REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
6 April 2012

2. REPORT TYPE
Briefing Charts

3. DATES COVERED *(From - To)*
1 April 2012 – 6 April 2012

4. TITLE AND SUBTITLE
SOLVCON: An Unstructured PDE Framework (BRIEFING CHARTS)

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Chen, Y. and Bilyeu, D.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
23011158

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory (AFMC)
AFRL/RZSS
1 Ara Drive
Edwards AFB CA 93524-7013

8. PERFORMING ORGANIZATION REPORT NO.

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory (AFMC)
AFRL/RQR
5 Pollux Drive
Edwards AFB CA 93524-7048

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-RZ-ED-VG-2012-102

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution A: Approved for public release; distribution unlimited PA# 12229

13. SUPPLEMENTARY NOTES
Presentation for Ohio State University, 6 April 2012

14. ABSTRACT
This presentation discusses the construction of the new software framework that supports, pluggable multi-physics, hybrid parallelism for HPC, and productive work flows, to deliver analyzed results by using high-fidelity solutions of hyperbolic conservation laws. The new software framework is called SOLVer CONstructor, i.e., SOLVCON; it is a platform to construct PDE-solving codes.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON / 19b. TELEPHONE NO *(include area code)*
Unclassified	**Unclassified**	**Unclassified**	SAR	20	Jean-Luc Cambier / NA

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

SOLVCON: An Unstructured PDE Framework

Yung-Yu Chen, David Bilyeu

Department of Mechanical Engineering
The Ohio State University

October 2011

New Software Framework for Solving Hyperbolic Conservation Laws

- This presentation discusses the construction of the new software framework that supports
  - pluggable multi-physics,
  - hybrid parallelism for HPC, and
  - productive work flows,

  to deliver analyzed results by using high-fidelity solutions of hyperbolic conservation laws.
- The new software framework is called SOLVer CONstructor, i.e., SOLVCON; it is a platform to construct PDE-solving codes.

Outline

1. SOLVCON Framework
   - Design of the Software Framework
   - Code Development for SOLVCON

Python Programming Language

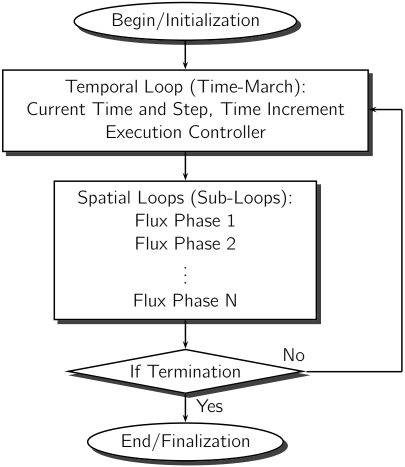
- Python enables high-level constructs:
  - Pluggable multi-physics.
  - Automatic hybrid parallelism.
  - Parallel I/O and in situ analysis/visualization.
- Python is a dynamically-typed programming language.
  - Support multiple programming paradigms: procedural (like Fortran or C), object-oriented (like C++ or Java), and functional (like Lisp or Scheme).
  - Realize high-level construct: type registries, plug-ins, etc.
- Python is designed to glue multiple programming languages together.
  - Use CUDA, C, pthread, and MPI simultaneously.
- Python is suitable to extend SOLVCON's functionalities:
  - 100+ packages in standard library and 13000+ 3rd-party packages.
  - Wrappers to many existing toolkits: VTK, netCDF, MPI, etc.

