REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 09-12-2019	2. REPORT TYPE Final Report	3. DATES COVERED (From - To) 16-Sep-2014 - 15-Sep-2019

4. TITLE AND SUBTITLE Final Report: Multiscale Methods for the Reliable Simulation of Multiphase Processes	5a. CONTRACT NUMBER W911NF-14-1-0301
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 611102
6. AUTHORS	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES AND ADDRESSES Rensselaer Polytechnic Institute 110 8th Street Troy, NY 12180 -3522	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) 65296-MA.31

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Mark Shephard
UU	UU	UU	UU		19b. TELEPHONE NUMBER 518-276-8044

Standard Form 298 (Rev 8/98)
Prescribed by ANSI Std. Z39.18

RPPR Final Report
as of 16-Dec-2019

Agency Code:

Proposal Number: 65296MA **Agreement Number: W911NF-14-1-0301**

INVESTIGATOR(S):

Name: Assad Oberai
Email: aoberai@usc.edu
Phone Number: 2137401882
Principal: N

Name: Onkar S. Sahni
Email: sahni@rpi.edu
Phone Number: 5182768560
Principal: N

Name: Mark S. Shephard shephard@s
Email: shephm@rpi.edu
Phone Number: 5182768044
Principal: Y

Organization: **Rensselaer Polytechnic Institute**
Address: 110 8th Street, Troy, NY 121803522
Country: USA

DUNS Number: 002430742 EIN: 141340095
Report Date: 15-Dec-2019 Date Received: 09-Dec-2019
Final Report for Period Beginning 16-Sep-2014 and Ending 15-Sep-2019
Title: Multiscale Methods for the Reliable Simulation of Multiphase Processes
Begin Performance Period: 16-Sep-2014 **End Performance Period:** 15-Sep-2019
Report Term: 0-Other
Submitted By: Mark Shephard Email: shephm@rpi.edu
Phone: (518) 276-8044
Distribution Statement: 1-Approved for public release; distribution is unlimited.

STEM Degrees: 1 **STEM Participants:** 1

Major Goals: Multiphase problems are ubiquitous in science and engineering. In all cases these problems are characterized by interfaces that evolve, thus any simulation procedure developed to address them must be able to effectively track the interfaces. In cases where the key interface physics is limited to the motion of the interface between phases (e.g., breaking waves, hydraulic jumps, plunging jets, etc.) modeling methods that employ an implicit representation of the phase interfaces have been successfully developed. These methods are popular because they avoid the geometric complications of explicitly tracking interfaces. Since the implicit methods smear the interface physics over a mesh size dependent portion of the domain, they are less well suited for use in simulating problems where the physics of interest is highly localized to the interface and/or the details of the interface geometry must be carefully reflected in the models and discretizations (meshes) being used. Many problems of interest involve reactive interfaces where different physics on each side of the interface must be addressed in the phase models (e.g. a burning interface). The application of methods that employ an explicit representation of the interfaces can more directly, and accurately, model the desired interface physics. An explicit representation of the interface geometry is also of importance to maintain when there is relative motion of objects in close proximity to each other and the quantities of interest are a function of the shape of the interfaces and quality of meshes used between the interacting objects. The central focus of this project has been the development of a set of models and model discretization methods for the effective simulation of multiphase of three-dimensional problems in which interfaces are explicitly represented and tracked.

To effectively meet the goals the following technologies needed to be developed:
- Thermodynamically consistent mathematical models capable of representing the physics of reactive interfaces while accounting for the mass, momentum and energy transfer on each side of the interface.
- Discretization methods that can effectively represent the interface physics, including appropriate discontinuities without introducing undue cost through the entire domain discretization when using mesh based methods.

• Mesh control methodologies that can be effectively coupled with explicit interface geometry tracking methods to adaptively control anisotropic mesh configurations.

To satisfy the computational requirements the resulting procedures have been developed to execute in parallel as part of a coordinated simulation workflow. A final aspect of the project was the demonstration of the methods developed on problems with reactive interfaces and problems with evolving geometries where accurate geometry tracking and mesh configurations are needed.

Accomplishments: In this project we have developed adaptive methods to solve the multiphase problem where the dynamic interface is undergoing a change of phase or involves relative motion between different surfaces/objects.

Finite element formulation for compressible phase change problems

We have developed a thermodynamically consistent formulation of jumps in thermo-mechanical quantities at an interface between two materials, where the interface represents a surface of phase change or that of a chemical reaction. We have done so for general (solid or fluid) compressible media. We have developed and implemented a variational formulation of these jump terms in a parallel fluid flow solver (PhaseChangePHASTA). These jump conditions translate to discontinuities in certain thermo-mechanical variables such as density, normal velocity, and pressure. Most prevalent implementations of these jump terms smear these discontinuities over a fictitious region confined around the interface whose width is numerically motivated. In this region, these approaches are forced to assume the existence of a mixture whose material behavior (constitutive models, equations of state) is an ad-hoc combination of the material on either side of the interface.

We have chosen not to adopt this approach. Rather, we allow for discontinuous interpolation across the evolving interface. As a result, we do not have to assume the existence of a fictitious material. We implement the jump terms using a discontinuous Galerkin (DG) finite element method and explicitly represent and track the interface. We note that within our approach the discontinuity is introduced only at the location of the interface (as it is explicitly tracked). In this way our approach is more economical than the standard DG finite element method.

We have analyzed the conservation properties of this method and demonstrated that up to errors in time-discretization it exactly balances the global mass, momentum and energy between the phases. We have also developed a priori error estimates for this method on a related, linear advection-diffusion problem and shown that the error in the numerical approximation converges at an optimal rate in an operator-induced discrete norm. In addition, we have also developed an inexpensive a posteriori error estimator (an explicit one) and used it to drive mesh adaptivity.

We note that our formulation can accommodate cases where the phase change rate at the interface is determined by the local thermodynamic variables, as is the case in Vielle’s law. Further, we have considered the limited application of our approach to problems where the dense phase is a solid. In order to model solids, we have developed a formulation wherein the velocities are mapped to displacements, which are necessary to evaluate the stress components for a solid.

Accompanying methods for evolving mesh and geometry

A necessary capability in an explicit interface tracking approach, like the one described above, is the evolution of the geometry and mesh during the simulation. The geometry can evolve in shape and topology. Similarly, the mesh can evolve in shape and topology/connectivity. Our focus to this point has been on developments and problems that involve shape changes in the geometry along with shape and topological changes in the mesh. For these capabilities we have integrated PhaseChangePHASTA with geometry and mesh libraries developed at RPI (i.e., PUMI) and Simmetrix (i.e., GeomSim and MeshSim). The latter set of libraries were developed under an Army STTR Phase II project lead by Simmetrix and were fully utilized in this ARO project.

The geometry of the evolving interface due to phase change is based on a discrete boundary description where the mesh moves with the interface, while within the volume arbitrary mesh motion is used based on an arbitrary Lagrangian-Eulerian (ALE) frame. To tackle this most effectively the mesh motion procedures employed used a variable stiffness elasticity analogy which does a good job of controlling element shapes for interface evolution over a number of time steps. However, this form of mesh motion does not fully maintain the structure and desired normal resolution in the layered portion of the mesh at the interface. Thus, a procedure that uses connectivity of growth curves (in the layered mesh) has been derived and implemented to explicitly control the structure and normal resolution (including gradation) near the interface when the interface is changing shape.

Mesh motion alone can only support a given amount of evolution in the interface/geometry. Thus, during the simulation it is required to employ methods that modify the mesh topology to maintain mesh quality and validity, as well as to ensure the appropriate mesh resolution based on adaptivity. We have integrated the parallel mesh adaptation procedures with PhaseChangePHASTA to support mesh adaptation steps with topological changes. In addition, the overall control loop, with all steps executing in parallel has been developed. In this loop, mesh adaptations with topological changes are carried out in an outer loop inside which an inner loop of mesh motion is employed over multiple time steps of flow solver. Furthermore, procedures have been developed for an efficient monitoring of the mesh to determine mesh adequacy during the simulation in order to automatically trigger mesh adaptation with topological changes.

In addition to phase change-related geometric changes, geometric changes can occur due to relative motion between different surfaces. An example is the case of the projectile moving down the tube or barrel of a firearm. In such cases the geometry is effectively described using a parametric boundary description (in contrast to a discrete representation of an arbitrarily evolving interface). This parametric representation is used to advantage when the elasticity based mesh motion predicts off-surface locations for boundary mesh vertices. To maintain locations on the appropriate surface we have integrated PhaseChangePHASTA with the geometry library and use geometric interrogations to ensure boundary mesh vertices remain on the parametric.

Simulation of problems of interest

We have worked on several problems that are used to verify, validate and demonstrate the tools we have developed. These include, (a) Stefan problem, which is a one-dimensional problem whose results are used to verify our formulation. (b) Exothermic phase change of a spherical droplet, which is a three-dimensional counterpart of the Stefan problem in spherical coordinates. (c) Burning of multiple grains in a chamber. This is a fully three-dimensional problem that is used to test the performance of our algorithms in a complex setting. It is also used to verify whether we are able to accurately predict the pressure and temperature rise in the chamber. (d) Modeling of the motion of a projectile down the barrel of a firearm. This problem thoroughly tests our mesh motion and adaptation capabilities along with mesh snapping to parametric geometry.

We have also demonstrated these technologies and problems to the US Army researchers at Benet Labs. In general, these capabilities are of direct interest to US Army CCDC Armaments Center researchers including those at Benet Labs. In addition, the developed technologies are also of interest to other US Army researchers and code developers, and specific relevant procedures have been integrated with the CREATE-AV Helios code and ERDC Coastal and Hydraulics Lab’s Proteus code under a DoD HPCMP PETTT project.

Training Opportunities: The current project has provided direct support for three doctoral students who have been trained in the broad area of computational science and engineering. Further, one postdoctoral researcher was trained, and has been provided professional opportunities (advising/mentoring, conference presentations, etc.) to enhance his career. In addition, several (bachelors, masters and doctoral) students, who were supported on other related DoD projects (including a DoD HPCMP PETTT project), have been able to interact with this project's team and develop supporting simulation technologies for other application areas of direct interest to the US Army. Some of these students have been hired, or offered a position, by different US Army groups including Steven Tran who was hired in the CREATE-AV Helios code team, Alvin Zhang who has accepted a position at ERDC's Coastal and Hydraulics Lab, and Erik Larrabee who is being considered for a position at Benet Labs.

Results Dissemination: The results of this project have been disseminated through the following mechanisms:
- Publications
- Presentations
- Software made available to Army collaborators
- Meetings with Army collaborators

See the technology transfer section for a summary of the interactions with Army collaborators.

The following indicates publications related to this project in three groups:
- Direct project publications
- Subset of publications on the technologies being developed that support this project
- Publications discussing applications of interest to the Army

There is also a list of project conference and other relevant presentations.

Project Papers

Granzow, B.N., Oberai, A.A, & Shephard, M.S. (2019). An Automated Approach for Parallel Adjoint-Based Error Estimation and Mesh Adaptation. Accepted Eng. Computers.

Chandra, A., Keblinski, P., Sahni, O., & Oberai, A. A. (2019). A continuum framework for modeling liquid-vapor interfaces out of local thermal equilibrium. Int. J. Heat and Mass Transfer, 144, 118597.

Zhang, Y., Chandra, A., Yang, F., Shams, E., Sahni, O., Shephard, M., & Oberai, A. A. (2019). A locally discontinuous ALE finite element formulation for compressible phase change problems. J. of Comp. Physics, 393, 438-464.

Carson, R. A., & Sahni, O. (2017). Scaling Laws for the Peak Overpressure of a Cannon Blast. J. of Fluids Eng., 139(2), 021204.

Yang, F., Chandra, A., Zhang, Y., Tendulkar, S., Nastasia, R., Oberai, A.A., Shephard, M.S., & Sahni, O. (2019). A Parallel Interface-tracking Approach for Evolving Geometry Problems. Accepted Eng. Computers.

Supporting Technology Development Papers

Granzow BN, Oberai AA, Shephard MS. Adjoint-based error estimation and mesh adaptation for stabilized finite deformation elasticity. Comp. Meth. App. Mech. Eng.. 2018 1;337:263-80.

Granzow BN, Shephard MS, Oberai AA. Output-based error estimation and mesh adaptation for variational multiscale methods. Comp. Meth. App. Mech. Eng.. 2017 1;322:441-59.

Li J, Jagalur-Mohan J, Oberai AA, Sahni O. Stochastic variational multiscale analysis of the advection–diffusion equation: Advective–diffusive regime and multi-dimensional problems. Comp. Meth. App. Mech. Eng.. 2017 1;325: 766-99.

Smith, C.W., Granzow, B., Diamond, G., Ibanez, D., Sahni, O., Jansen, K.E. and Shephard, M.S., 2018. In? memory integration of existing software components for parallel adaptive unstructured mesh workflows. Concurrency and Computation, 30(18), p.e4510.

Smith, C.W., Rasquin, M., Ibanez, D., Jansen, K.E. and Shephard, M.S., 2018. Improving unstructured mesh partitions for multiple criteria using mesh adjacencies. SIAM J. Scientific Computing, 40(1), pp.C47-C75.

Sahni, O., Ovcharenko, A., Chitale, K.C., Jansen, K.E. and Shephard, M.S., 2017. Parallel anisotropic mesh adaptation with boundary layers for automated viscous flow simulations. Eng. Computers, 33(4), pp.767-795.

Ibanez, D.A., Seol, E.S., Smith, C.W. and Shephard, M.S., 2016. PUMI: Parallel unstructured mesh infrastructure. ACM Trans. Math. Software, 42(3), p.17.

Related Papers of interest to the Army

Misiorowski, M, Gandhi F, Oberai AA. Computational Study on Rotor Interactional Effects for a Quadcopter in Edgewise Flight. AIAA J. 2019 Aug 23:1-1.

Misiorowski, M., Gandhi, F., and Oberai, A., “Computational Analysis and Flow Visualization of Ducted a Rotor in Edgewise Flight,” J. AHS, 64, 042004 (2019).

Misiorowski, MP, Gandhi FS, Oberai AA. Computational Study of Diffuser Length on Ducted Rotor Performance in Edgewise Flight. AIAA J.. 2018 Dec 31;57(2):796-808.

Tran, S. and Sahni, O., “Finite Element based Large Eddy Simulation using a Combination of the Variational Multiscale Method and the Dynamic Smagorinsky Model”, J. Turbulence, 18(5), 391-417, 2017.

Tran, S., Cummings, R. and Sahni, O., “Finite Element based Large Eddy Simulation of Flow over Bluff Bodies”, Comp. Fluids, 158, 221-235, 2017.

Conference Presentations

Exploring the effect of temperature discontinuity at interfaces in transient liquid-vapor phase change processes. C., Anirban, Z. Liang, A. Oberai, O. Sahni, and P. Keblinski. APS (2019).
Analysis-driven geometry and meshing for large scale simulations. S. Tendulkar, M. Beall, R. Nastasia, B. Downie, O. Klaas, M. Shephard, and O. Sahni. Proc. NAFEMS World Congress, Quebec, Jun 2019.
New Application Domains for the Variational Multiscale Method. Assad A. Oberai. WCCM 2018, NY, NY, July 2018.

Solving phase change problems using stabilized finite elements with improved interfacial conditions. C., Anirban, A. Oberai, P.l Keblinksi, and O. Sahni. APS DFD (2018).

An interface-tracking computational infrastructure for compressible multiphase processes. Yang, F., Chandra, A., Zhang, Y., Tendulkar, S., Nastasia, R., Oberai, A., Sahni, O. (2018, November). APS DFD.

Recent Efforts on Curved Element Mesh Adaptation, M.H. Siboni, K. Kamran, A. Dahale, O. Sahni & and M.S. Shephard, WCCM, NY, NY, July 23, 2018.

Geometry and Mesh Representation for High Order Methods. M.S. Shephard and M.W. Beall, WCCM, NY, NY, July 25, 2018.

A Parallel Geometry and Mesh Infrastructure for Explicit Phase Tracking in Multiphase Problems. Yang, F., Chandra, A., Zhang, Y., Shams, E., Tendulkar, S., Nastasia, R., Sahni, O. (2017, November). APS DFD.

Mesh Motion and Adaptation for Two-Phase Flow Problems with Moving Objects, O. Sahni, A. Zhang, M.S. Shephard and C. Kees, SIAM CSE17, 2017 SIAM Conf. on Comp. Sci. and Eng., Atlanta, GA, USA, Feb-Mar 2017.

A Stabilized Finite Element Method for Compressible Phase Change Problems. Zhang, Y., Yang, F., Chandra, A., Shams, E., Shephard, M., Sahni, O., & Oberai, A. (2017, Nov.). APS DFD.

Unstructured Finite Elements and Dynamic Meshing for Explicit Phase Tracking in Multiphase Problems. Chandra, A., Yang, F., Zhang, Y., Shams, E., Sahni, O., Oberai, A., & Shephard, M. (2017, Nov.). APS DFD.

Stabilized Finite Element Methods for Compressible Phase Change Phenomena, E. Shams, F. Yang, Y. Zhang, A. Chandra, M. Shephard, O. Sahni & A. Oberai, 14th US Nat. Con. Comp. Mech., Montreal, July 2017.

Fast Dynamic Load Balancing Tools for Extreme Scale System. C. Smith, G. Diamond and M.S. Shephard. SIAM Conf. on Comp. Science and Eng., Atlanta, GA, March 1, 2017

Parallel Geometry and Meshing Adaptation with Application to Problems with Evolving Domains. S. Tendulkar, O. Klaas, and M.W. Beall, and M.S. Shephard. SIAM Conf. on Comp. Sci.and Eng., Atlanta, GA, March 3, 2017.

Unstructured finite element simulations of compressible phase change phenomena. Shams, E., Yang, F., Zhang, Y., Sahni, O. Shephard, M., Oberai, A. APS DFD, 2016, Boston, MA.

A Finite Element Method for Simulation of Compressible Cavitating Flows, E. Shams, F. Yang, Y. Zhang, O. Sahni, M. Shephard and A. Oberai, APS DFD, Portland, OR, Nov. 20, 2016.

C.W. Smith, B. Granzow, D. Ibanez, O. Sahni, K.E. Jansen and M.S. Shephard, “Building Parallel Unstructured Adaptive Workflows Using In-memory Integration of Existing Software Components”, NSF XSEDE 16 Conf., Miami, FL, USA, July 2016

M.S. Shephard, C.W. Smith, D.A. Ibanez, B. Granzow, M.W. Beall, and S. Tendulkar, “Developing scalable components for massively parallel adaptive simulations”, Proc. NAFEMS World Congress, San Diego, CA, 10 pages, 2015.

Unstructured finite element simulations of compressible phase change phenomena. Shams, E., Yang, F., Zhang, Y., Sahni, O., Shephard, M., & Oberai, A., 2015, APS DFD.

Additional Relevant Presentations

New Application Domains for the Variational Multiscale Method. A.A. Oberai. CSRI, Sandia National Labs, Albuquerque, July, 2017.

Adaptive Unstructured Approaches for Problems with Fluid-structure and Multiphase Interactions at Extreme Scale. O. Sahni and M.S. Shephard. ERDC Fluid-Structure Interaction Workshop (organized by Chris Kees), Vicksburg, MS, USA, Apr. 2016

Towards High-order Adaptive Analysis of Complex Boundary Layer Flows at Extreme Scale. O. Sahni. NASA Langley, Hampton, VA, USA, Feb. 2016

Honors and Awards: Assad Oberai: American Institute for Medical and Biological Engineering (AIMBE) College of Fellows, 2016.

Mark Shephard: Member of the faculty team that won the 2016 School of Engineering Outstanding Team Award, RPI.

Mark Shephard: Associate Editor, Siam Journal on Scientific Computing, 2018

Protocol Activity Status:

Technology Transfer: The simulation software developed in this project (i.e., PhaseChangePHASTA) has been made available to the US Army researchers, specifically to the technical staff at Benet Labs. PhaseChangePHASTA code is provided in a pre-installed fashion including a set of relevant test cases on RPI’s computer systems and along with the latest release of meshing and geometry libraries (that were developed by Simmetrix under a complementary STTR Phase II project) in the form of a complete simulation workflow.

Several presentations and demonstrations were provided to US Army researchers at Benet Labs providing project accomplishments and detailing simulations of burning grains and moving projectile. Further, video tutorials have been created and provided for all the relevant steps ranging from geometry and mesh creation to performing adaptive analysis to post-processing. A pair of live demonstrations were also made for the simulation of moving projectiles.

We further note that the simulation technologies developed in the current project are also of interest to other US Army researchers and code developers, and under a synergistic DoD HPCMP PETTT project certain procedures have been integrated with the CREATE-AV Helios code and ERDC Coastal and Hydraulics Lab’s Proteus code. We continue to interact with the Helios and Proteus code developers including Dr. Andrew Wissink, Dr. Vinod Lakshminarayan and Dr. Chris Kees. Specifically, we continue to interact to provide near-body error estimation and mesh adaptivity for the Helios code to support rotory wing simulations as well as error estimation, predictive resolution and mesh adaptivity for the Proteus code to support two-phase simulations of three-dimensional wave and river current processes interacting with floating structures (i.e., evolving geometry).

PARTICIPANTS:

Participant Type: PD/PI
Participant: Assad A. Oberai
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Co PD/PI
Participant: Onkar Sahni
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Postdoctoral (scholar, fellow or other postdoctoral position)
Participant: Ehsan Shams
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Co PD/PI
Participant: Mark S Shephard
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Graduate Student (research assistant)
Participant: Fan Zhang
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Graduate Student (research assistant)
Participant: Anirban Chandra
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:

National Academy Member: N
Other Collaborators:

Participant Type: Graduate Student (research assistant)
Participant: Yu Zhang
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

CONFERENCE PAPERS:

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: USNCCM 13
Date Received: 30-Aug-2016 Conference Date: 26-Jul-2015 Date Published:
Conference Location: San Diego
Paper Title: Variational Multi-Scale Analysis of Stochastic Partial Differential Equations
Authors: J. Li, A.A. Oberai and O. Sahni
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: ERDC Fluid-Structure Interaction Workshop
Date Received: 30-Aug-2016 Conference Date: 01-Apr-2016 Date Published: 01-Apr-2016
Conference Location: Vicksburg
Paper Title: Adaptive Unstructured Approaches for Problems with Fluid-structure and Multiphase Interactions at Extreme Scale
Authors: O. Sahni and M.S. Shephard
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: NASA Langley Research Center Technical Talk
Date Received: 19-Nov-2019 Conference Date: 02-Feb-2016 Date Published: 02-Feb-2016
Conference Location: Hampton, VA
Paper Title: Towards High-order Adaptive Analysis of Complex Boundary Layer Flows at Extreme Scale
Authors: Onkar Sahni
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: APS Division of Fluid Dynamics Meeting
Date Received: 30-Aug-2016 Conference Date: 02-Nov-2015 Date Published: 02-Nov-2015
Conference Location: Boston, MA
Paper Title: Unstructured finite element simulations of compressible phase change phenomena
Authors: Shams, E., Yang, F., Zhang, Y., Sahni, O. Shephard, M., Oberai, A
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: NSF XSEDE 16 Conference
Date Received: 30-Aug-2016 Conference Date: 01-Jul-2016 Date Published: 01-Jul-2016
Conference Location: Miami, FL
Paper Title: Building Parallel Unstructured Adaptive Workflows Using In-memory Integration of Existing Software Components
Authors: C.W. Smith, B. Granzow, D. Ibanez, O. Sahni, K.E. Jansen and M.S. Shephard
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: Proc. NAFEMS World Congress
Date Received: 30-Aug-2016 Conference Date: 01-Sep-2015 Date Published: 01-Sep-2015
Conference Location: San Diego
Paper Title: Developing scalable components for massively parallel adaptive simulations
Authors: M.S. Shephard, C.W. Smith, D.A. Ibanez, B. Granzow, M.W. Beall, and S. Tendulkar
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 72nd Annual Meeting of the APS Division of Fluid Dynamics
Date Received: 20-Nov-2019 Conference Date: 25-Jan-2019 Date Published:
Conference Location: Seattle, Washington
Paper Title: Exploring the effect of temperature discontinuity at interfaces in transient liquid-vapor phase change processes
Authors: Anirban Chandra, Zhi Liang, Assad Oberai, Onkar Sahni, Pawel Keblinski
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 71st Annual Meeting of the APS Division of Fluid Dynamics
Date Received: 20-Nov-2019 Conference Date: 21-Nov-2018 Date Published:
Conference Location: Atlanta, Georgia
Paper Title: Solving phase change problems using stabilized finite elements with improved interfacial conditions
Authors: Anirban Chandra, Assad Oberai, Pawel Keblinksi, Onkar Sahni
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 71st Annual Meeting of the APS Division of Fluid Dynamics
Date Received: 20-Nov-2019 Conference Date: 20-Jan-2018 Date Published:
Conference Location: Atlanta, Georgia
Paper Title: An interface-tracking computational infrastructure for compressible multiphase processes
Authors: Fan Yang, Anirban Chandra, Yu Zhang, Saurabh Tendulkar, Rocco Nastasia, Assad Oberai, Mark S
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 70th Annual Meeting of the APS Division of Fluid Dynamics
Date Received: 20-Nov-2019 Conference Date: 21-Nov-2017 Date Published:
Conference Location: Denver, Colorado
Paper Title: A Parallel Geometry and Mesh Infrastructure for Explicit Phase Tracking in Multiphase Problems
Authors: F. Yang, A. Chandra, Y. Zhang, E. Shams, S. Tendulkar, N. Rocco, A.A. Oberal, M.S. Shephard, O. Sal
Acknowledged Federal Support: **Y**

RPPR Final Report
as of 16-Dec-2019

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 70th Annual Meeting of the APS Division of Fluid Dynamics
Date Received: 20-Nov-2019 Conference Date: 21-Nov-2017 Date Published:
Conference Location: Denver, Colorado
Paper Title: Stabilized Finite Element Method for Compressible Phase Change Problems
Authors: u Zhang, Fan Yang, Anirban Chandra, Ehsan Shams, Mark Shephard, Onkar Sahni, Assad Oberai
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 70th Annual Meeting of the APS Division of Fluid Dynamics
Date Received: 20-Nov-2019 Conference Date: 21-Nov-2017 Date Published:
Conference Location: Denver, Colorado
Paper Title: Unstructured Finite Elements and Dynamic Meshing for Explicit Phase Tracking in Multiphase Problems
Authors: Anirban Chandra, Fan Yang, Yu Zhang, Ehsan Shams, Onkar Sahni, Assad Oberai, Mark Shephard
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 69th Annual Meeting of the APS Division of Fluid Dynamics
Date Received: 20-Nov-2019 Conference Date: 21-Nov-2016 Date Published:
Conference Location: Portland, Oregon
Paper Title: A Finite Element Method for Simulation of Compressible Cavitating Flows
Authors: Ehsan Shams, Fan Yang, Yu Zhang, Onkar Sahni, Mark Shephard, Assad Oberai
Acknowledged Federal Support: **Y**

Copies of the papers that have appeared, or been accepted, since the last progress report are attached.

Chandra, A., Keblinski, P., Sahni, O., & Oberai, A. A. (2019). A continuum framework for modeling liquid-vapor interfaces out of local thermal equilibrium. *International Journal of Heat and Mass Transfer*, 144, 118597.

Zhang, Y., Chandra, A., Yang, F., Shams, E., Sahni, O., Shephard, M., & Oberai, A. A. (2019). A locally discontinuous ALE finite element formulation for compressible phase change problems. *Journal of Computational Physics*, 393, 438-464.

Granzow, B.N., Oberai, A.A., & Shephard, M.S. (2019). An Automated Approach for Parallel Adjoint-Based Error Estimation and Mesh Adaptation. Accepted for publication in *Engineering with Computers*.

Yang, F., Chandra, A., Zhang, Y., Tendulkar, S., Nastasia, R., Oberai, A.A., Shephard, M.S., & Sahni, O. (2019). A Parallel Interface-tracking Approach for Evolving Geometry Problems. Acceptted for publication in *Engineering with Computers*.

International Journal of Heat and Mass Transfer 144 (2019) 118597

Contents lists available at ScienceDirect

International Journal of Heat and Mass Transfer

journal homepage: www.elsevier.com/locate/ijhmt

A continuum framework for modeling liquid-vapor interfaces out of local thermal equilibrium

Anirban Chandra [a], Pawel Keblinski [b], Onkar Sahni [a], Assad A. Oberai [c,*]

[a] Department of Mechanical, Aerospace, and Nuclear Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, USA
[b] Department of Material Science and Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, USA
[c] Department of Aerospace and Mechanical Engineering, University of Southern California, Los Angeles, CA 90089, USA

ARTICLE INFO

Article history:
Received 18 April 2019
Received in revised form 16 August 2019
Accepted 16 August 2019
Available online 30 August 2019

Keywords:
Liquid-vapor phase change
Finite elements
Interface
Schrage

ABSTRACT

Continuum numerical methods modeling liquid-vapor phase change processes typically assume local thermal equilibrium at the liquid-vapor interface – continuity of temperature at phase interfaces, and a relation between interfacial saturation pressure and temperature based on phase co-existence curves. Several standard macroscopic problems have been solved accurately by adhering to this assumption. However, in micro-scale and certain non-standard macro-scale applications, significant jumps in temperature are observed at liquid-vapor interfaces during phase change, and vapor pressure can be far from equilibrium values. In this paper, we present a locally discontinuous arbitrary Lagrangian Eulerian finite element formulation with temperature discontinuities at interfaces. A penalty-based approach is used to weakly enforce the temperature jump condition. Furthermore, we use kinetic-theory-based Schrage relationship to evaluate the rate of phase change. We apply our methodology to solve the problem of flowing vapor in a planar heat pipe. The flow rates predicted by our continuum simulations are in excellent agreement with recently published molecular dynamics simulation results of the same problem. Interestingly, accounting for temperature discontinuities leads to only a slight improvement in the prediction of mass fluxes as compared to the case when temperature continuity is assumed. This is in contrast to the large improvement in the prediction of temperature profiles and is a consequence of a weak dependence of the evaporation/condensation rates on the vapor temperature.

1. Introduction

Phase change processes such as evaporation, condensation and sublimation are ubiquitous and have found applications in varied disciplines including climatology [1], agriculture [2], astronomy [3], medical therapy [4], thermal management [5] etc. Understanding phase transitions at liquid-vapor interfaces has been the focus of experimental [6,7], numerical [8,9] and theoretical [10,11] studies. The resulting behavior is often analyzed in terms of kinetic models for evaporation/condensation. The first model was formulated over a century ago based on the work of Knudsen and Hertz, resulting in the Hertz-Knudsen relation [12]. This relationship was developed based on thermodynamic and kinetic considerations, and was later modified by Schrage [13] to account for non-zero macroscopic velocity of the vapor which resulted in the Hertz-Knudsen-Schrage relationship.

The validity of the Hertz-Knudsen relation and Schrage relationships is still a subject of active research [14] and debate. Nevertheless, it is generally accepted that during a non-equilibrium evaporation process there is a temperature discontinuity (temperature jump) at the liquid-vapor interface [6]. Both, positive [15] and negative [7] temperature jumps have been observed experimentally. However, most continuum numerical methods for modeling liquid-vapor phase change assume continuity of temperature at phase interfaces [16,17]. Moreover, the value of interfacial temperature often is determined by the saturation pressure of the liquid [18]. In other words, the liquid-vapor interfaces is assumed to be in local thermal equilibrium. Sometimes deviation from equilibrium, $T^l \neq T_{sat}$, is considered while still maintaining temperature continuity across interfaces [19]. In microscopic phase change problems such as droplet evaporation, the assumption of continuity of temperature at the liquid vapor interface might lead to non-negligible errors [20,21].

In this article we introduce a stabilized finite element based continuum numerical method capable of modeling liquid-vapor phase change where we relax the assumption on the temperature

* Corresponding author.
E-mail address: aoberai@usc.edu (A.A. Oberai).

https://doi.org/10.1016/j.ijheatmasstransfer.2019.118597
0017-9310/© 2019 Elsevier Ltd. All rights reserved.

continuity at the interface. In the development of our approach, due to lack of a generalized law for prescribing the temperature jumps at the interface [19,11], we use an approximate law for temperature jump proposed by Liang et al. [8] which was derived based on the results of molecular dynamics (MD) simulations and thermodynamic analysis. Liang et al. also observed that Schrage relationship very accurately describes the evaporation/condensation rates, and presented detailed results on mass and heat flow in the planar heat pipe geometry. Therefore we use the results from Liang et al. as "numerical experiments" to validate our continuum-level framework.

The format of the remainder of this manuscript is as follows. In the next Section we present mathematical formulation of the problem. The model system is described in Section 3, while results and associated discussion are presented in Section 4. Finally, in Section 5 we present summary and conclusions.

2. Mathematical formulation

In this paper, we closely follow the formulation of the locally discontinuous arbitrary Lagrangian Eulerian (ALE) finite element method [22] which was developed for modeling generalized phase change processes. Let Ω^l and Ω^v denote the liquid and vapor phases respectively, and Γ^l the singular interface(s) between them. The Navier-Stokes equations are solved in each phase, Ω^l and Ω^v, and can be expressed as,

$$\boldsymbol{U}_{,t} + (\boldsymbol{F}_i^{adv} - \boldsymbol{F}_i^{diff})_{,i} = \boldsymbol{S}, \tag{1}$$

where $\boldsymbol{U}$ represents the vector of conservative variables for the mass, momentum and energy equations. Flux vector in each Cartesian direction is denoted by $\boldsymbol{F}_i$ with superscripts '*diff*' and '*adv*' representing the diffusive and advective contributions respectively. The source term is indicated by $\boldsymbol{S}$. The subscript '$,t$' denotes a temporal derivative. Subscripts '$,i$' with $i = 1, 2, 3$ denote spatial derivatives; repeated indices imply a summation. The fluxes and conservative variables are defined as,

$$\boldsymbol{U} = \rho \begin{bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e_{tot} \end{bmatrix}, \quad \boldsymbol{F}_i^{adv} = \rho u_i \begin{bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e_{tot} \end{bmatrix} + p \begin{bmatrix} 0 \\ \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \\ u_i \end{bmatrix}, \quad \boldsymbol{F}_i^{diff} = \begin{bmatrix} 0 \\ \tau_{1i} \\ \tau_{2i} \\ \tau_{3i} \\ \tau_{ij}u_j \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -q_i \end{bmatrix}, \tag{2}$$

where p is pressure, ρ is density, $\boldsymbol{u}$ is the flow velocity vector, δ_{ij} is the Kronecker delta, τ_{ij} are components of the viscous stress tensor for a Newtonian fluid, $e_{tot} = e_{int} + \frac{1}{2}u_i u_i$ is the total energy per unit mass, and $q_i = -\kappa T_{,i}$ are components of the heat flux vector with κ being the thermal conductivity.

Conservation laws at the interface can be written as,

$$[\![\boldsymbol{F}_i^{adv} - \boldsymbol{F}_i^{diff}]\!]_i = v_i^I [\![\boldsymbol{U}]\!]_i, \tag{3}$$

where the jump operator, $[\![\cdot]\!]_i$, is defined as,

$$[\![\cdot]\!]_i = (\cdot)^l n_i^l + (\cdot)^v n_i^v. \tag{4}$$

Here $\boldsymbol{n}^l$ is outward normal to the liquid at Γ^l, with $\boldsymbol{n}^v = -\boldsymbol{n}^l$, and v_i^I represents the components of interface velocity. Our prescription of interface velocity is described later in this Section. Superscripts represent the phase – liquid or vapor. To solve these equations, we introduce a 5×1 vector, $\boldsymbol{Y}$, of primitive variables - pressure, components of velocity, and temperature. Introduction of these variables will simplify prescription of certain interfacial conditions.

In addition to the above conservation laws and appropriate boundary conditions, kinematic conditions at the interface are required. A generalized version of this condition can be written as,

$$[\![\mathbb{C}\boldsymbol{Y}]\!]_i = \Theta_i, \tag{5}$$

where the operator $\mathbb{C}$, a 5×5 tensor, is defined as,

$$\mathbb{C} = \begin{bmatrix} 0 & \boldsymbol{0}^T & 0 \\ \boldsymbol{0} & \boldsymbol{I} - \boldsymbol{n}^l \otimes \boldsymbol{n}^l & \boldsymbol{0} \\ 0 & \boldsymbol{0}^T & 1 \end{bmatrix}. \tag{6}$$

Θ_i as,

$$\Theta_i = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \theta_T \end{bmatrix} n_i^l = \theta n_i^l. \tag{7}$$

and the variable $\boldsymbol{Y}$ as,

$$\boldsymbol{Y} = \begin{bmatrix} p \\ u_1 \\ u_2 \\ u_3 \\ T \end{bmatrix}. \tag{8}$$

The tensor $\boldsymbol{I}$ in Eq. (6) is a 3×3 identity tensor.

The left hand side of Eq. (5) evaluates the jump in tangential velocities and temperature across Γ^l while the right hand side prescribes the value of this jump. In general, the 2nd, 3rd and 4th entries of $\boldsymbol{\theta}$ can be non-zero if we have a slip-boundary condition; this effect is not considered in the current study. We do not explicitly control the jump in interfacial pressure, so the LHS and RHS of first row of Eq. (5) is identically zero. θ_T, the fifth component of $\boldsymbol{\theta}$, represents the jump in temperature at the liquid vapor interface as demonstrated next. Expanding the fifth component/row of Eq. (5) results in,

$$\begin{aligned} [\![T]\!]_i &= T^l n_i^l + T^v n_i^v = \theta_T n_i^l \\ \left(T^l - T^v\right) n_i^l &= \theta_T n_i^l \\ \left(T^l - T^v\right) &= \theta_T. \end{aligned} \tag{9}$$

As seen from this illustration, θ_T represents the difference between the liquid and vapor temperatures at the interface. It should be noted that θ_T can be a function of $\boldsymbol{Y}^v$ and $\boldsymbol{Y}^l$ and is not necessarily constant. In this paper we focus on a particular definition of this temperature jump which was proposed by Liang et al. [8]. Finally, to close the set of equations, the following equations of state are utilized – the vapor is assumed to be a perfect gas and the density of liquid is allowed to vary linearly with pressure and temperature. This is discussed in detail in Section 3.

2.1. Prescription of phase change rate

Without an independent expression for the interface velocity ($\boldsymbol{v}^I$) our formulation is incomplete. We define $\boldsymbol{v}^I$ as,

$$\boldsymbol{v}^I = \boldsymbol{u}^l + v_p \boldsymbol{n}^l, \tag{10}$$

where v_p represents the velocity at which the interface moves from the liquid phase into the gas phase because of phase change. In case there is no phase change, $\boldsymbol{v}^I$ will be equal to $\boldsymbol{u}^l$. Using this relation and conservation of mass at the interface, it can be shown that $\boldsymbol{u}^v = \boldsymbol{v}^I$; which is consistent with the fact that discontinuity in normal velocity arises only due to phase change. An expression for mass flux, considering a stationary interface, can be obtained from classical kinetic theory. This expression with some considerations on evaporation/condensation probabilities (accommodation coefficients) and bulk vapor velocity is given by the Schrage Relationship,

$$J = \frac{2\alpha(T^l)}{2-\alpha(T^l)}\sqrt{\frac{k_B}{2\pi m}}\left(\rho_{sat}(T^l)\sqrt{T^l} - \rho^v\sqrt{T^v}\right), \tag{11}$$

where the symbols have the following interpretations: α – mass accommodation coefficient, k_B – Boltzmann constant, m – mass of a molecule/atom, J – mass flux, ρ_{sat} – saturated vapor density. For a more detailed discussion on Schrage Relationships, the reader is referred to [8,12]. The mass flux in the liquid phase in a frame moving with the interface is given by,

$$\rho^l(u_i^l - v_i^l)n_i^l = J \tag{12}$$

Using Eq. (10) in Eq. (12) an expression for v_p (phase change rate) is obtained as

$$v_p = -\frac{J}{\rho^l} \tag{13}$$

2.2. Temperature jump at interfaces

As discussed earlier, θ_T represents the jump in temperature at the interface. In addition, the average interfacial temperature is defined as $\langle T\rangle = (T^l + T^v)/2$. When $|\theta_T|/\langle T\rangle \ll 1$ and if ρ^v is assumed to be equal to $\rho_{sat}(T^v)$, then an approximate expression for θ_T can be obtained [8]. This can be written as,

$$\theta_T = \psi J, \tag{14}$$

where

$$\frac{1}{\psi} = \frac{2\alpha}{2-\alpha}\sqrt{\frac{k_B}{2\pi M\langle T\rangle}}\left(\langle T\rangle \left.\frac{d\rho_{sat}}{dT}\right|_{\langle T\rangle} + \frac{\rho_{sat}(\langle T\rangle)}{2}\right) \tag{15}$$

It should be noted that the above expression provides only a satisfactory estimate of the magnitude of θ_T but captures its direction correctly [8]. Another definition of θ_T was proposed by Beardeaux and Keljstrup [23] based on non-equilibrium thermodynamics, but its applicability is limited by the need for accurate determination of the parameters in their expression.

2.3. Numerical method

We solve the 3D compressible Navier Stokes equations using a stabilized finite element with sharp interfaces separating the different phases. We allow discontinuities in the primitive variables ($\boldsymbol{Y}$) only at the interface (Γ^I); inside each phase (Ω^l and Ω^v) the primitive variables are continuous. At the interface, continuity of tangential velocities and temperature jump is imposed weakly using a penalty method. Details our approach can found in our earlier work [22], and our meshing strategy is described in [24]. In the current paper we modify the original formulation to account for temperature discontinuities at the interface as described in Eq. (5).

3. Problem setup

Our problem of interest is a one-dimensional planar nano-channel consisting of two interfaces. Evaporation occurs at one interface and condensation at the other. This setup is the same as used in the MD simulations reported by Liang et al. [8].

3.1. Geometry and mesh

The entire domain, Ω, can be subdivided into three sub-domains: Ω^{l1}, Ω^{l2} and Ω^v, with $\Omega^{l1} \cup \Omega^{l2} = \Omega^l$, and $\Omega^{l1} \cap \Omega^{l2} = \varnothing$. Total length of the nano-channel is 117 nm; width and height (y and z directions) are both 10 nm. Initial thickness of Ω^{l1} and Ω^{l2} are 6.5 nm and 4.5 nm respectively. A 2D schematic of the problem is shown in Fig. 1. As the simulation progresses, evaporation occurs at Γ^{I1} and condensation at Γ^{I2}. At the leftmost boundary of Ω^{l1} and rightmost boundary of Ω^{l2}, zero velocity and constant temperature ($\mathcal{T}^e$ and $\mathcal{T}^c$) BCs are prescribed. $\mathcal{T}^e$, the source temperature, is the specified temperature of the boundary close to the interface where evaporation occurs (Γ^{I1}). Similarly, $\mathcal{T}^c$ is the sink temperature. On all other boundaries, appropriate BCs are prescribed such that the problem becomes one-dimensional. A complete 3D model of the problem with the mesh is shown in Fig. 2.

3.2. Properties and parameters

The density of the liquid is evaluated using a first order expansion of $\rho(P,T)$ which results in an expression for ρ^l in terms of thermal expansion coefficient (α_P), isothermal compressibility (β_T), and the reference values of temperature, pressure and density. Since the liquid phase velocity is almost zero, the effect of dynamic viscosity is negligible. This suggests that accurate imposition of only κ, c_p, and ρ is important. Numerical values of these parameters are obtained from MD studies that use Lennard Jones' potential to model liquid argon. MD simulations were performed with the exact potential parameters used by Liang et al. and good agreement was found between our simulations and the values reported in literature. We use $c_p \approx c_v$ and set the value of α_P to be an order higher than the reported values. The variation of temperature in the liquid is small and we are interested in the steady state behavior of the system, so the assumptions made on these properties are not expected to cause any significant anomalies in the response of the system. All other parameters are obtained from MD/experimental works as convenient.

The argon vapor is modeled as a calorically perfect gas with $\gamma = 1.67$. The exact values of κ^v and μ^v are not deemed to be critical in the current problem of interest as in the steady state no gradients in temperature and velocity are expected in the gas. So, as in the case of the liquid, these values are obtained from MD/experiments as convenient.

All the physical parameters are shown in Table 1. It will be shown later in this manuscript that despite these 'modeling' assumptions, our simulations are able to replicate MD simulations quite accurately.

3.3. Accommodation coefficient and saturation curve

As described in the Section 2, mass accommodation coefficient (α) and saturation density ($\rho_{sat}(T^l)$) are parameters in Eqs. (11) and (15). The mass accommodation coefficient can be defined as the ratio of vapor molecules condensing at a liquid-vapor interface to the total number of vapor molecules colliding with this interface. Limiting values of this accommodation coefficient signify specular ($\alpha = 0$, no phase change) and diffuse ($\alpha = 1$) collisions. The variation of α with temperature can be attributed to the change in nature of collisions with increasing thermal energy of the molecules. Typical values of these coefficients are between the limits (0 to 1), and are often treated as fitting parameters in experimental studies [7]. However, using MD simulations, these coefficients can be evaluated precisely. Both, saturated vapor den-

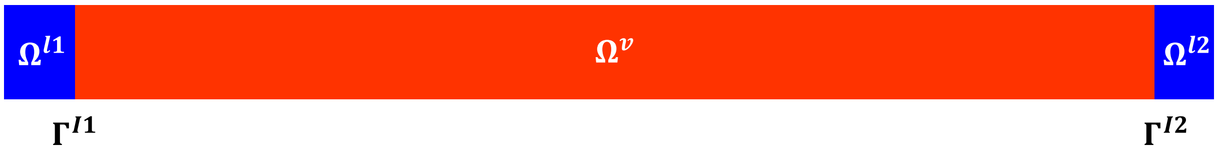


Fig. 1. 2D schematic of the considered problem. Ω^{l1} and Ω^{l2} are the two liquid (blue) regions, Ω^{v} is the vapor (red) region. Γ^{l1} (evaporation) and Γ^{l2} (condensation) are the liquid-vapor interfaces. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

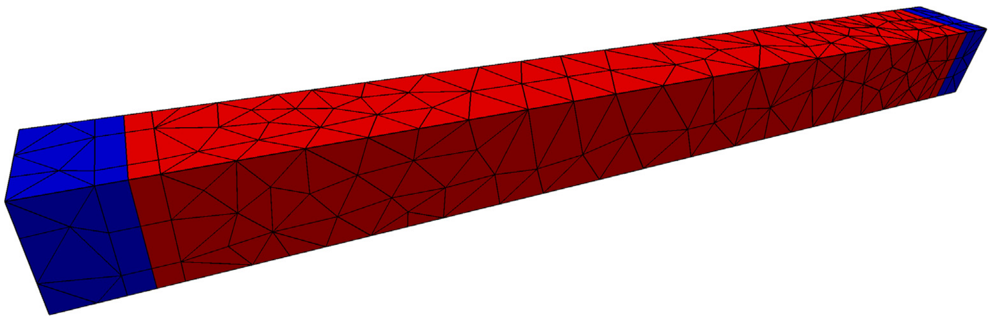


Fig. 2. 3D model of the problem with the mesh. We use hybrid meshes: layered mesh with wedges at interfaces and tetrahedrons elsewhere. A combination of mesh motion and mesh adaptation is used to account for the moving interface.

