLGrid Filesystem

Jeremy Kepner, Chansup Byun, William Arcand, William Bergeron, Matthew Hubbell, Andrew McCabe, Peter Michaleas, and Albert Reuther
MIT Lincoln Laboratory, Lexington, MA
{kepner, warcand, bbergeron, cbyun, mhubbell, amccabe, reuther}@ll.mit.edu

Abstract

Lincoln Laboratory's mission is to prototype advanced sensor systems. Developing the algorithms and software for these systems requires large parallel computing and data storage capabilities. The LLGrid supercomputer allows hundreds of researchers' simultaneous interactive access to thousands of processors and disks for development and testing of their sensor processing algorithms. The storage requirements of the LLGrid user-base are as diverse as the sensors they are developing: sonar, radar, infrared, optical, hyper-spectral, video and cyber. However, there are three common elements: delivering large amounts of data interactively to hundreds of processors, high-level user interfaces that require minimal user training, and auditable security. The key innovation that allows LLGrid to achieve these goals is a ubiquitous use of the file system abstraction. All components of the LLGrid supercomputer are presented to the user as file systems. In addition, all critical parallel functions (parallel launch, parallel messaging, system monitoring) are done via file systems. The LLGrid storage system has developed high-level interfaces for a wide-range of storage technologies: parallel file systems, file-based distributed file systems, block-based distributed file systems, peer-to-peer file systems, parallel virtual memory, SQL and NoSQL databases, file-based launching, file-based messaging passing, and file-based secure portals. This paper describes the LLGrid data processing requirements, architecture, technologies, interfaces, usability and performance. As a case study, LLGrid shows that storage technologies can perform all the critical functions in a supercomputer; these functions can be incorporated into high-level array-based programming environments; and keeping user effort to a few hours is the most important predictor of success with this user-base.

1. Introduction

Sensor prototyping at Lincoln spans a wide-range of sensor modalities: sonar, radar, infrared, optical, hyper-spectral, video and cyber. The goal is to demonstrate sensor systems that push the limits of size, resolution and bandwidth. Development of these sensor systems requires access to parallel computing and parallel data storage by hundreds of users. In addition, the timeframe for sensor algorithm development is often very short (weeks to months). Thus, the algorithm developers need to work interactively in high-level languages (e.g., MATLAB, Java, and Python). These requirements have led to the LLGrid supercomputer. The LLGrid parallel high-level language work and interactive scheduler have been described extensively in the literature (see References 1, 3, 2, and references therein). This paper is the first comprehensive presentation of the LLGrid interactive data storage system.

The key innovation that allows LLGrid to achieve its interactive performance goals is the ubiquitous use of file system abstractions. All components of the LLGrid system are presented to the user as file systems (to the point that most users never actually login into the supercomputer). In addition, all critical parallel functions (parallel launch, parallel messaging, parallel system monitoring) are done via file systems (see Figure 1). Briefly, a user connects to LLGrid by mounting the LLGrid file system on their desktop. The software tools for writing parallel programs are found in their LLGrid home directory. The user copies their data and programs to their LLGrid home directory. The user launches a parallel program on LLGrid by running commands (found in the LLGrid file system) that write launch scripts to the LLGrid file system. The LLGrid scheduler detects these scripts (in the LLGrid file system), finds the users programs (in the LLGrid file system) and launches the program on the LLGrid compute nodes. The compute nodes exchange messages with each other and the user desktop via the LLGrid file system. The program's output is written to the LLGrid file system, which is visible to the user in their home directory. All of these steps occur interactively. It is not uncommon for a user to go through this process many times in an hour while they are developing their algorithms. As a case study, LLGrid shows that file system abstractions can perform all the critical functions in a supercomputing system, and are a powerful tool for minimizing user effort.

Figure 1. LLGrid file-based parallel computing system. All components of LLGrid are presented to the user via file system abstractions.

The LLGrid file system abstraction places enormous pressure on the performance of the LLGrid data storage system. LLGrid storage is a diverse mix of industry-standard storage technologies (Lustre[4], BitTorrent[5], PostgreSQL[6], MySQL[7], and GRSecurity[8]) and emerging technologies (HadoopDFS[9,21], Tokutera[10], Sector/Sphere[11], HBase[12], and other Big-Table-like databases[13]). Likewise, the LLGrid storage interface uses industry standard interfaces (NFS, Samba, FUSE, and WebDAV) and high-level interfaces that we have developed (MatlabMPI[15], pMatlab[3], pMatabXVM[14], gridMatlab[2], D4M, LLGridZF[16]). 85% of the LLGrid user-base consists of Matlab users; thus the majority of these interfaces are written for MATLAB. However, these interfaces could be implemented in any high-level language such as Java, Python, or Perl.

LLGrid storage technologies and interfaces address a range of data request patterns (see Section 3). These technologies are by no means unique (or static), and there are a number of industry standards and emerging technologies that have been described in the literature that could be integrated into LLGrid in the future.

Fortunately, there is no shortage of groundbreaking work in the data storage area. For LLGrid, the limiting factor is the ability to integrate and put high-level interfaces on these technologies that allow users to take advantage of them with minimal training.

Lincoln fielded-sensor-system requirements are very similar to to Mahanaxar[17] in that both have time-indexed data and non-time-indexed data. However, the LLGrid sensor development work is different during development, and can't use a ring buffer to overwrite old data that will be needed for validating algorithmic changes. Nevertheless, it might be helpful to integrate Mahanaxar to allow our users to develop against a storage system that could be run on fielded sensors. Other storage technologies of interest include active storage[22], linear tape[19], solid state[20], fast RAID recovery schemes[23], and security aware partitioning[18].

In the database arena, there are a number technologies of interest that range from commercial databases (e.g., Oracle, Ingres, Vertica, and VoltDB) to open-source databases (e.g., SciDB).

2. Requirements

The LLGrid storage architecture is driven by usability, performance, and security requirements. Usability is measured by training time and the extent a user has to modify their code to work well on the LLGrid supercomputer. Performance is measured with capacity, bandwidth, latency and operations/sec. Security is measured by the adversary effort required to mount a successful attack, and the ability to detect intrusion. Technologies are assessed by implementing standard benchmarks and measuring the usability, performance and security achieved on those benchmarks. For example, typical usability targets take less than three hours to learn and are less than a 5% increase in the size of the user code. Typical performance goals are usually to double one element of performance, while not halving any of the other elements. Typical security goals are to increase the adversary effort while increasing intrusion detection.

A. Usability

A typical LLGrid user begins with a data-set (simulated or real) that represents the current output from a prototype sensor. This data-set can range from of a few large files to many small files or a database. The user's goal is to discover an algorithm that will detect items of interest in the data. This discovery process is manifest by writing a program (typically in a high-level language such as MATLAB, Java, or Python). The majority of the code development occurs on the user's

desktop system. The size of the data typically drives the need to write a parallel program so that the algorithm can be executed in reasonable time. The overall time-scale of this process is very short. A user will typically need to have a reasonable first algorithm in a matter of weeks. During this period the user will run their program hundreds of times. Most runs are very short and are for de-bugging purposes. Once the program is running reasonably, the program will often run for hours on hundreds of processors to build up statistics on the quality of the algorithm.

The users' primary concern is time. Using a parallel computer is only effective to them if it reduces their overall time-to-solution relative to what they could have done on their desktop. It is imperative that the time to learn how to use the parallel computer be short (typically a few hours). During the initial design of LLGrid it became apparent that GUI file system interfaces were intuitive to the users, and presenting the parallel computing system as a file system would minimize training time. Likewise, the interfaces to all performance-enhancing technologies must be made available to users via high-level programming environments that can be taught quickly. This is has typically meant creating array-based parallel MATLAB interfaces to file systems and databases.

B. Performance

The performance requirements of each LLGrid user are unique to the project they are working. To capture these requirements, the LLGrid team has written software for a number of open benchmark activities (HPC Challenge[24], HPEC Challenge[25], and Graph500[26]). These are computation benchmarks. However, by simply scaling the problem sizes to the point where they don't fit in RAM, which is reflective of the real-world from which they are drawn, these computation benchmarks become input/output (IO) benchmarks.

HPC Challenge consists of four primary benchmarks (Top500, Stream, FFT, and RandomAccess) that are designed to stress different parts of the memory hierarchy of a parallel computing system. Increasing the size of the problems naturally turns them into IO benchmarks that stress different data access patterns. For large problems, Stream becomes a local IO bandwidth test from a single compute node. Stream is representative of an LLGrid user reading and writing large intermediate files to local node storage during the course of their computation. For large problems, FFT becomes a global IO bandwidth test from all compute nodes. FFT is representative of an LLGrid user broadcasting and collecting large files to all nodes during the course of their computation. For large problems RandomAccess measures IO latency from all compute nodes. RandomAccess is representative of an LLGrid user broadcasting and collecting many small files to all nodes during the course of their computation. In addition, to the four primary benchmarks HPC Challenge has numerous secondary benchmarks. One particularly useful one is the ping-pong message passing benchmark that is useful for characterizing device latency and bandwidth. Ping-pong sets up a "ring" of processors, where each processor sends messages of difference sizes to its neighbors.

HPEC Challenge consists of several benchmarks. The synthetic aperture radar (SAR) benchmark has an explicit IO component to it. The SAR benchmark input consists of a number of large-input data files. Typically, each file is read by a separate processor from a central storage system. Reading the raw data input taxes the ability of a file system to support many simultaneous large-reads, and is essentially a read bandwidth test. The SAR benchmark turns the raw data into coherent images, searches the images for items of interest, and then writes out a small image containing each item of interest. Writing these many small output files tests the ability of a file system to support many file open/close requests.

Graph500 is derived from the Graph Analysis benchmark[26] and is reflective of a new class of graph-based computations. For large problems, Graph500 becomes a database benchmark. Specifically, the initialization phase measures the insert rate into a database. Likewise, the edge traversal phase of the benchmark measures the latency of queries. Both of these operations are reflective of LLGrid user applications.

Finally, for more a fundamental characterization of storage the IOzone (www.iozone.org) and the Sort benchmarks (Sort-Benchmark.org) are also used. IOzone measures the read and write performance to a storage device over different data sizes. The Sort benchmarks measure the largest file that can be sorted under a variety of constraints (e.g., time, size , and cost).

3. Architecture

There are a large number of possible dimensions to consider when evaluating storage technologies. For LLGrid applications, three dimensions appear to be the most important in characterizing the patterns of storage usage: data-request size, data-request offset, and ACID (atomicity, consistency, isolation, durability). Data-request size is the typical amount of data in a single request from a user application. Data-request size can be measured in bytes. However, given the rapid rate at which storage technologies advance, it is often more useful to think about request size in terms of the largest technology

that can accommodate the request (e.g., cache, single-node memory, multi-node memory, a single storage device or multiple storage devices). For example, a typical LLGrid application (as represented by the SAR benchmark) requests a small number of very large files that can fit on a single storage advice, and a large number of small requests where each request can easily fit in main memory.

Data-request offset is the typical byte "distance" between data-request from a user application. Data request offset can be thought of as either streaming (i.e., highly-cacheable) or random (i.e., cannot be cached). For example, a typical LLGrid application (as represented by the SAR benchmark) requests a small number of very large files that can be read as sequential blocks, and large number of small requests where each request may be in very different parts of the storage system.

ACID refers to the level of consistency required by the application. Historically, SQL databases have served read/modify/write applications that have required a high-level of ACID. However, many data mining applications are statistical in nature, and a new class of relaxed-consistency "NoSQL" databases has emerged to address these needs. Note: SQL is simply an interface and could be placed on any NoSQL database. Likewise, a NoSQL interface could be used with an SQL database. In either case, the ACID-level of the database is independent of the interface used to access it. The LLGrid database interface (D4M) exploits this fact in developing our own high-level interface that works with any database that stores data as tables.

Figure 2 shows the above dimensions with different storage technologies placed approximately in their "sweet" spots. The corners of Figure 2 are labeled with some of the standard approaches for achieving high-performance in that corner. Small data-requests with small offsets readily lend themselves to caching. Small data-requests with large offsets are often addressed with hashes that allow specific data requests to be found quickly. Large data-requests with small offsets are well served by using large datablocks that can be accessed sequentially. Large data-requests with large offsets are often addressed with files that allow the data to be large, while preserving an ability to randomly select different files.

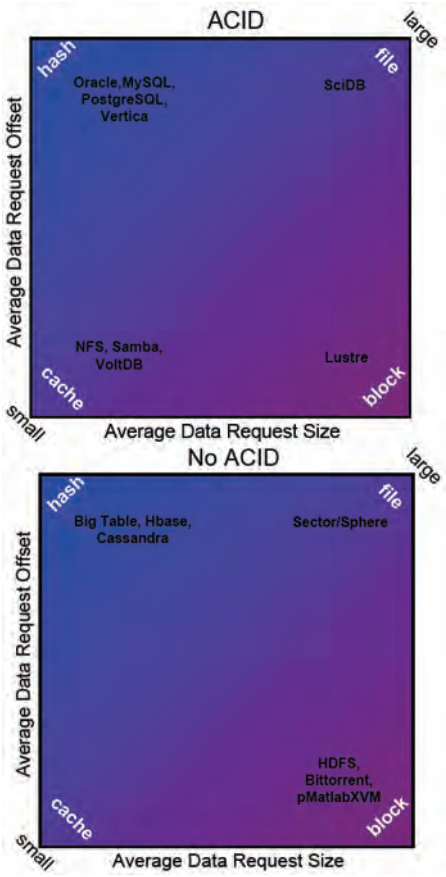


Figure 2. Storage evaluation dimensions

The benchmarks, likewise, can also be mapped onto this framework (see Figure 3). This mapping between technologies and benchmark-driven requirements provides a framework for technology evaluation and selection. The LLGrid data storage technologies have been selected to span this space of requirements.

It is worth noting that the location of most common IO interfaces (lower-left) is outside of the benchmarks.

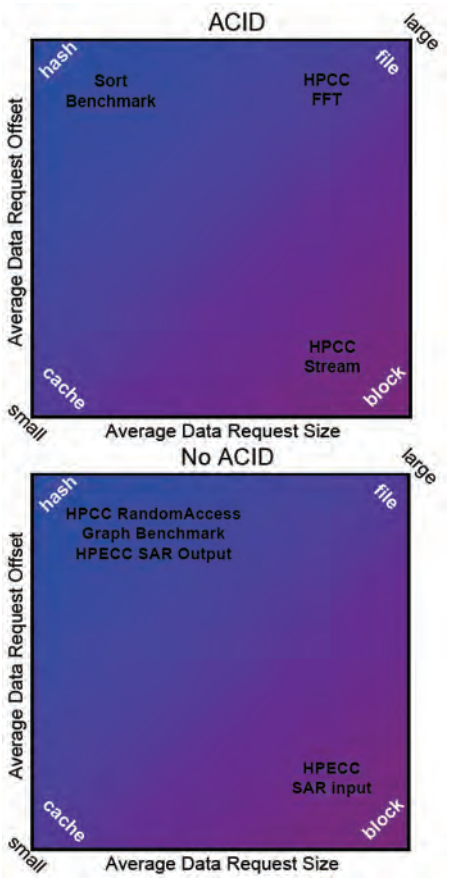


Figure 3. Benchmark dimensions

Thus, it is a significant challenge to use these interfaces to the technologies that meet the requirements.

4. Storage Technologies

The LLGrid file system abstraction places enormous pressure on the performance of the LLGrid data storage system. LLGrid storage is a diverse mix of commodity hardware, industry-standard storage technologies and emerging technologies.

A. Hardware

The LLGrid storage hardware is made up of commodity components with two primary types of storage: global and local. The global storage consists of high-device-count RAIDs (>100) that have their own data servers (~10) and metadata servers (~2). The global storage is visible to all compute nodes and to the user's desktop via industry-standard interfaces. The local storage consists of small-device-count RAIDs (<10) on the LLGrid compute nodes. The local storage is visible to programs running on the compute node, and can be made to appear as a single distributed storage system via a wide-range of interfaces.

The storage on an individual compute node has less capacity and performance that the global storage. However, the aggregate capacity and performance of the local storage on hundreds of compute nodes is larger than the global storage. A key challenge for the LLGrid storage system is finding combinations of storage technologies and interfaces that make this potential available to LLGrid users.

LLGrid is not limited to commodity clusters. Components of the LLGrid storage system have been successfully demonstrated on the IBM Blue Gene series[27] and the Cray XT series. This will allow LLGrid users to transparently take advantage of the benefits of these computing systems when they become available.

The physical specifications of the LLGrid storage system technology used in the tests is given in the Appendix.

B. Industry Standards

The most important element of LLgrid is the global storage that hosts user home directories, the scheduling system, the parallel messaging system, and the monitoring system. LLGrid uses the industry-standard Lustre open-source parallel file system to present the global storage array to the compute nodes and the user desktops. Lustre uses data and metadata servers to present many storage devices as single storage system.

Local node storage is used to supplement the global storage, and is most commonly used to store temporary intermediate results from a computation running on the node. That said, all local node storage is visible to all other nodes via NFS cross-mounts. Every user has private directories on every node and can access the local node storage from any other node at anytime. The mounts are maintained by the NFS auto-mounter. One advantage of these cross-mounts is their potential use an inter-node message passing (see Figure 4).

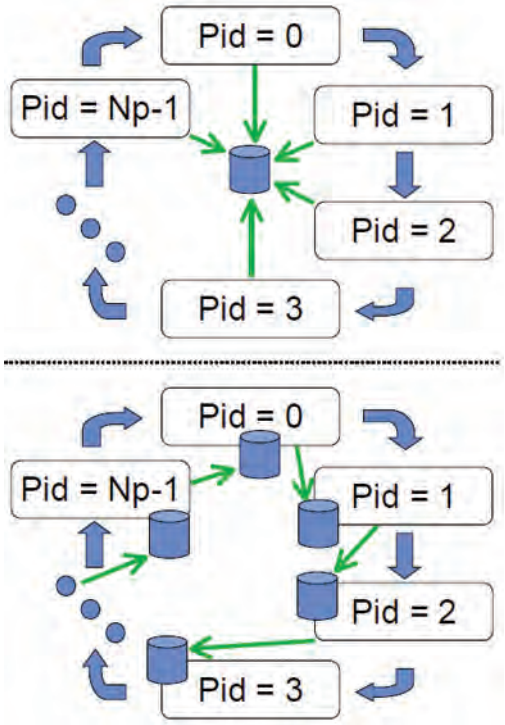


Figure 4. Execution of the ping-pong benchmark using a central global file system (top) versus cross-mounts of local node storage (bottom). Pid is the processor id and Np is the total number processors participating in the computation.

Most LLGrid user applications are written with high-level programming environments that are large software packages. Launching hundreds of copies of these programs can put a lot of pressure on the central file system. It is more efficient if these packages can be stored and executed from the local node storage. In addition, LLGrid users run their parallel programs from these high-level environments running on their desktops. LLGrid must match the user's desktop version of these environments, which entails storing several versions of the packages on each node. For all of these reasons, the LLGrid compute node system image is large, and needs to be updated frequently with large software updates. The BitTorrent[5] peer-to-peer file system is used to improve the performance of distributing the OS image to each of the compute nodes. BitTorrent allows every node to serve-up files to other nodes. Typically, a small number of LLGrid nodes are "seeded" with the new LLGrid image from the Lustre file system. These seed nodes then propagate the new OS image update to all other nodes. Using this approach, the LLGrid system can be completely reimaged in a few minutes.

To meet the security requirements the LLGrid uses the GRSecurity[8] OS enhancements that use a "white-list" model for accessing all files and functions in the system. Changing the white-list requires rebooting the system.

Databases play an important role for LLGrid users. For most database needs the PostgreSQL[6] and MySQL[7] databases provide the required functionality. The database servers can run using either global or local storage.

C. Emerging Capabilities

LLGrid has tested a number of emerging storage capabilities to supplement the above industry-standards. The goal is to make the capacity and performance potential of the local node storage available to users whose requirements exceed what can be provided by the Lustre global storage or the NFS cross-mounts of the local nodes. The majority of these technologies exploit some variation of the concept of sending the computation to the data (see Figure 5). This is in contrast to the traditional "send the data to the compute" approach.

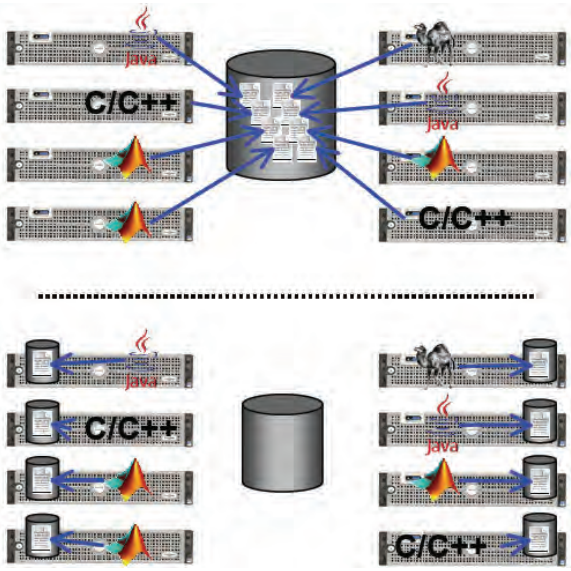


Figure 5. Traditional supercomputing applications (top) launch programs onto compute nodes and data is moved to these nodes. A different approach (bottom) is to put the data on the local storage of the compute nodes and then send the programs to nodes having the data.

The Hadoop[9] distributed file system (HadoopDFS) is a well-established open-source distributed file system modeled on the Google file system[13]. Hadoop is a replicated block-based distributed file system optimized for handling very large blocks, and is well-suited for managing large files that are larger than a typical storage device. HadoopDFS has mechanisms for managing block replication that can give it a high-level of fault tolerance. HadoopDFS is designed to support specialized applications that can take advantage of such a file system. Typically, these applications are launched using the Hadoop map/reduce execution framework, which tries to send the computation to the data blocks.

The Sector/sphere[11] distributed file system and execution framework is a file-based distributed file system based on peer-to-peer technology. Files are not broken up across storage nodes. Such a system is well-suited for storing many large binary files (e.g., image data) where computations are performed against a large set of files that will be statistically-distributed across the nodes. Sector/sphere presents a straightforward model that is easy to optimize with a high level of performance and security.

Relaxed consistency databases (i.e., No ACID) are one application that is well suited to HadoopDFS. HBase[12] and other "Big-Table-like" databases leverage the HadoopDFS by modeling Google Big Table. These databases are designed for data mining applications that do little read-modify-write, and where statistical consistency is more than adequate. Under these the relaxed restrictions there is potential to get a sizable increase in performance. These databases are typically NoSQL databases that have their own interfaces.

TokuSampleSort[10] is a parallel sorting engine that can be used for sorting very large files. The developers of this technology have also created the TokuDB database.

5. Interfaces

Tapping the performance potential of the storage technologies requires interfaces that can be used by the LLGrid users with minimal training. The standard file system graphics user interface (GUI) abstraction is one such interface that is widely familiar with the LLGrid user-base. LLGrid users are primarily scientists and engineers with a high-level of mathematical training, and high-level array-based languages (e.g., MATLAB) are also easy for them to learn.

A. Low-Level

Most LLGrid users access the system via their desktop computers (see Figure 1). For users on the LLGrid local-area network (LAN) this consists of mounting the LLGrid Lustre storage via the appropriate GUI interface (Samba or NFS). For users on the LLGrid wide-area network (WAN) this consists of mounting the LLGrid Lustre storage via a FUSE module for the SSHFS encrypted file system protocol.

Parallel programs are launched by a user typing a run command from their desktop that writes launch files into the mount-point. An SSH command is then issued that tells the scheduler to execute these files. For MATLAB users, this functionality is found within the gridMatlab software. A user issues a run command with a script they wish to execute and the number of processors needed. The system runs the command immediately, allowing for interactive parallel computing. An extension to gridMatlab allows the user to run against a set of files instead of a set of compute nodes, allowing gridMatlab to look up the locations of the files and send the computation to those nodes so that file IO can be local to the node.

In response to increased security threats, user desktop privileges are being curtailed. The above interfaces and administrative privileges (i.e., the ability to install software) are often not permitted. Thus, a new WebDAV-based interface to LLGrid has been developed called LLGridZF (Zero-Footprint). LLGridZF uses public key infrastructure (PKI)-based authentication and limits all communication to the http and https protocols. To support this capability, the Apache WebDAV server extension has been modified so that when a launch file is created on the mount point, the launch command is automatically run. This eliminates the need for SSH (which may not be allowed). In addition, LLGridZF requires that no software be installed on the user desktop. The overall operation of LLGridZF is shown in Figure 6. LLGridZF takes file-base supercomputing a step further by integrating LLGrid operations into a modified Apache server. Thus, at the server level it is possible to set the specific attributes of files that are allowed to be on the LLGrid system and the actions they trigger. Finally, it is a simple matter to record all interactions with this system in a manner that is readily auditable by not deleting files.

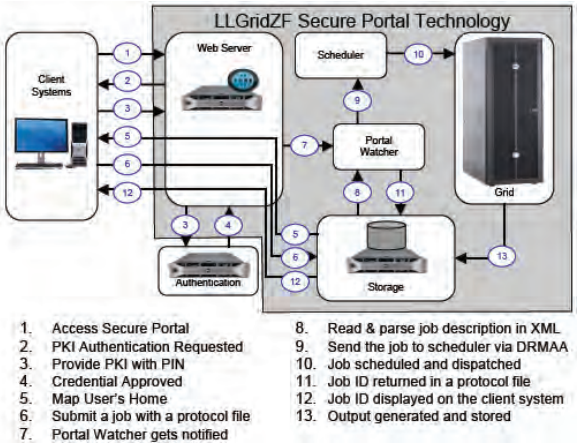


Figure 6. LLGridZF operation

B. High-Level

Making the LLGrid data storage system available to user applications consists of providing a number of MATLAB interfaces. The lowest-level interface is MatlabMPI for point-to-point communications between multiple instances of Matlab running on the LLGrid compute nodes. Variables are sent by the sender writing a buffer-file containing the variables followed by writing a lock-file (see Figure 7). The receiver polls for the existence of the lock-file, when it appears, the receiver reads in the buffer-file containing the variables.

pMatlab provides a higher-level parallel programming environment based on arrays. The central concept of pMatlab is using a "map" to distribute arrays across multiple processors (see Figure 8). Each instance of the program then operates on its local part of the array. Most parallel programs can be implemented in pMatlab with just seven pMatlab library functions: `Np` (number or processors), `Pid` (processor ID), `map` (create a distributed array map), `local` (get the local part of a distributed array), `put_local` (put local data into a distributed array), `global_index` (get the indices on a particular processor ID), and `agg` (aggregate a distributed array onto one processor ID). In addition, complex data movements can be performance by simply assigning one distributed array to another distributed array using an overloaded = operator.

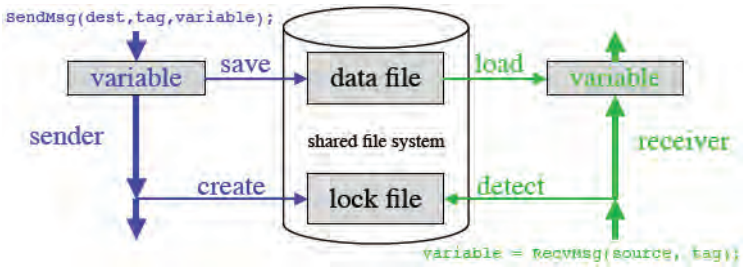


Figure 7. MatlabMPI file-based messaging. Sender writes variables to a buffer-file followed by writing a lock-file. Receivers polls for lock-file, when it appears, it reads in the buffer-file.

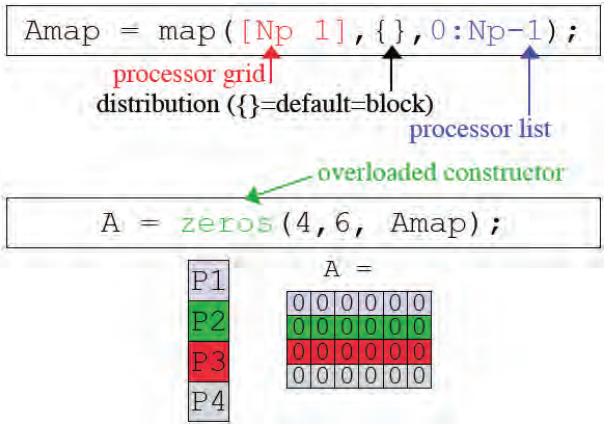


Figure 8. Anatomy of a pMatlab distributed array map. A map is used describe how an array is distributed across multiple processors.

pMatlabXVM (eXtreme Virtual Memory) extend pMatlab by providing a hierarchical array interface to the local storage on each node. A user constructs arrays with hierarchical maps that describe how the array is broken up across nodes and between memory and disks (see Figure 9). This allows the user to create arrays that are size of the total storage across all the nodes. The user then issues commands to the array for storing or retrieving blocks of data in parallel.

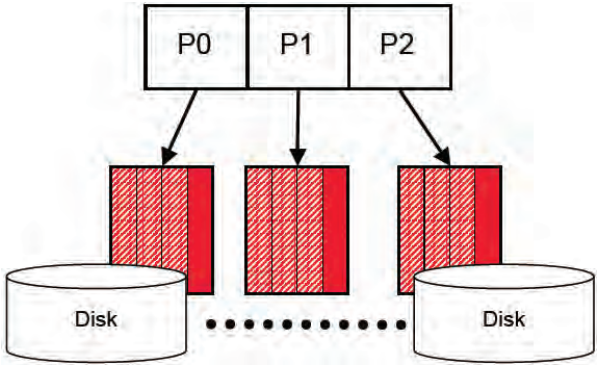


Figure 9. pMatlabXVM uses hierarchical descriptions of arrays to manage data across processors and between memory and disk

D4M (Dynamic Distributed Dimensional Data Model) provides a unified array-based interface to both SQL and NoSQL databases. D4M allows users to work on character strings the same way they would work with arrays of numbers. D4M is particularly useful in exploiting the increased capability offered by NoSQL databases or triple stores. D4M also provides a natural abstraction for exploiting the distributed array model found in pMatlab.

6. Usability

The LLGrid user-base is primarily deadline-driven, which creates a tension between usability and increased performance. In this section, the usability measurements (effort, code volume and training) of the LLGrid interfaces of LLGrid are discussed.

A. Low-Level File system Interface

There are a large number of steps between a user deciding they need to use a supercomputer and the user running a well-performing parallel code. Figure 10 shows these steps with typical times for a traditional supercomputer and the LLGrid file system supercomputer. The LLGrid file system-based approach allows the majority of these steps to be eliminated or significantly reduced by presenting the system as a single mount-point with all required programs and data prepositioned in the file system. The result is that the time from deciding to use a supercomputer to effectively running on it can occur in a few hours. This capability has been highly-successful with the LLGrid user-base and reduces the user support costs of maintaining the system.

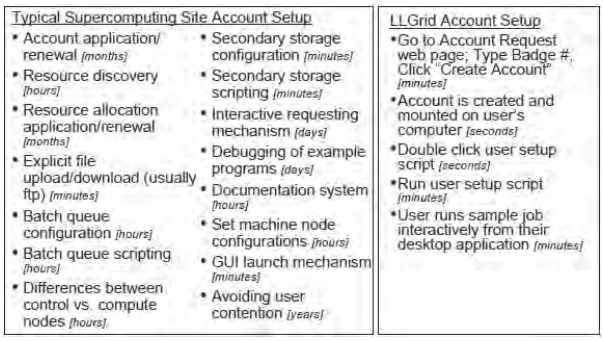


Figure 10. Comparison of the steps necessary to set-up a user on a typical supercomputer versus what is required for LLGrid

LLGridZF takes the above philosophy a step further (see Figure 6). LLGridZF eliminates steps from the baseline LLGrid interface and exposes fewer interfaces to an adversary. In addition, these interfaces are used in precise patterns defined by LLGridZF that allows reduced versions of these interfaces to be used (e.g., only certain arguments to scheduler are allowed). Finally, deviation from these patterns can be detected automatically. As a result, of the 13 steps carried out by LLGridZF, only three steps have user involvement at set-up time, and only one step requires regular user involvement during execution. Thus, LLGridZF simultaneously presents a heightened security profile along with an easier set-up and usage model. LLGridZF is in its early phases of deployment. It is expected that this capability will be well-received by the LLGrid user-base as PKI is deployed, since it allows the user to meet increased security requirements while lessening the number of steps they need to perform to get their work done.

A number of emerging low-level file system capabilities (e.g., Hadoop, Sector/Sphere) allows the user to take advantage of the high-bandwidth available on local file systems. However, the interfaces for these file systems are generally not integrated with standard parallel programming models. As described earlier, gridMatlab has been modified to incorporate the concept of launching a parallel program against a set of files. This is a fairly intuitive concept for the user-base. However, the core idea of positioning copies of data files across the nodes of a file system is still somewhat new to this user-base, and only relevant to those users whose file IO requirements are significantly larger than what can be offered by traditional parallel file system. Thus, direct access to these types of file systems may not be their most popular use. It may be that providing access to applications that sit on top of these emerging distributed file systems will be the most beneficial.

B. High-Level Programming Interface

Detailed analysis of the usability of LLGrid's high-level interfaces have been previously described in the literature. The most relevant results are summarized here.

MatlabMPI uses the look and feel of the MPI interface to allow file-based message passing between results. The usability of the MPI interface has been studied extensively[28]. It is typical for a serial program to increase in size by 1.5X when it is made parallel with MPI. MatlabMPI is no different, and thus is not very popular with the LLGrid user-base. The primary benefit of a file-based MPI is that it significantly reduces the size of the MPI implementation (~500 SLOCs versus

~5,000 SLOCs), which has made MatlabMPI an easy to maintain low-level messaging technology.

The array-based pMatlab interface is much more compact and intuitive way to write parallel programs. pMatlab hides the details of MatlabMPI message passing from the user. Studies of LLGrid pMatlab programs indicate that a serial program increases in size by just a few percent when it is made parallel with pMatlab[2]. The pMatlab capability has been highly-successful with the LLGrid user-base and reduces the user support costs of maintaining the system.

The goal of pMatlabXVM is to provide a high-level application interface to the raw IO power of the local disks. The XVM interface extends the pMatlab array interface by providing hierarchical abstractions to manage data in memory and on disk. Analysis of XVM implementations of HPC Challenge indicates that the effort of applying XVM is about 2X that of using pMatlab[29], and translates into a ~10% increase in code-size when applied to an application. This code-size increase may be acceptable. However, the training for XVM requires a more detailed understanding of the computing architecture. Finally, the core idea of treating memory as a hierarchical decomposition across the local disks on nodes of a supercomputer is still somewhat new to this user-base and is only relevant to those users whose memory requirements are significantly larger than what can be offered by traditional approaches.

D4M provides a compact interface to a variety of databases (both SQL and NoSQL). Early results indicate that database applications written with D4M are smaller than applications which use standard approaches (e.g., Java+SQL). Comparing Java and D4M implementations of the Graph Analysis benchmark using a NoSQL database showed that the D4M implementation was ~10X smaller. Thus, D4M simultaneously presents an increased capability along with decreased application size. D4M is in the early phases of deployment. It is expected that this capability will be well-received by the LLGrid user-base.

7. Performance

This section describes the IO performance results of LLGrid using a number of interfaces and benchmarks. The performance of the low-level hardware is described first, as this sets the baseline for performance for the high-level interfaces.

A. Low-Level File system Interface

The lowest-level performance measurement the LLGrid team has performed is to run the IOzone performance tool across all the disks in the system. The IO size was twice that of the available memory on the local node to minimize caching effects while still running in a reasonable time. Figure 11 shows results for the 400 nodes in the system are consistent with the hardware specifications of the devices. The device read-performance varies by ~2X and the write-performance varies by ~1.5X. The aggregate of these devices sets the maximum overall capability that can be provided by the distributed file systems that are built with these devices.

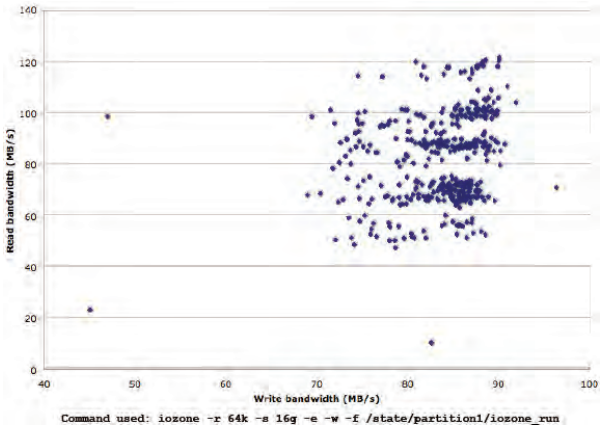


Figure 11. IOzone read versus write performance for the 400 local storage devices on LLGrid

In addition, to these distributed nodes, there is a large central parallel file system that delivers that vast majority of the day-to-day interactions to the LLGrid user-base. The performance of the Lustre system is ~20X the local node performance. However, in aggregate, the local node storage could theoretically deliver 20X the performance and the capacity of the Lustre file system.

The final low-level performance experiment was to conduct the Sort benchmark which gauges the accessibility of the potential offered by the large amount of local node storage. This experiment[10] produced results that set records for every category in the Sort benchmark, and demonstrates the performance offered by the local node storage is accessible in practice. As discussed earlier, the key challenge is to provide this performance via an interface that is readily-accessible to the LLGrid user-base.

B. High-Level Programming Interface

The performance of the low-level interfaces sets the bounds on what can be achieved in the high-level interfaces. Within the high-level interfaces, message passing is the core operation upon which the other high-level interfaces are built upon. The ping-pong benchmark is a good way to measure this performance and it establishes the asymptotic latency and bandwidth of the point-to-point file-based messaging.

Figure 12 shows the latency and bandwidth as function of message size obtained between two LLGrid compute nodes exchanging messages using the local storage and the Lustre file systems. The latency and bandwidth are determined by the flat parts in the curve. As expected, the performance of both file systems are comparable because both are limited by the link bandwidth from the compute nodes to the file systems and are achieving approximately 80% of that peak performance. Figure 13 shows the aggregate bandwidth and median latency as a function of the compute nodes participating in the benchmark. The performance scales linearly within the processor range measured.

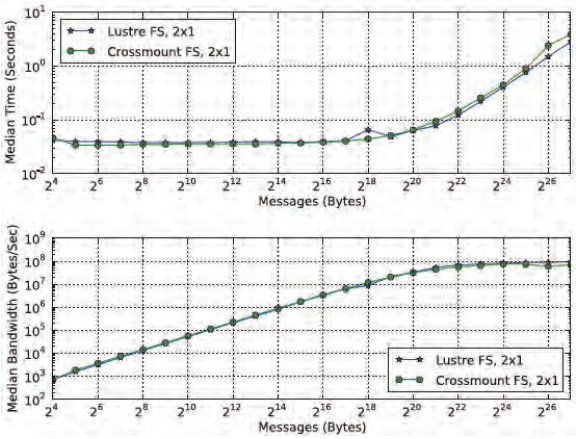


Figure 12. Ping-pong latency (top) and bandwidth (bottom) versus message size for the local nodes and the Lustre file system

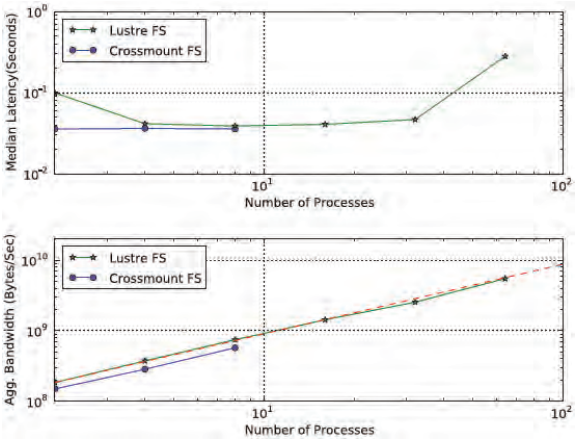


Figure 13. Ping-pong median latency (top) and aggregate bandwidth (bottom) versus the number of compute nodes participating in the benchmark

pMatlab provides a high-level interface to message passing. The most commonly used IO function in pMatlab is aggregation which collects all parts of a distributed array to a leader processor. Aggregation is found in all the benchmarks for collecting results at some point in the computation. Because the aggregated array must fit onto one processor, the data

size on each processor is small. Thus aggregation tends to be a latency dominated function. Figure 14 shows the latency of the pMatlab aggregation function versus the number of processors.

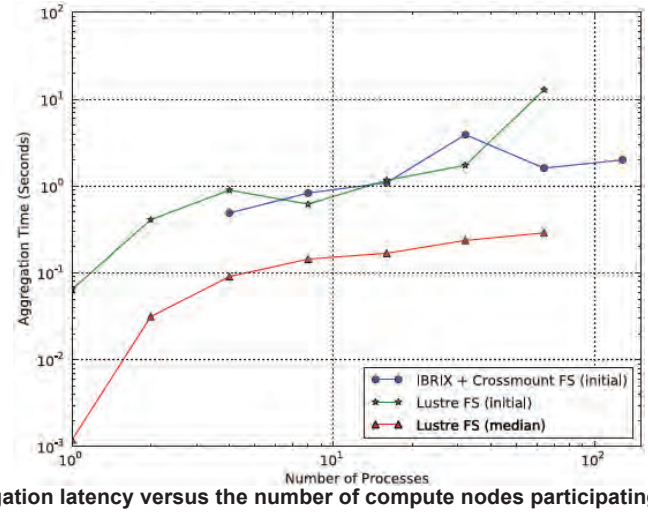


Figure 14. Aggregation latency versus the number of compute nodes participating in the benchmark

XVM provides a high-level interface to the IO potential offered by the local node storage. The HPC Challenge Stream benchmark is a good tool to measure the effectiveness of XVM at providing this performance. Figure 15 shows the performance of XVM on small and large problems for different numbers of processors. In both cases XVM achieves near perfect linear speed-up consistent with specifications of the devices.

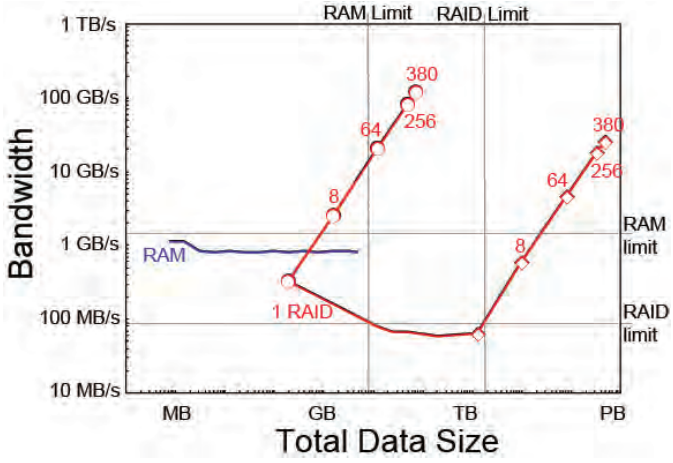


Figure 15. XVM Stream bandwidth versus total data-size. Bottom curve shows performance on 1 node for a range of problem sizes. Diagonal lines show the performance from 1 to 380 nodes for small (barely fits in memory) and large (barely fits in disk) problem sizes. RAM curves shows in memory Stream performance.

D4M provides a database neutral high-level interface to SQL and NoSQL databases. D4M syntax is particularly well-suited to exploiting the increased performance offered by NoSQL databases. The performance of D4M is measured by implementing the Graph Analysis benchmark. Figure 16 shows the single node insert and query performance of standalone D4M (in memory) and D4M connected to a NoSQL database (in storage). These results are consistent with the performance of the native Java interface and deliver near the theoretical performance limits of the hardware.

8. Conclusion

Lincoln Laboratory's advanced sensor prototyping requires interactive access to large parallel computing and data storage capabilities. The LLGrid supercomputer provides this capability to hundreds of researchers. The LLGrid user-base is deadline-driven. New technologies are embraced by the LLGrid user-base if they require minimal training. Ubiquitous

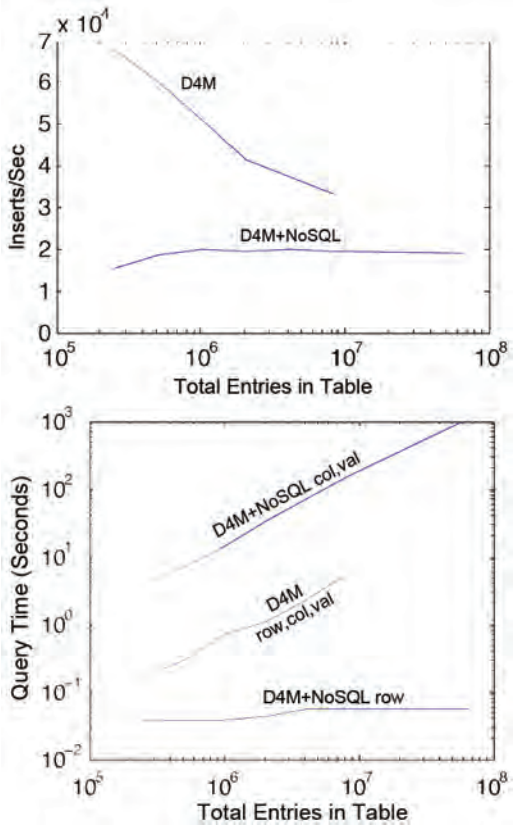


Figure 16. Single node D4M insert rates (top) and query times for row, column and value queries (bottom) versus the number of table entries. Curves are shown for stand-alone D4M (in memory) and D4M connected to a NoSQL database (in storage).

use of the file system abstraction is the key innovation that allows LLGrid to achieve these goals. All components of the LLGrid supercomputer are presented to the user as file systems. The LLGrid storage system has high-level interfaces for a wide-range of storage technologies: parallel file systems, file based distributed file systems, block-based distributed file systems, peer-to-peer file systems, parallel virtual memory, SQL and NoSQL databases, file-based launching, file-based messaging passing, and file-based secure portals. As a case study, LLGrid shows that storage technologies can perform all the critical functions in a supercomputer; these functions can be incorporated into high-level array-based programming environments; and keeping user effort to a few hours is the most important predictor of success with this user-base.

Acknowledgments

The authors would like to thank Kirk Jordan (IBM) and Vipin Sachdeva (IBM) for their help with porting LLGrid to the IBM Blue Gene. We are also indebted to Cray, Inc. for allowing us to port LLGrid to their XT line. Finally, Bradley Kuszmaul (TokuTek) was invaluable in implementing the Toku Sample-Sort on the LLGrid system. This work is sponsored by the Department of the Air Force under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author, and are not necessarily endorsed by the United States Government.

Appendix A: Hardware Details

All results presented in the paper are against the current LLGrid flagship system TX-2500 (all LLGrid systems have the TX prefix in homage of the first interactive supercomputers developed Lincoln TX-0 and TX-2). TX-2500 refers to the over 2,500 spindles that existing the over 400 RAID systems found in TX-2500. Each TX-2500 node has two processors with 4GB of RAM per processor core. Each node contains six 10,000 rpm serial attached SCSI (SAS) drives. Asymptotic data rate of each drive is 40 MB/sec. The asymptotic data rate for the RAID controller is 200MB/sec. The nodes are connected via a GigE interconnect.

The global storage array consists of a DDN10000 storage controller with 80 7200 rpm SATA2 and 80 7200 rpm SAS drives.

References

1. Kepner, J., *Parallel Matlab for Multi-core and Multi-node System*, SIAM Press, 2009

2. Bliss, N., R. Bond, H. Kim, A. Reuther, and J. Kepner, "Interactive Grid Computing at Lincoln Laboratory", *Lincoln Laboratory Journal*, Volume 16, Number 1, 2006.

3. Bliss, N. and J. Kepner, "pMatlab Parallel Matlab Library", *International Journal of High Performance Computing Applications: Special Issue on High-Level Programming Languages and Models*, J. Kepner and H. Zima (editors), Winter 2006 (November).

4. Wang, F., S. Oral, G. Shipman, O. Drokin, T. Wang, and I. Huang, "Understanding Lustre Filesystem Internals", *Technical Report ORNL/TM-2009/117*, Oak Ridge National Lab, April 2009.

5. Wu, D., P. Dhungel, X. Hei, C. Zhang, and K. Ross, "Understanding Peer-Exchange in BitTorrent Systems", *IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, 25–27 August, 2010.

6. Stonebraker, M. and L. Rowe, "The design of POSTGRES", *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data*, 2 June, 1986.

7. Axmark, D. and M. Widenius, "MySQL Introduction", *Linux Journal*, Issue 67, Nov. 1999

8. GRSecurity, http://grsecurity.net/.

9. HadoopDFS, http://hadoop.apache.org/hdfs/.

10. Kuszmaul, B., "TeraByte TokuSampleSort sorts 1TB in 197s", *SPAA 09 Proceedings of the Twenty-First Annual Symposium on Parallelism in Algorithms and Architectures*, 11–13 August, 2009

11. Grossman, R. and Y. Gu, "Data-mining using high-performance data clouds: experimental studies using sector and sphere", *KDD 2008 Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 24–27 August, 2008.

12. HBase, http://hbase.apache.org/.

13. Chang, F., J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber, "Bigtable: A Distributed Storage System for Structured Data", *ACM Transactions on Computer Systems*, Volume 26, Issue 2, June 2008.

14. Kim, H., C. Kahn, and J. Kepner, "pMatlabXVM: Parallel Matlab for Extreme Virtual Memory", *OSC Parallel Matlab Workshop*, Supercomputing 2005, Seattle, WA, November 15.

15. Kepner, J. and S. Ahalt, "MatlabMPI", *Journal of Parallel and Distributed Computing (JPDC)*, Volume 64, Issue 8, August, 2004.

16. Reuther, A., W. Arcand, C. Byun, B. Bergeron, M. Hubbell, J. Kepner, A. McCabe, P. Michaleas, J. Mullen, and A. Prout, *DDR&E LLGrid Portal: Interactive Supercomputing for DoD*, HPEC Workshop, 15–16 September, 2010.

17. Bigelow, D., S. Brandt, J. Bent, and HB Chen, "Mahanaxar: Quality-of-Service Guarantees in High-Bandwidth, Real-Time Streaming Data Storage", *26th IEEE Symposium on Mass Storage Systems and Technologies*, 3–7 May, 2010.

18. Parker-Wood, A., C. Strong, E.L. Miller, and D.D.E. Long, "Security-Aware Partitioning for Efficient File System Search", *26th IEEE Symposium on Mass Storage Systems and Technologies*, 3–7 May, 2010.

19. Pease, D., A. Amir, L.V. Real, B. Biskeborn, M. Richmond, and A. Abe, "The Linear-Tape File System", 2*6th IEEE Symposium on Mass Storage Systems and Technologies*, 3–7 May, 2010.

20. Seppanen, E., M.T. O'Keefe, and D.J. Lilja, "High-Performance Solid-State Storage Under Linux", 2*6th IEEE Symposium on Mass Storage Systems and Technologies*, 3–7 May, 2010.

21. Shvachko, K., H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System", 2*6th IEEE Symposium on Mass Storage Systems and Technologies*, 3–7 May, 2010.

22. Woo Son, S., S. Lang, P. Carns, R. Ross, R. Thakur, B. Ozisikyilmaz, P. Kumar, W.-K. Liao, and A. Choudhary, "Enabling Active Storage on Parallel I/O Software Stacks", 2*6th IEEE Symposium on Mass Storage Systems and Technologies*, 3–7 May, 2010.

23. Wan, J., J. Wang, Q. Yang, and C. Xie, "S2-RAID: A New RAID Architecture for Fast Data Recovery", 2*6th IEEE Symposium on Mass Storage Systems and Technologies*, 3–7 May, 2010.

24. Luszczek, P., J. Dongarra, and J. Kepner, "Design and Implementation of the HPC Challenge Benchmark Suite", *CT Watch*, Vol. 2, Number 4A, November 2006

25. Haney, R., T. Meuse, and J. Kepner, "The HPEC Challenge Benchmark Suite", *HPEC Workshop*, 19–21 September, 2006.

26. Bader, D., K. Madduri, J. Gilbert, V. Shah, J.y Kepner, T. Meuse, and A. Krishnamurthy, "Designing Scalable Synthetic Compact Applications for Benchmarking High-Productivity Computing Systems", *CT Watch*, Vol. 2, Number 4A, November, 2006.

27. Byun, C., J. Kepner, V. Sachdeva, and K. Jordan, "Toward Mega-Scale Computing with pMatlab", *HPEC Workshop*, 19–21 September, 2010.

28. Funk, A., V. Basili, L. Hochstein, and J. Kepner, "Analysis of Parallel Software Development using the Relative-Development Time-Productivity Metric", *CT Watch*, Vol. 2, Number 4A, November 2006.

29. Kim, H. and J. Kepner, "Parallel Out-Of-Core Programming in Matlab Using the PGAS Model", *Second Conference on Partitioned Global Address Space Programming Models*, 3–4 October, 2006.

HPCMP UGC 2011

5. Signal/Image Processing (SIP) and Sensors; Electronics, Networking, and Systems/C4ISR (ENS) and Testing

Electronics

Distributed Integrated Circuit Design

David A. Richie
Brown Deer Technology, Forest Hill, MD
drichie@browndeertechnology.com

Eric MacDonald and Matthew Markulik
University of Texas El Paso, TX
{emac, mcmarkulik}@utep.edu

Joseph Neff and Eric Bozeman
Space and Naval Warfare Systems Center Pacific (SSC Pacific), San Diego, Ca
{jdneff, eric.bozeman}@spawar.navy.mil

Abstract

Software for integrated circuit (IC) design has traditionally focused on the use of workstation-scale computing since this has been the most pervasive resource available to commercial end-users. We report on an investigation into the use of large-scale high performance computing (HPC) resources in support of IC design and modeling to reduce time-to-solution and expand the design space. Results are reported for a target workflow supporting the ongoing development of an unattended magnetic field detector at SSC-Pacific in support of the Marines' Tactical Remote Sensing System (TRSS). A Message Passing Interface (MPI) code was developed for driving parallel Verilog simulations of the detector circuit subject to an input step signal with noise, suitable for large-scale Monte Carlo (MC) simulations. Noise was generated using unique random seeds based on the parallel random number generator SPRNG. This code was adapted to perform Markov Chain MC (MCMC) simulations exploring the optimum detector threshold and algorithm variation. Using 128 cores on the US Army Research Laboratory Department of Defense (DoD) Supercomputing Resource Center (ARL DSRC) SGI Altix 8200 (Harold) over 4,000 Verilog simulations were performed in less than 10 minutes. Based on these initial results, the approach has been extended to higher-fidelity SPICE simulations in an effort to study a sub-threshold logic operation that offers the potential for dramatically-reduced power consumption in electronics applications.

1. Introduction

Many DoD research and engineering centers are charged with developing future combat systems that have Application-Specific Integrated Circuits (ASICs) as fundamental components. Reducing the time-to-solution during the design and modeling stages would enable accelerated schedules and expand the solution space available for exploration in optimizing the designs. At present, the large-scale computing resources of the High Performance Computing Modernization Program (HPCMP) are not extensively utilized for such work, despite the significant need for raw computing power.

Sub-threshold logic operation is gaining attention, particularly in the Department of Defense (DoD), for providing significantly-reduced energy per operation for applications in which high-performance is a secondary consideration. The property of low power consumption is particularly important in hand-held or unattended ground sensor applications, which operate on batteries or energy harvesting from the environment. One significant challenge with this new circuit regime is that digital circuits are now more dramatically affected by environmental (Vdd, temperature), design (load capacitance, input transition time), and process (threshold voltage, channel length) variations[1–3] and; therefore, require more accurate modeling and simulation. Sub-threshold circuits have exponential dependencies on variation, and require large circuits to be simulated with high-fidelity SPICE (Simulation Program with Integrated Circuit Emphasis) simulators.

The challenge of sub-threshold operation from a simulation perspective is that the model reduction used to generate timing constraints on components of a circuit for Register-Transfer-Level (RTL) simulation is conservative and does not reflect the true limits of operation in an application that requires reduced-power operation. Historically, the software developed for IC design and modeling has focused on workstation-scale computing platforms most typical across the commercial industry. Design methodologies over the past three decades have evolved to include simulation with tiers of fidelity, with a tradeoff between accuracy and circuit size. In order to simulate larger circuits, levels of abstractions were necessary to simplify the simulations to match the available computing resources. Simulation of sub-threshold behavior requires large circuits to be simulated with high-fidelity SPICE simulators.

Used as a representative system, the unattended magnetic field detector under development for the Marines' Tactical Remote Sensing System (TRSS)[4] has a stringent power requirement. Low-power, highly-sensitive, magnetometer sensors are crucial to the Warfighter, in order to provide immediate information regarding slight changes in the near (<30 feet) magnetic field. The fluxgate magnetometer is the design currently under test in this research effort, and serves as a test-bed. Although this magnetometer does not require high-performance control circuits, the necessary electronics must operate on close to zero power based on energy constraints. One possible solution is to run the digital electronics at very-low voltages, far below traditional levels, as power is quadratically related to the supply voltage.

For the Advanced Dynamic Magnetometer (ADM) system, the majority of complexity is captured in a single digital chip that controls a small set of analog electronics to drive and monitor current through a coil with material having a high magnetic susceptibility. The majority of the complexity of this ASIC is confined to the detection algorithm, which detects a change in the incoming signal, while ignoring significant noise and the Earth's normal magnetic fields. The detection algorithm is comprised of a moving-average algorithm. The detection algorithm is implemented with approximately 20,000 transistors. In contrast, typical SPICE simulations in industry are limited to less than 1,000 transistors. The full ASIC is approximately 100,000 transistors, which is substantially larger than typical industrial simulations.

In this work, we employ RTL and SPICE simulation of large digital integrated circuits within a parallel Monte Carlo framework, in order to exploit the large numbers of cores provided by HPC platforms aimed at the systematic exploration of the design space for integrated circuits.

2. Methodology

2.1 Conventional Simulation Techniques for Modeling Digital Integrated Circuits

This work employs the techniques of RTL and SPICE to model large digital integrated circuits. RTL simulation models a digital circuit by propagating, in time, the binary representation of state information within the digital circuit, subject to strict timing constraints with one or more clocks driving the dynamic operation of the circuit. RTL simulation is well-suited for complex technology and environmental conditions by incorporating high-level models for the components of the circuit. The operation of the circuit proceeds in discrete steps with an overriding requirement that all propagation delays and other factors not exceed timing constraints that would render the behavior of the circuit ill-defined. RTL simulation requires the specification and synthesis of the circuit in either Verilog or VHDL. In this work, Verilog is used along with the Verilog simulators VeriWell (open-source)[5] and Cadence NC VERILOG (commercial).

SPICE is a general-purpose open-source analog electronic circuit simulator, and provides greater fidelity for modeling digital circuits, especially when the circuit is to be operated under extreme conditions or in sub-threshold operation. SPICE models (BSIM3.3 - Berkeley Short-channel Igfet Models) remain the standard for all circuit simulators. SPICE simulations utilize detailed models of the transistors used to construct large-circuits and target a specific fabrication technology.

The higher-fidelity modeling provided by SPICE simulation relative to RTL simulation coincides with longer simulation times and greater memory requirements. Although having a more effective simulation allows for better analysis of circuit behavior under varying conditions, the amount of time taken to explore these varying conditions can amount to weeks or months of simple testing. SPICE must calculate transistor model parameters for each transistor at each transient step, in order to arrive at a high-fidelity solution. With varying levels of transistor models, the accuracy of each simulation will vary. Transistor models include many factors such as intrinsic capacitance, channel dimensions, junction sizes and additional relevant transistor properties. Each transistor model is specific for a target fabrication technology, e.g., the 250nm channel length that is the target of the SPAWAR ADM ASIC.

2.2 Large-Scale Integrated Circuit Simulation

The approach taken in this work differs from the conventional use of RTL and SPICE simulation in two respects. First, much larger circuits are targeted for direct SPICE simulation without model reduction. Second, both the RTL and SPICE simulations are employed as sub-simulations within a larger parallel Monte Carlo simulation for studying design and operating parameters.

As shown in Figure 1, we use the results of an RTL Verilog synthesis step to construct a full SPICE net list of the larger circuit and perform high-fidelity transistor-level SPICE simulations. Synthesis is a step performed for RTL simulation that takes in an abstract behavioral Verilog description of a large circuit and generates a gate-level net list. Performing SPICE simulations of an entire digital circuit, presents a significant technical challenge in constructing a suitable representation.

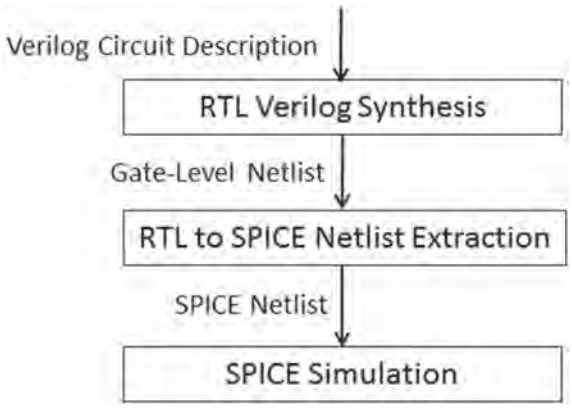


Figure 1. Workflow required to perform high-fidelity transistor-level SPICE simulation of a large digital circuit. A Verilog circuit description is synthesized into a gate-level net list. The result of the synthesis step is used to extract a SPICE net list suitable for transistor-level SPICE simulation of the entire circuit.

A key practical issue is that a synthesis net list uses named unordered arguments (inputs and outputs) for sub-modules, whereas SPICE modules require ordered argument lists for sub-modules. This adds to the complexity of extracting a net list for SPICE simulation from the synthesis of a large circuit. A tool was created for performing this function and verified with several SPICE simulators to ensure that simulator-specific syntax and options were respected.

Another challenge for SPICE simulation of large circuits based on RTL synthesis involved the initial DC convergence required as a pre-condition for transient analysis. An initial condition must be established self-consistently for each node in the circuit. For input nodes connected to a voltage source such as a flip-flop, which are bi-stable and have no *a priori* voltage without a forcing function, additional specifications are required within the SPICE net list. Moreover, many such nodes exist deep within the hierarchical design generated by the synthesis step. Significant effort was required to incorporate an automated mechanism for ensuring DC convergence.

For RTL simulations of large digital circuits, recording the output of the simulator involves recording the digital state at every clock transition, i.e., one or more binary values are recorded at a relatively limited number of discrete steps in time. Recording the output of a SPICE simulation for an equivalent circuit requires recording the dynamic voltage at each respective bit as a function of time, causing a tremendous increase in the amount of data generated by the higher-fidelity simulation. When scaling-up to running large numbers of SPICE simulations in parallel for large digital circuits, the treatment of the data generated becomes a significant issue. Options for handling this flood of data include either down-sampling the data to perform a crude reduction, or simply throwing the data away after suitable metrics are extracted. Down-sampling has the negative consequence of losing fast transients that might be key to understanding the circuit behavior or failure mode. Simply throwing the dynamic data away prevents the ability to go back and examine anomalous behavior identified during large-scale simulations.

This issue is not unique to large-scale SPICE simulations, but is encountered in many HPC applications. The output routine of the SPICE simulator Ngspice[7] was modified to integrate a generalized compression engine provided by StormRT[8] designed for the real-time compression of time-series in high performance computing (HPC) applications. The modifications enable compressed output that can achieve compression factors on the order of 100x while still maintaining a faithful representation of fast transients in the circuit. By design, the compression algorithm requires a minimal memory footprint and imposes a negligible additional computational load on the simulator. This approach allows the very large amount of data generated in the simulations to be treated in a manageable way.

2.3 Parallel Monte Carlo Simulations for Integrated Circuit Design

In considering how to utilize HPC resources for IC design and modeling tasks, consideration was given to the fact that the most likely use for large-scale resources would be to run thousands of simulations required to sample the parameter space of a given design. The approach taken was to treat the RTL Verilog and SPICE simulations as sub-simulations within a larger parallel Monte Carlo (MC) simulation for which a Message Passing Interface (MPI)-based simulation framework was developed.

The basic structure of the parallel framework is shown in Figure 2 for the case of parallel MC SPICE simulation. The MPI run is first launched with *n* processes and a simple master-worker model is used to initiate repeated cycles of *n*-1 simulations. The master process, *proc[0]*, first sends instructions to each of the workers, *proc[1] ... proc[n-1]*, which can

be used to control the individual simulations for a given iteration. Each worker process then generates a unique test-bench for the simulation, which will include a net list description of the circuit and all input files and parameters controlling the simulation. Parameters are generated so as to explore state space following the standard Metropolis algorithm for generating a Markov Chain Monte Carlo (MCMC) simulation. The random selection of the MC algorithm employs SPRNG, which is a parallel random number generator designed for accurate large-scale parallel simulations and suitable for MC studies. Once the test-bench (circuit, parameters and input stimulus) has been generated, a sub-simulation is then initiated from the worker process. Upon completion, a post-processing script is run to extract information from the simulation that is used to evaluate a metric to determine the quality of the results. The results from individual workers can be packed and sent to the master MPI process, which can aggregate the results and alter the instructions for the next cycle.

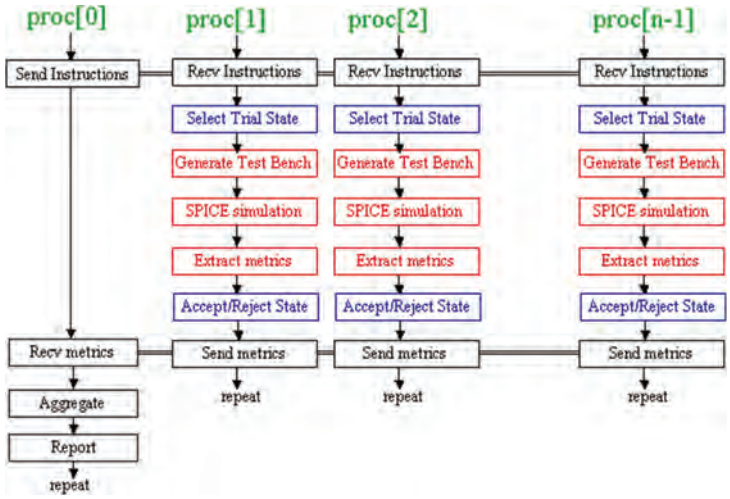


Figure 2. Program flow for the MPI Monte Carlo simulation code designed to drive SPICE simulations in parallel to perform a Markov Chain Monte Carlo simulation of a digital circuit

In order to demonstrate the approach, MC RTL simulations were used to study the detector threshold and permutations of the detection algorithm for the ADM ASIC. The state space consisted of: 1) an input data word representing a value between 0 and 1,023 as an offset to the dynamic threshold, and 2) a 4-bit configuration data word used to select one of four algorithm variants controlling the length of time transient changes are measured, and the number of consecutive rolling averages used to calculate the dynamic threshold. The standard Metropolis algorithm was used to sample the space spanned by the threshold and configuration data words. The threshold was treated as a pseudo-continuous variable where a random change was selected from a Gaussian proposal distribution. For the configuration data word, a proposal distribution was constructed as follows. With a 50% probability, no change at all was made to the configuration word; for the balance, there was an equal probability that a single bit was flipped. Each simulation was also unique due to the random noise that was superimposed on the magnetic step signal. The introduced noise in the signal affects the dynamic threshold of the circuit, and therefore permits an analysis of the circuit to determine a threshold that is best-suited for the noise environment.

A metric was required to measure the quality of a given state (threshold and configuration) based on the output of the simulation. The following metric was constructed to reflect a reasonable measure of the quality of the detection circuit's behavior,

$$F = \exp\left(+10 \cdot \left[N_r - N_f/50 - T_r/100 - 1\right]\right) \quad (1)$$

where $\boldsymbol{N_r}$ is the number of real-detects (0 or 1), $\boldsymbol{N_f}$ is the number of false-detects (unbounded), and $\boldsymbol{T_r}$ is the number of cycles until the real-detect. This metric has the property of being equal to 1 for the perfect case of a single real-detect, and remains positive for all cases, with strong weighting of the correct operation of the detector. The specific weights for real-detects (1), false-detects (1/50) and real-detect delay (1/100) were determined by the amount of impact they have on the overall results, based on an understanding of the application. This metric was used following the standard Metropolis algorithm to determine whether a new state should be accepted or rejected in the course of constructing the Markov chain exploring state space.

For higher-fidelity MC SPICE simulation, a demonstration was constructed using a digital circuit representative of the components used in the detector circuit that could be more easily scaled for systematic study. The objective was to

sample the state space spanned by V_{dd}, V_{enable}, and the transition time and period of the input clock for a large-counter, which incorporates an adder with a strong dependence on propagation delays with increasing numbers of bits. For each circuit parameter within this state space, a random change was selected from a Gaussian proposal distribution. These parameters were restricted to the ranges 0–2.5 V for V_{dd} and V_{enable} and 20-2,000 nsec for the clock-period.

The following metric was constructed to reflect a reasonable measure of the quality of the results with a focus on requiring that the circuit operated to completion with a preference for lower energy usage. The metric was,

$$F = \exp\left(+10 \cdot \left[-f_1^2/100 - f_2^2/500\right]\right) \quad (2)$$

where f_1 provides a penalty based on circuit failure and is proportional to the $\log_2$ of the clock-cycle where failure occurred, and f_2 provides a penalty for energy usage being proportional to the $\log_{10}$ of the total energy consumed in the operation of the circuit. This metric was used following the standard Metropolis algorithm to determine whether a new state should be accepted or rejected in the course of constructing the Markov chain exploring state space.

3. Results and Discussion

3.1 Parallel Monte Carlo RTL Simulation

Parallel MC RTL simulations were run on the ARL-DSRC SGI Altix ICE 8200 platform Harold using up to 256 cores. The detector was subject to a single magnetic step signal with a magnitude of 50, a rise-time of 5 and random noise with a standard deviation of 1.8. The state space of threshold and configuration were explored using the Monte Carlo technique described above in section 2.3. The rate of acceptance of proposed states was approximately 30%, which is well in-line with the conventional recommendation of approximately 25% for Monte Carlo simulations to provide a good sampling of state space. An examination of the resulting time-series for the configuration data word from a 128-core run, executing a total of 16,128 individual Verilog simulations, indicates that this component of state space is well-sampled. The distribution of threshold values for a given configuration data word are shown in Figure 3 for selected configurations. Here, 13 (binary 1101) and 5 (binary 0101) represent configurations strongly-favored, whereas 10 (binary 1010) was less favored. The distributions provide an initial glimpse of acceptable threshold ranges for a given configuration and demonstrate that the technique can be used in more ambitious simulations to determine more precisely the optimum operating configuration and threshold in the context of specified environmental factors such as noise and signal quality.

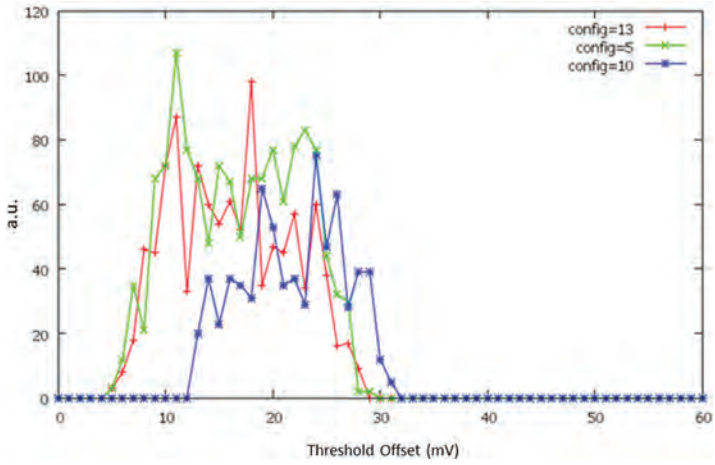


Figure 3. Distribution of threshold values for various configuration data words from Monte Carlo a simulation using 128 cores on the ARL-DSRC SGI Altix ICE 8200 platform Harold

3.2 Synthesis to SPICE Net list Extraction

Prior to performing MC SPICE simulations, synthesis to SPICE net list extraction was tested using a set of counters of varying sizes, an autoregressive moving average unit, and the full-detect algorithm from the ADM ASIC. The 64-bit counter contains a full-adder with significant propagation delay to warrant study in terms of reduced voltage and variable clock frequency operation. This circuit uses only fundamental logic gates, e.g., nands, ors, xors, flip-flops, and inverters. The net list for this circuit contains 1,602 nodes, 2,940 elements and 2,936 transistors. Once synthesized using a 40 ns clock, the RTL net list was converted to a SPICE net list with the Virginia Tech standard cell library. The resulting net list was verified

using HSPICE[7] and Ngspice to allow verification of correct operation and obtain performance benchmarks. Simulation times on a high-end dual-opteron workstation was 905 secs and 2,955 secs using HSPICE and Ngspice, respectively, where HSPICE used 6-cores and Ngspice used a single core.

The autoregressive moving average unit utilized multiplexers, flip-flops, and full-adders in addition to the basic logic gates. Using the same timing constraints, this more complex circuit required 1,316 nodes, 2,645 elements and 2,614 transistors. The resulting net list was also verified using HSPICE and Ngspice for comparison. Simulation times were 367 secs and 1,993 secs using HSPICE and Ngspice, respectively, where HSPICE used 5-cores, and Ngspice again used a single core.

The detect algorithm is a very complex circuit that utilizes a counter and moving average units as components. Within this circuit, all levels of logic gates are utilized from basic logic to more advanced combinatorial and cascaded logic. The SPICE net list contains 8,924 nodes, 16,871 elements and 16,832 transistors. The net list was verified using HSPICE and Ngspice, respectively, with simulation times of 32,451 secs and 98,405 seconds.

3.3 Integrated Data Compression for SPICE Simulation

The improved management of large data sets discussed in Section 2.2 was initially tested using the 64-bit counter, with Ngspice set-up to write the lowest N bits of the counter along with two additional signals. Comparison between the file size when using the original Ngspice output format and the MCF compression format used by the integrated data compression engine shows that increasing the number of bits from 4 to 64 resulted in a 10x increase in file size using the Ngspice format; whereas using the MCF compression, the file size increases by only 1%. The original format generates file sizes proportional to the number of signals, regardless of the dynamic activity of those signals. In contrast, the size of the compressed data files show almost no increase in size with the addition of signals expected to fluctuate at a decreasing rate. This is a reflection of the underlying wavelet transform used in the compression of the dynamic signals, which automatically limits the encoding of data to fluctuations across all time-scales.

Despite the significant reduction in file size, the signals can be faithfully reconstructed including fast transients. Shown in Figure 4 is the original and reconstructed signal from the MCF compression algorithm. The reconstructed signal closely approximates the original signal and captures all relevant features in the dynamic signal.

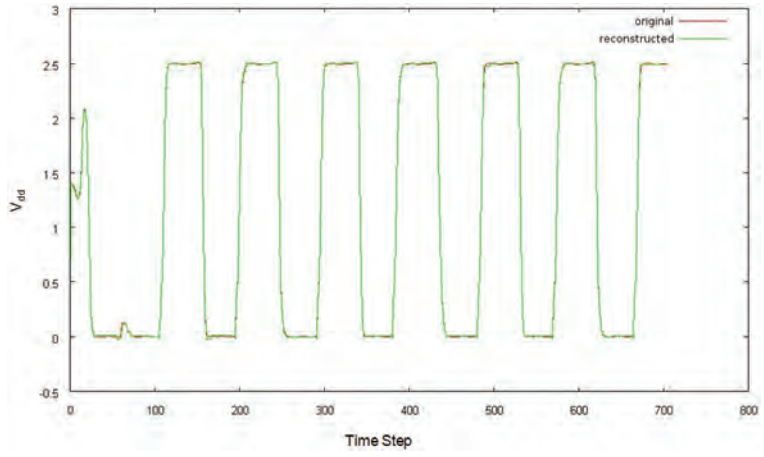


Figure 4. The original and reconstructed signal from a SPICE simulation where the reconstruction is based on the integrated MCF compression engine that significantly reduces the amount of data generated by the SPICE simulation

3.4 Parallel Monte Carlo SPICE Simulation

Parallel Monte Carlo SPICE simulations were run for a 64-bit counter using the method described in Section 2.3. A total of 5,080 SPICE sub-simulations were run on 128 cores of the ARL-DSRC Emperor platform that was used for the initial development and testing. The run-time for this simulation was approximately 7 hours. Of the 5,080 simulations generated in the Monte Carlo sampling, 4,469 simulations produced correct operation. The acceptance rate in the Monte Carlo sampling was initially 75%, which is somewhat higher than the 25% acceptance rate recommended for efficient Monte Carlo sampling. The consequence of a higher acceptance rate is one of efficiency insofar as the state space will

be explored more slowly than desired. Increasing the width of the trial distributions used to generate new states in the sampling reduced the acceptance rate to 64%, with 3,849 simulations showing correct operation. An examination of the autocorrelation functions for the state space parameters showed a corresponding reduction in the correlation between samples. The autocorrelation functions exhibited the correct exponential form expected for MCMC simulation. Despite these issues, the initial simulations showed a good sampling of the operating parameters of the circuit.

The inputs that were varied in the analysis include the supply voltage (V_{dd}), the input-enable voltage (V_{enable}), and the transition-time and period of the input clock. The correct functional operation of the device was checked, as the counters were held in reset for three cycles and then allowed to operate counting to the value 127 (final count of a seven-bit adder). Critical timing paths were anticipated to cause timing failures at the $\log_2$ transitions in the count value (15 to 16, 31 to 32, etc.) increasing in difficulty with a linear relationship with the bit value.

Figure 5 illustrates the relationship that emerges between supply voltage and total energy for successful simulations. A structure becomes apparent in which a local minimum energy is seen at 2.0V, and then the energy declines as supply voltage approaches zero. In this case, a well-understood relationship between supply voltage and dynamic energy explains the general decrease in energy consumption. The local increase, seen from 2.0 down to 1.5 V, is likely related to increased transition-times in the operation of the transistor, which increases short-circuit current. Below 1.5V the decrease in dynamic power begins to dominate.

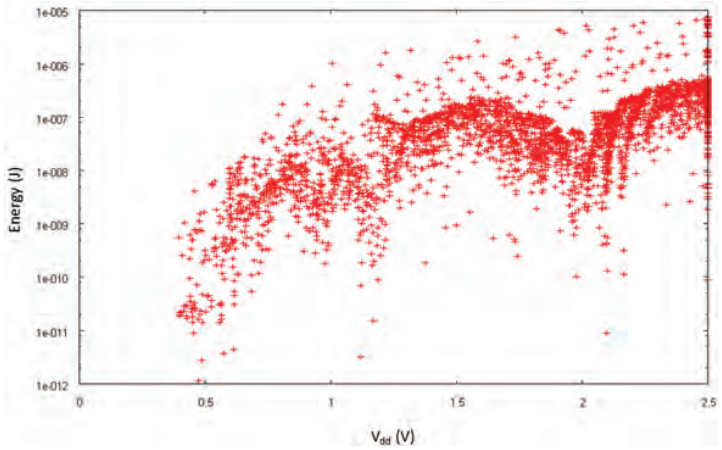


Figure 5. For each SPICE sub-simulation exhibiting correct operation, the energy consumed by the circuit is plotted as a function of reference voltage V_{dd}

Figure 6 illustrates that even though energy per operation should be unrelated to frequency or clock-period in a dynamic sense, static power (sub-threshold leakage) begins to increase the energy used as the simulated time is increased. The plot also shows two distinct lines, which likely are the clustering of simulations around two supply voltages that include the local minimum of 2.0V and the lower absolute minimum.

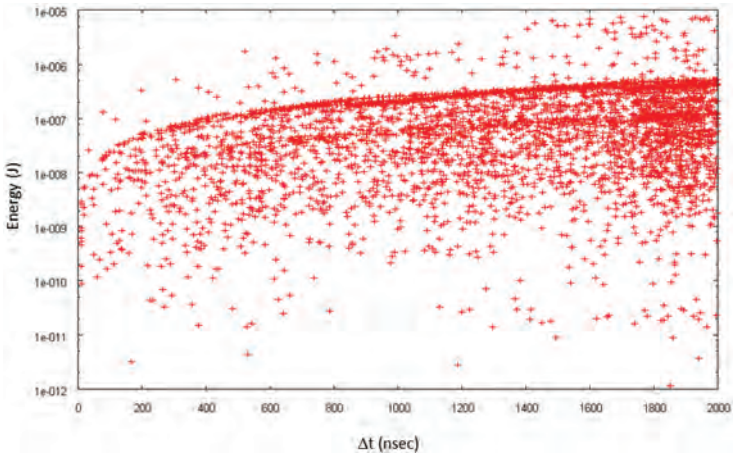


Figure 6. For each SPICE sub-simulation exhibiting correct operation, the energy consumed by the circuit is plotted as a function of the clock-period Δt

4. Conclusion

The work demonstrated a pragmatic approach that can integrate existing software to target project-specific analyses supporting IC design and modeling. Parallel Monte Carlo simulation using RTL and SPICE sub-simulations were demonstrated targeting large digital circuits of interest to the DoD. It was shown that ICs with large state spaces can be explored more rapidly using Monte Carlo techniques with high-fidelity, mathematically-complex circuit simulators for circuits the size of which would normally preclude simulation. These circuit simulations are in turn well-suited for HPC systems with very large numbers of cores.

The RTL simulation was based on the use of the open-source VeriWell Verilog simulator. Using 128 cores, more than 5,000 RTL sub-simulations were performed as part of a Monte Carlo sampling of parameters impacting the operation of the digital circuit. The SPICE simulation was based on an open-source simulator Ngspice for which the performance was within 50% of commercial tools, while providing similar fidelity. The methodology developed in this work encountered significant technical challenges that were successfully addressed. A method for automatically extracting a SPICE net list from the output of an RTL synthesis was developed. The treatment of large data sets generated when running thousands of SPICE simulations was addressed by integrating a real-time data compression engine into the Ngspice simulator. Circuits were simulated comprehensively that contained almost 3,000 transistors, being generally much larger than typical industrial circuit simulations, and provided a benchmark for further investigation.

5. Significance to DoD

This work will impact many DoD research and development centers distributed across the country that are charged with developing future combat systems for the Services. Since integrated circuits form the heart of most of these systems and custom ICs are required for many of them, it would greatly benefit DoD engineers to have access to the most advanced design tools available. This enables the use of the latest fabrication technologies to produce high-performance ICs. Development work on ICs is being done in all three Services (Army, Navy, and Air Force), as well as NSA and other associated government organizations.

This work will directly benefit efforts at SSC-Pacific related to ongoing projects that require the design and modeling of integrated circuits. One representative system that will benefit from this capability is a sensor communications system being developed at SSC-Pacific. This battery-powered system combines a receiver and transmitter along with digital control and signal processing into a product smaller than a deck of playing cards. This system is based on custom integrated circuits conceived and implemented by DoD engineers. Another representative sensor is an unattended magnetic field detector under development for the Marines' Tactical Remote Sensing System (TRSS) that has a stringent power requirement. The developed method and software will enhance the ability to model and develop electronic circuits that operate with extremely low power.

Acknowledgements

The authors would like to acknowledge the support received from the High Performance Computing Modernization Program Office (HPCMPO) for this work.

References

1. Wang, A. and A. Chandrakasan, "A 180-mV Subthreshold FFT Processor using a Minimum-Energy Design Methodology", *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, pp. 310–319, January 2005.

2. Soeleman, H., K. Roy, and B. Paul, "Robust Ultra-Low-Power Sub-threshold DTMOS Logic", *Proc. Int. Symp. Low-Power Electronics Design*, pp. 25–30, 2000.

3. B. Calhoun, A. Chandrakasan, "Ultra-Dynamic Voltage Scaling (UDVS) using sub-threshold operation and local voltage dithering", *IEEE Journal of Solid-State Circuits*, Vol. 41, No. 1, pp. 238–245, January 2006.

4. The Advanced Dynamic Magnetometer project (SSC-Pacific Code 71730) is a mature compact magnetic field sensor with stringent power requirements. The project can directly benefit from the effort described here by advancing the development of a low-power digital core as well as analog circuitry. The project is supported by the Marines and is in evaluation for use in their TRISS and TUSS systems.

5. VeriWell is an open-source Verilog simulator, http://sourceforge.net/projects/veriwell/develop.

6. Ngspice is an open-source SPICE 3F5 simulator, http://sourceforge.net/projects/ngspice/.

7. HSPICE is a commercial SPICE simulator provided by Synopsys, Inc., http://www.synopsys.com.

8. StormRT is an open-source library that supports real-time temporal data compression and feature detection, http://www.browndeertechnology.com/stormrt.htm.

Studies of Perovskite Materials for High Performance Piezoelectrics and Non-Volatile Memory

Tingting Qi, Joseph W. Bennett, Wissam Al-Saidi, Ilya Grinberg, and Andrew M. Rappe
The Makineni Theoretical Laboratories, Department of Chemistry, University of Pennsylvania, Philadelphia, PA
rappe@sas.upenn.edu

Abstract

Perovskite materials are crucial in a variety of important technological applications. Using quantum-mechanical simulations, we have computationally investigated ferroelectric materials for applications in computer memory and piezoelectric devices. We have determined that tetragonality of perovskite ferroelectrics, which is crucial for high piezoelectric performance, exhibits a quadratic dependence on the displacement of the B-site cations only. This provides guidance for the design of ferroelectrics and piezoelectrics with desired properties. We have also shown that a high piezoelectric response is likely to be present in lead-free, environmentally-friendly $Sn(Al_{1/2}Nb_{1/2}O_3)$ solid solutions. In a study of ultrathin ferroelectric films, we have shown that the details of metal-oxide bonding at the electrode-ferroelectric interface such as oxide termination play an important role in determining polarization stability and the miniaturization limit of ferroelectric devices.

1. Introduction

There is a need for perovskite oxides (formula ABO_3) with higher performance for use in a wide variety of applications such as actuators, sensors and data storage. For example, perovskite solid solutions with high piezoelectric response are used in current and next-generation military SONAR devices. When deformed by the external underwater sound vibrations, a piezoelectric material generates an electric field. This is then interpreted to gain information about depth, distance, and the identity of the source of the sound.

The flexible structure of perovskites lends itself to a variety of applications, depending upon the choice of *A*- and *B*-site atoms. In particular, the relationship between the sizes of the *A*-O and the *B*-O_2 sub-lattices plays an important role in determining the properties of the materials. This relationship is characterized by the tolerance factor (*t*) given by:

$$t = \frac{R_{A-O}}{R_{B-O}\sqrt{2}} \tag{1}$$

where R_{A-O} is the sum of *A* and O ionic radii and R_{B-O} is the sum of *B* and O ionic radii[1]. Tolerance factor $t<1$ usually leads to the rotation and expansion of the B-O_6 octahedra. Such octahedral rotations often generate a low-temperature anti-ferroelectric phase (e.g., $PbZrO_3$). If $t > 1$, the B-O_6 octahedra are stretched from their preferred B-O bond lengths, promoting B-cation distortions by creating room for the B-cations to move off-center. Therefore, simple perovskites with $t >1$ are usually ferroelectric.

For example, when Pb and Ti are paired as *A* and *B*, respectively, these cations move in a concerted manner, so $PbTiO_3$ (t = 1.06) is ferroelectric. Ferroelectric ABO_3 oxides are sensors in SONAR devices and are also useful for non-volatile random access memory (NVRAM) devices. When external underwater sound vibrations deform such a SONAR device material, it generates an electric field which can then be interpreted to gain information about depth and distance and the identity of the source of the sound. Ferroelectric random access memory (FeRAM) is one of a growing number of alternative non-volatile memories, as it offers lower power usage, faster write-speed and a much greater maximum number of write-erase cycles.

Highly-accurate modeling is necessary to understand the origin of the various properties exhibited by perovskites used in the current state-of-the-art technology, in order to speed up the design of new materials with enhanced performance for

future devices. Over the last decade, first-principles calculations have emerged as a vital tool for understanding complex solid-state systems due to a combination of methodological improvements and faster computer speeds. In particular, density functional theory (DFT) is a powerful technique for the description of ground-state properties of metals, semiconductors, and insulators[2,3] due to a combination of accuracy and computational efficiency. Here, we report on our first-principles DFT studies of materials design of new, lead-free piezoelectrics and relationships between composition and behavior of ferroelectric solid solutions.[4,5] We also investigate polarization stability in ultra-thin ferroelectric films, important for miniaturization of non-volatile memory devices.[6]

2. Results and Discussion

2.1 Correlations Between Tetragonality, Polarization, and Ionic Displacement in $PbTiO_3$-Derived Ferroelectric Perovskite Solid Solutions

The tetragonal lattice distortion plays a crucial role in the ferroelectric and piezoelectric properties of perovskites. For example, the best piezoelectric performance is found at the morphotropic phase boundary (MPB) of a solid solution between the tetragonal and the rhombohedral phases. Previous theoretical work has shown, that the strain-polarization coupling is vital for stabilizing the tetragonal phase relative to the rhombohedral one, and in enhancing the magnitude of polarization of the material.[7] Recently, several Bi-based materials with extremely large polarization and *c*/*a* values were synthesized.[8–10] These properties can be useful for applications such as birefringent optics and negative thermal expansion materials.[11]

To enable rational materials design, it is necessary to understand how changes in a material's composition change the relevant properties. Previously, we demonstrated that there is a simple universal relationship between the ferroelectric-paraelectric transition temperatures in $PbTO_3$-based solid solutions and their 0K polarization (P^2), and elucidated how the crystal chemical changes due to compositional variations affect P and therefore T_c.[12,13] Here, we study how crystal chemical properties of the material relate to its tetragonality.

Twenty-five different tetragonal ferroelectric materials were examined with DFT[2,14] calculations and Berry's phase calculations[15] to compute the polarization. The results of our calculations are presented in Table 1.