Python and C/C++

- Python and C - Ctypes
  - Compile the C code as a shared object
  - Python/ctypes cannot read in c header files so some of the contents of the header files will need to be rewritten in python, function definitions do not need to be rewritten.
  - Load the library from python using ctypes module and save to an object
  - This object contains each subroutine from the shared object file.
- Python and C++ - Boost
  - This is accomplished by writing wrappers to the C++ classes
  - Does not require any changes to the C++ code
  - I have not used this before as python handles the objects

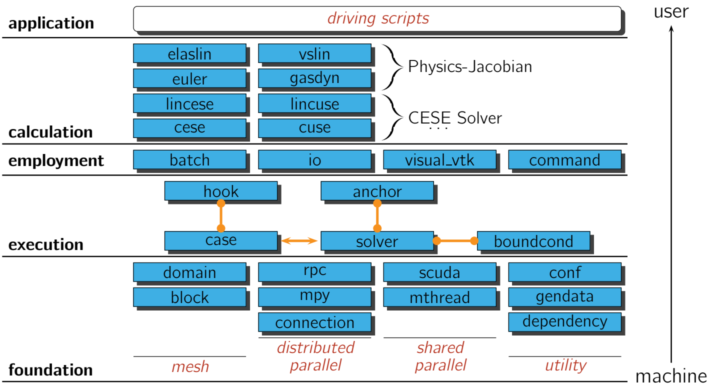
Two-Loop Structure

- All time-accurate finite-volume methods contain two loops.
  - Temporal loop time-marches for temporal integration.
  - Spatial loops iterate over elements to calculate flux.
- These two loops form the basic structure of SOLVCON.



Five-Layer Architecture

- Code is organized by using Python modules (blue solid boxes).
- A module depends only on other modules in the same layer or in the lower layers.
- The two-loop structure is hosted in the execution layer.



Driving Scripts

- The driving scripts are the highest-level construct of SOLVCON.
- A driving script must create a `Case` object and call its (i) `init()`, (ii) `run()`, and (iii) `cleanup()` methods.
  - The `Case` object represents the overall execution flow of the simulation, and contains the temporal loop.
- The driving scripts can specify logic to the simulations in addition to parameters.
  - Anything higher than the foundation layer (the lowest layer) can be replaced by code written in driving scripts.
  - Including but not limited to `Case`, `Solver`, `BC` classes, `Hook` and `Anchor` classes.
- SOLVCON does not use input files, but uses driving scripts instead (because Python code needs no explicit compilation).

Calculation

- Solver
  - This is a generic hyperbolic non-linear solver that has been optimized for both CPU and GPU and has been simplified for linear equations.
  - This is accomplished through the use of pre-processors that optimize/simplified certain portions of the code
  - Through the use of functional pointers these routines will call the required Jacobian subroutines.
- Constitutive Equations
  - These subroutines are called by the solver schemes and calculate the Jacobian and fluxes
  - SOLVCON has built in support for Euler and linear solvers.
  - New physics can be added by creating new Jacobian routines and by modifying certain parts of the default case and solver codes.
  - This is accomplished through inheritance

Employment

- batch: Used to submit parallel jobs on a cluster. Built in support for Torque
- io: Reads in the mesh and writes the VTK output files
  - Input capable of reading in: Gambit Neutral and Genesis/Exodus.
  - Output capable of writing: VTK in both serial and parallel, ASCII and binary.
- visual_vtk: Provides real time visualization and access to VTK through the VTK python wrappers.
- command: Provides the infrastructure for command line arguments

Execution

- hook: Allocates locations where the user can insert code into the temporal loop. Code inserted here can run in serial mode only
- anchor: Similar to the hook but the code is inserted into spatial loop. Code inserted here runs on each compute node.
- case: Provides the basic helper subroutines to support the solver such as:
  - cfl calculations
  - Convergence checks
  - I/O post processing support such as track results along a line or at a particular point
- solver: Defines the structure of the main program.
  - Defines the main data structure
  - Initializes/creates the data in the structure
  - Provides the routine to be called in the marching routine

Boundary-Condition Treatments

- SOLVCON uses ghost cells to treat boundary conditions (BC).
  - BC treatments depend on (i) numerical algorithms, (i) physical models, and (iii) mesh data structures.
- SOLVCON decouples BC treatments from numerical algorithms.
  - The `BC` class hierarchy is used to hold the code.
- A BC treatment is a spatial sub-loop that iterates over only boundary cells.

```
ConcreteSolver(Solver):
  def bound_flux():
    for bc in self.bclist:
      bc.bound_flux()
  def bound_gradient():
    for bc in self.bclist:
      bc.bound_gradient()
```

```
ConcreteBC(BC):


  def bound_flux():
    ...

  def bound_gradient():
    ...
```