Table 1
Material properties for two phases.

Property		Liquid	Vapor	Units
Molecular weight	M_w	–	0.039948	kg/mol
Density	ρ	1421 [25]	$\sim$ 5 (85 K, 1 atm)	kg/m^3
Heat capacity	c_v	1160 [26–28]	$\frac{R}{\gamma-1}$	J/kg·K
Specific heat ratio	γ	–	1.67	
Molecular viscosity	μ	2.5e−3 [29,30]	7e−6 [31]	N·s
Thermal conductivity	κ	0.13 [25]	5.5e−3 [31]	W/m·K
Thermal expansion coeff.	α_p	1e−4 [32]	$1/T$	1/K
Isothermal compressibility	β_T	3e−9 [32]	$1/p$	1/Pa
Internal Energy diff.	Δe_{gf}	144 (87.178 K, 1 atm) [28]	–	kJ/kg

sities and accommodation coefficients for this specific fluid were obtained by Liang et al. using MD simulations. Figs. 3 and 4 show the variation of these parameters with T^l (values from Liang et al. [8]) and polynomial fits to these data points are used as inputs to continuum simulations.

4. Results and discussion

4.1. Pressure, temperature, velocity and density profiles

Two interfacial kinematic conditions on temperature considered here are: (a) Continuous temperature, $T^l = T^v$ (b) Discontinu-

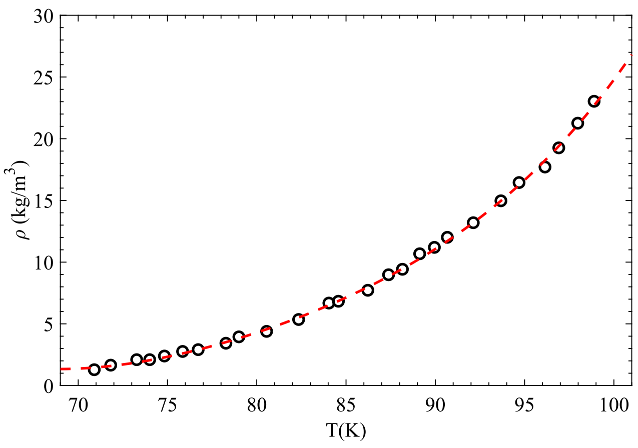


Fig. 3. Density of saturated vapor as a function of temperature. The black circles represent data from Liang et al. [8] and the red dashed line is a 4th order polynomial fit to that data. This polynomial can be expressed as, $\rho = 0.0000T^4 - 0.0072T^3 + 0.8732T^2 - 47.5221T + 972.7469$ (coefficients are correct up to 4 decimal places).

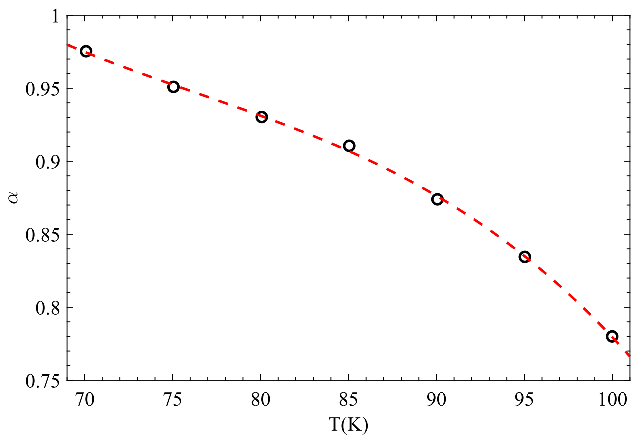


Fig. 4. Mass accommodation coefficient (α) vs. temperature. A 3rd order polynomial fit to the data from Liang et al. [8] (black circles) is indicated by the red dashed line. The polynomial takes the form, $\alpha = -0.0000T^3 + 0.0012T^2 - 0.0943T + 3.5633$, with coefficients correct up to the 4th decimal point.

ous temperature, where the magnitude and direction of the jump is determined by Eq. (14). Results of part (a) are shown in Fig. 6, and part (b) in Fig. 5. MD data reported in Liang et al. was averaged region(bin)-wise over the data collection period (8 ns) and therefore, the exact location of the interface cannot be determined from the data reported by them. We compare their averaged data with a specific timestep in our continuum simulations which is representative of the time instant from MD simulations. Fig. 5 shows a comparison of MD and continuum results – a very good agreement can be observed for the velocity and density profiles. The minor differences in the temperature profile can be attributed to the approximate temperature jump condition as discussed in Section 2. Furthermore, the error in pressure profile is likely associated with two factors; (i) error in temperature and (ii) the perfect gas law for the vapor phase equation of state. To corroborate the latter assumption, we note that under standard conditions Argon gas is well characterized by the ideal gas law. We refrain from considering more complicated equation of states because this simple gas law gives predicts mass fluxes with good accuracy, as shown later. At later times, the trends in the profiles remain the same with only minor changes in the magnitude of the variables.

The temperature continuity assumption (Fig. 6) produces a temperature profile which is quite different from those observed in MD simulations. Interestingly, other variables are not affected significantly by this assumption. As before, at later times, only minor deviations are observed as compared to early times. The reason behind this rather minor effect of the temperature continuity assumption is in the fact that the phase change rate in Schrage formula is mostly driven by the difference between saturation and actual vapor pressure, and is only weakly sensitive to the temperature changes as long as the changes are small relative to the absolute temperature.

In addition to discontinuities at liquid-vapor interfaces, a jump in temperature of about $\sim 1K$ was observed at the solid-liquid interface (Fig. 1 in [8]). Accounting for this boundary slip is necessary as it affects the liquid temperature, saturation pressures and thus, driving forces. To mimick the effect of the temperature slip at solid-liquid interfaces in our quasi-steady continuum simulations, we set the temperatures of the source ($\mathcal{T}_e$) and sink ($\mathcal{T}_c$) to be 91.36 K and 78.77 K respectively, instead of 92.5 K and 77.5 K as used in [8].

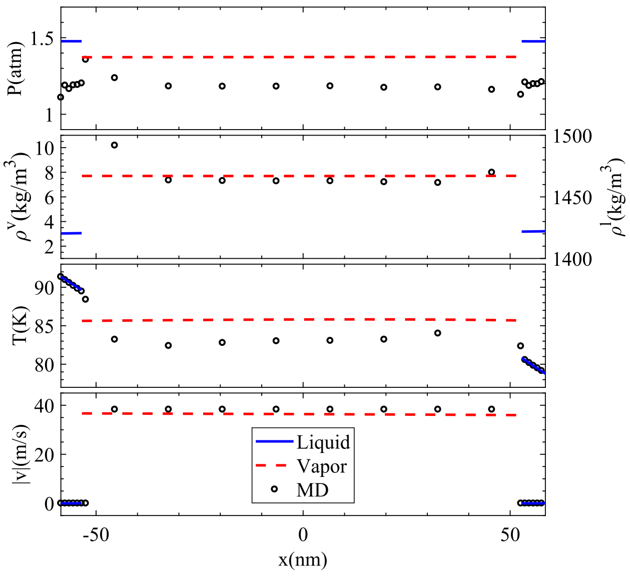


Fig. 5. Profiles of velocity, pressure, temperature and density at 7 ns when continuity of temperature is not assumed at the interface. The open black circles correspond to MD simulations performed by Liang et al.

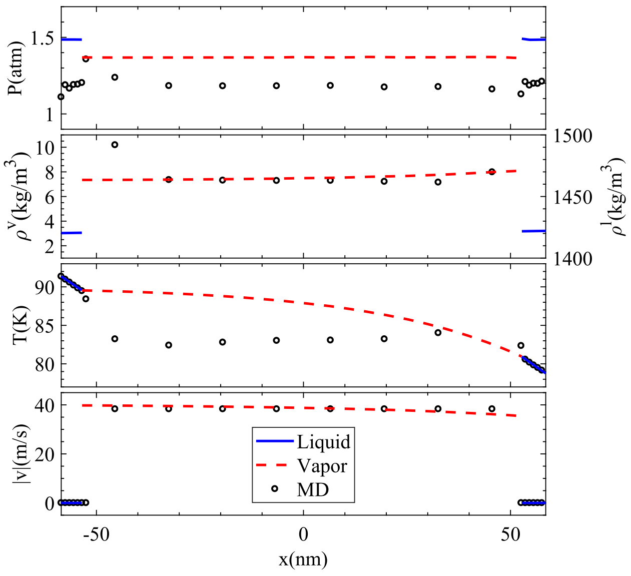


Fig. 6. Velocity, pressure, temperature and density profiles at 7 ns when continuity of temperature is assumed at the interface. At later times, similar trends are observed.

4.2. Mass fluxes

We commence our simulations from an artificial initial condition, so the initial transients ($t < 5ns$) are not considered in our analysis. Variation of the vapor phase mass flux at the two interfaces with time is shown in Fig. 7. Our simulations enter a quasi-steady regime only after $t = 5ns$ and mass fluxes vary almost linearly with time beyond this time instant. Liang et al. used a different initial condition and their simulations entered the quasi-steady regime earlier ($t = 3ns$). Therefore, a precise direct comparison with MD results in the transient regime is not possible. Furthermore, the continuum formulation during transient might be not as reliable due to the assumption on heat capacities and moderately large timesteps ($CFL_{max} = max\left\{CFL^l, CFL^v\right\} \sim 25$ with $\Delta t = 0.1$ ns). $CFL^{l/v}$ is defined as $\frac{c^{l/v}\Delta t}{\Delta x^{l/v}}$, where, $c^{l/v}$ is the speed of

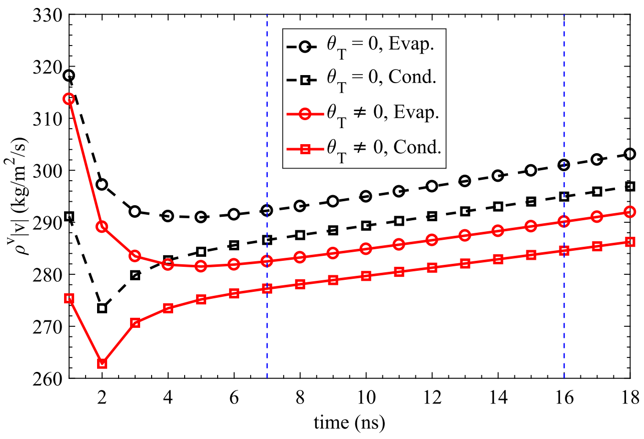


Fig. 7. Vapor phase mass flux at the two, evaporating (circles) and condensing (squares), interfaces as a function of time. The black dashed lines correspond to the continuous temperature assumption and the red solid lines represent cases when a temperature discontinuity is imposed using Eq. (14). The vertical blue lines correspond to the time instances where comparisons are done with MD. Due to the initial conditions some artificial transients are observed in the profiles which vanish as the system enters the quasi-steady regime at about 5 ns. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Comparison of vapor phase mass fluxes obtained from MD and Continuum simulations. All fluxes are in kg/m^2/s.

θ_T	J_{md}	t = 7 ns				t = 16 ns			
		J_e	err. (%)	J_c	err. (%)	J_e	err. (%)	J_c	err. (%)
$\theta_T \neq 0$	278.44	281.87	1.23	277.25	−0.43	290.11	4.19	284.55	2.19
$\theta_T = 0$		292.23	4.95	286.60	2.93	300.98	8.09	294.94	5.93

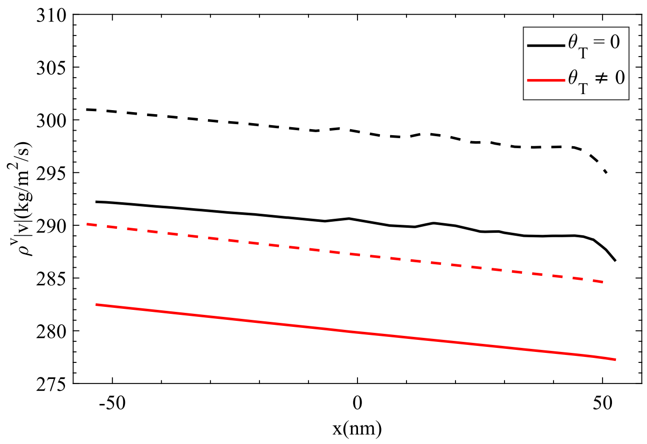


Fig. 8. Variation of mass flux in the vapor phase. The lines in this figure have the following meanings: solid lines - profiles at 7 ns, dashed lines - profiles at 16 ns, red lines - discontinuous temperature, black lines - continuous temperature. A minor variation in the mass flux along the channel is observed due the quasi-steady nature of the problem. The dashed lines (16 ns) are left-shifted as compared to the solid lines (7 ns) because of sustained evaporation at the left interface and condensation at the right. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sound in the liquid/vapor phase at $\sim 85K, \Delta t$ is the timestep used in our simulations, and $\Delta x^{l/v}$ is the minimum mesh size in liquid/vapor domain. Also, the magnitude of temperature slip at the boundary is unknown in the transient regime. Therefore, we just focus on comparison of our results with MD in the quasi-steady state. Liang et al. reports only one (temporally and spatially) averaged value of the mass flux with a time-averaging interval of 8 ns starting at $t = 3$ ns. So the representative time instant for MD results is 7 ns. Comparison of continuum results with MD is done at $t = 7$ ns and $t = 16$ ns (close to the end of the continuum simulation), as shown in Table 2. At $t = 7ns$ the continuum model accounting for the temperature discontinuity at liquid-vapor interface yields remarkably good agreement with the results of MD simulations (see Table 2).

As seen in Fig. 7, the evaporation flux is always higher than the condensation flux; so a build-up of density and pressure with time is expected. This is consistent with another observation – the variation of mass flux in the vapor phase along the length of the channel, plotted in Fig. 8. Both effects arise due to quasi-steady flows. The temperature continuity assumption yields higher mass fluxes as compared to the discontinuous temperature case, and also has larger errors when compared to MD simulations (refer Table 2); however, even in this case the errors are limited to about 5%. This implies that the evaporation/condensation mass fluxes can be predicted in continuum simulations with good accuracy using the Schrage relationship even if no discontinuity in temperature is considered at the interface. Of course, the temperature profile will be inaccurate.

5. Summary and conclusions

We present a stabilized finite element based method capable of modeling liquid-vapor interfaces out of local thermal equilibrium. Using this method and a physically motivated prescription of temperature discontinuity at the interface, we show that the assumption of the temperature continuity does not result in large variations in mass fluxes, even though the dissimilarities in temperature profiles is significant. This originates from the fact that the evaporation/condensation processes in our simulations of a heat pipe with planar geometries are predominantly driven by the difference between the actual and saturation vapor density. This strongly implies that the equilibrium relationship between interfacial temperature and vapor density at the interface should to be relaxed.

Our continuum is capable of accurately predicting evaporation/-condensation rates in quasi-steady flows when compared against “numerical MD experiments”. In particular, with the temperature continuity assumption, relative error in mass fluxes is $< 5\%$, and with temperature discontinuity at the interface the error is under 2%.

Although we use the Schrage relation as the “phase change law”, this methodology is capable of accommodating any other equation. Similarly, more accurate versions of the interfacial temperature discontinuity relation can be easily used. This method can be applied and extended to model more complicated systems such as collapsing vapor bubbles, spray cooling etc. where the effect of temperature discontinuities and local thermal equilibrium at the liquid vapor interface can be investigated.

Acknowledgement

This work was supported by the Office of Naval Research Thermal Science Program, Award No. N00014-17-1-2767, and U.S. Army grants W911NF1410301 and W911NF16C0117. We are grateful to Dr. Zhi Liang (California State University, Fresno) for providing us the data of his molecular dynamics simulations.

References

[1] A. Kay, H. Davies, Calculating potential evaporation from climate model data: a source of uncertainty for hydrological climate change impacts, J. Hydrol. 358 (3–4) (2008) 221–239.
[2] J.D. Valiantzas, Simplified versions for the penman evaporation equation using routine weather data, J. Hydrol. 331 (3–4) (2006) 690–702.
[3] F.M. Richter, Timescales determining the degree of kinetic isotope fractionation by evaporation and condensation, Geochim. Cosmochim. Acta 68 (23) (2004) 4971–4992.
[4] K. Sefiane, On the formation of regular patterns from drying droplets and their potential use for bio-medical applications, J. Bionic Eng. 7 (2010) S82–S93.
[5] S.G. Kandlikar, Z. Lu, Thermal management issues in a pemfc stack–a brief review of current status, Appl. Therm. Eng. 29 (7) (2009) 1276–1280.
[6] G. Fang, C. Ward, Temperature measured close to the interface of an evaporating liquid, Phys. Rev. E 59 (1) (1999) 417.
[7] V. Badam, V. Kumar, F. Durst, K. Danov, Experimental and theoretical investigations on interfacial temperature jumps during evaporation, Exp. Therm. Fluid Sci. 32 (1) (2007) 276–292.
[8] Z. Liang, T. Biben, P. Keblinski, Molecular simulation of steady-state evaporation and condensation: validity of the schrage relationships, Int. J. Heat Mass Transf. 114 (2017) 105–114.
[9] R. Hołyst, M. Litniewski, D. Jakubczyk, A molecular dynamics test of the Hertz–Knudsen equation for evaporating liquids, Soft Matter 11 (36) (2015) 7201–7206.
[10] M. Bond, H. Struchtrup, Mean evaporation and condensation coefficients based on energy dependent condensation probability, Phys. Rev. E 70 (6) (2004) 061605.

[11] H. Struchtrup, S. Kjelstrup, D. Bedeaux, Analysis of temperature difference driven heat and mass transfer in the Phillips–Onsager cell, Int. J. Heat Mass Transf. 58 (1–2) (2013) 521–531.
[12] A.H. Persad, C.A. Ward, Expressions for the evaporation and condensation coefficients in the Hertz-Knudsen relation, Chem. Rev. 116 (14) (2016) 7727–7767.
[13] R.W. Schrage, A Theoretical Study of Interphase Mass Transfer, Columbia University Press, 1953.
[14] A.P. Polikarpov, I. Graur, E.Y. Gatapova, O. Kabov, Kinetic simulation of the non-equilibrium effects at the liquid-vapor interface, Int. J. Heat Mass Transf. 136 (2019) 449–456.
[15] E.Y. Gatapova, I.A. Graur, O.A. Kabov, V.M. Aniskin, M.A. Filipenko, F. Sharipov, L. Tadrist, The temperature jump at water–air interface during evaporation, Int. J. Heat Mass Transf. 104 (2017) 800–812.
[16] C.R. Kharangate, I. Mudawar, Review of computational studies on boiling and condensation, Int. J. Heat Mass Transf. 108 (2017) 1164–1196.
[17] F. Gibou, L. Chen, D. Nguyen, S. Banerjee, A level set based sharp interface method for the multiphase incompressible Navier–Stokes equations with phase change, J. Comput. Phys. 222 (2) (2007) 536–555.
[18] S.W. Welch, J. Wilson, A volume of fluid based method for fluid flows with phase change, J. Comput. Phys. 160 (2) (2000) 662–682.
[19] D. Juric, G. Tryggvason, Computations of boiling flows, Int. J. Multiph. Flow 24 (3) (1998) 387–410.
[20] B.-Y. Cao, J.-F. Xie, S.S. Sazhin, Molecular dynamics study on evaporation and condensation of n-dodecane at liquid–vapor phase equilibria, J. Chem. Phys. 134 (16) (2011) 164309.
[21] S. Hardt, F. Wondra, Evaporation model for interfacial flows based on a continuum-field representation of the source terms, J. Comput. Phys. 227 (11) (2008) 5871–5895.
[22] Y. Zhang, A. Chandra, F. Yang, E. Shams, O. Sahni, M. Shephard, A.A. Oberai, A locally discontinuous ale finite element formulation for compressible phase change problems, J. Comput. Phys. 393 (2019) 438–464.
[23] D. Bedeaux, S. Kjelstrup, Transfer coefficients for evaporation, Phys. A: Stat. Mech. Appl. 270 (3–4) (1999) 413–426.
[24] F. Yang, A. Chandra, Y. Zhang, S. Tendulkar, R. Natasia, A.A. Oberai, M. Shephard, O. Sahni, A parallel interface-tracking approach for moving boundary problems, Eng. Comput. (submitted for publication).
[25] A. McGaughey, M. Kaviany, Thermal conductivity decomposition and analysis using molecular dynamics simulations. Part i. Lennard-Jones argon, Int. J. Heat Mass Transf. 47 (8–9) (2004) 1783–1798.
[26] A. Granato, The specific heat of simple liquids, J. Non-crystal. Solids 307 (2002) 376–386.
[27] S. Nosé, A molecular dynamics method for simulations in the canonical ensemble, Mol. Phys. 52 (2) (1984) 255–268.
[28] C. Tegeler, R. Span, W. Wagner, A new equation of state for argon covering the fluid region for temperatures from the melting line to 700 k at pressures up to 1000 mpa, J. Phys. Chem. Ref. Data 28 (3) (1999) 779–850.
[29] S.H. Lee, J. Kim, Molecular dynamics simulation study of transport properties of diatomic gases, Bull. Korean Chem. Soc. 35 (12) (2014) 3527–3531.
[30] E. Kirova, G. Norman, Viscosity calculations at molecular dynamics simulations 653 (1) (2015) 012106.
[31] H.J. Hanley, the viscosity and thermal conductivity coefficients of dilute argon, krypton, and xenon, J. Phys. Chem. Ref. Data 2 (3) (1973) 619–642.
[32] W. Streett, L. Staveley, Experimental study of the equation of state of liquid argon, J. Chem. Phys. 50 (6) (1969) 2302–2307.

Journal of Computational Physics 393 (2019) 438–464

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A locally discontinuous ALE finite element formulation for compressible phase change problems

Yu Zhang [a], Anirban Chandra [a], Fan Yang [a], Ehsan Shams [a], Onkar Sahni [a], Mark Shephard [a], Assad A. Oberai [b,*]

[a] Scientific Computation Research Center (SCOREC), Rensselaer Polytechnic Institute, Troy, NY 12180, United States
[b] Aerospace and Mechanical Engineering, University of Southern California, Los Angeles, CA 90089, United States

A R T I C L E   I N F O

Article history:
Received 4 September 2018
Received in revised form 11 March 2019
Accepted 15 April 2019
Available online 2 May 2019

Keywords:
Multiphase
Finite element method
Discontinuous interpolations
ALE
Compressible

A B S T R A C T

The simulation of phase change processes that occur at high rates, like the collapse of a vapor bubble or the combustion of dense energetic materials, poses significant challenges that include strong discontinuities in select field variables at the interface, high-speed flows in at least one phase, significant role of compressibility, disparate phase rate and acoustic time scales and advection dominated processes. In this paper we present a finite element based method that addresses these challenges. The discretization is continuous everywhere except at the interface and it inherits its stability properties from both continuous and discontinuous finite element formulations like the SUPG and interior penalty methods. We track the evolution of the interface mesh and accommodate its motion in the volume by moving the mesh in accordance with an elastic analogy within an arbitrary eulerian lagrangian (ALE) framework. This motion is interspersed with a few select steps of mesh modification. We demonstrate that the proposed method has desirable discrete conservation properties and outline a proof for its stability. We also describe how this method is implemented in a finite element code within an implicit predictor-corrector time stepping scheme. Finally we apply this method to a series of phase change problems involving an energetic material, where we verify its implementation and demonstrate its utility.

1. Introduction

Multiphase processes are ubiquitous in science and engineering. Examples include bubbly flows in oceans, rivers and streams, flows around ships and submarines, flows in the thermo-hydraulic components of nuclear reactors, and other processes involving evaporation, condensation and sublimation. Multiphase processes necessarily involve a dynamically evolving interface between two phases. In order to accurately simulate these processes, particularly when there are physically processes occurring at the interface, it is necessary to model the evolution of this interface and to accurately predict the transport of mass, momentum and energy across it. For this reason computational methods that attempt to simulate multiphase processes tend focus on these two aspects of multiphase flows.

Several techniques have been developed, refined and applied to modeling the evolution of the interface. These include volume of fluid methods [36,28,48,1], phase-field methods [4,7,24], level-set based methods [47,46,33,34] and interface

* Corresponding author.
E-mail address: aoberai@usc.edu (A.A. Oberai).

https://doi.org/10.1016/j.jcp.2019.04.039
0021-9991/© 2019 Elsevier Inc. All rights reserved.

tracking methods [52,14,51,8]. Each type comes with its own advantages and disadvantages, and we refer the reader to the excellent articles cited above that describe these methods and their characteristics. In this manuscript we work with an interface-tracking method where we discretize the interface with a surface mesh and track its evolution in time. We note that our focus is on the techniques used to simulate the transport of mass momentum and energy across the interface and not on the method used for modeling its evolution. Thus the techniques developed in this paper could be used with other interface modeling approaches including the level set method.

It is logical to divide multiphase problems into those with phase change and those without. In processes without phase change, the interface and each phase move with the same velocity at the interface; whereas in processes with phase change all three velocities are different. This leads to the obvious challenge of keeping track of three different velocities at the interface. It also leads to other challenges including:

1. The requirement of modeling strong discontinuities in pressure, density and the normal component of material velocity at the interface, while at the same time enforcing continuity in temperature and the tangential components of material velocity.
2. In phase change processes with large density differences or high rates of phase change (the phase change of a liquid or a solid propellant, for example), phase change leads to very high speeds in the lighter phase. This in turn leads to compressible behavior in that phase and means that it must be modeled as a compressible continuum.
3. The inclusion of compressibility leads to another challenge which stems from the fact that the ratio of the speed of sound in either phase to that of the phase change velocity is typically very large. This means that in order to model an appreciable amount of phase change it is necessary to simulate over a long time period relative to the time it takes for an acoustic wave to traverse the computational domain. In other words, it leads to a problem with multiple temporal scales where we are interested in the long-term behavior of a system with significant small time-scale dynamics.
4. Finally, since the speed in one or both phases is large, these problems tend to be convection dominated, which poses challenges to the stability of the numerical technique.

In this manuscript we propose and implement a finite element based method that addresses these challenges. We allow for discontinuity in some variables at the interface by using discontinuous basis functions only at the interface. We restrict these basis functions only to the interface, and use C^0 finite element basis functions elsewhere, in order to keep the total number of degrees of freedom under control. The mesh at the interface, and therefore the discontinuous basis functions, are constrained to move with the interface velocity. Everywhere else in the domain, the motion of the mesh is determined by solving a linear elastic problem. The effect of the moving mesh is captured in the weak formulation by posing the equations of motion and thermodynamics in a arbitrary Lagrangian Eulerian (ALE) framework [20,27,18].

The interface terms associated with our weak form are motivated by a discontinuous Galerkin formulation for the scalar advection diffusion problem [3,6,21]. Due to the use of discontinuous interpolations we are able to prove certain discrete conservation properties for the resulting method for each phase. The interface terms contain contributions that enforce the appropriate jump in the fluxes at the interface and the continuity of temperature and the tangential components of material velocity. They also include a stabilizing term that is motivated by the interior penalty method [2,3]. The volumetric terms, defined over each phase, contain the streamline-upwind Petrov Galerkin (SUPG) stabilization [19]. We solve the nonlinear ordinary differential equations resulting from the finite element discretization of the conservation laws and mesh motion using the Generalized-alpha method [25]. This leads to a staggered predictor-corrector approach within each time step and permits the use of implicit second-order time schemes which retain stability at very large values of the CFL number (≈ 20). This feature is essential for our application, since interface evolves very slowly when compared with the propagation of an acoustic wave.

When compared with incompressible phase change problems, there is a relatively small body of work dedicated to the simulation of compressible phase change problems. Among these, a significant proportion make use of multi-fluid or volume-of-fluid approaches, where multiple fluids are assumed to exist at any given spatial location (see for example [10,54, 40,13] and references therein). The method developed in this manuscript is different in that at a given spatial location only one phase is allowed to exist. A similar approach is developed in [31], where the authors have developed a finite volume method with a local Riemann solver in conjunction with a cut-cell method and applied it to the simulation of several interesting two-dimensional problems including that of bubble collapse. In contrast to this, in this manuscript, instead of a finite volume method we use a finite element method with a discontinuous basis that is confined to the interface, and instead of a cut-cell method we use an interface-fitted mesh that moves with the interface. The mesh around the interface is modified, when due to excessive mesh motion, its quality is no longer satisfactory. This modification step is triggered occasionally only as needed, thus most of the motion of the interface is accommodated by moving the mesh.

The remainder of this paper is as follows. In Section 2, we present the mathematical equations that describe the compressible phase change problem. We present both the strong and the weak form of these equations, with a special emphasis on the equations at the interface. We also examine the conservation properties of the finite element discretization based on these equations and demonstrate it stability in the context of a simple model problem. In Section 3, we discuss some of the computational aspects of the proposed formulation. These include our treatment of the discontinuities at the interface, mesh motion, and the time integration scheme for the problem. In Section 4, we apply the computational method to solve problems of interest. These include a simple phase change Stefan problem, where we can verify the numerical solution, a

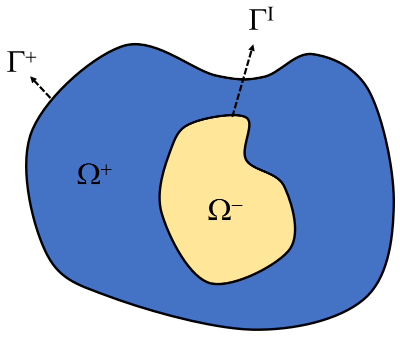


Fig. 1. Schematic diagram of the problem of interest.

spherical counterpart of this problem, and problems where we model phase change of single and six grains of a dense, energetic material enclosed in an adiabatic chamber. We end with conclusions in Section 5.

2. Mathematical model

Our mathematical model comprises of the compressible Navier-Stokes equations in the domain occupied by each phase (Ω^{+} and Ω^{-}) and two sets of interface conditions (see Fig. 1). The first is obtained by imposing conservation laws at the interface between the phases, while the second is obtained by requiring the continuity of certain field variables at the interface. In the following section, we first describe the strong form of these equations. Thereafter we present the weak from that leads to a finite element method.

2.1. Conservation laws

The Navier-Stokes equations can be written in the general form of an advective-diffusive system with a source term as

$$\boldsymbol{U}_{,t} + \boldsymbol{F}_{i,i} = \boldsymbol{S}, \tag{1}$$

where $\boldsymbol{U}$ is the vector of conservative variables, $\boldsymbol{F}_i = \boldsymbol{F}_i^{adv} - \boldsymbol{F}_i^{diff}$ are the flux vectors, where the superscripts *adv* and *diff* denote the advective and diffusive parts, respectively, and $\boldsymbol{S}$ is the source term. The subscripts "$,i$" with $i = 1, 2, 3$ denote spatial derivatives, where the repeated index implies a summation, and the subscript "$,t$" denotes a temporal derivative.

The conservative variables and the fluxes are defined as,

$$\boldsymbol{U} = \rho \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e_{tot} \end{Bmatrix} \tag{2}$$

$$\boldsymbol{F}_i^{adv} = \rho u_i \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e_{tot} \end{Bmatrix} + p \begin{Bmatrix} 0 \\ \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \\ u_i \end{Bmatrix} \tag{3}$$

$$\boldsymbol{F}_i^{diff} = \begin{Bmatrix} 0 \\ \tau_{1i} \\ \tau_{2i} \\ \tau_{3i} \\ \tau_{ij} u_j \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -q_i \end{Bmatrix}, \tag{4}$$

where ρ is density, $\boldsymbol{u}$ is the flow velocity vector, p is pressure, τ_{ij} are components of the viscous stress tensor, δ_{ij} is the Kronecker delta, e_{tot} is the total energy per unit mass, and $q_i = -\kappa T_{,i}$ are components of the heat flux vector.

The equations above are derived from the conservation of mass and energy and the balance of linear momentum for each phase. Thus they are valid in Ω^{+} and in Ω^{-} the two distinct volumes occupied by the two phases. At the interface between these phases, denoted by Γ^{I}, the same conservation laws lead to the following jump condition [9],

$$[\![\boldsymbol{F}_i]\!]_i = v_i [\![\boldsymbol{U}]\!]_i. \tag{5}$$

Here the jump operator $[\![\cdot]\!]_i$ is defined as,

$$[\![\cdot]\!]_i = (\cdot)^+ n_i^+ + (\cdot)^- n_i^-, \tag{6}$$

where, the superscripts $+$ and $-$ denote two phases, and n_i are the components of the outward normal vector at the interface. We note that $\boldsymbol{n}^+ = -\boldsymbol{n}^-$. Further, in equation (5) $\boldsymbol{v}$ is the interface velocity, expressed as,

$$\boldsymbol{v} = \boldsymbol{u}^- + u_p \boldsymbol{n}^+, \tag{7}$$

where the speed u_p is the speed at which the interface retreats from the $+$ phase into the $-$ phase (denser) due to phase change. This speed, which is a direct measure of the rate of phase change, is determined by the local thermodynamic variables on either side of the interface. The expression for this speed can be viewed as a constitutive equation that must be supplied in addition to the jump conditions described above. In this manuscript we have simulated problems where this speed is held constant, as well as problems where it is determined by the gas pressure.

In order to understand the jump conditions implied by (5) it is instructive to consider the first of the five equations implied by it,

$$\begin{aligned}(\rho u_n)^+ - (\rho u_n)^- &= (\rho^+ - \rho^-)(u_n^- + u_p)\\ \Rightarrow \rho^+(u_n^+ - u_n^-) &= (\rho^+ - \rho^-)u_p,\end{aligned} \tag{8}$$

where u_n denotes the velocity component along $\boldsymbol{n}^+$. Equation (8) makes it clear that when the interface is evolving due to phase change, that is $u_p \neq 0$, there is a discontinuity in the normal component of velocity. Further, the magnitude of this jump increases as the difference in the density between the two phases grows. For $\frac{\rho^+}{\rho^-} = 0.01$, this difference is $99 \times u_p$. Thus even if u_p is relatively small, this difference can be large. We note that a similar discontinuity exists in the pressure across the interface.

The phase change process is marked by changes in the molecular structure of the material which lead to changes in its thermodynamic properties and behavior. This includes sharp changes in density and pressure, as well as changes in the equations of state used for the two phases. The rate of phase change is determined by the thermodynamic variables at the interface, and in the examples considered in this paper, it is dependent only on the pressure of the vapor phase. We also assume that phase change is in local thermal equilibrium which ensures a continuous temperature at the interface. In many microscale applications this assumption may be violated and a jump in temperature might be observed. We note that this scenario can be easily accommodated with the approach developed in this manuscript.

Along with the jump conditions, we also need to enforce the “kinematic conditions” at the interface that stipulate the continuity of temperature and tangential velocity across the interface. For the primitive variable set,

$$\boldsymbol{Y} = \left\{ \begin{array}{c} p \\ u_1 \\ u_2 \\ u_3 \\ T \end{array} \right\}, \tag{9}$$

where p and T denote pressure and temperature, respectively, these equations may be written as

$$[\![\,\mathbb{C}\boldsymbol{Y}]\!]_i = \boldsymbol{0}, \tag{10}$$

where the operator $\mathbb{C}$ is defined as

$$\mathbb{C} = \begin{bmatrix} 0 & \boldsymbol{0}^T & 0 \\ \boldsymbol{0} & \boldsymbol{I} - \boldsymbol{n}^+ \otimes \boldsymbol{n}^+ & \boldsymbol{0} \\ 0 & \boldsymbol{0}^T & 1 \end{bmatrix}. \tag{11}$$

The diagonal sub-block in the center of the operator $\mathbb{C}$ projects out the normal component of the velocity at the interface and ensures the continuity of the tangential component. Further, the number 1 in the diagonal ensures the continuity in temperature at the interface. We note that since $\mathbb{C}$ is quadratic in the normal vector, its value on either side of the interface is the same. That is we could have used $\boldsymbol{n}^-$ in the equation above without changing the definition of $\mathbb{C}$. In summary the equations we wish to solve are given by (1), (5) and (10).

2.2. Equations in an arbitrary frame

In the approach described in this manuscript the mesh moves with the interface (interface tracking approach). Thus we wish to compute an equation for the rate of change of variables in a frame that is moving with an arbitrary velocity $\tilde{\boldsymbol{u}}$ with respect to the fixed frame [20]. We denote the time derivative in this frame with the subscript “$,\tilde{t}$,” and note that

$$\boldsymbol{U}_{,\tilde{t}} = \boldsymbol{U}_{,t} + \tilde{u}_i \boldsymbol{U}_{,i}. \tag{12}$$

Using this in (1) we arrive at

$$\boldsymbol{U}_{,t} + \tilde{\boldsymbol{F}}_{i,i} + \tilde{u}_{i,i}\boldsymbol{U} = \boldsymbol{S}, \tag{13}$$

where the total flux is defined as before as the sum of advective and diffusive fluxes, that is $\tilde{\boldsymbol{F}}_i = \tilde{\boldsymbol{F}}_i^{adv} - \boldsymbol{F}_i^{diff}$; however the advection is relative to the velocity of the moving frame and is given by

$$\tilde{\boldsymbol{F}}_i^{adv} = \rho\left(u_i - \tilde{u}_i\right)\begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e_{tot} \end{Bmatrix} + p\begin{Bmatrix} 0 \\ \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \\ u_i \end{Bmatrix}. \tag{14}$$

Assuming that the frame moves with the interface, that is on Γ^I, $\tilde{\boldsymbol{u}} = \boldsymbol{v}$, we may move the term on the right hand side of (5) to the left hand side, and write this condition in terms of the modified flux,

$$[\![\tilde{\boldsymbol{F}}_i]\!]_i = \boldsymbol{0}. \tag{15}$$

Thus the equations that we wish to solve in a frame that is arbitrary, up to the restriction that it moves with the interface, are given by (13), (15) and (10).

2.3. Weak form

We wish to solve the equations for each phase, and the interface conditions using the finite element method. For this we work with a weak formulation of the original differential equations. We derive the weak form for the domain depicted in Fig. 1. On the outer boundary of this domain for simplicity we assume that the outward fluxes are prescribed, that is

$$\tilde{\boldsymbol{F}}_i n_i^+ = \boldsymbol{H}, \text{ on } \Gamma^+. \tag{16}$$

In the weak form, which is posed for the primitive variables, $\boldsymbol{Y}$, we work with functions that are continuous over the domain occupied by each phase, and are discontinuous across the interface. The space of weighting functions is given by

$$\mathcal{V} = \left\{\boldsymbol{V} \,|\, \boldsymbol{V} \in H^1(\Omega^+)^3, \boldsymbol{V} \in H^1(\Omega^-)^3, \boldsymbol{V} \in L_2(\Omega)^3\right\}, \tag{17}$$

where $\Omega \equiv \Omega^- \cup \Omega^+$. The space of trial solutions, denoted by $\mathcal{S}$ is identical to $\mathcal{V}$. It too is comprised on functions that are continuous in the domain occupied by each phase, and discontinuous across the interface.

The weak form of the problem written for a finite element discretization of the function spaces is given by: find $\boldsymbol{Y} \in \mathcal{S}^h$ such that $\forall \boldsymbol{W} \in \mathcal{V}^h$,

$$\begin{aligned}
&\int_{\Omega} \Big(\boldsymbol{W} \cdot \boldsymbol{U}_{,\tilde{t}} - \boldsymbol{W}_{,i} \cdot \tilde{\boldsymbol{F}}_i + \boldsymbol{W} \cdot (\tilde{u}_{i,i}\boldsymbol{U})\Big)\,\mathrm{d}\Omega \\
&\qquad + \int_{\tilde{\Omega}} \Big(\hat{\boldsymbol{\mathcal{L}}}\boldsymbol{W} \cdot \boldsymbol{\tau}(\boldsymbol{\mathcal{L}}\boldsymbol{Y} - \boldsymbol{S})\Big)\,\mathrm{d}\Omega \\
&\qquad + \int_{\tilde{\Omega}} \Big(\nu^h g^{ij} \boldsymbol{W}_{,i} \cdot \boldsymbol{A}_0 \boldsymbol{Y}_{,j}\Big)\,\mathrm{d}\Omega \\
&+ \int_{\Gamma^I} \Big(\langle \tilde{\boldsymbol{F}}_i \rangle \cdot [\![\boldsymbol{W}]\!]_i + \langle \boldsymbol{K}_{ij}^T \boldsymbol{W}_{,j} \rangle \cdot [\![\mathbb{C}\boldsymbol{Y}]\!]_i\Big)\,\mathrm{d}\Gamma \\
&\qquad + \int_{\Gamma^I} \big([\![\mathbb{C}\boldsymbol{W}]\!]_i \cdot \boldsymbol{D}[\![\mathbb{C}\boldsymbol{Y}]\!]_i\big)\,\mathrm{d}\Gamma = \int_{\Omega} \boldsymbol{W} \cdot \boldsymbol{S}\,\mathrm{d}\Omega \\
&\qquad\qquad + \int_{\Gamma^+} \boldsymbol{W} \cdot \boldsymbol{H}\,\mathrm{d}\Gamma.
\end{aligned} \tag{18}$$

In the equation above the $\cdot$ symbol denotes the dot product between two vectors, the average operator that is defined as $\langle\cdot\rangle \equiv \frac{1}{2}[(\cdot)^+ + (\cdot)^-]$, $\tilde{\Omega}$ is the interior of all elements in Ω, $\boldsymbol{\tau}$ is a diagonal matrix of stabilization parameters, and the operator $\boldsymbol{\mathcal{L}}$ is defined as,

$$\mathcal{L} \equiv \boldsymbol{A}_0 \frac{\partial}{\partial \tilde{t}} + \tilde{\boldsymbol{A}}_i \frac{\partial}{\partial x_i} - \frac{\partial}{\partial x_i}\left(\boldsymbol{K}_{ij} \frac{\partial}{\partial x_j} \right) + \tilde{u}_{i,i} \boldsymbol{U}, \tag{19}$$

where $\boldsymbol{A}_0 = \boldsymbol{U}_{,\boldsymbol{Y}}$, $\tilde{\boldsymbol{A}}_i = \tilde{\boldsymbol{F}}^{adv}_{i,\boldsymbol{Y}}$, and $\boldsymbol{K_{ij}}$ is defined such that, $\boldsymbol{K}_{ij}\boldsymbol{Y}_{,j} = \boldsymbol{F}^{diff}_i$. We note that $\mathcal{L}\boldsymbol{Y}$ represents the Navier-Stokes equations written using the primitive variables. The operator $\hat{\mathcal{L}}$ is defined to be

$$\hat{\mathcal{L}} \equiv \tilde{\boldsymbol{A}}_i^T \frac{\partial}{\partial x_i} \tag{20}$$

In (18) the terms on the first line represent the Galerkin contribution from the original PDE, the term on the second line is the streamline upwind Petrov-Galerkin (SUPG) term [19], the term on the third line is the discontinuity-capturing (DC) operator which adds isotropic diffusion that is proportional to a residual-based eddy viscosity [22], the first term on the fourth line weakly enforces the jump condition for the fluxes (15), while the remainder of the terms on the left hand side weakly enforce the continuity condition (10). Of these, the last term on the left hand is the equivalent of the interior penalty in a typical discontinuous Galerkin method [2]. In this term $\boldsymbol{D}$ is a diagonal matrix whose entries are given by

$$\boldsymbol{D} = \mathrm{diag}(0, D_m, D_m, D_m, D_e). \tag{21}$$

The parameters D_m and D_e are determined by analyzing the stability of the weak form for a related scalar advection-diffusion problem as described in Section 2.6. The volumetric terms in our formulation represent an SUPG-based stabilized finite element method for the pressure-primitive variables with a DC term [16,17]. We note that there are several forms of the stabilization parameters $\boldsymbol{\tau}$ that have been proposed. In our formulation, for $\boldsymbol{\tau}$ we have used the expression from [15]. For a detailed discussion of different types of DC operators, the reader is referred to [22,49,37,44].