Table 1. DFT and experimental data for tetragonal $PbTiO_3$-derived ferroelectric perovskite solid solutions. *A*- and *B*-cation averaged displacements (D_A, D_B)and polarization (P), averaged over several different cation arrangements are obtained from DFT-relaxed structures. The *c*/*a*-1 and Curie temperature (T_c) data are from experimental literature[8,16]. In the table, ABO_3 - (1-x) PT means x ABO_3 - (1-*x*) PT. The transition temperature datum for $BiZn_{1/2}Ti_{1/2}O_3$ is omitted, since this compound decomposed before undergoing phase transition[9].

	c/*a*-1	*P*	T_c	D_B	D_A
PT	0.065	0.87	765	0.280	0.450
$PbZn_{1/3}Nb_{2/3}O_3$ - 0.25 PT	0.033	0.66	547	0.218	0.461
$PbZrO_3$ – 0.5 PT	0.023	0.76	659	0.165	0.440
$PbSc_{1/2}Nb_{1/2}O_3$ - 0.5 PT	0.020	0.50	560	0.142	0.296
$PbIn_{1/2}Nb_{1/2}O_3$ – 0.5 PT	0.028	0.45	623	0.129	0.255
$PbSc_{2/3}W_{1/3}O_3$ – 0.625 PT	0.020	0.61	517	0.176	0.350
$PbMg_{1/3}Nb_{2/3}O_3$ - 0.625 PT	0.044	0.66	583	0.201	0.387
$PbZn_{1/3}Nb_{2/3}O_3$ - 0.625 PT	0.048	0.74	643	0.241	0.424
$PbZrO_3$ – 0.67 PT	0.046	0.84	700	0.210	0.450
$PbSc_{1/2}Nb_{1/2}O_3$ - 0.75 PT	0.041	0.74	640	0.220	0.412
$PbIn_{1/2}Nb_{1/2}O_3$ - 0.75 PT	0.046	0.65	695	0.208	0.387
$BiZn_{1/2}Ti_{1/2}O_3$	0.220	1.34		0.489	0.903
$BiMg_{1/2}Ti_{1/2}O_3$ - 0.5 PT	0.047	0.88	733	0.200	0.515
$BiZn_{1/2}Ti_{1/2}O_3$ - 0.5 PT	0.120	1.17	1100	0.365	0.675
$BiMg_{1/2}Zr_{1/2}O_3$ - 0.75 PT	0.041	0.86	721	0.216	0.478
$BiMg_{1/2}Ti_{1/2}O_3$ - 0.75 PT	0.061	0.93	803	0.258	0.505

	c/*a*-1	*P*	T_c	D_B	D_A
$BiZn_{1/2}Zr_{1/2}O_3$ - 0.75 PT	0.064	0.93	740	0.263	0.508
$BiZn_{1/2}Ti_{1/2}O_3$ - 0.75 PT	0.088	1.04	875	0.319	0.550
$BiScO_3$ – 0.75 PT	0.040	0.84	768	0.220	0.488
$BiGaO_3$ – 0.75 PT	0.057	0.84	768	0.226	0.447
$BiInO_3$ - 0.75 PT	0.080	0.90	856	0.257	0.539
$BiAlO_3$ - 0.875 PT	0.050	0.84	758	0.243	0.473
$BiScO_3$ – 0.875 PT	0.053	0.85	780	0.257	0.442
$BiGaO_3$ – 0.875 PT	0.057	0.83	757	0.241	0.427
$BiInO_3$ - 0.875 PT	0.077	0.94	847	0.295	0.535

Inspection of the data in Table 1 shows that there is a general trend of higher *c*/*a* for materials with high *P* and large average *A*- and *B*-site off-center displacement magnitudes D_A^2 and D_B^2 as calculated from the ground-state structures obtained by our calculations. A closer examination of the correlation between tetragonality *c*/*a*-1 and *P*, D_A^2, and D_B^2 shows that tetragonality is controlled by D_B. For example, for *P* and D_A there is a general trend of higher *P* or D_A values corresponding to higher *c*/*a*-1 (Figure 1), but with large deviations from the trend at high *P* values. On the other hand, for D_B the correlation with *c*/*a*-1 is strong and all of the data points fall close to the fit. The quality of the fit is not improved when D_A^2, D_B^2 and the $D_A * D_B$ cross-terms are included; this proves that the dependence of *c*/*a*−1 on D_A is weak, and that the correlation between *c*/*a*-1 and D_A^2 is due to the coupling between the *A*- and *B*-site displacements.

The strong correlations observed in Figure 1 imply that for perovskite ferroelectrics there is a universal scaling of tetragonality with cation displacement and polarization. Tetragonality of different compositions can therefore be predicted from a modified Landau-Ginzburg-Devonshire (LGD) theory with the B-site displacement as the order parameter. The strain-displacement coupling contribution to the free energy can be described by:

$$G = -\gamma_A s D_A^2 - \gamma_B s D_B^2 - \gamma_{AB} s D_A D_B + \frac{1}{2} K s^2 \tag{2}$$

where *s* is the tetragonality *c*/*a*-1 and γ and *K* are the strain-displacement coupling and the elastic constants, respectively. Minimizing the free energy with respect to *s*, we get:

$$s = \left(\gamma_A D_A^2 + \gamma_B D_B^2 + \gamma_{AB} D_A D_B\right) / K. \tag{3}$$

The high quality of the fit to D_B data in Figure 1b means that γ_A and γ_{AB} are small compared to γ_B and can be neglected.

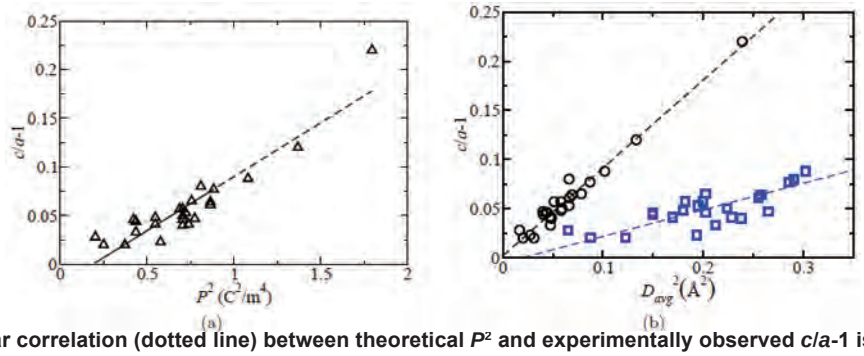


Figure 1. (a) A linear correlation (dotted line) between theoretical P^2 and experimentally observed *c*/*a*-1 is shown. (b) A linear correlation (dashed line) between the average cation displacement (D_A^2 in blue squares D_B^2 in black circles) and experimentally observed *c*/*a*-1 is shown. Better than D_A^2 and P^2, D_B^2 most closely correlates with *c*/*a*-1.

On the other hand, for the standard form of the LGD theory of ferroelectrics where P_{tot} is the order parameter, the minimum-energy strain s is given by:

$$s = \gamma \left(Z_A^* D_A + Z_B^* D_B \right)^2 / K \tag{4}$$

where we rewrite P_{tot} as the sum of the *A*- and *B*-site contributions given by the average product of cation displacement and its Born effective charge *Z**.

This form enforces a scaling of the coupling between *s* and off-center displacements to be given by the ratio of the *A*- and *B*-site Born effective charges. However, from fit 3 in Table 2 it is clear that the γ_B/γ_A ratio is ≈13, much larger than the ratio $\left(Z_B^* / Z_A^*\right)^2 = 3.4$ for $PbTiO_3$. This disagreement is more severe for BZT $\left(\left(Z_B^* / Z_A^*\right)^2 = 1.4\right)$. It is the overestimation of the *A*-site contribution to the total strain-polarization coupling that weakens the correlation and makes P_{tot} a less-accurate predictor of *c*/*a*.

Table 2. *R* is the notation for correlation coefficient. D_B^2 exhibits the best linear correlation with *s*. Fitting parameters *a*, *b*, and *d* are in the unit of Å^{-2}, *e* is in the unit of m^{4}C^{-2}, and *c* is unitless.

Fitting functions	*R*
$a * D_A^2 + c$ a = 0.2703, c = -0.0054	0.958
$b * D_B^2 + c$ b = 0.8938, c = 0.0020	0.984
$a * D_A^2 + b * D_B^2 + c$ a = 0.0553, b = 0.7236, c = -0.0003	0.985
$d * D_A * D_B + c$ d = 0.5016, c = -0.0025	0.983
$a * D_A^2 + b * D_B^2 + d * D_A * D_B + c$ a = 0.0000, b = 0.4796, d = 0.2346, c = -0.0003	0.985
$e * P^2 + c$ e = 0.1108, c = -0.0208	0.924

The difference in the strain dependence on the *A*- and *B*-site displacements is due to the geometry of the perovskite unit cell. For the *B*-cations, the *B*-O bonds lie along the Cartesian axes. Therefore, an off-center displacement along a Cartesian direction strongly affects the *B*-O bonds along that direction. Increased strain along a Cartesian direction increases the lattice constant and creates more space for the *B*-cation distortion. This makes the *B*-cation off-centering displacement more energetically favorable[7]. However, for the *A*-site, the *A*-O bonds lie along the (110) directions. Therefore, a tetragonal strain makes relatively small changes in the *A*-O bond length and strength. This makes the displacement-strain coupling for the *A*-cations weak.

Our understanding of how cation characteristics relate to the technologically important T_c and *c*/*a* properties of perovskite ferroelectrics provides a rational path for materials design. For example, since tetragonality scales with the *B*-site displacement, extreme tetragonality is not limited to Pb-or Bi-based materials but can be found even in compositions with a small charge on the *A*-site (e.g., Ag^{+1}-based ABO_3). For high T_c piezoelectric materials, an additive perovskite has to be chosen that will reduce *c*/*a* to create a morphotropic phase boundaries (MPB), while maintaining high T_c. To reduce *c*/*a*, the *B*-site displacement must be decreased. This; however, also tends to reduce T_c as it reduces the *B*-site contribution to polarization. Therefore, to keep T_c high, the *A*-site displacement must be increased by the additive perovskite. Such a combination of increased *A*-site displacement and decreased *B*-site displacement is in fact present for the two best high-temperature piezoelectric solid solutions available now – $BiScO_3PbTiO_3$[17] and $BiMg_{1/2}Ti_{1/2}O_3$ - $PbTiO_3$[16,18]. Upon addition of increasing among $PbTiO_3$, these exhibit an immediate reduction of *c*/*a* accompanied by an initial rise in T_c. This leads to an enhancement of T_c^{MPB} compared to the classic $Pb(Zr,Ti)O_3$ solid solution.

In conclusion, we have revealed a universal quadratic relationship between the average displacement of the cation *B*-site and *c*/*a* in ferroelectric perovskites. The composition-structure-property relations discovered in this and previous work provide guidance for systematic design of new ferroelectric and piezoelectric materials.

2.2 Pb-free Ferroelectrics Investigated with Density Functional Theory: $Sn(Al_{1/2}Nb_{1/2})O_3$ Perovskites

The search for more environmentally-friendly, lead-free piezoelectrics has led to the exploration of many new different perovskites with *A*-site ions such as Bi, Ag, and most recently, Sn. Investigations of Pb substitution by Sn are motivated by the chemical similarity between Pb^{2+} and Sn^{2+} cations. Both contain a stereochemically-active lone electron pair. Therefore, Sn atoms should exhibit large off-center displacements similar to the large Pb displacements in ferroelectric perovskites. Such displacements are favorable for a large polarization and a high ferroelectric-to-paraelectric transition temperature, T_c. The Sn analog of the classic $PbTiO_3$ material, $SnTiO_3$ has recently been studied with first-principles calculations and it was found that strong Sn off-centering is present, increasing both the c/a ratio and polarization relative to PT[19]. Experimentally, synthesis of $SnTiO_3$ is difficult, due to the hypothesized small ionic size of Sn which decreases the tolerance factor below the optimal value, t=1, that predicts perovskite phase stability. It should be noted that the radius of Sn^{2+} in a 12-coordinate cage has not yet been tabulated. Another hindrance towards experimental synthesis is the possibility of either partial reduction or disproportionation of Sn at elevated temperatures. Therefore, stable Sn-based perovskites should be favored for *B*-site cations with a small ionic radius (to help increase t closer to 1) that tend to prevent reduction of Sn at high temperatures. We therefore choose $Sn(Al_{1/2}Nb_{1/2})O_3$ (SAN) for our investigation of Sn-based ferroelectrics, and use DFT calculations to study the properties of the SAN solid solution.

To study the impact of *B*-cation ordering on SAN properties, we perform calculations for all possible Sn/Nb arrangements in a 40-atom supercell. Of the 10 supercells studied, our calculations indicate that only 6 (supercells 1–6) are found to be unique; the remaining four (supercells 7–10) relax back to one of the unique set of supercells. The results of these calculations are presented in Table 3. We find that for the majority of supercells, the difference between lattice constants *c* and *a* are small (1.01–1.03), unlike the large *c*/*a* found for $SnTiO_3$. All SAN supercells are ferroelectric with polarization magnitudes of 0.58–0.77 C/m^2 and a polarization vector close to either (110) or (111) directions.

Table 3. Data for the relaxed SAN supercells. Lattice constants, angles between axes and polarization values are given in Å, degrees and C/m^2, respectively. The energy differences relative to supercell 1 (ground state) are in eV per 40-atom supercell.

Supercell	*a*	*b*	*c*	α	β	γ	Px	Py	Pz	Ptot	Ediff
1	7.789	7.887	7.887	89.5	90.0	90.0	0	0.43	0.43	0.61	0.000
2	7.938	7.933	7.777	90.0	90.0	89.5	0.54	0.54	0	0.77	0.039
3	7.806	8.033	7.825	89.3	90.0	90.0	0	0.52	0.34	0.62	0.699
4	7.831	7.942	7.858	89.5	90.0	90.0	0.0	0.46	0.41	0.62	0.408
5	7.887	7.874	7.868	90.1	90.2	89.7	0.44	0.28	0.25	0.58	0.406
6	7.917	7.784	7.918	90.0	90.5	90.0	0.53	0	0.52	0.74	0.067
7	7.938	7.777	7.933	90.0	89.5	90.0	0.54	0	0.54	0.77	0.039
8	7.806	7.825	8.033	89.3	90.0	90.0	0	0.34	0.52	0.62	0.699
9	7.831	7.858	7.941	89.5	90.0	90.0	0	0.41	0.46	0.62	0.408
10	7.869	7.875	7.887	89.8	90.2	90.1	0.25	0.28	0.44	0.58	0.406

For supercell 5, we find a triclinic ground-state structure. This is similar to the low-symmetry structures found at the MPB of $Pb(Mg_{1/3}Nb_{2/3})O_3$-$PbTiO_3$ (PMN-PT), PZT and other solid solutions. This low-symmetry is symptomatic of a flat potential energy surface, favorable for polarization rotation, which leads to a high electromechanical response at the MPB. Thus, our DFT results for supercell 5 indicate that pure SAN is close to the MPB, and a slight doping of a tetragonal additive will result in an MPB composition with good electromechanical properties.

High values of piezoelectric coefficients are an indication of proximity to the MPB; to prove our hypothesis, we therefore calculate the piezoelectric coefficients of SAN supercells. To obtain piezoelectric coefficients one needs to employ a response function calculation, which is computationally expensive for *each* 40-atom supercell. In the first step, displacements are made to the well-converged ground-state structure, for each atom, in each direction to generate a force constant matrix. Using this information, a series of mixed derivatives (second-derivative of total energy with respect to finite electric field, displacements, strain, etc.) are then used to obtain the response of the system in the framework of perturbation theory. In the case of piezoelectric coefficients, the mixed second-derivative of interest is total energy with respect to electric field and strain.

We find that for all supercells, the diagonal piezoelectric coefficients are larger than e_{33}=2.7 C/m^2 found for $SnTiO_3$ by Uratani, et al.[19] and with the exception of supercells 1 and 4, are also larger than e_{33} coefficient of $PbTiO_3$. As expected, the similarity of supercell 5 to an MPB material leads to a particularly large e_{33}=11.8 C/m^2, essentially identical to the e_{33}=12.6 C/m^2 value of the monoclinic morphotropic phase boundary 50/50 PZT.[20] The off-diagonal elements of the e tensor are mostly comparable to those found in $PbTiO_3$. However, for supercells 1 and 5 we find large values (10.5 C/m^2) and e_{31} (7.3 C/m^2) for some off-diagonal coefficients. The large piezoelectric coefficients for some of the SAN Al/Nb arrangements support our hypothesis that SAN is close to the MPB. All values for piezoelectric tensors can be found in the Supplemental material of Reference 5.

In order to predict the location of the MPB in the SAN solid solution, in which a tetragonal additive is added, such as adding $PbTiO_3$ to $PbZrO_3$ to create PZT, we use a previously developed theoretical model.[21] If SAN is close to an MPB, then addition of tetragonal $ATiO_3$ would increase the electromechanical response. For a solid solution derived from PbTiO3 (PT), the PT content at the MPB $\left(x_{\mathrm{PT}}^{\mathrm{MPB}}\right)$ is given by:

$$x_{\mathrm{PT}}^{\mathrm{MPB}} = 1 - 1/\left(0.34 + 3.31R_{\mathrm{B,avg}} - 7.49D_{\mathrm{B,avg}}^{0}\right), \tag{5}$$

where $R_{\mathrm{B,avg}}$ is the average Shannon ionic radius for the *B*-site and $D_{\mathrm{B,avg}}^{0}$ is the average reference ionic displacement of the *B*-site.

Due to the similar behavior of Pb and Sn on the perovskite *A*-site, we expect that Equation 5 will be valid for Sn-based ferroelectrics as well. Comparison of Al and Nb displacements in our relaxed SAN supercells shows that the Al displacements are 0.06 Å smaller on average than the Nb displacements. Therefore, we assign the reference Al displacement as 0.11 Å, 0.06 Å smaller than the 0.17 Å reference displacement for Nb. Substituting the D_{B}^{0} values and the ionic radii (0.53 Å and 0.64 Å for Al and Nb, respectively) into Equation 5, we obtain $x_{\mathrm{PT}}^{\mathrm{MPB}} = 0.18$. Since the *c*/*a* ratio for $SnTiO_3$ is higher than that of PT, most likely less than 18% doping of $SnTiO_3$ will be needed to induce an MPB in a solid solution of SAN with $SnTiO_3$.

Local structure motifs in SAN are similar to those previously found for Pb-based perovskites.[12] The polarization is caused by the concerted displacements of Sn, Al, and Nb cations from the center of their oxygen cages. Similar to $PbTiO_3$, the displacements of the *A*-site Sn atoms are much larger than those for the B-cations. The Sn displacements are sensitive to the local *B*-cation arrangement. For example, inspection of the Sn-Al and Sn-Nb partial-pair distribution functions (PDFs) obtained from supercell 3 (Figure 2) shows that the Sn-Al peaks are located at significantly shorter distances than the Sn-Nb peak. This is because the Sn atoms are off-center toward the underbonded O atoms with two Al neighbors, and away from the overbonded O atoms with two Nb neighbors.

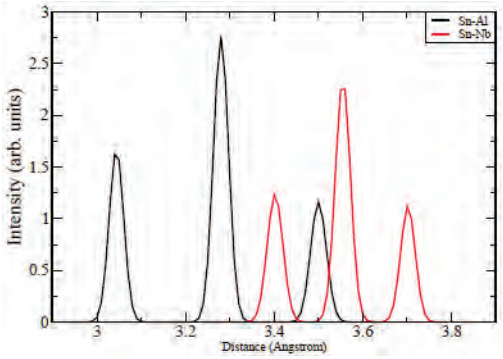


Figure 2. Sn-*B*-cation partial-pair distribution functions (PDF). Sn-Al and Sn-Nb partial PDFs are shown as solid black and dashed red lines respectively. On average, Sn-Nb distances are larger than Sn-Al distances, demonstrating that Sn displacements are directed away from Nb and toward Al.

To assign an ionic radius to the twelve-fold coordinated Sn^{+2} cation, we compare the Sn-O partial PDFs (Figure 3) with Pb-O and Bi-O partial PDFs from previous work[11,22]. For all three PDFs, there are three sets of peaks indicating a presence of strong, medium and weak *A*-O bonds created by the *A*-cation displacements away from the center of the O_{12} cage. Both the peaks for the strong and the weak Sn-O bonds are located at smaller distances than their Pb-O counterparts. This means that the ionic radius of 12-fold coordinated Sn^{2+} is smaller than that of 12-fold coordinate Pb^{2+} (r_{Pb}=1.49 Å). On the other hand, the Sn^{2+} ionic radius is larger than that of Bi^{3+} (r_{Bi}=1.36 Å). This is obvious from a comparison of the location of the

peak for the short Sn-O distances (2.3 Å) to the 2.1 Å location of the peak for the shortest Bi-O distances. The ionic size of Sn^{+2} is therefore ≈1.4 Å in the 12-fold coordination of the perovskite *A*-site, yielding *t*=1.00 for SAN solid solution and *t*=0.99 for $SnTiO_3$.

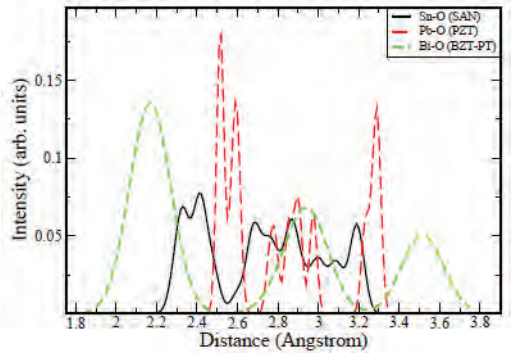


Figure 3. A comparison of Sn-O, Pb-O, and Bi-O partial PDFs. Sn-O shown in solid black line is obtained from current work, while Pb-O and Bi-O data from previous work[11,22] are shown as dashed short (red) and long (green) lines respectively. Sn-O distances are smaller than those of Pb-O in PZT, but larger than the Bi-O distances in BZT-PT.

In conclusion, we have investigated the ground-state structure of the Pb-free ferroelectric material $Sn(Al_{1/2}Nb_{1/2})O_3$. We show that the electromechanical properties of SAN are similar to those found in ferroelectrics in close proximity to an MPB, which can be reached with by mixing SAN with a small amount of tetragonal additive. This is favorable for the development of Sn-based piezoelectrics to replace the Pb-based materials used in current devices.

2.3 Density Functional Study of $PbTiO_3$ Nanocapacitors with Pt and Au Electrodes

Ferroelectric (FE) nano-capacitors have been under intense investigation in the last decade due to their potential use for nonvolatile (FE) random-access memory (FERAM) devices.[23–25] A crucial issue for FERAM miniaturization is the critical thickness at which ferroelectricity and polarization can no longer be sustained due to the presence of a strong depolarizing field. We investigate nano-capacitors consisting of $PbTiO_3$ ferroelectric with Pt and Au electrodes to gain a thorough understanding of the electrode-FE interfaces, and to elucidate the role of the interfacial chemical bonding and charge transfer in stabilizing the FE polar phase.

We use DFT calculations with the recently developed PBEsol functional[26] to study properties of several PTO-based nano-capacitor-like structures. As shown by Perdew and collaborators[27], PBEsol is more accurate for solids and surfaces, while the well-known PBE functional[27] gives better results for atoms and molecules.

To represent the capacitors, we used a supercell containing an alternating stack of PbO and TiO_2 layers sandwiched between metal electrodes. We study both TiO_2 and PbO terminations of $PbTiO_3$ and FE thicknesses of *n*=1…11 unit cells. For the TiO_2 termination, the metal atoms are located on top of the oxygen atoms and for the PbO termination they are on top of the *A*-cation and oxygen atoms.

The results for the nano-capacitors at the critical thickness are presented in Table 4. We find that the critical thickness is 24 Å for TiO_2 termination. We also present the data for PbO termination and 28 Å for interfacial polarization at the positive and negative surfaces, and the average polarization of the FE thin films. Our results show that the critical thickness of ferroelectricity is not only a function of composition of the oxide and metal, but also of the termination of the oxide film (TiO_2 vs. PbO). The FE-electrode interfaces are significantly different for the two terminations. For example, for Pt electrodes, the layer polarization magnitudes at the positive and negative surfaces are much smaller for the TiO_2 termination than for the PbO termination. Similarly, for the TiO_2 termination, Pt electrodes stabilize a larger fraction of bulk *P* than the Au electrodes. This is in contrast to the PbO termination for which our calculations find Au electrodes to be better.

Table 4. Computed properties of PTO-based nano-capacitors with different electrodes. N_c is the smallest number of unit cells for which the film has a FE ground state. The layer polarization at the negative (P_-) and positive (P_+) interfaces, and the average polarization (P_{av}) of the FE film, are all in units of the bulk polarization P_{bulk}. The layer polarization, given in units of the bulk polarization, is defined as the ratio of the rumplings of the cations from their oxygen cages in the thin film to those in the bulk. c/c_{bulk} measures the tetragonality of the FE film in units of the bulk value, and Δ (meV) is the well depth of the FE film per surface unit cell.

	N_c	P_-/P_{bulk}	P^+/P_{bulk}	P_{av}/P_{bulk}	c/c_{bulk}	Δ
TiO_2 term.						
Pt/PTO/Pt	7	0.50	0.62	0.62	0.94	7
Au/PTO/Au	7	0.77	0.86	0.80	0.96	16
PbO term.						
Pt/PTO/Pt	6	0.99	0.97	0.95	0.96	45
Au/PTO/Au	6	0.44	0.38	0.35	0.96	31

We use average polarization P of the nano-capacitors shown in Table 4 to estimate the effective screening length λ of the metallic electrodes. The depolarizing field is given by:

$$E_d = -\frac{2\lambda P}{\epsilon_0 \ell}, \quad (6)$$

where ℓ is the thickness of the FE film[28]. We obtain Ed by macroscopic-averaging[29] of the electrostatic potential. We find λ=0.045 and 0.028 Å for Pt and Au with TiO_2 termination, and 0.007 and 0.034 Å with PbO termination, respectively. These effective screening lengths are not intrinsic properties of the electrode, but of the electrode-oxide interface, and are much smaller than the standard screening lengths of metals (≈0.5 Å for Pt and ≈0.8 Å for Au).

Analysis of structural data shows that the deformations from the parent oxide and metal geometry are localized at the metal-oxide interface. For the oxide, the layer polarization for the 1–2 layers close to the interfaces is either enhanced or diminished compared to the polarization in the layers far from the electrode (Figure 4). Similar effect is seen in the buckling of the electrode layers (Table 5) for the PbO terminations. For the TiO_2 termination there is no buckling in the electrode layers because the metal atoms are in a symmetric chemical environment. The magnitude of the buckling for the Pt and Au electrodes is correlated with the strength of the metal-oxide bonds at the interface. These are stronger for Pt-O (bond enthalpy of 347 kJ/mole) than for Au-O (bond enthalpy of 222 kJ/mol).

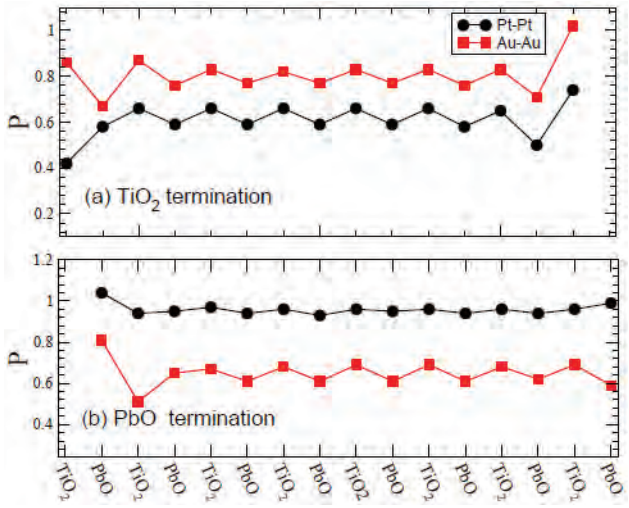


Figure 4. The layer polarization profile for the TiO_2 and PbO nano-capacitors with Pt and Au electrodes. We measure the polarization with respect to the bulk polarization of PTO. All films have seven unit cells.

Table 5. Buckling (in Å) of the electrode layers that are in registry with the PbO layers. We report these values for the PE nano-capacitors (β_0), and for the negative (β_-) and positive (β_+) surfaces of the FE nano-capacitors.

	PbO termination			TiO_2 termination		
	β_0	β_-	β_+	β_0	β_-	β_+
PtPt						
1st	−0.347	−0.314	−0.023	0.012	0.007	0.017
2nd	0.019	0.013	0.021	−0.005	0.008	−0.015
3rd	0.000	−0.011				
AuAu						
1st	−0.195	−0.247	0.074	−0.062	−0.015	0.068
2nd	−0.041	−0.064	0.006	−0.013	−0.008	−0.005
3rd	−0.000	−0.043				

Our results show that that the chemical bonding at the interface significantly changes the electronic surface of the interface oxide layers. Figure 5 shows layer-projected density-of-states (LPDOS) for the TiO2- and PbO-terminated FE nano-capacitors with Pt and Au electrodes. In these films, the polarization is directed upwards, i.e., the lower interfacial layer is at the negative surface and the upper one is at the positive surface. Examination of the LPDOS shows two main effects of the metal-oxide bonding. First, the interfacial oxide layers are metallic with a large density-of-states at the Fermi level. This is due to the charge transfer taking place at the interface, and is also the case for the paraelectric structure. Second, for the ferroelectric films, the conduction band on the positive surface is shifted downward and the valence band at the negative surface is shifted upwards. Similar to the structural changes, these effects are localized at the interface as can be seen from the LPDOS of the oxide layers in the bulk of the film which are similar to those of bulk PTO.

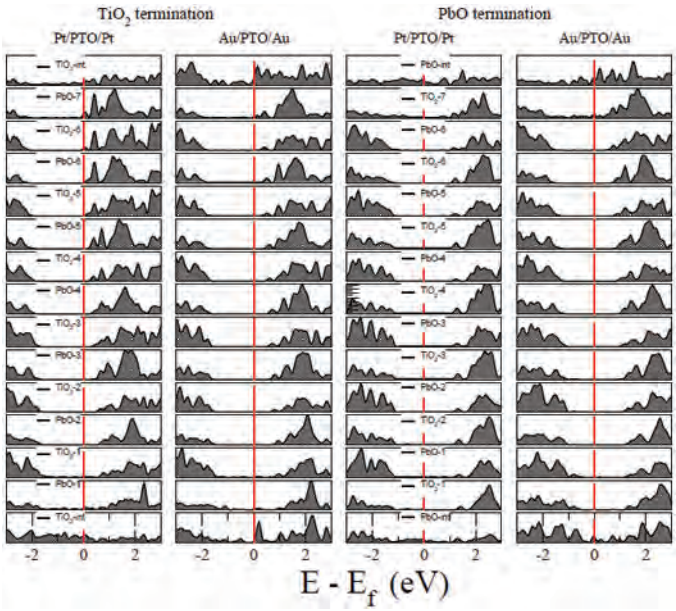


Figure 5. The layer-projected density of states for the TiO_2 and PbO terminated FE nano-capacitors with Pt and Au electrodes. The polarization is directed upwards i.e., the lower TiO_2 interfacial layer is at the negative surface of the capacitor and the upper one is at the positive surface. All the nano-capacitorshave 15 oxide layers (seven unit cells).

In conclusion, we presented an *ab initio* investigation of ferroelectricity in single-domain $PbTiO_3$ (PTO) based nano-capacitors. The ferroelectric properties of the thin films depend on the details of the electrodeperovskite interface, with difference in screening properties between the TiO_2 and PbO terminations of the oxide. The critical thickness is 24–28 Å.

3. Significance to DoD

Perovskite oxides form the basis for a range of military-relevant technologies such as dielectric ceramics for capacitor applications, ferroelectric thin-film non-volatile memories, piezoelectric materials for SONAR and actuators, and electro-optic materials for data storage and displays. A microscopic understanding of perovskite oxides properties is critical for the goal of designing new materials. Once the relationships between the atomic composition, structure and materials properties are understood, new materials that improve upon existing technology can be designed. The US Navy would reap a considerable military advantage from more robust ferroelectric memory impervious to electromagnetic pulse attacks, higher-performance SONAR for detection of submarines, and photovoltaic materials for energy generation. Our investigations of composition-structure-property relationships in perovskite ferroelectrics provide a rational path for design of new piezoelectric SONAR materials with enhanced sensitivity and robustness. Our study of lead-free tin-based piezoelectrics shows that this class of materials is likely to exhibit high piezoelectric performance similar to that of the current lead-based perovskites. Our examination of ferroelectric nano-capcitors shows the importance of the oxide termination for the behavior of the metal-oxide interface that controls polarization stability and sets the limit to the miniaturization of nonvolatile memories and other ferroelectric-based devices.

Computational Technology Area

Computational Material Science

Acknowledgements

This work was supported by the Office of Naval Research under grants N00014-09-1-0157 and N00014-09-1-0455, by the Air Force Office of Scientific Research Grant No. FA9550-10-1-0248, the MEREST grant, and the DMR05-20020. This work was also supported by the US Department of Energy Office of Basic Energy Sciences, under grant number DE-FG0207ER15920 and DE-FG02-07ER46431. Computational support was provided by a Challenge grant from the High Performance Computing Modernization Program of the US Department of Defense.

References

1. Goldschmidt, V.M., *Die Naturwissenschaften*, 21, 477, 1926.

2. Hohenberg, P. and W. Kohn, *Phys. Rev.*, 136, B864, 1964.

3. Kohn, W. and L.J. Sham, *Phys. Rev.*, 140, A1133, 1965.

4. Bennett, J.W., I. Grinberg, P.K. Davies, and A.M. Rappe, *Phys. Rev. B*, 83, 144112, 2011.

5. Qi, T., I. Grinberg, and A.M.Rappe, *Phys. Rev. B*, 82, 134113, 2010.

6. Al-Saii, W.A. and A M. Rappe, *Phys. Rev. B*, 82, 155304, 2010.

7. Cohen, R.E., *Nature*, 358, 136, 1992.

8. Suchomel, M.R. and P.K. Davies, *Appl. Phys. Lett.*, 86, 262905, 2005.

9. Suchomel, M.R. A.M. Fogg, M. Allix, H. Niu, J.B. Claridge, and M.J. Rosseinsky, *Chem. Mater.*, 18, 4987, 2006.

10. Stein, D.M., M.R. Suchomel, and P.K. Davies, *Appl. Phys. Lett.*, 89, 132907, 2006.

11. Grinberg, I., M.R. Suchomel, W. Dmowski, S.E. Mason, H. Wu, P.K. Davies, and A.M. Rappe, *Phys. Rev. Lett.*, 98, 107601, 2007.

12. Grinberg, I. and A.M. Rappe, *Phys. Rev. B*, 70, 220101, ISSN 0163-1829, URL http://dx.doi.org/10. 1103/PhysRevB.70.220101, 2004.

13. Grinberg, I. and A.M. Rappe, *Phys. Rev. Lett.*, 98, 037603, 2007.

14. Perdew, J.P. and A. Zunger, *Phys. Rev. B*, 23, 5048, 1981.

15. Vanderbilt, D. and R.D. King-Smith, *Phys. Rev. B*, 48, 4442, 1993.

16. Suchomel, M.R. and P.K. Davies, *J. Appl. Phys.*, 96, 4405, 2004.

17. Eitel, R.E., S.J. Zhang, T.R. Shrout, C.A. Randall, and I. Levin, *J. Appl. Phys.*, 96, 2828, 2004.

18. Randall, C.A., R. Eitel, B. Jones, T.R. Shrout, D.I. Woodward, and I.M. Reaney, *J. Appl. Phys.*, 95, 3633, 2004.

19. Uratani, Y., T. Shishidou, and T. Oguchi, *Jap. J. Appl. Phys.*, 47, 7735, 2008.

20. Wu, Z. and H. Krakauer, *Phys. Rev. B*, 68, 014112, 2003.

21. Grinberg, I., M.R. Suchomel, PK. Davies, and A.M. Rappe, *J. Appl. Phys.*, 98, 094111, 2005.

22. Grinberg, I., V.R. Cooper, and A.M. Rappe, *Phys. Rev. B*, 69, 144118, 2004.

23. Junquera, J. and P. Ghosez, *Nature*, 422, 506, 2003.

24. Sai, N., A.M. Kolpak, and A.M. Rappe, *Phys. Rev. B Rapid Comm.*, 72, 020101, R, 2005.

25. Stengel, M., D. Vanderbilt, and N. Spaldin, *Nat. Mater.*, 8, 392, 2009.

26. Perdew, J., A. Ruzsinszky, C. G/'abor, O. Vydrov, G. Scuseria, L. Constantin, X. Zhou, and K. Burke, *Phys. Rev. Lett.*, 100, 136406, 2008.

27. Perdew, J.P., K. Burke, and M. Ernzerhof, *Phys. Rev. Lett.*, 77, 3865, 1996.

28. Mehta, R.R., B.D. Silverman, and J.T. Jacobs, *J. Appl. Phys.*, 44, 3379, 1973.

29. Baldereschi, A., S. Baroni, and R. Resta, *Phys. Rev. Lett.*, 61, 734, 1988.

HPCMP UGC 2011

5. Signal/Image Processing (SIP) and Sensors; Electronics, Networking, and Systems/C4ISR (ENS) and Testing

Computational Electromagnetics and Acoustics (CEA)

Radar Signature Prediction for Sensing-Through-the-Wall by Xpatch and AFDTD – Part III

Traian Dogaru, Anders Sullivan, Calvin Le, and Chris Kenyon
US Army Research Laboratory (ARL), Adelphi, MD
{traian.dogaru, anders.sulivan, calvin.le, christopher.kenyon}@us.army.mil

Abstract

This paper presents computer simulations of Sensing-Through-the-Wall (STTW) radar for building imaging applications. In this study, we model a two-story building that includes many construction details, as well as furniture, kitchen appliances, bathroom installations and several humans. We use two radar signature prediction codes for this analysis, namely Xpatch and Army finite-difference time-domain (AFDTD). Based on the electromagnetic simulation results, we create synthetic aperture radar (SAR) images of the buildings for various depression look-angles, emulating both ground-based and airborne radar data collection geometries. We discuss both phenomenological aspects of this radar imaging application and the accuracy of the electromagnetic modeling codes. Our results demonstrate good accuracy for both simulation methods and point out interesting effects specific to building imaging by STTW radar.

1. Introduction

In the modern battlefield environment, the requirement to counter asymmetric threats has resulted in a greatly increased likelihood that military ground forces will need to engage and neutralize hostile elements in an urban setting, where the enemy may be hiding in one of a number of buildings. In this scenario, the soldier's situational awareness is often severely reduced, resulting in heightened personal danger and increased risk of collateral damage. Therefore, many defense agencies have demonstrated keen interest in developing Sensing-Through-the-Wall (STTW) technologies. Among these, the low-frequency (typically 4GHz or below), ultra-wideband (UWB) microwave radar demonstrated great potential for building mapping, as well as remotely detecting and tracking enemy personnel who are inside the building or behind a wall. Major STTW radar-related development programs are currently sponsored by the Army, Navy and Defense Advanced Research Projects Agency (DARPA) (Farwell, et al., 2008).

At the US Army Research Laboratory (ARL), we have been actively involved in several major Department of Defense (DoD) programs, developing computer models and algorithms in order to predict STTW system performance. Our models have included electromagnetic (EM) scattering from various realistic building structures, human meshes, weapons and furniture. Typically, we would use the EM modeling results obtained over a large number of frequencies and aspect angles in high-resolution synthetic aperture radar (SAR) image formation. In 2009, the High Performance Computing Modernization Program (HPCMP) awarded us a 3-year Challenge Project entitled "Radar Signature Prediction for Sensing-through-the-Wall", with the main goal of validating our radar signature computing codes for STTW applications. During the first year of the project, we considered a radar imaging scenario involving a moderately complex one-story building structure containing humans and furniture (Dogaru, et al., 2009). In the second year, we investigated building structures of increased complexity, including pipes, air ducts, and a realistic roof (Dogaru, et al., 2010a). In the third and final year of the project, we apply our radar modeling and imaging algorithms to a two-story building containing a large number of construction details.

2. Computational Methods and Resources

In this paper, we create SAR images of building structures based on computer-simulated radar signature data. In order to create the SAR image of a target, we first need to evaluate its radar response over a wide range of frequencies and aspect angles. The EM scattering models are performed via two computational electromagnetic (CEM) codes: AFDTD and Xpatch. AFDTD (Dogaru, 2010) was developed at ARL, sponsored by HPCMP under the Virtual Electromagnetic Design (VED) Portfolio. It implements a fully-parallelized version of the finite-difference time-domain (FDTD) algorithm

(Taflove and Hagness, 2000). Its major disadvantage consists of requiring very large computational resources, both in terms of CPU-time and memory. Xpatch (SAIC, 2009) was developed by Demaco/SAIC under a grant from the US Air Force and is based on ray-tracing combined with physical optics (PO). Unlike FDTD, which is a full-wave (exact) technique, these are approximate EM field calculation methods and their validity is normally restricted to the high-frequency regimes. A major purpose of this study is to validate the Xpatch computations for the STTW radar application, which involves relatively low operational frequencies (1–4GHz). This is achieved by comparing the SAR images obtained from Xpatch radar signature data with those based on AFDTD simulations.

In a previous paper, (Dogaru, et al., 2009), we performed this type of analysis for a single-floor building structure that included humans and furniture objects. In the current investigation, we keep that structure as the bottom floor of a two-story building and add another entire floor on top of it. The upper-floor has an increased level of complexity, including two fully-equipped bathrooms, a kitchen with appliances, furnished bedrooms and a dining area. The bathrooms contain a shower stall, a bathtub, sinks with vanity cabinets and mirrors, as well as toilets. Connecting the two floors we add an interior staircase. Between the two floors we place a ceiling/floor structure supported by beams in both directions. The overall building ceiling is flat, made of a 3-inch (7.5-cm) thick concrete slab. The overall building dimensions are 396 in by 276 in by 189 in (10 m by 7 m by 4.9 m). We left out several construction details such as pipes, air ducts and electrical wires, since those were studied separately in another paper (Dogaru, et al., 2010a).

The materials are typical for this kind of construction. The exterior walls are made of 8-inch brick and considered uniform. Some walls are equipped with glass windows and wooden doors. The interior walls are made of drywall material. The furniture is made of wood and fabric, the kitchen appliances are mostly metallic, while the bathroom installations contain porcelain, wood, glass and metal. The staircase is entirely made of wood. There are six humans inside the building: four at the ground-floor and two at the upper-floor. We also added an overall dielectric ground plane, which is important for the airborne radar simulations. The dielectric properties of the materials involved in this mesh were listed in Dogaru and Le (2010) and Le, et al. (2009). The general building layout is shown in Figure 1.

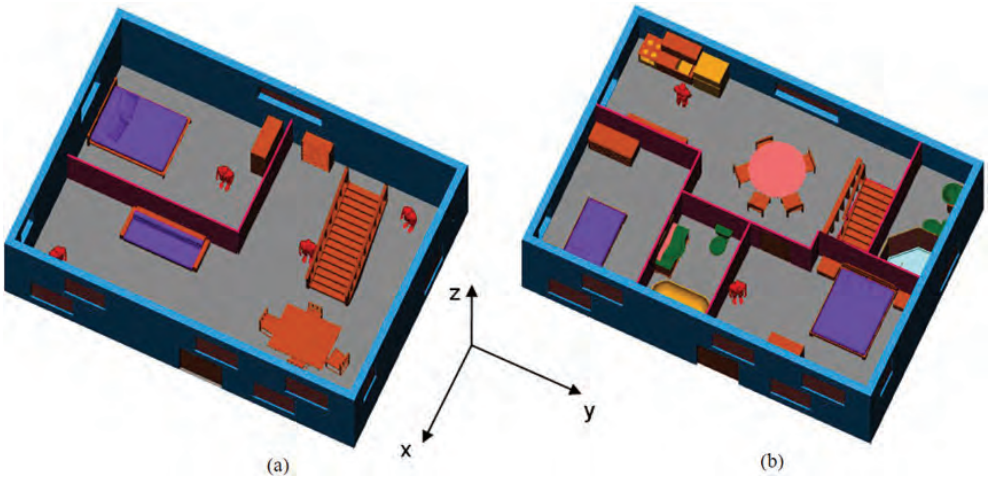


Figure 1. View of the computational mesh for the two-story building showing: (a) the lower-floor and (b) the upper-floor

The AFDTD computational meshes are made of small cubic cells with 5 mm sides. The cell size is dictated by the need to keep the numerical dispersion errors under control at the frequencies of interest (Taflove and Hagness, 2000). The overall grid involved just over 3 billion cells, making it the largest radar signature modeling problem that our research group at ARL has attempted to date. The wideband calculation of the radar response at one aspect angle was performed on 128 cores, requiring a total memory of about 180GB (or 1.4GB per core). In order to obtain most of the SAR images shown in Section III, the far-field backscattering radar return was calculated between 0.5 and 4.5GHz with a 6.67MHz frequency step, and between -30° and 30° in azimuth, with a 0.5° angular step. In some cases, we imaged the building from two sides by adding the contributions from apertures of -30° to 30° and 60° to 120° in azimuth, measured relative to the *x* axis in Figure 1. The SAR images were created assuming a spotlight configuration and the polar format algorithm (Carrara, et al., 1995). The images are always represented in the slant plane. The image resolution is approximately 3 inches (7.5-cm) in down-range and 6 inches (15-cm) in cross-range.

The simulations presented in this paper were performed primarily at the ARL DoD Supercomputing Resource Center (DSRC). Specifically, we used the Harold SGI Altix ICE 8200 system. Obtaining one of the SAR images shown in Section 3, based on AFDTD data, for vertical-vertical (V-V) polarization typically took about 77,000 CPU-hours. Xpatch is a much more efficient code and uses only a fraction of the computational resources required by AFDTD (typically 1/100 to 1/1,000). In this study, Xpatch used about 1,300 CPU-hours per image. All the graphics in this report were done with Pioneer RCS. The pre- and post-processing were performed on Dell workstations running Windows Vista.

Our final goal is to simulate the radar response of the building over a two-dimensional (2D) aperture (both in elevation and azimuth) that would allow us to create full three-dimensional (3D) SAR images of the building and its interior (Ahmad, et al., 2008). The simulations will be performed over a -30° to 30° aperture in azimuth (with 0.5° angular step) and 15° to 55° depression (with 1° angular step). The total CPU time necessary for the AFDTD simulations to cover these angular ranges is approximately 3.2 million hours. At the time of this paper's submission, these simulations were still ongoing. Therefore, we present here only 2D images obtained from apertures placed at a constant depression angle.

3. Numerical Results

The modeling results are presented as 2D SAR images in the slant plane, with the synthetic aperture placed on the left side of the page in most cases (unless specified otherwise). The pseudo-color maps represent pixel intensity in dB. All dimensions are in inches. Except for the first set of images, we generally avoid adding the mesh overlays since those would make the images very difficult to interpret.

We first present the case of a ground-based radar system, which corresponds to θ=0° (the radar moves in the horizontal plane). The SAR images of the lower floor were presented in the first part of this three-part study (Dogaru, et al., 2009). In Figure 2, we show the SAR images of the upper-floor based on AFDTD modeling data (this time we overlay the mesh contours), for apertures centered on two different sides of the building (in the page, the aperture is always centered on the left side). As we noticed in previous investigations, the walls (exterior and interior) perpendicular to the line of sight show prominently in the SAR images. We also notice other bright objects in the images, such as the side walls of the kitchen appliances, certain cabinets and the shower stall. Another interesting effect is the multiple reverberations created between some interior walls and the front wall, which appear as replicas of the interior walls at equal distances behind the front wall. The effect is an additional source of clutter in the image as we go deeper in down-range. We infer that, for a complex building, the radar images can only convey reliable information for a limited range behind the front wall (typically up to the second wall encountered by the radar waves). Combining images created from apertures placed on different sides of the building could allow more details to be picked up by the SAR system.

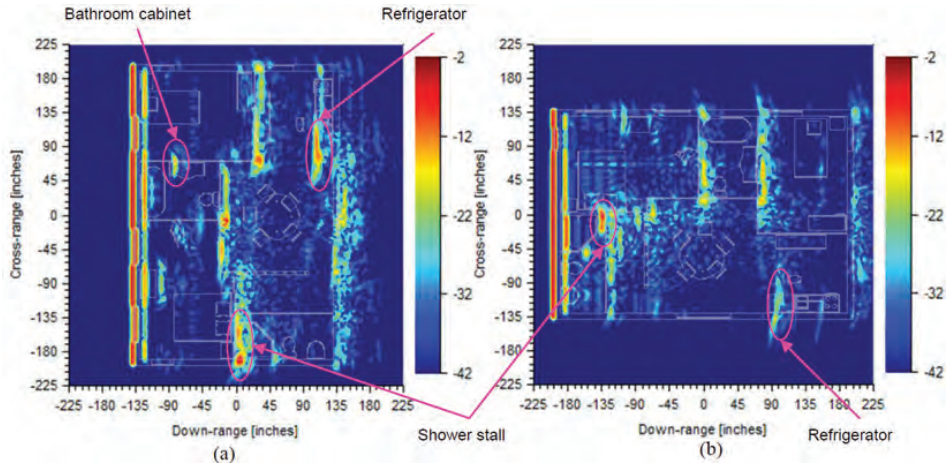


Figure 2. SAR images of the upper-floor, for θ=0° and V-V polarization, based on AFDTD data, with the aperture centered (a) along the x axis and (b) along the y axis

In Figure 3, we present the SAR images of the entire building for θ=0° and the same apertures as in the previous figure. This time we do not overlay the mesh contours, but we show both the AFDTD and Xpatch-based results. In general, the images obtained by the two methods are very similar. We notice that Xpatch overestimates the radar return from certain features (typically inner-corners), and therefore the Xpatch images look more "cluttered". One immediate observation about these images is that they do not allow us to discriminate targets (or even interior walls) placed on the lower-floor from

those at the upper-floor. This is a major limitation of a 2D SAR imaging system that does not allow resolving targets in the vertical dimension. To overcome this, the radar data must be collected over apertures at different heights (or depression angles), effectively creating a 2D aperture. Also important is the observation that the human targets (assumed here as stationary) are very difficult to distinguish from other features of the images. Most likely, a STTW radar system would have to rely on motion for detection and tracking of human targets.

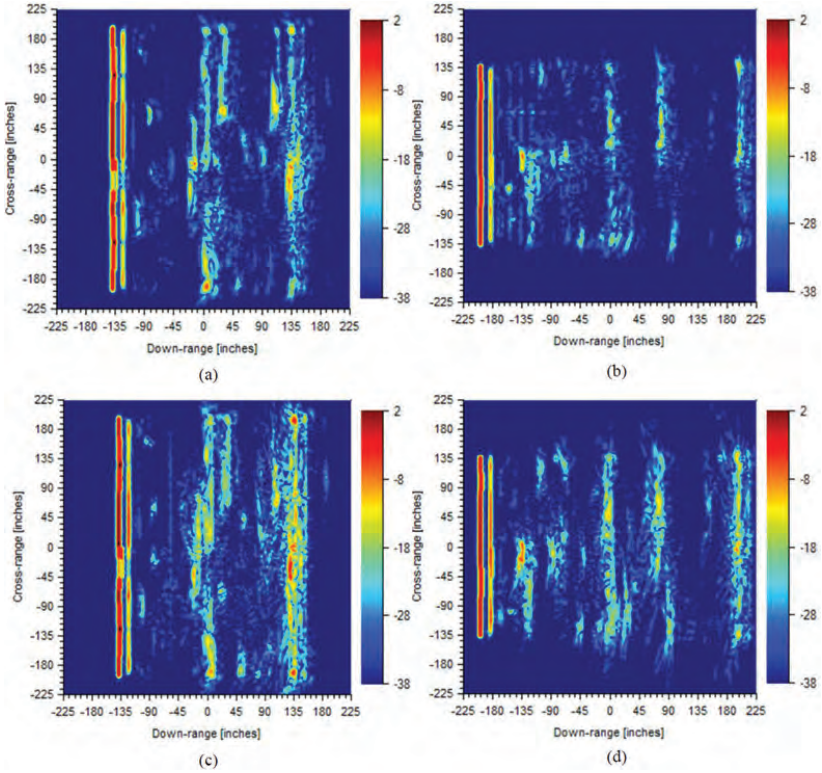


Figure 3. SAR images of the entire building, for θ=0° and V-V polarization, showing: (a) aperture along x axis, AFDTD data; (b) H aperture along y axis, AFDTD data; (c) aperture along x axis, Xpath data; and (d) aperture along y axis, Xpatch data

Figure 4 displays the SAR images of the two-story building for an airborne scenario for θ=20° (only one side of the building). As shown in previous work (Dogaru and Le, 2010), there are significant differences between the V-V and horizontal-horizontal (H-H) polarizations, due to the proximity of the Brewster angle to this direction of incidence. Thus, the ground bounce is very strong in H-H polarization and the bottom edge of the front wall (the brightest feature in the images) appears much more intense than for the V-V polarization (notice the difference in dB scales between the two polarizations). If the radar has a fixed dynamic range, the V-V mode allows more details to be picked up in the SAR image than the H-H mode at this incidence angle.

In Figure 5 we show the SAR images obtained at θ=40° based on AFDTD data only, for both polarizations. We again notice differences between the two polarizations in the front wall brightness, although these are not as large as for the previous case (about 10 dB for θ=40° versus 20 dB for θ=20°). However, more striking in these images is the "shrinking" of the building footprint in the cross-range dimension. While the down-range shrinking of the target dimensions in the slant plane is always present in an SAR imaging system, the cross-range shrinking is specific to the spotlight mode (Soumekh, 2001). This is a consequence of the imaging algorithm working directly in the slant plane, without taking into account the depression angle of this plane. To be more exact, the 2D imaging algorithm considers that the azimuth angle in the slant plane (used for image formation) is the same as the azimuth angle in the ground plane (as used by the radar signature simulation code). This assumption is generally incorrect, except when the slant and ground planes coincide θ=0°). The net effect of this artifact is that the image appears as "shrunk" by a cos θ factor in the cross-range dimension. This problem can easily be corrected by stretching the cross-range of each pixel from the middle of the image by a factor of 1/cos θ. We demonstrate this in Figure 6, where we also overlay the mesh contours in order to confirm our assumptions. Notice that the same effect occurs for the θ=20° case, although in that situation the image distortion is very small, since the "shrinkage" factor is about 0.94.

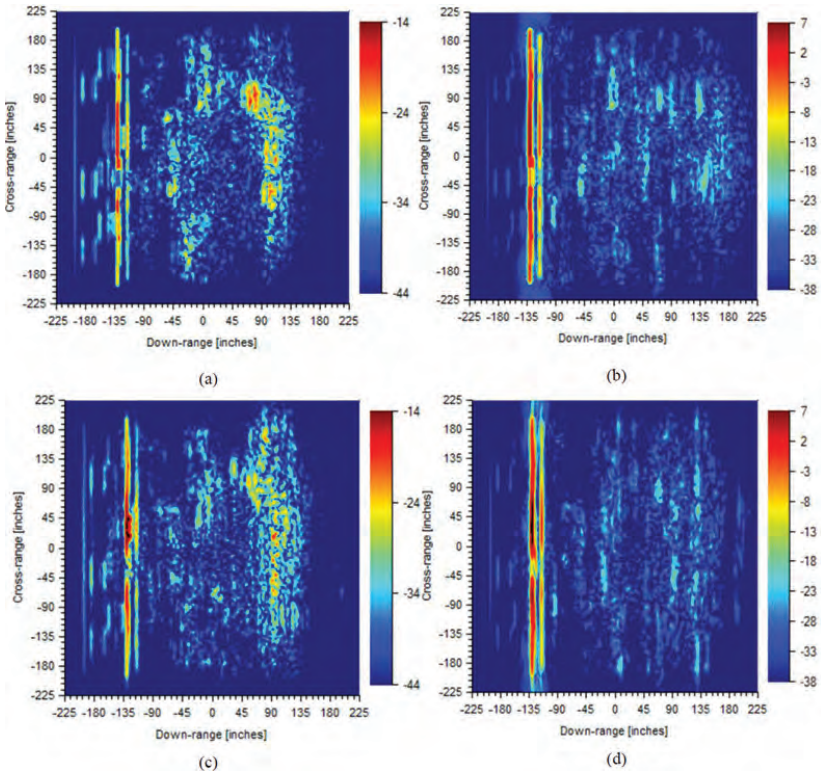


Figure 4. SAR images of the entire building, for θ=20° with the aperture centered along the x axis, showing: (a) V-V image based on AFDTD data; (b) H-H image based on AFDTD data; (c) V-V image based on Xpatch data; and (d) H-H image based on Xpatch data

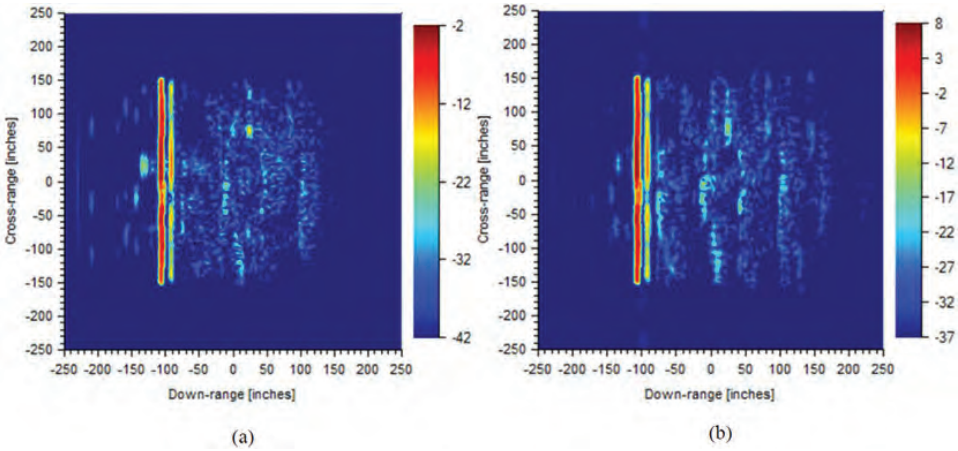


Figure 5. SAR images of the entire building based on AFDTD data, for θ=40° with the aperture centered along the x axis, showing: (a) V-V polarization; (b) H-H polarization. Note: no cross-range correction was applied to these images.

In order to quantify the differences between the AFDTD and Xpatch prediction results, we compared the pixel intensity in the areas highlighted in Figure 7. (To be more exact, we are comparing the maximum-intensity pixels within the approximate circled areas in Figure 7.) The features include exterior walls (#1), the bathtub (#2), two human targets (#3–4), two corners (#5–6) and one interior wall (#7). The results are presented in Table 1, for the three depression angles considered in this section (θ=0°, 20°, and 40°), for V-V and H-H polarization. Notice that we did not consider the H-H polarization for θ=0°, since we know from previous studies that there are no significant differences between the two polarizations for that scenario (Dogaru, et al., 2009). In the airborne scenarios (θ=20° and 40°) we were not able to discriminate the pixels representing certain features, so we left those entries in the table blank. This underscores again the difficulty to identify image features that are placed sufficiently far from the front wall, especially at steeper elevation angles.

The data in Table 1 demonstrate a good match between the images obtained by the two methods, particularly for areas of walls and the human bodies. However, significant discrepancies are noticed for the corner areas (#5–6), consistent with our previous experience with Xpatch (Dogaru, et al., 2009). The explanation is that Xpatch does not accurately account for corner diffraction at oblique incidence angles, since the PO technique cannot handle this phenomenon by itself. The bathtub (#2) also shows significant differences at θ=20° and 40°, where the scattering mechanisms from this cavity-like object become very complex, and therefore difficult to handle by Xpatch.

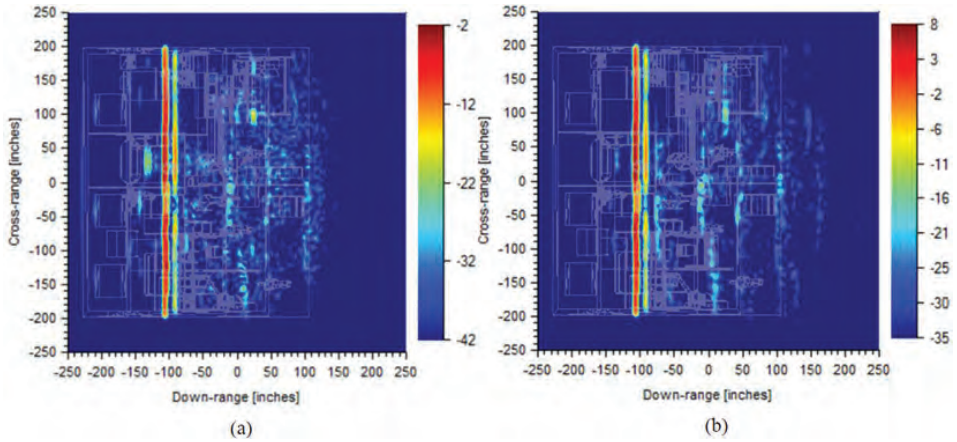


Figure 6. Cross-range-stretched SAR images of the entire building based on AFDTD data, for θ=40° with the aperture centered along the x axis, showing: (a) V-V polarization; (b) H-H polarization

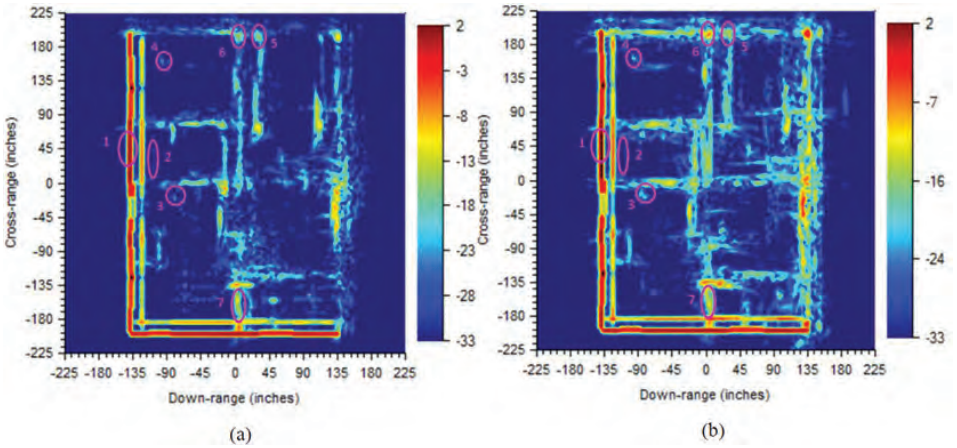


Figure 7. Combined SAR images of the entire building with apertures on two sides, for θ=0° and V-V polarization, showing: (a) image based on AFDTD data; (b) image based on Xpatch data. The areas selected for code comparison are highlighted by pink circles.

We also compare the SAR images obtained via AFDTD and Xpatch by calculating the cross-correlation coefficient between the 2D pixel maps. We perform this at the bottom of Table 1. The results demonstrate excellent correlation between the images obtained by AFDTD and Xpatch, except for the case of θ=20°, V-V polarization. It appears that in that case, Xpatch overestimates the ground-front-wall corner reflection, which has the largest contribution to the overall image intensity. In general, the good image correlation indicates that both codes correctly localize the position of the important features in the images (especially the walls). This is a key issue for an algorithm that tries to extract features, such as walls, wall edges or corners, based on the SAR images of the building.

4. Conclusion

In this paper, we considered a two-story building that includes many complex details and created 2D SAR images based on computer models of radar returns. We applied two CEM codes for the electromagnetic scattering calculations, AFDTD (an exact solver) and Xpatch (an approximate solver). The purpose was two-fold: to understand phenomenological aspects

of imaging large and complex targets at various depression angles, and to compare the accuracy of the modeling software. Although AFDTD provides the correct solution, it is known to require very large computational resources. On the other hand, Xpatch is a much more efficient field calculation tool, but its accuracy must be evaluated before applying it to this class of problems.

Table 1. Comparison between AFDTD and Xpatch imaging results in the areas marked in Figure 7, for the three depression angles considered in Section III (data in dB). The bottom row contains the cross-correlation coefficient between the images created by the two methods.

Scenario	θ=0°		θ=20°				θ=40°			
Polarization	V-V		V-V		H-H		V-V		H-H	
Code	FDTD	Xpatch	FDTD	Xpatch	FDTD	Xpatch	FDTD	Xpatch	FDTD	Xpatch
1	2.2	2.1	-16.8	-13.9	5.5	7.0	-3.4	-2.6	7.1	7.5
2	-30.9	-31.5	-25.4	-18.4	—	—	-21.4	-14.9	-22.5	-17.7
3	-23.6	-20.7	—	—	—	—	—	—	—	—
4	-23.5	-19.5	-37.1	-34.6	-25.6	-26.4	—	—	—	—
5	-15.6	-13.9	—	—	-26.9	-19.0	—	—	—	—
6	-14.4	-6.6	—	—	—	—	—	—	—	—
7	-11.9	-11.3	—	—	—	—	—	—	—	—
Correlation	0.955		0.814		0.960		0.983		0.988	

From a phenomenological standpoint, increasing the building complexity from one to two floors creates significant confusion into a 2D SAR image. The main reason is that all image features appear now projected onto one plane (the slant plane) and they cannot be resolved in the height dimension. This issue becomes more pronounced in the airborne case, as discussed in Section 3. In order to resolve individual features or targets in the vertical direction, at least two 2D images taken at different depression angles are needed. Better still, a 2D aperture would allow the creation of a full 3D image. This is the remaining goal within this research project, which we plan to achieve and report on once we obtain the entire set of simulation data.

The 2D images presented in Section 3 also emphasize the fact that stationary human targets would be very difficult to detect directly in the SAR images, when the building includes a large amount of clutter represented by interior walls, furniture, appliances, etc. Most likely, a STTW radar system would take advantage of human motion and apply moving-target indicator (MTI) techniques in order to detect and track personnel inside a building. However, the radar imaging techniques mentioned in this paper could be very useful in creating the floor layout of a multi-story building as a first step in a surveillance scenario. Once the layout is obtained, detecting and tracking a moving human becomes a much better determined problem, since knowledge of the wall positions can be used to mitigate the multipath propagation problem (Dogaru, et al., 2010b).

An interesting effect for airborne radar imaging is the target "shrinking" in cross-range, which becomes more visible at steep depression angles (such as θ=40°). As we demonstrated, "stretching" the cross-range image dimensions by $1/\cos\theta$ seems to realign the targets with the ground truth. We conjecture that this kind of "stretching" will also be necessary to process data from a 2D aperture into a 3D image.

Finally, by comparing the images obtained from the two modeling codes we can state that both AFDTD and Xpatch correctly locate the major features in the SAR image of the building. As expected, there are some differences in the pixel intensity of certain image areas. While many important features match within 2–3 dB between the two methods, fairly large differences are noticed at room corners, which is a well-known problem with Xpatch. In general, the findings in this paper on the code accuracy are consistent with those of our previous work (Dogaru, et al., 2009, 2010a). Consequently, we can state that Xpatch offers a fast and reasonably accurate tool for predicting the radar return in complex STTW scenarios. We think that it could prove a particularly efficient forward-modeling tool in building layout extraction applications, where the location of a specific scattering feature in an image is more important than its pixel intensity.

Acknowledgements

The DoD HPC Modernization Program supported this project by supplying supercomputer time under the Computing Challenge Project C3Z. The simulations were performed at the US Army Research Laboratory (Aberdeen Proving Ground) and the US Air Force Research Laboratory (Wright-Patterson Air Force Base) DoD Supercomputing Resource Centers.

References

Ahmad, F., Y. Zhang, and M.G. Amin, "Three-dimensional wideband beamforming for imaging through a single-wall", *IEEE Geoscience and Remote Sensing Letters*, vol. 5, pp. 176–179, 2008.

Carrara, W., R. Goodman, and R. Majewski, *Spotlight Synthetic Aperture Radar – Signal Processing Algorithms*, Artech House, Boston, MA 1995.

Dogaru, T., "AFDTD user's manual", *ARL Technical Report, ARL-TR-5145*, Adelphi, MD, 2010.

Dogaru, T., A. Sullivan, C. Le, and C. Kenyon, "Radar signature prediction for sensing-through-the-wall by Xpatch and AFDTD – Part I", *Proc. 2009 HPC Users Group Conference*, San Diego, CA, 2009.

Dogaru, T., A. Sullivan, C. Le, and C. Kenyon, "Radar signature prediction for sensing-through-the-wall by Xpatch and AFDTD- Part II", *Proc. 2010 HPC Users Group Conference*, Schaumburg, IL, 2010a.

Dogaru, T. and C. Le, "Through-the-wall radar simulations for complex room imaging", *ARL Technical Report, ARL-TR-5205*, Adelphi, MD, 2010.

Dogaru, T., C. Le, and L. Nguyen, "Synthetic aperture radar images of a simple room based on computer models", *ARL Technical Report, ARL-TR-5193*, Adelphi, MD, 2010b.

Farwel,l M., J. Ross, R. Luttrell, D. Cohen, W. Chin, and T. Dogaru, "Sense-through-the-wall system development and design considerations", *Journal of the Franklin Institute*, vol. 345, pp. 570–591, 2008.

Le, C., T. Dogaru, L. Nguyen, and M. Ressler, "Ultra-wideband imaging of building interior: measurements and predictions", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, pp. 1409–1420, 2009.

SAIC, http://www.saic.com/products/software/xpatch, accessed 2009.

Soumekh, M., *Synthetic Aperture Radar Signal Processing*, John Wiley & Sons, 1999.

Taflove, A. and S.C. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain*, Artech House, Boston, MA, 2000.

Rapid Antenna Design Enabler (RADE)

Steve Wong
High Performance Technologies, Inc. (HPTi), Wright-Patterson AFB, OH
swong@hpti.com

Charles Macon
US Air Force Research Laboratory (AFRL), Wright-Patterson AFB, OH
charles.macon@wpafb.af.mil

Abstract

Rapid Antenna Design Enabler (RADE) is a plug-in tool developed by the Department of Defense (DoD) High Performance Computing Modernization Program's User Productivity Enhancement, Technology Transfer and Training (PETTT) Initiative for use with the CUBIT meshing and CAD software from Sandia National Laboratories. RADE can be used to rapidly generate geometry models (and meshes) compatible with major DoD computational electromagnetics codes. Geometric primitives that are useful for electromagnetic analysis are parameterized and captured as templates in RADE. By simply providing parameters to these templates via their graphical user interfaces, geometry models will be generated automatically in CUBIT. These geometric primitives can then be used as building blocks to construct even more complex models for computational electromagnetics (CEM) analysis. RADE is useful for performing design or parametric studies. It can automate and reduce the labor required to manually create geometry models in an iterative design process. The RADE template library currently consists of over 40 geometry templates for antenna, radio-frequency (RF), and microwave applications. An overview of RADE will be presented in this paper.

1. Introduction

Geometry and mesh generations are usually the most labor-intensive part of any computational electromagnetics (CEM) analysis. In computer-aided design (CAD) of electromagnetic hardware such as antennas, radio-frequency (RF) and microwave circuits; multiple design iterations requiring CAD models, and meshes for CEM analysis are typical. Therefore, reducing the time needed for an analyst to create these electromagnetic CAD models can lead to enhanced productivity and reduced design-cycle time.

One approach to reducing the amount of manual labor involved in creating CAD models has been previously described in Reference 1. In this approach, a complete antenna CAD model can be generated by simply entering the desired parametric values, and following a clearly laid-out, interactive step-by-step process (the "antenna creation wizard"). This scheme was implemented for four classes of common antennas in a software tool. While this approach works well for generating CAD models that have been completely parameterized, strict adherence to the antenna creation wizard has its drawbacks. Namely, the wizard does not allow creation of designs that are outside the pre-programmed scope. The rigidity of the wizard also makes custom code modification difficult.

Building upon this previous work, a more flexible approach to antenna and electromagnetic CAD model creation has been implemented. This new software tool, named Rapid Antenna Design Enabler (RADE), permits a much greater degree of flexibility for users to generate the desired models. More importantly, new geometric model templates can be easily added to the new tool, and user customization can be quickly accomplished.

2. Technical Approach

RADE is a plug-in tool that works with the CUBIT geometry and mesh generation software from Sandia National Laboratories (version 12 and above)[2]. RADE has a graphical user interface which allows the user to interact with the meshing and geometry engine of CUBIT to rapidly create CAD models and meshes for electromagnetic analysis. The basic philosophy of RADE is to build complex CAD models from simple parameterized geometric primitives frequently encountered in CEM. This approach to model construction is not unlike the use of primitives such as sphere, cylinder, brick, etc., that are commonly found in many CAD software for building complex structures through Boolean operations. However, instead of just simple geometric shapes like sphere and cylinder, RADE uses more elaborate models of antennas,

RF and microwave components as geometric primitives. These electromagnetic (EM) geometric primitives can be combined through Boolean-type operations to build even more complex models for CEM analysis. This idea is illustrated in Figure 1, where two EM primitives: metallization pattern and a rectangular substrate are combined to produce a model of a frequency selective surface.

EM geometric primitives are parameterized and captured as templates in RADE. Users provide the parametric values that define the primitives' geometry via the template graphical user interface. By entering these parametric values, primitives are created automatically in RADE within the CUBIT program environment. From the users' perspective, the details of creating potentially complicated primitives are hidden, and the whole process consists of simply typing in the parametric values and a few button clicks. The resultant EM geometric primitives can then be treated like any other CAD geometries in CUBIT. They can be manipulated freely and further operated on by the users to produce the desired final CAD models.

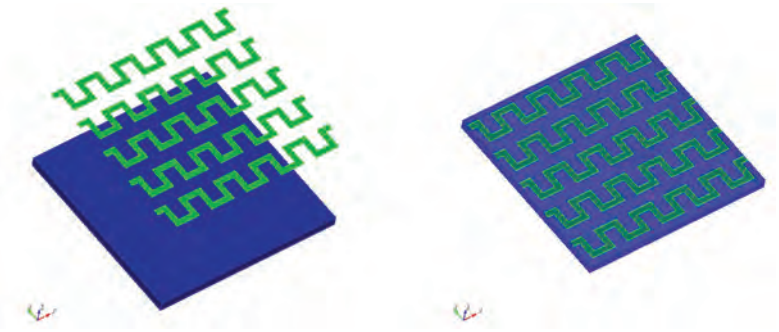


Figure 1. Electromagnetic geometric primitives are combined in RADE to rapidly construct CAD models

3. RADE Template Library

RADE is invoked inside the CUBIT program by loading it into the CUBIT/CLARO framework. After RADE is launched, a main window will appear (Figure 2), which consists of drop-down menus for users to select the various EM geometric primitive templates.

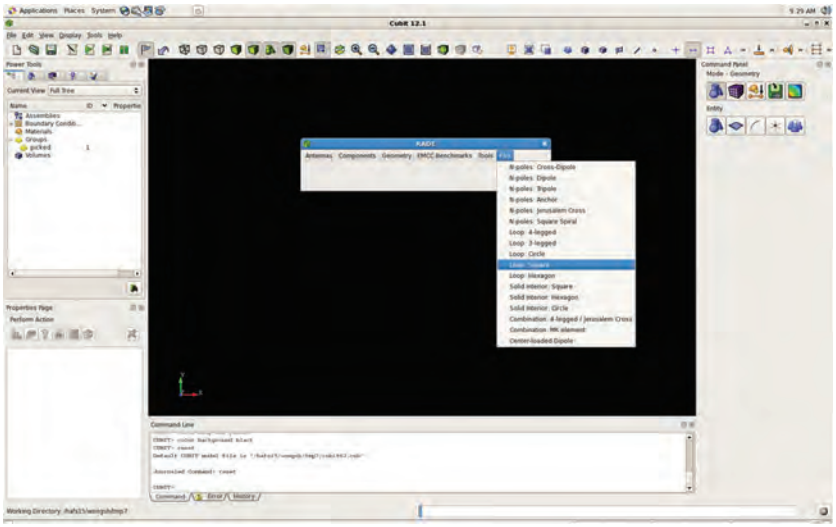


Figure 2. RADE main window (center) consists of drop-down menus of templates for electromagnetic geometric primitives

RADE is, in essence, a library of custom templates to create specialized geometric primitives that are useful for CEM analysis. Although it is possible for a template to create a complete parameterized CAD model suitable for CEM analysis in one step, such a template may only have limited general applications, as it may be difficult to add or subtract features from the pre-programmed model. Instead, RADE templates are meant to create primitives of lower-level building blocks,

from which a complete CEM CAD model can be assembled. The primitives are, in principle, independent of each other, and hence new templates can be added to the RADE library without affecting any of the existing ones. This modular approach will facilitate the customization of RADE.

The graphical user interface of a typical RADE template is shown in Figure 3 (the sub-window on the left). It consists of a set of input fields to enter the parametric values defining the geometry, and a set of buttons to create or delete the resultant model. As illustrated in Figure 3, which is a template to create various tapering profiles for tapered slot antennas, a wide-range of geometries can be realized from a single template with minimal effort from the user. Currently, there are 43 templates in the RADE template library. The templates are grouped into five categories: *Antennas*, *Components*, *Geometry*, *Tools*, and *Frequency Selective Surface (FSS)*. Details of these templates can be found in the *RADE User Manual* provided with the RADE code distribution. A brief description and some examples are given below.

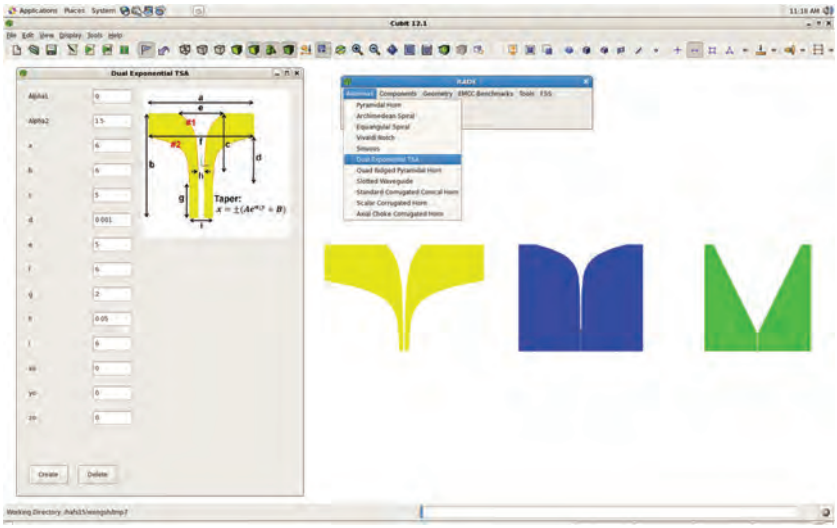


Figure 3. Tapered slot antenna (TSA) template

Typical EM geometric primitives under the *Antenna* template category are metallization patterns for different classes of planar antennas such as spiral, Vivaldi, TSA etc. Examples from the TSA template are shown in Figure 3. In additional to these two-dimensional (2D) planar geometries, three-dimensional (3D) antenna structures such as ridged-horn, corrugated horns are also available. An example of a conical corrugated horn primitive generated from a template is shown in Figure 4a.

Component templates are used to produce geometric primitives of electrical and circuit components such as coaxial feed, micro-strip, dielectric substrate, etc. These primitives are used in conjunction with primitives from other templates (such as the *Antenna* templates) to construct complete CAD models for CEM analysis. An example of the micro-strip template is shown in Figure 4b. With this particular template, users can rapidly layout the geometries of micro-strip transmission lines with the specified widths and mitered corners.

Geometry and *Tool* templates are support utilities in RADE for manipulating or creating new arbitrary geometries and meshes. These templates are essentially pre-programmed Python and CUBIT command scripts to simplify certain laborious or repetitive tasks in CUBIT. One example is the *Draw* template shown in Figure 4c. By using this template, users can quickly construct line segments or curves (for example, exponential curves used in many planar tapered slot antenna designs), or to create arbitrary shape planar surfaces. This general-purpose template is useful for creating CAD models of planar structures.

Lastly, *FSS* templates consist of commonly used FSS elements[3]. With these templates, users can create either a model of a single FSS element for unit-cell analysis or groups of elements tessellated in various configurations. An example is shown in Figure 4d, where models of a single "anchor" element, as well as an array of anchor elements, can be created by simply entering the geometric parameters and clicking a button in the template graphical user interface (GUI).

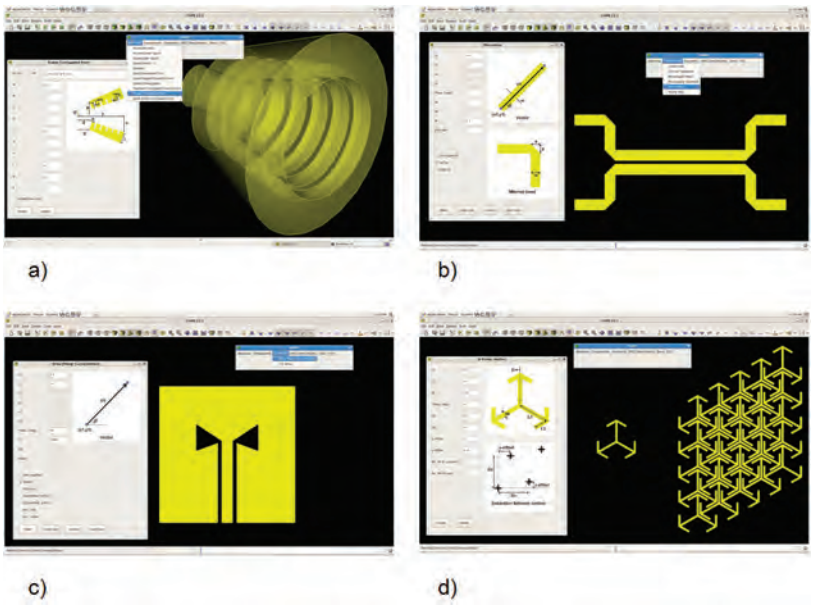


Figure 4. Examples from the RADE template library; a) Corrugated horn antenna; b) Microstrip; c) "Draw" template; and d) "Anchor" FSS element

4. RADE Implementation

RADE is written with a mixture of languages including Python, CUBIT commands, and CUBIT's native scripting language APREPRO[2]. Python is used to perform numerical calculations on the geometry for the templates. Python is also used to interface with CUBIT, via PyQt4, to create graphical user interfaces such as the RADE main window menus and template dialog GUIs. All graphical user interfaces are displayed within the CUBIT/CLARO graphical environment. APREPRO scripts are used to program the execution of CUBIT commands for geometry and mesh generation. Since RADE is written in scripting languages, it is completely text-based. It is interpreted by CUBIT; therefore, no code compilation is necessary. RADE will run on any computing platform that runs CUBIT.

There are two approaches in which RADE can be utilized to speed-up the CAD model generation process. Different methodologies will require different levels of complexity for the templates. On the one hand, having lots of simple low-level EM geometric primitives are desirable, as they provide great flexibility in constructing complex models. But at the other extreme, it may be more preferable to have a single template to generate a complete CAD model with one click of the button, which may be the case for parametric design study where the entire design can be captured into a single template. In either situation, the design of novel electromagnetic devices will inevitably require novel geometric templates that will not be available from the RADE library, no matter how comprehensive the library may be. Therefore, it is important that RADE be able to permit easy incorporation of new customized templates. To satisfy this requirement, RADE has adopted a modular approach, and is structured such that new templates can be added easily. A schematic of the organization of RADE is shown in Figure 5. It should be noted that although templates can be reused by other templates, they are fundamentally independent entities. Therefore, users can supply their own templates to the library regardless of whether the templates are for simple low-level EM geometric primitives or complete CAD models for parametric studies. The process of creating new templates is relatively simple, and is explained in detail in the RADE documentation *RADE Programmer's Guide* that is distributed along with the RADE source codes.

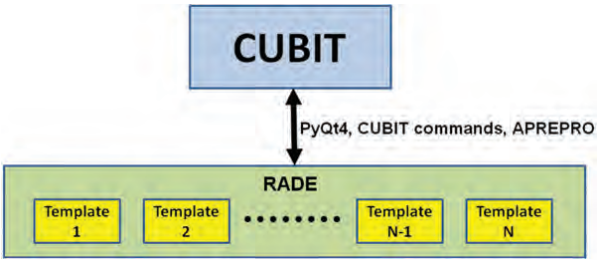


Figure 5. RADE template library interacts with CUBIT through Python and CUBIT commands

The RADE code distribution also includes an Exodus-to-ACAD file format conversion program called *exodus2acad*. This program converts the native Exodus mesh file format of CUBIT to the ACAD facet format commonly used in many CEM codes. The converter is a standalone program and is not a prerequisite to execute RADE. However, the converter can be invoked by templates in the RADE library to automate the export of surface mesh generated in CUBIT to the facet format used in CEM codes.

5. Vivaldi Antenna Example

In this section, an example is given to illustrate the steps of building a Vivaldi antenna CAD model in RADE. For this example, the model is composed of four primitives: the Vivaldi antenna metallization, substrate, micro-strip feed, and radial stub. The construction process is illustrated in Figure 6. The RADE template for each of these primitives is invoked inside CUBIT, and parametric values for the desired geometry entered into the input dialogs. The order in which these templates are invoked is not important in this example, as the primitives are completely independent. After the parametric data are entered, one or several button clicks in each template will automatically generate, respectively, the geometry of the metallization pattern of the antenna (Figure 6a), substrate (Figure 6b), layout of the micro-strip feed line (Figure 6c), and radial stub termination for the feed line (Figure 6d). These primitives are then maneuvered into the proper positions by manually issuing standard CUBIT commands via the CUBIT command line user interface (translation commands in this case). With the primitives in the correct locations, CUBIT operations such as “imprint” and “merge” will then assemble them into a final 3D model of a Vivaldi antenna (Figure 6e). The resulting CAD model can be exported to other meshing software, or meshed directly in CUBIT itself. From the users’ point-of-view, the entire modeling process basically requires only the input of parametric values and the manual translation/orientation of the primitives into their proper positions. The details of creating the more complex and time-consuming geometries, such as the metallization and micro-strip transmission lines in the above example, are performed automatically in RADE. This can result in significant time savings for the user.

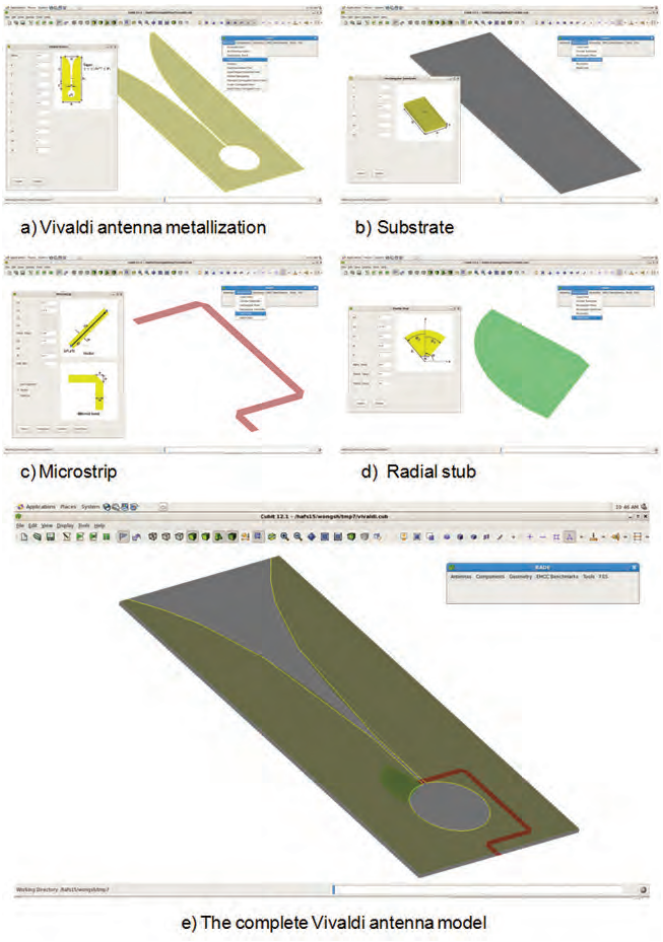


Figure 6. The steps in RADE to rapidly construct a Vivaldi antenna model

6. Conclusion

A productivity enhancement tool for creating antennas and general electromagnetic CAD models for CEM analysis has been developed. Complex CAD models are constructed from simple building blocks of pre-defined parameterized EM geometric primitives. The creation of these EM geometric primitives is performed automatically in CUBIT and the details are hidden from the users. With this modular approach, significant time savings in manual construction of the models can be realized. The RADE template library currently consists of 43 templates for various antenna, RF and microwave component primitives. User customization is an important aspect of RADE. Therefore, the RADE library code structure has been organized in such a way that new templates for the users' specific applications can be easily incorporated. RADE can benefit DoD CEM analysts by reducing the labor involved in creating CAD models for simulations. It can also serve as a useful tool for generating models for parametric design studies.

Acknowledgments

The authors gratefully acknowledge funding support from the DoD HPCMP's PETTT initiative for this work through PETTT pre-planned effort: PP-CEA-KY02-002-P3.

References

1. Macon, C., D. Henn, S. Wong, C. Kung, and J. Jin, "Rapid Antenna Model Creation", *Proceedings of the 2009 DoD High Performance Computing Modernization Program Users Group Conference*, June 2009.

2. http://cubit.sandia.gov.

