

# LA-UR-12-25760

Approved for public release; distribution is unlimited.

Title: Final Reports from the Los Alamos National Laboratory Computational Physics Student Summer Workshop

Author(s): Runnels, Scott R.  
Amelang, Jeffrey S.  
Ingraham, Daniel J.  
Jemison, Matthew B.  
McDermott, Danielle M.  
Pusateri, Elise N.  
Wang, Matthew Y.  
David, Sean  
Hoey, William  
Lung, Tyler  
Marcath, Matthew  
Matern, William  
Melvin, Jeremy  
Miller, Sean  
Shaner, Samuel  
Smith, Jeffrey  
Snyder, Evan  
Trettel, Ben  
Winters, Andrew

Intended for: To provide a record of workshop accomplishments and to use in promoting the workshop.  
Report



**Disclaimer:**

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.



# Final Reports

From the

## Los Alamos National Laboratory Computational Physics Student Summer Workshop

Assembled by: Scott R. Runnels, Ph.D.  
Workshop Coordinator and  
University Liaison for LANL's Advanced  
Scientific Computing Program

### Included in this Report

---

#### (1) Background Information

Philosophy of the Workshop  
Funding and Participation Profile  
Lecture Schedule

#### (2) Student Reports

# Table of Contents

## Background (Scott Runnels)

---

Philosophy of the Workshop	4
Funding and Participation Profile	5
Lecture Schedule	6

## Student Reports by Project

---

### Capturing Material Discontinuities and GPU Programming (Bob Robey, mentor)

“Numerically Tracking Contact Discontinuities” Sean Davis and Will Matern	11
--	----

### Methods Supporting Multi-Material Cell Calculations (Misha Shashkov, mentor)

“Moment of Fluid Interface Reconstruction with Increased Sub-gridcell Resolution, Part I”, Matthew Jemison	20
“Moment of Fluid Interface Reconstruction with Increased Sub-gridcell Resolution, Part II”, Andrew Winters	43

### Advanced Cell-Centered Hydro Methods (Nathaniel Morgan, mentor)

“Report on an FCT Method with Vorticity Control” Tyler Lung	81
“Validation of Two Hydrocodes with a Bi-Metallic Shaped Charge Experiment”, Daniel Ingraham	98

### Rad-Hydro Verification (Scott Ramsey, mentor)

“Development and Implementation of Radiation-Hydrodynamics Verification Test Problems”, Matthew Marcath and Matthew Wang	123
---	-----

**EMP Simulations****(Heidi Tierney, mentor)**

“Electromagnetic Pulse Simulations – Swarm Electron Time Delay in Thunderstorm Environment”, Elise Pusateri and Evan Snyder 192

**Grain Boundary Formation Simulation****(Cynthia Reichhardt, mentor)**

“Pattern Formation in a 2D Colloid System” 213  
Danielle McDermott and Jeffrey Amelang

**Turbulent Mixing****(Rob Gore, mentor)**

“Turbulent modeling of a plane mixing layer” 226  
Jeffrey Smith  
“RANS modeling of RTI and HVDT with BHR<sub>3</sub>”, 245  
Ben Trettel

**Development of an ICF Mix Code****(Erik Vold, mentor)**

“Development of ICF Mix Code”, 268  
Jeremy A. Melvin and Sean T. Miller

**Verification of Shocks in Plasmas****(Tom Masser, mentor)**

“Verification of Planar Shocks through Dense Plasma” 290  
William A. Hoey  
“Verification Study of Planar Shocks in Dense Plasmas” 305  
Samuel Shaner

# Background

## Philosophy of the Workshop

The two primary purposes of LANL's Computational Physics Student Summer Workshop are (1) To educate graduate and exceptional undergraduate students in the challenges and applications of computational physics of interest to LANL, and (2) Entice their interest toward those challenges. Computational physics is emerging as a discipline in its own right, combining expertise in mathematics, physics, and computer science. The mathematical aspects focus on numerical methods for solving equations on the computer as well as developing test problems with analytical solutions. The physics aspects are very broad, ranging from low-temperature material modeling to extremely high temperature plasma physics, radiation transport and neutron transport. The computer science issues are concerned with matching numerical algorithms to emerging architectures and maintaining the quality of extremely large codes built to achieve multi-physics calculations. Although graduate programs associated with computational physics are emerging, it is apparent that the pool of U.S. citizens in this multi-disciplinary field is relatively small and is typically not focused on the aspects that are of primary interest to LANL. Furthermore, more structured foundations for LANL interaction with universities in computational physics is needed; currently interactions rely almost solely on individuals' personalities and personal contacts. Thus a tertiary purpose of the Summer Workshop is to build an educational network of LANL researchers, university professors, and emerging students to advance the field and LANL's involvement in it.

This was the second year for the Summer Workshop. Like the previous year, the workshop's goals were achieved by bringing into LANL a select group of students recruited from across the United States and immersing them for ten weeks in lectures and interesting research projects. The lectures provided an overview of the computational physics topics of interest this year along with some detailed instruction while the projects gave the students a positive experience accomplishing technical goals. Each team consisted of two students working under one or more mentors from LANL on specific research projects associated with predefined topics. This year, the topics were on verification of simulations in plasma and radiation-hydrodynamics codes, hypervelocity material deformation, plasma physics, graphical processor unit ("GPU") programming, multi-material mixing, turbulence modeling, computational material science, and electromagnetic pulse simulations.

The students' growth was furthered by their participation on teams where their teammates were sometimes of different academic rank. It also developed them by requiring them to produce written and oral reports that they presented to peers, mentors, and management.

---

## Funding and Participation Profile

### LANL Staff

The Advanced Scientific Computing (ASC) Program at Los Alamos National Laboratory, under charge code JPDJ, sponsors the Summer Workshop by funding the workshop coordinator, paying for the lease at the University of New Mexico – Los Alamos campus, and also funding twelve of the eighteen students. The remaining six students were funded by various projects (some of them under ASC and some not), as shown below. This year, there were nine mentors, up from six the previous year. Also, while last year's mentors were almost solely in XCP, participation this year spread to include mentors from XTD, T, and CCS. This broad participation is welcome and it is hoped that it continues in future years. The details of the funding and divisional participation are summarized below.

#### Charge Code JPDJ (ASC Sponsoring Code):

Capturing Material Discontinuities and GPU Programming (Robey, XCP-2)  
Development of ICF Mix Code (Vold, XCP-2)  
Rad-Hydro Verification Test Problems (Ramsey, XCP-8)  
Methods Supporting Multi-Material Cell Calculations (Shashkov, XCP-4)  
Grain Boundary Formation Simulation (Reichhardt, T-1)  
Verification Study of Planar Shocks in Dense Plasmas (Masser, CCS-2)

#### Charge Code J466:

Electromagnetic Pulse Simulations (Tierney, XCP-6)

#### Charge Code J444:

Turbulent Mixing (Gore, XTD-6)

#### Charge Code X96H:

Advanced Cell-Centered Hydro Methods (Morgan, XCP-8)

### Students

Forty-one students applied for admission to the workshop, all eligible U.S. citizens with the breakdown shown in the chart below. The eighteen that ultimately were selected and participated were from the following schools: Caltech, San Diego State, Notre Dame, University of Toledo, Florida State, University of Michigan, University of New Hampshire, Stony Brook State University of New York, University of Washington, RPI, MIT, University of Toledo, UCLA, University of Maryland, and Middle Tennessee State. Their rank breakdown is also shown in the chart below.

---

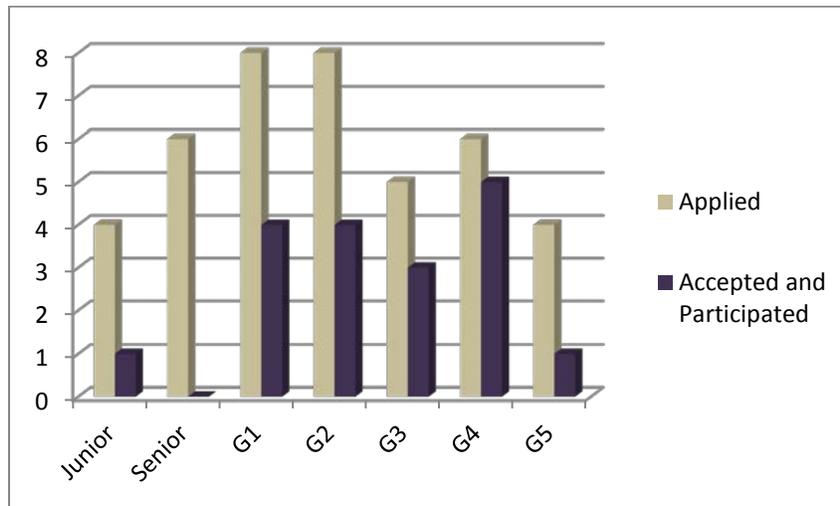


Chart showing the academic rank breakdown of the applicant pool and the ultimate participants. G1 means “1<sup>st</sup> year graduate student.” G2 means “2<sup>nd</sup> year graduate student,” and so on.

## Lecture Schedule

The workshop coordinator and participants greatly appreciated the contributions made by several lecturers, including some from outside LANL. The lectures were scheduled so the students could obtain the most benefit from them. Specifically, they were most frequent in the beginning of the workshop, when the students’ research was just getting started and they needed the most background information. Then, their frequency dropped dramatically and finally to no lectures towards the end so the students could complete their research without interruption. The lectures given and an image of the lecture schedule follow on the next two pages.

Name	Affiliation	Topic	Length in hours
Scott Runnels	LANL	Intro. to C++	2
		Intro. to Grid Data Structures	1
		Intro. to Hydro Terminology	1
		Diffusion on Two Grids	1
		Intro. to Artificial Viscosity	1
Erik Vold	LANL	Computational Transport	2
Bob Robey	LANL	Parallelism: MPI & GPUs	1
Nathaniel Morgan	LANL	Intro. to Lagrange Hydro	1
		Lagrange SGH in r-z	1
		Intro. to ALE	1
		Intro. to CCH Hydro	1
		Test-Driven Hydrocode Development	1
Misha Shashkov	LANL	Interface Reconstruction	1
		Remapping	1
Jim Kamm	Sandia	Verification	3
Greg Weirs	Sandia	Verification	2
Scott Ramsey	LANL	Verification	2
Rob Gore	LANL	Turbulence Modeling	8
Tom Masser	LANL	Shocks in Simple Plasmas	1
Cynthia Reichhardt	LANL	Material Modeling	1
Bill Rider	Sandia	History of Hydrocodes	4
Chris Simmons	UT-Austin	Software Quality	1
Nick Malaya	UT-Austin	Verification w/ Manufactured Sol'ns	1
Bhuvana Srinivasan	LANL	Multi-Fluid Plasmas	1
Ben Bergen	LANL	Computational Sciences	1
Heidi Tierney	LANL	EMP	1

---



# Student Reports

The reports that follow are assembled from separate PDF files. The table of contents at the beginning of this document uses page numbers in this fully assembled PDF file. In other words, it is recommended that the reader use the page indicator in the PDF viewer as the page number when navigating this combined document.

**Capturing Material Discontinuities  
and GPU Programming**

**(Bob Robey, mentor)**

# Numerically Tracking Contact Discontinuities

Sean Davis\* and Will Matern\*

August 15, 2012

## Abstract

We review some of the classic numerical techniques used to analyze contact discontinuities and compare their effectiveness. Several finite difference methods (the Lax-Wendroff method, a Multidimensional Positive Definite Advection Transport Algorithm (MPDATA) method and a Monotone Upstream Scheme for Conservation Laws (MUSCL) scheme with an Artificial Compression Method (ACM)) as well as the finite element Streamlined Upwind Petrov-Galerkin (SUPG) method were considered. These methods were applied to solve the 2D advection equation. Based on our results we concluded that the MUSCL scheme produces the sharpest interfaces but can inappropriately steepen the solution. The SUPG method seems to represent a good balance between stability and interface sharpness without any inappropriate steepening. However, for solutions with discontinuities, the MUSCL scheme is superior. In addition, a preliminary implementation in a GPU program is discussed.

## 1 Introduction

The earthquake and resulting tsunami in Japan caused a release of radioactive material into the Pacific ocean. This radioactive waste was carried by the Kuroshio current into the North Pacific current and is making its way to the US coastline. These currents can produce sharp flow discontinuities with the surrounding ocean water. The idea of tracking these flows and accurately resolving the interfaces between the radioactive and non-radioactive fluids provides a motivation for resolving contact discontinuities.

Many classic techniques have been developed to analyze contact discontinuities. The most simple scheme is a first order upwind method. First order schemes can be proven to preserve their order across discontinuities (monotonicity). Although this method is monotone, a large amount of diffusion error is introduced at each time step. Therefore, higher order methods have been developed to solve problems without introducing such large errors. The difficulty with higher order methods is that, according to Godunov's Theorem, they cannot be monotone across discontinuities. Therefore, limiters have been introduced and will be discussed in Section 2.1.

Several higher order finite difference methods (the Lax-Wendroff method with limiters, a Multidimensional Positive Definite Advection Transport Algorithm (MPDATA) method and a Monotone Upstream Scheme for Conservation Laws (MUSCL) with an Artificial Compression Method (ACM)) as well as the finite element Streamlined Upwind Petrov-Galerkin (SUPG) method are considered and compared for their effectiveness. These methods were applied to solve the 1D advection equation with three types of waves: a square wave, a triangular wave and an exponential smooth wave. Based on these results, we were able to conclude that the MUSCL scheme with artificial compression produces the sharpest interfaces but can inappropriately steepen the solution. The SUPG method seems to represent a good balance between stability and interface sharpness without any inappropriate steepening but is a non-conservative method. However, for solutions with discontinuities, the MUSCL scheme is still superior. Each of these methods is detailed in Section 2 followed by a brief description of the results from the different methods in Section 4. We then discuss the conclusions and future directions we hope to take with the research in Section 5.

---

\*Mentor: Dr. Bob Robey, Los Alamos National Lab

## 1.1 Introduction to GPUs

Graphics Processing Units (GPU) are typically used for accelerating graphics intensive applications. While these devices have been optimized for such applications, their ability to process significant data in parallel can also be harnessed for more general use. Because of recent advances in software, general purpose computing on graphics processing units (GPGPU) is an alternative to using additional CPUs for executing a program. GPGPU has the additional economic advantage of providing a greater number of floating point operations per second (FLOPS) per dollar. The two main languages used to program on GPUs are CUDA and OpenCL. While CUDA has more libraries available, it has the disadvantage of only being able to be used on NVIDIA GPU cards while OpenCL can be used with any programmable GPU. A GPU enabled code was developed to solve the shallow water equations and is described in Section 3.

## 2 Discription of Methods

### 2.1 Lax-Wendroff

The Lax-Wendroff method is a finite difference method designed for hyperbolic PDEs that is second order accurate in both space and time. This scheme can be understood as a two-step method. On the first step the fluxes at a half step in both space and time are calculated. On the second step of the method the half-step fluxes are used to calculate the new value of the conserved variable at each point in the domain. This scheme works well for smooth data but can run into trouble when the solution develops a discontinuity. When this occurs, the solution becomes highly oscillatory around the discontinuity.

It has been shown by numerous authors [4,5] that there is a simple fix for oscillations around discontinuities. First order schemes for hyperbolic PDEs (such as upwind finite differencing) do not lead to oscillations like higher order schemes (Lax-Wendroff). Flux limiters work by interpolating between the flux calculated by a first order scheme and the flux calculated by a higher order scheme. Oscillations near a discontinuity can be eliminated by making the approximation mimic a first order scheme near a steep gradient using a flux limiter and while higher order accuracy is achieved in smoother regions.

It has been mathematically shown that certain formulas for flux limiters in linear PDEs lead to schemes which are monotonicity preserving. This means that if the solution is monotonic at a certain time step it will remain monotonic at the next time step. This property is equivalent to saying that the scheme will not introduce new extrema and that the scheme is Total Variation Diminishing (TVD). We explored several limiters to preserve the monotonicity of our solutions and settled on the Superbee method, which produces the sharpest possible gradients while still being TVD.

### 2.2 MPDATA

Developed by Smolarkiewicz beginning in the early 1980s [6–8], the MPDATA algorithm is a finite-difference approximation for the advective terms in the fluid equations. A donor-cell approximation to the equation is defined in terms of the local Courant number. The resulting equation is a first order upwind finite difference scheme, which is very diffusive.

By analyzing the approximation from the first step using a modified equation analysis, it can be seen that a diffusive convective flux term is added to the model equation. This erroneous diffusion damps out nonphysical oscillations but also overly smears the solution over time.

As an example, the pure advection equation,

$$\frac{\partial \Psi}{\partial t} = -\frac{\partial}{\partial x}(u\Psi)$$

becomes,

$$\frac{\partial \Psi}{\partial t} = -\frac{\partial}{\partial x}(u\Psi) - \frac{\partial}{\partial x} \left( -K \frac{\partial \Psi}{\partial x} \right)$$

using an upstream approximation. The added dissipation error,  $\frac{\partial}{\partial x} \left( K \frac{\partial \Psi}{\partial x} \right)$ , is then subtracted out of the solution using an antidiffusive velocity to cancel the additional convective flux.

### 2.3 MUSCL with ACM

The MUSCL that we used is a modified Osher-Chakravarthy scheme [12], which is a second order TVD approximation to the scalar conservation laws. The modifications produce a third order accurate MUSCL solver [9]. The method is upwind biased and uses a minmod function to avoid numerical oscillations in the solution.

Hartens ACM [1] was initially intended to be used to sharpen contact discontinuities for first order accurate schemes. However, this technique has been extended to higher order methods such as Essentially Non-Oscillatory (ENO) schemes [3], MUSCL [9] and RAGE [2], which is widely used at LANL. This slope modification method relies on a switch to increase or decrease the slope of a function near a discontinuity. For the purposes of this presentation, the advection equation is solved using a MUSCL scheme with ACM.

### 2.4 SUPG

In the finite element method a set of trial functions are assumed to represent the solution of a differential equation over a region. These trial functions have certain degrees of freedom that must be solved for. The differential equation is multiplied by a test function and the new equation is then integrated over the region that the original differential equation is to be solved on. This generally yields one unique algebraic equations for each test function used. The resulting system of algebraic equations are then solved for the degrees of freedom. In Galerkin approaches, the test functions are assumed to be the same as the trial functions. However, when sharp gradients are present Galerkin methods produce highly oscillatory solutions. This can be helped in the case where the differential equation is an advection/diffusion equation by introducing the more general Petrov-Galerkin (PG) approach in which the test functions can be different from the trial functions. The streamline upwind Petrov-Galerkin (SUPG) [10] approach chooses a specific set of test functions which can be observed to significantly reduce oscillations in the solution. For the 2D advection equation these test functions are the same as the trial functions with two additional terms:

$$W = N + \frac{\alpha h}{2} \left( \frac{u}{\sqrt{u^2 + v^2}} \frac{\partial N}{\partial x} + \frac{v}{\sqrt{u^2 + v^2}} \frac{\partial N}{\partial y} \right) \quad (1)$$

Where  $W$  is test function,  $N$  is one of the trial functions,  $\alpha$  is an arbitrary parameter usually taken to be 1 for pure advection,  $h$  is a 1D length of an element, and  $u-v$  are the velocity in the  $x-y$  direction, respectively.

## 3 Wave Code

### 3.1 GPGPU Basics

A GPU can be visualized as thousands of weak processors operating on data that all of the processors can access (global memory). However, there is an important division of these processors that limits what can be processed on a GPU. A warp (Nvidia) or wavefront (AMD) refers to groups of 32 or 64 processors, respectively, that must all run the exact same code but can begin with different data. GPUs are therefore ideal for doing SIMD (Single Instruction Multiple Data) operations [11]. Processors in the same warp/wavefront can also use special shared memory that is much faster than global memory. In OpenCL you must define the number of instances of a kernel (called the work group size) you want to run on the GPU. Choosing a multiple of the warp/wavefront size is the most efficient choice as this means none of the processors will be producing output that won't be used. An important point for GPU programming is that the CPU must move all memory to the GPU card before the GPU can process it. A typical bottleneck for GPU applications is the latency of transferring data between the GPU memory and the CPU memory. Thus it behooves a programmer to make as few data transfers to and from the GPU as possible.

### 3.2 Model Equations and Method

The GPU enabled code we have been working with models the shallow water equations using a Lax-Wendroff method with a Minmod TVD Limiter, as described in Section 2.1. This simplification of the Euler Equations assumes that the length of waves is far greater than the depth of the fluid. This allows for an incompressible model:

$$\begin{aligned} \frac{\partial h}{\partial t} + \frac{\partial(uh)}{\partial x} + \frac{\partial(vh)}{\partial y} &= 0 \\ \frac{\partial(uh)}{\partial t} + \frac{\partial(u^2h + 1/2gh^2)}{\partial x} + \frac{\partial(uvh)}{\partial y} &= 0 \\ \frac{\partial(vh)}{\partial t} + \frac{\partial(uvh)}{\partial x} + \frac{\partial(v^2h + 1/2gh^2)}{\partial y} &= 0 \end{aligned}$$

where  $h$  is the height,  $u$  is the velocity in the  $x$ -direction,  $v$  is the velocity in the  $y$ -direction and  $g$  is the gravitational acceleration. With the introduction of an advection equation for a passive tracer concentration,  $\psi$ :

$$\frac{\partial \psi}{\partial t} + u \frac{\partial \psi}{\partial x} + v \frac{\partial \psi}{\partial y} = 0$$

we can look at our motivational problem, namely the advection of radioactive water across the Pacific Ocean more closely in the future.

## 4 Results

In order to analyze the various methods described in Section 2, we looked at a 1D advection problem. Although we developed our code in 2D, the advection of waves in 1D highlights the similarities and differences between the methods. While we were most interested in resolving sharp discontinuities, we wanted to be sure that the method didn't artificially sharpen other types of waves. We have therefore included a square wave, a triangular wave and an exponential smooth wave in Figures 1-5.

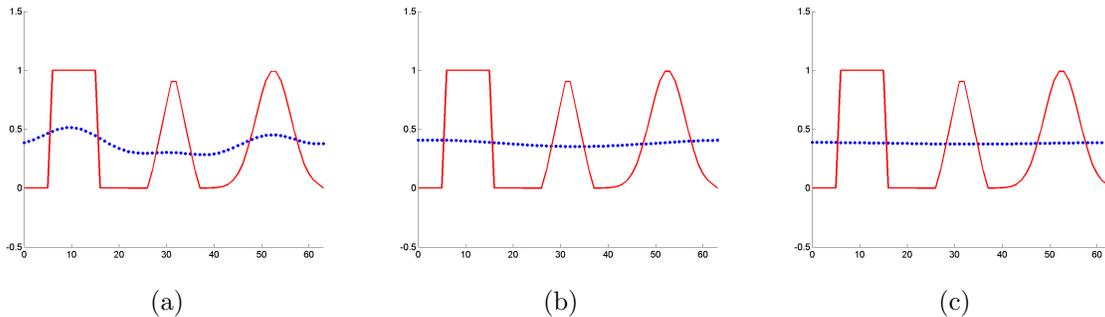


Figure 1: Advection of a square wave, a triangular wave and an exponential wave over time with periodic boundary conditions after (a) 1 pass through the domain, (b) 5 passes through the domain and (c) 10 passes through the domain using a first order upwind method.

A graphics package called MPE is employed in the Wave Code described in Section 3 to show realtime images of the desired variable. The evolution of the height in the shallow water equations as a wave moves from left to right is shown in Figure 6 and a passive tracer concentration is shown in Figure 7 for the same time steps.

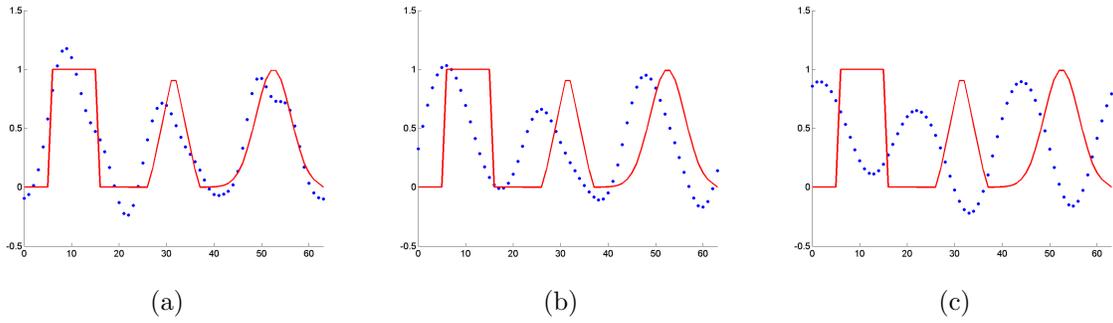


Figure 2: Same caption as Figure 1 except that a Lax-Wendroff method without any slope limiters was used to advect the waves.

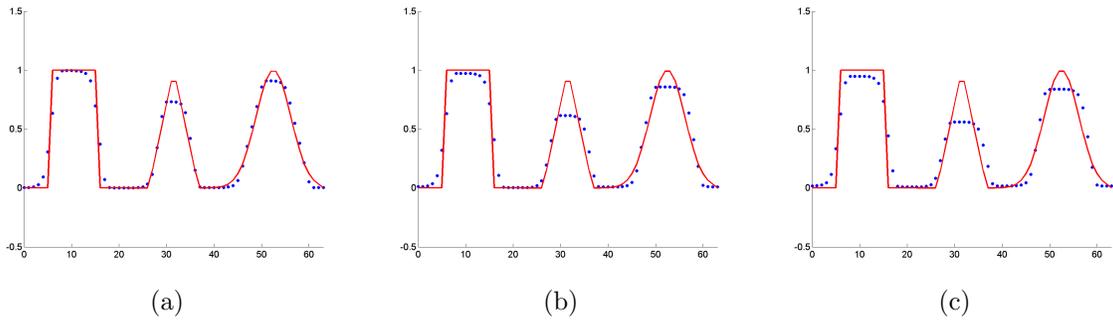


Figure 3: Same caption as Figure 1 except that a Superbee Limiter was used along with the Lax-Wendroff method to advect the waves.

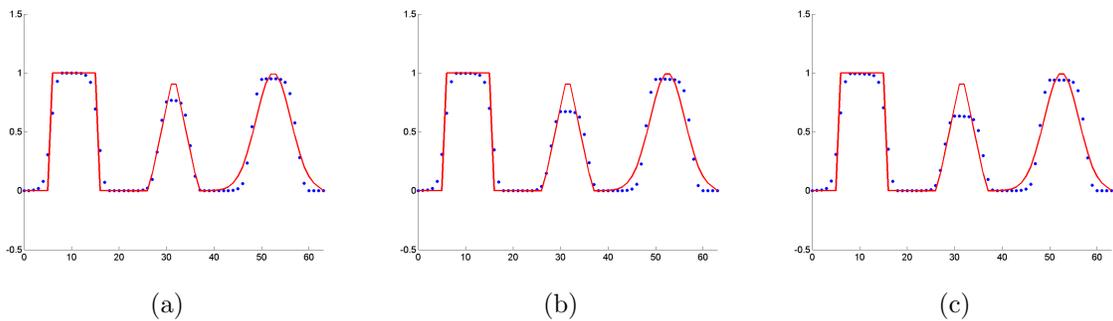


Figure 4: Same caption as Figure 1 except that a MUSCL with an ACM were used advect the waves.

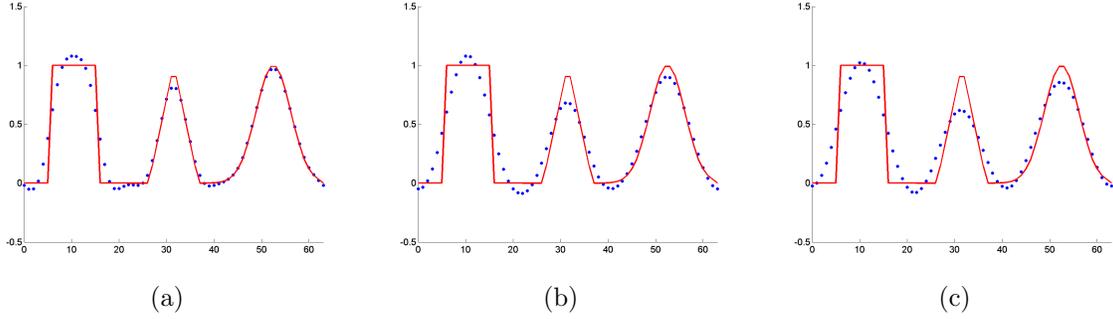


Figure 5: Same caption as Figure 1 except that SUPG was used to advect the waves.

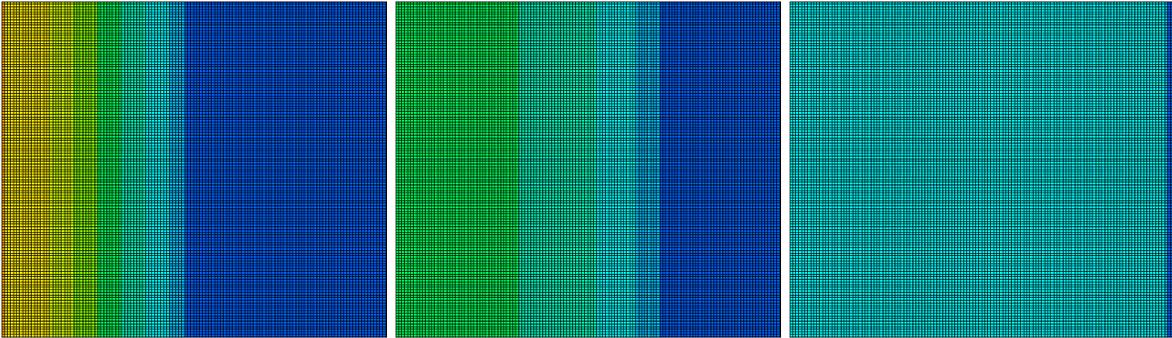


Figure 6: Evolution of the height of a wave over time using the GPU enabled wave code.

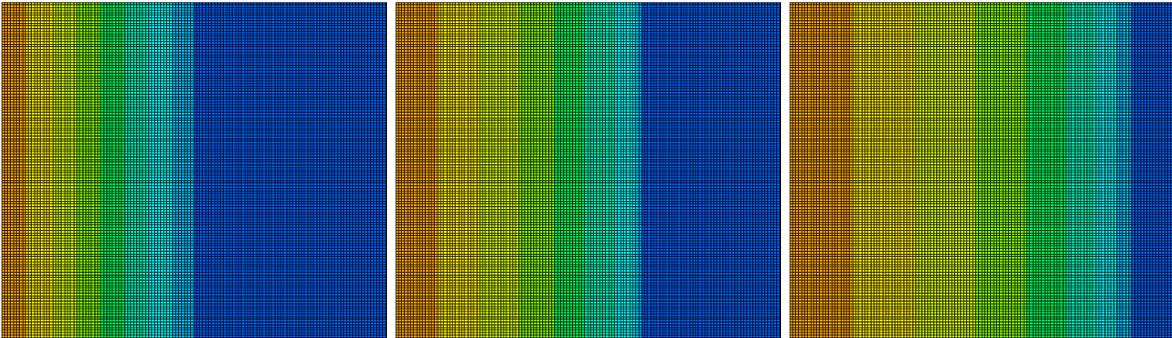


Figure 7: Evolution of the concentration of a dye over time using the GPU enabled wave code.

## 5 Conclusions and Future Directions

As was described in the introduction, the first order scheme shown in Figure 1 will remain stable across a sharp discontinuity. When higher order methods are considered, this is not necessarily the case. The second order Lax-Wendroff scheme does not exhibit monotonicity by itself as can be seen in Figure 2. When a Superbee limiter is introduced as in Figure 3, the method drops to first order near the steep gradients and the method can be considered TVD. Superbee is the TVD best limiter for sharp discontinuities albeit still diffuse but it over sharpens smooth waves.

Both the pros and the cons of the MUSCL are starker than the Lax-Wendroff TVD scheme used. The MUSCL is even more effective at preventing diffusion across sharp interfaces. On the down side, the MUSCL unnaturally sharpens normally smooth waves as is evident from the evolution of the initially exponential distribution in Figure 4.

The SUPG scheme provides a compromise between the over sharpening and diffusion errors by gradually diffusing the solution. However, it is evident from Figure 2.4 that the method is also not conservative. The choice of method is therefore problem specific and most production scale codes should have the option to switch between algorithms.

While, the Lax-Wendroff scheme has been applied to a GPU code, as shown in Figures 6 and 7, other methods will be ported to the GPU. MPDATA has the option of iterating the anti diffusion step and increasing its order of accuracy. Because FLOPS are essentially free on a GPU, this iterative method could produce highly accurate results with little computational time. We will also be looking into new methods, in particular a comparison of the above methods with an H-WENO scheme is being worked on.

## 6 Acknowledgements

We would like to thank our mentor Bob Robey for all of his help and advice throughout the summer as well as assistance from David Nicholaeff. We would also like to thank Scott Runnels for organizing the Computational Physics Summer Workshop, which allowed us to come to Los Alamos National Lab to complete this work.

## References

- [1] Harten A. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 49:357–393, 1983.
- [2] Clover M. Betlach T. Byrne N. Coker R. Dendy E. Hueckstaedt R. New K. Oakes W. R. Ranta D. Gittings M., Weaver R. and Stefan R. The rage radiation-hydrodynamic code. *Computational Science and Discovery*, 1, 2008.
- [3] Yang H. An artificial compression method for eno schemes: The slope modification method. *Journal of Computational Physics*, 89:125–160, 1990.
- [4] Leveque R. J. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.
- [5] Rider W. J. A comparison of tvd lax-wendroff methods. *Communications in Numerical Methods in Engineering*, 9:147–155, 1993.
- [6] Smolarkiewicz P. K. A fully multidimensional positive definite advection transport algorithm with small implicit diffusion. *Journal of Computational Physics*, pages 325–362, 1984.
- [7] Smolarkiewicz P. K. Multidimensional positive definite advection transport algorithm: An overview. *International Journal for Numerical Methods in Fluids*, 50:1123–1144, 2006.

- [8] Smolarkiewicz P. K. and Wojciech G. W. The multidimensional positive definite advection transport algorithm: Nonoscillatory option. *Journal of Computational Physics*, 86:355–375, 1990.
- [9] Wu C. Lin J. Chen Y. Lin S., Chin T. A pressure correction-volume of fluid method for simulation of two-phase flows. *International Journal for Numerical Methods in Fluids*, 68:181–195, 2012.
- [10] Zienkiewicz O. and Taylor R. *The Finite Element Method Volume 2*. Butterworth-Heinemann, 2005.
- [11] Satyamoorthy P. Stt-ram for shared memory in gpus. *University of Virginia*, 2011.
- [12] Osher S. and Chakravarthy S. High resolution schemes and the entropy condition. *Siam Journal of Numerical Analysis*, 21, 1984.

**Methods Supporting Multi-Material  
Cell Calculations**

**(Misha Shashkov, mentor)**

# Moment of Fluid Interface Reconstruction with Increased Sub-gridcell Resolution

Matthew Jemison  
Department of Applied & Computational Mathematics  
Florida State University

Mentor: Dr. Mikhail Shashkov

August 15, 2012

## Abstract

A new approach to increase sub-gridcell resolution of the Moment of Fluid (MOF) Interface Reconstruction is proposed. Instances when separation exists between like-material regions in a cell are detected. Fictitious material labels are introduced to model the system as a multimaterial cell for the purposes of interface reconstruction. Efforts are then made to capture the material configuration using multiple piecewise-linear interfaces that divide a cell into three distinct regions. The procedure for grouping like-material regions and reconstructing the material configuration after advection is described. The method is tested by applying a known, divergence-free velocity to a scalar field representing a material configuration that is deformed by the flow, after which the material interface is reconstructed. Numerical results are presented, along with analysis of the data.

## 1 Introduction

The Moment of Fluid (MOF) method is a piece-wise linear interface reconstruction technique introduced in [5]. Given a mixed cell (i.e. a cell with more than one material), we wish to reconstruct the interface between the materials. Given the volume fraction  $F_1$  and reference centroid  $\mathbf{x}_{ref,1}$  of Material 1 in a cell (Eqn. 1.1,1.2), MOF finds the interface that exactly captures the volume and minimizes the error between the reference (input) centroid and computed centroid. Here,  $\Omega$  refers to the cell, and  $\Omega_1$  refers to the portion of the cell occupied by Material 1.

$$F_1 = \frac{\int_{\Omega_1} d\mathbf{x}}{|\Omega|} \quad (1.1)$$

$$\mathbf{x}_{ref,1} = \frac{\int_{\Omega_1} \mathbf{x} \cdot d\mathbf{x}}{|\Omega|F_1} \quad (1.2)$$

Multimaterial MOF [4] applies the principle of using moment and volume data in interface reconstruction to situations when more than two materials occupy a cell. In general, if  $N$  materials exist in a cell,  $N - 1$  piecewise-linear interfaces must be found. In the case when more than two materials occupy a cell, the final material configuration will be dependent on the order of reconstruction. The process of nested dissection is used to prevent overlap of interfaces. Consider the case where a cell is occupied by Materials 1, 2, 3, etc. At each step, a material is chosen for reconstruction. After the interface is found, the material is removed from the cell domain. Assume without loss of generality that Material 1 is chosen first, Material 2 is chosen second, etc. Then the domain will be dissected as in (1.3).

$$\Omega \cap \Omega_1^C \rightarrow (\Omega \cap \Omega_1^C) \cap \Omega_2^C \rightarrow \text{etc.} \quad (1.3)$$

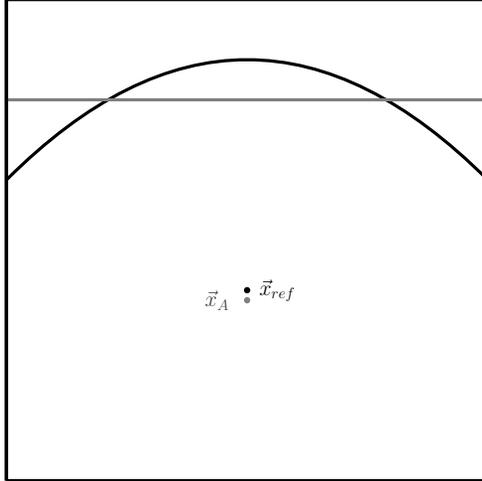


Figure 1: Given the volume under the dark, curved interface and the reference centroid,  $\vec{x}_{ref}$ , MOF will find the piecewise-linear interface that exactly captures the volume and minimizes error between the actual centroid  $\vec{x}_A$  of the reconstructed interface and the reference centroid.

Note that different orderings can result in different material configurations, as different portions of the cell may have been removed after a dissection.

Moment of Fluid is to be contrasted with Volume of Fluid (VOF) Methods [2]. Volume of Fluid methods exactly capture the volume of a material in a cell and use volume fraction information from surrounding cells to compute interface orientation. These techniques are cheaper than Moment of Fluid because they use a closed form to obtain the normal direction of the material interface. However, because they use non-local information in to determine the orientation of the interface, they cannot resolve features smaller than the stencil used in the calculation of the normal. They will not preserve linearity in cases where sharp interfaces lie on cell boundaries, a case well resolved by MOF [5]. Additionally, use of only local information in the reconstruction of a cell interface makes MOF more readily parallelizable.

Both MOF and VOF techniques are capable of modeling multi-material cells (cells that contain more than two materials). VOF uses the ‘onion skin’ model, which requires that at each reconstruction, all previously reconstructed fluids lie behind the interface. This can result in material overlap and fails to capture a T-junction topology or ‘triple point’ where three fluids meet [2]. Moment of Fluid is capable of resolving such material configurations through the use of nested dissection [1]. The local nature of the Moment of Fluid reconstruction coupled with its ability to resolve more complex material configurations makes it preferable for the purposes of this study.

The Moment of Fluid reconstruction breaks down for under-resolved features. Consider the case in Fig. 2. MOF uses volume fraction and centroid information, but has no way to detect connectedness of groups of the same material. In essence, all regions of a given material in a cell are treated as a single entity for reconstruction. In principle, we would like to detect when like-material regions are disconnected and use multiple interfaces in reconstruction, rather than a single interface.

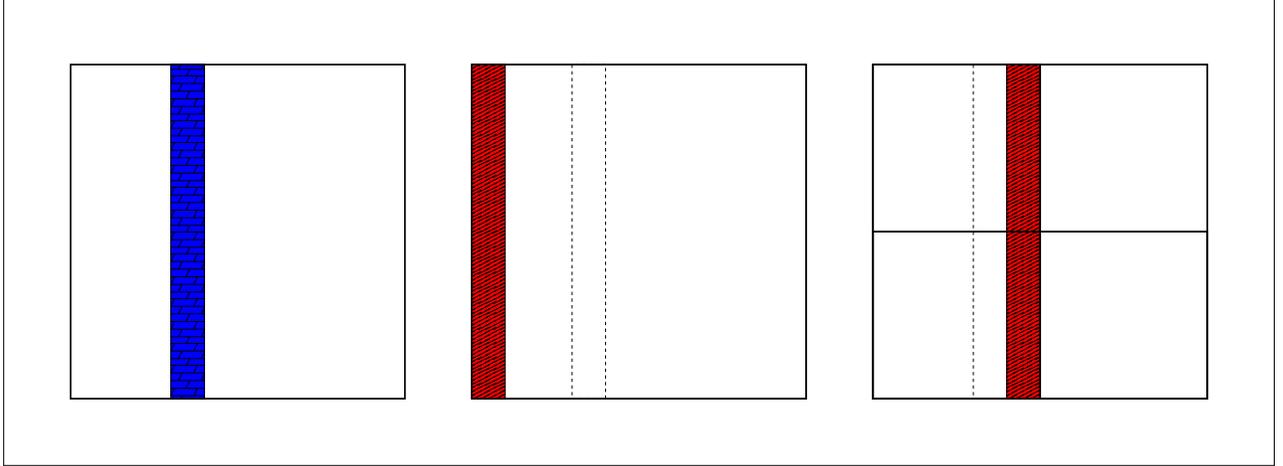


Figure 2: Left: A filament in the interior of a cell. Center: MOF reconstruction of the filament shown in red, with the true solution in dotted lines. Right: MOF reconstruction of the filament in red at higher mesh resolution, with the true solution in dotted lines.

## 2 Advection Procedure

The ideas of Le Chenadec and Pitsch [3] are used in the context of unsplit advection. First, a forward sweep is performed at time level  $t^n$ . Material is advected forward based on the velocities located at cell vertices, then a conservative remapping procedure is applied. This may be viewed as determining where material from a given cell will be transported. Formally, finding the nodes of the target region is equivalent to solving the ODE in Eqn. 2.1 for position  $\mathbf{X}(t^{n+1})$  with the initial condition (2.2) that nodes are located at position  $\mathbf{x}$  at time  $t^n$  [7].

$$\frac{d\mathbf{X}}{dt} = \mathbf{u}(\mathbf{X}, t) \quad (2.1)$$

$$\mathbf{X}(t^n) = \mathbf{x} \quad (2.2)$$

The updated values are used as a predictor for the solution at time level  $t^{n+1}$ , and a backward sweep is performed. The predicted velocity at time level  $t^{n+1}$  is used to advect material at  $t^n$ . This may be viewed as determining what material will arrive at any given cell at  $t^{n+1}$ . Formally, finding the nodes of the departure region is equivalent to solving Eqn. 2.1 for  $\mathbf{X}(t^n)$  with the condition (2.3) that the nodes  $\mathbf{x}$  of the cell are the solution to the ODE at time  $t^{n+1}$ .

$$\mathbf{X}(t^{n+1}) = \mathbf{x} \quad (2.3)$$

For the passive advection problems tested in this report, velocity is known in an analytical form. This analytical form is used to find velocity at corner nodes in the computation of departure and target advection regions.

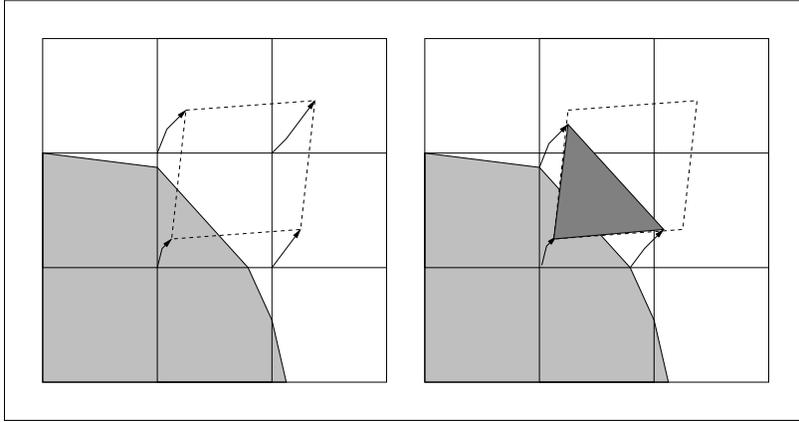


Figure 3: Left: Material configuration at time level  $t^n$ , with forward sweep. The target region is shown as a dotted line. Right: Remapping of the original cell to the target region.

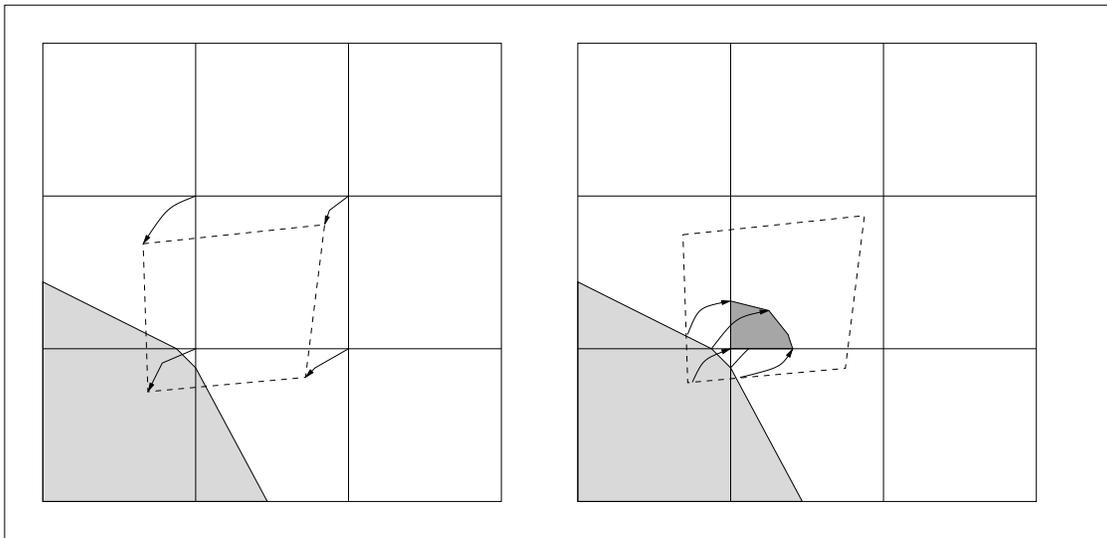


Figure 4: Left: Material configuration at time level  $t^n$ , with backward sweep. The departure region is shown as a dotted line. Right: Remapping of the departure region to the target cell.

### 3 Conglomeration Algorithm

In general, the advection procedure maps portions of material from multiple cells to a single cell (the target cell). Each of these cells is potentially cut by an interface and intersected with the region that will be mapped to the target cell. Once all portions of the intersected cells have been mapped forward to the target cell, the target cell is occupied by a set of polygons.

A conglomeration procedure is used to identify which polygons should be grouped together when reconstructing the material interface. Each polygon carries a material ID, and adjacent polygons with matching material ID's should be grouped into the same region, or conglomerate. The simplest algorithm for forming these material conglomerates is as follows.

1. Choose a starting polygon  $P_i$
2. Identify all ungrouped polygons  $P_j$  that are adjacent to  $P_i$  and have the same material ID.
3. If new polygons have been identified, repeat from Step 1 for each  $P_j$ . If no new polygons are identified, then continue.
4. Denote grouped polygons a completed conglomerate  $C_i$ .
5. Compute volume fraction  $F_i$  and centroid  $\mathbf{x}_i$  of conglomerate  $C_i$ .
6. Identify any ungrouped  $P_{i^*}$ . If no such polygon exists, all conglomerates have been found; exit. If there does exist such a polygon, return to Step 1.

#### 3.1 Computational Considerations

Consider two triangles  $T_i$  and  $T_j$  with sides represented by line segments  $S_{1,i}$ ,  $S_{1,j}$ ,  $S_{2,i}$ , etc. Triangles  $T_i$  and  $T_j$  are considered adjacent if for some pair of sides  $S_{k_1,i}$  and  $S_{k_2,j}$ , (3.1) holds, for some open set  $(a, b)$ .

$$(a, b) \subset S_{k_1,i} \cap S_{k_2,j} \quad (3.1)$$

In essence, triangles should not be conglomerated in the case that the intersection of their sides is the empty set or a single point. Numerically, we cannot directly compare values to test for adjacency of sides, due to round-off errors that are inherent to calculation with real numbers. Instead, the sides are taken as vectors. If the cross-product between sides (3.2) falls below a tolerance, the sides are said to be parallel.

$$\vec{S}_{k_1,i} \times \vec{S}_{k_2,j} \quad (3.2)$$

If sides are found to be parallel, then the projection of the endpoints of one side onto the linear extension of the other side is found. If the difference between the endpoint and the projection of the endpoint falls below a certain tolerance, the sides are said to be co-linear. Finally, co-linear points can be parametrized, so that one may find the intersection of the two line segments. If the intersection is some open set, the triangles are said to be adjacent.

Computing  $F_i$  and  $\mathbf{x}_i$  in the form presented (1.1-1.2) amounts to integrating over a (potentially non-convex) polygon. In practice, it is more computationally straightforward to triangularize the polygons  $P_i$ , compute  $F_i$  and  $\mathbf{x}_i$  for each triangle, and recombine the information to find the volume fraction and centroid of the polygonal element. There exists an analytical form for the volume and centroid of a triangle. This can be used to compute the integral over the entire polygonal element.

Suppose a polygon  $P_i$  is defined as the union of a set of  $N$  triangles (3.3).

$$P_i = \cup_{j=1}^N T_j \quad (3.3)$$

For each triangle, compute volume  $V_{T_j}$  and centroid  $\mathbf{x}_{T_j}$  as in (3.4 - 3.5). Let triangle  $T_j$  have vertices  $\mathbf{v}_{j,1}$ ,  $\mathbf{v}_{j,2}$ , and  $\mathbf{v}_{j,3}$ .

$$\mathbf{x}_{T_j} = \frac{\mathbf{v}_{j,1} + \mathbf{v}_{j,2} + \mathbf{v}_{j,3}}{3} \quad (3.4)$$

$$V_{T_j} = \frac{1}{2}|(\mathbf{v}_{j,2} - \mathbf{v}_{j,1}) \times (\mathbf{v}_{j,3} - \mathbf{v}_{j,1})| \quad (3.5)$$

Then, it is straightforward to compute the integral in (1.2) over the triangle.

$$\int_{T_j} \mathbf{x} \cdot d\mathbf{x} = V_{T_j} \mathbf{x}_{T_j} \quad (3.6)$$

Once centroids and volumes have been computed for all triangles  $T_j$ , the volume  $V_{P_i}$  and centroid  $\mathbf{x}_{P_i}$  of the polygon can be easily calculated.

$$V_{P_i} = \int_{P_i} d\mathbf{x} = \sum_{j=1}^N V_{T_j} \quad (3.7)$$

$$\mathbf{x}_{P_i} = \frac{\int_{P_i} \mathbf{x} \cdot d\mathbf{x}}{|P_i|} = \frac{\sum_{j=1}^N V_{T_j} \mathbf{x}_{T_j}}{V_{P_i}} \quad (3.8)$$

This procedure is then used to find the volume fraction (3.10) and centroid (3.11) for a conglomerate  $C_i$ , defined as (3.9), in cell  $\Omega$ .

$$C_i = \cup_{j=1}^M P_j \quad (3.9)$$

$$F_{C_i} = \frac{\sum_{j=1}^M V_{P_j}}{|\Omega|} \quad (3.10)$$

$$\mathbf{x}_{C_i} = \frac{\sum_{j=1}^M V_{P_j} \mathbf{x}_{P_j}}{F_{C_i} |\Omega|} \quad (3.11)$$

## 3.2 Conglomerate Configuration

Once all conglomerates have been found, several situations can exist.

1. There is only one conglomerate, i.e. the cell is a pure cell with only one material.
2. There are two conglomerates with different material ID's. In this case, the algorithm reverts to two material MOF interface reconstruction.
3. There are multiple conglomerates with different material ID's. In this case, a decision about interface topology must be made.

In the case of a pure cell, there is nothing to be done in terms of either conglomeration or interface reconstruction because no interface is present. This can be identified by preprocessing during the conglomeration routine to boost performance. If all polygonal elements have the same material ID, perform no conglomeration; write the input ID's to the output and continue. In the case of one conglomerate of each material, the cell is cut by a single interface. This is the situation for which Moment of Fluid Interface Reconstruction was designed. The conglomeration algorithm will perform the grouping, discover that only one conglomerate of each material exists, then write the material ID's to output. The interface reconstruction will be identical to MOF because no fictitious material ID's are introduced. This is computationally less efficient because more work will be expended with no gain in accuracy.

In the final case, multiple conglomerates of a given material exist, meaning a filament may form. Let us define a filament as a material region that separates two or more like-materials such that they are not adjacent. We will denote the material that is separated by the filament as the "bulk" fluid. Note that the labels filament and bulk make no assumption of relative size of the regions (Fig. 5).