2.4. Consistency

It is easy to see that the Euler-Lagrange equations associated with this weak formulation are the strong form of the original PDEs. This is done by performing integration by parts on the first term on the left hand side and collecting terms in the interior and the interface. This yields,

$$\begin{aligned}
\int_{\Omega} \boldsymbol{W} \cdot \left(\boldsymbol{U}_{,\tilde{t}} + \tilde{\boldsymbol{F}}_{i,i} + \tilde{u}_{i,i}\boldsymbol{U} - \boldsymbol{S} \right) \mathrm{d}\Omega \\
+ \int_{\tilde{\Omega}} \left(\hat{\mathcal{L}}\boldsymbol{W} \cdot \boldsymbol{\tau}(\mathcal{L}\boldsymbol{Y} - \boldsymbol{S}) \right) \mathrm{d}\Omega \\
+ \int_{\tilde{\Omega}} \left(\nu^h g^{ij} \boldsymbol{W}_{,i} \cdot \boldsymbol{A}_0 \boldsymbol{Y}_{,j} \right) \mathrm{d}\Omega \\
+ \int_{\Gamma^I} \left(\langle \tilde{\boldsymbol{F}}_i \rangle \cdot [\![\boldsymbol{W}]\!]_i + \langle \boldsymbol{K}_{ij}^T \boldsymbol{W}_{,j} \rangle \cdot [\![\mathbb{C}\boldsymbol{Y}]\!]_i \right) \mathrm{d}\Gamma \\
+ \int_{\Gamma^I} \left([\![\mathbb{C}\boldsymbol{W}]\!]_i \cdot \boldsymbol{D} [\![\mathbb{C}\boldsymbol{Y}]\!]_i - [\![\boldsymbol{W} \cdot \tilde{\boldsymbol{F}}_i]\!]_i \right) \mathrm{d}\Gamma \\
- \int_{\Gamma^+} \boldsymbol{W} \cdot \left(\tilde{\boldsymbol{F}}_i n_i - \boldsymbol{H} \right) \mathrm{d}\Gamma = 0.
\end{aligned} \tag{22}$$

Using the definition of the jump and average operators it is easy to show that

$$[\![\boldsymbol{W} \cdot \tilde{\boldsymbol{F}}_i]\!]_i = \langle \boldsymbol{W} \rangle \cdot [\![\tilde{\boldsymbol{F}}_i]\!]_i + \langle \tilde{\boldsymbol{F}}_i \rangle \cdot [\![\boldsymbol{W}]\!]_i. \tag{23}$$

Using this in the last term of (22), and recognizing that the first term of (22) is the dot product of the weighting function with the residual of the Navier Stokes equation and may be rewritten as $\boldsymbol{W} \cdot (\mathcal{L}\boldsymbol{Y} - \boldsymbol{S})$, we arrive at,

$$\begin{aligned}
\int_{\Omega} \boldsymbol{W} \cdot (\mathcal{L}\boldsymbol{Y} - \boldsymbol{S})\, \mathrm{d}\Omega + \\
\int_{\tilde{\Omega}} \hat{\mathcal{L}}\boldsymbol{W} \cdot \boldsymbol{\tau}(\mathcal{L}\boldsymbol{Y} - \boldsymbol{S}) \mathrm{d}\Omega + \int_{\tilde{\Omega}} \left(\nu^h g^{ij} \boldsymbol{W}_{,i} \cdot \boldsymbol{A}_0 \boldsymbol{Y}_{,j} \right) \mathrm{d}\Omega
\end{aligned}$$

$$+\int_{\Gamma^I}\Big(-\langle \boldsymbol{W}\rangle\cdot[\![\tilde{\boldsymbol{F}}_i]\!]_i+(\langle \boldsymbol{K}_{ij}^T\boldsymbol{W}_{,j}\rangle+\boldsymbol{D}[\![\mathbb{C}\boldsymbol{W}]\!]_i)\cdot[\![\mathbb{C}\boldsymbol{Y}]\!]_i\Big)\,\mathrm{d}\Gamma$$

$$-\int_{\Gamma^+}\boldsymbol{W}\cdot\Big(\tilde{\boldsymbol{F}}_i n_i-\boldsymbol{H}\Big)\,\mathrm{d}\Gamma=0. \tag{24}$$

Once it is recognized that the numerical time-scale ν^h vanishes whenever the volumetric residual vanishes, the equation above makes it clear that Euler Lagrange equations implied by the weak from in the volume occupied by the two phases are the Navier Stokes equations (13). Whereas on the interface the Euler-Lagrange equations imply the jump (15) and the continuity (10) equations.

The interior stability of the formulation described in (18) can be attributed to the diffusive flux in the Galerkin term, the SUPG term and the DC term. The stability on the interface is obtained from the interior penalty term. Another mechanism of obtaining enhanced stability on the interface would be to replace the average flux on the interface with an upwinded or other stable numerical fluxes that are popular in finite volume methods [29,31]. While this is an option, we have not exercised it in this study.

2.5. Conservation properties

Since the weighting function and trial solution spaces are designed to be discontinuous across the interface, we are able to prove discrete conservation properties for each phase.

To see this in the context of conservation of mass, we begin by setting $\boldsymbol{W}=\boldsymbol{e}_1$ in Ω^+ and zero elsewhere. Here $\boldsymbol{e}_j$ is the j-th Euclidean vector, and selecting $j=1$ allows us to "pick" the conservation law for mass. Using this in (18), we are led to the equation,

$$\int_{\Omega^+}(\rho_{,\tilde{t}}+\nabla\cdot\tilde{\boldsymbol{u}}\rho)d\Omega+\int_{\Gamma^I}\langle\tilde{F}_{1i}\rangle n_i^+ d\Gamma+\int_{\Gamma^+}H_1 d\Gamma=\int_{\Omega^+}S_1 d\Omega. \tag{25}$$

Here $\langle\tilde{F}_{1i}\rangle=\langle\rho(u_i-\tilde{u}_i)\rangle$ is the average mass flux at the interface in a frame that moves with interface. Now consider the first term in the expression above,

$$\begin{aligned}\int_{\Omega^+}(\rho_{,\tilde{t}}+\nabla\cdot\tilde{\boldsymbol{u}}\rho)d\Omega&=\int_{\Omega^+}(\rho_{,t}+\tilde{\boldsymbol{u}}\cdot\nabla\rho+\nabla\cdot\tilde{\boldsymbol{u}}\rho)d\Omega\\&=\int_{\Omega^+}\big(\rho_{,t}+\nabla\cdot(\tilde{\boldsymbol{u}}\rho)\big)d\Omega\\&=\int_{\Omega^+}\rho_{,t}d\Omega+\int_{\Gamma^I}\tilde{\boldsymbol{u}}\cdot\boldsymbol{n}^+\rho d\Gamma\\&=\frac{d}{dt}\int_{\Omega^+}\rho d\Omega.\end{aligned} \tag{26}$$

In deriving the relation above in the first line we have used (12), in the third line we have used the divergence theorem and the fact that $\tilde{u}=0$ on the fixed boundary Γ^+, and in the fourth line we have made use of Leibniz theorem for evaluating the rate of change of integrated quantity when the domain of integral is changing with time. Using (26) in (25), we arrive at the statement of conservation of mass for the + phase in our numerical method,

$$\frac{d}{dt}\int_{\Omega^+}\rho d\Omega=-\int_{\Gamma^I}\langle\tilde{F}_{1i}\rangle n_i^+ d\Gamma-\int_{\Gamma^+}H_1 d\Gamma+\int_{\Omega^+}S_1 d\Omega. \tag{27}$$

This states that the rate of change of the total mass of the + phase is balanced by mass flux through the interface, the boundary Γ^+ and any mass sources.

Now setting $\boldsymbol{W}=\boldsymbol{e}_1$ in Ω^- and zero elsewhere, and repeating the derivation above, we arrive at

$$\frac{d}{dt}\int_{\Omega^-}\rho d\Omega=\int_{\Gamma^I}\langle\tilde{F}_{1i}\rangle n_i^+ d\Gamma+\int_{\Omega^-}S_1 d\Omega, \tag{28}$$

where we have made use of fact that $\boldsymbol{n}^-=-\boldsymbol{n}^+$. By comparing the first term on the right hand side of (27) with its counterpart in (28), we note that the mass lost by the + phase at the interface is exactly gained by the − phase. Further by summing the two mass conservation results we arrive at mass conservation over the total volume:

$$\frac{d}{dt}\int_{\Omega} \rho d\Omega = -\int_{\Gamma^+} H_1 d\Gamma + \int_{\Omega} S_1 d\Omega, \tag{29}$$

which states that the total mass of the two phases is balanced by the mass flux through the external boundary and volumetric sources of mass.

We now derive a conservation statement for the j-th component of momentum in the $+$ phase by setting $\boldsymbol{W} = \boldsymbol{e}_{1+j}$ in Ω^+ and repeating the steps described above. This leads to

$$\begin{aligned}\frac{d}{dt}\int_{\Omega^+} \rho u_j d\Omega = &-\int_{\Gamma^I} \left(\langle \tilde{F}_{(1+j,i)}\rangle n_i^+ + D_m(\hat{u}_j^+ - \hat{u}_j^-)\right) d\Gamma \\ &-\int_{\Gamma^+} H_{1+j} d\Gamma + \int_{\Omega^+} S_{1+j} d\Omega.\end{aligned} \tag{30}$$

Here $\langle \tilde{F}_{(1+j,i)}\rangle$ is the average momentum flux at the interface in a frame that moves with the interface and $\hat{u}$ is the projection of the velocity of each phase at the interface in the tangential directions. We note that the total interface flux is the integral of the average momentum flux and a term that is the integral of the difference in the tangential velocity components. Through the kinematic penalty term we drive this difference to zero with mesh refinement, and as a result the momentum flux reduces to its actual value (see Figs. 7 and 14 for example). Setting $\boldsymbol{W} = \boldsymbol{e}_{1+j}$ in Ω^-, we arrive at

$$\frac{d}{dt}\int_{\Omega^-} \rho u_j d\Omega = +\int_{\Gamma^I} \left(\langle \tilde{F}_{(1+j,i)}\rangle n_i^+ + D_m(\hat{u}_j^+ - \hat{u}_j^-)\right) d\Gamma + \int_{\Omega^-} S_{1+j} d\Omega. \tag{31}$$

By comparing (27) with (31) we conclude that once again there is a perfect cancellation of fluxes at the interface. That is the momentum flux lost by the $+$ phase is gained by the $-$ phase. Consequently, the total momentum conservation statement is obtained by summing the statement over each phase and is given by

$$\frac{d}{dt}\int_{\Omega} \rho u_j d\Omega = -\int_{\Gamma^+} H_{1+j} d\Gamma + \int_{\Omega} S_{1+j} d\Omega. \tag{32}$$

Similar conservation expressions can be derived for energy. In the $+$ phase it is given by

$$\begin{aligned}\frac{d}{dt}\int_{\Omega^+} \rho(e + \frac{|\boldsymbol{u}|^2}{2}) d\Omega = &-\int_{\Gamma^I} \left(\langle \tilde{F}_{5i}\rangle n_i^+ + D_e(T^+ - T^-)\right) d\Gamma \\ &-\int_{\Gamma^+} H_5 d\Gamma + \int_{\Omega^+} S_5 d\Omega.\end{aligned} \tag{33}$$

We again observe that the numerical energy flux at the interface is modified by a contribution that is proportional to the jump in temperature. This jump is penalized in our numerical method, and disappears with mesh refinement (see Figs. 7 and 14 for example). The energy conservation statement in the $-$ phase is given by

$$\frac{d}{dt}\int_{\Omega^-} \rho(e + \frac{|\boldsymbol{u}|^2}{2}) d\Omega = \int_{\Gamma^I} \left(\langle \tilde{F}_{5i}\rangle n_i^+ + D_e(T^+ - T^-)\right) d\Gamma + \int_{\Omega^-} S_5 d\Omega. \tag{34}$$

Once again we retain the desirable property of perfect cancellation of fluxes between the two phases across the interface. The equation for the total conservation of energy is obtained by summing (33) and (34) and is given by

$$\frac{d}{dt}\int_{\Omega} \rho(e + \frac{|\boldsymbol{u}|^2}{2}) d\Omega = -\int_{\Gamma^+} H_5 d\Gamma + \int_{\Omega} S_5 d\Omega. \tag{35}$$

The conservation properties for the methods may be summarized as follows:

1. For all conserved quantities (mass, momentum and energy) there is a perfect cancellation of flux at the interface. That is the flux entering the $+$ phase through the interface is exactly equal to the flux leaving the $-$ phase through the interface.
2. The mass flux at the interface is equal to the average of the flux of the two phases.
3. The momentum flux at the interface is equal to the average of the flux of the two phases plus a contribution that is proportional to the jump in the tangential component of the velocity. This jump is weakly enforced to be zero and it is driven to zero with mesh refinement.

4. The energy flux at the interface is equal to the average of the flux of the two phases plus a contribution that is proportional to the jump in the temperature. This jump is also weakly enforced to zero and it is driven to zero with mesh refinement.

2.6. Stability

We do not provide a detailed analysis of stability properties of the proposed method here; instead we consider its application to a simple linear scalar advection-diffusion problem and examine the stability properties of the resulting discretization. This gives points to the origins of stability for the underlying discretization and also guides us in selecting the parameters D_m and D_e that weakly enforce continuity across the interface.

Consider a simple linear advection-diffusion equation for a scalar ϕ,

$$\nabla \cdot (\boldsymbol{a}\phi - \kappa \nabla \phi) = 0, \text{ in } \Omega^+ \bigcup \Omega^-. \tag{36}$$

Here $\boldsymbol{a}$ is the advective parameter which is assumed to be divergence free, and κ is the diffusivity. On the interface Γ^I, we have the flux continuity condition

$$[\![a_i\phi - \kappa\phi_{,i}]\!]_i = 0, \tag{37}$$

and the kinematic continuity condition,

$$[\![\phi]\!]_i = 0. \tag{38}$$

In addition, on Γ^+ we have the flux prescribed as

$$(\boldsymbol{a}\phi - \kappa\nabla\phi) \cdot \boldsymbol{n} = h. \tag{39}$$

The weak form for this problem is given by: find $\phi \in \mathcal{S}^h$ such that $\forall w \in \mathcal{V}^h$,

$$B(w, \phi) = \int_{\Gamma^+} wh \, \mathrm{d}\Gamma, \tag{40}$$

where the bilinear form $B(\cdot,\cdot)$ is defined as,

$$\begin{aligned} B(w,\phi) \equiv & \int_{\Omega} -w_{,i}(a_i\phi - \kappa\phi_{,i})\mathrm{d}\Omega \\ & + \int_{\tilde{\Omega}} (\boldsymbol{a}\cdot\nabla w)\tau\big(\nabla\cdot(\boldsymbol{a}\phi - \kappa\nabla\phi)\big)\mathrm{d}\Omega + \int_{\tilde{\Omega}} \nu^h h^2 \nabla w \cdot \nabla\phi \mathrm{d}\Omega \\ & + \int_{\Gamma^I} \big(\langle a_i\phi - \kappa\phi_{,i}\rangle [\![w]\!]_i + \langle \kappa w_{,i}\rangle [\![\phi]\!]_i + D[\![w]\!]_i[\![\phi]\!]_i\big)\,\mathrm{d}\Gamma \end{aligned} \tag{41}$$

By comparing the terms in (41) with the terms in (18) it can be seen how the two formulations are related.

We now examine the coercivity of this bilinear form by setting the trial solution equal to the weighting function and assuming a linear finite element basis so that $\Delta\phi = 0$ within each element. This yields,

$$B(w,w) = \left\|\kappa^{1/2}\nabla w\right\|^2 + \left\|\tau^{1/2}\boldsymbol{a}\cdot\nabla w\right\|^2 + \left\|\nu^{h1/2}h\nabla w\right\|^2 + \left\|D^{1/2}[\![w]\!]_i\right\|^2_{\Gamma^I} \tag{42}$$

We note that all terms are positive and contribute to the stability of the numerical scheme. The first term, which comes from the diffusion operator provides stability in the volume in the diffusive limit. The second term, which comes from the SUPG term provides stability along the streamlines of $\boldsymbol{a}$ in the advective limit. The third term, which comes from the DC operator provides residual-driven isotropic diffusion. The fourth term provides control on the jump in the solution at the interface. We would like this term also to be active in both the advective and diffusive limits. Driven by this objective, the dimensionally consistent expression for the parameter in this term is determined to be

$$D = \max\{\frac{\kappa}{h_n}, |a_n|\}. \tag{43}$$

Generalizing this form of the parameter to the compressible phase change problem we select

$$D_m = \max\{\frac{\mu^+}{h_n}, \rho^+|u_n^+|, \frac{\mu^-}{h_n}, \rho^-|u_n^-|\}, \tag{44}$$

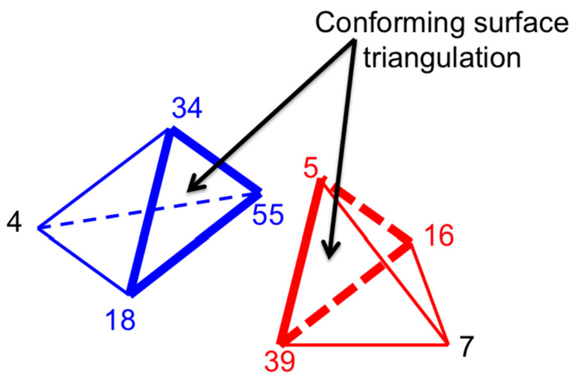


Fig. 2. The concept of a macro-element. The elements above share a common face, which happens to be on the phase interface, and whose edges are marked by thick lines. The evaluation of all the integrals on the interface requires information from all nodes marked on this figure. This requirement, which is non-standard in the finite element method, is met by defining a macro-element whose connectivity information includes all the nodes.

where $\mu^{\pm}$ is the viscosity in the two phases, and h_n is the mesh size in the normal direction at the interface. Similarly, for the energy equation

$$D_e = \max\{\frac{\kappa^+}{h_n}, \rho^+ C_v^+ |u_n^+|, \frac{\kappa^-}{h_n}, \rho^- C_v^- |u_n^-|\}, \tag{45}$$

where $\kappa^{\pm}$ is the thermal diffusivity in the two phases, and $C_v^{\pm}$ is the coefficient of specific heat at constant volume.

In recent studies on stabilized ALE finite element methods for compressible Navier Stokes equations, it has been shown that establishing Galilean invariance for a given formulation provides certain benefits for stability [41–43]. Applying a similar analysis to the formulation presented in this paper will be a useful addition as it might lead to the development of new, more stable methods. However, this is complicated by the fact that we are working with primitive variables (where proving invariance is harder) and the fact that we have introduced an interface term within a discontinuous formulation. However, it is something that will be explored in a future work.

3. Computational aspects

3.1. Finite element discretization at the phase interface

As described in the previous section, the finite element basis functions are continuous on the domains occupied by each phase. Thus the finite element connectivity within these domains is the same as that of a standard finite element method. However, the basis functions are discontinuous across the phase interface. Thus the evaluation of the integrals at the phase interface, as well as the connectivity along the interface warrants some discussion.

The term,

$$\int_{\Gamma^I} \left([\![\mathbb{C}\boldsymbol{W}]\!]_i \cdot \boldsymbol{D} [\![\mathbb{C}\boldsymbol{Y}]\!]_i \right) \mathrm{d}\Gamma, \tag{46}$$

involves the dot product of the jump in the weighting function with the jump in the trial solution. This couples the degrees from one side of the interface with the degrees of freedom on the other side. Referring to Fig. 2, where we have shown two tetrahedral finite elements with a conforming face that lies on the interface, this term will couple the degrees of freedom on nodes 18, 34 and 55 with the degrees of freedom on nodes 39, 5 and 16.

The term,

$$\int_{\Gamma^I} \langle \tilde{\boldsymbol{F}}_i \rangle \cdot [\![\boldsymbol{W}]\!]_i \mathrm{d}\Gamma, \tag{47}$$

involves the dot product of the jump in the weighting function with average of the fluxes on either side of the interface. The advective part of the flux only involves the trial solution (and not its derivatives) and therefore through this term it only couples the degrees of freedom at the interface with each other (same as the term above). However, the diffusive part of the flux involves gradients of the primitive variables, and therefore evaluating it at the interface involves contributions from nodes that are not on the interface. Consequently this term couples degrees of freedom at the interface with those that are not on the interface. Referring again to Fig. 2, this term will couple the degrees of freedom on nodes 18, 34 and 55 with the degrees of freedom on nodes 39, 5, 16, and 7, similarly degrees of freedom on nodes 39, 5, and 16 with those on 18, 34, 55 and 4. It is easy to see that the term,

$$\int_{\Gamma^I} \langle \boldsymbol{K}_{ij}^T \boldsymbol{W}_{,j} \rangle \cdot [\![\mathbb{C}\boldsymbol{Y}]\!]_i \mathrm{d}\Gamma \tag{48}$$

which involves derivatives of the weighting functions and jumps in the trial solution will also generate a similar type of coupling.

In summary then, for those elements that share a face that lies on the phase interface, all the degrees of freedom (dofs.) on the bounding elements are coupled. One way to accommodate this coupling, which is the strategy we have selected, is to create macro-elements at the interface, whose connectivity list contains all the nodes associated with these two elements. In the context of Fig. 2, the macro-element will span both elements, and its connectivity list will read 4, 18, 34, 55, 39, 5, 16, and 7.

3.2. Mesh motion and modification

As described above, the location of the interface at any given instant is represented by the common faces of the macro-elements. At every time step, the location of the nodes on these faces is updated by computing the interface velocity via Equation (7). This gives us the updated information of the interface. Based on this update a new location is determined for every node in the finite element mesh through an elastic analogy [45]. The entire domain is treated like an elastic solid and Dirichlet (displacement) data is prescribed for the nodes on the interface and for the nodes that are on a fixed boundary. The updated location of all other nodes is determined by approximately solving an elastic problem for an isotropic linear elastic material.

In defining the elastic parameters for the mesh motion problem, we have to select the Young's modulus, E and the Poisson's ratio, ν, for the problem. We set the Young's modulus to be inversely proportion to the volume of an element, and the Poisson's ratio to a constant value of 0.2. This specific choice for the Young's modulus ensures that the smaller elements deform the least, and the larger elements deform the most. Since the larger elements can deform more without becoming invalid, this allows for the simulation to proceed for a significant time interval without the need for mesh modification. We do however recognize that as the interface deforms significantly, the element quality deteriorates, and we perform mesh modification in those regions [30,38,39]. Yet another update, which is not considered in this paper, would require modifying the underlying geometric model when entities within it undergo topological changes. This addition to our approach will be considered in the future.

Changes in mesh topology are based on local mesh modification operations. These are applied in an incremental fashion and the solution is transferred in a local fashion (i.e., at the level of the mesh cavity) whenever a mesh modification operation is applied. A callback mechanism is invoked each time a mesh modification takes place. For example, when an element is refined by splitting its edge(s), a list of the old elements as well as elements of the new mesh are handed into the callback function which then accesses the solution values on the old elements and transfers them onto the new elements according to procedures encoded into the callback function. A linear interpolation is currently used to define solution values for the newly created nodes. This approach resulted in the desired accuracy for the cases considered in this paper. A more rigorous analysis of solution transfer, under the full set of local mesh modification operations, will be considered in the future (see [23], for example).

We note that in contrast to the displacement-based mesh motion method we have used in this manuscript, several authors have proposed the use of velocity-based mesh motion methods [32,53,5]. These methods are particularly useful for problems where the mesh velocity is driven by, and close to the complex velocity field for the material within the computational volume. That is, the problem is solved in a setting that is close to a Lagrangian formulation. In contrast to this, in our application the mesh velocity is driven by the interface velocity, which is not as complex, and about two orders of magnitude smaller than the fluid velocity in the lighter phase. Thus in the lighter phase, where the velocity field is complex, we are close to an Eulerian setting and the application of these methods is not necessary. However, as we move to more complex applications, we will consider these methods.

3.3. Time integration scheme

The weak form in (18) leads to a set of nonlinear ordinary differential equations that need to be solved in order to determine the evolution of the nodal values of the primitive variables. These equations are integrated using the Generalized-alpha method [11,25]. This method is a general method which is defined by 4 parameters that determine the specific time-integration scheme that is being exercised. Many popular time integration schemes, such as the Newmark method, the trapezoidal method, and the mid-point rule can be obtained as special cases of the Generalized-alpha method. This approach is particularly useful in designing second-order time integration methods that are dissipative for high-frequency modes while retaining their accuracy for the lower-frequency modes.

In Section 2.5 we established the conservative properties for the semi-discrete (or time-continuous) version of our formulation. The conservation properties of the complete formulation, that is the fully-discrete formulation, are determined by this analysis and the time integration scheme. As demonstrated in that section, the semi-discrete formulation is conservative; however the application of the time-integrator to this formulation results in a loss of conservation, and the extent of this loss is determined by the accuracy of the time-integrator. In order to the quantify this loss, for the complex problems

Table 1
Material properties for two phases.

Property		grains	gas	Units
Molecular weight	M_w	–	0.029	kg/mol
Density	ρ	100	$\sim$ 1 (at NTP)	kg/m^3
Heat capacity	C_p	1460	$\frac{\gamma R}{\gamma-1}$	J/kg K
Specific heat ratio	γ	–	1.4	
Molecular viscosity	μ	10	1e-5	Pa s
Thermal conductivity	λ	31.1	0.04	W/m K
Compressibility	α_p	1e-4	$1/T$	1/K
Compressibility	β_T	1e-7	$1/p$	1/Pa

considered in this paper, we have plotted the variation of the conserved quantities as a function of time in the numerical examples section.

As described in [25], the Generalized-alpha method is implemented in a predictor-corrector format. Using this approach, every time-step begins with a predictor that is derived from the current estimates of the rate of change of variables. This is followed by a corrector step. For implicit methods applied to linear systems, the corrector step involves solving a linear system of equations. For nonlinear systems, it leads to a nonlinear algebraic problem, which is solved through approximate Newton iterations. In evaluating the left hand side for this problem, a consistent tangent is not evaluated; rather approximations are made to retain the important terms. Several (2–4) approximate Newton iterations are done to ensure that the residual for the nonlinear systems drops by three to four orders of magnitude. These steps are interlaced with mesh update steps in a staggered approach that ensures that both the fluid variables and the mesh coordinates are converged at the end of each time step. Each linear system is solved using the GMRES iterative solver along with a block-diagonal preconditioner.

4. Results

In this section we present numerical results that verify the formulation developed in the previous section and demonstrate its utility in simulating phase change problems, especially those involving strong variations in density and velocities. In all the examples, we consider a gas phase whose properties are close to that of air with an ideal gas equation of state, and a condensed phase whose properties are close to that of grains found in explosives (HTPB) [35,12], with a few differences. First, the density is chosen to be about a tenth of the actual value. Second, the grains are modeled as a very viscous fluid rather than a viscoelastic solid. And finally, for the last two examples considered, the phase change rate of the grains is increased from its original value to ensure that the phase change velocity, u_p, is sufficiently large. For the grains, the equations of state for density and internal energy are linear expansions in temperature and pressure about a reference value. All the material parameters are listed in Table 1. For all simulations we use finite elements with linear interpolations.

4.1. Stefan's problem

This problem is used to verify that the conditions at the interface are correctly modeled. It comprises of a dense phase with the material properties of the grain and a much lighter gas phase. The interface is a plane with the normal oriented along the x-direction. We model a length of 2 mm of each phase with a square cross-section, assume a constant interface velocity of 1 m/s. In this problem, the dense phase is almost stationary and we apply a zero-velocity boundary condition at $x = -2$ mm. We also prescribe the temperature at this surface to be 288 K. On the boundary at $x = 2$ mm, we specify pressure and a zero heat-flux boundary condition. We also increase the thermal conductivity of the liquid phase by a factor of 100 in order to enhance diffusion so that we can compute an analytical solution to verify the numerical method.

The mesh used to solve this problem is shown in Fig. 3. We solve the problem with CFL $\approx$ 20 and accommodate the motion of the interface through mesh motion at every time step. We perform a mesh modification step every 200 time steps where the boundary layer mesh at the interface and at the domain boundaries is maintained. We perform a total of 7 mesh modification steps. The mesh after 3 such mesh modification steps is shown in Fig. 3.

We test the performance of the mesh motion and mesh modification algorithms by demonstrating they approximate the so-called geometric conservation law (GCL) with good accuracy. According to the integral form of this law, the volume of a given phase at time t, denoted by $V^{\pm}(t)$, must satisfy the relation [50],

$$V^{\pm}(t) - \int_{0}^{t} \int_{\Gamma^I} \boldsymbol{v} \cdot \boldsymbol{n}^{\pm} d\,\Gamma = V^{\pm}(0). \tag{49}$$

Recall that $\boldsymbol{v}$ is the velocity of the interface. In Fig. 4 we have plotted the left hand side of this equation as a function of time for the liquid and the gas phase, scaled by the initial volume. In this figure the red dots indicate instances when mesh modification is performed. From this figure we observe that for both phases the GCL is satisfied to good accuracy throughout the simulation. In fact, for this problem, since the interface starts and remains planar, the GCL is satisfied to within numerical precision during the entire simulation.

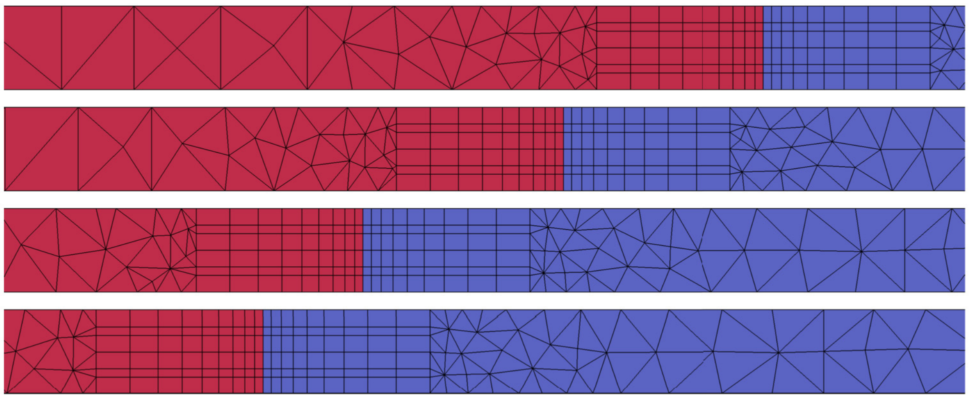


Fig. 3. A zoomed-in cross-sectional view of the mesh for Stefan's problem at time-steps $N = 20, 820, 1250$ & 1590 (from top to bottom). The colors indicate different phases. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

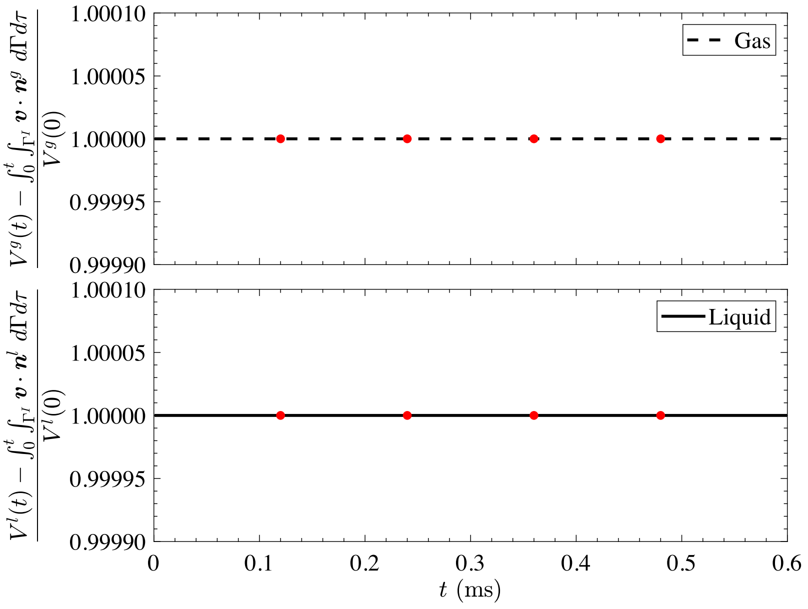


Fig. 4. Illustration of the GCL for each phase in the Stefan problem. Red dots indicate instances when the mesh is modified.

To verify our numerical results we analytically solve a one-dimensional Stefan problem with some simplifying assumptions. In the liquid we assume $\rho = 100$ kg/m^3, $v = 0$ and $\frac{dT}{dx} =$ constant, whereas in the gas we use the ideal gas equation of state to relate, ρ, T and P. The boundary conditions for this problem are same as that of the Stefan problem described above. The flow is assumed to be uniform in the gas since the momentum transfer is dominated by advection. As a result, the pressure in the gas phase is 1 atm, which is the boundary condition at the outlet. Given these assumptions, solving the quasi-steady Navier Stokes equations reduces to solving the interface conditions – a system of non-linear algebraic equations. These equations are solved in MATLAB and compared to the numerical solution as shown in Fig. 5. We observe good agreement and note that the minor differences between analytical and numerical solutions can be attributed to the assumptions made when evaluating the former. We observe similar agreement at other times (not shown here). This problem can be solved without including the DC term, and this solution, as well as the solution with the DC term are shown in Fig. 5. There is no discernible difference between these two solutions, indicating that this problem does not need the added stability from the DC term. We observe a significant jump in the velocity magnitude at the interface that occurs due to the significant difference in density among the two phases. We observe a jump in pressure that arises from balancing the momentum flux. We also note that even though the temperature was only weakly enforced to be continuous at the interface, it is in fact very close to being continuous.

We have performed a convergence study for the problem by refining the mesh at the interface in the normal and tangential directions, and computing the error in enforcing the flux and continuity (in temperature and tangential velocity) conditions. In Fig. 6 we have plotted the normalized L_2 norm of the error in the flux condition for mass, momentum and energy as a function of mesh size, for formulations with and without a DC term. For the formulation without a DC term, we

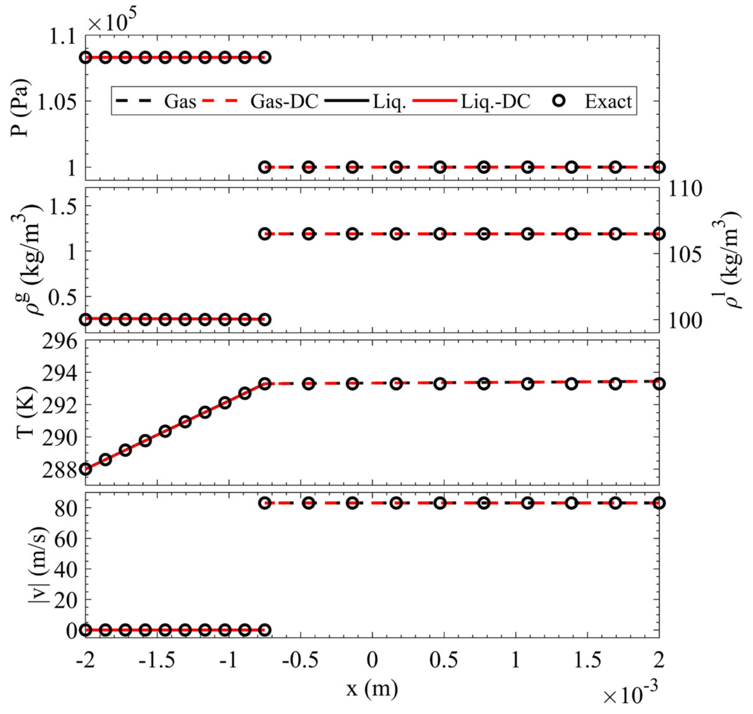


Fig. 5. Comparison of analytical and numerical solutions for the Stefan problem at $t = 0.75$ ms. Black dashed and solid lines represent the numerical solution without the DC term, while red dashed and solid lines represent the numerical solution with the DC term, and the symbols represent the analytical solution.

note that the error in mass and momentum flux conditions reduces like $O(h^2)$, while the error in the energy flux condition varies as $O(h^1)$. The lower convergence in the energy flux condition is expected since it is dominated by the diffusive flux in the heavier phase, which in turn depends on the spatial derivative of temperature. In contrast to this the mass and momentum fluxes are dominated by advective components which only depend on the variables and not their derivatives. For the formulation with the DC term, we observe similar trends, but with two caveats. First, for mass conservation the convergence rate appears to drop from approximately 2 to 1. This drop is likely due to the fact that without the DC term the equations for the conservation of mass do not contain any diffusive terms; however the DC term introduces a diffusive term and therefore changes the character of this equation. Second, though the rates for momentum and energy remain the same with the addition of the DC term, the constant in front of all rates increases, thereby increasing the error in conserving different quantities at the interface. This poses a challenge. Understanding this effect, and then modifying the DC formulation to overcome it, is an interesting avenue for future work.

Another measure of the accuracy of our method at the interface is the imposition of the kinematic conditions. These are the continuity in temperature and the tangential components of the velocity vector. The normalized L_2 in enforcing these conditions is shown in Fig. 7. From this figure we observe that both these errors are very small, and appear to be converging at $O(h^2)$. We also observe that these rates do not change with the introduction of the DC term, though the constant in front of them increases somewhat.

4.2. Phase change of a spherical droplet

Next we consider a problem described in spherical coordinates where we include additional physics when compared with Stefan's problem. The mesh for this problem is displayed in Fig. 8. We consider a droplet of the denser phase with diameter 1 mm, surrounded by the larger sphere of the lighter phase with diameter 2 mm. On the surface of the outer sphere we prescribe zero velocity and zero heat flux condition. The problem starts with both phases at rest, and is driven by phase velocity of $u_p = 1$ m/s. When compared with the Stefan problem we include a heat of reaction (~ 2 MJ/kg) term in the energy balance which represents the conversion of gaseous phase into the products of combustion. This term is accounted in the formulation by ensuring that the difference in the internal energy per unit mass between the liquid and the gas phase includes a contribution from this term. Since the emphasis of this manuscript is on modeling phase change, we do not consider more refined models of combustion. We do however note that the phase change methodology described in this paper can be combined with such models. We also note that in this problem, and all subsequent problems we have included the DC term in our formulation.

In Fig. 9 we examine the adherence of the numerical method to the GCL for this problem. We plot the sum of the volume and the integrated volume flux for each phase, scaled by its initial volume, as a function of time. According to the GCL (49)

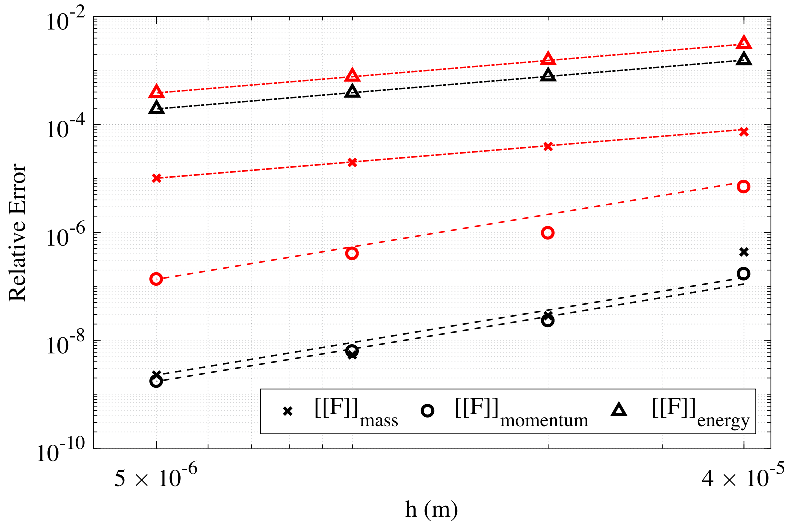


Fig. 6. Normalized L_2 error in enforcing interface mass, momentum and energy flux conditions at $t = 0.12$ ms as a function of mesh size (Black - without DC; Red - with DC). The dash-dot line represents a slope of h^1 and the dashed lines represent a slope of h^2.

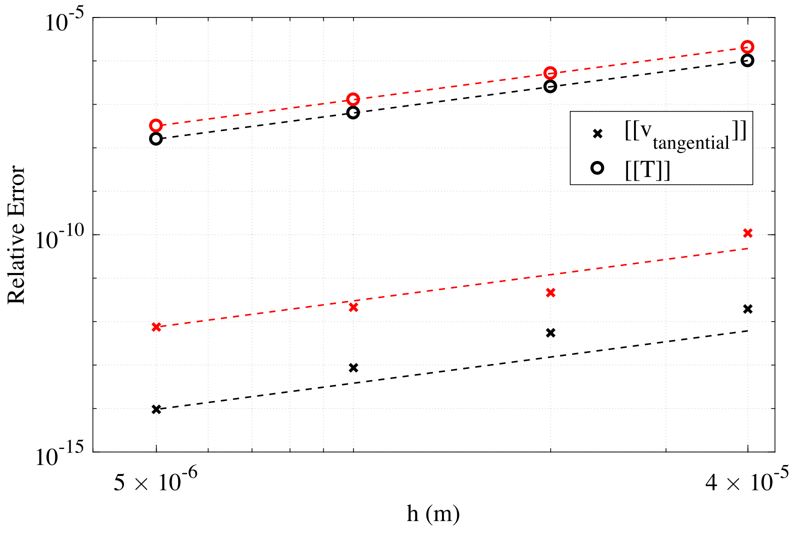


Fig. 7. Normalized L_2 error in enforcing continuity of temperature and tangential velocity at the interface at $t = 0.12$ ms as a function of mesh size (Black - without DC; Red - with DC). All lines represent a slope of h^2.

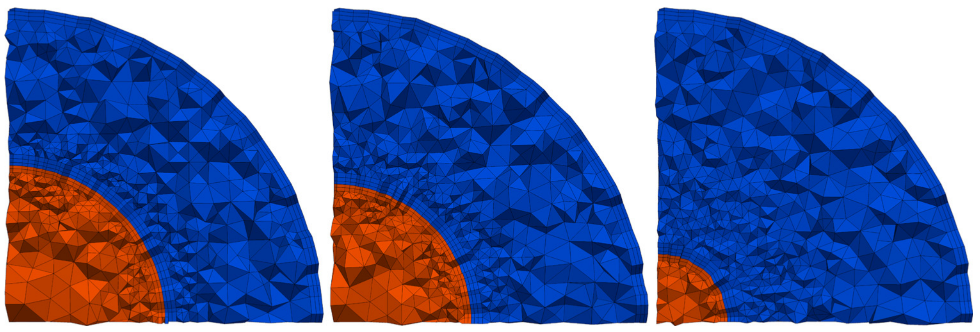


Fig. 8. A cross-sectional quadrant view of the mesh for the spherical droplet at time-steps $N = 10, 100$ & 430 (from left to right). The colors indicate different phases.

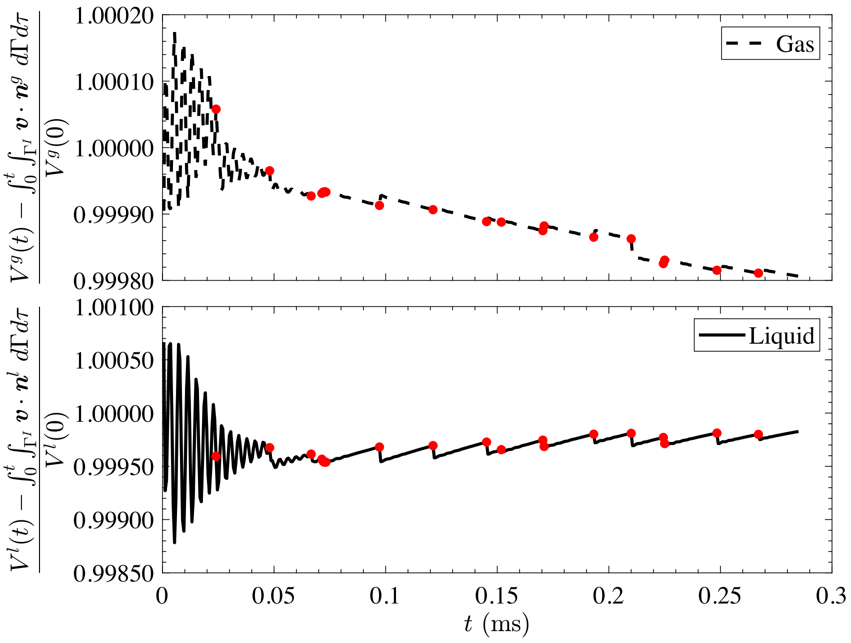


Fig. 9. Illustration of the GCL for each phase in the sphere problem. Red dots indicate instances when the mesh is modified.

this quantity must be equal to unity. We observe that it stays close to unity throughout the simulation. We also note that due to the impulsive start to this problem, pressure waves are set up in the domain and are damped after some time. These waves cause rapid (though small) variations in the conserved geometric quantities at early time. The red dots in this plot represent instances when the mesh is modified. At these instances we observe changes in the volume for each phase. This is to be expected, since both the outer surface and the interface in this problem are curved surfaces that are approximated by linear finite elements. A change in these elements leads to a slightly changed overall volume for each phase.

The problem commences with the gas pressure at 1 atm. The phase change causes the droplet to shrink and the lighter phase created at the interface is expelled outward, while the fluid within the droplet is relatively stationary. It also results in the generation of heat due to the heat of reaction which causes the temperature to increase. Since all this occurs in a closed, adiabatic chamber, it causes the pressure within the gas phase to rise. This can be seen in the variation of physical quantities along a radial line in Figs. 10–12.

At early time ($t = 0.006$ ms in Fig. 10) the density of the liquid phase is around 100 times the gaseous phase. Given that the liquid phase is at rest, and the phase change velocity is 1 m/s, conservation of mass at the interface implies that the speed in the gaseous phase must be around 100 m/s. Further, conservation of mass in the gas implies that this speed decay as r^{-2}, as long since the density in the gas is mostly uniform. This can also be observed in Fig. 10. Also, since at this time the phase change has just commenced, we do not observe appreciable changes in pressure and temperature beyond their initial values.

Next we consider the solution at $t = 0.06$ ms when the liquid sphere has lost around 38% of its initial volume (see Fig. 11). Due to sustained phase change and the corresponding generation of heat, the temperature at the interface has risen to 1,200 K. We note that enough time has passed for the hot gas to be convected to the outer boundary of the spherical domain, where we see a distinct front in the temperature profile. Within the droplet, the temperature at the interface has diffused into the core. Also, since the gas is contained within an enclosed chamber, the retardation of the gaseous phase at the outer boundary has led to an increase in the pressure to around 15 atm. The elevated pressure leads to elevated density at the interface, which in turn leads to the lowering of the speed in the gaseous phase to around 25 m/s.

These trends are seen to persist at $t = 0.26$ ms when the liquid volume is down to around 6% of its initial volume (see Fig. 12). The temperature at the interface has risen to 4,500 K, and the pressure everywhere is around 80 atm. The density of the gas at the interface has increased and its speed is now down to around 20 m/s. We also observe that the density variation is in inverse proportion to the temperature variation, leading to a more-or-less uniform pressure profile in the gas.

The process of mesh motion and modification is similar to that for the Stefan problem. As the interface moves, the mesh at the interface moves with it, while the mesh motion in the volume is determined by solving a linear elasticity problem. When the mesh quality starts to deteriorate a mesh modification step is performed in which the boundary layer mesh structure on either sides of the interface is preserved. We note that in the time takes for spherical droplet to shrink to 40% of its original diameter, which corresponds to 430 total time steps, we need only about 15 mesh modification steps.