3. Munk, B.A., *Frequency Selective Surfaces -Theory and Design*, Wiley-Interscience, New York, 2000.

HPCMP UGC 2011

6. Software and Hardware Infrastructure

A Fault Detecting Diagnostic Tool for Python-driven Multi-language Scientific Code

Sameer Shende and Allen D. Malony
ParaTools, Inc., Eugene, OR
{sameer, malony}@paratools.com

Andrew Wissink
US Army Aeroflightdynamics Directorate, Ames Research Center, Moffett Field, CA
andrew.m.wissink@us.army.mil

Abstract

Traditional debugging tools are limited in their ability to relate fault exceptions, such as numerical or memory errors, to their sources in a multi-language application, such as those being developed under the CREATE program. A new tool for multi-language scientific computing applications driven by Python is presented that reports useful diagnostic and performance information when the program experiences some form of anomalous operation. Built upon the TAU performance system, it operates across all code layers and generates a diagnostic file containing information about memory usage, call-stack, and input/output (I/O) gathered during execution. The tool is useful to both software developers as well as users who experience software issues, as it provides a way to exchange execution information between the user and the development team without requiring disclosure of potentially-sensitive application data. Furthermore, it provides support for retaining performance measurements and other execution information that would otherwise be lost.

1. Introduction

Modern software tools typically utilize multi-language components, linking software written in different languages. These can be native languages (e.g., C, C++, FORTRAN), scripting or interpreted languages (e.g., Python, Java), or other high-level languages (e.g., domain-specific languages). It is common practice for a program written in one language to use libraries written with different languages. Scientific software often uses scripting languages such as Python to drive high-level operations, while applying lower-level libraries written in C, C++, and FORTRAN for the compute-intensive numerics. Use of this multi-language paradigm facilitates modular software and facilitates more straightforward exchange of software between different development groups. However, it introduces a number of complexities in terms of debugging and memory management.

In general, tools for multi-language application execution, such as performance measurement and debugging tools, have the challenge of interpreting information gathered in relation to the language context where the information is observed. For parallel performance tools, the problems are ones of associating performance effects and behaviors to where they are manifested in the program. In the TAU Performance System®[2], we have developed sophisticated techniques for instrumentation, measurement, and analysis of multi-language parallel programs that can track execution, input/output (I/O), memory, and communication operations and attribute performance data across language levels[14]. We have demonstrated the use of these techniques for Helios[1], a high-fidelity multi-disciplinary software framework for rotorcraft analysis[1], which uses different components implemented with different languages, all orchestrated by a high-level Python script.

For multi-language debugging tools, the problems of determining execution context in relation to program parts, and the language features used to create them are non-trivial. At present, if a software code experiences a run-time issue, the program generates a core file which, while useful for debugging, contains limited information about the status of the execution, the memory profile, the process call-stack, and the system resources in use. Software developers need to understand the nature of the exception, where it occurred, how long the program executed, and the routines invoked prior to the error. This requires a full-view of the execution state that might come from a multi-language program to gain a

[1]Helios is being developed by the HPC Institute for Advanced Rotorcraft Modeling and Simulation (HI-ARMS)[1,2] as part of the CREATE-Air Vehicles (AV) program.

better insight into layered workings of the application. This also extends to the importance of retaining other run-time program information that might be useful in understanding what led up to the error. For example, the ability to save parallel performance measurements after a long-running Helios execution encounters an exception could be extremely valuable.

There are other issues that complicate multi-language debugging matters. Exchanging large core files with the developers is cumbersome, especially when working with parallel applications that use dynamic shared objects (DSOs). Furthermore, applications failing on restricted computer systems can't be debugged off-site with present techniques by developers who do not have the privileges either to access the machine or the program input. This is particularly important for the large amount of classified or proprietary work that takes place within the Department of Defense (DoD).

The goal of our work is to create a diagnostic tool for run-time faults that replaces the standard core file output with more effective and concise information gathered during execution and at the time of the errant behavior, including data about the computational performance, memory usage, call-stack, and I/O operation. The diagnostic file would be self-contained and interchangeable between machines in the TAU standard profile format, so a user that experienced an execution problem could send the diagnostic file to the development team for further analysis to determine the source of the problem. This format will be displayed in a human readable form and will be open. In this way, the new tool provides a means of exchanging diagnostic information between DoD software developers and users.

The next section presents the design approach and discusses the main issues that need to be resolved. Section 3 describes the development of a preliminary prototype for multi-language applications based on a Python-driver model. We have applied this prototype to a parallel benchmark to gain early evaluation of the techniques. It reports on our findings from initial experiments. Section 4 describes future work with the Helios application. Related work is presented in Section 5. We conclude the paper with a discussion of the next development and evaluation steps.

2. Design Approach

When the application experiences a run-time error such as a segmentation violation (e.g., caused by an access through a null pointer), execution of an illegal instruction, or floating-point exception (e.g., division by zero), it is critical to diagnose the problem with respect to the executing context. Merely reporting the text output of the execution is rarely sufficient to fix the problem. The developers typically need to understand the nature of the exception, where it occurred, how long the program executed, and the routines invoked prior to the error. This requires a full-view of the execution-state that might come from a multi-language program to gain a better insight into the program's layered workings. In addition, system information such as the operating system kernel version, the extent of heap memory utilization, number of cores, and other application specific parameters is important in fully-documenting the error.

The key to solving the problem of observing execution-state is in capturing (unwinding) the call-stack at the location of the error. The diagnostic tool integrates a call-stack capture module in TAU to accomplish this. TAU unwinds the calling stack of each thread of execution and records it in the profile file format with context-specific information, such as the calling routine name, file name, and where available, the source line number for each frame in the program's calling stack. The tool operates at run-time, automatically interposing with the application to generate diagnostics. This involves extracting addresses and mapping addresses to useful program information before the information is lost and the program terminates.

However, a tool to recover state information from an errant parallel program must consider the cases where a subset (perhaps just one) of the threads experiences a failure. Here, it is necessary to inform the other processes to gracefully terminate, capturing relevant diagnostic information as they do so.

2.1 Signals

The diagnostic tool must gain control of the program once a failure has occurred. To do so, a handler to intercept signals generated by the OS when some form of anomalous operation takes place must be registered. We assume that all failures of concern will generate an error signal and that those signals will be made available to the tool signal handler. Registering for signals is accomplished by TAU at the beginning of the program.

When a signal occurs, the diagnostic tool gains control through the handler and is able to interrogate the calling stack of the halted thread of execution. What about the other threads of execution? Consider the case of a process with multiple threads. Signals thrown for any thread within a process are caught by the process, which in turn halts its other threads. Then the calling stacks of those threads can be observed.

Now consider the case of a parallel program with multiple processes using Message Passing Interface (MPI). When the diagnostic tool gains control of the violating process, the other processes must somehow be made to stop. If a broken

pipe occurs between the processes because the processes have halted, as will be the case with an MPI program, the OS with throw a *SIGPIPE* signal. MPI implementations typically record their own signal handlers and do not pass the signal to any tool's handler. To operate with MPI programs, the diagnostic tool must re-register the signal handler after a call to the *MPI_Init* or *MPI_Init_thread* routine inside an MPI interposition library. The MPI standard provides a name-shifted *PMPI* interface[2] that allows tools to transparently intercept and override the native MPI implementation calls made by the application during linking. Internally, TAU's MPI wrapper library invokes timing and signal handling calls and calls the name-shifted PMPI interface that provides an alternate entry point to the same MPI routines. This interception of MPI calls can take place either during linking or at run-time by pre-loading TAU's shared object. When the MPI program executes and reaches an error condition, it triggers the signal handler to report the error by traversing and recording the program call-stack.

2.2 Call-Stack

Each thread of execution that encounters an error will dump its performance data annotated with the call-stack back-trace. There are several ways to get a back-trace. Debuggers such as *gdb*[17] can generate a back-trace at a breakpoint. The *GLIBC* library[18] provides a back-trace application programming interface (API) that allows the program to examine the call-stack comprising of a set of tuples of addresses and mangled routine names. TAU uses the *backtrace* API and de-mangles the C++ routine names using the C++ *ABI* de-mangler. Other sources of de-mangling the routine name include the GNU *binutils* de-mangler available as a library or the *c++filt* tool.

2.3 Address Translation

The program counter (PC) in the stack frame identifies where in the program the fault occurred. However, translating the binary addresses to source location information (such as function name, executable or library name and line number) is non-trivial. The address returned by the back-trace API typically refers to the address of the next instruction in the frame. Thus, to get an accurate source line number, we need to query the address of the previous instruction in the given frame. To query the address, TAU uses the BFD library and parses the symbol table to access sections of the executable. Within each section, it examines the nearest line to a known address. This symbol information may be missing from executables that are stripped or not compiled with the `-g` option. Compiling an application with the `-g` compiler flag does not preclude the use of optimization flags. External tools such as *addr2line* or *eu-addr2line* may also be used. To improve the accuracy of address translation in cases where complete symbol information is not accessible through this interface, TAU requests additional address mappings by invoking a debugger and capturing a backtrace. Unlike static executables where tools such as *nm* and *addr2line* can map the addresses back to the function name and source line numbers, our tool integrates a more sophisticated mechanism to query the information from multiple sources.

To build the table of address mappings, it is essential to identify the full path of the executable or the DSO. Multilanguage applications using Python typically use DSOs and need special care with address translation. DSOs by their very nature load position-independent object code at addresses that are assigned from an offset using a runtime preloader. The same routine may be loaded at a different address in different executing contexts (ranks) of a parallel MPI application. Furthermore, in a single context, the same address may be re-used by different routines as shared objects are loaded and unloaded dynamically during the course of program execution. The address needs to be resolved during execution by reading in the file mapping address ranges for each DSO loaded. This map changes as DSOs are loaded and unloaded, and can also differ from rank-to-rank in a parallel application. (In contrast, addresses from static executables can be more easily mapped to source line numbers post-mortem.) Thus, a strategy based on capturing an address at run-time to be decoded later is not feasible for dynamic executables. To translate the address correctly, TAU first identifies the full path to the DSO or the executable for each frame of the system call-stack. To do this, TAU reads the address map at the location of the error. This address map is typically available under the */proc* file system in Linux, and is refreshed to reflect current address mappings as the application executes and uses and discards DSOs. TAU's signal handler reads this map and generates a mapping table that is used in conjunction with the addresses from the back-trace to translate the address to the source line number. In this manner, TAU constructs the complete context information at the location of the error.

2.4 Integration with the Instrumentation and Measurement System

The call-stack data collected by the diagnostic tool is stored as metadata fields in the profile data generated for each executing context by the TAU measurement system. The TAU architecture, shown in Figure 1, consists of three layers: instrumentation, measurement, and analysis. Each layer uses multiple modules that can be configured in a flexible manner

under user control. This design makes it possible for TAU to easily provide alternative instrumentation techniques that target a common measurement API. The role of the instrumentation layer is to insert code (a.k.a. probes) to make performance events visible to the measurement layer. Performance events can be defined and instrumentation inserted in a program at several levels of the program transformation process. A complete performance view may require contribution of event information across code levels. To support this, TAU's instrumentation mechanisms are based on the code-type and transformation level: source (manual, preprocessor), object (compiler-based instrumentation), library interposition, binary rewriting and run-time instrumentation, interpreter, virtual machine, and operating system (kernel-level instrumentation).

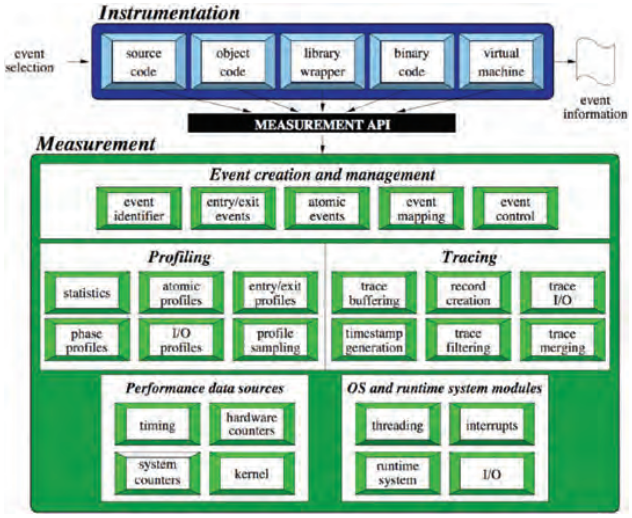


Figure 1. TAU framework architecture – instrumentation and measurement

Being able to retain run-time performance measurements made by TAU together with the call-stack information at the time of the error improves, significantly, the level of information that can be used in understanding failure scenarios.

3. Prototype

To demonstrate the feasibility of building a failure diagnostic module within the TAU framework, we created a prototype that is compatible with all modes of TAU instrumentation. It activates when a user sets an environment variable (*TAU_TRACK_SIGNALS*). Figure 2 shows the execution of an MPI application under TAU using *tau_exec* to track the signals, as well as heap memory information. To expand the scope of instrumentation, TAU supports binary rewriting using the DyninstAPI library[22]. We inject a fault in the LU NAS Parallel Benchmark[21] by dividing a number by zero. This triggers the call-stack capture module by catching the floating-point exception signal *SIGFPE*. TAU registers several signal handlers. This fault occurs in 6 of the 8 MPI ranks, yet TAU is able to generate performance data for all 8 ranks using signal handlers that intercept additional signals related to changes in the program state.

```
[sameer@mist-0-6 bin]$ setenv TAU_TRACK_SIGNALS 1
[sameer@mist-0-6 bin]$ mpirun -np 8 tau_exec -memory ./lu.i


 NAS Parallel Benchmarks 3.1 -- LU Benchmark

 Size:  33x 33x 33
 Iterations: 300
 Number of processes:     8

TAU: Caught signal 8 (Floating point exception), dumping profile with stack trace: [rank=5, pid=29368, tid=0]...
TAU: Caught signal 8 (Floating point exception), dumping profile with stack trace: [rank=6, pid=29369, tid=0]...
TAU: Caught signal 8 (Floating point exception), dumping profile with stack trace: [rank=2, pid=29351, tid=0]...
TAU: Caught signal 8 (Floating point exception), dumping profile with stack trace: [rank=1, pid=29355, tid=0]...
TAU: Caught signal 8 (Floating point exception), dumping profile with stack trace: [rank=4, pid=29367, tid=0]...
TAU: Caught signal 8 (Floating point exception), dumping profile with stack trace: [rank=0, pid=29335, tid=0]...
--------------------------------------------------------------------------
mpirun has exited due to process rank 5 with PID 29239 on
node mist-0-6.local exiting without calling "finalize". This may
have caused other processes in the application to be
terminated by signals sent by mpirun (as reported here).
--------------------------------------------------------------------------
[sameer@mist-0-6 bin]$ ls profile*
profile.0.0.0  profile.2.0.0  profile.4.0.0  profile.6.0.0
profile.1.0.0  profile.3.0.0  profile.5.0.0  profile.7.0.0
[sameer@mist-0-6 bin]$ paraprof --pack lu.i.signal.ppk
```

Figure 2. TAU captures the information in the stack trace with a floating-point exception signal handler

Figure 3 shows a back-trace of MPI rank 1 integrated in the metadata of the TAU profiles and viewed in ParaProf, TAU's profile browser. Here we can clearly see the location of the error as shown by the cursor at line number 66 of *exchange_3.f* file in the *exchange_3_* routine within the *lu.i* executable. It also shows the nature of the error (floating-point exception) in the *SIGNAL* field of the metadata. Figure 4 shows the dynamic call graph of the program recorded until the point of failure, and Figure 5 shows the exclusive time spent in this MPI rank.

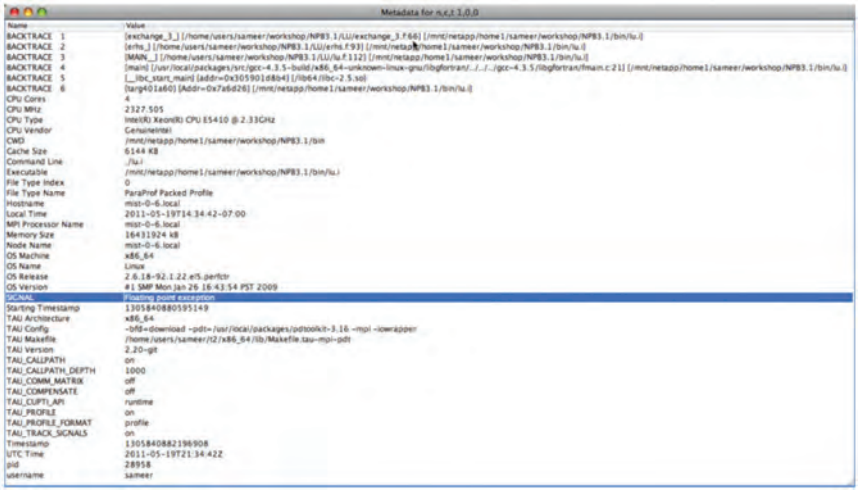


Figure 3. TAU's ParaProf browser displays the back-trace as well as signal information captured in TAU profiles

Figure 4. Program call graph for rank 1

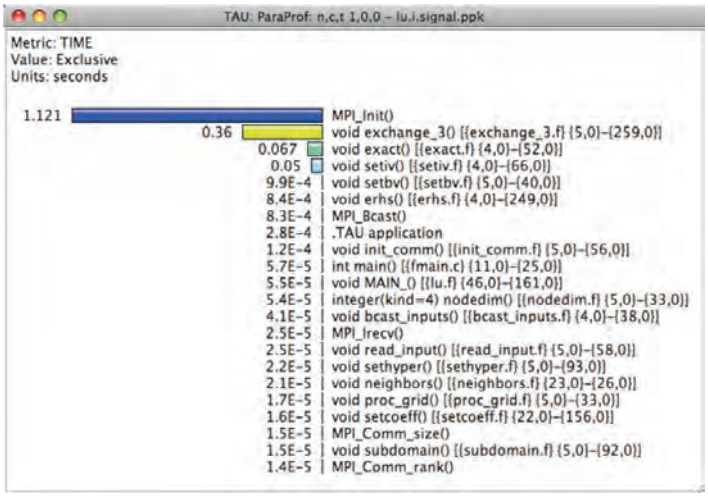
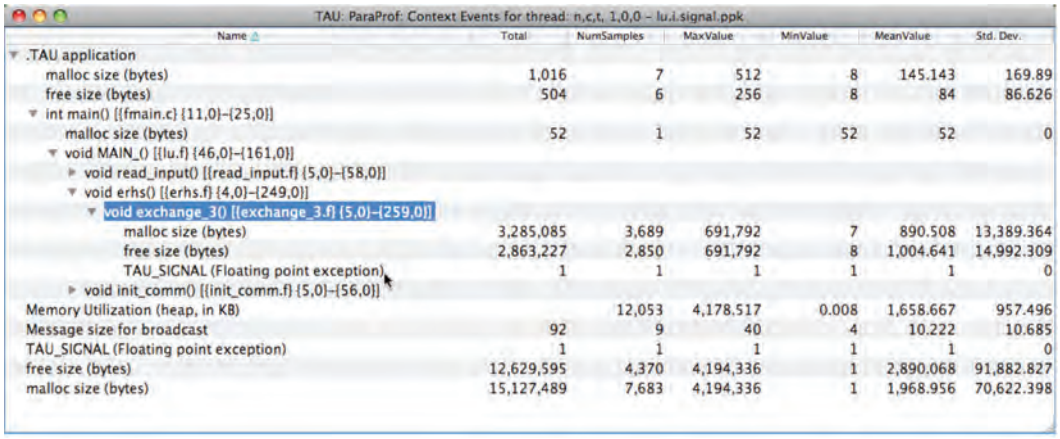


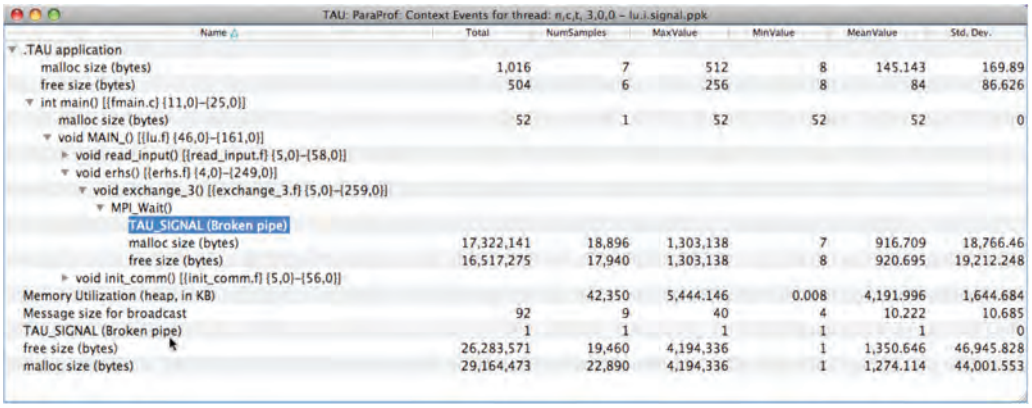
Figure 5. Performance data captured at the point of failure shows the exclusive time spent in routines on a given rank

Figure 6 shows the context event window where we see the execution context of TAU integrated with the location of the error. At the error, TAU triggers a context event with the name of the signal. This allows us to capture the TAU calling stack along with other performance metrics, such as the extent of memory and I/O. In this figure we see the peak heap memory utilization (of 4,178KB) at the location of the error. We also see the total memory allocated and de-allocated, and the memory operations along the program's calling path. In our example, a subset of MPI ranks experience the floating-point exception. Figure 7 shows the context event window on a rank that does not experience this error. TAU can generate this performance data, although no anomalous program event takes place on this rank. It captures this data by intercepting the SIGPIPE signal that is generated when the other end of the channel of communication drops in the MPI program. We see this "broken pipe" signal and where it occurs in the program stack, as shown by the highlighting and the cursor location. This context event window allows the user to at once see the top-level events and then progressively expand and contract the call-stack by opening and closing the nodes in the program callgraph.



Name	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
.TAU application						
malloc size (bytes)	1,016	7	512	8	145.143	169.89
free size (bytes)	504	6	256	8	84	86.626
int main() [{fmain.c} {11,0}-{25,0}]						
malloc size (bytes)	52	1	52	52	52	0
void MAIN_() [{lu.f} {46,0}-{161,0}]						
void read_input() [{read_input.f} {5,0}-{58,0}]						
void erhs() [{erhs.f} {4,0}-{249,0}]						
void exchange_3() [{exchange_3.f} {5,0}-{259,0}]						
malloc size (bytes)	3,285,085	3,689	691,792	7	890.508	13,389.364
free size (bytes)	2,863,227	2,850	691,792	8	1,004.641	14,992.309
TAU_SIGNAL (Floating point exception)	1	1	1	1	1	0
void init_comm() [{init_comm.f} {5,0}-{56,0}]						
Memory Utilization (heap, in KB)		12,053	4,178.517	0.008	1,658.667	957.496
Message size for broadcast	92	9	40	4	10.222	10.685
TAU_SIGNAL (Floating point exception)	1	1	1	1	1	0
free size (bytes)	12,629,595	4,370	4,194,336	1	2,890.068	91,882.827
malloc size (bytes)	15,127,489	7,683	4,194,336	1	1,968.956	70,622.398

Figure 6. TAU's context event window shows the location of the error relative to TAU's call-stack



Name	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
.TAU application						
malloc size (bytes)	1,016	7	512	8	145.143	169.89
free size (bytes)	504	6	256	8	84	86.626
int main() [{fmain.c} {11,0}-{25,0}]						
malloc size (bytes)	52	1	52	52	52	0
void MAIN_() [{lu.f} {46,0}-{161,0}]						
void read_input() [{read_input.f} {5,0}-{58,0}]						
void erhs() [{erhs.f} {4,0}-{249,0}]						
void exchange_3() [{exchange_3.f} {5,0}-{259,0}]						
MPI_Wait()						
TAU_SIGNAL (Broken pipe)						
malloc size (bytes)	17,322,141	18,896	1,303,138	7	916.709	18,766.46
free size (bytes)	16,517,275	17,940	1,303,138	8	920.695	19,212.248
void init_comm() [{init_comm.f} {5,0}-{56,0}]						
Memory Utilization (heap, in KB)		42,350	5,444.146	0.008	4,191.996	1,644.684
Message size for broadcast	92	9	40	4	10.222	10.685
TAU_SIGNAL (Broken pipe)	1	1	1	1	1	0
free size (bytes)	26,283,571	19,460	4,194,336	1	1,350.646	46,945.828
malloc size (bytes)	29,164,473	22,890	4,194,336	1	1,274.114	44,001.553

Figure 7. Context event window on rank 3 showing the signal and memory information

It is important to note that this call graph differs from the back-trace described earlier. There are two distinct call-stack entities—the system call-stack and the call-stack maintained by TAU. The TAU call-stack represents the sequence of events that are recorded by entry and exit instrumentation in interval timers. Because TAU supports a wide variety of programming languages, this call-stack is independent of the language and we can see a combination of events from C, C++, FORTRAN, and Python in this view. The Python events represent routine names in the Python scripts that TAU gets from the interpreter. These events are typically not visible to traditional debuggers such as *gdb*-based on compiled executables. By combining the two call-stacks consistently within one tool, TAU permits debugging of system back-traces along with performance data generated by higher language-level interval and atomic events providing a consistent view that will allow developers to get an accurate view of the program execution with or without fault diagnostic information.

4. Performance Evaluation of Helios Using TAU

As described in Reference 1, Helios is a code being developed for high-fidelity modeling of rotorcraft aero and structural dynamics. It consists of multiple modules written in different languages—FORTRAN90, C, and C++—that are integrated through a high-level Python-based infrastructure. This loosely coupled implementation has the advantage that each module can be developed separately from one another, but as with any parallel code, a single poor-performing module can hinder the performance and scalability of the suite as a whole. Figure 8 shows the architecture of the Helios software integration framework (SIF). Instead of the traditional model, where different physics are coupled together within a single code, Helios links together separate physics modules—computational fluid dynamics (CFD), computational structural dynamics (CSD), six degree-of-freedom dynamics (6DOF), and various interface routines—through a high-level Python-based integration framework.

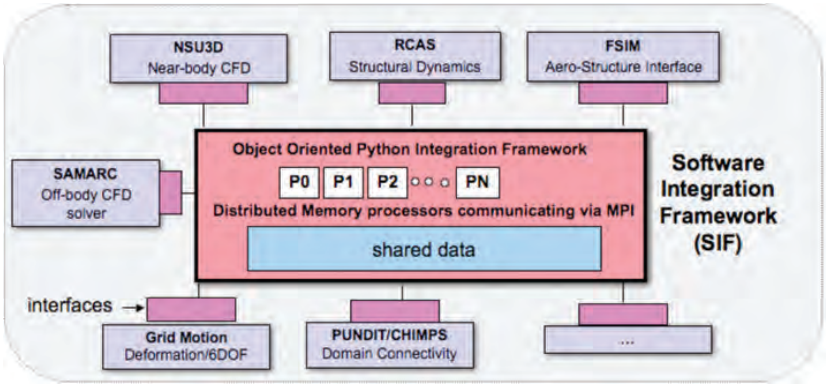


Figure 8. Architecture of Helios

The Helios software consists of multiple modules, some legacy, and some new. The CFD modules use a mixed-grid overset solution strategy. Near the body-surface, the Reynolds-averaged Navier-Stokes (RANS) solver NSU3D[5] is applied. The far-field wake is resolved using a high-order block-structured adaptive Cartesian Euler solver SAMARC[6]. The PUNDIT[7] software manages chimera-style interpolation between the different grid systems. The RCAS software[8] supplies structural loads and 6DOF information to Helios. Further details on the implementation and validation of Helios are in Reference 1.

To assess the performance of Helios, we execute the TRAM test case using tau_exec. This builds on our prior work with Helios[14]. Figure 9 shows the profile from Helios with TAU instrumentation at the Python, pyMPI[13], MPI, C++, C, and FORTRAN. We plan to extend our prototype implementation to support Python-based executions and show an integrated view of fault diagnostic information with the TAU call-stack that shows Python entities clearly in the profile.

5. Related Work

Debugging mixed-language parallel programs that use Python is a daunting task. Commercial debuggers such as TotalView[19], DDT[20], and open-source debuggers such as *gdb*[17] excel at generating back-traces for compiled executions. It is difficult, if not impossible at this time, to stop a program at a breakpoint, and move up or down the frames traversing Python and C boundaries in the same debugger, and examining and invoking Python routines and data structures. Python-level entities are visible to performance evaluation tools that operate at the Python interpreter level. By merging the back-trace operations traditionally in the debugging domain with performance introspection, we create a hybrid tool capable of diagnosing fault information based on performance instrumentation. Extensive work has been done in the area of call-stack unwinding and sampling-based measurements in the DyninstAPI, TAU, and HPCToolkit projects[23].

To better understand the performance characteristics of an application, both profiling and tracing are relevant. While profiling shows summary statistics, tracing can reveal the temporal variation in application performance. Among tools that use the direct measurement approach, the VampirTrace[15] package provides a wrapper interposition library that can capture the traces of I/O operations using the Linux pre-loading scheme used in *tau_exec*. Scalasca[10] is a portable and scalable profiling and tracing system that can automate the detection of performance bottlenecks in message-passing and shared memory programs. Like many other tools, including VampirTrace, it uses library wrapping for MPI.

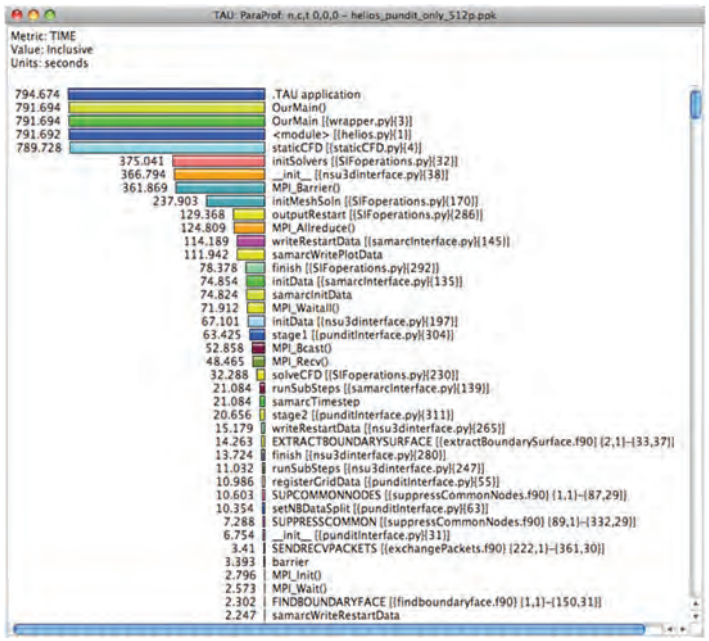


Figure 9. ParaProf shows the inclusive time for a TRAM test case in Helios using TAU instrumentation

TAU may be configured to use Scalasca or VampirTrace internally. TAU, VampirTrace, and Scalasca internally use the PAPI[3] library to access hardware performance counters present on most modern processors. However, only the *tau_exec* scheme provides the level of integration of all sources of performance information—MPI, I/O, and memory—of interest to us, with the rich context provided by TAU. With this support, we can utilize the VampirServer[9] robust parallel trace visualization system to show the performance data through scalable time-line displays of the state transitions of each process along a global timeline. Profile performance data can also be easily stored in the PerfDMF database[10]. TAU's profile browser, ParaProf, and its cross-experiment analysis and data-mining tool PerfExplorer[9] can interface with the performance database to help evaluate the scalability of an application.

6. Significance to DoD

When application software experiences a run-time failure or performance problem, it is important for concise information about the error to be communicated to the development team. Current solutions are inadequate, leaving a gap in communication between users experiencing bugs and/or performance issues and the code development team. The project aims to deliver a tool that consolidates necessary execution data for diagnostic purposes utilizing more powerful techniques for comprehensive measurement in the presence of execution errors. The goal is to close the loop with developers for more rapid turnaround of bug fixes. If no errors are encountered, the tool will report diagnostic information to users about the computational performance, memory usage, and I/O. Such information is useful for users to understand the computational characteristics of their application and for planning their computing requirements.

The other issue addressed by the new tool is it avoids the need for the user to open up problem inputs to the development team in diagnosing problems. The diagnostic file contains only run-time information, so it is more easily exchangeable to members of the development team that may not have requisite permissions to see the problem data. This is particularly important for the large amount of classified or proprietary work that takes place within the DoD.

We have extended TAU to simplify assessment of error diagnostics coupled with I/O and memory inspection for un-instrumented and instrumented applications. It allows the users to ask questions such as:

- Where and when did the program experience an anomalous operation?
- What was the nature of the fault?
- What is the heap memory utilization in the application at the time of failure?
- Were there any memory leaks in the application?
- What was the level of nesting of the call-stack?
- What was the routine name, source-file name, line number and module name at the fault location?
- What were the performance characteristics of the application at that time?
- How much time did the application spend in I/O, and communication operations at the time of fault?

It is now possible to answer these questions under Linux without re-compiling or re-linking the application. It is our goal to be able to evaluate the fault diagnostics and performance of codes that use multiple languages such as Python, FORTRAN, C, and C++.

7. Conclusion

Modern scientific software requires software components written in different languages to interact with one another. For instance, software being developed by the CREATE air vehicles program involves high-level Python scripts executing lower-level C, C++, and FORTRAN90 software components. While this multi-language paradigm enhances the re-usibility and extensibility of software, programming it is complex due to a lack of debugging tools available for inter-language execution and memory leak analysis. Also, the software is intended to run on high-end parallel computer systems which demand a high-level of sophistication in the performance evaluation tools. In this paper, we describe a new diagnostic tool under development for multi-language applications. The tool, which builds on our previous efforts using TAU for performance and memory analysis, reports memory and IO information, as well as useful diagnostic call-stack information when the code experiences some form of anomalous operation such as a segmentation fault. The tool is primarily intended to assist software developers within the DoD working in programs like CREATE, but it is also intended to be run by users as well. As CREATE software gets deployed for classified and proprietary projects within the DoD, users that experience bugs or anomalous behavior can provide details on the code execution through the diagnostic report generated by the tool, without revealing details about the application that the code is being applied.

Acknowledgements

This work was supported by the DoD High Performance Computing Modernization Program (HPCMP) User Productivity Enhancement, Technology Transfer and Training (PETTT) program and through support provided by the DoD HPCMP Office to the HIARMS Institute and the CREATE program.

References

1. Sankaran, V., A. Wissink, A. Datta, J. Sitaraman, B. Jayaraman, M. Potsdam, A. Katz, S. Kamkar, B. Roget, D. Mavriplis, H. Saberi, W. Chen, W. Johnson, and R. Strawn, "Overview of the Helios Version 2.0 Computational Platform for Rotorcraft Simulations", *AIAA-2011-1105*, 49th AIAA Aerospace Sciences Meeting, Orlando, FL, Jan 2011.

2. Shende, S. and A.D. Malony, "The TAU Parallel Performance System", *Int'l Journal of High Performance Computing Applications*, SAGE Publishers, 20(2), pp. 287–311, http://tau.uoregon.edu, Summer 2006.

3. Browne, S., J. Dongarra, N. Garner, G. Ho, and P. Mucci, "A Portable Programming Interface for Performance Evaluation on Modern Processors", *Int'l Journal of High Performance Computing Applications*, 14(3), pp. 189–204, http://icl.cs.utk.edu/papi/, 2000.

4. Knupfer, A., R. Brendel, H. Brunst, H. Mix, and W. Nagel, "Introducing the Open-Trace Format (OTF)", *Proc. ICCS 2006*, LNCS 3992, Springer, 2006.

5. Mavriplis, D.J., "Results from the Third Drag-Prediction Workshop using the NSU3D Unstructured Mesh Solver", *AIAA-2007-0256*, 45th AIAA Aerosciences Conference, Reno, NV, Jan 2007.

6. Wissink. A., Kamkar, S., Pulliam, T., Sitaraman, J., and Sankaran, V., "Cartesian-Adaptive Mesh Refinement for Rotorcraft Wake Resolution", *AIAA Paper 2010-4554*, 28th AIAA Applied Aerodynamics Conference, Chicago, IL, June 2010.

7. Sitaraman, J., M. Floros, A. Wissink, and M. Potsdam, "Parallel Domain Connectivity Algorithm for Unsteady Flow Computations using Overlapping and Adaptive Grids", *Journal of Computational Physics*, 229 (12), pp. 4703–4723, June 2010.

8. *RCAS Theory Manual, Version 2.0*, United States Army Aviation and Missile Command, AeroFlightDynamics Directorate (USAAMCOM/AFDD) Technical Report 02-A-005, US Army Aviation and Missile Command, Moffett Field, CA, June 2002.

9. Huck K. and A. Malony, "PerfExplorer: A Performance Data-Mining Framework for Large-Scale Parallel Computing", *Proc. ACM/IEEE Conference on Supercomputing (SC'05)*, 2005.

10. Huck, K., A. Malony, R. Bell, L. Li, and A. Morris, "PerfDMF: Design and Implementation of a Parallel Performance Data Management Framework", *Proc. ICPP 2005*, IEEE Computer Society, 2005.

11. Knupfer, A., H. Brunst, and W. Nagel, "High-Performance Event Trace Visualization", Proc. PDP 2005, IEEE, http://www.vampir.eu, 2005.

12. Geimer, M., F. Wolf, B. Wylie, and B. Mohr, "Scalable Parallel Trace-Based Performance Analysis", *Proc. EuroPVM/MPI 2006*, LNCS 4192, Springer, pp. 303–312, http://www.scalasca.org, 2006.

13. SourceForge, "pyMPI: Putting the py in MPI", http://pympi.sourceforge.net, 2008.

14. Wissink, A. and S. Shende, “Performance Evaluation of the Multi-Language Helios Rotorcraft Simulation Software”, *Proc. DoD HPCMP UGC 2007 Conference*, 2007.

15. Shende, S., A.D. Malony, A. Morris, and A. Wissink, “Simplifying Memory, I/O, and Communication Performance Assessment using TAU”, *Proc. DoD HPCMP UGC 2010 Conference*, 2010.

16. *GNU Binutils*, http://www.gnu.org/software/binutils/, 2011.

17. *GDB – The GNU Project Debugger*, http://www.gnu.org/software/gdb/, 2011.

18. *Backtraces – GLIBC*, http://www.gnu.org/software/libc/manual/html_node/Backtraces.html# Backtraces, 2011.

19. *TotalView*, http://www.roguewave.com/products/totalview-family.aspx, 2011.

20. *DDT*, http://www.allinea.com/products/ddt/, 2011.

21. *NAS Parallel Benchmarks*, http://en.wikipedia.org/wiki/NAS_Parallel_Benchmarks, 2011.

22. *DyninstAPI*, http://www.dyninst.org, 2011.

23. Malony, A.D., J. Mellor-Crummey, and S. Shende, “Measurement and Analysis of Parallel Program Performance using TAU and HPCToolkit”, *Performance Tuning of Scientific Applications*, (eds. D. Bailey, R. Lucas, and S. Williams), CRC Press, pp. 49–86, 2010.

A Microsoft® HPC Portal

Pat Collins and Thomas Kendall
US Army Research Laboratory (ARL), Aberdeen Proving Ground, MD
{pat.collins4, thomas.kendall}@us.army.mil

Jim Waterman and Mike Knowles
Lockheed Martin, Aberdeen Proving Ground, MD
{james.l.waterman, mike.knowles}@us.army.mil

Rob Fisher
BAE Systems, Vienna, VA
jrfisher@hpcmo.hpc.mil

Andy Greenwell and John Kreatsoulas
Microsoft, Cambridge, MA
{andy.greenwell, john.kreatsoulas}@microsoft.com

Tom Quinn
Microsoft, Baltimore, MD
thomas.quinn@microsoft.com

Abstract

A web portal for integrating high performance computing (HPC) applications with the desktop is described. It is based on Microsoft's HPC server and RemoteApp technology. A typical portal application has two components: a client and an HPC component. When users click an application icon in the web page, the associated client runs on the portal and through Microsoft's RemoteApp technology the graphical user interface (GUI) is display on the desktop. It appears to users as if the application is running locally. The users interact with the client as usual but now, by virtue of the HPC component, users have the capability of running large, computationally-intensive jobs on backend HPC servers right from the GUI. The users never have to learn more than what is required by the application itself to take advantage of the power of HPC. Parallel computing and job submission details are essentially hidden from the user.

1. Introduction

High performance computing (HPC) plays a key role in developing the advanced weapons systems that keep the US Armed Forces the best in the world. New software tools are constantly being written to help design these systems and as each new tool emerges the computational requirements increase; consequently, HPC has become a necessity. Many DoD scientists and engineers (S&Es) already utilize HPC through resources provided by the High Performance Computing Modernization Program (HPCMP); however, it is widely recognized that many others who could easily benefit from HPC do not use it. The main reason is because traditional HPC requires a certain level of knowledge in parallel computing, UNIX, job scheduling, and scripting. Most S&Es prefer to focus on their specific areas of expertise and do not have the time or organization mandate to become fluent in parallel computing. This is as it should be. The goal of the Microsoft ARL HPC Portal is to remedy this by making HPC resources easily accessible to all.

There are many ways to approach making HPC easier to use, and each has their distinct advantages. Most are based around simplifying and minimizing steps required for an end-user to "submit" and run an application on a traditional cluster by providing web-based interfaces to traditional cluster job management tools and schedulers. Examples for generic portal solutions include Platform Application Center [Platform Computing, 2011], Ganglia's web front-end [Ganglia, 2011], ezHPC [ERDC, 2011], and Microsoft's SharePoint web-parts for HPC Server [Microsoft, 2010]. Many institutions have created customized portal interfaces specific to one or several applications and/or user communities such as BioHPC [Cornell University, 2011], Genomeweb [Genomeweb LLC, 2011], and Cornell's MATLAB implementation on the TeraGrid [Cornell University, 2011]. The approach we take is based on remote application serving, which provides the end-users the application client they use every day on their desktops.

A suite of Microsoft technologies is used here to build an HPC Portal. These technologies provide the mechanism for securely delivering HPC-aware applications to the user's desktop without having to install any software on the desktop or using any network transport protocols other than HTTPS. HPC-aware applications come fully-integrated with the Windows HPC server scheduler, and this allows users to setup and run HPC jobs through the application interface alone. The Portal provides the means and the infrastructure to take advantage of them easily. It presents the graphical user interface (GUI) to the user, while hiding the communication layer and HPC job scheduling tasks. Applications are run on a portal server, not the desktop, but their interfaces are fully-integrated with the desktop through the use of the RemoteApp/Remote Desktop Protocol technologies. Local disk drives and printers can be mapped to the remote session. All this combined makes the user experience almost identical to running applications locally, but with added HPC capabilities.

There are a number of independent software vendor applications that are HPC-aware and so are candidates for the Portal. MATLAB® is one such application. Its relatively large user-base makes it the primary application for the initial instance of this Portal. Using the MathWorks® Parallel Computing Toolbox™ (PCT) and the MATLAB Distributed Computing Server™ (MDCS), users can setup and run parallel jobs through MATLAB constructs only. Other applications that have hooks for the Windows HPC scheduler and are easily delivered by the portal are: Microsoft's Excel®; Ansys's CFX®, HFSS™, and Fluent®; Simulia's Abaqus®; CD-Adapco's Star-Cd® and Star-CCM+®; and COMSOL Multiphysics® just to name a few. See Microsoft, 2011 for a more complete list.

2. Concept of Operations

From the user's perspective, the concept of operations is very simple. Users:

- connect to the Portal web site with a browser,
- use their Common Access Card (CAC) to authenticate,
- are presented with a list of applications they have been granted access to,
- click and run an application,
- copy data to and from the Portal through an explorer window also published as an application on the portal,
- set up and run HPC jobs through procedures defined by the application itself, and
- post-process the results on the Portal using the application itself, or copy results back to the desktop.

Once the user clicks on an icon and the GUI appears, the user's interaction with the application is no different from interacting with the same application running locally, except now it has HPC resources at its disposal. The procedures to take advantage of HPC do not require the user to learn anything other than those defined by the application itself. Generally, this is minimal and does not require the user to become a parallel programming expert or to interface directly with the scheduler.

3. Portal Architecture and Design

The main architectural components are shown in Figure 1 with brief descriptions of their functions given in Table 1. It is based on many mature Microsoft technologies that are widely-used in industry; however, this architecture is the first instance of using these technologies in concert for delivering HPC applications to the desktop. The rest of this section describes how these components work together to form the Portal.

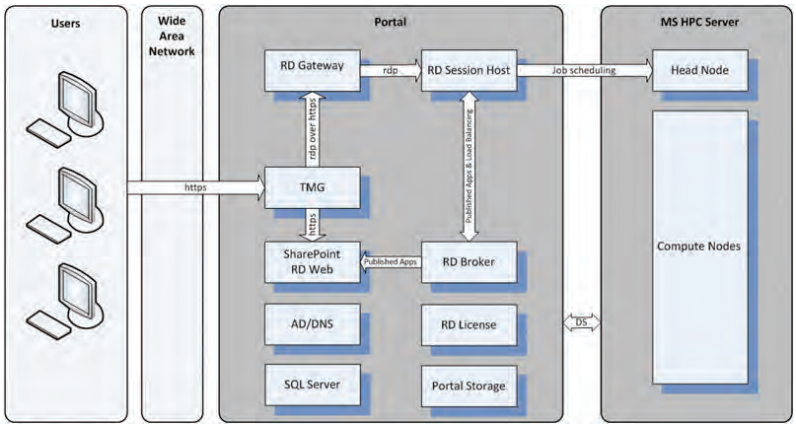


Figure 1. Portal architecture

Table 1. HPC Portal architectural components

Component	Functions
RD Gateway (Remote Desktop Gateway)	Provides a gateway for users to connect to remote application servers using RDP encapsulated in https connections.
TMG (Threat Management Gateway)	The TMG is a Forefront threat management gateway firewall that provides access controls to the Portal and CAC authentication for the website and RemoteApp applications.
SharePoint/RD Web	A SharePoint website that is the entry point into the Portal. It presents the user with a list of authorized applications.
RD Web	Integrates Remote Apps with the SharePoint website.
AD/DNS (Active Directory/Domain Name Services)	Provides Domain Services (DS) and DNS services for the Portal domain.
SQL Server	Used for implementing dual HPC server head nodes.
RD Session Host	A ortal server that runs client applications.
RD Broker	Collection point for all published applications. Provides the list of applications and Session Host servers to the "RD Web" component on the Portal website. Load balances connections across multiple RD Session Hosts.
RD License	Provides client access licenses (CAL) for RD RemoteApp and SharePoint access.
Portal Storage	Main storage system for the Portal. It provides shared file space for HPC applications as well as user home directories.
Head Node	Runs the HPC scheduler and provides a single point of control for the HPC cluster.
Compute Nodes	Provides CPU resources for computationally-intensive Windows applications.

Microsoft's RemoteApp technology is used to publish applications on a SharePoint web Portal. It is based on the Remote Desktop Protocol (RDP) and provides the mechanism for integrating the application's GUI with the user's desktop. To publish an application, for example the MATLAB client, the software is installed on the Remote Desktop (RD). Session Host servers and any associated HPC parts, for example the MATLAB Distributed Computing Server (MDCS), are installed on the shared file system. The RemoteApp Managers on each RD Session Host server are configured to publish the applications and all RD Session Hosts are put into a load-balancing farm. The Remote Desktop Broker communicates with all RemoteApp Managers collecting information about published applications and providing it to SharePoint. The broker also keeps track of Session Host connections, and that information is used for connection load-balancing. SharePoint presents the applications to the users through icons on the web page using the TSPortalWebPart web part (an ActiveX control). Details of the RemoteApp connection process including load-balancing can be found in Anderson & Griffin, 2010.

Using Active Directory group policies and other security features, the applications presented are restricted to those that the user has been authorized to run. In addition, AppLocker is utilized to control what applications are allowed to run on the RD Session Hosts through an application white list, blocking potentially malicious actions, protecting the systems, and maintaining stability across the platform for all users sharing the system.

To access and run a published application, the user first surfs to the Portal website. The Threat Management Gateway accepts the web connection on behalf of the SharePoint Server, inspects the traffic, and requests smart card authentication using the DoD Common Access Card (CAC). Kerberos Constrained Delegation is used to convert the smart card web authentication into a Kerberos ticket that is sent to the SharePoint server by the Threat Management Gateway on behalf of the user. Once access to the SharePoint Portal is granted, a list of application icons appear. Clicking an icon downloads a RemoteApp configuration file that is processed by an ActiveX browser plug-in, providing an interface to the Microsoft Remote Desktop Connection client (mstsc.exe) component of the Windows desktop. This component establishes a bridged https connection between the client and the RD Gateway through the Threat Management Gateway. An initial RDP connection is then made to a member of the RD Session Host farm which acts as a "redirector". The redirector authenticates the user on the RD farm, but does not actually launch the application; instead, it directs the session to the RD Connection Broker. The RD Connection Broker then determines which server in the RD Session Host farm will host the user's application session. An end-to-end RemoteApp RDP connection is then established between the client and the assigned Session Host, through the bridged https connection channel. The client application is started and its GUI appears on the desktop.

At the current time, single sign-on is not available. Dual sign-on is required, once for authentication to SharePoint, and another to the Remote Desktop Gateway. This is because of the Kerberos Constrained Delegation hop limit, and because there are no domain trusts configured between users systems and the Portal System. If the user's organizations were to establish domain trusts between the Portal System and their own, this second prompt would be eliminated. This; however, isn't practical, given that users would be coming from a variety of domains or other configurations. So, when a user, who already CAC-authenticated to the website, runs an application, it must also be CAC-authenticated. This is only required when the first application is launched. Subsequent applications run will not require authentication if at least 1 authenticated RemoteApp session remains up.

The Microsoft HPC server and the Portal storage system, shown in Figure 2, are central to the Portal. The head node runs the job scheduler and controls access to the HPC resources. It is also the control point for the HPC cluster where cluster management is performed. The compute nodes provide the computational resources necessary to run parallel applications. HPC-aware applications have built-in interfaces to the head node scheduler. It is through this interface the user submits HPC jobs. The procedures for doing this depend on the application. In the next section, a simple example will show how this can be done in MATLAB.

Data transfers to and from the Portal is accomplished using Windows Explorer and the Remote Desktop Protocol. The clipboard must be available for use in the Remote Desktop Session, in order to copy and paste files back and forth between the Portal systems and the users local desktop. The Portal file system is accessible to users through a Windows Explorer application that is published on the portal. Running this application gives users access to their home directory. File transfers occur by copying and pasting files to this window. The user's local desktop file systems are available to Portal applications such as the Windows explorer and MATLAB client; however, these file systems are not accessible on the HPC compute nodes, so data required for an HPC job must exist on the Portal file system. Portal home directories are accessible from all the HPC compute nodes and all RD Session Hosts.

This Microsoft HPC Portal imposes constraints on the client's desktop. To use the Portal, a user must run Internet Explorer version 6.0 or higher, and the desktop must have installed a version of Remote Desktop Connection (RDC) that supports Remote Desktop Protocol (RDP) 6.1 or higher. RDC 6.1 supports Remote Desktop Protocol 6.1. RDC 6.1 is included with Windows Server 2008, Windows Vista with Service Pack 1, and Windows XP with Service Pack 3. Windows 7 and Windows Server 2008 R2 run a version of RDC that supports RDP version 7.0. These are the supported platforms. Additionally, the Remote Desktop Services ActiveX Client control must be enabled. The appropriate version of the ActiveX control is included with RDC version 6.1 and higher.

4. Implementation

The Portal is currently built on an IBM eServer 1350 cluster with 2 BladeCenter H chassis, each with 8 HS22 blade servers, and 7 IBM System x3650 M3 rack-mounted servers. Two RD Session Host servers and 14 compute nodes are on the BladeCenters. Each of these nodes has dual 6-core Intel x5670 2.93GHz processors with 24GB of memory. The rest of the infrastructure is built on the x3560 rack-mounted servers, including a 2-node General Parallel File System (GPFS) cluster running RedHat Linux that serves 36TB of storage from an IBM DS3400 storage system.

The Portal will soon be augmented with 12 additional IBM BladeCenter nodes, and up to 20 more nodes from US Army Research Laboratory's (ARL's) Utility server built by Appro. This new hardware will provide more RD Session Hosts and compute nodes, as well as redundancy for the infrastructure. Figure 2 shows the planned implementation with the new hardware. Dual Windows HPC server head nodes and dual Threat Management Gateways will be implemented on four of the x3650 servers. A single x3650server will be used for management functions such as providing operating system and virus definition updates and central logging. The other portal functions will be moved to the BladeCenters. The IBM system alone will provide 4 Session Host servers and 16 compute nodes, and will have a total of 562.56 Ghz and 384GB of memory.

As mentioned above, the system will also be augmented, as needed, with an additional 16 compute nodes and 4 Session Host servers from ARL's Utility server. These nodes will have dual boot capabilities so that they can function as either Linux nodes on the Utility server, or Windows nodes on the Portal. Each Utility server node has dual AMD Magny-Cours, 8-core processors with 128GB of memory.

MATLAB licenses are provided for up to 16 concurrent users and MDCS worker licenses for parallel jobs that can utilize all available cores. All MATLAB toolboxes will be available for a limited period of time, enabling the Portal team to gauge which toolboxes and what quantity should be purchased for the Portal.

Windows Server 2008 R2 is used for all Windows systems. The Enterprise edition is used on the Portal infrastructure and the Windows HPC Server 2008 R2 suite is installed on the HPC servers. Two RedHat Linux nodes provide the GPFS cluster.

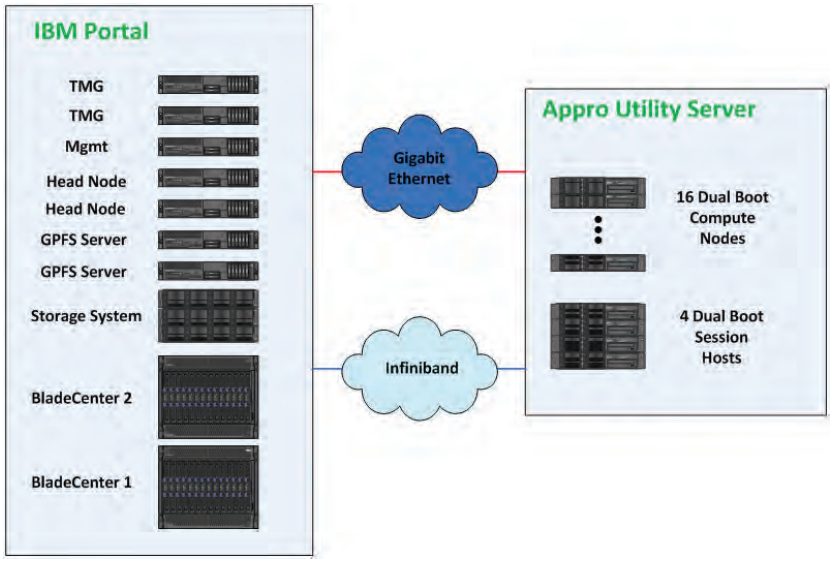


Figure 2. Planned implementation of the Microsoft HPC Portal

5. MATLAB

To illustrate a simple example of using the portal, a MATLAB-authorized user goes to the portal website, CAC-authenticates and is presented the webpage shown in Figure 3. Clicking on the MATLAB icon starts the application running on a Session Host server with the GUI, shown in Figure 4, presented on the user's desktop. The application starts up in the pre-assigned "MATLAB" directory off of the user's home directory. By running the explorer application, see Figure 3, the user can copy codes and data to the Portal.

To run MATLAB jobs in parallel, the user first parallelizes his code using constructs from the Parallel Computing Toolbox. These constructs hide much of the details of parallel programming, allowing the user to focus on the application. When ready, the user submits a parallel job to the cluster using only the mechanisms provided by the PCT. System MATLAB startup scripts will provide a default HPC scheduler that is needed by PCT for submitting jobs to the cluster. The user will not have to configure a scheduler and so job submission details are essentially hidden from the user.

A simple example for creating and submitting a MATLAB pool job is shown below.

```
function job = prun_cluster( mpool )
    sched = findResource();
    job = sched.createMatlabPoolJob( ...
        'MaximumNumberOfWorkers', mpool, ...
        'MinimumNumberOfWorkers', mpool);

    job.createTask(@MyCode,2);
    job.submit
    job.wait;

    data = job.getAllOutputArguments();

    % Post process data
```

The function shown is called prun_cluster. It takes 1 argument, mpool, that controls the number of MATLAB workers requested. The pre-defined HPC scheduler is attached to the job with the findResource() command. The MATLAB pool job and the tasks defined by the user's code, MyCode, are created. The job is submitted and the client is instructed to wait for the results. Output data is returned to the client using the job.getAllOutputArguments() command. Subsequent code, not shown, can be used for plotting, post-processing, or saving this data. Because the MATLAB GUI is tightly integrated with the desktop, output data and graphics can be sent to the local printers or stored on the desktop.

This is a very simple example, and does not include any details about how to use PCT to make the user's code, MyCode, run efficiently in parallel. It illustrates one simple way that PCT can be used to submit an HPC job. The point is that MATLAB user needs only learn the MATLAB Parallel Computing Toolbox to take advantage of HPC. The PCT does an excellent job of hiding the complexity of parallel computing from the user. Documentation on the PCT/MDCS can be found at MathWorks, Inc., 2011.

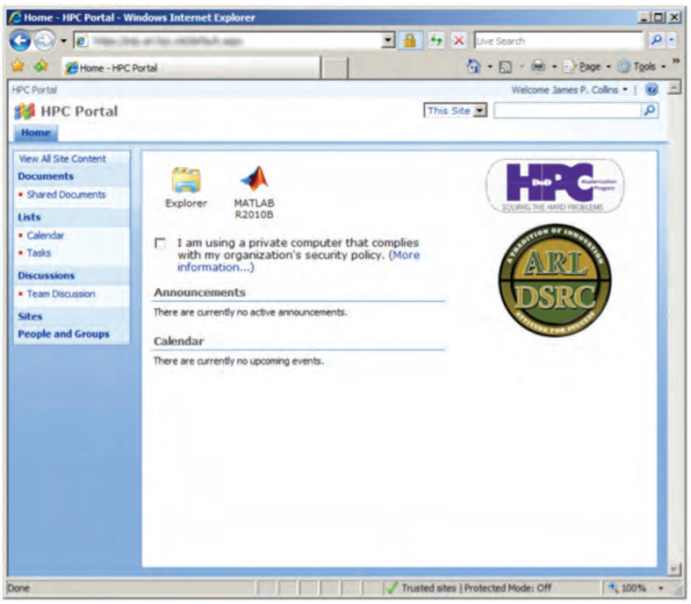


Figure 3. The HPC Portal home page that MATLAB users see

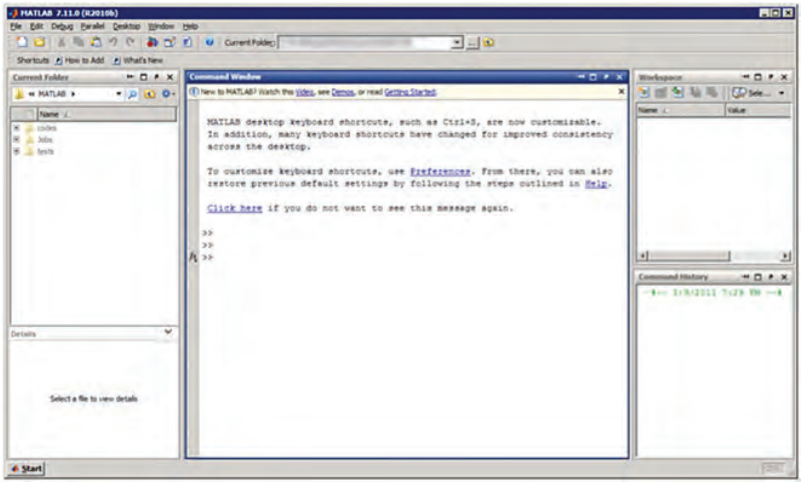


Figure 4. The MATLAB application interface that is presented to the users' desktop. It is the same interface users see when running MATLAB on their desktop.

6. Summary

Many DoD S&E's are familiar and comfortable with the Windows desktop. Some run computationally-intensive Windows applications on their desktop because they have no alternative. This Microsoft-based Portal, while focused initially on MATLAB, will eventually provide computational resources for other Windows applications. These could be highly-parallel scientific application codes or just serial codes used in large parametric studies. It may also include Microsoft Excel running in parallel. An up-to-date list of applications for Windows HPC Server can be found at http://www.microsoft.com/hpc/en/us/solutions/all-applications.aspx. In any case, a significant advantage will be realized.

Through this Portal, users will have a place to go for the computational resources they need without leaving their familiar Windows desktop environment, and without having to become HPC experts. Because applications do not need to be installed on the desktop, users will have much more flexibility. From home or while on travel, users will be able to continue their work and not be tied to the desktop. The computational resources, the ease-of-use, and the flexibility will be a significant benefit to researchers engaged in developing new technologies that enable the US Warfighters to better perform their mission.

Acknowledgements

The authors would like to thank Charlie Nietubicz for his vision and support of a Microsoft HPC portal, the US Army Research Laboratory and the High Performance Computing Modernization Program for their financial support, and to Niraj Srivastava, Technology Officer, Raytheon Corporation, for his early efforts in defining the architecture and setting up the initial prototype. We would also like to thank Curt Aubley and Troy Landry from the Lockheed Martin NexGen for allowing us to prototype a similar system at their facility, and to Kevin Crowley from HP for lending us the prototype hardware.

References

Anderson, C. and Griffin, K., *Windows Server 2008 R2 Remote Desktop Services Resource Kit*, Microsoft Press, Redmond, WA, 2010.

Cornell University, *BioHPC*, retrieved May 13, 2011, from http://biohpc.org/, 2011.

Cornell University, *MATLAB on the TeraGrid*, retrieved May 13, 2011, from http://www.cac.cornell.edu/matlab/, 2011.

ERDC, *ezHPC Wiki*, retrieved May 13, 2011, from https://ezhpc.hpc.mil/EZHPCv2/doc/ezhpc_wiki.html, 2011.

Ganglia, *Ganglia Monitoring System*, retrieved May 13, 2011, from http://ganglia.sourceforge.net/, 2011.

Genomeweb LLC, *GenomeWeb*, retrieved May 13, 2011, from http://www.genomeweb.com/, 2011.

MathWorks, Inc., *Parallel Computing Toolbox*, retrieved May 13, 2011, from http://www.mathworks.com/products/parallel-computing/, 2011.

Microsoft, Microsoft *Windows HPC Server Applications*, retrieved May 13, 2011, from http://www.microsoft.com/hpc/en/us/solutions/all-applications.aspx, 2011.

Microsoft, *What is Included in the HPC Pack 2008 SharePoint Integration Sample*, retrieved May 16, 2011, from http://technet.microsoft.com/en-us/library/ff601833(WS.10),aspx, 2010.

Platform Computing, *Platform Application Center*, retrieved May 16, 2011, from http://www.platform.com/workload-management/high-performance-computing/add-on-products/platform-application-center, 2011.

A Practical Application of the Computational Science Environment (CSE)

John Vines, Kelly Kirk, and Eric Mark
US Army Research Laboratory (ARL), Aberdeen Proving Ground, MD
{john.m.vines, kelly.t.kirk, eric.mark}@us.army.mil

Carrie Spear and Joel Martin
Lockheed Martin/US Army Research Laboratory (ARL), Aberdeen Proving Ground, MD
{carrie.spear, joel.martin5}@us.army.mil

Abstract

The Computational Science Environment (CSE) is a collection of open-source software tools and utilities, and encompasses a large number of state-of-the-art APIs, (i.e., Qt, Python and SciPy). The CSE software development system fosters the development of modern software applications, and is structured to support individuals, small teams or large distributed development groups. An integral piece of CSE is its intrinsic support for software testing. All production software development efforts will require testing for verification and validation of functionality and results. The CSE provides tools for extensive software testing suites and quality assurance dashboards to post results.

Extensibility is another core capability of the CSE. Software development projects, or existing production-level applications, can leverage the CSE's dynamic environment through the use of "add-ons". A CSE add-on provides the capability to incorporate established applications and previously-developed utilities. A good example of a CSE add-on has been developed for the HPCMP's Multi-Scale Reactive Modeling (MSRM) Institute for the Multiple Object Evolutionary Strategies (MOES) code.

The MSRM's Infrastructure team has worked closely with the CSE team and MOES developers to design a cross-platform build-and-testing system for the MOES code. The CSE MOES add-on provides the MSRM institute with the ability to utilize, develop, build and test the entire MOES system.

1. Introduction

The Computational Science Environment (CSE) provides users with a standard development environment, as well as open-source software tools and utilities for data analysis and visualization. A significant advantage of using CSE is its ability to provide researchers and engineers with a development foundation capable of supporting their software project. This is supported through a CSE subproject or "add-on". CSE provides a straightforward framework for the creation and automatic testing of add-ons, along with the ability to customize the build process by specifying particular compilers, optimization flags, and external libraries to meet the researcher's needs. CSE add-ons are a simple way for researchers and developers to extend the common baseline set of tools, libraries, and applications in CSE. The CSE team has worked closely with the MOES developers, as well as the MSRM's Infrastructure team, to design a modular build-and-test system that simplifies the entire build process, and make the projects easier to maintain, and modify.

2. Configuration Management

2.1 The Software Repository

Source code management for the MSRM project is done through GIT. GIT is a distributed version-control system unlike Subversion, which is a centralized version-control system. Because GIT is a distributed version-control system, when a user performs a checkout (clone), every revision of every file is available as part of the local copy. An additional benefit of GIT is branching. While branching is not unique to GIT, one benefit is being able to completely remove a branch (there is no trace like there is in SVN), so branches can be easily created and discarded as necessary.

GIT Repository permissions are handled at the OS- and file-system-level through the use of Linux groups and File-system Action Control Lists (ACLs). Each project is designated a unique group name for the facilitation of permissions, only individuals designated by the project POC will be added to the group.

ACL - a list of permissions attached to an object, this list specifies which users or groups are granted permissions to an object and what actions can be performed on that object

The following commands are used to provide a specific group with read- and write-permission to their repository:

setfacl -R -m g:cse:rwX cse_stable
find repo -type d | xargs setfacl -R -m d:g:cse:rwX

There are two software repositories for each Atomistic add-on. Each add-on has a development and a production repository. All developers have read- and write-permissions to the development repository. Only individuals deploying the add-on have full-access to the production repository. Each of these repositories is built, tested, and reported on nightly. The production repository is built, tested, and deployed on various HPC systems as directed.

2.2 Software Configuration Management

Because the team is "geographically-distributed" an online interface was necessary to simplify communication. The team uses Redmine (http://www.redmine.org) an open-source, database-driven project management web application. The Redmine application offers a substantial preconfigured solution that is easy to set up and can be customized to better fit a particular project. Redmine was chosen as it provides a number of features that are of interest for team interaction. The most significant features were the ability to support multiple projects, a customizable issue-tracking system, email and news notifications, wiki support, access control, and integration with several source-code management systems (e.g., git, as described above). Redmine's tracking system provides a simple solution for managing bugs, general support issues, and requests for new features allowing the entire team to determine priorities, provide updates, and know the status of any outstanding items. As items in the tracking system are updated the responsible personnel and others with an interest are informed via email. Redmine also provides a common place for current source code via the connected git repository, as well as code and user documentation via its wiki and file storage features. Collectively, these features reduce the time needed to oversee and manage multiple projects, allow the team to keep track of issues specific to their assigned projects, and provide a common place for project-related communication.

3. CMake and Modules

The CSE MOES add-on utilizes CMake to drive the build process. CMake is an open-source cross-platform build system with strong system introspection capabilities. There are numerous Macros distributed with CMake to discover what packages are installed on a system, their version numbers, and where these packages are installed. Because the MOES code was needed by the team's scientists on several different HPC architectures, the system introspection capabilities of CMake significantly simplified deployment. There is a top-level CMakeLists.txt file that includes all of the sub-packages, as well as the module installation code.

The following is from the top-level CMakeLists.txt project. It sets up the default installation prefix and includes all of the sub-package directories. The CSE provides template files for the top-level CMakeLists.txt file:

```
# Add-on CSE_ADDON_CVS Packages
cmake_minimum_required(VERSION 2.8)
# The name of the build project
project(CSE_ADDON_MOES)
set(SUBPROJECT "moes")
set(HOME_DIR MOES)
include(CTest)

# Enable testing for the project
ENABLE_TESTING()

include(ExternalProject)
set(base "${CMAKE_BINARY_DIR}/CMakeExternals")
set_property(DIRECTORY PROPERTY EP_BASE ${base})

# Find the CSE installation on this system

find_path(CSE_HOME  Release  $ENV{CSE_HOME} /usr/cta/CSE $ENV{HOME}/CSE)
list(APPEND CMAKE_MODULE_PATH "${CSE_HOME}/Misc/CMake")
```

```
find_package(CSE)
# Default installation prefix is /home/userid, this can be changed at configure time
# To change the Install prefix cmake -DMOES_INSTALL_PREFIX=/prefix/you/want
set (MOES_INSTALL_PREFIX "$ENV{HOME}/${HOME_DIR}" CACHE PATH "Install Path")
set (CMAKE_INSTALL_PREFIX ${MOES_INSTALL_PREFIX} CACHE INTERNAL "" FORCE)

find_program(PATCH_PROGRAM patch
      PATHS /usr/bin)

# include all subdirectories
add_subdirectory(airebo)
add_subdirectory(lp_solve)
add_subdirectory(reac)
add_subdirectory(moes)
```

Modules are automatically built and installed with each CSE add-on. Modules are a command-line tool providing dynamic modification of a user's environment, thus eliminating the need for the user to modify their own environment variables (PATH, LD_LIBRARY_PATH, PYTHONPATH) to compile, test, and run code. The CSE team provides a default file for each module that is populated by package-specific variable names at configure time.

The MOES project installs the following modules for use by MOES developers and individuals performing batch-runs utilizing the MOES code:

```
---------------------- /usr/cta/CSE.atomistic.2011-04-22/MOES/modules ----------------------

cse-msrm-atomistic/airebo/1.1      cse-msrm-atomistic/moes/1.0
cse-msrm-atomistic/airebo/latest   cse-msrm-atomistic/moes/latest
cse-msrm-atomistic/lp_solve/5.5    cse-msrm-atomistic/reac/1.2
cse-msrm-atomistic/lp_solve/latest   cse-msrm-atomistic/reac/latest
```

4. Automated Build-and-Test

CTest is a testing tool distributed with CMake. This tool can be used to automate building and testing of a project. CTest can also submit testing results to a dashboard for display and review. CDash is a product distributed with Kitware's CMake. CDash is a web-based, open-source software testing server. CDash organizes and displays testing results on a simple, easy to understand web page. The immediate feedback that developers receive on the dashboard helps to encourage careful testing and code review prior to submitting code modifications to the repository.

In order to submit to a local dashboard, it is necessary to have a CTestConfig.cmake file. The following is a basic sample of a CTestConfig.cmake file:

```
set(CTEST_PROJECT_NAME "MOES_Development")
set(CTEST_NIGHTLY_START_TIME "21:00:00 EDT")
#use https
set(CTEST_DROP_METHOD "https")
set(CTEST_DROP_SITE "your.web.dash.site")
set(CTEST_DROP_LOCATION "/CDash/submit.php?project=MOES_Development")
set(CTEST_DROP_SITE_CDASH TRUE)
set(CTEST_CURL_OPTIONS "CURLOPT_SSL_VERIFYPEER_OFF;CURLOPT_SSL_VERIFYHOST_OFF")
```

For the CSE MOES add-on, each package is tested individually and the entire project is also tested as a whole. Because all of the packages in MOES use CMake to build all of the regression tests leverage CTest, the CTest output is stored locally in an XML file that gets submitted to the CDash quality-assurance dashboard upon completion of the builds and tests. In order to ensure that software modifications did not impact numerical accuracy, test results are automatically validated against a known-set of values. This verification is done through a bash-script that gets executed by CTest; value tolerances can be set in this script if necessary. The following line needs to be added to the CMakeLists.txt file to enable testing for the project:

```
enable_testing()
```

The CMake tests are added to the build simply by adding the following lines to the CMakeLists.txt file, these lines call the bash-scripts that execute the tests:

```
add_test(TestMoesCalculate test_moes.sh)
add_test(TestAireboDriver test_airebo.sh)
```

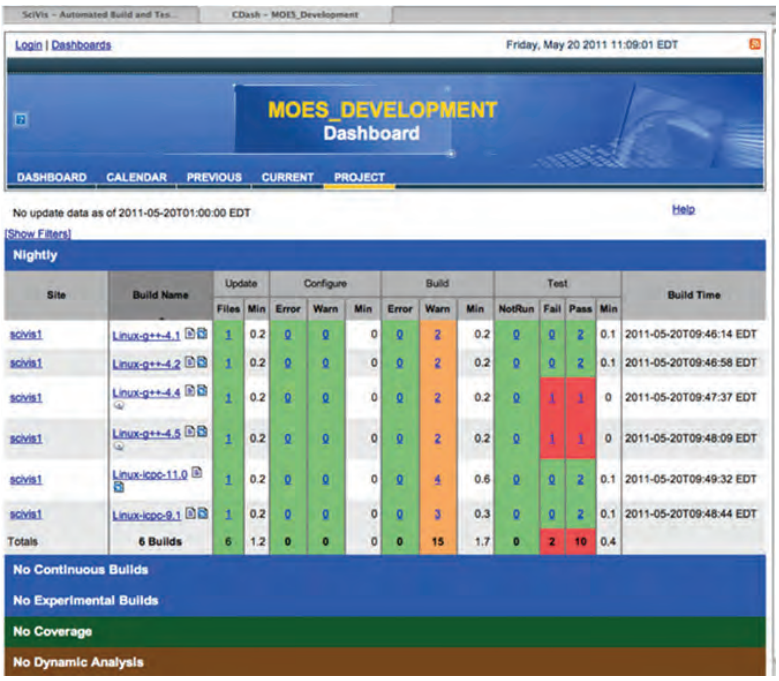


Figure 1. The quality-assurance dashboard reports the results of the test

5. Data Analysis and Visualization

Utilizing several of the tools available in CSE (python, numpy, pyqt, and matplotlib), the CSE team worked with the MSRM Infrastructure team and the MOES developers to assist with the creation of data conversion and visualization programs. A python program was developed to translate the MOES output code into a CSV file that can be used by numerous spreadsheet and graphing packages. A python program was also developed to automatically generate graphs of the data output using python, numpy, and matplotlib.

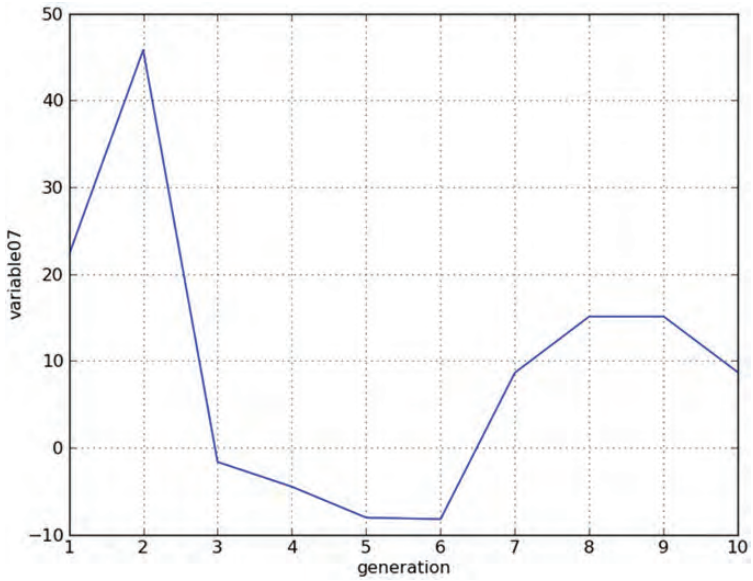


Figure 2. Example plot of MOES data

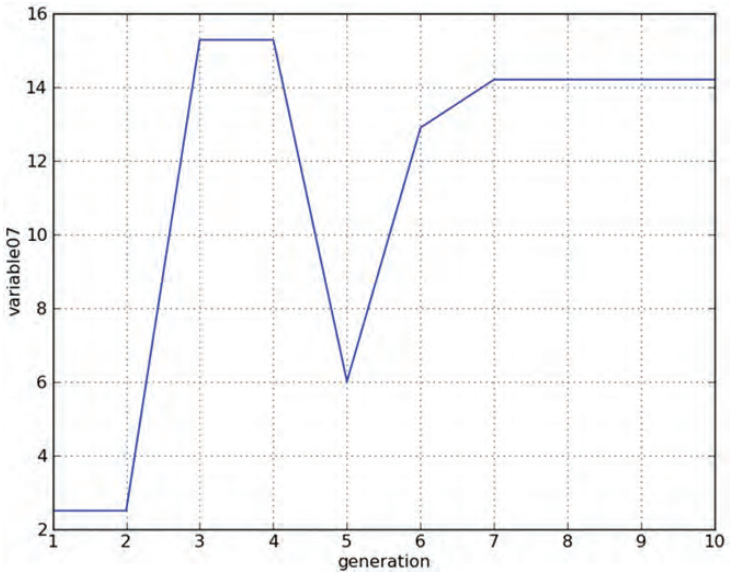


Figure 3. Example plot of MOES data

6. Conclusion

The CSE offers a comprehensive software environment providing a flexible environment, modern software APIs, and supporting development efforts scaling from individuals to large geographically-dispersed teams.

An integral part of the CSE is the automated building, testing, and reporting system for all software development projects.

The CSE "add-on" framework allows users and developers to create domain-specific capabilities outside of the system-level architecture.

Acknowledgments

The authors would like to thank the High Performance Computing Modernization Program for supporting this work through computing resources.

Energy Efficiency Evaluation and Benchmarking of AFRL's Condor High Performance Computer

Ryan Luley, Courtney Usmail, and Mark Barnell
US Air Force Research Laboratory, Information Directorate, Computing Technology Applications Branch (AFRL/RITB), Rome, NY
{ryan.luley, courtney.usmail, mark.barnell}@rl.af.mil

Abstract

Emerging supercomputers strive to achieve an ever-increasing performance metric at the cost of excessive power consumption and heat production. This expensive trend has prompted an increased interest in green computing. Green computing emphasizes the importance of energy conservation, minimizing the negative impact on the environment, while achieving maximum performance and minimizing operating costs.

The Condor Cluster, a heterogeneous supercomputer composed of Intel Xeon X5650 processors, Cell Broadband Engine processors, and NVIDIA general-purpose graphical processing units was engineered by the Air Force Research Laboratory's Information Directorate, and funded with a Dedicated High Performance Computer Project Investment (DHPI). The 500 TeraFLOPS Condor was designed to be comparable to the top-performing supercomputers using only a fraction of the power. The objective of this project was to determine the energy efficiency as a function of performance per Watt of Condor.

The energy efficiency of Condor was determined using the Green500 test methodology, in particular measuring power consumption during maximum performance on the High Performance LINPACK (HPL) Benchmark. The HPL Benchmark measures computing performance in floating point operations per second while solving random dense linear equations. A power meter was used to measure the average energy consumption of a single node of the system over the duration of the execution time of the benchmark. Using the energy consumption from a single node and assuming each node to draw equal amounts of energy, the efficiency performance of the entire system was calculated. We demonstrate that Condor achieves an energy efficiency performance comparable to the top supercomputers on the Green500 List.

1. Introduction

The past 20 years have seen a seemingly unstoppable increase in computer performance; we have witnessed a remarkable 10,000-fold improvement in the peak performance of a high-end supercomputer (Feng and Cameron, 2007). The drive in computer advancements has been strictly performance-based, doing anything necessary to achieve a maximum number of floating-point operations per second (FLOPS). What has not been heavily considered throughout these advancements; however, is the energy efficiency of the supercomputer. Green computing takes a new view of high performance computing by considering the energy consumption required to achieve maximum performance goals (Feng, et al., 2008).

The drive for energy efficient computing has been increasingly present in the past several years for a number of reasons. We continue to observe an immense increase in the peak performance of computers on the TOP500 list over the years. While this speed-up is a great feat, the cost to run these powerful computers is an unavoidable roadblock for sustainability in terms of total cost of ownership. Consider that the price of electrical energy per megawatt is estimated to be approximately $1 million per year (Feng, et al., 2008). According to the TOP500 list of November 2010, the top-performing supercomputer in the world used 4.04 MW of power; and this system was not the most power-hungry on the list (http://www.top500.org/list/2010/22/100).

Since 2006, there has been an evident drive for energy efficient computing by the US Government. In December of 2006, the US Congress passed Public Law 109-431 "to study and promote the use of energy efficient computer servers in the United States" (http://energystar.gov). The law emphasizes the need for energy efficient improvements for government and commercial servers and data centers, and required a study be done by the Environmental Protection Agency (EPA) Energy Star Program to analyze the areas of potential impacts in energy efficiency improvements, as well as recommendations for

incentive programs to advance the transition to energy efficient computing. The EPA Energy Star program submitted the "Report to Congress on Server and Data Center Energy Efficiency" in 2007 where energy use and cost for data centers in the US was extensively examined, and prospective areas of improvement were addressed (http://www.energystar.gov). In addition, AMD, Dell, IBM, Sun Microsystems, and VMware formed the Green Grid consortium in 2007. The mission of the Green Grid is to improve the energy efficiency of data servers and computer ecosystems (Kurp, 2008).

The Green500 List was started in April 2005 to encourage energy efficiency as a first-class design consideration in emerging supercomputer construction, and to provide a ranking of the top-performing supercomputers with respect to an energy efficiency metric (www.green500.org). Similar to the well-known TOP500 List that ranks high performance computers based on peak performance, the Green500 list measures the peak performance of a system running the High Performance LINPACK (HPL) benchmark, while also measuring the energy consumed to achieve such performance. Supercomputers are ranked by MegaFLOPS (MFLOPS) per Watt, with the minimum criteria to be accepted on the Green500 List, being that the supercomputer must achieve HPL performance great enough to appear on the most recent TOP500 list.

With energy efficiency in mind, the US Air Force Research Laboratory's Information Directorate engineered the *Condor* Cluster, a heterogeneous supercomputer composed of Intel Xeon X5650 processors, Cell Broadband Engine (Cell BE) processors, and NVIDIA general-purpose graphical processing units. This project was funded with a High Performance Computing Modernization Program (HPCMP) Dedicated HPC Project Investment (DHPI), and has a theoretical single-precision peak performance of 500 TeraFLOPS (TFLOPS). This paper examines the energy efficiency of *Condor* using the Run Rules for the Green500 List, to demonstrate the total cost of ownership efficiency of this unique system design.

2. The *Condor* Cluster

The *Condor* Cluster is a heterogeneous supercomputer composed of 94 NVIDIA Tesla C2050's, 62 NVIDIA Tesla C1060's, 78 Intel Xeon X5650 dual-socket processors, and 1,716 Sony PlayStation 3s (PS3s), adding up to a total of 69,940 cores and a theoretical single-precision peak performance of 500 TFLOPS. There are 84 subcluster head nodes, of which six are gateway nodes that do not perform computations, while the other 78 compute head nodes are capable of 230 TFLOPS of theoretical peak processing performance. Each of the 78 compute head nodes are composed of two NVIDIA general-purpose graphical processing units (GPGPUs) and one Intel Xeon X5650 dual-socket hexa-core processor (i.e.,12 cores per Xeon). Of the 78 compute head nodes, 47 contain dual NVIDIA Tesla C2050 GPGPUs while 31 contain dual NVIDIA Tesla C1060 GPGPUs. The head nodes are connected to each other via 40 Gbps InfiniBand and 10Gb Ethernet. Additionally, each compute node is connected to a 10GbE/1GbE aggregator that provides communication to a subcluster of 22 PS3s. In total, the PS3s can achieve a theoretical peak performance of 270 TFLOPS.

The NVIDIA GPGPUs in *Condor*, the Tesla C1060 and the newer model Tesla C2050, share similar architectures but vary in performance. Both the C1060 and C2050 have the same Tesla architecture based on a scalable processor array (Lindholm, 2008). The architecture can be broken down into independent processing units called texture/processor clusters (TPCs). The TPCs are made up of streaming multiprocessors which perform the calculations for the GPGPU. The streaming multiprocessors can be broken down further into streaming processors or cores; these are the main units of the architecture (Maciol, 2008). The GPGPU communicates with the CPU via the host interface (Lindholm, 2008). However, the C1060 model has 240 cores while the C2050 has 448 cores (http://www.nvidia.com).

The Intel Xeon processors on each head node are built on the energy efficient Intel Nehalem microarchitecture. This architecture was made with several Intel technologies that adjust performance and power usage based on application needs. When not in use, the processor is capable of drawing a minimal amount of power and also capable of operating above the rated frequency when necessary. The *Condor* Cluster is equipped with the six-core Intel Xeon dual-socket X5650, giving a total of 12 cores per processor (http://www.intel.com).

The Sony Toshiba IBM (STI) Cell BE is a nine-core heterogeneous processor that consists of one PowerPC Processing Element (PPE) and eight Synergistic Processing Elements (SPEs). The PPE is based on the open-source IBM Power Architecture processor, and is responsible for controlling and coordinating the SPE tasks and runs the operating systems on the processor (Buttari, et al., 2007). The eight SPEs are responsible for the majority of the compute power on the processor (Gschwind, et al., 2007). All code executed by the SPE is done in the 256KB software-controlled local store (Buttari, et al., 2007). The SPEs consist of a Synergistic Processing Unit (SPU) and Memory Flow Controller (MFC). The MFC transfers data between the SPE cores, as well as between the local store and the system memory (Gschwind, et al., 2007). Connection from PPE to SPEs is made via the Element Interconnect Bus (EIB), which has a peak bandwidth of 204.8GB/s (Buttari, et al., 2007).

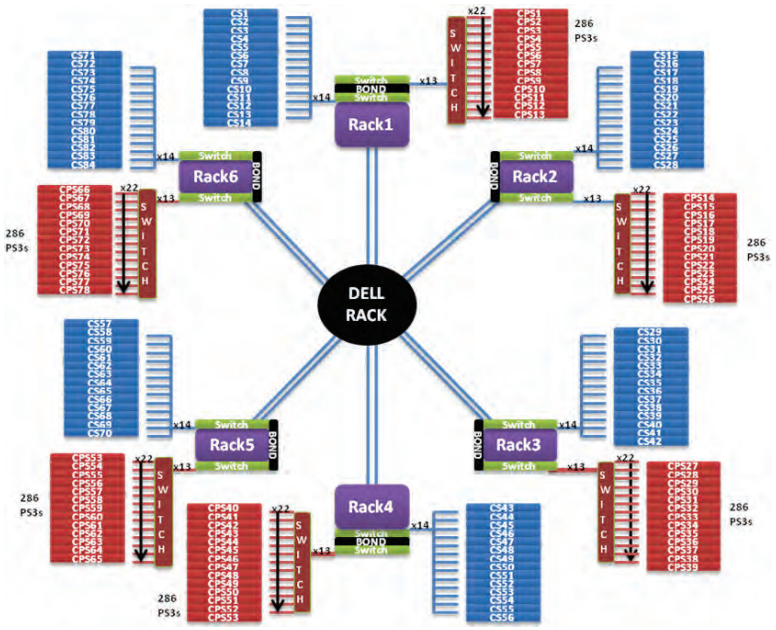


Figure 1. Condor Cluster Configuration Diagram. The Condor Cluster is arranged in a fat tree topology, with six gateway head nodes each providing access to a rack of 13 compute head nodes. Each head node is capable of accessing 40 Gbps InfiniBand and dual 10 Gb Ethernet links. In addition, each head node is connected via a 10 GbE/1 GbE aggregator to provide communication to a subcluster of 22 Sony PlayStation 3s. The InfiniBand mesh network is providing single-hop connection between head nodes, while the 10 GbE is a dual-hop connection providing linkage between all PS3s.

Condor utilizes the PS3 as a computing platform for access to the Cell BE. The PS3 is equipped with the Cell BE with minor alterations. Only six of the eight SPEs are available for use in the PS3; one SPE is disabled for yield reasons at the hardware level, and one SPE is reserved solely for the GameOS (Buttari, et al., 2007). For use in the *Condor* Cluster, CentOS Linux was installed on the PS3s. Additionally, of the total 256MB of available memory for the Cell Broadband Engine, only 200MB is accessible to Linux (Buttari, et al., 2007).

The *Condor* Cluster was engineered to increase the combat effectiveness of the Department of Defense (DoD) through technological advances supported by high-performance computing. Next-generation synthetic aperture radar (SAR) sensors strive to provide surveillance of larger areas (30 km diameter) with smaller targets at resolutions close to one foot. Applications such as this demand real-time processing of over 200 sustained TFLOPS. This surveillance capability can be achieved using the SAR backprojection algorithm, a computationally-intensive algorithm that enables every pixel to focus on a different elevation to match the contour of the scene. The SAR backprojection algorithm has been optimized by the AFRL Information Directorate to eliminate nearly all double- precision operations, favoring application on the Cell Broadband Engine. NVIDIA Tesla GPGPU cards also have a preference for single-precision operations, which critically enhances the algorithm and consequently the number of pixels generated for a 30 km surveillance circle.

3. High Performance LINPACK

The LINPACK benchmark has become the de facto standard for measuring real peak computational performance of high-performance computers for nearly twenty years. HPL introduced the ability to address scalability in the LINPACK testing environment, in order to accurately measure the performance of larger, parallel distributed memory systems. Since 1993, HPL has been used to formulate the TOP500 list of the most powerful supercomputers in the world (Dongarra, et al., 2001).

HPL provides an implementation of the LU decomposition for solving a system of equations. The benchmark includes the ability to measure the accuracy of the solution, as well as the time required to compute it. In addition, HPL requires the use of the Message Passing Interface (MPI) for providing inter-process communication, and an implementation of the Basic Linear Algebra Subprograms (BLAS) for the linear algebra operations library.