If a cell is cut by a filament with interfaces of zero curvature, we wish to be able to recover the material configuration exactly. With Multimaterial Moment of Fluid, it is possible to capture the interfaces within the tolerance of volume and centroid error in the reconstruction process. By assigning a fictitious third material ID to one of the bulk regions, the procedure of nested dissection [4] can be used to recover the two interfaces. As with a true multimaterial case, it is necessary to choose the ordering that minimizes reconstruction error. If the filament material is selected first for reconstruction, the resulting material configuration may differ significantly from the true material configuration. While no *a priori* form exists for the optimal ordering for reconstruction, the conglomerate with the centroid farthest from the cell center is less likely to be the filament.

### 3.3 Disconnected Conglomerates

A common occurrence in numerical simulations of deforming interfaces is the formation of “flotsam,” small regions of material in a cell of a different material [9]. Flotsam can separate from an interface and merge at a later time. Suppose a piece of flotsam of Material 1 enters a cell that contains both Material 1 and Material 2, and the flotsam does not touch the boundary of the cell. This case cannot be captured exactly with either one interface (as with MOF) or with two interfaces (using filament machinery). In this scenario, the location and orientation of the interface will be affected proportionally by the mass and location of the flotsam. Because a filament (as defined in Section 3) does not exist if a piece of flotsam is interior to the cell, we conglomerate the flotsam and the larger material and perform the reconstruction. The Moment of Fluid reconstruction will find the interface in the cell that exactly captures the volume of the material and minimizes centroid error [5].

### 3.4 General Conglomeration Algorithm

Consider now the general case, with multiple conglomerates of two materials. We seek to identify “optimal” groupings in the sense that the conglomerates formed can best be represented by one or two interfaces. The full conglomeration algorithm is presented.

1. Form all Material 1 and 2 conglomerates: 1, 2, 1', 2', 1'', 1''', 2''', etc.
2. Choose at most the two largest conglomerates of each material that are touching a cell edge. These are the bulk candidates. If one material has no conglomerates adjacent to the cell edge (i.e. all conglomerates are interior to the cell), use the standard MOF reconstruction and skip to the final step.
3. Attach all remaining conglomerates to the ‘nearest’ like-material bulk candidate.
4. Introduce fictitious material ID’s to account for any material with more than one conglomerate. Up to 4 materials may exist in a cell: 1, 2, 1', 2'.
5. If two conglomerates of each material exist, then two separate interface reconstructions must be performed; continue to the next step. Otherwise, perform an interface reconstruction as in Multimaterial MOF and skip to the final step.
6. Perform an interface reconstruction assuming that Material 1 is the bulk material and Material 2 is the filament. Perform another reconstruction assuming that Material 2 is the bulk material and Material 1 is the filament.
7. Accept the interface reconstruction that minimizes centroid error.

For work presented in this report, assigning remaining conglomerates to the ‘nearest’ bulk candidate was performed based on distances between conglomerate centroids, as computed in Eqn. 3.11. The fictitious fluid labels allow for differentiation between separated conglomerates of the same material. Multimaterial MOF will see them as different fluids and perform the dissection and reconstruction procedure accordingly.

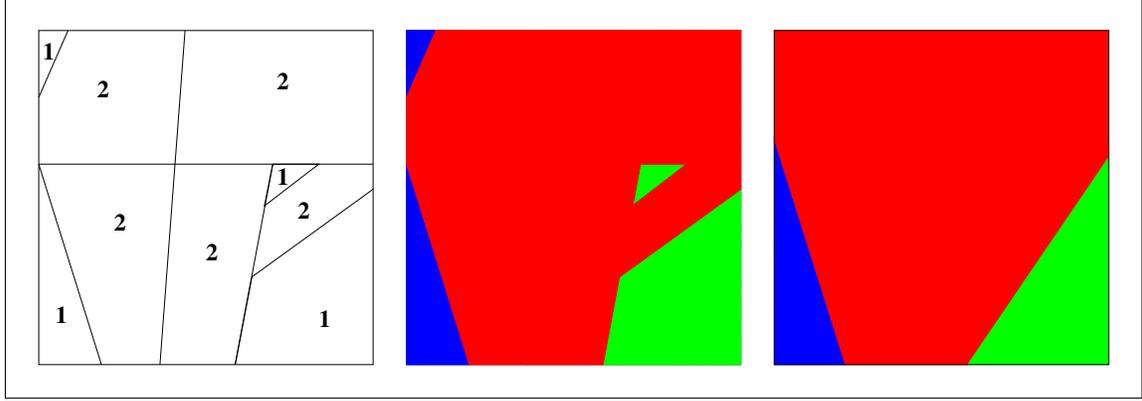


Figure 5: Left: Cell after advection. Polygons are labeled with material ID's. Center: Colors denote grouping of material conglomerates according to the general conglomeration algorithm. Right: Potential reconstruction of material configurations. Red material is a filament, separating the blue and green bulk materials.

This is different from the procedure of standard MOF. During computation of volumes and moments of the materials in the cell, only the material ID is taken into account when grouping polygonal elements. With no respect to the adjacency of like-material elements in a cell, all polygonal elements of the same material will be grouped into a single conglomerate, to use the language of this paper. In a two fluid simulation, this will result in two conglomerates in all mixed cells, regardless of the topology of the material configuration.

If a filament forms and generates a new, fictitious material ID, this should not prevent material from reforming at a later time step to a single material, eliminating the fictitious material ID. In addition and prior to the detection of pure cells, a preprocessing step should be carried out. All secondary, fictitious material ID's should be replaced by the primary material ID before the conglomeration procedure begins.

1. Begin preprocessing.
2. Assign any material with fictitious ID 1' an ID of 1.
3. Assign any material with fictitious ID 2' an ID of 2.
4. Scan all polygonal elements. If all elements have the same material ID, denote the cell 'pure.'
5. If the cell is pure, do not perform conglomeration. Write element ID's to output and proceed to the next cell.
6. If the cell is not pure, perform conglomeration and interface reconstruction.

Grouping should not differentiate between a material and its counterpart that was assigned a fictitious label at a previous time step. Physically, they represent the same material, and they should be conglomerated if adjacent.

Consider the more severe case of disconnected conglomerates, where one or more conglomerates of Material 1 exists in a cell and none of the Material 1 conglomerates is adjacent to the boundary of the cell. Material 2 makes up the remainder of the cell. This case is indicative of flotsam. It is possible that the flotsam may occupy a large portion of the cell or nearly form a filament, but no attempt is made to detect this situation. The proposed algorithm is designed to resolve certain sub-cell features, but this case of flotsam does not fall under this set. The cell is not necessarily segregated into distinct regions that are well-modeled by linear interfaces. For these reasons, this case is handled by using a standard MOF reconstruction. After

performing the preprocessing step, material ID’s are written to output and interface reconstruction is carried out. This amounts to grouping all like-material polygonal elements in a single conglomerate.

Other techniques for resolving such sub cell features may include mesh refinement; quantification of orientation and closeness to the boundary to attempt a fit to a filament; or tracking methods that do not perform a reconstruction in the under-resolved cell in the hope that at the next iteration it is more obvious what should be done. These techniques are beyond the scope of this report.

## 4 Adaptive Mesh Refinement with Filaments

### 4.1 Refinement Criteria

Adaptive Mesh Refinement (AMR) is a dynamic technique for increasing the resolution in cells where some feature is under-resolved. Motivations for introducing mesh refinement may include small material volume fractions, MOF centroid error, interface curvature, and change in topology [1]. The development of filaments can lead to cells with small material volumes that are captured adequately, so refinement based solely on volume fractions may be undesirable. Instead, refinement is made based on error in the MOF centroid reconstruction, as in Ahn and Shashkov [1]. The difference between the reference and actual centroid can be viewed as a means of detecting curvature and change in topology of the interface. Define MOF error as in (4.1).

$$E_{MOF} = \frac{1}{\min(\partial\Omega_i)} \sqrt{\sum_i F_i^2 \|\mathbf{x}_{ref,i} - \mathbf{x}_{act,i}\|^2} \quad (4.1)$$

Value  $\min(\partial\Omega)$  is taken as the minimum side length of cell  $\Omega$ . The sum is taken over all material indices. The volume fraction of material  $i$  is denoted  $F_i$ . The reference and actual centroids of material  $i$  are denoted  $\mathbf{x}_{ref,i}$  and  $\mathbf{x}_{act,i}$  respectively. Once the interface has been reconstructed, compute  $E_{MOF}$ . If  $E_{MOF}$  exceeds some tolerance  $\epsilon$ , mesh refinement is performed. Mesh refinement is also performed in a buffer region around the cell.

### 4.2 Regridding

Regridding is done after every iteration. This gives the mesh a chance to coarsen in the event that an under-resolved feature and its buffer region have left the cell. The regridding procedure is as follows.

1. Perform reconstruction at the coarsest mesh level.
2. Check the MOF error (4.1). Tag cells whose error exceeds a threshold. Tag cells within a ‘buffer’ distance of the tagged cell as well.
3. Generate the next finer level. Copy data from previous finer level meshes where two ‘finer’ levels overlap. Otherwise, interpolate data from the coarser level.
4. Go back to step 2 until maximum refinement level is reached.

## 5 Error Formulation

An analytical solution is not available at intermediate times for all test problems. To approximate the error for passive-advection, deforming-interface problems, a set of evenly spaced points is initialized on the  $T = 0$  interface. The points are advected according to the flow field using Fourth Order Runge-Kutta. At the stopping time, these points represent the zero valued contour of the signed distance, or “level set,” function. Note that segments between points have no relation to the underlying grid. To find the location of the interface in a cell, it must be possible to evaluate the level set function at an arbitrary point. One could construct the level set function at a discrete set of nodes and perform extrapolation to evaluate the function

elsewhere, but this will introduce error based on the method of extrapolation. Rather, a method similar to [6] is used. Lagrangian surface data is used to determine distances on the background Eulerian grid.

First, divide the interior of the deformed interface into triangles. Next, using the mesh resolution of the calculation, tag any cells that are “close” to a given triangle. A number of cells surrounding the triangle should be tagged so that points contained in nearby cells will include the triangle in determining the distance function. See Figure 6 for an example of the tagging procedure. A buffer region of 3 cells was used in testing. The size of this buffer region effectively determines the resolution of the level set function near the surface. A routine to determine if the edge of a triangle is interior or exterior to the deformed figure must be available, as well as a routine to determine if a point is interior to a triangle. The distance function is then computed as follows.

1. Given a point  $P$ , determine in which grid cell it is located.
2. Determine all triangles  $T_i$  that lie near the cell.
3. For triangle  $T$ , calculate the shortest distance from  $P$  to any exterior edge. Project the point onto the extension of the line segment. If the projection lies on the segment, this is the shortest distance associated with point  $P$  and triangle  $T$ . If the projection does not lie on the line segment, then calculate the distance from  $P$  to the endpoints of the segment.
4. Take the minimal such distance over all triangles  $T_i$  for point  $P$ .
5. Determine if  $P$  lies interior to any triangle. If so, take the sign of the distance to be positive. Otherwise, take the sign to be negative.
6. If  $P$  lies in a cell not tagged to be ‘near’ any triangle, then take the distance to be a fixed negative number associated with the size of the buffer.

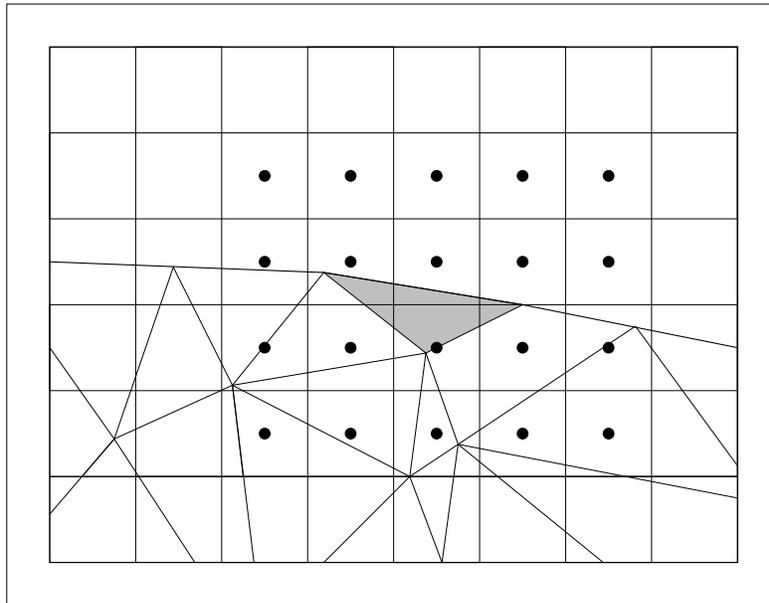


Figure 6: Cells in the buffer region of the shaded triangle are tagged with a black dot. Here, a buffer of 1 cell is shown. The region is taken as a bounding box defined by the minimal and maximal cell indices of the vertices of the shaded triangle.

The level set function can be used to compute the symmetric difference error of the computed solution, defined as in (5.1). The level set function is sampled at cell corners. If the sign of the level set is not the same at all corners, the cell is cut by a material interface. Any cell that is cut is subdivided, and the corners of each subcell are test. This process can be repeated to arbitrary resolution. In this fashion, the level set function is used to locate regions of discrepancy between the computed and exact solution. Quadrature is used to find the magnitude of these regions. Here,  $\Omega_C$  and  $\Omega_E$  represent the computed and exact solutions for material configurations.

$$E_{SD} = |(\Omega_E \cup \Omega_C) / (\Omega_E \cap \Omega_C)| \quad (5.1)$$

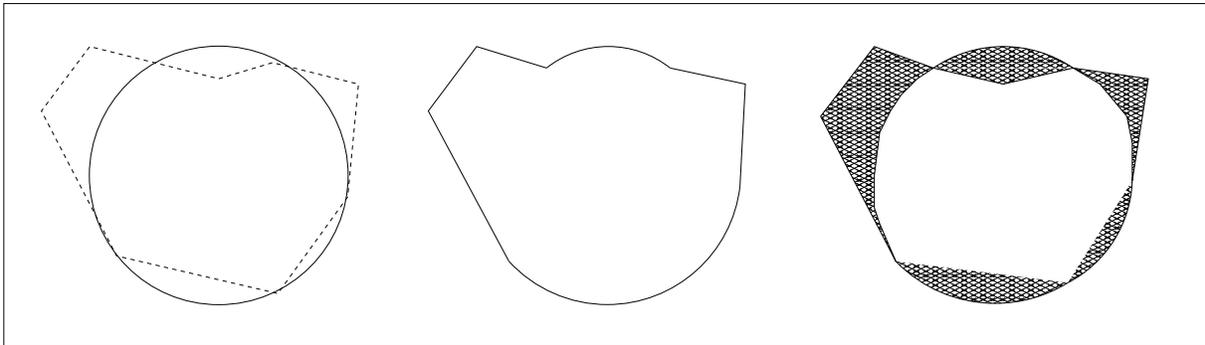


Figure 7: Left: Exact solution (solid) and computed solution (dashed). Center: Union of exact and computed material configurations. Right: Intersection of exact and computed solutions shown in white. Error regions shaded.

## 6 Test Problems

### 6.1 Finite Filament

A filament of length 0.5, width 0.02 is initialized in the unit square domain,  $[0, 1] \times [0, 1]$  with center at  $(0.5, 0.5)$ . The filament is advected such that at time  $T = 1.0$ , the filament has returned to its initial position. The filament is advected horizontally with the top and bottom of the filament conforming to grid lines. Error for the test (Table 1) using filament machinery is on the order of the tolerance in the MOF optimization procedure, as expected.

	AMR Level	Error
Filament MOF	0	$1.481 \times 10^{-8}$
	1	$1.481 \times 10^{-8}$
	2	$1.481 \times 10^{-8}$
Standard MOF	0	$4.969 \times 10^{-2}$
	1	$4.969 \times 10^{-2}$
	2	$4.969 \times 10^{-2}$

Table 1: Results for the finite filament test with horizontal advection. Starting mesh resolution is  $32^2$ . Refinement tolerance is  $\epsilon = 0.0003$ .

A filament of length 0.5, width 0.02 is initialized in the unit square domain,  $[0, 1] \times [0, 1]$  with center at  $(0.51562, 0.5)$ . The filament is advected vertically in a uniform velocity field such that at time  $T = 1.0$ , analytically, the filament has returned to its initial position. The top and bottom of the filament conform with

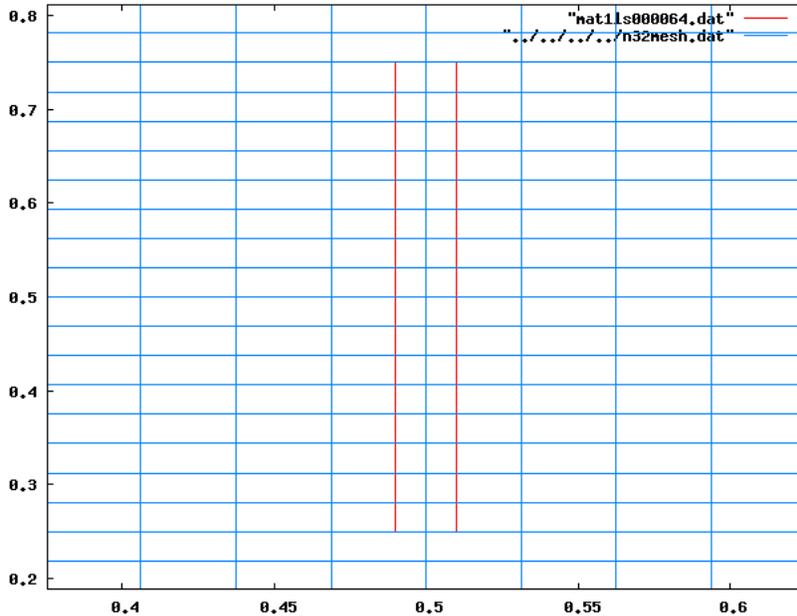


Figure 8: Horizontally advected filament with Filament MOF,  $32^2$  resolution. Results shown after 64 iterations, with structure returning to the starting position.

grid lines, but the structure is initialized slightly right of center in a cell. Error for the test (Table 2) using filament machinery is more severe than the horizontally advected case. For both tests,  $CFL = \frac{|\mathbf{u}|\Delta t}{\Delta x} = 0.5$ , meaning the structure is advected one half of a grid length at each time step on the uniform grid. In the horizontal case, due to structural conformity to grid lines at the top and bottom, the structure was captured to numerical precision at every iteration. This is not the case with vertical advection. The first time step moves the structure such that the top and bottom no longer conform to grid lines. Without additional refinement, this situation cannot be resolved exactly.

	AMR Level	Error
Filament MOF	0	$1.108 \times 10^{-3}$
	1	$4.065 \times 10^{-4}$
	2	$5.498 \times 10^{-5}$
Standard MOF	0	$1.100 \times 10^{-2}$
	1	$1.100 \times 10^{-2}$
	2	$1.100 \times 10^{-2}$

Table 2: Results for the finite filament test with vertical advection. Starting mesh resolution is  $32^2$ . Refinement tolerance is  $\epsilon = 0.0003$ . It appears that error is being dominated by the error in step 1, when the filament tip fills the bottom of the cell. It may be preferable to have a smaller tolerance, but there are already few filament cells.

Lack of convergence under grid refinement for both advection tests in the case of Standard MOF indicates an error in the testing. To properly initialize each test for Standard MOF, three materials must be used. However, at all future steps only two materials may exist. Additionally, during vertical advection using filament machinery, one would expect convergence to the error tolerance, as in horizontal advection, at one level of mesh refinement. This is due to the fact that at  $CFL = 1/2$ , odd advection steps will conform with

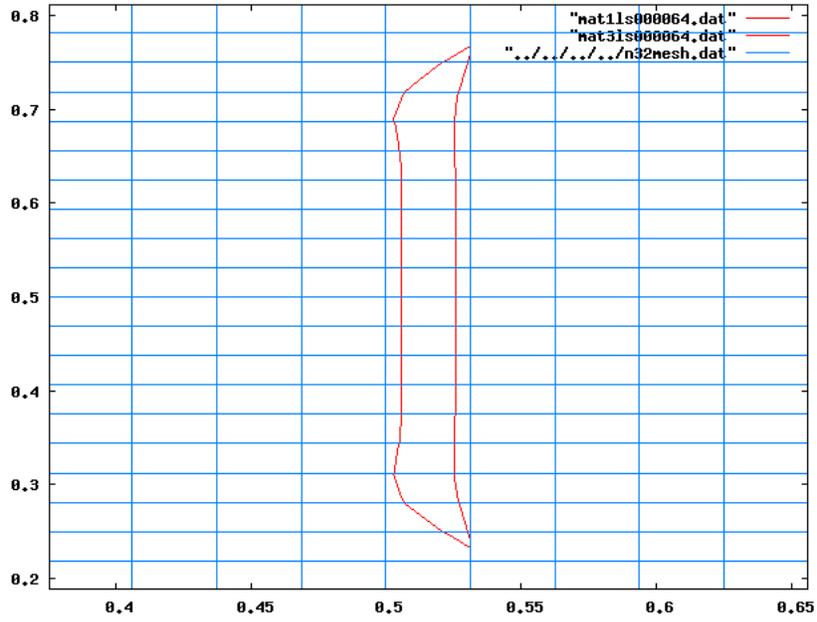


Figure 9: Vertically advected filament with Filament MOF,  $32^2$  resolution. Results shown after 128 iterations, with structure returning to the starting position.

AMR level 1 grid lines. This indicates an unexpected interaction between filaments and the AMR procedure, regridding process, or advection. More investigation must be done.

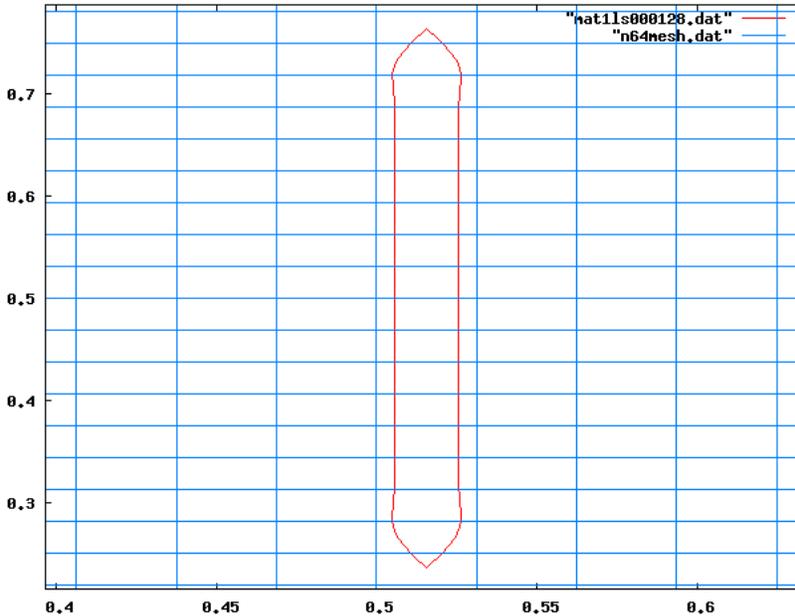


Figure 10: Vertically advected filament with Filament MOF,  $32^2$  resolution. Results shown after 64 iterations, with structure returning to the starting position.

## 6.2 Reversible Vortex

The reversible vortex problem was introduced by Rider and Kothe [8]. The initial material configuration is defined by a circle of radius  $R = 0.15$  with center  $(0.50, 0.75)$  located within the unit square domain  $[0, 1] \times [0, 1]$ . The circular region is deformed by a non-linear, unsteady velocity field given by stream function (6.1).

$$\psi(x, y, t) = \frac{1}{\pi} \sin^2(\pi x) \sin^2(\pi y) \cos\left(\frac{\pi t}{T}\right) \quad (6.1)$$

Parameter  $T$  defines the time for the full period of the flow field, with larger  $T$  leading to more deformation. For the long period reversible vortex, we take  $T = 8$ . Time  $t = T$  would see the analytical solution returning to the initial material configuration. Time  $t = \frac{T}{2}$  defines the time of maximum material deformation.

Error results at full reversal (Table 3) show that the coarsest level of Filament MOF is comparable to Standard MOF with an effective mesh resolution 4 times finer. Visual inspection of the results in Fig. 12 - 13 shows the discrepancy between the interface reconstruction at the same effective mesh resolution. At the same effective mesh resolution, Filament MOF shows a factor of 3 – 5 improvement over Standard MOF.

Error at maximum deformation is examined in Table 4. The analytical solution to the flow field is not available at maximum deformation,  $t = \frac{T}{2}$ , so the procedure in Section 5 is used to approximate an exact solution (Fig. 11). While the the solution for Standard MOF shows break-up in the thin regions near the tail not exhibited in Filament MOF (Fig. 16 - 17), the difference in symmetric difference error for shows only a factor of 2 – 3 improvement when using filaments.

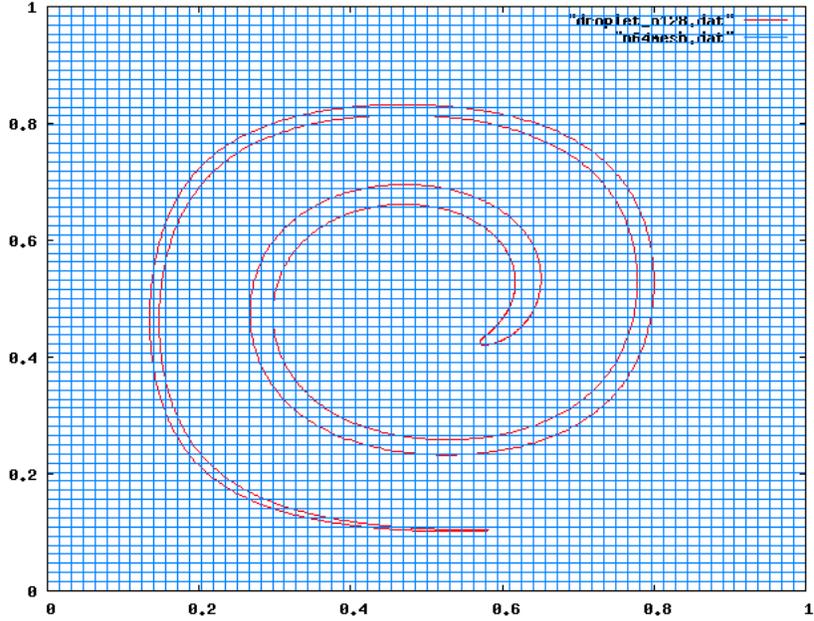


Figure 11: ‘Exact’ solution for the reversible vortex problem, computed as in Section 5, with 2000 points initialized on the circle boundary and time step of  $\Delta t = \frac{1}{3200}$  using RK4. This is a factor of 16 smaller than the smallest time step used. Note the fine, subcell structure at the tail.

	AMR Level	Error	Runtime (sec)
Filament MOF	0	$2.908 \times 10^{-3}$	207.850
	1	$1.098 \times 10^{-3}$	657.587
	2	$8.582 \times 10^{-4}$	1690.734
Standard MOF	0	$1.569 \times 10^{-2}$	183.140
	1	$9.926 \times 10^{-3}$	672.626
	2	$2.871 \times 10^{-3}$	1975.072

Table 3: Results for the long-period deforming vortex at  $t = 8$ . Starting mesh resolution is  $32^2$ . Refinement tolerance is  $\epsilon = 0.0003$ .

	AMR Level	Error
Filament MOF	0	$1.453 \times 10^{-2}$
	1	$6.980 \times 10^{-3}$
	2	$4.923 \times 10^{-3}$
Standard MOF	0	$7.036 \times 10^{-2}$
	1	$1.382 \times 10^{-2}$
	2	$7.381 \times 10^{-3}$

Table 4: Results for the long-period deforming vortex at  $t = 4$ . Starting mesh resolution is  $32^2$ . Refinement tolerance is  $\epsilon = 0.0003$ .

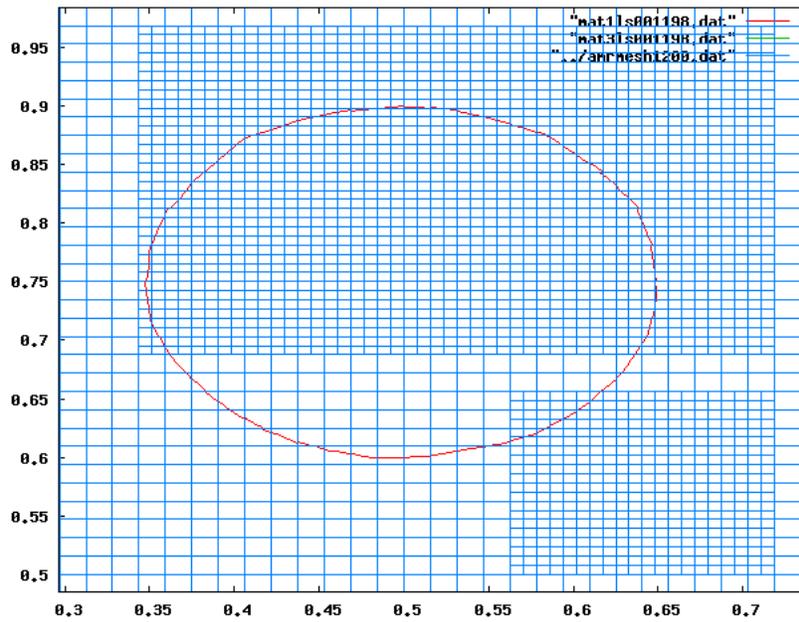


Figure 12: Reversible vortex after full-reversal with 2 levels of AMR using filaments. Base resolution is  $32^2$ .

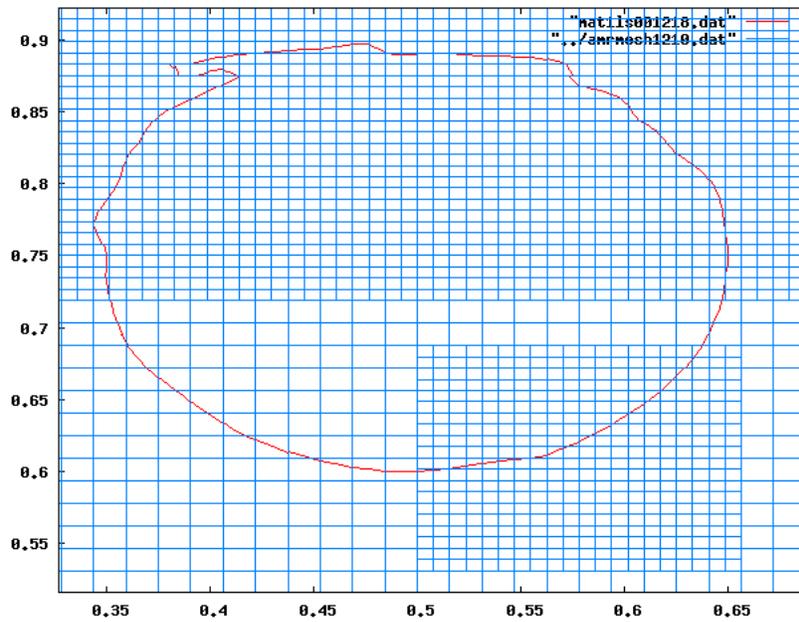


Figure 13: Reversible vortex after full-reversal with 2 levels of AMR without filaments. Base resolution is  $32^2$ . Note the manifestation of breakup at maximum deformation in the form of material reattached at the northwest edge of the structure.

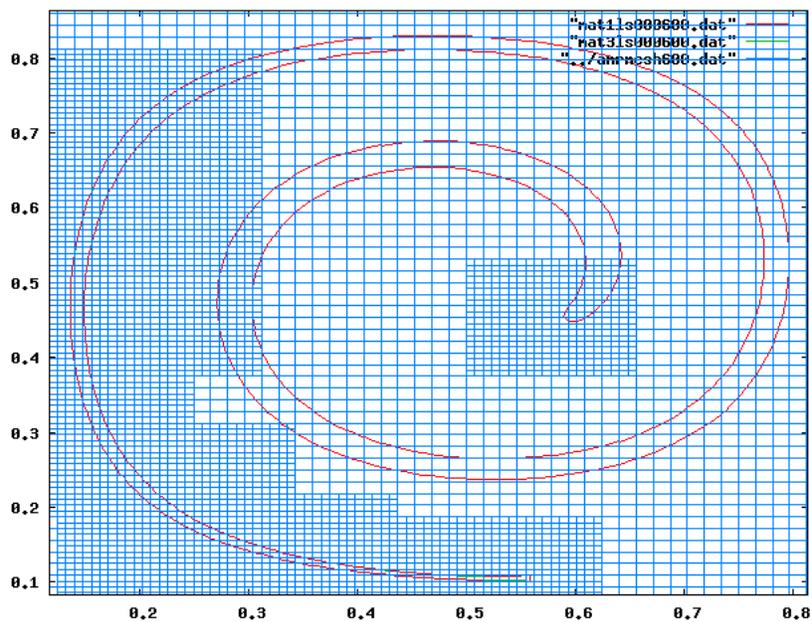


Figure 14: Reversible vortex at maximum deformation with 2 levels of AMR using filaments. Base resolution is  $32^2$ .

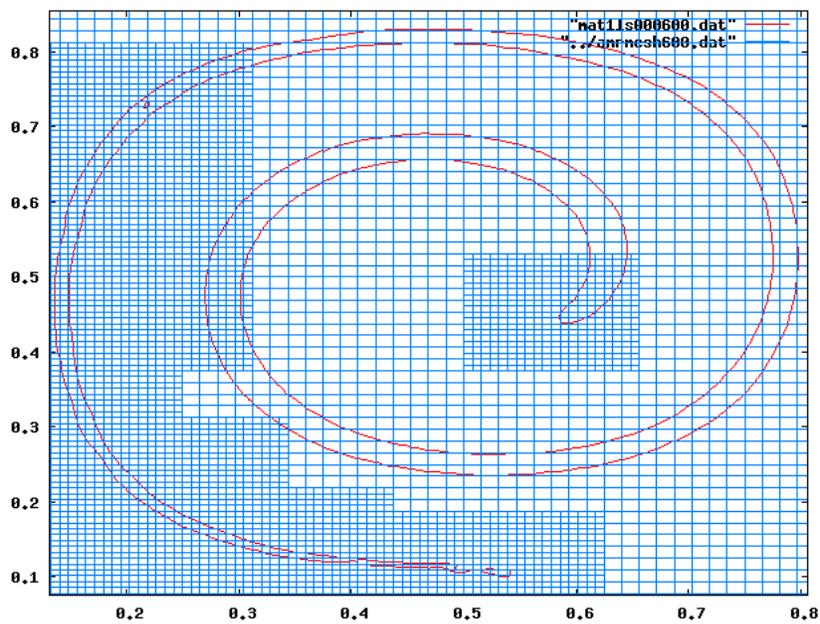


Figure 15: Reversible vortex at maximum deformation with 2 levels of AMR without filaments. Base resolution is  $32^2$ .

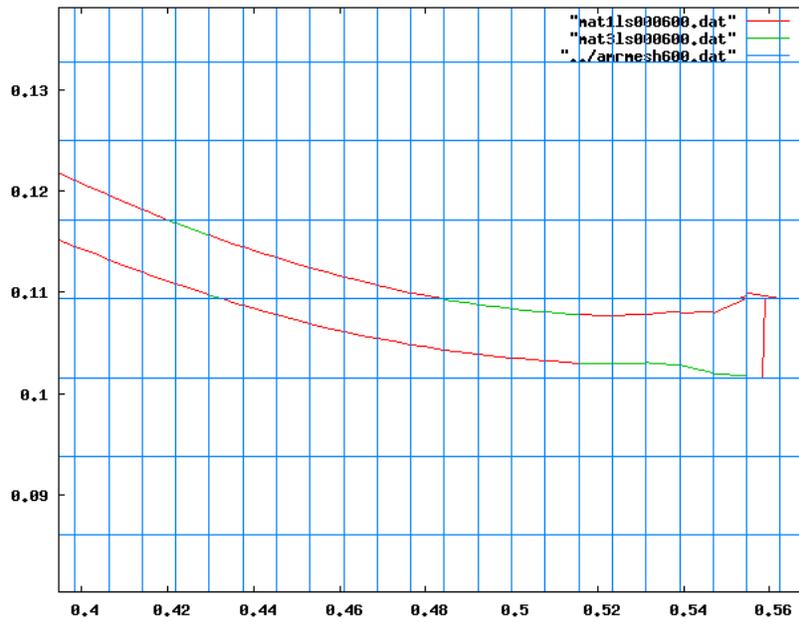


Figure 16: Reversible vortex at maximum deformation with 2 levels of AMR using filaments. Base resolution is  $32^2$ . Note the resolution of the filament structures at the tail.

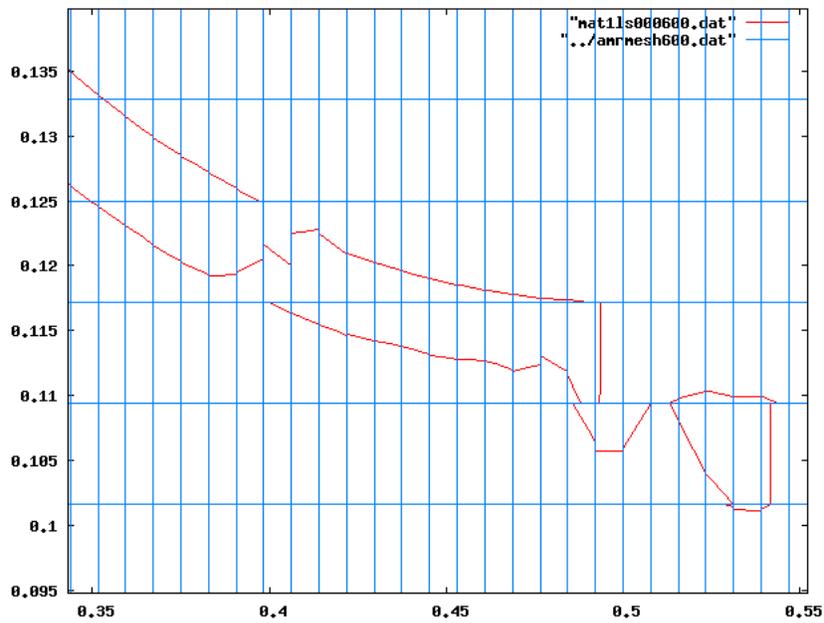


Figure 17: Reversible vortex at maximum deformation with 2 levels of AMR without filaments. Base resolution is  $32^2$ . Note the breakup of the fine structure in the tail due to the standard MOF algorithm.

### 6.3 Droplet Flow

The droplet flow test case was introduced in [1]. The flow is defined by a steady, divergence-free, non-linear velocity field. The initial material configuration of a circle with radius  $R = 0.125$  centered at  $(0.50, 0.50)$  in the unit square  $[0, 1] \times [0, 1]$  develops two sharp edges that extend over time (Fig. 18).

The edges present problems similar to the vertically advected filament at the tip of the reversible vortex. When the tip of a filament enters a cell but does not span the cell, only two conglomerates are formed: the tip of the filament material and the background material. This is a situation that is not well-resolved by a single interface. A second interface could improve accuracy in principle, but it would require partitioning the background material into adjacent conglomerates such that it is amenable to linear interface reconstruction. Alternatively, tracking the material and delaying the reconstruction, as mentioned in the discussion of flotsam (Section 3.4) is possible. For the results presented, AMR was used as a means to attempt to resolve the interface near the sharp tips.

$$\mathbf{v} = \left[ \begin{array}{c} \frac{1}{8}(8x - 4) \\ \frac{1}{8}(- (8y - 4) - 4 - (1 - (8x - 4)^2 - (8x - 4)^4)) \end{array} \right] \quad (6.2)$$

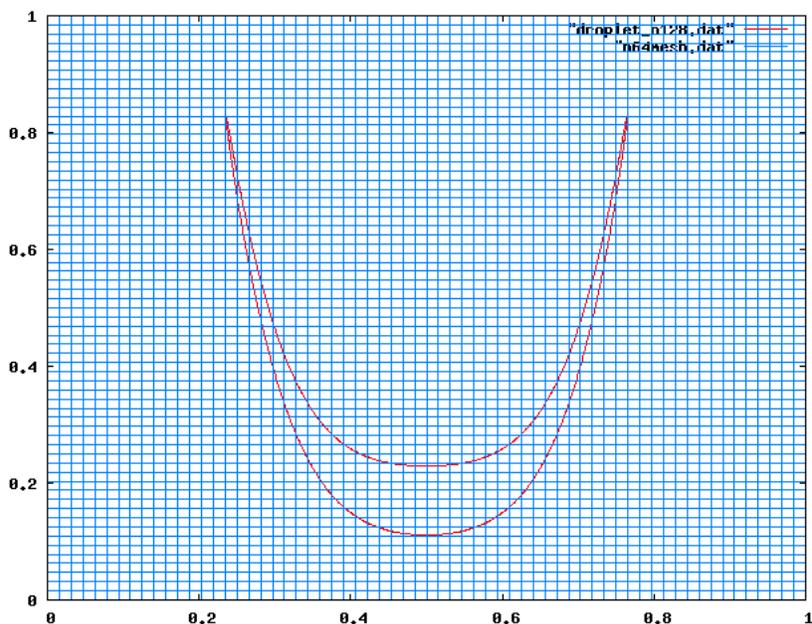


Figure 18: ‘Exact’ solution for the droplet problem, computed as in Section 5, with 2000 points initialized on the circle boundary and time step of  $\Delta t = \frac{1}{3200}$  using RK4. Note the subcell structure at the tips.

The errors presented in Table 5 shows a factor of 3 improvement between Filament and Standard MOF for the two coarsest mesh resolutions. However, errors at the finest mesh resolution are comparable between the two methods. This indicates that the tips are increasingly well-resolved, and few filaments are being formed. This highlights the fact that the filament machinery is an acceleration technique that is most cost-effective when features are under-resolved. As the mesh is refined, results between Filament MOF and Standard MOF should converge. Figures 19 - 20 show results for 2 levels of AMR. Break-up in the thin structures near the tip of the deforming droplet when filaments are not used. However, material topology is not explicitly taken into account in the error metric, so errors are comparable.

	AMR Level	Error	Runtime (sec)
Filament MOF	0	$7.059 \times 10^{-3}$	1578.734
	1	$3.158 \times 10^{-3}$	3129.463
	2	$2.217 \times 10^{-3}$	5285.594
Standard MOF	0	$1.479 \times 10^{-2}$	1728.755
	1	$9.143 \times 10^{-3}$	3375.182
	2	$2.679 \times 10^{-3}$	5024.604

Table 5: Results for the deforming droplet at  $t = 0.75$ . Starting mesh resolution is  $32^2$ . Refinement tolerance is  $\epsilon = 0.0003$ . Note: Runtime is dominated by error testing, due to the recursive nature of the integration procedure (Section 5).

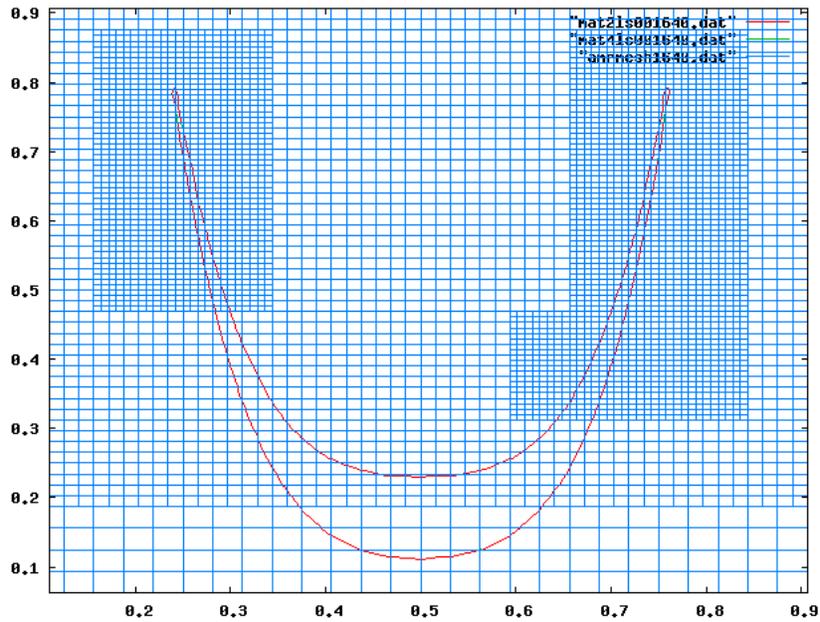


Figure 19: Droplet at time  $t = 0.75$  with 2 levels of AMR and filament capability. Red and green lines denote filament material interfaces.

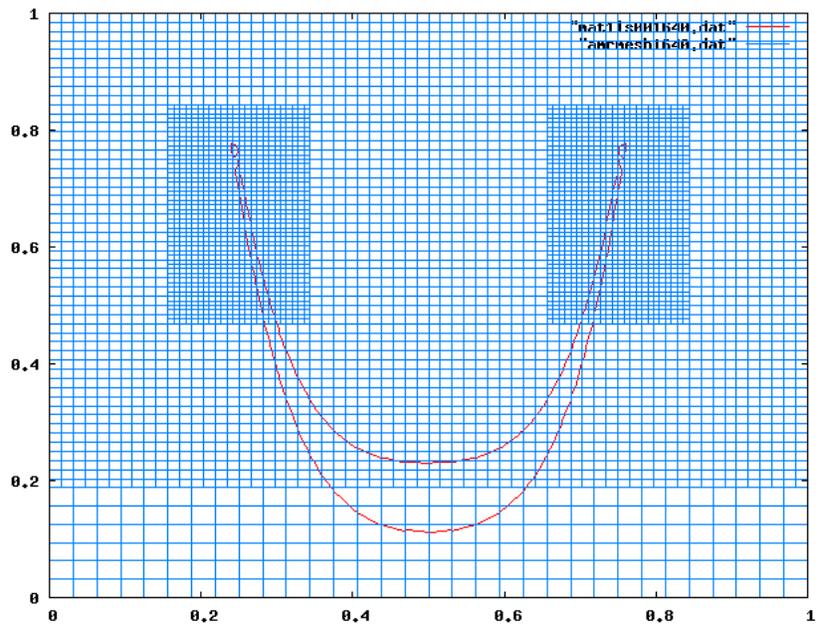


Figure 20: Droplet at time  $t = 0.75$  with 2 levels of AMR and no filament capability. Note the breakup of the structure near the tips, similar to the reversible vortex in Fig. 17.

## 7 Conclusions & Future Work

The use of Multimaterial Moment of Fluid interface reconstruction with filaments allows for acceleration of convergence of the standard Moment of Fluid method. The improvement in accuracy of using filaments vs. using standard MOF with mesh refinements is typically equivalent to 1 – 2 levels of mesh refinement. For meshes of the same effective resolution, this comes at an increase in runtime of approximately 10%, but an increase in accuracy by a factor of 3 – 5 at coarse mesh resolution. The additional cost of forming conglomerates and performing multiple interface reconstructions when 2 or more conglomerates of each material exist is less than the increase seen from allowing additional mesh refinement, which can increase runtime by a factor of 2 – 3. Cases where thin structures enter a cell but do not divide the cell into three or more distinct regions (i.e. a well-resolved filament does not form) are handled no differently than standard MOF and yield no gains in accuracy. Problems for which filaments are well-suited would include problems where large structures are deformed to a point where they are under-resolved and breakup of the structure is undesirable.

These results are encouraging, but it is important to keep in mind that filaments are an acceleration technique for capturing thin, under-resolved interface features. If all features are fully resolved (i.e. no cell is cut by more than one interface), then there will be no gains in accuracy to mitigate the additional computational overhead of the conglomeration procedure. As seen in Section 6.3, the thin structures in the material were captured to similar precision with or without filaments because all features were fairly well-resolved at the highest mesh resolution. Convergence in symmetric difference error of Filament MOF to Standard MOF should be expected as the mesh is refined.

Future work will include incorporation of filament techniques into multi-physics codes that account for buoyancy, surface tension, and transport by mechanisms other than passive advection. The current implementation of filaments is in a block-structured square mesh. However, the technique is not dependent on the mesh. Both MOF and the filament algorithm described in this report are applicable to general, unstructured meshes ???. The technique can also be generalized to  $3D$  by finding the intersection of faces of polyhedra and forming conglomerates in a similar fashion. However, this will be a more computationally intensive procedure than in  $2D$ . More can be done in the case discussed in Section 3.4, when one material is composed entirely of flotsam that is non-adjacent to the boundary. The current implementation reverts to MOF reconstruction when this situation is detected. As stated, there are other techniques that can be considered such as mesh refinement and tracking methods. Detection and error-minimizing resolution of other problematic cases is an active question. The current method for handling cases of 2 or more conglomerates of each material is to form a single filament from one of the materials and accept the configuration with the lowest error. In some cases, standard MOF may produce lower error. Another alternative is to not form a filament and apply Multimaterial Moment of Fluid for the 4 materials. Finally, the generalization of filaments to the true multimaterial case with  $N$  fluids and  $N$  fictitious labels has yet to be implemented.

## Acknowledgments

The author would like to thank his mentor, Dr. Mikhail Shashkov, for providing the direction for this project and the motivation to continue when problems arose; collaborator Dr. Mark Sussman for technical advice on how to proceed; partner Andrew Winters for many fruitful discussions; program coordinator Dr. Scott Runnels for organizing the Computational Physics Summer Workshop; and Los Alamos National Laboratories for providing this wonderful opportunity.

## References

- [1] H. Ahn and M. Shashkov. Adaptive moment-of-fluid method. *J. Comput. Phys.*, 228(8):2792–2821, 2009.
- [2] D. Benson. Volume of fluid interface reconstruction methods for multi-material problems. *Applied Mechanics Reviews*, 55:151–165, 2002.

- [3] V. L. Chenadec and H. Pitsch. A hybrid lagrangian-eulerian method for multiphase flow computation. *Journal of Computational Physics*, preprint, 2012.
- [4] V. Dyadechko and M. Shashkov. Multi-material interface reconstruction from the moment data. Technical Report LA-UR 06-5846, Los Alamos National Laboratory, 2006.
- [5] V. Dyadechko and M. Shashkov. Moment-of-fluid interface reconstruction. Technical Report LA-UR 07-1537, Los Alamos National Laboratory, 2007.
- [6] M. Jemison, E. Loch, M. Sussman, M. Shashkov, M. Arienti, M. Ohta, and Y. Wang. A coupled level set-moment of fluid method for incompressible two-phase flows. *Journal of Scientific Computing*, 2012.
- [7] O. Pironneau. On the transport-diffusion algorithm and its applications to the navier-stokes equations. *Numerische Mathematik*, 32, 1982.
- [8] W. Rider and D. Kothe. Reconstructing volume tracking. *J. Comput. Phys.*, 141:112–152, 1998.
- [9] M. Sussman. A second order coupled levelset and volume of fluid method for computing growth and collapse of vapor bubbles. *Journal of Computational Physics*, 187:110–136, 2003.

# SUPPORT OPERATORS METHOD FOR THE DIFFUSION EQUATION IN MULTIPLE MATERIALS

Andrew Winters<sup>1,2</sup> and Mentor: Mikhail Shashkov<sup>2</sup>

<sup>1</sup>Department of Applied & Computational Mathematics, Florida State University

<sup>2</sup>XCP-4, Methods and Algorithms, Los Alamos National Laboratory

August 14, 2012

---

## Abstract

A second-order finite difference scheme for the solution of the diffusion equation on non-uniform meshes is implemented. The method allows the heat conductivity to be discontinuous. The algorithm is formulated on a one dimensional mesh and is derived using the support operators method. A key component of the derivation is that the discrete analog of the flux operator is constructed to be the negative adjoint of the discrete divergence, in an inner product that is a discrete analog of the continuum inner product. The resultant discrete operators in the fully discretized diffusion equation are symmetric and positive definite. The algorithm is generalized to operate on meshes with cells which have mixed material properties. A mechanism to recover intermediate temperature values in mixed cells using a limited linear reconstruction is introduced. The implementation of the algorithm is verified and the linear reconstruction mechanism is compared to previous results for obtaining new material temperatures.

---

## 1 Introduction

We collaborated with Dr. Mikhail Shashkov on problems in heat diffusion, principally concerned with diffusion processes, linear and non-linear, across a material interface. To describe the heat diffusion process we solve the diffusion equation

$$a \frac{\partial u}{\partial t} = \nabla \cdot (\mathbf{K} \nabla u) + f, \quad (1.1)$$

where  $u$  is the temperature,  $\mathbf{K}$  is the material properties tensor,  $a = c\rho > 0$ , where  $c$  is the heat capacity and  $\rho$  is the density, and  $f$  is a forcing function. The flux vector  $\mathbf{w}$  plays an important role in the numerical scheme to be derived, so it is introduced as

$$\mathbf{w} = -\mathbf{K} \nabla u. \quad (1.2)$$

The flux (1.2) can, and will be, used to write the diffusion equation (1.1) as a first order system in Sec. 2. Applying the method of support operators we obtain a set of discrete equations to

solve the diffusion equation numerically where, for generality, the material properties tensor  $\mathbf{K}$  may be discontinuous.