We have also performed a convergence study for this problem by considering a hierarchy of meshes and computing the errors in enforcing the flux and continuity conditions at the interface. For the coarsest mesh the ratio of $D/h = 5$ (here D

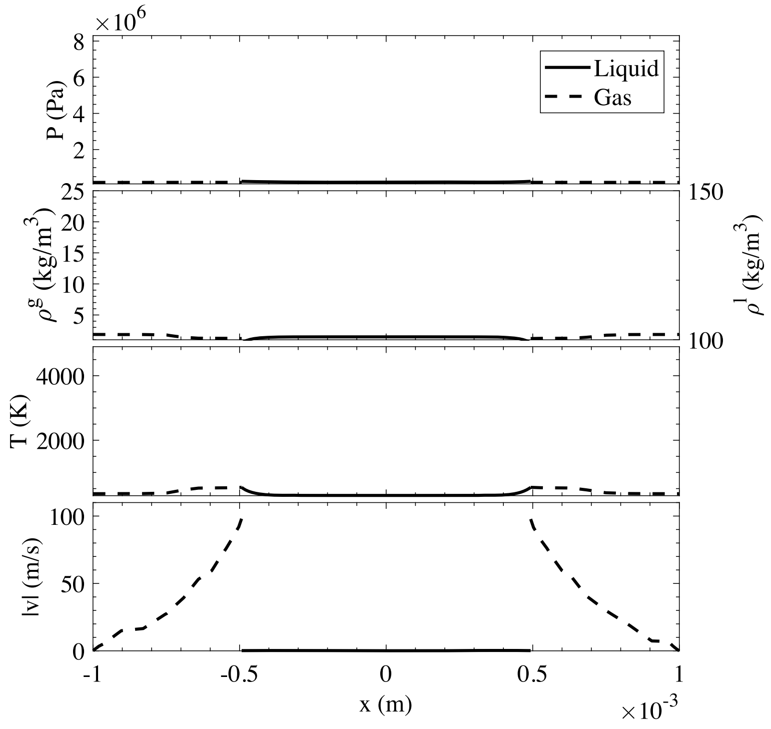


Fig. 10. Variation of the numerical solution along a radial line at $t = 0.006$ ms ($N = 10$). The solution in the dense liquid phase is denoted by a solid line and the gaseous phase is denoted by a dashed line.

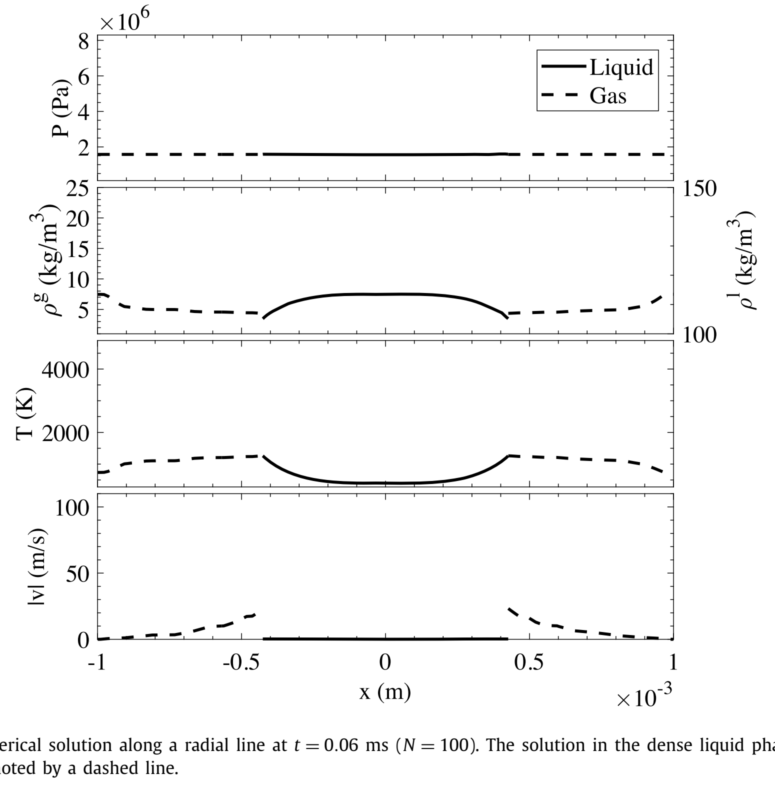


Fig. 11. Variation of the numerical solution along a radial line at $t = 0.06$ ms ($N = 100$). The solution in the dense liquid phase is denoted by a solid line and the gaseous phase is denoted by a dashed line.

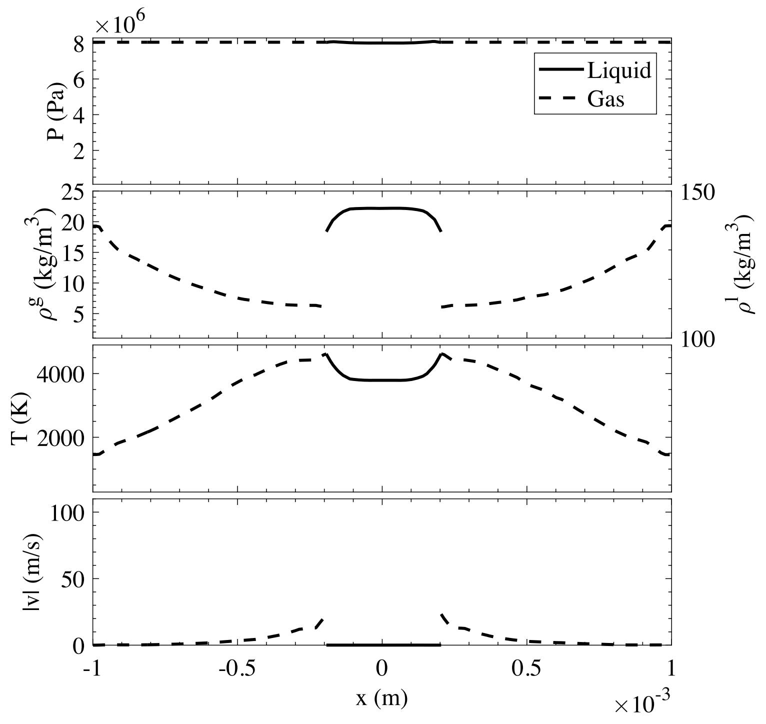


Fig. 12. Variation of the numerical solution along a radial line at $t = 0.26$ ms ($N = 430$). The solution in the dense liquid phase is denoted by a solid line and the gaseous phase is denoted by a dashed line.

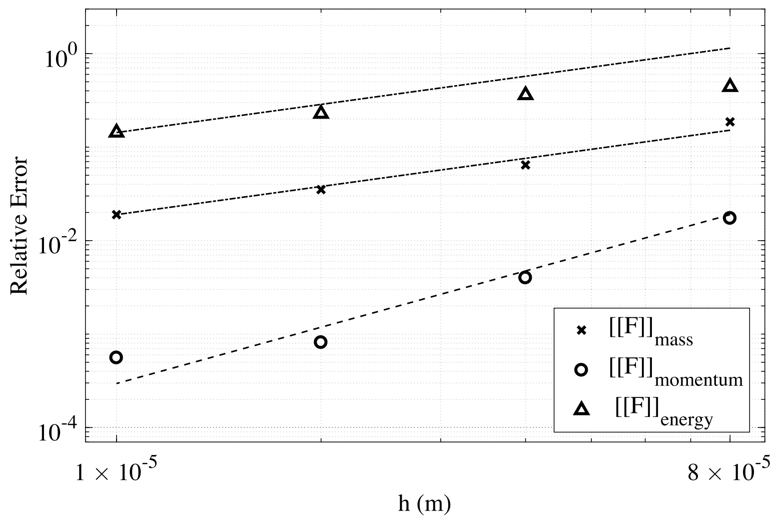


Fig. 13. Normalized L_2 error in enforcing interface mass, momentum and energy flux conditions at $t = 0.12$ ms as a function of mesh size. The dash-dot line represents a slope of h^1 and the dashed lines represent a slope of h^2.

is the initial diameter of the sphere, and h is tangential mesh size) while for the finest mesh $D/h = 40$. The results of this study are shown in Figs. 13 and 14. From Fig. 13 we observe that the error in the momentum flux condition reduces as $O(h^2)$ and the error in the mass and energy flux conditions reduces as $O(h)$. While from Fig. 14 we observe that the error in enforcing continuity in temperature and tangential velocities reduces as $O(h^2)$. All these rates are similar to the rates observed for the Stefan problem.

4.3. *Phase change of a cylindrical grain*

Next we consider the phase change and combustion of a cylindrical grain with hemispherical end caps enclosed in a small cylindrical chamber. The length of grain is 2 mm and its radius is 0.5 mm, while the length of chamber is 4 mm

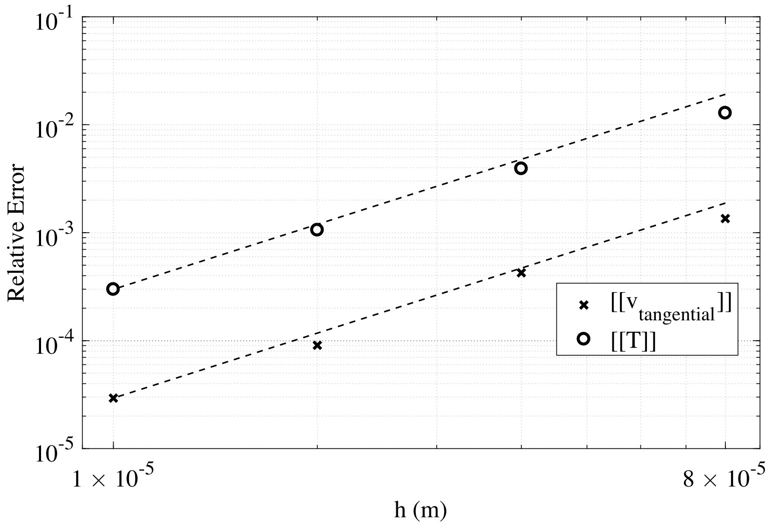


Fig. 14. Normalized L_2 error in enforcing continuity of temperature and tangential velocity at the interface at $t = 0.12$ ms as a function of mesh size. Both lines represent a slope of h^2.

and its radius is 1 mm. Once again we include the heat of reaction term. In addition we use a more realistic phase change model by making use of Vieille's law, where the phase change speed is given by $u_p = a(p^+)^n$, where p^+ is the pressure of the gas. Here we select $n = 0.7$ and $a = 7.9 \times 10^{-5}$ m/(s Pa^n). These values are selected based on typical values found for an energetic material like AP/HTPB/AL [35,12]. We note that we have increased the value of a by a factor of 3000 in order account for the fact that we have used a lower density material (this accounts for a factor of 10) and to speed up the phase change process (this accounts for a factor of 300).

The initial pressure within the chamber is set to 1 atm. At this pressure the speed of phase change is 0.25 m/s, which is rather small. However, as phase change and reaction occur, the pressure in the chamber increases due to the deceleration of high speed gasses in a closed chamber, and the confinement heat released due to reaction. This causes the rate of phase change to increase due to the pressure dependence of Vieille's law thereby accelerating the entire process. This can be seen by plotting the average gas pressure and temperature as a function of time in Fig. 15. We observe that both these quantities rise slowly initially and then increase exponentially, finally slowing down when the interface area becomes very small. We terminate the simulation when the volume of the grain reduces to about 3.5% of its initial volume.

The exponential increase in pressure and temperature introduces very strong gradients in the solution. In this case we have found that the SUPG term alone is unable to provide sufficient stability. For this reason we have made use of the so-called discontinuity (DC) capturing operators [22,49,37,44] in solving the single and multiple grain problems (considered next). Like the SUPG term the DC term is also residual based and is therefore consistent; however unlike the SUPG term which provides stability only in the upwind direction, this term, which takes the form of an isotropic diffusion, acts in all directions. Also, since this term involves gradients of the weighting function, it vanishes when the weighting function is set to unity to determine the conservative properties of the method. Thus the results of Section 2.5 hold even with the addition of this term.

The mesh used to solve this problem comprises of around 2.3 million linear tetrahedral and wedge elements. At the interface, on the side of the propellant grain, there are two layers of boundary layer mesh, with a total thickness of 2.2×10^{-5} m. On the gas side, there are three layers of boundary layer mesh with a total thickness of 3.64×10^{-5} m.

The variation of pressure, temperature and velocity within the gas phase inside the chamber is shown in Figs. 16, 17 and 18 respectively at two different time instances ($t = 0.122$ and 0.1907 ms). For pressure, we scale the pressure field with its average value, and plot this value. With this approach, we are able to discern variations in pressure are within 0.8–1.4 times the average pressure, and are therefore not significant. We also observe that due to the presence of strong acoustic waves that bounce back and forth within the chamber, the pressure distribution within the chamber does not display a consistent pattern that is sustained over the course of the simulation. The only trend that is observed is that along the axis of the grain, the pressure close to the center is elevated, and because of this and the dependence of phase change rate on pressure, this region of the grain is consumed faster than other regions. This gives the grain a pinched-off appearance at later times.

The velocity field in a plane within the chamber can be seen by visualizing the line integral convolution (LIC) of velocity in Fig. 17. At both times we observe high speeds of around 200 m/s. This is in contrast to the spherical droplet problem where the gas speed reduced with time. The difference between these two problems is the speed of phase change, which was held constant in the special droplet problem and allowed to vary as a function of pressure in the cylindrical grain problem. As a result in the cylindrical grain problem the reduction in the speed at the interface due to the increase in gas density is offset by the increase in the phase change speed due to increasing pressure. A close look at the flow patterns at different times reveals certain interesting features. These include, a stagnation flow on the curved surface of the chamber

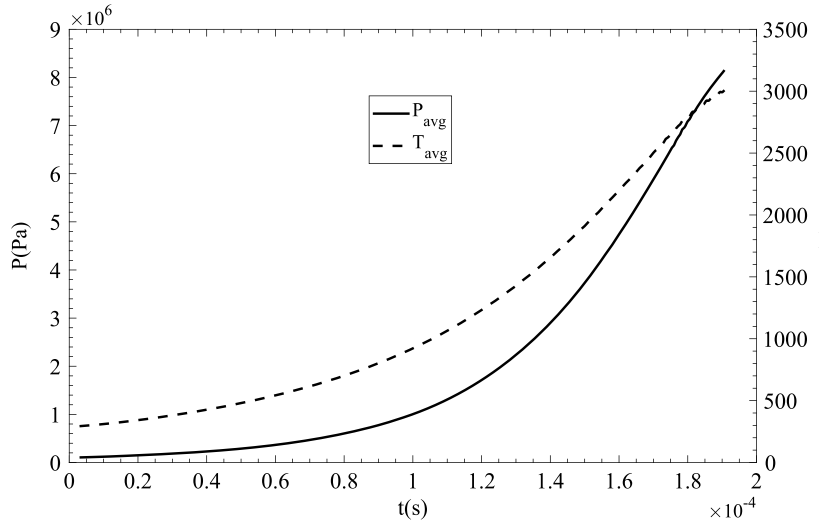


Fig. 15. Average pressure and temperature in the gas as a function of time.

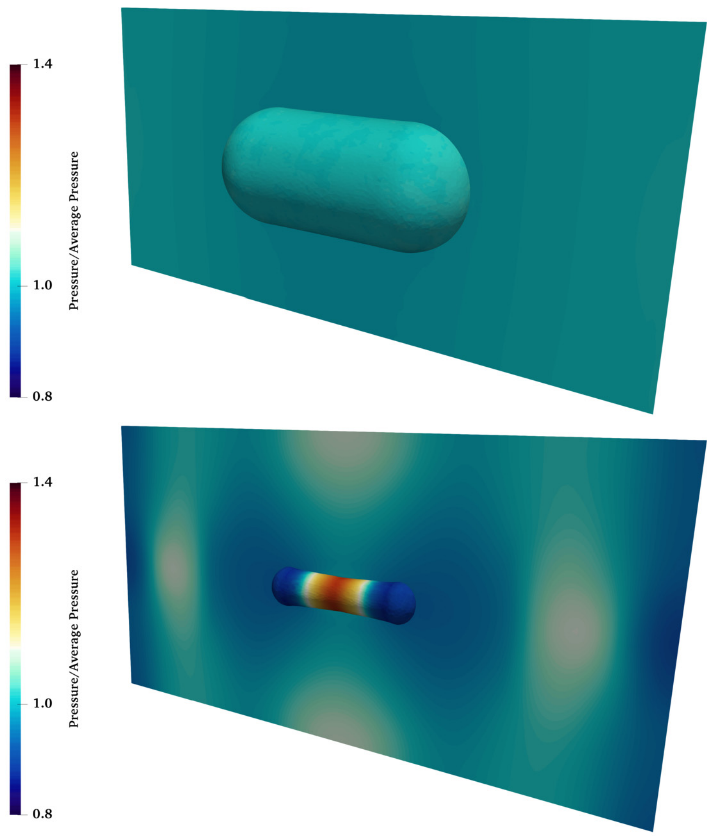


Fig. 16. Pressure, scaled by its average value in the chamber, at $t = 0.122$ and 0.1907 ms for the cylindrical grain problem (time step $N = 410$ and 1190).

surrounding the grain with the stagnation point (curve) directly above and below the grain, the presence of vortex cores (curves) towards the ends of the chamber that appear to move with time, and the presence of 2–3 stagnation points along the long axis of the cylinder that also appear to move with time.

The distribution of the temperature within the chamber is shown in Fig. 18 where we note that in contrast to the pressure distribution it is much more non-uniform. This can be understood by viewing temperature as a scalar that is generated at the interface (due to phase change and reaction) and is convected in the chamber by the gas (diffusion plays

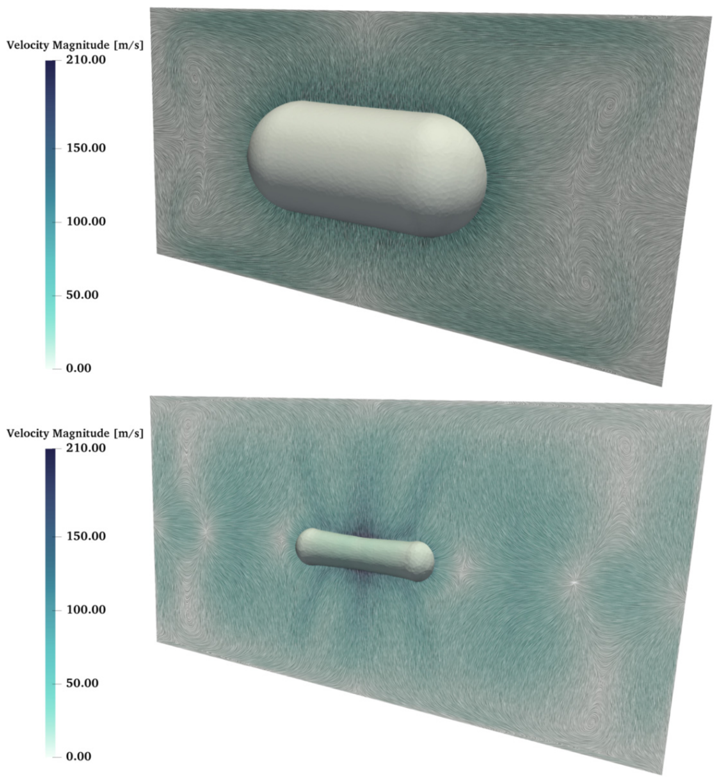


Fig. 17. Line integral convolution (LIC) of velocity at $t = 0.122$ and 0.1907 ms for the cylindrical grain problem (time step $N = 410$ and 1190).

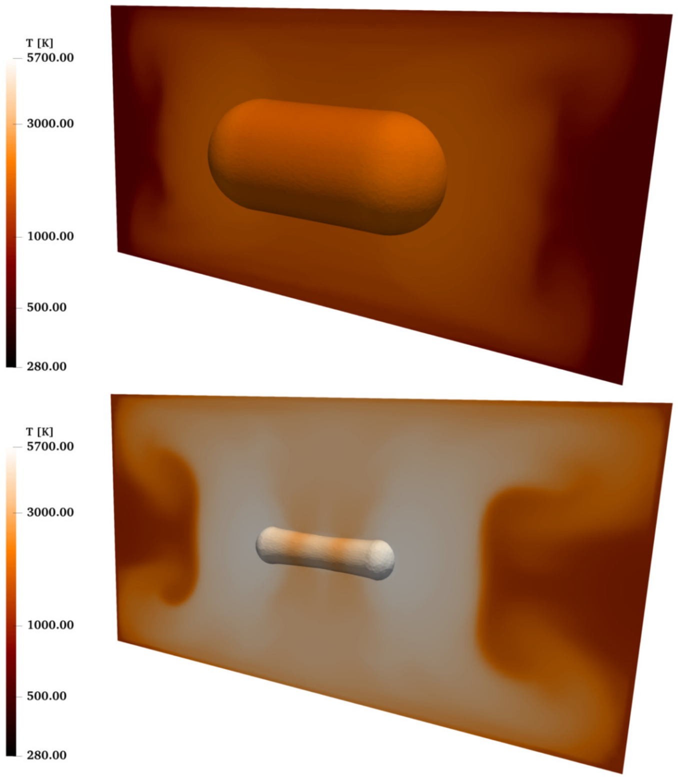


Fig. 18. Temperature at $t = 0.122$ and 0.1907 ms for the cylindrical grain problem (time step $N = 410$ and 1190).

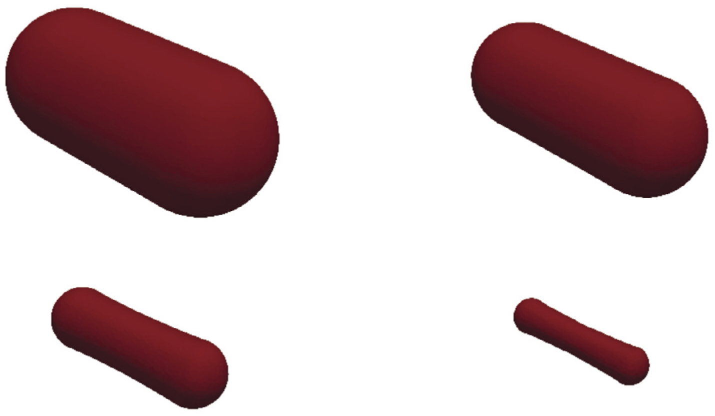


Fig. 19. Shape of the grain at $t = 0.003, 0.122, 0.1706$ and 0.1907 ms (time steps $N = 10, 410, 660$ and 1190).

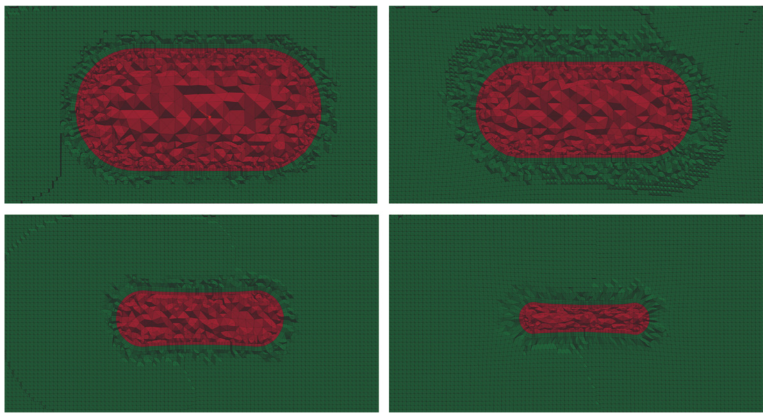


Fig. 20. Mesh along a cross-section at $t = 0.003, 0.122, 0.1706$ and 0.1907 ms (time steps $N = 10, 410, 660$ and 1190).

very little role). Since the flow contains several regions of stagnant fluid that are not connected to the interface, these regions also correspond to locations where the temperature does not rise significantly.

The shape of the grain at 4 different times ($t = 0.003, 0.122, 0.1706$ and 0.1907 ms) is shown in Fig. 19. We observe that the grain retains its initial shape through most of the process. However, towards the end we begin to see a reduction in its thickness along its center which corresponds to the slightly higher pressure that is observed in this region.

The mesh for this problem along a cross-section for the same time instances is shown in Fig. 20. We undertake 5 mesh modification steps for the total of 1190 time steps that this problem is run. In every mesh modification step we maintain three layers of boundary layer mesh outside and two layers inside the grain.

For this problem, the total mass and energy of the system are conserved during the simulation, while the momentum is not. In Fig. 21, we have plotted the variation of these quantities for each phase, and the total system, as a function of time. As the simulation proceeds the mass and the energy of the system are transferred from the liquid to the gas phase. Since the energy density of the liquid phase is significantly larger, the cross-over point for the total energy, that is the time where the total amount of the conserved quantity in the gas phase exceeds that of the liquid phase, is later than the cross-over point for the total mass. The error in conservation during the course of the simulation for mass is 3.1% and for energy it is 7.2%. This error can be reduced by increasing the accuracy of the time-integrator or by simply reducing the time-step in the simulation.

4.4. Phase change of multiple grains

Finally, we consider the phase change and burning of several (six) cylindrical grains. Each grain is a 2 mm long circular cylinder capped with hemispherical end caps of 0.5 mm radius. The grains are placed symmetrically in a cylindrical chamber that is 8.4 mm long and has a radius of 1.9 mm. The material properties of the grains are the same as in the previous example and as before the chamber has adiabatic boundary conditions.

The mesh used to solve this problem comprises of around 6.2 million linear tetrahedral and wedge elements. At the interface, on the side of the propellant grain, there are two layers of boundary layer mesh, with a total thickness of 2.2×10^{-5} m. On the gas side, there are three layers of boundary layer mesh with a total thickness of 3.64×10^{-5} m.

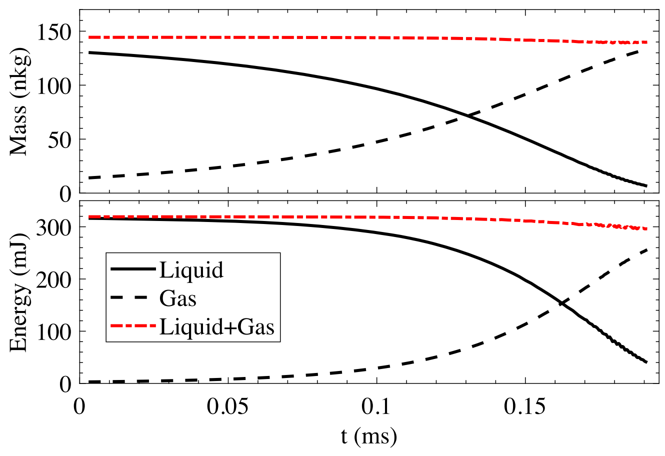


Fig. 21. Variation of the total mass and energy in the gas, the grain, and the total system as a function of time.

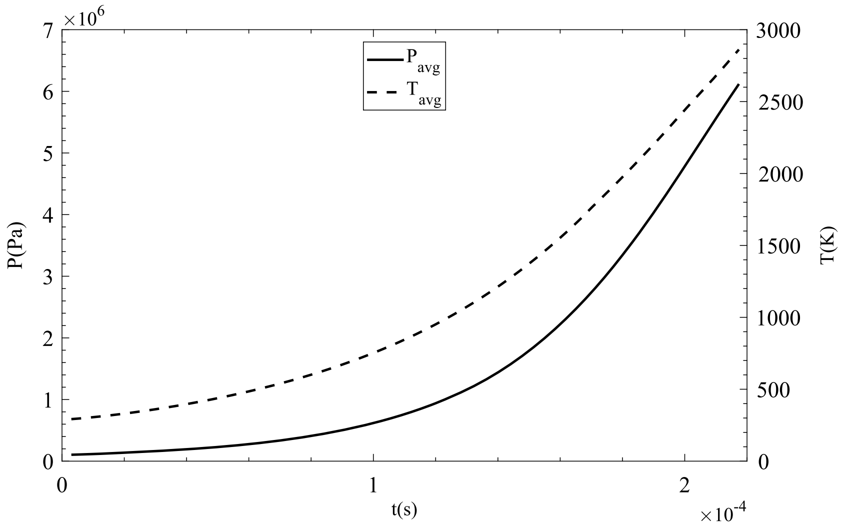


Fig. 22. Average pressure and temperature in the gas as a function of time.

The phase change process commences at 1 atm and increases exponentially until the interface area starts to reduce significantly. The variation of the average pressure and temperature in the gas phase is shown in Fig. 22 where, as for the one grain case, we observe a ramp up period followed by exponential increase in pressure and temperature. We terminate the simulation when the grains reduce to about 6% of their initial volume.

The distribution of pressure at the end of the simulation is shown in Fig. 23. From this figure observe that the pressure within the gas phase is quite uniform and varies within 0.8 to 1.3 of its average value. We also observe no sustained trends in the pressure distribution, primarily due to the multiple scattering of strong acoustic waves within the chamber.

The velocity distribution at the end of the simulation can be observed in a plot of the line integral convolution (LIC) of velocity in Fig. 24. In this figure we observe a very complex flow pattern that is formed from the interaction of the jet of fluid emanating from each grain with each other and with the surface of chamber.

The corresponding temperature distribution is shown in Fig. 25. In this figure we observe that the temperature distribution is much more uniform when compared with the single grain case. We still observe pockets of low temperature that probably correspond to pools of stagnant gas, however the extent of these regions is much smaller than in Fig. 18.

The evolution of the shape of the grains is shown in Fig. 26 for $t = 0.003, 0.132, 0.1886$ and 0.2174 ms. Once again we observe that for a significant duration of the simulation the grains shrink while retaining their shape. However, towards the end they appear to become thinner along their centers when compared to their ends. Further, towards the end they also display asymmetry in their orientation with respect to each other.

In Fig. 27 we have shown a close-up of the mesh around one of the grains at four different instances. The mesh was modified at least once between each of these instances. We observe that through each modification cycle the boundary layer on either side of the phase interface has been preserved. We also note that even though the total simulation was

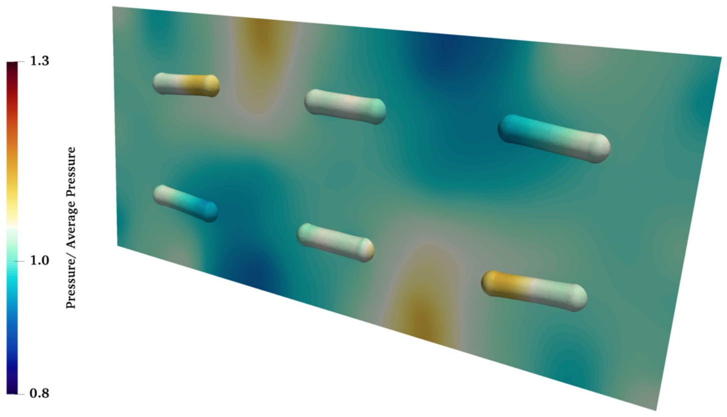


Fig. 23. Pressure, scaled by its average value in the chamber, at $t = 0.2174$ ms for the six grains problem (time step $N = 1620$).

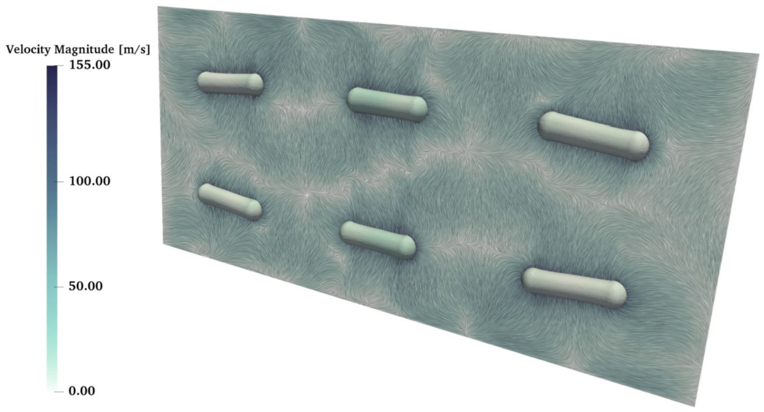


Fig. 24. Line integral convolution (LIC) of velocity at $t = 0.2174$ ms for the six grains problem (time step $N = 1620$).

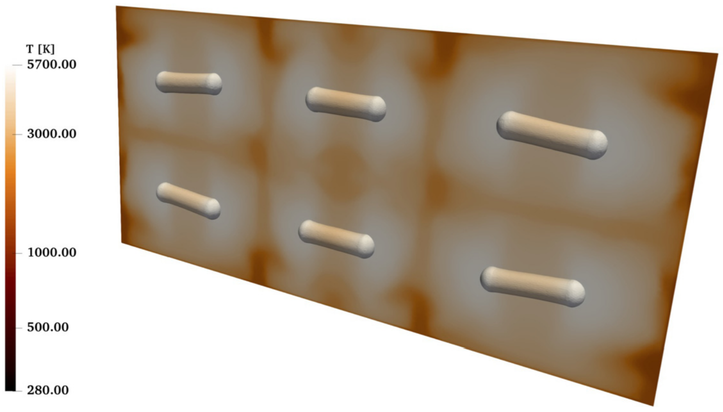


Fig. 25. Temperature distribution at $t = 0.2174$ ms for the six grains problem (time step $N = 1620$).

run for 1620 time steps, the mesh was modified only a total of 5 times. Thus the major portion of interface motion and deformation was accommodated via mesh motion, which significantly less expensive that mesh modification.

In Fig. 28, we have plotted the variation of the total mass and energy for each phase, and the total system, as a function of time. Once again, we observe that as the simulation proceeds the mass and the energy of the system are transferred from the liquid to the gas phase, and that the cross-over time for the total energy is larger than for the total mass. The error in conservation during the course of the simulation for mass is 1.3% and for energy it is 2.1%.

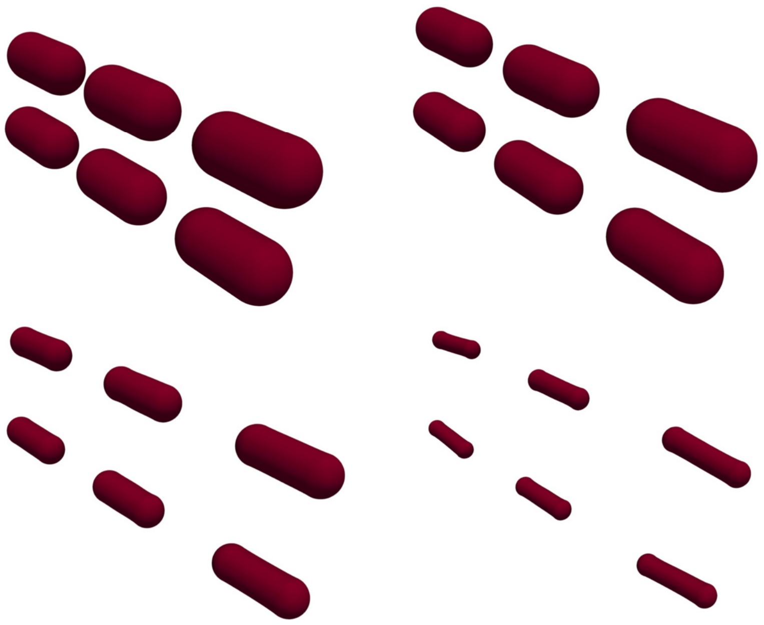


Fig. 26. Shape of the grains at $t = 0.003, 0.132, 0.1886$ and 0.2174 ms.

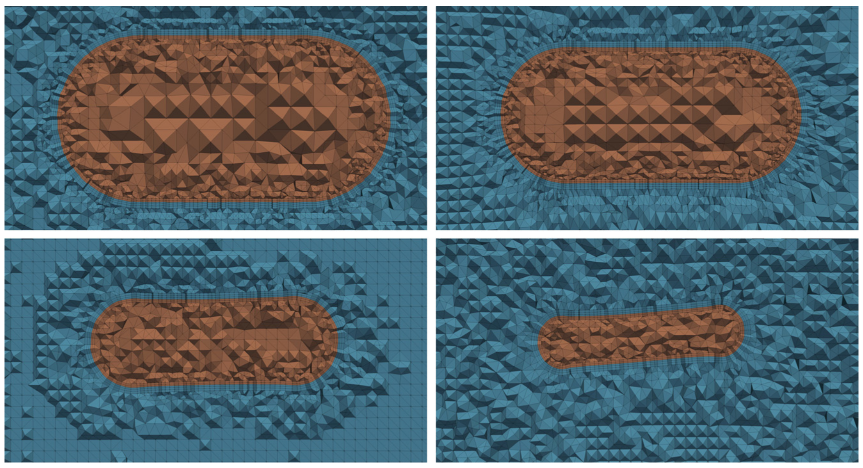


Fig. 27. Mesh along the cross-section of a grain at $t = 0.003, 0.132, 0.1886$ and 0.2174 ms.

5. Conclusions

In this manuscript we have developed and implemented a finite element method to model compressible fluid phase change problems. The finite element interpolations are continuous everywhere except at the interface, where discontinuous interpolations are used to model the naturally occurring strong discontinuities. In this regard the method can be considered a combination of standard and discontinuous finite element methods and it inherits its stability from the SUPG and interior penalty methods. Within this method, at the interface, appropriate flux conditions as well as certain continuity conditions are imposed weakly. It is also shown that the proposed method possess certain desirable discrete conservation properties.

The interface between the two phases is a represented by a surface mesh which is evolved in time. The volumetric mesh is allowed to deform in response to this motion in analogy to a linear elastic solid. The motion of the mesh is accounted for through the ALE formulation. Mesh modification is performed only when it is found that the mesh motion alone is unable to deliver a mesh of desired quality.

This formulation is implemented in a compressible Navier Stokes finite element code within an implicit predictor-corrector time integrator. This is particularly useful in working the disparate time scales of interface motion and acoustic waves while still retaining stability.

The performance of this method is verified using a simple Stefan's problem for which an approximate analytical solution is computed. Furthermore, for this problem and its counterpart in spherical coordinates, the convergence of the errors in

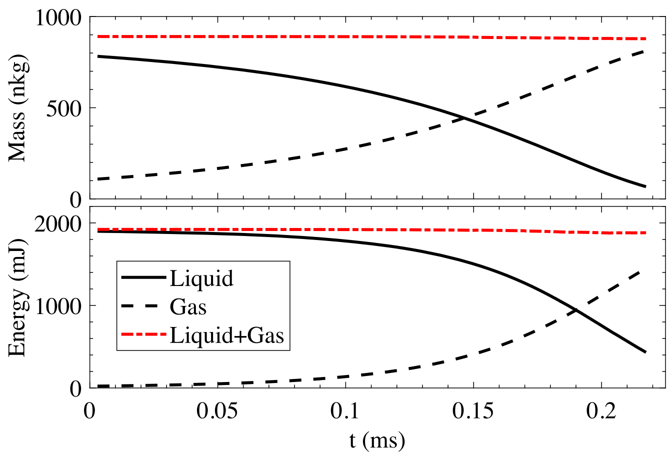


Fig. 28. Variation of the total mass and energy in the gas, the grain, and the total system as a function of time.

enforcing the interface flux and continuity conditions with decreasing mesh size is quantified. Thereafter this method is applied to simulating the combustion of a single and several grains of a dense energetic material enclosed in a closed chamber with adiabatic walls. Interesting results that describe the change in the shape of the grains, the evolution of average pressure and temperature, and non-uniformity in the temperature field are obtained.

While the method presented in this paper is promising, we believe that there are several ways in which it can be improved, refined and extended to solve additional compressible phase change problems. These include the addition of a surface tension term in order to simulate the interaction and collapse of vapor bubbles in water. The extension of the gas phase description to include multiple components (air, gaseous propellant and the products of combustion, for example) so that processes that occur after phase change, like the combustion of the gaseous phase can be modeled more accurately. In addition, there are several interesting numerical extensions that can be explored as well. These include addition of interface stabilization terms that are the interface analogs of the SUPG and discontinuity capturing operators. These would be particularly useful for flows with strong shocks that run through interfaces.

Finally, we note that the methods that explicitly track interfaces, like the one presented here, are either limited to cases where the change in topology of the phase interfaces do not occur (examples in the current paper) or must include techniques that explicitly account for the topological changes. Developments required to support topological changes include procedures needed to track the geometry and mesh for an accurate detection of an imminent topological change, and procedures needed to update the geometry and mesh during a topological change (see [26] for an application in crack propagation). It is worth noting that although the mesh updates required in cases involving topological changes are more extensive as compared to cases with shape changes, such mesh updates can still be carried out by local mesh modification operations.

Acknowledgements

This work is supported by the U.S. Army grants W911NF1410301 and W911NF16C0117.

References

[1] Gilou Agbaglah, Sébastien Delaux, Daniel Fuster, Jérôme Hoepffner, Christophe Josserand, Stéphane Popinet, Pascal Ray, Ruben Scardovelli, Stéphane Zaleski, Parallel simulation of multiphase flows using octree adaptivity and the volume-of-fluid method, C. R., Méc. 339 (2011) 194–207.
[2] Douglas N. Arnold, An interior penalty finite element method with discontinuous elements, SIAM J. Numer. Anal. 19 (4) (1982) 742–760.
[3] Douglas N. Arnold, Franco Brezzi, Bernardo Cockburn, L. Donatella Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM J. Numer. Anal. 39 (5) (2002) 1749–1779.
[4] V.E. Badalassi, H.D. Ceniceros, Sanjoy Banerjee, Computation of multiphase systems with phase field models, J. Comput. Phys. 190 (2) (2003) 371–397.
[5] Jozsef Bakosi, Jacob Waltz, Nathaniel Morgan, Improved ale mesh velocities for complex flows, Int. J. Numer. Methods Fluids 85 (11) (2017) 662–671.
[6] Carlos Erik Baumann, J. Tinsley Oden, A discontinuous *hp* finite element method for convection–diffusion problems, Comput. Methods Appl. Mech. Eng. 175 (3–4) (1999) 311–341.
[7] William J. Boettinger, James A. Warren, Christoph Beckermann, Alain Karma, Phase-field simulation of solidification, Annu. Rev. Mater. Res. 32 (1) (2002) 163–194.
[8] B. Bunner, G. Trygvasson, Effect of bubble deformation on the properties of bubbly flows, J. Fluid Mech. 495 (2003) 77–118.
[9] Peter Chadwick, Continuum Mechanics: Concise Theory and Problems, Courier Corporation, 2012.
[10] Chih-Hao Chang, Meng-Sing Liou, A robust and accurate approach to computing compressible multiphase flow: stratified flow model and AUSM$^+$-up scheme, J. Comput. Phys. 225 (1) (2007) 840–873.
[11] Jintai Chung, G.M. Hulbert, A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method, J. Appl. Mech. 60 (2) (1993) 371–375.
[12] Paul W. Cooper, Stanley R. Kurowski, Paul W. Cooper, Introduction to the Technology of Explosives, VCH, New York, 1996.

[13] Anastasios Georgoulas, Manolia Andredaki, Marco Marengo, An enhanced VOF method coupled with heat transfer and phase change to characterise bubble detachment in saturated pool boiling, Energies 10 (3) (2017) 272.
[14] James Glimm, John W. Grove, Xiao Lin Li, Keh-ming Shyue, Yanni Zeng, Qiang Zhang, Three-dimensional front tracking, SIAM J. Sci. Comput. 19 (3) (1998) 703–727.
[15] G. Hauke, Simple stabilizing matrices for the computation of compressible flows in primitive variables, Comput. Methods Appl. Mech. Eng. 190 (51–52) (2001) 6881–6893.
[16] G. Hauke, T.J.R. Hughes, A unified approach to compressible and incompressible flows, Comput. Methods Appl. Mech. Eng. 113 (3–4) (1994) 389–395.
[17] Guillermo Hauke, Thomas J.R. Hughes, A comparative study of different sets of variables for solving compressible and incompressible flows, Comput. Methods Appl. Mech. Eng. 153 (1–2) (1998) 1–44.
[18] Howard H. Hu, Neelesh A. Patankar, M.Y. Zhu, Direct numerical simulations of fluid–solid systems using the arbitrary Lagrangian–Eulerian technique, J. Comput. Phys. 169 (2) (2001) 427–462.
[19] Thomas J.R. Hughes, Recent progress in the development and understanding of SUPG methods with special reference to the compressible Euler and Navier-Stokes equations, Int. J. Numer. Methods Fluids 7 (11) (1987) 1261–1275.
[20] Thomas J.R. Hughes, Wing Kam Liu, Thomas K. Zimmermann, Lagrangian-Eulerian finite element formulation for incompressible viscous flows, Comput. Methods Appl. Mech. Eng. 29 (3) (1981) 329–349.
[21] Thomas J.R. Hughes, Guglielmo Scovazzi, Pavel B. Bochev, Annalisa Buffa, A multiscale discontinuous Galerkin method with the computational structure of a continuous Galerkin method, Comput. Methods Appl. Mech. Eng. 195 (19–22) (2006) 2761–2787.
[22] T.J.R. Hughes, M. Mallet, A new finite element formulation for computational fluid dynamics: IV. A discontinuity-capturing operator for multidimensional advective-diffusive systems, Comput. Methods Appl. Mech. Eng. 58 (3) (1986) 329–336.
[23] D.A. Ibanez, E. Love, T.E. Voth, J.R. Overfelt, N.V. Roberts, G.A. Hansen, Tetrahedral mesh adaptation for lagrangian shock hydrodynamics, Comput. Math. Appl. (2018).
[24] David Jacqmin, Calculation of two-phase Navier–Stokes flows using phase-field modeling, J. Comput. Phys. 155 (1) (1999) 96–127.
[25] Kenneth E. Jansen, Christian H. Whiting, Gregory M. Hulbert, A generalized-α method for integrating the filtered Navier–Stokes equations with a stabilized finite element method, Comput. Methods Appl. Mech. Eng. 190 (3–4) (2000) 305–319.
[26] O. Klass, R. Nastasia, M. Beall, A. Loghin, J. Laflen, J. LeMonds, Robust tools for crack insertion and propagation, in: NAFEMS World Congress, Boston, USA, 2011.
[27] Patrick Knupp, Len G. Margolin, Mikhail Shashkov, Reference Jacobian optimization-based rezone strategies for arbitrary Lagrangian Eulerian methods, J. Comput. Phys. 176 (1) (2002) 93–128.
[28] Bruno Lafaurie, Carlo Nardone, Ruben Scardovelli, Stéphane Zaleski, Gianluigi Zanetti, Modelling merging and fragmentation in multiphase flows with surfer, J. Comput. Phys. 113 (1) (1994) 134–147.
[29] R.J. LeVeque, Numerical Methods for Conservation Laws, 2nd edition, Birkhauser Verlag, Basel, 1992.
[30] Xiangrong Li, Mark S. Shephard, Mark W. Beall, 3d anisotropic mesh adaptation by mesh modification, Comput. Methods Appl. Mech. Eng. 194 (48) (2005) 4915–4950.
[31] Jian-Yu Lin, Yi Shen, Hang Ding, Nan-Sheng Liu, Xi-Yun Lu, Simulation of compressible two-phase flows with topology change of fluid–fluid interface by a robust cut-cell method, J. Comput. Phys. 328 (2017) 140–159.
[32] Rainald Löhner, Chi Yang, Improved ale mesh velocities for moving bodies, Commun. Numer. Methods Eng. 12 (10) (1996) 599–608.
[33] S. Nagrath, K. Jansen, R.T. Lahey, I. Akhatov, Hydrodynamic simulation of air bubble implosion using a level set approach, J. Comput. Phys. 215 (1) (2006) 98–132.
[34] S. Nagrath, K.E. Jansen, R.T. Lahey, Computation of incompressible bubble dynamics with a stabilized finite element level set method, Comput. Methods Appl. Mech. Eng. 194 (42–44) (2005) 4565–4587.
[35] M.S. Patil, Haridwar Singh, Ballistic and mechanical properties of HTPB based composite propellants, J. Hazard. Mater. 19 (3) (1988) 271–278.
[36] Yuriko Renardy, Michael Renardy, PROST: a parabolic reconstruction of surface tension for the volume-of-fluid method, J. Comput. Phys. 183 (2) (2002) 400–421.
[37] Franco Rispoli, Alessandro Corsini, Tayfun E. Tezduyar, Finite element computation of turbulent flows with the discontinuity-capturing directional dissipation (DCDD), Comput. Fluids 36 (1) (2007) 121–126.
[38] Onkar Sahni, Kenneth E. Jansen, Mark S. Shephard, Charles A. Taylor, Mark W. Beall, Adaptive boundary layer meshing for viscous flow simulations, Eng. Comput. 24 (3) (2008) 267–285.
[39] Onkar Sahni, Aleksandr Ovcharenko, Kedar C. Chitale, Kenneth E. Jansen, Mark S. Shephard, Parallel anisotropic mesh adaptation with boundary layers for automated viscous flow simulations, Eng. Comput. 33 (4) (2017) 767–795.
[40] Richard Saurel, Olivier Lemetayer, A multiphase model for compressible flows with interfaces, shocks, detonation waves and cavitation, J. Fluid Mech. 431 (2001) 239–271.
[41] Guglielmo Scovazzi, A discourse on Galilean invariance, SUPG stabilization, and the variational multiscale framework, Comput. Methods Appl. Mech. Eng. 196 (4–6) (2007) 1108–1132.
[42] Guglielmo Scovazzi, Galilean invariance and stabilized methods for compressible flows, Int. J. Numer. Methods Fluids 54 (6–8) (2007) 757–778.
[43] Guglielmo Scovazzi, Edward Love, A generalized view on galilean invariance in stabilized compressible flow computations, Int. J. Numer. Methods Fluids 64 (10–12) (2010) 1065–1083.
[44] Farzin Shakib, Thomas J.R. Hughes, Johan Zdeněk, A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier-Stokes equations, Comput. Methods Appl. Mech. Eng. 89 (1) (1991) 141–219.
[45] Keith Stein, T. Tezduyar, Richard Benney, Mesh moving techniques for fluid-structure interactions with large displacements: flow simulation and modeling, J. Appl. Mech. 70 (1) (2003) 58–63.
[46] M. Sussman, A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, M.L. Welcome, An adaptive level set approach for incompressible two-phase flows, J. Comput. Phys. 148 (1) (1999) 81–124.
[47] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1) (1994) 146–159.
[48] Mark Sussman, Elbridge Gerry Puckett, A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows, J. Comput. Phys. 162 (2) (2000) 301–337.
[49] T.E. Tezduyar, Y.J. Park, Discontinuity-capturing finite element formulations for nonlinear convection-diffusion-reaction equations, Comput. Methods Appl. Mech. Eng. 59 (3) (1986) 307–325.
[50] P.D. Thomas, C.K. Lombard, Geometric conservation law and its application to flow computations on moving grids, AIAA J. 17 (10) (1979) 1030–1037.
[51] Grétar Tryggvason, Bernard Bunner, Asghar Esmaeeli, Damir Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, Y-J. Jan, A front-tracking method for the computations of multiphase flow, J. Comput. Phys. 169 (2) (2001) 708–759.
[52] Salih Ozen Unverdi, Grétar Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, J. Comput. Phys. 100 (1) (1992) 25–37.
[53] Jacob Waltz, Nathaniel R. Morgan, Thomas R. Canfield, Marc R.J. Charest, L.D. Risinger, John G. Wohlbier, A three-dimensional finite element arbitrary Lagrangian–Eulerian method for shock hydrodynamics on unstructured grids, Comput. Fluids 92 (2014) 172–187.
[54] Ali Zein, Maren Hantke, Gerald Warnecke, Modeling phase transition for compressible two-phase flows applied to metastable liquids, J. Comput. Phys. 229 (8) (2010) 2964–2998.