Because of the general acceptance of HPL as the standard measure of computational performance, Feng, et. al. chose to adopt the benchmark to provide the FLOPS metric for scalable system performance as it relates to energy efficiency (Feng and Cameron, 2007).

3.1 HPL CUDA

Fatica (2009) describes an implementation of HPL for NVIDIA Tesla series graphics processing units (GPUs). The approach described utilizes the CUBLAS library for the BLAS implementation, and requires only minor modifications to the HPL code. In particular, the implementation utilizes the GPU as a co-processor to the CPU, executing the benchmark simultaneously on both architectures. Thus, a critical component to achieving maximum performance is to find the optimum division of processing load between the CPU and GPU.

The only modification to the HPL source code required to enable execution on the Tesla series GPUs was changing memory allocation calls to *cudaMallocHost* calls. Subsequent acceleration of the benchmark is achieved by intercepting calls to DGEMM and DTRSM to utilize the CUBLAS library routines. Fatica's implementation exploits the independence of DGEMM operations by overlapping them on the CPU and GPU.

We used CUDA 3.2 and Open MPI 1.4.3 to execute the implementation of HPL on *Condor's* GPGPU compute nodes.

3.2 HPL Cell Broadband Engine Architecture

To execute HPL on the Cell BE of the PS3 we used a modified implementation of the one described by Kistler, et al. (2009). The approach described was targeted for the IBM BladeCenter QS22, with two IBM PowerXCell 8i processors. The PowerXCell 8i is a component of several of the top 10 computers on the Green500 List (http://www.green500.org). Our implementation has been modified to run on the Cell BE available in the PS3, a variant similar to the IBM BladeCenter QS21. As previously mentioned, the PS3 Cell only has 6 synergistic processing elements (SPEs) available for computation, as opposed to the eight SPEs available on the PowerXCell 8i. In addition, the PowerXCell 8i has an enhanced double-precision unit which the PS3 Cell does not have (Kistler, et al., 2009).

Contrary to the approach used to implement HPL for the Tesla series GPUs, Kistler, et al. implemented the benchmark through multiple kernel modifications. In particular, the most compute-intensive kernels were modified to exploit the key architectural characteristics of the PowerXCell 8i. The result was the creation of an HPL acceleration library (Kistler, et al., 2009).

We used the IBM Cell SDK 3.1 and Open MPI 1.4.3 to execute the implementation of HPL on *Condor's* PS3 nodes.

4. Test Methodology

To measure the energy efficiency of the *Condor* Cluster, we followed the Run Rules for submission to the Green500 List. This consists of two basic steps: 1) executing the HPL benchmark capable of achieving peak performance on the supercomputer and 2) measuring the energy consumption of the supercomputer while running the benchmark. It is understood that in many cases measuring the total system energy consumption is not feasible. Therefore, the Run Rules allow for measuring power at a subcomponent (e.g., 1U node, rack, etc.) and then extrapolating this measurement across the entire system (Run Rules, http://www.green500.org).

Given the uniqueness of the system and its heterogeneous nature, the HPL benchmark could not be run across the entire system at one time. Additionally, we were not able to measure the power for the entire system at a central location. Furthermore, there is a significant difference in the power draw between the PS3's and head compute nodes as well as the computational performance, particularly as a result of the memory limitations of the PS3 architecture. Therefore, in order to measure the total power consumed by the system, the supercomputer had to be broken down into three subcomponents: two PS3's, one NVIDIA C1060 compute node with two NVIDIA C1060s and one Intel Xeon processor, and one NVIDIA C2050 compute node with two NVIDIA C2050s and one Intel Xeon processor. The benchmark was executed on each of the three subcomponents and the power for each unit was measured in isolation. The total power for *Condor* was then determined using the following equation where P is power, R_{max} is the maximum performance achieved by HPL, and N is the number of units:

$$P_{total}(R_{max}) = N_{PS3} \cdot P_{PS3}(R_{maxPS3}) + N_{C1060} \cdot P_{C1060}(R_{maxC1060}) + N_{C2050} \cdot P_{C2050}(R_{maxC2050}) \quad (1)$$

We use a similar equation to Equation 1 to estimate the peak performance R_{max} of *Condor*.

Prior to obtaining the results reported below, the HPL benchmark was optimized for each of the three subcomponents. Tuning HPL to achieve the maximum performance on each subcomponent consisted of varying a selection of parameters and running several cases to observe the peak FLOPS that could be attained. Documentation on performance tuning and setting up the input data file for HPL was referenced to assist in this process. One of the most critical parameters is determining the matrix size, N, to run. This decision is largely determined by the size of RAM for the processor being tested. A listing of the parameters used for our study is seen in Table 1. In addition, we show the memory available to each subcomponent, and the percentage of memory which the matrix requires when running HPL.

Table 1. Parameters used for HPL execution (Dongarra, et al., 2001)

Subcomponent	Problem Size	Block Size	NBMIN	NDIV	Panel factorization	Recursive factorization	Broadcast	RAM	%RAM for *NxN*
SONY PS3	5440	128	4	2	R	L	Bandwidth Reducing	256MB	88%
NVIDIA C2050 compute node	51080	512	8	2	L	R	Increasing Ring	24GB	81%
NVIDIA C1060 compute node	51080	256	2	2	R	L	Increasing Ring	24GB	81%

To measure the power consumption of the subcomponents we used the "Watts Up? Pro ES" and followed the Power Measurement Tutorial by Ge, et al. (2006). The Watts Up? Pro ES is a digital power meter with a PC interface. The meter collects data in one-second intervals and stores the results in internal memory until connected to a PC. Upon completion of a set of tests the data was downloaded to the PC via USB for recording and processing power data; Watts Up? Download Software was used to collect the data from the device.

The same method was used for capturing the power consumption of each subcomponent. Prior to powering on and executing HPL, the subcomponent power cord was connected to the power meter, which was subsequently connected to the on-rack power strip. The only difference was for the PS3s, in which we connect two PS3s to a power strip and then connected the power strip to the meter. Two PS3s were monitored because the HPL implementation used was written for a QS22 containing two Cell BE processors. Each subcomponent was then powered on and allowed to run for approximately 15 minutes. This allowed the computers to stabilize and to get accurate readings of the average idle power consumption of each subcomponent. After the stabilization period, we executed the HPL code for each particular subcomponent using the parameters determined above for achieving maximum performance. Though the Green500 Run Rules state that it is sufficient to measure power consumption for a minimum of 20% of the HPL runtime, we measured consumption over the entire run. In addition, the Run Rules state that only two runs are necessary—given a tolerance of less than 1% in power variation between the two—yet we chose to run these tests 10 and 20 times for the Tesla GPUs and PS3s, respectively.

5. Results

The results presented below show the energy efficiency performance of *Condor* at the subcomponent level. We present the energy consumption of the subcomponent running HPL versus the average idle consumption, and calculate the energy efficiency in GFLOPS/W using the peak performance achieved on HPL.

Over the course of 20 runs on two PS3 nodes, the average power consumption showed little variation while executing the HPL benchmark. The average power draw for two PS3s while running the benchmark at peak performance was observed to be 199.95 W. As compared to the power draw while idle, the increase in the amount of power required to execute the peak performance of the HPL benchmark is very low, as shown in Figure 2. This demonstrates the efficiency of the PS3 while running computationally-intensive problems.

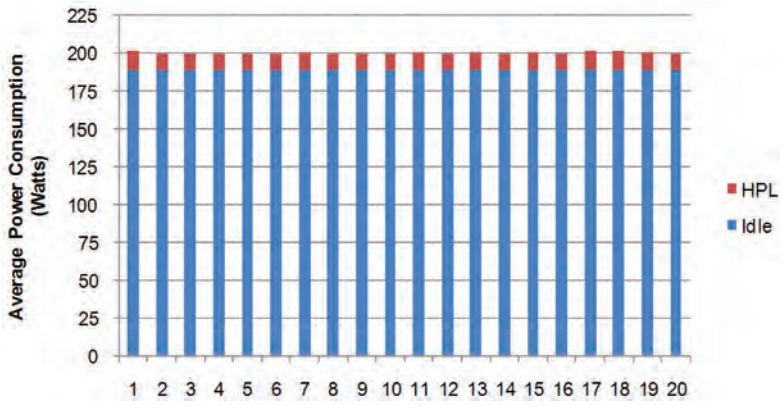


Figure 2. Power consumption of two PlayStation 3 nodes executing the HPL benchmark. When idle, the two PS3s consume 188.49 W on average. At peak HPL performance, the nodes draw an average of 199.95 W, an additional load of approximately 5.73 W per node.

Figure 3 shows the results of each run on the PS3, in terms of GigaFLOPS (GFLOPS) achieved and the average power consumption over the entire run. There is an apparent relationship between the peak performance that is achieved and the power consumed by the nodes. In most cases, slightly higher power consumption was witnessed when the performance was greater. A similar relationship was observed on each of the subcomponents tested.

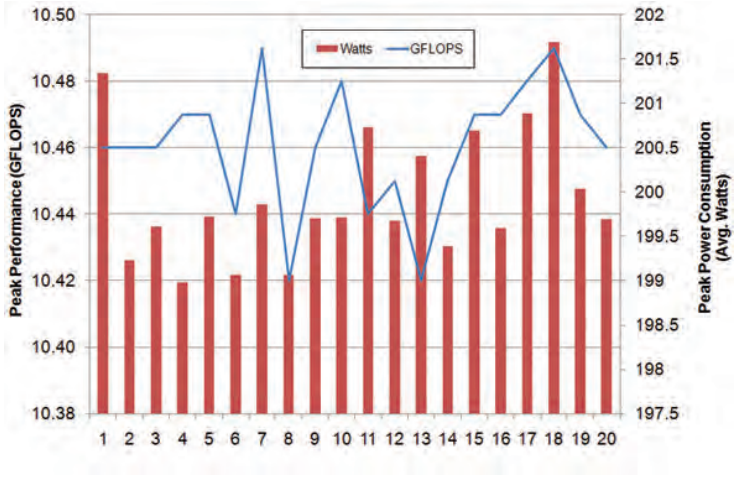


Figure 3. Performance of the HPL benchmark on two PlayStation 3 nodes. Peak performance measured as output from HPL, while power consumption is measured as the average over the duration of the HPL execution.

The experimental average peak performance of the PS3s was determined to be 10.46 GFLOPS. Thus, at an average rate of 199.95 W consumed, the energy efficiency for the PS3s can be calculated as .052 GFLOPS/W (52 MFLOPS/W). Such a rating would be sufficient to place the PS3 nodes in the 20th percentile of the November 2010 Green 500 List.

For comparison, the theoretical peak performance of a single PS3 node is 10.97 GFLOPS. Thus, the peak performance for two PS3s is 21.9 GFLOPS. Experimentally, we achieved 48% of peak performance for the HPL benchmark. However, we expected this poor performance because the PS3 Cell BE is not optimized for double- precision computation. On the other hand, a single PS3 node could achieve 153 GFLOPS in single-precision.

The NVIDIA C2050 compute nodes demonstrated higher power consumption, particularly when compared to consumption over idle use, but also showed significant improvements in HPL performance. Figure 4 shows that the average idle power consumption of the NVIDIA C2050 compute nodes is 368 W. When operating at peak performance, we observed that the nodes consumed 639 W on average. This represents a 73% increase in consumption.

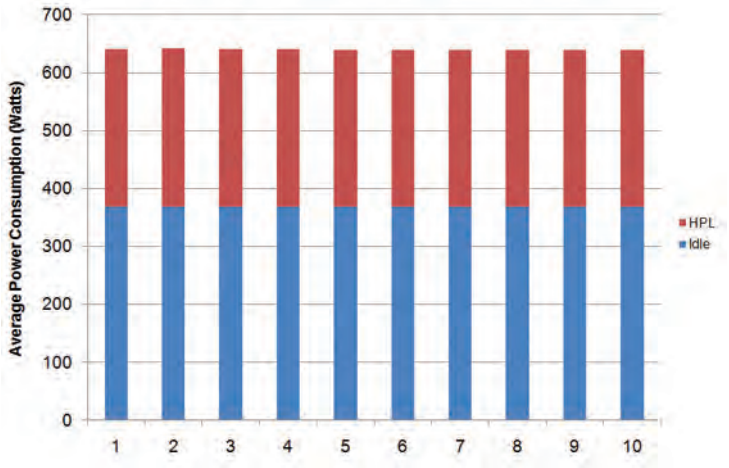


Figure 4. Power consumption of a compute node with dual NVIDIA C2050 GPUs executing the HPL benchmark. When idle, the node consumes 368.991 W on average. At peak HPL performance, the node draws an average of 639.59 W, an additional load of approximately 270.6 W.

Figure 5 shows the results of each run on the C2050 compute. The experimental average peak performance for the C2050 compute node was observed to be 619.5 GFLOPS, which equates to 54% of the theoretical 1.158 TFLOPS for these nodes (i.e., 128 GFLOPS for the Intel processor and 515 GFLOPS per NVIDIA C2050). The energy efficiency for the C2050 compute nodes can be calculated as .966 GFLOPS/W (966 MFLOPS/W). This efficiency would place the C2050 compute nodes in the 99th percentile of the November 2010 Green500 List.

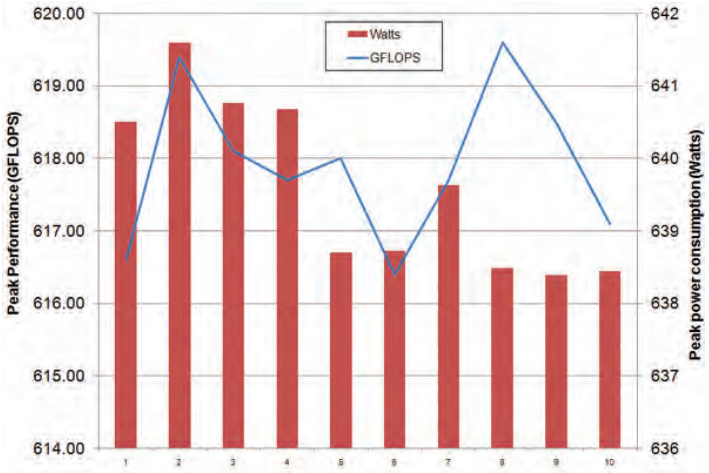


Figure 5. Performance of the HPL benchmark on a compute node with dual NVIDIA C2050 GPUs. Peak performance measured as output from HPL, while power consumption is measured as the average over the duration of the HPL execution.

The NVIDIA C1060 compute nodes demonstrated lesser power consumption to the C2050 compute nodes. Figure 6 shows the average consumption of the C1060 compute nodes when idle, as compared to the average consumption for each run of HPL. The average idle power consumption of the NVIDIA C1060 compute nodes is 337 W. When operating at peak performance, we observed that the nodes consumed 506 W on average. This represents an approximate 50% increase over idle performance.

However, unlike the C2050 compute nodes, the C1060 nodes are not fully-optimized for double-precision computations. In particular, it is the C1060 which does not perform optimally, as the Intel processors are the same as those on the C2050 nodes. The theoretical peak performance of a C1060 for single-precision is 933 GFLOPS. However, the theoretical peak performance for double-precision is 78 GFLOPS (http://www.nvidia.com). Conversely, the C2050 performs at 1.3 TFLOPS in single-precision and 515 GFLOPS for double-precision (http://www.nvidia.com). As a result, we observed much lower performance on the C1060 compute nodes at an average of 118 GFLOPS, or 42% of the peak.

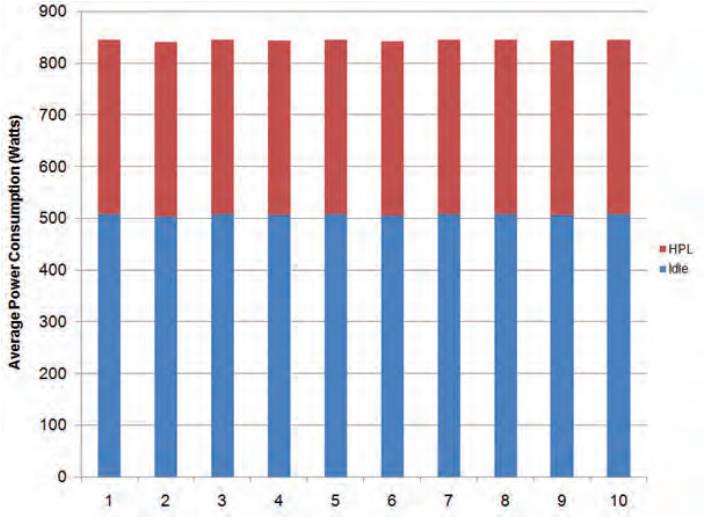


Figure 6. Power consumption of a compute node with dual NVIDIA C1060 GPUs executing the HPL benchmark. When idle, the node consumes 336.94 W on average. At peak HPL performance, the node draws an average of 506.85 W, an additional load of approximately 169.85 W.

Figure 7 shows the results of each run on the C1060 compute. With an average performance of 118 GFLOPS and average power consumption of 506 W, the energy efficiency for the C1060 compute nodes can be calculated as .223 GFLOPS/W (223 MFLOPS/W). This efficiency would place the C1060 compute nodes in the 75th percentile of the November 2010 Green500 List.

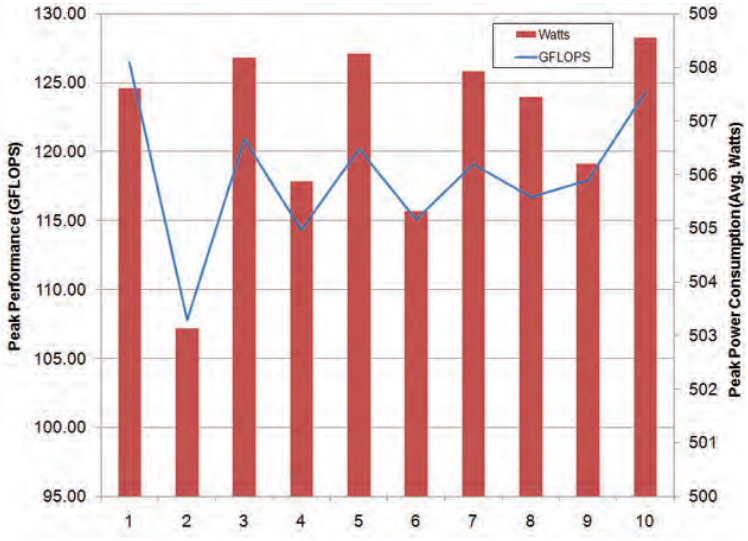


Figure 7. Performance of the HPL benchmark on a compute node with dual NVIDIA C1060 GPUs. Peak performance measured as output from HPL, while power consumption is measured as the average over the duration of the HPL execution.

Our results across all three node classes are shown in Table 2. Using Equation 1, we can calculate the overall energy efficiency of *Condor* to be approximately .192 GFLOPS/W (192 MFLOPS/W). This rating reflects 41.7 TFLOPS of double-precision performance and 217.3 KW of consumed power.

Table 2. Observed energy efficiency of *Condor* by subcomponent

Subcomponent	# of Nodes	Avg Watts Per Node	GFLOPS Per Node	GFLOPS/W
SONY Playstation 3	1716	99.98	5.23	.052
NVIDIA C2050 compute node	47	639.59	619.5	.966
NVIDIA C1060 compute node	31	506.85	118.3	.233

While our method for measuring the average power consumption of the nodes is consistent with the methodology prescribed by the Green500, we realize that isolation of a single node for running HPL and then extrapolating the results across the entire supercomputer is not consistent with the TOP500 run rules. Parallelization of the benchmark across the entire supercomputer would introduce degradations on the overall performance, e.g., due to communication and coordination between the nodes. What we present here can thus be described as an experimentally-rooted theoretical maximum for the energy efficiency performance of *Condor*. In practice, we would expect the overall peak performance of HPL to drop slightly when utilizing the full cluster.

6. Conclusion and Future Work

In a time where the drive for advancing computer systems has been dominated by peak performance at any cost, the Green500 List challenges emerging developers to examine another key aspect to advanced computing; namely, energy efficiency. Not only has the cost to operate top-of-the line supercomputers soared beyond a million dollars per year, but the excessive power consumption of these emerging supercomputers is negatively impacting the environment, making energy efficiency a necessity in system design. The Green500 List provides a ranking system where the performance per Watt metric has not only taken precedence over other metrics, but has been encouraged as a primary consideration in new designs.

We demonstrated here that the *Condor* Cluster is capable of achieving energy efficiency performance that would place in the top 35% of the most recent Green500 list (http://www.green500.org). However, the computational performance is limited with respect to HPL because the Cell BE and NVIDIA C1060 are not optimized for double-precision floating-point operations.

However, for the majority of the applications run on the *Condor* Cluster single-precision operation is sufficient; as such, the design model for the supercomputer was not intended to achieve extraordinary double-precision performance. We consider exploration of mixed-precision approaches to HPL (Kurzak & Dongarra, 2006) or other single-precision benchmarks as an area of future research to demonstrate the efficiency of *Condor* in its targeted niche of computation.

A key design concept of *Condor* was to bring the three critical drivers in supercomputer design—peak performance, price/performance, and performance/Watt—together into a unique and highly-sustainable system capable of solving some of the military's most critical information processing problems. High performance computing systems are designed to achieve a peak performance based on their desired applications. *Condor* is capable of sustaining a peak performance of 200–300 TFLOPS required to perform several important military applications. While the cost of engineering a high-performing supercomputer can be very expensive, *Condor* was built using commodity game consoles and graphics processors that achieve performance comparable to specialized architectures at a fraction of the cost. With a total cost of $2.5M, the price/performance ratio far exceeds that of comparable systems. Finally, we have demonstrated the energy efficiency of *Condor* to be 0.192 GFLOPS/W. The energy needs of *Condor* can be translated into sustainability costs on the order of $0.5M per year. Thus, the *Condor* Cluster is a powerful, yet highly sustainable asset for the US Air Force Research Laboratory (AFRL) and the Department of Defense (DoD).

Acknowledgements

The authors would like to acknowledge Qing Wu and Gaurav Khanna for their assistance in getting the HPL implementations running on Condor, and to NVIDIA for providing the Tesla implementation.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL.

References

Board Specification Tesla C1060 Computing Processor Board, NVIDIA Corporation, January 2009, http://www.nvidia.com/docs/IO/56483/Tesla_C1060_boardSpec_v03.pdf.

Board Specfications Tesla C2050 and Tesla C2070 Computing Processor Board, NIVIDIA Corporation, July 2010, http://www.nvidia.com/docs/IO/43395/BD-04983-001_v03.pdf.

Buttari, A., J. Dongarra, and J. Kurzak, "Limitations of the PlayStation 3 for High Performance Cluster Computing", *LAPACK Working Note 185*, 2007.

Dongarra, J., P. Luszczek, and A. Petit, "The LINPACK Benchmark: Past, Present, and Future", *Concurrency and Computation: Practice and Experience*, 15, 9, pp. 803–820, 2003.

Fatica, M., "Accelerating Linpack with CUDA on heterogeneous clusters", *GPGPU '09*, Washington D.C. 2009.

Feng, W. and K. Cameron, "The Green500 List: Encouraging Sustainable Supercomputing", *Computer*, 40, 12, pp. 50–55, 2007.

Feng, W., X. Feng, and R. Ge, "Green Supercomputing Comes of Age.' *IT Professional*, 10, 1, pp. 17–23, 2008.

Ge, R., H. Pyla, K. Cameron, and W. Feng, "Power Measurement Tutorial for the Green500 List", http://www.green500.org/docs/pubs/tutorial.pdf, 2007.

Gschwind, M., D. Erb, S. Manning, and M. Nutter, "An Open-Source Environment for Cell Broadband Engine System Software", *Computer*, 40, 6, pp. 37–37, 2007.

Intel Xeon Processor 5600 Series: The Next Generation of Intelligent Server Processors, Intel Corporation, 2010, http://www.intel.com/assets/PDF/prodbrief/323501.pdf.

Kistler, M., J. Gunnels, D. Brokenshire, and B. Brenton, "Programing the Linpack benchmark for the IBM PowerCXell 8i processor", Scientific Programming, 17, pp. 43–47, 2009.

Kurp, P., "Green Computing: Are you ready for a personal energy meter?", *Communications of the ACM*, 51, 10, pp. 11–13, 2008.

Kurzak, J., and J. Dongarra, "Implementation of the Mixed-Precision High-Performance LINPACK on the CELL Processor", *LAPACK Working Note 177*, 2006.

Lindholm, E., J. Nickolls, S. Obermand, and J. Montrym, "NVIDIA Tesla: A Unified Graphics and Comptuing Architecture", *IEEE Micro*, 28, 2, pp. 39–55, 2008.

Maciol, P. and K. Banaś, "Testing Tesla Architecture for Scientific Computing: the Performance of Matrix-Vector Product", *Proceedings of the International Multiconference on Computer Science and Information Technology*, pp. 285–291, 2008.

Public Law 109-431, (120 Stat. 2920, Date: 20 December 2006) 109th Congress, http://www.energystar.gov/ia/products/downloads/Public_Law109-431.pdf.

Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431, U.S. Environmental Protection Agency ENERGY STAR Program, 2007, http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf.

"Run Rules for the Green500: Power Measurement of Supercomputers Version 0.9", http://www.green500.org/docs/pubs/RunRules_Ver0.9.pdf.

The Green500 List, http://www.green500.org.

Top500 Supercomputing Sites, http://www.top500.org.

Matrix Multiplication using Virtex-4 FPGAs on SGI RASC RC100 at the Naval Research Laboratory

Stephen Bique and Robert Rosenberg
US Naval Research Laboratory (NRL-DC), Washington, DC
{stephen.bique, robert.rosenberg}@nrl.navy.mil

Abstract

Matrix multiplication is a common operation for which the RISC architecture is well-designed, and for which good performance is achieved using established libraries. This paper describes and analyzes different techniques to implement matrix multiplication on Virtex-4 FPGAs on the SGI RASC RC100 at the US Naval Research Laboratory, which is significant because of the advantages of low-powered hardware. Several implementations written in Mitrion-C are described. Performance results are reported to compare these different implementations.

1. Introduction

Given a real $m \times n$ left-hand matrix $A=(a_{i,j})$and an $n \times p$ right-hand matrix $B=(b_{i,j})$, the matrix product AB is an $m \times p$ matrix $C=(c_{i,j})$whose i, j-th entry for $i=1,2,\ldots, m$, and $j=1,2,\ldots, p$ is:

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j}.$$

The task is to compute many matrix products assuming a single-process holds the data before and after the calculations. This problem contrasts sharply with an embarrassingly parallel one, in which each process uses data obtained independently, such as random inputs in a simulation. On the other hand, the basic computation permits a high-degree of parallelism since each element of a matrix product is an inner-product that maybe computed independently.

The chosen programming environment consists of the Mitrion-C programming language and Xilinx synthesis software. More specifically, the available tools include Mitrion SDK 2.0 and Xilinx ISE Design Suite 10.1[10,13,14]. The target devices are the thirty-two Xilinx Virtex 4 LX200 FPGAs (Field-Programmable Gate Arrays) on the SGI reconfigurable application-specific computing platform (RASC) RC100, one of the high performance computing (HPC) resources at the US Naval Research Laboratory (NRL).

An evaluation of the SGI RASC RC100 in this programming environment, based on an application involving floating-point arithmetic, is given in Reference 6. A programmer needs to be aware of various hardware and software details to develop efficient code. An overview of reconfigurable supercomputing using FPGAs, which discusses the technologies, tools, languages and includes examples, recently appeared[7].

An implementation on a Xilinx XUP development board with 256MB of DRAM demonstrates slightly faster performance using their hardware-versus-software pipelined design for sufficiently-large matrices[2]. Their performance results should improve using more-current hardware technology. Their design in Bluespec SystemVerilog (BSV) was based on $O(n^3)$ algorithms.

Consider square $n \times n$ matrices. The time-complexity using sequential analysis and brute force, which is also known as the naive, trivial or cubic method, is $O(n^3)$ as there are $O(n^2)$ elements in the result and each of them requires $O(n)$ additions. In a parallel setting, the time-complexity is $O(\log n)$ using an excessive number $O(n^3)$ processors to perform all of the multiplications in parallel and concurrently combining all of the terms in the summations using a binary tree algorithm.

Compared to the cubic method, there exists slightly faster and more complicated algorithms such as Strassen's $O(n^{2.807})$ method published in 1969, which is not as accurate as other methods[5]. Such complex designs as Strassen's algorithm are typically avoided on embedded systems, as routing them is impractical. The Coppersmith-Winograd algorithm published in 1987 is $O(n^{2.38})$ but the matrix order must be astronomically-large so that this algorithm is not useful for numerical

computations[11]. Note the optimal time-complexity cannot be better than $O(n^2)$, as there are $O(n^2)$ elements in the input matrices that must be read and processed.

Typically, local (SRAM) memory banks are used for input/output (I/O), which is especially true when designing applications using Mitrion tools on the target FPGAs. An important limitation for an FPGA or any other accelerator is the number of available ports for I/O, which is typically limited to a couple ports per local memory bank. Hence, an accelerator reads and writes a large matrix in $O(n^2)$ time. To realize more significant speed-up using an FPGA, it is necessary to add more ports to the local memories or (even better) avoid SRAMs and add more channels to faster external memory, as time spent doing computations needs to be balanced with time spent on I/O operations. The HC-1 system developed by Convey avoids the SRAM bottleneck.

I/O is an important consideration in designing an application. Each Virtex 4 LX200 FPGA on the SGI RASC RC100 can access five 8MB SRAM memory banks[9]. These "local" (external) memory banks can be accessed using different access modes. Typically, pairs of physical banks, each with 8MB of memory allowing access to 1,048,576 64-bit words, are combined into logical banks with 16MB allowing access to 1,048,576 128-bit words. This means during any clock-cycle, it is possible to perform the same number of accesses of 128-bit words in a logical bank as 64-bit words in a physical bank. This fact follows, since a logical bank provides exactly twice the number of ports in a physical bank. In particular, it is possible to access two 128-bit words in a logical bank during any clock-cycle.

In a Mitrion-C program, it is possible to define either four 64-bit banks, or two combined 128-bit banks, and optionally one 64-bit bank. The "extra" 64-bit bank can be accessed only by the coprocessor (not the host)[9]. It is also possible to use streams for I/O. Using streams bypasses the local memory banks. In practice, streams are not useful whenever the program structure is iterative involving blocks of data such as matrices.

Increasingly, floating-point computations on an FPGA are practical as technology advances[8]. A Xilinx Virtex II Pro FPGA implementation based on a block-matrix multiplication algorithm using 64-bit floating-point numbers demonstrates relatively good performance[4]. Their design was done using Xilinx ISE 6.0. The scope of this study is limited to floating-point calculations. In particular, single-precision numbers are used. The primary reason for using single-precision numbers is that more numbers can be read and written to memory during a clock-cycle. If more ports were available on the local memories of the FPGA, then it would be possible to use double-precision while performing the same memory accesses per clock-cycle. Hence, each computer word holds multiple matrix elements.

Ideally, the sizes of the matrices should be sufficiently large to enhance opportunities for parallelism. In practice, multiplying large matrices using single-precision involves accumulation and propagation of round-off errors, which is beyond the scope of this investigation. When comparing the results obtained using two different implementations (for example, using two different algorithms), the primary interest is performance, assuming both implementations always produce correct results. To avoid issues related to errors and simplify verification, the size of the matrices and the magnitudes of the numbers are limited by trial-and-error to avoid errors. Thus, if a new implementation consistently produces exactly the same results on random inputs, we can be more confident the implementation is correctly designed and coded.

We investigate "multi-buffering" which is also referred to as "double-buffering" and "streaming." Multi-buffering works automatically, provided the FPGA is configured to use at most half of each memory bank, has only read-access to one of the banks and write-access to the other bank. These requirements actually allow the host to utilize more memory than physically exists, even though the FPGA is configured to have less memory. The other half of the memory banks are used as buffers.

Using multi-buffering, data transfers take place and the coprocessor is restarted automatically whenever the data written does not fit into the local memory of the FPGA. Thus, the overhead of using FPGAs is reduced by overlapping data processing and data transfers. The biggest overhead is the time it takes to load the logic, but this cost is a one-time charge.

The Basic Linear Algebra Subprograms (BLAS) are portable, efficient and widely-used routines for performing vector and matrix operations[3]. A comparison of BLAS Level 2 on CPU, FPGA, and processing unit (GPU) demonstrates similar performance and higher energy-efficiency on FPGA[12]. CPU implementations using BLAS routines, Message Passing Interface (MPI) processes and OpenMP threads are also run on the SGI RC100. Performance results are reported to see the relative performance of these implementations on the same system.

2. Implementations

Matrix multiplication is implemented on FPGAs using various computational patterns and I/O schemes. To optimize a design, it is usually necessary to fix various parameters such as the size of blocks for the case of block-matrix multiplication. For different input sizes, different techniques may yield better performance. Finding an optimal implementation usually requires some good guesswork based on the given architecture as an exhaustive study is impractical.

A. Block-Matrix Multiplication

Several implementations are discussed briefly. In every implementation, each full-matrix product is computed iteratively by applying a function to multiply a block of rows from the left-hand matrix A by a block of columns from the right-hand matrix B to compute the corresponding block of the resulting matrix C as shown in Figure 7.

Blocks are used because there are insufficient resources on the FPGA to operate on large matrices. Other types of blocks, such as segments of rows and columns are not investigated to avoid saving and combining such blocks. In particular, the algorithm cannot be easily decomposed and mapped to this particular device using shorter blocks in such a way that enhances parallelism in view of I/O considerations, an important topic that is addressed later. For other devices, it may be advantageous to use such shorter blocks.

It is convenient to choose block sizes that evenly divide the dimensions of the matrices, and matrix sizes that evenly divide the available memory sizes. These assumptions are not serious restrictions, as zeros can always be appended to fill the matrices as desired, provided the resulting rows and columns are ignored. This strategy is recommended when developing codes and comparing relative performance of different implementations. In the final implementation, the codes may be adapted to use the desired sizes.

B. Programming an FPGA

Consider data aggregates, which are called "collections" in Mitrion-C, such as lists and vectors, and suitable program structures on these data types. Informally, a matrix is a list of lists. A function that implements the naive method (brute force) for matrix multiplication using lists is given in Figure 1.

```
matmul(matrix_a, matrix_b)
{
  matrix_c = foreach(row_a in matrix_a)
  {
    row_c = foreach(column_b in matrix_b)
        {
          FLOAT sum = (FLOAT) 0;
          element = for( a, b in
                         row_a, column_b )
          {
             sum = a * b + sum;
           } sum;
        } element;
   } row_c;
} matrix_c;
```

Figure 1. Naive method in Mitrion-C

A typical optimization technique for a traditional processor stores the transpose of the right-hand matrix to take advantage of locality of reference to reduce cache misses. It is assumed only the transpose of the right-hand matrices are stored for this naive implementation. In all other implementations described, assume the right-hand matrix is not transposed.

Upon execution of this naive implementation, elements are computed serially, although the exact order is not specified. This implementation is interesting as it involves powerful features of the language, such as a loop-dependent variable, to perform a reduction. Regardless, this function is not promising. To achieve better performance compared to the host, the coprocessor must perform many operations concurrently as it runs at a slower clock-speed. This design is especially poor, as none of the floating-point operations which take longer than one clock cycle are performed concurrently.

A "vectorized" function that performs matrix multiplication is displayed in Figure 2. The current row of the result is updated during each iteration of the for loop by multiplying an entire row of the right-hand matrix by the current element in the corresponding row of the left-hand matrix. Despite this vectorized approach, execution remains serial.

A common technique to enhance parallelism is to "unroll" the loops. The standard way to unroll a for loop in Mitrion-C is to iterate over vectors instead of lists. The innermost for loop in the preceding vectorized function is unrolled as shown in Figure 3. Although corresponding terms are now computed in parallel, the summation for each inner-product is executed sequentially. To fully unroll the inner-most loop as shown, the number of columns in the right-hand block must be small; otherwise, the logic will not fit on an FPGA chip.

```
matmul(matrix_a, matrix_b, ncols)
{
  matrix_c = foreach(row_a in matrix_a)
  {
   FLOAT< ncols > tmp_row =
      foreach(i in < 1..ncols >) (FLOAT) 0.0;
   row_c = for ( a, row_b in row_a, matrix_b)
   {
    tmp_row = foreach( b,c in row_b, tmp_row)
              c + b * a;
   } tmp_row;
  } row_c;
} matrix_c;
```

Figure 2. Vectorized matrix multiplication in Mitrion-C

```
matmul(matrix_a, matrix_b, ncols)
{
  matrix_c = foreach(row_a in matrix_a)
  {
    FLOAT[ ncols ] tmp_row_c =
      foreach(i in [ 1..ncols ]) (FLOAT) 0.0;
    row_c = for ( a,row_b in row_a, matrix_b)
    {
      tmp_row_b = reformat(row_b, [ ncols ]);
      tmp_row_c = foreach( b,c in
                           tmp_row_b, tmp_row_c)
                  c + b * a;
    } tmp_row_c;
     RowC = reformat(row_c, < ncols >);
  } RowC;
} matrix_c;
```

Figure 3. Vectorized matrix multiplication with inner-most loop unrolled

If the right-hand block is large, the for loop can be partially unrolled as shown in Figure 4. In this implementation, each segment of the vector is processed serially while all of the terms in each segment are computed in parallel. Note that the rows are processed serially in some order.

```
matmul(matrix_a, matrix_b)
{
  mat_B=reshape(matrix_b,<NROWB><ITER><UNROLL>);
  matB = reformat(mat_B, <NROWB><ITER>[UNROLL]);
  FLOAT<ITER>[UNROLL] trowC =
    foreach(dummy in < 1 .. ITER > ) {
      z = foreach(e in [1 .. UNROLL]) 0;
    } z;
  matrixC = foreach(rowA in matrix_a)
  {
    matC = for(row_B, a in matB, rowA)
    {
      trowC = foreach(colBunroll, trowc in
                      row_B, trowC ) {
        t = foreach(c, b in trowc, colBunroll)
            c + b * a;
      } t;
    } trowC;
  } matC;
  mat_C=reformat(matrixC,
                 <SLICE_A_ROWS><ITER><UNROLL>);
  matC=reshape(mat_C,
                 <SLICE_A_ROWS><SLICE_B_COLS>);
} matC;
```

Figure 4. Vectorized matrix multiplication with partial unrolling

Instead of processing rows serially as observed in Figure 4, it is also possible to process several rows in parallel by partially unrolling the outer loop. A function that involves unrolling of both inner- and outer-loops is given in Figure 5. As this implementation is more complicated, it takes much longer to place and route the design. In general, nesting for loops should be avoided to simplify placement and routing.

In the function given in Figure 5, it is possible to concurrently perform the same number of multiply-accumulate (MAC) operations by using either more rows and less columns, or less rows and more columns. It follows that by choosing block-sizes carefully (see Figure 7), it is not necessary to partially unroll more than one loop and still perform an optimal number of MAC operations per clock- cycle. A simpler function that involves partial unrolling of the outer loop is given in Figure 6.

```
matmul(matrix_a, matrix_b)
{
  mat_A = reshape(matrix_a,<ITERA><UNROLLA><NCOLA>);
  matA  = reformat(mat_A,  <ITERA>[UNROLLA]<NCOLA>);
  mat_B = reshape(matrix_b,<NROWB><ITERB><UNROLLB>);
  matB  = reformat(mat_B,  <NROWB><ITERB>[UNROLLB]);
  FLOAT<ITER>[UNROLL] trowC =
     foreach (dummy in < 1 .. ITERB > ) {
       z = foreach(e in [1 .. UNROLLB]) 0;
     } z;
  matrixC = foreach(rowAunrolled in matA)
    {
      matc = foreach(rowA in rowAunrolled)
      {
        matC = for(row_B, a in matB, rowA)
          {
            trowC = foreach( colBunroll, trowc in
                             row_B, trowC ) {
              t = foreach(c,b in trowc,colBunroll)
                  c + b * a;
            } t;
          } trowC;
      } matC;
    } matc;
  mat_C = reformat(matrixC,
                   <ITERA><UNROLLA><ITERB><UNROLLB>);
  matC = reshape(mat_C,
                   <SLICE_A_ROWS><SLICE_B_COLS>);
} matC;
```

Figure 5. Vectorized matrix multiplication with unrolling of inner- and outer-loops

```
matmul(matrix_a, matrix_b)
{
  mat_A=reshape(matrix_a,<ITER><UNROLL><NCOLA>);
  matA =reformat(mat_A,  <ITER>[UNROLL]<NCOLA>);
  FLOAT<SLICE_B_COLS> trowC =
    foreach(e in <1 .. SLICE_B_COLUMNS>) 0;
  matrixC = foreach(rowAunrolled in matA)
    {
      matC = foreach(rowA in rowAunrolled)
      {
        rowC = for(rowB, a in matrix_b, rowA)
          {
            trowC = foreach(c,b in trowC, rowB)
                    c + b * a;
          } trowC;
      } rowC;

    } matC;

  mat_C = reformat(matrixC,
                   <ITER><UNROLL><SLICE_B_COLS>);
  matC = reshape(mat_C,
                   <NROWS_A_SLICE><SLICE_B_COLS>);
} matC;
```

Figure 6. Vectorized matrix product with unrolling of outer-loop

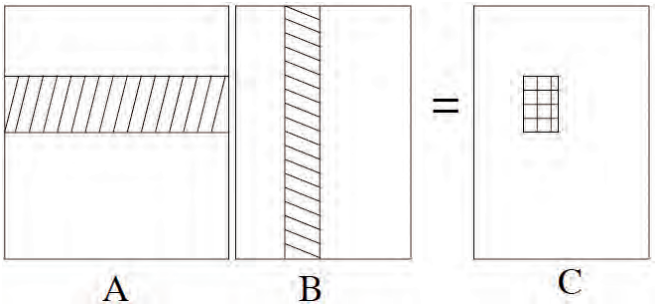


Figure 7. Block matrix multiplication *AB=C*

3. I/O Configuration

Computations take place after reading the input data and before writing the results. Ideally, the number of accesses to a memory bank at any given time equals the maximum number allowed to enhance bandwidth. Whenever reading and writing to the same memory bank, it is convenient to attempt half as many concurrent reads as possible to allow the same number of concurrent writes while avoiding deadlock situations.

For each implementation, the memory banks need to be configured and the data mapped to the local memories of the FPGA. Configuration includes defining the number of banks, their sizes, and the data-type of each word in memory. The memory map must be decided in advance in order to separately develop the code for both the host and coprocessor. Performance will be determined not only by the coding described in the preceding section, but also by the configuration of the local memories and the mapping of the data to these banks. To simplify the discussion of different memory models, consider 256×256 matrices.

A. Case I

In this case, both combined memory banks are configured to use all available memory. The first combined bank holds all of the input matrices, i.e., the combined bank is completely filled with input data. Multi-buffering is not used as more than half the available memory is used; otherwise, all FPGA accesses meet the requirements for multi-buffering. In particular, 32 left-hand matrices are stored in the first half of the first combined memory bank, and 32 right-hand matrices are stored in the second half as depicted in Figure 8. The resulting matrices are stored in the first-half of the second combined memory bank.

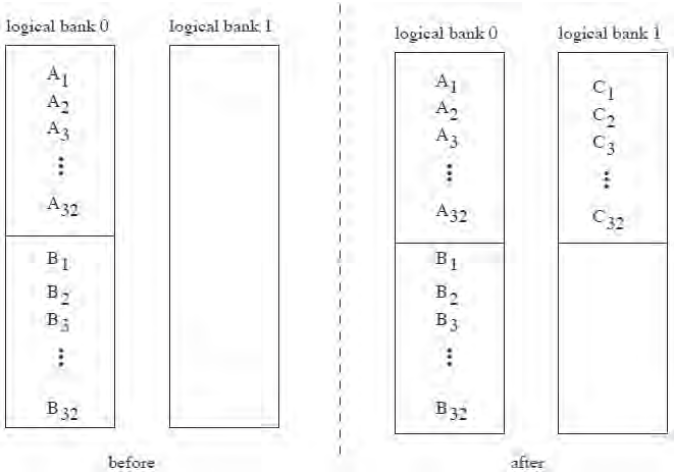


Figure 8. Memory Map for Case I

Pseudo-code for the program structure is given in Figure 9.

```
For each pair of matrices A and B,
   For each slice As, a block of rows of A,
      For each slice Bs, a block of columns of B,
         Read Bs and calculate As · Bs.
```

Figure 9. Pseudo-code for Case I

Note the slices of the left-hand matrix A are reread when successive slices of the right-hand matrix are read. Although the first logical bank is fully utilized, only half of the second logical bank is utilized.

B. Case II

As in the previous case, both combined memory banks are configured to use all available memory. The first combined bank holds all of the left-hand matrices, and the second combined bank holds all of the right-hand matrices so that each combined bank is completely filled with input data. The extra 64-bit memory bank is used to hold the results until the corresponding matrix product is computed, and then the results are copied over the right-hand matrix. As the right-hand matrix is reread many times, it cannot be overwritten until the matrix product is computed.

Multi-buffering is not used because not only more than half the available memory is used, but also the FPGA is both reading from and writing to the second combined bank. In particular, 64 left-hand matrices are stored in the first combined memory bank, and 64 right-hand matrices are stored in the second combined memory bank as depicted in Figure 10. The contents of the 64-bit memory bank are not depicted since the host does not access this bank. The resulting matrices are stored in the second combined memory bank. Unlike the preceding case, both logical banks are fully utilized so the coprocessor is able to perform twice as many matrix products during a single run.

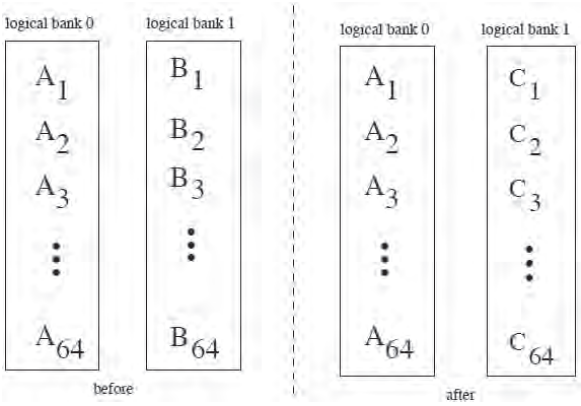


Figure 10. Memory Map for Case II

The program structure is almost identical as in the previous case, except the results are written to the physical bank instead of directly to a logical bank. Note copying requires two reads from the physical bank for each write to the combined bank, as a word in the physical bank is half the size of a word in a logical bank.

C. Case III

As in the previous cases, both combined memory banks are configured to use all available memory. The first combined bank holds all of the left-hand matrices, and the second combined bank holds all of the right-hand matrices, so that both combined banks are completely filled with input data. Instead of overwriting the right-hand matrices as in the previous case, the left-hand matrices are overwritten. Multi-buffering is not used because not only more than half the available memory is used, but also the FPGA is both reading from and writing to the first combined bank. In particular, 64 left-hand matrices are stored in first combined memory bank, and 64 right-hand matrices are stored in second combined memory bank as depicted in Figure 11. The resulting matrices are stored in the first combined memory bank.

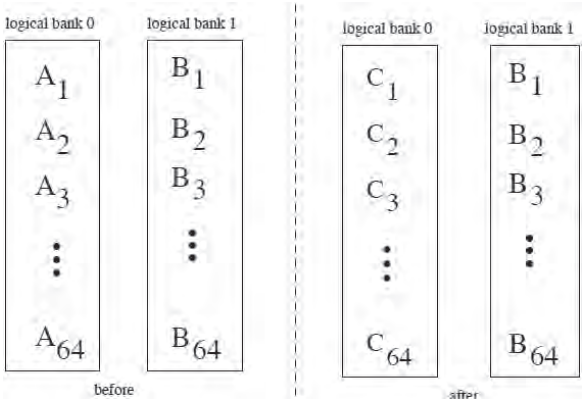


Figure 11. Memory Map for Case III

Pseudocode for the program structure is similar as in previous cases except the slices of the left-hand matrices are not reread and are overwritten. See Figure 12.

```
For each pair of matrices A and B,
   For each slice As, a block of rows of A,
      Read As
      For each slice Bs, a block of columns of B,
         Read Bs and calculate As · Bs.
```

Figure 12. Pseudo-code for Case III

D. Case IV

In this case, both combined memory banks are configured to use half of the available memory. The first combined bank holds all of the input matrices. Although the requirements for multi-buffering are met, the input data fits entirely into the configured memory of the FPGA so that the coprocessor is not restarted. In particular, 16 left-hand matrices are stored in the first-half of the first combined memory bank, and 16 right-hand matrices are stored in the second half of the first combined memory bank as depicted in Figure 13. The resulting matrices are stored in the first quarter of the second combined memory bank.

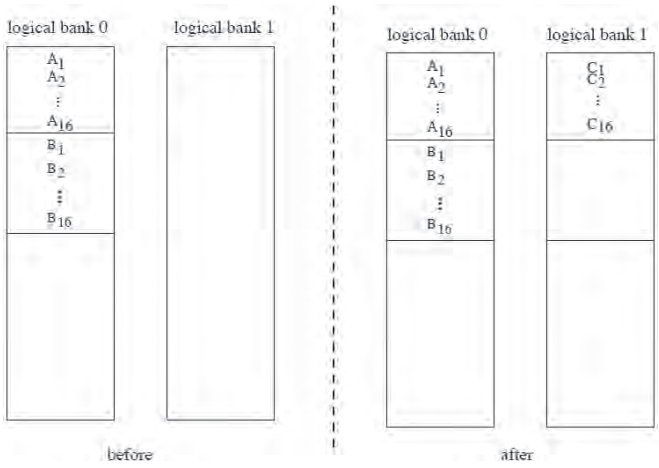


Figure 13. Memory map for Case IV

As far as programming the FPGA is concerned, the only difference between this case and Case I is the configuration of the size of local memory, so that only half as many matrix products are computed per run.

E. Case V

This case involves the same bit-stream used in the previous case, i.e., the logic loaded on the FPGA is the same. In this case, multi-buffering is used. The host writes so much data as to fill the configured memory of the FPGA multiple times. The RASC tools automatically handle the data transfers and restart the FPGA.

The host needs to set up memory before sending data, and then extract the appropriate chunks after the FPGA completes processing. The memory layout is depicted in Figure 14 when "calling" an FPGA to perform 32 matrix products, even though the configured (combined) memory holds only half the input data. For each run on the coprocessor, the corresponding segment of memory mapped to the local memory of the FPGA must contain a "block" of left-hand "A" matrices followed by a block of corresponding right-hand "B" matrices. The results received by the host processor will contain "empty" blocks that must be ignored, as the FPGA writes to only half of a memory bank.

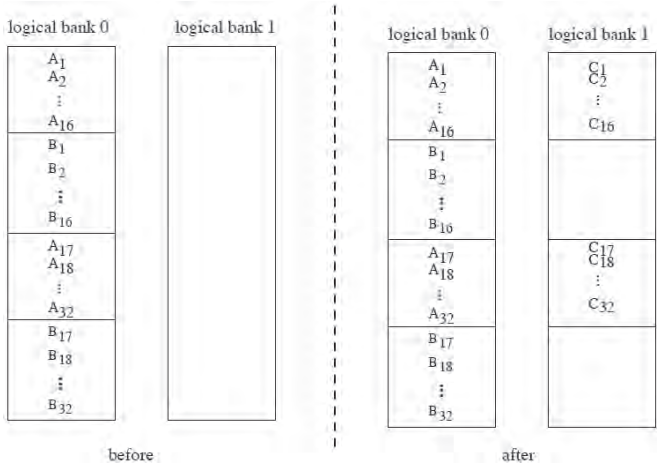


Figure 14. Memory map for Case V

4. Performance Evaluation

The matrix order (256×256) was fixed based on empirical tests. Less performance improvement was observed for smaller matrices. Larger matrices were not used, as it was more difficult to fit the logic onto the chip using the described implementations.

Even though floating-point operations are used, the (matrix) elements are randomly chosen integers in the range 0 to 255. Using such input data, the results are accurate using a variety of methods to perform the computations. We applied a fast multiplicative congruential generator (MCG) to generate random numbers[1]. In particular, the next matrix element is x_{n+1} *mod* 256, where x_{n+1} is the next pseudo-random number given by Equation 1.

$$x_{n+1} = 1425040946427180923 \cdot x_n \; mod \; 2305843009213328881 \quad (1)$$

Comparing the described implementations, the best relative performance is observed using a vectorized approach with unrolling of outer-loop (see Figure 6). Although we successfully synthesized code based on the naive method (see Figure 1), the coprocessor did not terminate upon execution. Regardless, we would not expect to see speed-up using the naive implementation due to the poor design. Using the Xilinx tools, we successfully fully placed the design involving unrolling of both inner- and outer-loops (see Figure 5); however, routing was unsuccessful, which is not surprising due to the complexity of the design.

To measure the performance, we ran codes on the RASCRC100 ten times and computed an average speed-up. Codes were synthesized using "normal" placement and routing effort (op med) unless otherwise specified. We recommend using "standard" placement and routing effort (-op std).

Consider the first four cases (I-IV). In each case, we used slices with 32 rows/columns. The results shown in Table 1 indicate Case II and III yield the best performance of these four cases. Case III is preferred as the extra memory bank is not used and slightly fewer flip-flops are needed.

Table 1. Average speed-up compared to naive method

	Case			
	I	II	II	IV
Matrix Products	32	64	64	16
Average Speedup	0.997	3.08	3.09	2.73
Standard Deviation ($\times 10^{-2}$)	1.86	2.0	1.5	5.2

Case V is suitable whenever all of the input data does not fit into the available memory on the FPGA. In this case, multi-buffering reduces the overhead of data transfers and restarting the device. Table 2 shows performance improvement using multi-buffering. Speed-up tends to increase with the number of times the memory is refilled with improvement becoming nearly constant after refilling memory about 16 times. Table 3 shows that there is performance improvement using multi-buffering compared to using BLAS routines.

Table 2. Speed-up using multi-buffering over naive method

	Runs							
	1	4	8	12	16	20	32	32
Products	16	64	128	192	256	320	512	512
Speedup	2.75	3.31	3.42	3.33	3.49	3.50	3.52	4.26
Standard Deviation ($\times 10^{-2}$)	5.91	2.92	1.91	2.85	0.96	0.73	0.62	28.6
Effort‡	N	N	N	N	N	N	N	S

‡synthesis: N=normal, S=standard

Table 3. Speed-up using a single FPGA over BLAS

	Runs				
	4	8	12	16	32
Matrix Products	64	128	192	256	512
Average Speedup	1.00	1.03	1.04	1.05	1.05
Standard Deviation ($\times 10^{-2}$)	1.03	1.38	0.891	0.758	0.267

Lastly, consider using multiple processors and coprocessors. Instead of running a single coprocessor 32 times, use 32 coprocessors. For comparison purposes, use also 32 MPI processes and 32 OpenMP threads. In each case, 16 matrix products are computed concurrently by each process, thread or device. Although we do see speedup using OpenMP threads compared to using BLAS routines, the best performance is achieved using multiple FPGAs as shown in Table 4. In an MPI implementation, the scatter/gather operations are too costly. Recall, the requirement is that one process must hold all of the data before and after the computation. If this requirement were relaxed, then it would be possible to obtain better performance results using MPI and OpenMP.

Table 4. Speed-up using MPI, OpenMP, and multiple FPGAs compared to BLAS

	MPI	Open MPI	FPGA
Matrix Products per Process†	16	16	16
Average Speedup	0.979	2.34	4.1
Standard Deviation ($\times 10^{-2}$)	8.49	49.3	13.5
†per process, thread or device			

5. Conclusion

This paper presents practical techniques to efficiently implement solutions to problems that rely on matrix multiplication and similar computations. The best relative performance is observed using a vectorized approach with unrolling of the outer-loop (see Figure 6). If the input data fits in the local memory of the FPGA, then it is beneficial to fully utilize the memory banks (Case III). If more matrix products are computed than fit into the available memory, then multi-buffering (Case V) is beneficial.

We have observed speed-up using FPGAs. Due to the flexibility of FPGAs, it is time-consuming to conduct an exhaustive study even for a single target system. The focus of this study is limited to a particular matrix order based on empirical tests using a particular combination of hardware and software. Nevertheless, the same techniques may be applied using different matrix orders on newer chips.

It is possible designs other than the ones described will yield better performance improvement. Using the higher-level designs described, it is also possible to improve performance by rebuilding the codes in an HDL language, such as VHDL, thereby eliminating the Mitrion Virtual Processor to more efficiently utilize the chip. A sound design approach begins with a higher-level language such as Mitrion-C for initial studies to find the most suitable design, and ends with a "translation" of the best design into a HDL to obtain maximum speed-up.

Although the BLAS routines are fast and portable, FPGAs yield better performance. Our case study involved older technology (Virtex 4). As technology advances, especially with new designs that circumvent the memory bottleneck, FPGAs offer greater potential as application accelerators.

Acknowledgments

This work was performed on the SGI RASC RC100 system at NRL-DC under the auspices of the US Department of Defense (DoD) High Performance Computer Modernization Program (HPCMP).

References

1. Bique, S. and R. Rosenberg, "Fast Generation of High-Quality Pseudo-random Numbers and Permutations Using MPI and OpenMP on the Cray XD1", *CUG Proceedings*, 2009.

2. Dave, N., K. Fleming, M. King, M. Pellauer, and M. Vijayaraghavan, "Hardware Acceleration of Matrix Multiplication on a Xilinx FPGA," *Proceedings of Formal Methods and Models for Codesign (MEMOCODE)*, Nice, France, 2007.

3. Dongarra, J.J., J. Du Croz, I.S. Duff, and S. Hammarling, "A Set of Level-3 Basic Linear Algebra Sub-programs", *ACM Trans. Math. Soft.*, 16, 1990.

4. Dou, Y., S. Vassiliadis, G.K. Kuzmanov, and G.N. Gaydadjiev, "64-bit floating-point FPGA matrix multiplication", *ACM/SIGDA Field-Programmable Gate Arrays*, ACM Press, pp. 86–95, 2005.

5. Higham, N.J., *Accuracy and Stability of Numerical Algorithms*, 2nd edition, SIAM, Philadelphia, PA, 2002.

6. Kindratenko, V.V., R.J. Brunner, and A.D. Myers, "Mitrion-C Application Development on SGI Altix 350/RC100", *International Symposium on Field-Programmable Custom Computing Machines*, IEEE Xplore, 2007.

7. Lanzagorta, M., S. Bique, and R. Rosenberg, "Introduction to Reconfigurable Supercomputing", *Synthesis Lectures on Computer Architecture*, series editor M. Hill, Morgan & Claypool, 2009.

8. Roesler, E. and B. Nelson, "Novel Optimizations for Hardware Floating-Point Units in a Modern FPGA Architecture", *Lecture Notes In Computer Science*, Vol. 2438, Proc. FPL, Springer-Verlag, 2002.

9. Mitrionics, Inc., *Running the Mitrion Virtual Processor on SGI RASC RC100*, 2.0.3-001, 2009.

10. Mitrionics, Inc., *The Mitrion-C Programming Language*, 2.0.3-001, 2009.

11. Robinson, S., "Toward an Optimal Algorithm for Matrix Multiplication", *SIAM News*, Vol. 38(9), 2005.

12. Underwood, K.D. and K.S. Hemmert, "Closing the gap: CPU and FPGA Trends in sustainable floating-point BLAS performance", *Proceedings ofthe IEEE Symposium on Field-Programmable Custom Computing machines (FCCM 2004)*, April 2004.

13. Xilinx, Inc., *ISE Design Suite Software Manuals and Help -PDF Collection*, UG681, v 12.2, 2010.

14. Xilinx, Inc., *Free ISE WebPACK -Earlier Software Releases*, http://www.xilinx.com/webpack/classics/wpclassic, 2010.

Monotonic Lagrangian Grid Application on Virtex-4 FPGAs of SGI RASC RC100 at the Naval Research Laboratory

Stephen Bique and David Fyfe
US Naval Research Laboratory (NRL-DC), Washington, DC
{stephen.bique, david.fyfe}@nrl.navy.mil

Abstract

Many computational physics applications, e.g., molecular dynamics, require access to nearest-neighbor information. Determining nearest-neighbors can be compute-intensive. The Monotonic Lagrangian Grid (MLG) is designed to sort data in such a way that one can locate nearest-neighbors with index offsets in the data storage. In time-dependent calculations, such as molecular dynamics, the relationships between the particles do not change much from one time-step to the next. Hence, simple sorting algorithms like a binary-sort are adequate; although we examine other more powerful choices. We discuss implementing these sorting algorithms on Virtex-4 Field-programmable Gate Array (FPGA) of SGI RASC RC100system. We analyze techniques especially suitable for FPGA programming. FPGAs are of interest because of the advantages of low-powered hardware. Implementations are written in Mitrion-C, which is challenging because although the language is suitable to process loops in parallel on small collections of data, the proposed application involves random-access on a large collection of data. Different solutions exist, depending on the order of processing, so it is reasonable to consider the quality of the MLG. This application requires only simple operations and is amenable to a massively-parallel implementation. Yet, typical high performance computing (HPC) paradigms such as Message Passing Interface (MPI) or OpenMP are not useful to find a single MLG on the target system because of the well-known gather/scatter problem. Indeed, a large amount of data needs to be modified iteratively, and there exist data dependencies which create the gather/scatter problem. We report the speed-up using up to 32 FPGAs.

1. Introduction

In this section, we first introduce the application. Second, we look briefly at prior work. Third, we examine Field-Programmable Gate Arrays (FPGAs) for those readers who are not familiar with them. Fourth, we provide a quick overview of the software approach.

A. Philosophy and Motivation for the MLG Data Structure

The Monotonic Lagrangian Grid (MLG) data structure was developed for molecular dynamics to simplify the required N body-force calculations[3]. For N independent objects, there exists the possibility for $\frac{N(N-1)}{2}$ interactions, which is called the N^2 problem. However, for most problems, only the strong or "near-neighbor" interactions need to be considered. For most near-neighbor algorithms, this fact requires that a list of objects near in space be kept and updated whenever the objects move. The MLG data structure performs the task of organizing these N objects so that near-neighbors can be found without the need for near-neighbor lists. The MLG data structure is a dynamic data structure that maps objects in physical space into computer memory with well-defined memory offsets for near-neighbors. By mapping nearby objects into contiguous memory locations, the data structure allows the vectorization and parallelization of algorithms for the generation, maintenance and search of the data. Datasets can be partitioned and merged easily, which is especially important for distributed and parallel algorithms.

Small changes in the data, such as those that would occur for a spatial position during a particle dynamics simulation, do not trigger global changes in the MLG data structure. While the MLG data structure was originally developed for three-dimensional (3D) spatial-based data, its application can be expanded easily to incorporate other types of data in an arbitrary number of dimensions.

Motivation for the general definition of an MLG can be obtained by discussing its original 3D spatial definition[3]. Suppose $N = N_x \cdot N_y \cdot N_z$ objects (particles) are moving in space. Because of the form of N, it is natural to reference each object with three subscripts (i, j, k). For lattice particle simulations, it is also natural to associate each subscript with a physical dimension. Each of the objects has three physical coordinates $x_{i,j,k}$, $y_{i,j,k}$, and $z_{i,j,k}$, as well as other data as depicted in Figure 1.

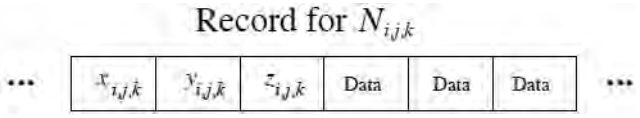


Figure 1. Object data record as stored in computer memory

An MLG construction orders the N objects such that the x positions of all objects increase monotonically with index i, the y positions of all objects increase monotonically with index j, and the z positions of all objects increase monotonically with index k. Mathematically, N objects in three-dimensional space with locations $x_{i,j,k}$, $y_{i,j,k}$, and $z_{i,j,k}$, constitute an MLG if, and only if:

$$\begin{aligned} x_{i,j,k} &\le x_{i+1,j,k} \quad \text{for} \quad 1 \le i \le N_x - 1, \\ y_{i,j,k} &\le y_{i,j+1,k} \quad \text{for} \quad 1 \le j \le N_y - 1, \\ z_{i,j,k} &\le z_{i,j,k+1} \quad \text{for} \quad 1 \le k \le N_z - 1, \end{aligned} \tag{1}$$

Given N random locations, the spatial lattice defined by an MLG is irregular. When the N object locations satisfy Equation 1 and any additional constraints or relations specifying other than infinite-space boundary-conditions, they are in "MLG order." This ordering is useful because the direction for going from one node to another in space and in the MLG is the same. Further, nodes which lie between two nodes in space will also be between them in the MLG. Thus, neighbors in real-space also have neighboring address indices in the MLG.

B. Constructing an MLG Data Structure

An existence proof for an MLG in the form of a construction procedure for any data is well-known[3]. The basic procedure is to sort all objects according to one of the relations of the MLG. Iteratively for each relation, the objects are partitioned into subsets and sorted according to the appropriate relation. This iterative procedure is carried out recursively until each relation is fully-satisfied. An MLG construction is only as efficient as each of the individual sorts for each relation. If the 1D sorts are $O(N \log N)$, then the algorithm is $O(N \log N)$.

Continuing the example with 3D spatial data, first sort the objects by their z coordinate. To do so, assign the first $N_x \cdot N_y$ objects the index k=1, the next $N_x \cdot N_y$ objects the index k=2, and so on. For each of the $N_x \cdot N_y$ planes, sort the objects by their y coordinate, assigning the first N_x objects of each of these sorted lists the index j=1, and so on. Lastly, sort each of the $N_x \cdot N_z$ subsets of N_x objects assigning the first object in each subset the index i=1, the second object the index i=2, etc.

A pictorial view of this construction algorithm in two dimensions for a 4×4 MLG on 16 objects is given by Picone, et al.[12]. Sixteen objects, labeled from **A** to **P** in Figure 2, are associated with a position (x, y) in two-dimensional (2D) space, as explained next.

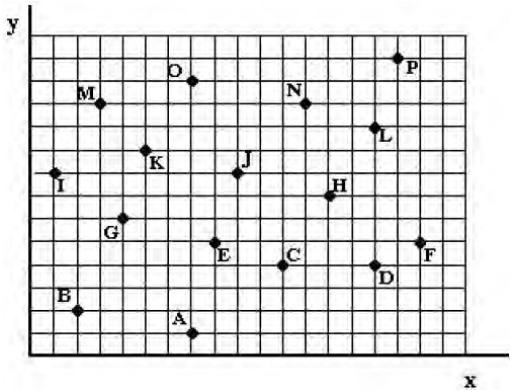


Figure 2. 2D space of 16 objects

The construction algorithm for this 2D example proceeds as follows:

1. Order all 16 objects according to increasing y-coordinate, i.e., from lowest to highest *y*-value.
2. Give the objects with the four lowest y-values (i.e., objects **A**, B, **C**, and **D**) a *j*-index of one. Assign those with the next four lowest *y*-values (**E**, **F**, **G**, and **H**) a *j*-index of two, and so on. Figure 3 shows each set of our objects connected with line segments.
3. Assign an *i*-index to each object in a given set of our objects with the same *j*-index, such that the *i* indices increase monotonically with the *j* indices. This assignment is shown in Figure 4. Note that those objects with the same *i*-index are connected via line segments.

A two-dimensional MLG in index space looks as depicted in Figure 5, where the letters are the labels of objects in previous figures. Objects (letters) that are adjacent in memory (index space) are connected by lines in Figures 3 and 4. Thus, adjacency of the object data in computer memory corresponds to geometric adjacency of the objects.

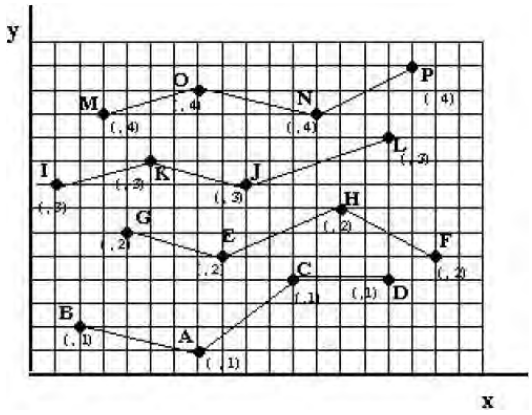


Figure 3. 2D space of 16 objects in groups of 4 by y

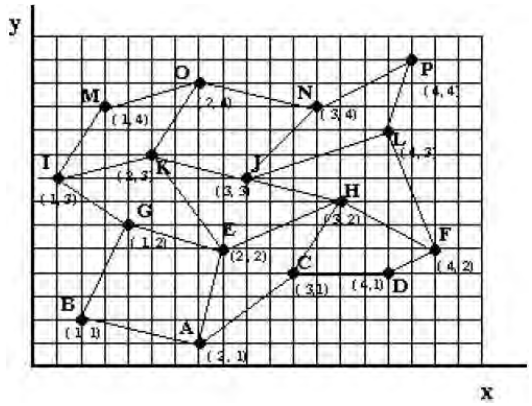


Figure 4. 2D space of 16 objects with (i, j) based on MLG rules

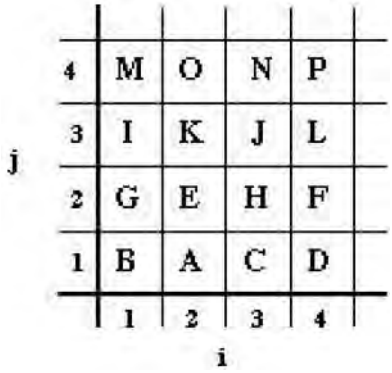


Figure 5. 2D MLG for 16 objects

The eight near-neighbors of object **H** in index space are objects **A**, **C**, **D**, **E**, **F**, **K**, **J**, and **L** in real-space as shown in Figure 6. The dashed-line in this figure encloses the near-neighbors of **H** in index space (compare with Figure 5).

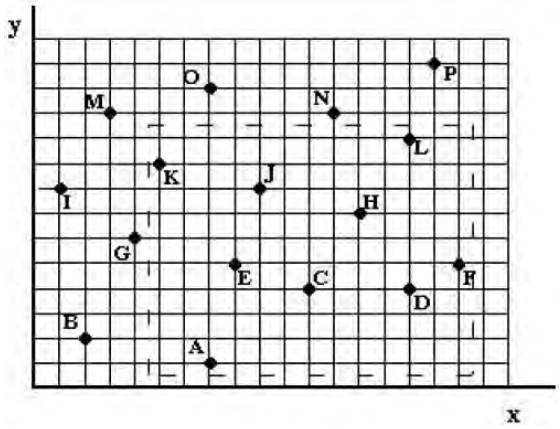


Figure 6. Near-neighbors of object H in real-space

In general, more than one MLG data structure can exist for a given set of data. The description of the construction of a 2D MLG data structure just given is simple and easy to follow, but may not produce the "best" MLG data structure for the problem. The choice as to which MLG is best must be based on the data, the problem being solved, and any additional constraints.

Based on prior work, this construction algorithm is not optimal for objects which are already "almost" in order. Data of this type occur in molecular dynamics calculations where the relative particle (object) positions change slowly over the course of one time-step for reasons of accuracy and stability. In this case, only a few particles are locally out of MLG order. The previous general construction algorithm can result in objects moving long distances in the MLG data structure before order is restored. Boris suggests using a bubble-sort algorithm in this case to restore MLG order[3]. A few iterations of a bubble-sort algorithm will recover an MLG order in $O(N)$ time. However, on completely random data, the bubble-sort algorithm is not optimal on a sequential computer. An MLG library developed by David Fyfe, Ronald Kolbe, Jay Boris, and Robert Sinkvits, includes a Shell-sort routine for such datasets that require non-local sorting. The MLG data structure was originally developed to facilitate algorithm writing and accessing near-neighbors on the Cray, a vector computer. Other sorting algorithms have not yet been implemented in the MLG package. The MLG library includes a routine to measure the quality of an MLG, and to generate better MLGs.

Shell-sort operates by doing a sort, such as bubble-sort or insertion-sort, on objects using successively smaller regular intervals until the interval length is one; whence, the last sort is a full- (bubble or insertion) sort. For example, if the size of the problem is N=32, then a choice for interval lengths consists of 8, 4, 2, and 1. In the case of an interval length of 8, eight sorts are carried out, starting at positions 0,1,2,…,7, where each sort involves items at four positions, such as 0, 8, 16, 24, and then the remaining sorts for smaller intervals are carried out. An advantage of this scheme is that objects require fewer swaps when their location is far from their final position in an MLG.

Shell-sort is an unstable algorithm. In a stable-sort, two records maintain their relative order when the keys are the same. Recently, we have found that unstable algorithms can cause infinite recursion when doing multi-dimensional sorting. The Shell-sort provided in the MLG library is not susceptible to this problem. This routine actually carries out Shell-sort only during initial recursive stages, which makes sense for random data. After sorting the entire axes once using Shell-sort, it is expected that the objects are more nearly in MLG order.

The time complexity of Shell-sort depends on the "increments" used. Using suitable increments, the time complexity is $O\left(N^{\frac{3}{2}}\right)$ in the worst-case, and $O(N^{1.25})$ for randomly-ordered data with N<60,000[14]. The flexibility that Shell-sort provides for the choice of increments is beneficial for parallel computing.

C. Field-Programmable Gate Arrays

The target devices are the thirty-two Xilinx Virtex-4 LX200 FPGAs on the SGI reconfigurable application-specific computing (RASC) platform RC100 at the US Naval Research Laboratory (NRL). Much like a CPU, FPGAs come in many different varieties constructed with different building blocks, and their performance depends on the entire system,

including the board, memory and input/output (I/O) systems. Unlike CPUs which are programmed in software, FPGAs are programmed in hardware and do not fetch, decode or execute instructions. Currently, CPUs and FPGAs are being built with billions of transistors on a single die.

An FPGA is a co-processor that may be seen as a large dense array of configurable logic blocks (CLBs) and programmable switch matrices (PSMs), plus horizontal and vertical buses which consist of multiple wires. A simplified sketch is depicted in Figure 7. FPGAs also have I/O blocks (IOBs), and random access memory (RAM) blocks on the periphery of the chip (not shown), plus multiply-accumulate circuits (MACs).

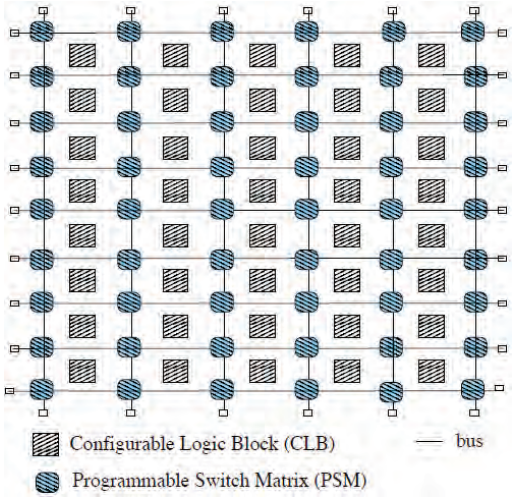


Figure 7. FPGA: an array of CLBs and buses

In early FPGA development, the number of wires was small, so that FPGAs were limited to operating only at the level of bits. As technology advanced, it became feasible to use multiple wires to operate at the level of words so that common data-types could be handled. Short wires (not shown) are used to connect adjacent CLBs, whereas PSMs are used to make longer connections between horizontal and vertical wires and to implement fan-out so that the same output is sent to multiple CLBs.

A CLB consists of a number of "slices". Each slice holds up to a few logic cells. Configure a cell to perform a desired logic function. Combine configured cells within a CLB to perform more complex logic functions.

Unlike the components inside an application-specific integrated circuit (ASIC) which is designed for a singular purpose, components are configured to perform different operations. The buses are used to connect individual components to provide the desired inputs and outputs, depending on the application for which the device is programmed. Hence, programming an FPGA refers to configuring the components and switches on the device.

An FPGA is ideally-suited for parallel computing, partly due to the vast number of components which can be operating in parallel. In particular, an FPGA is especially useful for fine-grained parallelism and pipelining. Although FPGAs currently run at slower clock-speeds, they are low-powered. Due to standards such as VHDL (VHSIC Hardware Description Language), FPGAs are important to reduce the time-to-market at lower cost[17].

In contrast, a traditional von Neumann CPU is poorly-suited for parallel computations, due to the limited number of functional units and the fetch-decode-execute cycle, which creates a well-known bottleneck. While Moore's law will likely hold beyond the next decade, it is expected that it will become increasingly difficult to significantly improve the performance of CPUs due to physical laws. Yet as technology relentlessly advances, the demand for processing power is ever-increasing. For this reason, accelerators such as FPGAs and GPUs (Graphics Processing Units) will continue to be studied.

How is an FPGA programmed at a high-level? A program may be viewed as a graph where the nodes are the "black boxes" that perform functions on inputs and the edges define the program flow. This graph must be mapped to the hardware target, which involves "synthesis, place and route". During this mapping process, the physical hardware configuration is inferred from a high-level description. Once all of the interconnections have been worked out in a high-level description (synthesis), the final stages include placement of all components, and routing of all connections to meet timing requirements.

A graph both models the desired computation and captures the inherent data dependencies. To obtain good performance, the graph has to be more than simply correct, i.e., always produce the desired outputs for valid inputs. A graph must be

"good" in the sense of exploiting the architecture to enhance opportunities for parallelism. For example, an FPGA is likely to perform poorly for any graph that may be chiefly characterized as sequential.

In practice, a good graph is constructed with advance knowledge of how the mapping and routing can be realized. The trick is to arrange the nodes to take advantage of potential parallelism and pipelining. Many well-known examples include "signal flow graphs" (SFGs) in the case of digital signal processing (DSP) applications[13]. To design a SFG, a programmer must have a sufficiently good model of the hardware device. In particular, it is necessary to analyze the timing in the network.

Popular programming tools include low-level hardware description languages (HDLs) such as Verilog (commonly used by vendors) and VHDL (commonly used by individuals), electronic system level (ESL) synthesis tools such as Bluespec System Verilog and DSPLogic, and high-level tools such as Impulse-C and Mitrion-C. A programmer using high-level tools does not need to study the hardware at the level of switches and transistors. This process is akin to writing a single high-level instruction such as a C or FORTRAN statement which will be translated to a number of microprocessor instructions.

Typically, many stages are involved from compiling to synthesis, and place and route of a high-level program. An experienced programmer is aware of the program structures, such as loops that typically yield good performance. Compilers for high-level languages perform various optimizations, especially for such program structures as loops.

Due to the large number (millions) of components and gates, synthesis, placement, and routing are exponentially hard problems. Exhaustive searching techniques are impractical. Hence, synthesis, placement and routing are not deterministic processes. Automatic synthesis tools can take hours or days, which will likely get worse as technology advances, as there will be more resources to manage. Nevertheless, automatic synthesis tools yield relatively quick design compared to manual design by an engineer[17].

D. Mitrion-C

Mitrion-C is a novel high-level programming language to easily express the type of pipelining and parallelism that is especially suitable for FPGAs. This type of fine-grained parallelism is frequently derived from loops. Not surprisingly, the main control structures in Mitrion-C are loops over the few main data structures, which are known as collections (lists, vectors and streams).

Unlike in traditional high-level languages, collections are immutable objects that can be defined, but not modified. Generally, the entire collection is processed in a sequential or parallel way. Indexing is costly and avoided. So a "good" Mitrion-C program is just a chain of loops for which the inputs of each loop are the collections defined in preceding loops in the chain.

Compiling a program in Mitrion-C generates a VHDL file that is used by the low-level synthesis, place and route tools to infer a configuration of the device. To produce such an input file requires solving several NP-hard problems. Hence, building a bit-stream, which is loaded to program the device, is a nondeterministic process that can be successful even when there have been previous unsuccessful attempts using the same source file. Although such failures often occur during synthesis, place or route, whenever utilization of resources is high, they can also occur when generating the initial VHDL file.

Mitrion-C cannot be used to design a circuit. The reason is a hardware abstraction layer known as Mitrion Virtual Processor (MVP) is used to run user code. This means that user code is mapped to "processing elements" in the MVP. It also means parts of the resources on the chip are used by the MVP. This overhead will become less significant as technology advances in the same way that memory is less scarce today compared to decades ago.

The chosen programming environment consists of the Mitrion-C programming language under Mitrion SDK 2.0, and Xilinx synthesis software ISE Design Suite 10.1[11,15,16]. For discussion of Mitrion-C including examples, see References 9 and 1. An evaluation of the SGI RASC RC100 in this programming environment based on an application involving floating-point arithmetic is given in Reference 8.

2. Programming Challenges

In this section, the focus is on the hurdles a traditional programmer encounters during FPGA programming. The starting point for analysis is the FORTRAN code provided in the MLG library. We discuss some of the challenges porting this code to an FPGA.

A. Problem Size, Data-type, and I/O Configuration

The MLG library was designed to handle an arbitrary number of dimensions. An executable provided as a test case generated random real 3D data with each axis having length 50. We applied a fast multiplicative congruential generator (MCG) to generate random numbers[2]. In particular, we filled linear memory with a sequence (x_{n+1} mod 500), where xn+1 is the next pseudorandom number given by Equation 2.

$$x_{n+1} = 1425040946427180923 \cdot x_n \bmod 2305843009213328881 \tag{2}$$

We fixed the number of dimensions at three. An axis length of 50, which was used in the test case, is unsatisfactory because the number is not a power of two. Choosing a power of two is convenient to take advantage of multi-buffering capabilities, which are provided by the RASC tools. In particular, the size of an MLG should evenly divide memory so that more data can be read and written to an FPGA than fits into local memory; additionally more MLGs can be easily computed using multiple FPGAs.

Initially, the length of each axis was fixed with 64 points per axis. Later the length was reduced to 32. This choice left more BlockRAMs available, which made it possible to test more designs. Moreover, it is convenient to arrange that the length of each axis is divisible by the number of BlockRAMs.

It is a programming challenge to make effective use of 32BlockRAMs using the chosen tools on the target FPGAs. The purpose of multiple BlockRAMs is to arrange parallel access. This effectively means that processes must be duplicated for every BlockRAM. This duplication is costly in terms of number of flip-flops. For example, given source code that builds properly, simply adding an extra reader in a loop (overall BlockRAMs) may be impossible because the reader is duplicated for every BlockRAM.

BlockRAMs are small memories built into the fabric on the chip, not the local SRAM banks. The local memory banks are used for multi-buffering so that the host writes and reads SRAMs. As there are only a couple SRAMs available on the target device, and each of them provides only a couple ports for I/O, the coprocessor cannot use them for parallel access. Many BlockRAMs are created to permit parallel access, which is needed for this application. So the coprocessor transfers data between SRAMs and BlockRAMs in order to have parallel access.

Although we initially developed codes using floating-point, the chosen data-type is integers. This choice of data-type made synthesis easier since simpler logic was required, i.e., 16-bit integer comparisons instead of 32-bit floating-point comparisons. Because Virtex-4s have relatively much fewer gates than newer FPGA variants, floating-point is feasible on newer chips. If there was not the overhead of the MVP, then synthesis would have been easier using floating-point even on Virtex-4s.

The chosen map of the memory is explained next. Although there are five 8 MB SRAM banks on-board the chips, we use only four of them. Two physical banks form each logical bank. The FPGA is configured to use only half of each logical bank in order to implement multi-buffering. In particular, the input data is stored in the first half of one logical bank, and the results are stored in the first half of the other bank as shown in Figure 8. For a discussion of issues to consider in deciding the memory map, see Reference 1.

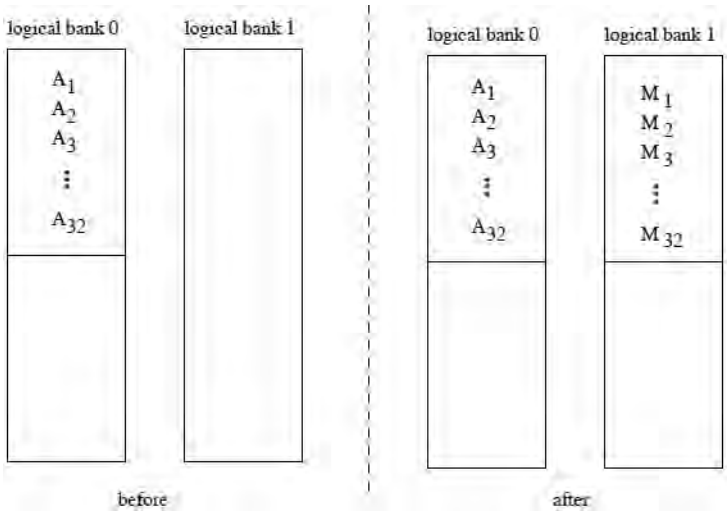


Figure 8. Memory Map of Input (left) and Output (right)

B. Analysis and Profiling

A popular proposal is to profile a code to find the portions of code suitable for acceleration. Such technique is practical on systems for which run-time reconfiguration is fast, which is hardly the case on the chosen system. The construction of an MLG requires a large multidimensional array to be iteratively modified. The main operations are comparisons and swaps, which take relatively little time compared to the overhead of transferring a large multidimensional array and starting the coprocessor. On the target system, this overhead is prohibitive so that the only option is to run the entire code on the coprocessor.

After profiling the code in the MLG library, two functions were observed to account for the bulk of the runtime. A code fragment for the function that performed a fast scan to determine which swaps to perform (i.e., which pairs are out of MLG order) appears in Figure 9. A code fragment for the other function that calculated addresses and performed the swaps is shown in Figure 10. After revising these functions, some small improvement was realized. But then it was noticed that the same improvement could be obtained by simply adding compiler switches to create the executable. Hence, the code was optimally written.

```
      do k=1,Nhigh
        do i=1,Nlow
          swap_if(i,k) =
&           MLGleft(axis,i,1,k) .gt. MLGright(axis,i,1,k)
        end do
      end do
```

Figure 9. FORTRAN code fragment for scan

```
do is=1,vlen
  if(swap_it(is))  then
    indexl = is - 1
    k = (indexl/nlow)*kfac
    indexl = indexl + jk + k
    indexr = indexl + slen
    hold = MLG_index(indexl)
    MLG_index(indexl) = MLG_index(indexr)
    MLG_index(indexr) = hold
    do l=1,swap_words
      hold = MLG(swap_data(l),indexl)
      MLG(swap_data(l),indexl) =
                          MLG(swap_data(l),indexr)
      MLG(swap_data(l),indexr) = hold
    end do
    MLG_INDEX_SWAP =
      MLG_INDEX_SWAP + 1
  end if
end do
```

Figure 10. FORTRAN code fragment for swaps

Although the code in the provided MLG library is optimized, the algorithmic design is unsuitable for the target device. The problem with this design is twofold. Firstly, although the function that computes addresses and performs the swaps is a data parallel implementation, which is well designed for a vector computer, the function that performs the scan is sequential. These functions would create a bottleneck on the FPGA, as there would be loops producing and consuming data at different rates. Secondly, the address calculations involving divisions are expensive to perform on the FPGA and can be avoided. Thus, simply "translating" the FORTRAN code into Mitrion-C is unlikely to yield acceptable performance on the FPGA.

A simple algorithm performs the swaps in-place as the scan is done and avoids unnecessary divisions. A code fragment in C is given in Figure 11. This code performs a single pass of the bubble-sort algorithm. This approach makes it possible to find the same MLG solution using the FORTRAN code (without Shell-sort) since the order of the swap operations can be maintained.

C. Constant Expressions in Loops

The code fragment in Figure 11 cannot be easily ported to Mitrion-C. The problem is that the expressions in **for** loops have variables instead of constant expressions. For example, the variables **nhigh** and **nlow** depend on the axis along which sorting is carried out.

```
for( k=0; k < nhigh; k++ )
   {
     j = k * kfac + jk;
     for( i=j; i < j+nlow; i++  )
         {
           index1 = i * _RECORD_LENGTH;
           if( MLG[index1 + axis] > MLG[index1 + slena] )
             {
              for( l=index1; l < index1+swap_words; l++ )
                {
                  hold = MLG[l];
                  MLG[l] = MLG[l+slen];
                  MLG[l+slen] = hold;
                }
              Swap_Words_Result += 1;
             }
         }
   }
```

Figure 11. Code fragment in C doing swaps-in-place

One possible solution is to fix the values of the variables for each axis, and write separate loops for sorting instead of a single nested loop. However, this solution is expensive in terms of code duplication. Sometimes, it is possible to recombine after rearrangement to reduce duplication. For instance, after writing separate loops for sorting along each axis, it is possible to combine a couple loops in order to sort along more than one axis, but not all axes.

Unlike the coprocessor, a CPU exhibits excellent performance when processing irregular loops due to advances in hardware design and optimizing compilers. As there is no instruction stream for an FPGA, the coprocessor operates quite differently. Regular loops more naturally fit the model of a circuit, and make it possible to achieve greater efficiency. Although it is frequently possible to remove an irregular structure by unrolling loops, the cost in hardware resources is frequently prohibitive. Instead of rewriting code, a programmer typically seeks another algorithm with more regular control structure.

D. No Function Calls

In the Shell-sort routine in the MLG library, eventually the sorts are single passes of the classical bubble-sort algorithm. These sorts are recursively applied to every axis until an MLG is found. In this context, the term sort is used to refer to a partial sort as in such a pass of the bubble-sort algorithm. A default traversal order is used. This order is explained using the notation:

$$\vec{x}\,\vec{y}\,\vec{z},\ \vec{z}\,\vec{x}\,\vec{y},\ \vec{y}\,\vec{z}\,\vec{x},\ \vec{x}\,\vec{y}\,\vec{z},\ \vec{z}\,\vec{x}\,\vec{y},\ \vec{y}\,\vec{z}\,\vec{x},\cdots$$

where the letters correspond to the axes and the arrows indicate the direction. Changing direction should not be confused with changing sorting order, as in decreasing or increasing, but rather the traversal direction. This symmetrical pattern is repeated as needed. First, sort all of the x-axes. Second, sort all of the y-axes. Third, sort all of the z-axes. If no swaps are made after visiting all of these axes, then an MLG has been found; otherwise, sort all of the z-axes in the reverse direction, and so forth. Note even if all of the axes along one of the dimensions are sorted, they can become unsorted after sorting axes along a different dimension.