The support operators method (SOM) takes advantage of the fact that most partial differential equations are formulated in terms of the invariant differential operators divergence  $\nabla \cdot$ , gradient  $\nabla$ , and curl  $\nabla \times$ . The SOM provides an approach for spatial differencing by constructing discrete analogs of the aforementioned differential operators. The discrete operators satisfy discrete versions of important differential and integral identities satisfied by the continuum operators. In essence, the SOM constructs a discrete version of the differential operator calculus. Conservation laws, solution symmetries and adjoint relationships between differential operators are examples of properties we want the discrete operators to mimic. For the linear diffusion problem (1.1) the mimetic discretization mimics

1. The Gauss-Green theorem to enforce the local conservation law
2. The negative adjoint relationship between the flux and divergence operators,  $-\mathbf{K}\nabla = (\nabla \cdot)^*$
3. Guaranteed symmetry and positivity of the product of the discrete divergence and discrete flux, the discrete operator  $\mathcal{D}\mathcal{G}$
4. The null space of the discrete flux operator is the constant functions.

Full details on the mimetic discretization of the diffusion equation are found in, for example, [8, 12].

The construction of a discrete calculus proceeds in two steps. First we choose a discrete form for one of the fundamental operators, termed the *prime* operator. Then, based on some subset of differential and integral identities we choose to maintain, we construct the other fundamental operator(s), termed the *derived* operators. The choice of the prime operator is application and discretization dependent. In a sense, the prime operator ‘supports’ the construction of the derived operators.

The remainder of this report is organized as follows. In Sec. 2 we provide a derivation of a mimetic finite difference (MFD) scheme for the one dimensional diffusion equation (1.1) using the SOM. Sec. 3 gives an overview of extensions to the MFD scheme from Sec. 2 to non-linear diffusion problems. In Sec. 4 we discuss generalizations of the MFD scheme to meshes where cells have mixed material properties. Sec. 5 provides numerical results. Finally, Sec. 6 discusses conclusions and future work.

## 2 Global Support Operators Method

We consider the diffusion equation (1.1) in one spatial dimension

$$a \frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left( k \frac{\partial u}{\partial x} \right) + f, \quad (2.1)$$

with some initial condition  $u = u_0$  at  $t = 0$  and set of boundary conditions. The diffusion equation (2.1) is solved on a one dimensional domain  $\Omega = [x_1, x_N]$ . For generality, we allow the heat conductivity  $k$  to be discontinuous. We use the one dimensional analog of the flux (1.2) to rewrite the diffusion equation (2.1) as a first-order system

$$\begin{cases} a \frac{\partial u}{\partial t} = -\frac{\partial w}{\partial x} + f, \\ w = -k \frac{\partial u}{\partial x}. \end{cases} \quad (2.2)$$

We select general Robin type boundary conditions

$$\beta \left( k \frac{\partial u}{\partial x} \right) \hat{n} + \alpha u = \psi, \quad (2.3)$$

where  $\alpha$ ,  $\beta$ , and  $\psi$  are given smooth functions at the boundary of the one dimensional domain  $\Omega$  and  $\hat{n}$  is the unit outward normal to the boundary. In one dimension we know  $\hat{n} = -1$  at  $x_1$  and  $\hat{n} = 1$  at  $x_N$ . If we represent the boundary conditions (2.3) with the flux  $w$ , choose  $\beta = 1$ , and assume  $\alpha \geq 0$ , we find

$$\begin{cases} w + \alpha u = \psi & \text{at } x = x_1, \\ -w + \alpha u = \psi & \text{at } x = x_N. \end{cases} \quad (2.4)$$

The SOM requires that the fully discrete problem mimic the semi-discrete problem, where time, but not space, has been discretized [12]. We perform our analysis on the fully implicit time discretization of (2.2). First we write the diffusion equation (2.1) and first-order system (2.2) in terms of abstract operators. Next, we examine a few important properties of the abstract operators we wish the discrete operators to mimic. Then, following the SOM, we select a prime operator and use it to support the construction of the derived operator to arrive at a fully discrete equation. For the derivations that follow we adopt the notation that continuum operators are capital bold letters and discrete operators are capital script letters.

## 2.1 The Abstract Operators

The fully implicit semi-discrete diffusion equation and Robin boundary conditions give

$$\begin{cases} a \frac{u^{n+1} - u^n}{\Delta t} - \frac{\partial}{\partial x} \left( k \frac{\partial u^{n+1}}{\partial x} \right) = f, & \text{on } (x_1, x_N), \\ -k \frac{\partial u^{n+1}}{\partial x} + \alpha u^{n+1} = \psi, & \text{at } x = x_1, \\ k \frac{\partial u^{n+1}}{\partial x} + \alpha u^{n+1} = \psi, & \text{at } x = x_N, \end{cases} \quad (2.1.1)$$

where the index  $n$  corresponds to the time level  $t_n = n\Delta t$  and  $u^n = u(x, t_n)$ . We assume that  $a$ ,  $k$ ,  $f$ ,  $\alpha$ , and  $\psi$  are functions of space and time, but, as we are mainly concerned with the spatial discretization, we suppress the time index. We write (2.1.1) in operator form

$$\mathbf{A}u^{n+1} = \mathbf{F}^{n+1}, \quad (2.1.2)$$

where the operator  $\mathbf{A}$  is

$$\mathbf{A}u = \begin{cases} \frac{au}{\Delta t} - \frac{\partial}{\partial x} \left( k \frac{\partial u}{\partial x} \right), & \text{on } (x_1, x_N), \\ -k \frac{\partial u}{\partial x} + \alpha u, & \text{at } x = x_1, \\ k \frac{\partial u}{\partial x} + \alpha u, & \text{at } x = x_N, \end{cases} \quad (2.1.3)$$

and the right hand side of (2.1.2) is

$$\mathbf{F}^{n+1} = \begin{cases} f + \frac{au^n}{\Delta t}, & \text{on } (x_1, x_N), \\ \psi, & \text{at } x = x_1, \\ \psi, & \text{at } x = x_N. \end{cases} \quad (2.1.4)$$

The fully implicit semi-discrete flux form of the diffusion equation and Robin boundary conditions are

$$\begin{cases} a \frac{u^{n+1} - u^n}{\Delta t} + \frac{\partial w^{n+1}}{\partial x} = f, & \text{on } (x_1, x_N), \\ w^{n+1} = -k \frac{\partial u^{n+1}}{\partial x}, & \text{on } (x_1, x_N), \\ w^{n+1} + \alpha u^{n+1} = \psi, & \text{at } x = x_1, \\ -w^{n+1} + \alpha u^{n+1} = \psi, & \text{at } x = x_N. \end{cases} \quad (2.1.5)$$

The semi-discrete equation (2.1.5) is written in the operator form

$$w^{n+1} - \mathbf{G}u^{n+1} = 0, \quad \mathbf{D}w^{n+1} + \mathbf{\Upsilon}u^{n+1} = \mathbf{F}^{n+1}, \quad (2.1.6)$$

where the operators  $\mathbf{G}$ ,  $\mathbf{D}$ , and  $\mathbf{\Upsilon}$  are defined as

$$\begin{aligned} \mathbf{G}u &= -k \frac{\partial u}{\partial x}, & \text{on } (x_1, x_N), \\ \mathbf{D}w &= \begin{cases} \frac{\partial w}{\partial x}, & \text{on } (x_1, x_N), \\ w, & \text{at } x = x_1, \\ -w, & \text{at } x = x_N, \end{cases} \\ \mathbf{\Upsilon}u &= \begin{cases} \frac{au}{\Delta t}, & \text{on } (x_1, x_N), \\ \alpha u, & \text{at } x = x_1, x_N. \end{cases} \end{aligned} \quad (2.1.7)$$

If we eliminate the flux  $w$  from (2.1.6) we relate the operators  $\mathbf{A}$ ,  $\mathbf{G}$ ,  $\mathbf{D}$ , and  $\mathbf{\Upsilon}$  by

$$\mathbf{A} = \mathbf{\Upsilon} + \mathbf{D}\mathbf{G}. \quad (2.1.8)$$

On the other hand, if  $u$  is eliminated from (2.1.6) then we obtain a new operator equation

$$\mathbf{B}w^{n+1} = (\mathbf{I} + \mathbf{G}\mathbf{\Upsilon}^{-1}\mathbf{D})w^{n+1} = \mathbf{G}\mathbf{\Upsilon}^{-1}\mathbf{F}^{n+1}. \quad (2.1.9)$$

The Hilbert space  $H$  is the closure of scalar functions  $u$  that are smooth on  $\Omega$  with the inner product

$$(u, v)_H = \int_{x_1}^{x_N} uv \, dx + uv \Big|_{x_1}^{x_N}, \quad u, v \in H, \quad (2.1.10)$$

and the Hilbert space  $\mathbf{H}$  is the closure of the scalar functions  $w$  that are smooth on  $\Omega$  with the inner product

$$(w, z)_{\mathbf{H}} = \int_{x_1}^{x_N} wk^{-1}z \, dx, \quad w, z \in \mathbf{H}. \quad (2.1.11)$$

The inner product (2.1.11) is weighted by the inverse of the heat conductivity  $k$ . Because  $k$  is uniformly bounded above and below, so is  $k^{-1}$ . Thus,  $(\cdot, \cdot)_{\mathbf{H}}$  is an inner product even when  $k$  is discontinuous.

As we only consider one spatial dimension the functions  $u$  and  $w$  are both scalars. However, in higher dimensions the temperature  $u$  is a scalar and the flux  $\mathbf{w}$  is a vector. The fact that the functions  $u$  and  $w$  are not both scalars in higher dimensions is the motivation for defining two Hilbert spaces  $H$  and  $\mathbf{H}$ . To keep the discussion general, we need to track which functions are analogs of scalars and which are analogs of vectors as well as which derivative operations correspond to the divergence or the gradient. Derivatives which act on the scalar function  $u$  are analogous to the gradient and derivatives which act on  $w$  are analogous to the divergence. For details of a two dimensional SOM derivation see, for example, Shashkov [12].

### 2.1.1 Important Properties of the Abstract Operators

In the SOM we choose a prime operator, a derived operator, an integral identity connecting the prime and derived operator, discretize the prime operator, and discrete analogs of the inner products (2.1.10) and (2.1.11). We select the divergence as the prime operator which is discretized with a one dimensional analog of the divergence theorem (the Fundamental Theorem of Calculus in one dimension)

$$\int_{x_1}^{x_N} \frac{\partial w}{\partial x} dx = w \Big|_{x_1}^{x_N}. \quad (2.1.12)$$

We select the flux  $\mathbf{G} = -k \frac{\partial}{\partial x}$  as the derived operator. The required integral identity is the Gauss-Green theorem, which in one dimension is just integration by parts

$$\int_{x_1}^{x_N} u \frac{\partial w}{\partial x} dx + \int_{x_1}^{x_N} w \frac{\partial u}{\partial x} dx = uw \Big|_{x_1}^{x_N} \quad (2.1.13)$$

If we multiply the integrand of the second integral in (2.1.13) by  $k^{-1}k$  and rearrange terms we find

$$\int_{x_1}^{x_N} u \frac{\partial w}{\partial x} dx - uw \Big|_{x_1}^{x_N} = - \int_{x_1}^{x_N} wk^{-1}k \frac{\partial u}{\partial x} dx. \quad (2.1.14)$$

From the definition of the operators  $\mathbf{D}$  and  $\mathbf{G}$  in (2.1.7), the inner products (2.1.10) and (2.1.11), and the integral identity (2.1.14) we find

$$(\mathbf{D}w, u)_H = (w, \mathbf{G}u)_H, \quad (2.1.15)$$

which implies that  $\mathbf{G} = \mathbf{D}^*$ . Note the divergence property (2.1.12) is equivalent to

$$(\mathbf{D}w, 1)_H = 0, \quad (2.1.16)$$

where 1 is the constant function with value 1. Finally, we assume  $a > 0$  and  $\alpha \geq 0$ , to conclude  $\mathbf{\Upsilon} = \mathbf{\Upsilon}^* \geq 0$ . Further, if we take  $\alpha > 0$ , then  $\mathbf{\Upsilon} > 0$ .

To summarize, the continuum operators have the divergence property (2.1.16) and if  $\alpha > 0$ , then

$$\mathbf{G} = \mathbf{D}^*, \quad \mathbf{\Upsilon} = \mathbf{\Upsilon}^* > 0, \quad (2.1.17)$$

and because

$$\begin{aligned} \mathbf{A} &= \mathbf{\Upsilon} + \mathbf{D}\mathbf{G} = \mathbf{\Upsilon} + \mathbf{D}\mathbf{D}^*, \\ \mathbf{B} &= \mathbf{I} + \mathbf{G}\mathbf{\Upsilon}^{-1}\mathbf{D} = \mathbf{I} + \mathbf{D}^*\mathbf{\Upsilon}^{-1}\mathbf{D}, \end{aligned} \quad (2.1.18)$$

it follows that

$$\mathbf{A} = \mathbf{A}^* > 0, \quad \mathbf{B} = \mathbf{B}^* > 0. \quad (2.1.19)$$

For the operator  $\mathbf{B}$  in (2.1.19) we use the result, if  $\mathbf{\Upsilon}$  is a symmetric positive definite operator, then so is  $\mathbf{\Upsilon}^{-1}$ . The properties (2.1.16) - (2.1.19) are the important properties of the continuum operators that we want the fully discrete operators to mimic. The properties of  $\mathbf{A}$  and  $\mathbf{B}$  follow from the properties of  $\mathbf{\Upsilon}$ ,  $\mathbf{D}$ , and  $\mathbf{G}$  and the definitions of the inner products (2.1.10) and (2.1.11). The goal of the SOM is to build the discrete analogs of the operators  $\mathbf{\Upsilon}$ ,  $\mathbf{D}$ , and  $\mathbf{G}$  and the inner products (2.1.10) and (2.1.11) that satisfy analogs of (2.1.16) and (2.1.17).

## 2.2 A Mimetic Finite Difference Scheme in One Dimension

Before we proceed with the derivation of the numerical scheme we define the notation used for the one dimensional mesh. We consider the domain  $\Omega = [x_1, x_N]$ , which we divide into  $N - 1$  non-overlapping, one dimensional cells. We assume the one dimensional mesh is not uniform. We derive a staggered grid scheme and thus require a convenient notation for cell edge and cell-centered values. We adopt the convention that cell edge values carry integer indices and cell-centered values carry half-integer indices. The cells, the one dimensional area of each cell, denoted by  $h$ , and the heat conductivity in each cell also carry a half-integer index.

### 2.2.1 The Degrees of Freedom

First we define the degrees of freedom for physical variables temperature  $u$  and flux  $w$  in the discretization. For now we assume that each cell is homogeneous but material properties can vary between cells. This assumption is removed and extensions of the MFD scheme to a mesh where a cell may contain a material interface is discussed in Sec. 4. The discrete temperature  $U$  is defined at cell-centers and the endpoints of the domain  $\Omega$  to capture boundary conditions. The discrete flux  $W$  is defined at cell faces.

### 2.2.2 The Discrete Inner Products

The second step of the SOM is to equip the spaces of discrete temperatures and discrete fluxes with discrete inner products. We define discrete analogs of the two continuous inner products (2.1.10) and (2.1.11). Effectively this means we select a quadrature technique on each cell to approximate the integrals in the continuous inner products. We denote the spaces of discrete scalar functions as  $HC$  and  $\mathbf{HC}$ . Because the discrete information for  $U$  is cell-centered, it is natural to select midpoint rule for the first inner product (2.1.10)

$$(U, V)_{HC} = \sum_{i=1}^{N-1} U_{i+\frac{1}{2}} V_{i+\frac{1}{2}} h_{i+\frac{1}{2}} + U_N V_N + U_1 V_1, \quad U, V \in HC. \quad (2.2.1)$$

The discrete inner product (2.2.1) is symmetric,  $(U, V)_{HC} = (V, U)_{HC}$ , and if all the areas  $h_{i+1/2}$  are positive, then  $(U, U)_{HC} \geq 0$  and  $(U, U)_{HC} = 0$  if and only if  $U = 0$ . So the discrete inner product is well-defined. The integral in the continuous inner product (2.1.11) is discretized using trapezoid rule because the discrete information for the fluxes is located at the edges of the cell

$$(W, Z)_{\mathbf{HC}} = \sum_{i=1}^{N-1} \frac{W_{i+1} Z_{i+1} + W_i Z_i}{2k_{i+\frac{1}{2}}} h_{i+\frac{1}{2}}, \quad W, Z \in \mathbf{HC}, \quad (2.2.2)$$

where  $k_{i+1/2}$  is the heat conductivity and  $h_{i+1/2}$  is the area of cell  $i + \frac{1}{2}$ . The discrete inner product (2.2.2) is well defined provided  $k_{i+1/2}$  and  $h_{i+1/2}$  are positive.

### 2.2.3 The Formal Inner Products

For the computation of adjoint relationships and entries of matrices corresponding to the discretized operators, we introduce the formal inner products, denoted by  $[\cdot, \cdot]$ , in the two spaces of discrete scalar functions  $HC$  and  $\mathbf{HC}$ . In the space  $HC$  we define the formal inner product

$$[U, V]_{HC} = \sum_{i=1}^{N-1} U_{i+\frac{1}{2}} V_{i+\frac{1}{2}} + U_N V_N + U_1 V_1, \quad (2.2.3)$$

and similarly in the space  $\mathbf{HC}$  we define the formal inner product

$$[W, Z]_{\mathbf{HC}} = \sum_{i=1}^N W_i Z_i. \quad (2.2.4)$$

We relate the discrete inner products (2.2.1), (2.2.2) and the formal inner products (2.2.3), (2.2.4) by

$$(U, V)_{HC} = [\mathcal{M}U, V]_{HC}, \quad (W, Z)_{\mathbf{HC}} = [\mathcal{S}W, Z]_{\mathbf{HC}}, \quad (2.2.5)$$

where  $\mathcal{M} = \text{diag}(1, h_{3/2}, \dots, h_{N-1/2}, 1)$ . Thus, the matrix  $\mathcal{M}$  is symmetric and positive definite. The discrete inner product  $(\cdot, \cdot)_{\mathbf{HC}}$  (2.2.2) is well defined, so the matrix  $\mathcal{S}$  must be symmetric positive definite as  $[\mathcal{S}W, Z]_{\mathbf{HC}}$  is also a well defined inner product. For the one dimensional case it is straightforward to derive the matrix  $\mathcal{S}$ . However, we delay the derivation until Sec. 2.2.5.

We have the symmetric positive definite operators  $\mathcal{M}$  and  $\mathcal{S}$ . It can be shown that there exist positive constants  $\Xi_i$  and  $\xi_i$ ,  $i = 1, 2$  such that the formal inner products satisfy the bounds

$$\begin{aligned} \xi_1 [U, V]_{HC} &\leq [\mathcal{M}U, V]_{HC} \leq \Xi_1 [U, V]_{HC}, \\ \xi_1 [W, Z]_{\mathbf{HC}} &\leq [\mathcal{S}W, Z]_{\mathbf{HC}} \leq \Xi_2 [W, Z]_{\mathbf{HC}}, \end{aligned} \quad (2.2.6)$$

where the constants  $\Xi_i$  and  $\xi_i$ ,  $i = 1, 2$  depend on the upper and lower bounds of  $k$  and the size of the one dimensional cell.

In higher dimensions and on an arbitrary mesh it is extremely difficult to find such a matrix  $\mathcal{S}$  that the discrete inner product approximates the continuous inner product with sufficient accuracy in the space  $\mathbf{HC}$ , see, for example, [8, 12]. Also, Hyman et. al. showed that the accuracy of mimetic discretizations strongly depends on a choice of the matrix  $\mathcal{S}$  [5].

#### 2.2.4 The Discrete Divergence - the Prime Operator

The third step is to discretize the prime operator, which we selected to be the extended divergence operator  $\mathbf{D}$  in (2.1.7). We call  $\mathbf{D}$  the extended divergence operator because it includes boundary conditions. We denote the discrete divergence by  $\mathcal{D}$ . On the interior of the domain  $\mathcal{D}$  is

$$(\mathcal{D}W)_{i+\frac{1}{2}} = \frac{W_{i+1} - W_i}{h_{i+\frac{1}{2}}}, \quad (2.2.7)$$

for  $i = 1, \dots, N - 1$ . On the boundary of the domain  $\mathcal{D}$  is

$$(\mathcal{D}W)_1 = W_1, \quad (\mathcal{D}W)_N = -W_N. \quad (2.2.8)$$

Now if we use the discrete inner product (2.2.1) in conjunction with the discrete divergence operator  $\mathcal{D}$  we determine from telescoping sums that

$$(\mathcal{D}W, 1)_{HC} = 0, \quad (2.2.9)$$

which is the divergence property of the discrete divergence  $\mathcal{D}$ . The discrete property (2.2.9) is the direct analog of the continuum property (2.1.16) for the operator  $\mathbf{D}$ .

#### 2.2.5 The Discrete Flux - the Derived Operator

The discrete flux operator, denoted  $\mathcal{G}$ , is the derived operator in the SOM. The operator  $\mathcal{G}$  is the discrete analog of the continuum operator  $\mathbf{G} = -k \frac{\partial}{\partial x}$  and is defined by  $\mathcal{G} = \mathcal{D}^*$ . The adjoint is taken with respect to the discrete inner products (2.2.1) and (2.2.2). As will be seen, the

operator  $\mathcal{G}$  can be expressed as the product of a banded and an inverse of a banded matrix, a fact we exploit in the numerical algorithm. We note that on a one dimensional mesh, which is always orthogonal and rectangular, the operator  $\mathcal{G}$  remains banded. In two dimensions for a general nonorthogonal, logically rectangular grid the operator  $\mathcal{G}$  is not banded [12].

From the discrete operator  $\mathcal{D} : \mathbf{HC} \rightarrow HC$  and the definition of adjoint we have

$$(\mathcal{D}W, U)_{HC} = (W, \mathcal{D}^*U)_{\mathbf{HC}}, \quad (2.2.10)$$

which we translate to the formal inner products as

$$[\mathcal{D}W, \mathcal{M}U]_{HC} = [W, \mathcal{S}\mathcal{D}^*U]_{\mathbf{HC}}. \quad (2.2.11)$$

In the translation to the formal inner products (2.2.11) we use the fact that  $\mathcal{M}$  and  $\mathcal{S}$  are symmetric. The symmetricity allows us to swap the location of the operators in the formal inner products without apologizing. We define the formal adjoint  $\mathcal{D}^\dagger$  of  $\mathcal{D}$  to be the adjoint in the formal inner product, so

$$[W, \mathcal{D}^\dagger \mathcal{M}U]_{HC} = [W, \mathcal{S}\mathcal{D}^*U]_{\mathbf{HC}}. \quad (2.2.12)$$

The relationship (2.2.12) must be true for all  $W$  and  $U$ , thus

$$\mathcal{D}^\dagger \mathcal{M} = \mathcal{S}\mathcal{D}^*, \quad (2.2.13)$$

and then the discrete analog of the operator  $\mathbf{G}$  is given as

$$\mathcal{G} = \mathcal{D}^* = \mathcal{S}^{-1}\mathcal{D}^\dagger \mathcal{M}. \quad (2.2.14)$$

In general,  $\mathcal{S}$  is banded and consequently  $\mathcal{S}^{-1}$  is not usually banded, so  $\mathcal{G}$  is not usually banded [12]. That is, the discrete flux  $\mathcal{G}$  has a nonlocal stencil.

The expression for the discrete gradient (2.2.14) contains two unknown operators  $\mathcal{S}$  and  $\mathcal{D}^\dagger$  as well as the known diagonal operator  $\mathcal{M}$ . To determine  $\mathcal{S}$  and  $\mathcal{D}^\dagger \mathcal{M}$  we apply a summation by parts method to the Gauss-Green theorem (2.1.15). We use the discrete inner products (2.2.1), (2.2.2) and the discrete operators  $\mathcal{D}$  and  $\mathcal{G}$  to rewrite the Gauss-Green theorem (2.1.15) in a discrete form

$$(\mathcal{D}W, U)_{HC} = (W, \mathcal{G}U)_{\mathbf{HC}}. \quad (2.2.15)$$

We expand the discrete integral identity (2.2.15) to find

$$\sum_{i=1}^{N-1} U_{i+\frac{1}{2}} (W_{i+1} - W_i) - U_N W_N + U_1 W_1 = \sum_{i=1}^{N-1} \frac{W_{i+1}(\mathcal{G}U)_{i+1} + W_i(\mathcal{G}U)_i}{2k_{i+\frac{1}{2}}} h_{i+\frac{1}{2}}. \quad (2.2.16)$$

If we separate the sums in (2.2.16) and reindex any terms with an  $i+1$  index to an  $i$  index we find

$$\begin{aligned} \sum_{i=2}^{N-1} W_i \left\{ U_{i-\frac{1}{2}} - U_{i+\frac{1}{2}} - (\mathcal{G}U)_i \left( \frac{1}{2} \left[ \frac{h_{i+\frac{1}{2}}}{k_{i+\frac{1}{2}}} + \frac{h_{i-\frac{1}{2}}}{k_{i-\frac{1}{2}}} \right] \right) \right\} + \left( -(\mathcal{G}U)_N \left\{ \frac{1}{2} \left[ \frac{h_{N-\frac{1}{2}}}{k_{N-\frac{1}{2}}} \right] \right\} \right. \\ \left. + U_{N-\frac{1}{2}} - U_N \right) W_N + \left( U_1 - U_{\frac{3}{2}} - (\mathcal{G}U)_1 \left\{ \frac{1}{2} \left[ \frac{h_{\frac{3}{2}}}{k_{\frac{3}{2}}} \right] \right\} \right) W_1 = 0. \end{aligned} \quad (2.2.17)$$

The relationship (2.2.17) must hold for any  $W \in HC$ . Thus, we determine the form for  $\mathcal{G}$  by solving for  $(\mathcal{G}U)_i$  in (2.2.17) to find

$$\begin{aligned} (\mathcal{G}U)_1 &= -\left(\frac{1}{2} \left[ \frac{h_{\frac{3}{2}}}{k_{\frac{3}{2}}} \right]\right)^{-1} (U_{\frac{3}{2}} - U_1), \\ (\mathcal{G}U)_i &= -\left(\frac{1}{2} \left[ \frac{h_{i+\frac{1}{2}}}{k_{i+\frac{1}{2}}} + \frac{h_{i-\frac{1}{2}}}{k_{i-\frac{1}{2}}} \right]\right)^{-1} (U_{i+\frac{1}{2}} - U_{i-\frac{1}{2}}), \\ (\mathcal{G}U)_N &= -\left(\frac{1}{2} \left[ \frac{h_{N-\frac{1}{2}}}{k_{N-\frac{1}{2}}} \right]\right)^{-1} (U_N - U_{N-\frac{1}{2}}), \end{aligned} \quad (2.2.18)$$

where  $i = 2, \dots, N-1$ .

So in one dimension we find that the matrix  $\mathcal{S}$  to be banded with the form

$$\mathcal{S} = \text{diag} \left( \frac{1}{2} \left[ \frac{h_{\frac{3}{2}}}{k_{\frac{3}{2}}} \right], \dots, \frac{1}{2} \left[ \frac{h_{i+\frac{1}{2}}}{k_{i+\frac{1}{2}}} + \frac{h_{i-\frac{1}{2}}}{k_{i-\frac{1}{2}}} \right], \dots, \frac{1}{2} \left[ \frac{h_{N-\frac{1}{2}}}{k_{N-\frac{1}{2}}} \right] \right). \quad (2.2.19)$$

The operator  $\mathcal{S}$  is diagonal and thus trivial to invert. The matrix  $\mathcal{D}^\dagger \mathcal{M}$  is banded with the form

$$\begin{aligned} -(\mathcal{D}^\dagger \mathcal{M}U)_1 &= (U_{\frac{3}{2}} - U_1), \\ -(\mathcal{D}^\dagger \mathcal{M}U)_i &= (U_{i+\frac{1}{2}} - U_{i-\frac{1}{2}}), \\ -(\mathcal{D}^\dagger \mathcal{M}U)_N &= (U_N - U_{N-\frac{1}{2}}), \end{aligned} \quad (2.2.20)$$

for  $i = 2, \dots, N-1$ . Thus, the matrix  $\mathcal{D}^\dagger \mathcal{M}$  is bidiagonal.

### 2.2.6 The Discrete $\Upsilon$ Operator

The discrete form of  $\Upsilon$  (2.1.7) is

$$(\Upsilon U)_{i+\frac{1}{2}} = \begin{cases} \frac{a_{i+\frac{1}{2}}}{\Delta t} U_{i+\frac{1}{2}}, & \text{on } (x_1, x_N), \\ \alpha_1 U_1, & \text{at } x = x_1, \\ \alpha_N U_N, & \text{at } x = x_N. \end{cases} \quad (2.2.21)$$

The operator  $\Upsilon$  is diagonal, therefore symmetric, and easy to invert. It is assumed that  $a > 0$  and  $\alpha \geq 0$ , so  $\Upsilon \geq 0$ . Also note that if both  $a$  and  $\alpha$  are bounded below by  $c$  and above by  $C$

$$c(U, V)_{HC} \leq (\Upsilon U, V)_{HC} \leq C(U, V)_{HC}, \quad U, V \in HC. \quad (2.2.22)$$

### 2.2.7 The Discrete $\mathbf{F}^{n+1}$ Operator

The discrete form of the right hand side  $\mathbf{F}^{n+1}$  (2.1.4) is

$$\mathcal{F}^{n+1} = \begin{cases} f_{i+\frac{1}{2}} + \frac{a_{i+\frac{1}{2}}}{\Delta t} U_{i+\frac{1}{2}}^n, & \text{on } (x_1, x_N), \\ \psi_1, & \text{at } x = x_1, \\ \psi_N, & \text{at } x = x_N. \end{cases} \quad (2.2.23)$$

The discrete operator  $\mathcal{F}^{n+1}$  is the cell-centered values of the right hand side of the fully implicit equation (2.1.1) and the right hand side of the boundary conditions (2.4).

### 2.2.8 The Fully Discretized Diffusion Equation

We've constructed the necessary discrete operators to determine the fully discretized forms of the semi-discrete diffusion equation (2.1.2) and (2.1.9). The first order system (2.1.6) is discretized as

$$W^{n+1} - \mathcal{G}U^{n+1} = 0, \quad \mathcal{D}W^{n+1} + \Upsilon U^{n+1} = \mathcal{F}^{n+1}. \quad (2.2.24)$$

If the flux is eliminated from (2.2.24), then we obtain a fully discretized form of the diffusion equation (2.1.2)

$$\mathcal{A}U^{n+1} = (\Upsilon + \mathcal{D}\mathcal{G})U^{n+1} = \mathcal{F}^{n+1}. \quad (2.2.25)$$

However, if the discrete temperature  $U$  is eliminated from (2.2.24) we find a single equation for the flux analogous to (2.1.9)

$$\mathcal{B}W^{n+1} = (\mathcal{I} + \mathcal{G}\Upsilon^{-1}\mathcal{D})W^{n+1} = \mathcal{G}\Upsilon^{-1}\mathcal{F}^{n+1}, \quad (2.2.26)$$

where  $\mathcal{I}$  is the discrete identity operator and we assume the operator  $\Upsilon$  is positive definite.

The discrete operators  $\mathcal{A}$  and  $\mathcal{B}$  have symmetry and positivity properties. In the discrete inner product space  $HC$  we have

$$\begin{aligned} (\mathcal{A}U, V)_{HC} &= (\Upsilon U, V)_{HC} + (\mathcal{D}\mathcal{G}U, V)_{HC} \\ &= (\Upsilon U, V)_{HC} + (\mathcal{G}U, \mathcal{D}^*V)_{HC} \\ &= (\Upsilon U, V)_{HC} + (\mathcal{G}U, \mathcal{G}V)_{HC}, \end{aligned} \quad (2.2.27)$$

so clearly  $\mathcal{A}$  is symmetric and positive, and positive definite if either  $\Upsilon$  or  $\mathcal{G}$  is positive definite. Also,

$$\begin{aligned} (\mathcal{B}W, Z)_{HC} &= (\mathcal{I}W, Z)_{HC} + (\mathcal{G}\Upsilon^{-1}\mathcal{D}W, Z)_{HC} \\ &= (W, Z)_{HC} + (\Upsilon^{-1}\mathcal{D}W, \mathcal{G}^*Z)_{HC} \\ &= (W, Z)_{HC} + (\Upsilon^{-1}\mathcal{D}W, \mathcal{D}Z)_{HC}, \end{aligned} \quad (2.2.28)$$

where it is clear that  $\mathcal{B}$  is symmetric and positive definite if  $\Upsilon$  is positive definite.

Regrettably, both operators  $\mathcal{A}$  and  $\mathcal{B}$  are nonlocal whenever  $\mathcal{G}$  is nonlocal. This is not a problem in one dimension because the operator  $\mathcal{G}$  is always local. However, in higher dimensions we need an efficient numerical method with local operators. We show later that if we rearrange the assembly of  $\mathcal{A}$ , then  $\mathcal{A}U$  can be computed efficiently. The computation is efficient because we can use a large class of fast iterative solvers. For the operator  $\mathcal{B}$ , we note

$$\mathcal{I}\mathcal{B} = \mathcal{I} + \mathcal{D}^\dagger \mathcal{M}\Upsilon\mathcal{D}, \quad (2.2.29)$$

is banded so that  $\mathcal{B}^{-1} = (\mathcal{I}\mathcal{B})^{-1}\mathcal{I}$  is the product of the inverse of a banded matrix and a banded matrix. The structure of the the operator  $\mathcal{B}$  is related to compact finite difference schemes. More explicitly, we take the discrete flux

$$W = \mathcal{G}U = \mathcal{I}^{-1}\mathcal{D}^\dagger \mathcal{M}U. \quad (2.2.30)$$

Applying the operator  $\mathcal{I}$  to both sides of (2.2.30) yields

$$\mathcal{I}W = \mathcal{D}^\dagger \mathcal{M}U, \quad (2.2.31)$$

where  $\mathcal{I}$  and  $\mathcal{D}^\dagger \mathcal{M}$  are banded. So (2.2.31) is a compact representation of the fluxes.

### 2.2.9 The Computational Sequence

We can solve the diffusion equation using two approaches: solve (2.2.25) for  $U$ ; or solve (2.2.26) for  $W$  and then recover  $U$  from  $W$ . We have observed that the operator  $\mathcal{A} = \Upsilon + \mathcal{D}\mathcal{G}$  is generally nonlocal, particularly in higher dimensions, and thus is not practical. A major difficulty is rearranging the equation (2.2.25) so that  $\mathcal{G}$  is to the left of  $\mathcal{D}$ , for which the solution would be efficient to compute.

To compute  $\mathcal{A}U$  efficiently we reintroduce the flux  $W = \mathcal{G}U$  and note that  $W$  satisfies (2.2.31),  $\mathcal{S}W = \mathcal{D}^\dagger \mathcal{M}U$ , which can be solved for  $W$  with standard methods because  $\mathcal{S}$  is symmetric positive definite. Then  $\mathcal{A}U = \Upsilon U + \mathcal{D}W$ . This approach requires the computation of  $W$  as an intermediate step so we will devote our attention to a flux based algorithm and not pursue this approach further.

The operator  $\mathcal{B}$ , as noted earlier, is nonlocal, but since  $\mathcal{S}\mathcal{B}$  is local we combine (2.2.26) multiplied by  $\mathcal{S}$  to obtain

$$(\mathcal{S} + \mathcal{D}^\dagger \mathcal{M}\Upsilon^{-1}\mathcal{D})W^{n+1} = \mathcal{D}^\dagger \mathcal{M}\Upsilon^{-1}\mathcal{F}^{n+1}, \quad (2.2.32)$$

which is convenient for computation because (2.2.32) has a local stencil. In fact, in one dimension the matrix  $(\mathcal{S} + \mathcal{D}^\dagger \mathcal{M}\Upsilon^{-1}\mathcal{D})$  is tridiagonal. To see the properties of the coefficient matrix for  $W$  in (2.2.32), we note

$$\begin{aligned} [(\mathcal{S} + \mathcal{D}^\dagger \mathcal{M}\Upsilon^{-1}\mathcal{D})W, W]_{\mathbf{HC}} &= [\mathcal{S}W, W]_{\mathbf{HC}} + [\mathcal{D}^\dagger \mathcal{M}\Upsilon^{-1}\mathcal{D}W, W]_{\mathbf{HC}} \\ &= [\mathcal{S}W, W]_{\mathbf{HC}} + [\mathcal{M}\Upsilon^{-1}\mathcal{D}W, \mathcal{D}W]_{\mathbf{HC}} \\ &\geq \xi_2 \|W\|_{\mathbf{HC}} + \frac{\xi_1}{\mathcal{C}} \|\mathcal{D}W\|_{\mathbf{HC}}, \end{aligned} \quad (2.2.33)$$

where the constants  $\xi_1$ ,  $\xi_2$  and  $\mathcal{C}$  are defined in (2.2.6) and (2.2.22) respectively. Thus, the coefficient matrix for  $W$  is symmetric and positive definite.

We solve the matrix equation (2.2.32) with Gauss-Seidel; however other iterative algorithms for the symmetric positive system are available. For example, SOR or conjugate gradient [11]. The coefficient matrix in (2.2.33) is symmetric positive definite for any type of boundary conditions - Dirichlet, Neumann, or Robin - which is convenient for computation.

After the fluxes  $W^{n+1}$  are computed from (2.2.33), the temperature is computed from the explicit formula

$$U^{n+1} = \Upsilon^{-1} (\mathcal{F}^{n+1} - \mathcal{D}W^{n+1}). \quad (2.2.34)$$

### 2.3 Null Space of the Discrete Flux Operator

An important property of the gradient is that  $\nabla u = 0$  implies that  $u$  is a constant function. Because  $k > 0$ , this is the same as  $-k\nabla u = 0$  implies that  $u$  is a constant. The same result holds for the discrete flux operator derived in Sec. 2.2.5. To see this, assume that  $\mathcal{G}U = 0$  where  $\mathcal{G}$  is defined in (2.2.18). Thus, we have on the interior

$$(\mathcal{G}U)_i = - \left[ \frac{1}{2} \left( \frac{h_{i+1/2}}{k_{i+1/2}} + \frac{h_{i-1/2}}{k_{i-1/2}} \right) \right]^{-1} \left( U_{i+\frac{1}{2}} - U_{i-\frac{1}{2}} \right) = 0. \quad (2.3.1)$$

Then Eq. (2.3.1) gives

$$U_{i+\frac{1}{2}} = U_{i-\frac{1}{2}}, \quad (2.3.2)$$

for  $i = 2, \dots, N-1$ . A similar result holds for the boundary terms. The Eq. (2.3.2) implies that  $U$  is constant in  $i$ , so  $U$  is a constant function.

We have shown that the constant functions are in the null space of the discrete flux. However, we must show that they are the *only* members of the null space. It is sufficient to show that the null space of the operator  $\mathcal{G}$  is one dimensional. The discrete flux  $\mathcal{G} : HC \rightarrow \mathbf{HC}$  maps cell-centered values to cell face values. If we include the boundary conditions in the discrete temperatures  $U$ , then the matrix  $\mathcal{G} \in \mathbb{R}^{N \times N+1}$ . We note that the matrix components of  $\mathcal{G}$  have dimension  $\mathcal{S}^{-1} \in \mathbb{R}^{N \times N}$  and  $\mathcal{D}^\dagger \mathcal{M} \in \mathbb{R}^{N \times N+1}$ . Clearly the matrix  $\mathcal{S}$  has  $\text{rank}(N)$  as it is invertible. From the bidiagonal structure of the differencing operator  $\mathcal{D}^\dagger \mathcal{M}$  it is straightforward to determine the matrix  $\mathcal{G}$  has  $\text{rank}(N)$ . Thus the null space of  $\mathcal{G}$  is one dimensional and contains the constant functions.

The property that the null space of the discrete flux contains only the constant functions, just as the differential operator  $-\mathbf{K}\nabla$ , is important in numerical schemes. For numerical schemes that do not have this property typically the highest-frequency mode is also in the null space of the discrete gradient [12]. A special filtering procedure is required, like that of Margolin [10].

### 3 Non-Linear Heat Conductivity

The diffusion equation (2.1) only models linear heat conductivity. In Sec. 2 we allow  $k$  to be discontinuous at cell interfaces. A discontinuous  $k$  gives the ability to model linear diffusion across material interfaces. In certain applications, like hydrodynamics, problems with non-linear diffusion become important [2]. To generalize the equation (2.1) to admit non-linear diffusion we take the heat conductivity to be a function of the temperature, i.e., define the heat conductivity  $\kappa \equiv \kappa(u)$ . We present numerical results for non-linear heat conductivity in Secs. 5.2.3 and 5.3.

We assume that there is no forcing of the non-linear equation, i.e,  $f \equiv 0$  in (2.1). Next, we introduce the non-linear heat flux

$$w = -\kappa(u) \frac{\partial u}{\partial x}, \quad (3.1)$$

to rewrite the diffusion equation (2.1) as the first order system

$$\begin{cases} a \frac{\partial u}{\partial t} + \frac{\partial w}{\partial x} = 0, \\ w = -\kappa(u) \frac{\partial u}{\partial x}, \end{cases} \quad (3.2)$$

with some initial condition and set of boundary conditions. As a simplifying assumption we take the non-linearity in the heat flux (3.1) to manifest itself as a power  $\alpha \in \mathbb{R}$  on the temperature  $u$ . That is, we consider the non-linear flux to be of the form

$$w = -\kappa(u) \frac{\partial u}{\partial x} = -ku^\alpha \frac{\partial u}{\partial x}. \quad (3.3)$$

The non-linearity in the first-order system (3.2) makes the heat equation more difficult to solve numerically. Thus, we employ a non-linear transformation

$$\theta = u^{\alpha+1}. \quad (3.4)$$

From the chain rule we find the time derivative of  $\theta$

$$\frac{\partial \theta}{\partial t} = u^\alpha (\alpha + 1) \frac{\partial u}{\partial t}, \quad (3.5)$$

and similarly the spatial derivative of  $\theta$

$$\frac{\partial \theta}{\partial x} = u^\alpha (\alpha + 1) \frac{\partial u}{\partial x}. \quad (3.6)$$

We use the time and space derivatives of the non-linear transformation (3.5) and (3.6) to rewrite the first-order system (3.2) in such a way that the heat conductivity  $\kappa$  does not depend on the temperature. The transformed system in the new variable  $\theta$  is

$$\begin{cases} \bar{a} \frac{\partial \theta}{\partial t} + \frac{\partial w}{\partial x} = 0 \\ w = -\bar{k} \frac{\partial \theta}{\partial x} \end{cases}, \quad (3.7)$$

where

$$\begin{aligned} \bar{a} &= \frac{a}{\theta^{\alpha/\alpha+1}(\alpha+1)} \\ \bar{k} &= \frac{k}{\alpha+1}. \end{aligned} \quad (3.8)$$

Again we assume that we have some initial condition and set of boundary conditions to close the system (3.7).

The system of equations (3.7) has an identical form to the linear heat equation written as a first-order system. The only difference is the constants  $a$  and  $k$  are altered by the non-linear transformation (3.4). Conveniently we can apply the same MFD scheme, with altered constants  $\bar{a}$  and  $\bar{k}$ , to find the discrete non-linear fluxes  $W$ . Another advantage is the MFD scheme treats the computation of the fluxes in a fully implicit way. Thus, we know that the numerical method is stable and accurate when applied to non-linear problems.

On a general, non-uniform one dimensional grid the non-linear fluxes are found using the scheme

$$\begin{aligned} -\frac{\Delta t}{\bar{a}_{i-\frac{1}{2}}^n h_{i-\frac{1}{2}}} W_{i-1}^{n+1} + \left[ \frac{1}{2} \left( \frac{h_{i-\frac{1}{2}}}{\bar{k}_{i-\frac{1}{2}}^n} + \frac{h_{i+\frac{1}{2}}}{\bar{k}_{i+\frac{1}{2}}^n} \right) + \frac{\Delta t}{\bar{a}_{i-\frac{1}{2}}^n h_{i-\frac{1}{2}}} + \frac{\Delta t}{\bar{a}_{i+\frac{1}{2}}^n h_{i+\frac{1}{2}}} \right] W_i^{n+1} \\ - \frac{\Delta t}{\bar{a}_{i+\frac{1}{2}}^n h_{i+\frac{1}{2}}} W_{i+1}^{n+1} = \Theta_{i-\frac{1}{2}}^n - \Theta_{i+\frac{1}{2}}^n, \end{aligned} \quad (3.9)$$

where  $\Theta_{i+1/2}^n$  is the cell-centered value of the non-linear transformation (3.4) in cell  $i + \frac{1}{2}$  at time level  $n$ . The implicit scheme (3.9) yields a tridiagonal, symmetric positive-definite matrix to compute the discrete non-linear fluxes  $W$ . We solve the system for  $W$  using Gauss-Seidel. Once the non-linear fluxes are computed, we recover the discrete temperature function  $U$  in each cell from a discretization of the first equation in (3.2)

$$U_{i+\frac{1}{2}}^{n+1} = U_{i+\frac{1}{2}}^n - \frac{\Delta t}{a_{i+\frac{1}{2}}} \frac{W_{i+1}^{n+1} - W_i^{n+1}}{h_{i+\frac{1}{2}}}. \quad (3.10)$$

## 4 Generalizations of the Mimetic Finite Difference Scheme

A problem of interest is to accurately model the heat diffusion across a material interface located inside a cell. The global method derived in Sec. 2 accurately handles material discontinuities as long as the material interface coincides with a cell interface, i.e., all cells are pure. We verify the accuracy of the global method in Sec. 5.1. However, it may not be the case that the material interface lies on a cell interface, for example, if the material moves in time.

Another interesting problem arises after we compute the temperature on a mesh that contains cells with a mixture of material properties, deemed a mixed cell. We have a single, cell-centered value for the temperature in a mixed cell, which we think of as a total amount of heat. How can we recover the temperature values on either side of the interface from the total heat? We describe a limited linear reconstruction technique to recover the values on either side of a material interface in a mixed cell.

## 4.1 Diffusion in a Mixed Cell

An option for multiple material problems is to move the mesh and track the material interface to ensure it is always placed on a cell interface. But moving mesh methods may become computationally expensive in higher dimensions [3]. Also, we do not want to introduce extra degrees of freedom due to possible mesh refinement required to place a material interface on a cell boundary. Instead we describe two methods to generalize the MFD scheme to operate on meshes with mixed cells, homogenization and condensation.

For the remainder of this report we assume any mixed cell has only two components with heat conductivities  $k_L$  and  $k_R$ . Also, we denote the location of the material interface as  $x^* \in (x_i, x_{i+1})$  when the  $i + \frac{1}{2}$  cell is mixed. We denote the distances on either side of the material interface in the mixed cell as  $h_L = x^* - x_i$  and  $h_R = x_{i+1} - x^*$ .

### 4.1.1 Technique of Homogenization

A tactic to extend the MFD scheme is to homogenize the material properties in the mixed cell. The discrete flux operator in the global MFD method (2.2.18) requires a heat conductivity value  $k$  for each cell. However, in a mixed cell it is not immediately clear what value to choose for  $k$ . Obviously we could just assign the left or right material property to the mixed cell but this is unphysical. In homogenization we determine an average material property  $\bar{k}$  which we assign to the mixed cell.

There are many choices for the calculation of  $\bar{k}$ . A few of the choices are the volume weighted harmonic mean

$$\bar{k} = \frac{h_{i+1/2}}{\left(\frac{h_L}{k_L} + \frac{h_R}{k_R}\right)}, \quad (4.1.1)$$

the volume weighted arithmetic mean

$$\bar{k} = \frac{h_L}{h_{i+1/2}}k_L + \frac{h_R}{h_{i+1/2}}k_R, \quad (4.1.2)$$

and volume weighted geometric mean

$$\bar{k} = \exp\left(\frac{h_L \log(k_L)}{h_{i+1/2}} + \frac{h_R \log(k_R)}{h_{i+1/2}}\right). \quad (4.1.3)$$

With a heat conductivity value assigned to any mixed cells we can proceed in the normal fashion with the global discretization outlined in Sec. 2. In mixed cells, we calculate the cell-centered value which represents the total heat. The recovery of the temperature on either side of the material interface is a post-processing step. We explore the effect the homogenization has on the order of the method in Sec. 5.2.1 and which volume weighted mean coupled with a limited linear reconstruction produces the most accurate results for the recovered temperature values in Secs. 5.2.2 and 5.2.3.

### 4.1.2 A Local Reformulation with Condensation

Rather than homogenizing the material properties in a mixed cell and applying the global method derived in Sec. 2, we rederive the MFD scheme using a local version of the SOM. The local derivation gives extra flexibility in the mixed cell and allows us to alter the stencil of the scheme to respect the material interface. The stencil is augmented by introducing intermediate degrees of freedom at the material interface. We eliminate the new degrees of freedom using continuity assumptions. This is the process of condensation, for details in two dimensions see the work by Kuznetsov [7, 8].

We begin with the one dimensional domain  $\Omega = [x_1, x_N]$  which we break into  $N - 1$  non-overlapping cells denoted  $e_{i+1/2}$ . The local method is derived on a single cell  $e_{i+1/2}$ . We will show that the local and global support operators method are equivalent, on pure cells, in that they produce the same flux components and cell-centered temperature values. That is, the two methods represent the same discrete operator but differ in their choice of independent variables.

We derive the local method in cell  $e_{i+1/2}$  which we assume is mixed. In the mixed cell there is a material interface located at  $x^* \in (x_i, x_{i+1})$ . We define the degrees of freedom in our problem to be a cell-centered value for temperature  $U_{i+1/2}$  and face centered values for the temperature  $U_i, U_{i+1}$  and flux  $(\mathcal{G}U)_i, (\mathcal{G}U)_{i+1}$ . In the condensation method we introduce an intermediate discrete flux value,  $(\mathcal{G}U)^*$ , at  $x^*$  which we eliminate later.

With the degrees of freedom defined we discretize the Gauss-Green theorem, which in one spatial dimension is just integration by parts, on a single cell

$$\int_{x_i}^{x_{i+1}} u \frac{\partial w}{\partial x} dx + \int_{x_i}^{x_{i+1}} w \frac{\partial u}{\partial x} dx = uw \Big|_{x_i}^{x_{i+1}}. \quad (4.1.4)$$

The cell is mixed, so we split the integrals on the left side of (4.1.4) along the interface to obtain

$$\int_{x_i}^{x^*} u \frac{\partial w}{\partial x} dx + \int_{x^*}^{x_{i+1}} u \frac{\partial w}{\partial x} dx + \int_{x_i}^{x^*} wk_L^{-1} \left( k_L \frac{\partial u}{\partial x} \right) dx + \int_{x^*}^{x_{i+1}} wk_R^{-1} \left( k_R \frac{\partial u}{\partial x} \right) dx = uw \Big|_{x_i}^{x_{i+1}}, \quad (4.1.5)$$

where we have scaled the last two integrals by the heat conductivity coefficient  $k$  on either side of the material interface.

To simplify the expressions in (4.1.5) we introduce the flux  $\mathbf{G} = -k \frac{\partial}{\partial x}$  for either value of  $k$ . To keep the discussion general and extendable to two dimensions we need to keep track of which functions are analogs of scalars and which are analogs of vectors as well as which derivative operations in (4.1.5) correspond to the divergence or the gradient. In this section we take  $u$  to be a scalar function and  $w, \mathbf{G}u$  to be vector functions. Just as in Sec. 2, derivatives that act on  $u$  are analogous to the gradient and derivatives that act on  $w$  are analogous to the divergence. As with the global derivation using the support operators method we select the divergence to be the prime operator. The divergence is defined as before in (2.2.7) by

$$\frac{\partial w}{\partial x} \approx \frac{w_{i+1} - w_i}{h_{i+\frac{1}{2}}}. \quad (4.1.6)$$

From the discrete divergence operator we determine the discrete flux operator.

First we equip the spaces of scalar and vector functions with discrete inner products. Effectively this means we select a quadrature technique to approximate the integrals in (4.1.5). It is natural to select midpoint rule for the first two integrals in (4.1.5) because the discrete information for  $u$  is cell-centered. Similarly, the last two integrals in (4.1.5) are discretized using trapezoid rule because the discrete information for fluxes is at the edges of the cell. Applying

the two quadrature rules we find

$$\begin{aligned}
\int_{x_i}^{x^*} u \frac{\partial w}{\partial x} dx &\approx U_{i+\frac{1}{2}}(w^* - w_i) \\
\int_{x^*}^{x_{i+1}} u \frac{\partial w}{\partial x} dx &\approx U_{i+\frac{1}{2}}(w_{i+1} - w^*) \\
\int_{x_i}^{x^*} w k_L^{-1} \mathbf{G} u dx &\approx \frac{w_i(\mathcal{G}U)_i + w^*(\mathcal{G}U)^*}{2k_L} h_L \\
\int_{x^*}^{x_{i+1}} w k_R^{-1} \mathbf{G} u dx &\approx \frac{w^*(\mathcal{G}U)^* + w_{i+1}(\mathcal{G}U)_{i+1}}{2k_R} h_R,
\end{aligned} \tag{4.1.7}$$

where  $h_L = x^* - x_i$  and  $h_R = x_{i+1} - x^*$ .