An Automated Approach for Parallel Adjoint-Based Error Estimation and Mesh Adaptation

Brian N. Granzow[a,*], Assad A. Oberai[c], Mark S. Shephard[b]

[a]*Sandia National Laboratories*
P.O. Box 5800
Albuquerque, NM 87185-1321
[b]*Scientific Computation Research Center*
Rensselaer Polytechnic Institute
110 8th Street
Troy, NY 12180
[c]*Aerospace and Mechanical Engineering*
University of Southern California,
Los Angeles, CA 90089

Abstract

In finite element simulations, not all of the data is of equal importance. In fact, the primary purpose of a numerical study is often to accurately assess only one or two engineering output quantities that can be expressed as functionals. Adjoint-based error estimation provides a means to approximate the discretization error in functional quantities and mesh adaptation provides the ability to control this discretization error by locally modifying the finite element mesh. In the past, adjoint-based error estimation has only been accessible to expert practitioners in the field of solid mechanics. In this work, we present an approach to automate the process of adjoint-based error estimation and mesh adaptation on parallel machines. This process is intended to lower the barrier of entry to adjoint-based error estimation and mesh adaptation for solid mechanics practitioners. We demonstrate that this approach is effective for example problems in Poisson's equation, nonlinear elasticity, and thermomechanical elastoplasticity.

Keywords: automated, adjoint, a posteriori, error estimation, adaptation, finite element

1. Introduction

Numerical simulation has become ubiquitous in engineering and scientific practice due to the continuing increase in power and accessibility of computational resources. For scientific and engineering applications, ensuring the accuracy and reliability of the computed numerical solution is of primary importance. For example, it is often necessary to design structural components for which the von Mises stress in the component's domain is less than a given material yield strength when subjected to a variety of loading conditions.

In the context of finite element methods, *a posteriori* error estimation and mesh adaptation provide useful tools for approximating and controlling the discretization error when solving partial differential equations (PDEs) (*cf.* [1, 26, 61]).

In an abstract finite element setting, one seeks to solve the variational problem: find $u \in \mathcal{S}$ such that

$$\mathcal{R}(u; w) = 0 \quad \forall w \in \mathcal{V}, \tag{1}$$

where $\mathcal{R} : \mathcal{S} \times \mathcal{V} \to \mathbb{R}$ is a semilinear form, potentially nonlinear in its first argument and linear in its second, and $\mathcal{S}$ and $\mathcal{V}$ are Hilbert spaces. Given the exact solution u to this problem and a corresponding finite element

*Corresponding author, bngranz@sandia.gov approximation u^H, the discretization error is defined as $e := u - u^H$. Traditional *a posteriori* error estimates attempt to bound or approximate the discretization error in a global norm $\|e\|$, such as the energy norm induced by the underlying partial differential operator. In the past twenty years, though, adjoint-based techniques have been developed and utilized to obtain approximations η or approximate bounds $\hat{\eta}$ for the error $|J(u) - J(u^H)|$ in some physically meaningful functional $J : \mathcal{S} \to \mathbb{R}$, such that

$$|J(u) - J(u^H)| \approx \eta < \hat{\eta}. \tag{2}$$

The use of adjoint (or duality) techniques for *a posteriori* error estimation can be traced back to Babuška and Miller [3, 4, 5] who explored the post-processing of functional quantities and Gartland [18] who explored estimating point-wise errors. These ideas were then expanded to the context of adaptive finite element methods by Johnson et al. [15]. Becker and Rannacher [8] developed an approach to functional *a posteriori* error estimation for Galerkin finite element methods called the *dual weighted residual* method. Giles and Pierce [21] developed an approach conceptually similar to the dual weighted residual method, but additionally concerned themselves with discretizations that lack Galerkin orthogonality, such as Godunov finite volume methods. Venditti and Darmofal [58, 59, 60] developed adjoint-based error estimates for arbitrary discretizations using discrete adjoint equations based on two-level discretization schemes. Prudhomme and Oden [42, 37] developed adjoint techniques to determine guaranteed upper and lower bounds for linear functionals in the context of linear variational problems.

Adjoint-based error estimation and mesh adaptation have been heavily adopted by the computational fluid dynamics (CFD) community [17]. This can in part be explained by the fact that the current increase in computing power alone may not be sufficient to accurately resolve quantities of interest (QoIs) in CFD applications, even when very finely generated *a priori* meshes are used [17]. However, despite its popularity in CFD, adjoint-based error estimation is not regularly applied to solid mechanics applications.

In the context of solid mechanics, adaptive adjoint-based error estimation has been used to study linear elasticity in two [45, 55, 22] and three [20] dimensions, two [46, 47] and three [19] dimensional elasto-plasticity, two dimensional thermoelasticity [43], two dimensional nonlinear elasticity [32], and two dimensional hyperelasticity [62]. We remark that in the majority of this literature, mesh adaptation is performed with structured adaptive mesh refinement using quadrilateral or hexahedral elements and without any discussion of parallelization. Additionally, none of these investigations apply the current state of the art capabilities of parallel three-dimensional unstructured mesh adaptation, as is currently the norm for CFD applications.

Perhaps one reason for this discrepancy is the fact that adjoint-based error estimation requires the development and implementation of a number of non-trivial steps. From a high-level, the following steps must be carried out to perform an adaptive adjoint-based error analysis:

1. Solve the original (primal) PDE of interest.
2. Using the primal solution, derive and solve an auxiliary adjoint PDE.
3. Enrich the solution to the adjoint PDE in some manner.
4. Compute error estimates using an adjoint-weighted residual method.
5. Localize the error to contributions at the mesh entity level.
6. Adapt the mesh based on the local error contributions.

Additionally, modern solid mechanics applications necessitate the use of parallel analysis, meaning each of these steps must execute effectively and efficiently on parallel machines.

Further, both the primal residual and functional QoI can be highly nonlinear in solid mechanics. Solving the primal and adjoint problems requires the linearization of the primal residual and the functional QoI with respect to the degrees of freedom of the problem. The derivation and numerical implementation of these linearizations can be time consuming and error-prone. As a result, adjoint-based error control has typically remained a tool for expert analysis with a high barrier of entry.

As a final observation, we note that unstructured mesh generation and adaptation is robust, reliable, and scalable for simplicial elements. However, for solid mechanics applications with incompressibility constraints, such as isochoric plasticity or incompressible hyperelasticity, standard displacement-based Galerkin finite element discretizations are known to be unstable when using simplicial elements. This fact may have further hindered the adoption of unstructured mesh adaptation for previous adaptive adjoint-based solid mechanics applications.

To make adjoint-based error estimation and mesh adaptation more accessible to solid mechanics practitioners, we seek to fully automate its steps for execution on parallel machines. Specifically, we seek to develop software that automates steps 1-6 outlined above based solely on the inputs of a semilinear form $\mathcal{R}$ and a functional QoI J. We endeavor to develop this software to be applicable to both Galerkin and stabilized finite element methods.

Recently, Rognes and Logg [50] introduced a fully automated approach to goal-oriented error estimation and mesh adaptation for Galerkin finite element methods within the FEniCS [35] finite element framework. In this approach, the adjoint problem is derived in a discrete manner based on a user-implemented residual $\mathcal{R}$ and a functional J. The adjoint problem is then solved on the same finite element space as used for the primal problem and enriched to a higher order polynomial space by solving local patch-wise problems. Based on the given semilinear form $\mathcal{R}$, error contributions are then localized to the element-level by solving local element problems to recover the strong form of the residual operator over element interiors and element boundaries. The total error in the functional QoI is then computed as the sum of these error contributions. As a final step, the mesh is adapted using conforming unstructured mesh *refinement*.

In this work, we present an approach for automating goal-oriented analysis that is distinct in several ways. First, we consider adjoint-based error estimation in the context of both Galerkin and stabilized finite element methods. Additionally, we propose solving the adjoint problem in a richer finite element space, obtained via uniform refinement, than the space used for the primal problem. To localize the error to the mesh entity level, we utilize a partition of unity based approach proposed by Richter and Wick [49]. This allows us to directly re-use the implemented semilinear form $\mathcal{R}$ for error localization, eliminating the need to solve local element problems to recover the strong form residual. We also take advantage of fully unstructured conforming mesh adaptation, where the mesh can be *coarsened* as well as refined. As a final distinction, we highlight the ability of the proposed approach to execute on parallel machines.

In totality, our new approach can be described as follows. First, the primal problem is solved via Newton's method, where the Jacobian of the semilinear form $\mathcal{R}$ is obtained via automatic differentiation. The adjoint problem is derived in a discrete manner in a richer finite element space obtained by a uniform refinement of the initial mesh. The adjoint operator is derived by an application of automatic differentiation to the semilinear form $\mathcal{R}$, and the right-hand side of the adjoint problem is obtained by applying automatic differentiation to the functional J. An approximate error η is computed as a modified discrete adjoint weighted residual evaluated on the fine space. The error is then localized to mesh vertices using a variational localization approach by introducing a partition of unity into the weighting slot of the semilinear form $\mathcal{R}$. An estimator of the error bound $\hat{\eta}$ is obtained by summing the absolute values of localized error contributions. Finally, the mesh is adapted by specifying a *mesh size field*, which defines the length of mesh edges over the mesh. We have implemented this approach in a `C++` finite element application which we have called Goal [23].

Underlying this approach is the concept of *template-based generic programming* (TBGP) [38, 39], which has previously been used to automate the solution of PDEs as well as embedded advanced analysis features, such as sensitivity analysis and uncertainty quantification. From a high-level the TBGP approach consists of a *gather* phase, a *compute* phase, and a *scatter* phase. The present work extends the TBGP approach to include the automation of error localization, as required to drive mesh adaptation.

The remainder of this paper is outlined as follows. First adjoint-based error estimation is reviewed for abstract Galerkin and stabilized variational problems. Next, a description of the software components utilized in this work is provided. In particular, each step of the automated adjoint-based analysis is discussed with respect to its utilized software components. A review of the concept of TBGP is then provided and its extension for the purposes of adjoint-based error estimation is discussed. A detailed description of each step in the adaptive adjoint-based process is then described. First the automated solution of the primal problem is discussed. Then the automated derivation and solution of the adjoint problem is described. Next, the automation of error localization to drive mesh adaptation is outlined and mesh adaptation procedures are discussed. The implementation of several QoIs in the Goal application is reviewed. Finally, the effectiveness of the proposed automated approach is demonstrated for several applications.

2. A Review of Adjoint-Based Error Representations

In this section, a brief review of the derivation of adjoint-based error representations is provided for Galerkin finite element methods as outlined by Becker and Rannacher [8], and for stabilized finite element methods as outlined by Cyr et al. [12]. This review is intended to give context and serve as a road map for the remaining sections in this chapter.

Let $\mathcal{S}$ and $\mathcal{V}$ be Hilbert spaces, $\mathcal{R}_g : \mathcal{S} \times \mathcal{V} \to \mathbb{R}$ and $\mathcal{R}_\tau : \mathcal{S} \times \mathcal{V} \to \mathbb{R}$ be semilinear forms. Here the subscript g is used to denote a residual that corresponds to a standard Galerkin method and the subscript τ is used to denote a residual that arises due to numerical stabilization. Let $\mathcal{S}^H \subset \mathcal{S}$ and $\mathcal{V}^H \subset \mathcal{V}$ be classical finite element function spaces, where H is a mesh-dependent parameter that denotes the fineness of the discretization. We introduce the following variational abstract model problem: find $u \in \mathcal{S}$ such that

$$\mathcal{R}_g(u; w) = 0 \quad \forall w \in \mathcal{V}. \tag{3}$$

Similarly, we introduce the following abstract *adjoint problem*: find $z \in \mathcal{V}$ such that

$$\mathcal{R}_g'[u^H](w, z) = J'[u^H](w) \quad \forall w \in \mathcal{V}, \tag{4}$$

where the prime indicates Fréchet linearization about the argument in square brackets, which can equivalently be expressed as the Gâteaux derivatives

$$J'[u^H](w) := \lim_{\epsilon \to 0} \frac{J(u^H + \epsilon w) - J(u^H)}{\epsilon} \tag{5}$$

and

$$\mathcal{R}_g'[u^H](w, z) := \lim_{\epsilon \to 0} \frac{\mathcal{R}_g(u^H + \epsilon w; z)}{\epsilon} \tag{6}$$

Here, $u^H \in \mathcal{S}^H$ denotes some finite element approximation to the true solution u. The purpose of the adjoint problem is to relate the original problem of interest to the functional quantity J, and it is this relationship that allows us to derive adjoint-based error representations. Further, we note that the adjoint solution z can be interpreted as the sensitivity of the QoI to perturbations in the primal PDE residual [17].

2.1. *Galerkin Finite Element Methods*

The corresponding Galerkin finite element formulation of the abstract problem (3) can be stated as: find $u^H \in \mathcal{S}^H$ such that

$$\mathcal{R}_g(u^H; w^H) = 0 \quad \forall w^H \in \mathcal{V}^H. \tag{7}$$

Let $e := u - u^H$ denote the discretization error. We can then derive an error representation for the functional J in terms of the adjoint solution z as follows:

$$\begin{aligned} J(u) - J(u^H) &= J'[u^H](e) + \mathcal{O}(e^2) \\ &= \mathcal{R}_g'[u^H](e, z) + \mathcal{O}(e^2) \\ &= \mathcal{R}_g(u; z) - \mathcal{R}_g(u^H; z) + \mathcal{O}(e^2) \\ &= -\mathcal{R}_g(u^H; z) + \mathcal{O}(e^2) \\ &= -\mathcal{R}_g(u^H; z - z^H) + \mathcal{O}(e^2). \end{aligned} \tag{8}$$

Here, the first equality is due to the linearization [8] of the functional J, the second equality is due to the introduced adjoint problem (4), the third equality is due to the linearization [8] of the residual $\mathcal{R}_g$, the fourth equality holds due to the definition of the abstract primal problem (3), and the fifth equality is due to Galerkin orthogonality, where z^H denotes the projection of z onto the space $\mathcal{V}^H$.

2.2. *Stabilized Finite Element Methods*

A corresponding stabilized finite element method of the abstract problem (3) can be expressed as: find $u^H \in \mathcal{S}^H$ such that

$$\mathcal{R}_g(u^H; w^H) + \mathcal{R}_\tau(u^H; w^H) = 0 \quad \forall w^H \in \mathcal{V}^H. \tag{9}$$

Here $\mathcal{R}_\tau$ denotes a consistent *stabilization residual* that adds stability to the numerical scheme. We say that the stabilization is *consistent* if $\mathcal{R}_\tau(u; w^H) \to 0$ as $H \to 0$.

Again, we let $e := u - u^H$ denote the discretization error, and derive an error representation for the functional J as follows

$$\begin{aligned} J(u) - J(u^H) &= J'[u^H](e) + \mathcal{O}(e^2) \\ &= \mathcal{R}'_g[u^H](e, z) + \mathcal{O}(e^2) \\ &= \mathcal{R}_g(u; z) - \mathcal{R}_g(u^H; z) + \mathcal{O}(e^2) \\ &= -\mathcal{R}_g(u^H; z) + \mathcal{O}(e^2) \\ &= -\mathcal{R}_g(u^H; z - z^H) + \mathcal{R}_\tau(u^H; z^H) + \mathcal{O}(e^2). \end{aligned} \tag{10}$$

Here, the first four equalities are obtained exactly as in the corresponding Galerkin finite element method. However, when we subtract z^H in the fifth equality, an additional term remains because the numerical scheme (9) lacks Galerkin orthogonality.

3. Software Components

An adaptive adjoint-based simulation requires the implementation and coordination of a number of non trivial components. Namely, the solution of a primal problem, the construction and solution of an auxiliary adjoint problem, an enrichment of the adjoint solution, the estimation and localization of the error, and mesh adaptation are all required steps in the adjoint-based adaptive process.

To implement each of these components for effective execution on parallel machines, we make use of two state of the art software suites. The first is PUMI [31], which contains tools to support unstructured mesh services on massively parallel machines. In particular, PUMI provides all of the necessary machinery to store, query, adapt, and dynamically load balance parallel unstructured meshes via a collection of modern `C` and `C++` libraries. The second is Trilinos [28, 29], which provides a large variety of `C++` packages to support multiphysics simulations on parallel machines. In particular, Trilinos provides the ability to store and solve sparse parallel linear systems, as well as tools to perform automatic differentiation.

Using these two software suites as building blocks, we have written a new `C++` application for adjoint-based error estimation and mesh adaptation with an emphasis on nonlinear solid mechanics. We have called this application Goal [23]. Below, we describe how these software components are utilized for each portion of the adaptive adjoint-based process, where the analysis is automated based only on the inputs of a semilinear form $\mathcal{R}$ and a functional QoI J.

3.1. *The Primal Problem*

Based on an implemented weighted residual operator $\mathcal{R}$, the Goal application computes element-level residual vectors and element-level Jacobian matrices. The element-level Jacobian matrices are computed via automatic differentiation using the Trilinos library Sacado. Sacado provides efficient automatic differentiation using a `C++` meta-programming technique called expression templates [40].

After the computation of a single element's residual vector and Jacobian matrix, the Goal application performs a finite element assembly step to sum contributions to the global residual vector and global Jacobian matrix. To store and modify the global linear algebra objects, we utilize the Tpetra library provided by Trilinos. In particular, the Jacobian matrix is stored as a sparse compressed row storage matrix in parallel.

The primal problem is solved via Newton's method, which requires iterative evaluations of the global residual vector and Jacobian matrix. For each Newton iteration, a global linear system must be solved.

We solve this linear system iteratively in parallel using either a CG or GMRES solver provided the Trilinos library Belos [7]. Additionally, we perform algebraic multigrid preconditioning using the Trilinos library MueLu [41].

Once the primal problem has been solved, we utilize the PUMI library APF to store the finite element solution information at nodes and if necessary secondary solution information at integration points. Additionally, the APF library is used to provide shape function information and to query stored solution information during residual and Jacobian evaluations. Throughout the entire solution process, the PUMI mesh data structure is utilized to query mesh specific information.

3.2. The Adjoint Problem

To solve the adjoint problem in a richer finite element space than the one used for the primal problem, we make use of the underlying PUMI mesh data structure [30] and the PUMI MeshAdapt software to create and store a uniformly refined nested mesh with parent-child relations back to the original mesh. This relational information is implemented in the Goal application, as it falls outside of the normal intended use case of the MeshAdapt software, which concerns itself with fully unstructured conforming mesh adaptation via edge splits, swaps, and collapses. However, the flexibility of the PUMI software allows us to additionally construct data structures similar to those used in traditional adaptive mesh refinement (AMR) with little implementation effort. Using the APF library and the parent-child relational information, we are able to interrogate stored solution fields on both the parent and nested meshes, which is required during the assembly of the adjoint problem.

On this finer mesh, element-level Jacobian matrices are computed using the Sacado library, based on the Goal implementation of the operator $\mathcal{R}$. Additionally, element-level derivatives of the functional quantity of interest J are computed with respect to degrees of freedom of the problem, resulting in an element-level functional derivative vector.

After the computation of a single element's Jacobian matrix and functional derivative vector, the Goal application performs a finite element assembly step to sum contributions to the global discrete adjoint matrix and the global functional derivative vector. Like the primal problem, these global parallel linear objects are stored using the Tpetra library. The global discrete adjoint operator and functional derivative vector fully define the linearized adjoint problem, which we again precondition with algebraic multigrid techniques using the MueLu library and solve using either CG or GMRES iterations using the Belos library. The fine-space adjoint solution is then attached to the mesh using the APF library.

3.3. Error Estimation and Localization

The error estimation and localization routines are implemented entirely in the Goal application. The error is localized by an evaluation of the stabilized weighted residual operator $\mathcal{R}$, where the weight is chosen to be the adjoint solution multiplied by a partition of unity. This error localization is discussed in further detail in Section 7. Based on these element-level residual vectors, Goal performs finite element assembly of the global residual vector, which is stored as a Tpetra vector. This vector represents an adjoint-weighted residual error estimate at each mesh vertex for each PDE equation in the fine mesh. The error is attached to the vertices of the fine mesh using the APF library.

3.4. Mesh Adaptation

Once the error is stored on the vertices of the fine mesh, we interpolate it to element centers of the coarse mesh. The fine mesh data structures are then destroyed and the Goal application computes a mesh size field that seeks to equidistribute the error for an output mesh with N elements. This mesh size field is given as the input to the PUMI MeshAdapt software, which adapts the mesh with a sequence of edge splits, swaps, and collapses [33, 2] to satisfy the given mesh size field. As a final step, we utilize the PUMI library ParMA [54, 13] to perform diffusive load balancing to ensure parallel partitioning quality.

3.5. In-Memory Integration of Components

The coupling of the software components described above is done *in-memory* [53]. That is, there is no file-based communication of data from one analysis component to the next in the automated process. This in-memory coupling is a key ingredient for parallel analysis, where filesystem bandwidth is a critical bottleneck.

4. Template-Based Generic Programming

In this section, we provide a review of the concept of *template-based generic programming* for the evaluation and solution of PDEs [38, 39] and how it has been extended in the Goal application to automate the process of adjoint-based error estimation and mesh adaptation. For PDE applications, TBGP is broken into three major components, a *seed* or *gather* phase, a *compute* phase, and a *scatter* phase. The seed and scatter operations must be programmed specifically for each evaluation purpose. In contrast, the compute phase, where the PDE and QoI expressions are implemented, are written in a totally generic manner. Figure 1 pictorially represents this design philosophy.

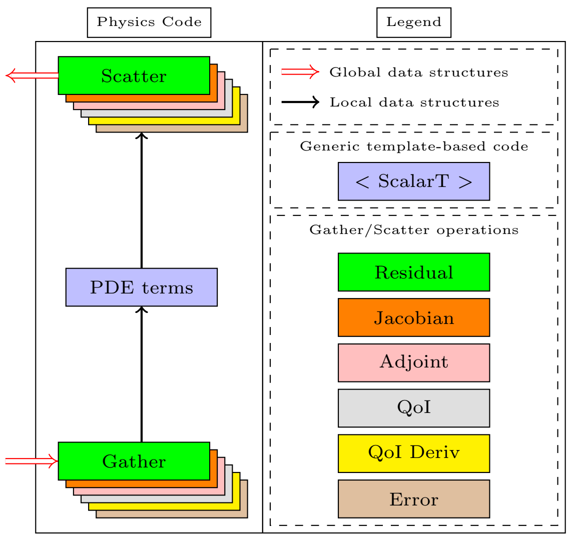


Figure 1: A schematic for the generic programming model of PDEs.

We invoke this approach at the element-level, meaning for each element we perform the process: Gather $\rightarrow$ Compute $\rightarrow$ Scatter. By doing so, we reduce memory overhead and eliminate complications introduced by parallel computation [39]. Underlying the TBGP approach is the use of forward automatic differentiation (FAD) [27]. The Goal application utilizes the Trilinos library Sacado [40] to perform automatic differentiation.

The purpose of the gather/seed operation is to collect information from global storage containers and 'gather' it to local element-level data structures. Further, any FAD derivative information is *seeded* during this operation, if necessary. For each evaluation type, the gather/seed operation initializes an array that physically represent the degrees of freedom associated with the current element, and initializes FAD variables' derivative arrays to physically represent derivatives with respect to the degrees of freedom associated with the current element, when necessary. This degree of freedom array is templated on a scalar type `ScalarT`. For the Residual, QoI, and Error evaluation types, this scalar type corresponds to a `C++` double. For the remaining evaluation types, this scalar type corresponds to a `Sacado::FAD` forward automatic differentiation variable type.

The compute phase computes local contributions to the equations or expressions of interest at the element level in terms of the degrees of freedom, as collected by the gather operation. The code for the compute

Evaluation Type	Scalar Type	Input	Output
Residual	`double`	$\boldsymbol{u}^H, \boldsymbol{s}^H$	$\boldsymbol{R}^H$
Jacobian	`Sacado::FAD`	$\boldsymbol{u}^H, \boldsymbol{s}^H$	$\frac{\partial \boldsymbol{R}^H}{\partial \boldsymbol{u}^H}$
Adjoint	`Sacado::FAD`	$\boldsymbol{u}^h_H, \boldsymbol{s}^h_H$	$\left[\frac{\partial \boldsymbol{R}^h}{\partial \boldsymbol{u}^h}\Big\|_{\boldsymbol{u}^h_H}\right]^T$
QoI	`double`	$\boldsymbol{u}^H, \boldsymbol{s}^H$	J^H
QoI Deriv	`Sacado::FAD`	$\boldsymbol{u}^h_H, \boldsymbol{s}^h_H$	$\left[\frac{\partial J^h}{\partial \boldsymbol{u}^h}\Big\|_{\boldsymbol{u}^h_H}\right]^T$
Error	`double`	$\boldsymbol{u}^h_H, \boldsymbol{s}^h_H, \boldsymbol{z}^h$	$\boldsymbol{R}^h$

Table 1: A list of TBGP evaluation operations used in the Goal application. In this table $\boldsymbol{u}^H$ is the primal solution vector, $\boldsymbol{u}^h_H$ is the prolongation of the solution vector to a richer space, $\boldsymbol{s}^H$ is a (potentially empty) vector of history-dependent mechanics state variables, $\boldsymbol{s}^h_H$ is the prolongation of the state to a richer space, $\boldsymbol{z}^h$ is the adjoint solution vector, $\boldsymbol{R}^H$ is the residual vector evaluated on the coarse space, $\boldsymbol{R}^h$ is the residual vector evaluated on the fine space, and J^H is the scalar QoI.

Listing 1: The abstract Goal integrator class interface.

```
class Integrator {
  public:
    Integrator();
    virtual ~Integrator();
    std::string const& get_name() { return name; }
    virtual void set_time(double, double) {}
    virtual void pre_process(SolInfo*) {}
    virtual void set_elem_set(int) {}
    virtual void gather(apf::MeshElement*) {}
    virtual void in_elem(apf::MeshElement*) {}
    virtual void at_point(apf::Vector3 const&, double, double) {}
    virtual void out_elem() {}
    virtual void scatter(SolInfo*) {}
    virtual void post_process(SolInfo*) {}
  protected:
    std::string name;
```

phase is written in an entirely generic fashion, and is templated on a scalar type `ScalarT`. Templating the code used for the compute phase, along with appropriately chosen gather and scatter operations, allows the same code to be re-used for the distinct evaluation purposes listed in Table 1.

The scatter phase takes the local element-level data evaluated in the compute phase and 'scatters' it to the appropriate global data structure, as determined by the current evaluation operation. For instance, for the residual evaluation operation, local element-level residuals are evaluated in the compute phase and then summed into appropriate locations in the global residual vector during the scatter phase. Similarly, for the Jacobian evaluation operations, local element-level Jacobians are evaluated in the compute phase and the scatter opeation sums these local contributions to appropriate locations in the global Jacobian matrix.

In the Goal application, we have considered six specific gather/scatter evaluation operations corresponding to the evaluation of the global residual vector, evaluation of the global Jacobian matrix, evaluation of the adjoint of the global Jacobian matrix, evaluation of the functional QoI, evaluation of the derivative of the functional QoI, and evaluation of localized adjoint-weighted residual error estimates. Table 1 lists the inputs and outputs for these specific evaluation operations.

To realize these specific gather/scatter operations, we have implemented an abstract degree of freedom class and an abstract quantity of interest class that are both templated on a scalar type `ScalarT`. This scalar type is explicitly instantiated to either be a `C++` double or a `Sacado::FAD` forward automatic differentiation variable type. Both the degree of freedom and QoI classes are equipped with `gather` and `scatter` methods, whose behavior changes based on an input parameter given to the class constructor. For the degree of freedom class, this parameter selects gather/scatter operations for either the residual, Jacobian, adjoint Jacobian, or adjoint-weighted residual error evaluations. Similarly, for the QoI class, this input parameter selects gather/scatter operations for either the evaluation of the QoI or the derivative of the QoI with respect to the problem degrees of freedom.

Previously, TBGP has been utilized in the multiphysics code Albany [51, 57] with the capability to perform the Residual, Jacobian, Adjoint, QoI, and QoI derivative evaluation operations shown in Table 1. To extend the abilities of TBGP to include adjoint-based error estimation, the Goal application implements the ability to perform the Adjoint and QoI derivative evaluations in a richer finite element space, as discussed in Section 6, a feature not previously available in existing TBGP codes. Further, the Goal application implements a novel evaluation type for the localization of the error, referred to as the Error evaluation type in Table 1. For this purpose, we have implemented an abstract weighting function class whose behavior changes based on the chosen evaluation type. This class evaluates the appropriate finite element weighting function values and gradients based on linear Lagrange basis functions for the Residual, Jacobian and Adjoint evaluation types. However, for the Error evaluation type, the behavior of the weighting function class is modified such that it evaluates the value and gradient of the adjoint solution z^h multiplied by a partition of unity. This abstraction of the weighting function class allows us to re-use the PDE implementation of the semilinear form $\mathcal{R}$ to assemble a residual vector $\boldsymbol{R}^h$ that represents an adjoint-weighted residual error estimate at each mesh vertex for each PDE equation in the richer finite element space, which is then used to drive mesh adaptation.

Listing 1 demonstrates the abstract integrator interface that has been implemented in the Goal application. The abstract degree of freedom, QoI, and weighting function classes inherit from this base class. For each of these classes, the `gather` and `scatter` methods are implemented specifically for each appropriate evaluation type. The PDE equations in the Goal application are written as a combination of `Goal::Integrator`s. Given an ordered array of integrators, the Goal application performs finite element assembly for every evaluation type in a generic manner, as outlined by Algorithm 1.

Algorithm 1 Assembly algorithm used in the Goal application

```
Given a mesh $M$ and an ordered array of integrators $I$:
Call pre_process for each integrator $i$ in $I$.
for each element set $es$ in mesh $M$ do
    Call set_elem_set for each integrator $i$ in $I$.
    for each element $e$ in element set $es$ do
        Call gather for each integrator $i$ in $I$.
        Call in_elem for each integrator $i$ in $I$.
        for each integration point $ip$ in element $e$ do
            Call at_point for each integrator $i$ in $I$.
        end for
        Call out_elem for each integrator $i$ in $I$.
        Call scatter for each integrator $i$ in $I$.
    end for
end for
Call post_process for each integrator $i$ in $I$.
```

5. The Primal Problem

5.1. Galerkin Finite Element Methods

We recall the definition of the abstract Galerkin finite element model problem, given by equation (7). In this context, the weighted residual form $\mathcal{R}_g$ is implemented in the Goal application. As an example, Listing 2 demonstrates the implementation of the Poisson residual $\mathcal{R}_g(u; w) := (\nabla u, \nabla w) - (w, f)$ in the Goal application.

Listing 2: Poisson residual.

```
template <typename ScalarT>
void Residual<ScalarT>::at_point(
    apf::Vector3 const& p, double ipw, double dv) {
  apf::Vector3 x(0,0,0);
  apf::mapLocalToGlobal(elem, p, x);
  double fval = eval(f, x[0], x[1], x[2], 0.0);
  for (int n = 0; n < u->get_num_nodes(); ++n)
  for (int i = 0; i < num_dims; ++i)
    u->resid(n) += u->grad(i) * w->grad(n, i) * ipw * dv;
  for (int n = 0; n < u->get_num_nodes(); ++n)
    u->resid(n) -= fval * w->val(n) * ipw * dv;
}
```

Listing 3: Pressure stabilization residual for mechanics.

```
template <typename ScalarT>
void Stabilization<ScalarT>::at_point(
    apf::Vector3 const&, double ipw, double dv) {
  double h = get_size(mesh, elem);
  double tau = 0.5*c0*h*h/mu;
  auto J = k->get_det_def_grad();
  auto F = k->get_def_grad();
  auto Cinv = inverse(transpose(F)*F);
  for (int n = 0; n < p->get_num_nodes(); ++n)
  for (int i = 0; i < num_dims; ++i)
  for (int j = 0; j < num_dims; ++j)
    p->resid(n) += tau * J * Cinv(i, j) *
      p->grad(i) * w->grad(n, j) * ipw * dv;
}
```

5.2. Stabilized Finite Element Methods

We recall the definition of the abstract stabilized finite element model problem, given by equation (9). In this context, both the weighted residual statement $\mathcal{R}_g$ and the stabilized weighted residual form $\mathcal{R}_\tau$ are implemented in the Goal application. As an example, Listing 3 demonstrates the implementation of the pressure stabilization [44] residual $\mathcal{R}_\tau(w;u)$ term used in the Goal application for finite deformation solid mechanics. This stabilization term is discussed in greater detail in Section 10.2.

5.3. Automated Solution Based on Residual Implementation

For each element, we compute element level Jacobian matrices by applying automatic differentiation [27] to element-level contributions to the residual vector. For example, Listing 2 demonstrates how contributions to the element-level Poisson's equation residual $\mathcal{R}(w;u) = (\nabla w, \nabla u) - (w, f)$ are implemented. The element level Jacobian matrices are then assembled into the global system Jacobian operator $\boldsymbol{\mathcal{J}}^H \in \mathbb{R}^{N\times N}$, given by

$$\boldsymbol{\mathcal{J}}^H = \frac{\partial \boldsymbol{R}^H(\boldsymbol{u}^H)}{\partial \boldsymbol{u}^H} \tag{11}$$

Listings 2 and 3 both demonstrate how element-level contributions to the semilinear forms $\mathcal{R}_g$ and $\mathcal{R}_\tau$, respectively, are computed in the Goal application. Notice that this code is templated on a scalar type `ScalarT`. When the scalar type is chosen as a `C++ double`, element-level contributions to the residual vector $\boldsymbol{R}^H$ are computed. When the scalar type is chosen as a Sacado forward automatic differentiation variable, element-level contributions to the Jacobian matrix $\boldsymbol{\mathcal{J}}^H$ are computed. This illustrates a key concept of template-based generic programming, in that the governing equations need only be implemented once to compute a variety of additional information.

With the ability to fully assemble the Jacobian matrix $\boldsymbol{\mathcal{J}}^H$ and the residual vector $\boldsymbol{R}^H$, we solve the governing equations with Newton's method, where we iterate over the steps

$$\begin{aligned} \boldsymbol{\mathcal{J}}^H(\boldsymbol{u}_k^H)\,\delta\boldsymbol{u}_k^H &= -\boldsymbol{R}^H(\boldsymbol{u}_k^H) \\ \boldsymbol{u}_{k+1}^H &= \boldsymbol{u}_k^H + \delta\boldsymbol{u}_k^H, \end{aligned} \tag{12}$$

unitl the convergence criterion $\|\boldsymbol{R}^H(\boldsymbol{u}^H)\|_2 < \epsilon$ is met for a user-specified tolerance ϵ. Here $\boldsymbol{u}_k^H$ denotes the solution vector at the k^{th} iteration obtained by solving the Newton linear system. For linear variational problems, we simply restrict ourselves to a single Newton linear solve, which reduces exactly to classical FEM assembly for linear problems.

6. The Adjoint Problem

6.1. A Richer Space via Uniform Refinement

The adjoint solution must be represented in a richer space than the one used for the primal problem to obtain meaningful error estimates. There are several strategies that are commonly used to obtain such a representation. First, the adjoint problem can be solved in the same finite element space as the primal problem and then be enriched to a higher order polynomial space [8] or a nested mesh [36] by some local patch-wise operation, or variational multiscale enrichment [25] can be used in the context of stabilized finite elements. Alternatively, the adjoint problem can be solved in a higher order polynomial space [16], which we will refer to as p-enrichment. As a final option, the adjoint problem can be solved on a uniformly refined mesh [11], which we will refer to as h-enrichment.

In this work, we choose the h-enrichment approach for several reasons. First, we would like the adjoint solution to be as accurate as possible for error estimation purposes, so we choose to solve the adjoint problem in a globally richer finite element space. Additionally, for stabilized finite element methods, the use of p-enrichment would in general necessitate the use of higher order stabilization terms that vanish for lower-order finite element methods with simplicial elements. These higher order terms are typically more difficult to implement than their lower order counterparts. Further, we remark that higher-order stabilized finite element methods are rarely used in practice, as stable higher-order mixed methods can usually be derived with fewer overall degrees of freedom [56]. Finally, we note that the unstructured mesh adaptation capabilities of the PUMI software make the h-enrichment approach readily available.

In the present context, we consider the term *uniform refinement* for triangles and tetrahedra to mean splitting each edge of the parent element at its midpoint. Or, in other words, creating new edges by connecting the midpoints of the parent element's existing edges. The uniform refinement of a triangle results in 4 nested triangles and the uniform refinement of a tetrahedron results in 8 nested tetrahedra.

We have denoted the trial and test spaces used for the primal problem as $\mathcal{S}^H$ and $\mathcal{V}^H$, respectively. We denote the trial and test spaces on the uniformly nested mesh as $\mathcal{S}^h$ and $\mathcal{V}^h$, respectively, where $h < H$ is representative of a finer mesh size. Figure 2 illustrates the discretization for the coarse and fine trial and test spaces defined for a three dimensional geometry with a complex void inclusion.

6.2. Discrete Adjoint Approximation

Let $\boldsymbol{R}^h : \mathbb{R}^n \to \mathbb{R}^n$ denote the residual form of the system of nonlinear algebraic equations arising either from the Galerkin (7) or stabilized (9) model problem posed on the uniformly nested mesh. Let $\boldsymbol{u}_H^h := I_H^h \boldsymbol{u}^H$ denote the prolongation of the primal finite element solution onto the richer space $\mathcal{S}^h$ via interpolation. Let $J^h : \mathbb{R}^n \to \mathbb{R}$ denote the discretization of the functional QoI on the uniformly nested fine space. We approximate the adjoint problem (4) in a discrete manner [17, 58, 59, 60], by solving

$$\left[\left. \frac{\partial \boldsymbol{R}^h}{\partial \boldsymbol{u}^h} \right|_{u_H^h} \right]^T \boldsymbol{z}^h = \left[\left. \frac{\partial J^h}{\partial \boldsymbol{u}^h} \right|_{u_H^h} \right]^T. \tag{13}$$

This allows us to automate the process of solving the adjoint problem, as discussed below. Here $\boldsymbol{z}^h \in \mathbb{R}^n$ denotes the adjoint solution vector on the nested discretization.

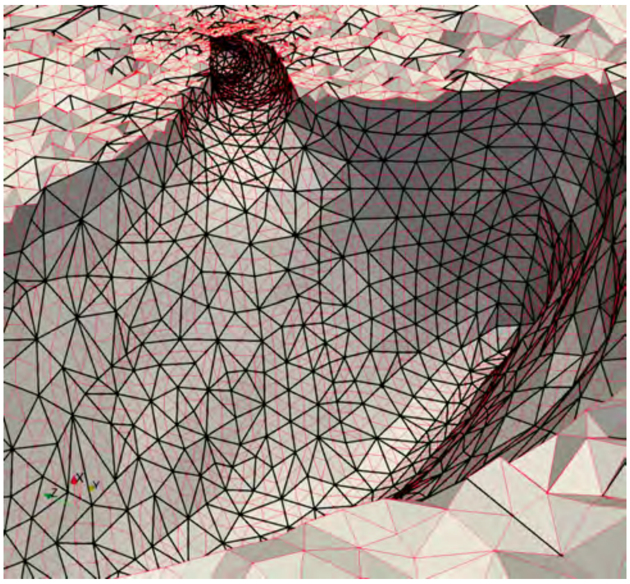


Figure 2: Example of a nested mesh (red edges) obtained via a uniform refinement of a base mesh (black edges) in three dimensions.

We remark that in the case of stabilized finite element methods, the matrix on the left hand side of equation (13) corresponds to the transpose of the discrete Jacobian matrix that arises from the stabilized finite element method. In this manner, stabilization is in some manner incorporated into the adjoint problem. This approach for approximating the adjoint problem for stabilized methods has been demonstrated to be effective for error estimation [12].

6.3. Automated Solution Based on Residual Formulation

The construction of the Jacobian transpose matrix $\left[\partial \boldsymbol{R}^h / \partial \boldsymbol{u}^h\right]^T$ is performed in the same automated manner as the Jacobian for the primal problem. That is, for each element, we compute consistent element tangent stiffness matrices via automatic differentiation of element-level contributions to the residual vector. However, during the *scatter* phase of the template-based generic programming process, we transpose the element-level tangent matrices and sum them into global Jacobian adjoint matrix. The computation of the Jacobian adjoint is done using the same templated code that is used to compute the primal residual vector and the Jacobian matrix, as illustrated by listings 2 and 3.

Similarly, the construction of the functional derivative vector $\left[\partial J^h / \partial \boldsymbol{u}^h\right]^T$ is done by evaluating derivatives of element-level contributions to the functional via automatic differentiation. This results in element-level derivative vectors that are then assembled into the global functional derivative vector. Listings 4 and 5 illustrate the implementation of two quantities of interest in the Goal application. Once the Jacobian transpose matrix and functional derivative vector have been assembled, we solve the adjoint problem (13) using a sparse iterative solver in parallel.

7. Error Estimation

7.1. Two-Level Error Estimates

Following Venditti and Darmofal [58, 59, 60], we review adjoint-based error estimation using two discretization levels. The discrete residual form of the governing equations for a Galerkin (7) finite element method or a stabilized finite element method (9) posed on the fine space can be expressed as

$$\boldsymbol{R}^h(\boldsymbol{u}^h) = \boldsymbol{0}. \tag{14}$$

Taking Taylor expansions of the discrete residual $\boldsymbol{R}^h$ evaluated on the fine space and the discrete functional J^h evaluated on the fine space about the point $\boldsymbol{u}_H^h$ yields

$$\boldsymbol{R}^h(\boldsymbol{u}^h) = \boldsymbol{R}^h(\boldsymbol{u}_H^h) + \left[\left. \frac{\partial \boldsymbol{R}^h}{\partial \boldsymbol{u}^h} \right|_{\boldsymbol{u}_H^h} \right] (\boldsymbol{u}^h - \boldsymbol{u}_H^h) + \dots \tag{15}$$

and

$$J^h(\boldsymbol{u}^h) = J^h(\boldsymbol{u}_H^h) + \left[\left. \frac{\partial J^h}{\partial \boldsymbol{u}^h} \right|_{\boldsymbol{u}_H^h} \right] (\boldsymbol{u}^h - \boldsymbol{u}_H^h) + \dots \tag{16}$$

respectively.