To implement the described traversal order in a traditional programming language, simply write a loop (e.g., a **for** loop) that produces the desired ordering of directions and axes, and call a routine passing the axis and the direction for each iteration. As there is no instruction stream, there are no true function calls on an FPGA. In other words, each "function call" in source code targeting an FPGA requires the same amount of resources as the function itself. Furthermore, there is no such thing as passing a memory reference as the connections to memories are always fixed.

It is not clear how to efficiently implement the described traversal order on current FPGAs. Presumably many functions would be needed to arrange efficient parallel access depending on the traversal order. Writing so many functions leaves less space for duplication; whence, even if there are sufficient resources, it is unlikely that performance will be satisfactory due to the low-degree of parallelism.

Because it does not seem feasible to implement the same traversal order on an FPGA and achieve acceptable speed-up, a different ordering will be applied. Using a different traversal order means a different MLG will typically be found. Our primary interest is performance. Nevertheless, it is interesting to measure the quality of the results when they differ.

It might be surprising that there exists an exponentially-large number of ways of choosing a traversal order. There is no known analytical reason to choose any particular order among the vast majority of such traversals. Yet, some traversal orders yield better performance and quality compared to other traversals. Plainly, the cost of computing the travels impacts performance. The default symmetrical traversal order is a candidate when searching for a better MLG. Other possible traversals include:

- $\vec{x}\,\vec{y}\,\vec{z},\cdots$,
- $\vec{z}\,\vec{y}\,\vec{x},\cdots$,
- sort all x-axes and y-axes in alternating fashion within same z-plane, visiting each z-plane in a natural order, and then sort all z-axes in a natural order, and repeat,
- sort all z-axes in a natural order, and then sort all x-axes and y-axes in alternating fashion within same z-plane, visiting each z-plane in a natural order, and repeat,
- sort all x-axes in first z-plane, then sort all y-axes in same plane, for all z-planes, and lastly sort all z-axes in a natural order, and repeat,
- sort all y-axes in first z-plane, then sort all x-axes in same plane, for all z-planes, and lastly sort all z-axes in a natural order, and repeat,
- sort all z-axes in a natural order, and then sort all x-axes in first z-plane, then all y-axes in same plane, for all z-planes, and repeat,
- sort an x-axis, a y-axis and a z-axis in that order, until all axes are visited, listing all x-axes in each x-plane, all y-axes in each y-plane, and all z-axes in natural order, and repeat,
- sort an x-axis, a y-axis and a z-axis in that order, and repeat until all axes are visited, listing all x-axes in each z-plane, all y-axes in each z-plane, and all z-axes in natural order, and repeat,
- sort all z-axes in a natural order, and then sort alternating groups of four axes with gap of 8 within same z-plane, sorting successively four x-axes, and then four y-axes, until all axes in same z-plane are sorted, proceeding plane by plane, and repeat,
- sort all x-axes in first y-plane, then sort all z-axes in same plane, repeating for all y-planes, and lastly sort all y-axes, and repeat.

To reduce the amount of hardware resources, visitation strategies that require less logic are preferred. However, using more resources for control might be acceptable if faster convergence is observed, or better MLGs are found. We compared performance and a statistical measure on the quality of MLGs computed on a CPU using various sorting orders. We found that both performance on CPU and quality of MLGs are better using only the simplest strategies without changing directions, such as $\vec{x}\,\vec{y}\,\vec{z}$, $\vec{z}\,\vec{y}\,\vec{x},\cdots$.

E. Complexity

Although the number of recursive stages depends on both the input data and the sorting algorithms used, we observed consistent empirical evidence that the number of iterations is fairly constant for a given routine. This observation is congruous with the statement that time-complexity to find an MLG is the same as the complexity of the one-dimensional (1D) sorting algorithm applied to each axis.

We investigated performance of different sorting algorithms on a CPU. One of the authors developed optimized C code for various algorithms. In some of the experiments, two functions are passed to a common routine: a function that carries out a 1D sort on an axis, and a function that computes the next axis. In addition, we ran experiments using a similar common routine except that the sorting function that was passed as an argument was applied exactly once to all axes, and all remaining sorts were stable, such as a full bubble-sort, or insertion-sort, or a partial-sort such as a single pass of the bubble-sort algorithm. For all of these tests, we especially focused on the traversal order $\vec{z}\,\vec{y}\,\vec{x},\cdots$.

More efficient sorting algorithms, such as quick-sort and Shell-sort, are unstable. Although it is not difficult to modify these algorithms to enforce stability, the performance of such modified algorithms will be significantly impacted. Indeed, the efficiency of quick-sort is due to its simplicity. Nonetheless, it is reasonable to apply an unstable algorithm initially on multi-dimensional data, provided that recursion eventually utilizes only stable algorithms to avoid infinite recursion in cases when the keys are not all distinct on every axis.

Based on run-times, we list algorithms in increasing order of observed performance in Table 1. When two algorithms are listed, the first one is applied only during initial sorting phase. Shell-sort and quick- sort applied initially on random data yield nearly the same CPU performance. After the initial phase, a single pass of the bubble-sort algorithm applied to each axis yields optimal performance based on our tests.

Table 1. Approximate relative performance of algorithms

Algorithms	Performance
bubblesort	worst
selection sort	
Shell sort & bubblesort	
stable Shell sort	
stable Shell sort with merge	
stable quicksort	
“minimizing swaps” Shell sort	
parallel bubblesort & insertion sort	
single pass bubblesort	
parallel bubblesort	
insertion sort	
selection sort & insertion sort	
Shell sort & insertion sort	
quicksort & insertion sort	
parallel bubblesort & single pass bubblesort	
insertion sort & single pass bubblesort	
selection sort & single pass bubblesort	
quicksort & single pass bubblesort	
Shell sort & single pass bubblesort	best

We found that the revised FORTRAN Shell-sort routine using simple traversals produced better quality results compared to using the same routine with bubble-sorts (submlgs=0) instead of Shell-sorts. We also found that the revised FORTRAN Shell-sort routine on average produced nearly the same or slightly better quality results compared to using insertion-sort, parallel bubble-sort, selection-sort or quick-sort. Our Shell-sort implementation based on insertion-sort using an efficient merge for the final phase is approximately twice as fast as the revised FORTRAN version.

Unfortunately, few sorting algorithms exhibit a regular structure which is ideal for implementation an FPGA. Among the listed algorithms in Table 1, two are highly-suitable for implementation on an FPGA, namely, parallel bubble-sort and single pass bubble-sort.

3. Matrix Transpose

The solution requires iteratively sorting all axes. When sorting along a different dimension, address calculations are different because different strides are used for each dimension. Due to hardware and software limitations, it does not seem possible to optimize parallel accesses for every dimension.

Matrix transposition provides a means to reduce the problem from 3D to 1D, ignoring the cost of matrix transpose. By transposing, it is necessary to sort along only one of the dimensions. Using this technique, it is easy to optimize parallel accesses and avoid rewriting code to handle special memory access patterns for each dimension.

```
repeat
    repeat
        sort all z-axes,
        transpose so that z-axes become x-axes,
    until all z-axes are restored to their original positions
until no swaps are performed during sorting
```

Figure 12. Pseudo-code utilizing Matrix Transpose

In general, a matrix transpose operation presents technical challenges in FPGA programming. It is convenient to partition data among BlockRAMs so as to arrange parallel access for the main operations (sorting). In order to perform matrix transpose efficiently, parallel access is also needed. Currently, it is too costly to arrange parallel access along more

than one-dimension for large multidimensional datasets using a single set of BlockRAMs. Furthermore, it is unclear how to decompose the matrix transpose operation in order to overlap sorting operations and transpose operations; whence, a full matrix transpose must be computed before the next sort commences.

In particular, it is necessary to use twice the number of BlockRAMs to store the data, since it is not possible to efficiently perform matrix transpose in place. Hence, it is impossible to avoid copying data whenever using matrix transpose. If data is sorted in one set of BlockRAMs, and then transposed in the other set, the data would be in the wrong place! The reason is because the memory references are fixed. So in this case it would be necessary to copy the data to the other BlockRAMs which are referenced in the sorting routine.

A useful optimization is explained next. Assume that only z-axes are sorted. Partition the data so that pairs of z-hyperplanes, which are perpendicular to z-axes, are stored in separate BlockRAMs. Hence, there are 16 BlockRAMs that permit parallel read access as illustrated in the Mitrion-C code fragment in Figure 13. In this figure, **mem_V** is a vector of "instance tokens" which is a syntactic description, not a semantic one. The **foreach** loop over this vector implements32 concurrent reads as there are two reads for each of the 16 BlockRAMs. After reading and sorting an axis, instead of writing the results back to the same set of BlockRAMs, write the results in parallel to the "transpose" set of BlockRAMs as shown in Figure 14. For this transpose set, partition the data so that pairs of x-hyperplanes are stored in separate BlockRAMs. The **foreach** loop in Figure 14 performs 32 writes, since there are two writes for each of the 16 BlockRAMs.

```
// read in parallel from every BlockRAM
( MLG_record[16] evens, MLG_record[16] odds, mem_V ) =
  foreach( t in mem_V ) {
     (x1, m_0) = memread(t[0], indexl); // evens
     (y1, m_1) = memread(t[0], indexr); // odds
      m_2 = [ memsync(m_0,m_1) ];
  } (x1,y1,m_2);
```

Figure 13. Mitrion-C code to read an axis

```
// Write to transpose (optimized for parallel write access)
(transpose) = foreach( e, d, instok  in
                       sorted_evens, sorted_odds, transpose )
              {
                 im_0 = memwrite(instok[0], indexl2, e );
                 im_1 = memwrite(instok[0], indexl2 + 1, d );
                 im_2 = [ memsync(im_0,im_1) ];
              } (im_2);
```

Figure 14. Mitrion-C code to write an axis

In this way, the transpose is virtually free! However, the transpose is in the wrong place since the memory references cannot be used in sorting. Furthermore, the data is not properly partitioned for parallel access to sort the data.

The actual cost of using matrix transpose is the time and resources needed to transfer the data between the sets of BlockRAMs. By using different parallel access patterns for the different sets of BlockRAMs, it is possible to read in parallel from one set of BlockRAMs, and write in parallel to the other set. Thus, transferring data is highly-parallel as depicted in Figure 15. In this figure, observe that 16 axes are read concurrently in 16 iterations of a **foreach** loop with a "gap" size of 64 as two columns are stored per z-hyperplane in each BlockRAM. This code shows conversion of a vector to a list in a sequence of steps, which is more efficient than doing the conversion in a single step. Notice also from this code that there are 32 writers to write a single axis with 32 elements using a gap size of 1,024 as an entire z-hyperplane is stored in each BlockRAM destination.

Nonetheless, matrix transpose is expensive both in terms of resources and time. Parallel access requires many readers and writers. Because the size of the multi-dimensional array is large, the time-to-transfer data is significant even using multiple readers and writers.

We ran different codes on an FPGA using matrix transpose as described. We found that these implementations were not as fast as other implementations which did not involve matrix transpose. We observed only slight speed-up (around 1.4×) running an FPGA once to compute 32 MLGs compared to finding 32 MLGs using a CPU on same system.

4. Implementation on FPGA

Numerous designs were coded in Mitrion-C. Synthesis was not always successful due to insufficient resources; consequently, some designs cannot be tested on the target system using the same tools. Notwithstanding this caveat, it is possible that some slight modification or clever trick would make synthesis possible in some cases. In some cases, when

```
// Copy transpose to blockRAMs for parallel read access

(mem_T, mem_V) = for( uint:11 row in <0..31> ) {

  uint:11 read_offset = row << 1;
  uint:11 write_offset = row << 5;

  (mem_t, mem_v)  = for( col in <0,1> ) {

    // read 16 axes
    (hm,pm,mem_t) = foreach( rt in  mem_t ) {// PAR

      (h,p,mhp_) = foreach( uint:11 k in <0..15> ) {// SEQ
        rd_offset = (k << 7)+read_offset; // skip 2 planes
        (h_el, mh) = memread( rt[0], rd_offset );
        (p_el, mp) = memread( rt[0], rd_offset + 64 );
         mhp = memsync(mh,mp);
      } (h_el, p_el, mhp);

      mhv = [ mhp_ ];

    } (h, p, mhv);

    // convert vector of 16 axes to list of 16 axes
    hl1 = reshape( hm, [4][4]<16> );
    pl1 = reshape( pm, [4][4]<16> );
    hl2 = reformat( hl1, [4]<4><16> );
    pl2 = reformat( pl1, [4]<4><16> );
    hl3 = reformat( hl2, <4><4><16> );
    pl3 = reformat( pl2, <4><4><16> );
    hl = reshape( hl3, <16><16> );
    pl = reshape( pl3, <16><16> );

    // write 16 axes
    mem_v = for( lh,lp in  hl, pl by uint:11 position ) {
      vh = reformat( lh, [16] );
      vp = reformat( lp, [16] );
      wr_offset = (position << 1) + write_offset;
      mem_v = foreach( e, f, wt  in  vh, vp, mem_v  ) {
        wt1  = memwrite( wt[0], wr_offset, e );
        wt2  = memwrite( wt[0], wr_offset + 1024, f );
        swt = [memsync( wt1, wt2 )];
      } swt;
    } mem_v;

    read_offset = read_offset + 1;
    write_offset = write_offset + 1;

  } (mem_t, mem_v); // foreach col

} (mem_t, mem_v); // foreach row
```

Figure 15. Mitrion-C code to copy matrix transpose

building the initial VHDL file used by the Xilinx synthesis software, an output message contained the text "INTERNAL COMPILER ERROR". A programmer seeing this message might attempt to revise the source code to avoid this error. However, we found that by repeating the build process two or more times without making any changes, a VHDL file could be produced. The reason the build process can sometimes fail is that the algorithm used to build a VHDL file is non-deterministic solving a number of NP-hard problems.

Although synthesis was successful in many cases, the resulting bit-streams did not always run on hardware as expected, which is probably due to bugs in the software, or programming errors. In less severe cases, the program did not terminate. In more severe cases, the program crashed the system. Nevertheless, many different designs run successfully on hardware. Among these designs, the best implementation is described next.

The size of every MLG is fixed so as to evenly divide the size of memory. In particular, up to 32 MLGs can be computed during a single-run of an FPGA. More MLGs can be computed using multiple FPGAs as well as the multi-buffering feature provided by the RASC API to automatically transfer data and restart FPGAs. Pseudo-code for the main loop is shown in Figure 16.

```
for each input array
    transfer input data from SRAMs to BlockRAMs
    find MLG using data in BlockRAMs
    transfer MLG from BlockRAMs to SRAMs
```

Figure 16. Loop in main program

Figure 17 shows Mitrion-C code to transfer input data from an SRAM bank to BlockRAMs. The data in an SRAM bank is read sequentially. Each 128-bit word in an SRAM bank holds two 64-bit records. After reading 32 records into a collection, it is reshaped into a vector that is written in parallel to 32 BlockRAMs.

```
copy_combined_bank2BRAMs( MEM_SRAM mem_a_00, offset,
                          BRAM[BlockRAM_NUM][1] mem_c )
{
(mem_ext, mem_int) = for( Reg index in < 0.. HYPERPLANE - 1  > )
    // HYPERPLANE = size of plane = number of axes per dimension
    {
      Index = (index << 4) + offset;

      // Read values from extram mem_a_00 (one at a time)
      (words0, mem_a_00) = foreach(num in <0..15>)
      {
        (word0, a1) = memread(mem_a_00, Index + num);
        bits:64[2] vec2 = word0;
        v0 =  [ (MLG_record)vec2[0], (MLG_record)vec2[1] ];
      }(v0, a1);
      words0_v = reformat( words0, [16][2] );
      words1_v = reshape( words0_v, [32] );

      // Write values to blockrams
      mem_c = foreach(w0, mem_c_cur in  words1_v, mem_c)
      {
         c01 = [ memwrite(mem_c_cur[0], index, w0) ];
      } c01;

    } (mem_a_00, mem_c);

} ( mem_ext, mem_int );
```

Figure 17. Mitrion-C code to read data from SRAM

Figure 18 shows Mitrion-C code to transfer an MLG from BlockRAMs to an SRAM bank. A vector of 32 records is read in parallel from the BlockRAMs. This vector is efficiently reshaped into a list, which is a suitable format for writing sequentially. This list holds pairs of records, which are sequentially written as single 128-bit words to the proper external memory bank.

```
copy_BRAMs2combined_bank( BRAM[BlockRAM_NUM][1] mem_itv,
                          MEM_SRAM mem_b0, offset )
{
(mem_b, mem_c) = for(Reg index in < 0.. HYPERPLANE - 1 >)
  {
    // Read values from BlockRAMs
    (x1_v, mem_itv) = foreach(mem_el in mem_itv)
    {
       (data, memtoken) =  memread(mem_el[0], index);
       token_ = [ memtoken ];
    } (data, token_);
    x1_r = reshape(x1_v,[4][4][2]);
    x1_1 = reformat( x1_r, [4]<4>[2] );
    x1_2 = reformat( x1_1, <4><4>[2] );
    x1_l = reshape( x1_2, <16>[2] );
    address = (index << 4) + offset;

    // Write values to SRAMs
    mem_b0 = foreach( pair in x1_l by Reg position)
    {
      result =  val_MLG_Typeto128(pair[0], pair[1]);
      b01 = memwrite(mem_b0, address+position, result);
    } b01;
  } (mem_b0,  mem_itv);
} (mem_b, mem_c);
```

Figure 18. Mitrion-C code to write data from SRAM

Figure 19 shows the main control structure to find an MLG. The **while** loop implements the recursion. After the first iteration, the body of the **while** loop is executed only if a swap occurred during the previous iteration. During each iteration, every axis is sorted exactly once.

The first **for** loop in Figure 19 is executed twice, firstly, sorting all z-axes, and secondly, sorting all y-axes. Most of the **for** body is omitted from this figure and displayed in Figure 21. The second **for** loop in Figure 19 sorts all x-axes, and the loop body is omitted from this figure, and shown in Figure 20.

The data is partitioned along the x-axis so that each x-hyperplane is stored in a separate BlockRAM. A z-axis or y-axis is read from every BlockRAM and sorted concurrently, although each individual axis is read and sorted sequentially. Hence, 32 sorts are carried out in parallel. The offsets for each element along the axes that are read concurrently are the same and are computed in the first for each loop shown in Figure 21.

The “head” of each axis is read in the first statement of the second **foreach** loop, and the “tail” of the axis is read in the subsequent **foreach** loop of Figure 21. A bubble-sort is carried out concurrently for each axis in the **for** loop shown in Figure 21. Lastly the sorted axes are written concurrently back to BlockRAMs in the last **foreach** loop at the bottom of the figure.

```
mlg( BRAM[BlockRAM_NUM][1] mem_V )
{
  // Initialize loop-dependent variables
  bool  swapped = true;

  mem_result = while( swapped ) {

  bool swap_yz = false;
  ( BRAM[BlockRAM_NUM][1] memx, Swap_yz) =
      for( key,inc,offset  in  <true,false>, <1,32>,
         < [ 32, 64, 96,128,160,192,224,256,288,320,
             352,384,416,448,480,512,544,576,608,640,
             672,704,736,768,800,832,864,896,928,960,
             992], [1..31] > ) {

        uint:11 start_location = 0;
        (mem_V, swap_yz) = for( axes in <0..31> ) {

             // Sort all y-axes and z-axes

             start_location = start_location + inc;
             swap_yz = swap_yz ||
                       ((bits:32) Bits != zero32);
        } (mem_V, swap_yz);

  } (mem_V, swap_yz);


  bits:1[16] Swap= zero16;
  ( mem_V, swap ) = for( row_number in <0..1023> ) {

         // Sort all x-axes

   } ( mem_V, Swap );

   swapped = Swap_yz || ((bits:16)swap != zero16);

 } mem_V;  // while loop;
} mem_result;
```

Figure 19. Mitrion-C function fragment to find an MLG

```
vector_read = foreach( rd_token in  memx ) {
   element = memread( rd_token[0], row_number );
} element;

MLG_record[16][2] vector_p =
    reshape( vector_read, [16][2] );
(MLG_record[16] evens_next,
 MLG_record[16]  odds_next) =
    foreach( pair in vector_p ) {
      even_element = pair[0];
       odd_element = pair[1];
} (even_element, odd_element);

(xt__,yt__,Swap_)=foreach( e1,e2,f in evens_next,
                           odds_next, Swap) {
  e1_1 = untup(e1);
  e2_1 = untup(e2);
  (e1_,e2_,f_) = if( e1_1 > e2_1 ) (e2,e1,bit1)
                 else (e1,e2,f);
} (e1_,e2_,f_);

(odds_, evens_, Swap__)=foreach( e1,e2,f in
          yt__ </ 15, xt__ /< 1, Swap_ </ 15) {
  e1_1 = untup(e1);
  e2_1 = untup(e2);
  (e1_,e2_,f_) = if( e1_1 > e2_1 ) (e2,e1,bit1)
                 else (e1,e2,f);
} (e1_, e2_, f_);
Swap=foreach(bit in Swap__ >< (Swap_ /< 15)) bit;

vector_of_pairs=foreach(even_elt,odd_elt in
 (xt__ </ 1) >< evens_, odds_ >< (yt__ /< 15)) {
    paired = [ even_elt, odd_elt ];
} paired;
sorted_vector = reshape( vector_of_pairs, [32] );

mem_V = foreach( a_elt, wr_token in
          sorted_vector, memx ) {
  wr_tok = [ memwrite( wr_token[0],
                       row_number, a_elt) ];
} wr_tok;
```

Figure 20. Mitrion-C loop body to sort x-axes

As there is a one-to-one mapping between the x-hyperplanes and the BlockRAMs, all of the elements along an x-axis are read in parallel, as seen in the first **foreach** of Figure 20. The elements read are then formed into two groups by the even and odd positions along an axis, as seen in the second **foreach** loop in Figure 20. Next, a single pass of the parallel bubble-sort algorithm is carried out, which consists of two phases. In the first phase, 16 "even" pairs are sorted in the third **foreach** loop. In the second phase, 15 "odd" pairs are sorted in the fourth **foreach** loop. The sorted elements are recombined into a single vector in the fifth **foreach** loop. Lastly, all sorted elements are written in parallel back to BlockRAMs, as seen in the **foreach** loop at the bottom of the figure.

```
address_vector = foreach( uint:11 j in offset )
                    {a = j + start_location;} a;
address_list = reformat( address_vector, <31> );

(mem_V, Bits) = foreach(instance_token in mem_V ) {
  MLG_record e = memread(instance_token[0],
                         start_location);
  list_yz = foreach( j in  address_list) {
      elem = memread( instance_token[0], j );
  } elem;

  bits:1 switch = bit0;

 (List,last,switch_result) = for(enext in list_yz)
      {
       (d1_1,d1_2,d1_3) = untup(e);
       (d2_1,d2_2,d2_3) = untup(enext);
       (u1,u2) = if ( key ) (d1_3,d2_3)
                 else (d1_2,d2_2);
       (First,e,switch)=if(u1 > u2)(enext,e,bit1)
                        else (e,enext,switch);
      } (>< First, e, switch) ;

  token=foreach(j,x in
                <start_location> >< address_list,
                    List >< <last>  ) {
          tok = memwrite(instance_token[0], j, x);
  } tok;
  tv = [ token ];
} (tv, switch_result);
```

Figure 21. Mitrion-C fragment to sort y-axes and z-axes

Approximate speed-up for the described implementation was calculated using the average of ten runs on the system. The results are shown in Table 2. Significant speed-up is observed, even using a single FPGA, provided enough MLGs are found. Although the quality of an MLG produced by an FPGA is not as good as the quality of an MLG found using Shell-sort, the quality was better than MLG found using only bubble-sorts.

Table 2. Approximate speed-up of FPGA over CPU

MLGs	FPGAs	Runs of each FGPA	Speedup
1	1	1	0.91
32	1	1	4.8
1152	1	36	5.9
1408	4	11	24
6144	32	6	56

5. Conclusion

The MLG application involves iteratively modifying a large multi-dimensional array. Each iteration has a number of stages that corresponds to the shape of the array. Each stage permits a high degree of parallelism as every parallel axis can be sorted in parallel. Yet there exists data dependencies between stages, which create a gather/scatter problem.

Typical parallel programming paradigms such as MPI and OpenMP are not useful to accelerate finding a single MLG on the target system due to the scatter/gather problem; i.e., the communication costs are excessive relative to the computational cost. Indeed, it is difficult to beat a CPU's performance, which is outstanding due to excellent hardware design and the success of optimizing compilers, especially handling large sequential random access memory.

We discussed practical design decisions to implement an MLG application on an FPGA. We observed performance and quality improvements using simple traversals without changing directions, such as firstly sorting (in a natural order) all x-axes, secondly sorting all y-axes, and thirdly sorting all $(\vec{x}\ \vec{y}\ \vec{z},\cdots)$. This fact makes implementation on an FPGA easier.

We found that the FORTRAN Shell-sort routine using simple traversals produced better quality results compared to using the same routine with only bubble-sorts. We also observed that the revised FORTRAN Shell-sort routine, on average, produced nearly the same or slightly better quality results compared to using insertion-sort, parallel bubble-sort, selection-sort or quick-sort. A Shell-sort implementation in C-based on insertion-sort using an efficient merge for the final phase is nearly twice as fast as the revised FORTRAN version.

After applying an unstable algorithm such as Shell-sort during the initial sorting phase, a single pass of the bubble-sort algorithm is iteratively applied to each axis. This stable algorithm yields optimal performance based on our tests. In most cases, doing more than a partial sort along any axis is inefficient unless the data is random; which cannot hold after recursive calls since the data is partially sorted.

We discussed using matrix transpose for this application. We observed slight speed-up (around 1.4×) running an FPGA once to compute 32 MLGs, compared to finding 32 MLGs using a CPU on the same system. For these comparisons, only the FPGA code involved matrix transpose. A CPU does not perform well doing matrix transposes for this application, based on runs on the target system. There is no reason to use matrix transpose for this application on a CPU because it has exceptional capabilities handling random accesses. For this application, the main operations are comparisons and swaps. If the main operations were more expensive to compute, then it is likely that our technique to handle matrix transpose would be useful for acceleration.

In usual practice, only a single MLG is computed. It may be desirable to find multiple MLGs by making various perturbations of the input data in order to choose the best MLG. We described an implementation for which significant speed-up is observed provided enough MLGs are found.

Although the SRAM banks are useful in connection with the multi-buffering feature, these banks are the bottleneck for this application. However, FPGAs are flexible devices and SRAMs are not needed. For example, Convey's HC-1system provides eight channels to external memory without any SRAMs on-board the chips. Moreover, Virtex-6 FPGAs, which are available in the HC-1ex, offer greater potential for speed-up due to the larger number of gates.

Acknowledgments

Gopal Patnaik proposed both the problem and matrix transpose as part of the solution. Stefan Möhl provided useful support revising the I/O code to transfer data between SRAMs and BlockRAMs. Eric Kinzie set up the environment to interactively run Xilinx synthesis software. Peter Tripician performed the setup to run the Xilinx tools on the compute nodes under PBS queuing system. Jeanie Osburn, head of the Operational Computer Section of CCS, provided resources to work on this project. This work was performed on the SGIRASC RC100 system and the SGI Altix ICE system at NRL-DC under the auspices of the US Department of Defense (DoD) High Performance Computer Modernization Program (HPCMP).

References

1. Bique, S. and R. Rosenberg, "Matrix Multiplication using Virtex-4 FPGAs on SGI RASC RC100 at the Naval Research Laboratory", *Proceedings of the HPCMP Users Group Conference 2011*, 2011.

2. Bique, S. and R. Rosenberg, "Fast Generation of High-Quality Pseudo-random Numbers and Permutations Using MPI and OpenMP on the CrayXD1", *CUG Proceedings*, 2009.

3. Boris, J., "A Vectorized "Near-Neighbors" Algorithm of Order N Using a Monotonic Logical Grid", *J. of Computational Physics*, 66, p. 1–20, 1986.

4. Dave, N., K. Fleming, M. King, M. Pellauer, and M. Vijayaraghavan, "Hardware Acceleration of Matrix Multiplication on a Xilinx FPGA", *Proceedings of Formal Methods and Models for Codesign (MEM-OCODE)*, Nice, France, 2007.

5. Dongarra, J.J., J. Du Croz, I.S. Duff, and S. Hammarling, "A set of Level3 Basic Linear Algebra Subprograms", *ACM Trans. Math. Soft.*, 16, 1990.

6. Dou , Y., S. Vassiliadis, G.K. Kuzmanov, and G.N. Gaydadjiev, "64-bit floating-point FPGA matrix multiplication", *ACM/SIGDA Field-Programmable Gate Arrays*, ACM Press, pp. 86–95, 2005.

7. Higham, N.J., "Accuracy and Stability of Numerical Algorithms", *SIAM*, 2002.

8. Kindratenko, V.V., R.J. Brunner, and A.D. Myers, "Mitrion-C Application Development on SGI Altix 350/RC100", *International Symposium on Field-Programmable Custom Computing Machines*, IEEE Xplore, 2007.

9. Lanzagorta, M., S. Bique, and R. Rosenberg, "Introduction to Reconfigurable Supercomputing", *Synthesis Lectures on Computer Architecture*, series editor Hill, M., Morgan & Claypool, 2009.

10. Mitrionics, Inc., *Running the Mitrion Virtual Processor on SGI RASCRC100, 2.0.3-001*, 2009.

11. Mitrionics, Inc., *The Mitrion-C Programming Language, 2.0.3-001*, 2009.

12. Picone, J.M., S.G. Lambrakos, and J.P. Boris, "Timing Analysis of the Montonic Logical Grid for Many-body Dynamics", *SIAM J. Sci. Stat Comput.*, 11, No. 2, pp. 368–388, 1990.

13. Pirsch, P., *Architectures for Digital Signal Processing*, Wiley, 1996.

14. Press, W.H., S.A Teukolsky, W.T. Vettering, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press Cambridge, pp. 331–332, 1992.

15. Xilinx, Inc., *ISE Design Suite Software Manuals and Help -PDF Collection*, UG681, v 12.2, 2010.

16. Xilinx, Inc., *Free ISE WebPACK -Earlier Software Releases*, http://www.xilinx.com/webpack/classics/wpclassic, 2010.

17. Yalamanchili, S., *Introductory VHDL: From Simulation to Synthesis*, Prentice Hall Xilinx Design Series, Prentice Hall, 2001.

ProjectHPC: A Multi-Tier Architecture for Simulation and Analysis

Jerry Clarke, Kelly Kirk, and James Collins
US Army Research Laboratory (ARL), Aberdeen Proving Ground, MD
{jerry.clarke, kelly.t.kirk, pat.collins4}@us.army.mil

Ankur Chopra
High Performance Technologies, Inc. (HPTi), Aberdeen Proving Ground, MD
ankur.chopra.ctr@us.army.mil

Kenneth Renard
WareOnEarth Communications, Inc., Aberdeen Proving Ground, MD
kenneth.renard@us.army.mil

Abstract

*For Mobile Network Modeling, Multi-Scale Modeling, and Blast Protection for Platforms and Personnel, US Army Research Laboratory is developing and providing a variety of codes and tools that require a wide-range of computer sophistication from the user. This diverse and complex software supports a range of interfaces that extend from simple, intuitive interfaces with a minimal number of options to scalable, parallel simulators that are submitted via a batch queuing system. Subject matter experts with little to no computer science experience will benefit from the delivered software environment, as well as expert users with intimate knowledge of the inner working of a parallel simulator. By definition, high performance computing attempts to attain the most performance from computational resources as possible. Usability is not a priority, which is absolutely as it should be; however, while a huge number of analysts and periodic users could potentially benefit from access to high performance computers, the "cost of entry" is just too high. Our solution is a multi-tier architecture that provides **expert user** interfaces for code developers, **computer scientist** interfaces for sophisticated users, and **analyst** interfaces for subject matter experts in fields other than computer science.*

1. Introduction

In 1963, before ZIP codes were introduced in the US, Glen Culler and Burt Fried described ***An On-Line Computing Center for Scientific Problems*** (Culler and Fried, 1963). They lamented about the difficulty scientists and mathematicians have taking full advantage of the "impressive achievements in computer hardware and programming techniques". They wrote:

> *The source of the difficulty is basically the poor communications between the "user" ... and the computer ... as well, upon the inherent difficulty of imparting to a programmer the detailed and specialized knowledge one acquires about a particular problem area after working in it for some time.*

Naturally, their problems were complicated by the fact that their RW-400 computer had only 1,024 words of memory and stored functions on an 80,000 word magnetic drum. Their interface was two 17-inch CRT oscilloscopes for graphics and an 8-inch CRT for alpha-numeric output.

So how far have we progressed from those heady days when the favorite programming language was solder? Even after the introduction of the Graphical User Interface (GUI), Windowing Systems, Client-Server Software Architectures, Web Services, Grid Computing and more, many of today's scientists and mathematicians would lament the same problems. In fact, the gap between "user" and "programmer" is widening. Effectively programming today's parallel systems with tens-of-thousands of cores and providing a simple-to-use interface to the desktop is challenging for even the most skilled programming teams.

During the mid 1990s and early 2000s, many meta-computing tools were available (and some still exist) that allowed the computer scientist to design complex distributed computing systems providing a variety of reusable layers. Most of these were designed by computer scientists, for computer scientists. Indeed, examining system diagrams and application programming interface (API) specification for many of these is daunting. Some of these tools relevant to meta-computing applications included (Allan and Ashworth, 2001): Legion, GLOBUS, AppLeS, CORBA, Nimrod, NetSolve, Synthetix, Chorus, InfoSpheres, Amoeba, MILAN, Arjuna, Apertos, GrassHopper, WAVE, Locust, HPVM, HPC++, CC++, MIST, GA, Fortran-M, HPF, Java, Raja/RMI, Jini, ANDF, DQS, NQS, LSF, Condor, NQE, LoadLeveler, and Cumulvs. These tools helped address some of the technical challenges and architectural issues involved in designing systems that can bring the full power of today's computing systems to users that are not computer scientists. Technical challenges; however, are only a piece of the overall problem. Local policy restrictions, particularly with computer security are a major concern. Additionally, systems that "make HPC easier to use" still require an in-depth knowledge of the target application, high performance computing (HPC) platforms, or both.

Today, multi-tier architectures have graduated from the halls of computer science departments and have become widely-used in the business community. They provide separation of the data, presentation, and business logic layers making it possible to design robust, maintainable application-specific systems that are used by subject matter experts in fields like accounting and medicine. They provide the scalability, flexibility and modularity that is difficult to reproduce in simple client-server architectures. They provide an excellent model for deploying systems of interest to the Department of Defense (DoD) that deliver a top-down solution to a scientific problem as opposed to a bottom-up solution that makes HPC easier to use.

Recent advancements in Python communication and Web frameworks, as well as advancements in JavaScript toolkits, have made developing and deploying multi-tier architectures easier. Building upon the widely-used *Computational Science Environment* (CSE) (Clarke and Vines, et al., 2010), the US Army Research Laboratory (ARL) is developing and deploying ProjectHPC. ProjectHPC is a multi-tier architecture that provides ***expert*** interfaces for code developers, ***computer scientist*** interfaces for sophisticated users, and ***analyst*** interfaces for subject matter experts in fields other than computer science. Maintaining multiple interfaces for various levels of expertise is imperative since exposing all functionality at all levels is counterproductive. This is similar in concept to various Linux distributions, where usability approaches the familiar Microsoft Windows (Holmes and Cox, 2011) for the general user community without removing the command-line interface beloved by experts.

The goal of ProjectHPC is ***not*** to design a generalized framework and service provider like nanoHUB (Klimeck, et al., 2006), rather it is to acquire and/or develop tools that take advantage of advancements in JavaScript toolkits and Python frameworks to deliver targeted interfaces to complex, distributed functionality. Initially, the focus is in three primary scientific areas: Multi-Scale Modeling, Mobile Network Modeling, and Blast Protection for Platforms and Personnel. While not inhibiting the expert user or computer scientist, ProjectHPC endeavors to deliver an interface to the analyst or subject matter expert that is focused on problem-solving, thus relieving the user from the complex tasks of marshaling the proper resources and managing the complex interactions between parallel computer simulations, data management, and a responsive user interface.

2. Computational Science Environment (CSE)

The Computational Science Environment (Clarke and Vines, et al., 2010) is a Python-aware environment developed around the common data model using the eXtensible Data Model and Format (XDMF). It provides a standard environment for data analysis, visualization, and software testing and evaluation. It incorporates many industry-standard software development APIs, cross-platform scripting languages, a large number of available analysis and visualization libraries and tools, code-control repositories, a build-system, and much more. It is the foundation for ProjectHPC.

XDMF provides the glue for CSE. It ties together scientific application codes, data analysis codes, and visualization tools in multiple ways. Codes that output their data in XDMF format can take advantage of visualization tools such as ParaView and Ensight. It also provides mechanisms for stand-alone, scientific application codes to share data during the course of a calculation, thus producing coupled calculations. The sharing of data can be through memory or disk, and on-the-fly. XDMF is flexible. One example is sharing data at well-defined simulation times between computational fluid dynamics (CFD) and computational structural mechanics (CSM) codes.

ProjectHPC builds upon the successes of CSE by taking advantage of XDMF and the many analysis tools and the rich development environment it provides. The core applications of ProjectHPC define an API for interfacing scientific application codes. Using this API, application developers can quickly build Web-based tools for solving complex engineering problems and can take advantage of the many tools CSE has to offer.

3. ProjectHPC

3.1 Architecture

ProjectHPC is a multi-tier architecture consisting of three main tiers (or layers): a User Interface Layer (UIL), a Logic Layer (LL), and a Compute Layer (CL), see Figure 1. The Compute Layer is not formally part of ProjectHPC; but, because it is central to the system, we refer to it as the third tier of the architecture. As mentioned in the Introduction, there are multiple interfaces into this architecture serving a variety of user needs. These interfaces are shown schematically in Figure 1 and described below.

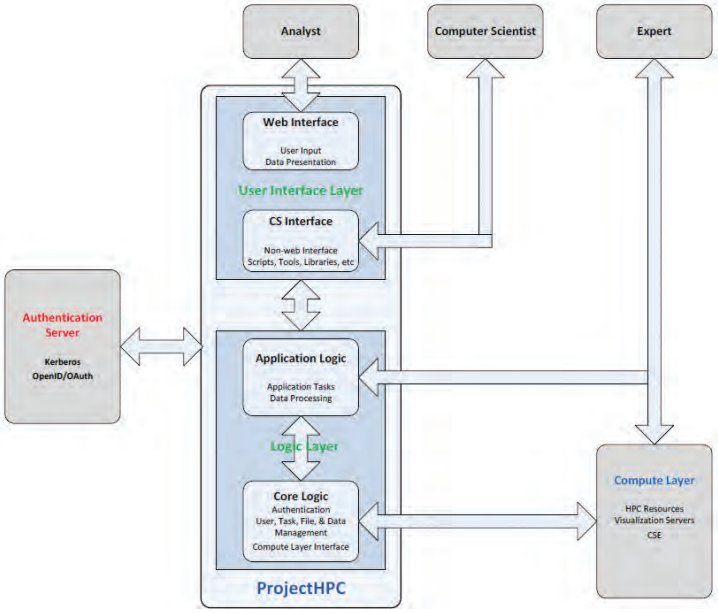


Figure 1. ProjectHPC Architecture

The User Interface Layer provides the "Analyst" and "Computer Scientist" interfaces into the system. The "Analyst" interface is web-based and gives access to workflows that solve specific DoD problems. This is for analysts who are only interested in results, and do not have the time or organization mandate to become HPC experts. To use the system, an analyst interacts with the webpage by selecting a problem, providing input and initiating tasks. Upon completion of the tasks, results are presented to the user in some useful format.

A second interface is provided for computer scientists. This interface gives access to the framework, and is for users who need more flexibility than the web interface provides. Computer scientists will have access to scripts, frameworks, and libraries that facilitate data manipulation, data modeling and visualization. They can take advantage of this layer by building web or other interfaces into workflows, or building additional application logic as necessary. A workflow defines a solution to a particular problem. For example, a workflow might consist of integrated simulations from multiple disciplines, such as CSM, CFD, Forces Modeling and Simulation (FMS), etc., just to solve a single problem. A major goal of ProjectHPC is to construct a framework so that workflows that solve problems of interest to DoD can be easily built and used. Computer scientists working with application code developers will be able to rapidly develop web interfaces to workflows that help design the next-generation military systems.

The Logic Layer is the middleware between the user interface layer and the compute layer. There are two main parts to this tier: the core logic and application logic. Core logic implements basic user, task, file, authentication and data management services. It also manages the interface into the Compute Layer. An API into the core logic is well defined. Application logic takes advantage of this API by implementing specific workflows of interest to end-users. At this layer, inputs are processed from the UIL, and associated tasks are created and managed. These tasks may include submitting jobs to HPC systems and providing the results back to the UIL.

The Compute Layer provides the computational resources necessary for solving many problems of interest to DoD scientists and engineers. While not formally part of the architecture, ProjectHPC's core logic will provide the hooks into the large High Performance Computing Modernization Program (HPCMP) HPC systems and an API to build interfaces to other Linux systems.

3.2 System Design

The design of ProjectHPC is based on the Django (Django Software Foundation, 2011) Web Framework. A set of core logic applications have been developed that extend, but do not alter, the Django code base. These core applications form the basis of the development environment for the application framework. Currently, the core applications are the User Manager, File Manager and Job Manager. The User Manager is integrated with Django's built-in user authentication system and extends it to include features like custom user permissions, custom user groups, supplemental user data and support for future OpenID integration. The File Manager manages all aspects of the user's interaction with the file system including adding and deleting multiple files, assigning home directories, etc. It mimics the UNIX hierarchical file structure in the database. The Job Manager defines what a Job is, sets the specifications for extended jobs that are defined in framework applications, registers extended jobs for polymorphic access, defines the operations that each job must perform and provides an API for job execution. It also provides the interface to the compute layer which includes HPC systems.

The application framework is designed so that HPC applications can be presented with relative minimal programming effort. As with other frameworks, applications must conform to a set of rules. In the case of ProjectHPC, they must conform to the core logic layer API. For each application that runs under ProjectHPC, application logic Python scripts are developed that define the application's tasks and that interface with the core logic layer. The underlying framework then provides abstractions for a number of activities needed to run the applications, such as interpreting requests, processing requests, storing data and producing responses. Thus, a developer writing code for the framework concentrates on the logic of the application instead of worrying about the server-side infrastructure.

ProjectHPC is designed so that published applications are completely self-contained and do not interfere with other applications running on the system. They can be added or removed without breaking the system, and all relational dependencies are confined within the application. This modular architecture not only makes it easy to deploy and retire applications, but also simplifies developing, updating and testing applications on the framework.

Users can interact with ProjectHPC at different levels of the User Interface (UI). The primary Web interface is designed for analysts that are only interested in obtaining results. The Computer Scientist (CS) interface provides the ability to communicate with applications using custom scripts and tools through a well-defined API. This provides the capability of tailoring web interfaces to applications and also building non-web interfaces to them.

Tasks, or execution units, are fundamental to the system. Tasks are generated when job creation and initiation requests are received from the UI Layer (browser or CS interface). The logic layer processes and tracks these tasks, and generates others based on the request. One example of a task is to run a job on the Compute Layer. Each job may have one or more such tasks; therefore, the core logic must run these tasks in the most efficient manner possible and must keep track of their status so that results are retrieved as soon as they are available. Once a job is completed, the results are accessible at the UI layer. Tasks are generated to do this.

ProjectHPC depends on open-source software for task management. Tasks are queued on to a message queue that implements the Advanced Message Queuing Protocol (AMQP). A listener monitors the message queue and delegates processing to available workers. One or more worker servers then concurrently process these tasks. Software called Celery (Celery, 2011) and its Django integration piece Django-Celery handles task processing. Rabbit MQ (Springsource, 2011) handles messaging and Kombu (Kumbu, 2011) provides the listener.

ProjectHPC relies on other open-source software as well. It runs on the Apache Web server with the Python's Web Server Gateway Interface module, mod_wsgi. It uses ParaViewWeb (Kitware, 2011) and other packages to display graphical results. More details on ProjectHPC can be found in Reference Clark and Kirk, et al., 2011.

An application called NetSim was developed for the Mobile Network Modeling Institute (MNMI) and integrated with ProjectHPC. Its purpose is to generate dynamic visualizations for both live data-feeds and NetDMF file-playback. Figure 2 and Figure 3 are screen-shots that show results visualized on ProjectHPC. Figure 2 is a snapshot of a communication matrix taken from a simulation. It shows live-device communication data from an NS3 (NS-3 Project, 2011) simulation. The x- and y-axis are IP addresses of the communicating devices.

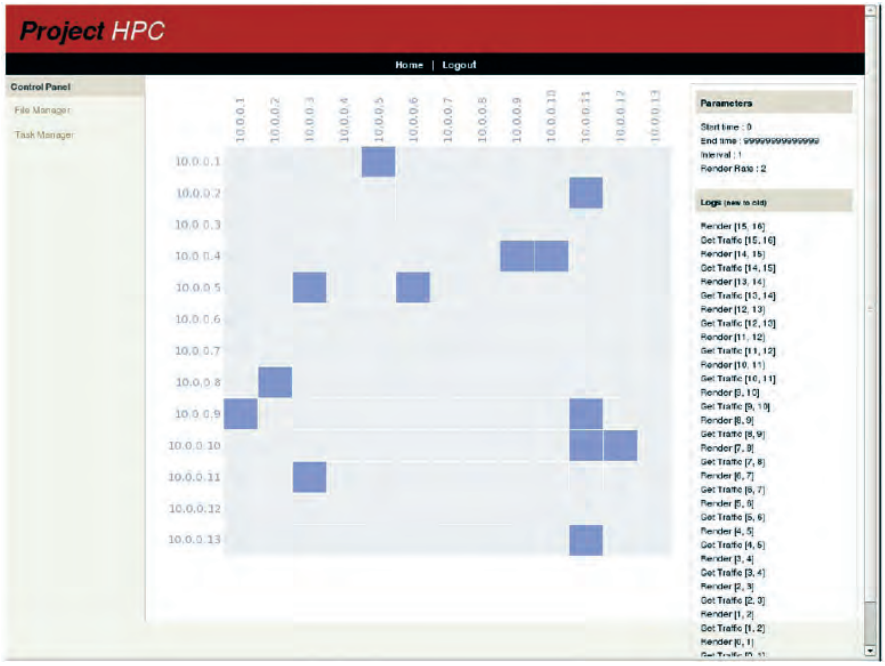


Figure 2. A snapshot of a device communication matrix taken from a live NS3 simulation. The colored boxes indicate active communication links between devices whose IP addresses are given on the x- and y-axis.

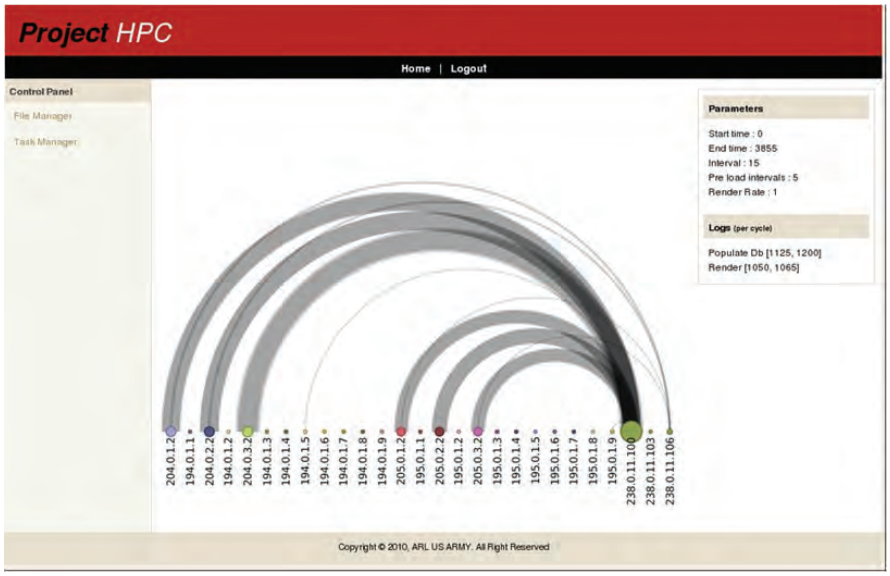


Figure 3. An arc diagram visualization playing an NetDMF communication file

Figure 3 shows communication between devices. The width of the connecting line represents the relative amount of data transferred. The size of the colored balls indicates the amount of data transferred and the darkness of the lines indicates overlaps.

Figures 2 and 3 show some of the capabilities of ProjectHPC as it exists today, but the vision is much greater. A glimpse of the future is presented in Figure 4. It shows a battlefield network simulation launched from a website where the user has simulation time access to the data and can easily interact with it. This example has a two-dimensional (2D) graphic showing connectivity between nodes on a battlefield with color-coded data transfer rates, and a three-dimensional (3D) graphic showing the terrain. The user will be able to interact with the data as the simulation progresses; in particular, the user will be able to view 3D data from whatever viewpoint that makes sense and make decisions based on the results. Simulations can be controlled with ease, all from a web interface.

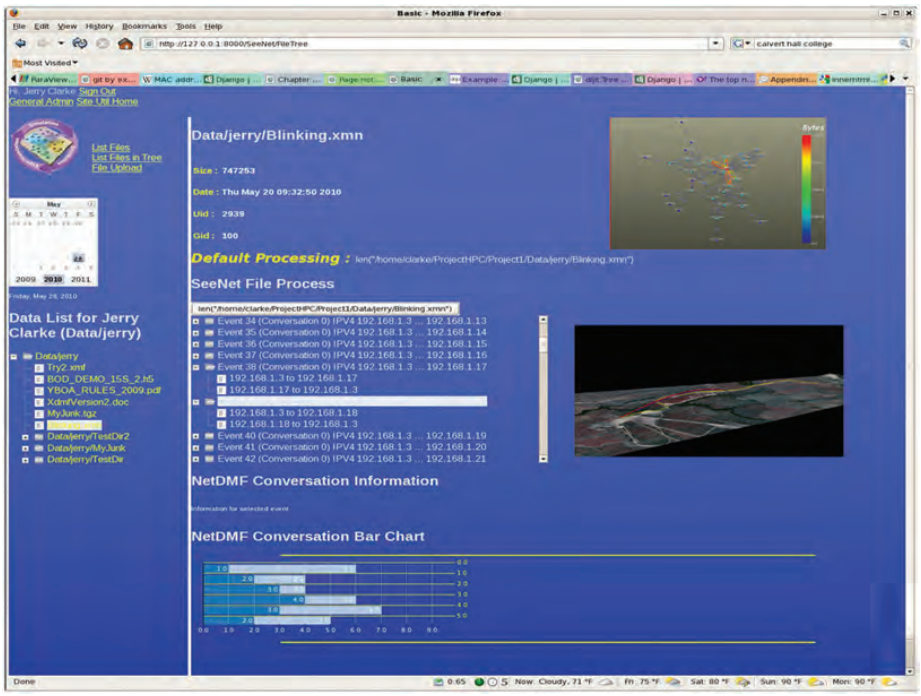


Figure 4. ProjectHPC's vision for the future

3.3 ProjectHPC Authentication

Multi-tiered web applications pose additional security challenges, since a web server is attempting to execute an action on the back-end tier on behalf of the user. This requires the web server to possess and forward authentication information that originates from the user. Typically, this information includes a secret that is meant to be shared only between the user and the back-end tier. These credentials can be vulnerable to a rogue web server or an impostor claiming to be the legitimate one. Even on the legitimate web server, a collection of credentials from a large set of users becomes a high-value target for attackers. A complicating factor is that most authentication systems do not limit the access granted by the use of their credentials. For example, a password or Kerberos ticket is usually capable of granting full user permissions to do anything that the user is authorized for. During transmission and storage on the web server, protection of these valuable credentials is essential.

Our approach to implementing security measures for ProjectHPC is to start with the current practice as developed and deployed in the User Interface Toolkit (UIT) project. This involves obtaining Kerberos credentials for the user on the web server by passing in the Kerberos principal name, Kerberos password, and a SecurID passcode. Users that do not have SecurID cards are registered to use HSAM (HPCMP Single-Use Authentication Method), which provides a passcode in place of the SecurID passcode. The Kerberos ticket obtained is first validated against a Kerberos principal known to the web server and is used for local authentication. The Kerberos ticket and secret key are then encrypted and encoded as a cookie that is sent back to the user. The encryption key stays on the server, only to be used by authenticated client connections when needed for authentication to the back-end system. This approach limits the storage of credential information on the server, as well as protects the credential in transit to and from the client.

This project is also looking forward to addressing other challenges of multi-tiered authentication systems by prototyping the use of OAuth as a way to limit credential capabilities, and to avoid sending credential information (passwords, passcodes) to the Web server. Currently, OAuth version 1 is being used in prototypes that rely on authentication directly to the back-end tier where the user must specifically authorize access to the back-end tier from the web server. The OAuth protocol flow first establishes a "request token" by authenticating the MTS to the back-end system, and then re-directs the user to the back-end system to authenticate themselves and authorize the pending transaction. Upon successful authentication and authorization, an "access token" is granted to the web server which it uses to authenticate on behalf of the user to the back-end system. The access token is limited in time and in scope based on the decision the user made at the time of authorization. The authentication and authorization steps are meant to protect and limit the authority granted to the web server and be quick and easy for the user to execute.

To that end, authentication to the back-end system could be provided by a PKI authentication or another web-based authentication such as OpenID. We intend to integrate OpenID as an authentication system to this environment as implemented by the HPCMP security infrastructure. Coordination and cooperation with the HPCMP security staff is the foundation of these efforts and can serve as a model for future authentication of web services.

4. Conclusion

ProjectHPC implements an architecture that can provide targeted interfaces to complex software to analysts and subject matter experts who are only interested in results, and who do not have the time or organization mandate to become HPC experts. These targeted interfaces fit into the NIST definition for cloud computing (Mell and Grance, 2011). It provides convenient, on-demand access to shared applications that solve complex military problems using the "Sofware-as-a-Service" service model. ProjectHPC, however, goes beyond that. ProjectHPC provides multiple interfaces into the architecture. Computer scientists that need more than just web access will have an interface into the system that allows them to take advantage of the framework API to build their own applications and interfaces. Expert users will have access to the application interfaces. In short, ProjectHPC targets the analyst but doesn't lose sight of the more sophisticated user's needs.

ProjectHPC will eventually be bundled with the Computational Sciences Environment and will be made widely available to the DoD community.

References

Allan, R.J. and M. Ashworth, *A Survey of Distributed Computing, Computational Grid, Meta-computing and Network Information Tools Survey*, Computational Science and Engineering Department, Daresbury: UKHEC, 2001.

Celery, *Celery - The Distributed Task Queue*, 2011, http://www.celeryproject.com/ (accessed May 13, 2011).

Clarke, Jerry, et al., "A Common Computational Science Environment for High Performance Computing Centers", *HPCMP User Group Conference*, Chicago, 2010.

Clarke, Jerry, Kelly Kirk, James Collins, Ankur Chopra, and Kenneth Renard, "ProjectHPC: A Multi-Tier Architecture for Simulation and Analysis", Report, APG: Army Research Laboratory, 2011.

Culler, Glen J. and Burton D. Fried, "An On-line Computing Center for Scientific Problems", *Proc. 1963 Pacific Computer Conf.*, Piscataway, N.J., IEEE, pp. 221–242, 1963.

Django Software Foundation, *Django - The Web Framework for Perfectionist with Deadlines*, 2011, http://www.djangoproject.com/ (accessed May 13, 2011).

Holmes, Marc and Simon Cox, "Delivering End-to-End High-Productivity Computing", *Microsoft Architect Journal*, 2011.

Kid, *Kid*, 2011, http://kid-templating.org/ (accessed May 13, 2011).

Kitware, ParaViewWeb, June 1, 2011, http://www.vtk.org/Wiki/ParaViewWeb (accessed Sept 21, 2011).

Klimeck, Gerhard, et al., "NEMO 3-D and nanoHUB:Bridging Research and Education", *Sixth IEEE Conference (IEEE-NANO 2006)*, pp. 441–444, 2006.

Kumbu, *Python Package Index: kumbu 1.1.3*, 2011, http://pypi.Python.org/pypi/kombu (accessed May 13, 2011).

Mell, Peter and Timothy Grance, *The NIST Definition of Cloud Computing (Draft)*, NIST, Gaithersburg, MD, 2011.

NS-3 Project, *The NS-3 Network Simulator*, 2011, http://www.nsnam.org/ (accessed May 13, 2011).

Springsource, *RabbitMQ - Messaging That Just Works*, 2011, www.rabbitmq.com (accessed May 13, 2011).

Secure Remote Visualization Services with PKIVNC

Randall Hand
US Army Engineer Research and Development Center, DoD Supercomputing Resource Center (ERDC DSRC), Data Analysis and Assessment Center (DAAC), Vicksburg, MS
randall.e.hand@usace.army.mil

Dave Pratt
Secure Mission Solutions
david.pratt.ctr@hpcmo.hpc.mil

Abstract

In this paper, we present a new service available on the Enhanced User Environment (EUE) Utility Server (US) called PKIVNC, short for Private Key Infrastructure Virtual Network Computing. This new service will allow users to connect remotely to the Utility Server for remote display access at higher frame rates than previously available, enabling access to visualization applications previously difficult or impossible to use. In addition, the service will enable remote access to high-speed graphics accelerators suitable for high-quality rendering, GPGPU development, and simulation acceleration. We will cover the design and security considerations of this new service, as well as the user experience and capabilities.

1. Introduction

The principal objective of the Secure Remote Visualization Services initiative is to provide highly interactive rendering of locally processed data on a remote researcher's desktop. Traditional visualization techniques (Desktop Sharing, X11) are plagued with performance problems for all but the shortest of network paths. Variations on the Virtual Network Computing (VNC) protocol exist to address much of the bandwidth issue, but these solutions fail to meet Department of Defense (DoD) directives for secure access to the High Performance Computing Modernization Program (HPCMP) systems. The PKIVNC system addresses both concerns, providing the framework for encrypted, authenticated data channels between remotely administered, specialized graphics compute engines and a researcher's workstation.

The PKIVNC toolset is a group of scripts and utilities to coordinate the creation, control, and tear down of the pipelines and data sources running on each of the communication endpoints. Users interact with a single client on their Mac, Linux, or Windows XP workstation communicating with the PKIVNC tools on the remote sites. Together, they automate the connection semantics, establishing secure connections, managing remote processes, and spawning local clients to view the results. This frees researchers from the connection details, allowing them to focus their efforts where they belong—on their work.

The PKIVNC toolset was built to run on the new Enhanced User Environment (EUE) Utility Servers, a collection of HPC-like resources deployed at each DSRC within the program. These Utility Servers come in three "flavors" enabling use of large-memory, high core-counts, and graphics hardware acceleration. The Utility Servers share a common filesystem with the HPCs in the "Center-wide File System" (CWFS), enabling local filesystem access to data from HPC simulations without data transfer or migration.

2. System Overview

At a minimum, a PKIVNC system consists of three nodes: a client node, a login node, and a compute node. External access is limited to the login node, to which multiple remote users may simultaneously connect. The compute nodes have full TCP/IP connectivity, but will generally not be externally visible. Access to them is solely through the login node. One is free to place the login node anywhere within the network partition as long as external users can ssh to it. Similarly, compute nodes need only ssh connectivity to the login node. A queuing system (e.g., PBS Pro) running on the login node allocates exclusive access to a set of compute nodes to a given user for a specified time.

Communication between a client node and compute node is proxied through the login node. Command and control messages passed between all nodes are transactional, encapsulated within an XML data block. They provide the

infrastructure support necessary to prepare the remotely rendered image stream for display on a researcher's desktop. In short, a client "talks" to the login node, which in turn, makes requests of the queuing system and the compute nodes, ultimately establishing a VNC connection between the two.

3. User Connection Mechanisms

As with any remote visualization technique, network topology plays a significant role in the responsiveness of an application. A PKIVNC system can be thought of as having two separate connection components; the first provides the OpenGL optimizations necessary to quickly render three-dimensional (3D) images, and the second, a network backbone upon which that information is transported.

The network bridge between a client and the compute nodes at a PKIVNC site is Kerberized ssh. As such, no special software or login mechanism is required for access from any DSRC. A PKIVNC user simply obtains a Kerberos ticket normally, and subsequent authentication and credential forwarding are transparent. Upon connection, TCP ports are forwarded between hosts to establish a secure, authenticated data pathway for the compute server and the client to communicate.

The OpenGL component is handled by an application suite called TurboVNC/VirtualGL.[1] Briefly, this works its magic by intercepting an application's OpenGL calls (via LD_PRELOAD) and rendering them into a pixel buffer (pbuffer) off screen. The beefed-up Utility Server compute nodes do the graphics heavy lifting, render into the pbuffer, and the software suite reads that buffer directly and sends the now two-dimensional (2D) version of the image down the pipeline.

The PKIVNC system supports two connection methods, each aimed at a different network topology. These are as follows:

a. **VNC Mode**
   To take full advantage of the processing power on the compute nodes on a Utility Server, VNC mode provides the TurboVNC/VirtualGL coupling described above. This mode requires only a TurboVNC client on the researcher's desktop. For most users, this will be their primary connection method. VNC mode provides reasonable performance for high-latency, low-bandwidth networks. The performance may be further fine-tuned by altering one of the many TurboVNC viewing parameters available. As this document is not intended to serve as an abbreviated version of the TurboVNC manual,[1] interested users are referred to the official documentation for a complete listing. This mode is labeled "X11 Image Transport" in the TurboVNC documentation.

b. **X11 Mode**
   X11 mode consists of a straight, X11 connection, forwarded via ssh. The client need only have an X server installed to access an EUE site. This connection mechanism is intended for fast networks with little latency. Performance is greatly influenced by the third-party X application used to visualize data. Rendering is client-side, with the local X server doing the bulk of the work. As a general rule, applications with limited user interfaces and modest user interactions will perform well under X11 mode.

4. Installation/Configuration Overview

Compute and login nodes software requirements consist of a cluster environment, PBSPro, to manage jobs on that cluster and Perl. Configuration of the compute node cluster and login node queuing system is outside the scope of this paper and will be assumed already established. A single file at each node provides the majority of the configuration parameters for PKIVNC, primarily defining paths to the locally installed tools. Compute nodes require slightly more work to install and configure, owing to the addition of the TurboVNC/VirtualGL software. In particular, access to the high-speed graphics hardware must be enabled. As this exposes some additional security risk, the ramifications are more fully described in the next section.

With the software suite installed server-side, attention can be turned to the clients. Here, too, workstations need a current Perl installation, with Windows users having a choice between ActiveState2 and Cygwin3 Perl. A Java client (JRE 1.6) is also available for those who shun the command line or cannot install Perl directly. As with the server, each client is configured through a single file. The TurboVNC viewer is the final client-software requirement.

[1] http://www.virtualgl.org/

[2] http://www.activestate.com/activeperl

[3] http://www.cygwin.com/

5. Security Concerns

Connection security is paramount in the PKIVNC design. The guiding principle from the start has been to minimize deviations from the existing HPCMP security infrastructure. This section aims to provide a high-level assessment of the security framework PKIVNC is built upon and the strategies in place to mitigate any variances. It is meant as a brief overview for security personnel and system administrators to weigh the risk of implementing the system at their site. The definitive document for security-related information remains the PKIVNC Security Access Guideline.[4]

a. **Network Infrastructure**
First, and foremost, the communication pathway between PKIVNC nodes is through the existing Kerberos ssh infrastructure. No processes require privilege escalation, and all PKIVNC commands run as the user. As the PKIVNC system is a client/server application, it relies heavily on forwarding the data stream through the encrypted, authenticated ssh pipeline. This necessitates enabling both TCP and X11 forwarding on the sshd daemons at both the login node and the compute nodes. It is anticipated that most HPCMP sites currently allow X11 forwarding, and a large percentage currently have TCP forwarding on as well. Little operational risk is anticipated from enabling either.

b. **Direct Hardware Access**
In order for users to directly access the high-performance graphics hardware on a compute node, they must be granted trusted access to the X11 server. The ramifications are that the currently executing user, armed with the X11 daemon's magic cookie, will have full access to all X sessions on that node. This risk is mitigated by granting access to the cookie via a PBS prologue script running as root. Every time a job leaves the PBS queue, a new X11 cookie for the console display is generated on disk, made readable by the job's owner, and is merged into the user's own .Xauthority file. That access remains valid until the next job is released. That user also becomes the sole member of group 'vgluser', the only group granted read access to the hardware device driver. This mechanism is sufficient to ensure only a single user has trusted access to the X11 daemon at a time.

c. **Local Host TCP Ports**
The client and the compute nodes open TCP ports at each endpoint, and these are forwarded through the ssh tunnels. The tunnels are both encrypted and authenticated from client to login and from login to compute node. At each host; however, the interprocess TCP ports are unencrypted. As a result, it is possible for a logged-in third party on the compute, login, or client node to intercept and relay the VNC data exchanged. This effectively permits a read-only compromise of the 2D image stream. The risk is equivalent to existing port forwarding threats. If this is deemed unacceptable for your site, the following additional steps may be pursued:

At the compute node, the queuing system should be configured to ensure that node is accessible only to the specified user for the lifetime of the job. As the VNC server opens only one port on a single compute node, this is sufficient to remove the threat completely. This feature is currently not implemented.

A login node cannot be locked down in similar fashion, as it must provide multiuser access. The risk is partially mitigated, as this node is limited to HPCMP-authorized users only. As all login nodes are under our complete control, kernel-level logging of any attempt to connect to a forwarded ssh port by a user who does not own that port should be flagged. Such access should be reviewed as a potential attack. This feature is currently not implemented.

The client nodes provide the greatest threat for compromise, as these systems are frequently out of an organization's control. For that reason, no viable risk reduction exists to address the potential for a read-only compromise of the 2D image streaming back from the compute node.

d. **VNC Authentication**
Full VNC session hijacking on the client node is prevented by a triple-des encrypted, eight-character, one-time password challenge/response between the VNC client and the server. This password is stored on the compute node filesystem, and the VNC server will begin listening for connections only if that file is read-only and owned by the specified user.

The password is regenerated for every connection attempt and is invalidated if no connection is made within an administrator-defined timeout. The client receives this password along the encrypted ssh tunnel and, depending on implementation, writes it to a temporary, automatically deleted file, delivers the password directly to the VNC

[4] https://www.hpcmo.hpc.mil/security/kerberos/

server via stdin, or requires the user to manually enter the password.

It should be noted that one-time password authentication is scheduled for removal in the next release of PKIVNC. Authentication will be replaced with a PAM module looking solely at process ownership on the compute node. If the process making the connection is owned by anyone other than the user running the VNC server, authentication fails.

The network flow between the VNC viewer and the VNC server is illustrated in Figure 1.

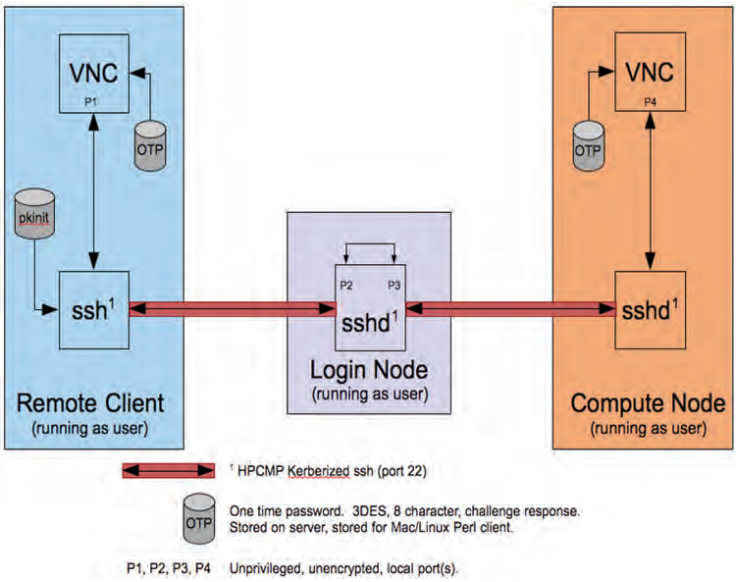


Figure 1. PKI/VNC network connectivity

6. User Interface

The PKIVNC client comes in two flavors: a Java application, and a command line Perl interface. Your system administrator will select the client most appropriate for your site. Both are functionally equivalent, with the Java client hiding the underlying PKIVNC command requests.

a. **Perl Client**

The Perl client is generally located on your site's standard path and is named "**clientNode.pl**". This script provides the means to query, submit, connect, and remove jobs from the login node. The syntax is simply as follows:

clientNode.pl *command options*

where the available commands include INFO, SUBMIT, STATUS, CONNECT, …, and CLEAN. Commands are meant to be self-explanatory, with terse help available as shown below. The PKIVNC User Guide[5] provides a detailed explanation of each command and its options.

clientNode.pl help *command*
clientNode.pl help help

In the absence of an explicit command on invocation, clientNode.pl will enter interactive mode and begin a prompt, read, execute loop of the commands typed. Depending on your terminal type, this gains you a command history and additional line editing functions. It also provides greater diagnostics to help isolate error conditions. At most sites, your system administrators will have configured the PKIVNC software suite such that end-users need only execute it from their favorite shell. If this is not the case, refer to the PKIVNC User Guide for full details.

b. **Java Client**

Help is available from within the Java client and will not be repeated here. To launch the client, navigate to the directory in which you unpacked the distribution and type

[5] https://www.hpcmo.hpc.mil/security/kerberos/. Included in all PKIVNC packages.

java –jar Harda.jar

First time users, however, have one optional step to complete first. As described earlier, the PKIVNC clients need to know what remote PKIVNC sites are available, how to access them, and where the supporting executables are located on your workstation. For Perl users, your system administrator will have taken care of these details for you. The Java clients; however, store this information in a system-defined, local "preferences file". The exact location is immaterial, but you need a means of providing the initial values for your site. A template file exists within the distribution directory named "*HardaClient.xml*". Edit the file with a text editor replacing the highlighted entries with the full pathnames to the specified executables on your workstation:

<entry key="vncViewer" value="/opt/TurboVNC/bin/vncviewer"/>
<entry key="sshCommand" value="/usr/KRB5/bin/ssh"/>

Once configured for your workstation, you may initialize the client with the following command:

java –jar Harda.jar HardaClient.xml

Once loaded, you can drop the last argument for subsequent invocations. Future changes to the preference files are best accomplished within the **View/Preferences** dialog, described more fully in the Help dialog within the application.

7. Job Script

The PKIVNC environment attempts to be nearly invisible to the user's environment. In general, no special requirements or setup are needed to access and use the system. Further, the applications you run on the compute nodes and the scripts used to ready, launch, and process your data should require no modification. The single "hook" into the PKIVNC environment rests within the job script submitted to the queuing system. The job script may include any and all commands normally available to you outside of a PKIVNC run. It may also include any legal queuing directive, with the exception of a queue specifier (e.g., #PBS –q foo). The final line of the script, however, must be:

/app/SRVS/pkivnc/bin/hardaRun.pl *someExecutable*

In general, *someExecutable* will be the full path to the OpenGL application used to render your data or the wrapper script to launch it. This job script will remain running throughout the lifetime of your job submission and should not be backgrounded. If your site has the PKIVNC suite installed to a non-standard location, the path to hardaRun.pl will need to be updated.

8. Conclusion

The PKIVNC environment entered Beta Testing in early 2011 with great success. Reports from users were almost entirely positive, focusing on the highly interactive frame rates they were able to achieve from distant locations. The Beta Testbed was located in Vicksburg, Mississippi, with Beta tests conducted with users from Maryland, Ohio, and Pennsylvania; with satisfactory results from all.

Users have so far been able to execute standard visualization applications like ParaView, VisIt, and Ensight, as well as "home-grown" applications thanks to the universal support of VirtualGL. In addition, some users have begun using interactive IDE and debuggers like TotalView for their applications, reducing the time spent debugging and improving their time-to-solution.

Future work on this product includes an effort to remove the need for a user-installed client entirely, pushing it onto the web via Java Applets and standard web server technologies. Code-named SPARROW, this product will reutilize large portions of the PKIVNC effort to create a thin-install system available entirely within a standard Java-capable web browser.

Supercomputing: SaaS, PaaS, or IaaS?

Albert Reuther and Jeremy Kepner
MIT Lincoln Laboratory, Lexington, MA
{reuther, kepner}@ll.mit.edu

Abstract

Cloud computing has been a ubiquitous topic in the press. The technical underpinnings have certainly solidified in the past two years. Recently, the National Institute of Standards and Technology (NIST) has published a series of articles on cloud computing, including one on the definition of cloud computing. This definition includes essential characteristics, delivery models, and service models. In this article we will cover these parts of the NIST cloud computing definition. Next, we will discuss what concepts have been a part of supercomputing and high performance computing.

Then we will discuss our experiences in applying cloud computing concepts to our high performance computing capability, LLGrid. Using the LLGrid platform, we have been experimenting with providing services in each of the three service models. We have deployed LLGrid Portal and LLData along with some web services as Software-as-a-Service (SaaS), and we have enabled the installation of third-party software in the Infrastructure-as-a-Service (IaaS) model. But the most impact we have had is in providing a Platform-as-a-Service (PaaS) capability for writing research code using pMatlab and MATLAB as the primary application programming interface (API). With this PaaS, several of our users have prototyped SaaS services. Through this experimentation, we believe that some aspects of cloud computing are very applicable to supercomputing enabling greater flexibility and availability of high performance computing capabilities.
Keywords: cloud computing, high performance computing, LLGrid, Software-as-a-Service, Platform-as-a-Service, Infrastructure-as-a-Service

1. Introduction

The topic of cloud computing has been pervasive in the technical press and has even entered popular advertising. ("To the Cloud!" exclaim the Microsoft ads.) While some of the press and advertising have stretched the definitions of cloud computing, the technical underpinnings have certainly solidified in the past two years. Back then, two types of cloud computing were clear: utility cloud computing vs. data-intensive cloud computing. Utility computing involves the outsourcing of IT services, while data-intensive cloud computing entails an architecture for large-scale data analysis (like Apache Hadoop[1] and Google File System[2], BigTable[3], and MapReduce[4]).

There has been much progress in understanding and categorizing cloud computing offerings in the past two years. While there are many definitions of cloud computing, the definition set forth by the National Institute of Standards and Technology (NIST) captures the technology suite well:

> "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services), that can be rapidly provisioned and released with minimal management effort or service-provider interaction. This cloud model promotes availability, and is composed of five essential characteristics, three service models, and four deployment models."[5]

In Section 2, we will expand on this NIST definition to populate the taxonomy for cloud computing offerings. Once we have explained this taxonomy, we will be able to recognize the types of services that traditional high performance computing (HPC) systems can deliver to scientific computing communities, which we will discuss in Section 3. The discussion in that section will include how we have experimented with several cloud services on LLGrid. Finally, we will end with a summary and conclusion in Section 4.

2. Taxonomy of Cloud Computing

2.1 Essential Characteristics

Let us revisit the definition of cloud computing from NIST from Section 1. There are a several phrases to note in this definition, but they are brought out and expanded in the an explanation of five essential characteristics, three service models, and four deployment models mentioned at the end of the NIST definition. First, NIST enumerates the five essential characteristics as the following: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. The following paragraphs further develop each of these essential characteristics.

1. *On-demand self-service*: Users are able to automatically self-provision resources on-demand without human interaction with service-providers; so users generally do not need to interact with individuals via phone or email to change their resource utilization. Also, there is no delay in such changes; the update is immediate.
2. *Broad network access*: The capabilities are available over the network via thin or thick clients such as smart phones, desktops, laptops, etc. Anywhere there is a network connection and device; users should be able to access their cloud resources. Generally, users should not be limited to accessing their cloud resources from only certain devices, operating systems, platforms, etc.
3. *Resource pooling*: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, and the consumer generally has no knowledge of the location of resources that are made available to them. The resources could be anywhere in the world. Naturally, cloud service providers only put a finite (though seemingly-infinite) number of computers into service, and their resource pool is sized to accommodate the statistical worst-case scenario of all users over time, not the sum of all worst-case scenarios. Further, users generally do not know whether they are sharing certain compute nodes, network switches, disk drives, etc. with other users.
4. *Rapid elasticity*: Each user can increase or decrease resources quickly, as needed. This can be automatic, though it does not necessarily need to be, and total resource capacity appears to be unlimited to the users. As one would expect, there are limits to how many new resources a provider can bring online in a given time-frame, but generally providers have done the necessary analysis to cover virtually all user needs for a given time-frame.
5. *Measured service*: The provider monitors, controls, and reports resource usage both for managing the amount of resources each user consumes, and for which each user is charged. In other words, the providers are able to quickly determine how their resources are being used and can predict when more resources need to be brought online to accommodate user needs.

These five essential characteristics mimic those of a utility company. For example, with electrical service, we can determine how much electricity we want to use without first calling the electric company every time we want to turn on the toaster. Electricity is ubiquitous and accessible from every electrical outlet. We all share the generator station and delivery network, and we generally do not know where our electricity is coming from. The supply of electricity can be increased or decreased readily by the electric company by bringing more generators online up to a limit, but to users the supply appears unlimited. And finally, the electricity meters on our houses and workplaces measure how much electricity we are consuming for billing and resource allocation purposes. (And with smart grid technologies that will be deployed in the near future, the feedback of usage to the electric companies will be continuous and in real-time rather than discrete monthly meter-reading events.)

This utility-like nature is the foundation of the commercial cloud computing capabilities. Utility cloud computing is designed to host traditional server applications, enable portability of data with applications, offer capacity on-demand, and deliver a lower total cost of ownership for many traditional enterprise applications. At its root, utility cloud computing enables outsourcing of information technology services, and generally it is designed to work for a great number of concurrent, independent users operating on millions of data sets involving terabytes of data.

However, there is another paradigm of cloud computing that has received some press coverage, though not nearly as much as utility cloud computing. This other paradigm is data-intensive cloud computing. Data-intensive cloud computing is the large-scale data analysis that occurs within the utility cloud computing provider companies like Google, Yahoo, Facebook, Amazon, and Microsoft Azure. These capabilities are built to handle billions of records per day, trillions of stored records, and petabytes of storage. The first papers that provided insight into this cloud computing paradigm was about the Google File System[2], Google BigTable[3], and Google MapReduce[4]. Yahoo, Facebook, Microsoft, open-source developers, and other companies established the Apache Hadoop project[1] to create an open-source version of the technologies that Google described in their papers. These systems are designed for performance and scalability, optimized for data ingest and query, and present a simplified application programming interface (API). These data-intensive cloud

computing capabilities also embody the utility nature of cloud computing for the smaller communities for whom they are built. In many ways, this data-intensive cloud computing infrastructure has similarities to cluster-based high performance computing environments.

2.2 Service Models

Within the cloud computing paradigm, three service models have emerged: Software-as-a-Service (SaaS)—"run apps on the cloud"; Platform-as-a-Service (PaaS)—"write apps for the cloud"; and Infrastructure-as-a-Service (IaaS)—"install apps in the cloud". In SaaS, consumers use applications provided to them by the cloud. Examples include Google Docs, Gmail, SalesForce.com, Facebook, Amazon Simple Storage Service (S3), and NetFlix Streaming. When consumers use provided tools and development environments to create and deploy their own applications, this is PaaS.Such platform services which include Amazon Web Services (AWS), Facebook Apps, Microsoft Windows Azure, and Force.com. Finally, users can deploy their own application, generally within virtual machine (VM) instances, in the IaaS model. Providers of IaaS are Amazon Elastic Compute Cloud (EC2), Joyent, Rackspace, VMWare, and others. In order to better understand these service categories, we must first explain a generic computer system stack including VMs, which are commonly used in all three service models. This system stack is depicted in Figure 1, and each of the layers of the stack is described below:

- Applications and their User Interfaces – Programs that users run;
- Application Programming Interface (API) – Rules and specifications for libraries and frameworks;
- Libraries/Frameworks – Reusable software routines for building applications;
- Operating System (OS) – Manager of computer hardware resources, and of common services for application software;
- Virtual Machine (VM) – Software implementation of a computer that executes programs like a computer machine;
- Hypervisor – Virtual operating platform that monitors the execution of one or more VMs; and
- Hardware – Physical components of the computer.

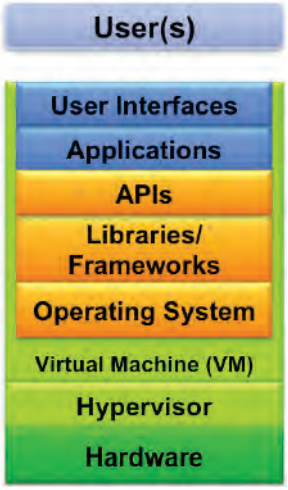


Figure 1. Computer system stack

The key interfaces for cloud computing are the User Interfaces, APIs, and Hypervisor for SaaS, PaaS, and IaaS, respectively. These key interfaces are depicted in Figure 2. Using user interfaces, SaaS delivers entire software packages to users across the network. The consumer is generally not authorized to manage or control any of the underlying infrastructure including libraries, operating system, virtual machines; nor computational, network or storage hardware. For certain applications, though, consumers do have control over their application preferences. PaaS shares a set of APIs and tools with which applications can be developed and executed. While developers often have some control over their application deployment, they do not have any control over the distribution of their application; nor do they have control of the hardware on which their application executes. For IaaS, the Hypervisor interface enables the execution of VMs provided by users and executes the VMs on infrastructure hardware. Also, IaaS often allows users to deploy virtual private networks (VPNs) to segregate their system environment from other system environments. Simply put, SaaS enables the running of applications, PaaS enables the writing of applications, and IaaS enables the installation of applications.

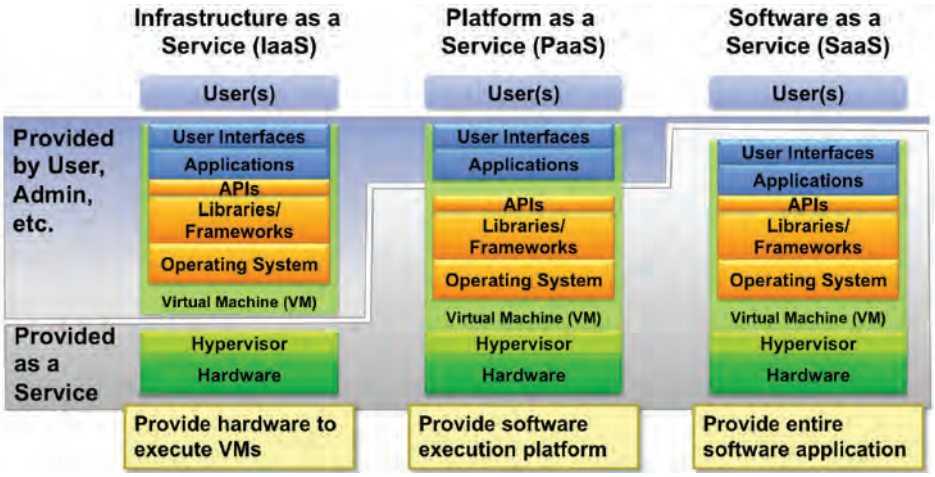


Figure 2. Key interfaces for cloud computing service models

2.3 Deployment Models

The final dimension of cloud computing is the deployment models. These four models explain what communities of users have access to the cloud computing resources, who is responsible for deploying the cloud resources, and the relationship between the two. The four models are public, private, community, and hybrid cloud deployments.

- We are all familiar with public clouds like Amazon Web Services and Google App Engine. Public clouds are accessible by the general public, and they are deployed and managed by public cloud companies that sell their services.
- Private clouds are accessible only within an organization. The cloud resources may be deployed by the same organization that can access it, or it may be contracted to another organization.
- Community clouds are established for a certain community of organizations with a similar mission. The DoD is an example of a community that has established several community cloud instances. As with private clouds, the resources may be deployed by one of the community organizations or may be contracted to another organization.
- Hybrid clouds are composed of combinations of two or more of the above three models. For example, an organization with a private cloud may establish the technical and business relationship to overflow into a public cloud infrastructure when the demand for cloud resources overflows the internal private cloud resources (often called cloud bursting). This requires a standardized cloud interface and infrastructure between the cloud infrastructures involved.

To summarize, Figure 3 depicts the three dimensions of the cloud computing taxonomy: the two utility paradigms, three service models, and four deployment models. In the next section, we will discuss the similarities of this taxonomy with high performance computing. Also we will discuss how we have experimented with high performance computing within the two utility paradigms and the three service models, all delivered with a private cloud deployment model.

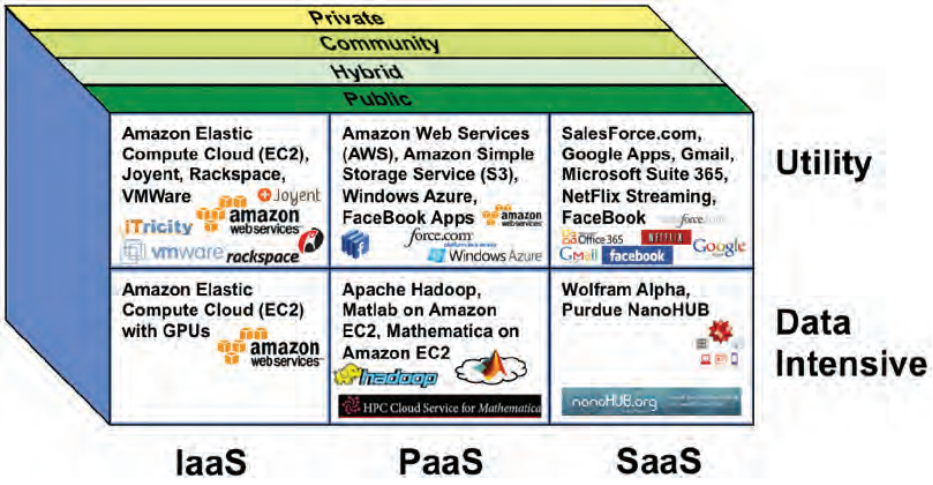


Figure 3. Cloud computing taxonomy cube

3. Cloud Computing and HPC

In both the research and commercial cloud computing communities, there has been debate as to whether research computing, and high performance computing in particular, can take advantage of cloud computing. How feasible is it to provide SaaS, PaaS, and/or IaaS in an HPC environment?

In conversations with fellow HPC and cloud computing researchers, the idea that cloud computing has some of its roots in supercomputing has been shared by many. Since the days of the first supercomputers over five decades ago, central supercomputing capabilities have been deployed and shared within an organization (like a private cloud) or research community (like a community cloud). The grid computing technologies that came to the forefront in the late 1990s and early 2000s enabled organizations to overflow their HPC jobs from their organizational supercomputing assets to other supercomputers in their community (like a hybrid cloud). Some of these grid computing communities, like TeraGrid, include so many organizations and research teams that they look a lot like public clouds.

Generally, these supercomputing assets are a mix between utility computing and data-intensive computing. These supercomputing centers generally reflected the five essential characteristics of cloud computing, at least to the extent that they could before commodity clusters and virtual machines made the realization of the five characteristics much easier. At supercomputing centers, users can submit jobs of various sizes to the batch queue system for execution, though more sophisticated services of the supercomputer required some interaction with the help desk. The supercomputers often are on the same network as the researcher's terminals or desktop computers, so there is generally broad network access. Resources are pooled for the use of all the users of the supercomputer systems, and if programmed appropriately, users can scale-up the execution of their applications from using a few processors to as many as the supercomputer can accommodate; thereby delivering reasonably rapid elasticity. And most supercomputing centers employ an accounting process of allocating CPU-hours to each user or research group per year, and measuring how much of that allocation they used during the course of their processing.

With regard to service models, most supercomputing centers provide common libraries such as mathematics libraries (like Atlas, FFTW, and MKL) and communication libraries (like Message Passing Interface (MPI) and CORBA) as the platform with which applications are developed and deployed. In this way, supercomputing has foreshadowed PaaS capabilities. It is often difficult, if not forbidden, to install your own commercial applications, as in the IaaS service model on supercomputing systems, without interacting with the administrators and managers of the supercomputer. However, with permission, such applications can be and are deployed, thereby showing how IaaS can be delivered in supercomputing environments. Finally, some organizations have deployed SaaS-like services by providing web-based and GUI-based interfaces to common parallel supercomputing codes. Examples of this include the Platform Computing Platform Application Center[6] and the Purdue University/Florida State University/NSF nanoHub[7].

At Lincoln Laboratory, we have also wrestled with the question of providing SaaS, PaaS, and/or IaaS in an HPC environment. Because of security considerations, we can deliver only private and community cloud deployments, and our community is generally other Department of Defense (DoD) users. Further, due to the research nature of Lincoln Laboratory work, our cloud research tends toward the data-intensive cloud computing paradigm, while many of the possible cloud services will strongly display utility computing characteristics. That leaves the bulk of the discussion to determining which service model(s) are applicable to LLGrid. Using the LLGrid platform, we have been experimenting with providing services in each of the three service model categories. The following subsections will describe the services with which we have experimented and deployed.