Boundary-Conditions

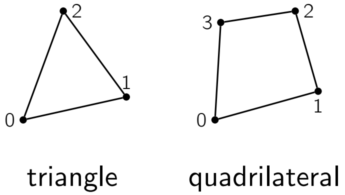
- Boundary conditions can be defined by either the solver or the physics.
- Boundary conditions specified by the solver are generic and are applicable to all physics. These boundary conditions are:
  - Non-reflecting
  - Periodic
- Boundary conditions specified by the physics are only available to the physics that creates them. Some examples are:
  - Non-Viscous wall
  - Pressure inlet/outlet
  - Non-Conducting
  - ...

Foundation

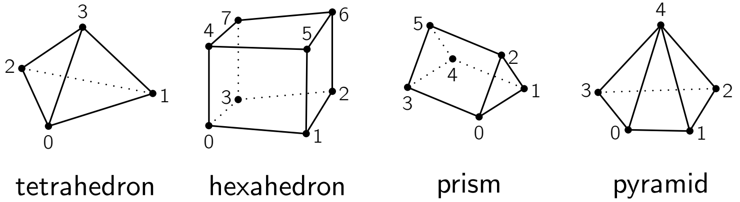
- Mesh
  - domain: oversees the domain decomposition using Metis or Scotch and distributes the domains over the network
  - block: Provides the data structures for the unstructured mesh
- distributed parallel
  - rpc: Remote Procedure call and inter-process communication
  - mpy: Python wrappers to MPI
  - connection: Remote connections and communications between nodes
- Shared parallel
  - scuda: A wrapper to the CUDA shared libraries through ctypes
  - mthread: Multi-threading though pthreads, OpenMP, MPI
- utility
  - conf: Info about the configuration of SOLVCON
  - gendata: Generic data structure
  - dependency: Manages the external shared libraries

Currently Supported 2/3D Primitive shapes

- Two-dimensional elements:



- Three-dimensional elements:



Data Structures of Unstructured Meshes

- Three types of entities: nodes, faces, and cells.
- The spatial domain of interest is covered by non-overlapping cells.
- Two sets of arrays define the meshes.

Connectivity

- `clnds`: nodes in each cell.
- `clfcs`: faces in each cell.
- `fcnds`: nodes in each face.
- `fccls`: cells related by each face.

Geometry

- `ndcrd`: coordinates of each node.
- `fccnd`: center of each face.
- `fcnml`: unit normal vector of each face.
- `fcara`: area of each face.
- `clcnd`: center of each cell.
- `clvol`: volume of each cell.

On-the-Fly Analysis

- Solution processing is not part of the numerical algorithms.
- SOLVCON uses the callback mechanism to separate the supportive functionalities from numerical algorithms and physical models.
  - `Hook`: The outer temporal loop.
  - `Anchor`: The inner spatial loops.
- Example 1: Initial condition.
  - SOLVCON calls the `preloop()` method before entering the temporal loop.
- Example 2: Calculate physical quantities.
  - SOLVCON calls the `postmarch()` method after finishing all spatial loops for each time step.
- The analysis code can be packaged with solver kernels.

Coding for In Situ Visualization

- SOLVCON directly calls external visualizing libraries by using Python.
  - Currently support VTK.
- VTK interface is provided in SOLVCON.
  - Provide one-way data converter from SOLVCON to VTK.
  - Use VTK's official Python wrappers to access all VTK functionalities, e.g., cut surface, contour, iso-surface, etc.
- In situ visualization is programmed in driving scripts.
  - Visualization differs from one case to another.
  - No hard-wired code.
- The driving scripts become an application program that can deliver analyzed results including graphic files.

SOLVCON is Open-Sourced

- Released under GNU GPL v2 with full source.
  - Freely available at `http://solvcon.net/`
- Systematic open-source practices: (i) Distributed version control system, (ii) unit testing, (iii) issue tracking, (iv) continuous integration, (v) auto-generated API documentation.