We eliminate the extra degree of freedom introduced at the material interface with the assumption that the divergence in the left portion of the mixed cell is the same as the divergence across the entire cell, i.e.,

$$\frac{w^* - w_i}{h_L} = \frac{w_{i+1} - w_i}{h_{i+\frac{1}{2}}}. \tag{4.1.8}$$

The same assumption about the divergence in a portion of the mixed cell is made by Kuznetsov [7, 8]. We solve for the intermediate flux value  $w^*$  to find

$$w^* = \frac{h_R w_i + h_L w_{i+1}}{h_{i+\frac{1}{2}}}. \tag{4.1.9}$$

A similar argument as (4.1.8) is used to find an identical expression to (4.1.9) for the intermediate flux  $(\mathcal{G}U)^*$ . The expressions for  $w^*$  and  $(\mathcal{G}U)^*$  are substituted in the last two integral approximations in (4.1.7). Noting that the values for  $w_{i+1}$  and  $w_i$  are arbitrary and combining the integral identity (4.1.5), the integral approximations (4.1.7), and the intermediate flux value (4.1.9) we arrive at two equations for the two unknowns  $(\mathcal{G}U)_i$  and  $(\mathcal{G}U)_{i+1}$

$$\begin{aligned}
\frac{h_L}{2k_L} \left( \frac{h_L h_R}{h^2_{i+\frac{1}{2}}} (\mathcal{G}U)_i + \frac{h_L^2}{h^2_{i+\frac{1}{2}}} (\mathcal{G}U)_{i+1} \right) + \frac{h_R}{2k_R} \left( \frac{h_L h_R}{h^2_{i+\frac{1}{2}}} (\mathcal{G}U)_i + \left[ 1 + \frac{h_L^2}{h^2_{i+\frac{1}{2}}} \right] (\mathcal{G}U)_{i+1} \right) \\
= -(U_{i+1} - U_{i+\frac{1}{2}}) \\
\frac{h_L}{2k_L} \left( \left[ 1 + \frac{h_R^2}{h^2_{i+\frac{1}{2}}} \right] (\mathcal{G}U)_i + \frac{h_L h_R}{h^2_{i+\frac{1}{2}}} (\mathcal{G}U)_{i+1} \right) + \frac{h_R}{2k_R} \left( \frac{h_R^2}{h^2_{i+\frac{1}{2}}} (\mathcal{G}U)_i + \frac{h_L h_R}{h^2_{i+\frac{1}{2}}} (\mathcal{G}U)_{i+1} \right) \\
= -(U_{i+\frac{1}{2}} - U_i).
\end{aligned} \tag{4.1.10}$$

We collect like terms and rewrite (4.1.10) as a matrix equation

$$\begin{bmatrix} \frac{h_L^2 h_R}{2k_L h^2_{i+\frac{1}{2}}} + \frac{h_L h_R^2}{2k_R h^2_{i+\frac{1}{2}}} & \frac{h_L^3}{2k_L h^2_{i+\frac{1}{2}}} + \frac{h_R}{2k_R} \left( 1 + \frac{h_L^2}{h^2_{i+\frac{1}{2}}} \right) \\ \frac{h_L}{2k_L} \left( 1 + \frac{h_R^2}{h^2_{i+\frac{1}{2}}} \right) + \frac{h_R^3}{2k_R h^2_{i+\frac{1}{2}}} & \frac{h_L^2 h_R}{2k_L h^2_{i+\frac{1}{2}}} + \frac{h_L h_R^2}{2k_R h^2_{i+\frac{1}{2}}} \end{bmatrix} \begin{bmatrix} (\mathcal{G}U)_i \\ (\mathcal{G}U)_{i+1} \end{bmatrix} = - \begin{bmatrix} U_{i+1} - U_{i+\frac{1}{2}} \\ U_{i+\frac{1}{2}} - U_i \end{bmatrix}. \tag{4.1.11}$$

The matrix equation (4.1.11) is always non-singular, so we can find general expressions for the fluxes  $(\mathcal{G}U)_i$  and  $(\mathcal{G}U)_{i+1}$  in the mixed cell. The expressions are unwieldy and not illuminating,

so they are omitted. In one dimension it is laborious, but not difficult to create explicit formulae for the edge fluxes in the mixed cell. However, in two dimensions such analytical computations will become unnecessarily difficult and labor intensive. Thus, it is likely that in higher dimensions the matrix system to determine the edge fluxes will be assembled and solved numerically.

Rather than repeating the local derivation for a pure cell we consider, without loss of generality,  $h_L = 0$  and  $h_R = h_{i+1/2}$ . The equations for the edge fluxes given by (4.1.11) simplify considerably to

$$\begin{aligned} (\mathcal{G}U)_i &= - \left[ \frac{h_{i+1/2}}{2k_{i+1/2}} \right]^{-1} \left( U_{i+\frac{1}{2}} - U_i \right), \\ (\mathcal{G}U)_{i+1} &= - \left[ \frac{h_{i+1/2}}{2k_{i+1/2}} \right]^{-1} \left( U_{i+1} - U_{i+\frac{1}{2}} \right). \end{aligned} \quad (4.1.12)$$

We have a local approximation on each cell whether it is mixed or pure. Next we create a global method on the entire domain  $\Omega = [x_1, x_N]$  by connecting the local approximations using continuity assumptions. We assume that the face temperatures and normal fluxes are equal at cell interfaces. For the one dimensional problem the continuity assumptions give

$$U_i^{i-\frac{1}{2}} = U_i^{i+\frac{1}{2}} \quad \text{and} \quad (\mathcal{G}U)_i^{i+1/2} - (\mathcal{G}U)_{i+1}^{i-1/2} = 0, \quad (4.1.13)$$

where the superscripts denote the cell in which the temperature and flux are located. The continuity conditions (4.1.13) allow us to find an expression for the cell interface temperature values in terms of the cell-centered values.

First we examine the pure cells as the algebra is simpler. From the continuity assumptions (4.1.13) and the local fluxes (4.1.12) we find the cell edge temperature values to be

$$U_i = \left[ \frac{k_{i+1/2}}{0.5h_{i+1/2}} + \frac{k_{i-1/2}}{0.5h_{i-1/2}} \right]^{-1} \left( \frac{k_{i+1/2}}{0.5h_{i+1/2}} U_{i+\frac{1}{2}} + \frac{k_{i-1/2}}{0.5h_{i-1/2}} U_{i-\frac{1}{2}} \right), \quad (4.1.14)$$

where  $i = 2, \dots, N-1$ . The values for  $U_1$  and  $U_N$  are handled by the boundary conditions. If we use the expression for the cell interface temperature values (4.1.14) in the first expression for the fluxes in (4.1.12) we find

$$(\mathcal{G}U)_i = - \left[ \frac{1}{2} \left( \frac{h_{i+1/2}}{k_{i+1/2}} + \frac{h_{i-1/2}}{k_{i-1/2}} \right) \right]^{-1} \left( U_{i+\frac{1}{2}} - U_{i-\frac{1}{2}} \right), \quad (4.1.15)$$

which is the same discrete flux operator (2.2.18) at interior cell interfaces derived in the global SOM described in Sec. 2. Thus, the local derivation using condensation and the global method are equivalent in the pure cells as both derivations lead to the same discrete operator. However, the local SOM allows us the freedom to develop a more complicated discrete flux that alters the differencing at a material interface located within a cell.

Next we consider the flux at the cell interfaces of the mixed cell. Again, we use the continuity assumption (4.1.13) to determine the interface temperature values in terms of the neighboring cell-centered values. Let us assume that cell  $i + \frac{1}{2}$  is mixed. The process to determine analytical expressions for  $U_i$  and  $U_{i+1}$  is fairly involved and the final expressions are unwieldy so we omit them. In turn, we use the analytical expressions for  $U_i$  and  $U_{i+1}$  in the mixed cell to rewrite the fluxes  $(\mathcal{G}U)_i$  and  $(\mathcal{G}U)_{i+1}$ , which are the solution of the matrix equation (4.1.11), to depend only on cell-centered values. The process of rewriting the mixed cell fluxes to depend only on cell-centered temperature values is especially algebraically involved. However, it is possible to determine analytical expressions of the fluxes  $(\mathcal{G}U)_i$  and  $(\mathcal{G}U)_{i+1}$  in one dimension. We omit the analytical expressions of the mixed cell fluxes, but discuss some of the important results.

The local condensation method alters the coefficients and differencing in the discrete flux for the mixed cell. The flux calculation will depend on  $U_{i-1/2}$ ,  $U_{i+1/2}$ , and  $U_{i+3/2}$ . The local method changes the entries of the operator  $\mathcal{S}$  in (2.2.19) near the mixed cell. The matrix  $\mathcal{S}$  is still predominantly diagonal but has non-trivial blocks for the calculation near the mixed cell. The newly derived method augments  $\mathcal{D}^\dagger \mathcal{M}$  in (2.2.20) to be tridiagonal, though  $\mathcal{D}^\dagger \mathcal{M}$  remains largely bidiagonal with the only non-trivial subdiagonal entries occurring around the indices of the mixed cell. So the discrete gradient calculation in the mixed cell is no longer a backward difference operation as we saw in the global method and (4.1.15). Instead we have a centered difference with more complicated coefficients than the harmonic average found in (2.2.18).

## 4.2 New Material Temperatures in Mixed Cells

Either homogenization or condensation coupled with the MFD scheme allow us to apply the numerical method to meshes with mixed cells. Now we address the problem of how to recover the temperature on either side of the material interface in a mixed cell. We denote the reconstructed temperature values on the left or right of the material interface as  $U_L$  and  $U_R$  respectively. We need two equations to form a closed system to determine the two unknown values  $U_L$  and  $U_R$ .

Without loss of generality we assume that the  $i + \frac{1}{2}$  cell is mixed. The cell-centered value  $U_{i+1/2}$  represents the total heat in the mixed cell. One of the equations in the system to determine  $U_L$  and  $U_R$  guarantees that we do not create any new heat in the mixed cell

$$h_L U_L + h_R U_R = h_{i+\frac{1}{2}} U_{i+\frac{1}{2}}. \quad (4.2.1)$$

For simplicity we assumed a mixed cell contains two components. It is reasonable to define the relationship

$$U_R = U_L + \Delta U, \quad (4.2.2)$$

where  $\Delta U$  represents some unknown change in the temperature across the material interface. We write the two equations (4.2.1) and (4.2.2) in matrix form to find

$$\begin{bmatrix} h_L & h_R \\ -1 & 1 \end{bmatrix} \begin{bmatrix} U_L \\ U_R \end{bmatrix} = \begin{bmatrix} h_{i+\frac{1}{2}} U_{i+\frac{1}{2}} \\ \Delta U \end{bmatrix}. \quad (4.2.3)$$

The determinant of the coefficient matrix in (4.2.3) is  $h_L + h_R = h_{i+1/2} \neq 0$ . So the system of the equations (4.2.3) always produces a unique solution for  $U_L$  and  $U_R$ . Inverting the  $2 \times 2$  system in (4.2.3) gives a closed form for  $U_L$  and  $U_R$

$$\begin{aligned} U_L &= U_{i+\frac{1}{2}} - \frac{h_R}{h_{i+\frac{1}{2}}} \Delta U, \\ U_R &= U_{i+\frac{1}{2}} - \frac{h_L}{h_{i+\frac{1}{2}}} \Delta U. \end{aligned} \quad (4.2.4)$$

We introduce a new degree of freedom  $\Delta U$  and require a method to determine its value. Once that value is found it is simple to calculate  $U_L$  and  $U_R$  from (4.2.4). To determine  $\Delta U$  we use a limited linear reconstruction of the solution in the mixed cell

$$U_R = U_L + \left( \frac{\partial U}{\partial x} \right)^{lim} (x_R - x_L), \quad (4.2.5)$$

where  $(\partial U / \partial x)^{lim}$  is the limited derivative approximation,  $x_R = (x_{i+1} + x^*)/2$ , and  $x_L = (x_i + x^*)/2$ . From (4.2.2) and (4.2.5) we know,

$$\Delta U = \left( \frac{\partial U}{\partial x} \right)^{lim} (x_R - x_L). \quad (4.2.6)$$

There is some freedom in how we choose the limited derivative reconstruction  $(\partial U/\partial x)^{lim}$ . We discuss the *minmod* and Barth-Jespersen limiters but other options are available.

The idea of a limited linear reconstruction is more general than given in (4.2.5). A linear reconstruction valid anywhere in cell  $i + \frac{1}{2}$  is given by

$$U(x) = U_{i+\frac{1}{2}} + \left(\frac{\partial U}{\partial x}\right)^{lim} (x - x_{i+\frac{1}{2}}), \quad x \in [x_i, x_{i+1}], \quad (4.2.7)$$

where we recover the reconstruction in (4.2.5) if we center the general reconstruction (4.2.7) at  $x_L$  and evaluate at  $x_R$ . The limited linear reconstruction used to determine  $\Delta U$  and solve for  $U_L$  and  $U_R$  in (4.2.4) is discussed in one dimension but the extension to two dimensions is straightforward. The conservation condition (4.2.1) changes slightly to involve volumes and the linear reconstruction becomes more complicated, for details, see [9].

The assumption of two components in a mixed cell does not constrain the linear reconstruction procedure, only simplifies the discussion. For example, if the mixed cell contained three materials, then we would want to determine three intermediate values, say,  $U_L$ ,  $U_M$ , and  $U_R$ . The heat conservation condition (4.2.1) simply has  $h_M U_M$  added to the left hand side. Next we introduce two equations to close the system

$$U_M = U_L + \Delta U_1 \quad \text{and} \quad U_R = U_M + \Delta U_2, \quad (4.2.8)$$

where we now have two unknown changes in temperature across the material interfaces  $\Delta U_1$  and  $\Delta U_2$ . These two unknowns are found from two separate limited linear reconstructions. One can see how the process can generalize to an arbitrary number of materials. Thus, the method for recovering temperature values in mixed cells we discuss is generalizable to higher dimensions as well as an arbitrary number of materials.

#### 4.2.1 *minmod* in One Dimension

A popular choice for the cell-centered derivative of temperature,  $(\partial U/\partial x)^{lim}$ , is termed the *minmod* method [13]. The method compares one-sided derivative approximations with respect to cell  $i + \frac{1}{2}$

$$\left(\frac{\partial U}{\partial x}\right)^L = \frac{U_{i+\frac{1}{2}} - U_{i-\frac{1}{2}}}{x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}} \quad \text{and} \quad \left(\frac{\partial U}{\partial x}\right)^R = \frac{U_{i+\frac{3}{2}} - U_{i+\frac{1}{2}}}{x_{i+\frac{3}{2}} - x_{i+\frac{1}{2}}}. \quad (4.2.9)$$

The *minmod* derivative is defined as

$$\left(\frac{\partial U}{\partial x}\right)^{lim} = \begin{cases} 0, & \text{if } \left(\frac{\partial U}{\partial x}\right)^L \cdot \left(\frac{\partial U}{\partial x}\right)^R < 0 \\ \text{sign} \left( \left(\frac{\partial U}{\partial x}\right)^L \right) \cdot \min \left( \left| \left(\frac{\partial U}{\partial x}\right)^L \right|, \left| \left(\frac{\partial U}{\partial x}\right)^R \right| \right), & \text{otherwise} \end{cases}. \quad (4.2.10)$$

If the derivative changes sign across the mixed cell, then  $(\partial U/\partial x)^{lim}$  is set to zero. When the left and right derivatives (4.2.9) have the same sign, then the one with the smaller absolute value is chosen. We explore the accuracy and convergence using the minmod limiter (4.2.10) to determine  $\Delta U$  and compare it to Barth-Jespersen in Sec. 5.2.2 for linear problems and Sec. 5.2.3 for non-linear problems.

#### 4.2.2 Barth-Jespersen in One Dimension

An alternative approach to determine  $(\partial U/\partial x)^{lim}$  is based on the Barth-Jespersen algorithm [1]. We start from the general linear reconstruction in cell  $i + \frac{1}{2}$  given in (4.2.7). We impose two

conditions on the reconstruction. First, it must be exact for linear functions. Second, it should not create any new local extrema. We define a maximum and minimum value from cell-centered values in the neighbors of cell  $i + \frac{1}{2}$  as follows

$$U^{min} = \min_{k=\pm 1} \left( U_{i+\frac{1}{2}+k} \right) \quad \text{and} \quad U^{max} = \max_{k=\pm 1} \left( U_{i+\frac{1}{2}+k} \right), \quad (4.2.11)$$

then

$$U^{min} \leq U(x) \leq U^{max}, \quad x \in [x_i, x_{i+1}]. \quad (4.2.12)$$

Note that if all the cell-centered values are positive, then the reconstruction should produce a positive function.

The first step in the Barth-Jespersen reconstruction is to create an unlimited linear reconstruction using an unlimited derivative

$$\left( \frac{\partial U}{\partial x} \right)^{unl} = \frac{U_{i+\frac{3}{2}} - U_{i-\frac{1}{2}}}{0.5 \left( h_{i+\frac{3}{2}} + h_{i-\frac{1}{2}} \right) + h_{i+\frac{1}{2}}}. \quad (4.2.13)$$

With the unlimited derivative (4.2.13) we create an intermediate unlimited linear reconstruction

$$U^{unl}(x) = U_{i+\frac{1}{2}} + \left( \frac{\partial U}{\partial x} \right)^{unl} \left( x - x_{i+\frac{1}{2}} \right), \quad x \in [x_i, x_{i+1}]. \quad (4.2.14)$$

For the limited reconstruction (4.2.5) we represent the derivative as

$$\left( \frac{\partial U}{\partial x} \right)^{lim} = \Phi_{i+\frac{1}{2}} \left( \frac{\partial U}{\partial x} \right)^{unl} \quad (4.2.15)$$

where the limiter function  $\Phi_{i+1/2}$  is contained in the interval  $0 \leq \Phi_{i+1/2} \leq 1$ , and is chosen to satisfy our two conditions – the reconstruction is exact for linear functions and does not create any new local extrema. Because the reconstruction (4.2.7) is linear it is sufficient to enforce our conditions at the endpoints of the one dimension cell (vertices of the cell in two dimensions).

Let us consider the endpoint  $x_i$ . From the unlimited linear reconstruction (4.2.14) we have

$$U^{unl}(x_i) = U_{i+\frac{1}{2}} + \left( \frac{\partial U}{\partial x} \right)^{unl} \left( x_i - x_{i+\frac{1}{2}} \right), \quad (4.2.16)$$

and for the limited reconstruction we have

$$U^{lim}(x_i) = U_{i+\frac{1}{2}} + \Phi_{i+\frac{1}{2}} \left( \frac{\partial U}{\partial x} \right)^{unl} \left( x_i - x_{i+\frac{1}{2}} \right). \quad (4.2.17)$$

Manipulating (4.2.16) and replacing in (4.2.17) we obtain

$$U^{lim}(x_i) - U_{i+\frac{1}{2}} = \Phi_{i+\frac{1}{2}} \left( U^{unl}(x_i) - U_{i+\frac{1}{2}} \right). \quad (4.2.18)$$

Now, because  $0 \leq \Phi_{i+1/2} \leq 1$  we infer that if the unlimited function is increasing/decreasing in the center of cell  $i + \frac{1}{2}$  to the endpoint  $x_i$ , then the limited reconstruction is also increasing/decreasing.

Next, suppose

$$U^{unl}(x_i) - U_{i+\frac{1}{2}} > 0, \quad (4.2.19)$$

that is, the unlimited and limited functions are both increasing. We require, as one of our conditions, that the limited value at  $x_i$  be smaller than the maximum

$$U^{lim}(x_i) \leq U^{max}. \quad (4.2.20)$$

From (4.2.18) we find a sufficient bound on  $\Phi_{i+1/2}$  to ensure no new local extrema to be

$$\Phi_{i+\frac{1}{2}} \leq \frac{U^{max} - U^{lim}(x_i)}{U^{unl}(x_i) - U_{i+\frac{1}{2}}}. \quad (4.2.21)$$

In order for the reconstruction to preserve linear functions we have the more restrictive condition on  $\Phi_{i+1/2}$

$$\Phi_{i+\frac{1}{2}} \leq \min \left( 1, \frac{U^{max} - U^{lim}(x_i)}{U^{unl}(x_i) - U_{i+\frac{1}{2}}} \right). \quad (4.2.22)$$

A similar process can be applied to the case where

$$U^{unl}(x_i) - U_{i+\frac{1}{2}} < 0, \quad (4.2.23)$$

to find the condition

$$\Phi_{i+\frac{1}{2}} \leq \min \left( 1, \frac{U^{min} - U^{lim}(x_i)}{U^{unl}(x_i) - U_{i+\frac{1}{2}}} \right). \quad (4.2.24)$$

Finally, if

$$U^{unl}(x_i) - U_{i+\frac{1}{2}} = 0, \quad (4.2.25)$$

then we choose  $\Phi_{i+1/2} = 1$ . Summarizing,

$$\Phi_{i+\frac{1}{2}}^i = \begin{cases} \min \left( 1, \frac{U^{max} - U^{lim}(x_i)}{U^{unl}(x_i) - U_{i+\frac{1}{2}}} \right), & \text{if } U^{unl}(x_i) - U_{i+\frac{1}{2}} > 0 \\ \min \left( 1, \frac{U^{min} - U^{lim}(x_i)}{U^{unl}(x_i) - U_{i+\frac{1}{2}}} \right), & \text{if } U^{unl}(x_i) - U_{i+\frac{1}{2}} < 0 \\ 1, & \text{if } U^{unl}(x_i) - U_{i+\frac{1}{2}} = 0 \end{cases}. \quad (4.2.26)$$

To satisfy our conditions at all endpoints of the one dimensional cell we define

$$\Phi_{i+\frac{1}{2}} = \min_{k=0,1} \left( \Phi_{i+\frac{1}{2}}^{i+k} \right) \quad (4.2.27)$$

We use  $\Phi_{i+1/2}$  from (4.2.27) to compute the limited cell-centered derivative  $(\partial U / \partial x)^{lim}$  given in (4.2.15). Now, we have all the pieces necessary to compute the value of  $\Delta U$  shown in (4.2.6) and finally recover the values of  $U_L$  and  $U_R$  from (4.2.4). We explore the accuracy and convergence of the Barth-Jespersen as a mechanism for obtaining new material temperatures for linear and non-linear problems in Secs. 5.2.2 and 5.2.3 respectively.

## 5 Numerical Results

We test the MFD scheme on non-uniform one dimensional meshes for a variety of problems. In Sec. 5.1 we confirm the theoretical second-order convergence rate of the algorithm [4]. Also, we test to ensure that the method recovers linear and piecewise linear solutions exactly (to numerical double precision). In Sec. 5.2.2 we examine the convergence and accuracy of the limited linear reconstruction techniques to recover temperature values in mixed cells for linear diffusion. In Sec. 5.2.3 we perform a similar analysis on non-linear diffusion. Finally, in Sec. 5.3 we perform a numerical experiment where we propagate a heat wave across a material interface.

## 5.1 Convergence of Global Mimetic Finite Difference Method

For the numerical results throughout this section we take a test problem for the diffusion equation (2.1) on the unit interval  $\Omega = [0, 1]$ . We consider Dirichlet boundary conditions where

$$U_1 = 0 \quad \text{and} \quad U_N = 1, \quad (5.1.1)$$

so we take the discrete divergence  $\mathcal{D}$  to be 0 at  $x_1$  and  $x_N$ , in the discrete operator  $\Upsilon$  (2.2.21) we take  $\alpha_1 = \alpha_N = 1$ , and in the discrete right hand side  $\mathcal{F}^{n+1}$  (2.2.23) we take  $\psi_1 = 0$ ,  $\psi_N = 1$ . Also, we take  $a = 1/30$ .

### 5.1.1 Second-Order Convergence on Non-Uniform Mesh

We test the MFD scheme on a non-uniform mesh to ensure we obtain the second-order convergence of the method. For the convergence test we take the heat conductivity to be a constant  $k = 1/100$ . We use the method of manufactured solutions to create a test problem for the diffusion equation (2.1). The exact solution is prescribed as

$$u = \frac{1}{6}x^3. \quad (5.1.2)$$

For (5.1.2) to be a solution to the diffusion equation (2.1) the forcing function  $f$  must be

$$f = -kx. \quad (5.1.3)$$

To verify the second-order convergence of the algorithm we use the  $\infty$ -norm of the error. To compute the error we evaluate the exact solution (5.1.2) at the cell-centers and physical boundary. Tab. 1 gives the results of the convergence analysis. We obtain second-order convergence, if the number of mesh points is doubled the error is reduced by approximately a factor of four.

$N$	$\ u - U\ _\infty$	Rate
10	3.0066E-03	—
20	7.3227E-04	2.06
40	1.7964E-04	2.04

Table 1: SECOND-ORDER CONVERGENCE FOR MIMETIC FINITE DIFFERENCE SCHEME.

### 5.1.2 Linear and Piecewise Linear Solutions

The MFD scheme is second-order, a fact we verify in the previous section. Therefore the method should recover linear and piecewise linear solutions exactly, to numerical precision. First we verify this fact for a linear problem.

We take the heat conductivity to be a constant  $k = 1/100$ . The linear test problem has the linear steady-state solution

$$u = x. \quad (5.1.4)$$

As before, we compute the  $\infty$ -norm of the error between the exact solution (5.2.3) and the computed solution  $U$ . We verify the numerical double precision accuracy of the method to recover a linear solution for two values on  $N$ . We present the results for recovering a linear solution in Tab. 2(A).

Next we test the recovery of a piecewise linear solution. We take the heat conductivity to be piecewise constant

$$k = \begin{cases} k_L = 1/10, & \text{if } 0 \leq x < 0.5, \\ k_R = 1/100, & \text{if } 0.5 < x \leq 1. \end{cases} \quad (5.1.5)$$

The piecewise linear test problem has the steady-state solution

$$u = \begin{cases} \frac{2k_R}{k_L+k_R}x, & \text{if } 0 \leq x < 0.5, \\ \frac{2k_L}{k_L+k_R}x + \frac{k_R-k_L}{k_L+k_R}, & \text{if } 0.5 < x \leq 1. \end{cases} \quad (5.1.6)$$

In the case when  $k_L = k_R = k$ , the exact solution (5.1.6) is identical to the solution (5.1.4). The test problem with piecewise constant solution is solved on a non-uniform one dimensional mesh where we ensure a cell interface coincides with the material discontinuity at  $x = 0.5$ . The method is exact to numerical double precision and we present results in Tab. 2(B).

Linear Solution		Piecewise Linear Solution	
$N$	$\ u - U\ _\infty$	$N$	$\ u - U\ _\infty$
13	3.8304E-15	13	6.2728E-15
26	1.0325E-14	26	1.6764E-14

(A)
(B)

Table 2: THE MFD SCHEME RECOVERS LINEAR AND PIECEWISE LINEAR SOLUTIONS TO NUMERICAL DOUBLE PRECISION.

## 5.2 Numerical Tests on a Mixed Cell

In Sec. 5.1.2 we saw the MFD scheme recover piecewise linear solution exactly, to numerical double precision, if the material discontinuity is placed on a cell interface. Pursuant to the discussion in Sec. 4 we relax this requirement and allow the material discontinuity to lie somewhere in a cell. Recall we denoted a cell that contains a material interface as mixed. We test the effect the presence of a mixed cell has on the order of the method. We also test the convergence rate when we handle the mixed cells using the homogenization technique from Sec. 4.1.1. The limited linear reconstruction mechanism outlined in Secs. 4.2 is used to obtain new material temperatures.

For tests in this section we measure the error in our computation using the  $\infty$ -norm to test convergence. So we may compare our results to the computational results found in Kucharik [6] we also measure the error with the relative  $L_1$  norm

$$L_1 = \frac{1}{N_{\text{cells}}} \frac{\sum_{\text{cells}} |u - U|}{\sum_{\text{cells}} u}, \quad (5.2.1)$$

where  $u$  is the analytical solution evaluated at the cell centers and  $U$  is the computed solution. Note we omit the values at the boundary as they are exact, so the contribution to the overall error is zero.

### 5.2.1 Convergence Analysis of Homogenization Methods

We investigate the homogenization procedure and its effect on the order of the numerical method. To analyze the convergence we solve the linear diffusion equation (2.1) on the unit interval

$\Omega = [0, 1]$ . We again consider Dirichlet boundary conditions (5.1.1) and  $a = 1/30$ . We place a material interface at  $x = 0.6125$ . We take the heat conductivity to be

$$k = \begin{cases} k_L = 1/10, & \text{if } 0 \leq x < 0.6125, \\ k_R = 1/100, & \text{if } 0.6125 < x \leq 1. \end{cases} \quad (5.2.2)$$

The test problem has the analytical solution (5.1.6). The problem is solved up to time  $t_{final} = 300$  with a constant  $\Delta t = 1/100$ .

We present the convergence results for each homogenization method coupled with the *minmod* limited linear reconstruction in Tabs. 3 and the Barth-Jespersen reconstruction in Tabs. 4. All the methods have first order convergence properties. The results in Tab. 3(A), Tab. 3(C), Tab. 4(A), and Tab. 4(C) show if we use either the weighted arithmetic mean or weighted geometric mean as the homogenization technique and either limited linear reconstruction, then the errors and convergence rate is unaffected. We see from Tab. 3(B) and Tab. 4(B) that the choice of limiter is a factor is the size of the error for the weighted harmonic mean homogenization. The Barth-Jespersen limiter produces slightly smaller errors, but the convergence rate is identical to that of *minmod*.

The drop in the order of the method is possibly due to our imperfect knowledge of the location of the material interface. We only know that the interface lies somewhere in the interval  $[x_i, x_{i+1}]$  for the mixed cell  $e_{i+1/2}$ . The lack of foreknowledge of the interface location introduces an error on the order of  $\mathcal{O}(h_{i+1/2})$  for the mixed cell calculation. Thus, first order is likely the best we can obtain for a numerical method coupled with homogenization.

Arithmetic Mean			Harmonic Mean			Geometric Mean		
$N$	$\ u - U\ _\infty$	Rate	$N$	$\ u - U\ _\infty$	Rate	$N$	$\ u - U\ _\infty$	Rate
10	5.682E-02	—	10	3.505E-02	—	10	3.505E-02	—
20	2.970E-02	0.96	20	1.615E-02	1.06	20	1.861E-02	0.94
40	1.506E-02	0.99	40	7.867E-03	1.03	40	9.503E-03	0.98

(A)
(B)
(C)

Table 3: CONVERGENCE RESULTS FOR *minmod* LIMITED RECONSTRUCTION AND THE THREE HOMOGENIZATION TECHNIQUES.

Arithmetic Mean			Harmonic Mean			Geometric Mean		
$N$	$\ u - U\ _\infty$	Rate	$N$	$\ u - U\ _\infty$	Rate	$N$	$\ u - U\ _\infty$	Rate
10	5.682E-02	—	10	2.273E-02	—	10	3.505E-02	—
20	2.970E-02	0.96	20	1.077E-02	1.06	20	1.861E-02	0.94
40	1.506E-02	0.99	40	5.245E-03	1.03	40	9.503E-03	0.98

(A)
(B)
(C)

Table 4: CONVERGENCE RESULTS FOR BARTH-JESPERSEN LIMITED RECONSTRUCTION AND THE THREE HOMOGENIZATION TECHNIQUES.

### 5.2.2 Recover Intermediate Temperature Values - Linear Problem

Next we test the accuracy of the limited linear reconstruction mechanism to obtain intermediate temperatures. We solve the linear diffusion equation (2.1) on the unit interval  $\Omega = [0, 1]$  on a

mesh of 99 cells where the middle cell is twice the size of the remaining cells. Just as in Sec. 5.1 we take Dirichlet boundary conditions (5.1.1) and  $a = 1/30$ . We place a material interface at  $x = 0.5$  where the heat conductivity changes abruptly. We take the heat conductivity to be the piecewise constant given in (5.1.5). The test problem has the analytical solution (5.1.6). The problem is solved up to the final time  $t = 300$  with  $\Delta t = 1/100$ . The three homogenization techniques from Sec. 4.1.1 and the two limiting strategies *minmod* and Barth-Jespersen are compared.

We present the complete error analysis including the error in the recovered temperatures  $U_L$  and  $U_R$ , the  $\infty$ -norm over the whole mesh, the  $\infty$ -norm over the pure cells, and the relative  $L_1$  norm in Tab. 5. In general, for a given homogenization method, the error in the recovered temperature values  $U_L$  and  $U_R$  is larger for the *minmod* limited reconstruction than the Barth-Jespersen reconstruction. Overall the weighted harmonic mean produces the best results. Although the weighted arithmetic mean and the weighted geometric mean produce accurate results too. All three homogenization techniques work well to solve the diffusion equation on a mesh with a mixed cell. However, with any of the homogenization strategies the method loses the ability to recover a piecewise linear solution exactly, to numerical double precision.

Method	$U_L$ error	$U_R$ error	$\ \cdot\ _\infty$	$\ \cdot\ _\infty$ pure cells	$L_1$
Arithmetic, <i>minmod</i>	2.227E-03	4.925E-03	1.197E-02	1.197E-02	1.096E-04
Harmonic, <i>minmod</i>	5.909E-03	2.273E-03	5.909E-03	2.075E-12	2.698E-06
Geometric, <i>minmod</i>	3.415E-03	2.108E-03	7.560E-03	7.560E-03	6.969E-05
Arithmetic, Barth-Jesp	1.975E-03	4.674E-03	1.197E-02	1.197E-02	1.096E-04
Harmonic, Barth-Jesp	4.091E-03	4.091E-03	4.091E-03	2.075E-12	2.698E-06
Geometric, Barth-Jesp	2.754E-03	1.447E-03	7.560E-03	7.560E-03	6.926E-05

Table 5: ERROR ANALYSIS OF THE LIMITED LINEAR RECONSTRUCTION MECHANISM TO RECOVER TEMPERATURE VALUES IN A MIXED CELL.

The  $\infty$ -norm over the pure cells is included in Tab. 5 to quantify the effect the homogenization technique has on the computation of the discrete temperatures in the pure cells. Again, we see the weighted harmonic mean produces the best results. From Tab. 5 we see the weighted harmonic mean homogenization still recovers a piecewise linear solution exactly, to numerical double precision, in the pure cells. Also, comparing the relative  $L_1$  results from Tab. 5 with the  $L_1$  results presented in the report by Kucharik we find that the limited linear reconstruction mechanism to obtain intermediate temperatures outperforms any other mechanism discussed and tested [6].

Figs. 1 - 6 plot close-ups of the results of the piecewise linear test problem on the 99 cell mesh. As noted previously, the *minmod* limiting technique produces less accurate results than the more sophisticated Barth-Jespersen limiter. We verify this fact in the “eyeball” norm. Figs. 1 - 6 also show the effect the different homogenization techniques have on the computation of the discrete temperature in the pure cells. Generally, the weighted harmonic mean (4.1.1) produces more accurate results in the “eyeball” norm than either the weighted arithmetic mean (4.1.2) or the weighted geometric mean (4.1.3), just as we saw in the error analysis given in Tab. 5

Recover  $U_L$  and  $U_R$  in Mixed Cell using *minmod* and Weighted Arithmetic Mean

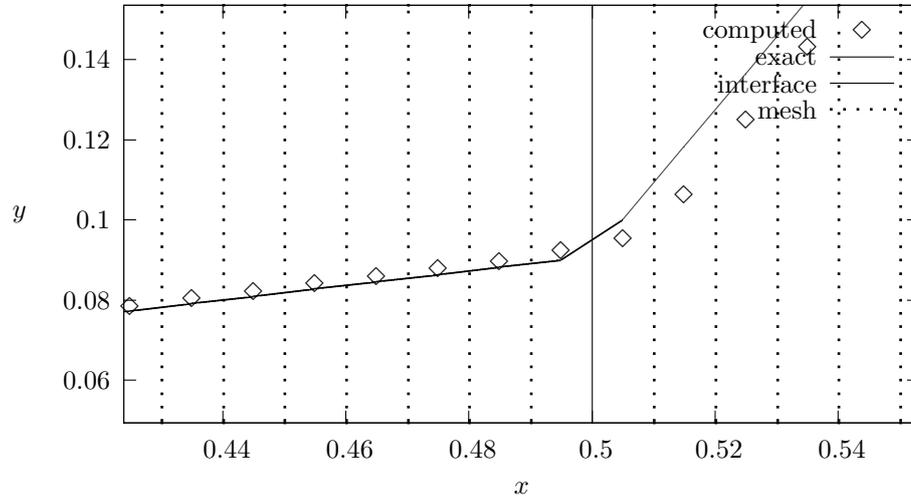


Figure 1: RECONSTRUCTED TEMPERATURE VALUES IN MIXED CELL USING WEIGHTED ARITHMETIC MEAN AND *minmod* LIMITER. THE MESH IS THE OVERLAY OF DOTTED LINES.

Recover  $U_L$  and  $U_R$  in Mixed Cell using Barth-Jespersen and Weighted Arithmetic Mean

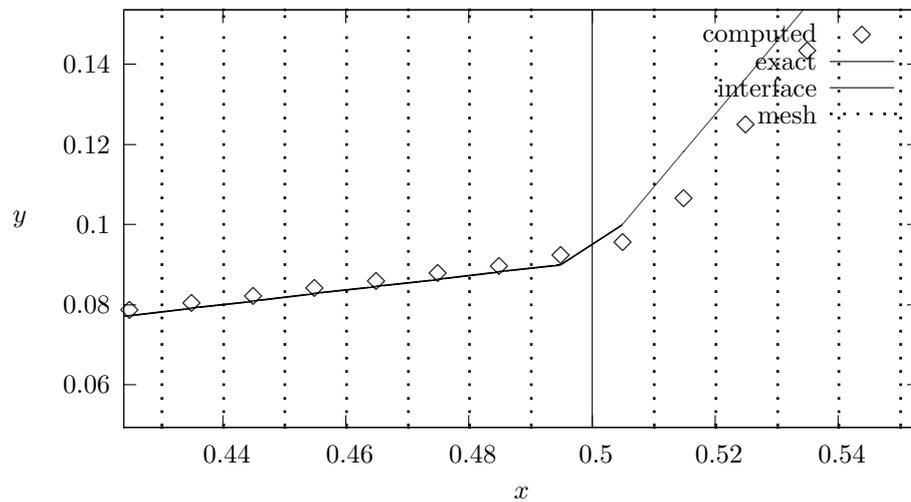


Figure 2: RECONSTRUCTED TEMPERATURE VALUES IN MIXED CELL USING WEIGHTED ARITHMETIC MEAN AND BARTH-JESPERSEN LIMITER. THE MESH IS THE OVERLAY OF DOTTED LINES.

Recover  $U_L$  and  $U_R$  in Mixed Cell using *minmod* and Weighted Harmonic Mean

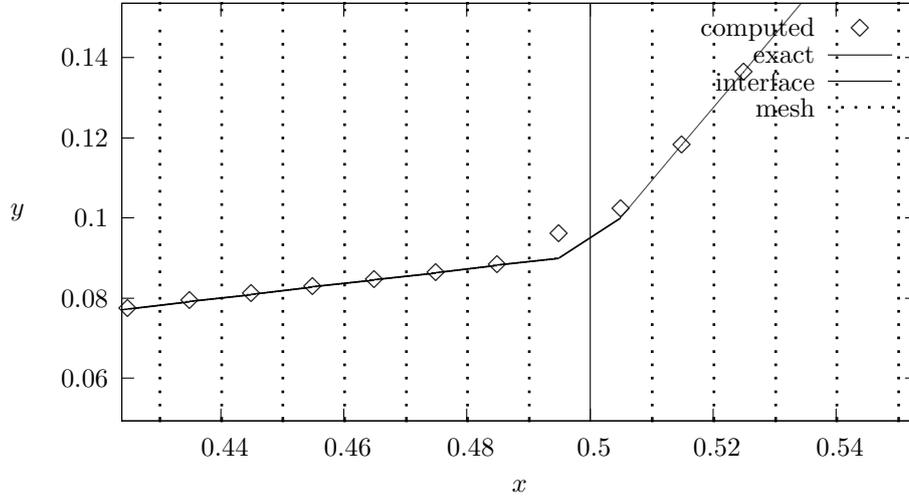


Figure 3: RECONSTRUCTED TEMPERATURE VALUES IN MIXED CELL USING WEIGHTED HARMONIC MEAN AND *minmod* LIMITER. THE MESH IS THE OVERLAY OF DOTTED LINES.

Recover  $U_L$  and  $U_R$  in Mixed Cell using Barth-Jespersen and Weighted Harmonic Mean

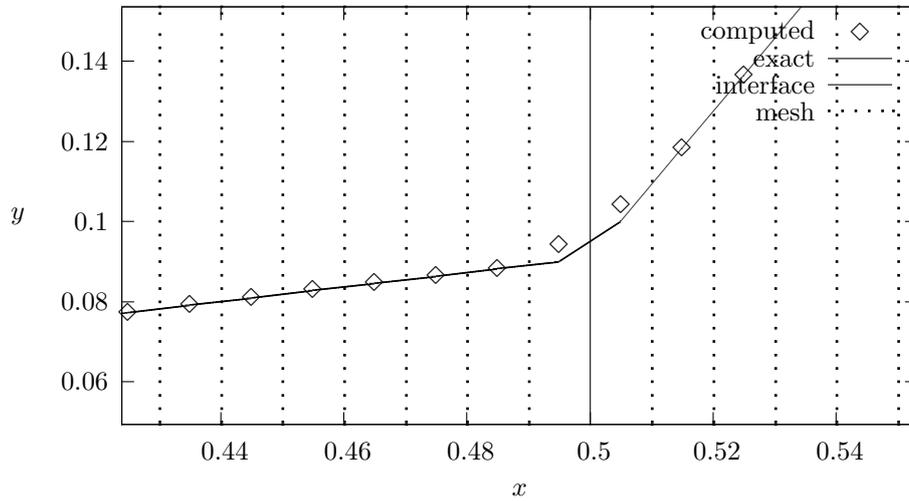


Figure 4: RECONSTRUCTED TEMPERATURE VALUES IN MIXED CELL USING WEIGHTED HARMONIC MEAN AND BARTH-JESPERSON LIMITER. THE MESH IS THE OVERLAY OF DOTTED LINES.

Recover  $U_L$  and  $U_R$  in Mixed Cell using *minmod* and Weighted Geometric Mean

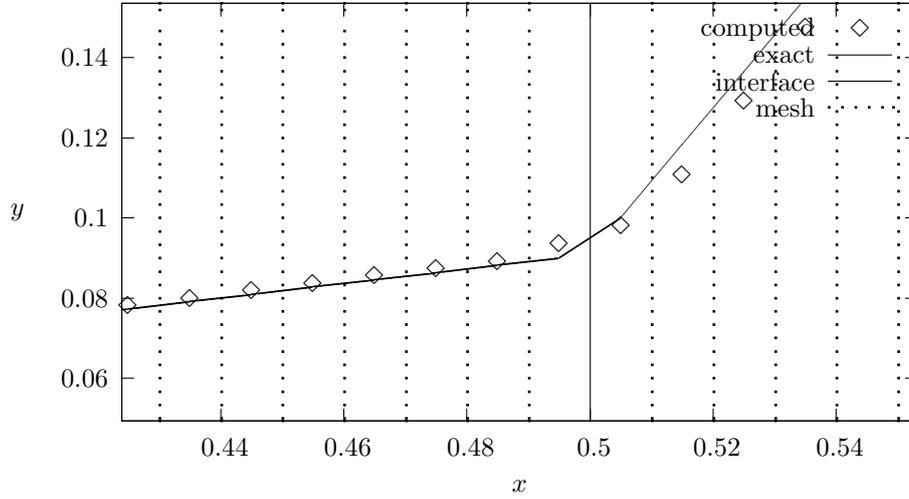


Figure 5: RECONSTRUCTED TEMPERATURE VALUES IN MIXED CELL USING WEIGHTED GEOMETRIC MEAN AND *minmod* LIMITER. THE MESH IS THE OVERLAY OF DOTTED LINES.

Recover  $U_L$  and  $U_R$  in Mixed Cell using Barth-Jespersen and Weighted Geometric Mean

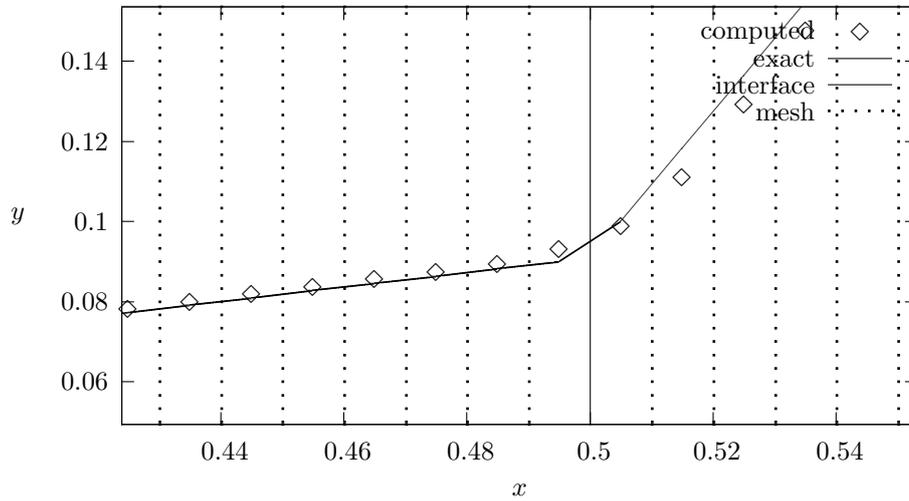


Figure 6: RECONSTRUCTED TEMPERATURE VALUES IN MIXED CELL USING WEIGHTED GEOMETRIC MEAN AND BARTH-JESPERSON LIMITER. THE MESH IS THE OVERLAY OF DOTTED LINES.

### 5.2.3 Recover Intermediate Temperature Values - Non-Linear Problem

We investigate the accuracy of using a limited linear reconstruction technique, both *minmod* and Barth-Jespersen, to recover intermediate heat values in a mixed cell for the non-linear heat equation (3.7). To perform the numerical tests we use the analytical solution

$$u(x, t) = \begin{cases} \left(\frac{\alpha D}{k} [Dt + s + x]\right)^{\frac{1}{\alpha}}, & \text{if } x > -Dt - s, \\ 0, & \text{if } x < -Dt - s. \end{cases} \quad (5.2.3)$$

The function (5.2.3) is a heat wave which solves the non-linear heat equation (3.7) everywhere except at the point  $x = -Dt - s$  where the derivatives are not defined. This point corresponds to the location of the head of the heat wave. The domain is taken to be  $\Omega = [0, 6.2]$ . In the computational experiments we use the analytical solution (5.2.3) evaluated at  $t = 0$  as our initial condition and the analytical solution evaluated at the endpoints of the one dimensional domain  $u(x_1, t)$  and  $u(x_N, t)$  as the Dirichlet boundary conditions for the system (3.7). For all the tests in this section we measure the error in our computation using the relative  $L_1$  norm given in (5.2.1). The parameters in the analytical solution (5.2.3) are taken to be

$$\alpha = \frac{5}{2}, D = 5, s = 1, k = \frac{1}{1000}. \quad (5.2.4)$$

We solve the non-linear heat equation (3.7) on two meshes: a uniform mesh with 62 cells and a mesh of 61 cells where the middle cell is twice as big as all other cells. On the second mesh we artificially split the mesh into two materials. This is only a formality as the material properties are the same in every cell. The split is performed because we are unable to construct the analytical solution for a non-smooth conductivity coefficient. So even though material properties are the same we use the limited linear reconstruction to recover the non-linear solution in a mixed cell. We solve the problem up to time  $t_{final} = 0.8$  with  $\Delta t = 1/10000$ . The solution on the mesh with 62 cells does not have a mixed cell and is used as a baseline case.

In Tab. 6 we present a comparison of the relative  $L_1$  errors of the baseline, pure cell case and all the different methods for homogenization and linear reconstruction. We see that all the errors are comparable, a direct result of the constant conductivity coefficient. We account for the slightly smaller error in the baseline, pure cell computation as a combination of the method calculating on a pure cell mesh and the mesh contained one more cell than the other computations, so the error is scaled by a slightly smaller number, as seen in (5.2.1). We compare

Method	$L_1$ error
Baseline, Pure Cell	2.17026E-06
Arithmetic Mean, <i>minmod</i>	2.62244E-06
Harmonic Mean, <i>minmod</i>	2.62244E-06
Geometric Mean, <i>minmod</i>	2.62244E-06
Arithmetic Mean, Barth-Jespersen	2.62244E-06
Harmonic Mean, Barth-Jespersen	2.62244E-06
Geometric Mean, Barth-Jespersen	2.62244E-06

Table 6: COMPARISON OF THE RELATIVE  $L_1$  ERROR FOR EACH METHOD OF RECOVERING TEMPERATURE VALUES.

our results in Tab. 6 to the results for an identical test problem in the technical report by Kucharik [6]. Again, we find that the limited linear reconstruction mechanism for obtaining new material temperatures for a non-linear problem is more accurate than all the methods described in [6].

In Figs. 7 - 13 we plot the results for the numerical experiments. In each of the figures the diamonds represent the computed solution and the solid line represents the exact solution. The overlay of vertical lines (solid in Fig. 7 or dotted in Figs. 8 - 13) indicate the cell interfaces. Fig. 7 shows the baseline, pure cell case where we have a mesh of 62 pure cells and computed the solution up to a final time  $t = 0.8$ . The remaining figures are all calculated on a 61 cell mesh where the middle cell is twice as large as the other cells.

Fig. 8 and 9 show a close-up of the linear reconstruction for the temperature on either side of the interface for *minmod* and Barth-Jespersen limiters using the weighted arithmetic mean. Fig. 10 and 11 show the linear reconstruction on either side of the interface for both types of limiters using the weighted harmonic mean. Finally, Fig. 12 and 13 show the results for both types of reconstruction using the weighted geometric mean. Note that because the material is actually homogeneous and the interface is artificial. The three homogenization strategies (4.1.1)-(4.1.3) yield the same result. Each weighted mean collapses to the conductivity constant  $k$  in the mixed cell. The difference between using the *minmod* limiter and Barth-Jespersen limiter in these computational experiments is very subtle, usually somewhere in the third decimal place. So the results for the two linear reconstruction strategies look identical in the “eyeball” norm.

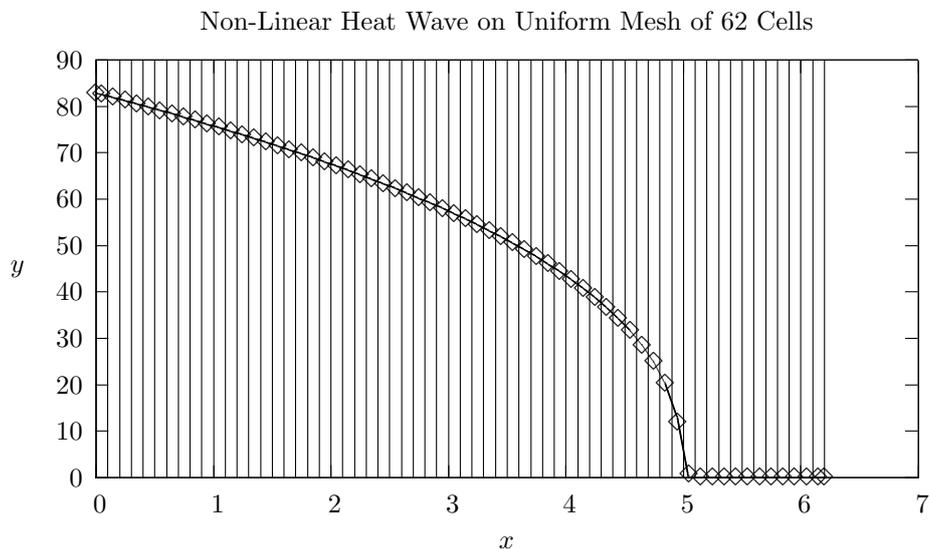


Figure 7: THE BASELINE, PURE CELL CASE WHERE THE ARTIFICIAL MATERIAL INTERFACE ALIGNS WITH A CELL INTERFACE.

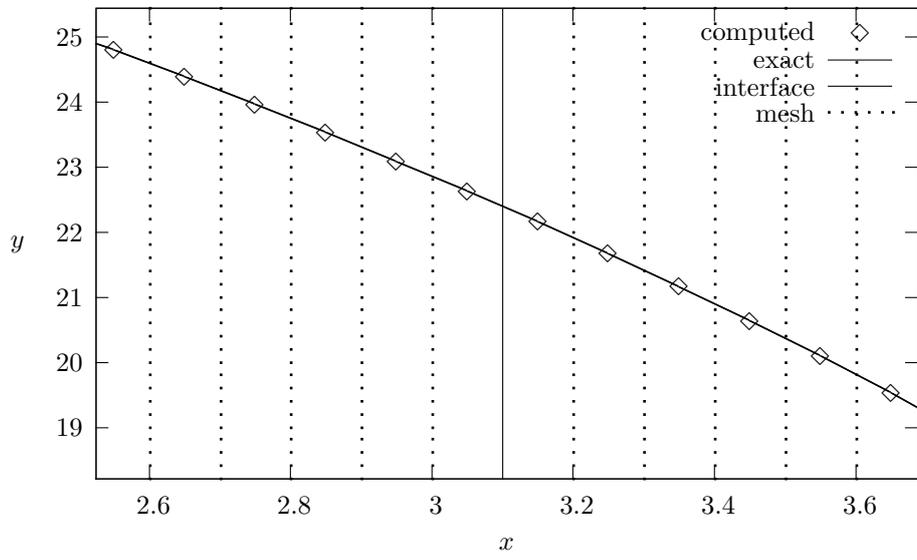


Figure 8: RECONSTRUCTED TEMPERATURE VALUES IN MIXED CELL USING WEIGHTED ARITHMETIC MEAN AND *minmod* LIMITER.