3.1 PaaS and LLGrid

LLGrid provides the usual set of libraries, compilers and tools that other HPC systems have like MPI, Atlas, C/C++/FORTRAN compilers, etc., as well as a full Java development platform. But by far, the most impact we have had for Lincoln Laboratory is in providing a PaaS capability for writing interactive, on-demand prototype and research code using pMatlab and MATLAB as the primary API. We developed three technologies that enable algorithm developers to develop parallel MATLAB codes with minimal changes to their serial codes and enable them to run parallel MATLAB jobs transparently on the LLGrid On-Demand Grid Computing system[8]. The three toolboxes that we developed are:

- **MatlabMPI** for point-to-point messaging (available at http://www.LL.mit.edu/MatlabMPI),
- **pMatlab**[9] for global array semantics (similar to High Performance FORTRAN) (available at http://www.LL.mit.edu/pMatlab and includes MatlabMPI), and
- **gridMatlab** for integrating user's computers into the LLGrid and automatically allocating grid computing resources.

These technologies have combined to create a unique on-demand grid computing experience, whereby running a parallel MATLAB job on the grid is identical to running MATLAB on the desktop. Users can run their parallel MATLAB jobs from their desktops on Windows, Linux, Solaris, and Mac OS X computers. The users launch their jobs onto any of a number of LLGrid clusters. The LLGrid cluster nodes and user desktop/laptop computers all mount a large shared central parallel file system.

Figure 4 depicts the PaaS software stack for pMatlab applications. The kernel layer is comprised of MATLAB along with its many toolboxes, MatlabMPI for messaging, and gridMatlab for launching onto clusters. MatlabMPI consists of a set of MATLAB scripts that implement a subset of MPI, allowing any MATLAB program to be run on a parallel computer. The key innovation of MatlabMPI is that it implements the widely-used MPI "look and feel" on top of standard MATLAB file input/output (I/O), resulting in a "pure" MATLAB implementation that is exceedingly small (~300 lines of code). Thus, MatlabMPI will run on any combination of computers that MATLAB supports. MatlabMPI uses a common file system as the communication fabric, which enables easier debugging of parallel program messaging and compatibility across any OS platform that runs MATLAB or Octave. The gridMatlab toolbox transparently integrates the MATLAB on each user's desktop with the shared grid cluster through a cluster resource manager like LSF, PBS, PBS Pro, and GridEngine; when a MatlabMPI or pMatlab job is run by the user in his or her MATLAB session, gridMatlab automatically amasses the requested LLGrid computational resources from the shared grid resources to process in parallel with the user's MATLAB session. The gridMatlab toolbox interfaces with the underlying scheduling resource manager to run interactive, on-demand jobs. While the cluster system and the MatlabMPI and gridMatlab toolboxes are integral to the functionality of the PaaS capability, the pMatlab toolbox is the application programming interface with which algorithm designers write their parallel MATLAB codes, as depicted in Figure 4. pMatlab is a partitioned global address space (PGAS) toolbox, which allows a MATLAB user to parallelize their program by changing a few lines of code.

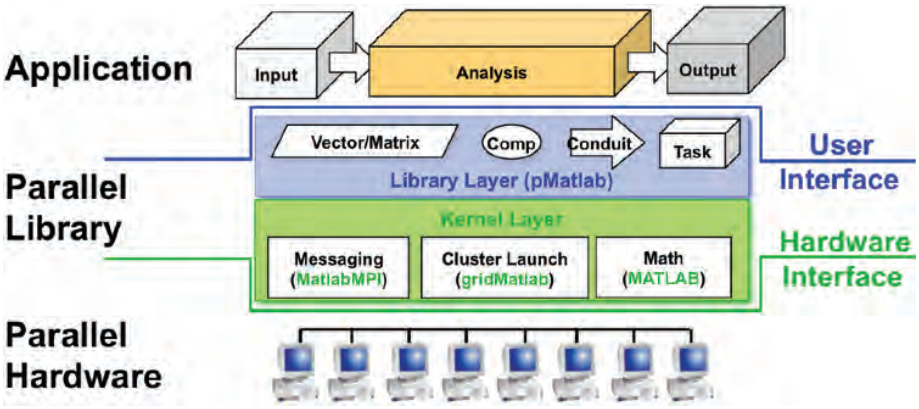


Figure 4. pMatlab/MATLAB software stack

3.2 SaaS and LLGrid

Over the past three years, we have been using LLGrid as a research platform for service-oriented architecture (SOA) application development research. Back then, it involved running multiple Java service applications using Apache Tomcat as LLGrid jobs through the scheduler and making service calls to those dynamically running application servers. Apache Tomcat is an open-source Java application server environment that is particularly useful for deploying web services and server-side web applications. There were two primary challenges to this effort. The first was taking advantage of certain configuration parameters to log each server's output to separate directory structures. This was not the default configuration for Tomcat, because at that time it was usually not deployed on shared resources. The second was determining to which computer the web service client should send the service request because the LLGrid scheduler did not launch a given service on the same compute node every time. We solved this hurdle by developing and deploying our own database-backed dynamic domain name service (D-DNS) within Lincoln.

3.2.1 LLGridPortal – LLGrid as a Service

Since then, we have expanded our Software-as-a-Service research efforts. Another SaaS capability that we have developed is LLGridPortal. The goal of the LLGridPortal project was to make HPC resources easily accessible to a wide population of scientists and engineers, regardless of their computational background and network security requirements.

The Portal interface was to provide the look and feel of running applications on the desktop, when in reality they are running on a Department of Defense (DoD) supercomputer or other HPC system with higher security requirements. Several considerations had to be made in order to adapt the LLGrid system to DoD HPC resources. These considerations are mainly concerned with security:

- DoD systems require authentication with common access card (CAC) credentials;
- Mounting network file systems via NFS, CIFS, etc. across organizations has historically been difficult;
- DoD networks only allow certain network ports to pass through routers, and this varies by organization;
- The number of additional computer applications installed on the desktop should be minimized; and
- The applications that users can execute on the HPC system and file access of the HPC file system should be limited.

With these security requirements in mind, we have implemented several unique LLGrid components that enable the portal technology, each of which will be described in the following paragraphs: CAC-enabled Apache Web-based Distributed Authoring and Versioning (WebDAV) server, Linux file system watcher, gridMatlab for Portal, and grsecurity kernel patches.

Web-based Distributed Authoring and Versioning (WebDAV) is a set of web protocol extensions that enable remote file system access over the SSL web port 443. All modern operating systems, including Windows7, Linux, and Mac OS X, provide WebDAV support natively by enabling WebDAV shares to be mounted as a network file system. The Apache web server can be configured to provide WebDAV access, but it did not include CAC authentication. We added CAC authentication to the Apache web server so that anyone with a CAC can authenticate and mount the prototype LLGrid file system onto their computer via WebDAV.

We needed to avoid installing and enabling secure shell (ssh) on each of the user desktops (especially the Windows desktops), but we needed a way to get HPC system status, launch jobs, abort jobs, and get job status[10] without making a remote procedure call. By modifying the Apache WebDAV server, we designed and implemented a Linux file system watcher. The watcher monitors all file system activities, and executes actions depending on the activity and filename. We have defined filename patterns and XML file contents forgetting HPC system status, launching jobs, aborting jobs, and getting job status. When a matching filename appears in the file system, the watcher executes scheduler scripts and returns a response XML file into the same folder as the activity file.

The third part of this effort was to modify gridMatlab to write and read the aforementioned activity and response XML files. The gridMatlab toolbox enables desktop MATLAB environments to interact with HPC schedulers to launch and abort parallel MATLAB jobs, get HPC system status, and get job status.

The above technologies limit users to only execute the applications for which XML file-based launching is enabled. To further secure access to HPC system resources, we have deployed the grsecurity kernel patches[11]. Grsecurity provides white-list role-based access control (RBAC) for all operating resources including file system access, process tables, process status (/proc), stack memory, socket access, etc. By white-listing access to all resources on the HPC system, we have highly-granular resource access control for each user and have monitoring capability that can indicate user errors and system exploitation. With these technologies, we have enabled the LLGrid system on a prototype system, and we are capable of deploying HPC systems up to a protection level 2 (PL2), which enables access for multiple programs with different need-to-know levels at the same classification level on the same system.

With LLGridPortal, we are able to deliver an on-demand, interactive, parallel rapid-prototyping to DoD scientists and engineers and others with rapid-prototyping needs in a higher-security HPC environment. All of the services are delivered over port 443 using the ubiquitous interface of a mounted drive (network share)—a very powerful service with a common, simple interface.

3.2.2 LLData – Large-Datasets as a Service

In image processing research programs that have enormous datasets such as persistent surveillance and LiDAR, data is usually stored on many enterprise-class storage appliances (that is, inexpensive but big file systems) that deliver a great amount of affordable storage but relatively slow throughput, thereby frustrating analysts and algorithm developers. (We usually see throughput between 100–250MB/s on such appliances.) Furthermore, the information about the datasets, the dataset metadata, is often stored in relational databases that refer to files in the large-file systems. (Or the dataset metadata is stored in the memory of the users, a sort of tribal knowledge.) Even when the dataset metadata and dataset are stored together within a relational database, the scalability of the database is rather low. Performance suffers as the dataset grows. Our efforts to overcome these challenges come under our LLData project, which aims to deliver large-datasets as a service—a kind of SaaS.

To address the poor-performance and scalability of enterprise-class storage appliances and their file systems, we have experimented with and deployed many instances of the Lustre parallel file system. Lustre is an open-source file system that is deployed over a set of servers. One of the servers is the file system metadata server (MDS), which holds directory and file information. For high-availability installations, a second MDS can be deployed as a hot spare by connecting it to the same disk array as the primary MDS. The rest of the servers are deployed as object storage servers (OSSs), which house all of the data. File blocks are striped across the OSSs so that the read and write activities of a given file are shared across all of the OSSs on which the file has been striped. With a properly-configured Lustre file system and similarly well-configured network infrastructure, data throughput and latency performance is tremendous. We have built Lustre clusters from commodity servers as well as Data Direct Network file servers, and have achieved throughput in excess of 1.3GB/s, even with sets of 16 commodity servers and only using the disks within the commodity server chassis. We have heard talks by Department of Energy (DOE) lab researchers that have achieved over 100GB/s throughput with very-large Lustre file systems with hundreds of OSSs. For the foreseeable future, we are planning to use Lustre file systems as our central storage capability in LLGrid. While this solution addresses the file server performance problems many image processing research teams face, it does not address the dataset metadata challenges.

Even with database clustering technologies, the scalability of relational databases is limited both in performance and manageability. This has prompted efforts in data-intensive cloud computing to break through these bottlenecks. We are exploring two such technologies: Apache Hadoop[1] and SciDB[12].

There are two components of Apache Hadoop that are applicable to this problem, Hadoop Distributed File System (HDFS) and HBase. HDFS is a block-based, distributed cloud file storage service. HDFS is extremely scalable, and it is integrated with Hadoop MapReduce and other Hadoop services, and it underlies HBase. Using the Java or command-line interfaces, users can insert and retrieve files from HDFS. By default, file blocks are chunked into 64 MB blocks, and these blocks are replicated across HDFS servers. While the block size can be modified to be very large, this may be a hindrance to image processing within HDFS. Also, the lack of a mountable file system interface can make it frustrating to work with.

HBase is a low-latency, loosely-structured distributed key1/key2/value triple store. It stores these triples across distributed servers with replication. It features write-once/read-many storage with versioned entries. It has a robust Java API, and there are ways to interface to it with an SQL-like interface (like using Hive[13]). HBase is a scalable alternative to relational databases, but it is not intended for large data objects like images. HBase is best for storing very large amounts of sparse metadata.

An open-source array-based distributed database called SciDB shows promise for providing a scalable means for storing image metadata and data in the same database. SciDB is being developed by a team of researchers, including MIT Prof. Michael Stonebraker, the developer of the Ingres and Postgres databases. In SciDB, the basic building-block is an array instead of tables and rows within the tables, and it allows these arrays to be nested as arrays-of-arrays. Partitions of each array are subdivided into tiles and stored on distributed servers, and this tiled-data can include overlapping data regions across the distributed servers. Similar to HDFS, there is no data overwriting. SciDB includes data versioning as well as provenance, the lineage of the data. SciDB will have multiple language bindings including R,C++, Ruby, Python, MATLAB, IDL, and others; and the SciDB arrays will have native data structure binding in code similar to that of Hibernate and Ruby-on-Rails. Beyond the native data structure interface, SciDB also has an SQL-like interface for ingestion, search, operation, and retrieval. With respect to array operations, SciDB will support array joins as well as array mathematics within the distributed database. Over the next year, we intend to further explore the applicability of both the Hadoop and SciDB technologies to large-image database storage, searching, and processing.

3.2.3 Other Current SaaS Efforts

Other efforts include incorporating Jetty and Apache Tomcat servers as a Java thread within MATLAB to receive and respond to SOA client-server service requests. This will allow us to deploy rapidly-developed MATLAB algorithms and other software-as-software services. We are prototyping image data services and image processing algorithm services that are based on the pMatlab/MATLAB PaaS environment described in Section 3.1. In the coming years, we intend more LLGrid users to have prototyped SaaS services using the pMatlab/MATLAB PaaS.

3.3 IaaS and LLGrid

At the most basic level, we have enabled the installation of third-party software in an IaaS model since the beginning of LLGrid. Users are encouraged to install whatever software packages they need in their own accounts with little or no interaction with the LLGrid team. Taking the IaaS service model further, we recently started experimenting with launching virtual machines (VMs) as job processes on LLGrid. We are collaborating with several cyber security teams that need to

be able to scale their testing environment from a few VM instances on their desktops and group servers to thousands of VMs on a large shared computational resource, i.e., LLGrid. The researchers need to configure their own VMs to launch on LLGrid, and LLGrid provides a virtual private network (VPN) on which the VMs connect their network interfaces. The VPN includes a dynamic host configuration protocol (DHCP) server so that each of the VMs will acquire their own unique identity. Eventually, the cyber security teams would like to emulate tactical networks like disadvantaged users, by including virtual network link emulation switches and routers into the VPNs.

We are just in the initial stages of evaluating the launching of VMs on LLGrid. One of the primary concerns we have is whether we can make VM launching and configuration easy enough for most users to do by themselves. This may become a capability that is reserved for research teams that are very familiar with utilizing VMs.

4. Summary and Conclusion

In this paper, we pondered whether high performance computing and cloud computing could intermingle. We set the stage by presenting a taxonomy of cloud computing based on the NIST definition of cloud computing. We used the five essential characteristics of cloud computing (on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service) to present the similarities and difference between utility cloud computing and data-intensive cloud computing—the use paradigm. The second-dimension of the taxonomy is the service models of PaaS, SaaS, and IaaS. SaaS enables the running of applications; PaaS enables the writing of applications; and IaaS enables the installation of applications. Finally, the delivery model explains what communities of users have access to the cloud resources. The clouds can be private, public, community clouds, or combinations of these as in the hybrid case.

We then discussed the similarities between this cloud computing taxonomy and high performance computing and supercomputing. Finally, we presented a number of capabilities that we have developed or will develop using LLGrid as a foundation.

Through our experience, we are confident that some aspects of cloud computing are very applicable to supercomputing for enabling greater flexibility and availability of high performance computing capabilities. In the commercial world, many organizations have embraced cloud computing because it allows them to outsource their information technology infrastructure, and it allows them to concentrate on their core business rather than IT. Some DoD entities will likely follow the same rationale. They have become reasonably convinced that the cloud security concerns have been addressed, and they have begun deploying some of their software applications on clouds. However, DoD HPC providers are not to this point. Our experience suggests that some HPC projects could be launched that could explore the employment of cloud computing concepts, but that further effort and research is required to offer the right services.

Acknowledgements

This work is sponsored by the Department of the Air Force under US Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.

References

1. Apache Hadoop Project, http://apache.hadoop.org/.

2. Ghemawat, S., H. Gobioff, and S.T. Leung, "The Google File System", *19th ACM Symposium on Operating Systems Principles*, Lake George, NY, Oct. 2003.

3. Chang, F., J. Dean, and S. Ghemawat, et al., "BigTable: A Distributed Storage System for Structured Data", *OSDI'06: Seventh Symposium on Operating System Design and Implementation*, Seattle, WA, Nov. 2006.

4. Dean, J. and S. Ghemawat, "MapReduce: Simplified Data Processing on Large-Clusters", *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, San Francisco, CA, Dec. 2004.

5. Mell, P. and T. Grance, "The NIST Definition of Cloud Computing (Draft)", *NIST Special Publication 800-145*, National Institute of Standards and Technology, January 2011.

6. Platform Application Center, http://www.platform.com/workload-management/high-performance-computing/add-on-products/platform-application-center.

7. nanoHub.org, Online Simulation and More for Nanotechnology, http://nanohub.org/.

8. Bliss, N.T., R. Bond, J. Kepner, H. Kim, and A. Reuther, "Interactive Grid Computing at Lincoln Laboratory", *Lincoln Laboratory Journal*, vol. 16, no. 1, 2006.

9. Kepner, J., “Parallel MATLAB for Multi-core and Multi-node Computers”, *SIAM*, June 2009.

10. Reuther, A., J. Kepner, A. McCabe, J. Mullen, N.T. Bliss, and H. Kim, “Technical Challenges of Supporting Interactive HPC”, *Proceedings of the High Performance Computing Modernization Office (HPCMO) Users Group Conference (UGC) 2007*, Pittsburgh, PA, 18–22 June 2007.

11. grsecurity, http://www.grsecurity.org/.

12. Cudre-Mauroux, P., H. Kimura, K.-T. Lim, J. Rogers, R. Simakov, E. Soroush, P. Velikhov, Wang, M. Balazinska, J. Becla, D. DeWitt, Heath, D. Maier, S. Madden, J. Patel, M. Stonebraker, S. Zdonik, “A Demonstration of SciDB: A Science-Oriented DBMS”, *VLDB’09*, Volume 2, Number 1, 1534-1537, Lyon, France, August 2009.

13. Apache Hive, http://hive.apache.org/.

Utilizing Advanced FORTRAN 2003/2008 for High Performance Computing

Michael List and David Car
US Air Force Research Laboratory, Propulsion Directorate (AFRL/RZ), Wright-Patterson AFB, OH
{michael.list, david.car}@wpafb.af.mil

Abstract

FORTRAN has been a principle language for High Performance Computing since its formative years. The number of computational fluid dynamics codes and related utilities written in FORTRAN is truly immense. With recent advances in compiler implementation of the 2003 and 2008 standards, it has now become possible to greatly enhance the software development process using advanced data structures and to interface directly with C routines and libraries. Examples of these advances are the polymorphic capabilities and the intrinsic ISO_C_BINDING module. Utilization of advanced programming practices greatly enhances programmer productivity. Software becomes extensible, maintainable, and testable. As the current era of rapid hardware architecture development continues it is both prudent and necessary for the high performance computing software developer to adapt.

1. Introduction

FORTRAN has been one of the most widely-used programming languages for scientific computing for decades. It is commonplace in many computational fluid dynamics research codes, as well as countless geometry libraries, mesh-generators, and post-processors. Since its inception, it has been improved and modernized, with many new language features added due to the success of such features in other major languages.

Advanced FORTRAN 2003 concepts have been used to formulate tools and to allow access to the general-purpose computing on graphics processing units (GPGPU) directly from FORTRAN. Specifically, the topics of advanced programming patterns, meta-programming through Python pre-processing, and direct access of C libraries are discussed. Illustration of these advanced concepts is made through implementation of a basic matrix solver and sample programs using bindings to CUDA. These basic building blocks are critical to formulation of adaptive software tools which can work on multiple hardware architectures. With the introduction of FORTRAN 90 capabilities (modules, pointers, types), concepts such as reference counting and creating modules which function as classes have been introduced. These concepts are crucial for taking full-advantage of features introduced in FORTRAN 2003 and 2008, specifically type-extension (inheritance). The addition of powerful preprocessing to implement templating in FORTRAN makes the language competitive with C++. A brief overview of use of a C interface generation utility is also provided, which utilizes the ISO_C_BINDING module to allow existing FORTRAN programs to take advantage of the rapidly growing set of C and C++ libraries.

This paper offers a brief survey of some of the new features of FORTRAN 2003 and 2008, as well as a brief description of utilities and practices which can increase the effectiveness of computational software. The discussion relates directly to enabling technologies for development of complex data structures for computational fluid dynamics (CFD) solvers and tools. More detailed descriptions of some of the new standard features can be found in References 14 and 15.

2. Methodology

2.1 Advanced Programming Patterns

Many existing CFD codes use outdated programming patterns to produce difficult to extend “spaghetti” code. Such pieces of software fail to encapsulate data or take advantage of the surprisingly complex and useful features of FORTRAN 90. These features include modules, types, and interfaces.

2.1.1 Modules as Classes

Akin[1] discussed the use of FORTRAN 90 modules to program using the object-oriented paradigm. Modules were used to manage a set of related functions and subroutines and the types upon which they operated. This fundamental concept was identical to the concept of a C++ class. The public and private attributes were used to restrict user access to portions of the module, thereby protecting portions of the software defined by the programmer. This practice became essential when designing library components to be used and extended.

Car[2] described a reference counting pattern which utilized the concept of modules as classes to manage memory for FORTRAN types. This was a practical way to track access of data that was shared amongst multiple portions of a piece of software and ensure proper de-allocation.

2.1.2 Type-Extension

The introduction of type-extension (polymorphism and inheritance), in conjunction with the use of type-bound procedures, greatly enhanced the capabilities of FORTRAN. As many other modern languages have embraced object-oriented concepts, this feature has helped to matriculate FORTRAN into the modern era of programming.

Polymorphism allows a number of patterns to be utilized and followed. One such pattern is the Strategy and Puppeteer design pattern[7,16]. This set of patterns has been used to develop a solver framework which can accommodate from reduced-order models to full three-dimensional (3D) solvers.

The Strategy pattern gave a consistent interface to the user of the pattern—pre-update, update, and post-update. Each algorithm implemented its own unique set of the three functions. During any of the three steps execution may pass to another strategy, perform a calculation or communication, or simply return. Figure 1 demonstrates this. The benefit of this type of implementation was the decoupling of the specific algorithms from the calling code—allowing the algorithms to be easily interchangeable. Examples for implementation include abstraction of perfect, ideal, and real gas models in CFD codes; turbulence models; integration schemes; and matrix solvers.

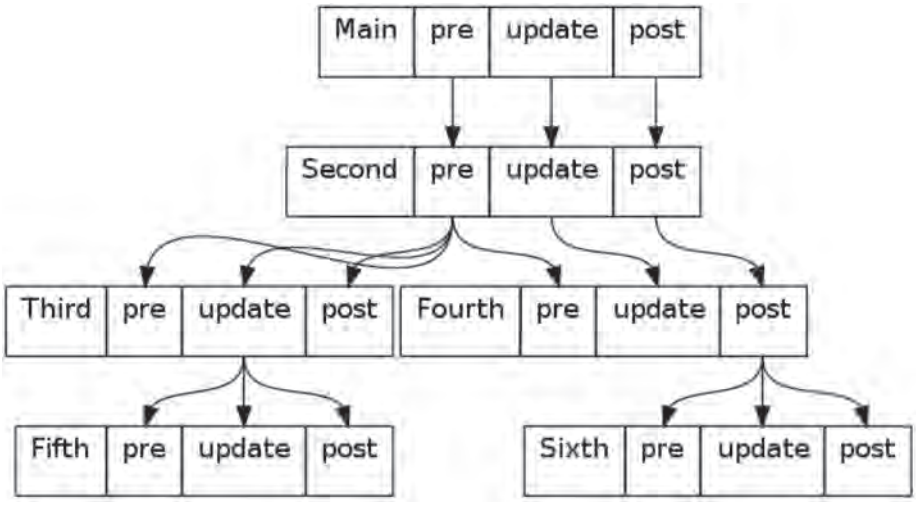


Figure 1. A simple strategy pattern tree structure example

The puppeteer pattern, shown in Figure 2, provided a control structure and intermediary for a number of disparate systems (each having a predictable and well-defined application programming interface [API]). In this sense, a puppeteer acted as an aggregation device, for instance collecting information into one packet for a Message Passing Interface (MPI) send or distributing after a receive. It has interfaces for data exchange and management of pieces, without exposing the pieces to one another. This keeps those pieces from becoming circularly dependent upon one another.

The key application of polymorphism to such patterns in FORTRAN is the creation of an abstract base-type to selectively control the way in which all user-defined patterns will operate. Developers can then enforce a specific API for each type in a program through proper use of `abstract`, `interface`, and deferred type-bound procedures.

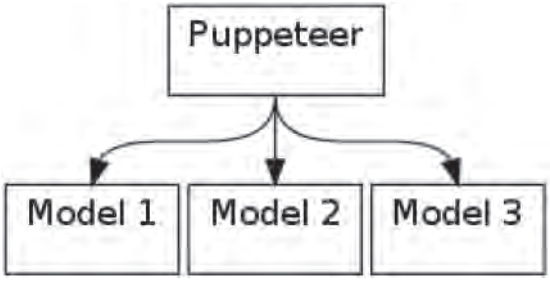


Figure 2. Simple example of a puppeteer pattern

2.1.3 Type-Bound Procedures

Type-bound procedures enhance the object-orientation concept. The process of binding functions and subroutines to types allowed such entities to have access to "methods" which may or may not have been included within the scope in which they were invoked. This (combined with type-extension) made it possible to write portions of code which did not necessarily know upon exactly what *type* of object they were operating, simply what *class*. This became important for construction of libraries and utility classes.

2.2 Meta-Programming

Meta-programming is the process of "writing code that writes code." Modern scripting languages, such as Python and TCL, and most compiled programming languages are excellent tools for generating sections of code based upon user inputs. Such methodologies have the option of utilizing existing token parsing libraries to create special markups within a standard programming language. This effectively extends a programming language.

To give an example of implementation of the meta-programming principle, templating capabilities have been produced for FORTRAN[3,11] and have allowed generation of exceptionally reusable code. Making use of the library of code corresponding to[3] grants access to hash tables, linked lists, and string utilities. From these basic building blocks a series of tools useful for CFD and data reduction can be generated. These units can be reused and can encapsulate various intrinsic and user-defined types—the primary advantage being the ability to follow the Do not Repeat Yourself (DRY) principle (see also Reference 8). This approach was key to the success of the aforementioned solver framework.

2.3 ISO_C_BINDING Module

The ISO_C_BINDING module provides a standard in-source method for calling C code directly from FORTRAN, as well as share interoperable variables. Prior to the FORTRAN 2003 standard, it was necessary to write preprocessing macros within the C code that would be called from FORTRAN. It was potentially difficult to properly select the types that would be sent between the two languages and to ensure bytes were properly aligned. With the new standard, a set of kind specifiers for the intrinsic FORTRAN types was used to make variables interoperable. Also, a bind (C) attribute can be used to make type, function, and subroutine declarations which allow C structs and functions to be accessed.

Other advantages to using the ISO_C_BINDING module exist:

- It is possible to call C library functions directly from FORTRAN by providing the correct interface. The standard C library contains a large number of capabilities important for operating-system utilities. Some of these, for instance regular expression matching, could be useful for interactive scientific tools.
- Existing solvers and tools written in C or a C-accessible language can be quickly wrapped to give access to components.
- Newer architectures, such as general-purpose computing on graphics processing units (GPGPU), use C or C-like programming languages to create code for the devices. Having an ability to access such pieces makes it possible to more quickly improve and adapt existing software.

2.4 Unit-Testing

Unit-testing is the systematic testing of individual components—typically methods, functions, or subroutines—of a software project with the intent of ensuring successful operation or functionality[9]. Implementation implies that small programs were written to check all of the components of a software program by providing sample inputs which caused

known results or side-effects. The outputs or results were then checked against these known values, and their compliance was reported to the user. The degree to which a component's functionality was tested is known as "code coverage" and is an important part of the extreme programming methodology[6].

A number of FORTRAN unit-testing frameworks have been developed in recent years[4,5,10,13]. Each took advantage of meta-programming principles. Unit-testing has been extensively used in developing many CFD components, including those described in the next section.

3. Results

The remainder of this document discusses examples of advanced programming methods which can be used to gain efficiency in developing software without severely degrading performance.

3.1 Matrix Solver

A number of matrix solving utilities were developed, originally as a means of comparing performance of highly optimized C++ and FORTRAN code. The FORTRAN version of the solver used the templating concept and pointers to cleanly access data for a simple Gaussian elimination with pivoting. It was found that the pointer usage did not negatively impact the speed of the software, but did make the code easier to modify.

Traditionally the elements of a matrix would simply be housed in an array. Incorporating more complex data structures and module construction habits; however, it was possible to generate a consistent interface for many types of matrices, including both full- and sparse-block matrices, all of which could be composed of integers, reals, or double-precision values. The change of intrinsic type was handled through a template argument[3]—this essentially means that a pre-processor performed string substitution to ensure the correct `integer`, `real (kind=4)`, or `real (kind=8)` declaration.

The outline of a sparse-block matrix class which employed the compressed row storage scheme is given in Appendix A. A module was defined containing types and functions and subroutines which operate upon those types. While this may seem excessive, definition of accessors and setters made management of the data simpler and contained. It also provided a consistent API which could be used when defining, for instance, a tri-diagonal matrix or a septa-diagonal matrix class. These could all then be used interchangeably in various generic functions. When matrix operations were performed; however, the internal data storage array was used directly making any computations as efficient as if the data was simply stored in an array. Because all operations occurred through well-defined interfaces contained in a single module rather than sporadically in many routines, it was much easier to avoid mistakes and inconsistencies during software development. It also greatly improved the ability to attain high code-coverage by unit tests.

3.2 CUDA Bindings

CUDA bindings were generated by parsing the C header files and producing modules containing types, enumerators, functions, and subroutines. The parser was constructed using Pyparsing[12], a pure-Python library, which tokenizes strings, a lexical analysis technique where text streams are broken into meaningful pieces. Simply put, a source file is broken into groups of words, keywords, or symbols which may define declarations, functions, modules, structures, enumerators, etc. Using the tokenized results of several parsing passes, information contained in the CUDA header files provided as part of the software development kit (SDK) were parsed and converted to FORTRAN 2003 interoperable function and subroutine calls.

Appendix B describes a very simple example of using the C-interoperability wrappers discussed above. The CUDA calls are nearly identical. While this did not permit kernels to be written directly in FORTRAN , it did provide a consistent interface to the functions, regardless of the language used to call the CUDA routines. This goes a long way towards providing a well-defined interface.

4. Conclusion

A discussion of several FORTRAN advancements and design patterns useful for computational science and engineering has been presented. These advanced concepts can greatly increase productivity of programmers. Coding errors can be eliminated through the modularization and decomposition of code into testable units. User errors can be eliminated through appropriate use of abstraction, interfaces, and FORTRAN-style object-orientation.

Some recommendations for effective FORTRAN software development discussed in this paper are summarized below:

- Manipulating many related data arrays and variables is clumsy and error-prone. Performing many function calls within nested loops is inefficient. The object-oriented approach to using modules offers the best of both worlds. Management can be performed using a type and related functions to ease programmer burden and effectively self-document source code, and core algorithms may still be implemented by accessing the data arrays directly.
- Use the `public` and `private` attributes to control access to module components. Specifying `private` at the top of a module makes all module members private by default.
- Utilize language extensions and standards. Even if they are not just as fast as hard-coding a loop or an interface now, the compiler vendors are working diligently and the hard-coding may only be efficient on current generation machines. Use of standard features also reduces programmer burden for maintaining such interfaces, which across a wide-array of systems may be quite different.
- Management of interfaces through type extension and type bound procedures provides great benefit at a low overhead cost when designed well.
- Take time to model and document source code, and rewrite when things become difficult to maintain or develop.
- Meta-program. Even small sections of software which are auto-generated can greatly increase productivity. It is well worth taking the time to learn Python, Ruby, Perl, TCL, Lisp, Haskell, or one of the many other languages amenable to writing code that writes code.
- Take advantage of existing C libraries and generate wrapper interfaces using the ISO_C_BINDING module.
- Test as much software as possible using automated frameworks. This can greatly reduce the number of bugs and can help eliminate large sections of code as potential locations for infestation.

Acknowledgments

The authors would like to thank the High Performance Computing and Modernization Program Office for supporting this work through computing resources, dedicated support staff, and generous allocations for both traditional and challenge projects. The authors would also like to thank the US Air Force Research Laboratory DoD Supercomputing Resource Center for support, assistance, and guidance.

References

1. Akin, Ed, *Object-Oriented Programming via FORTRAN 90/95*, Cambridge University Press, 2003.

2. Car, David, "A reference-counting implementation in FORTRAN 95/2003", *ACM FORTRAN Forum*, 29(1), pp. 21–28, April 2010.

3. Car, David and Michael G. List, "PyF95++: A Templating Capability for the FORTRAN 95/2003 Language", *SIGPLAN FORTRAN Forum*, 29(1), pp. 2–20, April 2010.

4. Chen, Andrew and Phillip David, *FORTRAN Unit Test Framework*, http://sourceforge.net/projects/ fortranxunit, September 2010.

5. Clune, Tom and Brice Womack, *pFUnit*, http://opensource.gsfc.nasa.gov/projects/FUNIT/index.php, March 2010.

6. Cornett, Steve, *Code coverage analysis*, http://www.bullseye.com/coverage.html, September 2010..

7. Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, Reading, MA, 1995.

8. Hunt, Andrew and David Thomas, *The Pragmatic Programmer: From Journeyman to Master*, Addison-Wesley Long-man Publishing Co., Inc., Boston, MA, USA, 1999.

9. Kolawa, Adam and Dorota Huizinga, *Automated Defect Prevention: Best-Practices in Software Management*, Wiley-IEEE Computer Society Press, p. 75, 2007.

10. List, Michael G. and David Car, "A FORTRAN Unit-Testing Framework Utilizing Templating and the PyF95++ Toolset", *SIGPLAN FORTRAN Forum*, 30(1), pp. 3–15, April 2011.

11. McCormack, Drew, "Generic programming in FORTRAN with forpedo", *SIGPLAN FORTRAN Forum*, 24, pp. 18–29, August 2005.

12. McGuire, Paul, *Getting Started with Pyparsing*, O'Reilly Media, Inc., 2007.

13. NASA.rb, *NASA.rb: A home for NASA's open-source Ruby software*, http://nasarb.rubyforge.org, November 2009.

14. Reid, John, "The new features of FORTRAN 2003", *SIGPLAN FORTRAN Forum*, 26, pp. 10–33, April 2007.

15. Reid, John, "The new features of FORTRAN 2008", *SIGPLAN FORTRAN Forum*, 27, pp. 8–21, August 2008.

16. Rouson, Damian W.I., Helgi Adalsteinsson, and Jim Xia, "Design patterns for multi-physics modeling in FORTRAN 2003 and C++", *ACM Trans. Math. Softw.*, 37, pp. 3:1–3:30, January 2010.

A. Sparse-Block Matrix Class Outline

An outline of a sparse-block matrix class is included below. This class consists of a module with functions defining a type for the sparse-block matrix and a number of functions and subroutines[1].

```
! _ _TYP _ _ is a string which can be substituted through the use of
! a Python−based preprocessor. Valid arguments are integer, real (4)
! and real (8) .
template <class _ _TYP _ _>
module class_SparseBlockMatrix
# include "PyF95_macros . inc"

    use Parameters
    use class_Exception
    implicit none

!!
!! Reference counted internal type for sparse block matrix
!!
    type SparseBlockMatrix _
      integer :: refcount = 0 !< Reference count
      integer :: nmats =0 !< Total number of blocks in the sparse matrix
      integer :: nrows =0 !< Total number of rows in the sparse matrix
      integer :: blockNrows = 0 !< Number of rows in each block
      integer :: blockNcols = 0 !< Number of columns in each block
      integer :: ni=0, nj=0, nk=0 !< Not necessary but useful for lookups

      ! Index pointer to location , length nmats
      integer , dimension (:) , pointer :: idx => null ()
      ! Index pointer for begining of row , length nrows
      integer , dimension (:) , pointer :: rowidx => null ()

      ! Matrix buffer (blockNrows , blockNcols , nmats)
      ! This stores the matrix data in the same fashion that a typical
      ! Fortran 77 code would store data. The data is encapsulated in
      ! a type , however , for management purposes. Management functions
      ! can be used to improve object manipulability , but computations
      ! only use mtx and are still very efficient .
      _ _ TYP _ _, dimension (:, :, :), pointer :: mtx => null ()
    end type SparseBlockMatrix

!!
!! User-accessible external type for sparse block matrix
!!
    type SparseBlockMatrix
      type (SparseBlockMatrix_ < _ _TYP _ _>), pointer :: shared => null ()
    end type SparseBlockMatrix

    ! ... More types and interfaces omitted

contains

    !>
    !! A set of accessors are defined to return the values of each of
    !! the members of SparseBlockMatrix type .
    !<
    function getNmats( this ) result ( nmat ) as getNmats
      ! ...
      ! ...
      ! ...

    !>
    !! Several functions exist to return pointers to the sections of
    !! the data array, mtx , based on several types of inputs. These
```

```
    !! functions are wrapped in an interface as "getBlock"
    !<
    function  getBlockPtr ( this , ind )  result ( ptr )
      ! ... Takes an index in the matrix
    function  getBlockIJKPtr (this , row , i , j , k)  result (ptr)
      ! ... Takes a row number and an ijk index
    function  getBlockRowColPtr( this , row, col )  result (ptr)
      ! ... Takes a row and column number
    !>
    !!  Provides a method for multiplying a sparse matrix and a vector.
    !<
    subroutine  multiply ( this , vec , ptr )
      ! ...
    ! ... More functions and subroutines omitted
end  module class SparseBlockMatrix
end  template
```

B. Checking CUDA Devices

A simple CUDA code is given in C in Section B.1. An equivalent FORTRAN version is given in Section B.2. In-lined comments are provided to describe the components of each.

The two codes are very similar and translation from one to the other could be accomplished quite easily. The use of the ISO_C_BINDING module greatly simplified accessing the CUDA functions. This would eliminate the need for manually-creating C wrapper functions and the appropriate preprocessing macros which are typically necessary for working with GPGPU devices (or simply C) from FORTRAN.

B.1 C Version

```
/*
 * This utility was visciosly stolen from stackoverflow .
 *  It was graciously posted by aaa on 17 Feb 2010.
 *  http :// stackoverflow.com/questions/2285185/easiest -way-to-test -for -existence -of-cuda- capable -gpu-from-cmake
 */
# include  < stdlib .h>
# include  <stdio .h>
# include  <cuda .h>
# include  <cuda runtime.h>

int  main (int argc , char ** argv ) {
   int  ct = 0;
   int  dev= 0;
   cudaError t code ;
   struct cudaDeviceProp prop;

   //  Obtain the device count and check for errors
   cudaGetDeviceCount(&ct ) ;
   code = cudaGetLastError () ;
   if (code)   printf ("%s \n",  cudaGetErrorString(code));

   if ( ct == 0) {
      printf ("Cuda  device not found. \ n") ;
      exit (0) ;
   }
   printf ("Found %i  Cuda  device (s). \ n" , ct ) ;

   //  Loop through the devices and display properties
   for (dev = 0; dev < ct ; ++dev) {
      printf ("Cuda  device  %i \n", dev);

      cudaGetDeviceProperties (&prop ,dev);
```

```
    printf (“ \ tname : %s \n”, prop. name );
    printf (“ \ ttotalGlobablMem  :  %lu \n”,  (unsigned long) prop.totalGlobalMem);
    printf (“ \ tsharedMemPerBlock  :  %i \n” , prop . sharedMemPerBlock ) ;
    printf (“ \ tregsPerBlock  :  %i \n” , prop . regsPerBlock ) ;
    printf (“ \ twarpSize  :  %i \n” , prop . warpSize ) ;
    printf (“ \ tmemPitch   : %i \n” , prop . memPitch ) ;
    printf (“ \ tmaxThreadsPerBlock  :  %i \n” , prop . maxThreadsPerBlock ) ;
    printf (“ \ tmaxThreadsDim  :  %i , %i , %i \n”, prop.maxThreadsDim[0], prop.maxThreadsDim[1],    prop . maxThreadsDim [2]) ;
    printf (“ \ tmaxGridSize  :  %i , %i , %i \n” , prop . maxGridSize [0] , prop . maxGridSize [1] , prop .     maxGridSize [2]) ;
    printf (“ \ tclockRate  :  %i \n” , prop . clockRate ) ;
    printf (“ \ ttotalConstMem  :  %i \n” , prop . totalConstMem ) ;
    printf (“ \ tmajor  :  %i \n”, prop.major); printf (“ \ tminor : %i \n”, prop.minor);
    printf (“ \ ttextureAlignment  :  %i \n” , prop . textureAlignment ) ;
    printf (“ \ tdeviceOverlap  :  %i \n”, prop . deviceOverlap);
    printf (“ \ tmultiProcessorCount  :  %i \n” , prop . multiProcessorCount ) ;
  }
}
```

B.2 FORTRAN Version

The FORTRAN 2003 interfaces are included after the FORTRAN utility source.

```
! ##############################################################################
    program  check_cuda_device
! ##############################################################################
    !  The intrinsic ISO_C_BINDING module provides a number of kind parameters
    !  which are used to define in this case C−interoperable integers.
      use , intrinsic  : :  ISO_C_BINDING
    !  The CUDA module is a wrapper around the interfaces , types , and
    !  procedures that were auto−generated from cuda.h and cuda_runtime.h.
    !  This module takes the place of the C #include macros.
      use  CUDA
      implicit  none

      integer  : :  ct=0 !< CUDA device count
      integer  : :  dev =0
      integer  : :  gpuDeviceCount =0

      !  The cudaDeviceProp is an interoperable type and matches the C
      !  declaration “struct cudaDeviceProp prop;”
      type ( cudaDeviceProp )  : :  prop
      integer ( c int )  ::  code

    !  Obtain the device count from the system and check for errors .
    !  Note that the ct variable is already passed by reference (FORTRAN‘s
    !  default) and it is necessary to store the function result.
      code = cudaGetDeviceCount ( ct )
      if  (code /= cudaSuccess )  then
        ct =0
      end if

      print ∗ , “Found “, ct , “ CUDA device(s)”

    !  Loop over the devices and print their properties.  Note again
    !  that the result of the CUDA call must be stored and that the
    !  first argument is already passed by reference .
      do  dev =0, ct − 1, 1
      code = cudaGetDeviceProperties(prop,dev)
      if  ( prop%major == 9999) cycle

      print ∗ , ”Cuda device ” ,  dev
      print ∗ , ”   name : ”,  prop%name
      print ∗ , ”   totalGlobablMem : ” ,  prop%totalGlobalMem
```

```
      print * , "   sharedMemPerBlock : " ,  prop%sharedMemPerBlock
      print * , "   regsPerBlock:",  prop%regsPerBlock
      print * , "   warpSize : " ,  prop%warpSize
      print * , "   memPitch : " ,  prop%memPitch
      print * , "   maxThreadsPerBlock : " ,  prop%maxThreadsPerBlock
      print * , "   maxThreadsDim:",  prop%maxThreadsDim(1:3)
      print * , "   maxGridSize:",  prop%maxGridSize(1:3)
      print * , "   clockRate:",  prop%clockRate
      print * , "   totalConstMem : " ,  prop%totalConstMem
      print * , "   major : " ,  prop%major
      print * , "   minor : " ,  prop%minor
      print * , "   textureAlignment:",  prop%textureAlignment
      print * , "   deviceOverlap:",  prop%deviceOverlap
      print * , "   multiProcessorCount:",  prop%multiProcessorCount
      end  do

  end program  check cuda device

!>
!! The cuda runtime h module is part of an automatically − generated
!! set of modules included in a wrapper module , called CUDA, which
!! provides C interoperability with the actual CUDA libraries .
!<
!@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    module  cuda_runtime_h
!@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

      use ,  intrinsic  : :  ISO_C_BINDING

      implicit  none

      ! ... Enums and types omitted

      enum ,  bind (C)  !  ::  cudaError
        enumerator  : :  cudaSuccess=0
        ! ... More members omitted
      end  enum

      type ,  bind (C) : : cudaDeviceProp
        character (c char)  : :  name (256)
        integer ( c int )  : :  totalGlobalMem
        integer ( c int )  : :  sharedMemPerBlock
        integer ( c int )  : :  regsPerBlock
        integer ( c int )  : :  warpSize
        integer ( c int )  : :  memPitch
        integer ( c int )  : :  maxThreadsPerBlock
        integer ( c int )  : :  maxThreadsDim(3)
        integer ( c int )  : :  maxGridSize(3)
        integer ( c int )  : :  clockRate
        integer ( c int )  : :  totalConstMem
        integer ( c int )  : :  major
        integer ( c int )  : :  minor
        integer ( c int )  : :  textureAlignment
        integer ( c int )  : :  deviceOverlap
        integer ( c int )  : :  multiProcessorCount
        integer ( c int )  : :  kernelExecTimeoutEnabled
        integer ( c int )  : :  integrated
        integer ( c int )  : :  canMapHostMemory
        integer ( c int )  : :  computeMode
        integer ( c int )  : :  maxTexture1D
        integer ( c int )  : :  maxTexture2D(2)
        integer ( c int )  : :  maxTexture3D(3)
        integer ( c int )  : :  maxTexture2DArray(3)
```

```
        integer ( c int ) : : surfaceAlignment
        integer ( c int ) : : concurrentKernels
        integer ( c int ) : : ECCEnabled
        integer ( c int ) : : pciBusID
        integer ( c int ) : : pciDeviceID
        integer ( c int ) : : cudaReserved (22)
    end type cudaDeviceProp
    interface
        function cudaGetDeviceCount( count ) result ( res ) bind (C, name="cudaGetDeviceCount" )
          use , intrinsic : : ISO_C_BINDING
          import cudaSuccess
          implicit none
          integer ( c int ) : : count
          integer (KIND( cudaSuccess ) ) : : res
        end function cudaGetDeviceCount
      end interface

      interface
        function cudaGetDeviceProperties(prop ,device) result ( res ) bind (C, name=" cudaGetDeviceProperties")
          use , intrinsic : : ISO C BINDING
          import cudaSuccess
          import cudaDeviceProp
          implicit none
          type ( cudaDeviceProp ) : : prop
          integer ( c int ) , value : : device
          integer (KIND( cudaSuccess ) ) : : res
        end function cudaGetDeviceProperties
      end interface

end module cuda_runtime_h
```

HPCMP UGC 2011

7. Software Performance and Performance Modeling

MATLAB and Python for GPU Computing

Jose Unpingco and Juan Carlos Chaves
High Performance Technologies, Inc. (HPTi), Wright-Patterson AFB, OH
{junpingco, jchaves}@hpti.com

Abstract

Recent trends in hardware development have led to graphics processing units (GPUs) evolving into highly-parallel, multi-core computing platforms suitable for computational science applications. Recently, GPUs such as the NVIDIA Tesla 20-series (with up to 448 cores) have become available to the High Performance Computing Modernization Program (HPCMP) user community. Traditionally, NVIDIA GPUs are programmed using Compute Unified Device Architecture (CUDA). CUDA is a parallel programming model and software environment, developed by NVIDIA Corporation, that enables programmers to take advantage of the multi-core GPU using the C language. CUDA provides extensions to the C programming language that enable the programmer to write fine-grained parallel algorithms that can be executed using multiple, simultaneous threads on the GPU. Usually, this requires a deep understanding of the CUDA environment. Productivity is strongly influenced by the workflow of the user (e.g., time spent running vs. time spent programming). Therefore, in the Signal/Image Processing (SIP) and other related communities, most users prefer high-productivity languages such as MATLAB or Python for their scientific and technical computation needs. In this paper we study the feasibility of exploiting NVIDIA GPUs for typical SIP applications using MATLAB or Python. Programming GPUs with MATLAB or Python is a relatively recent development and only a few solutions are available. PyCUDA allows accessing the NVIDIA CUDA parallel computation API from Python. On the other hand, the MathWorks, Inc. has added MATLAB support for NVIDIA CUDA-enabled GPUs through the Parallel Computing Toolbox (PCT). We investigate the use of these technologies for typical SIP applications paying special attention to performance and productivity aspects of these approaches. Our preliminary results show that these technologies are viable and show potential to preserve the high-productivity advantages of MATLAB and Python with GPUs with relatively few, if any, performance tradeoffs.

1. Introduction

The new emphasis on high-end computing systems is rapidly evolving towards productivity and value, rather than traditional high performance computing (HPC) standards such as raw theoretical peak computing performance. Researchers' idea-to-solution or time-to-solution is becoming a more important metric than old fashioned raw computing capacity. Moreover, total end-user computing life-cycle costs and mission responsiveness are becoming increasingly critical to operational scenarios of modern Department of Defense (DoD) and homeland defense systems. HPC has recently seen a surge of interest in heterogeneous systems, with an emphasis on modern graphics processing units (GPUs). With the introduction of full double-precision support on GPUs and IEEE754-2008 compatibility, GPUs have evolved towards general-purpose graphical processing units (GPGPUs) that can be leveraged for mathematical and scientific computing. Recently HPC systems based on GPUs such as the NVIDIA Tesla 20-series (with up to 448 cores) have become available to the High Performance computing Modernization Program (HPCMP) user community. These devices offer tremendous potential for performance and efficiency in important DoD applications. However, exploiting this potential can be challenging. Traditionally, NVIDIA GPUs are programmed using the Compute Unified Device Architecture (CUDA). CUDA is a parallel programming model and software environment developed by NVIDIA Corporation that enables programmers to take advantage of the multi-core GPU using the C language. CUDA provides extensions to the C programming language that enable the programmer to write fine-grained parallel algorithms that can be executed using multiple, simultaneous threads on the GPU. Usually, this is a complex proposition that requires deep understanding of the CUDA environment. In our experience, the inherent complexity in utilizing and programming HPC systems is the main obstacle to widespread exploitation of HPC resources and technologies in the DoD. Consequently, there is the persistent need to simplify the programming interface for the generic user. This need is particularly acute in the Signal/Image Processing (SIP) and other related communities where typical users have lack a well-established codebase. Mastering the complexity of the

GPU programming environment offered by C/CUDA may be interpreted as misplaced effort that could be applied to the study of the given scientific domain. Many SIP users instead prefer high-productivity/high-level languages with integrated development environments, such as MATLAB and Python.

Programming GPUs with MATLAB or Python is a relatively recent development and only a few solutions are available. PyCUDA allows accessing the NVIDIA CUDA parallel computation API from Python. On the other hand, the MathWorks, Inc. has added MATLAB support for NVIDIA CUDA-enabled GPUs through the Parallel Computing Toolbox (PCT). In this preliminary study, we explore the feasibility of MATLAB and Python for exploiting the GPU systems recently available at the High Performance Computing Modernization Program (HPCMP). It is well known that productivity is strongly influenced by the workflow of the user (e.g., time spent running vs. time spent programming). Therefore, we want to study performance and productivity aspects of using GPUs through MATLAB and Python to understand the advantages and possible disadvantages of this approach. Moreover, we want to document these particular approaches to GPU computing (PCT, PyCUDA) presenting practical use information on typical SIP core tasks for the benefit of the HPCMP community.

The MathWorks, Inc., the developer of MATLAB, recently introduced support for programming NVIDIA GPUs on its PCT corresponding to the MATLAB R2010b release. The PCT only supports NVIDIA hardware meeting the CUDA 1.3 or greater hardware specifications. The main reason for this requirement is that the CUDA release 1.3 or greater supports floating-point doubles (the base data type in MATLAB) and is IEEE compliant. On the other hand PyCUDA, an open-source package, supports the whole universe of CUDA-enabled GPU hardware including the 1.0 release. In Section 2, we present a brief overview of the HPCMP hardware and software environment where we performed our study, including the recently announced Condor cluster at the US Air Force Research Laboratory (AFRL). In Section 3, we present an overview of the MATLAB PCT and its GPU support. In addition, we provide specific information to enable MATLAB-based GPU applications on HPCMP HPC platforms, paying special attention to the performance and productivity aspects of this new approach. In Section 4, we explored the PyCUDA approach to GPU computing, including details of PyCUDA-based GPU computation, and presenting performance and productivity results as well. In Section 5, we present analysis and discussion of our experiments, with special emphasis in understanding the impact of using high-productivity/high-level languages for GPU computing for productivity, portability and performance. Finally, we present our conclusions.

2. Hardware and Software Environment

As hardware platform for this study, we used the Condor Cluster at the US Air Force Research Laboratory, Information Directorate (AFRL.RI) at Rome, New York. The Condor Cluster is made of 78 custom-built servers (2U Dual six-core Intel Xeon 5,650 CPUs) with 94 NVIDIA GPU Tesla C2050s (dual cards per node) and 62 NVIDIA GPU Tesla C1060s (dual cards per node). The Condor Cluster also contains 1,716 cell BE(ps3) systems, although we do not use this part of the system for our studies. In addition, and mostly for our PyCUDA experiments, we used the smaller GPU systems available at the Arctic Region Supercomputing Center, C2050, nemo, and sectest which contain Tesla 2050 GPUs, as well as older NVIDIA GeForce GPUs. At the time of our study, the C2050 GPU was the most current version of the NVIDIA GPU architecture. The key specifications for the GPU used in our study are shown in Table 1.

Table 1. Key system characteristics for the GPUs used for the study

Name	Tesla C2050	Tesla C1060	GeForce 9800M GTS	GeForce 9800 GX2
Compute Capability	2.0	1.3	1.1	1.1
Multiprocessor Count	14	30	8	16
Total Memory	3GB	4GB	0.5GB	0.5GB
Core Frequency	1.15GHz	1.3GHz	1.5GHz	1.5GHz
GFLOPs (single)	1030	933	288	768
ECC Memory	Yes	No	No	No
IEEE Double	Yes	Yes	No	No

Regarding the software environment, to conduct the MATLAB study we used release R2010b and the Parallel Computing Toolbox (PCT) 5.0 running on top of 64-bit Linux. On the other hand, for the Python study we used Python 2.6 and PyCUDA 0.94.

3. MATLAB for GPU Computing

The PCT is the official parallel toolbox for MATLAB. This toolbox consists of the standard interactive MATLAB front-end and a backend computing resource that is divided into "*workers*." It supports interactive development with up to eight workers as well as batch mode, which can be configured to use a scheduling batch system such as PBS. The PCT supports both task parallelism and data parallelism. In addition to task parallelism and data parallelism, the toolbox also implements commands similar to those used for MPI. Recently as of MATLAB-released R2010b, the PCT 5.0 also supports GPU computation. As we mentioned, the PCT only supports NVIDIA hardware meeting CUDA 1.3 or greater hardware specifications. The PCT enables users to write and run code on the GPU in the MATLAB native M-language. In the simplest mode of operation, MATLAB commands are transformed into GPU functions by casting input data to PCT's GPU data structure. The interpretive nature of the MATLAB language is maintained by providing real-time, transparent access to the GPU compiler. The GPU functions and operations implemented within the PCT are transparent to the user: functions and operations calls are very similar to the CPU MATLAB implementation. The advantage of this approach allows a high-level language such as MATLAB to utilize the GPU without writing C/CUDA code. Productivity is maintained, as MATLAB users do not need to perform extensive modifications to run MATLAB code with the PCT. The PCT allows defining GPU variables in the usual MATLAB workspace, and when operations or functions are performed on these variables the execution is automatically performed in the GPU instead of the CPU. Moreover, the casting of variables makes the transfer of data from the CPU to the GPU transparent to the user.

3.1 The *gpuArray()* Command

The PCT offers several options to enable GPU computing on MATLAB programs: the first one, already mentioned, is the transferring of data between the MATLAB workspace and the GPU by simply casting MATLAB workspace arrays using the *gpuArray()* command. Figure 1 presents a code snippet that illustrates a GPU computation using typical SIP core algorithms.

```
1  % A is an array in the MATLAB workspace
2  A = someArray(1000, 1000);
3  % Push to GPU memory
4  G = gpuArray(A);
5  % Computation at the GPU using overloaded functions
6  F = fft(G);
7  x = G\b;
8  % Bringing back into the MATLAB workspace
9  z = gather(x);
```

Figure 1. The PCT *gpuArray()* command

In line #2 of the code snippet in Figure 1, a regular MATLAB array ***A*** is created in the MATLAB workspace. By a simple casting operation though the *gpuArray()* command, the array is transferred to the GPU where it exists as ***G***. Subsequently, an operation performed on G such as the fast Fourier transform (FFT) will be performed on the GPU instead of the CPU by the overloaded *fft()* function. Currently, there are over 100 GPU overloaded MATLAB built-in functions with support for integer and double variables. Many important functions for SIP such as *fft(), fft2(), ifft(), ifft2()*, matrix multiplication (*A*B*), matrix left division (*A\b*) and LU factorization are supported. In addition, there is also support to bring arrays back from the GPU to the regular MATLAB workspace for further processing (on the CPU), as illustrated in line #9 of the code snippet.

There are also a number of static methods on the *GPUArray* MATLAB class that allow directly constructing arrays on the GPU without having to transfer them from the MATLAB workspace. These constructors require only array size and data class information, so they can construct an array without any element data from the workspace, if necessary. A complete list of available static methods can be displayed by invoking the command: *methods('parallel.gpu.GPUArray')*.

3.2 Running MATLAB User's Code on the GPU

The PCT also offers the option of invoking element-wise MATLAB functions on the GPU, including code written by the users. The PCT allows executing user-created MATLAB function on the GPU by calling *arrayfun()* with a function handle to the MATLAB function as the first input argument:

> result = arrayfun(@myFunction, arg1, arg2);

Additional arguments provide inputs to the MATLAB function. If any of the input arguments is of the class *GPUArray*, the function executes on the GPU and returns a *GPUArray*. On the other hand, if none of the inputs is of the class *GPUArray*, then *arrayfun()* executes in the CPU. Figure 2 presents a code snippet of a GPU computation that takes advantage of this PCT feature.

```
% Parallel GPU: using FFT on the GPUs for SIP
% Computing the Total Energy and the Average Power of a set of 100 signals
% Simulating the signals with data sampled at 1000 Hz.
...
% Transfering time vector to the GPU
timeVec    = gpuArray( (0:numSamples-1) * sampleTime );
...

% Taking advantage of the 2 GPU available to compute in parallel
parfor i = 1:100
% Creating a set of signals to process
freq1(i)      = 2 * pi * i*10;
freq2(i)      = 2 * pi * (i/2)*10;
signal     = sin( freq1(i) .* timeVec ) + sin( freq2(i) .* timeVec );
% Adding random Noise to the Signal
signal  = signal + 2 * i*randn( size( timeVec ) );

% Computing the Total Energy and the Average Power of the Signal
% in the frequency domain on the GPUs
```

Figure 2. The *arrayfun()* command

3.3 Parallel GPU: MATLAB Programs on Multiple GPU Cards

As mentioned before, the PCT supports task and data parallelism. The *matlabpool* command opens or closes a pool of MATLAB sessions (called workers) for parallel computation. These workers are essentially headless MATLAB sessions with most of the functionality of a regular MATLAB client except the graphic capabilities. Up to 8 workers can be opened on a local multicore node and more using several nodes if the MATLAB Distributed Computing Server (MDCS) is available. For systems that have multiple GPU cards the *matlabpool* command allows leveraging several GPUs, each associated with a MATLAB worker, for parallel GPU programs.

The easiest way of enabling task parallelism on several GPUs is through the *parfor* loop command. The PCT executes a series of statements (the loop body) over a range of values. Part of the *parfor* body is executed on the MATLAB client GPU (where the *parfor* is issued) and part is executed in parallel on MATLAB workers' GPUs. Because several MATLAB workers can be computing concurrently on the same loop, a *parfor* loop can provide significantly better performance than a regular *for* loop on a single GPU. Figure 3 presents a code snippet of a GPU computation that takes advantage of the PCT *parfor* loop to exploit multiple GPUs for a typical SIP core task.

```
% Parallel GPU: using FFT on the GPUs for SIP
% Computing the Total Energy and the Average Power of a set of 100 signals
% Simulating the signals with data sampled at 1000 Hz.
...
% Transfering time vector to the GPU
timeVec    = gpuArray( (0:numSamples-1) * sampleTime );
...

% Taking advantage of the 2 GPU available to compute in parallel
parfor i = 1:100
% Creating a set of signals to process
freq1(i)      = 2 * pi * i*10;
freq2(i)      = 2 * pi * (i/2)*10;
signal     = sin( freq1(i) .* timeVec ) + sin( freq2(i) .* timeVec );
% Adding random Noise to the Signal
signal  = signal + 2 * i*randn( size( timeVec ) );

% Computing the Total Energy and the Average Power of the Signal
% in the frequency domain on the GPUs
transformedSignal = fft( signal );

% Due to limited support for complex arithmetic for GPUArray objects, we
% use |real| and |imag| to perform the computation
realPart      = real( transformedSignal );
imagPart      = imag( transformedSignal );
% Magnitude of transformedSignal
X = sqrt(realPart .* realPart + imagPart .* imagPart);
%Total Energy for set of signals:
E = E + sum((X.^2) / (numSamples^2) * (sampleTime^2) * df);
% Average Power for set of signals:
AP = AP + sum((X.^2)/(numSamples^2));
end
```

Figure 3. The *parfor* loop command

The command *parfor* in line #10 will execute in parallel on the GPUs if the *matlabpool* command has been previously executed as shown in this example:

> matlabpool open local 2

This assumes that the system has 2 available local GPUs for computation like in the Condor Cluster. In modern multi-core heterogeneous HPC systems, many configurations of CPU and GPUs are possible. As mentioned, the PCT supports exploitation of local and remote GPU cards. The main limitation is that each MATLAB worker can, at most, use one GPU at a time. To run in an HPC Cluster with GPUs on its nodes required not only the PCT but the MDCS. Depending on the configuration of the HPC Cluster and the needs of the users, 3 cases may be possible:

- Less likely: each node has only a single GPU and is made of a single-core processor. In this configuration, there will be no issues.
- Likely: each node has a multi-core processor and a single GPU per node. In this configuration, two uses are possible: to run only one MATLAB worker per each node at a time, or to allow multiple workers to run on the same node at the same time and share the GPU. However, then the GPU computation will be serialized across workers.
- Most likely: each node has a multi-core processor and has multiple GPUs per node. In this situation, it is necessary to manually assign GPUs to workers on each node to ensure that each worker gets an independent GPU card.

3.4 MATLAB GPU Computing: Performance Aspects

We ran computation typical of signal and image processing to investigate the productivity and performance aspects of the MATLAB approach to GPU computing. First, we benchmarked the AFRL Condor and the ARSC C2050 systems by solving a linear system (solve for *x* in *A*x=b*). We ran the standard benchmark provided by the MathWorks, Inc. hundreds of times to ensure statistical significance of the results. In addition in all our benchmarks we included the transfer time to the GPU. The MATLAB algorithm executed on the GPU and CPU is very simple as shown in the code snippet in Figure 4.

```
function [A, b] = getData(n, clz)
    fprintf('Creating a matrix of size %d-by-%d.\n', n, n);
    A = rand(n, n, clz) + 100*eye(n, n, clz);
    b = rand(n, 1, clz);
end
function time = timeSolve(A, b)
    tic;
    x = A\b;
    time = toc;
end
```

Figure 4. The *backslash* operator benchmark

The number of floating-point operations per second is used as a measure of performance. It allows comparing the performance of the algorithm for different matrix sizes. Floating-point operations are counted as in the HPC Challenge benchmark. For an n-by-n matrix, the number of floating point operations is given by *2/3*n^3 + 3/2*n^2*. Memory limitations on the GPUs allowed a maximum matrix size of 8,192×8,192 in double precision (512 megabytes). Ten matrices from 32×32 to 8,192×8,192 were used in single- and double-precision to complete the benchmark. In Figure 5, we present typical results for the Tesla C1060 and Tesla C2050 nodes at the AFRL Condor Cluster. The results are very similar for the the Tesla C2050 based machine at ARSC:

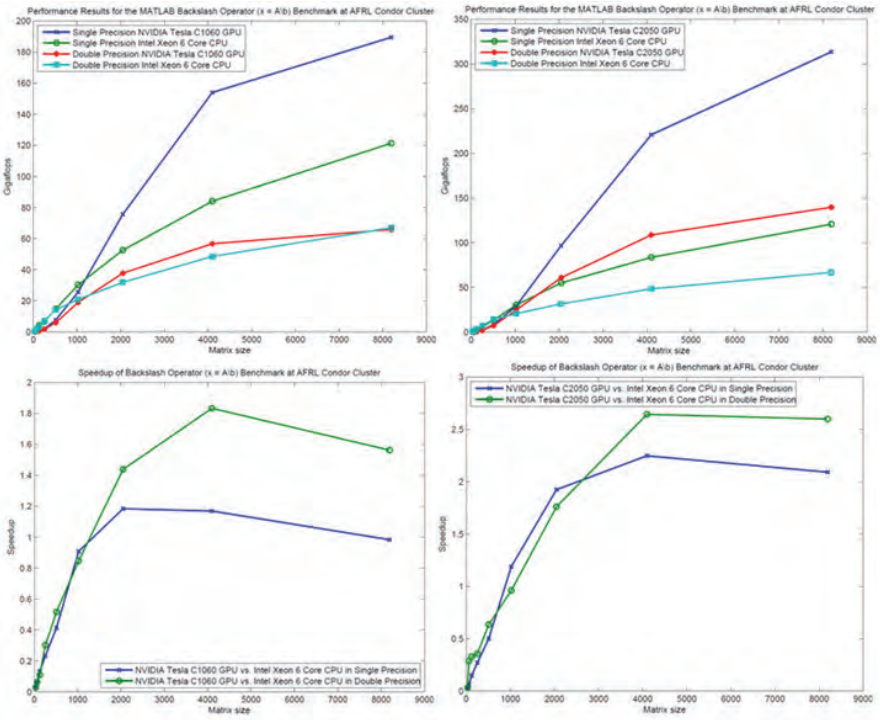


Figure 5. The *backslash* operator benchmark results for the Condor C2050 GPU nodes

We notice that for large matrices the performance of the GPUs (for the C2050 cards) is more than twice the performance of the 6-core Intel Xeon processor, despite of the fact that the MATLAB backslash operator is multithreaded on the CPU and we have included the GPU to CPU transfer times our benchmarks.

In addition we ran typical SIP tasks used for spectral analysis (as shown in Figure 3) hundreds of times to obtain statistical significant performance results for the case when 2 GPU cards are exploited to enable task parallelism. The results are displayed in Table 2.

Table 2. FFT based benchmark at AFRL Condor

MATLAB PCT FFT-based benchmark (double precision)	Wall-clock average times (seconds)
1-Tesla C1060 GPU	44.50
2-Tesla C1060 GPUs	25.30
1-Tesla C2050 GPU	35.10
2-Tesla C2050 GPUs	21.50
1-6 core Intel Xeon CPU (running multithreaded FFT)	117.20

Exploiting two C2050 GPU cards provides a significant reduction in the wall-clock time when compared to the multi-core Intel processor even though again the MATLAB FFT is multithreaded. The reduction is about 82%. These results are consistent with the *backslash* operator benchmark results.

4. Python for GPU Computing

Python is a high-level language that has gained an enormous following in the last decade due to its simple syntax and powerful design that allows complex codes to be written in relatively few lines. As an interpreted language, there is no separate compile-and-link process, since the Python interpreter itself actively builds the byte-code that is executed on a particular hardware platform. Consequently, Python is a portable language because it is the interpreter that mediates between the code and the hardware, which means that Python code runs wherever it can find an installed interpreter (e.g., Python runs on iPhones, desktops, HPCs, embedded systems). Python is noted as a “glue” language because it can call

library functions in languages such as C and FORTRAN. This has been a particular boon for the scientific community because it allows for migrating well-established scientific codes into Python. In the last five years, there has been a concerted effort in the scientific Python community to consolidate the best-of-breed numerical libraries in a number of packages such as scipy and SAGE. The end result is that now Python and its scientific modules constitute an open-source scientific software development platform on par with MATLAB. For example, Figure 6 shows the Python code to compute the FFT of data stored in a file.

```
from scipy import fft, loadtxt
data = loadtxt('datafile.txt') # open and read data
y = fft(data,1024)             # compute 1024-point FFT
```

Figure 6. FFT on data stored in a file in Python

Thus, Python is not hard to learn, has a wide-ranging and ever-growing set of scientific modules, and is supported by a vibrant user and developer community. This brings us to the combination we investigate in this paper, CUDA and Python: PyCUDA.

4.1 PyCUDA for Scientific Computing on GPUs

CUDA and Python are complementary because, as an interpreted language, Python itself (as opposed to the libraries Python may call), is slower than a compiled language, whereas CUDA is a fast, compiled language. This sets up a productive workflow wherein the slower staging of computations, say by utilizing the existing scipy C/FORTRAN library calls, is complemented with the faster execution of GPU-primed computations for specific algorithms. This strategy uses the strength of both coding techniques for a smooth PyCUDA optimized workflow as shown in the Figure 7.

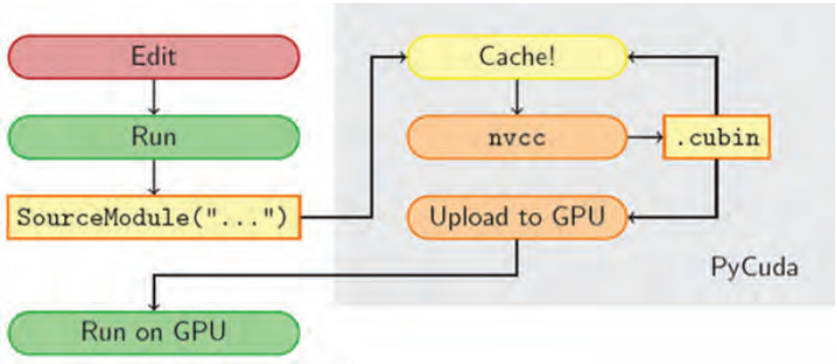


Figure 7. PyCUDA optimized workflow

The PyCUDA workflow begins by editing and running a Python text script. Embedded in the Python code are calls to the PyCUDA module that contain a mixture of high-level PyCUDA code and potentially low-level CUDA code fragments. Python wraps the embedded CUDA code with extra CUDA boilerplate to create a complete CUDA source file that is then compiled using the nvcc compiler, just as for the basic CUDA workflow. The resulting code is then injected into a PyCUDA cache, uploaded to the GPU, and then ultimately run with results returning to the Python interpreter. Note that Python handles all of the above steps without user intervention. The PyCUDA cache maintains a history of pre-compiled CUDA code, so that as long as the Python script does not alter functions that are ultimately injected into the GPU, then the nvcc compile step is not re-run. This is powerful because it means that data can be manipulated using Python code and operated upon by cached CUDA code.

When embedded CUDA code is created, PyCUDA generates a wrapper that manages CUDA tasks such as memory allocation and garbage collection. Thus, the nvcc compiler actually builds from a much larger source code file than is shown in the Python source fragment, but this entire process is hidden from the user. Note that PyCUDA provides incrementally more and deeper access to the full CUDA API.

In a heterogeneous computing environment with multiple potentially dispersed compute nodes with multiple GPUs, PyCUDA can be used in concert with the built-in Python multiprocessing module (or, other third-party modules such as IPython) to disperse and manage remote CPU/GPU computations. It is important to emphasize that this not built into PyCUDA, but is a separate technique.

4.2 PyCUDA *gpuarray* Programming

As opposed to writing embedded C/CUDA, the easiest way to use PyCUDA is to use the exposed *gpuarray* convenience functions directly. For example, Figure 8 shows a complete PyCUDA program to square an array of numbers.

```
import pycuda.autoinit        # sets up interface to GPU
import numpy                  # numerical Python arrays
from pycuda import gpuarray   # gpuarray is the GPU-based equivalent of numpy arrays
# create array of 0's on GPU, float32 is the type supported by the GPU
a=gpuarray.zeros((10,10),dtype=numpy.float32)
a = a*a                       # this happens on the GPU
# pulls data from GPU for further processing as a numpy array
values = a.get()
```

Figure 8. PyCUDA program to square an array

Although this computation is simple, this shows how a short PyCUDA program can accomplish many tedious tasks. For example, *gpuarray.zeros* allocates and fills GPU memory in one line (c.f. the above CUDA code sample). The following lines square the variable a on the GPU, but we are not required to manage this computation explicitly by specifying its memory access, or bother with garbage cleanup, all of which is handled automatically by PyCUDA. Figure 9 shows a more interesting example, where a matrix of random numbers is created on the CPU, transferred to the GPU, and the squared on the GPU.

```
x = numpy.random.rand(10,30).astype(numpy.float32) # creates random CPU matrix
a=gpuarray.to_gpu(x)                               # transfer numpy array to GPU
b=a**2                                             # compute square
```

Figure 9. A more interesting PyCUDA program

The *gpuarray.to_gpu* call transfers data from the CPU to GPU. Thus, the usual basic arithmetic operations are supported by PyCUDA, as we have shown. Additionally, PyCUDA contains a *cumath* math library with the following extended mathematical functions named after their C-language analogues:

pycuda.cumath.tan	*pycuda.cumath.tan*
pycuda.cumath.sin	*pycuda.cumath.asin*
pycuda.cumath.cos	*pycuda.cumath.floor*
pycuda.cumath.acos	*pycuda.cumath.ldexp*
pycuda.cumath.exp	*pycuda.cumath.atan*
pycuda.curandom.rand	*pycuda.cumath.log*
pycuda.cumath.sinh	*pycuda.cumath.fmod*
pycuda.cumath.log10	*pycuda.cumath.cosh*
pycuda.cumath.frexp	*pycuda.cumath.sqrt*
pycuda.cumath.tanh	*pycuda.cumath.modf*
pycuda.cumath.fabs	*pycuda.cumath.ceil*

This summarizes the easiest workflow for PyCUDA based upon using the exposed functions and the *gpuarray* structures to build scientific computations. For a large data-parallel application, just using these features is enough to quickly obtain a 5x to 10x speedup. Thus, with relatively little investment (e.g., inexpensive graphics card, open-source PyCUDA, easy-to-use *gpuarray* interface), one can determine whether or not subsequent investment in GPU algorithm development is warranted. If so, then PyCUDA provides incrementally more fine-grained algorithm control via the *ElementWiseKernel*, *ReductionKernel*, and *SourceModule* features in which more and more C/CUDA source code can be included in the Python script.

4.3 PyCUDA GPU Computing: Performance Aspects

To investigate the performance aspects of the PyCUDA approach to GPU computing we benchmarked the evaluation of a 10th-order polynomial *(1+x**2+..+x**10)* on the ARSC systems: C2050, nemo (GeForce 9800M GTS) and sectest (GeForce 9800 GX2) described at Table 1. We could not run PyCUDA at the AFRL Condor Cluster due to compatibility issues with the gcc library versions currently at that machine. Figure 10 shows the results for this benchmark. The graph on the left is for the simple float32 add operation, and the graph on the right shows the 10th-order polynomial valuation. Note that each point on each graph corresponds to an average over 1,000 trials used to obtain statistically meaningful results.

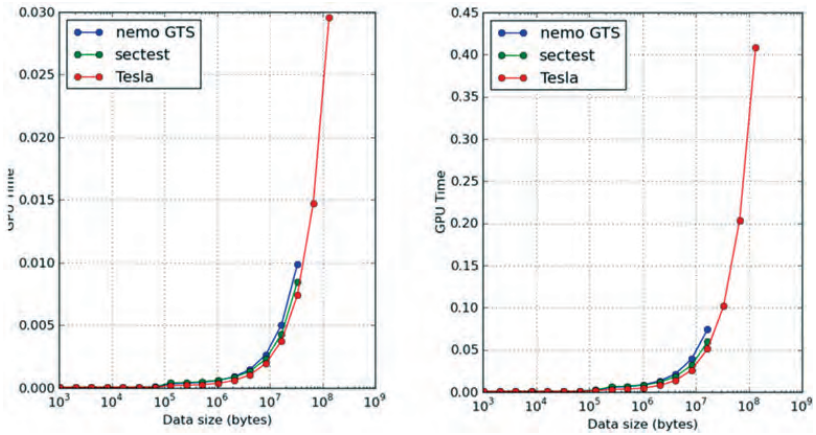


Figure 10. GPU wall-clock times (milliseconds): float32 add operation (left) and 10th-order polynomial valuation (right)

Figure 11 shows the respective benefit of using the GPU as opposed to the CPU for the computation. The polynomial evaluation indicates about 10-times the gain as compared to the simple float32 add.

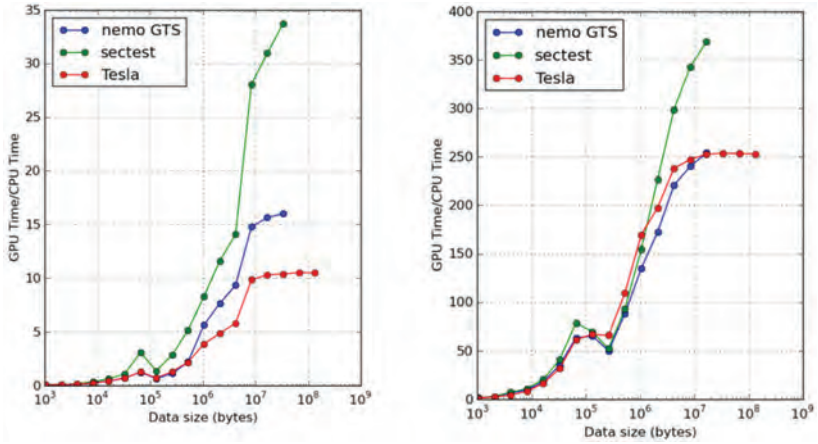


Figure 11. GPU vs. CPU wall-clock times: float32 add operation (left) and 10th-order polynomial valuation (right)

5. Discussion: High-Productivity Languages for GPU Computing

Both MATLAB and PyCUDA provide relief from having to learn the full C/CUDA API in order to use GPUs effectively for scientific computing. On the one hand, since PyCUDA and Python are freely available open-source projects and the cost of starter GPUs is relatively inexpensive, they provide a cost-effective entry point into GPU computing for those who want to investigate whether or not the data-parallel format of GPU computing is usable for their application. Further, unlike MATLAB, PyCUDA supports both single- and double-precision computing on a wider variety of platforms. On the other hand, PyCUDA does not automatically manage distributing CPU/GPU computations across multiple heterogeneous clusters with multiple GPUs, as this requires a separate technique outside the scope of PyCUDA.

Ultimately, both MATLAB and PyCUDA alleviate the effort involved in getting started with GPU computing for scientific applications. As high-level languages, both provide interactive access to the GPU and corresponding computing so that individual data can be pushed and pulled from the GPU and then manipulated further on the CPU, if need be. Once the setup is complete, neither provides substantial or prohibitive run-time overhead for GPU computing. Finally, some tightly-coupled CPU-bound applications, simply do not fit into the data parallel paradigm of GPU computing, regardless of the workflow.

Talented computer scientists continue to extend traditional scientific libraries such as BLAS to GPUs and this trend is accelerating. Though not suitable for all scientific work, low-cost GPU computing is here to stay and is further expedited using high-level languages such as MATLAB and Python.

6. Conclusion

GPU technology is evolving into highly parallel, multicore computing devices suitable for computational science applications. Recently GPUs such as the NVIDIA Tesla 20-series have become available to the HPCMP user community in the form of new high-end Linux Clusters. High-level/High-Productivity Languages such as MATLAB and Python offer easy access to GPUs while preserving user productivity and allowing a relative transparent exploitation of these new systems. Our preliminary results show that the PCT and PyCUDA are viable technologies and show potential to preserve the high-productivity advantages of MATLAB and Python with GPUs with relatively few, if any, performance tradeoffs. This may ease and even encourage the migration of non-traditional CTA communities such as SIP to these new high-end platforms fostering the exploitation of HPCMP HPC resources for the benefit of the Warfighter.

Acknowledgments

The authors would like to thank and acknowledge Dr. Greg Newby (ARSC) for his extraordinary support. In addition, we want to acknowledge the helpdesk staff at AFRL Information Directorate for their timely guidance with the new GPU system. This publication was made possible through support provided by the DoD HPCMP PETTT program through High Performance Technology, Inc (HPTi) under contract. The opinions expressed herein are those of the authors and do not necessarily reflect the views of the DoD or HPTi.

References

Parallel Computing Toolbox User's Guide (Release 2010b), http://www.mathworks.com/help/pdf_doc/distcomp/distcomp.pdf, retrieved December 17, 2010.

PyCUDA's Documentation (Release 0.94), http://documen.tician.de/pycuda/, retrieved December 10, 2010.

PyCUDA: Even Simpler GPU Programming with Python, Nvidia GPU Technology Conference, San Jose, CA, October 2010.

Half-day Tutorial on GPU Computing using PyOpenCL, Conference on Scientific Computing in Python (SciPy 2010), Austin, TX, June 2010.

Programming and Benchmarking Domain Decomposition Codes on the Cray XE6

Robert Rosenberg, Stephen Bique, and Kris Andersen
US Naval Research Laboratory (NRL-DC), Washington, DC
{robert.rosenberg, steven.bique, kris.andersen.ctr}@nrl.navy.mil

Matt Koop and Chris Kung
High Performance Technologies, Inc., Reston, VA
{mkoop, ckung}@hpti.com

Abstract

The High Performance Computing Modernization Program (HPCMP) Technology Insertion-10 (TI-10) acquisition saw the procurement of three Cray XE6 systems. We perform a benchmark comparison of these new systems against the older TI-09 SGI Altix ICE/Nehalem systems using four three-dimensional (3D) codes. These four codes, employing domain decomposition on uniform grids, demonstrate good parallelization and scaling. Along the way we discover some recommended programming practices for this type of Message Passing Interface (MPI) parallelization. The resulting benchmark runs indicate that the latest Cray systems compare well with the SGIs on a node-by-node basis.

1. Introduction

In January of 2011, the procurement assets of the High Performance Computing Modernization Program (HPCMP) Technology Insertion-10 (TI-10) acquisition began to appear on the floors of their respective Centers: Chugach at Arctic Region Supercomputing Center (ARSC), Garnet at the US Army Engineer Research and Development Center (ERDC) and the largest system, Raptor at US Air Force Research Laboratory (AFRL). They are all Cray XE6 systems with AMD Opteron/Magny-Cour processors. In a similar manner, the TI-09 acquisition procured all Intel/Nehalem systems (two SGI and one DELL): Mana at Maui High Performance Computing Center (MHPCC), Harold at the US Army Research Laboratory (ARL), and the largest system, Diamond, at ERDC. Table 1 displays the characteristics of these platforms.

Table 1A. Platform characteristics

CPU	Vendor	Name	Type	Number of Nodes	Number of CPUs/Node	GHz	Number of Cores/Node	Total Number of Cores
Itaniums	sgi	hawk	SGI Altix 4700	18×128	2	1.6	dual	9216
Xeons	lnx	mjm	LNX ATC	1100	2	3	dual	4400
	dell	mana	Dell PowerEdge	1152	2	2.8	quad	9216
	sgi	harold	SGI Altix ICE 8200	1344	2	2.8	quad	10752
	sgi	diamond	SGI Altix ICE	1920	2	2.8	quad	15360
Opterons	cray	sapphire	CRI XT3	4096	1	2.6	dual	8192
	cray	jade	CRI XT4	2146	1	2.1	quad	8584
	cray	einstein	CRI XT5	1592	2	2.3	quad	12736
	cray	chugach	CRI XE6	727	2	2.4	octa	11648
	cray	garnet	CRI XE6	1264	2	2.4	octa	20224
	cray	raptor	CRI XE6	2732	2	2.4	octa	43712
PowerPCs	ibm	davinci	IBM HydroCluster	152	16	4.7	dual	4864

Table 1B. Platform characteristics

CPU	Vendor	Name	Interconnect	Topology	Memory/Node	Memory Type
Itaniums	sgi	hawk	Numa Link 4	Fat Tree	16	DDR
Xeons	lnx	mjm	Infiniband	Fat Tree	8	DDR2
	dell	mana	Infiniband DDR1	Fat Tree	24	3xDDR3
	sgi	harold	infiniband DDR4	Hypercube	24	3xDDR3
	sgi	diamond	infiniband DDR4	Hypercube	24	3xDDR3
Opterons	cray	sapphire	SeaStar	3D torus	4	DDR
	cray	jade	SeaStar2	3D torus	8	DDR2
	cray	einstein	SeaStar2+	3D torus	16	DDR2
	cray	chugach	Gemini	3D torus	32	4xDDR3
PowerPCs	ibm	davinci	HPS (FED)	omega-network	64	

The most obvious difference between the two systems is that the Xeon systems are quad-core, while the latest Opteron systems are octa-core. The table shows that the Opterons have a slower clock than the Xeons, but the former have 4 channels to 32GBs of memory per node, while the latter have only 3 channels to 24GBs of memory per node. Given these characteristics we would expect that benchmarks on the two systems would be comparable on a core-by-core basis.

2. Simple Benchmarks

The HPCMP Benchmarks[1], before the extant of the Cray XE6s, showed that the Intel/Nehelam systems, overall, performed best across 13 Department of Defense (DoD) HPCMP applications (standard and large runs). In a presentation at last year's Users Group Conference[2], we built a suite of simple benchmarks, ranging from serial programs like the Linpack benchmark to a large parallel matrix-matrix multiply running on 16 cores, to a two-dimensional (2D) user's code running on 64 cores. The results were in agreement with the HPCMP Benchmark results. The superior performance of the Intel/Nehalem was attributed to the fact that they integrated a memory controller onto the chip, resulting in excellent memory bandwidth[3]. When the Cray XE6s were first made available to users, we were eager to include their performance in our simple benchmark results.

Figure 1 shows the simple benchmark results from last year with the Garnet XE6 results added in next to Einstein, a Cray XT5. The plot incorporates 14 DoD Supercomputing Resource Center (DSRC) platforms, some of which have been retired, and 4 Affiliated Resource Centers (ARCs) platforms from the US Naval Research Laboratory-Washington, DC (NRL-DC). The vertical axis indicates the 10 simple benchmarks. Performance on all benchmarks was scaled between 0 and 1, for each benchmark.

We see that the XE6 runs are only marginally faster than those of the XT5, and significantly slower than the Intel/Nehalem (mana, Harold, and diamond). This is the first time that the latest procured systems actually ran slower than the previous year's acquisition.

The procurement process is not solely based on performance, but on the performance/price ratio too. A Google search on the costs of these two processors[4,5] indicated a significant price difference, with the Xeon almost twice as expensive as the Opteron, commensurate with the performance difference.

We concluded that users would have to run on more cores to maintain performance. This could be a performance issue for codes that do not scale well. We decided to develop some more realistic benchmarks to explore scaling on the XE6 platforms.

3. Domain Decomposition

Domain decomposition is merely a spatial subdivision of the computational domain across processors. The scalability of such codes for large-scale computation has been demonstrated.[6] In the four codes under consideration in this paper, all the computational grids are uniform, so the partitioning is straightforward.

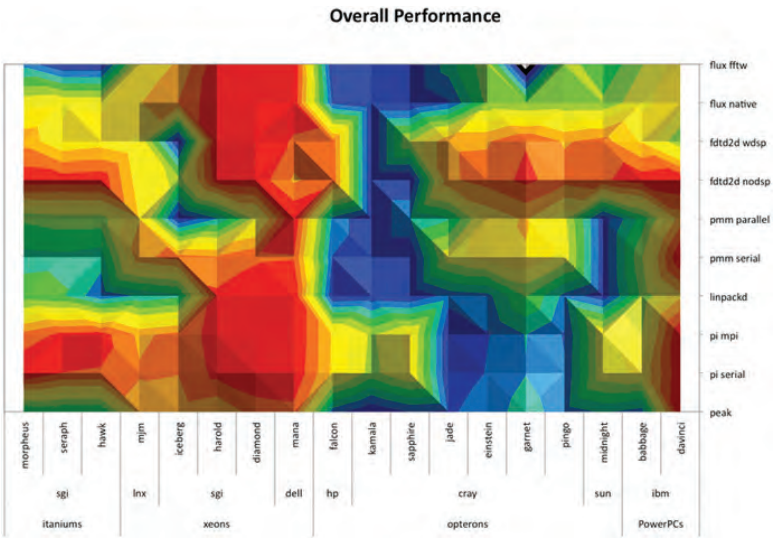


Figure 1. Overall Performance for the Simple Benchmarks. Performances for each benchmark were scaled from 0 to 1. Garnet, the Cray XE6 appears under the Opterons/Cray next to Einstein. Garnet's performance is only marginally better than Einstein's.

3.1 Partitioning

Let **npr** be the total number of processors and **ipr** the process rank. In Figure 2, we show the Message Passing Interface (MPI) calls and FORTRAN source code to accomplish this decomposition.

```
call mpi_dims_create(npr,ndim,dims,ierr)
ngr(1)=dims(3)
ngr(2)=dims(2)
ngr(3)=dims(1)

lperiod=.false.
call mpi_cart_create(mpi_comm_world,ndim,ngr,lperiod,.false.,comm3d,ierr)
call mpi_cart_coords(comm3d,ipr,ndim,igr,ierr)

do idim=1,ndim
    call mpi_cart_shift(comm3d,idim-1,1,iprv(idim),inxt(idim),ierr)
end do

nc =(/nx,ny,nz/)
nlo=float(  igr)*nc/ngr+1
nhi=float(1+igr)*nc/ngr
nn =nhi-nlo+1
```

Figure 2. FORTRAN code fragment displaying the MPI calls for domain decomposition on a uniform grid

In the figure, **ndim** is the number of dimensions, 3, and **dims** is the number of processors along each dimension. MPI_DIMS_CREATE assumes that the call is from a 'C' program and so the order of **dims** must be reversed when invoked from 'FORTRAN.'

In the call to MPI_CART_CREATE, the '.false.' value specifies that the task ranks not be reordered for optimizing communication. It has been our experience that when '.true.' is specified, only the IBM systems actually reorder the ranks. But when they do so, the call to MPI_CART_COORDS has a bug in it, such that the original topology is used, resulting in incorrect coordinate assignments. The work around is to use the MPI_CART_GET call.

iprv and **inxt** are length 3 arrays which indicate the nearest-neighbor processor ranks in each direction (previous and next for x, y, and z), and are necessary when we perform the border exchange of ghost cells.