Using equation (14), the discretization error between the two spaces can be approximated to first order as

$$(\boldsymbol{u}^h - \boldsymbol{u}_H^h) \approx - \left[\left. \frac{\partial \boldsymbol{R}^h}{\partial \boldsymbol{u}^h} \right|_{\boldsymbol{u}_H^h} \right]^{-1} \boldsymbol{R}^h(\boldsymbol{u}_H^h), \tag{17}$$

which can then be substituted into the functional Taylor expansion (16) to obtain the so-called *adjoint weighted residual*

$$J^h(\boldsymbol{u}^h) - J^h(\boldsymbol{u}_H^h) \approx -\boldsymbol{z}^h \cdot \boldsymbol{R}^h(\boldsymbol{u}_H^h) \tag{18}$$

where $\boldsymbol{z}^h$ is the solution to the adjoint problem (13).

7.2. Modified Functional Error Estimate

Assume that the QoI converges at the rate k, such that $J - J^h(\boldsymbol{u}_H^h) = cH^k$ and $J - J^h(\boldsymbol{u}^h) = ch^k$, where J denotes the exact value of the QoI. If the fine space is obtained via uniform mesh refinement, then the ratio of the fine mesh size to the coarse mesh size is given as $\frac{h}{H} = \frac{1}{2}$. Consider the ratio

$$\begin{aligned} \frac{J^h(\boldsymbol{u}^h) - J^h(\boldsymbol{u}_H^h)}{J - J^h(\boldsymbol{u}_H^h)} &= \frac{\left[J - J^h(\boldsymbol{u}_H^h)\right] - \left[J - J^h(\boldsymbol{u}^h)\right]}{J - J^h(\boldsymbol{u}_H^h)} \\ &= \frac{cH^k - ch^k}{cH^k} \\ &= 1 - \left(\frac{h}{H}\right)^k \\ &= 1 - \left(\frac{1}{2}\right)^k \end{aligned} \tag{19}$$

in the limit as $H \to 0$ [17]. We call this ratio $\alpha := 1 - (1/2)^k$. Let η denote an approximation to the functional error $J - J^h(\boldsymbol{u}_H^h)$. Let $\mathcal{I}$ denote the effectivity index given by

$$\mathcal{I} = \frac{\eta}{J - J^h(\boldsymbol{u}_H^h)}. \tag{20}$$

We would like to obtain error estimates η that lead to effectivity indices of $\mathcal{I} = 1$ as $H \to 0$. To this end, we recall $J^h(\boldsymbol{u}^h) - J^h(\boldsymbol{u}_H^h) \approx -\boldsymbol{z}^h \cdot \boldsymbol{R}^h(\boldsymbol{u}_H^h)$ from equation (18) to obtain the scaled adjoint weighted residual error estimate

$$\eta = -\frac{1}{\alpha} \boldsymbol{z}^h \cdot \boldsymbol{R}^h(\boldsymbol{u}_H^h). \tag{21}$$

7.3. Error Localization for Galerkin Methods

Following the approach of Richter and Wick [49], we introduce a partition of unity ϕ_i, such that $\sum_i \phi_i = 1$, into the weighting function slot for the error estimate to localize the error. In this work, the partition of unity is realized as linear Lagrange basis functions. This yields local error contributions η_i at the n_{vtx} mesh vertices in the mesh. Let $z^h \in \mathcal{V}^h$ be the finite element solution obtained by solving the discrete adjoint problem (13). We assume that this solution well approximates the continuous adjoint problem (4), such that $z \approx z^h$. Let z^H denote the interpolant of z^h onto the coarse space $\mathcal{V}^H$. Recalling the error representation (8) for Galerkin finite elements, we obtain partition of unity-based correction indicators η_i in the following manner

$$J(u) - J(u^H) \approx \sum_{i=1}^{n_{vtx}} \underbrace{-\mathcal{R}_g(u^H\,;\,(z^h - z^H)\phi_i)}_{\eta_i}. \tag{22}$$

7.4. Error Localization for Stabilized Methods

Error localization for the stabilized finite element formulation (9) proceeds in the same manner as the previous section. Let $z^h \in \mathcal{V}^h$ denote the finite element solution obtained by solving the discrete adjoint problem (13) and let z^H denote the interpolant of z^h onto the coarse space $\mathcal{V}^H$. Introducing a partition of unity into the error representation (10) for stabilized finite element methods with the approximation $z \approx z^h$ yields the vertex-based correction indicators η_i:

$$J(u) - J(u^H) \approx \sum_{i=1}^{n_{vtx}} \underbrace{-\mathcal{R}_g(u^H\,;\,(z^h - z^H)\phi_i) + \mathcal{R}_\tau(u^H\,;\,z^H\phi_i)}_{\eta_i}. \tag{23}$$

Once correction indicators η_i have been evaluated, an approximate upper bound $\hat{\eta}$ for the error is computed by summing the absolute value of the error contributions over all mesh vertices

$$\hat{\eta} = \sum_{i=1}^{n_{vtx}} |\eta_i|. \tag{24}$$

7.5. Automated Error Localization Based on Residual Implementation

During the assembly of the adjoint problem (13), the evaluation of element-level contributions to the residual vector evaluated on the fine space $\boldsymbol{R}^h(\boldsymbol{u}^h_H)$ are necessarily computed by the machinery of forward automatic differentiation. Thus, during the `scatter` phase for the adjoint problem computation, we additionally sum element-level contributions to the fine residual to assemble the global vector $\boldsymbol{R}^h(\boldsymbol{u}^h_H)$. This, along with the solution $\boldsymbol{z}^h$ to the adjoint problem (13) provides enough information to compute the adjoint-weighted residual estimate (21) in an automated fashion.

Again, we let $z^h \in \mathcal{V}^h$ denote the finite element solution to the discrete adjoint problem (13) and let z^H denote the interpolant of z^h onto the coarse space $\mathcal{V}^H$. We refer again to Listings 2 and 3, which illustrate implementations of Galerkin and stabilized semilinear forms $\mathcal{R}_g$ and $\mathcal{R}_\tau$, respectively, in the Goal application. Specifically, we remark that these residual evaluations contain the evaluation of weighting functions and their derivatives, given with calls to the methods `w->val(node)` and `w->grad(node, dim)`, respectively. To localize the error in an automated fashion, we override the calls to these methods such that they return values of the adjoint solution multiplied by a partition of unity. For instance, at a given reference location $\boldsymbol{\xi}$ in a given element, the partition of unity-based weight for the Galerin residual $\mathcal{R}_g$ is computed as

$$\texttt{w->val(n)} = \left[(z^h - z^H) \cdot \phi_n\right]\Big|_{\boldsymbol{\xi}}, \tag{25}$$

and its corresponding gradient is computed as

$$\texttt{w->grad(n)} = \nabla\left[(z^h - z^H) \cdot \phi_n\right]\Big|_{\boldsymbol{\xi}}. \tag{26}$$

Similarly, for the stabilized residual $\mathcal{R}_\tau$, the partition of unity-based adjoint weight is computed as

$$\texttt{w->val(n)} = \left[z^H \cdot \phi_n \right]\Big|_{\boldsymbol{\xi}}, \tag{27}$$

and its corresponding gradient is computed as

$$\texttt{w->grad(n)} = \nabla \left[z^H \cdot \phi_n \right]\Big|_{\boldsymbol{\xi}}. \tag{28}$$

In this manner, we have introduced partition of unity-based adjoint weights that have the same data type as the weights used for the computation of the primal and adjoint problems.

Using the adjoint weights in the error localization evaluation results in element-level residual vectors that correspond to contributions to the localized correction indicators η_i. During the `scatter` phase of the error localization evaluation, we sum these element level contributions to the appropriate mesh vertices to compute the localized correction indicators η_i.

8. Mesh Adaptation

Given localized correction indicators η_i at mesh vertices, we compute element-level correction indicators η_e for $e = 1, 2, \dots, n_{el}$, where n_{el} is the number of elements in the coarse discretization, by interpolating the vertex-based indicators to element centers and taking the result's absolute value.

We then specify a *mesh size field* that defines the desired value of edge lengths over the mesh. From a high-level, we would like to specify this size field such that areas of the mesh that contribute strongly to the error in the QoI are refined, and areas of the mesh that are insensitive to the error are coarsened. Following Boussetta et al. [10], we specify a size field that attempts to equidistribute the error in an output adapted mesh with N target elements. Let p be the polynomial interpolant order for the chosen finite element method. In the present setting, $p = 1$. We first define the global quantity G as

$$G = \sum_{e=1}^{n_{el}} (\eta_e)^{\frac{2d}{2p+d}}. \tag{29}$$

This global quantity arises by considering *a priori* convergence rates for the input mesh and attempting to find an optimal mesh size for an output mesh with N elements [10]. With this global quantity, new element sizes H_e^{new} are computed by scaling the previous element size H_e

$$H_e^{\text{new}} = \left(\frac{G}{N} \right)^{\frac{1}{d}} (\eta_e)^{\frac{-2}{2p+d}} H_e. \tag{30}$$

Finally, to prevent excessive refinement or coarsening in a single adaptive step, we clamp the element size such that it is no smaller than one quarter and no greater than twice the previous element size. This clamping is performed to ensure that mesh adaptation is being driven by accurate error indicators.

$$\frac{1}{4} \leq \frac{H_e^{\text{new}}}{H_e} \leq 2. \tag{31}$$

9. Quantities of Interest

In this section, we review three quantities of interest that we have implemented in the Goal application. One benefit of the current automated approach is that additional quantities of interest can be rapidly prototyped and investigated with relative ease. Here, we refer to the domain discretized by the finite element mesh as Ω.

Listing 4: Evaluation of the integrated displacement over a sub-domain.

```
template <typename ScalarT>
void AvgDisp<ScalarT>::at_point(
    apf::Vector3 const&, double w, double dv) {
  for (int i = 0; i < num_dims; ++i)
    this->elem_value += u->val(i) * w * dv;
  this->elem_value /= num_dims;
}
```

9.1. Point-Wise Solution Component

First, we consider the evaluation of the solution u at a given spatial location $\boldsymbol{x}$. This functional can be expressed as

$$J(u) = \int_{\Omega} \delta(\boldsymbol{x} - \boldsymbol{x}_0)\, u \, \mathrm{d}\Omega, \tag{32}$$

where δ is the Dirac delta function. We implement this quantity of interest as a discrete delta function, such that the right-hand side for the adjoint problem takes the form

$$\frac{\partial J^h}{\partial \boldsymbol{u}^h} = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 0 \end{bmatrix}. \tag{33}$$

For this implementation, a mesh vertex is always placed at the spatial location $\boldsymbol{x}_0$, the QoI derivative vector is zeroed out, and we place a one in the row of the QoI derivative vector that corresponds to the solution at the vertex.

9.2. Integrated Solution Over a Sub-Domain

Next, we consider the integrated solution over a sub-domain $\Omega_0 \subset \Omega$, which can be expressed as

$$J(u) = \int_{\Omega_0} \frac{1}{n_c} \sum_{i=1}^{n_c} u \, \mathrm{d}\Omega. \tag{34}$$

Here, n_c denotes the number of components for the solution vector. As an example, Listing 4 demonstrates the Goal implementation for the QoI corresponding to the integrated displacement over a sub-domain.

9.3. Integrated von Mises Stress Over a Sub-Domain

Finally, specifically for mechanics problems, we consider the evaluation of the von Mises stress integrated over a sub-domain $\Omega_0 \subset \Omega$, given as

$$J(u) = \int_{\Omega_0} \sigma_{vm} \, \mathrm{d}\Omega, \tag{35}$$

where the von Mises stress σ_{vm} is defined as

$$\sigma_{vm} := \sqrt{\frac{3}{2} \boldsymbol{\sigma}'_{ij} \boldsymbol{\sigma}'_{ij}}. \tag{36}$$

Here summation over repeated indices is implied and $\boldsymbol{\sigma}' = \boldsymbol{\sigma} - \frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})\boldsymbol{I}$ denotes the deviatoric part of the Cauchy stress tensor $\boldsymbol{\sigma}$. The von Mises stress is often used in yield criterion for elastoplastic constitutive models, and is hence of particular interest for solid mechanics design applications.

We note that this function $J(u)$ has sources of nonlinearities from the deviatoric stress tensor $\boldsymbol{\sigma}'$ and further nonlinearites introduced by the definition of the von Mises stress, which includes the square of deviatoric stress components and a square root operation. The linearization and implementation of this QoI, as required for adjoint-based error estimation, would be cumbersome at best without some kind of automated approach. In contrast, Listing 5 illustrates the simplicity of the relevant `C++` code that implements integration point contributions to this specific QoI in the Goal application.

Listing 5: Evaluation of the integrated von Mises stress over a sub-domain.

```
template <typename ScalarT>
void AvgVM<ScalarT>::at_point(
    apf::Vector3 const&, double w, double dv) {
  auto sigma = model->get_cauchy();
  ScalarT vm = compute_von_mises<ScalarT>(sigma);
  this->elem_value += vm * w * dv;
}
```

10. Results

10.1. Poisson's Equation

As a first example, we investigate error estimation and mesh adaptation in Poisson's equation for the model problem

$$\begin{cases} -\nabla^2 u = f & \boldsymbol{x} \in \Omega, \\ \quad u = 0 & \boldsymbol{x} \in \partial\Omega. \end{cases} \tag{37}$$

This model problem leads to the Galerkin finite element method: find $u^H \in \mathcal{V}^H$ such that $\mathcal{R}_g(u^H; w^H) = 0$ for all $w^H \in \mathcal{V}^H$. Here the residual $\mathcal{R}_g$ is defined as

$$\mathcal{R}_g(u^H; w^H) := (\nabla u^H, \nabla w^H) - (w^H, f), \tag{38}$$

and the space $\mathcal{V}^H$ is given by

$$\mathcal{V}^H := \{u^h \in H^1(\Omega) : u^H = 0 \text{ on } \partial\Omega\,,\ u^H|_{\Omega_e} \in \mathbb{P}^1\}. \tag{39}$$

Here Ω_e denotes an element in a decomposition of the domain Ω into n_{el} non-overlapping elements such that $\cup_{e=1}^{n_{el}} \Omega_e = \Omega$ and $\Omega_i \cap \Omega_j = \varnothing$ if $i \neq j$. Additionally, $\mathbb{P}^1$ denotes the space of piecewise linear polynomials.

The domain is chosen to be $\Omega := [-1, 1] \times [-1, 1] \setminus [-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$ as shown in Figure 3. The data is chosen to be $f = 1$ and we consider a point-wise QoI of the form $J(u) = \int_\Omega \delta(\boldsymbol{x} - \boldsymbol{x}_0) u \, \mathrm{d}\Omega$, where the point of interest $\boldsymbol{x}_0$ is chosen to be $\boldsymbol{x}_0 = (0.75, 0.75)$. This problem was initially studied in the reference [6], where the QoI was determined to have a reference value of $J(u) = 0.0334474 \pm 1e\text{-}7$. Presently, we demonstrate that our automated approach can reproduce the results for traditional adjoint-based error estimation found in [6].

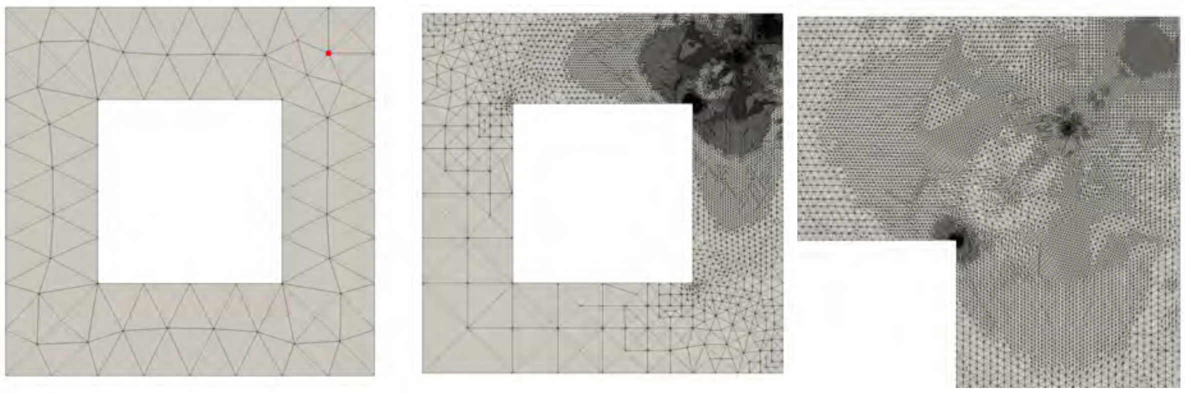


Figure 3: Domain and initial mesh (left) for the Poisson's equation example with the QoI point indicated in red, final adapted mesh (middle), and a close up of the upper-right hand corner of the final adapted mesh (right).

Starting from the initial mesh shown in Figure 3, the steps:

Solve Primal $\rightarrow$ Solve Adjoint $\rightarrow$ Estimate Error $\rightarrow$ Adapt Mesh were performed seven times. The adaptive simulation was run using 4 MPI ranks. The mesh size field was set according to equation (30) so that the target number of elements is twice that of the current mesh. Figure 3 also shows the final adapted mesh resulting from this procedure. We remark that the distribution of degrees of freedom in this mesh closely resembles the results obtained in reference [6].

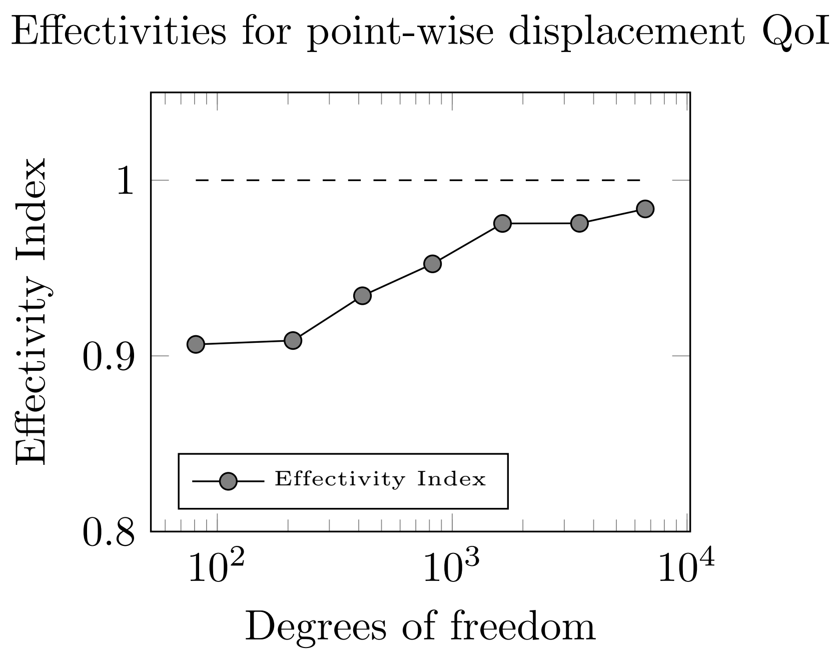


Figure 4: Effectivity indices for the adaptive Poisson's equation example.

We naively choose $k = 2$ for the functional convergence rate, although the presence of re-entrant corners may be reasonably expected to lessen this convergence rate. Despite this assumption, we demonstrate reasonable and useful results in terms of error estimation and mesh adaptivity later in this section. The choice of k yields a scaling factor $\alpha = \frac{3}{4}$ to be used in the estimate (21). We consider the "exact error" $\mathcal{E} = J(u) - J(u^H)$ and the effectivity index $\mathcal{I} = \frac{\eta}{\mathcal{E}}$, where η is the estimate given by equation (21). Here we have placed quotations around the term exact error because we have only approximated the exact value of the QoI $J(u)$, and not truly recovered its exact value. Figure 4 plots the effectivity index $\mathcal{I}$ versus the number of degrees of freedom in the adaptive process. This plot demonstrates the ability of the error estimate to recover the "exact error" as $H \to 0$.

Figure 5 displays the evolution of various errors during the adaptive process. The "exact error" $\mathcal{E}$ and the estimated error η are very close, as previously demonstrated by the effectivity index $\mathcal{I}$. Additionally, the approximated upper bound on the error $\hat{\eta}$ overestimates the error, but not to a large degree. This provides some indication that the correction indicators are effective in that they do not drastically overestimate error. Finally, we remark that an improved *corrected* QoI functional value can be computed as $J^*(u^H) = J(u^H) + \eta$. Figure 5 demonstrates that this corrected value is nearly an order of magnitude more accurate than the computed functional value $J(u^H)$ during the adaptive process.

Finally, Figure 6 demonstrates the evolution of the "exact error" for two adaptive strategies. The first strategy uniformly refines the mesh at each adaptive step and the second strategy performs the adjoint-based adaptive scheme developed in this work. We note that the error for the adjoint-based adaptive scheme converges faster than the uniform refinement scheme. Further, this convergence plot is consistent with the reference [6].

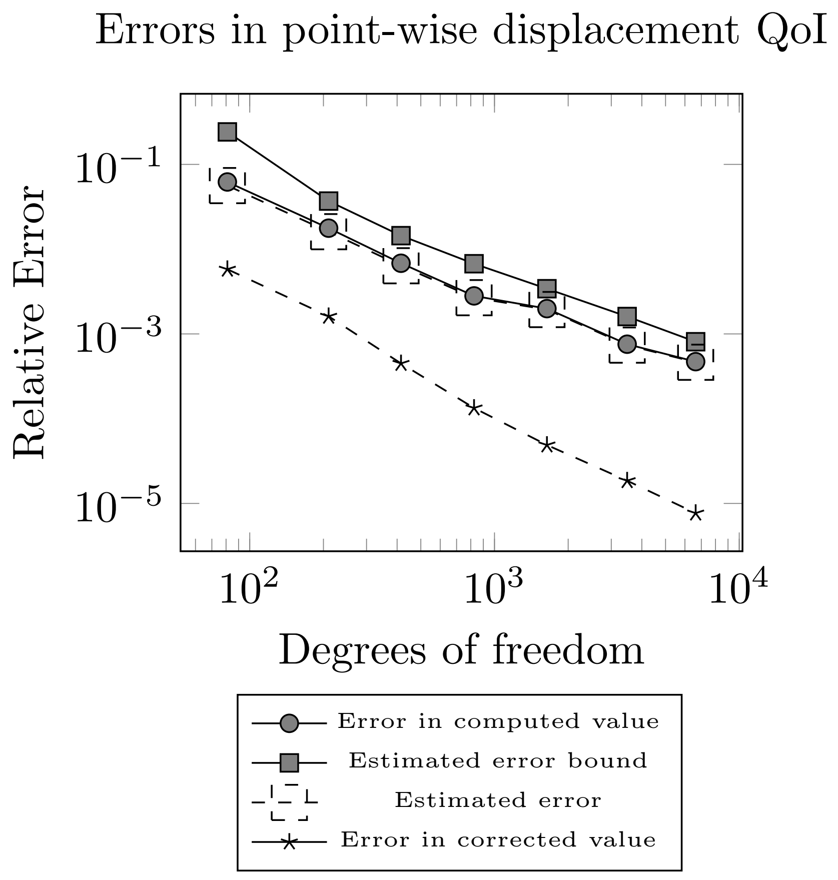


Figure 5: Errors for the point-wise QoI for the adaptive Poisson's equation example.

10.2. A Cell Embedded in a Matrix

Recently, the automated approach developed in this paper was applied to a stabilized mixed pressure-displacement finite element formulation [44] for the governing equations of finite deformation elasticity in a total Lagrangian setting [24]. For two and three dimensional problems in nonlinear elasticity, the automated approach was shown to effectively estimate the error and provide improved error convergence rates via adjoint-based mesh adaptation over uniform refinement.

In this section, the parallelization of a biomechanical application presented in the reference [24] is discussed. First, the governing equations are briefly reviewed. For mixed pressure-displacement formulations in the Goal application, the Galerkin residual is defined as:

$$\begin{aligned}&\mathcal{R}_g(\boldsymbol{U}^H;\boldsymbol{W}^H) := \\ &\int_\Omega \boldsymbol{P} : \nabla \boldsymbol{w}^H \,\mathrm{d}\Omega + \int_\Omega \left[\frac{p^H}{\kappa} - \frac{1}{2j}(j^2-1)\right] q^H \,\mathrm{d}\Omega - \int_{\partial\Omega_h} \boldsymbol{h}\cdot\boldsymbol{w}\,\mathrm{d}\Gamma,\end{aligned} \tag{40}$$

and the stabilization residual is defined as:

$$\mathcal{R}_\tau(\boldsymbol{U}^H;\boldsymbol{W}^H) := \sum_{e=1}^{n_{el}} \int_{\Omega_e} \tau_e(j\boldsymbol{F}^{-1}\boldsymbol{F}^{-T}) : (\nabla p^H \otimes \nabla q^H)\,\mathrm{d}\Omega. \tag{41}$$

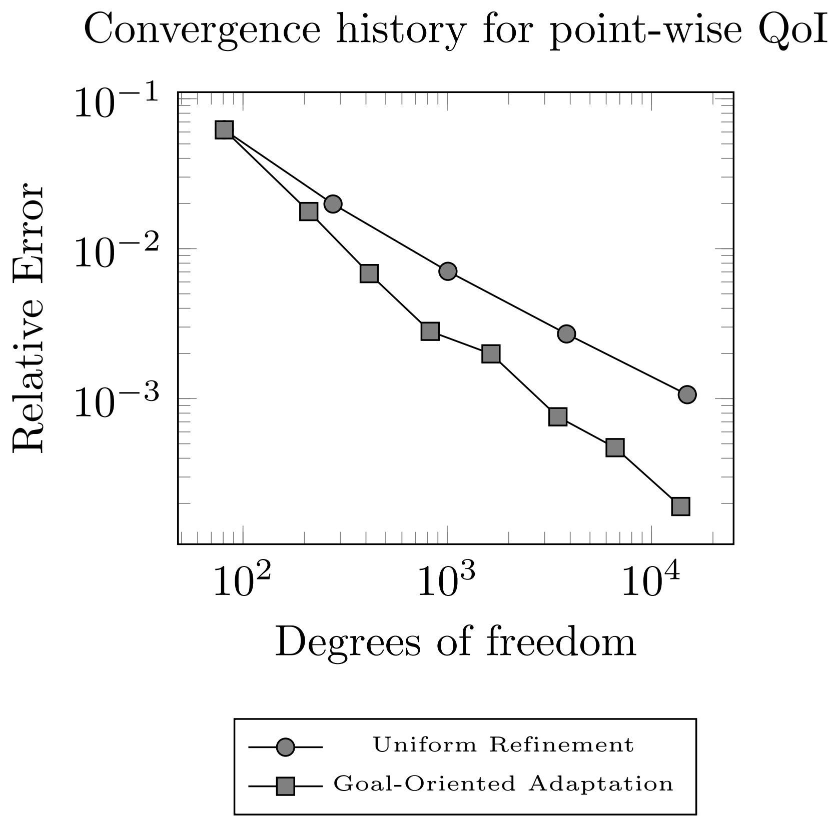


Figure 6: Error convergence using uniform mesh refinement and adjoint-based error estimation for the adaptive Poisson's equation example.

Here, $\boldsymbol{F}$ is the deformation gradient, $j := \det(\boldsymbol{F})$, $\boldsymbol{h}$ is an applied traction over the boundary $\partial\Omega_h$, $\boldsymbol{P} := j\boldsymbol{\sigma}\boldsymbol{F}^{-T}$ is the first Piola-Kirchhoff stress tensor, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, n_{el} is the total number of elements in the mesh, and $\tau_e := \frac{c_0 H_e^2}{2\mu}$ is a mesh-dependent stabilization parameter, where c_0 is a non-negative stability constant, H_e denotes an element mesh size and μ denotes the bulk modulus. The Cauchy stress tensor is defined via a neo-Hookean constitutive relationship. The total solution vector is defined as $\boldsymbol{U}^H := [\boldsymbol{u}^H, p^H]$, where $\boldsymbol{u}^H$ corresponds to displacements and p^H corresponds to pressures. Similarly, the total weighting vector is defined as $\boldsymbol{W}^H := [\boldsymbol{w}^H, q^H]$, where $\boldsymbol{w}^H$ denotes a weighting function corresponding to displacements and q^H is a weighting function corresponding to pressures.

We focus on a microglial cell with dimensions of about 20 µm ×20 µm ×20 µm embedded in an extracellular matrix of dimension 100 µm ×100 µm ×100 µm. The QoI is chosen to be a local integrated displacement $J(\boldsymbol{U}) = \int_{\Omega_0} \frac{1}{3}(u_x + u_y + u_z)\, d\Omega$, defined over a box Ω_0 with dimensions 30 µm ×30 µm ×30 µm that bounds the microglial cell. Figure 7 shows the geometry defining the microglial cell, the bounding box Ω_0, and the extracellular matrix. The shear modulus is defined as $\mu = 600$ Pa and Poisson's ratio is set to be $\nu = 0.4999$.

To drive the problem, traction boundary conditions are imposed along the surface of the microglial cell. The magnitude of the applied traction $\boldsymbol{h}$ is defined to be 10 times the distance to the center of the cell and its direction points inward towards the cell center. This traction is consistent with observed physical behavior [14]. Displacements $u_x = 0$, $u_y = 0$, and $u_z = 0$ are applied to the faces with constant minimum x-coordinate value, constant minimum y-coordinate value, and constant minimum z-coordinate value, respectively, to constrain rigid body rotations and translations.

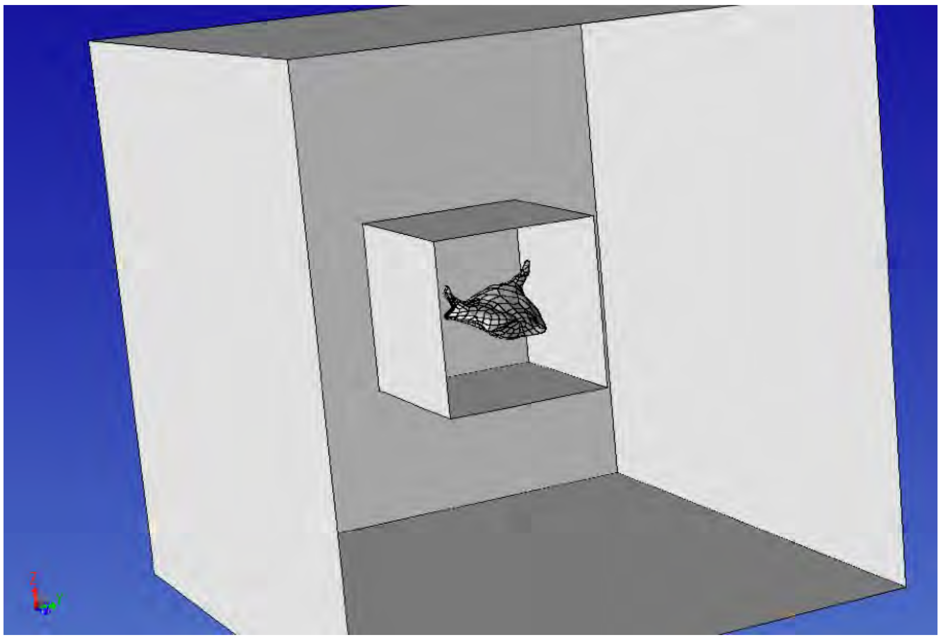


Figure 7: Domains for the microglial cell example.

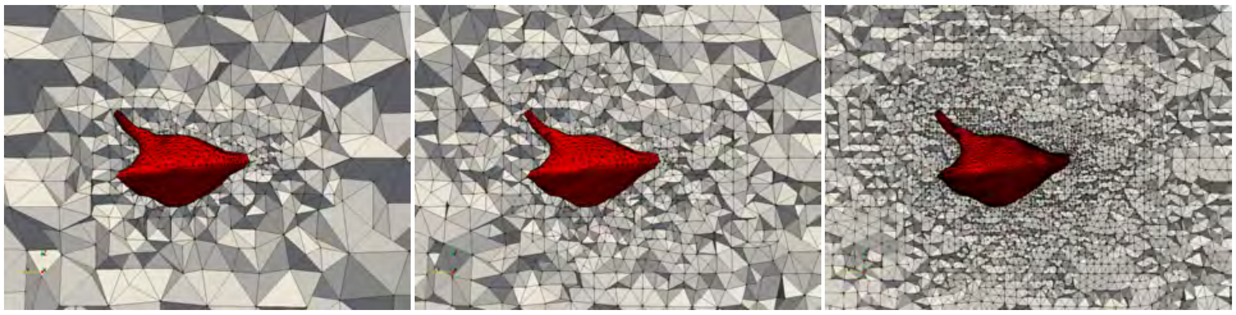


Figure 8: A close-up of the initial mesh (left) the mesh after 5 adaptive iterations (center) and the final adapted mesh (right) for the microglial cell example.

Figure 8 demonstrates an initial mesh, which contains around $30,000$ degrees of freedom. From this initial mesh, the steps

$$\text{Solve primal PDE} \rightarrow \text{Solve adjoint PDE} \rightarrow \text{Localize error} \rightarrow \text{Adapt mesh}$$

were successively performed 10 times. During the adapt stage, the mesh size field was set such that desired number of elements N in the output mesh is 1.5 times the number of elements in the previous mesh, according to equation (30). Figure 8 additionally demonstrates the adapted meshes obtained at the fifth and final adaptive iteration. In particular, both *coarsening* and *refinement* is performed during the adaptive iterations.

The problem was run using 16 MPI ranks. Figure 9 demonstrates the parallel partitioning for the initial mesh and for the final adapted mesh obtained after 10 adjoint-based adaptive iterations. To ensure partitioning quality, ParMA was utilized to guarantee the imbalance of vertices and elements across parallel partitions is no greater than 5%.

Figure 10 presents a breakdown of the total percentage of CPU time spent on each step in the adaptive analysis. For every adaptative iteration, the error localization (23) takes only a small percentage of the

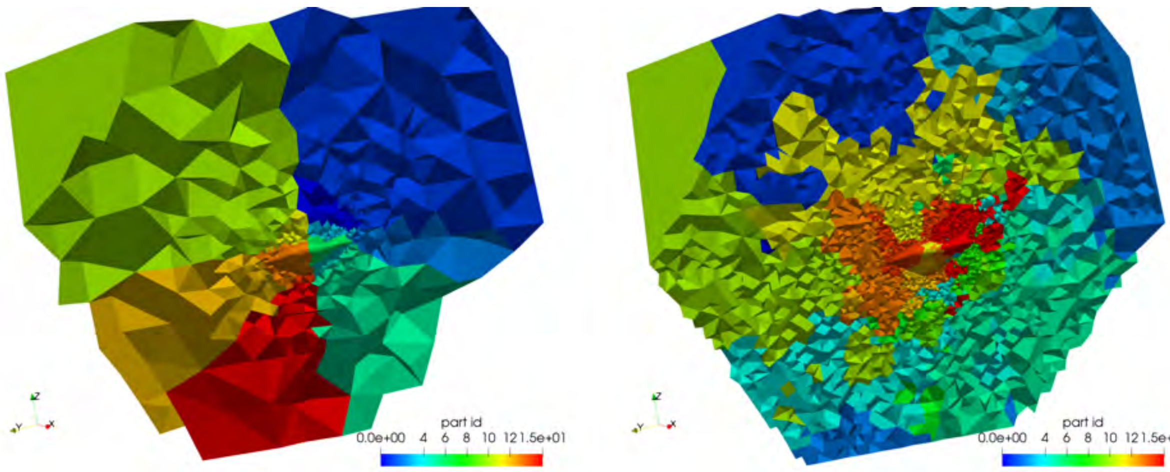


Figure 9: The parallel mesh partitioning for the initial mesh (left) and the final adapted mesh (right) for the microglial cell example.

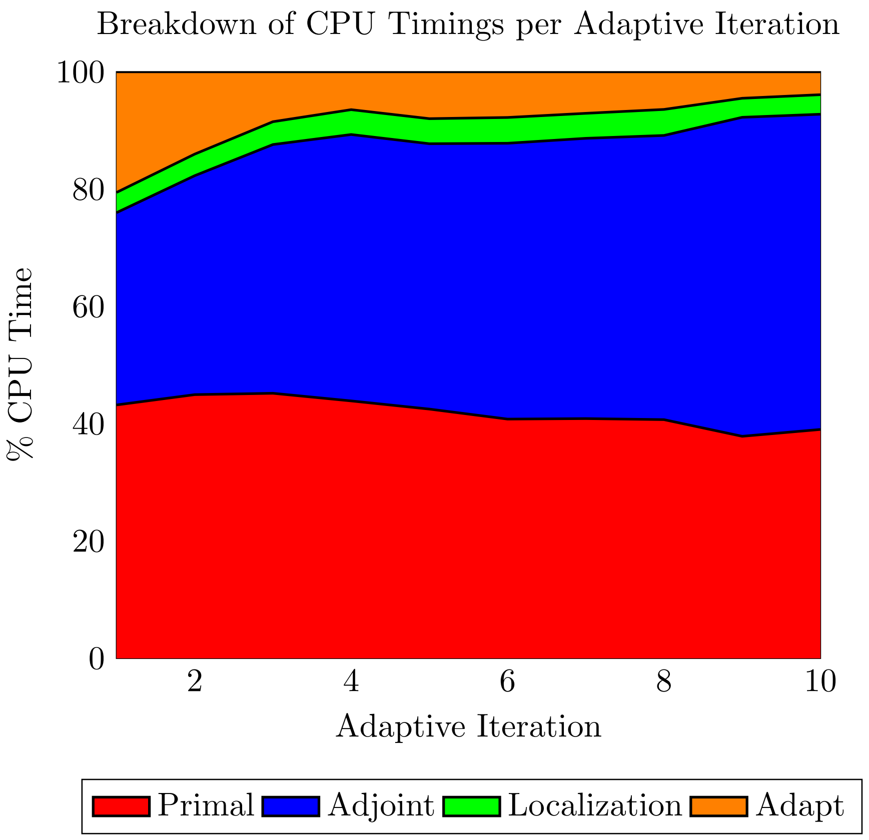


Figure 10: Breakdown of the CPU time spent for each portion of the adaptive process for the microglial cell example.

total CPU time, as it essentially amounts to an evaluation of the residual vector on the fine space. More interestingly, mesh adaptation initially accounts for about 20 percent of the total CPU time but decreases as the adaptive simulation progresses. This is explained by the fact that the initial adaptive iteration requires more work to optimally distribute the degrees of freedom for the functional QoI as compared to subsequent adaptive iterations. In addition to refinement and coarsening operations, the mesh adaptation step also performs *shape correction* to ensure elements are not too heavily skewed [33]. Finally, we note that the adjoint problem accounts for roughly 40 to 50 percent of the CPU time over the course of the adaptive simulation. While process of adjoint-based error estimation is not cheap for this example, we provide two justifying remarks. First, this problem required only 3 to 4 Newton iterations for each primal solve. For constitutive models with higher degrees of nonlinearity or for problems loaded to higher strains, it is not uncommon for Newton's method to converge in 7 to 10 iterations. In these scenarios, the relative cost of adjoint-based error estimation is not as extreme. Second, for this computational price, we have achieved very accurate error estimates as shown in the reference [24].

10.3. Elastoplasticity in an Array of Solder Joints

In this section, we investigate the utility of adjoint-based mesh adaptation for a thermomechanical analysis of an array of solder joints used in microelectronics fabrication. We consider a 6×6 array of solder joints sandwiched between two materials with distinct thermomechanical properties to model a portion of the process of 'flip-chip' manufacturing [9]. The full geometry is shown in Figure 11. We consider an elastoplastic constitutive model with a von Mises yield surface and linear isotropic hardening, as given by Simo and Hughes [52] with a temperature correction for the stress tensor [34]. The top slab, solder joints, and bottom slab are modeled with the distinct material properties given in reference [9].

To drive the problem, the entirety of the domain is cooled from a reference temperature $T_{ref} = 393$ K to a resting temperature of $T_f = 318$ K in a single load step. The faces with minimum x, y, and z coordinate values were constrained to have zero displacements in the x, y, and z directions, respectively. As a QoI, we consider the integrated von Mises stress given by equation (35) over three solder joints shown in yellow in Figure 11.

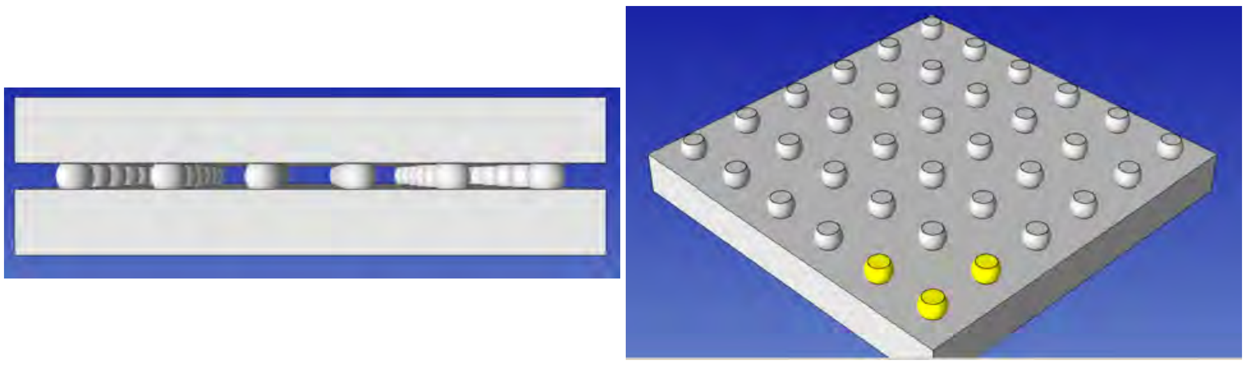


Figure 11: The solder joint array geometry (left) and the geometric specification of the integrated von Mises QoI (right).

The primal problem was solved on a sequence of uniformly refined meshes, starting with an initial mesh with about 1 million elements distributed over 16 MPI ranks, and finalizing with a mesh with over half a billion elements distributed over 8192 MPI ranks. For each solve, the work load for each mesh part (MPI rank) was held constant at approximately $70,000$ elements. Figure 12 demonstrates weak scaling timing results for various aspects of the primal solve. In particular, we remark that the assembly of the residual vector and Jacobian matrix scale well as the number of MPI ranks increases. The preconditioning routine shows a slight increase in time as the number of MPI ranks increases, but this increase is not drastic. The time to solve the linear system, however, does not scale optimally. Improvements to parallel performance could likely be made by more finely tuning the preconditioning and linear solver routines for the specific problem, but this is outside the scope of the present work.

The approximate QoI, $J^H(\boldsymbol{u}^H)$, was computed at each primal solve. Using the QoI evaluations from the finest three meshes, we performed Richardson extrapolation [48] to obtain a more accurate representation

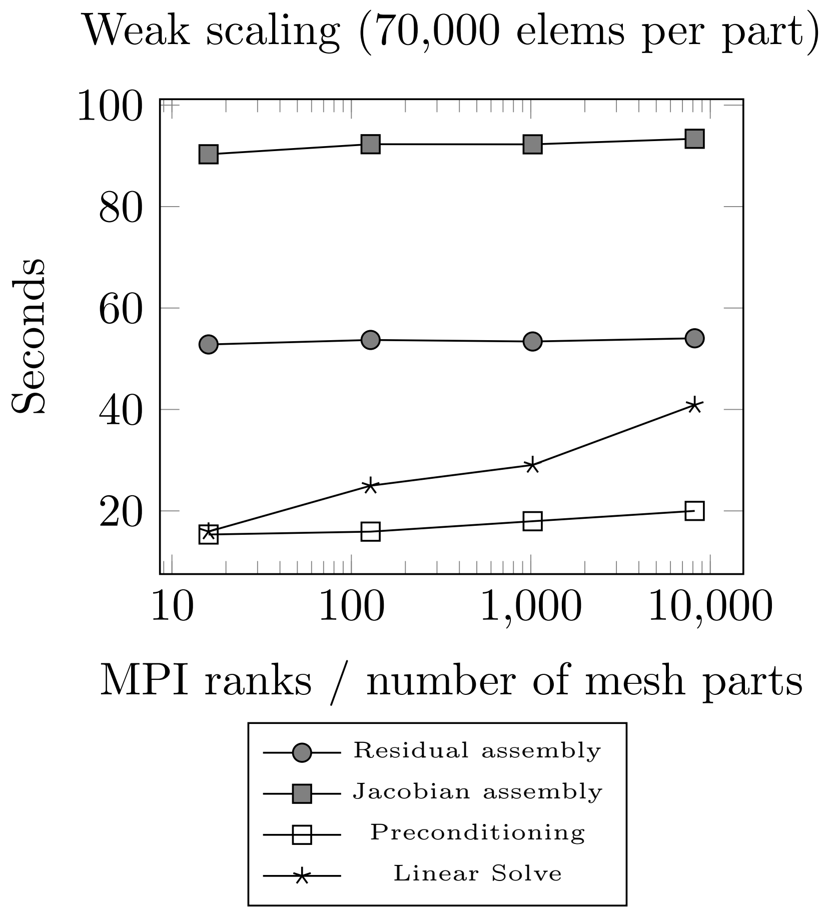


Figure 12: Weak scaling for the Goal application.

of the QoI. This value was given as $J(u) = 328.9$. We consider the extrapolated value to be the "true" QoI value and measure errors with respect to it. The expected convergence rate of the QoI is $k = 1$, which is confirmed by the Richardson extrapolation procedure.

From the same initial mesh used in the weak scaling study, we iteratively performed the steps:

Solve Primal → Solve Adjoint → Estimate Error → Adapt Mesh

with a restart after each mesh adaptation using 128, 256, and 512 MPI ranks, such that the output number of elements N in the adapted mesh was targeted to be 1 million, 2 million, and 4 million elements, respectively. Figure 13 shows different components of the adjoint solution obtained during the adjoint-based adaptive process. Figure 14 shows the spatial distribution of the error for the given QoI as computed by the adjoint-based error estimation process. Unsurprisingly, the majority of the error is localized to the area which geometrically defines the QoI. However, there are also contributions to the error from nearby solder joints that decrease as the distance from the 3 QoI solder joints increases. These additional contributions to the error are mostly gathered at the interface between solder joints and the underlying material slab, where von Mises stress concentrations exist.