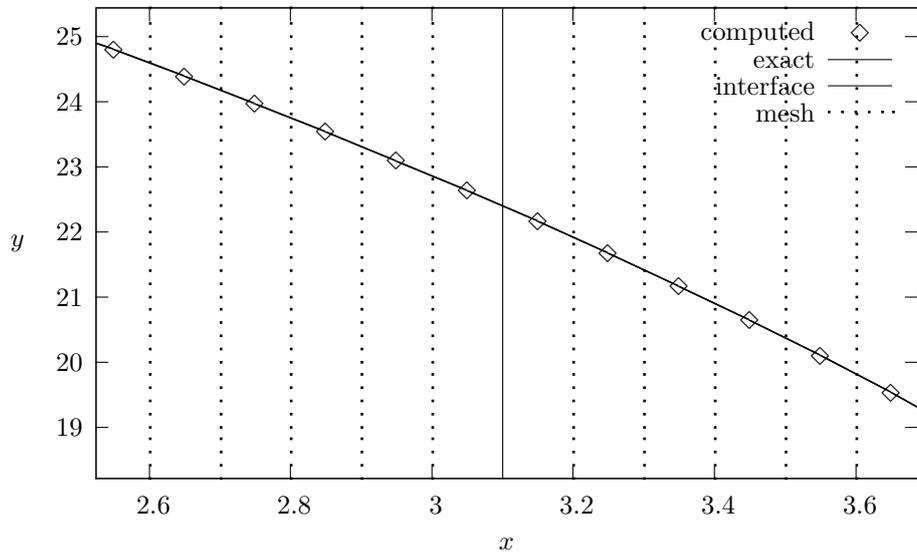


Figure 9: RECONSTRUCTED TEMPERATURE VALUES IN MIXED CELL USING WEIGHTED ARITHMETIC MEAN AND BARTH-JESPERSON LIMITER.

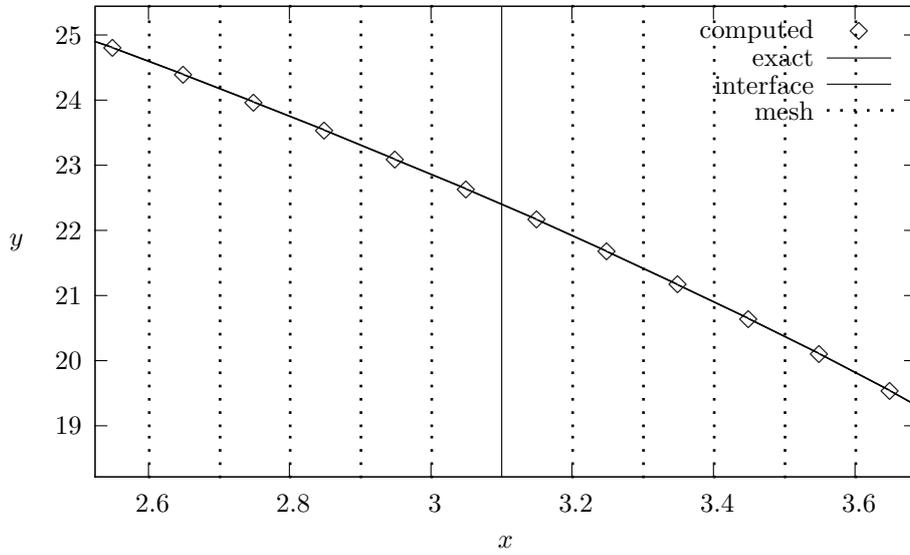


Figure 10: RECONSTRUCTED TEMPERATURE VALUES IN MIXED CELL USING WEIGHTED HARMONIC MEAN AND *minmod* LIMITER.

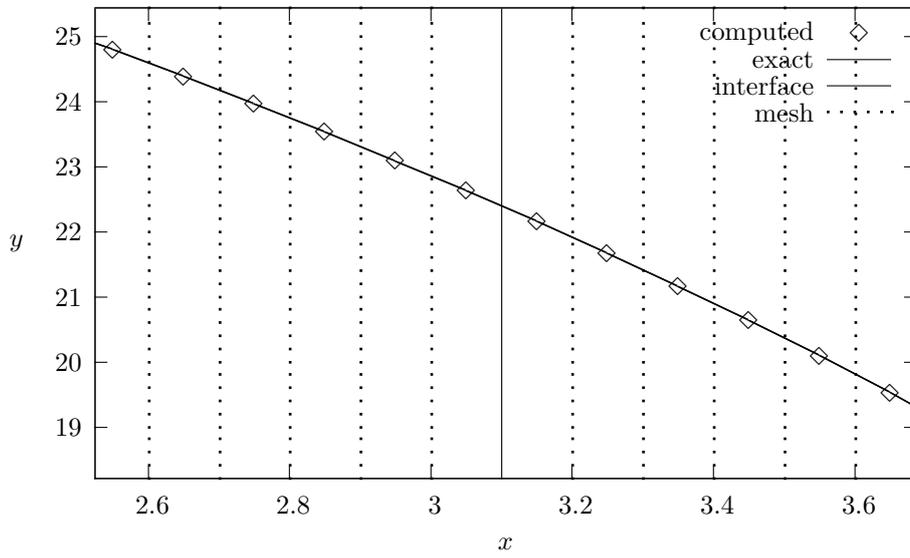


Figure 11: RECONSTRUCTED TEMPERATURE VALUES IN MIXED CELL USING WEIGHTED HARMONIC MEAN AND BARTH-JESPERSON LIMITER.

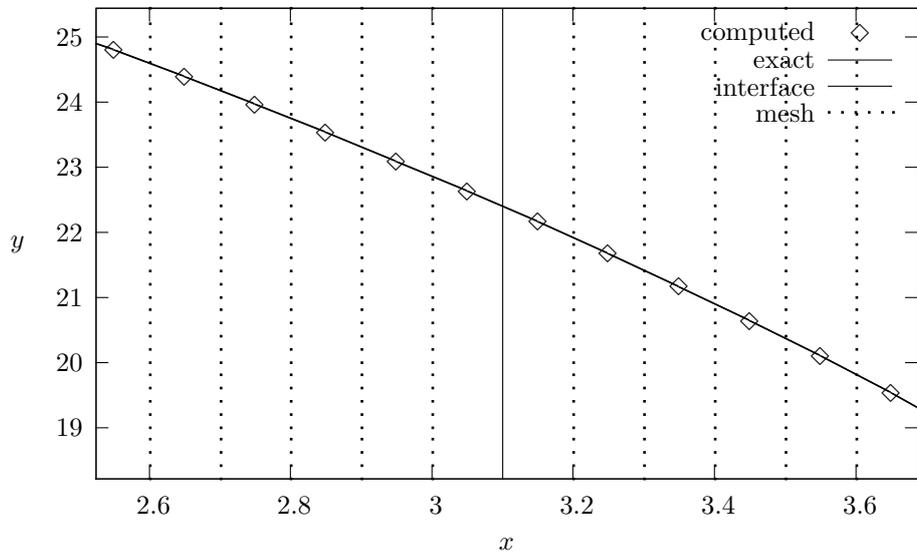


Figure 12: RECONSTRUCTED TEMPERATURE VALUES IN MIXED CELL USING WEIGHTED GEOMETRIC MEAN AND *minmod* LIMITER.

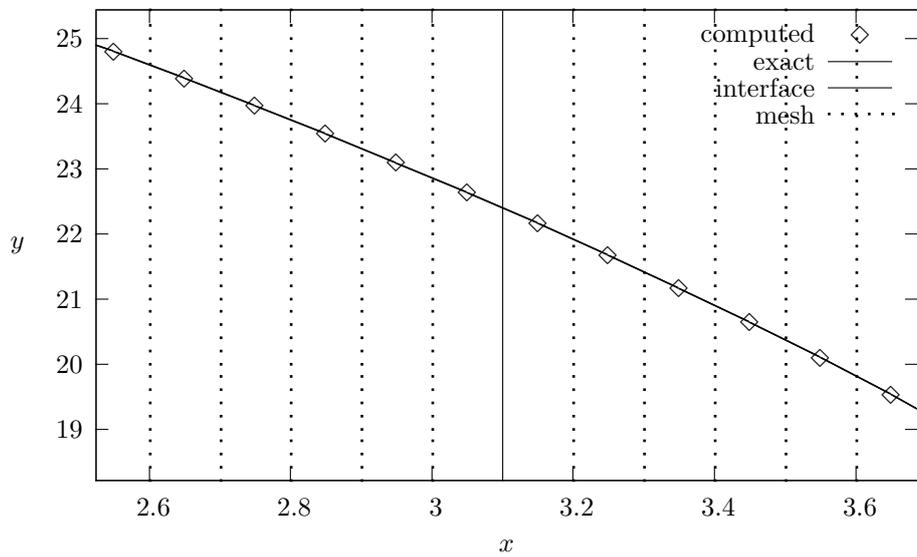


Figure 13: RECONSTRUCTED TEMPERATURE VALUES IN MIXED CELL USING WEIGHTED GEOMETRIC MEAN AND BARTH-JESPERSEN LIMITER.

### 5.3 Non-Linear Heat Wave in Multiple Materials

As a numerical experiment we use the MFD scheme for non-linear problems given in (3.9) to propagate the heat wave (5.2.3) across a material interface. We take the parameters for the wave to be  $D = 5$ ,  $s = 1$ ,  $\alpha = 5/2$ . In Fig. 14 we plot the result of propagating the heat wave across a single material as a benchmark. For the calculation in Fig. 14 we take  $k = 1/100$ . We solve

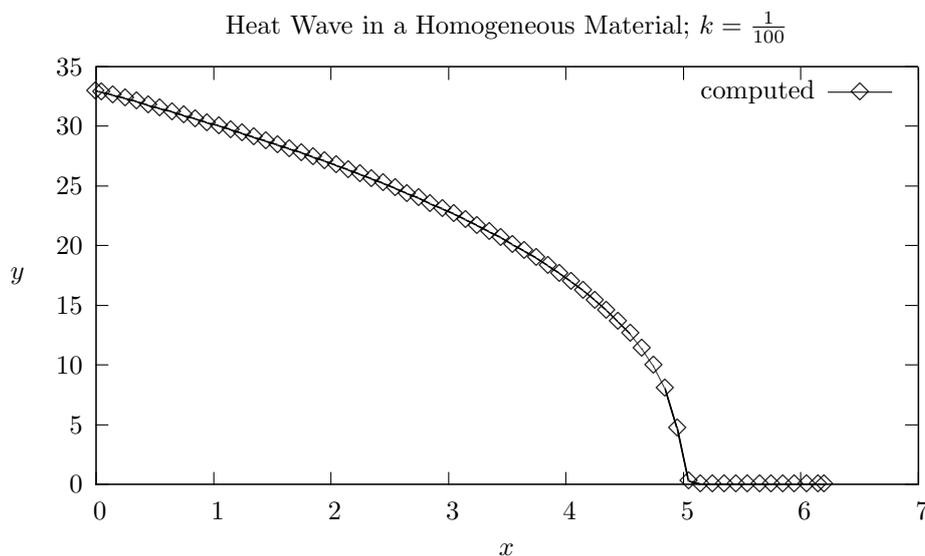


Figure 14: BENCHMARK HEAT WAVE PROPAGATES THROUGH A HOMOGENEOUS MATERIAL WITH  $k = 1/100$ .

the non-linear heat equation (3.7) on a mesh of 61 cells where the middle cell is twice as big as all other cells.

The plot in Fig. 15 presents the result of the numerical experiment where we propagate the heat wave across a material interface located at  $x = 3.1$ . For the experiment we take

$$k = \begin{cases} \frac{1}{100} & \text{if } x < 3.1 \\ \frac{1}{1000} & \text{if } x > 3.1 \end{cases} . \quad (5.3.1)$$

The numerical experiment shown in Fig. 15 does not have an analytical solution (that we are aware of) so it is presented as a qualitative experiment with which we compare the result Fig. 14.

As the heat wave interacts with the material interface, where the diffusion coefficient shrinks by a factor of 10, the amplitude of the heat wave raises considerably compared to the original amplitude in Fig. 14. The head of the heat wave continues to slowly propagate in the second material. The tail of the heat wave is still located in the first material and continues diffusing at the faster rate allowed by the first material. Essentially the tail is trying to ‘catch-up’ to the head of the heat wave; however, the tail slows down once it interacts with the material interface. The continuing interaction of the tail with the material interface causes a continuing rise in the amplitude.

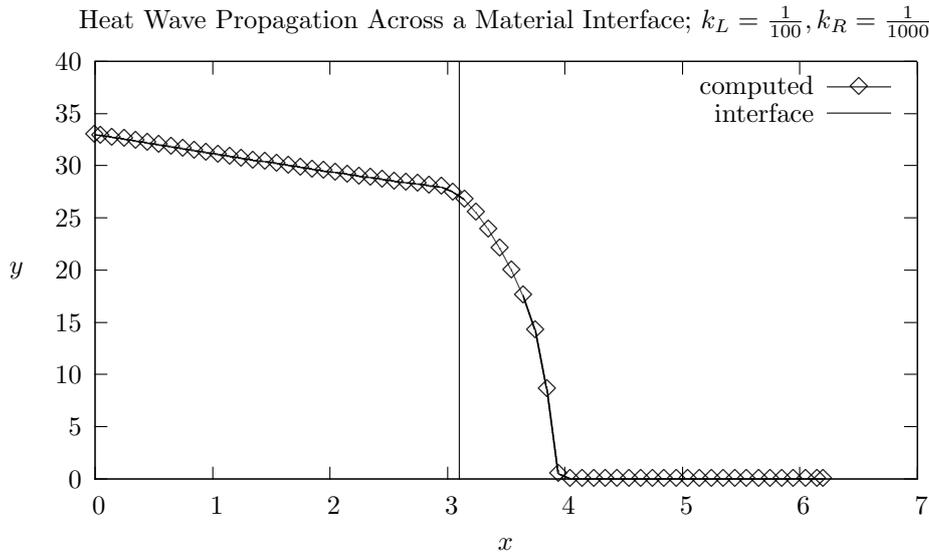


Figure 15: A HEAT WAVE PROPAGATES ACROSS A MATERIAL INTERFACE LOCATED AT  $x = 3.1$ .

## 6 Conclusions & Future Work

We used the support operators method and derived a numerical algorithm to solve the diffusion equation in one spatial dimension. It was shown that the resulting mimetic finite difference scheme satisfied a discrete version of the divergence theorem. The discrete operators mimic the negative adjoint property of the continuum flux and divergence operators. We showed that the product of the discrete divergence operator and discrete flux operator is symmetric positive definite, just as in the continuum. Finally, we ensured that the null space of the discrete flux matched the null space of the continuum flux operator.

We gave a brief overview of the process to extend the algorithm to solve the non-linear heat equation. We introduced the concepts of homogenization and condensation to extend the MFD scheme to handle meshes where material interfaces may not coincide with cell interfaces. Once the method could handle mixed cells the question arose of how to recover temperature values in the pure parts of the mixed cell. In answering this question we avoided introducing extra degrees of freedom in the matrix equations of the MFD scheme. Instead we outlined a limited linear reconstruction mechanism to recover the unknown temperatures. We discuss extensions of homogenization, condensation and a limited linear reconstruction to higher spatial dimensions.

The analysis of Secs. 2 - 4 was followed by a battery of numerical results. We verified the predicted second-order convergence rate of the MFD scheme. Also, we checked to ensure that the algorithm recovers linear and piecewise linear solution exactly, to numerical double precision, provided any material discontinuities coincide with cell interfaces.

We investigated the effect mixed cells have on the convergence rate of the method. We found the homogenization techniques tested drop the method from second-order to first-order. We analyzed how accurately the limited linear reconstruction mechanism recovered intermediate temperatures in mixed cells. From the numerical experiments, on linear and non-linear diffusion, we found the Barth-Jespersen limiter produced more accurate results than the *minmod* limiter overall. Additionally, the weighted harmonic mean homogenization left the calculation in the

pure cells of the mesh untouched. That is, the method was able to exactly, to numerical double precision, recover the piecewise linear solution in the pure cells; the error is dominated by the mixed cell. We compared the limited linear reconstruction mechanism results against previous results for other methods of recovering intermediate temperature values. We found our method outperformed the others in every case.

Lastly, we performed a qualitative numerical experiment propagating a heat wave across a material interface. In the absence of an analytical solution we check the computed solution against our physical intuition of how the heat wave should behave as it interacts with the material interface.

For future work, one would implement the local condensation method and compare the numerical results with the homogenization techniques. The implementation of the local method would involve rewriting several subroutines in our code. A bulk of the changes would occur in the routines which assemble the coefficient matrix  $\mathcal{S} + \mathcal{D}^\dagger \mathcal{M} \mathcal{D}$  and the right hand side vector  $\mathcal{D}^\dagger \mathcal{M} \Upsilon^{-1} \mathcal{F}^{n+1}$ . We would require a general mechanism to detect the location of a mixed cell and alter the operators  $\mathcal{S}$  and  $\mathcal{D}^\dagger \mathcal{M}$  accordingly. We present the full derivation of the local method in Sec. 4.1.2, but we ran out of time this Summer to implement it.

## Acknowledgments

The author thanks Dr. Mikhail Shashkov and Matt Jemison for fruitful discussions on the support operators method, mixed cell techniques, and limiters.

## References

- [1] T. J. Barth. Numerical methods for gasdynamics systems on unstructured grids. In *Lecture Notes in Computational Science and Engineering, "An Introduction to Recent Developments in Theory and Numerics for Conservation Laws, Proceedings of the International School on Theory and Numerics for Conservation Laws*, pages 195–285, Freiburg/Littenweiler, 1997. Springer Berlin.
- [2] M. Holec. *Modeling of Diffusive Problems in Laser Plasma*. PhD thesis, Czech Technical University, 2012.
- [3] J. Hyman. Numerical methods for tracking interfaces. *Physica D: Nonlinear Phenomena*, 12(1 - 3):396 – 407, 1984.
- [4] J. Hyman, M. Shashkov, and S. Steinberg. The numerical solution of diffusion problems in strongly heterogeneous non-isotropic materials. *Journal of Computational Physics*, 132(1):130 – 148, 1997.
- [5] J. Hyman, M. Shashkov, and S. Steinberg. The effect of inner products for discrete vector fields on the accuracy of mimetic finite difference methods. *Computers & Mathematics with Applications*, 42(12):1527 – 1547, 2001.
- [6] M. Kucharik. Comparison of multi-material heat conductivity models. Technical report, Los Alamos National Laboratory, (unpublished), 2012.
- [7] Y. Kuznetsov. Mixed finite element method for diffusion equations on polygonal meshes with mixed cells. *Journal of Numerical Mathematics*, 14:305–315, 2006.

- [8] Y. Kuznetsov, K. Lipnikov, and M. Shashkov. The mimetic finite difference method on polygonal meshes for diffusion-type problems. *Computational Geosciences*, 8:301–324, 2004.
- [9] L. G. Margolin and M. Shashkov. Second-order sign-preserving remapping on general grids. Technical report, Los Alamos National Laboratory, LAUR-02-525, 2003.
- [10] L. G. Margolin and T. F. Tarwater. A diffusion operator for Lagrangian meshes. Technical report, Lawrence Livermore National Laboratory, LAUR-02-525, 1986.
- [11] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Springer, 2007.
- [12] M. Shashkov and S. Steinberg. Solving diffusion equations with rough coefficients in rough grids. *Journal of Computational Physics*, 129:383–405, 1996.
- [13] P. K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, 21(5):995–1011, 1984.

**Advanced Cell-Centered Hydro  
Methods**

**(Nathaniel Morgan, mentor)**

# Final Report - Los Alamos National Laboratory Computational Physics Summer Student Workshop

Student: Tyler Lung  
PhD Pre-candidate  
Department of Aerospace Engineering  
University of Michigan  
tblung@umich.edu

Graduate Advisor: Professor Phil Roe  
Department of Aerospace Engineering  
University of Michigan  
philroe@umich.edu

Mentor: Dr. Nathaniel Morgan  
XCP-8: Verification and Analysis  
Los Alamos National Lab  
nmorgan@lanl.gov

August 17, 2012

## Abstract

The numerical solution of highly compressible, multi-material flows is an ongoing research area. These types of flows can be solved with a Lagrangian type mesh which moves with the material in a simulation to allow precise material interface tracking. Currently, researchers at Los Alamos National Laboratory and elsewhere are investigating cell-centered Lagrangian algorithms with the aim of producing methods that have second-order accuracy, preserve symmetry, and do not generate spurious vorticity. The new cell-centered algorithms solve a Riemann-like problem at the vertex of a cell. Professor Phil Roe at the University of Michigan has proposed a new structure for Lagrangian hydrodynamic algorithms that does not rely on the solution of the Riemann problem. The new approach utilizes Flux Corrected Transport (FCT) and it implements a form of vorticity control. The first step in the development of this method has been to construct an algorithm that solves the acoustic equations on an Eulerian mesh. The algorithm, which builds on the work of Morton and Roe [1], calculates fluxes at cell vertices, attains second-order accuracy using FCT, and has the special property of preserving vorticity. Results are presented that confirm the second order accuracy of the scheme and the vorticity preserving properties. The results are compared to the output produced by a MUSCL-Hancock algorithm. Some discussion of limiting methods for the FCT algorithm is also given.

# 1 Introduction

The development of improved algorithms for the simulation of compressible multi-material flows is of high importance to Los Alamos National Laboratory. The algorithms form the basis for production computer codes which are heavily relied upon to successfully carry out the lab's mission. The long term goal of the research presented here is to develop a new Lagrangian compressible flow algorithm that has three distinguishing characteristics:

1. Uses Flux Corrected Transport (FCT) to achieve second order accuracy while incorporating multi-dimensional physics.
2. Does not rely on the solution of the traditional one dimensional Riemann problem.
3. Implements a form of vorticity control.

This is an ambitious goal, and in the opinion of the author and his graduate advisor, it would be foolish to jump straight to an effort to implement a multi-dimensional Lagrangian algorithm. A better procedure is to start with simple implementations of the ideas stated above and make jumps in complexity as ideas and methods are proven and understood. The first step in this process was implementing one dimensional linear advection solvers of FCT and MUSCL-Hancock types on Eulerian meshes and comparing their performance. The results obtained showed that the FCT algorithm matched or exceeded the performance of the MUSCL-Hancock algorithm in all areas. Next, one dimensional solvers for the acoustic equations were implemented in both the FCT and MUSCL-Hancock flavors, again on Eulerian meshes. The results here were sufficiently encouraging to warrant the extension of the algorithms to two dimensions. This work was begun about one month prior to the start of the 2012 Computational Physics Summer Workshop at LANL and continued for its duration. As such, the 2D vorticity preserving FCT algorithm for the acoustic equations is the focus of this report.

It is worth discussing the impetus for developing an Eulerian solver for the acoustic equations. After all, it has already been stated that the ultimate goal of this research is to produce a Lagrangian algorithm. Such an algorithm must incorporate mesh movement and non-linear physics, two things which are not present in the acoustic solver. The key here is to realize that the acoustic equations are a system of partial differential equations that exhibit symmetric wavespeeds. This is a characteristic they share with the Lagrangian equations of gas dynamics. The plan is to exploit this trait when developing a flux limiting procedure for the Lagrangian algorithm. The acoustic equations provide a simplified development environment for such a limiting procedure.

## 2 Vorticity Preserving Flux Corrected Transport Algorithm

The numerical method presented in the following section builds on the work of Morton and Roe [2], which was published in 2001. In that paper, the authors present a Rotated Richtmeyer scheme which is formulated using corner fluxes. The Rotated Richtmeyer scheme is a two dimensional variant of the Lax-Wendroff method. By computing and storing fluxes at cell vertices, Morton and Roe were able to produce a scheme that, by construction, guarantees that the discrete vorticity

$$\zeta = \delta_x \mu_y v - \delta_y \mu_x u \tag{1}$$

is preserved.<sup>1</sup> This fits the physics of the acoustic equations perfectly provided that the background state of the fluid of interest has a uniform density. Note that if the method were extended to

---

<sup>1</sup>See Appendix A for a definition of the compact differencing operators used in equations throughout this document.

solve a system of governing equations that allow vorticity to evolve in time, a vorticity correction mechanism would need to be developed. There are benefits beyond the vorticity preserving property when working in terms of nodal fluxes. This eliminates the need to solve traditional one dimensional Riemann problems at cell interfaces as is customary in Godunov based finite volume methods. There are two main reasons why we seek to distance ourselves from using a Riemann solver to compute fluxes. First, strictly speaking, the Riemann problem is one dimensional in nature. In this research, emphasis is placed on making the new algorithm as multi-dimensional as possible. Second, when considering Lagrangian applications, the use of Riemann solutions at cell interfaces forces one to relate face velocities around a node to a unique nodal velocity. This can be a difficult task and it is one that we wish to avoid. By storing fluxes at vertices, the unique velocity of each node is already defined.

The ideas of Flux Corrected Transport, originally developed by Boris and Book [2] were chosen to achieve second order accuracy. This was due to the multi-dimensional and adaptive limiting framework that it provides. An FCT algorithm computes a first order solution which is known to be free of spurious oscillations and then uses that information to relax limiting constraints where it is appropriate. This stands in contrast to other limiting schemes which apply limiting whenever a new maximum or minimum could develop without giving a thought to whether the extremum should exist or not. The first order scheme used here is equivalent to the First Order Upwind Method, but it does not require explicit upwinding. The high order scheme used to define antidiffusive fluxes is the Rotated Richtmeyer scheme, mentioned previously.

## 2.1 Problem Statement

In two dimensions the acoustic equations read

$$p_t + \rho_0 a_0^2 (u_x + v_y) = 0 \quad (2)$$

$$u_t + \frac{1}{\rho_0} p_x = 0 \quad (3)$$

$$v_t + \frac{1}{\rho_0} p_y = 0 \quad (4)$$

Note that  $u, v$ , and  $p$  represent perturbations to a fluid with a uniform background state described by  $u_0 = 0, v_0 = 0$  and  $p_0 = \text{constant}$ . Zero subscripts always denote a property of the background state. For example,  $\rho_0$  and  $a_0$  give the density and sound speed of the background state, respectively.

Assume a structured quadrilateral mesh with  $\Delta x = \Delta y = h$ . Define the timestep as  $k$ . The conserved variables  $\mathbf{u} = (p, u, v)^T$  are stored at cell centers and the fluxes  $\mathbf{f} = (\rho_0 a_0^2 u, 0, \rho_0^{-1} p)^T$  and  $\mathbf{g} = (0, \rho_0 a_0^2 v, \rho_0^{-1} p)^T$  are stored at vertices.

## 2.2 Numerical Recipe

A step by step numerical recipe for the Vorticity Preserving Flux Corrected Transport Algorithm (VPFCT) will now be given.

### 2.2.1 Compute Low Order Solution

1. Calculate the fluxes at each vertex from adjacent cell centered values.

$$\hat{\mathbf{f}} = \mu_x \mu_y \mathbf{f} \quad (5)$$

$$\hat{\mathbf{g}} = \mu_x \mu_y \mathbf{g} \quad (6)$$

2. Evolve each vertex flux through a partial time step.

$$\hat{\mathbf{F}} = \hat{\mathbf{f}} - \frac{\nu h}{2k} \begin{pmatrix} \delta_x \mu_y u + \delta_y \mu_x v \\ 0 \\ \delta_x \mu_y p \end{pmatrix} \quad (7)$$

$$\hat{\mathbf{G}} = \hat{\mathbf{g}} - \frac{\nu h}{2k} \begin{pmatrix} 0 \\ \delta_x \mu_y u + \delta_y \mu_x v \\ \delta_y \mu_x p \end{pmatrix} \quad (8)$$

$$\nu \equiv \frac{a_0 k}{h} \quad (9)$$

3. Loop over cells to update.

$$\mathbf{u}^* = \mathbf{u}^n - \frac{k}{h} (\delta_x \mu_y \hat{\mathbf{F}} + \delta_y \mu_x \hat{\mathbf{G}}) \quad (10)$$

### 2.2.2 Compute High Order Solution

1. Define antidiffusive fluxes such that  $\widehat{\mathbf{F}}^{\text{AD}*} = \widehat{\mathbf{F}}^{\text{H}*} - \widehat{\mathbf{F}}^{\text{L}*}$  and  $\widehat{\mathbf{G}}^{\text{AD}*} = \widehat{\mathbf{G}}^{\text{H}*} - \widehat{\mathbf{G}}^{\text{L}*}$ .

$$\widehat{\mathbf{F}}^{\text{AD}*} = \frac{a_0(\nu^2 - \nu)}{2\nu} \begin{pmatrix} \delta_x \mu_y u^* + \delta_y \mu_x v^* \\ 0 \\ \delta_x \mu_y p^* \end{pmatrix} \quad (11)$$

$$\widehat{\mathbf{G}}^{\text{AD}*} = \frac{a_0(\nu^2 - \nu)}{2\nu} \begin{pmatrix} 0 \\ \delta_x \mu_y u^* + \delta_y \mu_x v^* \\ \delta_y \mu_x p^* \end{pmatrix} \quad (12)$$

2. Compute limiting coefficients,  $\omega$ , for the antidiffusive fluxes.

See Section 3 for more details.

3. Update the solution to the  $n + 1$  time level.

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \frac{k}{h} (\delta_x \mu_y \omega_f \widehat{\mathbf{F}}^{\text{AD}*} + \delta_y \mu_x \omega_g \widehat{\mathbf{G}}^{\text{AD}*}) \quad (13)$$

## 3 Flux Limiting

### 3.1 Procedure for One Dimension

The simplest flux limiting case, that which occurs in one dimension, will be considered first. The goal is to implement a method that will ensure that monotone data will remain monotone while advancing in time. In one dimension, the definition of monotonicity is simple and well defined. When computing the limiting coefficients for a given cell, the first order solution will be used to determine if the cell value falls on the interval defined by its left and right neighbors. If it does, the data is monotone and we wish to keep it that way. Mathematically, this can be expressed with the inequality

$$0 \leq \frac{u_j^{n+1} - u_{j-1}^*}{u_{j+1}^* - u_{j-1}^*} \leq 1 \quad (14)$$

By defining the quantities  $\delta u_L^* = u_j^* - u_{j-1}^*$  and  $\delta u_R^* = u_{j+1}^* - u_j^*$  and substituting them, along with the update equation, into the previous inequality we obtain

$$0 \leq \frac{2}{Q^+} + \omega_L - \omega_R \frac{\delta u_L^*}{\delta u_R^*} \leq \frac{2}{Q^+} \left(1 + \frac{\delta u_R^*}{\delta u_L^*}\right) \quad (15)$$

Where

$$Q^+ = Q - \nu^2 \quad Q = \nu \quad \nu \equiv \frac{a_0 k}{h} \quad (16)$$

Note that the limiting coefficients  $\omega_L$  and  $\omega_R$  have been applied to the fluxes in the update equation prior to substitution. The coefficients must fall between zero and one. A value of zero reduces the solution to first order, while a value of one gives the full second order method. The goal is to find the largest allowable coefficients that will still prevent non-physical oscillations from developing in the solution.

Noting that all the quantities in the inequalities are positive, one can use "worst case scenarios" to arrive at expressions for the limiting coefficients. The left inequality is

$$0 \leq \frac{2}{Q^+} + \omega_L - \omega_R \frac{\delta u_L^*}{\delta u_R^*} \quad (17)$$

The positive term  $\omega_L$  can only help this inequality, so the worst case of  $\omega_L = 0$  is assumed to give

$$\omega_R \leq \frac{2}{Q^+} \frac{\delta_L^*}{\delta_R^*} \quad (18)$$

Similar logic is used on the right inequality to give

$$\omega_L \leq \frac{2}{Q^+} \left(1 + \frac{\delta u_R^*}{\delta u_L^*}\right) \quad (19)$$

### 3.1.1 Procedure for Two Dimensions

The challenge now, which is of critical importance, is to develop a limiting method which can be applied to two dimensional, and eventually three dimensional, problems. The method and reasoning presented in the previous section becomes much more difficult in two dimensions since each flux depends on four neighboring cells. This adds a great deal of complexity to the inequalities and makes it difficult or impossible to find explicit expressions for the limiting coefficients. To begin analyzing the two dimensional case we will first consider the second order pressure correction

$$p^{n+1} = p^* - \frac{k}{h} (\delta_x \mu_y \omega_f \widehat{F^{AD*}} + \delta_y \mu_x \omega_g \widehat{G^{AD*}}) \quad (20)$$

Where the fluxes are given by

$$\widehat{F^{AD*}} = \frac{a_0(\nu^2 - \nu)}{2\nu} (\delta_x \mu_y p^*) \quad (21)$$

$$\widehat{G^{AD*}} = \frac{a_0(\nu^2 - \nu)}{2\nu} (\delta_y \mu_x p^*) \quad (22)$$

Performing a fair amount of algebra, the second order pressure update simplifies to

$$p^{n+1} = p^* - \frac{(\nu^2 - \nu)}{4} [\omega_1(p_1 - p_5) + \omega_2(p_3 - p_5) + \omega_3(p_9 - p_5) + \omega_4(p_7 - p_5)] \quad (23)$$

This is a notable result and shows that the fluxes for the second order pressure update depend only on the diagonal neighbors of the cell of interest. Furthermore, this implies that there are

actually two distinct pressure fluxes per node. Since these two fluxes depend only on the diagonal neighbors, they can be limited independently while still preserving the vorticity preserving properties of the scheme. Using this interpretation of the fluxes, it can be observed that we nearly have a one dimensional limiting problem along each cell diagonal. In fact, if the cell centered pressure could be intelligently split between the two sets of diagonal “flux pairs”, two one dimensional limiting problems would result. Since we know how to solve one dimensional limiting problems, it is this procedure we seek to implement. Note that this technique could be easily generalized to three dimensions.

In order to distribute the cell centered pressure between the two one dimensional limiting problems, define two weights  $w_a$  and  $w_b$ . The simplest way to approach this would be to assign values of 0.5 to each weight, which would evenly split the cell centered value between the limiting problems. However, it seems that this would be overly simplistic and that it would be better to assign more of the cell centered value to the flux pair that has the largest net flux value. The logic here is that the flux pair with the largest net value will have the greatest effect during the final cell update and therefore should be assigned a larger portion of the cell centered value for limiting purposes. Now define

$$f_a^{net} \equiv abs(f_1 + f_3) \quad f_b^{net} \equiv abs(f_2 + f_4) \quad (24)$$

From here the distribution weights can be defined as

$$w_a = \frac{f_a^{net}}{f_a^{net} + f_b^{net}} \quad w_b = \frac{f_b^{net}}{f_a^{net} + f_b^{net}} \quad (25)$$

and the pressures assigned to each limiting problem are given by

$$p_a = w_a p_5 \quad p_b = w_b p_5 \quad (26)$$

To summarize, the two dimensional pressure limiting problem has now been reduced to two one dimensional limiting problems which are solved along cell diagonals. This technique results in two limiting coefficients being computed per node, one for each diagonal flux. Implementation could be such that the two fluxes at each node are limited independently or the most conservative coefficient could be applied to both fluxes. Next it is necessary to turn our attention to the velocities. The natural question to ask here is if the velocity components can be limited in the same manner as the pressure. To investigate this question, we turn to the continuous form of the equations and analyze the second order terms.

The acoustic system is repeated here for convenience in vector form

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{1}{\rho_0} \nabla(p) = 0 \quad (27)$$

$$\frac{\partial p}{\partial t} + \rho_0 a_0^2 \nabla \bullet \mathbf{u} = 0 \quad (28)$$

Differentiating (1) with respect to time we have

$$\frac{\partial^2 \mathbf{u}}{\partial t^2} + \frac{1}{\rho_0} \frac{\partial}{\partial t} \nabla(p) = 0 \quad (29)$$

Take the gradient of (2) to obtain

$$\nabla \left( \frac{\partial p}{\partial t} \right) + \rho_0 a_0^2 \nabla (\nabla \bullet \mathbf{u}) = 0 \quad (30)$$

Noting that partial differentiation is commutative we have

$$\frac{\partial}{\partial t} \nabla(p) = -\rho_0 a_0^2 \nabla(\nabla \bullet \mathbf{u}) \quad (31)$$

Substitute (5) into (3) to show

$$\frac{\partial^2 \mathbf{u}}{\partial t^2} = a_0^2 \nabla(\nabla \bullet \mathbf{u}) \quad (32)$$

Recall the vector identity

$$\nabla(\nabla \bullet \phi) = \nabla \bullet (\nabla \phi) + \nabla \times (\nabla \times \phi) \quad (33)$$

Applying (7) to the RHS of (6) we have

$$\frac{\partial^2 \mathbf{u}}{\partial t^2} = a_0^2 \nabla \bullet (\nabla \mathbf{u}) + \nabla \times (\nabla \times \mathbf{u}) \quad (34)$$

Noting that  $\nabla \times \mathbf{u} = \boldsymbol{\omega}$  and  $\nabla \bullet (\nabla) \equiv \nabla^2$  we write

$$\frac{\partial^2 \mathbf{u}}{\partial t^2} = a_0^2 \nabla^2 \mathbf{u} + \nabla \times \boldsymbol{\omega} \quad (35)$$

Some vector manipulation will show that  $\nabla \times \boldsymbol{\omega}$  is given in 2D by

$$\frac{\partial}{\partial y} \left( \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) \hat{\mathbf{i}} - \frac{\partial}{\partial x} \left( \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) \hat{\mathbf{j}} \quad (36)$$

The x and y components of the second order velocity equation are therefore

$$u_{tt} = a_0^2 (u_{xx} + u_{yy} + v_{yx} - u_{yy}) \quad (37)$$

$$v_{tt} = a_0^2 (v_{xx} + v_{yy} - v_{xx} + u_{xy}) \quad (38)$$

which can also be written as

$$\frac{\partial^2 u}{\partial t^2} = a_0^2 (\nabla^2(u) + \frac{\partial}{\partial y} \omega_i) \quad (39)$$

$$\frac{\partial^2 v}{\partial t^2} = a_0^2 (\nabla^2(v) - \frac{\partial}{\partial x} \omega_j) \quad (40)$$

Note at this point that the second order velocity corrections are composed of two terms. The first is Laplacian and the second term contains a component of the vorticity curl. The algebra will be omitted for brevity, but if one follows an analogous procedure for the pressure, you will find that

$$p_{tt} = a_0^2 (p_{xx} + p_{yy}) \quad (41)$$

It is now understood that the tidy form of the equation for the second order pressure update used in constructing the limiting scheme was due to the the Laplacian form of the continuous equation. While the velocities have a Laplacian term also, there is an additional vorticity term that must also be accounted for. Perhaps it should be pointed out that this term is, obviously, only important for vortical flows. In the absence of vorticity the velocity terms could be limited in a way that is completely analogous to the pressure. In fact, the current implementation of the code does just

this, and vorticity is neglected when limiting the velocity. Improving this procedure is one focus of ongoing work.

There is one last issue related to velocity limiting that deserves some attention. The velocity limiting must be done in such a way that does not destroy the vorticity preserving property of the scheme. A natural question that arises here is whether or not different limiting coefficients can be applied to the velocity components at a given node while still keeping this important property intact.

Recall the compact vorticity, which we seek to preserve

$$\zeta = \delta_x \mu_y v - \delta_y \mu_x u \quad (42)$$

Vorticity preservation implies that the following condition should be enforced for any two successive time levels

$$\zeta^{n+1} = \zeta^n \implies \delta_x \mu_y v^{n+1} - \delta_y \mu_x u^{n+1} = \delta_x \mu_y v^n - \delta_y \mu_x u^n \quad (43)$$

This is accomplished, per the work of Morton and Roe [1], by evaluating the divergence of pressure via vertex fluxes. To understand how to keep this property while limiting the velocity components, first take a look at the first order scheme. Recall that the update equations for the first order scheme look like

$$u^* = u^n - \frac{k}{h} [\delta_x \mu_y (\mu_x \mu_y \rho a^2 u - \frac{Qh}{2k} (\delta_x \mu_y u^n + \delta_y \mu_x v^n))] \quad (44)$$

$$v^* = v^n - \frac{k}{h} [\delta_y \mu_x (\mu_x \mu_y \rho a^2 v - \frac{Qh}{2k} (\delta_x \mu_y u^n + \delta_y \mu_x v^n))] \quad (45)$$

Substitute (44) and (45) into (43) and cancel terms

$$\begin{aligned} & \delta_x \mu_y [v^n - \frac{k}{h} [\delta_y \mu_x (\mu_x \mu_y \rho a^2 v - \frac{Qh}{2k} (\delta_x \mu_y u^n + \delta_y \mu_x v^n))] \\ & - \delta_y \mu_x [u^n - \frac{k}{h} [\delta_x \mu_y (\mu_x \mu_y \rho a^2 u - \frac{Qh}{2k} (\delta_x \mu_y u^n + \delta_y \mu_x v^n))] \\ & = \delta_x \mu_y v^n - \delta_y \mu_x u^n \end{aligned} \quad (46)$$

$$\begin{aligned} & \delta_x \mu_y [-\frac{k}{h} [\delta_y \mu_x (\mu_x \mu_y \rho a^2 v - \frac{Qh}{2k} (\delta_x \mu_y u^n + \delta_y \mu_x v^n))] \\ & - \delta_y \mu_x [-\frac{k}{h} [\delta_x \mu_y (\mu_x \mu_y \rho a^2 u - \frac{Qh}{2k} (\delta_x \mu_y u^n + \delta_y \mu_x v^n))] \\ & = 0 \end{aligned} \quad (47)$$

$$-\delta_x \mu_y \frac{k}{h} [\delta_y \mu_x \mu_x \mu_y \rho a^2 v] + \delta_y \mu_x \frac{k}{h} [\delta_x \mu_y \mu_x \mu_y \rho a^2 u] = 0 \quad (48)$$

If vorticity is to be preserved, the LHS of (48) must evaluate to zero. Experiments with the existing code have shown that this is indeed the case. Now consider the second order FCT scheme. The update equations here are

$$u^{n+1} = u^* - \frac{k}{h} [\delta_x \mu_y \omega_u \frac{aQ^+}{2\nu} (\delta_x \mu_y u^* + \delta_y \mu_x v^*)] \quad (49)$$

$$v^{n+1} = v^* - \frac{k}{h} [\delta_y \mu_x \omega_v \frac{aQ^+}{2\nu} (\delta_x \mu_y u^* + \delta_y \mu_x v^*)] \quad (50)$$

Substitute (49) and (50) into (43)

$$\begin{aligned} & \delta_x \mu_y [v^* - \frac{k}{h} [\delta_y \mu_x \omega_v \frac{aQ^+}{2\nu} (\delta_x \mu_y u^* + \delta_y \mu_x v^*)]] \\ & - \delta_y \mu_x [u^* - \frac{k}{h} [\delta_x \mu_y \omega_u \frac{aQ^+}{2\nu} (\delta_x \mu_y u^* + \delta_y \mu_x v^*)]] \\ & = \delta_x \mu_y v^n - \delta_y \mu_x u^n \end{aligned} \quad (51)$$

Substitute (44) and (45) into (51) and then cancel terms

$$\begin{aligned} & \delta_x \mu_y [v^n - \frac{k}{h} [\delta_y \mu_x (\mu_x \mu_y \rho a^2 v - \frac{Qh}{2k} (\delta_x \mu_y u^n + \delta_y \mu_x v^n))] \\ & \quad - \frac{k}{h} [\delta_y \mu_x \omega_v \frac{aQ^+}{2\nu} (\delta_x \mu_y u^* + \delta_y \mu_x v^*)]] \\ & - \delta_y \mu_x [u^n - \frac{k}{h} [\delta_x \mu_y (\mu_x \mu_y \rho a^2 u - \frac{Qh}{2k} (\delta_x \mu_y u^n + \delta_y \mu_x v^n))] \\ & \quad - \frac{k}{h} [\delta_x \mu_y \omega_u \frac{aQ^+}{2\nu} (\delta_x \mu_y u^* + \delta_y \mu_x v^*)]] \\ & = \delta_x \mu_y v^n - \delta_y \mu_x u^n \end{aligned} \quad (52)$$

$$\begin{aligned} & \delta_x \mu_y [-\frac{k}{h} [\delta_y \mu_x (\mu_x \mu_y \rho a^2 v)]] - \frac{k}{h} [\delta_y \mu_x \omega_v \frac{aQ^+}{2\nu} (\delta_x \mu_y u^* + \delta_y \mu_x v^*)] \\ & - \delta_y \mu_x [-\frac{k}{h} [\delta_x \mu_y (\mu_x \mu_y \rho a^2 u)]] - \frac{k}{h} [\delta_x \mu_y \omega_u \frac{aQ^+}{2\nu} (\delta_x \mu_y u^* + \delta_y \mu_x v^*)] \\ & = 0 \end{aligned} \quad (53)$$

From (48) we can infer that terms 1 and 3 of (53) combine to give zero, leaving

$$\begin{aligned} & -\delta_x \mu_y \frac{k}{h} [\delta_y \mu_x \omega_v \frac{aQ^+}{2\nu} (\delta_x \mu_y u^* + \delta_y \mu_x v^*)] \\ & + \delta_y \mu_x \frac{k}{h} [\delta_x \mu_y \omega_u \frac{aQ^+}{2\nu} (\delta_x \mu_y u^* + \delta_y \mu_x v^*)] \\ & = 0 \end{aligned} \quad (54)$$

Factoring like terms and operators gives

$$\delta_x \mu_y \delta_y \mu_x \frac{Q^+}{2} [\omega_v (\delta_x \mu_y u^* + \delta_y \mu_x v^*) - \omega_u (\delta_x \mu_y u^* + \delta_y \mu_x v^*)] = 0 \quad (55)$$

Inspection of (55) implies that the limiting coefficients must be equal for the equality to be true. This analysis suggests that vorticity will be generated if dissimilar limiting coefficients are applied to the velocity components. This conclusion agrees with experiments run to date with the research code.

## 4 Numerical Results

Results obtained from the most recent implementation of the code are now presented. First, convergence analysis will be discussed to gain confidence that the Vorticity Preserving FCT (VPFCT) and MUSCL-H algorithms are working properly. Then results will be presented that compare the performance of the VPFCT and MUSCL-H scheme for the following classes of problems: smooth initial data, discontinuous initial data, and vortical flows.

### 4.1 Convergence Analysis

Some convergence analysis was done using Richardson's deferred approach to the limit. This notes that the following equation can be used to relate an exact solution, its numerical approximation, and the design convergence rate of the numerical method. For both VPFCT and MUSCL-H we have

$$\phi^{exact} = \phi^{approx.} + O(h^2) \quad (56)$$

Recognizing that the exact solution is a constant we can rearrange this expression to get

$$\phi^{approx} = c_1 h^2 + c_2 \quad (57)$$

When plotting some code output as a function of the mesh parameter,  $h$ , a linear relationship should exist. This procedure was performed by taking the code output of interest to be the average velocity magnitude squared over the entire domain. Velocity magnitude squared was chosen so that a zero average would not be obtained on problems with cylindrical symmetry. The velocity magnitude was left squared to avoid a square root sign that complicates the analysis. It should be noted since the velocity magnitude was squared, there is a fourth order deviation from the perfect linear relationship shown above. This was assumed to be negligible. The test problem had smooth initial data: a Gaussian pressure disturbance to a fluid at rest.

A number of interesting conclusions can be reached from studying the convergence results presented in Figure 1. On an encouraging note, this figure shows that the unlimited versions of both VPFCT and MUSCL-H are converging at second order and to the same value. In addition, the constant in the leading error term of the VPFCT scheme is smaller than that of the MUSCL-H, as evidenced by the fact that the VPFCT line falls above that for MUSCL-H on the plot. The results for the limited VPFCT scheme were disappointing as the convergence failed on finer meshes. Understanding this failure and fixing it is an area of ongoing work. The VPFCT limiter which produced these results used the distribution weights defined using net flux values. In addition, it used the most conservative pressure limiting coefficient at each node. Experimentation has shown that this approach is preferred over using two limiting coefficients at each node. The implementation using distinct limiting coefficients has shown to increase the leading error constant during experimentation. Ironically, the convergence behavior of the VPFCT scheme can be improved by setting the distribution weights to 0.5. At the current time, it appears that using the more complicated weights introduces a large amount of noise into the velocity divergence and throws off the convergence of the scheme. Convergence results when the VPFCT limiting scheme uses 0.5 for the distribution weights are shown in Figure 2.

### 4.2 Smooth Initial Data

Tests were run using a problem with smooth initial data in which a Gaussian Pressure perturbation was introduced to a fluid at rest. The pressure field was initialized according to

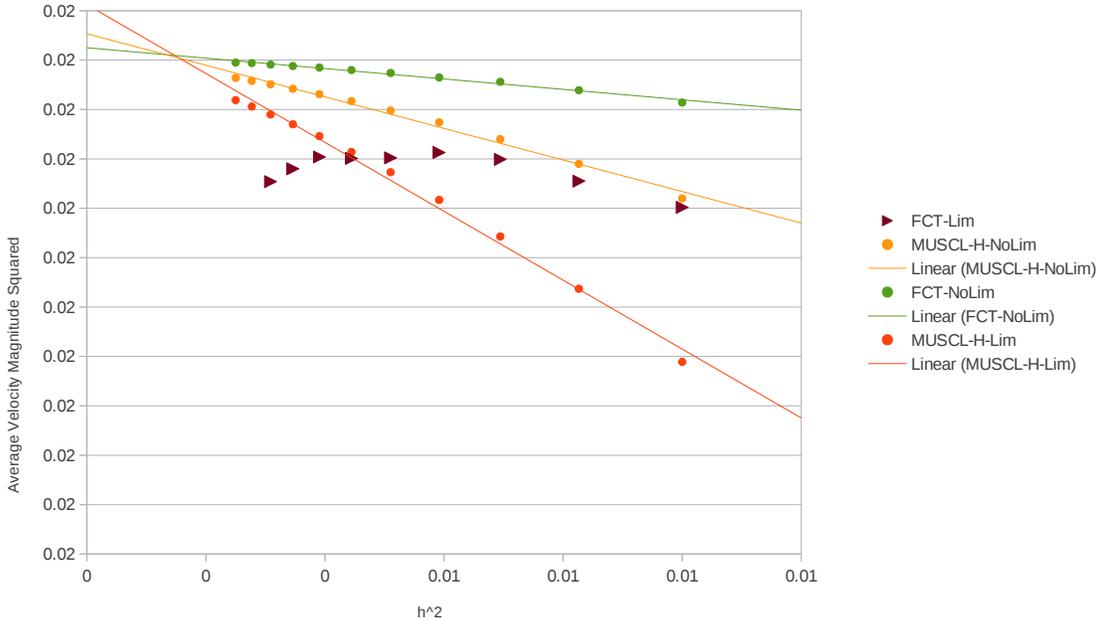


Figure 1: Convergence Study - Convergence Problem with Limited VPFCT

$$p(x, y, 0) = 2exp(x^2 + y^2) \quad (58)$$

Results were plotted as function of radius to help visualize the symmetry preservation ability of the algorithms. Figure 3 shows the pressure and velocity magnitude profiles for the the VPFCT and MUSCL-H schemes. Note that the VPFCT scheme has a slightly higher peak pressure and velocity. Perhaps of more interest was verifying the vorticity preserving properties of the VPFCT scheme. Figure 4 compares the compact vorticity contours produced by each scheme after three time units. The VPFCT maintains zero vorticity to double precision, while the MUSCL-H scheme creates vorticity in an interesting pattern.

### 4.3 Discontinuous Initial Data

A problem with discontinuous initial data was run in order to better exercise the limiting mechanisms of each scheme. Here a cylindrical pressure perturbation was introduced to a fluid at rest. The pressure field was initialized according to

$$p(x, y, 0) = 2 \quad r \leq 2 \quad p(x, y, 0) = 0 \quad r \geq 2 \quad (59)$$

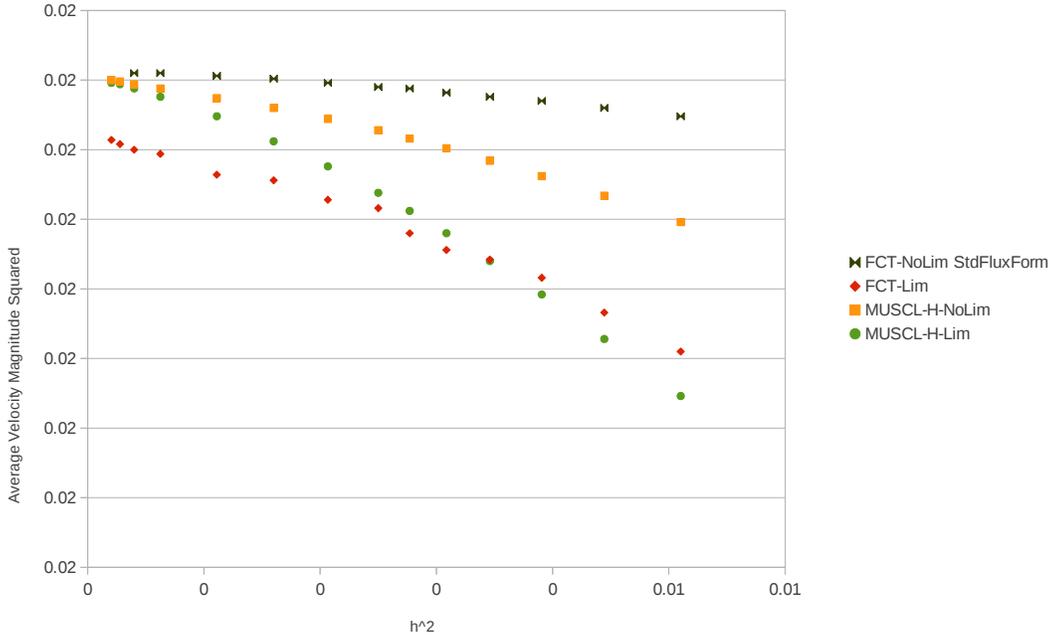


Figure 2: Convergence Study - VPFCT Distribution Weights set to 0.5

Inspection of the pressure and velocity profiles in Figure 5 shows that while the VPFCT result is reasonable, more work is still needed on the limiting procedure. Some overshoots still develop, especially at the top of the shock front, in both the pressure and velocity magnitude plots. The vorticity plot shown in Figure 6 are even more encouraging than for the smooth problem. The vorticity preserving property of the VPFCT is again verified, while the MUSCL-H scheme manufactures significant vorticity.