3.2 MPI Derived Data-types

There are many ways to do the exchange of ghost cells, i.e., "Border Exchange." Previously we have coded the border exchange using MPI derived data-types.[7] The motivation was to simplify the code and take advantage of as many MPI features as we could. In Figure 3, we show the FORTRAN code for both the Subarray data-type and the Vector data-type.

```
      nnx=nprnx+2*ng
      nnx=nprnx+2*ng
      nny=nprny+2*ng
      nnz=nprnz+2*ng

      if (npr.eq.0) print*,'%grid, using subarray derived data type'

      sizes    =(/nnx,nny,nnz,5/)
      subs(:,1)=(/ ng,nny,nnz,1/)
      subs(:,2)=(/nnx, ng,nnz,1/)
      subs(:,3)=(/nnx,nny, ng,1/)
      starts   =0

      call
mpi_type_create_subarray(3,sizes,subs(1,1),starts,MPI_ORDER_FORTRAN,DTflt,DTghostx,ierr)
      call
mpi_type_create_subarray(3,sizes,subs(1,2),starts,MPI_ORDER_FORTRAN,DTflt,DTghosty,ierr)
      call
mpi_type_create_subarray(3,sizes,subs(1,3),starts,MPI_ORDER_FORTRAN,DTflt,DTghostz,ierr)

      else

      if (npr.eq.0) print*,'%grid, using vector derived data type'

      istride=(nnx)
      jstride=(nnx)*(nny)

      call mpi_type_vector((nnz)*(nny),            ng,istride,DTflt,DTghostx,ierr)
      call mpi_type_vector((nnz)      ,      ng*(nnx),jstride,DTflt,DTghosty,ierr)
      call mpi_type_vector( 1         ,ng*(nny)*(nnx),jstride,DTflt,DTghostz,ierr)

      end if

      call mpi_type_commit(DTghostx,ierr)
      call mpi_type_commit(DTghosty,ierr)
      call mpi_type_commit(DTghostz,ierr)
```

Figure 3. FORTRAN code fragment displaying the MPI calls for creating subarray and vector derived data types

Here, 'ng' is the number of ghost cells and nprnx, nprny and nprnz are the grid dimensions of the local arrays. One drawback for using MPI derived data-types is that not all MPI implementations support them as much as we like.

The SGI implementation, Message Passing Toolkit (MPT) provides an optimization for sending messages, such that for certain messages, it will perform single-buffer copy instead of the standard double-buffer copy. However, this optimization is not available for MPI derived data-types. The resulting double-buffer copying can also result in a performance bug.

In Figure 4, we see the wall-clock times for two of our four benchmark codes, FDTD3D and FCT3D. The border exchange was coded using MPI_SENDRECV and the non-blocking MPI_IRECV, MPI_ISEND and MPI_WAITALL. Two MPI implementations were timed, SGI-MPT and Intel-MPI on SGI Altix/ICE Diamond. The runtimes for the SGI-MPT are significantly slower than the Intel-MPI versions. The problem arises from the default number of buffers being too small. If we add the following environment variables to our run-script:

```
setenv MPI_BUFS_PER_PROC 1024
setenv MPI_BUFS_PER_HOST 1024
```

the performance bug disappears. SGI-MPT statistics can be obtained by setting the following environment variable:

```
setenv MPI_STATS 1
```

We also ran into a difficulty in using MPI-derived data-types with OpenMPI. The benchmarks were largely written in FORTRAN 90, and thus used the "use mpi" statement to include the MPI parameters and subroutine calls. However, the FORTRAN 90 version of the OpenMPI library does not support advanced data-types such as subarrays. The suggested workaround was to change all the "use mpi" statements to "include mpif.h," which accesses the FORTRAN 77 version of the OpenMPI libraries.

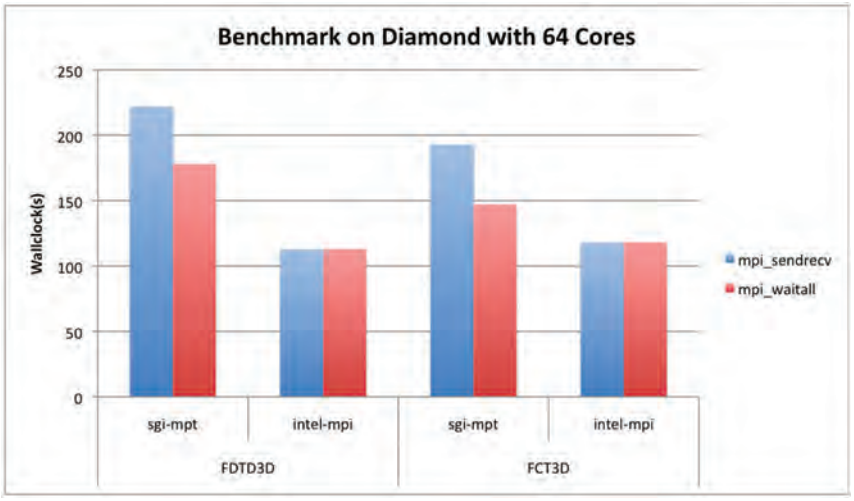


Figure 4. Performance bug for SGI-MPT using MPI-derived data-types. Wall-clock times for SGI-MPT are much slower than for Intel-MPI. Resolution of the bug results in SGI-MPT times similar to Intel-MPI.

3.3 Border Exchange

Figure 5 shows the source code for border exchange using MPI derived data-types. The code is much cleaner than that for packing by hand, but as mentioned earlier, using MPI derived data-types can cause unexpected problems.

```
call mpi_irecv(u(-2,-2,nprnz  ,m),1,DTghostz,inxt(3),6,comm3d,ireq(1),ierr)
call mpi_irecv(u(-2,-2,     -2,m),1,DTghostz,iprv(3),5,comm3d,ireq(2),ierr)
call mpi_isend(u(-2,-2,      0,m),1,DTghostz,iprv(3),6,comm3d,ireq(3),ierr)
call mpi_isend(u(-2,-2,nprnz-2,m),1,DTghostz,inxt(3),5,comm3d,ireq(4),ierr)
call mpi_waitall(4,ireq,mpi_statuses_ignore,ierr)

call mpi_irecv(u(-2,nprny  ,-2,m),1,DTghosty,inxt(2),4,comm3d,ireq(1),ierr)
call mpi_irecv(u(-2,     -2,-2,m),1,DTghosty,iprv(2),3,comm3d,ireq(2),ierr)
call mpi_isend(u(-2,      0,-2,m),1,DTghosty,iprv(2),4,comm3d,ireq(3),ierr)
call mpi_isend(u(-2,nprny-2,-2,m),1,DTghosty,inxt(2),3,comm3d,ireq(4),ierr)
call mpi_waitall(4,ireq,mpi_statuses_ignore,ierr)

call mpi_irecv(u(nprnx  ,-2,-2,m),1,DTghostx,inxt(1),2,comm3d,ireq(1),ierr)
call mpi_irecv(u(     -2,-2,-2,m),1,DTghostx,iprv(1),1,comm3d,ireq(2),ierr)
call mpi_isend(u(      0,-2,-2,m),1,DTghostx,iprv(1),2,comm3d,ireq(3),ierr)
call mpi_isend(u(nprnx-2,-2,-2,m),1,DTghostx,inxt(1),1,comm3d,ireq(4),ierr)
call mpi_waitall(4,ireq,mpi_statuses_ignore,ierr)
```

Figure 5. FORTRAN code fragment for border exchange using MPI-derived data-types

Figure 4 also demonstrates that there does not seem to be a performance difference between MPI_SENDRECV and non-blocking MPI_IRECV, MPI_ISEND, and MPI_WAITALL. In the literature, one can find strong recommendations for either method.[8,9] However, the former has an additional point of synchronization over the latter and, in the past, we have consistently implemented the latter.

3.4 MPI I/O

We also took this opportunity to assess the performance of MPI input/output (I/O). By fiat, MPI I/O uses MPI derived data-types. In Figure 6, we show the necessary code for writing subarray data-types to a single file. The resulting output file is suitable for reading back in with either standard FORTRAN I/O or MPI-I/O.

```
integer(kind=MPI_OFFSET_KIND) :: idisp
real(4), allocatable :: v(:,:,:)

allocate(v(nn(1),nn(2),nn(3)))

sizes (1:3) = nc
subs  (1:3) = nn
starts(1:3) = nlo - (/1,1,1/)

call mpi_type_create_subarray(ndim,sizes,subs,starts,MPI_ORDER_FORTRAN,MPI_REAL4,DTFout,ierr)
call mpi_type_commit(DTFout,ierr)

v=a(1:nn(1),1:nn(2),1:nn(3))

idisp= 0
call mpi_file_open(mpi_comm_world,trim(afile)//'.dat',mpi_mode_create+mpi_mode_wronly,mpi_info_
null,ifile,ierr)
call mpi_file_set_view (ifile,idisp,MPI_REAL4,DTFout,"native",mpi_info_null,ierr)
call mpi_file_write_all(ifile,v,size(v),MPI_REAL4,istat,ierr)
call mpi_file_close    (ifile,                          ierr)
```

Figure 6. FORTRAN code fragment MPI-I/O

Table 2 shows the parallel files system and the MPI implementation that was used on each system. The Mesh benchmark code was used to compare MPI I/O to FORTRAN I/O. In each case a $1{,}024^3$ single- precision array (4GB) was written once to disk using 64 cores. Hence, for the FORTRAN I/O, 64 separate files were created. The timings include the time to create the single-precision array, excluding ghost cells, from the original double-precision array. Figure 7 shows the side-by-side comparison, and in most cases we see that the FORTRAN I/O was either comparable to the MPI-I/O, or significantly faster. As a result, we recommend that if users are performing a lot of I/O that they avoid MPI-IO, and forgo the convenience of having one file containing their output data. For the lustre file systems, we did briefly experiment with setting the striping parameter, but they did not seem to affect our timings.

Furthermore, we would also have to recommend against using MPI-derived data-types. From our own interactions with users, most do not use MPI-derived data-types, or MPI-IO. We speculate that vendors are aware of this and thus, do not prioritize their optimization.

Table 2. Benchmarked parallel file systems

Core	Vendor	Platform	Parallel File System	MPI Implementaion
itaniums	sgi	morpheus	cxfs	sgi-mpt
		seraph	cxfs	sgi-mpt
		hawk	gpfs	sgi-mpt
xeons	lnx	mjm	gpfs	openmpi
	sgi	harold	lustre	sgi-mpt
		diamond	lustre	sgi-mpt
	dell	mana	lustre	openmpi
opterons	cray	jade	lustre	xt-mpt
		einstein	lustre	xt-mpt
		garnet	lustre	xt-mpt
PowerPCs	ibm	davinci	gpfs	ibm

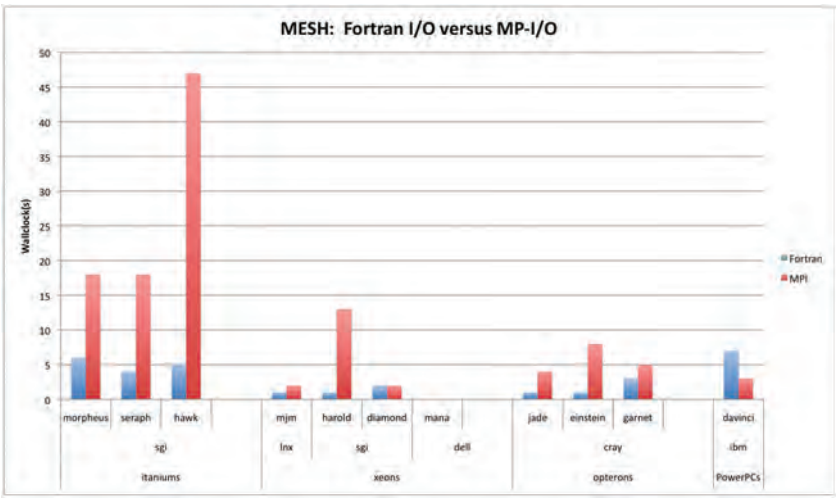


Figure 7. Comparison of Parallel FORTRAN I/O versus MPI-I/O for various DoD HPCMP platforms

Table 3. Benchmark characteristics

	mesh	kfdtd	fdtd3d	fct3d
Grid size	1024^3	1024^3	600^3	512^3
Number of arrays for Border exchange	1 Dead/alive value	6 Ex,Ey,Ez,Hx,Hy,Hz	1 Wave function	5 Density,velocity,energy
ghost cells	1	1	2	6,3
message size	.5MB	1.5MB	.3MB	.36MB
comm/comp	3%	15%	27%	5%
time-steps	1000	200	100	1000
wall clock time	1017s	258s	145s	199s

4. Benchmarks

In this section, we briefly describe the four benchmarks and try to characterize their computational aspects. The codes are meant to be easy to port, easy to modify with short runs that can scale up to 1,000s of cores.

4.1 mesh

The first benchmark, mesh, is a three-dimensional (3D) cellular automaton based on Conway's Game of Life.[10] This code was motivated by trying to debug the other three codes. Any problems we find in the other codes, we try to implement in this one. Since the code is exceptionally simple, it can be passed on to the help desk people at Consolidated Customer Assistance Center (CCAC)[11] without the need for a detailed explanation of the code.

4.2 kfdtd

The second benchmark, kfdtd, an electro-magnetics code that uses Finite-Difference Time-Domain (FDTD)[12] to advance the simulation of a single-point source at the center of the computational grid.

4.3 fdtd3d

This code uses FDTD to solve the acoustic wave equation in 3D.[13] An underwater acoustic wave impinges upon an ocean bubble plume with two different densities. For the benchmark version of the code, dispersion due to impact with the bubble plume has been turned off.

4.4 fct3d

This last code uses the Flux-Corrected Transport (FCT) algorithm[14] to simulate the "Bursting Diaphragm" problem in 3D. Computations are performed on one-dimensional (1D) arrays by looping over 3D arrays in each direction.

4.5 Characteristics

Table 3 attempts to characterize the four benchmarks in terms of a 64 core run on Raptor. The table shows the computational grid-size, the number of arrays required to be updated for border exchange, the number of ghost cells, message size, ratio of communication to computation, number of time-steps for the benchmark run, and the corresponding wall-clock time in seconds. All of the benchmarks have reasonably large computational domains, and should be accomplishing reasonable amount of computation up to 1000s of cores. The table also indicates that mesh and fct3d both do more computation per communication than the other two benchmarks.

5. Compilers

Table 4 displays the compilers and versions available on the two types of platforms, the Cray XE6s and the SGI Altix ICE/Nehalems. The benchmarks were run only on diamond and raptor, the two largest systems, respectively. Benchmarks were compiled with the Intel-11.1.072 and PGI-11.0.0 on their respective platforms.

Table 4. Compiler versions

	Intel	PFI	CCE
chugach@ARSC	12.0.0.084	11.2.0	7.2.8
garnet@ERDC		10.9.0	7.2.5
mana@MHPCC	11.1.073	11.0.0	7.3.1
harold@ARL	11.1.069		
diamond@ERDC	11.1.072		

Table 5 shows the compiler options that were employed. The left-hand column is an attempt to categorize the options in order to make comparisons across different compilers. OLVL is the optimization level used, MATH are the switches affecting floating-point performance, LOOP indicates the switches affecting loop optimization such as unrolling, INLN indicates the in-lining options, VECT indicates the vectorizing options, although this optimization may be included in other options, TUNE indicates the switches for a particular architecture and OTHR indicates commonly used switches not falling into the other categories. For the fct3d benchmark, the time-step is determined by the steep gradients in the simulation, and thus was very sensitive to floating-point optimization. Agreement between platforms could only be accomplished by substituting the MATH options with the IEEE switches.

Table 5. Compiler options

Category	ifort	pgf90	crayftn
MATH	-fp-model fast=2 -IPF fma -no-prec-div -IPF-fp-relaxed	-Mfprelaxed	-O fp3 -O scalar
LOOP	-unroll=2	-Munroll=c:1 -Mrl3	-O fusion2 -O nooverindex
INLN	-ip	-Mipa -Mautoinline	-O inlinelib -O ipa5
VECT		-Mvect=sse -Mscalarsse	-O vector3
TUNE	-xSSE4.2	-tp instanbul-64	-h cpu=mc8
OTHR	-opt-prefetch -ftz –pad –align –auto	-Mcache_align	-O cache3 -xomp
IEEE	-fp-model strict	-KIEEE	-O fp0

Category	ifort	pgf90	crayftn
MATH	-fp-model fast=2 -IPF fma -no-prec-div -IPF-fp-relaxed	-Mfprelaxed	-O fp3 -O scalar
LOOP	-unroll=2	-Munroll=c:1 -Mrl3	-O fusion2 -O nooverindex

6. Results

In order to ease the benchmarking process, we chose only to run on the two largest systems, Diamond and Raptor.

Figure 8 shows the wall-clock run-times, in seconds, for the four benchmarks running on core counts 64, 96, 128, 160, 192, 244, 256, 384, 512, 640, 768, 896, and 1,024. It seems for the lower core counts, benchmark runs for diamond are significantly faster than raptor and in some cases, twice as fast.

There appear to be "blips" in the plot for the fdtd3d benchmark runs on raptor for core counts of 256 and 384. These two core counts are significantly slower than their neighboring core counts. If we use core counts of 255 and 385, benchmark run-times fall in line with their neighboring runtimes. If we use the CCE compiler, crayftn, we again observe no blips.

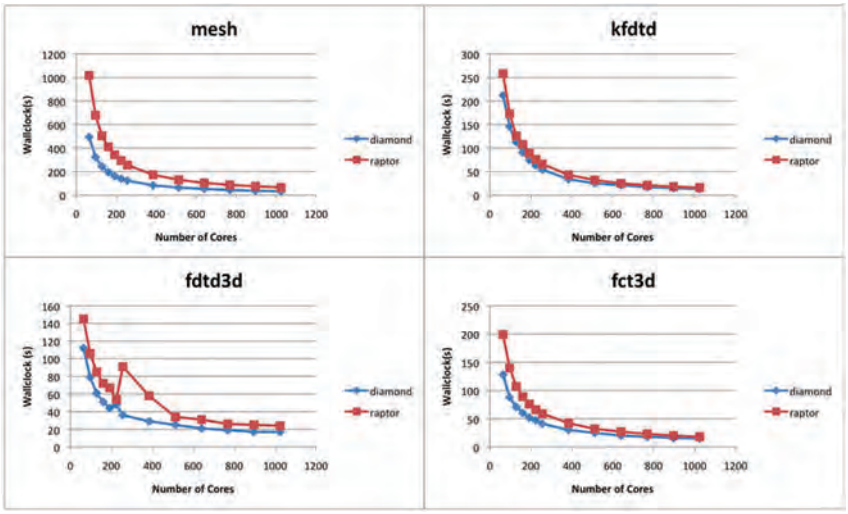


Figure 8. Core-by-core comparison for the four benchmarks

Because of the significant difference in run-times, we wanted to obtain a better assessment of the benchmark runs in terms of just scaling, so we developed another metric called the scale factor.

Let wc(n) be the wall-clock time, in seconds, of the benchmark run on n cores. Let tct(n) be the total computational time of the benchmark on n cores and let sf(n) be the scale factor on n cores. We define the scale factor as just the normalized total computational time to the run on 64 cores, our smallest core count:

$$tct(n) = wc(n) * n$$
$$sf(n) = tct(n) / tct(64)$$

A horizontal line indicates perfect scaling. Figure 9 shows the results for the four benchmarks and we observe very similar scaling for both systems, even though the raptor system is using only half the nodes of diamond. The mesh and kfdtd benchmarks exhibit very good scaling as their plots are almost completely horizontal. fdtd3d and fct3d do not scale as well. fdtd3d performs some single processor I/O to initialize (reading in the dispersion data for the bubble plume) the simulation, and this takes a constant amount of time, 11 seconds, regardless of the number of cores. fct3d scaling problems arise from the way border exchange is performed and also from integration along the z-axis.

The border exchange loads the relevant portions of the computational arrays into a buffer and then sends the buffer as an MPI argument. However, the way the computational arrays are set up, the left-most index is not the x-direction index, but rather the physical quantity index (from 1–5, density, all three velocities and the energy), resulting in non-unit memory striding.

Integration in the z-direction involves loading a 3D array into a 1D array along the z-direction. For large arrays, this will thrash cache. For the smaller arrays involved here, we still are seeing some cache misses and lack of stride-one memory access.

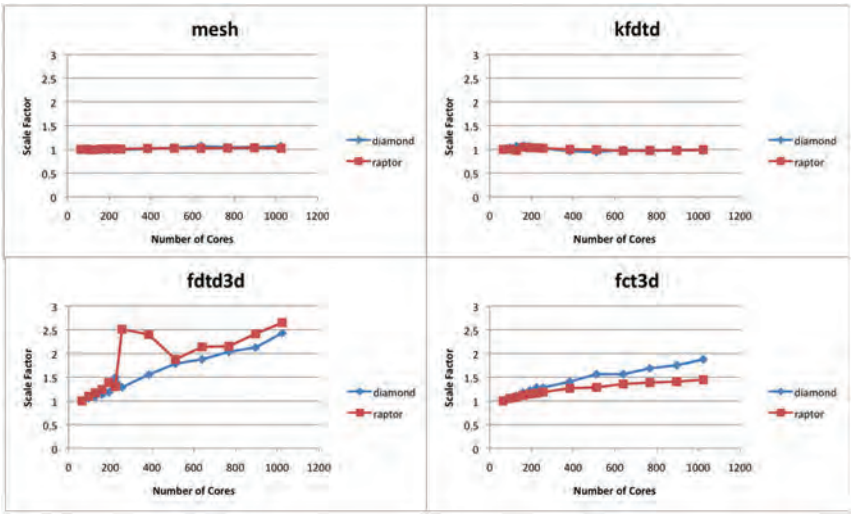


Figure 9. Scale factor for the four benchmarks

The last figure, Figure 10, compares the benchmarks on the two systems on a node-by-node basis. Here we see, at the larger node-counts, raptor either matches or exceeds the performance of diamond. This is quite different from our initial performance assessment.

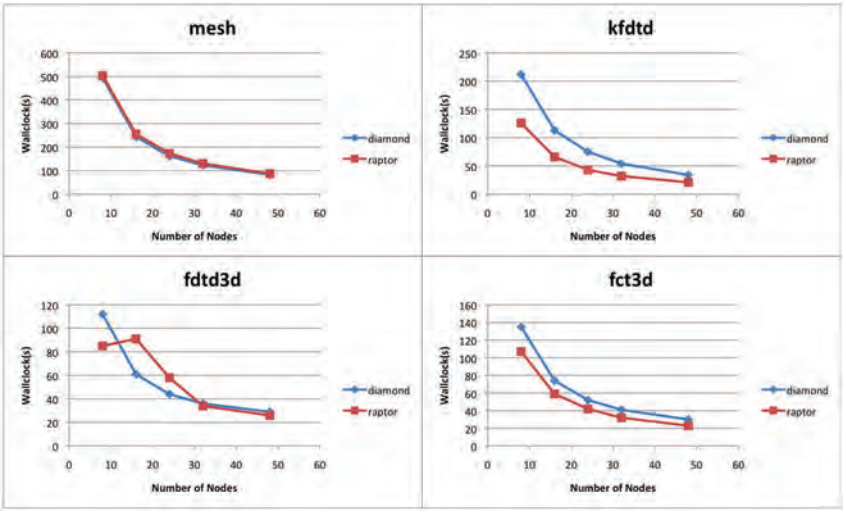


Figure 10. Node-by-node comparison for the four benchmarks

7. Conclusion

Although our initial comparison of the latest Cray XE6 systems to the older SGI Altix ICE/Nehalem systems was disappointing, we showed that the former systems compare well on a node-by-node basis, as opposed to a core-by-core basis. Due to the benchmarking process, we encountered some problems with using MPI-derived data-types for border exchange and MPI-IO. Thus we must recommend against using them until the various MPI implementations provide better support.

Acknowledgements

This work was performed on the SGI Altix ICE system at ERDC and the Cray XE6 system at AFRL under the auspices of the US Department of Defense High Performance Modernization Program.

References

1. HPCMP Benchmarks, *Making the Most of Your Allocation*, http://www.benchmarking.hpc.mil/.

2. Rosenberg, R., S. Bique, M. Koop, K. Andersen, and C. Kung, "Exploring Best-Practices for the DSRCs with Benchmarking", *DoD Proceedings of the HPCPM Users Group Conference 2010*, Schaumburg, IL, June.

3. Shrout, R., "Inside the Nehalem: Intel's New Core i7 Microarchitecture", http://www.pcper.com/reviews/Processors/Inside-Nehalem-Intels-New-Core-i7-Microarchitecture?aid=608, August 25, 2008.

4. Cost of an Intel Xeon x5560 Nehalem. 2.8GHz, http://www.newegg.com/Product/Product.aspx?Item=N82E16819117181.

5. *Cost of an AMD Opteron 6136 Magny-Cours 2.4GHz*, http://www.amazon.com/AMD-Opteron-Magny-Cours-Processor-OS6136WKT8EGOWOF/dp/B003BYRHMQ.

6. Foster, I.T., *Designing and Building Parallel Programs*, Addison-Weselye Publishing Company, Inc., p. 30, 1995.

7. Snir, M., S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI – The Complete Reference, Volume 1, The MPI Core*, 2nd Edition, The MIT Press, pp 123–189, 1998.

8. Rabenseifner, R., *Optimization of MPI Applications*, University of Stuttgart, High-Performance Computing Center, https://fs.hlrs.de/projects/par/par_prog_ws/pdf/mpi_optimize_3.pdf.

9. Kjostad, F.B. and M. Snir, "Ghost Cell Pattern", *ACM, ParaPLoP'10*, Carefree, AZ, March 30–31, 2010.

10. *Cellular Automata*, Random (Blog), http://nbickford.wordpress.com/2010/05/19/cellular-automata/.

11. Consolidated Customer Assistance Center (CCAC), http://www.ccac.hpc.mil/.

12. Taflove, A. and S.C. Hagness, *Compuational Electrodynamics, The Finite-Difference Time-Domain Method*, Artech House, Inc, Norwood, MA, 2000.

13. Rosenberg, R., G. Norton, J.C. Novarini, W. Anderson, and M. Lanzagorta, "Modeling Pulse Propagation and Scattering in a Dispersive Medium: Performance of MPI/OPenMP Hybrid Code", *IEEE Proceedings of Supercomputing 06*, Tampa, FL, 2006.

14. Boris, J.P. and D.L. Book, *J. Computational Physics*, Volume 11, p. 38, 1973.

Sustained Systems Performance Test: Delivering Performance to HPCMP Users

Paul M. Bennett
US Army Engineer Research and Development Center DoD Supercomputing Resource Center (ERDC DSRC), Computational Science and Engineering Group, Vicksburg, MS
paul.m.bennett@usace.army.mil

Abstract

The High Performance Computing Modernization Program (HPCMP) sustained systems performance (SSP) test continues to play an important role in monitoring the performance delivered to users of the HPCMP's big iron systems. The test benchmarks a subset of the codes used in the system acquisition cycle. The purpose is to quantitatively evaluate updates to system software, hardware repairs, modifications to job queuing policies, and revisions to the job scheduler. The codes chosen for SSP have proven migration capability to HPCMP HPC systems and non-empirical tests for numerical accuracy. Metrics such as compilation time, queue wait time, benchmark execution time, and total test throughput time are gathered and compared against data from previous tests to monitor the systems-under-test while minimizing impact to the users. Jobs failing to execute properly or in anomalously short or long times are investigated, and the results are reported to systems administrators and Center Directors at each Center for appropriate actions.

During the past year, the SSP test has been instrumental in surfacing configuration issues with certain implementations of Message Passing Interface (MPI) on HPCMP equipment, and detecting other performance issues on several HPC systems. In certain instances, effective work-around techniques have been implemented to recapture lost performance. Additionally, the SSP suites have been adapted for use by each Department of Defense Supercomputing Resource Center (DSRC) to locally monitor system health post-preventive maintenance.

1. Introduction

The Department of Defense (DoD) High Performance Computing Modernization Program (HPCMP) owns and operates eleven major high performance computing (HPC) systems at five DoD Supercomputing Resource Centers (DSRCs). Through these resources, the Program delivers HPC capability to its users in support of the modern Warfighter. Established practices for operating and maintaining the HPC systems dictate that updates should be regularly applied to the operating systems, libraries of numerical routines, communication libraries, other assorted software, and to the configuration and policies governing operation of the job schedulers, if warranted. The goals are typically to fix or work around as many bugs as possible, to implement more efficient numerical routines, to implement better communication algorithms, to close security holes, or to improve job throughput and queue wait times. Ideally, such updates would be expected to make the system-under-test (SUT) more stable and efficient over time, but this cannot be guaranteed. Moreover, improvements in numerical processing and throughput may be offset by more stringent security procedures or hardware failures of aging equipment. Nor do changes to scheduling policies or scheduler configurations necessarily lead to improved job throughput.

These considerations raise the issue of how to quantitatively monitor the cumulative effects of system updates and aging hardware of the SUT over its lifetime. Such measurement should be rigorous, to give good quantitative results, yet lightweight, to minimally impact the production environment. In Kramer, et al. (2005), at the National Energy Research Scientific Computing Center, a sustained systems performance (SSP) test is described, which is based upon application codes in use on the SUT. The application codes in use in the SSP test are determined by characterizing the workload on the SUT in some appropriate manner. Kramer, et al. cite a specific tool that gathers communication and hardware counter-profiling data unobtrusively. The tool generates data that are then used to help determine the specific codes, numeric algorithms, and the scientific purpose for which the codes are being used, along with input from the SUT's users. Kramer, et al. states that since it is impossible to get full-coverage of the complete workload, a representative subset is chosen based, in part, upon prominence. This is determined by the number of cycles each code uses, and its importance to the scientific community. Portability of the codes also plays a role in selection, but codes that merely perform the actions of a

computational kernel are inadequate to measure the performance of different system features as they interact in execution of a complex parallel code. The metric that Kramer, et al. describe is based upon the total wall-clock resource utilization at the scales used in production runs.

The DoD HPCMP SSP test is similar. Although the workload of each SUT is not covered fully, it is represented by a subset of codes chosen for portability and prominence in the workload anticipated during procurement. As such, the SSP codes are most conveniently taken from the set of the codes used in the Technology Insertion (TI) process. The TI codes themselves are selected based upon requirements in addition to usage, and not by gathering communication data or obtaining hardware counter-profiling data. The computational technology area (CTA) and the type of numerical research that is performed also play a role, as well as the prominence of the codes in the overall HPCMP research program. As the TI suite changes from procurement cycle to procurement cycle, the SSP suite based upon it also changes. For this reason, each SSP suite is named following the convention used to refer to the TI suite from which it derives. For example, SSP-07 is derived from the TI-07 suite, and SSP-08 is derived from the TI-08 suite. The latest addition is the SSP-10 suite, chosen from the TI-10 suite used during the latest complete procurement cycle.

In this paper, the codes and test cases comprising the SSP suites currently in use on the major DoD HPCMP HPC systems are described. A brief description of versions of the SSP suites that have been assembled for use by the DSRCs is included. Representative results are presented, and recent and ongoing developments in SSP testing across the Program are described. The paper concludes with a brief summary and statement of the current status of SSP testing in the HPCMP.

2. Formulation of the SSP Suites

Currently, there are four SSP suites used to conduct quarterly tests on the major HPCMP HPC systems. Summary reports from tests using these suites are delivered to the HPCMP and are disseminated to DSRC personnel. Four similar suites, essentially identical except lacking certain source codes covered under ITAR regulations governing dissemination of export-controlled software, have been provided to the HPCMP DSRCs to conduct performance tests on their own equipment following any of several events covered under the Baseline Configuration Team's SSP testing policy. A member of each DSRC has been named as the point-of-contact for SSP testing, and these individuals were trained in SSP testing procedures in October 2010 at the behest of the Baseline Configuration Team.

The SSP-07 suite is comprised of one user application from each of the HPCMP's principal CTAs; namely Computational Structural Mechanics (CSM); Computational Fluid Dynamics (CFD); Computational Chemistry, Biology, and Materials Science (CCM); Computational Electromagnetics and Acoustics (CEA); and Climate/Weather/Ocean Modeling and Simulation (CWO). The Air Vehicles Unstructured Solver (AVUS) is a CFD code, and CSQ to the Three-Halves (CTH) is used in CSM. The General Atomic and Molecular Electron Structure System (GAMESS) is used in CCM, and the HYbrid Coordinate Ocean Model (HYCOM) is a CWO code. The Out-Of-CORE (OOCORE) solver is a CEA code. Each application has two test cases, so-called standard and large. The codes use Message Passing Interface (MPI) for communications, although GAMESS can also use shmem, 1-sided MPI on IBM systems, and sockets under Transmission Control Protocol/Internet Protocol (TCP/IP), and HYCOM can use shmem and OpenMPI. The benchmarks are conducted at the distinguished core counts from TI-07, which is 64 for the AVUS, CTH, GAMESS, and OOCORE standard test cases, and 384 for the large. The HYCOM standard test case executes at 59 cores, and the large test case executes at 385. The SSP-08 suite uses the same codes and similar test cases, but the distinguished core counts for its benchmarks are 128 for most standard test cases except HYCOM, which uses 124, and 512 for most large test cases, again except for HYCOM, which uses 504.

The codes in the SSP-08 suite are the same as in SSP-07, but GAMESS and HYCOM are later releases. OOCORE is maintained by the Computational Science and Engineering group at US Army Engineer Research and Development Center (ERDC) DSRC for the purposes of performance benchmarking, so it does not have a release per se. However, lessons learned in TI-07 were applied to make minor modifications to OOCORE in TI-08, so the version in SSP-08 is not quite the same as in SSP-07. The SSP-08 benchmarks are executed on 128 cores for most standard test cases and on 512 cores for most large test cases. However, the HYCOM standard test case executes on 124 cores, and the HYCOM large test case executes on 504.

In practice, AVUS and HYCOM were found to surface many of the same performance issues. Additionally, GAMESS used TCP/IP protocol over sockets on many of the systems-under-tests, so AVUS was dropped from the SSP-09 suite, and one of the GAMESS test cases was removed. OOCORE was also dropped from the SSP-09 suite. Therefore, the codes in the SSP-09 suite are CTH, with a standard and large test case; GAMESS, with a large test case only; and HYCOM, with a standard and large test case. All three codes are at later releases than in the two prior SSP suites. In order to address certain

performance issues not easily surfaced by these user applications, SSP-09 also has two synthetic application codes, namely OSBENCH, which measures percentage performance losses arising from operating system jitter, and MultiMAPS, which measures the data bandwidths between the various levels of the system's memory hierarchy and the CPU. OSBENCH is essentially the code P-SNAP authored at the Los Alamos National Laboratory for the purpose of quantifying interference or noise by the operating system. MultiMAPS is developed by the Performance Modeling and Characterization Institute at the San Diego Supercomputing Center. The CTH standard test case executes on 256 cores, and the HYCOM standard test case executes on 250. On the other hand, the CTH and GAMESS large test cases execute on 1,024 cores, and the HYCOM large test case executes on 1006.OSBENCH requires 64 cores, but MultiMAPS requires exactly 1.

The SSP-10 suite is similar to the SSP-09 suite. The user application codes are CTH, GAMESS, and HYCOM, but CTH is a later release, that being the version that was used in acceptance testing. Additionally, the GAMESS and HYCOM codes include extensive modifications to the source that were made by the vendor to generate optimized benchmarks used in acceptance testing. The SSP-10 test cases are identical to the SSP-09 test cases, however, and the synthetic application codes are also the same as in SSP-09. The SSP-10 test cases require the same node counts as the corresponding SSP-09 test cases.

2.1 AVUS

AVUS computes discrete solutions, using a parallel implicit solver, to compressible Euler and Navier-Stokes equations subject to the ideal gas equation of state (Tomaro, et al., 1997). More information about AVUS' numeric algorithms can be found in Cable, et al. (2005), Tracy, et al. (2003), and Tracy (2005). AVUS' models can be two-dimensional (2D), three-dimensional (3D), or axi-symmetric spaces, and they permit unstructured grids with arbitrary cell types. The grid is broken into subdomains, called groups, blocks, or zones. Each zone resides on a separate processor in a domain decomposition enabling parallel processing. This is accomplished using ParMETIS, the MPI-based, parallel grid-partitioning library that performs both static and dynamic graph partitioning and fill-reducing reordering. ParMETIS is available from the Karypis Laboratory at the University of Minnesota. The zones produced by ParMETIS are roughly equally sized, ensuring good load balancing. Each zone's surface area is minimized, which reduces communications overhead to a minimum. Consequently, AVUS has good load balancing with excellent scalability that allows high-computational intensity requiring little communication. More detailed information on ParMETIS may be found in Karypis, et al. (1997), Karypis and Kumar (1998), Karypis and Kumar (1999), and Schloegel, et al. (2000), to name a few of numerous references.

Most of AVUS is written in FORTRAN90, but one auxiliary library and ParMETIS are written in C. AVUS uses non-blocking communication to perform point-to-point exchanges of messages.

Both standard and large test cases are three-dimensional (3D) and model turbulent viscous flow. The run-times of the cases are set by advancing the solutions forward in time-varying numbers of steps. The standard test case, referred to as the "wing-flap big" data set by the developers, is a wind tunnel model of a wing with a flap and endplates. It has 7,287,723 cells and runs for 200 time-steps in SSP-07. In SSP-08, it runs for 900 time-steps. The large test case, called the "waverider" dataset by the developers, is a generic configuration for a supersonic/hypersonic vehicle that "rides" a shock wave that forms below the vehicle at supersonic speeds, generating lift for the vehicle. This model has 31,080,000 cells and runs for 200 time-steps in SSP-07 and 400 time-steps in SSP-08.

2.2 CTH

CTH is a family of codes developed by Sandia National Laboratory to model complex multidimensional problems arising in the study of objects assembled from multiple materials subjected to large deformations and strong shocks. Therefore, problems modeling penetration and perforation, compression, and detonations can all be explored with the CTH software package. It features many different numerical models from which to choose, and it can solve problems on a static grid or on an adaptive mesh over a physical domain. More information about CTH can be found in Hertel, et al. (1992) and Hertel, et al. (1993).

CTH is written in C and FORTRAN. The particular code used for the benchmarks computes physical models. Thus, both standard and large test cases feature a rod 7.67 cm long, 0.767 cm in diameter, and made of 10 materials impacting a plate made of 8 materials that is 0.64 cm thick. The impact occurs at an angle of 73.5 degrees, and the initial speed of the rod is 1,210 m/s. The numerical results are validated by looking at the penetration depth of the rod at the final time-step.

The standard test case in all four SSP suites uses adaptive mesh refinement. In SSP-07, there are up to five mesh refinement levels, and a maximum of 480 8×8×8 blocks of data are allowed to each MPI process. The simulation runs for 500 time-steps. In SSP-08, there are also a maximum of five refinement levels, but each process is allowed up to 600 8×8×8

blocks of data; the simulation runs for 600 time-steps. In SSP-09 and SSP-10, there are a maximum of six refinement levels, each MPI process is allowed up to 225 8×8×8 blocks of data, and the simulation runs for 350 time-steps.

In contrast to the standard test problem, the large test problem uses a fixed-grid with 1,840×230×460 cells globally. The collective memory requirement is 204 GBytes of memory, and the problem scales strongly. There are 190 time-steps in the SSP-07 benchmark, 400 time-steps in the SSP-08 benchmark, and 300 time-steps in the SSP-09 and SSP-10 benchmarks.

2.3 GAMESS

GAMESS is a quantum chemistry code currently maintained by Ames Laboratory and Iowa State University. It is available online at http://www.msg.ameslab.gov/GAMESS/GAMESS.html. It originated primarily from an early version of HONDO, which was developed with funding provided by the National Science Foundation, the Department of Energy (DOE), and IBM. GAMESS' development is ongoing at Iowa State University with sponsorship from the US Air Force Office of Scientific Research and the DOE. Many individuals and several research organizations have contributed to the development of GAMESS, a complete listing of which may be found in the user's guide included in the source code distribution. The guide also has information on GAMESS' capabilities, which are extensive, how to use them, and installation instructions for many HPC platforms. More information can be found in Gordon and Schmidt (2005) and Schmidt, et al. (1993).

GAMESS is written almost exclusively in FORTRAN77, although the communication routines, forming the so-called Data Distribution Interface or DDI layer, are written in C. Its algorithms feature numerical quadrature rules applied to determine the electron shell structure of molecules resembling those of current interest to the DoD. The integrals are computed using analytic expressions and recursion formulae that employ a variety of basic functions to approximate the electron shells. After initially setting up the model, the integrals are computed using little or no communication. Results are assembled at the end using collective communication calls. Much of the numerical heavy lifting is performed by calls to Level 3 basic linear algebra routines in tuned numerical libraries, such as IBM's ESSL, Cray's libsci, or Intel's MKL.

The standard test case used in SSP-07 and SSP-08 performs a restricted Hartree-Fock calculation using density functional theory (DFT) to compute the nuclear gradient vector of a polyhedral oligomeric silsesquioxane. The electron shells are approximated by self-consistent field wave equations described by the split-basis 6-311G** functionals, as described in INPUT.DOC included in the source code distribution. The basis functionals use six Gaussian-type orbital (GTO) functions and two additional Slater-type orbitals to model the valence shells that hold the molecule together. For hydrogen atoms, the additional orbital is modeled by a p-type polarization function, but for all heavier atoms, it is modeled by a d-type polarization function, according to REFS.DOC in the source code distribution and the Computational Chemistry entry at http://en.wikipedia.org. More information about this test case can be found in Bennett, et al. (2006).

The large test case used in all four SSP suites also performs a restricted Hartree-Fock computation, but it uses a second-order, Moller-Plesset correction method to compute the nuclear gradient vector of an anion of a potent advanced chemical oxidizer. The computation employs perturbations in the form of excitations intended to account for electron correlation, according to REFS.DOC in the source code distribution. The basis set for the electron shells in SSP-07 and SSP-08 is the same as for the standard test case. However, in addition to the extra Slater-type valence orbitals found in the standard test case, there are also an extra diffuse s-shell on hydrogen atoms and an extra diffuse sp-shell on atoms in the second and third rows of the periodic table, i.e., lithium, beryllium, and heavier nonmetals, halogens, metalloids, and post-transition metals in those two rows, according to REFS.DOC in the source code distribution. More information about this test case can also be found in Bennett, et al. (2006). A different basis set is used for the SSP-09 and SSP-10 benchmarks, specifically a Dunning-type Correlation Consistent basis set with d-type polarization augmented with a set of diffuse functions. The molecule is the same as in SSP-07 and SSP-08, but it is in a slightly different configuration.

2.4 HYCOM

HYCOM was developed by the HYCOM Consortium, which is part of the US Global Ocean Data Assimilation Experiment funded by the National Ocean Partnership Program. It is an open-source code, available at http://www.hycom.rsmas.miami.edu. It uses traditional isopycnic coordinates in the open stratified ocean, like its predecessors NLOM, the Naval Layered Ocean Model, and MICOM, the Miami Isopycnic-Coordinate Ocean Model, both developed by the Naval Research Laboratory. However, HYCOM has the added capability that it can transition smoothly between different types of vertical coordinates in shallow or mixed-layer waters. It uses a generalized hybrid vertical coordinate system that transitions between traditional isopycnic vertical coordinates in deep stratified water to coordinates that follow terrain in shallow coastal regions, and also to z-level coordinates in the mixed layer and un-stratified ocean. Additional information about HYCOM may be found in Bleck (2002).

HYCOM consists of 31,000 lines of FORTRAN90 with a timing routine written in C. The models are fully-global and capable of resolving eddies. Laterally, HYCOM uses a conventional second-order, finite-difference hydrostatic primitive ocean scheme. Vertically, HYCOM uses an arbitrary Lagrange Eulerian coordinate system. The globe is tiled, and all tiles consisting solely of land are discarded. Each processor is assigned one tile, resulting in core counts that typically do not fully populate all cores of all the nodes used for processing HYCOM jobs. More detailed information may be found in Leach, et al. (2005).

The HYCOM test cases are both global ocean models. The standard test case has ¼-degree resolution at the equator, and it simulates one model day. It includes a representative amount of file input/output (I/O), computation, and communication. The initial state is generated from representative ocean profiles. The HYCOM large test case has 1/12-degree equatorial resolution. It simulates 18 hours, or ¾ day, and it also generates a representative amount of file I/O, computation, and communication. It is initialized using representative ocean profiles.

2.5 OOCORE

OOCORE is the primary numerical kernel of an electromagnetics code that may not be used in either system acquisition or in performance monitoring. It was developed by Oak Ridge National Laboratory, the University of Tennessee at Knoxville, and several other universities, as part of the ScaLAPACK project. It was included in the original TI-07 suite in order to represent the core execution of SWITCH, a proprietary electromagnetics code developed by Northrop Grumman that cannot be used as an application benchmark code. Its work is to solve a large linear system of equations using a parallel LU decomposition included in the ScaLAPACK package available from the University of Tennessee at Knoxville. More information on OOCORE can be found in Blackford, et al. (1997) and Fatoohi (2008). A single OOCORE input parameter allows the user to artificially restrict the amount of main memory available for storing the matrix, so large disk I/O can be ensured as needed for testing purposes such as performance benchmarking. Since OOCORE can be configured to perform by far the largest amount of disk I/O of any application code in the SSP-07 and SSP-08 suites, it is the best available test to monitor a platform's disk I/O performance.

The standard test case runs the solver in in-core fashion. That is, it uses main memory to store the LU decomposition. This is accomplished by setting the problem to use only 80 percent of the memory available per core, allowing the matrix to fit in memory. However, in the large test case, each core is forced to store a relatively small maximum of 1.8×10^6 matrix elements in main memory, forcing the calculation to be executed out-of-core. Thus, the SUT opens two files for each MPI process for file I/O. In SSP-07, the standard test case solves a linear system with 53,000 double-complex unknowns, and the large test case solves a system with 78,000 double-complex unknowns. In SSP-08, the standard test case uses 62,000 unknowns, and the large test case uses 82,000 unknowns.

3. Recent Results

This section discusses some recent performance measurements with the SSP suites. These results are not exhaustive.

3.1 SSP-07 on *Mjm* (Linux Networx Advanced Technology Cluster at ARL DSRC)

The SSP-07 has been used to monitor the performance of *Mjm*, the Linux Networx Advanced Technology Cluster at the US Army Research Laboratory (ARL) DSRC, since Spring 2008. Figure 1 charts the cumulative SSP performance of *Mjm* since inception. Whereas the intent is that a system's acceptance test benchmarks should be used as the reference data, other benchmarks must be used for those systems already in production mode when SSP testing is initiated. Therefore, the reference data were generated during the first SSP test performed on *Mjm*. During its lifecycle, *Mjm* has had a couple of major events with the potential to significantly modify its performance as measured by SSP. These are removal of the Infiniserv and SilverStorm MPI software, replaced by OpenMPI, and replacement of the LSF job scheduler by PBS. The first revision occurred when SilverStorm Technologies, Inc., was acquired by Qlogic. Shortly afterwards, an announcement was made that the MPI software would no longer be supported. The change in MPI software was made during the 2nd quarter FY2009. The second event was mandated by the HPCMP as part of the change to PBS program-wide, and it occurred at the end of FY2009. Consequences of the change in scheduler included disabling the GAMESS large test case from executing, and disabling both HYCOM test cases from executing, also. All of these issues came about because of the "nonstandard" manner in which these benchmarks are scheduled. The GAMESS large test case requires sufficiently large amounts of memory that only half of the cores on each node can be used, effectively allowing use of nearly twice as much memory by the compute processes. Such scheduling requires a nonstandard statement in the job script's preamble that tells PBS to schedule only 2 MPI processes on each node and not 4. This was not known during the first SSP test under PBS

on *Mjm*. On the other hand, the HYCOM benchmark jobs required 1 or 2 of the nodes to be underutilized, for the simple reason that 59 and 385 are not evenly divisible by 4. Necessarily, the standard test case required the 16th node to have only 3 MPI processes on it, and the large test case required a 97th node with 1 MPI process on it. The problem was then how to inform PBS that it must schedule and initialize such tasks. As with the GAMESS large test case, this was not known during the first SSP test under PBS on *Mjm*. It was fixed at that time, but broke again during the Q2 FY2010 test. There have also been numerous software updates, including revisions of the operating system, compilers, and numerical libraries.

Figure 1a presents the performance data for SSP testing on *Mjm* from inception up to the 4th Quarter, FY2010 inclusive. Of particular note is that the performance of every benchmark except for the GAMESS benchmarks was off from the previous quarter's by at least 20 percent. The performance of the HYCOM large test case was so slow that it was not completing in 4 hours, which was a multiple of the benchmark time by at least 8. A major clue to possible causes of the performance is indicated in the GAMESS benchmarks. They performed as usual, but the GAMESS binary did not use MPI at all, instead following the developer's advice to use TCP/IP protocol over sockets. The ARL DSRC was notified of the problem immediately, and upon investigation, determined that something had happened with OpenMPI 1.4, an upgrade to OpenMPI 1.3 the previous quarter. Indeed, James Ianni found references to OpenMP calls within the OOCORE binaries, which are supposed to use MPI only, not a mixture of MPI and OpenMP. The ARL DSRC determined the cause of the performance problem and repaired it. Figure 1b shows the performance after the repairs, and it also presents the performance in the first 2 quarters of FY2011, showing that indeed, *Mjm* is performing nicely as far as SSP is concerned. Additionally, James Ianni has determined that OOCORE benefits from setting certain environment variables to non-default values, which has led to significant performance improvements in both test cases.

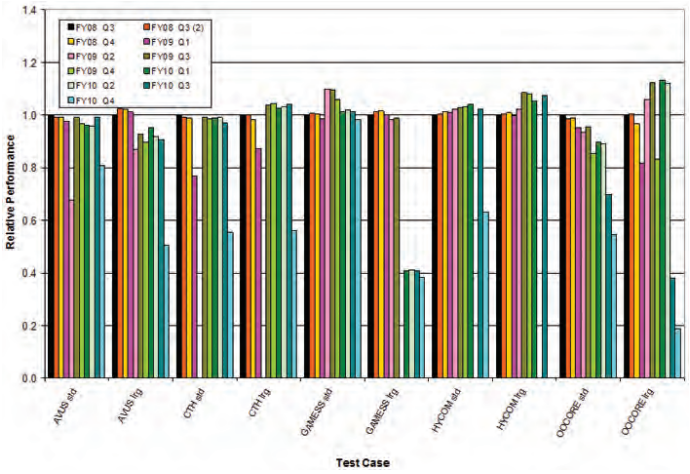


Figure 1a. Cumulative SSP performance on *Mjm*, Linux Networx Advanced Technology Cluster at ARL DSRC, from inception to Q4 FY2010 inclusive

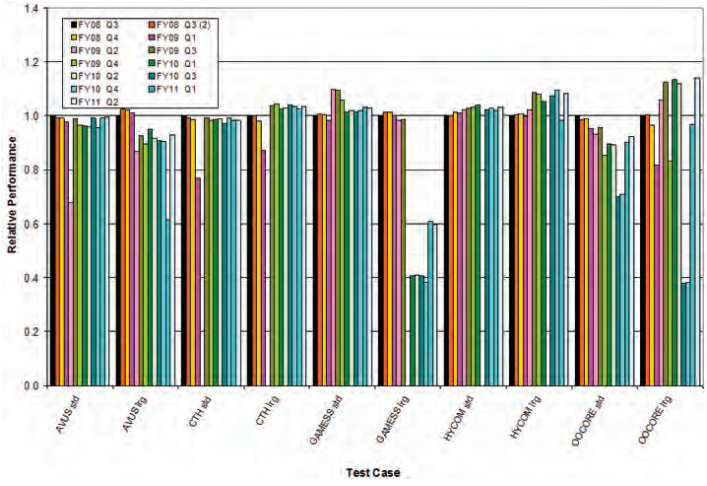


Figure 1b. Cumulative SSP performance on *Mjm*

3.2 SSP-08 on *Einstein* (Cray XT5 at Navy DSRC)

The SSP-08 suite has been used to conduct the SSP test on *Einstein*, the Cray XT5 at the NAVY DSRC, since *Einstein's* inception. Figure 2 charts the cumulative SSP performance. The reference data are the acceptance test benchmarks generated by the vendor. Cray tuned the AVUS and GAMESS codes to generate the corresponding acceptance test benchmarks. Furthermore, the GAMESS benchmarks were generated using a different communication mode than the developers recommended. This was not known at the time of the initial few SSP tests, so performance differences in the GAMESS benchmarks in the first few quarters are actually a result of different communication modes, shmem vs. MPI, and the tuned GAMESS code.

During the past year, the CTH benchmarks had some problems. The first noticeable problem arose when a system update to the Pathscale user environment forced that compiler to modify its licensing process with more rigorous checks for the pre-compiler, and this broke the compilation. The update occurred between the 1st and 2nd quarter FY2010 SSP tests. Although the issue was reported to the NAVY DSRC via Consolidated Customer Assistance Center (CCAC) as soon as it was determined to be a problem with the compiler and not user error, a solution was not found before the end of that quarter's benchmark window. The fix was to change the call to the pre-compiler in the CTH makefile to use "cc" and not "pathcc".

The second issue affecting the CTH benchmarks arose in the 4th quarter FY2010 SSP test. In that test, the Pathscale C compiler was not able to compile the CTH source code. The problem could not be diagnosed, and it was clearly not a problem with a bad definition of the pre-compiler. It was reported to the Navy DSRC via CCAC right away, and their response was that it was a known problem with the Pathscale compilers, a result of updating the operating system and communications libraries. A solution was to be provided during the next preventive maintenance cycle. However, this did not alleviate the problem of obtaining SSP benchmarks for CTH. Therefore, the user environment was changed to use the PGI compilers, and compilation was successful. Unfortunately, the first attempt to execute the CTH SSP benchmarks did not pass with a successful numerical check. The jobs were resubmitted, but only results of the large test case could be retrieved before the end of the fiscal year. The PGI compiler was used again in the 1st quarter of FY2011, and the CTH benchmarks were much more successful, passing the numerical and giving good performance. The PGI compiler suite had been updated in the meantime.

The only other problem affecting SSP benchmark performance was a noted issue with file I/O, surfaced by the OOCORE benchmarks. The problem was reported to the NAVY DSRC via CCAC. It was most likely a consequence of updates to communication routines or numerical routines in libsci. The performance was repeated in the 2nd quarter of FY2011, but shortly after that test, *Einstein's* architecture was changed from barcelona to shanghai, with attendant updates to the operating system, communications layer, PGI compilers, numerical library libsci, and an improvement in processor speed. Christine Cuicchi at the NAVY DSRC requested an additional SSP test to be performed shortly after the system update, and across the board, the performance of every benchmark improved significantly.

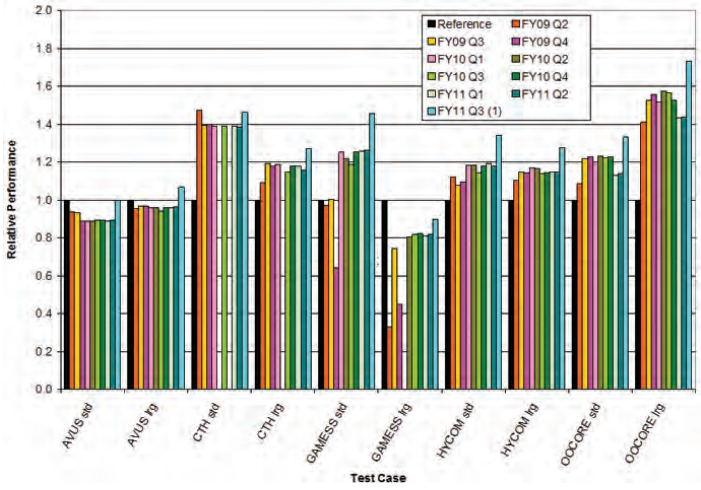


Figure 2. Cumulative SSP performance for *Einstein*, Cray XT5 at NAVY DSRC

3.3 SSP-09 on *Diamond* (SGI Altix ICE 8200 at ERDC DSRC)

The SSP-09 suite has been used to conduct the SSP test monthly on *Diamond*, the SGI Altix ICE 8200 at the ERDC DSRC, since that system's inception. The reference data for all of the user application codes and the MultiMAPS synthetic application code are the benchmarks generated by SGI during acceptance testing. Dave Anderson of SGI generated the reference data for OSBENCH. For the CTH large test case benchmarks, SGI assigned MPI processes to cores in such a way as to maximize the performance, and for HYCOM, SGI tuned the include files so as to achieve the best performance. Usage of tuned HYCOM include files was not understood for the first few SSP benchmarks. The MPI process assignment is not used for the CTH large test case benchmarks in SSP, but the tuned include files are used to generate the HYCOM binaries. For the first few months of production operation, the SSP test was conducted monthly. The period changed to quarterly for the 4th quarter FY2010 test.

Figure 3 charts the cumulative SSP performance of the user application codes against the reference performance since inception. Aside from transient issues related to contention for the interconnect, the user application codes have surfaced no major issues. The only point to be made for the user application benchmarks is that the large test cases have wider variation in times, which tends to obscure the presence of systemic issues.

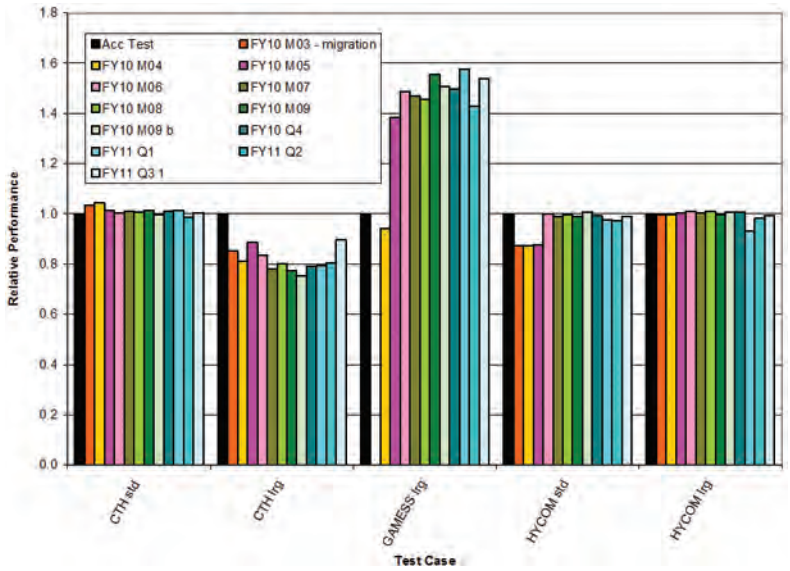


Figure 3. Cumulative SSP user application code performance on *Diamond*, SGI Altix ICE 8200 at ERDC DSRC

Figures 4a and 4b chart the cumulative performance of the SSP synthetic benchmark codes. In particular, Figure 4a indicates an issue surfaced by MultiMAPS during the test in the 2nd quarter FY2011. The problem was that the data bandwidth from every level of the memory hierarchy to the CPU was off from the previous quarter by about 18 percent, relative to the reference performance, for all three levels of cache, and off by about 37 percent from main memory. The issue was reported to the ERDC DSRC via CCAC. The ensuing investigation revealed that although system benchmarks had been executed following a preventive maintenance cycle during which some system software was updated, a condition had occurred in which settings to inform the Intel Nehalem architecture that it is to execute in "turbo" mode were overwritten with settings to perform in standard mode. Owen Lagarde, the system administrator, determined the conditions under which such settings were modified and repaired the problems. Additional MultiMAPS benchmarks were performed that indicated that the performance observed in the previous quarter had been achieved. He later requested an additional SSP test to check *Diamond's* performance following some additional maintenance. That test indicated satisfactory performance was maintained. The results are charted in Figure 4b.

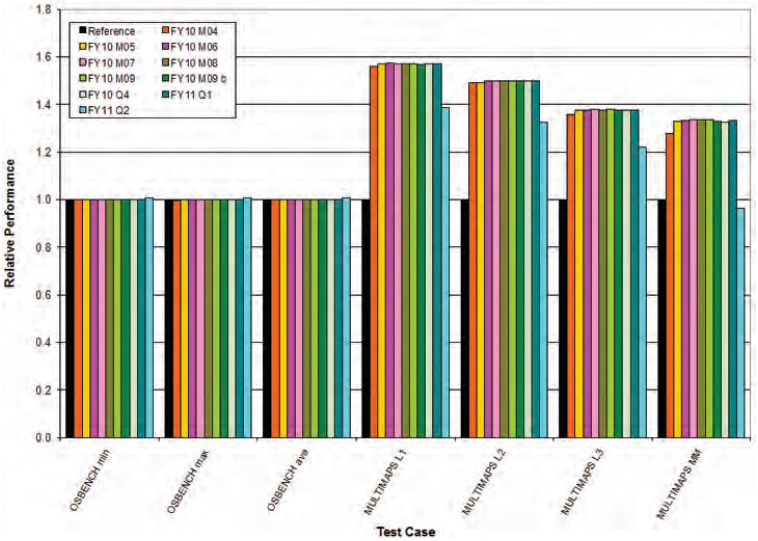


Figure 4a. SSP synthetic application performance inclusive from inception up to Q2 FY2011 for *Diamond*, SGI Altix ICE 8200 at ERDC DSRC

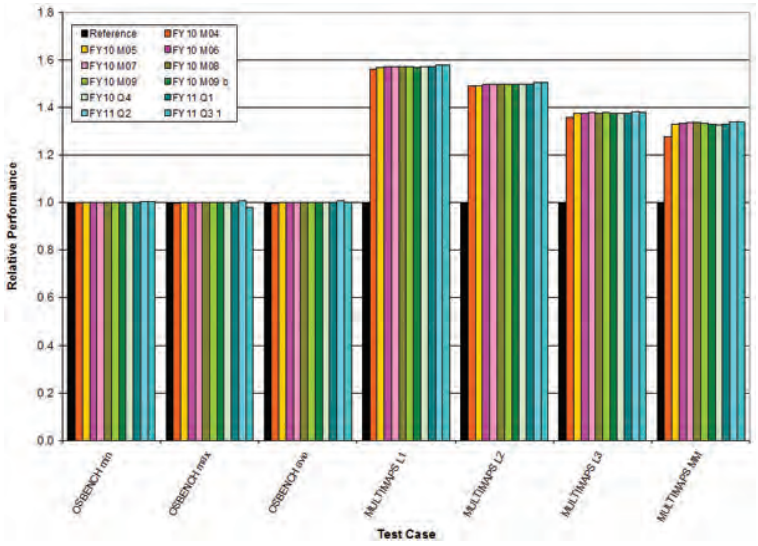


Figure 4b. Cumulative SSP synthetic application code performance on *Diamond*

3.4 SSP-10 on *Garnet*, *Raptor*, and *Chugach* (Cray XE6 systems at ERDC, AFRL, and ARSC DSRCs)

The SSP-10 suite has been assembled in draft form to execute upon the Cray XE6 systems procured in TI-10. The effort to get the individual codes to compile and execute on Garnet at the ERDC DSRC and *Raptor* at the AFRL DSRC began in February 2011. Unfortunately, several unexpected outages on each system, lengthy compilation times on both systems, and problems arising when the file system on *Raptor* inexplicably lost files made it impossible to finish the initial migration effort by the mid-April reporting deadline. However, the GAMESS large test case was benchmarked on *Raptor*, this success occurring on April 27, 2011. The benchmark time was 1,190 seconds using a binary compiled with the PGI compilers. These compilers are not the default, requiring the user to change the Cray compiler module to the PGI compiler module. Cray's benchmark time, obtained during acceptance testing, was 1,056 seconds, giving in turn 88.7 percent performance relative to the reference. The Cray compilers were used to generate the binary in acceptance testing.

4. Conclusion

The SSP testing program outlined in Kramer, et al. (2005) has been implemented, with appropriate modifications to procedure and constituent codes, for use in monitoring the performance of DoD HPCMP HPC systems. Several suites, named following the particular TI in which each SUT was procured, are currently in use to surface performance issues and bring them to the attention of system administrators and vendors if necessary. The performance effects of updates to operating systems, compilers, numerical libraries, and communication libraries are evaluated periodically, and the results are reported to the HPCMP DSRC Directors and other appropriate Center personnel. Copies of the suites lacking codes restricted from dissemination under ITAR rules governing handling of export-controlled codes have been posted on the SSP website to make them available to the SSP POC at each DSRC. The POCs have been so informed and have taken delivery of the suites. The intent is for each DSRC to administer the SSP test appropriate to each system upon occurrence of an event, as defined in the policy governing SSP testing issued by the Baseline Configuration Team, with the intent to shorten the time-to-correction or more rapidly enable amelioration of negative performance issues. However, routine monitoring by the Computational Science and Engineering group at the ERDC DSRC continues, keeping the HPCMP and the DSRCs informed of the overall performance of their systems. SSP testing continues to surface issues that can adversely impact user code performance, and thereby reduce the effectiveness of project allocations and the quality of service delivered to the users by the DSRCs, the system vendors, and the HPCMP.

Acknowledgments

This work was funded by the DoD HPCMP with grants of computer time and resources on HPC systems Program-wide under the project HPCMO02B. We would like to thank Brent Andersen and Rick Roberts at the AFRL DSRC; Dr. James Ianni, George Petit, Justin La Barre, and Duncan Schulze at the ARL DSRC; Bob Alter, Owen LaGarde, and Dr. Alan Scheinine at the ERDC DSRC; Michele Hershey, Leo Ohler, Chris Young, and Mark Payba at the MHPCC DSRC, and Christine Cuicchi, Sheila Carbonette, and Wes Tabor at the NAVY DSRC for their assistance in investigating and resolving performance issues. Thanks are also due to Carrie Leach of the ERDC DSRC and to Dr. Jerry Boatz at the US Air Force Research Laboratory at Edwards AFB for their assistance with migrating SSP-10 to the Cray XE6 systems, and to Dr. Alan Wallcraft at the US Naval Research Laboratory at Stennis Space Center for his assistance with HYCOM. We would like to thank Dave Anderson at SGI, Dr. Bill Ward at the HPCMP, and Mike Gough at the ERDC DSRC for their assistance in providing reference data for SSP benchmark analysis. We would like to thank Glen Browning at the ERDC DSRC and Greg Brewer at the AFRL DSRC for their assistance in administering a benchmark website to facilitate SSP suite delivery and reporting and for posting the SSP suites for the DSRCs. Finally, we would like to thank last, but most assuredly not least, Tim Sell and Lloyd Slonaker at the AFRL DSRC, Dr. James Ianni at the ARL DSRC, Ed Kornkven at ARSC, Bob Alter at the ERDC DSRC, Leo Ohler at the MHPCC DSRC, and Christine Cuicchi at the NAVY DSRC, who are the SSP POCs.

References

Bennett, P.M., S.B. Cable, R.W. Alter, M. Mahmoodi, and T.C. Oppe, "Targeting CCM-, CEA-, and CSM-Based Computing to Specific Architectures Based upon HPCMP Systems Assessment", *Proceedings of the HPCMP Users Group Conference 2006*, pp. 360–366, Denver, CO, 2006.

Blackford, L., A. Cleary, J. Choi, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R.C. Whaley, *ScaLAPACK Users' Guide*, User manual, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

Bleck, R., "An oceanic general circulation model framed in hybrid isopycnic-Cartesian coordinates", *Ocean Model*, 4(1), pp. 55–88, 2002.

Cable, S.B., T.C. Oppe, W.A. Ward, Jr., R.L. Campbell, Jr., R.E. Gordnier, V.S. Burnley, M.J. Grismer, and P.G. Buning, "CFD-Based HPCMP Systems Assessment Using AERO, AVUS, and OVERFLOW-2", *Proceedings of the HPCMP Users Group Conference 2005*, Nashville, TN, pp. 349–355, 2005.

Fatoohi, R., "Performance evaluation of NSF application benchmarks on parallel systems", *Proceedings of the 22nd IEEE International Symposium on Parallel and Distributed Processing*, IPDPS 2008, Miami, FL, 2008.

Gordon, M.S. and M.W. Schmidt, "Advances in electronic structure theory: GAMESS a decade later", *Theory and Applications of Computational Chemistry, the first forty years*, eds. C.E. Dykstra, G. Frenking, K.S. Kim, and G.E. Scuseria, Elsevier, Amsterdam, 2005.

Hertel, Jr., E., R. Bell, M. Elrick, A. Farnsworth, G. Kerley, J. McGiaun, S. Pemey, S. Silling, P. Taylor, and L. Yarrington, "CTH: A software family for multi-dimensional shock physics analysis", *Technical Report SAND-92-2089C*, Sandia National Laboratories, Albuquerque, NM, 1992.

Hertel, Jr., E.S. et al., "CTH: A Software Family for Multi-Dimensional Shock Physics Analysis", *Proceedings of the 19th International Symposium on Shock Waves*, Marseilles, France, Volume 1, pp. 377–382, 26–30 July 1993.

Karypis, G. and V. Kumar, "Multilevel k-way partitioning scheme for irregular graphs", *Journal of Parallel and Distributed Computing*, 48(1), 1998.

Karypis, G. and V. Kumar, "Parallel multilevel k-way partitioning scheme for irregular graphs", *SIAM Review*, 41(2), pp. 278–300, 1999.

Karypis, G., K. Schloegel, and V. Kumar, "PARMETIS: Parallel graph partitioning scheme and matrix ordering library", Technical report, University of Minnesota, Department of Computer Science and Engineering, 1997.

Kramer, W., J. Shalf, and E. Strohmaier, "The NERSC Sustained System Performance (SSP) Metric", *Paper LBNL-58868*, Lawrence Berkeley National Laboratory, 2005.

Leach, C.L., T.C. Oppe, W.A. Ward, Jr., and R.L. Campbell, Jr., "CWO-Based HPCMP Systems Assessment Using HYCOM and WRF", *Proceedings of the HPCMP Users Group Conference 2005*, Nashville, TN, pp. 356–359, 2005.

Schloegel, K., G. Karypis, and V. Kumar, "A Unified Algorithm for Load-balancing Adaptive Scientific Simulations", Supercomputing 2000.

Schmidt M.W., K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S.J. Su, T.L. Windus, M. Dupuis, and J.A. Montgomery, "General Atomic and Molecular Electronic Structure System", *J. Comput. Chem.*, 14, pp. 1347–1363, 1993.

Tomaro, R.F., W.Z. Strang, and L.N. Sankar, "An Implicit Algorithm for Solving Time-Dependent Flows on Unstructured Grids," *AIAA 97-0333*, 35th Aerospace Sciences Meeting and Exhibit, Reno, NV, Jan. 1997.

Tracy, F.T., T.C. Oppe, W.A. Ward, Jr., and R.E. Peterkin, Jr., "A survey of the algorithms in the TI-03 application benchmarking suite with emphasis on linear system solvers", *Proceedings of the 2003 Users Group Conference*, pp. 332–336, Bellevue, WA, 2003.

Tracy, F.T., "Role of Algorithms in Understanding Performance of the TI-05 Benchmark Suite", *Proceedings of the HPCMP Users Group Conference 2005*, Nashville, TN, pp. 420–426, 2005.

HPCMP UGC 2011

8. Computational Methods

Advancing Computational Capabilities with Heterogeneous Platforms

Song J. Park, Dale R. Shires, and Brian J. Henz
US Army Research Laboratory (ARL), Aberdeen Proving Ground, MD
{brian.j.henz, dale.shires}@us.army.mil

James A. Ross
High Performance Technologies, Inc., Aberdeen, MD
jross@hpti.com

David A. Richie
Brown Deer Technology, Forest Hill, MD
drichie@browndeertechnology.com

Abstract

The configuration of a general-purpose processor with graphics integrated is gaining popularity in portable consumer electronics and supercomputing centers. These heterogeneous systems are easily found in everyday consumer products, such as desktops, laptops, and smart-phones. As processing systems evolve toward increased parallel units, the correlation between software and performance needs to be addressed. The advancement of parallel processors has been placing a responsibility in software for extracting processor's full computing power. In this environment, leveraging the massively-parallel architectures is the key to enhancing the current computational capability. To reach maximum processing potential, a programmer needs to take specific details of hardware into account. Although several programming languages exist for targeting heterogeneous platforms (e.g., CUDA, DirectCompute, and OpenCL), the knowledge of hardware characteristics and execution models is applicable across different languages. Accordingly, hardware features and sustainable performance are examined for an architectural study of graphics processors. In conjunction, this research investigates the application of heterogeneous platforms toward a tactical environment and simplifying access to the power of graphics processing unit (GPU)-enhanced systems in the context of an algorithm for situational awareness.

1. Introduction

Embracing parallelism is the future of computing. This is, in part, due to the development of microprocessors increasing the number of cores without the advancement in clock frequency. A performance boost in multi- and many-core architectures requires parallelization of an algorithm to take advantage of all the available cores. For further acceleration, leveraging graphics processing units is another logical and attractive option due to graphics processing units' (GPUs) consumer market, where modern PCs and laptops are already equipped with a GPU. Moreover, graphics processing technology is being extended to areas such as Amazon Elastic Compute Cloud, Apple A4 chip, Intel's Sandy Bridge, and Tianhe-1A supercomputer. With Intel and AMD invested in graphics processing, future systems will most likely be composed of a mixture of computing resources or a heterogeneous platform. Therefore, these evidences suggest that an ideal compute platform would be a CPU/GPU combination system for achieving maximum raw-performance and software flexibility.

The ballistic threat simulation and Synthetic Aperture Radar (SAR) processing applications are analyzed for data points presented in this study. The ballistic algorithm is based on the first-hit ray-casting method with a quad-tree hierarchical search. The core kernel computes line-plane intersections and applies a ballistic-hit probability function to predict the threat. Moderately detailed three-dimensional representation of buildings and terrain map serves as an input for testing the ballistic threat simulation. Another application under analysis is a SAR image processing algorithm, which uses a back-projection method for output image reconstruction. The main computations within the back-projection routine are distance calculations to determine signal strength at points of interest. A comprehensive description is presented in Reference 1.

2. Architecture Analysis

In an attempt to understand the computational capabilities of processors, specifications of architecture resources are collected for analysis. Maximum theoretical processing power is calculated by examining the number of floating-point operations that can be completed by hardware per cycle. As a start, instruction throughput values for AMD Evergreen[2] and Nvidia compute capability 2.0[3] are summarized in Table 1. For simplicity, instructions MAD, ADD, and MUL are focused for instruction rate of the stream processor for AMD and the multiprocessor for Nvidia. For single-precision, both AMD's and Nvidia's processing unit can complete one 32-bit floating-point operation per cycle. As for double-precision, AMD has a higher throughput for addition operation than multiplication, and Nvidia's throughput is half of the single-precision numbers. In reference to integer arithmetic, addition instruction is the most efficient per core in the AMD and Nvidia cases.

Table 1. Throughput of arithmetic instructions

Hardware	Instruction	Single Precision	Double Precision	Integer
AMD Evergreen	MAD	5	1	1
	ADD	5	2	5
	MUL	5	1	1
Nvidia Compute Capability 2.0	MAD	32	16	16
	ADD	32	16	32
	MUL	32	16	16

In x86 processors with Streaming SIMD extensions (SSE) support, the instruction set uses 128-bit vector registers to pack floating-point operations. A single SSE instruction can operate on two packed double-precision operations or four packed single-precision values. With two SSE execution units, eight single-precision or four double-precision floating-point operations can be completed per cycle on Nehalem micro-architecture. Accordingly, to calculate the peak performance, the clock-frequency was multiplied by the number of cores, and this result is then multiplied by four for 64-bit precision and by eight for 32-bit precision.

For NVIDIA graphics processors, the number of shader processors and the shader clock-rate are considered for peak-performance calculation. Since each shader processor is capable of executing one fused multiply-add (FMA) per clock or one multiply-add (MAD) per clock, maximum single-precision floating-point operations can be calculated by multiplying the number of shader processors, shader clock-frequency, and two single-precision floating-point operations (FMA or MAD). As for double-precision, Fermi cards can compute 16 double-precision FMA per multiprocessor per clock, which leads to half of single-precision performance. However, for gamer GeForce boards, double-precision arithmetic is limited to 1/8 of the single-precision speed.

Similarly, the number of shaders and shader clock-frequency are used for determining the peak processing power for AMD graphics cards. Each processing element can execute floating-point or integer operations. The equation for peak single-precision rate can be determined by multiplying processing elements, clock-frequency, and two (FMA or MAD). In modern AMD architecture, double-precision floating-point operations are performed by connecting two or four processing elements[2].

To relate theoretical peak to experimental results, double-precision general-matrix multiply (DGEMM) implementation was searched in the literature. DGEMM consists mostly of MAD instructions, which should lead to an ideal correlation to peak marketed performance. AMD Cypress XT GPU is able to achieve 465 giga floating-point operations per second (GFLOPS) or 85% peak[4]. Intel Core2 Quad processor is reported to reach 42 GFLOPS or 88% peak[5]. Nvidia Fermi is measured to compute 350 GFLOPS or 70% peak[6].

Table 2 summarizes the hardware specifications collected for mid-level to high-end products (price checked in April of 2011) for Nvidia, AMD, and Intel processors. A list of products is separated by vendors and organized according to price. The purpose was to show a snapshot of processor options for computation-challenged applications.

Conventionally, GPU's theoretical-performance FLOP-rates are derived from its ability to complete one MAD or FMA instruction, which equates to two floating-point operations per cycle. If the benchmark consisted of only multiply instructions, then the achievable peak-performance would be reduced to half of the advertised speed. Here, the worst-case instruction throughput value is chosen to calculate the peak performance. Low-maximum-peak FLOP-rates that assume one floating-point operation per unit, as opposed to two floating-point operations, are presented in Figure 1 for single-

precision and Figure 2 for double-precision. From these floating-point performance numbers, the ranges of adjusted raw performance speed-up are 6.3x–12.7x (AMD single-precision), 2.5x–6.3x (AMD double-precision), 2.8x–8.7x (Nvidia single-precision), and 0.7x–4.8x (Nvidia double-precision). These speed-up values attempt to represent practical possible gains for real-world applications. For instance, to observe floating-point instruction distribution on an SAR imaging application, a parallel-thread execution file was examined for instruction breakdown. The chart shown in Figure 3 conveys the portion of various instruction types.

Table 2. Processor specifications for various cost range

Processor	1Q 2011 Price *(dollars)*	Processing Units	Clock Frequency *(GHz)*	TDP *(Watts)*
GeForce 460	200	336	1.35	150
GeForce 480	350	480	1.401	250
GeForce 580	500	512	1.544	244
Tesla C2050	2500	448	1.15	238
Radeon HD 6950	240	1408	0.8	200
Radeon HD 6970	360	1536	0.88	250
FireStream 9350	800	1440	0.7	150
FireStream 9370	2000	1600	0.825	225
Core i5 2500	210	32	3.3	95
Core i7 2600	300	32	3.4	95
Core i7 970	600	48	3.2	130
Xeon X5680	1700	48	3.33	130

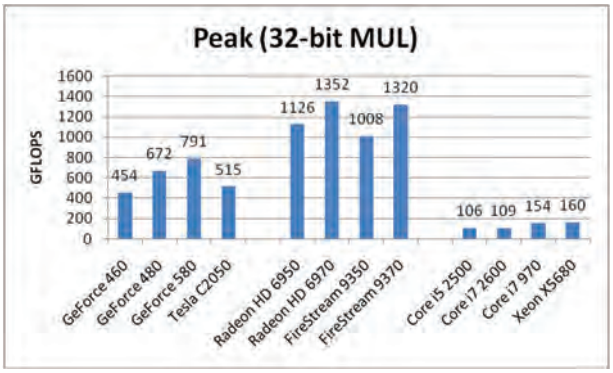


Figure 1. Single-precision multiply instruction theoretical

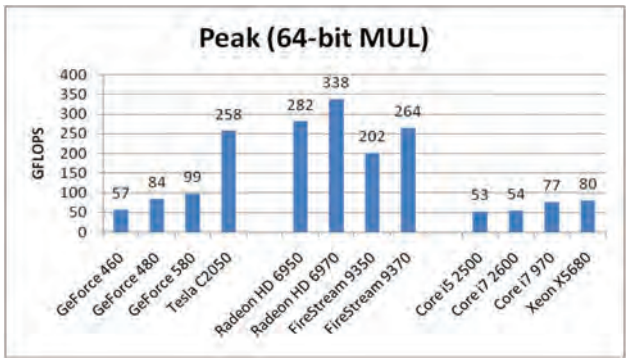


Figure 2. Double-precision multiply instruction theoretical

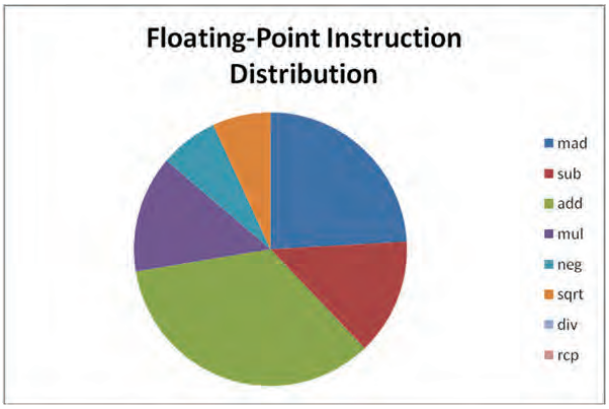


Figure 3. Floating-point instructions in SAR imaging routine

3. Run-time Results

With a broad range of hardware support, OpenCL framework has the notable advantage of portability across vendors and architectures. Therefore, ballistic threat simulation and SAR imaging algorithms were written in the OpenCL framework. For ballistic simulation, the execution time to generate a bitmap image of 448 × 240 pixels was measured on Intel, Nvidia, and AMD processors. For Intel X5570 Quad-Core, the runtime for ballistic threat probability calculation took 73.6 seconds. The ballistic OpenCL code was written with a float4 data-type to ensure SSE optimization for x86 processors, and optimal very-long instruction-word (VLIW) unit utilization for AMD graphics hardware. Excluding float4, no other optimization techniques were applied to this initial algorithm. GPU performance results are illustrated in Figure 4. Although OpenCL targets multiple vendors, the value of the local work group size was dependent on each compute device, due to architectural differences. In addition, the original kernel size written for the AMD graphics card was not compatible with Nvidia video cards, due to a 5-second kernel execution-time restriction imposed on Nvidia cards used for driving displays.

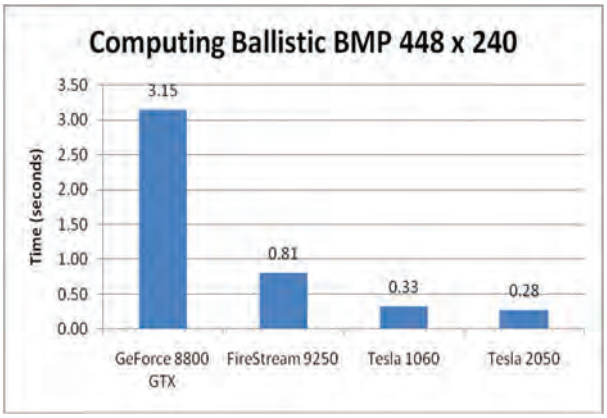


Figure 4. OpenCL execution-time for generating a ballistic threat image

The OpenCL performance results for SAR imaging algorithm are reported in Figure 5. First-time execution of OpenCL program results in a longer run-time, due to the initial compile and builds process of the OpenCL kernel. The reported measurements presented here are run-times measured following the initial run. A 4.2x speed-up is observed in kernel execution-time for dual-core and quad-core processor comparison. For Nvidia cards, the OpenCL kernel-time was decreased by 7.6x in Tesla 1060, but the algorithm is dominated by set-up and memory transfer (CPU to GPU) operations in Quadro and Tesla. Figure 6 lists total and kernel execution times for OpenCL and CUDA. The OpenCL version was an early implementation, and was not optimized to utilize local memory. For the Quadro and Tesla GPU cards, the 2-meter SAR imaging algorithm did not possess high-arithmetic-intensity or desired computational requirement.

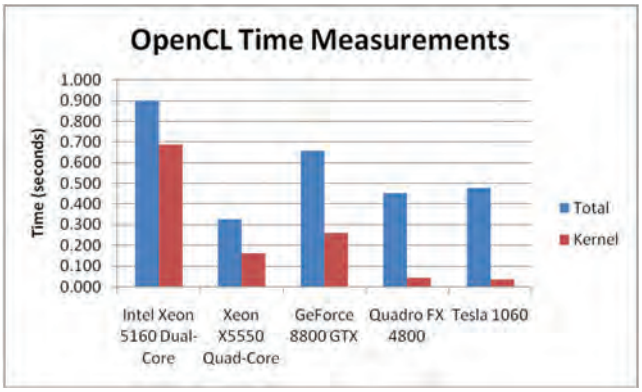


Figure 5. OpenCL total and kernel execution-time for 2 meter SAR image reconstruction

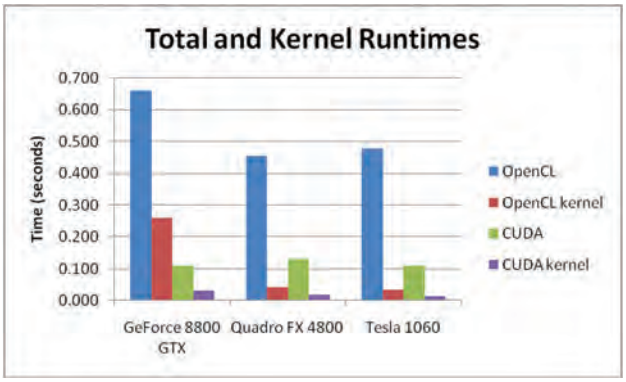


Figure 6. OpenCL and CUDA total and kernel execution-time comparison

4. Mobile Interface

The popularity of heterogeneous platforms seems to promise huge opportunity space for portable high-performance computing. With the Radeon HD 6990 GPU card advertised at 5 TFLOPS, a 20 TFLOP PC-size compute system can be easily constructed by inserting four GPU cards in a workstation. Five years ago, assuming 50% peak LINPACK on the GPUs, this computer occupying only a desk-space would be considered a supercomputer and would rank in the 2006 TOP500 list[7].

In an effort to extend the GPU-enhanced capability to average users, Web interface was developed for easy accessibility. The set-up allows any browser-capable device to connect and request GPU-accelerated ballistic application. Figure 7 shows a snapshot of an iPhone requesting the ballistic threat calculation via Safari browser. The image output of ballistic simulation is displayed on top of the Google maps background. The scenario exemplifies how mobile devices can interact with a GPU-equipped heterogeneous system. Apache Tomcat software facilitated the Web interface that placed a GPU application in a Web environment.

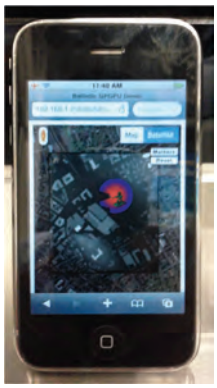


Figure 7. Mobile device access to ballistic threat simulation

5. Conclusion

Graphics processors are massively-parallel compute devices capable of providing high-throughput in floating-point and integer operations. Driven by the mass-market demand for graphics, video cards are commodity products commonly found in laptops, PCs, and workstations. Graphics chips are starting to take hold in smart-phones and tablet markets as well. With growing popularity and the ability to address general-purpose computing, GPUs have successfully assisted in many scientific and engineering fields[8]. Heterogeneous systems consisting of CPU and GPU architectures are manifesting themselves in mobile computing and supercomputing environments as an efficient computing resource. In our lab, efforts are underway to leverage heterogeneous platforms to advance soldiers' computing capability. The ballistic threat simulation and SAR image reconstruction algorithms were implemented to harness the parallelism in graphics processors. The Internet browser interface demonstrates the simple web-page access to GPU-accelerated computations, and addresses end-user interaction and interoperability. The research continues to explore parallel computing on heterogeneous systems to enhance support for tactical computing.

Acknowledgements

The authors would like to acknowledge the support provided by the DoD High Performance Computing Modernization Program office and the SAR development researchers at the RF Signal Processing & Modeling Branch in the Sensors and Electronic Devices Directorate.

References

1. Nguyen, L., "Signal and Image Processing Algorithms for the U.S. Army Research Laboratory Ultra-wideband (UWB) Synchronous Impulse Reconstruction (SIRE) Radar", *ARL-TR-4784*, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, April 2009.