Figures 15 and 16 demonstrate the intial mesh used for the solder joint problem and the final adapted mesh obtained via adjoint-based adaptation. These figures clearly demonstrate that the 3 solder joints that

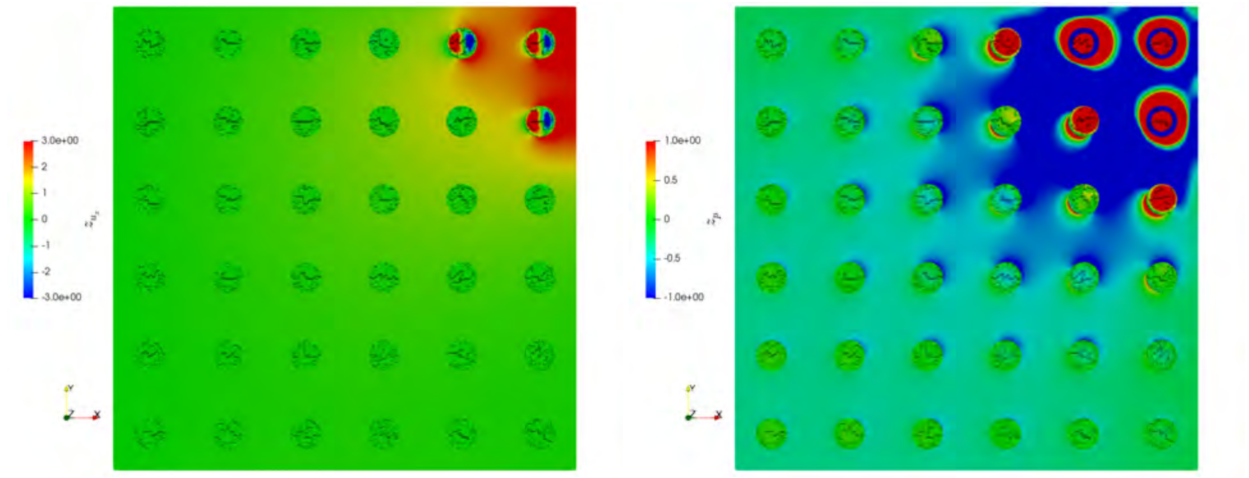


Figure 13: The x-component of the adjoint displacement solution (left), and the pressure component of the adjoint solution (right).

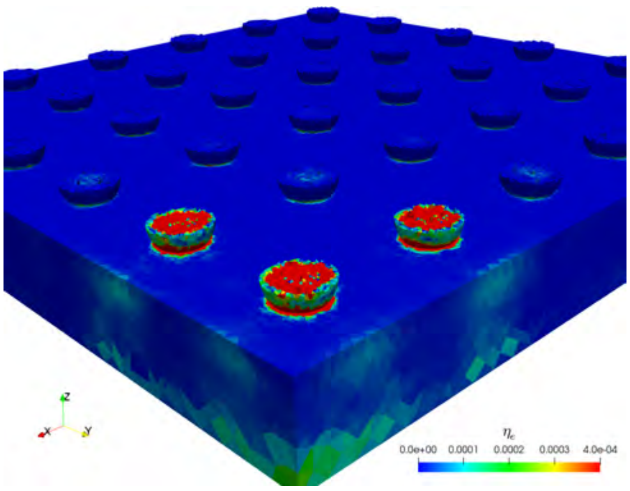


Figure 14: The spatial distribution of errors as computed by adjoint-based error estimation for the solder joint array.

define the QoI sub-domain are heavily refined, as expected. Additionally, notice that Figure 15 demonstrates that there is refinement at the left-most solder joint, which is not included in the geometric definition of the QoI. The adjoint-based error estimation procedure indicates that mesh must be refined in additional areas to accurately assess the QoI.

Figure 17 demonstrates the convergence history of the error in the functional QoI, as defined by the difference of the QoI obtained via Richardson extrapolation and the QoI approximated by the finite element solution. We compare the convergence for two adaptive schemes, one achieved by successive uniform refinements of the mesh and the other achieved by adjoint-based error estimation. After 4 adaptive iterations, the adjoint-based adaptive procedure achieves nearly the same degree of accuracy as the uniform refinement procedure with two orders of magnitude fewer degrees of freedom.

Finally, we remark that automated parallel adaptive workflows have been developed in reference [9]. As an avenue for future investigation, adjoint-based error estimation could be folded into these automated workflows. In particular, an automated primal analysis could be used to inform the actual selection of the QoI itself, which could then be accurately assessed using adjoint-based error estimation.

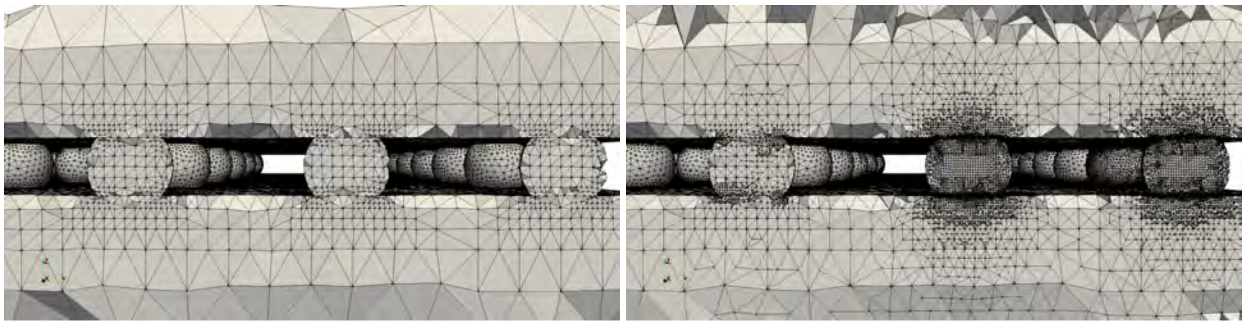


Figure 15: Cross-sectional view of the initial mesh for the solder joint geometry (left) and the final adapted mesh (right).

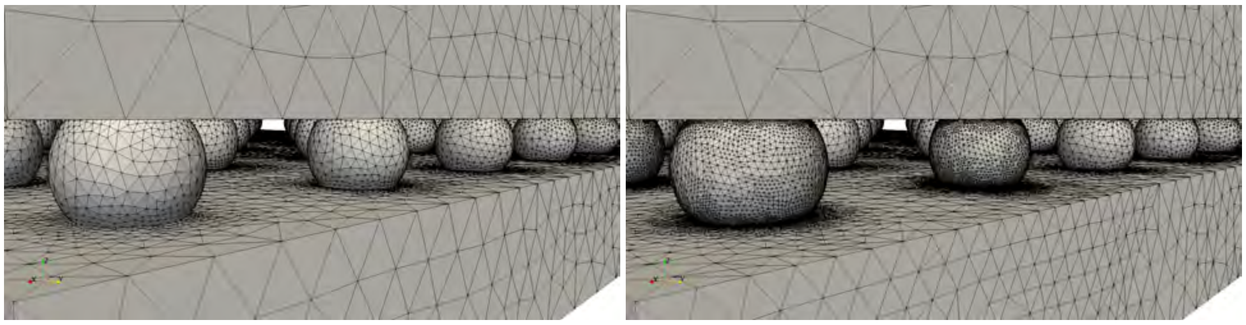


Figure 16: The initial mesh for the solder joint geometry (left) and the final adapted mesh (right).

11. Conclusions

In this work, we have developed an automated approach for adjoint-based error estimation and mesh adaptation for execution on parallel machines. We have developed this approach to be applicable to both Galerkin and stabilized finite element methods. To realize this approach, we have extended the concept of *template-based generic programming* for PDE models to include the automatic localization of error contributions using a partition of unity-based localization approach. We have demonstrated that this approach is effective for a variety of example applications, including nonlinear elasticity and elastoplasticity.

12. Acknowledgements

The authors acknowledge the support of IBM Corporation in the performance of this research. The computing resources of the Center for Computational Innovations at Rensselaer Polytechnic Institute are also acknowledged. The development of tools used in this work was partly supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under award DE-SC00066117 (FASTMath SciDAC Institute). The authors would like to thank Li Dong for providing the discrete geometric model used for the microglial cell example and Max Bloomfield for the geometric model used for the solder ball example.

This work was performed at the time at Rensselaer Polytechnic Institute before the authors moved to different organizations.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

This work is supported by the U.S. Army grants W911NF1410301 and W911NF16C0117.

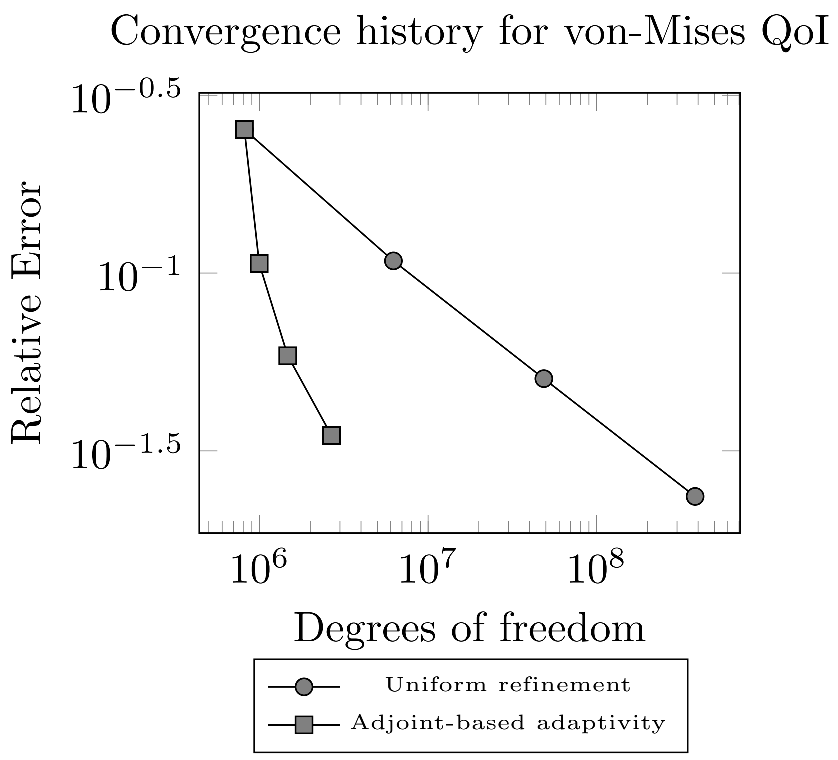


Figure 17: Error convergence histories for the solder joint example problem with the integrated von Mises stress QoI.

References

References

[1] Mark Ainsworth and J. Tinsley Oden. *A Posteriori Error Estimation in Finite Element Analysis.* John Wiley & Sons, Ltd, Hoboken, NJ, USA, 2011.

[2] Frédéric Alauzet, Xiangrong Li, E Seegyoung Seol, and Mark S Shephard. Parallel anisotropic 3D mesh adaptation by mesh modification. *Eng. with Comp.*, 21(3):247–258, Jan. 2006.

[3] Ivo Babuška and Anthony Miller. The post-processing approach in the finite element method, Part 1: Calculation of displacements, stresses and other higher derivatives of the displacements. *Int. J. for Numerical Methods. in Eng.*, 20(6):1085–1109, Jun. 1984.

[4] Ivo Babuška and Anthony Miller. The post-processing approach in the finite element method, Part 2: The calculation of stress intensity factors. *Int. J. for Numerical Methods. in Eng.*, 20(6):1111–1129, Jun. 1984.

[5] Ivo Babuška and Anthony Miller. The post-processing approach in the finite element method, Part 3: A posteriori error estimates and adaptive mesh selection. *Int. J. for Numerical Methods. in Eng.*, 20(12):2311–2324, Dec. 1984.

[6] Wolfgang Bangerth. Deal ii step 14, 2017.

[7] Eric Bavier, Mark Hoemmen, Sivasankaran Rajamanickam, and Heidi Thornquist. Amesos2 and Belos: Direct and iterative solvers for large sparse linear systems. *Scientific Programming*, 20(3):241–255, Jan. 2012.

[8] Roland Becker and Rolf Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica*, 10:1–102, May. 2001.

[9] Max O Bloomfield, Zhen Li, Brian Granzow, Daniel A Ibanez, Assad A Oberai, Glen A Hansen, Xiao Hu Liu, and Mark S Shephard. Component-based workflows for parallel thermomechanical analysis of arrayed geometries. *Eng. with Comput.*, 33(3):509–517, Jul. 2017.

[10] Ramzy Boussetta, Thierry Coupez, and Lionel Fourment. Adaptive remeshing based on a posteriori error estimation for forging simulation. *Comput. Methods in Appl. Mechanics and Eng.*, 195(48):6626–6645, Oct. 2006.

[11] Carsten Burstedde, Omar Ghattas, Georg Stadler, Tiankai Tu, and Lucas C Wilcox. Parallel scalable adjoint-based adaptive solution of variable-viscosity stokes flow problems. *Comput. Methods in Appl. Mechanics and Eng.*, 198(21):1691–1700, May. 2009.

[12] Eric C Cyr, John Shadid, and Tim Wildey. Approaches for adjoint-based a posteriori analysis of stabilized finite element methods. *SIAM J. on Scientific Comput.*, 36(2):A766–A791, Apr. 2014.

[13] Gerrett Diamond, Cameron W. Smith, and Mark S. Shephard. Dynamic load balancing of massively parallel unstructured meshes. In *Proc. of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, Denver, CO, USA*. Denver, CO, USA, Nov. 2017.

[14] Li Dong and Assad A Oberai. Recovery of cellular traction in three-dimensional nonlinear hyperelastic matrices. *Comput. Methods in Appl. Mechanics and Eng.*, 314:296–313, Feb. 2017.

[15] K Eriksson, D Estep, P Hansbo, and C Johnshon. *Computational Differential Equations*. Cambridge Univ. Press, New York, NY, USA, 2 edition, 1996.

[16] Krzysztof J Fidkowski. Output error estimation strategies for discontinuous galerkin discretizations of unsteady convection-dominated flows. *Int. J. for Numerical Methods in Eng.*, 88(12):1297–1322, May. 2011.

[17] Krzysztof J Fidkowski and David L Darmofal. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA J.*, 49(4):673–694, Apr. 2011.

[18] EC Gartland, Jr. Computable pointwise error bounds and the ritz method in one dimension. *SIAM journal on numerical analysis*, 21(1):84–100, 1984.

[19] S. S. Ghorashi and T. Rabczuk. Goal-oriented error estimation and mesh adaptivity in 3D elastoplasticity problems. *Int. J. of Fracture*, 203:3–19, Jan. 2017.

[20] S Sh Ghorashi, J Amani, AS Bagherzadeh, and T Rabczuk. Goal-oriented error estimation and mesh adaptivity in three-dimensional elasticity problems. In *WCCM XI-ECCM V-ECFD VI, Barcelona, Spain*, Barcelona, Spain, Jul. 2014.

[21] Michael B. Giles and Niles A. Pierce. *Chapter 2 - Adjoint Error Correction for Integral Outputs*, pages 47–95. Springer, Berlin, Germany, 2016.

[22] Octavio Andres González-Estrada, E Nadal, JJ Ródenas, Pierre Kerfriden, Stéphane Pierre-Alain Bordas, and FJ Fuenmayor. Mesh adaptivity driven by goal-oriented locally equilibrated superconvergent patch recovery. *Comput. Mechanics*, 53(5):957–976, May. 2014.

[23] Brian N. Granzow. Goal GitHub Repository, 2017.

[24] Brian N Granzow, Assad A Oberai, and Mark S Shephard. Adjoint-based error estimation and mesh adaptation for stabilized finite deformation elasticity. *Computer Methods in Applied Mechanics and Engineering*, 337:263–280, 2018.

[25] Brian N Granzow, Mark S Shephard, and Assad A Oberai. Output-based error estimation and mesh adaptation for variational multiscale methods. *Comput. Methods in Appl. Mechanics and Eng.*, 322:441–459, Aug. 2017.

[26] Thomas Grätsch and Klaus-Jürgen Bathe. A posteriori error estimation techniques in practical finite element analysis. *Comput. & Structures*, 83(4-5):235–265, Dec. 2005.

[27] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Soc. for Ind. & Appl. Math., Philadelphia, PA, USA, 2 edition, 2008.

[28] Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, and et al. An overview of the Trilinos project. *ACM Trans. on Math. Software*, 31(3):397–423, Sep. 2005.

[29] Michael A Heroux and James M Willenbring. A new overview of the Trilinos project. *Scientific Programming*, 20(2):83–88, Mar. 2012.

[30] Dan Ibanez and Mark S Shephard. Modifiable array data structures for mesh topology. *SIAM J. on Scientific Comput.*, 39(2):C144–C161, Apr. 2017.

[31] Daniel A Ibanez, E Seegyoung Seol, Cameron W Smith, and Mark S Shephard. PUMI: Parallel unstructured mesh infrastructure. *ACM Trans. on Math. Software*, 42(3):17–45, Jun. 2016.

[32] Fredrik Larsson, Peter Hansbo, and Kenneth Runesson. Strategies for computing goal-oriented a posteriori error measures in non-linear elasticity. *Int. J. for Numerical Methods in Eng.*, 55(8):879–894, Aug. 2002.

[33] Xiangrong Li, Mark S Shephard, and Mark W Beall. 3D anisotropic mesh adaptation by mesh modification. *Comput. Methods in Appl. Mechanics and Eng.*, 194(48):4915–4950, Nov. 2005.

[34] Zhen Li, Max O Bloomfield, and Assad A Oberai. Simulation of finite-strain inelastic phenomena governed by creep and plasticity. *Comput. Mechanics.* to be published.

[35] Anders Logg, Kent-Andre Mardal, and Garth Wells. *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*. Springer, Heidelberg, Germany, 2012.

[36] Marian Nemec and Michael J Aftosmis. Adjoint error estimation and adaptive refinement for embedded-boundary Cartesian meshes. In *18th AIAA Computational Fluid Dynamics Conf., Miami, FL, USA*, Miami, FL, USA, Jun. 2007.

[37] John Tinsley Oden and Serge Prudhomme. Goal-oriented error estimation and adaptivity for the finite element method. *Comput. & Math. with Applications*, 41(5-6):735–756, Mar. 2001.

[38] Roger P Pawlowski, Eric T Phipps, and Andrew G Salinger. Automating embedded analysis capabilities and managing software complexity in multiphysics simulation, Part I: Template-based generic programming. *Scientific Programming*, 20(2):197–219, Apr. 2012.

[39] Roger P Pawlowski, Eric T Phipps, Andrew G Salinger, Steven J Owen, Christopher M Siefert, and Matthew L Staten. Automating embedded analysis capabilities and managing software complexity in multiphysics simulation, Part II: Application to partial differential equations. *Scientific Programming*, 20(3):327–345, Jul. 2012.

[40] Eric Phipps and Roger Pawlowski. Efficient expression templates for operator overloading-based automatic differentiation. In *Recent Advances in Algorithmic Differentiation*, pages 309–319. Springer, Berlin, Germany, 2012.

[41] Andrey Prokopenko, Jonathan J. Hu, Tobias A. Wiesner, Christopher M. Siefert, and Raymond S. Tuminaro. MueLu user's guide 1.0. Technical Report SAND2014-18874, Sandia Nat. Lab., Albuquerque, NM, USA, Oct. 2014.

[42] Serge Prudhomme and J Tinsley Oden. On goal-oriented error estimation for elliptic problems: Application to the control of pointwise errors. *Comput. Methods in Appl. Mechanics and Eng.*, 176(1-4):313–331, Jul. 1999.

[43] E Rabizadeh, A Saboor Bagherzadeh, and T Rabczuk. Adaptive thermo-mechanical finite element formulation based on goal-oriented error estimation. *Comput. Materials Science*, 102:27–44, May. 2015.

[44] Binoj Ramesh and Antoinette M Maniatty. Stabilized finite element formulation for elastic–plastic finite deformations. *Comput. Methods in Appl. Mechanics and Eng.*, 194(6):775–800, Feb. 2005.

[45] Rolf Rannacher and F-T Suttmeier. A feed-back approach to error control in finite element methods: Application to linear elasticity. *Comput. Mechanics*, 19(5):434–446, Apr. 1997.

[46] Rolf Rannacher and F-T Suttmeier. A posteriori error control in finite element methods via duality techniques: Application to perfect plasticity. *Comput. Mechanics*, 21(2):123–133, Mar. 1998.

[47] Rolf Rannacher and Franz-Theo Suttmeier. A posteriori error estimation and mesh adaptation for finite element models in elasto-plasticity. *Comput. Methods in Appl. Mechanics and Eng.*, 176(1-4):333–361, Jul. 1999.

[48] Lewis Fry Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Trans. of the Royal Society of London*, 210:307–357, Jan. 1911.

[49] Thomas Richter and Thomas Wick. Variational localizations of the dual weighted residual estimator. *J. of Comput. and Appl. Math.*, 279:192–208, May. 2015.

[50] Marie E Rognes and Anders Logg. Automated goal-oriented error control I: Stationary variational problems. *SIAM J. on Scientific Comput.*, 35(3):C173–C193, May. 2013.

[51] Andrew G Salinger, Roscoe A Bartett, Quishi Chen, Xujiao Gao, Glen Hansen, Irina Kalashnikova, Alejandro Mota, Richard P Muller, Erik Nielsen, Jakob Ostien, et al. Albany: A component-based partial differential equation code built on trilinos. Technical Report SAND2013-8430J, Sandia Nat. Lab., Albuquerque, NM, USA, Nov. 2013.

[52] Juan C Simo and Thomas JR Hughes. *Computational Inelasticity.* Springer, New York, NY, USA, 2006.

[53] Cameron W Smith, Brian Granzow, Dan Ibanez, Onkar Sahni, Kenneth E Jansen, and Mark S Shephard. In-memory integration of existing software components for parallel adaptive unstructured mesh workflows. In *Proc. of the XSEDE16 Conf. on Diversity, Big Data, and Science at Scale, Miami, FL, USA*. Miami, FL, USA, Jul. 2016.

[54] Cameron W. Smith, Michel Rasquin, Dan Ibanez, Kenneth E. Jansen, and Mark S. Shephard. Improving unstructured mesh partitions for multiple criteria using mesh adjacencies. *SIAM J. Scientific Comput.* to be published.

[55] Erwin Stein, Marcus Rüter, and Stephan Ohnimus. Error-controlled adaptive goal-oriented modeling and finite element approximations in elasticity. *Comput. Methods in Appl. Mechanics and Eng.*, 196(37):3598–3613, Aug. 2007.

[56] Cedric Taylor and Paul Hood. A numerical solution of the Navier-Stokes equations using the finite element technique. *Comput. & Fluids*, 1(1):73–100, Jan. 1973.

[57] Irina K Tezaur, Mauro Perego, Andrew G Salinger, Raymond S Tuminaro, and Stephen F Price. Albany/FELIX: A parallel, scalable and robust, finite element, first-order Stokes approximation ice sheet solver built for advanced analysis. *Geoscientific Model Development*, 8(4):1197–1220, Apr. 2015.

[58] David A Venditti and David L Darmofal. Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow. *J. of Comput. Physics*, 164(1):204–227, Oct. 2000.

[59] David A. Venditti and David L. Darmofal. Grid adaptation for functional outputs: Application to two-dimensional inviscid flows. *J. of Comput. Physics*, 176(1):40–69, Feb. 2002.

[60] David A. Venditti and David L. Darmofal. Anisotropic grid adaptation for functional outputs: Application to two-dimensional viscous flows. *J. Comput. Phys.*, 187(1):22–46, May. 2003.

[61] Rüdiger Verfürth. A posteriori error estimation and adaptive mesh-refinement techniques. *J. of Comput. and Appl. Math.*, 50(1-3):67–83, May. 1994.

[62] JP Whiteley and SJ Tavener. Error estimation and adaptivity for incompressible hyperelasticity. *Int. J. for Numerical Methods. in Eng.*, 99(5):313–332, Apr. 2014.

Noname manuscript No.
(will be inserted by the editor)

A Parallel Interface Tracking Approach for Evolving Geometry Problems

Fan Yang · Anirban Chandra · Yu Zhang · Saurabh Tendulkar · Rocco Nastasia · Assad A. Oberai · Mark S. Shephard · Onkar Sahni

Received: date / Accepted: date

Abstract This paper presents a parallel interface tracking approach for evolving geometry problems where both the computational domain and mesh are updated as dictated by the analysis. An interface-fitted conforming hybrid/mixed mesh with anisotropic layered elements is used. A combination of mesh motion and mesh modification is employed to update the mesh to account for the interface motion. During mesh motion and modification the desired structure, shape and resolution of the anisotropic layered elements at interface are maintained. All steps are performed on partitioned meshes on distributed-memory parallel computers. The effectiveness of the current approach is demonstrated on problems with large motion or deformation in the geometry

1 Introduction

Problems with multimaterial, multiphase and fluid structure interactions arise in many engineering applications such as solid combustion, droplet evaporation, flow-driven projectile, blood flow through flexible arteries, flow-induced vibration, to name a few. Figure 1 shows two such problems. In such problems a common and important feature is that the interface evolves in time. Therefore, the geometry or spatial domain must be updated based on the interactions, for example, between an object and the surrounding fluid (i.e., between projectile and air) in the projectile case or between two phases (i.e., liquid and gas) in the droplet case.

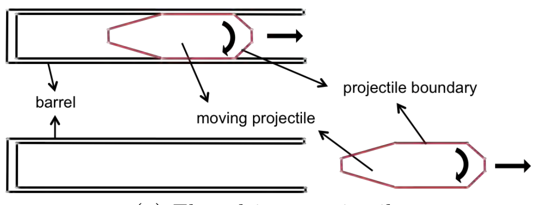
F. Yang, A. Chandra, Y. Zhang, M. Shephard and O. Sahni
SCOREC, RPI, 110 8th Street, Troy, NY 12180
Tel.: +1-518-276-8560
E-mail: sahni@rpi.edu

S. Tendulkar and R. Nastasia
Simmetrix Inc.

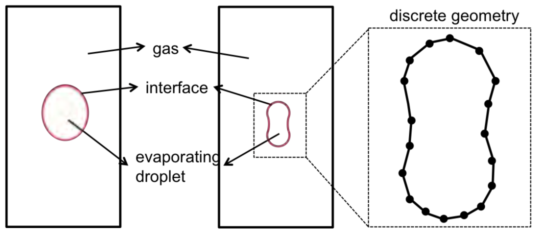
A. Oberai
University of Southern California

Two types of mesh-based approaches are commonly used for evolving geometry problems: explicit interface tracking, which is also referred to as front tracking, and implicit interface capturing. These approaches employ different frameworks to describe and simulate the transport equations of the material, namely the Lagrangian/material, the Eulerian/fixed or an arbitrary Lagrangian-Eulerian (ALE) [24, 26, 12] framework. In the case of an interface capturing approach, the interface is represented implicitly in the material/volumetric mesh that is typically fixed based on the Eulerian framework. Commonly used interface capturing methods include the level set [35, 45, 48], phase field [2, 5] and immersed boundary [36] methods. On the contrary, in an interface tracking method the evolving interface is explicitly represented in the underlying discretization at all times. Interface tracking method can be achieved by using any of the three frameworks involving a fixed or a moving mesh. Commonly used interface tracking methods include interface embedding methods [49, 19], interface reconstruction procedures based on volume-of-fluid and moment-of-fluid [44, 14, 6], interface enrichment approaches based on XFEM [8] and methods using interface-fitted mesh based on the Lagrangian or an ALE framework [17, 51, 37, 10, 11, 28].

A major advantage of implicit interface capturing is that topological changes in the interface are relatively easy to incorporate, even when they occur frequently (e.g., a large number of droplets interacting with each other causing topological changes). However, the drawback of implicit methods is the ad-hoc treatment of interface conditions as well as constitutive laws and equa-



(a) Flow-driven projectile



(b) Evaporating droplet

Fig. 1 Two examples of the evolving geometry problem with (a) fluid-structure interaction and (b) multiphase interaction (two instances are shown in each example)

tion of state in the so-called blending region around the interface. In contrast, interface tracking allows for a direct treatment of the interface physics including discontinuities and steep normal gradients. However, topological changes are more complex to manage in interface tracking.

We present an interface tracking approach that employs an ALE framework along with an interface-fitted conforming hybrid/mixed mesh with anisotropic layered elements. In such an approach the geometry of the computational domain must be updated as dictated by the analysis to properly represent the current state of the interface in terms of its location, shape and topology. We note that the current paper does not include general topological changes. The additional functions needed for topological changes are under development. In explicit interface tracking, the mesh must be updated to be consistent with the updated geometry at every instance. One way to update the mesh is to regenerate it entirely based on the updated geometry. Another option is to apply local mesh modifications. Regenerating or modifying a mesh and using this altered mesh in the analysis code at each time step is computationally expensive (i.e., both mesh regeneration or modification and re-initialization of data structures within the analysis code are time consuming). In contrast to mesh regeneration or modification, mesh motion can be used as the geometry evolves, based on the motion and deformation of the interface, while keeping the mesh connectivity the same (including the size of the data structures related to the mesh in the analysis code). Thus, mesh motion is efficient to use.

Several mesh motion methods have been proposed in the previous research, such as mesh elasticity analogy [34, 47, 52, 13], spring analogy [4, 7, 15, 53] and target-matrix paradigm [31].

Large motion and deformation in the geometry occur in many problems of interest. For example, a projectile moving from one end of the cannon to the other or droplets shirking significantly in volume due to phase change. For problems with large motion and deformation in the geometry, mesh motion alone is not sufficient while mesh regeneration or modification alone is computationally expensive. Thus, a combination of mesh motion and regeneration/modification is required for problems with large motion and deformation in the geometry [16, 25, 50, 37, 22, 33, 10, 21, 3, 28].

We have developed an approach that employs a combination of mesh motion and modification (instead of mesh regeneration) to be efficient. In our approach the mesh motion is based on mesh elasticity analogy and mesh modification is based on local cavity-based operations (e.g., see [32, 39, 43]) that offers specific advantages such as local solution transfer and parallelization. A mesh size field (e.g., see [32]), which describes the desired mesh resolution over the domain, is used to automatically trigger and drive mesh modification. The mesh size field is either prescribed, computed based on an error estimator, or a combination of the two.

This paper is organized as follows. Section 2 describes the overall simulation workflow and components. Section 3 discusses the details of the geometry and mesh updates. Results are presented in Section 4. Section 5 provides closing remarks.

2 Overall Simulation Workflow

The parallel interface tracking approach developed in this work is based on three primary simulation components or libraries. These components are shown in Figure 2 including: pre-processor, solver, and model and mesh adaptor. These three components are coupled/linked using a “driver” code. Further, all the components operate in parallel to target practical problems of interest.

The role of pre-processor is to convert input data (including mesh-related data) into a solver-suitable format and uses data streams to pass it to the solver. In addition to physics computation, the solver includes sub-components for mesh motion, error estimation and mesh modification trigger (to indicate that the mesh topology must be modified). These three additional sub-components are efficient to operate within the solver due to the direct access to the necessary data needed

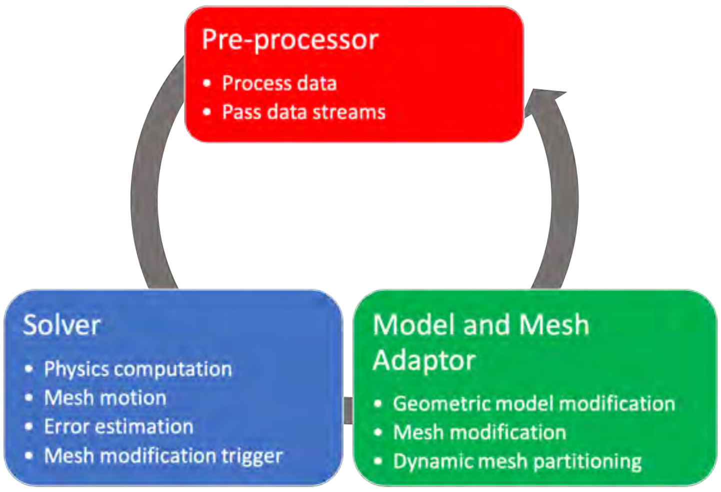


Fig. 2 Simulation components for interface tracking

for them, especially in a parallel distributed-memory environment. The third component of model and mesh adaptor updates the geometric model, applies mesh modification and performs dynamic mesh partitioning to maintain parallel load balance. The solver uses data streams to pass the necessary information to the adaptor including the current interface location and mesh coordinates to update the shape information in the geometry and mesh as well as the mesh size and solution fields to drive mesh modification and perform solution transfer. The driver code continues until the simulation completes spanning the desired interval of time.

Several of the sub-components have been developed in earlier works not involving interface tracking. For example, in-memory based coupling [46] is used between all the components (instead of a file-based coupling), which is crucial to be performant on parallel computers. The pre-processor and solver use data streams for in-memory coupling as mentioned above, while the adaptor uses well-defined functional interfaces (or application programming interfaces, APIs), see [46] for details. The solver employs an ALE finite element formulation [54] for the physics computation, while the error estimation is performed using an explicit variational multiscale (VMS)-based error estimator [23].

In this paper, we focus our attention on new developments made for parallel interface tracking. Details of the geometry and mesh updates are discussed in the next section, while the trigger for mesh modification along with the parallel programming model and mesh partition are discussed in this section.

2.1 Trigger for Mesh Modification

Geometry updates in the current problems of interest only involve shape changes and thus, can be performed at the necessary frequency (e.g., at every time step). However, mesh updates also include modification or connectivity changes that are time consuming to perform and update into the solver. Thus, it is desirable to apply mesh modification at a lower frequency (i.e., not at every time step) and a mechanism is needed that can trigger mesh modification only when necessary.

Two criteria are used to trigger mesh modification. If either of the criterion is met then mesh modification will be executed. Both criteria rely on the mesh size field. In this study, the mesh size field is determined at each time step, which is relatively inexpensive. It is either prescribed based on the current state of the geometry (i.e., it evolves in time with the geometry), computed based on an error estimator, or a combination of the two.

The first criterion is based on mesh resolution and a mesh modification is triggered when the current mesh resolution is not satisfactory with respect to the mesh size field at the given time step. This idea has been developed before and commonly used in an adaptive transient simulation (e.g., see [38]), and we follow the same in this work.

The second criterion targets mesh quality and requires element shape quality to be above a threshold. Several element shape quality measures are available in the literature. Note that it is necessary to use a signed quality measure such that any inverted/invalid element is detected with a negative value. In this work, we use a normalized mean ratio in the transformed/metric space defined by the given mesh size field (e.g., see [32]). Note that it can attain a maximum value of 1 for an element with the “best” shape (e.g., a regular tetrahedron). Note that any other suitable measure of the element shape quality can be used. Mesh modification is triggered when the shape quality measure for any element in the mesh is below the threshold value. The current procedure is summarized in Algorithm 1, where γ^M denotes the shape quality measure in the metric space while the subscripts e and 0 are used to denote an element value and the threshold value, respectively. For the current choice of element shape quality measure, 0.3 is used as the threshold value.

Algorithm 1 Quality-based mesh modification trigger

```
 1: triggerMeshModFlag = OFF
 2:
 3: for each element $e$ do
 4:     compute the element quality $\gamma_e^M$
 5:     if $\gamma_e^M < \gamma_0^M$ then
 6:         triggerMeshModFlag = ON
 7:         break
 8:     end if
 9: end for
10:
11: if triggerMeshModFlag == ON then
12:     return to driver and apply mesh modification
13: end if
```

2.2 Parallel Programming Model and Mesh Partition

The current parallel programming model builds upon earlier work not including interface tracking. It relies on each simulation component to use the same distributed-memory parallel mesh (e.g., see [27]) and thus, ensures that all simulation steps are efficiently performed in parallel. In this model, the mesh is partitioned into the desired number of parts and message passing interface (MPI) [20] is used such that each part is associated with a MPI rank or process. Hybrid parallel programming (including data parallelism) will be considered in the future.

The underlying parallel mesh database provides APIs to query the necessary information related to the inter-part boundaries, specifically each mesh entity on the inter-part boundaries (including mesh vertices, edges and faces in 3D) has multiple copies since it is duplicated on all residing parts. These copies are stored as remote copies in each residing part (e.g., see [27]). These APIs are directly used by the geometry and mesh adaptor to operate in parallel as discussed in the next section. Similarly, these APIs are used in the pre-processor to build parallel communication structures used by the solver.

In the pre-processor, owner relationship is established among remote copies of any given inter-part mesh entity such that one copy is assigned as the owner who is in-charge of communication between all the copies. In the solver, owner relationship is used to device a two-pass communication strategy to perform physics computation in parallel, see [41, 40] for details. Sub-components of mesh motion and error estimation follow the same form of parallel computation. This is the primary reason why these sub-components currently reside in the solver component. However, our workflow is flexible to support these sub-components to be defined outside of the solver such that other options can be incorporated.

Parallel interface tracking is supported by imposing two constrains on the parallel mesh partition. These constrains ensure that the overall parallelization strategy and communication structures used in each component remain the same with and without interface tracking. The first constrain is applied to support parallel discontinuous interpolation/solution for physically discontinuous fields at the interface such as density or normal velocity. The second constrain is applied to support parallel mesh updates of highly anisotropic layered elements around the interface. Each constrain is imposed by forming sets of related elements around the interface and enforcing the elements within any related set to be assigned to the same part during mesh partitioning. We note that, due to the locality of any constrained element set, the parallel load balance is not altered for practical problems of interest.

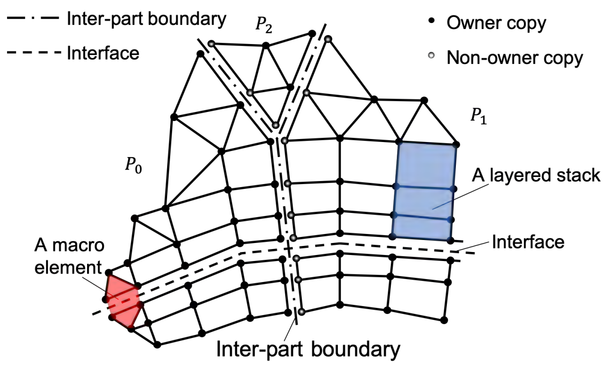


Fig. 3 A schematic showing a portion of the distributed mesh on three parts, P_0, P_1 and P_2, that consist of discontinuous interpolation and layered elements at the interface (note that an artificial gap is introduced at the interface and inter-part boundaries for clarity)

In the case of a discontinuous interpolation at the interface, the degrees-of-freedom on each side of the interface are different. Any mesh face on the interface is duplicated into two images and each image is used by the corresponding mesh region (i.e., on each side of the interface). Note that lower dimensional mesh entities (i.e., vertices and edges) can have more than two images for cases involving interactions between more than two materials at a physical location. For problems involving diffusion, the degrees-of-freedom of the two mesh regions sharing a mesh face on the interface are coupled in the current finite element formulation, see [54] for details. Thus, a macro element is formed based on the two mesh regions around the interface, see Figure 3. Each macro element containing two mesh regions is defined as an element set that is constrained to be on the same part during parallel mesh partitioning. This ensures that the parallel interactions between degrees-of-freedom remain the same to a parallel case with no interface and thus, the two-pass communication strategy discussed above is applicable during physics computation involving parallel interface tracking. Note that the geometric compatibility at the interface is ensured during mesh motion and modification, as discussed in the next section.

Element sets are also defined for the highly anisotropic layered elements around the interface to ensure their desired structure, shape and resolution are maintained during mesh motion and modification in parallel. In this case, an element set is defined based on a layered stack (see Figure 3), which contains all the elements through the layered height/thickness. Layered stacks on each side of the interface are considered separately. Note that the parallelization details of the geometry and mesh adaptor are discussed in the next section.

In summary, there are two types of element sets around the interface. One based on macro elements and the other based on layered stacks. An overlap can occur between different element sets. In such a situation, a superset containing all the relevant element sets is constructed and enforced to be on the same part during mesh partitioning. Note that the maximum possible overlap can be forced or limited to be local in nature such that the parallel load balance is not altered for practical problems of interest. For example, the maximum possible overlap can be forced to the most common situation which includes a macro element and two layered stacks from each side of the interface, which can be attained by a meshing process that ensures any mesh region around the interface only has one mesh face on the interface. It is also important to note that these two types of element sets provide flexibility to target a wide variety of problems. For example, some problems may not require macro elements (e.g., free surface problems) in which case element sets based on macro elements will not be present, or may not require layered elements (e.g., due to lack of anisotropy) in which case element sets based on layered stacks will not be present, while some problems may not involve macro elements on some part of the interface and may not require layered elements on some part of the interface.

3 Geometry and Mesh Updates

3.1 Geometry Updates

The computational domain is defined using a boundary representation-based geometric model. Such a domain definition includes a set of topological entities, each having shape information associated with them. Interface motion can result in shape and/or topological changes in the geometry. As mentioned above, we currently focus on problems where the geometry undergoes shape changes while the topology remains fixed. There are two classes of shape changes. One class where a rigid body motion is involved. The second class includes arbitrary deformations.

For problems involving a rigid body motion, a parametric representation is employed for the geometry (i.e., based on a CAD system), while in cases involving arbitrary deformations a parametric description is not suitable and therefore, a discrete representation is used (e.g., based on a triangulation).

In the case of a parametric geometry, the update is applied to the relevant objects or region entities in the geometry, e.g., when a projectile slides through the cannon. Parallelization is straightforward in this case by storing the entire geometry in each MPI rank or process and by using a synchronization/reduction step to update the shape of the geometry in each process. We note that a parallel partition may be required for a parametric geometric model with millions of geometric entities and will be consider in the future.

For problems involving arbitrary deformations, we currently use the evolving interface mesh to define and update the discrete geometric representation. Parallelization is achieved based on the partitioning of the interface mesh. It is also important to note that the relation of the mesh entities to the geometric model entities is currently maintained, specifically mesh edges relating to model edges and mesh faces relating to model faces. By doing this it is possible to employ methods that enrich the shape information for the discrete geometry beyond that defined by linear mesh facets. For example, methods like subdivision surfaces can be used to account for the local curvature [50]. Shape enrichment is particularly useful in case of a discrete geometry when applying mesh modification at the interface.

3.2 Mesh Updates

The mesh must be updated to remain consistent with the geometric model at every instant. In this study, the mesh is updated by mesh motion or, if indicated by the trigger, by mesh modification. Mesh modification is based on local cavity-based operations including non-manifold cases and layered elements [32, 50, 39, 42]. For parallel mesh modification based on local cavity-based operations see [1, 43]. For interface tracking, we employ the same parallel mesh modification procedures including at the interface. This is made possible with the two constrains applied on parallel mesh partition, as discussed in Section 2.2. In this paper, we focus our attention on mesh motion.

Mesh motion is divided into four steps: moving geometric entities, static geometric entities, layered elements, and tetrahedral elements. These steps are coupled with the mesh modification trigger to maintain mesh quality and validity, where the updated mesh is checked after applying all the four steps and the moved mesh is accepted and used only if it is deemed acceptable by the trigger. Otherwise the mesh state is reverted to the prior state (i.e., before applying mesh motion) and a mesh modification is instead triggered on the prior state of the mesh. This approach is found to be sufficient for the current problems of interest. Further, parallelization in each step is achieved by moving together all the copies of each mesh entity at the inter-part boundaries, which ensures geometric compatibility at the inter-part boundaries. Each mesh motion step is discussed below.

3.2.1 Moving Geometric Entities

In the present study, for each problem case a certain set of geometric entities are allowed to undergo a motion, specifically a set of geometric faces, edges and vertices are allowed to move. The motion for these moving geometric entities is computed as part of the physics computation, e.g., as part of a rigid body motion or of an arbitrarily deforming interface. Motion of each mesh entity residing on these geometric entities is set to be the same as that of the geometric entity. This motion is used as an input to drive the other three steps in mesh motion. It is important to note that for an interface supporting discontinuous interpolation, all images of each mesh entity on the interface move together ensuring geometric compatibility at the interface.

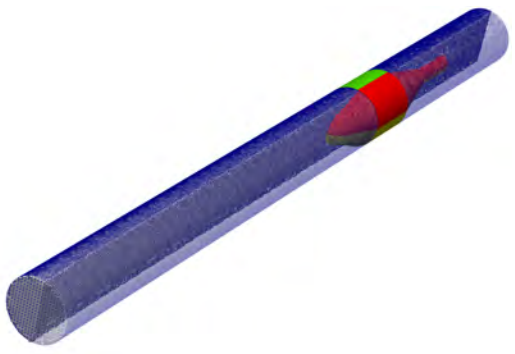
3.2.2 Static Geometric Entities

Mesh motion for each mesh entity residing on the static geometric entities, i.e., geometric faces, edges and vertices that are fixed, is computed such that the mesh entity is constrained to remain on the geometric entity. This is achieved in two steps. In the first step, we use the linear mesh elasticity analogy (including Jacobian-based stiffening [47]) to displace mesh entities without any constraints; given the input on mesh motion from the moving geometric entities discussed above. In the second step, updated locations from the first step (which can be off for the curved geometric entities) are projected back onto the geometric entities by employing a local search procedure provided by the geometry library. The projected locations are used to move the mesh entities residing on the static geometric entities. Essentially, during mesh motion the mesh on the static geometric entities is allowed to slide on the geometry. This feature provides a great deal of flexibility in mesh motion and enables the use of mesh motion alone (without any mesh modification) for a relatively longer duration. We note that in many problems of interest, absence of this feature may result in mesh modification at every time step. Figure 4(a) shows a translating and rotating projectile in a cannon. The projectile translates axially from the back/closed end to the open end by a distance approximately equal to its length. It rotates in the counter clockwise sense by 34.5° as shown by the 4 differently colored quadrants of the projectile. For this demonstration, a uniform tetrahedral mesh is used and the total motion is applied in about 400 steps; only mesh motion is used. The mesh on the projectile surface moves with it while the mesh on the cannon walls is allowed to slide on it and the tetrahedral volume mesh is free to move in all three directions. Overall a good quality mesh is maintained.

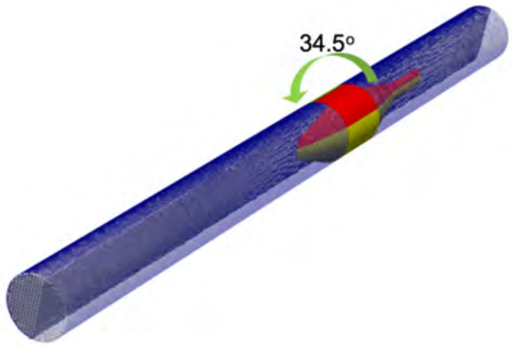
3.2.3 Layered Elements

Many problems of interest exhibit relatively strong gradients in certain directions as compared with other directions, such as a shear layer near the wall in a viscous flow or high normal gradient in temperature at a burning interface. In such cases, highly anisotropic layered elements are desired near the appropriate boundaries/interfaces.