#### 4.4 Vortical Flows

To this point, all the results that have been introduced were for problems with zero vorticity. For completeness, a test problem with finite vorticity is also included. Here the Taylor-Green vortex solution is modified for inviscid flow. The pressure and velocity fields were initialized according to

$$p(x, y, 0) = \frac{2\rho_0}{4}[\cos(2x) + \cos(2y)] \quad (60)$$

$$u(x, y, 0) = 2\sin(x)\cos(y) \quad (61)$$

$$v(x, y, 0) = -2\cos(x)\sin(y) \quad (62)$$

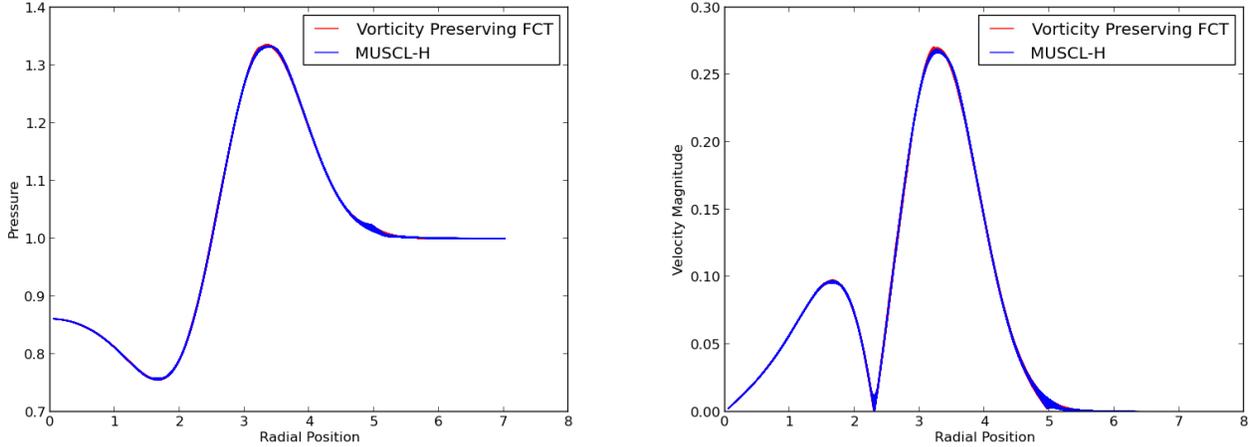


Figure 3: Pressure and Velocity Magnitude Profiles for Gaussian Pressure Disturbance,  $t = 3$  units

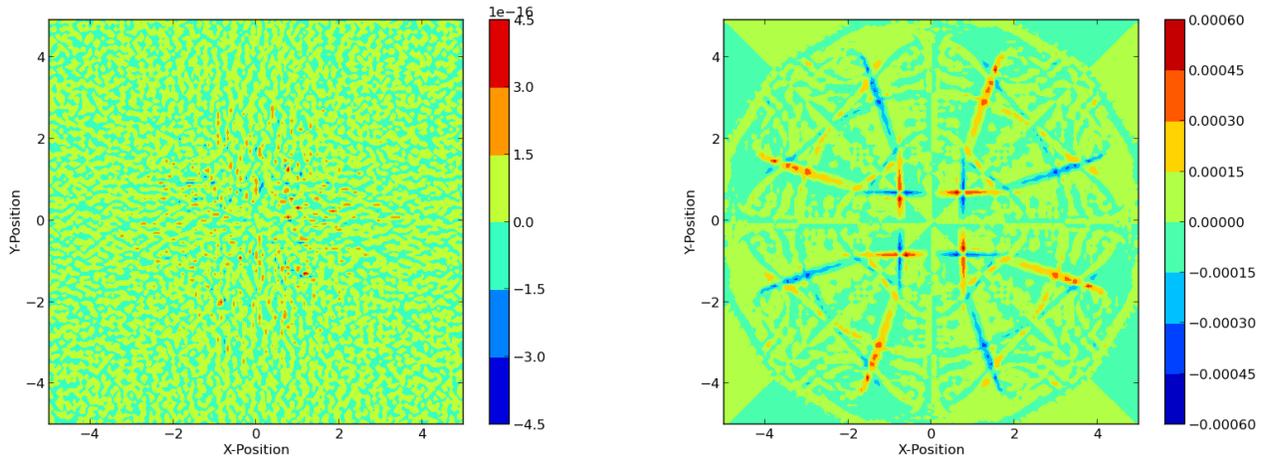


Figure 4: Compact Vorticity Contours for Gaussian Pressure Disturbance,  $t = 3$  units

The vorticity contours at  $t = 0$  are shown in Figure 7. The vorticity contours after 3 time units (or around 2000 timesteps) for the VPFCT and MUSCL-H schemes are shown in Figure 8. Note that the VPFCT scheme is able to maintain the vortices, while the MUSCL-H scheme distorts them. It is worth pointing out that the Taylor-Green solution assumes incompressible flow. Since the discrete operators in this scheme do not exactly evaluate the velocity divergence to zero and these algorithms are for compressible flow, small pressure waves propagate through the domain during the simulation. Even with the wave interactions, the VPFCT scheme is able to preserve vorticity.

## 5 Conclusions and Future Work

Significant progress has been made on the short term research goals discussed in the introduction. A two dimensional VPFCT algorithm has been implemented and compared to the performance of a MUSCL-H algorithm. The vorticity preserving properties of the VPFCT algorithm have been verified and its performance has shown promise relative to the MUSCL-H algorithm. Specifically,

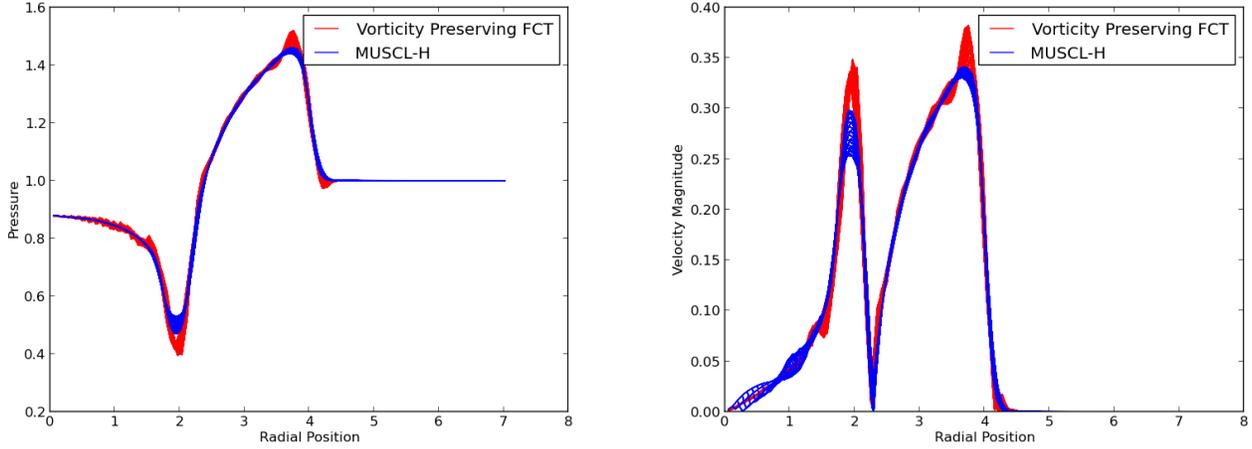


Figure 5: Pressure and Velocity Magnitude Profiles for Cylindrical Pressure Disturbance,  $t = 3$  units

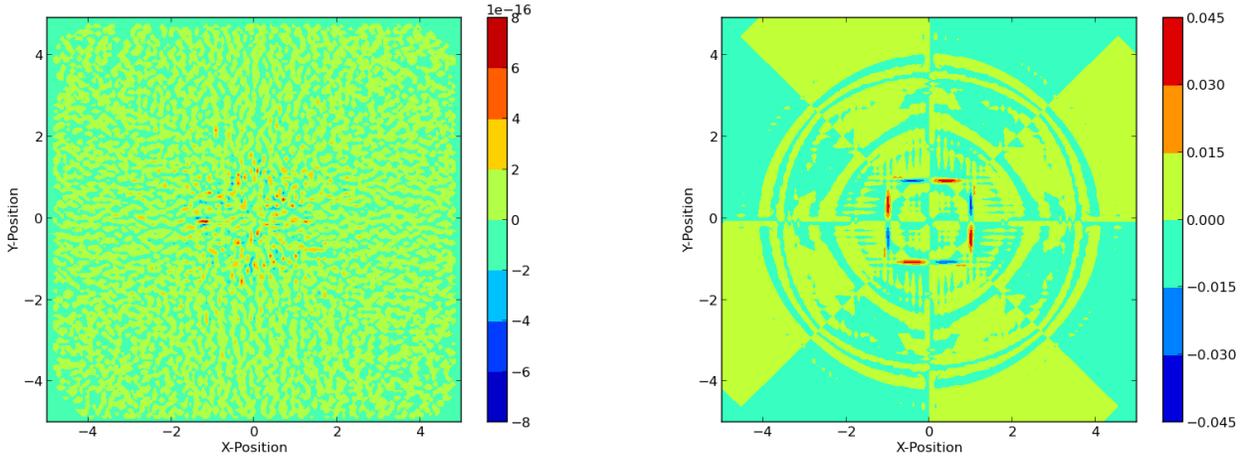


Figure 6: Compact Vorticity Contours for Cylindrical Pressure Disturbance,  $t = 3$  units

the unlimited version of VPFCT was shown to have a smaller leading error constant than the MUSCL-H scheme. There is still work to be done on the VPFCT limiting mechanism. The limiting scheme needs to be improved so that it does not adversely affect convergence. In addition, some overshoots are still appearing in the limited VPFCT solution so more aggressive limiting is needed in some instances. Once these issues are resolved in a satisfactory manner, work will begin on extending the method to the Lagrangian Euler equations.

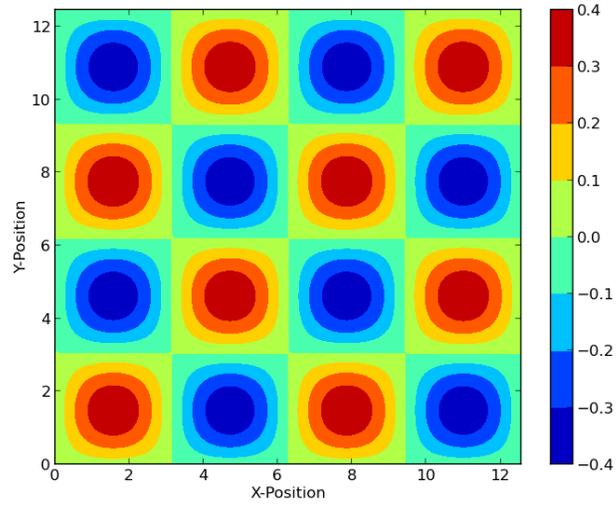


Figure 7: Initial Compact Vorticity Contours: Taylor-Green Vortex

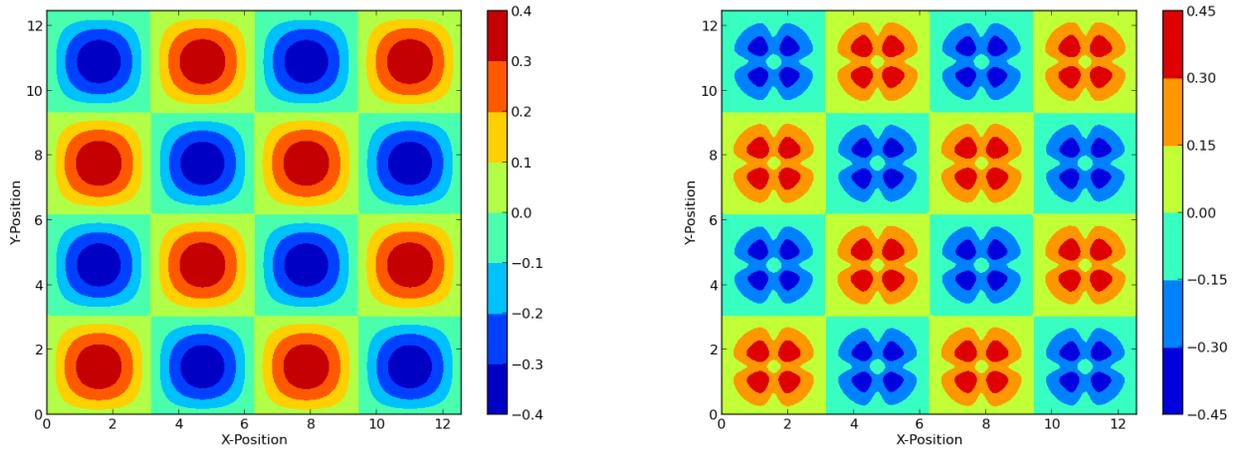


Figure 8: Compact Vorticity Contours for Taylor-Green Vortex,  $t = 3$  units

## References

- [1] Morton, K. W. and Roe, P.L.. *Vorticity-preserving Lax-Wendroff-type scheme for the system wave equation*. SIAM J. Sci. Comp., 2001, Vol. 23, No. 1, pp.170-191.
- [2] Book, D.L., Boris, J.P.. *Flux corrected transport: I. SHASTA, a fluid transport algorithm that works*. J. Comp. Phys., 1973, Vol. 11, pp. 38-69.

## Appendix A

Definition of compact differencing operators:

$$\delta_x(\cdot)_{j,k} \equiv (\cdot)_{j+1/2,k} - (\cdot)_{j-1/2,k} \quad (63)$$

$$\delta_y(\cdot)_{j,k} \equiv (\cdot)_{j,k+1/2} - (\cdot)_{j,k-1/2} \quad (64)$$

$$\mu_x(\cdot)_{j,k} \equiv \frac{1}{2}[(\cdot)_{j+1/2,k} + (\cdot)_{j-1/2,k}] \quad (65)$$

$$\mu_y(\cdot)_{j,k} \equiv \frac{1}{2}[(\cdot)_{j,k+1/2} + (\cdot)_{j,k-1/2}] \quad (66)$$

# Validation of Two Hydrocodes with a Bi-Metallic Shaped Charge Experiment

Daniel Ingraham

August 16<sup>th</sup>, 2012

## Abstract

Staggered grid (SGH) and cell-centered (CCH) Lagrangian Hydrodynamics are two approaches to modeling high explosives experiments. These experiments often involve complex flows with multiple materials with and without constitutive relationships. One example of complex flow phenomena is the discontinuous velocities along the tangential direction of a contact surface. Lagrangian methods combined with a contact surface algorithm are well-suited for these types of flows. In this work, the SGH and CCH schemes coupled with a contact surface algorithm are used to model a bi-metallic shaped charge experiment. The shaped charge experiment will be used to validate the contact surface methods, and is scheduled to be performed in the coming year at LANL. The experiment consists of a high explosive wrapped around a hemispherical shell of aluminum and an inner hemispherical shell of copper. The interface between the aluminum and copper shells is modeled both as a frictionless and “bonded” surface. The simulations are performed in a two-dimensional  $r$ - $z$  coordinate system using the production code FLAG. A modified Gurney solution for an imploding sphere and Richardson extrapolation is used to evaluate the reasonableness of the results.

## 1 Introduction

### 1.1 What is a hydrocode?

Hydrodynamic codes (hydrocodes) are used to simulate complex phenomena involving multiple materials with complex equations of state and constitutive relationships experiencing large-scale deformation. The term “hydrocode” comes from the behavior of a material when it is subjected to a load many times greater than its strength — the material will flow much like a fluid, where the material’s stress can be approximated by the “hydrodynamic” (pressure) component only. Most modern hydrocodes allow for constitutive relationships that model a material’s strength, however. See refs. [1, 14] for some good introductions to hydrocodes.

Hydrocodes could be divided into two categories: Eulerian codes and Lagrangian codes. Eulerian codes solve the governing equations in a static frame of reference (or, in the case of moving grids, one that is defined ahead of time), with the material moving through the computational cells. Lagrangian codes, on the other hand, adopt a reference frame that *moves with* the deforming material. In an Eulerian code, the location of the mesh nodes and cells is constant, and thus the *volume* of each cell is constant. For a Lagrangian algorithm, however, it is the *mass* of each computational cell is constant, and the mesh node and cell locations are one of the quantities calculated by the algorithm. The mathematical difference between the two perspectives is essentially found in the form of the governing equations each code type solves: the Eulerian codes contain an advection term, while the Lagrangian codes do not. For example, the momentum equation in Eulerian form could be written as

$$\rho \frac{\partial v_i}{\partial t} + \rho v_j \frac{\partial v_i}{\partial x_j} = \rho f_i + \frac{\partial \sigma_{ji}}{\partial x_j}, \quad (1)$$

and the same equation in Lagrangian form as

$$\rho \frac{dv_i}{dt} = \rho f_i + \frac{\partial \sigma_{ji}}{\partial x_j}. \quad (2)$$

The term  $\rho v_j \frac{\partial v_i}{\partial x_j}$  in (1) represents the net rate of momentum transfer per unit volume due to advection.

As with everything in life, both Eulerian and Lagrangian codes come with their own advantages and disadvantages. Most hydrodynamic calculations involve more than one material — thus at least one material interface is present. Because the mesh in a Lagrangian calculation moves with the material, the interface is sharply defined and handled in a natural way. In an Eulerian calculation the interface will become “blurred” as material of one type moves into cells originally containing only material of the “other” type.

One common difficulty with Lagrangian hydrocodes is “mesh tangling.” As a Lagrangian mesh deforms, it may happen that the lines connecting the nodes may cross, which will prevent the calculation from continuing (see Figure 1). Eulerian codes do not have this problem, as the mesh remains stationary, or at least moves in a well-defined way that does not allow it to tangle. A related problem is one where the cells are compressed so greatly that the largest stable timestep allowed by the time-marching scheme falls below a “reasonable” limit (likely specified by the code user), again halting the simulation. (Hydrocodes generally use explicit time-marching algorithms, which generally have relatively restrictive stability limits.)

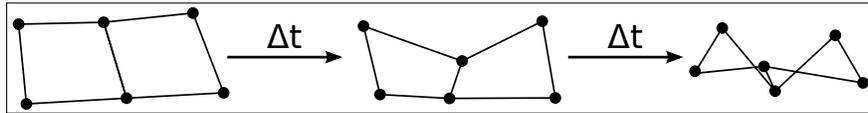


Figure 1: Example of a mesh becoming tangled during a Lagrangian calculation. The “ $\Delta t$  arrows” represent timesteps.

In this work, the LANL production hydrocode FLAG will be used. FLAG is actually an Arbitrary Lagrangian-Eulerian (ALE) code, meaning it can solve the flow equations in either (or both) reference frames. In this work, however, the Lagrangian solver of FLAG was used exclusively.

## 1.2 What is Solution Validation?

Solution validation is cleverly described by Bohem [3] and Blottner [2] (reported by Roache [11]) as ensuring one is “solving the right equations.” The goal is to show that the equations a simulation code is solving is an adequate representation of reality (i.e. experiment). Performing validation studies assumes that the code is “solving the equations right” — Bohem and Blottner’s definition of verification. Stated a slightly different way, one might say that the goal of validation is to estimate an quantity  $\delta$ , where

$$\delta = u_{\text{nature}} - u_{\text{gov eqns}}. \quad (3)$$

Here,  $u_{\text{nature}}$  is the value of a quantity of interest of some system that actually exists in nature (the “real world”), uninfluenced by any error, and  $u_{\text{gov eqns}}$  is the value of the same quantity of interest that would be found if the governing equations constructed to model the system were solved exactly. Equation 3 is inspired by Oberkampf and Trucano’s discussion of validation [8].  $\delta$  can be broken up further:

$$\delta = (u_{\text{nature}} - u_{\text{experiment}}) + (u_{\text{experiment}} - u_{\text{gov eqns}}) \quad (4)$$

where  $u_{\text{experiment}}$  is the value of the quantity of interest measured during an experiment. But (4) can be subdivided still further:

$$\delta = \underbrace{u_{\text{nature}} - u_{\text{experiment}}}_{E_1} + \underbrace{u_{\text{experiment}} - u_{\text{code}}}_{E_2} + \underbrace{u_{\text{code}} - u_{\text{gov eqns}}}_{E_3} \quad (5)$$

where  $u_{\text{code}}$  is the value of the quantity of interest found using the simulation code. Each of the underbraced terms is a component of error with a different interpretation.  $E_1$  is the error in measurement of the quantity of interest during the experiment (the “experimental error”),  $E_2$  is the difference between  $u$  found from experiment and the simulation code, and  $E_3$  is the error created by discretizing the governing equations and

solving them on a finite-precision computer (the “numerical error”).  $E_1$  can be estimated with knowledge of the precision of the devices used to perform the experiment and measure the desired data,  $E_2$  can be calculated directly from the experimental and simulation data, and  $E_3$  can be found using Solution Verification techniques (e.g., Richardson Extrapolation or Roache’s Grid Convergence Index). After these errors are quantified, an estimate of the errors arising from using the governing equation to model nature (i.e.,  $\delta$ ) can be found with (5).

In reference to (5), the goal of this work is to find  $u_{\text{code}}$  and provide an estimate for  $E_3 = u_{\text{code}} - u_{\text{gov eqns}}$  (in a general sense regarding  $u$ ). It is hoped that the researchers performing the experiment will provide  $u_{\text{experiment}}$  and  $E_1 = u_{\text{nature}} - u_{\text{experiment}}$ , and anyone with access to both the experimental and simulation data can find  $E_2 = u_{\text{experiment}} - u_{\text{code}}$ , and thus  $\delta$ .

## 2 Numerical Methods

In this work, the LANL simulation code FLAG will be used for all simulations. FLAG is an arbitrary Lagrangian-Eulerian (ALE) code, but only the Lagrangian solver was used in this work. Two different algorithms for solving the governing equations are implemented in FLAG and used in this work: the well-established compatible staggered grid hydrodynamics (SGH) scheme [5], and the new cell-centered hydrodynamics (CCH) scheme [4].

The SGH scheme, as the name implies, stores flow quantities on a staggered grid: position and velocity are located at the nodes, while density, stress, and internal energy are located at the cell centers. Because of this arrangement, the SGH method uses two control volumes in its spatial discretization of the flow equations: one for evaluating the momentum equation, and the other for evaluating an equation for the internal energy. The “semi-discrete” equations (i.e., discretized in space but not time) are shown in (6), and a diagram of the control volumes of the SGH scheme is shown on the left side of Figure 2. Any suitable time marching scheme can be used to integrate the pseudo-ODEs in (6) — popular choices are the leapfrog or explicit second-order Runge-Kutta methods.

$$\begin{aligned}
 \frac{dM_c}{dt} &= 0 \\
 \frac{d\vec{x}_p}{dt} &= \vec{u}_p \\
 M_p \frac{d\vec{u}_p}{dt} &= \sum_{i \in p} \sigma_c(i) \cdot d\vec{S}_i = \sum_{i \in p} \vec{F}_i \\
 M_c \frac{d\varepsilon_c}{dt} &= - \sum_{i \in c} \vec{F}_i \cdot \vec{u}_{p(i)}
 \end{aligned} \tag{6}$$

The CCH scheme, unlike SGH, stores the velocity of the material at the cell centers, along with density, stress, and total (not internal) energy. Thus the CCH scheme requires only one control volume to evaluate the “right-hand sides” of the semi-discrete equations found in (7).

$$\begin{aligned}
 \frac{dM_c}{dt} &= 0 \\
 \frac{d\vec{x}_p}{dt} &= \vec{u}_p^* \\
 M_c \frac{d\vec{u}_c}{dt} &= \sum_{i \in c} \sigma_p^*(i) \cdot d\vec{S}_i \\
 M_c \frac{dj_c}{dt} &= \sum_{i \in c} \sigma_{p(i)}^* \cdot d\vec{S}_i \cdot \vec{u}_{p(i)}^*
 \end{aligned} \tag{7}$$

To sum the forces and work done on each cell’s control surface, the CCH scheme uses a two-step process. First, the cell-centered velocity and stress values are projected to the nodes using a limited gradient. Next,

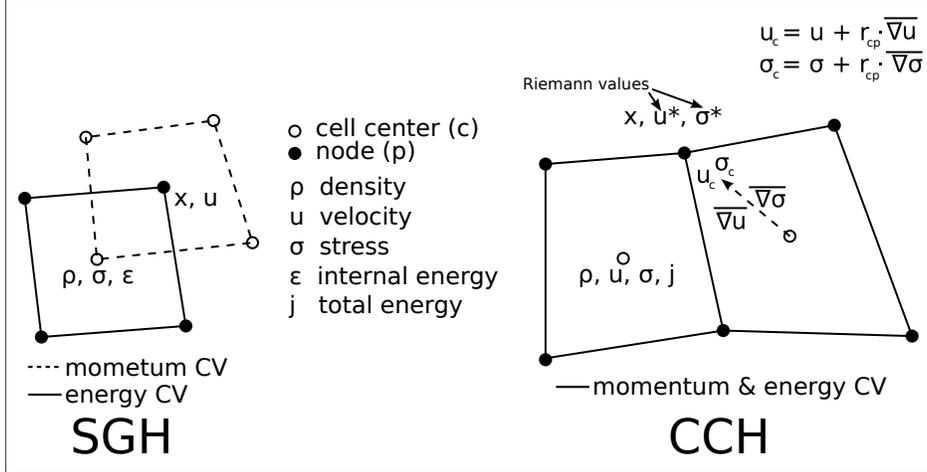


Figure 2: Control volumes and variable locations for the SGH (left) and CCH (right) schemes.

the projected values are used in a multi-dimensional Riemann-like solver to find the values of stress and velocity that will be summed around the cell control surfaces to find the temporal rate of change of velocity and total energy, i.e., the left-hand side of (7).

One difference between the SGH and CCH schemes is that SGH is *non-monotonic* — sharp flow discontinuities can lead to non-physical oscillations. This is not true of the CCH scheme. The practical effect of this difference will be seen in the results show later in this work.

Both the SGH and CCH schemes have contact surface algorithms that can be used to model the interface between two materials. Contact surface algorithms allow two materials to move relative to each other and separate if the materials are in tension. While the surfaces are in contact the components of stress and velocity normal to the interface are continuous, while those components parallel may be discontinuous. Figure 3 shows a conceptual diagram of a contact surface.

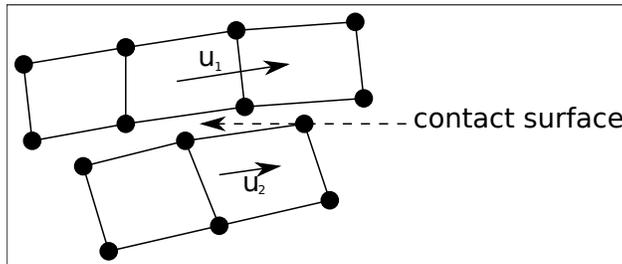


Figure 3: Diagram of a contact surface.

### 3 Shaped Charges

A shaped charge is a device used to penetrate a target using the energy stored in a high explosive (HE). Many different configurations of shaped charges exist, but they all share two common features: a mass of explosive material with a lined cavity at one end and a detonator at the opposite end. When the high explosive is detonated, the liner collapses, forming a dense, high-velocity “slug” of material that impacts and (hopefully)

burrows deeply into the target. Figure 4 shows the nomenclature of a shaped charge. Refs. [13, 12] contain excellent introductions to shaped charges.

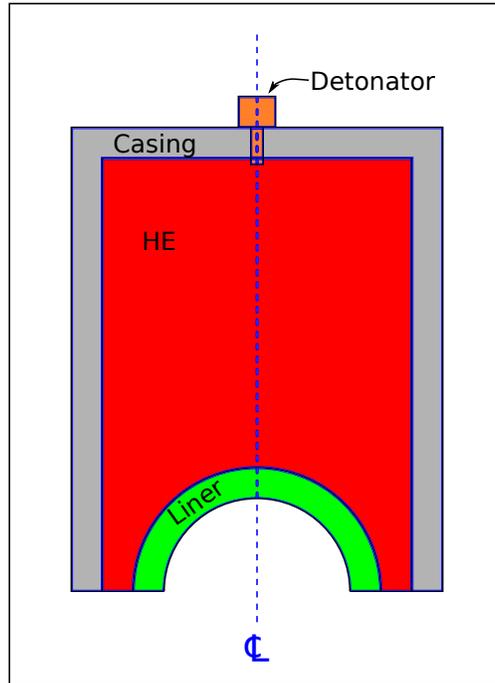


Figure 4: A diagram of a (cylindrical) shaped charge.

Shaped charges have numerous applications. They are used to “finish” oil wells, creating a channel that connects the well to the oil reservoir, and in demolition, especially for collapsing large structural columns. Perhaps shaped charges’ most well-known application, however, is their use in military munitions, such as missiles, torpedoes, and High Explosive Anti-Tank (HEAT) rounds.

## 4 Results and Discussion

### 4.1 Validation Case Setup

A diagram of the shaped charge geometry used in this work is shown in Figure 5. The geometry is defined in a  $r$ - $z$  axis-symmetric coordinate system —  $z$  is an axis of symmetry, and thus the copper and aluminum liners are actually hemispherical shells. The extended 3mm-thick aluminum plate was added to prevent mesh tangling in the HE region; this allows the simulation to run longer. The air region below the shaped charge was added to avoid a “triple point” at the intersection of the bottom edge of the aluminum endplate, the contact surface line, and the air.

A justification for treating the copper-aluminum interface as a frictionless contact surface is warranted. If the following assumptions are made:

- Clearance between parts  $\approx 1\mu m$
- Approximate viscosity of air  $\approx 5.0 \cdot 10^{-5} Pa \cdot s$
- After-shock particle velocity  $\approx 0.2cm/\mu s$

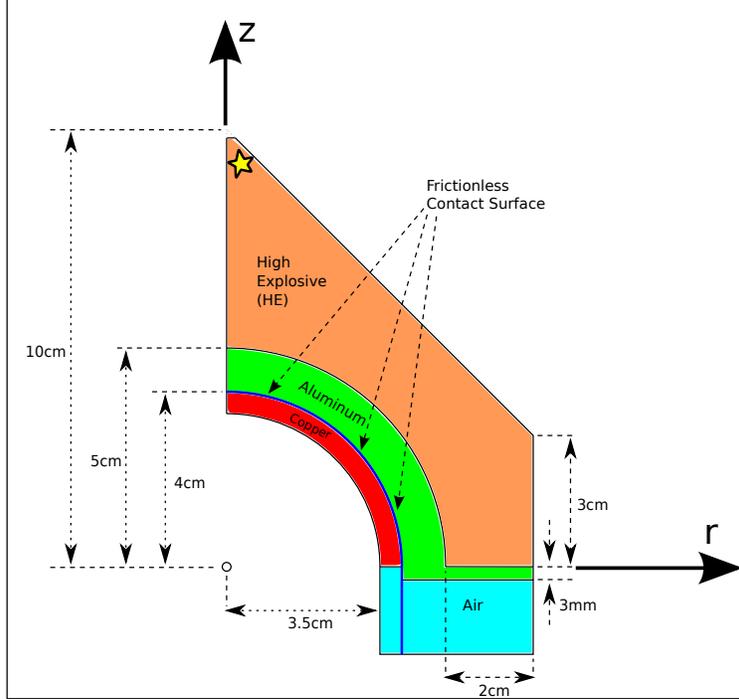


Figure 5: Diagram of the shaped charge geometry used in this work. The yellow star indicates the detonation point, and the blue curve is the locus of points where the contact surface algorithm is used.

then the shear stress at the copper-aluminum interface may be estimated as

$$\tau \approx \mu \frac{\Delta u}{\Delta y} = 5 \cdot 10^{-5} \text{Pa} \cdot \text{s} \frac{0.2 \frac{10^{-2} \text{m}}{10^{-6} \text{s}}}{1 \cdot 10^{-6} \text{m}} = 10^5 \text{Pa} = 1 \text{bar}. \quad (8)$$

The shock pressure for the high explosive used here is about 0.25 MBars, which is five orders-of-magnitude larger than the estimation in (8). Thus the shock will completely overwhelm any frictional forces at the interface, and the assumption of a frictionless contact surface is reasonable. Runs with a “bonded” surface (i.e., completely attached) were performed and will be discussed, however.

A structured mesh was created using the mesh generation software Altair version 5.1.0. Figure 6 shows a screenshot of the mesh. Ninety cells were used in the angular direction, and the radial spacing was set to  $\frac{1}{12}$  cm, giving approximately square cells in the “center” of the mesh (near the aluminum-HE interface). The mesh spacing in the  $r$ -direction was halved in the aluminum endplate and air region; this change was found to be more resistant to mesh tangling, allowing the simulation to run longer.

No casing was used in the shaped charge mesh. Some runs were performed using an outer aluminum casing, but it was found that the results differed little from runs without the casing, and actually made the simulation slightly less resistant to mesh tangling and cell collapse.

## 4.2 Results

The mesh of Figure 6 was used with the input files to simulate the detonation of the shaped charge. All results using the SGH scheme were run on LANL’s “Yellow Rail” cluster and with FLAG version 3.2.Beta.2; results using the CCH scheme were run on the “Turing” cluster and with FLAG version 3.3.Alpha.6. Simulations were run in parallel, typically with the same number of processes as execution cores available on one cluster

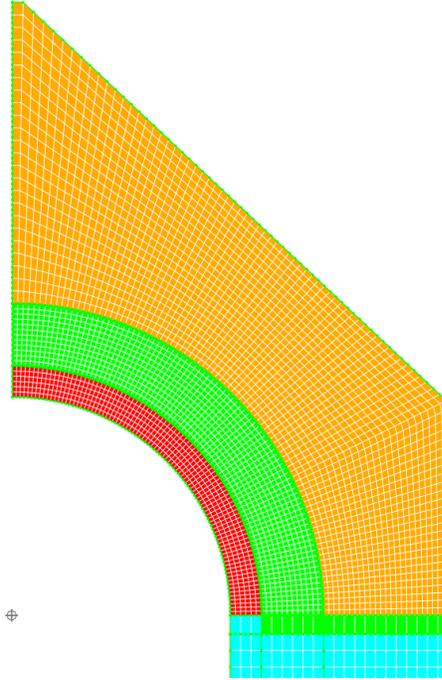


Figure 6: Screenshot of the shaped charge mesh created using Altair 5.1.0.

node (i.e., eight for Yellow Rail and sixteen for Turing). All runs typically took significantly less than the five minutes of walltime requested from the job scheduler.

Figures 7 and 8 show the deformation of the shaped charge assembly for  $t = 10\mu\text{s}$  and  $t = 20\mu\text{s}$ , respectively. The results from the two schemes are qualitatively very similar.

Figure 9 shows a series of pressure maps for the SGH and CCH results. Detonation begins at the “top” (maximum  $z$  location) of the high explosive, causing a shock to propagate downward (Figure 9a). The high explosive region expands as it interacts with the free boundary condition. The shock wave impacts and transmits through the aluminum and copper liner, then eventually reaches the free boundary condition at the inner edge of the copper liner where it is reflected as an expansion wave (Figure 9b). The shock finally encounters the endplate and air, partially transmitting through to the air region and reflecting as another expansion wave (Figures 9c and 9d).

Some differences between the SGH and CCH schemes can be observed through Figure 9. Less of the shock appears to transmit through the liner in the CCH results. Perhaps because of this, the contact surface appears to open up wider with the SGH scheme. Two of the most striking differences between the two schemes, however, are seen in the copper material region. For the SGH scheme, “wiggles” are found in the copper liner trailing the shock. There was some question as to whether these oscillations were a numerical artifact of the SGH scheme, or a physical effect of the copper/aluminum interface opening and closing after the passing of the shock wave. To investigate this, the mesh with the same geometry but increased mesh spacing was created and run with the same inputs as the results from Figure 9. “Wiggles” were still observed, but with a larger wavelength. This result combined with the lack of the “wiggles” in the CCH results strongly indicates the oscillations are non-physical.

The other noticeable difference between the two schemes was also found in the copper liner, this time in the CCH results: pressure “spikes” were observed at the inner and outer edge of the copper liner. These are not seen in the SGH results, and the reader is reminded that the contact surface algorithm is active at the inner edge of the copper liner. Perhaps the interpolation of the stress and velocity with the limited gradient

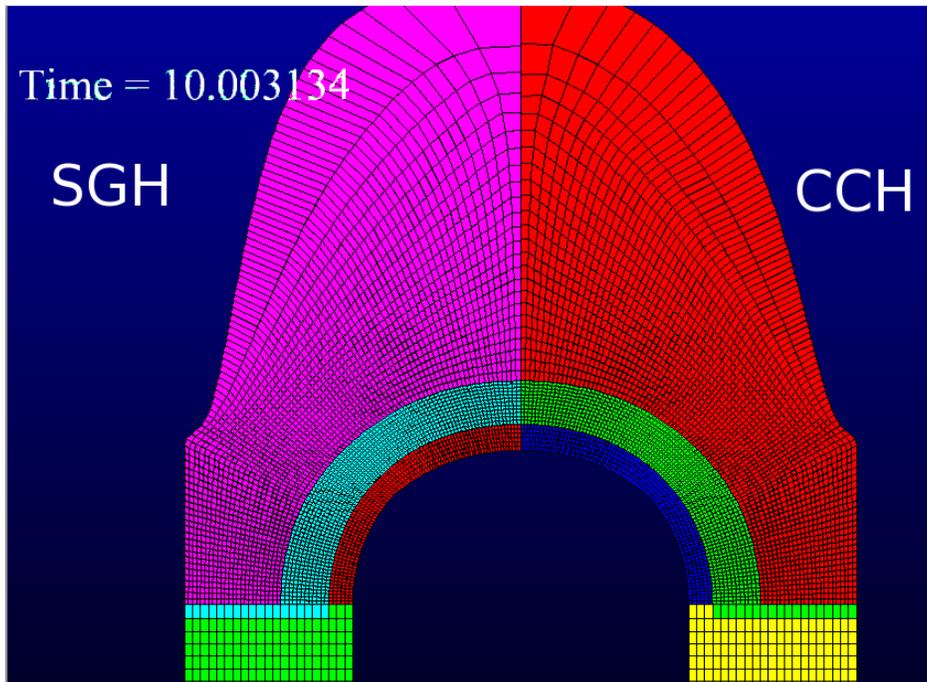


Figure 7: Comparison of the deformation of material at  $t = 10\mu s$  for the SGH and CCH schemes.

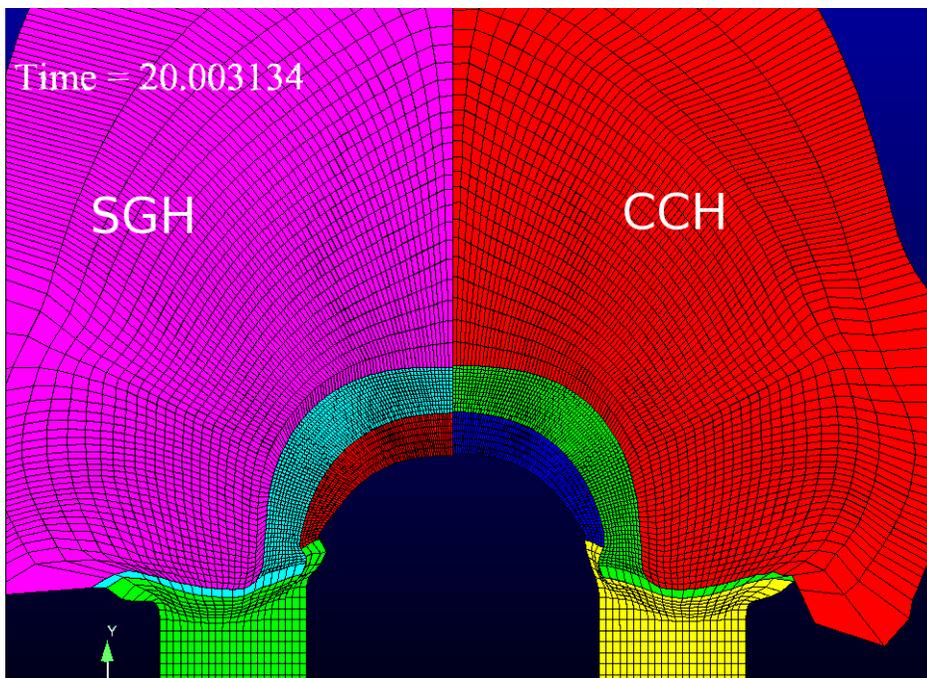


Figure 8: Comparison of the deformation of material at  $t = 20\mu s$  for the SGH and CCH schemes.

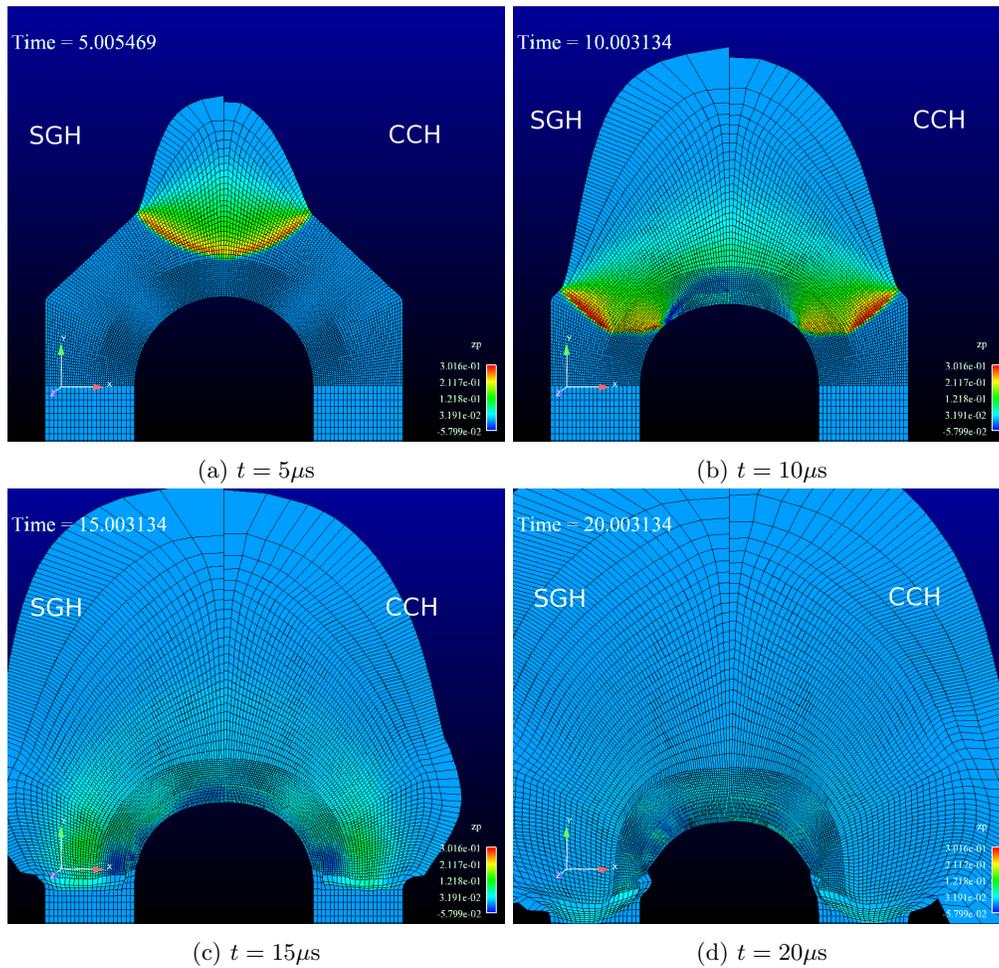


Figure 9: Sequence of pressure maps for the SGH and CCH runs. The units for the pressure scales are  $10^2\text{GPa}$ , and time is shown in  $\mu\text{s}$ .

in the CCH scheme is responsible for these effects?

To investigate the effect of the contact surface, the same mesh and configuration from the results of Figures 7-9 was run without a contact surface at the copper/aluminum interface. A comparison demonstrating the effect of the frictionless contact surface is shown for the SGH and CCH schemes in Figures 10 and 11, respectively. Overall the presence of the contact surface has little effect on the simulation results,

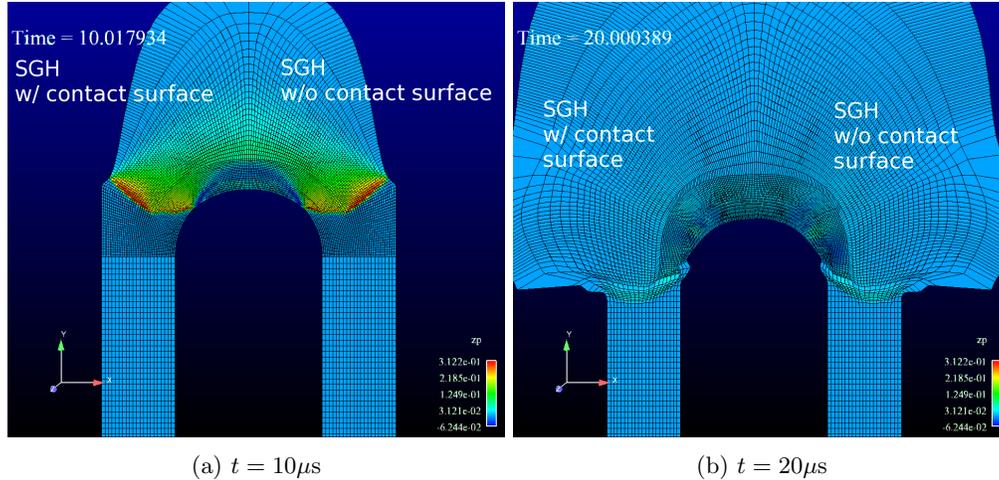


Figure 10: Effect of frictionless contact surface on the SGH runs.

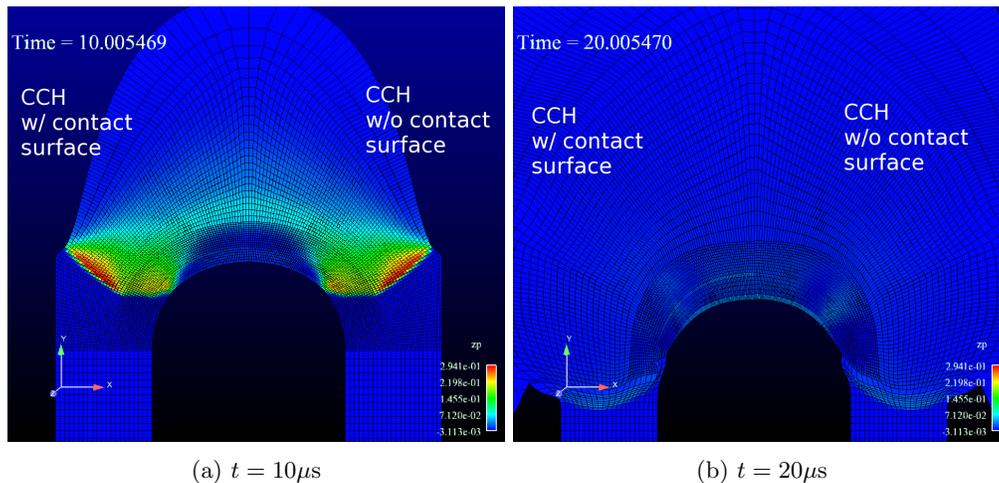


Figure 11: Effect of frictionless contact surface on the CCH runs.

but some differences can be seen. Slightly more of the shock wave is transmitted through the “solid” copper/aluminum interface without the frictionless contact surface than with it. Perhaps because of this, the series of weak compression/expansions reflected after the detonation wave impacts the aluminum endplate and air is stronger for the case without the contact surface. The “wiggles” trailing the shock wave in the SGH results are about the same wavelength as the previous results, but slightly larger in amplitude (again likely due to the stronger shock in the liner region). The most significant observation from the comparison may be that the pressure spike at the copper/aluminum interface in the CCH runs has disappeared with the removal of the contact surface, strongly suggesting that the contact surface algorithm is responsible for this phenomenon.

### 4.3 Sanity Check: Modified Gurney Solution

Gurney solutions [6] use a control volume approach with a momentum and energy balance to estimate the “steady” velocity of a metal slab accelerated by the detonation of a high explosive. Gurney solutions are typically restricted to infinite or symmetric geometries and “explosive” configurations; however, Hirsch [7] extended the Gurney approach to imploding cylindrical and spherical cases. The spherical case will be used here to obtain a solution that can be compared with a slightly modified version of the simulation results presented in Section 4.2.

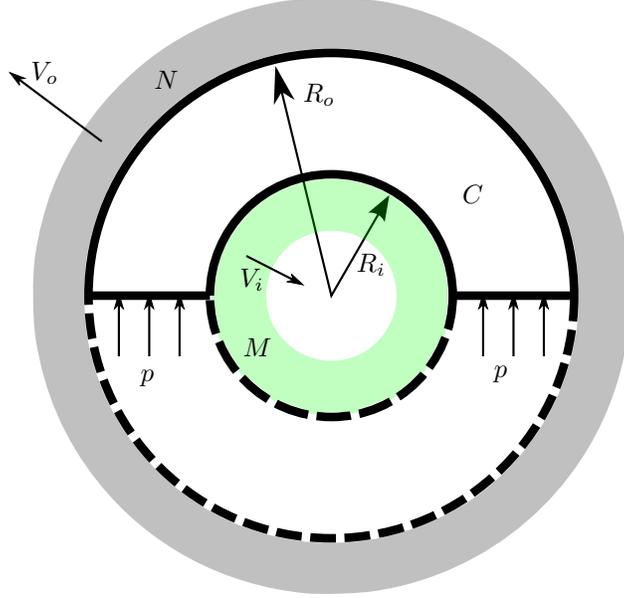


Figure 12: Diagram of Hirsch’s modified Gurney solution for an imploding sphere. The bounding surfaces of the control volume are represented by solid lines, and the grey and green shaded areas denote the outer casing and inner liner.  $M$ ,  $N$ , and  $C$  are the masses of the inner liner, outer casing, and explosive material, respectively.

Figure 12 shows the geometry and control volume associated with the Hirsch’s imploding sphere solution. The momentum balance for the control volume is

$$\frac{1}{2}NV_o - \frac{1}{2}MV_i + 2\pi \int_{R_i}^{R_o} V_{\text{gas}}(r)\rho_{\text{ex}}R^2 dR = \int_0^\infty \int_{R_i}^{R_o} p(r,t) dr dt, \quad (9)$$

and the energy balance,

$$\frac{1}{2}CE = \frac{1}{2} \left( \frac{1}{2}M \right) V_i^2 + \frac{1}{2} \left( \frac{1}{2}N \right) V_o^2 + \pi \int_{R_i}^{R_o} V_{\text{gas}}^2 \rho_{\text{ex}} R^2 dR, \quad (10)$$

where  $\rho_{\text{ex}}$  is the density of the explosive and most of the other terms are defined in Figure 12. The  $E$  in (10) is the “Gurney energy” of the explosive material, which is defined as the amount of chemical energy (per unit mass) in the explosive that is converted into the kinetic energy of the liner, casing, and explosive products. As with all Gurney solutions, the velocity of the explosive product gases  $V_{\text{gas}}$  is assumed to vary linearly with  $R$ :

$$V_{\text{gas}}(R) = (V_o + V_i) \frac{R - R_i}{R_o - R_i} - V_i. \quad (11)$$

Finally, the right-hand side of (9) (the impulse integral) is assumed to take the form

$$\int_0^\infty \int_{R_i}^{R_o} p(r, t) dr dt = a\pi m_o V_i (R_o^2 - R_i^2) \quad (12)$$

$$= a\pi m_o V_i R_i^2 \left( \left[ \frac{R_o}{R_i} \right]^2 - 1 \right) \quad (13)$$

$$= \frac{1}{2} aM \left( \left[ \frac{R_o}{R_i} \right]^2 - 1 \right) V_i \quad (14)$$

where  $m_o$  is the liner mass per unit projected area and  $a$  is a dimensionless constant of proportionality (taken to be unity here).