2. AMD, *Programming Guide AMD Accelerated Parallel Processing OpenCL*, May 2011.

3. Nvidia, *NVIDIA CUDA C Programming Guide*, March 2011.

4. Rohr, D., M. Kretz, and M. Bach, "Technical Report, CALDGEMM and HPL," Dec. 2010.

5. Volkov, V. and J. Demmel, "Benchmarking GPUs to Tune Dense Linear Algebra", Supercomputing, Austin, TX, 2008.

6. Phillips, E., "CUDA Accelerated Linpack on Clusters", GPU Technology Conference, Sep. 2010.

7. TOP500 Supercomputer Sites, TOP 500 List – June 2006, http://www.top500.org/list/2007/06/100.

8. Nickolls, J. and W. Dally, "The GPU Computing Era", *Micro, IEEE*, vol. 30, no. 2, pp. 56–69, March–April 2010.

An Intelligent Text Recognition System on the AFRL Heterogeneous Condor Computing Cluster

Qinru Qiu
Department of Electrical & Computer Engineering, Binghamton University, Binghamton, NY
qqiu@binghamton.edu

Qing Wu, Morgan Bishop, Mark Barnell, Robinson Pino, and Richard Linderman
US Air Force Research Laboratory, Information Directorate (AFRL/RI), Rome, NY
{qing.wu, morgan.bishop, mark.barnell, robinson.pino, richard.linderman}@rl.af.mil

Abstract

Modern image processing software performs image detection and pattern recognition with fairly high accuracy given the condition that the input image is clean and fully observable. Pattern recognition becomes extremely difficult, if not impossible, when the image is partially obscured or even partially missing. Compared to computer-based image processing algorithms, the human brain exhibits extraordinary ability for pattern recognition in noisy environments because it generates anticipations based on the context of the input and knowledge of the problem. This paper presents an Intelligent Text Recognition System (ITRS) that mimics the human information processing procedure to fill in the missing or damaged text by considering the word-level and sentence-level context. The system is built upon two cognitive computing models, the Brain-State-in-a-Box (BSB) Attractor Model and the Cogent Confabulation Model. The former performs character detection and later performs word and sentence completion. Given a scanned text image where each character is 15-by-15 pixels large, experimental results show that, when 20% of the character images are damaged by a 1-pixel-wide horizontal scratch running through the center of the image where most of the information to distinguish amongst various characters is found, the ITRS recognizes complete sentences at 92% accuracy. When 60% of the character images are damaged by a 3-pixel-wide horizontal scratch located at the center, the ITRS recognizes sentences at 64% accuracy. Furthermore, when 10% of the characters are completely occluded, the ITRS recognizes sentences and words at more than 60% and 95% accuracy, respectively. When the characters increase to 30% completely occluded, the sentence accuracy drops to 20% and the word accuracy drops to 85%.

1. Introduction

The extraordinary ability of humans to recognize patterns and objects in complex scenes, even when blurred or partially-occluded, is a high-priority objective for machine automation. One important aspect of this challenge involves reading text. A significant amount of research[1,2] has been done on the application of optical character recognition (OCR) for printed characters during the past years. Present software, such as the open-source Tesseract-OCR[9], works well on clean images but struggles when the images are degraded. For example, given the image in Figure 1(a), it would be impossible to recognize the characters that are smudged or missing using only image processing techniques.

In this paper, we present a context-aware Intelligent Text Recognition System (ITRS) that serves as the physical layer of machine reading. A neuromorphic system architecture is adopted that incorporates massively-parallel high-performance computing techniques with advances in neuromorphic models of computation. The proposed ITRS system reads and processes about 16–20 pages of scanned or photographed text image on the 500 TFLOPS (Tera Floating-point Operations Per Second) Condor computing cluster with medium effort put toward performance optimization. It learns from what has been read and based on the obtained knowledge, it forms anticipations and predicts the next input image (or the missing part of the current image). Such anticipation helps the system to deal with all kinds of noise that may occur during recognition.

This application is mainly built on two cognitive computing models. They are the *Brain-State-in-a-Box* (BSB) *Attractor Model*[4], and *Cogent Confabulation Model*[7]. The BSB model is a simple, auto-associative, nonlinear, energy-minimizing neural network. It provides the pre-processing of the image of each character for the matching pattern. Cogent confabulation is an emerging computational model that mimics the Hebbian learning, the information storage and inter-

relation of symbolic concepts, and the recall operations of the brain. As the model discovers the relations between symbols, it can naturally be used to complete the missing information in a word or a sentence.

The ITRS is divided into three layers as shown in Figure 1(b). The input of the system is the text image. The first layer is character recognition software based on BSB models. It tries to recall the input image with stored images of the English alphabet. If there is noise in the image, multiple matching patterns may be found. The ambiguity can be removed by considering the word-level and sentence-level context; which is achieved in the second and third layer, where word and sentence recognition is performed using cogent confabulation models. Image processing front-end software is designed to read in the scanned images of text and separate them into blocks of smaller images of single characters. The ITRS system is evaluated using images of scanned text with missing information, i.e., text with hard-to-recognize or missing characters. Its accuracy will be reported in Section 4.

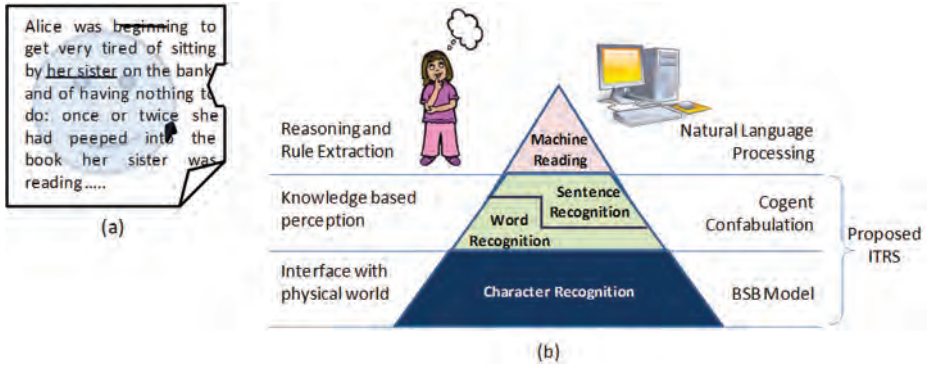


Figure 1. (a) An example of real-life text and (b) Layered architecture of intelligent text recognition

2. Background

2.1 Brain-State-in-a-Box

The BSB model is an auto-associative, nonlinear, energy-minimizing neural network[4–6]. A common application of the BSB model is to recognize a pattern from a given noisy version. The BSB model can also be used as a pattern recognizer that employs a smooth nearness measure and generates smooth decision boundaries.

There are two main operations in a BSB model, training and recall. In this paper, we will focus on the BSB recall operation. The mathematical model of a BSB recall operation can be represented in the following form[5]:

$$\boldsymbol{x}(t+1) = S\left(\alpha \cdot \boldsymbol{A} \cdot \boldsymbol{x}(t) + \lambda \cdot \boldsymbol{x}(t) + \gamma \cdot \boldsymbol{x}(0)\right)$$

where:

- $\boldsymbol{x}$ is an N dimensional real vector
- $\boldsymbol{A}$ is an $N \times N$ connection matrix
- $\boldsymbol{A} \times \boldsymbol{x}(t)$ is a matrix-vector multiplication operation
- α is a scalar constant feedback factor
- λ is an inhibition decay constant
- γ is a nonzero constant if there is a need to maintain the input stimulation
- $S(\ )$ is the "squash" function defined as follows:

$$S(y) = \begin{cases} 1 & \text{if } y \geq 1 \\ y & \text{if } -1 < y < 1 \\ -1 & \text{if } y \leq -1 \end{cases}$$

We implemented and optimized the recall operation of the BSB model on the Sony PS3® gaming consoles containing the Cell Broadband Engine processor[8]. The run-time measure shows that we have been able to achieve about 70% of the theoretical peak performance of the processor (approximately 100 out of 153 billion single precision floating-point operations per second).

2.2 Cogent Confabulation

Cogent confabulation is a connection-based cognitive computing model. It captures correlations between objects (or features) at the symbolic level and stores this information as a knowledge base. Given an observation, familiar information with high-relevancy will be recalled from the knowledge base. Based on the theory, the cognitive information process consists of two steps: learning and recall. During learning, the knowledge links are established and strengthened as symbols are co-activated. During recall, a neuron receives excitations from other activated neurons. A "winner-takes-all" strategy takes place within each lexicon. Only the neurons (in a lexicon) that represent the winning symbol will be activated and the winner neurons will activate other neurons through knowledge links. At the same time, those neurons that did not win in this procedure will be suppressed.

Figure 2 shows an example of lexicons, symbols and knowledge links. The three columns in Figure 2 represent three lexicons for the concept of shape, object and color with each box representing a neuron. Different combinations of neurons represent different symbols. For example, as shown in Figure 2, the pink neurons in lexicon I represent the cylinder shape, the orange and yellow neurons in lexicon II represent a fire extinguisher and a cup, while the red neurons in lexicon III represent the red color. When a cylinder-shaped object is perceived, the neurons that represent the concepts "fire extinguisher" and "cup" will be excited. However, if a cylinder shape and a red color are both perceived, the neurons associated with "fire extinguisher" receives more excitation and becomes activated while the neurons associated with the concept "cup" will be suppressed. At the same time, the neurons associated with "fire extinguisher" will further excite the neurons associated with its corresponding shape and color, and eventually make those symbols stand out from other symbols in lexicon I and III.

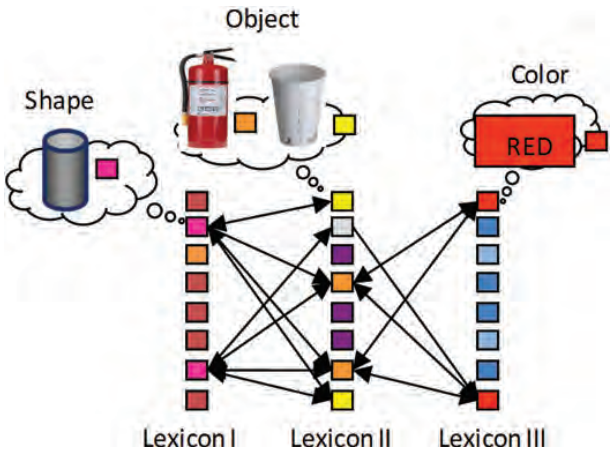


Figure 2. A simple example of lexicons, symbols and knowledge links

A computational model for cogent confabulation has been proposed by Hecht-Nielsen[7]. Based on this model, a lexicon is a collection of symbols. A *knowledge link* (KL) from lexicon I to II is a matrix with the row representing a source symbol in lexicon I, and the column representing a target symbol in lexicon II. The *ij*th entry of the matrix represents the strength of the synapse between the source symbol si and the target symbol $\boldsymbol{t}_j$. It is quantified as the conditional probability $P(s_i \mid \boldsymbol{t}_j)$. The collection of all knowledge links is called a *knowledge base* (KB). The knowledge bases are obtained during the learning procedure. During recall, the excitation level of all symbols in each lexicon is evaluated. Let l denote a lexicon, F_l denote the set of lexicons that have knowledge links going into lexicon l, and S_l denote the set of symbols that belong to lexicon l. The excitation level of a symbol t in lexicon l can be calculated as:

$$I(t) = \sum_{k \in F_l} \sum_{s \in S_k} I(s) \left[\ln\left(\frac{P(s \mid t)}{p_0} + B \right) \right], t \in S_l.$$

The function $I(s)$ is the excitation level of the source symbol s. Due to the "winner-takes-all" policy, the value of $I(s)$ is either "1" or "0". The parameter p_0 is the smallest meaningful value of $P(s_i \mid \boldsymbol{t}_j)$. The parameter B is a positive global constant called the *bandgap*. The purpose of introducing B in the function is to ensure that a symbol receiving N active knowledge links will always have a higher excitation level than a symbol receiving (N-1) active knowledge links, regardless of the strength of the knowledge links. For example, both symbols $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ belong to lexicon I. And both of them are activated by two symbols from other lexicons. However, $\boldsymbol{t}_1$ is activated by two symbols that belong to the same lexicon,

while t_2 is activated by two symbols that belong to different lexicons. We consider t_2 to be more highly activated than t1 and the bandgap B is introduced to increase the excitation level of t_2.

3. System Architecture

3.1 Overview of the ITRS

In this project, we developed the prototype of a context-aware Intelligence Text Recognition System (ITRS) that mimics the human information processing procedure. The ITRS system learns word-level and sentence-level knowledge from extensive reading. It recognizes partially-obscured characters or fills in completely occluded characters, based on the word- and sentence-level context and the high-level knowledge that has been obtained during training.

The input of the system is the text image. The first layer is character recognition software based on BSB models. It tries to recall the input image with a stored image of the English alphabet. In this work, a race model is adopted. The model assumes that the convergence speed of the BSB indicates the similarity between patterns. For a given input image, we consider all patterns that converge within 50 iterations as potential candidates that may match the input image. All potential candidates will be reported as the BSB results. Using the racing model, if there is noise in the image or the image is partially damaged, multiple matching patterns will be found. For example, a horizontal scratch in the letter "T" could make it appear to be the letter "F". In this case we have ambiguous information.

The ambiguity can be removed by considering the word-level and sentence-level context, which is achieved in the second and third layer where word and sentence recognitions are performed using cogent confabulation models. The models fill in the missing characters in a word, and missing words in a sentence. The three layers work cooperatively. The BSB layer performs the word recognition and it sends the potential letter candidates to the word level confabulation. The word recognition layer forms possible word candidates based on those letter candidates and sends this information to the sentence recognition layer. There could be feedback paths that send the sentence level confabulation results back to word-level, or send word confabulation results back to character-level. We believe that the feedback information can speed-up the recognition process, and its implementation will be our future task.

Figure 3 shows an example of using the ITRS to read texts that have been occluded. The BSB algorithm recognizes text images with its best-effort. The word-level confabulation provides all possible words that associate with the recognized characters, while the sentence-level confabulation finds the combination among those words that gives the most meaningful sentence.

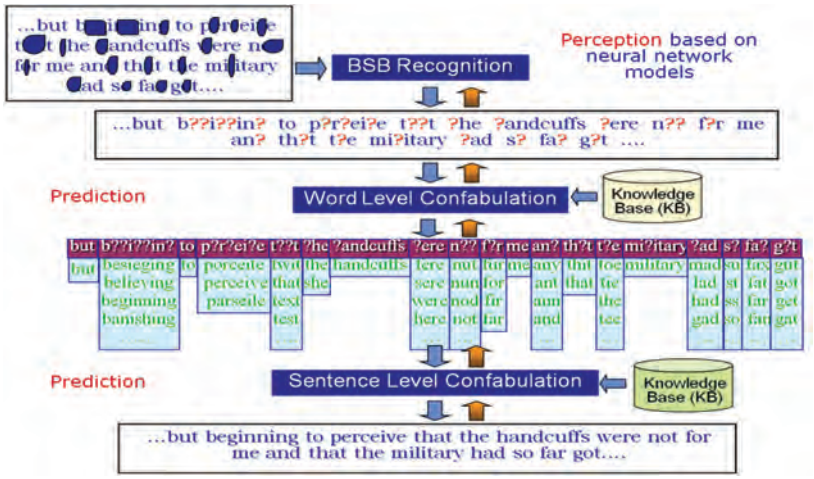


Figure 3. Overall architecture of the models and algorithmic flow

3.2 Front-End Image Processing

At the top level of the ITRS is the image processing function that reads in the image of the text, and separates it into blocks of smaller images that contain only one character. The image processing function distinguishes punctuations from texts and uses them to separate sentences. It also separates words based on white spaces. The output of the image processing function is a set of character images labeled by a triplet (i, j, k), where k is the position of the character in a word, j is the position of this word in a sentence, and i is the index of the sentence.

The flow of page image processing is given in Figure 4. The page image is initially in JPG format. It is converted into BMP format. The page is first divided horizontally into different lines and each line is then vertically divided into characters. The bit-map of each character will be stretched or scaled into 15×15 pixels. Finally the grey-scale value of each pixel will be discretized into binary, which is the required input format for the BSB model.

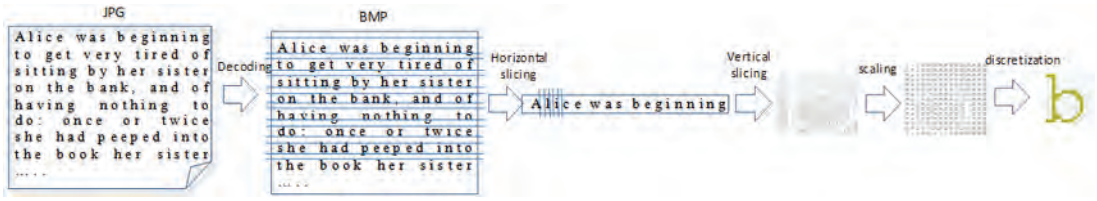


Figure 4. Processing flow of page image processing

3.3 Character-Level Image Perception

The output of the image processing software is the input of the BSB-based character recognition model. The Brain-State-in-a-Box model is a simple, non-linear, auto-associative neural network. Human memory is said to be associative; that is, it is capable of retrieving a piece of data upon presentation of only partial information from that piece of data. Given a vague, partially formed idea or input, an associative memory will compare it against all other stored content until a match is found. In this context, a match refers to the resolution, completion, or connection of the full-version, result, or 'answer', based upon the partial input located within memory. The BSB algorithm mimics this auto-associative behavior by storing the prototype patterns as vectors in the neural network, and recalling this vector when a 'noisy' or incomplete version of the pattern is presented to the system.

As mentioned in the system overview, BSB is used for character recognition within the ITRS. Characters in the system are represented by 15×15 pixel patterns. The system is trained with a character set, and when a letter image is presented to the BSB algorithm, it is compared against all models in the system. This comparison is called the ‹recall› stage. The ‹winning› candidate characters are those that converge, or match, the closest to the images trained in the system. More than one character can be sent to the word-level confabulation algorithm as a candidate, if multiple letters have the same degree of similarity to the input pattern. For severely damaged characters, all letters in the alphabet can be considered candidates.

3.4 Confabulation-Based Word/Sentence Level Prediction

The inputs of word confabulation are characters with ambiguities referred as candidates. For each input image, one or multiple character-level candidates will be generated by the BSB model. In this work, we assume that each word has less than 20 characters. Any word that is longer than this will be truncated. Currently, if a word has less than 20 characters, it will be padded with white spaces. In the future, the length of the word will be considered as an input to the confabulation model to avoid this type of padding and to speed-up the process.

The word-level confabulation model consists of three levels of lexicon units (LUs). There are 20 LUs in the first level, and the ith LU in the first level represents the ith character in the word. There are 19 LUs in the second level, and the ith LU in the second level represents a pair of adjacent characters at location i and i+1. Finally, there are 18 LUs in the third level, and the ith LU in the third level represents a pair of characters located at i and i+2.

A *knowledge link* (KL) from lexicon I to II is an $M \times N$ matrix, where M and N are the cardinalities of symbol sets S_I and S_{II}. The ijth entry of the knowledge link gives the conditional probability $P(i|j)$, where $i \in S_I$, and $j \in S_{II}$. Symbols i and j are referred to as *source symbol* and *target symbol*. Between any two LUs, there is a *knowledge link* (*KL*). If we represent the lexicons as vertices and represent the knowledge link from lexicon I to lexicon II as a directed edge from vertex I to vertex II, then we will obtain a complete graph.

Confabulation-based word-level and sentence-level prediction heavily relies on the quality of the *knowledge base* (*KB*). The training of the KB is the procedure to construct the probability matrix between source symbols and target symbols. The training program first scans through the training corpus and counts the number of co-occurrences of symbols in different lexicons. Then, for each symbol pair, it calculates their posterior probability.

The word-level recall algorithm finds all words from possible combinations of input character candidates. For example, if the input candidates of a 3-letter word are: (w t s r p o k e c a) for the first letter, (h) for the second letter, and (y t s r o m i h e a) for the third letter, then the word-level confabulation program will find 24 words, including "why", "who", "wha", "thy", "thi", "the", "tha", "shy", "sho", "she", "rho", "phr", "ohs", "oho", "ohm", "kho", "eht", "cha", "aht", "ahs", "ahr",

"ahm", "ahh", and "aha". Although some of them are not dictionary words, they appear at least once in the training corpus, which consists of more than 70 fiction books. Some of these non-dictionary words are names for special places or objects, and some of them are used to represent specific sounds. A few of them are typos or errors in the training file.

For each input candidate in each lexicon, the recall algorithm sets the corresponding symbols to be active. A lexicon that has multiple symbols activated is referred to as an *ambiguious lexicon* and the goal of the word level confabulation is to eliminate such character level ambiguity as much as possible or to transform it into word level ambiguity which can be further eliminated by sentence level confabulation.

For each lexicon that has multiple symbols activated, we calculate the *excitation level* of each activated symbol. The excitation level of a symbol *i* in lexicon *B* is defined as:

$$EL_B[i] = \sum_{A \neq B} \sum_{j \in \{active\ symbols\ in\ A\}} kl_{AB}[j][i]$$

where $kl_{AB}[j][i]$ is the knowledge value from symbol *j* in lexicon *A* to symbol *i* in lexicon *B*. The *N* highest excited symbols in this lexicon are kept active. These symbols will further excite the symbols in other ambiguous lexicons. This procedure will continue until the activated symbols in all lexicons do not change anymore. If convergence cannot be reached after a given number of iterations, then we will force the procedure to stop.

For each word in a test sentence, the word-level confabulation model generates one or multiple word candidates. They will be the input to the sentence level confabulation model.

The sentence-level confabulation model is very similar to its word-level counterpart, except that there are only two levels of LUs. The first level LUs represent single words, while the second level LUs represent adjacent word pairs. The training and recall functions of sentence-level confabulation have the same principle as these functions at word-level. It is important to point out that, each first-level lexicon for word confabulation contains at most 26 symbols representing 26 letters in the alphabet, and each second- and third-level lexicon for word confabulation contains at most 26×26=676 symbols; however, the maximum size of the symbols of each sentence-level lexicon equals to the total number of English words and word pairs. Without any compression, the sentence-level knowledge base will be extremely large. For example, the English version of the book "Round the Moon" has about 47×10^3 words. Our analysis shows that it has 23×10^3 distinguished symbols (i.e., words and word pairs). As we mentioned earlier, each knowledge link is an $M\times N$ matrix, where *M* and *N* are the symbol size of the source and target lexicons. Without any compression, the trained knowledge base will have 2.3×10^9 entries which are equivalent to be 9.2 GBytes. Fortunately, the knowledge links are sparse matrices. Only less than 0.1% of the matrix has non-zero values. Therefore, an option to reduce the memory cost is to store the knowledge using the *list-of-lists* (LIL) format, which has been widely used for sparse matrix storage. However, this leads to the second problem. As the size of the training corpus grows, the number of symbols of each lexicon can easily go up to hundreds of thousands. Even with the best search algorithm, the time to locate the entry in the compressed matrix grows logarithmically, and soon the algorithm will become prohibitively slow. In this work, two-level hash functions are used to speed-up the training and recall of the sentence-level confabulation model. It provides 10 to 15X speed-ups to locate a knowledge entry in a 6GB compressed knowledge base. More details of sentence-level confabulation can be found in our recent work[9].

4. Experimental Results

We implemented the ITRS on the 12,000 core heterogeneous high performance computing (HPC) cluster. The cluster consists of 78 sub-clusters, and each sub-cluster is composed of two Intel Xeon Hex-core processors as the server node, and 22 Sony PlayStation3 (PS3) computers based on the IBM Cell Broadband Engine processor. The structure of the heterogeneous HPC cluster is given in Figure 5(a). Each cell processor has one PowerPC processor and 6 available synergistic processing elements (SPE). Its block diagram is given in Figure 5(b). Each SPE processor is a self-contained vector processor that runs 8 floating-point operations at 3.2GHz. With 6 of these SPEs, a cell processor provides 153GFlops (Giga Floating-Point Operations Per Second) performance. Each PS3 only has 256MB memory, which is relatively small, compared to PCs and the server nodes. Overall, the 1,700 cell processors deliver more than 260 TFLOPs (Tera Floating-Point Operations Per Second) computing power, and form the first-layer hardware of a cognitive computing system. The Intel Xeon processor-based server nodes naturally form the second-layer. Each server node has 12 cores and 24GB memory. The memory access speed could reach as high as 2GB/s per core.

In this project, we map the BSB computation on the cell processors, because their vector processing capability provides great acceleration to the matrix vector multiplication of the BSB model. As we mentioned before, the cell processor sustains about 100 GFLOPS (i.e., 70% of the peak performance) while computing the BSB model. With the overhead for communication and handshaking between the BSB and confabulation models, each cell processor still sustains about

75 GFLOPS (i.e., 50% of the peak performance.) We map the word and sentence confabulation on the Xeon Hexa-core processors because they provide enough memory space for the storage of the knowledge base required by the confabulation model.

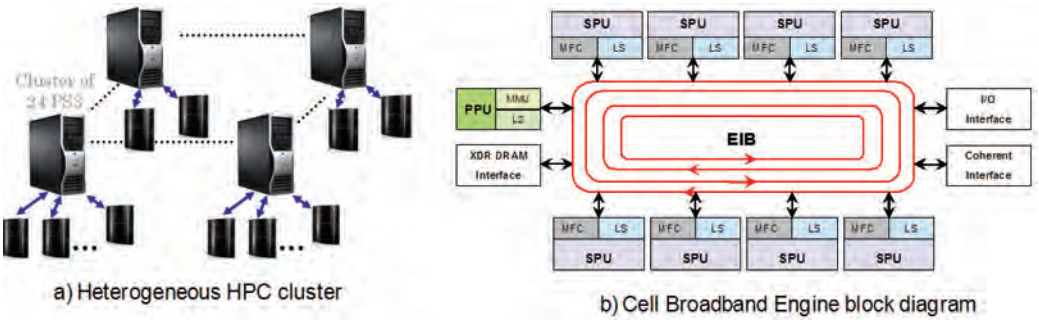


Figure 5. Hardware platform architecture

We evaluate the ITRS system for its performance and accuracy. Its word-level knowledge is obtained by "reading" an English dictionary, and its sentence-level knowledge base obtained by "reading" more than 70 classic literature texts.

In the first experiment, we test the ITRS system using text images with low-level noise. Our test case is extracted from the book "Great Expectations" by Charles Dickens. The text consists of 96,767 letters or 23,912 words. The text has not been read during the training process. In order to explicitly control the noise in the input, we use generated bit-maps of text images instead of scanned text images. Horizontal scratches are added to the images of letters selected randomly. The amount of noise in the input is controlled by two parameters: 1) the thickness of horizontal scratches varies from 1 pixel wide to 3 pixels wide, and 2) the probability that a character is scratched varies from 0.2, 0.4 to 0.6.

Figure 6 shows the examples of the three different types of horizontal scratches. Note that the scratches are located in the center of the text image, where most of the information to distinguish amongst various characters is found. Also note that each text image is 15×15 pixels large, 1–3 pixel scratches across the image is equivalent to 10–30% missing information.

Figure 6. Three different horizontal scratches

The outputs of ITRS are compared against the original text. A sentence (or a word) is considered in-accurately recognized if there is one word (or one letter) mismatch from the original text. Table 1 gives the accuracy of word and sentence confabulation. They are calculated as the number of sentences (words) that have been correctly "read", divided by the number of sentence (word) confabulations that have been invoked. The same table also gives the percentage of correct words. It is calculated as the total number of correct words (including both confabulated and non-confabulated), divided by the total number of words in the text. As we can see, the rate of accurate word confabulation and the overall percentage of correct words are very close to each other. This is because the majority of words have at least one scratch. Therefore, they all need to go through the word confabulation process.

Table 1. Recall accuracy at sentence- and word-level

Scratch prob.	Sentence confabulation accuracy			Word confabulation accuracy			Overall % correct words		
	1 scratch	2 scratch	3 scratch	1 scratch	2 scratch	3 scratch	1 scratch	2 scratch	3 scratch
0.2	0.92	0.90	0.86	0.98	0.98	0.97	0.99	0.99	0.98
0.4	0.87	0.82	0.76	0.98	0.97	0.95	0.98	0.98	0.96
0.6	0.82	0.74	0. 65	0.97	0.95	0.93	0.98	0.97	0.94

Figure 7 shows the rate of correct sentence and correct words found by the ITRS when a well-trained sentence-level knowledge base (i.e., "long KB") is used, and a poorly-trained sentence-level knowledge base (i.e., "short KB") is used. The size of the high-quality knowledge base ("long KB") is more than 6GB, while the size of the low-quality knowledge

base ("short KB") is only 2.7MB. The data series of "% improve" gives the percentage improvement of the results obtained using "long KB" over the results obtained using "short KB". These charts indicate that better knowledge at sentence-level improves the sentence accuracy up to 80%, and word accuracy up to 8%.

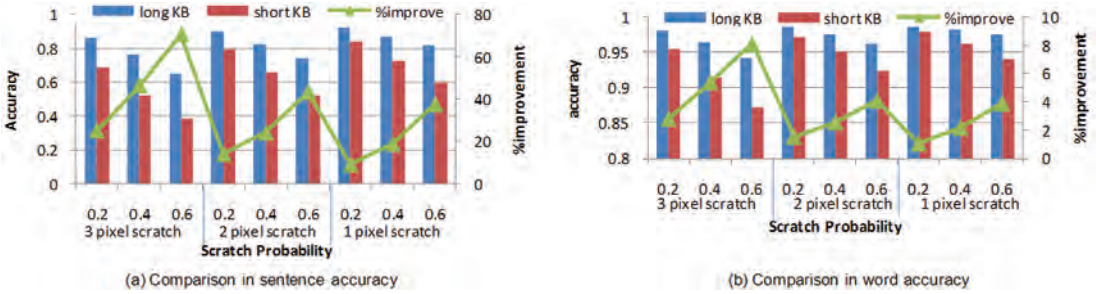


Figure 7. Impact of sentence knowledge on the confabulation accuracy

We define an *ambiguous candidate* at word- (or character-) level as one of the multiple candidates that associate to the same word (or character). Figure 8 shows the relationship between the word and sentence accuracy versus the amount of ambiguity in the input. The amount of ambiguity is measured using four different parameters: 1) "Cand./L" gives the average number of ambiguous character candidates for each character in the input text. Note that if a character associates to only one candidate (i.e., it has no ambiguity), then it contributes only to the denominator, not to the numerator in "Cand./L". Therefore, the value of "Cand./L" can be less than 1 for some clean inputs; 2) "Cand./XL" gives the average number of ambiguous character candidates for each scratched character in the input text; 3) "Cand./W" gives the average number of ambiguous word candidates for each word in the input text; and 4) "Cand./XW" gives the average number of ambiguous word candidates for each word that has scratched letters. The results show that the text recognition accuracy is a linear function with Cand./L and Cand./W. However, it has low-correlation with Cand./XL and Cand./XW. Note that Cand./XL and Cand./XW represent the ratio of the amount of ambiguous information versus the amount of damaged information (i.e., the average severity of damage), while Cand./L and Cand./W represent the ratio of the amount of ambiguous information versus the total amount of available information. This result indicates that the accuracy of ITRS is not determined simply by how many letters/words are scratched out and how severely each letter/word is scratched. It is determined by the ratio of the amount of ambiguous information versus the total available information. In other words, the ITRS performs letter/word recognition not only by looking at the ambiguous information itself, but also by considering the context of the ambiguous information.

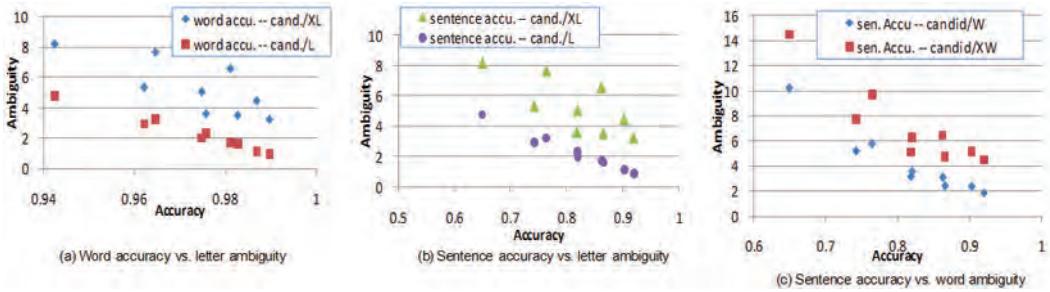


Figure 8. Word/sentence-level accuracy versus the ambiguity

In the second experiment, we test the ITRS system using text images with completely occluded characters. The two test files are extractions from the book "Great Expectations" by Charles Dickens and the book "Lost World" by Arthur Conan Doyle. Neither of these books had been read during the training. We increase the percentage of occluded letters from 10% to 30%. Table 2 gives the sentence-level and word-level accuracy for different input files. As we can see, even with 30% of the characters missing, the ITRS can recognize more than 85% words correctly.

Table 2. Sentence and word accuracy for input with occluded characters

Percentage of occluded letters	10%		20%		30%	
	Word accu.	Sentence accu.	Word accu.	Sentence accu.	Word accu.	Sentence accu.
Great Exp.	95.5%	60.2%	90.5%	33.6%	86.3%	23.9%
Lost World	97.0%	63.8%	92.7%	31.2%	86.4%	20.7%

The following two examples show the input text (with occluded letters) and the recognized text from ITRS:

Input: bu[redacted] [redacted] [redacted]new the s[redacted]unds by this t[redacted]me and co[redacted]ld [redacted]is[redacted]ociate th[redacted]m from the object o[redacted] [redacted]rs[redacted]it
Recognized sentence: but I knew the sounds by this time and could dissociate them from the object of pursuit
Input: gra[redacted]ious [redacted]o[redacted]dne[redacted]s [redacted]r[redacted]c[redacted]o[redacted]s me what[redacted] [redacted]one wit[redacted] th[redacted] pi[redacted]
Recognized sentence: gracious goodness gracious me what's gone with the pig
Correct sentence: gracious goodness gracious me what's gone with the pie

5. Conclusion

In this paper, we present an intelligent text recognition system which is based on two neuromorphic computing models, i.e., BSB and cogent confabulation. The BSB model performs the character image recognition, while the confabulation models perform the context-aware prediction based on word and sentence knowledge bases. The results show that the ITRS system can achieve more than 90% accuracy at sentence-level, and more than 98% accuracy at word-level.

Acknowledgements

The professor's work is supported by the US Air Force Research Laboratory, under contract number FA8750-09-2-0155. The DoD HPC Modernization Program supported this work under project AFIFR6632663 and by supplying time on the Condor supercomputer at the HPC Affiliated Resource Center at AFRL/RIT in Rome, NY.

References

1. Mantas, J., "An Overview of Character Recognition Methodologies", *Pattern Recognition*, 19(6), pp. 425–430, 1986.

2. George Nagy, "Optical Character Recognition – Theory and Practice", *Handbook of Statistics, Vol. 2*, Elsevier North-Holland, pp. 621–649, 1982.

3. *Tesseract-OCR*, http://code.google.com/p/tesseract-ocr/.

4. Schultz, A., "Collective recall via the Brain-State-in-a-Box network", *IEEE Transactions on Neural Networks*, Vol. 4, No. 4, pp. 580–587, July 1993.

5. Anderson, J.A., J.W. Silverstein, S.A. Ritz, and R.S. Jones, "Distinctive features, categorical perception, probability learning: Some applications of a neural model," *Neurocomputing; Foundations of Research,* J.A. Anderson and E. Rosenfeld (eds.), The MIT Press, Cambridge, MA, Chap. 22, pp. 283–325, 1989; reprint from *Psychological Review*, Vol. 84, pp. 413–451, 1977.

6. *Associative Neural Memories: Theory and Implementation*, Mohamad H. Hassoun, Editor, Oxford University Press, 1993.

7. Hecht-Nielsen, R., Confabulation Theory: The Mechanism of Thought, Springer, Aug. 2007.

8. Wu, Q., P. Mukre, R. Linderman, T. Renz, D. Burns, M. Moore, and Qinru Qiu, "Performance Optimization for Pattern Recognition using Associative Neural Memory," *Proc. of 2008 IEEE International Conference on Multimedia & Expo*, June 2008.

9. Qiu, Q., Q. Wu, D. Burns, M. Moore, M. Bishop, R. Pino, and R. Linderman, "Confabulation-Based Sentence Completion for Machine Reading", *Proc. of IEEE Symposium Series on Computational Intelligence*, April 2011 (to appear).

Context-Aware Parallel Pseudorandom Number Generators for Large Parallel Computations

Rajendra V. Boppana
CS Dept. and Inst. for Cyber Security, University of Texas at San Antonio, TX
boppana@cs.utsa.edu

Abstract

Design and testing of parallel pseudorandom number generators (PRNGs) that generate millions of parallel random number (RN) streams needed for large parallel computations is a nontrivial task if: a) the number of parallel streams are not fixed at the beginning of the program execution, and they are to be generated in a distributed manner with low communication overhead; and b) the correlations among the parallel streams must be very low. Furthermore, the current PRNGs require the user to manage the number of streams and their initialization, which can be onerous if each process or thread of a parallel application consumes RNs at multiple locations and, for better randomization, distinct RN streams must be used in each instance. In this paper, both problems are addressed using context-aware PRNGs. In this approach, each request for an RN stream by a process/thread results in the allocation of a large set of RN streams, so that each distinct program statement that calls for RN generation (denoted, RN context) is served with a distinct RN stream taken from the RN streams assigned to that process. A prototype context-aware parallel random number generators (CPRNGs) based on the multiplicative lagged Fibonacci generator is implemented for automatic RN stream generation based on RN contexts. A new parallel statistical test, called the inter-stream correlation (ISC) test, is designed and implemented to assess the degree of independence among a large number of parallel RN streams and provide an easy-to-use quality metric. Preliminary results indicate that the prototype CPRNG provides high-quality RN streams, and that the ISC test promises to be a highly-effective test to assess correlations among a large number of RN streams.

1. Introduction

Many scientific computing applications, business and finance applications, and complex systems modeling and analysis techniques use random number generators[1] (RNGs) extensively for simulations of various likely scenarios and estimations of potential outcomes. Often, these applications are highly-scalable and can take advantage of the availability of thousands of computing cores on heterogeneous systems comprising multi-core processors (CPUs) and highly-parallel general-purpose graphics processing units (GPGPUs), provided that suitable parallel random number generators (PRNGs) are available to simultaneously feed thousands of computing streams with high-quality random number (RN) streams with low intra- and inter-stream correlations.

We present context-aware parallel random number generators (CPRNGs) based on a new approach to allocate and manage RN streams by parameterized random number generators that can generate virtually unlimited numbers of distinct RN Streams. Lagged Fibonacci generators (LFGs), which generate a new RN by applying an arithmetic or logic operation on two or more previously generated RNs, can provide a large number of distinct RN streams, with each stream having a large cycle—the number of RNs that can be used before the sequence repeats. A prototype CPRNG, based on multiplicative lagged Fibonacci generator (MLFG), is implemented and evaluated. CPRNG provides two new features that a basic MLFG does not provide.

- CPRNG uses the program context, in which a request for a random number is made, to automatically select and use distinct RN Streams for distinct contexts.
- A typical PRNG requires the application to declare the maximum number of RN streams used in an execution run, and the number of distinct RN streams requested to be within this limit. However, this can be a significant constraint

[1]The random number generators we consider this paper are pseudorandom number generators. For easier description, however, we simply refer to them as random number generators.

for applications that may spawn additional processes during the execution, based on the intermediate results and use unpredictable number of RN streams.

- CPRNG relaxes this constraint and allows applications to request virtually unlimited number of RN streams beyond any initially-specified RN stream limit.

Another problem addressed in this work is the evaluation of intra-stream and inter-stream correlations—i.e., the quality of the random numbers generated. Several excellent statistical tests[1,11] are available to test intra-stream correlations of a sequential RNG. Test packages such as Diehearder[16] and TestU01[15] run a battery of such tests on an RN stream and provide pass/fail results from each test. If an RN stream fails any of the tests, then additional, more detailed tests are conducted. Otherwise, it is assumed that the RN stream is free of intra-stream correlations with very high probability.

To assess the quality of a PRNG, several parallel RN streams generated by it are interleaved using the perfect shuffle pattern to create a single RN stream, and single-stream test batteries are used to evaluate the inter-stream correlations among the RN streams[7,8]. A single-stream test package contains 20 different types of basic tests (which may be repeated with different parameters to create up to 150 test instances) and gives pass/fail status for each test applied to the interleaved stream. This provides a vector of pass/fail information that will be hard to use for comparisons of different PRNGs. Furthermore, these tests use a few billions of RNs from the test stream for these tests. Therefore, they are not suitable to test a large number of parallel RN streams; if a billion streams are interleaved to form a single stream, then these tests only examine a few numbers from each stream, which may not be enough to assess the inter-stream correlations. On the other hand, if a billion RN streams are partitioned into several subsets with each subset consisting of a small number of RN streams, and single-stream test batteries are applied on each set, then these tests will take several 100's of hours on a desktop machine and provide multiple vectors of pass/fail information that will be hard to combine into an easy-to-understand quality metric.

Another approach is to use thoroughly analyzed applications to test inter-stream and long-range correlations of RNGs. For example, a physics application involving simulations of two-dimensional (2D) Ising square lattice models with periodic boundary conditions, for which the exact solutions are known, is often used to evaluate PRNGs[3,4,7]. However, application-based tests are computationally-expensive and may not be adaptable to test billions of parallel streams at a time. Therefore, faster and more informative statistical tests of parallel RN streams are needed. Currently, there are very few parallel statistical tests that do not require serialization of RN streams and have the potential to evaluate inter-stream correlations.

We present a new inter-stream correlation (ISC) test that evaluates a large number of parallel RN streams simultaneously, and provides an easy-to-use quality metric. The ISC test divides the total streams to be evaluated into subsets of streams, and computes a correlation coefficient for each subset. These correlation coefficients are aggregated using a theoretically-sound test method such as the Donner and Rosner test (DR test)[13] or Kolmogorov-Smirnov test (KS test)[14] and a test statistic is obtained. If the test statistic is too high compared to a suitably determined critical value, the claim of independent RN streams is rejected. Lack of rejection indicates that the RN streams are likely to be independent.

We present preliminary results of the implementation of a prototype CPRNG and the application of ISC test. Timing tests show that CPRNG is nearly as fast as a basic PRNG, such as MLFG, and incurs only a small amount of overhead to provide the context-awareness. The ISC test found significant correlations in the RN streams generated by multiplicative Fibonacci lagged generator (MLFG), in the widely-used SPRNG package.

The rest of the paper is organized as follows. Section 2 presents the basics of parallel random number generators. Section 3 presents the context-aware PRNGs, and compares a prototype PRNG with the widely-used MLFG in the SPRNG package. Section 4 presents the ISC test and its application to CPRNG and MLFG, with up to 1.5 billion streams analyzed. Section 5 describes the related work in PRNGs and test methods. Section 6 concludes the paper.

2. Background

We are interested in parameterized RNGs that have the capability to generate a large number of RN streams with relatively simple changes to the initialization. Lagged Fibonacci generators[8,9,10] are easy to parameterize and, with careful selection of the parameters, can be used to generate virtually unlimited number high-quality distinct RN streams easily. In particular, we are interested in the multiplicative Fibonacci lagged generator (MLFG), which uses the recurrence relation:

$$x_n = x_{n-k} \times x_{n-l} \pmod{2^m}, \quad 0 \le k < l < n, \tag{1}$$

where l and k are the lags (or indices to the older random numbers used to generate the new random number), and x_i 's are positive and odd m-bit integers. This generator produces $2^{(m-3)*(l-1)}$ different RN streams, each with a cycle length of $2^{(m-3)} \cdot (2^l-1)$. Therefore, there are $(m-3) \cdot (l-1)$ bits that need be determined uniquely for each RN stream. (One of the initial lag

words and the least significant bits of all initial lag words are specified by the canonical form and parameters specified, and are common to all RN streams with those parameters[10].)

For a 64-bit MFLG with lag 17, there are $2^{61*16}=2^{976}$ different RN streams, each with a distinct 976-bit seed value and a cycle length of $2^{61} \cdot (2^{17}-1) \approx 2^{78}$. In practice, the upper or the middle b, $b<64$, bits of x_i's are extracted and supplied as the RNs to improve the randomness, since the lower bits are often less random owing to the arithmetic operation involved. We used the SPRNG package[8] and the MLFG available from its library, to implement a prototype CPRNG.

Additive lagged Fibonacci generators (ALFGs) are obtained by replacing the multiply operation in Equation 1 with an add operation; x_i's are positive m-bit integers with at least one odd integer in the first l lags. Compared to MLFGs, ALFGs provide more distinct RN streams with longer cycles for the same bit-size. However, the intra-stream and inter-stream correlations are more significant in ALFGs. To mitigate these issues, larger lags, l=1,279 or larger, are used. SPRNG package combines two ALFGs with different lag words to provide a higher-quality ALFG. On the other hand, MLFG can be used with smaller lags, e.g., l=17. Therefore, for the most commonly used configuration in SPRNG package, ALFGs take twice as much time to initialize a new RN stream and to generate RNs as MLFG.

SPRNG library package provides init_rng() and get_rn_dbl() function calls to initialize a new RN stream and to obtain the next RN in an already initialized stream, respectively. The init_rng function is called by specifying the seed, parameter sets that specify the lags and the locations of the odd-numbered words in the initial set of lag words, maximum number of RN streams (denoted max_str) that will be requested by the application, and cur_str, the RN stream number in the range [0, max_str) that needs to be initialized. The seed, parameter set, and max_str must be common in all init_rng() calls. For most parallel applications, it is easy to allocate the RN streams to processes based on the input data and/or computations allocated to them. For example, if a computational loop is partitioned cyclically among p processes, then iteration i is executed by process $i\%p$; in this case it is natural to allocate RN streams from the set $i, i+p, \ldots$ to process i.

Each call to init_rng() results in the initialization of the RN stream specified by the stream number, cur_str, and the calling code is given a pointer to the RN stream that should be used as argument in the function call get_rn_dbl() to obtain the next RN in the stream. (SPRNG library provides several other function calls including requests for integer RNs instead of reals, but they are not of interest in this paper.)

3. Context-aware Parallel Random Number Generators

If a process uses RNs in multiple locations for different purposes, then it is generally recommended that a distinct RN stream be used for each such context. However, with the current RNG packages, this requires the application to explicitly initialize the additional RN streams needed and, more importantly, use the appropriate RN stream pointer in each context.

This puts a significant burden on the application developer to manage the RN streams and contexts. Any changes to the code that change the number of contexts require additional work by the application developer to make suitable changes to the RN stream management. While it is natural and intuitive to partition RN streams based on the partitioning of input data or computations, explicitly managing multiple RN streams based on program contexts makes the application less portable and distracts the application developer.

To address these concerns and to improve the quality of RNs used by applications, we developed the CPRNG. The design methodology is to take a parameterized random number generator that has the capability to generate a large number of RN streams with relatively simple changes to the initialization and augment it with a scalable and automatic initialization process. Our first CPRNG is based on the MLFG described in Section 2. We used the SPRNG package and the MLFG available from its library, to implement the prototype CPRNG. The design of CPRNG is elaborated below.

3.1 CPRNG Design

In CPRNG implementation, each init_rng() call allocates not just one RN stream but a set of distinct RN streams and returns a pointer, str_ptr, to the set; the streams in this set can be customized with program context without further calls to init_rng(). The RN-context, the context or the program location from which a RN number is requested, is used in addition to the stream-set pointer, str_ptr, to determine the specific RN stream to be used. The RN context is derived from a combination of the program line number in the source code, the return address of the function call to get_rn_dbl(), the process/thread numbers, and any user supplied identifiers such as the iteration number. When the application requests for a random number using the function call get_rn_dbl(str_ptr), the RN-context will be used to determine the specific RN stream to be used in the set of streams pointed by str_ptr. The appropriate RN stream is automatically initialized with the RN-context, if it is the first call from this context, and a RN from the stream is returned.

Figure 1 describes the initialization process by CPRNG with lag parameters l and k, $0 < k < l - 2$. A call to init_rng() results initialization of $l - 3$ of the lag words[2] using a sequential RNG such as the recursive with carry (RWC) generator described in the Diehard package[2] seeded with the user-specified seed. These lag words are common to the initialization of all RN streams, regardless of the process number or RN-context. One of the remaining three lag words is filled with an ID that is guaranteed to be distinct for distinct cur_str numbers specified in init_rng(). The distinct ID word is common to the set of RN streams that are allocated in response to init_rng() call. The remaining two lag words are filled with the RN-context so that distinct RN-contexts result in distinct RN streams.

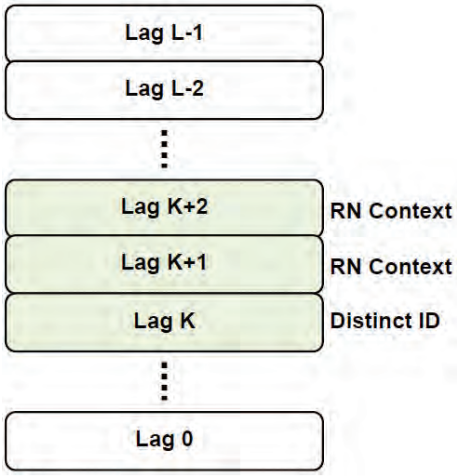


Figure 1. Initialization of RN stream lags by CPRNG. Each lag word is a 64-bit word with maximum lag L. L-3 of the lag words are filled randomly, based on the user-specified seed and a sequential RNG. These words are common to all RN streams used during the execution of the application. Lag K, K<L-2, is initialized with a unique and distinct ID that is associated with the cur_ str used in the init_rng() call. Lags K+1 and K+2 are initialized with RN-context to create a distinct RN stream fo reach distinct program context in each process.

For a CPRNG based on MLFG with maximum lag l=17 and 64-bit words, $2^{2\times61}=2^{122}$ distinct RN streams are allocated with each init_rng() call. Based on the context and str_ptr argument used in a call to get_rn_dbl(), an appropriate stream is selected, automatically initialized prior to first use, and the next RN in the stream is returned. CPRNG may be used without RN-contexts by choosing appropriate parameters to init_rng() call. If RN-contexts are not used, then the two lag words that are normally filled with RN-context are filled with the random bits generated by the sequential RWC generator; the lag word with distinct ID ensures that RN streams are distinct for distinct values of cur_str specified in the init_rng(). CPRNG will be simply a basic MLFG when used without context.

For applications that use a large and variable number of RN streams, having to specify the maximum number of streams used during an execution run is a limitation. Furthermore, certain large-scale parallel applications may spawn additional processes and threads dynamically depending on the input data and intermediate results. To accommodate such situations, CPRNG assigns 2^{10} distinct IDs for the lag word k upon a call to init_rng(), independent of any streams allocated to handle RN contexts. Typically, only one of these IDs is used by a process. However, if a process spawns threads or child processes and needs to use additional distinct RN streams without going through the initialization process, it can have them without any communication overhead by using the original initialization with the distinct ID word replaced with one of the unused IDs from its allocated IDs. This leads to faster initialization of the new RN streams on demand. If more RN streams are needed and init_rng() is called with cur_str value greater than max_str, a monotonically increasing counter is used to ensure that the lag word K is distinct. However, the access to this counter needs to be serialized by using appropriate mutex locks in threaded applications or by assigning it to a process to serve the counter-values to the other processes of the application. Only in these instances, an additional communication or serialization overhead is incurred by CPRNG, compared to the static methods used in the current packages such as SPRNG. On the other hand, CPRNG provides a virtually unlimited

[2]The initialization of the lag words is more complicated than the simpler description given here. For MLFG, all the lag words must be odd values. The two consecutive 32-bit RNs generated by the RWC generator are used form a 61-bit integer and a least significant bit determined by the canonical form and parameter set is appended to it to form a 62-bit number, say, z. The actual lag word is formed by using the operation $(-1)^y 3^z \bmod 264$, where y is a randomly generated 1 or 0. However, for easier description, we omit these implementation details. See Reference 10 for the complete details.

number of RN streams on demand, and avoids depletion of the available RN streams that can occur with static partitioning of the available RN streams for applications with many levels of dynamic process/thread creation.

CPRNG is implemented in the SPRNG package as an additional PRNG. The implementation provides the same application interface as the other PRNGS in the package. Any parallel application currently designed to use SPRNG generators can use CPRNG by using an appropriate RNG code. No additional application modifications are needed. Just like the other PRNGs in the SPRNG package, CPRNG produces consistent and predictable RN streams for an application regardless of the number of processes/threads used for parallel computation. The CPRNG implementation is free of memory leaks, is multithread safe, and works seamlessly with MPI-based applications. The GPU version of CPRNG is implemented as a different generator with some restrictions on features: no context-awareness, and the maximum number of streams needed by the application must be specified at the beginning of the program execution.

3.2 Timing Tests

Two computers, a desktop computer with Intel Core i7-870 CPU and a rack server with Xeon E630 CPU, are used to estimate the time required for initialization of a random number (RN) stream by calling init_rng(),and the time taken to generate a single random number from an already initialized stream for CPRNG with lag 17 and three generators from the SPRNG package: MLFG—multiplicative lagged Fibonacci generator with lag 17, ALFG—lagged Fibonacci generator which is a combination of two additive Fibonacci generators with lag 1,279, and ALFG_17—lagged Fibonacci generator with lag 17.

We used Intel CPU time-stamp counter for the time-stamps. For RN stream initialization test, the time taken to initialize a single RN stream is subtracted from the time taken to initialize two RN streams. This is repeated 100 times and the average of the times is taken as a sample point. This experiment is repeated 10 times and the average of the 10 samples and the corresponding 95% confidence interval is calculated. For RN generation test, the time taken to generate 1,000 RNs from an already initialized stream is subtracted from the time taken to generate 1,000 RNs each from two previously initialized RN Streams. This time is divided by 1,000 to get the time taken to generate a single RN. This is repeated 100 times and the average is taken as a single sample point. This experiment is repeated 10 times and the average of the 10 sample points and the corresponding 95% confidence interval is calculated.

Figure 2 gives the times in clock-ticks—2.93 ticks/ns for the Core i7-870, and 2.53 ticks/ns for the Xeon E630 machines. The chart on the left gives the initialization time of an RN stream, while the chart on the right gives the time taken to get an RN from an initialized stream. The initialization costs of CPRNG are about the same as those of MLFG, on which CPRNG is based. The cost of generating an RN is about 3 ticks higher compared to MLFG owing to the additional processing needed for context-aware RN generation. This can be easily eliminated if the application does not require context-aware RN generation. ALFG has high initialization overhead since it uses two additive Fibonacci generators with a large amount of lag (the oldest RN used in calculating the next RN) to provide high-quality RN streams. To rule out any experimental error, we tested ALFG with lag 17 (which is not recommended), whose initialization cost is comparable to those of MLFG and CPRNG.

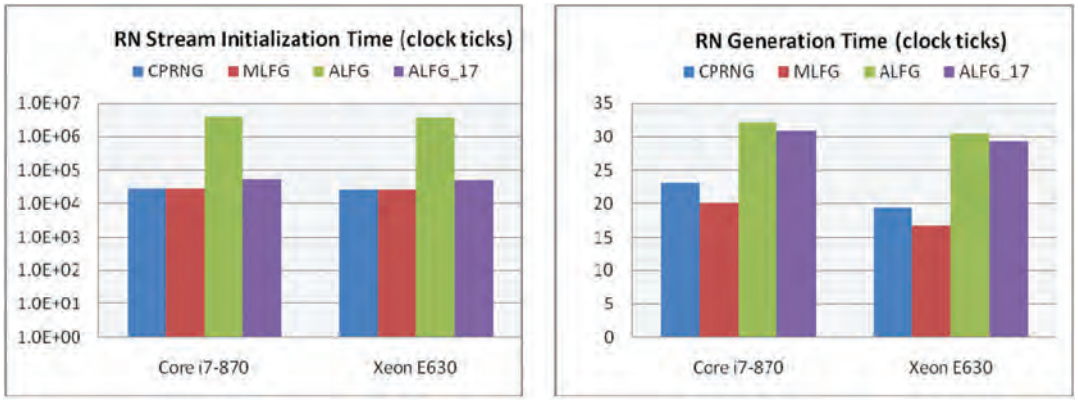


Figure 2. Time taken to initialize RN streams (left chart) and to generate RNs (right chart). A desktop computer with Intel Core i7-870 CPU and a rack server with Xeon E630 are used. The times are given in clock-ticks—2.93 ticks/ns for the Core i7-870, and 2.53 ticks/ns for the Xeon E630. The y-axis for the left chart is in log-scale. The 95% confidence intervals are ± 1% of the mean-values reported.

4. Inter-Stream Correlation Test

The inter-stream correlation (ISC) test evaluates the correlations among a large number of RN streams. The RN streams are divided into several subsets, and the streams in a subset are interleaved, using perfect shuffle or biased interleaving method. Consider three RN streams A, B and C with RNs $a_1, a_2, a_3,\ldots, b_1, b_2, b_3,\ldots, c_1, c_2, c_3,\ldots$, respectively. In perfect shuffle interleaving, a new stream $a_1, b_1, c_1, a_2, b_2, c_2, a_3,\ldots$ is created. In biased interleaving, $a_1, b_1, a_2, c_1, a_3, b_2, a_4,\ldots$ is created. The RNs in the odd-numbered positions form the X variates and the RNs in the even-numbered positions form the Y variates. These are transformed into normal bi-variates using Box-Muller transform[12]. Correlation coefficient, r, for the bi-variate pairs is computed. This is repeated several times to obtain multiple r's. Collectively, these r's are the samples that can be used to estimate ρ, the true common correlation coefficient among the parallel RN streams generated by the PRNG being evaluated.

The r's are aggregated using a theoretically-sound test method such as Donner and Rosner test (DR-test)[13] or Kolmogorov-Smirnov test (KS-test)[14] and a test statistic is obtained. The statistic for DR-test is denoted as t_H and the statistic for KS-test as D_{max}. For each test, there is a critical value that is computed based on the desired significance level and the number of r's used. For example, for DR-test at a significance level of 0.05, the critical value is 1.96 provided the number of bi-variate pairs used to calculate each r is large. If the test statistic is significantly above the critical value, then the RN streams generated by the PRNG are likely to have significant inter-stream correlations.

The DR-test combines the r's and gives the test statistic t_H, which is a standard normal variate. This can be used to test the null hypothesis $H_0 : \rho = 0$. Large absolute values of t_H will lead to the rejection of the null hypothesis and the acceptance of the alternative hypothesis H_1: $\rho \neq 0$. For the significance level α=0.05, absolute values of t_H above 1.96 lead to the rejection of the claim that parallel RN streams are independent; the probability that the rejection is erroneous is α=0.05. One could use different significance levels: for α=0.02, the absolute values of t_H above 2.33 will lead to rejection of the claim of independence of RN streams with only 0.02 probability of being wrong.

The distribution of r's is approximately normal. These r's can be converted into standard normal variates using sample variance of r's and the fact that we are testing for ρ=0. This enables us to apply the KS-test on the distribution of r's. In this test, the KS-test statistic, D_{max}, computed using the r's must be less than the critical value, $D_{a,n}$, for significance level α and n , the number of r's used. For KS-test, at a significance level of 0.01, the critical value is 0.0274 when the number of r's used is 1,500.

Figure 3 gives the results of the two tests for shuffle-interleaving of RN streams generated by CPRNG and MLFG. In our tests, we used 1,500 sets of random number streams with the set size varied from 10 to 1,000,000 to obtain $r_1, r_2,\ldots, r_{1,500}$. When the set size is 1,000,000, a million streams are used to obtain a single r, and a total of 1.5 billion streams are used to obtain the 1,500 r's used to calculate the test statistic. The dashed-lines indicate the critical values for the test statistics. Our results indicate that MLFG (the built-in random number generator in the SPRNG package) fails both DR- and KS-tests for test configurations that use 1.5 million or more streams. On the other hand, CPRNG, which is also based on the same theoretical foundation as that of MLFG, performs well; it narrowly fails the KS-test only for the largest test configurations we used.

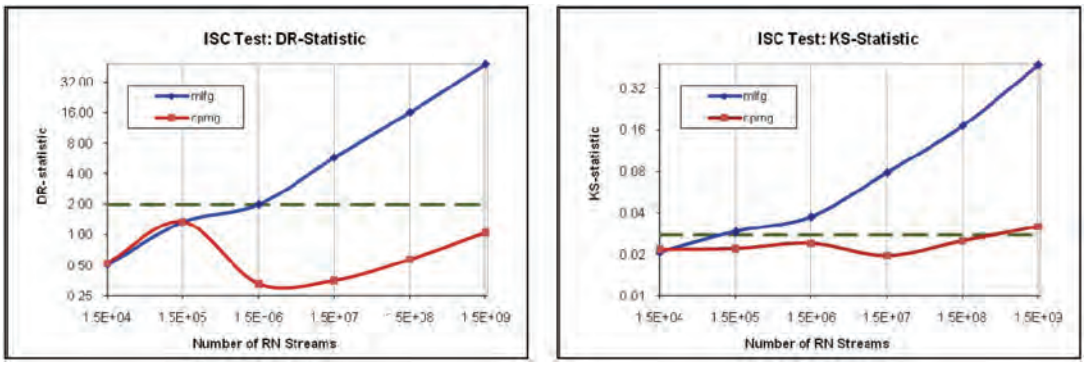


Figure 3. ISC tests for CPRNG and MLFG. The dashed-lines indicate the critical values. The y-axes for both charts are in log-scale. For both tests, MLFG's test statistic is significantly higher than the critical value, indicating that the RN streams generated by MLFG may have significant inter-stream correlations and must be tested further.

5. Related Work

The designs of sequential and parallel RNGs are extensively investigated by many researchers owing to their importance to computational science and to the elegant, mathematical nature of the problem. Knuth[1] discusses several RNGs, and many excellent single-stream test methods that are implemented in popular test batteries such as Dieharder[16] and TestU01[15]. Linear congruential generators that use a recursive equation of the form $x_n = a \cdot x_{n-1} + b \pmod{2^m}$, where a and b are constants, are commonly available as part of the C math library in a typical UNIX environment. One of 2^{19937} the most popular sequential RNGs is the Mersenne twister[5] which offers an RN stream with a cycle of length. A graphics processing unit (GPU) version of this RNG[6] is commonly used by applications designed to use GPUs.

Additive and multiplicative lagged Fibonacci generators[8–10] have been extensively investigated because of the ease with which they can be used to generate distinct RN streams. Of the two, MLFG is considered to be more robust, producing higher-quality RN streams. Both generators are implemented in the popular SPRNG package[8]. We have used the SPRNG package extensively. The prototype implementation of CPRNG is based on the MLFG implementation and supports SPRNG's application interface.

SPRNG also implements several sequential tests and provides a systematic way to interleave several streams into a single-stream and apply the sequential tests. However, owing to the availability of more exhaustive test packages, Dieharder and TestU01, we did not use the single-stream tests in SPRNG. Another important resource provided by SPRNG is the Ising model simulation codes based on Metropolis and Wolff algorithms. These applications are widely-used to evaluate sequential and parallel RNGs[3,4,7].

6. Conclusion

Context-aware parallel random number generators are based on a new approach to allocate and manage RN streams by parameterized random number generators that can generate virtually unlimited numbers of distinct RN Streams. Compared to the parallel random number generators in the current packages such as SPRNG, CPRNGs can automatically provide distinct RN streams depending on the program context to improve the quality of the RNs used. To achieve the same effect with the current PRNGs, the application needs to, explicitly, manage multiple streams and their usage based on the program context. Furthermore, CPRNGs support highly-complex parallel applications that require a large and variable number of RN streams by dynamically allocating RN streams beyond the maximum number of RN streams specified at the beginning of program execution. In contrast, the current PRNGs do not allow applications to request RN streams beyond the initially specified number of RN streams. Some implementations, e.g., SPRNG, handle this issue by partitioning the total RN streams using a binary partitioning scheme. For applications that have many levels of dynamic process/thread creation, this can result in depletion of RN streams available to dynamically-created processes/threads.

The inter-stream correlation test evaluates the correlations among a large number of RN streams. Using a well-known test method such as the Donner and Rosner test or the Kolmogorov-Smirnov test, it provides an aggregate PRNG quality metric. This test complements the existing sing-stream test batteries and application-based tests currently available. It is applied to evaluate inter-stream correlations among as many as 1.5 billion RN streams. The ISC test shows that the MLFG used in SPRNG has significant inter-stream correlations when 1.5 million or more streams are considered. In addition to providing an easy-to-use quality metric, the ISC test is fast and can be adapted to on-line testing, in which the actual RNs used by an application are fed to ISC test, and overall quality of the RNs used is provided at the end of the execution of the application.

In the future, we plan to revise the current implementation and release a CPRNG library package to the HPC community. We also plan to design and implement additional CPRNGs based on other RNGs. We will work with HPC practitioners in adapting new applications that use multi-scale simulation models to augment the current test methods.

Acknowledgments

This work was done as part of the STTR Phase I contract W911NF-11-C-0026 funded by the US Army Research Office (ARO) Program Manager, Dr. Joe Myers. The primary computer used for the development and testing was acquired with funds from National Science Foundation grant CNS-0551501. Additional computer resources were provided by UTSA Institute for Cyber Security. The contents of this paper do not necessarily represent the position or the policy of the Government and no official endorsement should be inferred. The author thanks Professors Ram Tripathi and Ravi Sandhu at UT San Antonio and Mr. Robert Keller and Mr. Hemant Trivedi at Silicon Informatics Inc. for their help and participation in this project. Prof. Tripathi identified the DR-test and helped the author implement the DR-test and the KS test. Mr. Trivedi helped with the implementation of CPRNG and the time-stamp counter used in the timing tests.

References

1. Knuth, K.E., *The Art of Computer Programming, Volume 2: Semi-Numerical Algorithms*, 3rd ed., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1997.

2. Marsaglia, G., *Diehard software package*, available at http://stat.fsu.edu/pub/diehard, 1995.

3. Ferrenberg, A.M., D. Landau, and Y. Wong, "Monte Carlo simulations: Hidden errors from 'good'random number generators", *Phys. Rev.*, 69, 23, pp. 3382–3384, 1992.

4. Coddington, P. "Tests of random number generators using Ising model simulations", *Int. J. of Mod. Phys.*, 7, 3, pp. 295–303, 1996.

5. Matsumoto, M. and T. Nishimura, "Mersenne twister: a 623-dimensionally equi-distributed uniform pseudo-random number generator", *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, v.8 n.1, pp. 3–30, Jan. 1998.

6. Podlozhnyuk, V., *Parallel Mersenne Twister, Version 1.0*, Nvidia, 2007.

7. Srinivasan, A., M. Mascagni, and D. Ceperley, "Testing parallel random number generators", *Parallel Computing*, vol. 29, pp. 69–94, 2003.

8. Mascagni, M. and A. Srinivasan, "Algorithm 806: SPRNG: a scalable library for pseudo-random number generation", *ACM Transactions on Mathematical Software (TOMS)*, 26(3), pp. 436–461, 2000.

9. Aluru, S., "Lagged Fibonacci Random Number Generators for Distributed Memory Parallel Computers", *Journal of Parallel and Distributed Computing*, 45(1), pp. 1–12, 1997.

10. Mascagni, M. and A. Srinivasan, "Parameterizing Parallel Multiplicative Lagged-Fibonacci Generators", *Parallel Computing*, vol. 30, pp. 899–916, 2004.

11. Marsaglia, G., "A current view of random number generators", *Computing Science and Statistics: Proceedings of the XVIth Symposium on the Interface*, pp. 3–10, 1985.

12. Box, G.E.P. and M. E. Muller, A note on the generation of random normal deviates", *The Annals of Mathematical Statistics*, 29(2), pp. 610–611, 1958.

13. Donner, A. and B. Rosner, "On inferences concerning a common correlation coefficient", *Applied Statistics*, 29(1), pp. 69–76, 1980.

14. Rao, P.V., *Statistical Research Methods in the Life Sciences*, Brooks/Cole, 1998.

15. L'Ecuyer, P. and R. Simard, "TestU01: A C library for empirical testing of random number generators", *ACM Trans. Math. Softw.*, 33, 4, August 2007.

16. Brown, R.G., D. Eddelbuettel, and D. Bauer, *Dieharder Test Package, v 3.29.0*, Duke University, online: http://www.phy.duke.edu/~rgb/General/dieharder.php, retrieved on Nov. 2010.

Probability of Collision using High Performance Computing and the Monte Carlo Method

Kevin P. Roe
Maui High Performance Computing Center (MHPCC), Kihei, HI
kevin.roe.ctr@mhpcc.hpc.mil

Chris Sabol
US Air Force Research Laboratory (AFRL), Kihei, HI
chris.sabol@maui.afmc.af.mil

Alan Segerman and Christopher Binz
US Naval Research Laboratory (NRL-DC), Washington, DC
{segerman@nrl.navy.mil, christopher.binz}@nrl.navy.mil

Abstract

The ability to estimate the probability of collision between two satellites is critical to space situational awareness. These calculations can be quite complex, essentially requiring the integration of a multi-dimensional probability distribution function over a complex overlap in multi-dimensional space. Current practice typically utilizes simplified analytical methods relying on assumptions that are not appropriate for all situations. Although these analytical methods can provide very reasonable estimates for many situations, they do not handle all situations well; for example, variations of up to 30% are found when nonlinear dynamics dominate. Analytical methods have been used because of their lower computational requirements as compared to more accurate brute-force numerical methods. The High Performance Computing Software Applications Institute for Space Situational Awareness (HSAI-SSA) is able to use high performance computing to make these brute-force methods viable. This paper explores the benefits of using a Monte Carlo method in conjunction with high-accuracy special perturbation propagation methods to estimate the probability of collision between two orbiting objects, as well as the necessary high performance computing techniques necessary to handle the high computational requirement of this calculation.

1. Introduction

The United States relies on space for both defense and civil purposes more than any other nation. Maintaining free access to space is of critical importance for safe operations of satellites (weather monitoring, television, GPS, Earth observation, etc.). As the number of orbiting space objects has grown over the past 50 years, the risk of inadvertent collisions between satellites has also grown. Since the unanticipated collision of an active US Iridium communications satellite with an inactive Russian satellite in February 2009, there has been heightened awareness of these risks and the potential impacts they have on our ability to maintain free access to space and safe satellite operations.

The ability to accurately predict the probability of collision between any two satellites is one of the most important space situational awareness products for two reasons. First, for maneuverable satellites, it provides the actionable information required on whether a collision avoidance maneuver is warranted. Second, for all cases, it allows for the posturing of space surveillance resources to observe and catalog potential debris, which is critical since debris pieces can pose a collision risk to other satellites.

Accurately calculating the probability of collision isn't straightforward, and requires the integration of a multi-dimensional probability distribution function over an overlap in multi-dimensional space. In practice, simplifying assumptions are made allowing for analytical estimates of this calculation. These assumptions may not always be appropriate, and thus rigorous verification and validation is required. This is where numerical methods can be utilized.

Monte Carlo methods can be used to generate probability of collision distribution functions that can be compared to analytic methods. Given two satellites of interest with known uncertainty distributions, sample states can be drawn and propagated forward in time. All-on-all conjunction analysis can then be performed between the two resulting populations, where simple geometric calculations determine closest-approach distances. The result of the complete all-on-all analysis

is a cumulative distribution function of miss-distance. As the number of samples increase, the cumulative distribution function of miss-distance approaches the true value of the probability of collision as a function of satellite combined radius. Figure 1 illustrates this concept. The target number of samples for this type of analysis is on the order of 10^4 or more for each satellite, resulting in the examination of 10^8 or more combinations. Given a large sample size, this approach lends itself very well to parallelization.

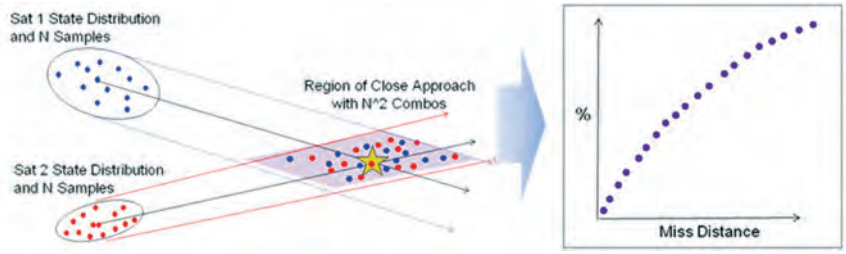


Figure 1. Monte Carlo Approach for Producing Probability of Collision Cumulative Distribution Function

This approach has been previously explored[1,2] but several assumptions and simplifications used in the analytical methods were retained in the numerical implementation, such as a two-body or SGP-4 dynamical model, in order to reduce computational requirements. In this work, no such assumptions are made, and high-fidelity special perturbations-based orbit dynamics are used; non-spherical gravity, atmospheric drag, and nonlinear dynamics are preserved for each sample. High performance computing is utilized to generate results in a timely manner.

The High Performance Computing Software Applications Institute for Space Situational Awareness (HSAI-SSA) was established to exploit the Department of Defense's high performance computing resources to help overcome our nation's SSA challenges. A combination of SSA subject matter expertise and computer science expertise allows the HSAI-SSA to address problems that otherwise could not be addressed, such as the Monte Carlo analysis described above.

2. Approach

Runs were performed on the US Air Force Research Laboratory Department of Defense (DoD) Supercomputing Resource Center's (AFRL DSRC) shared memory Altix 4700 system "Hawk", as well as the Maui High Performance Computing Center (MHPCC) DSRC distributed memory system "Mana".

In order to compare the results, an analytical method was selected from the literature. Although several of these methods exist, many rely on a similar set of simplifications and assumptions. Alfano's method of estimation[3] was chosen as a representative comparison that provides an estimate of the probability of collision. In order to compare this analytical estimate to the cumulative distribution function resulting from the Monte Carlo method, the combined object radius is varied, and the estimation is performed for each radius. The resulting plot of probability of collision against combined object radius is essentially another cumulative distribution function, which can then be directly compared against the Monte Carlo result.

3. Implementation

The Collision and Conjunction Analysis (Cocoa) code is broken into three segments: initialization, ephemeris generation, and all-on-all conjunction analysis. The initialization segment involves the ingestion of initial conditions and sampling within the uncertainty volumes for each satellite, which does not take a significant amount of time to execute. The ephemeris calculation and probability of collision segments are significantly more complicated and need to be discussed in detail.

The ephemeris generation segment relies on the special perturbations propagation program within the SPeCIAL-K catalog maintenance software suite[4–6], developed at the US Naval Research Laboratory (NRL) for operational use at DSC2 (Distributed Space Command and Control) Dahlgren. This program initially relied on the use of a database to provide necessary information. Since the time required to access disk is significantly longer than memory, an alternative method for getting necessary information to SPeCIAL-K was needed. A solution that attempted to put the database in a Ram disk, proved to be on the order of 3 times faster; although this is an improvement, it is not sufficient for the code to operate in a parallel fashion. A modified version of SPeCIAL-K was created that did not rely on accessing a database, but rather keeping

everything in memory. This proved to be an order-of-magnitude faster, and allowed the focus to be on improving the last segment of the Cocoa code, conjunction analysis. This was tackled by optimizing its memory accesses and, in the next segment, parallelizing the code to operate on many shared memory processors.

The conjunction analysis segment had to be tuned to optimally take advantage of the processors' caches. The target was to improve the use of the level 1 cache, which ultimately affects the performance of the other caches. This was accomplished by improving the spatial and temporal locality of data, as well as ensuring optimal data reuse. As this code was written in FORTRAN, memory was organized in a column-major order as opposed to row-major order in C (Figure 2).

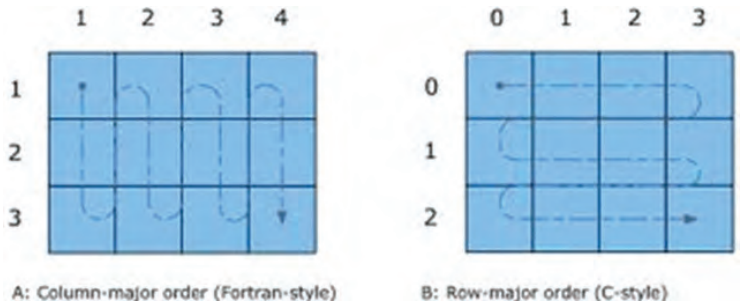


Figure 2. Column- vs. Row-major order

Following standard matrix notation, rows are identified by the first index of a two-dimensional array and columns by the second index. However, memory is linear and the mapping from array notation can be seen in Figure 3. In this case, S_{00} is in memory position 0, S_{10} is in position 1, S_{20} is in position 2, and S_{01} is in position 4.

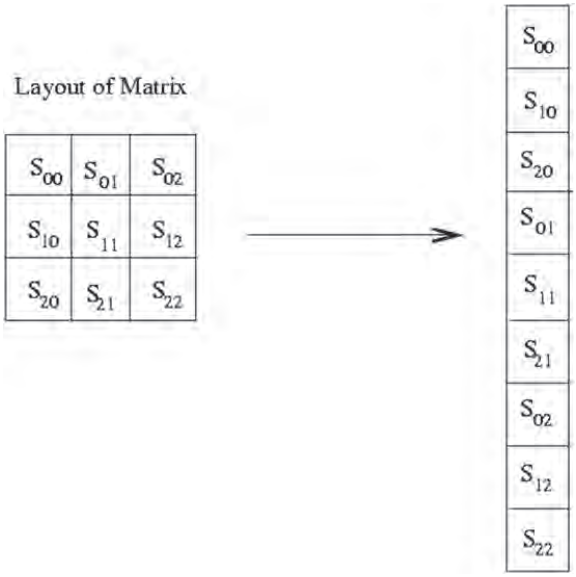


Figure 3. Visual description of column-major order

Since FORTRAN follows column-major order, it is important to make sure that array elements accessed are contiguous in memory, as this will be faster than accessing elements which are not, due to caching. The original Cocoa code was written in row-major order, even though it was coded in FORTRAN. In the HSAI-SSA, inefficient implementations such as this are often encountered when non-computationally-focused subject matter experts code algorithms in ways consistent with their understanding of the physics of the problem, rather than an understanding of the computations. The code was modified to access memory in a column-major order fashion, and performance significantly improved. The code execution time improved by 25% on "Hawk" and by 27% on "Mana".

4. Parallelization

As the number of samples, N, is increased, the number of trajectory comparisons grows at a factor N^2. A parallel implementation is required to operate as a larger number of samples are used. A parallel implementation of the Cocoa code with OpenMP libraries was chosen because the code was initially implemented on the "Hawk" Altix 4700 shared memory system. OpenMP was chosen because it requires fewer man-hours to implement by allowing incremental parallelization as opposed to MPI. The implementation of Cocoa on "Mana" is also OpenMP, but it was limited to operating within a single-node because it can only operate in a shared memory environment.

The focus of the initial parallelization effort and performance study was on the conjunction analysis segment; because the initialization segment takes very little time and the ephemeris generation segment is embarrassingly parallel. Fortunately, the conjunction analysis segment was dominated by operations that could be computed independently of each other, thus allowing for a straightforward parallelization approach. The main hurdle for achieving the best-performance is that there is a conditional statement within the main loops being parallelized. This conditional determines if the closest approach has happened between 2 samples. If the conditional is satisfied then additional computation is required. This additional work can create an imbalance in the workload each thread is assigned. Aside from this possible imbalance, the probability of collision is now written in a fashion that should achieve near-linear speed-ups until enough cores are assigned that each thread does not have enough data to keep it fully-utilized. The next section explains the benchmarking process, and shows the parallel efficiency achieved by the Cocoa code.

5. Benchmarking

The Cocoa software was implemented on MHPCC's "Mana" Linux cluster and ASC's "Hawk" SGI Altix 4700 shared memory system. Here are the details/specifications behind the systems and their benchmarks:

1. The SGI Altix 4700 system is a Shared Memory system with 500 cores per node. Each core is a 1.6 Gigahertz (GHz) Itanium2 processor and has a 32 kilobyte (KB) of Level 1 cache (16MB data/16MB instruction), 256KB of Level 2 data cache, 1 megabyte (MB) of Level 2 instruction cache, 9 megabytes (MB) of Level 3 cache, and access to either 4GB or 2GB (depending on which node the job is run) of system memory.
2. The "Mana" node is a special test node, unlike the batch nodes that consist of two quad cores processors. It is a Dell m610x, Dual Intel Xeon X5680 (6 cores)@3.33GHz, 96GB RAM. It consists of 2 processors with 6 cores per processor for a total of 12 cores. Affinity is used to best balance the workload per processor.
3. Parallel run-times are implemented using the Intel compilers and the OpenMP libraries.
4. Run-times are an average of 5 runs.
5. Speed-up is calculated from the sequential run-time divided by the parallel run-time.
6. Parallel Efficiency is a percentage relative to a linear speed-up.
7. 1,000 and 10,000 samples are tested for each satellite.
8. The Cocoa main loop was parallelized. The initialization routines that read in the data from file were not included as they took very little execution time.

The Cocoa code was run in parallel using OpenMP libraries and benchmarked on the single 12 core "Mana" node (Table 1 and Table 2) and the shared memory Altix, "Hawk" (Table 3 and Table 4). A variety of cores per processor combinations were examined. The sequential time is displayed at the bottom of the table.

Table 1 and Table 2 show the performance of the Cocoa algorithm on the 12 core "Mana" node for 1,000 and 10,000 samples to show that the code scales. The tables also show that the parallel efficiency drops steadily as more cores are utilized within the same node. This drop is most likely due to additional overhead incurred when a core accesses a cache-line from another core in the same processor, but it will be even larger when it has to access a cache-line from a core that exists on a different processor within the node (passing through Intel's QuickPath Interconnect). Distributed cache misses when data is not in the distributed cache, and must be sought in the shared cache; and shared cache misses occur when the data is not found in the distributed or shared cache and must be loaded into both the shared cache and then the distributed cache. In short, overhead occurs as more cores share the same L3 cache[7,8]. It is interesting to note the parallel efficiency is less for 4 cores per processor; this is most likely due to the fine-grained memory access governed by the memory model that is unaware of the non-uniform memory access characteristics of the underlying shared address space system[9].

Table 1. Benchmark of Cocoa Loop on a single 12 core Mana node for 1,000 samples/satellite

Processors	Cores Per Proc	Total Cores	Time: Average (s)	Relative Speed Up	Efficiency (%)
1	1	1	20.691	0.985	98.5%
2	1	2	11.119	1.833	91.6%
2	2	4	6.008	3.392	84.8%
2	3	6	4.503	4.526	75.4%
2	4	8	3.817	5.339	66.7%
2	6	12	2.429	8.390	69.9%
Sequential Performance					
1	1	1	20.380	1.00	100%

Table 2. Benchmark of Cocoa Loop on a single 12 core Mana node for 10,000 samples/satellite

Processors	Cores Per Proc	Total Cores	Time: Average (s)	Relative Speed Up	Efficiency (%)
1	1	1	2023.09	0.999	99.9%
2	1	2	1053.91	1.918	95.9%
2	2	4	562.86	3.591	89.8%
2	3	6	412.04	4.906	81.8%
2	4	8	328.68	6.150	76.9%
2	6	12	209.59	9.645	80.4%
Sequential Performance					
1	1	1	2021.41	1.00	100%

The Cocoa algorithm produced different results for the shared memory Altix system “Hawk”. Table 3 and Table 4 display the results of the algorithm for 1,000 and 10,000 samples. These tables show “Hawk” is able to achieve better overall parallel efficiency, even though its sequential performance is about 50% slower than that of “Mana”. The results show that the performance remained linear until 32 cores were used. The performance degradation beyond 16 cores is most likely due to a lack of work per core. To illustrate this, the 10,000 sample maintained a higher parallel efficiency as opposed to the 1,000 sample case.

Table 3. Benchmark of Cocoa Loop on Hawk for 1,000 samples/satellite

Processors	Total Cores	Time: Average (s)	Relative Speed Up	Efficiency (%)
1	1	30.221	0.995	99.5%
1	2	15.110	1.991	99.5%
2	4	7.579	3.969	99.2%
4	8	3.802	7.912	98.9%
8	16	1.904	15.798	98.7%
16	32	0.971	30.978	96.8%
32	64	0.509	59.096	92.3%
64	128	0.258	116.589	91.1%
128	256	0.130	231.385	90.4%
250	500	0.086	349.767	70.0%
Sequential Performance				
1	1	30.080	1.000	100%

Table 4. Benchmark of Cocoa Loop on Hawk for 10,000 samples/satellite

Processors	Total Cores	Time: Average (s)	Relative Speed Up	Efficiency (%)
1	3198.549	0.999	99.9%	3198.549
1	1598.852	1.999	99.9%	1598.852
2	801.829	3.987	99.7%	801.829
4	400.920	7.974	99.7%	400.920
8	200.481	15.945	99.7%	200.481
16	102.460	31.201	97.5%	102.460
32	53.247	60.038	93.8%	53.247
64	26.774	119.400	93.3%	26.774
128	13.393	238.695	93.2%	13.393
250	7.951	402.068	80.4%	7.951
Sequential Performance				
1	1	3196.840	1.00	100%

In addition, cases were examined with a single-core on a single-processor per node. These were tested to show the code's performance if memory from both processors were available to a single-core on a single-processor. If the code had been memory-bound, then it may have been beneficial. As the code was not memory-bound, this showed no performance gain. As a greater number of samples (N) are utilized, the problem size grows at a rate of N^2; given this growth rate, the problem may become memory-bound, and the use of fewer cores per processor may be necessary.

6. Results

In order to draw definitive conclusions about the applicability of analytic probability of collision methods, an extensive analysis must be undertaken and is currently underway within the HSAI-SSA. Some initial results are provided here as an example of how the Cocoa application is being used. Figure 4 plots the probability of collision as a function of combined object radius for the analytic probability of collision algorithm, along with the Cocoa miss-distance cumulative distribution function; which should be equivalent, for a geosynchronous satellite encounter with both reference orbits close to each other. This particular case stresses one of the major assumptions in the analytic approach, and one can see a significant difference in the resulting probability of collisions. For other less-stressing cases, excellent agreement has been observed. Efforts continue to analyze the applicability of analytic probability of collision algorithms across multiple orbit regimes and engagement scenarios.

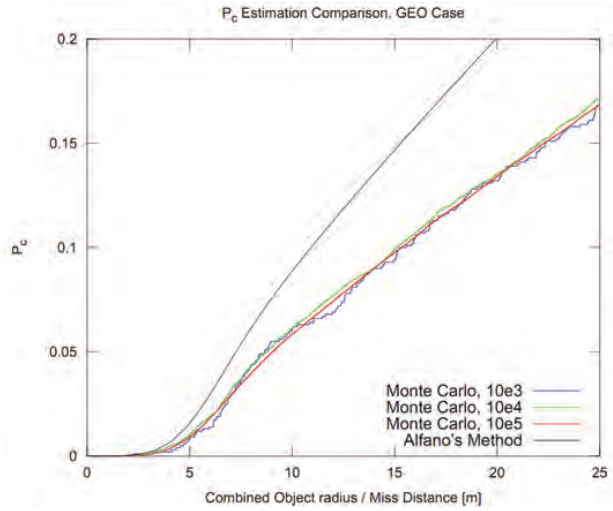


Figure 4. Comparing Cocoa Monte Carlo Results to Analytic Method for Geosynchronous Case

7. Future Work

There is additional work that can be done to improve the model's capabilities and execution time. A list of future work includes:

1. Parallelizing the SPeCIAL-K portion of the code. Although this is routine is utilized once during initialization, its execution time will increase when a greater number of samples are utilized at a rate proportional to N2. Embarrassingly parallel techniques are already being examined.
2. Modify the Cocoa software to identify close approaches between all orbiting satellites.
3. Port the code to a hybrid graphics processing unit (GPU)-based parallel architecture. CPU-intensive portions of the Cocoa code can take great advantage of GPU, as there is a high-level of independence from other concurrent calculations.
4. Re-architect the software to run in distributed memory environments.

8. Conclusion

A methodology has been created that incorporates the use of a brute-force numerical method in conjunction with a high-accuracy special perturbations propagation method to estimate the probability of collision between orbiting objects. The implementation of the Monte Carlo method now allows the Cocoa algorithm to handle a variety of cases that the less computationally-intensive analytical methods do not handle correctly. A parallel implementation of this numerical method was necessary because of the increased computational requirements. Parallelization utilizing the OpenMP libraries has been shown to achieve near linear speed-ups under certain circumstances. This implementation's speed-up allows for more accurate solutions to be obtained through the use of a greater number of samples.

Acknowledgments

This work was funded by the US Air Force Research Laboratory and the DoD High Performance Computing Modernization Program through the HSAI-SSA. The authors would like to acknowledge the support received from the Maui High Performance Computing Center, the US Air Force Academy, the Naval Research Laboratory, and the AFRL DSRC. Additional technical contributions were made by USAF Lt Tom Ainscough, Capt Lucas Roselius, Dr. Terry Alfriend, and Dr. Paul Schumacher.

References

1. Alfano, S., "Satellite Conjunction Monte Carlo Analysis," *Paper No. AAS 09-233 2009*, AAS/AIAA Space Flight Mechanics Meeting, February 2009.

2. deVries, W. and D. Phillion, "Monte Carlo Method for Collision Probability Calculations using 3D Satellite Models," *2010 AMOS Technical Conference*, Wailea, HI, September 2010.

3. Alfano, S., "A Numerical Implementation of Spherical Object Collision Probability", *Journal of the Astronautical Sciences*, Vol. 53, No. 1, pp. 103–109, January–March, 2005.

4. Neal, H.L., S.L. Coffey, and S.H. Knowles, "Maintaining the Space Object Catalog with Special Perturbations", *AAS 97-687, Astrodynamics 1997 v.97 Part II*, AAS/AIAA Astrodynamics Meeting, Sun Valley, ID, pp.1349–1360, Aug 4–7, 1997.

5. Coffey, S.L. and Neal, H.L., "An Operational Special-Perturbations-Based Catalog", *Paper AAS 98-113*, AAS/AIAA Space Flight Mechanics Conference, Monterey, CA, 1998.

6. Healy, L M. and Coffey, S.L., "Parallel Computing for Space Surveillance", *Proceedings of the Space Surveillance Workshop*, Project Report STK-193, 1992.

7. Mathias, Jacquelin, Loris Marchal, and Yves Raobert, "Complexity Analysis and Performance Evaluation of Matrix Product on Multicore Architectures", *The 38th International Conference on Parallel Processing*, Vienna, Austria, 2009.

8. Ulrich Drepper, "What Every Programmer Should Know About Memory", http://lwn.net/Articles/250967/, 2007

9. Haoqiang, Jin, Dennis Jespersen, Piyush Mehrotra, Rupak Biswas, Lei Huang, and Barbara Chapman, "High Performance Computing Using MPI and OpenMP on Multi-core Parallel Systems", *Parallel Computing*, February 2011.

Author Index

Author Index

Author Index

Author Index

UGC 2011