Pre-defined meshes with layered elements near the walls have been utilized in previous studies ([18, 30, 29]). However, for evolving geometry problems, a procedure is needed that tightly controls the anisotropic layered elements in an adaptive fashion when the shape of the geometric entities is changing. We aim to maintain the desired structure, shape and resolution of the anisotropic layered elements during mesh motion. As noted earlier, mesh modification for layered elements is based on earlier works [39, 42, 43]. Mesh motion



(a) Initial mesh



(b) Mesh after translation and rotation

Fig. 4 Mesh motion for a translating and rotating projectile in a cannon (half of the mesh with a cut through the center plane is shown and the cannon walls are kept translucent)

based on the linear mesh elasticity approach is not robust in maintaining or explicitly controlling the structure, shape and resolution of the anisotropic layered elements. Therefore, we have developed an explicit repositioning method for the layered elements that employs connectivity of the growth curves. Note that mesh motion for the rest of the interior mesh containing tetrahedral elements is discussed below, while transition elements between layered and tetrahedral elements move accordingly from each side (and does not require any additional consideration).

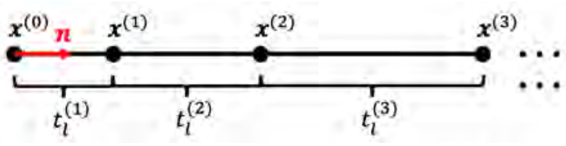


Fig. 5 Schematic of a growth curve

We first define a growth curve (see Figure 5). It is a collection or stack of mesh edges defined along the local normal direction to the boundary/interface. At an interface, growth curves on each side are treated differently. The mesh vertex on the boundary/interface from which a growth curve originates is called the base vertex. The direction of a growth curve is called the growth direction.

Currently, a growth curve is geometrically graded and defined by four parameters: the first layer thickness $t_l^{(1)}$, growth ratio r_l, total number of layers n_l and growth direction or local unit normal vector $\boldsymbol{n}$. The first three parameters ($t_l^{(1)}$, r_l, n_l) are either prescribed or computed adaptively [9]. The fourth parameter $\boldsymbol{n}$ is calculated based on the current geometry shape which is evolving with time. For a given mesh vertex on the boundary/interface, the local unit normal vector $\boldsymbol{n}$ is computed by a weighted average of the unit normal vectors of mesh faces on the boundary/interface around this mesh vertex. In case of a distributed/partitioned mesh, the two-pass communication strategy is used to computed the local unit normal vector along the growth direction.

First, the thickness of the ith layer in a growth curve is computed as:

$$t_l^{(i)} = r_l^{i-1} t_l^{(1)} = r_l t_l^{(i-1)} \qquad i = 2, 3, ..., n_l \tag{1}$$

Subsequently, the position of each vertex on a growth curve is given by:

$$\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + t_l^{(i)} \boldsymbol{n} = \boldsymbol{x}^{(i-1)} + r_l^{i-1} t_l^{(1)} \boldsymbol{n} \tag{2}$$

where $\boldsymbol{x}$ is the coordinate of a vertex. The superscript (i) indicates it is the ith vertex on a growth curve, where $i = 0$ implies the base vertex.

This procedure requires the connectivity of a growth curve, where all the mesh vertices on each growth curve are maintained in a list. In case of a distributed/partitioned mesh, a growth curve is allowed to be at the inter-part boundary. However, the entire growth curve (or layered stack) is constrained to be together on any residing part(s) as discussed in Section 2.2 (i.e., the list of vertices on a growth curve is complete on any given part). Figure 6 shows a partitioned growth curve for parallel mesh motion. This way the explicit repositioning for each growth curve is applied independently on each part (assuming that the position of the base vertex and the four growth-curve parameters (including local unit normal vector) are available on each part). The position of the top most vertex in case of a partitioned growth curve is also ensured to be consistent among all residing parts by using the two-pass communication strategy discussed in Section 2.2.

3.2.4 Tetrahedral Elements

After applying the above three mesh motion steps, all that is remaining is the mesh motion for the interior

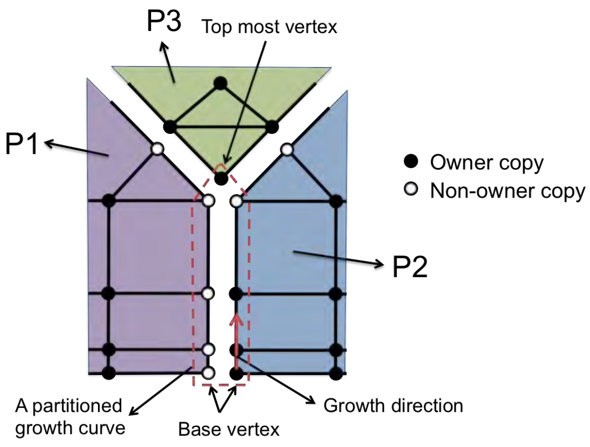
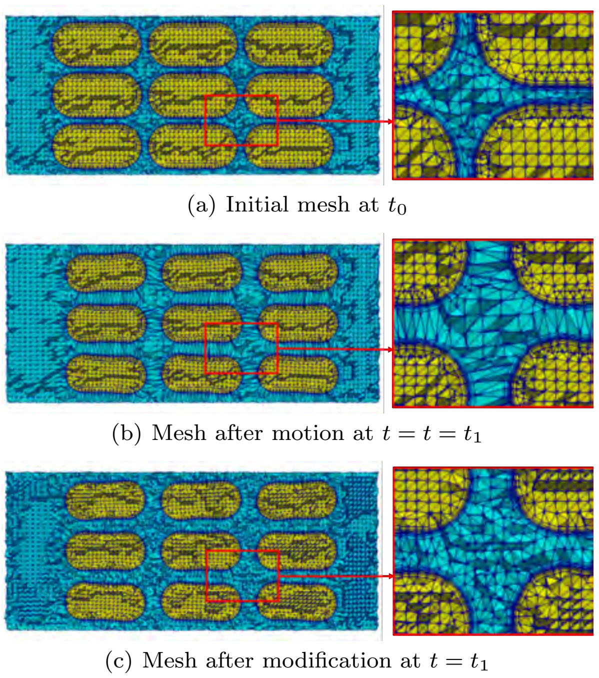


Fig. 6 A partitioned growth curve

mesh containing tetrahedral elements. It is performed using the linear mesh elasticity approach including Jacobian-based stiffening [47]. The input in this fourth mesh motion step is provided by the mesh motion or displacement computed in the above three steps.

3.2.5 Summary of Mesh Updates

We summarize this section with a demonstration of the entire mesh update process based on a combination of mesh motion and modification. A 9-grain case is used where the grains translate and shrink in a closed chamber. The interface motion is prescribed in this demonstration. In Figure 7, three meshes, including anisotropic layered elements at the interface, are shown at two instants ($t = t_0$ and t_1). Each mesh consists of about 220,000 and 1,000,000 number of vertices and regions, respectively. Mesh motion is applied on the initial mesh at $t = t_0$ to obtain a mesh at $t = t_1$, see Figure 7(b). In the mesh after motion, two aspects are important to note: 1) the structure, shape and resolution of the layered mesh is maintained, and 2) the elements become too distorted in between the grains (see the zoomed view on the right side in Figure 7(b)). We note that the mesh motion is exaggerated in this case to clearly show the distorted elements. Further, a mesh modification is applied at $t = t_1$ to obtain a mesh with the desired quality, see Figure 7(c).


(a) Initial mesh at t_0

(b) Mesh after motion at $t = t = t_1$

(c) Mesh after modification at $t = t_1$

Fig. 7 Mesh updates based on mesh motion and modification (including layered elements) for a 9-grain case

4 Results

Two problems are selected to demonstrate the utility of the current interface tracking approach. The first one involves 6 grains undergoing phase change with arbitrary deformations. The second one includes a rigid projectile moving/translating down the barrel.

4.1 Grain Case with Phase Change

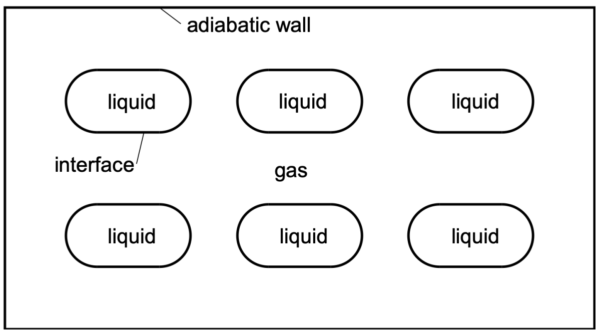


Fig. 8 Center plane of the 6-grain case at the beginning

In this section, we present a problem involving 6 grain-shaped droplets undergoing a phase change from denser liquid to lighter gas inside a camber, In this problem, predicting the exponential rise of pressure and temperature in the camber is typically of interest and thus, fully 3D transient simulations of phase change process is performed up to a certain duration, see [54] for more details. Figure 8 shows the center plane of the problem with 6 grain-shaped droplets in an adiabatic cylindrical camber with no-slip walls. At the beginning, each droplet is a 2 mm long circular cylinder capped with hemispherical end caps of 0.5 mm radius.

The phase change rate is governed by the Vieilles law:

$$u_p = a(p^+)^n \tag{3}$$

where p^+ is the pressure of the surrounding gas, while the exponent is set to be $n = 0.7$ and the pre-factor to be $a = 7.9\text{e} - 5$ m/(sPa^n). The initial pressure of the closed chamber is set to be 1 atm. As the droplets undergo phase change from denser phase to lighter phase, the pressure increases and speeds up the phase change rate in return. Discontinuous interface is utilized in this case. Further, layered elements are used on both sides of the interface.

Figure 9 shows meshes and solution fields around one of the grains at three instances ($t = t_0$, t_1 and t_2). A cut through the mesh is shown, where the change in grain size/volume is clear between the first and last instances. Solution fields of velocity magnitude and temperature are shown, where at each instance the former is shown in the left half and the latter in the right half. $t = t_0$ is near the beginning while at $t = t_1$ and t_2 mesh modification is triggered. Thus, there are two mesh and solution states for each instance of t_1 and t_2, one after mesh motion (only) and the other after mesh modification. In this case, the mesh size field is prescribed to be finer near the interface and is based on the current state of the geometry (i.e., it evolves in time with the geometry). Note that the structure, shape and resolution of the anisotropic layered elements are maintained during mesh motion and modification. The overall mesh quality is maintained to be above the threshold value of 0.3 as discussed in Section 2.1. Further, the solution fields are well resolved, especially at the interface including discontinuity in the normal component of the velocity and steep normal gradient in the velocity and temperature fields.

4.2 Projectile Case with Rigid Motion

A finned projectile inside a pressurized cannon is considered. In this case, the projectile velocity and flow field at the exit of the cannon are of interest (e.g., to design a muzzle brake). Figure 10 shows the problem setup for this case, where high pressure and temperature is set at the closed end of the cannon. The high pressure pushes the projectile towards the open end. The projectile is considered to be a rigid object. It translates axially along the tube. It is 711mm in length and has 8 fins. The inner length of the cannon is 2000 mm, while its inner diameter and thickness are 120 mm and 10 mm, respectively. The outer boundary (not shown) is set as outflow and placed at a sufficiently long distance away from the cannon. Currently, no-slip and zero heat flux conditions are set on all the walls, where no-slip condition due to the projectile is taken at the contact between projectile and inner side of the cannon. Since we are primarily interested in the projectile motion and gaseous flow, material inside the projectile and cannon walls are not included in the current simulations.

A stabilized finite element method for pressure-primitive variables with a discontinuity capturing operator is used, see [54] for more details. The projectile starts at 20 mm from the left inner side of the cannon and moves about 1300 mm in 6 ms such that the projectile reaches near the exit of the cannon. Two mesh size fields are used in this case. One that is prescribed to be finer near the projectile and is based on the current state of the geometry (i.e., it evolves in time with the geometry). Other which is computed adaptively using a VMS-based error estimator.

We first focus our attention on the geometry-based prescribed mesh size field. Evolution of the geometry and mesh is shown in Figure 11 at six different instances as the projectile moves from the left/closed end to the right/open end of the cannon. A cut through the mesh is shown. A zoomed view near the projectile nose is shown for all six instances at the top while a zoomed view near the fins is shown at the bottom. The mesh refinement around the projectile is maintained as dictated by the geometry-based prescribed mesh size field. Similarly, a finer mesh is maintained near the tip of the fins. Figure 12 shows the parallel mesh partition at the same instances (different colors indicate different parts). Throughout the simulation, the partitioned mesh consists close to 13.5 million elements in total on 256 parts.

Figure 13 shows the minimum mesh quality over time, it is clear that the threshold value is satisfied throughout the simulation. Note that the rate of mesh modification becomes relatively higher towards the end of the simulation, which is expected as the projectile reaches the exit of the cannon and a topological change in the geometry is imminent.

Figure 14 shows the evolution of the normalized projectile velocity over time. The maximum projectile velocity is roughly 330 m/s, which is reached near the exit of the cannon.

Figures 15 and 16 show the Mach number and numerical schlieren at four different instances. The precursor wave in front of projectile is clearly formed at t =2.5 ms and it propagates out of the cannon by t =3.5 ms. In the later two instances of t =4.5 ms and 5.5 ms, the barrel shock is clearly formed and at the last instance a complex shock structure is observed. These features

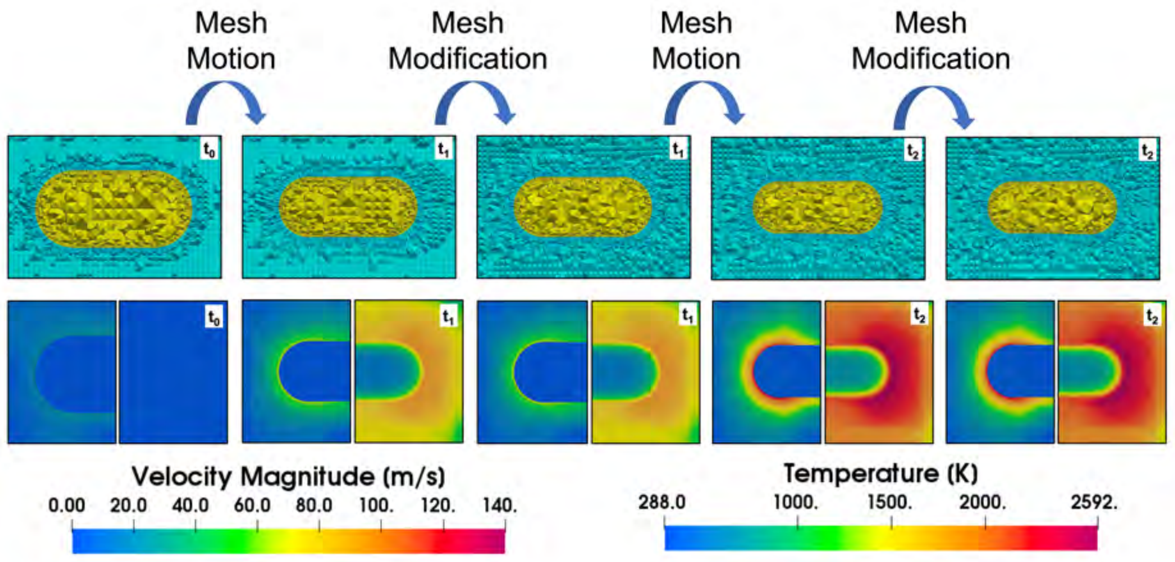


Fig. 9 Mesh cut (top row), and solution fields (bottom row), around one of the 6 grains at three instants ($t = t_0$, t_1 and t_2), where $t = t_0$ is near the beginning while at $t = t_1$ and t_2 mesh modification is triggered

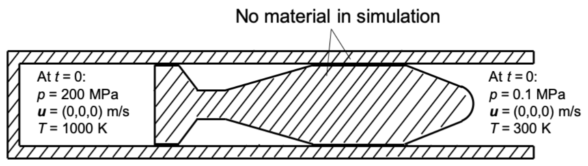


Fig. 10 Setup of the projectile case

are common for a case with a fast moving projectile in a cannon.

Next we focus our attention on the simulation employing the error-based mesh size field. In the initial instances, the error-based adapted mesh remains about the same to the mesh based on the prescribed mesh size field, where some additional refinements are observed near the exit of the cannon as the precursor wave propagates out of the cannon (e.g., at t =3.5 ms). However, with the formation of the barrel shock by t =4.5 ms and a complex shock structure by t =5.5 ms, the error-based adapted mesh is focuses resolution in the shock regions. The total number of elements reaches a maximum of about 8.5 million in the case of the error-based adapted mesh. We recall that the total number of elements remains about 13.5 million in the mesh based on the prescribed mesh size field. Figures 17 and 18 show a cut view of the mesh zoomed near the exit of the cannon at two instances of t =4.5 ms and 5.5 ms. Meshes based on both types of the mesh size field are presented to clearly show the utility of the error-based mesh adaptation. Figures 19 and 20 show a zoomed view of the Mach number near the exit of the cannon at the same two instances. As expected, the complex shock structure is resolved crisply for the error-based adapted mesh and thus, leads to a higher accuracy.

5 Closing Remarks

We presented a parallel interface tracking approach for evolving geometry problems. In our approach, the computational domain is defined using a boundary representation-based geometric model that is updated as dictated by the analysis. An interface-fitted conforming hybrid/mixed mesh with anisotropic layered elements is used. The mesh is updated to be consistent with the updated geometry at every instance. Mesh is updated using a combination of mesh motion and mesh modification. Mesh modification is triggered only when necessary. Further, during mesh motion and modification the desired structure, shape and resolution of the anisotropic layered elements at interface are maintained. All steps are performed on partitioned meshes on distributed-memory parallel computers.

We demonstrated our approach on two problems with large motion or deformation in the geometry. The first problem involved 6 grains undergoing phase change with arbitrary deformations. The second problem included a rigid projectile moving/translating down the barrel, where an error-based adapted mesh was shown to provide a highly accurate solution.

Acknowledgements This work is supported by the U.S. Army grants W911NF1410301 and W911NF16C0117.

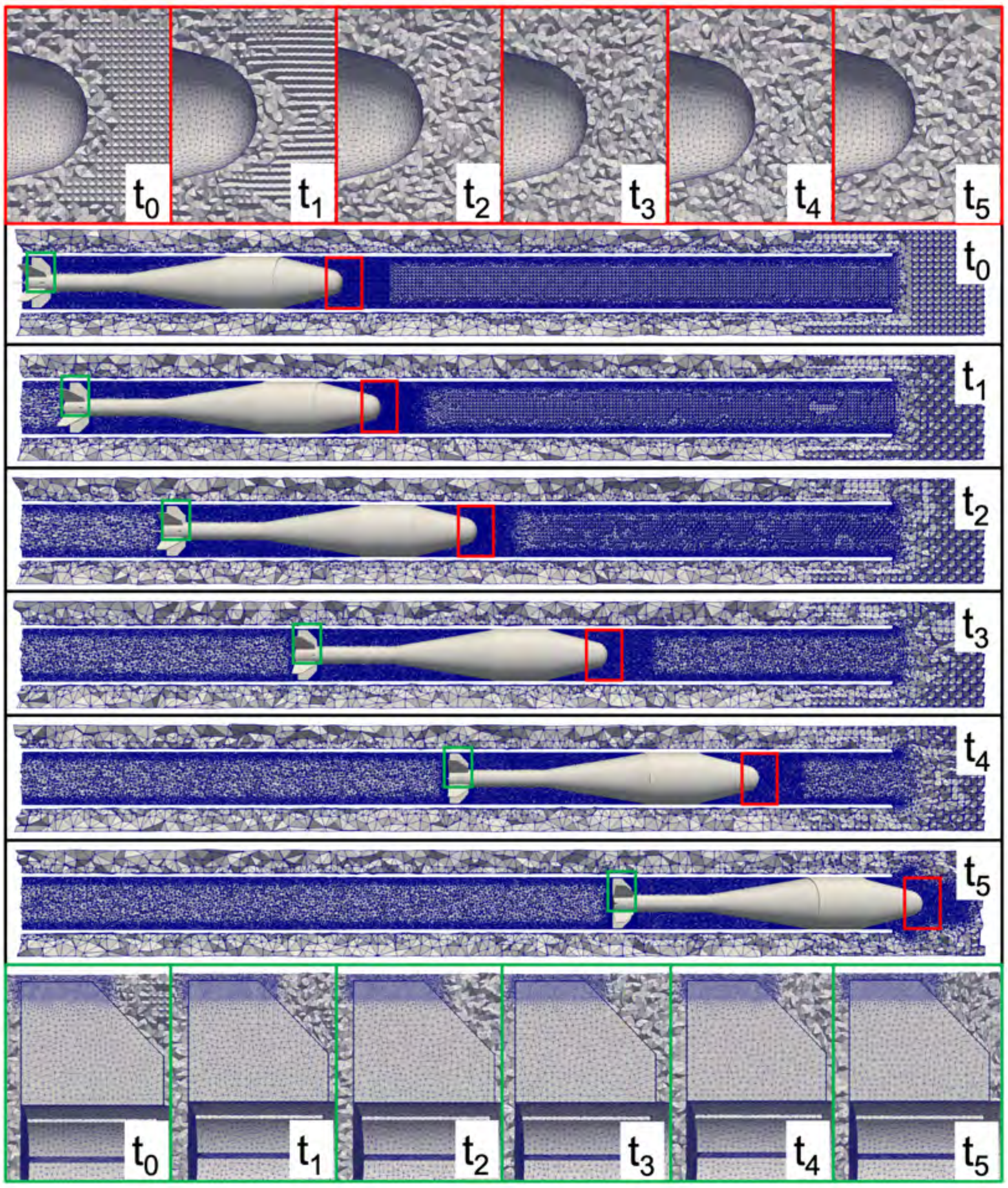


Fig. 11 Mesh at six different instances (cut view)

References

1. Alauzet F, Li X, Seol ES, Shephard MS (2006) Parallel anisotropic 3D mesh adaptation by mesh modification. Eng Comput 21(3):247–258
2. Anderson DM, McFadden GB, Wheeler AA (1998) Diffuse-interface methods in fluid mechanics. Annu Rev Fluid Mech 30(1):139–165
3. Barral N, Alauzet F (2019) Three-dimensional CFD simulations with large displacement of the geometries using a connectivity-change moving mesh approach. Eng Comput 35(2):397–422
4. Batina JT (1990) Unsteady euler airfoil solutions using unstructured dynamic meshes. AIAA J 28(8):1381–1388
5. Boettinger WJ, Warren JA, Beckermann C, Karma A (2002) Phase-field simulation of solidification. Ann Rev Mater Res 32(1):163–194
6. Breil J, Harribey T, Maire PH, Shashkov M (2013) A multi-material ReALE method with MOF interface reconstruction. Comput Fluids 83:115–125
7. Burg C (2004) A robust unstructured grid movement strategy using three-dimensional torsional springs. In: 34th AIAA Fluid Dynamics Conference and Exhibit, p 2529

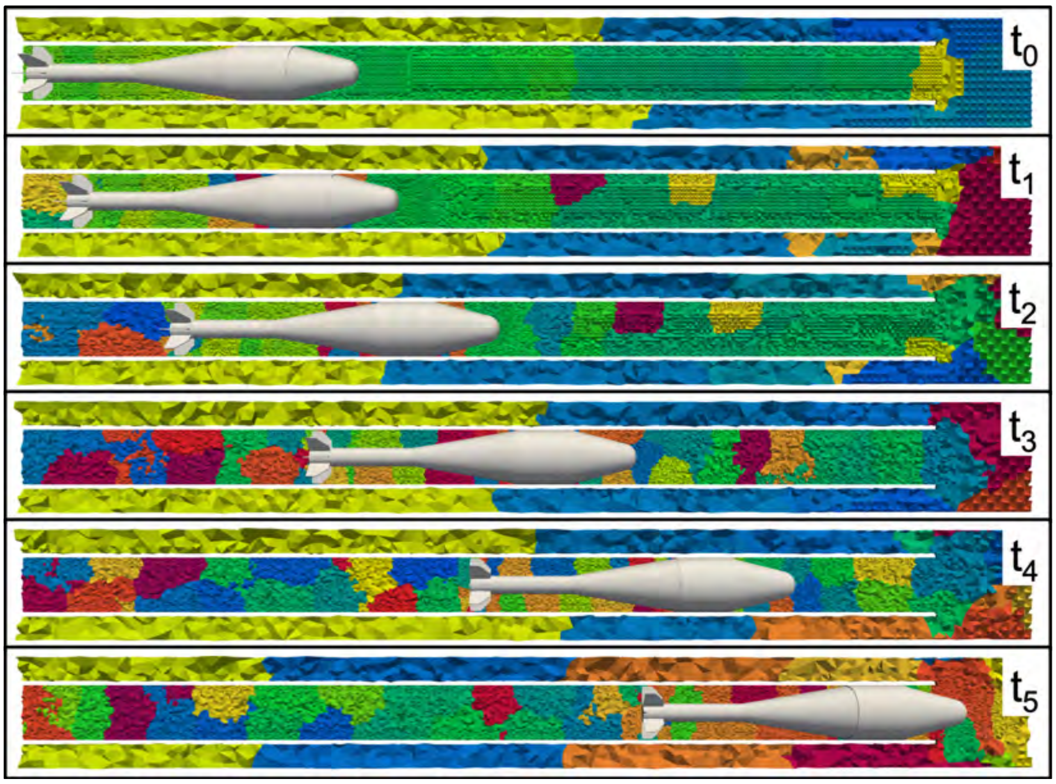


Fig. 12 Parallel mesh partition at six different instances (cut view)

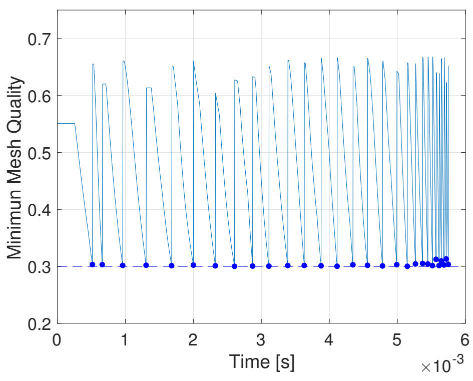


Fig. 13 Minimum mesh quality over time (the dashed line indicates the threshold value that is used to trigger mesh modification and solid dots are used to indicate the instances when mesh modification is applied)

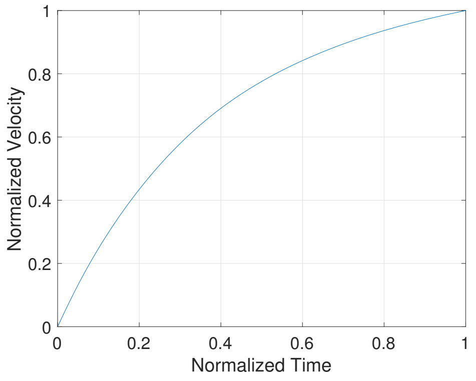
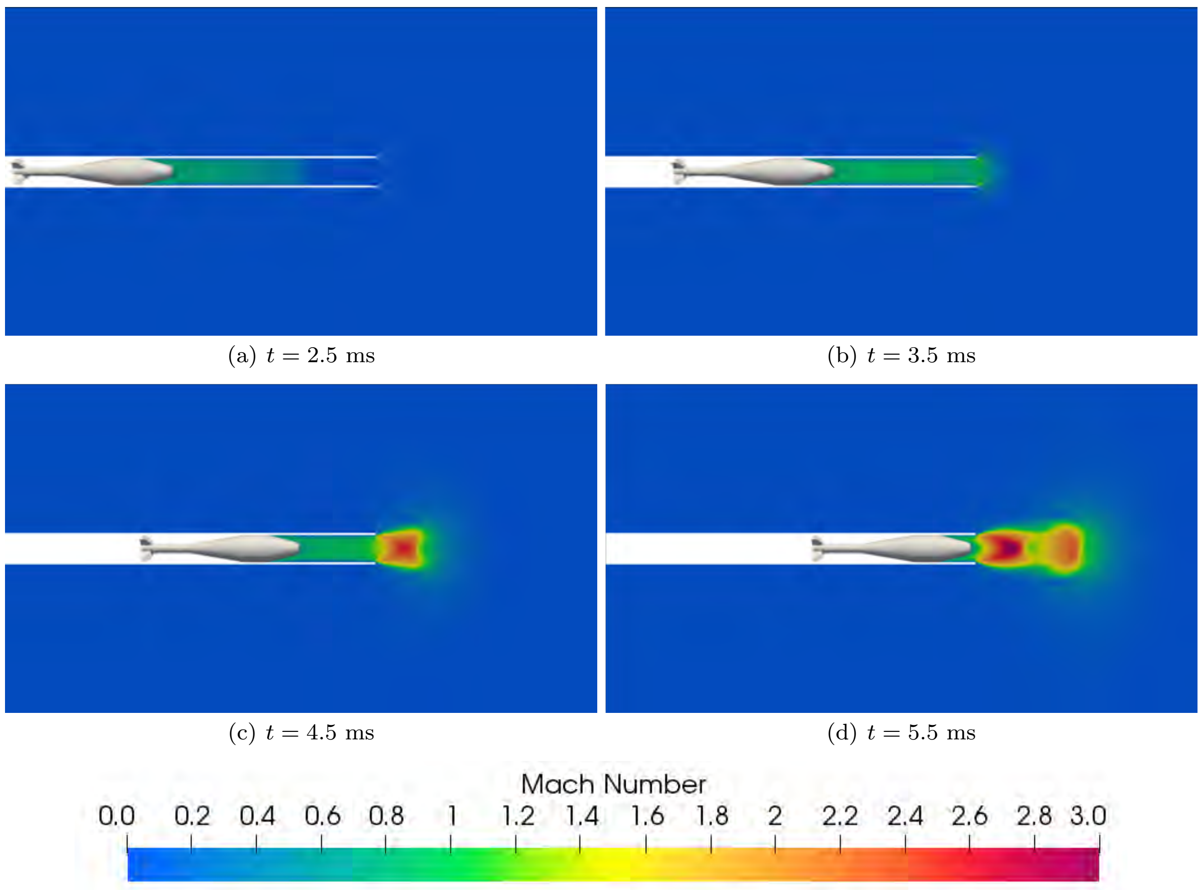


Fig. 14 Normalized projectile velocity with time

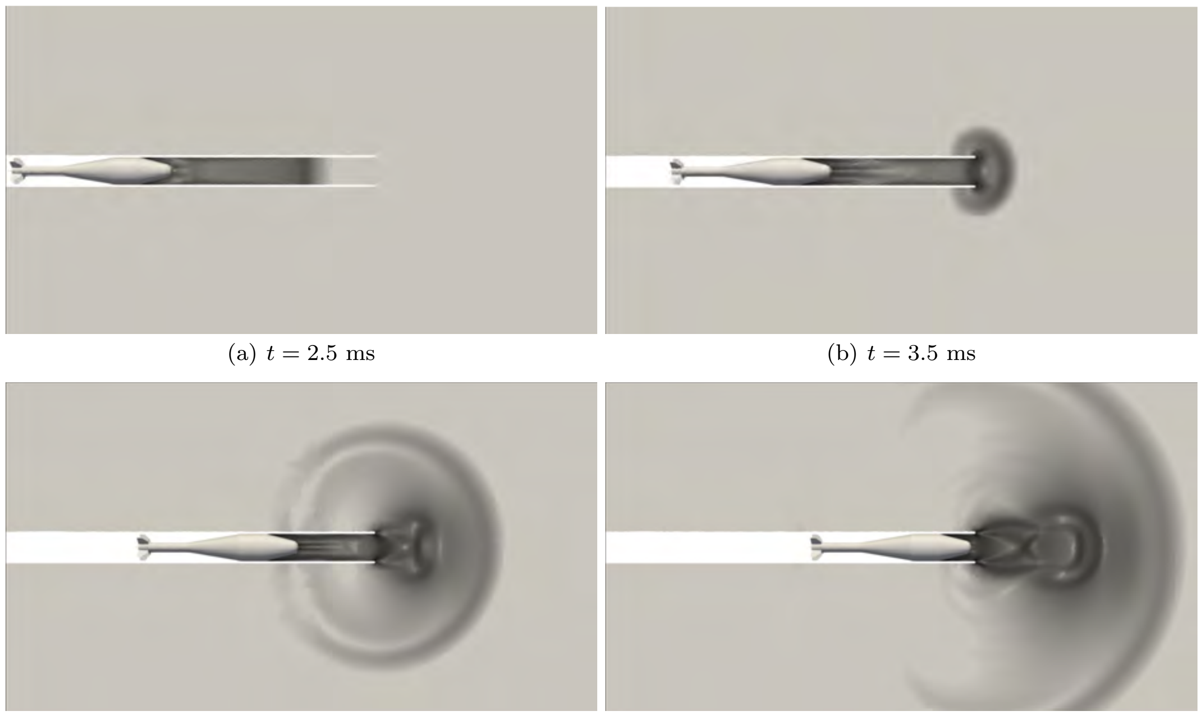
8. Chessa J, Belytschko T (2003) An extended finite element method for two-phase fluids. J Appl Mech 70(1):10–17
9. Chitale KC, Sahni O, Shephard MS, Tendulkar S, Jansen KE (2014) Anisotropic adaptation for transonic flows with turbulent boundary layers. AIAA J 53(2):367–378
10. Del Pino S (2011) Metric-based mesh adaptation for 2D Lagrangian compressible flows. J Comput Phys 230(5):1793–1821
11. Dobrev VA, Kolev TV, Rieben RN (2012) High-order curvilinear finite element methods for Lagrangian hydrodynamics. SIAM J Sci Comput 34(5):B606–B641
12. Donea J, Giuliani S, Halleux JP (1982) An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions. Comput Methods Appl Mech Engrg 33(1-3):689–723
13. Dwight RP (2009) Robust mesh deformation using the linear elasticity equations. Comput Fluid Dyn 2006 pp 401–406



(a) $t = 2.5$ ms (b) $t = 3.5$ ms

(c) $t = 4.5$ ms (d) $t = 5.5$ ms

Fig. 15 Mach number at four different instances (cut view), for clarity only ambient air region is shown



(a) $t = 2.5$ ms (b) $t = 3.5$ ms

(c) $t = 4.5$ ms (d) $t = 5.5$ ms

Fig. 16 Numerical schlieren at four different instances (cut view), for clarity only ambient air region is shown

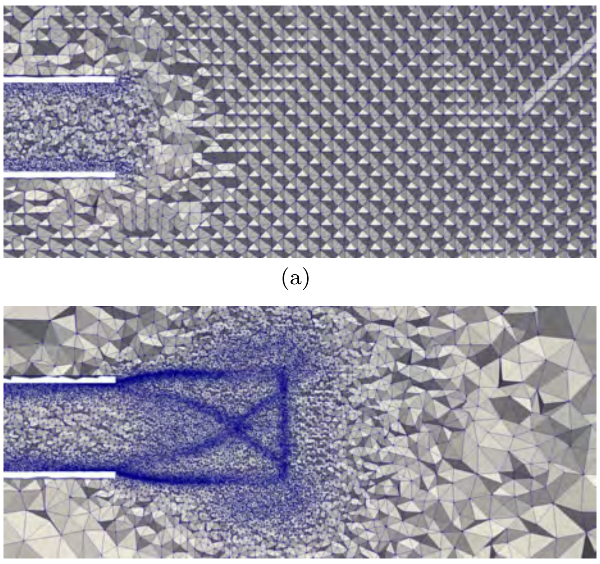


Fig. 17 Comparison of mesh near the exit (cut view) with prescribed (upper) and error-based (lower) mesh size fields at $t = 4.5$ ms

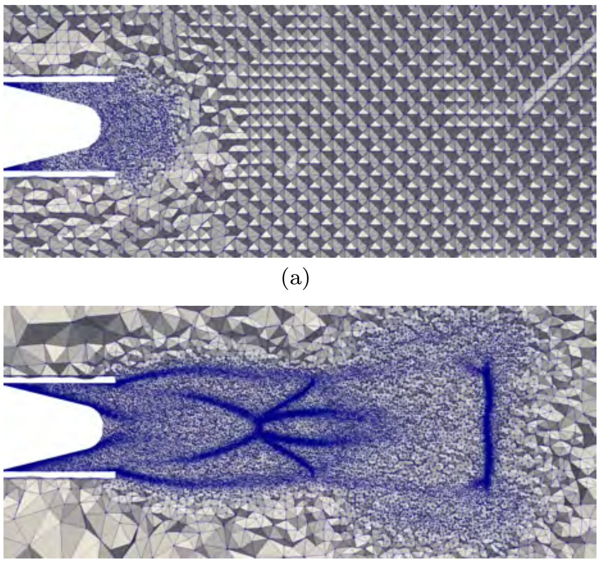


Fig. 18 Comparison of mesh near the exit (cut view) with prescribed (upper) and error-based (lower) mesh size fields at $t = 5.5$ ms

14. Dyadechko V, Shashkov M (2008) Reconstruction of multi-material interfaces from moment data. J Comput Phys 227(11):5361–5384
15. Farhat C, Degand C, Koobus B, Lesoinne M (1998) Torsional springs for two-dimensional dynamic unstructured fluid meshes. Comput Methods Appl Mech Engrg 163(1-4):231–245
16. Fritts M, Boris J (1979) The lagrangian solution of transient problems in hydrodynamics using a trian-

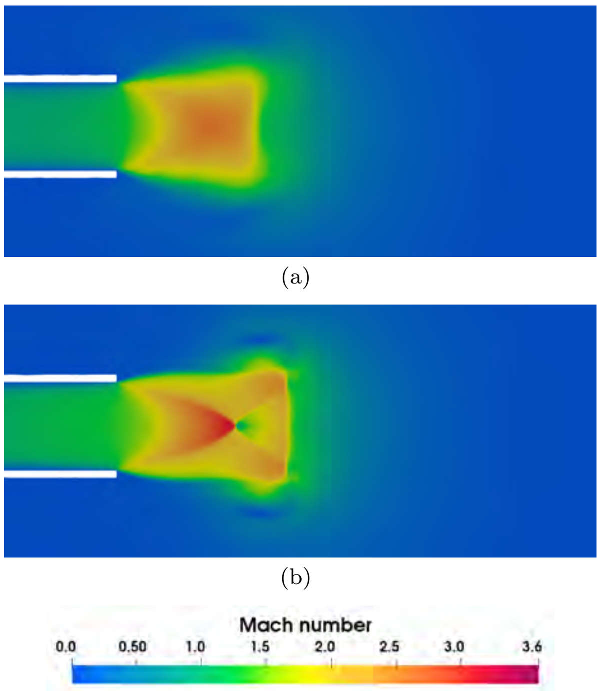


Fig. 19 Comparison of Mach number near the exit (cut view) with prescribed (upper) and error-based (lower) mesh size fields at $t = 4.5$ ms

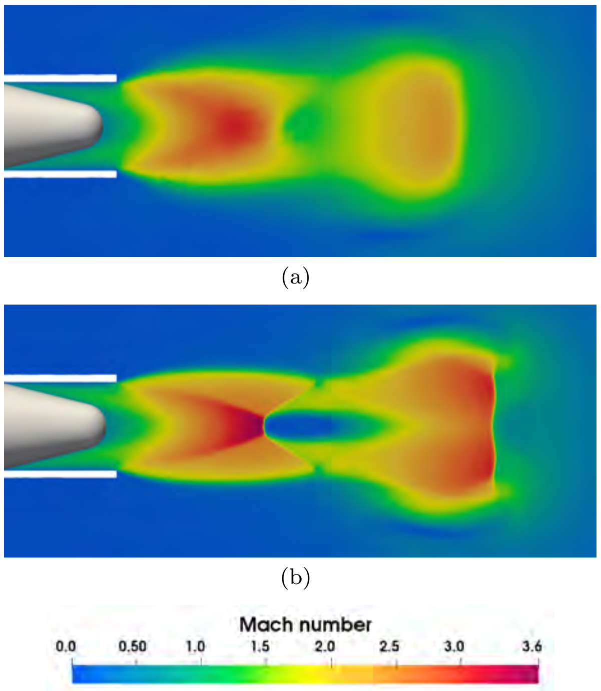


Fig. 20 Comparison of Mach number near the exit (cut view) with prescribed (upper) and error-based (lower) mesh size fields at $t = 5.5$ ms

gular mesh. J Comput Phys 31(2):173–215

17. Fyfe DE, Oran ES, Fritts M (1988) Surface tension and viscosity with Lagrangian hydrodynamics on a triangular mesh. J Comput Phys 76(2):349–384
18. Garimella RV, Shephard MS (2000) Boundary layer mesh generation for viscous flow simulations. Internat J Numer Methods Engrg 49(1-2):193–218
19. Glimm J, Grove JW, Li XL, Shyue Km, Zeng Y, Zhang Q (1998) Three-dimensional front tracking. SIAM J Sci Comput 19(3):703–727
20. Gropp W, Gropp WD, Lusk ADFEE, Lusk E, Skjellum A (1999) Using MPI: portable parallel programming with the message-passing interface, vol 1. MIT press
21. Guventurk C, Sahin M (2017) An arbitrary Lagrangian–Eulerian framework with exact mass conservation for the numerical simulation of 2D rising bubble problem. Internat J Numer Methods Engrg 112(13):2110–2134
22. Hassan O, Sørensen K, Morgan K, Weatherill N (2007) A method for time accurate turbulent compressible fluid flow simulation with moving boundary components employing local remeshing. Internat J Numer Methods Fluids 53(8):1243–1266
23. Hauke G, Fuster D, Lizarraga F (2015) Variational multiscale a posteriori error estimation for systems: The Euler and Navier–Stokes equations. Comput Methods Appl Mech Engrg 283:1493–1524
24. Hirt C, Amsden AA, Cook J (1974) An arbitrary Lagrangian-Eulerian computing method for all flow speeds. J Comput Phys 14(3):227–253
25. Hu HH, Patankar NA, Zhu M (2001) Direct numerical simulations of fluid–solid systems using the arbitrary Lagrangian–Eulerian technique. J Comput Phys 169(2):427–462
26. Hughes TJ, Liu WK, Zimmermann TK (1981) Lagrangian-Eulerian finite element formulation for incompressible viscous flows. Comput Methods Appl Mech Engrg 29(3):329–349
27. Ibanez DA, Seol ES, Smith CW, Shephard MS (2016) PUMI: Parallel unstructured mesh infrastructure. ACM Trans Math Software 42(3):17
28. Ibanez DA, Love E, Voth TE, Overfelt JR, Roberts NV, Hansen GA (2019) Tetrahedral mesh adaptation for lagrangian shock hydrodynamics. Comput Math Appl 78(2):402–416
29. Ito Y, Nakahashi K (2002) Unstructured mesh generation for viscous flow computations. In: IMR, pp 367–377
30. Jansen KE, Shephard MS, Beall MW (2001) On anisotropic mesh generation and quality control in complex flow problems. In: IMR, Citeseer
31. Knupp P (2012) Introducing the target-matrix paradigm for mesh optimization via node-movement. Eng Comput 28(4):419–429
32. Li X, Shephard MS, Beall MW (2005) 3D anisotropic mesh adaptation by mesh modification. Comput Methods Appl Mech Engrg 194(48-49):4915–4950
33. Loubère R, Maire PH, Shashkov M, Breil J, Galera S (2010) Reale: a reconnection-based arbitrary-lagrangian–eulerian method. J Comput Phys 229(12):4724–4761
34. Nielsen EJ, Anderson WK (2002) Recent improvements in aerodynamic design optimization on unstructured meshes. AIAA J 40(6):1155–1163
35. Osher S, Fedkiw RP (2001) Level set methods: an overview and some recent results. J Comput Phys 169(2):463–502
36. Peskin CS (2002) The immersed boundary method. Acta Numer 11:479–517
37. Quan S, Schmidt DP (2007) A moving mesh interface tracking method for 3D incompressible two-phase flows. J Comput Phys 221(2):761–780
38. Rodriguez JM, Sahni O, Lahey Jr RT, Jansen KE (2013) A parallel adaptive mesh method for the numerical simulation of multiphase flows. Comput Fluids 87:115–131
39. Sahni O, Jansen KE, Shephard MS, Taylor CA, Beall MW (2008) Adaptive boundary layer meshing for viscous flow simulations. Eng Comput 24(3):267–285
40. Sahni O, Carothers CD, Shephard MS, Jansen KE (2009) Strong scaling analysis of a parallel, unstructured, implicit solver and the influence of the operating system interference. Sci Program 17(3):261–274
41. Sahni O, Zhou M, Shephard MS, Jansen KE (2009) Scalable implicit finite element solver for massively parallel processing with demonstration to 160k cores. In: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, IEEE, pp 1–12
42. Sahni O, Luo X, Jansen K, Shephard M (2010) Curved boundary layer meshing for adaptive viscous flow simulations. Finite Elem Anal Des 46(1):132–139
43. Sahni O, Ovcharenko A, Chitale KC, Jansen KE, Shephard MS (2017) Parallel anisotropic mesh adaptation with boundary layers for automated viscous flow simulations. Eng Comput 33(4):767–795
44. Scardovelli R, Zaleski S (1999) Direct numerical simulation of free-surface and interfacial flow. Annu Rev Fluid Mech 31(1):567–603

45. Sethian JA, Smereka P (2003) Level set methods for fluid interfaces. Annu Rev Fluid Mech 35(1):341–372
46. Smith CW, Granzow B, Diamond G, Ibanez D, Sahni O, Jansen KE, Shephard MS (2018) In-memory integration of existing software components for parallel adaptive unstructured mesh workflows. Concurr Comp Pract E 30(18):e4510
47. Stein K, Tezduyar TE, Benney R (2004) Automatic mesh update with the solid-extension mesh moving technique. Comput Methods Appl Mech Engrg 193(21-22):2019–2032
48. Sussman M, Smereka P, Osher S (1994) A level set approach for computing solutions to incompressible two-phase flow. J Comput Phys 114(1):146–159
49. Tryggvason G, Bunner B, Esmaeeli A, Juric D, Al-Rawahi N, Tauber W, Han J, Nas S, Jan YJ (2001) A front-tracking method for the computations of multiphase flow. J Comput Phys 169(2):708–759
50. Wan J, Kocak S, Shephard MS (2005) Automated adaptive 3D forming simulation processes. Eng Comput 21(1):47–75
51. Welch SW (1995) Local simulation of two-phase flows including interface tracking with mass transfer. J Comput Phys 121(1):142–154
52. Yang Z, Mavriplis DJ (2007) Mesh deformation strategy optimized by the adjoint method on unstructured meshes. AIAA J 45(12):2885–2896
53. Zeng D, Ethier CR (2005) A semi-torsional spring analogy model for updating unstructured meshes in 3D moving domains. Finite Elem Anal Des 41(11):1118–1139
54. Zhang Y, Chandra A, Yang F, Shams E, Sahni O, Shephard M, Oberai AA (2019) A locally discontinuous ALE finite element formulation for compressible phase change problems. J Comput Phys 393:438–464