After considerable manipulation, equations (9)-(14) can be used to produce an expression for the liner velocity  $V_i$ :

$$V_i = \sqrt{2E} \left/ \left[ A \left\{ \left( M/C + \frac{\beta^2 + 3\beta + 6}{10(\beta^2 + \beta + 1)} \right) \right/ A + A \left( N/C + \frac{6\beta^2 + 3\beta + 1}{10(\beta^2 + \beta + 1)} \right) - \frac{3\beta^2 + 4\beta + 3}{10(\beta^2 + \beta + 1)} \right\} \right]^{\frac{1}{2}} \quad (15)$$

where  $\beta = R_o/R_i$  and  $A = V_o/V_i$ :

$$A = V_o/V_i = \left[ M/C + a(M/C)(\beta^2 - 1) + \frac{\beta^2 + 2\beta + 3}{4(\beta^2 + \beta + 1)} \right] \left/ \left( N/C + \frac{3\beta^2 + 2\beta + 1}{4(\beta^2 + \beta + 1)} \right) \right. . \quad (16)$$

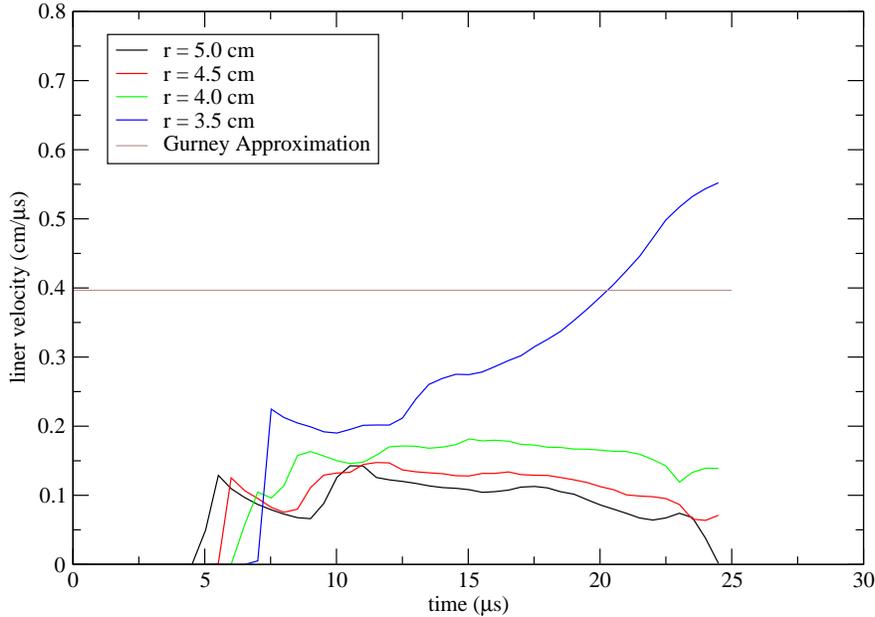


Figure 13: Velocity history of four evenly-spaced points in the all-aluminum liner of the modified shaped charge experiment compared to the Gurney solution for an imploding sphere. All points were  $90^\circ$  from the horizontal direction (i.e., at the “top” of the liner).

To use expressions (15) and (16) to predict the liner velocity  $V_i$ , the simulations from Section 4.2 had to be modified in two ways: the copper-aluminum liner was replaced by a all-aluminum liner, and the contact

surface was removed. The mass and geometric parameters of the shaped charge geometry were substituted into (15) and (16) to obtain a value for  $V_i$ . The results are shown in Figure 13 for the SGH scheme. The liner velocity predicted by the Gurney solution was compared to the velocity of the aluminum in the modified simulation at four evenly-spaced points  $90^\circ$  from horizontal, i.e. the points furthest from the “edges” of the shaped charge, and thus hopefully far enough away from the sources of asymmetry to be a good candidate for comparison with the symmetric Gurney solution.

The result from the Gurney solution was at least of the same order of magnitude as the simulation results — a reasonably good comparison for such a “back of the envelope” approximation to the shaped charge configuration. There are numerous explanations for the discrepancy between the Gurney solution and simulation results for the liner velocity:

- The geometry of the shaped charge and imploding sphere are significantly different — the shaped charge liner is a hemisphere, not a sphere, and the shape of the high explosive is only vaguely spherical,
- The Gurney solution assumes a linear velocity distribution in the HE products — the simulation does not,
- The Gurney solution assumes the liner is instantly accelerated by the expanding HE gasses, and thus predicts a “steady-state” liner velocity, while the simulation obviously accounts for the unsteady motion of the shaped charge liner, and appears to not proceed long enough to predict the “steady-state” liner velocity (notice how the liner velocity for the  $R = 3.5\text{cm}$  point is still trending upward at the end of the simulation).

Despite the above limitations, it is felt that the Gurney solution provides a useful “sanity check” for the simulations in the absence of experimental data.

#### 4.4 Numerical Error Estimation: Richardson Extrapolation

As discussed in Section 1.2, it is the job of the researcher running the simulation code to provide some estimation of the numerical error present in the results of the simulation (i.e.,  $E_3$ ). Richardson extrapolation [9], as presented by Roache [10], can be used to provide such an estimation. One first begins with a Taylor series expansion of a solution at two different “discretization measures”  $h$  (i.e., mesh spacings or time steps):

$$f_f = f_{\text{exact}} + c_1 (h)^p + \text{HOT} \quad (17)$$

$$f_c = f_{\text{exact}} + c_1 (rh)^p + \text{HOT} \quad (18)$$

where  $f_f$  and  $f_c$  are the “fine” and “coarse” numerical solutions,  $r$  is the ratio of the coarse-to-fine discretization measures,  $p$  is the formal order-of-accuracy of the schemes used (here,  $p = 2$ ), and HOT are higher-order terms. If one neglects HOT, then

$$f_c - f_f \approx c_1 (rh)^p - c_1 (h)^p \quad (19)$$

$$\approx c_1 h^p (r^p - 1) \quad (20)$$

and so

$$c_1 \approx \frac{f_c - f_f}{h^p (r^p - 1)} \quad (21)$$

and the leading error term for the coarse solution is

$$E_c = c_1 (rh)^p = \frac{r^p (f_c - f_f)}{r^p - 1} \quad (22)$$

and for the fine,

$$E_f = c_1 (h)^p = \frac{f_c - f_f}{r^p - 1}. \quad (23)$$

To use Richardson extrapolation, then, one must have (at least) two solutions for an identical simulation, differing only in discretization measure. For this work, the same shaped charge simulation described in Section 4.2 was used. The “fine” mesh was identical to that of Section 4.2, and the “coarse” mesh was like the fine, but with every other mesh line removed (i.e.,  $r = 2$ ). Any kind of solution could be taken as  $f$  — here the same data used in Section 4.3 was chosen at  $t = 20.0\mu s$  (i.e., the liner velocity of four evenly-spaced points  $90^\circ$  from the horizontal). A constant  $\Delta t$  was used in the fine and coarse simulation runs to ensure that both simulations ended at the same time level.

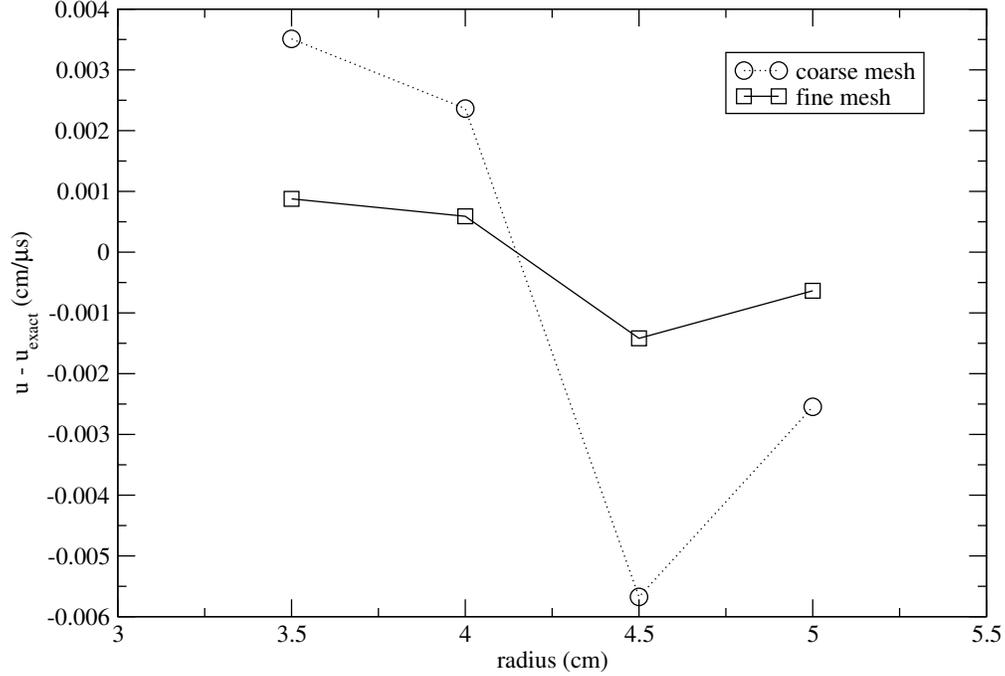


Figure 14: Liner velocity error as predicted by Richardson extrapolation.

Figure 14 shows the Richardson extrapolation results for SGH scheme. The magnitude of the error for the fine mesh is on the order of  $10^{-3}$ .

It must be noted that the use of Richardson extrapolation to estimate the error in a calculation implicitly assumes that the simulations are within the “asymptotic range,” i.e., HOT from (18) are small compared to the leading error term. One method of determining this (brought to the attention of the author by Tyler Lung), would be to first run the same simulation for a range of  $h$ , and then plot  $f$  as a function of  $h^p$ , i.e.,

$$\begin{aligned}
 f(h^p) &= f_{\text{exact}} + c_1 h^p + \text{HOT} \rightarrow 0 \\
 f(y) &\approx f_{\text{exact}} + c_1 y.
 \end{aligned}
 \tag{24}$$

where  $y = h^p$ . If HOT are small, then the curve-fit of the  $f$  data plotted on a  $h^p$  scale will appear as a straight line, with the slope of the line being  $c_1$  and the  $y$ -intercept an estimation of the exact solution. This linear region of the plot indicates the range of  $h$  for which the asymptotic range occurs.

## 5 Conclusions

In this work, the simulation results from a shaped charge validation test case were presented. After a brief explanation of the numerical methods used in the simulation code and the nature of a shaped charge device,

the simulation results were summarized. The Staggered Grid Hydro (SGH) and Cell-Centered Hydro (CCH) schemes as implemented in LANL’s FLAG code were used to simulate the detonation of a bi-metallic shaped charge with and without a frictionless contact surface in  $r$ - $z$  axisymmetric coordinates. Results between the two schemes were qualitatively similar, but some differences were observed. The SGH scheme produced “wiggles” in the solution trailing the detonation shock, and pressure “spikes” were found at the inner surface of the copper liner, and at the outer surface when the contact surface algorithm was used. A slightly modified shaped charge simulation was compared to a Gurney approximation of an imploding sphere. The Gurney approximation was within the same order-of-magnitude of the simulation data, indicating that the simulation results are not ridiculous. The numerical error of the simulation was estimated to be on the order of  $10^{-3}$  using Richardson extrapolation.

Future work clearly includes comparing the simulation data to the forthcoming experiment at LANL. The pressure “spikes” at the inner and outer edge of the copper liner are (at least to the author) unexplained and worth investigating. More work could be done to quantify the numerical error and ensure that the simulations are within the asymptotic range as outlined at the end of Section 4.4. Finally, it may be worthwhile to explore using ALE to extend the life of the simulations — perhaps this would allow more opportunity to compare the simulations to the experimental data.

## Acknowledgments

This work was performed during the 2012 Computational Physics Summer Workshop, XCP-4 at the Los Alamos National Laboratory, run by Dr. Scott Runnels. The author is thankful to Dr. Nathaniel Morgan (mentor, XCP-8) and Tyler Lung (workshop partner) for many useful discussions and frequent assistance.

## References

- [1] C.E. Anderson. An overview of the theory of hydrocodes. *International Journal of Impact Engineering*, 5(1):33–59, 1987.
- [2] F.G. Blottner. Accurate navier-stokes results for the hypersonic flow over a spherical nosetip. *Journal of Spacecraft and Rockets*, 27:113, 1990.
- [3] B.W. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.
- [4] D. Burton, T. Carney, N. Morgan, S. Runnels, S. Sambasivan, and M. Shashkov. A cell centered lagrangian godunov-like method for solid dynamics. *Submitted to the Journal of Computers and Fluids*, 2012.
- [5] D.E. Burton. Consistent finite-volume discretization of hydrodynamic conservation laws for unstructured grids. In *Presented at the Nuclear Explosives Code Developers Conference (NECDC), Las Vegas, NV, 25-28 Oct. 1994*, volume 1, pages 25–28, 1994.
- [6] R.W. Gurney. The initial velocities of fragments from bombs, shells and grenades. Technical Report BRL Report 405, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland, September 1943.
- [7] E. Hirsch. Simplified and extended gurney formulas for imploding cylinders and spheres. *Propellants, Explosives, Pyrotechnics*, 11(1):6–9, 1986.
- [8] W.L. Oberkampf and T.G. Trucano. Verification and validation in computational fluid dynamics. *Progress in Aerospace Sciences*, 38(3):209–272, 2002.
- [9] L.F. 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 Transactions of the Royal Society of London. Series A*, 210:307–357, 1911.

- [10] P.J. Roache. Quantification of uncertainty in computational fluid dynamics. *Annual Review of Fluid Mechanics*, 29(1):123–160, 1997.
- [11] P.J. Roache. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, 1998.
- [12] W. Walters and A.P. Ground. An overview of the shaped charge concept. In *11th Annual ARL/USMA Technical Symposium*, volume 5, 2003.
- [13] W.P. Walters and J.A. Zukas. *Fundamentals of Shaped Charges*. John Wiley & Sons, 1989.
- [14] J.A. Zukas. *Introduction to Hydrocodes*. Elsevier, 2004.

## A Altair Script

This section contains the Python script used with Altair 5.1.0 to generate the shaped charge mesh shown in Figure 6.

```
import math
from altair import *

# For easier coding:
pi = math.pi
tan = math.tan
cos = math.cos
sin = math.sin
atan = math.atan

MeshName = "shaped_charge"
altair.startModel(model = MeshName)

# Number of cells in the theta direction.
NTheta = 90
dTheta = 0.5*pi/float(NTheta)

# Where everything begins.
origin = (0.0, 0.0)#cm

# Dimensions of the geometry.
RinnerCopper = 3.5#cm
RouterCopper = 4.#cm
RinnerAl = RouterCopper
RouterAl = 5.#cm
ChargeTotalHeight = 10.#cm
ChargeBottomWidth = 2.#cm
ChargeSideHeightGiven = 3.#cm
HalfTopFlatWidthGuess = 0.125#cm

# Number of cells in the radial direction in the copper part.
Nr_copper = int(round((RouterCopper - RinnerCopper) / (0.5 * (RouterAl + RinnerAl) * dTheta)))
print("Nr_copper = %s" % Nr_copper)

dr = (RouterCopper - RinnerCopper) / float(Nr_copper)

# Need to decide how many cells to have along the arc.
gammaGiven = tan(ChargeSideHeightGiven/(RouterAl + ChargeBottomWidth))
```

```

# Number of cells in the theta direction for the charge 'side flat.'
NThetaChargeSideHeight = int(round(gammaGiven / dTheta))
# So now I can find what the actual ChargeSideHeight will be.
ChargeSideHeight = (RouterAl + ChargeBottomWidth) * tan(NThetaChargeSideHeight*dTheta)

# Find the 'HalfTopFlatWidth'.
phiGuess = tan(HalfTopFlatWidthGuess / (ChargeTotalHeight - HalfTopFlatWidthGuess))
NThetaHalfTopFlatWidth = int(round(phiGuess/dTheta))
if NThetaHalfTopFlatWidth < 1:
    raise ValueError(
        "Can't create reasonable HalfTopFlatWidth with NTheta = %s. Try increasing NTheta." % NTheta
    )
HalfTopFlatWidth = (
    (ChargeTotalHeight * tan(NThetaHalfTopFlatWidth*dTheta)) /
    (1. + tan(NThetaHalfTopFlatWidth*dTheta))
)

# The points for the 'flats' of the copper shell (i.e., the two flat parts
# joining the inner and outer surface). I'm numbering them from left to right.
p1Copper = (RinnerCopper, 0.)
p2Copper = (RouterCopper, 0.)
p3Copper = (0., RinnerCopper)
p4Copper = (0., RouterCopper)

# The points for the 'flats' of the aluminum shell (i.e., the two flat
# parts joining the inner and outer surface). I'm numbering them from left to
# right.
p1Al = (RinnerAl, 0.)
p2Al = (RouterAl, 0.)
p3Al = (0., RouterAl)
p4Al = (0., RinnerAl)

# There are lots of charge points (eight)! Here we go.
# p1Ch and p2Ch will form the 'flat' for the 'imin' surface of the charge
# block.
# (r , z)
p1Ch = (RouterAl, 0.)
p2Ch = (RouterAl + ChargeBottomWidth, 0.)#cm
p3Ch = (RouterAl + ChargeBottomWidth, ChargeSideHeight)
p4Ch = (HalfTopFlatWidth, ChargeTotalHeight - HalfTopFlatWidth)
p5Ch = (0., ChargeTotalHeight - HalfTopFlatWidth)
p6Ch = (0., RouterAl)

# It might be best to use contour rather than line to create the outer edge of
# the charge. So that means I'll need a list of r and z locations.
outer_charge_rz = [p2Ch, p3Ch, p4Ch, p5Ch]

shell_spacing = 0.3 # 3mm

# Spacing in the 'horizontal' (or r) direction for the air and aluminum
# endplate.
#dr_endplate = (RouterCopper - RinnerCopper) / 3. # This is the radial spacing
# for the previous runs.

# That was a bad idea. Changing it back.
dr_endplate = dr
# Maybe Dr. Morgan meant me to not refine the z direction? But that makes no

```

```

# sense... Hmm... I could *coarsen* the z direction:

# Spacing in the 'vertical' (or z) direction for the air and aluminum
# endplate.
#dz = (RouterAl) * (0.5 * pi) / float(NTheta)
#dz = 1.25 * (RouterAl) * dTheta
dz = 3. * (RouterAl) * dTheta

# These are the points that I'll use to define the bottom aluminum shell.
p1AlShellBot_Cu = (p1Copper[0], -shell_spacing)
p2AlShellBot_Cu = (p2Copper[0], -shell_spacing)
p1AlShellBot_Al = (p1Al[0], -shell_spacing)
p2AlShellBot_Al = (p2Al[0], -shell_spacing)
p1AlShellBot_Ch = (p1Ch[0], -shell_spacing)
p2AlShellBot_Ch = (p2Ch[0], -shell_spacing)

p1AirBot_Cu = (p1Copper[0], -50*shell_spacing)
p2AirBot_Cu = (p2Copper[0], -50*shell_spacing)
p1AirBot_Al = (p1Al[0], -50*shell_spacing)
p2AirBot_Al = (p2Al[0], -50*shell_spacing)
p1AirBot_Ch = (p1Ch[0], -50*shell_spacing)
p2AirBot_Ch = (p2Ch[0], -50*shell_spacing)

# Initialize some lists that I'll be using.
inner_copper_arc_r = []
inner_copper_arc_z = []
outer_copper_arc_r = []
outer_copper_arc_z = []
outer_aluminum_arc_r = []
outer_aluminum_arc_z = []
#theta = []

N = 360 # number of points on the curves that we're going to make. This is not
# the same as the number of cells in the theta direction.
delta = 0.5 * pi / float(N - 1) # N - 1 because python starts at 0 and ends at len - 1
for i in range(N):
    inner_copper_arc_r.append(RinnerCopper * cos(float(i)*delta))
    inner_copper_arc_z.append(RinnerCopper * sin(float(i)*delta))

    outer_copper_arc_r.append(RouterCopper * cos(float(i)*delta))
    outer_copper_arc_z.append(RouterCopper * sin(float(i)*delta))

    outer_aluminum_arc_r.append(RouterAl * cos(float(i)*delta))
    outer_aluminum_arc_z.append(RouterAl * sin(float(i)*delta))

# Now make the contours.
inner_copper_contour = contour('InnerCopperSurface',
                               rList=inner_copper_arc_r,
                               zList=inner_copper_arc_z,
                               origin=origin)
outer_copper_contour = contour('OuterCopperSurface',
                               rList=outer_copper_arc_r,
                               zList=outer_copper_arc_z,
                               origin=origin)
inner_aluminum_contour = contour('InnerAluminumSurface',
                                 rList=outer_copper_arc_r,

```

```

        zList=outer_copper_arc_z,
        origin=origin)
outer_aluminum_contour = contour('OuterAluminumSurface',
        rList=outer_aluminum_arc_r,
        zList=outer_aluminum_arc_z,
        origin=origin)
inner_charge_contour = contour('InnerChargeSurface',
        rList=outer_aluminum_arc_r,
        zList=outer_aluminum_arc_z,
        origin=origin)
outer_charge_contour = contour('OuterChargeSurface',
        rzList=outer_charge_rz,
        origin=origin)

# Next, the 'flats' of the copper shell.
left_flat_copper = line('LeftCopperFlat', p1Copper, p2Copper)
right_flat_copper = line('RightCopperFlat', p3Copper, p4Copper)

# The 'flats' of the aluminum shell.
left_flat_aluminum = line('LeftAluminumFlat', p1Al, p2Al)
right_flat_aluminum = line('RightAluminumFlat', p3Al, p4Al)

# This is the 'imin' edge/surface of the charge.
left_flat_charge = line('LeftChargeFlat', p1Ch, p2Ch)
# This is the 'imax' edge/surface of the charge.
right_flat_charge = line('RightChargeFlat', p6Ch, p5Ch)

# Adding a aluminum shell along the bottom of the shaped charge.
bottom_al_shell_Cu = line('BottomAluminumShell_Cu', p1AlShellBot_Cu, p2AlShellBot_Cu)
bottom_al_shell_Cu_rmin = line('BottomAluminumShell_Cu_rmin', p1Copper, p1AlShellBot_Cu)
bottom_al_shell_Cu_rmax = line('BottomAluminumShell_Cu_rmax', p2Copper, p2AlShellBot_Cu)

bottom_al_shell_Al = line('BottomAluminumShell_Al', p1AlShellBot_Al, p2AlShellBot_Al)
bottom_al_shell_Al_rmin = line('BottomAluminumShell_Al_rmin', p1Al, p1AlShellBot_Al)
bottom_al_shell_Al_rmax = line('BottomAluminumShell_Al_rmax', p2Al, p2AlShellBot_Al)

bottom_al_shell_Ch = line('BottomAluminumShell_Ch', p1AlShellBot_Ch, p2AlShellBot_Ch)
bottom_al_shell_Ch_rmin = line('BottomAluminumShell_Ch_rmin', p1Ch, p1AlShellBot_Ch)
bottom_al_shell_Ch_rmax = line('BottomAluminumShell_Ch_rmax', p2Ch, p2AlShellBot_Ch)

# Creating lines for air layer below aluminum shell.
bottom_air_Cu = line('BottomAir_Cu', p1AirBot_Cu, p2AirBot_Cu)
bottom_air_Cu_rmin = line('BottomAir_Cu_rmin', p1AlShellBot_Cu, p1AirBot_Cu)
bottom_air_Cu_rmax = line('BottomAir_Cu_rmax', p2AlShellBot_Cu, p2AirBot_Cu)

bottom_air_Al = line('BottomAir_Al', p1AirBot_Al, p2AirBot_Al)
bottom_air_Al_rmin = line('BottomAir_Al_rmin', p1AlShellBot_Al, p1AirBot_Al)
bottom_air_Al_rmax = line('BottomAir_Al_rmax', p2AlShellBot_Al, p2AirBot_Al)

bottom_air_Ch = line('BottomAir_Ch', p1AirBot_Ch, p2AirBot_Ch)
bottom_air_Ch_rmin = line('BottomAir_Ch_rmin', p1AlShellBot_Ch, p1AirBot_Ch)
bottom_air_Ch_rmax = line('BottomAir_Ch_rmax', p2AlShellBot_Ch, p2AirBot_Ch)

# Need to make some 'segments', I guess.
inner_copper_segment = segment(inner_copper_contour)
outer_copper_segment = segment(outer_copper_contour)

```

```

inner_aluminum_segment = segment(inner_aluminum_contour)
outer_aluminum_segment = segment(outer_aluminum_contour)

inner_charge_segment = segment(inner_charge_contour)
outer_charge_segment = segment(outer_charge_contour)

left_flat_copper_segment = segment(left_flat_copper)
right_flat_copper_segment = segment(right_flat_copper)

left_flat_aluminum_segment = segment(left_flat_aluminum)
right_flat_aluminum_segment = segment(right_flat_aluminum)

left_flat_charge_segment = segment(left_flat_charge)
right_flat_charge_segment = segment(right_flat_charge)

bottom_al_shell_Cu_segment = segment(bottom_al_shell_Cu)
bottom_al_shell_Cu_segment_rmin = segment(bottom_al_shell_Cu_rmin)
bottom_al_shell_Cu_segment_rmax = segment(bottom_al_shell_Cu_rmax)

bottom_al_shell_Al_segment = segment(bottom_al_shell_Al)
bottom_al_shell_Al_segment_rmin = segment(bottom_al_shell_Al_rmin)
bottom_al_shell_Al_segment_rmax = segment(bottom_al_shell_Al_rmax)

bottom_al_shell_Ch_segment = segment(bottom_al_shell_Ch)
bottom_al_shell_Ch_segment_rmin = segment(bottom_al_shell_Ch_rmin)
bottom_al_shell_Ch_segment_rmax = segment(bottom_al_shell_Ch_rmax)

bottom_air_Cu_segment = segment(bottom_air_Cu)
bottom_air_Cu_segment_rmin = segment(bottom_air_Cu_rmin)
bottom_air_Cu_segment_rmax = segment(bottom_air_Cu_rmax)

bottom_air_Al_segment = segment(bottom_air_Al)
bottom_air_Al_segment_rmin = segment(bottom_air_Al_rmin)
bottom_air_Al_segment_rmax = segment(bottom_air_Al_rmax)

bottom_air_Ch_segment = segment(bottom_air_Ch)
bottom_air_Ch_segment_rmin = segment(bottom_air_Ch_rmin)
bottom_air_Ch_segment_rmax = segment(bottom_air_Ch_rmax)

# Now, discretizations.

# This should discretize the 'flats' evenly cells with 'dr' spacing.
left_flat_copper_segment.equalArcDistrib(dr)
right_flat_copper_segment.equalArcDistrib(dr)
left_flat_aluminum_segment.equalArcDistrib(dr)
right_flat_aluminum_segment.equalArcDistrib(dr)
left_flat_charge_segment.equalArcDistrib(dr)
right_flat_charge_segment.equalArcDistrib(dr)
bottom_al_shell_Cu_segment.equalArcDistrib(dr_endplate)
bottom_al_shell_Cu_segment_rmin.equalArcDistrib(dz)
bottom_al_shell_Cu_segment_rmax.equalArcDistrib(dz)
bottom_al_shell_Al_segment.equalArcDistrib(dr_endplate)
bottom_al_shell_Al_segment_rmin.equalArcDistrib(dz)
bottom_al_shell_Al_segment_rmax.equalArcDistrib(dz)
bottom_al_shell_Ch_segment.equalArcDistrib(dr_endplate)

```

```

bottom_al_shell_Ch_segment_rmin.equalArcDistrib(dz)
bottom_al_shell_Ch_segment_rmax.equalArcDistrib(dz)
bottom_air_Cu_segment.equalArcDistrib(dr_endplate)
bottom_air_Cu_segment_rmin.equalArcDistrib(dz)
bottom_air_Cu_segment_rmax.equalArcDistrib(dz)
bottom_air_Al_segment.equalArcDistrib(dr_endplate)
bottom_air_Al_segment_rmin.equalArcDistrib(dz)
bottom_air_Al_segment_rmax.equalArcDistrib(dz)
bottom_air_Ch_segment.equalArcDistrib(dr_endplate)
bottom_air_Ch_segment_rmin.equalArcDistrib(dz)
bottom_air_Ch_segment_rmax.equalArcDistrib(dz)

# This will divide the inner and outer surfaces of the shell into NTheta cells.
inner_copper_segment.equalAngleDistrib(NTheta)
outer_copper_segment.equalAngleDistrib(NTheta)
inner_aluminum_segment.equalAngleDistrib(NTheta)
outer_aluminum_segment.equalAngleDistrib(NTheta)
inner_charge_segment.equalAngleDistrib(NTheta)
outer_charge_segment.equalAngleDistrib(NTheta)

# Make some slides. Or not, for now.
outer_copper_segment.slide("slide1") # use this only.
#inner_aluminum_segment.slide("slide")
#outer_aluminum_segment.slide("slide2") # and this one.
#inner_charge_segment.slide("slide")

# Final step: make the blocks.
copper_block = block('CopperBlock',
    iMin=left_flat_copper_segment,
    iMax=right_flat_copper_segment,
    iRes=len(inner_copper_segment),
    jMin=inner_copper_segment,
    jMax=outer_copper_segment,
    #jRes=len(left_flat_copper_segment),
    material='0003')

aluminum_block = block('AluminumBlock',
    iMin=left_flat_aluminum_segment,
    iMax=right_flat_aluminum_segment,
    iRes=len(inner_aluminum_segment),
    #jMin=inner_aluminum_segment,
    jMin=outer_copper_segment,
    jMax=outer_aluminum_segment,
    #jRes=len(left_flat_aluminum_segment),
    material='0002')

charge_block = block('ChargeBlock',
    iMin=left_flat_charge_segment,
    #iMax=right_flat_charge_segment,
    #iRes=len(inner_charge_segment),
    #feather='Distributed',
    jMin=outer_aluminum_segment,
    jMax=outer_charge_segment,
    #jRes=len(left_flat_charge_segment),
    material='0001')

```

```

aluminum_shell_Cu = block('AluminumShellCu',
    #iMin=left_flat_copper_segment,
    #iMax=bottom_al_shell_Cu_segment,
    iMax=copper_block.iMin().coarsen(),
    #iMin=bottom_al_shell_Cu_segment,
    jMin=bottom_al_shell_Cu_segment_rmin,
    jMax=bottom_al_shell_Cu_segment_rmax,
    #iRes=1,
    material='0004')

bottom_al_shell_Cu_segment_rmax.slide('slide1')

aluminum_shell_Al = block('AluminumShellAl',
    iMax=left_flat_aluminum_segment.coarsen(),
    iMin=bottom_al_shell_Al_segment.coarsen(),
    jMin=bottom_al_shell_Cu_segment_rmax,
    jMax=bottom_al_shell_Al_segment_rmax,
    material='0002')

aluminum_shell_Ch = block('AluminumShellCh',
    iMax=left_flat_charge_segment.coarsen(),
    iMin=bottom_al_shell_Ch_segment.coarsen(),
    jMin=bottom_al_shell_Al_segment_rmax,
    jMax=bottom_al_shell_Ch_segment_rmax,
    material='0002')

# Defining air blocks below the aluminum plate.
#air_Cu = block('AirCu',
    #iMin=bottom_al_shell_Cu_segment,
    #iMax=bottom_air_Cu_segment,
    #jMin=bottom_air_Cu_segment_rmin,
    #jMax=bottom_air_Cu_segment_rmax,
    #material='0004')

air_Cu = block('AirCu',
    iMax=aluminum_shell_Cu.iMin(),
    #iMax=bottom_air_Cu_segment,
    #dx1 = dr,
    jMin=bottom_air_Cu_segment_rmin,
    jMax=bottom_air_Cu_segment_rmax,
    material='0004')

bottom_air_Cu_segment_rmax.slide('slide1')

air_Al = block('AirAl',
    iMax=aluminum_shell_Al.iMin(),
    #iMax=bottom_air_Al_segment,
    #dx1 = dr,
    jMin=bottom_air_Cu_segment_rmax,
    jMax=bottom_air_Al_segment_rmax,
    material='0004')

air_Ch = block('AirCh',
    #iMin=aluminum_shell_Ch.iMin(),
    iMax=aluminum_shell_Ch.iMin(),
    #iMax=bottom_air_Ch_segment,

```

```

        #dx1 = dr,
        jMin=bottom_air_Al_segment_rmax,
        jMax=bottom_air_Ch_segment_rmax,
        material='0004')

# Finish up!
altair.endModel(dumpX3D=True,npes=30)

# vim:sw=4

```

## B Gurney Solution Script

The Gurney solution described in Section 4.3 was evaluated using a Python script. The value of  $\sqrt{E} = 2.90\text{km/s}$  was taken from the PBX-9404 entry in the table on page 51 of Ref. [13].

```

# This calculates the gurney solution for an imploding sphere.
from math import pi, sqrt

# Densities of the copper, aluminum and high explosive.
r_cu = 8.930 # g/cm3
r_al = 2.7 # g/cm3
r_he = 1.84 # g/cm3

# Volumes of the copper, aluminum, and high explosives
V_cu = 2.5 * pi # cm3
V_al = 6.0 * pi # cm3
#V_he = 59.0 / 3.0 * pi # cm3 # Error!
V_he = 178.0 * pi # cm3

# The Gurney velocity, sqrt(2*E), where E is the Gurney energy, or the amount of
# energy that is converted into kinetic energy of the liner and high explosive
# after detonation.
G = 2.9 # km/s

# b (beta in the reference I'm working off of) is the ratio of the outer and
# inner radius of the high explosive.
#b = (2.0 * sqrt(218.0)) / (5.0) # cm/cm
# Actually, beta might be the ratio of the outer and inner radius of the entire
# assembly. In that case, it would be
#b = (2.0 * sqrt(218.0)) / (3.5) # cm/cm
# From my better estimate of V_he, I calculated a "effective hemispherical
# radius" of the HE.
#radius_he_effective = (2. * 3./4. / pi * V_he)**(1./3.)
#print("radius_he_effective = %s cm" % radius_he_effective)
# Better "effective hemispherical radius" (earlier I was saying that the
# shaped charge was a *solid* hemisphere -- now I've removed the air and liner
# volume from the "effective hemisphere").
radius_he_effective = 7.87030458316
b = (radius_he_effective) / (3.5) # cm/cm

# a is a dimensionless constant of proportionality that is approximately unity in
# all the references I've seen.
#a = 1.0

# Mass of the liner. In the "real" shaped charge, there is an inner copper

```

```

# shell and outer aluminum shell, but here we'll say it's all aluminum.
M = r_al * (V_cu + V_al) # g

for a in [0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8,
         1.9, 2.0, 2.5, 3.0, 3.5]:
    #for a in [1.0,]:
        #for HE_vol_factor in [0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8,
                               #1.9, 2.0, 3.0, 4.0, 5.0, 10.0, 20.0, 30.0]:
            for HE_vol_factor in [1.0,]:

                # Mass of the high explosive.
                C = r_he * V_he * HE_vol_factor # g

                # A is an intermediate parameter that happens to be equal to the ratio of the
                # outer and inner velocity of the high explosive.
                A = (
                    (
                        M/C
                        + a*(M/C)*(b**2 - 1.0)
                        + (b**2 + 2.0*b + 3.0)/(4.0*(b**2 + b + 1.0))
                    )
                    / ((3.0*b**2 + 2.0*b + 1.0)/(4.0*(b**2 + b + 1.0)))
                )

                # Now get what we're here for: the velocity of the liner.
                Vi = (
                    G /
                    sqrt(A * ((M/C + (b**2 + 3.0*b + 6.0)/(10.0*(b**2 + b + 1.0)))/A +
                              A*(6.0*b**2 + 3.0*b + 1.0)/(10.0*(b**2 + b + 1.0)) - (3.0*b**2 +
                              4.0*b + 3.0)/(10.0*(b**2 + b + 1.0)))) # same units as 'G'.

                # Convert the liner velocity from km/s to cm/microsecs.
                Vi = 0.1 * Vi
                print("HE_vol_factor = %s, a = %s, Vi = %s cm/microsecs" % (HE_vol_factor, a, Vi))

```

**Rad-Hydro Verification**  
**(Scott Ramsey, mentor)**

# Development and Implementation of Radiation-Hydrodynamics Verification Test Problems

---

Computational Physics Student Summer Workshop

**Marcath, Matthew J.; Wang, Matthew Y.**

**7/20/2012**

Analytic solutions to the radiation-hydrodynamic equations are useful for verifying any large-scale numerical simulation software that solves the same set of equations. The one-dimensional, spherically symmetric Coggeshall #9 and #11 analytic solutions, cell-averaged over a uniform-grid have been developed to analyze the corresponding solutions from the Los Alamos National Laboratory Eulerian Applications Project radiation-hydrodynamics code xRAGE. These Coggeshall solutions have been shown to be independent of heat conduction, providing a unique opportunity for comparison with xRAGE solutions with and without the heat conduction module. Solution convergence was analyzed based on radial step size. Since no shocks are involved in either problem and the solutions are smooth, second-order convergence was expected for both cases. The global L1 errors were used to estimate the convergence rates with and without the heat conduction module implemented.

## Contents

Introduction.....	3
Verification .....	4
Initialization .....	5
Radiation-Hydrodynamic Equations.....	5
Coggeshall Equations.....	6
Discretization of Cell Quantities.....	10
Analysis and Data .....	10
Conclusions.....	12
Acknowledgements.....	13
References.....	14
Appendices.....	15
A: The Radiation Flux Expression.....	15
B: Heat Conduction Invariance.....	17
C: Initialization Discrepancies in xRAGE.....	19
D: Coggeshall Solution 9.....	20
Viewgraph Norms for Conserved Cell-Averages with and without the Heat Conduction Module Implemented .....	20
Viewgraph Norms for Unconserved Cell-Averages with and without the Heat Conduction Module Implemented .....	22
Viewgraph Norms with the Heat Conduction Module Not Implemented with and without Conserved Cell-Averages .....	24
Viewgraph Norms with the Heat Conduction Module Implemented with and without Conserved Cell-Averages .....	26
Conserved, Heat Conduction On .....	28
Non-Conserved, Heat Conduction On .....	32
Conserved, Heat Conduction Off.....	36
Non-Conserved, Heat Conduction Off.....	40
E: Coggeshall Solution 11 .....	44
Viewgraph Norms for Conserved Cell-Averages with and without the Heat Conduction Module Implemented .....	44

Viewgraph Norms for Unconserved Cell-Averages with and without the Heat Conduction Module Implemented .....	46
Viewgraph Norms with the Heat Conduction Module Not Implemented with and without Conserved Cell-Averages .....	48
Viewgraph Norms with the Heat Conduction Module Implemented with and without Conserved Cell-Averages .....	50
Conserved, Heat Conduction On .....	52
Non-conserved, Heat Conduction On .....	56
Conserved, Heat Conduction Off.....	60
Non-conserved, Heat Conduction Off .....	64

## Introduction

The Los Alamos National Laboratory Eulerian Applications Project radiation-hydrodynamics code xRAGE [1] solves a version of the radiation-hydrodynamics equations numerically [2]. Since the code solution is numerically derived, there are expected discretization errors associated with any solution found through iterating from initial conditions through time for a given spatial mesh. Since the spatial mesh and time stepping can be varied, the convergence rate of the code solution to an exact solution can easily be found. While the code does have adaptive mesh refinement and adaptive time stepping capabilities, only fixed time step and meshes for individual solutions will be used for ease of analysis.

Convergence rate analysis allows for quantified verification of the code. Verifying the code's numerical solution method ensures that the code is solving the intended equations correctly and as expected. Verification, however, does not insist that the code is producing solutions in agreement with experimental data. This analysis focuses on convergence rates as a function of spatial mesh size. Convergence rate refers to the rate at which the code approaches the exact solution over a series of spatial mesh reductions. Convergence can also be thought of as the rate at which error in the solution is reduced for each refinement of spatial mesh. Typically, the analytic solution used in convergence analysis is derived from the same set of equations that the code's numerical method is derived. There are expected rates of convergence for each solution method, and enabling different modules of the xRAGE code can affect these rates. In finding these rates, aspects of the code may be verified as functioning correctly. For smooth one-dimensional problems second-order convergence rates are expected of xRAGE's default hydrodynamics solver, which is a Godunov type solver. Shocks in the problem reduce the convergence rate to first order.

To analyze convergence of the xRAGE code it is convenient to use an analytically derived solution of the governing equations as the reference solution. The Coggeshall solutions nine and eleven [3] are a good choice for convergence analysis because they are smooth and physically meaningful. In the past, xRAGE has been successfully verified using solution eight, which has also been implemented in the regression test problem modules [4]. This analysis uses solutions nine and eleven. Both solutions are spherically symmetric and invariant with respect to radiation heat conduction, providing a unique opportunity to analyze heat conduction modules of xRAGE. Coggeshall created a closed-form radiation heat flux equation. The methods and assumptions that Coggeshall used to create a closed form of the radiation-hydrodynamics equations were scrutinized and confirmed to be reasonable in this analysis. Lie group theory was used to find analytic solutions to the set of equations. There are many possible analytic solutions, however only a fraction are physically meaningful.

xRAGE solves the radiation-hydrodynamics equations for a given discretization, meaning it produces discrete values for volume elements. These discrete values are analogous to cell-averaged quantities. It is necessary for practical analysis of xRAGE solutions to compare the xRAGE solutions to some form of cell-averaged quantities derived from the reference solution. There are many methods to find the cell-averages, however only the standard mean and conserved quantity weighting are included in this analysis. Conserved quantity weighting is thought to be most analogous to the modes with which xRAGE forms solutions. The algorithms that the code uses conserve mass, momentum, and energy. It has been proposed that computing cell-averages using the three conserved quantities will produce averages that most accurately show the convergence behavior of the code. The analysis presented compares the two methods of producing cell-averages.

## Verification

Code verification is the process of confirming that the code solves the prescribed equations as intended. xRAGE solves the radiation-hydrodynamics equations. Verification of xRAGE will involve determining whether xRAGE solves these equations properly and at the appropriate convergence rate. The convergence rate,  $n$ , of the spatial mesh is the exponential rate at which the code more accurately approximates the solution for subsequently smaller spatial mesh sizes with respect to the error metric,  $E$ . Convergence rate varies based on the numerical method used to solve the equations. For smooth problems with no shocks, second order convergence is expected for discretized spatial mesh refinements. However, slight deviations from this value are possible because of numerical errors.

$$E(\Delta x) \approx C\Delta x^n$$

$E = \text{error metric}$

$C = \text{convergence coefficient}$

$$(1) \quad \log(E) \approx n \log(\Delta x) + \log(C)$$

$\Delta x = \text{spatial mesh size}$

$n = \text{order of convergence}$

To verify a code, it is most useful to have an analytic solution to the equations to which the code produces an approximate solution. The analytic solution often must be found by making assumptions or simplifications in the radiation heat flux term and problem geometry. While the benefit of a numerical code package is to produce solutions to complicated problems, simple problems are useful for code verification purposes. The Coggeshall solutions [3] come from a few symmetry assumptions and the postulated closed-form of the radiation heat flux. The assumptions and simplifications must be justified and done appropriately to produce an analytic solution that is consistent with a numerical approximation.

The first step of verification is to compare the code solution to the expected analytic solution. Visualization of the code and analytic solutions is known as finding the “viewgraph norm;” while not a rigorous analysis of the code, this shows if the code solution is similar to the analytic solution. Once it is confirmed that the viewgraph norm is satisfactory, the convergence rate analysis should be performed. The convergence rate should perform as well as the solution algorithm predicts but can be different for each observed quantity and for each problem type. For Coggeshall solutions nine and eleven, the algorithm should produce solutions that converge at a second order rate in space.

Convergence rate is not constant for all mesh spacings. There are regions for very small and very large mesh sizes that converge irregularly or very little. In the case of very small spatial meshes, the convergence becomes dominated by the time step and not the spatial mesh. The case of very large spatial meshes has irregular and unpredictable convergence rates. Only between these two regions does a nearly constant spatial convergence rate occur. These regions occur at different size meshes for each numerical method and each test problem [5].

## Initialization

xRAGE presents difficulties at boundaries especially where problems that extend infinitely in space must be restricted to a finite region that is manageable by the code. The code treats the problem as symmetric about the origin. For infinite valued quantities near the origin, large numeric discrepancies are often observed. Usually these discrepancies dampen quickly to the analytic solution away from the origin, but numeric error in initialization can potentially propagate throughout a problem in time and space. At the outer boundary of the problem, frozen conditions are applied. A frozen outer boundary cell condition fixes the initial conditions at those outer cells for all time steps. This is a necessary approximation, but it is evident even in a viewgraph norm that it does not agree well with the analytic solution. However, in most cases the solution quickly approaches the analytic away from the boundary.

The outer boundary condition is non-physical and a result of simplifications that must be made to initialize the problem. The Coggeshall solutions are defined for all space, but must be restricted to a finite space in xRAGE. This restriction artificially imposes a false boundary condition that does not exist in the exact solution. Since the code solution is physically applicable in all ranges excluding near the outer boundary, it is appropriate to compare only these portions of the analytic and code solution.

## Radiation-Hydrodynamic Equations

xRAGE and Coggeshall solve some form of the non-relativistic radiation-hydrodynamic equations. The equations consist of two sets of terms: the material state terms and the radiation heat terms. The non-relativistic radiation-hydrodynamic equations are as follows.

### Conservation of Mass

$$(2) \quad \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0$$

### Conservation of Momentum

$$(3) \quad \frac{\partial}{\partial t} \left( \rho \vec{u} + \frac{\vec{F}}{c^2} \right) + \nabla P_{mat} + \nabla \cdot (\rho \vec{u} \vec{u} + P_{rad}) = 0$$

### Conservation of Energy

$$(4) \quad \frac{\partial}{\partial t} \left( \frac{1}{2} \rho u^2 + E_{mat} + E_{rad} \right) + \nabla \cdot \left[ \left( \frac{1}{2} \rho u^2 + E_{mat} + P_{mat} \right) \vec{u} + \vec{F} \right] = 0$$

Where the subscript “mat” refers to the gas material and subscript “rad” refers to the radiation.

$$c = \text{speed of light}$$

$$\rho = \rho(\vec{r}, t) \equiv \text{material density}$$

$$P = P(\vec{r}, t) \equiv \text{pressure}$$

$$\vec{u} = \vec{u}(\vec{r}, t) \equiv \text{flow velocity}$$

$$\vec{F} = \vec{F}(\vec{r}, t) \equiv \text{heat flux}$$

$$E = E(\vec{r}, t) \equiv \text{energy density}$$

The radiation-hydrodynamic equations can be reduced to the Euler hydrodynamic equations [6]. The equations are reduced to (5), (6), and (7) by neglecting the radiation energy density and radiation pressure. Also  $\frac{\vec{F}}{c^2} \ll 1$  is assumed in (3), so this term is neglected. Conversely, the Coggeshall solutions used in this analysis have the property of heat conduction invariance which ultimately reduces the set of radiation-hydrodynamic equations to the hydrodynamic equations.

#### Conservation of Mass

$$(5) \quad \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0$$

#### Conservation of Momentum

$$(6) \quad \frac{\partial}{\partial t} (\rho \vec{u}) + \nabla P_{mat} + \nabla \cdot (\rho \vec{u} \vec{u}) = 0$$

#### Conservation of Energy

$$(7) \quad \frac{\partial}{\partial t} \left( \frac{1}{2} \rho u^2 + E_{mat} \right) + \nabla \cdot \left[ \left( \frac{1}{2} \rho u^2 + E_{mat} + P_{mat} \right) \vec{u} + \vec{F} \right] = 0$$

$$\nabla \cdot \vec{F} = 0$$

## Coggeshall Equations

The radiation-hydrodynamic equations can be reduced to the form used by Coggeshall to find analytic solutions for a one-dimensional case [7]. Coggeshall assumes the ideal gas law.

$$P = \Gamma \rho T$$

$$\Gamma = \text{gas constant}$$

$$\gamma = \text{adiabatic exponent}$$

#### Conservation of Mass

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0$$

$$\nabla \cdot f = \frac{1}{r} \frac{\partial(rf)}{\partial r} \text{ for one-dimensional cylindrical coordinates}$$

$$\nabla \cdot f = \frac{1}{r^2} \frac{\partial(r^2 f)}{\partial r} \text{ for one-dimensional spherical coordinates}$$

The following will be carried out in spherical symmetry.

$$\frac{\partial \rho}{\partial t} + \frac{1}{r^2} \frac{\partial (r^2 \rho \bar{u})}{\partial r} = 0$$

$$(8) \quad \rho_t + \frac{2\rho u}{r^2} + \rho_r u + \rho u_r = 0$$

### Conservation of Momentum

$$\frac{\partial}{\partial t} (\rho \bar{u}) + \nabla P + \nabla \cdot \rho \bar{u} \bar{u} - \bar{u} \left[ \frac{\partial \rho}{\partial t} + \nabla (\rho \bar{u}) \right] = 0$$

$$\bar{u} \rho_t + \rho \bar{u}_t + \nabla P_{mat} + \bar{u} (\nabla \rho \cdot \bar{u}) + 2\bar{u} \rho (\nabla \cdot \bar{u}) - \bar{u} \rho_t - \bar{u} (\rho \nabla \cdot \bar{u}) - \bar{u} (\bar{u} \cdot \nabla \rho) = 0$$

$$\rho \bar{u}_t + \nabla P + \bar{u} \rho (\nabla \cdot \bar{u}) = 0$$

$$\rho \left( \frac{\partial}{\partial t} + \bar{u} \cdot \nabla \right) \bar{u} + \nabla P_{mat} = 0$$

$$\rho \bar{u}_t + \rho \bar{u} \bar{u}_r + \frac{\partial}{\partial r} (\Gamma \rho T) = 0$$

$$(9) \quad \bar{u}_t + \bar{u} \bar{u}_r + \frac{\Gamma \rho_r T}{\rho} + \Gamma T_r = 0$$

### Conservation of Energy

$$\frac{\delta E}{\delta t} + \bar{u} \cdot \nabla E + \frac{1}{\rho} P \nabla \cdot \bar{u} + \frac{1}{\rho} \nabla \cdot \vec{F} = 0$$

$$E = \frac{\Gamma}{\gamma - 1} T$$

$$\frac{\Gamma}{\gamma - 1} T_t + \bar{u} \frac{\partial}{\partial r} \left( \frac{\Gamma}{\gamma - 1} T \right) + \frac{\Gamma T}{\rho r^2} \frac{\partial}{\partial r} (r^2 \bar{u}) + \frac{1}{\rho r^2} \frac{\partial}{\partial r} (r^2 \vec{F}) = 0$$

$$(10) \quad \frac{\Gamma}{\gamma - 1} (T_t + \bar{u} T_r) + \Gamma T \bar{u}_r + \frac{2\Gamma T \bar{u}}{r} + \frac{\vec{F}_r}{\rho} + \frac{2\vec{F}}{\rho r} = 0$$

From these equations and a closure model for the radiation heat flux,  $\vec{F} = -K(\rho, T) \nabla T$  (appendix A), Coggeshall uses Lie group theory to find a set of analytic solutions. The general forms of the equations contain a geometry factor  $k$  ( $k = 1$  for cylindrical and  $k = 2$  for spherical), the adiabatic exponent  $\gamma$ , and conduction constants  $\alpha$  and  $\beta$ . The specific heat at constant volume,  $C_V$ , was set to unity in the code input deck, so specific internal energy equal to temperature.

$$(11) \quad e = C_V T$$

### Solution #9

$$\rho(r, t) = \rho_0 r^{-(2\beta+k+7)/\alpha} t^{-2[\alpha(k+1)-2\beta-k-7]/\alpha[2+(\gamma-1)(k+1)]}$$

$$u(r, t) = \frac{2r}{t[2+(\gamma-1)(k+1)]}$$

$$T(r, t) = \frac{2\alpha(\gamma-1)(k+1)r^2}{t^2\Gamma[2+(\gamma-1)(k+1)]^2(2\alpha-2\beta-k-7)}$$

For the analysis presented here, let  $\alpha = -1$ ,  $\beta = 2$ ,  $\gamma = 5/3$ , and  $k = 2$ .  $\alpha$  and  $\beta$  are experimentally derived quantities [3].  $\gamma=5/3$  indicates a monatomic gas.

$$(12) \quad \rho(r, t) = \rho_0 r^{13} t^{-8}$$

$$(13) \quad u(r, t) = \frac{r}{2t}$$

$$(14) \quad T(r, t) = \frac{r^2}{60\Gamma t^2}$$

$$(15) \quad P(r, t) = \frac{\rho_0}{60} r^{15} t^{-10}$$

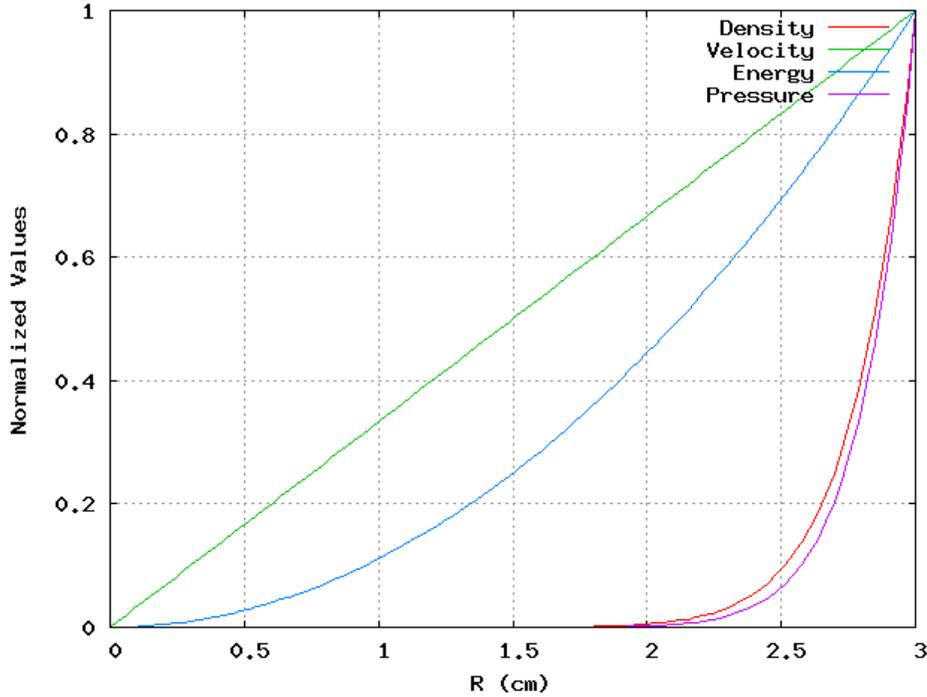


Figure 1: Coggeshall nine normalized solutions for key flow variables.

### Solution #11

$$\rho(r, t) = \rho_0 r^{(\gamma-1)(k+1)-2} t^{1-k-(\gamma-1)(k+1)}$$

$$u(r, t) = r/t$$

$$T(r, t) = T_0 r^{2-(\gamma-1)(k+1)} t^{-2}$$

With the constraint,  $\alpha = \beta + 4 + (k - 1)/[2 - (\gamma - 1)(k + 1)]$

For the analysis presented here, let  $\alpha = -1$ ,  $\beta = 2$ , and  $k = 2$  then  $\gamma = 12/7$ .

$$(16) \quad \rho(r, t) = \rho_0 r^{1/7} t^{-22/7}$$

$$(17) \quad u(r, t) = r/t$$

$$(18) \quad T(r, t) = T_0 r^{-\frac{1}{7}} t^{-2}$$

$$(19) \quad P(r, t) = \Gamma T_0 \rho_0 t^{-36/7}$$

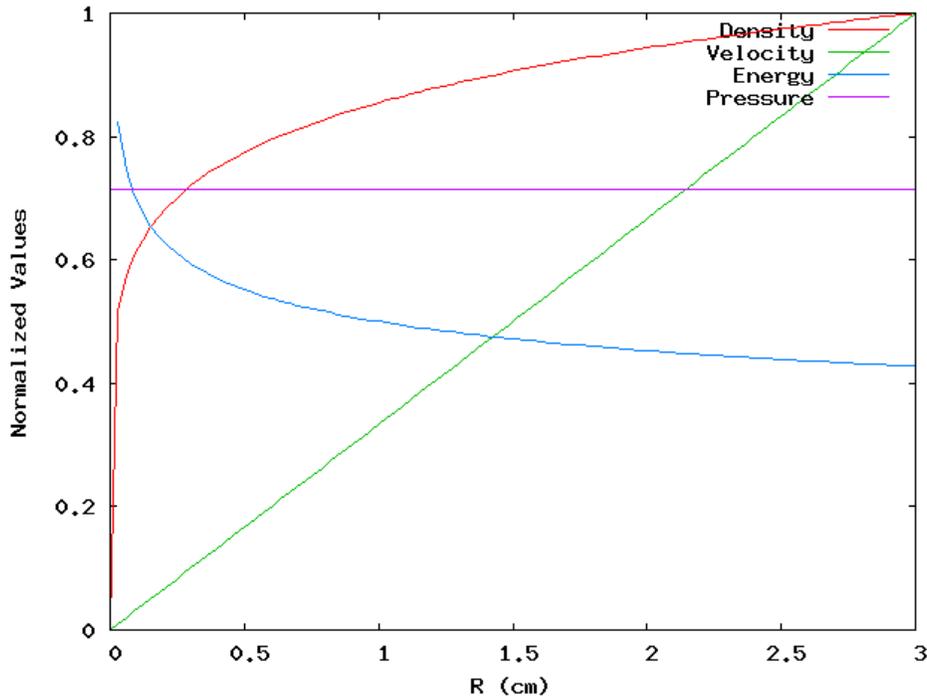


Figure 2: Coggeshall eleven normalized solutions for key flow variables.

With appropriate constants applied, these solutions are physically meaningful. The velocity solutions increase linearly through space and decrease inversely with time. This velocity behavior, with the density and temperature behaviors, are consistent with an expanding and cooling system for both solutions. Solution nine pressure increases with power law variation

with respect to radius, but decreases inversely with time. Whereas, solution eleven pressure decreases with power law variation with respect to time and is invariant in space. These solutions have the feature of being invariant with respect to heat conduction as indicated by the  $\nabla \cdot \vec{F} = 0$  condition that Coggeshall assumes. The heat flux into a volume is equal to the heat out of a volume, so there is no net effect on the volume by heat (appendix B).

## Discretization of Cell Quantities

Given a spatial discretization, xRAGE produces discrete solutions: one zone-averaged value for each flow variable for each mesh zone. Since the code solution is discretized, a method must be introduced to compose corresponding discrete values from the analytic solutions. Using zone center flow variable values directly from the reference solution can produce misleading convergence results [4]. It is more useful to compare a zone-averaged quantity with the code solution, because it is more analogous to the code solution output. This analysis uses cell volume-averaged quantities to compare with the code solution. Cell-averages used in this analysis were produced by weighting the average with conserved quantities and by a standard mean. The conserved quantities refer to the quantities that xRAGE conserves, which are mass, momentum, and energy. Cell-averaged density is conserved since it is already a function of a conserved quantity, mass.

**Table 1: Conserved and non-conserved cell-averaging equations over individual mesh zones used in the analysis.**

Cell-Average	Non-conserved	Conserved
Density	$\bar{\rho} = \frac{\int r^2 dr \rho(r)}{\int r^2 dr}$	$\bar{\rho} = \frac{\int r^2 dr \rho(r)}{\int r^2 dr}$
Velocity	$\bar{u} = \frac{\int r^2 dr u(r)}{\int r^2 dr}$	$\bar{u} = \frac{\int r^2 dr \rho(r) u(r)}{\int r^2 dr \rho(r)}$
Specific Internal Energy	$\bar{e} = \frac{\int r^2 dr e(r)}{\int r^2 dr}$	$TE = 4\pi \int r^2 dr \rho(r) [e(r) + \frac{1}{2} u^2(r)]$ $\bar{e} = \frac{(\frac{TE}{\bar{\rho}} - \frac{1}{2} \bar{u}^2)}{\int r^2 dr}$ [8]
Pressure	$\bar{P} = \frac{\int r^2 dr \Gamma \rho(r) T(r)}{\int r^2 dr}$	$\bar{P} = \Gamma \overline{\rho_{cons} e_{cons}}$

The cell-averaged quantities are used to provide initial conditions to input into xRAGE for each mesh cell. The reason for using cell-averaged quantities for initialization is to provide a quantity that is most similar to a spatially discretized quantity that the code would produce. Also, the cell averages are used as the exact solution to compare with the code solution and form error quantities. The cell-average method used for the initial conditions must be used in analysis of the code solution.

## Analysis and Data

xRAGE solutions for the quantities of interest were compared with their analytically derived counterparts to produce a volume weighted global L1 norm (21) for each spatial mesh size for a given time step. The global L1 norm is derived from the continuous L1 norm (20).

Since the outer boundary condition is not consistent with the exact solution of the outer boundary, points beyond a radius of 2.75 are not included in the convergence analysis. The plots for the L1 error norms at each time step as well as plots for convergence analysis at 3.0 s are attached in appendix D and E. The linear curve fits in the convergence analysis plots provide a slope that is used to represent the convergence rate for the given quantity at a given time in the solution. It was found that 50, 100, and 200 mesh zones appear to be in the convergence region dominated by the spatial mesh size for most state quantities.

$$(20) \quad L_1 = \frac{\int_{x_{min}}^{x_{max}} dV |f_{exact} - f_{xRAGE}|}{\int_{x_{min}}^{x_{max}} dV}$$

$$(21) \quad Error \equiv L_1 \approx \frac{\sum_{i=1}^N |f_{exact,i} - f_{xRAGE,i}| (x_{max,i}^3 - x_{min,i}^3)}{\sum_{i=1}^N (x_{max,i}^3 - x_{min,i}^3)}$$

**Table 2: Convergence rates at time 2.0 s for key quantities**

t = 2.0 s	Heat Conduction Module	Cell-Averages	Density	Velocity	Internal Energy	Pressure
Coggeshall #9	On	Conserved	2.19734	2.2009	1.94221	2.14571
		Unconserved	2.1924	2.19378	1.93846	2.14514
	Off	Conserved	2.19734	2.20084	1.94225	2.14571
		Unconserved	2.1924	2.19372	1.9385	2.14514
Coggeshall #11	On	Conserved	3.51083	2.60838	1.22401	2.09167
		Unconserved	3.19004	0.94538	1.9552	2.495
	Off	Conserved	3.76259	2.71418	1.23488	2.09128
		Unconserved	3.18902	0.96388	1.95858	2.48846

**Table 3: Convergence rates at time 3.0 s for key quantities**

t = 3.0 s	Heat Conduction Module	Cell-Averages	Density	Velocity	Internal Energy	Pressure
Coggeshall #9	On	Conserved	2.22078	2.9966	1.8986	2.17192
		Unconserved	2.2165	2.28098	1.89169	2.17158
	Off	Conserved	2.22078	2.29951	1.89866	2.17192
		Unconserved	2.2165	2.28084	1.89176	2.17158
Coggeshall #11	On	Conserved	2.80453	2.43544	1.86782	2.01101
		Unconserved	2.65554	0.82368	2.1264	1.62433
	Off	Conserved	3.1982	2.65904	1.8974	2.0056
		Unconserved	2.60006	0.905	2.14088	1.60959

Table 2 shows that at time 2.0 s the convergence rate of the Coggeshall nine solution is affected minimally by the use of the heat conduction module for both conserved and unconserved cell-averages. However, the Coggeshall eleven solution convergence rate is reduced by the use of the heat conduction module for both conserved and unconserved cell-averages for the majority of the quantities. The slight reduction in convergence rate suggests that the heat conduction module may have a negative effect on convergence. This effect is likely due

to the propagation of errors through the problem in space and time, caused by wall heating-like phenomenon near the origin.

In almost all cases the convergence rate is higher when using conserved cell-averages. Also, the convergence rates are usually second-order when using conserved cell averages. This is likely a result of the method being more consistent with the code methods; thus, it is more consistent with the order of accuracy. Coggeshall eleven velocity solutions with unconserved cell-averages exhibit less than first-order convergence, while the conserved counterparts exhibit second-order or higher rates. However, in Table 3 for Coggeshall eleven specific internal energy solutions at time 3.0 s the conserved cell-averages converge just below second-order, but the unconserved converge slightly above second order. The difference in convergence rates at this time dump for the specific internal energy solutions is minimal and one can reasonably accept that the convergence rate is still second-order for conserved averages. The only exception to the appearance of second-order convergence for conserved averages occurs at time 2.0 s for Coggeshall eleven specific internal energy solution. Both the higher convergence and the second-order behavior suggest that using conserved cell-averages is the best method to analyze convergence.

## Conclusions

xRAGE displayed second-order spatial convergence for both Coggeshall nine where  $\alpha = -1$ ,  $\beta = 2$ ,  $\gamma = 5/3$ , and  $k = 2$  and eleven problems where  $\alpha = -1$ ,  $\beta = 2$ ,  $k = 2$ , and  $\gamma = 12/7$ . Convergence analysis is best performed using conserved cell-averages for the majority of state quantities. There is a small reduction in convergence rate for Coggeshall solution eleven. The analysis of xRAGE convergence using Coggeshall nine and eleven solutions extended and reinforced existing results from the Coggeshall eight solution.

This work could be refined by improving the initialization of the flow variables in Coggeshall problem eleven near the origin. It would be especially useful for the initialization to closely match the analytic solution when the heat conduction module is being used. Investigation into the few flow variable convergence rates that were either far below or above second order is necessary. Convergence analysis with respect to the time step should be done. Also, extension of the analysis to two and three dimensions would be useful.

## **Acknowledgements**

This work was performed under the auspices of the United States Department of Energy by Los Alamos National Security, LLC, at Los Alamos National Laboratory under contract DE-AC52-06NA25396. The authors acknowledge the support of the US Department of Energy Advanced Strategic Computing Program Verification Project under project leader S. Doebbling, mentor S. Ramsey, and the support of the Los Alamos National Laboratory Computational Physics Student Summer Workshop under the direction of S. Runnels. The authors thank R.C. Hendon for valuable insights on these topics.

## References

- [1] M. Gittings, R. Weaver, M. Clover, T. Betlach, N. Byrne, R. Coker, E. Dendy, R. Hueckstaedt, K. New, W. Oakes, D. Ranta and R. Stefan, *The RAGE radiation-hydrodynamics code*, Computational Science Disc. 1, 015005, 2008.
- [2] Los Alamos National Laboratory Eulerian Applications Project, "xRAGE Users Manual (v1109.03)," LA-CP-11-00643, 2012.
- [3] S. Coggeshall, "Analytic Solutions of Hydrodynamics equations," vol. A, no. 3, 1991.
- [4] F. X. Timmes, M. R. Clover, J. R. Kamm and S. D. Ramsey, "On a Cell-Averaged Solution of the Coggeshall #8 Problem," LA-UR-09-004438, 2009.
- [5] R. Hendon and S. Ramsey, "Radiation Hydrodynamics Test Problems with Linear Velocity Profiles," Los Alamos National Laboratory, 2012.
- [6] R. Drake, *High-Energy-Density Physics: Fundamentals, Inertial Fusion, and Experimental Astrophysics*, Berlin: Springer, 2006.
- [7] S. Coggeshall and R. A. Axford, "Lie group invariance properties of radiation hydrodynamics equations and their associated similarity solutions," *American Institute of Physics: Physics of Fluids*, vol. 29, 1986.
- [8] S. D. Ramsey, J. R. Kamm and J. H. Bolstad, "The Guderley Problem Revisited," Los Alamos National Laboratory, 2006.
- [9] D. Mihalas and B. Weibel-Mihalas, *Foundations of Radiation Hydrodynamics*, Mineola, New York: Dover Publications, Inc., 1984.

## Appendices

### A: The Radiation Flux Expression

Coggeshall states that “heat flux  $F$  can be represented through a radiation diffusion approximation” [3]. The radiation diffusion approximation is an extension of Fick’s law.

$$\text{Fick's Law } \vec{F} = -D\nabla\phi$$

$$\vec{F} = \text{diffusion flux}$$

$$D = \text{diffusion coefficient}$$

$$\phi = \text{Concentration}$$

Fick’s law is used in conjunction with the Stefan-Boltzmann law which states that the power of the radiation emitted from a black-body is proportional to its temperature,  $T$ , to the fourth power [9]. In this case  $\phi \equiv \text{power of the radiation emitted} \propto T^4$ .

$$\vec{F} = -D\nabla T^4$$

At this point only the diffusion coefficient,  $D$ , remains to be rationalized. The diffusion coefficient should have units [ $Wm^{-2}K^{-4}$ ]. Classically the diffusion coefficient is composed of three main parts that are then multiplied: a factor that originates from classical electrodynamics where the radiation pressure is proportional to the internal energy density of a black-body; the characteristic velocity of the radiation,  $c$ ; and the characteristic length scale of the radiation,  $\lambda$ , or its mean free path in a given material. The energy and temperature units are accounted for by use of a radiation constant,  $a$ .

$$D = \frac{1}{3}c\lambda a$$

$$(22) \quad \vec{F} = -\frac{1}{3}c\lambda a\nabla T^4$$

The radiation mean free path  $\lambda$ , a function of density and temperature, is related to the Rosseland mean opacity,  $\kappa_\nu$  [9], and the material density,  $\rho$ .

$$\lambda = (\kappa_\nu\rho)^{-1}$$

The Rosseland mean opacity is also a function of density and temperature. A power-law fit of it in these variables is then assumed yielding the following.  $\alpha$  and  $\beta$  were determined experimentally [7].

$$(23) \quad \lambda(\rho, T) = \lambda_0\rho^\alpha T^\beta$$

Coggeshall uses (22) and (23) to obtain a form analogous to Fourier's law in one dimensional spherical coordinates [3].

$$\vec{F} = -\frac{1}{3}c\lambda a \frac{\partial}{\partial r} T^4$$

If the problem is spherically symmetric then the equation only has radial variance.

$$\vec{F} = -\frac{4}{3}c\lambda\alpha T^3 \frac{\partial T}{\partial r} \hat{r}$$

Otherwise direction must be retained.

$$\vec{F} = -\frac{4}{3}c\lambda\alpha T^3 \nabla T$$

The prefactors can be combined to obtain a form like Fourier's law where  $K(\rho, T)$  is the conductivity of the material.

$$\vec{F} = -K(\rho, T) \nabla T$$

## B: Heat Conduction Invariance

The Coggleshall eight solution for a spherical geometry,  $\alpha=-1$ ,  $\beta=2$ , and  $\gamma=5/3$  has been stated to be independent of conduction in literature [4], “The heat conduction is such that the area-weighted flux on each cell face is equal, that is, conduction moves as much energy into a cell as it removes.” A discussion of this feature of the Coggleshall eight solution follows.

To prove the statement, the area-weighted heat flux through a cell face in spherical symmetry must be shown to equal to the area-weighted heat flux outward through the opposing cell face. This implies that the heat deposited by conduction to the cell through the one face is removed through the other face by conduction at a particular instance in time, shown in Figure 1.

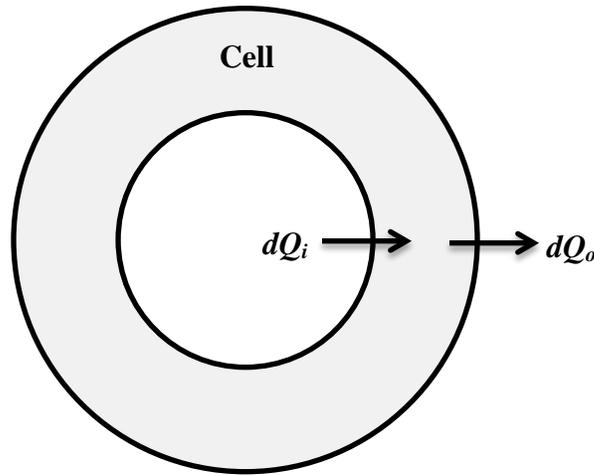


Figure 3: Area-weighted heat flux into and out of a cell is equal

Show that  $Q_{in} = Q_{out}$ .

$Q \equiv$  heat conducted

$A \equiv$  Cell face area

$F \equiv$  heat flux =  $\frac{Q}{A}$

$FA \equiv$  area – weighted flux =  $Q$

$F_{in}A_{in} = Q_{in} = Q_{out} = F_{out}A_{out}$

For spherical symmetry  $A = 4\pi r^2$ .

$F_{in}r_{in}^2 = F_{out}r_{out}^2$

$F = -\left(\frac{c\lambda}{3}\right) \nabla aT^4$  [3]

$r_{in}^2 \lambda_{in} \nabla T_{in}^4 = r_{out}^2 \lambda_{out} \nabla T_{out}^4$

$$\lambda(\rho, T) = \lambda_0 \rho^\alpha T^\beta \quad [3]$$

$$\lambda(\rho, T) = \lambda_0 \rho^{-1} T^2$$

$$\rho(r, t) = \rho_0 \frac{r^{\frac{1}{7}}}{t^{\frac{22}{7}}} \quad T(r, t) = T_0 (r^{-\frac{1}{7}}) (t^{-\frac{13}{7}}) \quad [4]$$

$$\lambda = \lambda_0 (\rho_0^{-1} r^{-\frac{1}{7}} t^{\frac{22}{7}}) \left( T_0^2 r^{-\frac{2}{7}} t^{\frac{13}{7}} \right)$$

$$\nabla T^4 = -\frac{4}{7} T_0^4 r^{-\frac{11}{7}} t^{-\frac{13}{7}}$$

$$\begin{aligned} r_{in}^2 \lambda_0 (\rho_0^{-1} r_{in}^{-\frac{1}{7}} t^{\frac{22}{7}}) \left( T_0^2 r_{in}^{-\frac{2}{7}} t^{\frac{13}{7}} \right) \left( -\frac{4}{7} T_0^4 r_{in}^{-\frac{11}{7}} t^{-\frac{13}{7}} \right) \\ = r_{out}^2 \lambda_0 (\rho_0^{-1} r_{out}^{-\frac{1}{7}} t^{\frac{22}{7}}) \left( T_0^2 r_{out}^{-\frac{2}{7}} t^{\frac{13}{7}} \right) \left( -\frac{4}{7} T_0^4 r_{out}^{-\frac{11}{7}} t^{-\frac{13}{7}} \right) \end{aligned}$$

$$r_{in}^2 \left( r_{in}^{-\frac{1}{7}} \right) \left( r_{in}^{-\frac{2}{7}} \right) \left( r_{in}^{-\frac{11}{7}} \right) = r_{out}^2 \left( r_{out}^{-\frac{1}{7}} \right) \left( r_{out}^{-\frac{2}{7}} \right) \left( r_{out}^{-\frac{11}{7}} \right)$$

$$1 = 1$$

The statement is true.

This method was also used to show that both Coggeshall solutions #9 and #11 are invariant to heat conduction.

## C: Initialization Discrepancies in xRAGE

There are three options for initialization parameters for the xRAGE code for material regions (computational zones):

1. rhoreg and siereg (Density and Specific Internal Energy)
2. rhoreg and tevreg (Density and Temperature)
3. prsreg and tevreg (Pressure and Temperature)

All initializations of xRAGE also require the initialization of velocity. xRAGE then uses equations of state and the provided parameters to calculate the rest the initialization values. Although the input values are provided by the user, it has been observed that xRAGE uses the equation of state to recalculate temperature and pressure when given as initialization parameters. The recalculated values do not exactly match the original input values. The order of accuracy is of order  $10^{-10}$ . Although the difference is very small, it does create errors in the simulation.

The xRAGE code manual [2] recognizes and discusses an analytic equation of state implementation error that may be responsible for these phenomena.

## D: Coggeshall Solution 9

### Viewgraph Norms for Conserved Cell-Averages with and without the Heat Conduction Module Implemented

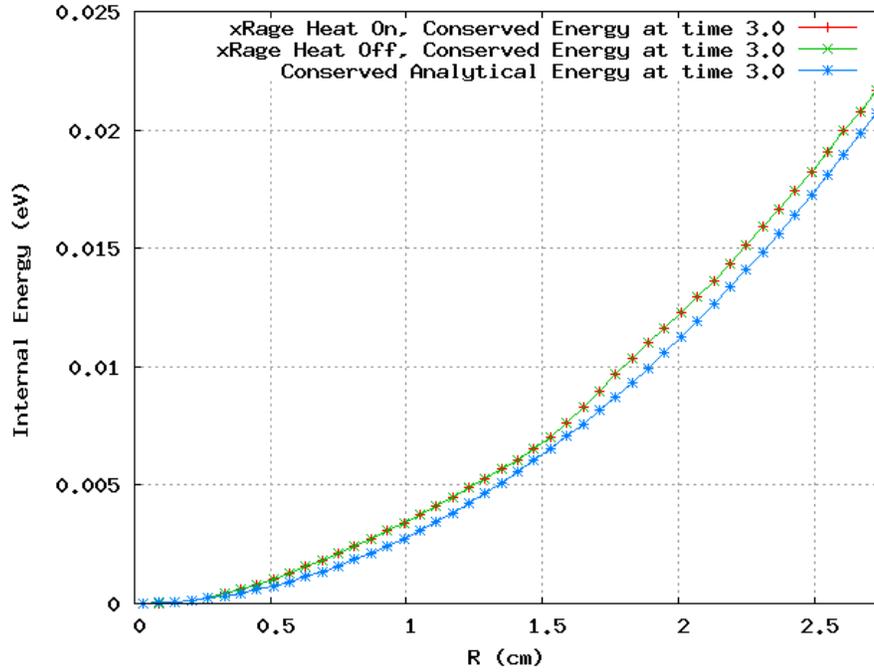


Figure 4: Viewgraph norm of the xRAGE specific internal energy solution with and without the heat conduction module implemented for conserved cell-average values at  $t=3.0$  s

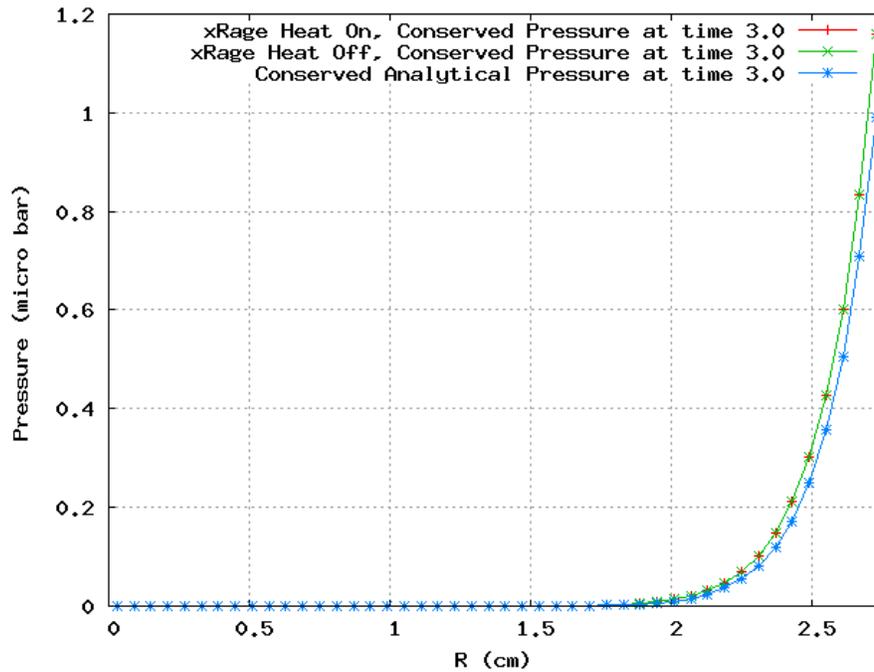


Figure 5: Viewgraph norm of the xRAGE pressure solution with and without the heat conduction module implemented for conserved cell-average values at  $t=3.0$  s

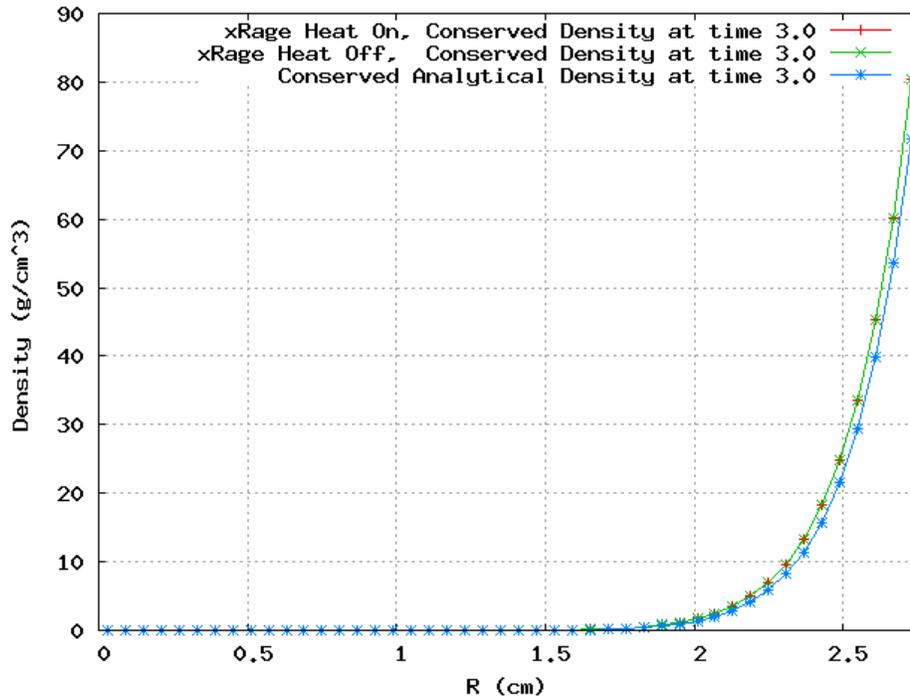


Figure 6: Viewgraph norm of the xRAGE density solution with and without the heat conduction module implemented for conserved cell-average values at  $t=3.0$  s

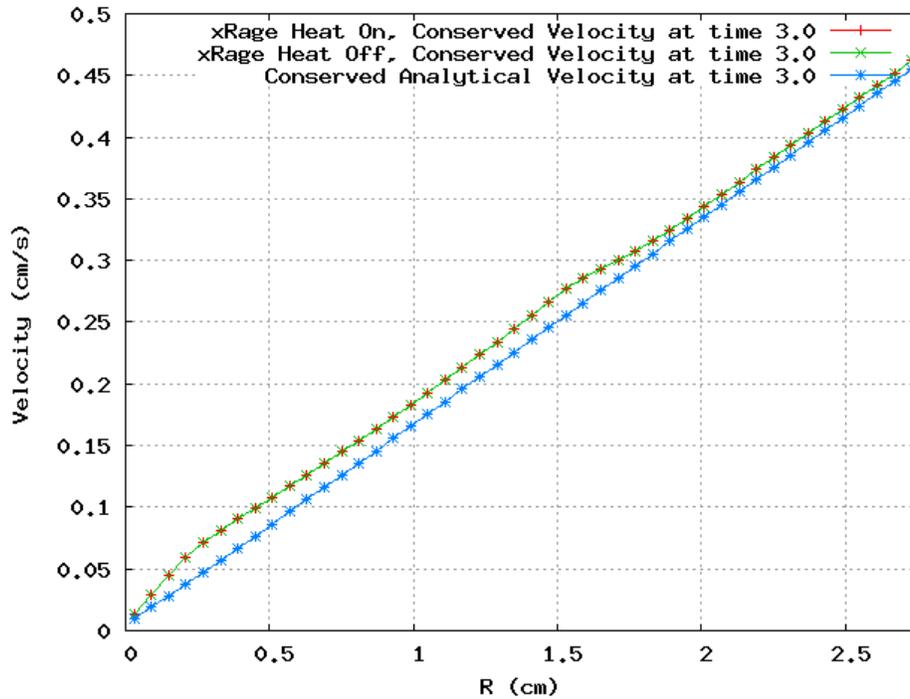


Figure 7: Viewgraph norm of the xRAGE velocity solution with and without the heat conduction module implemented for conserved cell-average values at  $t=3.0$  s

## Viewgraph Norms for Unconserved Cell-Averages with and without the Heat Conduction Module Implemented

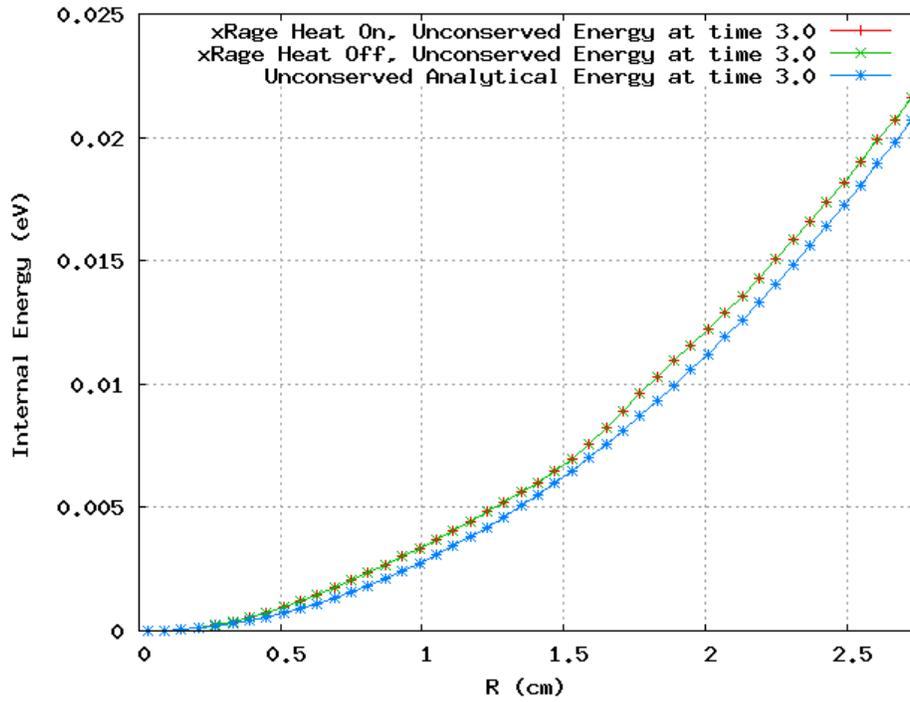


Figure 8: Viewgraph norm of the xRAGE specific internal energy solution with and without the heat conduction module implemented for unconserved cell-average values at  $t=3.0$  s

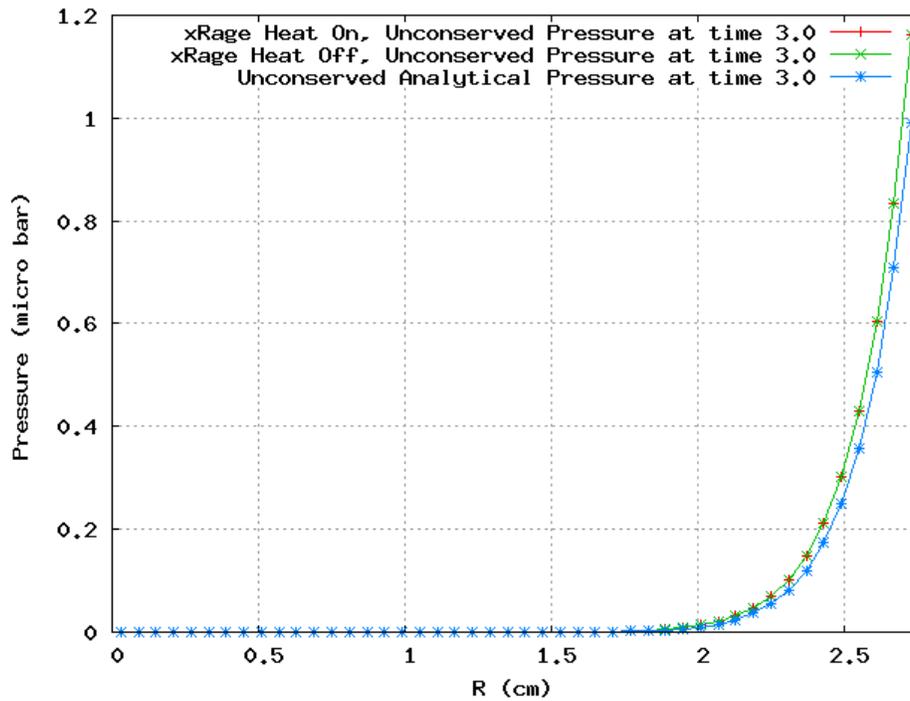


Figure 9: Viewgraph norm of the xRAGE pressure solution with and without the heat conduction module implemented for unconserved cell-average values at  $t=3.0$  s

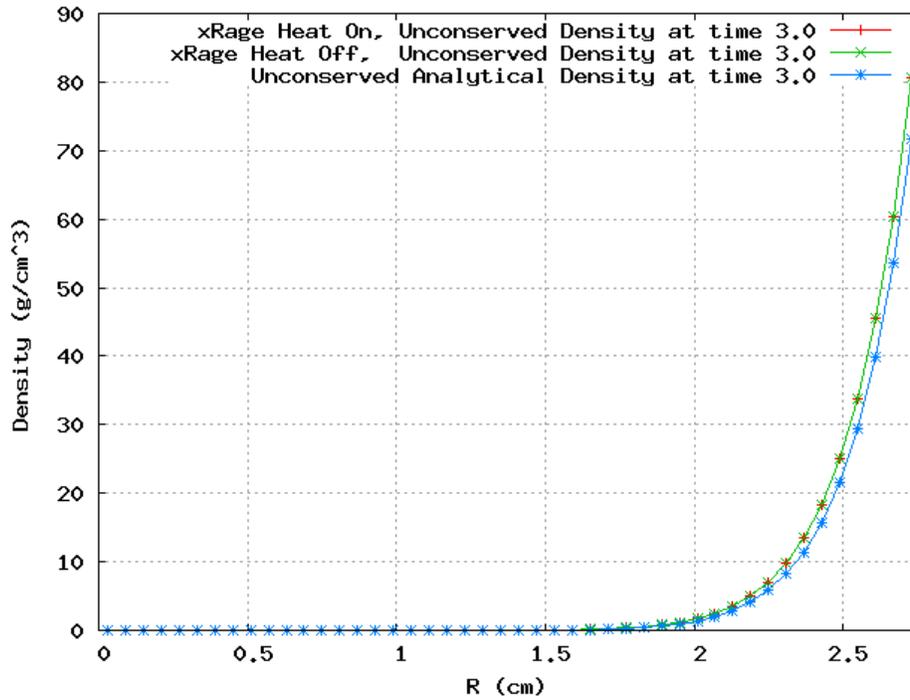


Figure 10: Viewgraph norm of the xRAGE density solution with and without the heat conduction module implemented for unconserved cell-average values at  $t=3.0$  s

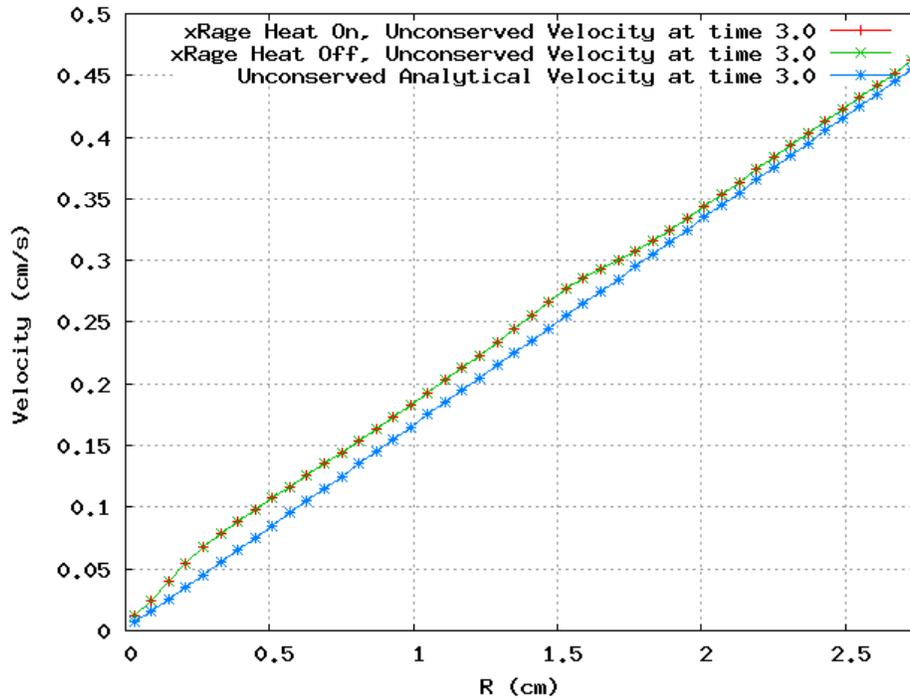


Figure 11: Viewgraph norm of the xRAGE velocity solution with and without the heat conduction module implemented for unconserved cell-average values at  $t=3.0$  s

## Viewgraph Norms with the Heat Conduction Module Not Implemented with and without Conserved Cell-Averages

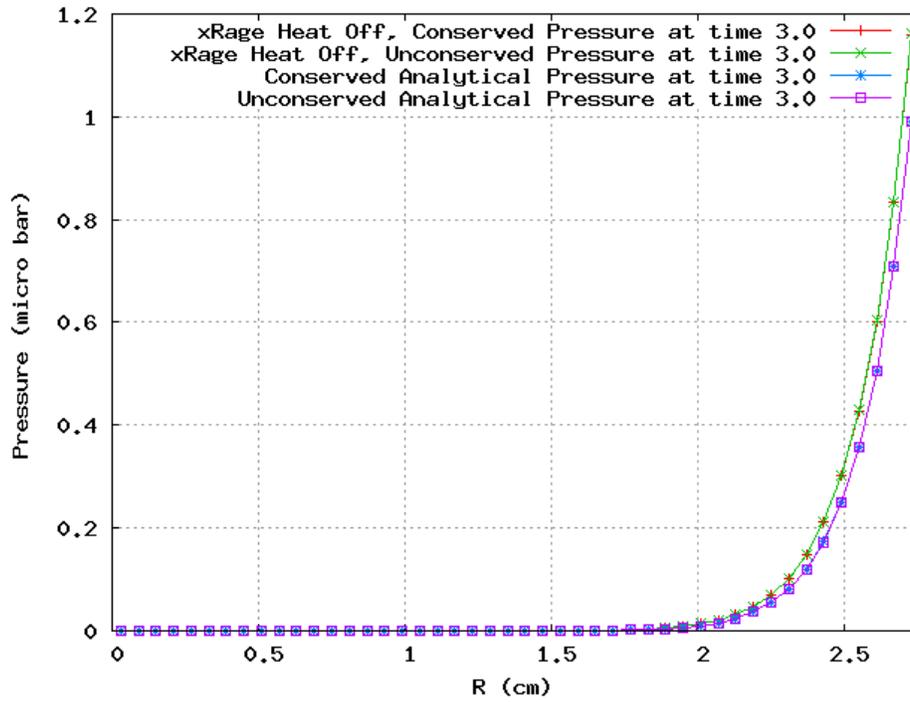


Figure 12: Viewgraph norm of the xRAGE pressure solution with and without conserved cell-averages with the heat conduction module not implemented at  $t=3.0$  s

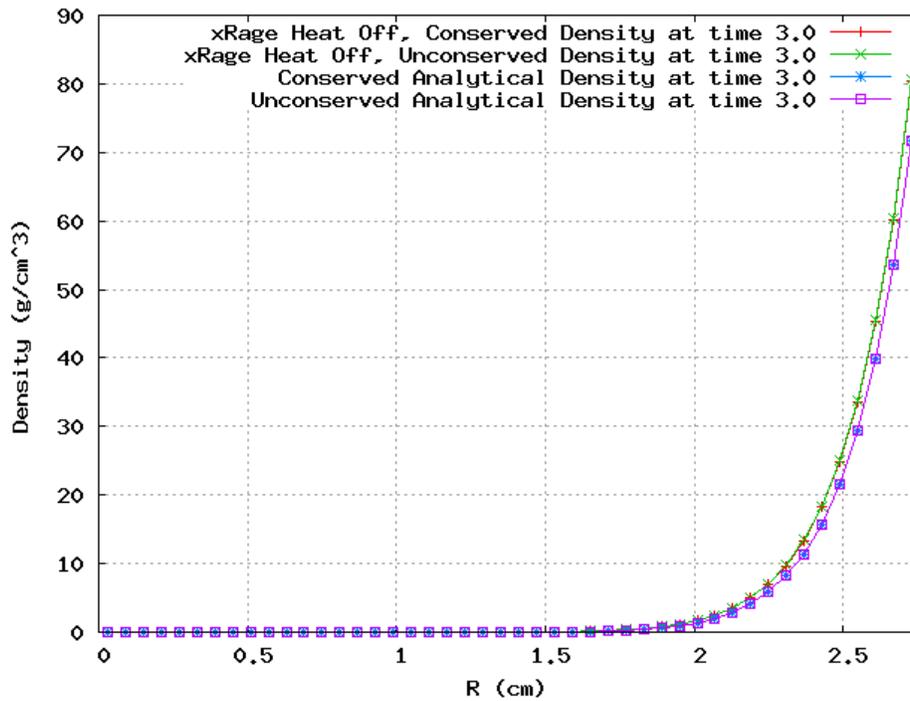


Figure 13: Viewgraph norm of the xRAGE density solution with and without conserved cell-averages with the heat conduction module not implemented at  $t=3.0$  s

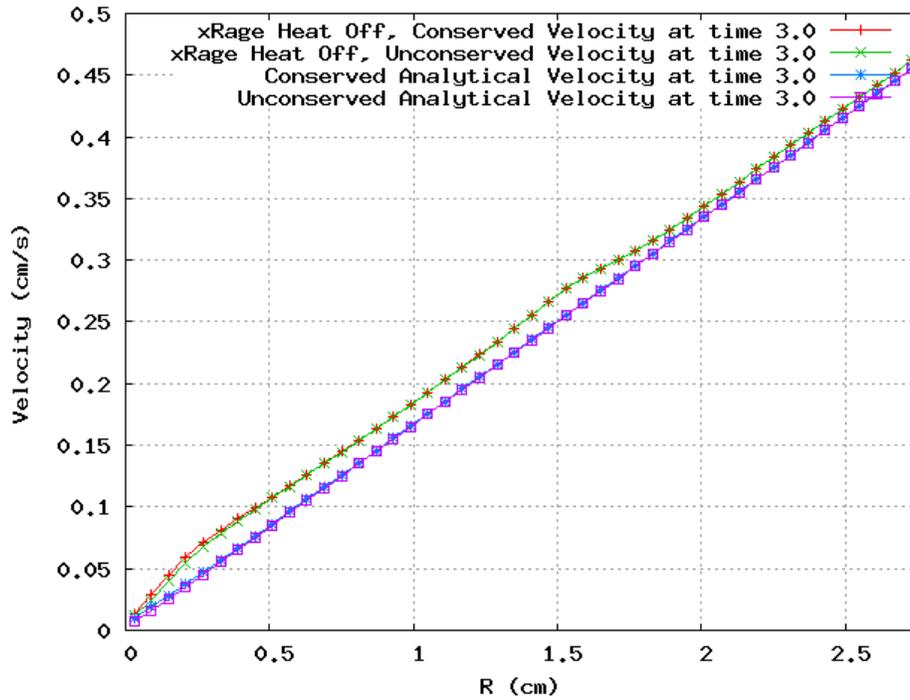


Figure 14: Viewgraph norm of the xRAGE velocity solution with and without conserved cell-averages with the heat conduction module not implemented at  $t=3.0$  s

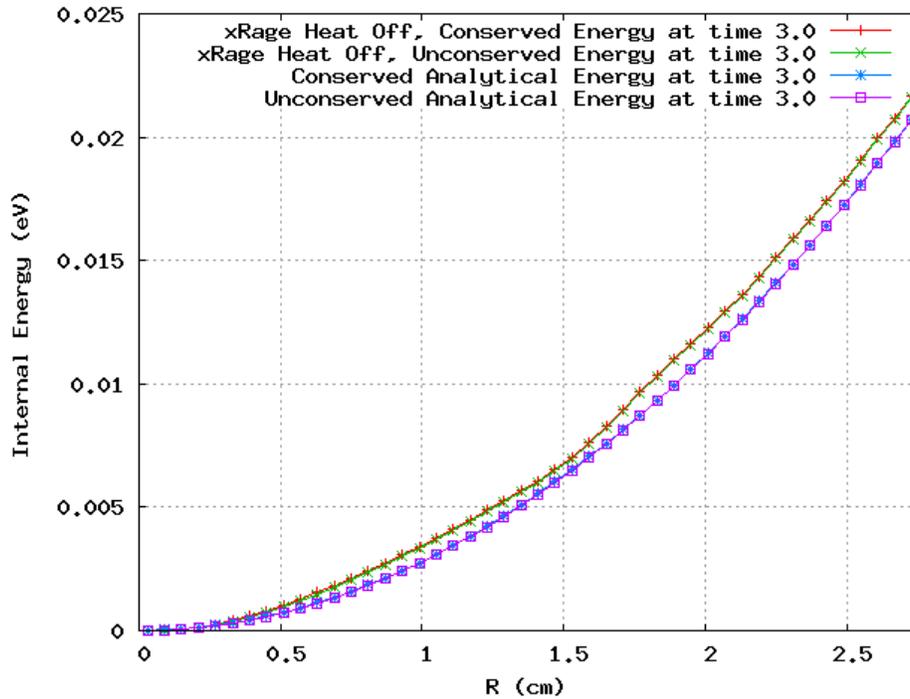


Figure 15: Viewgraph norm of the xRAGE specific internal energy solution with and without conserved cell-averages with the heat conduction module not implemented at  $t=3.0$  s

## Viewgraph Norms with the Heat Conduction Module Implemented with and without Conserved Cell-Averages

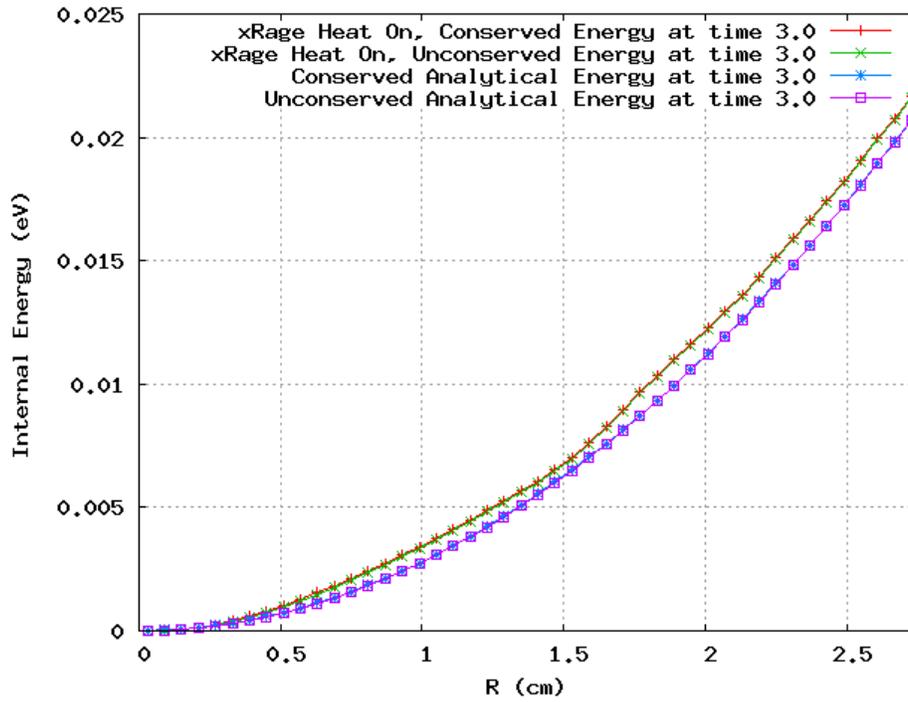


Figure 16: Viewgraph norm of the xRAGE specific internal energy solution with and without conserved cell-averages with the heat conduction module implemented at  $t=3.0$  s

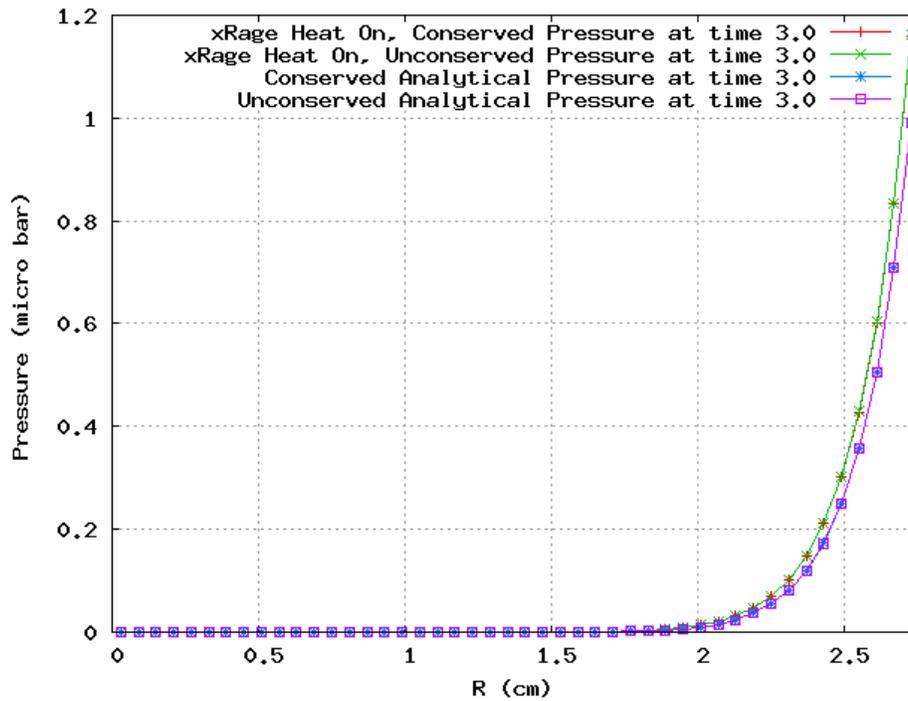


Figure 17: Viewgraph norm of the xRAGE pressure solution with and without conserved cell-averages with the heat conduction module implemented at  $t=3.0$  s

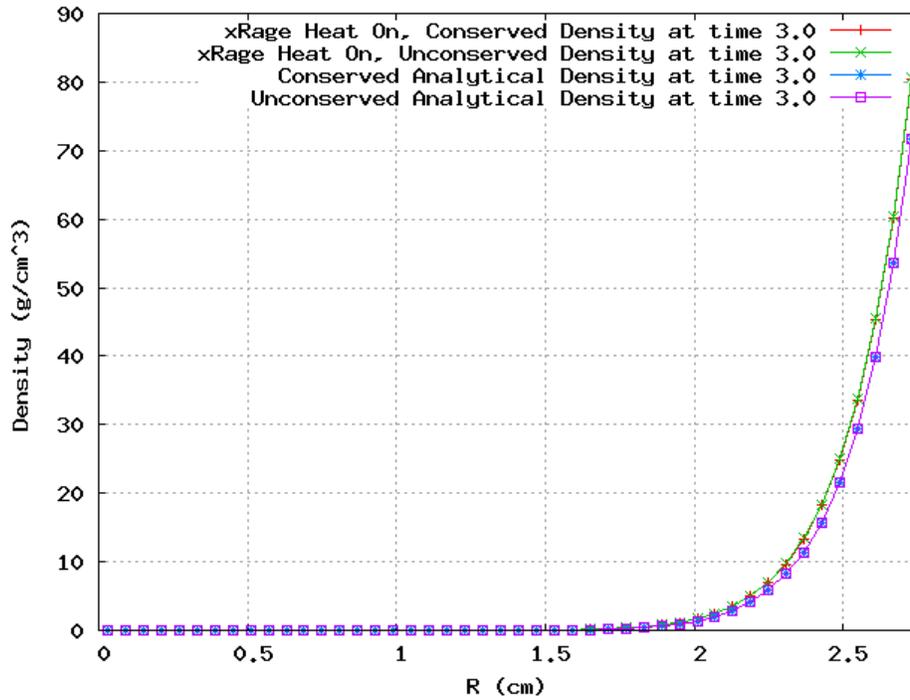


Figure 18: Viewgraph norm of the xRAGE density solution with and without conserved cell-averages with the heat conduction module implemented at  $t=3.0$  s

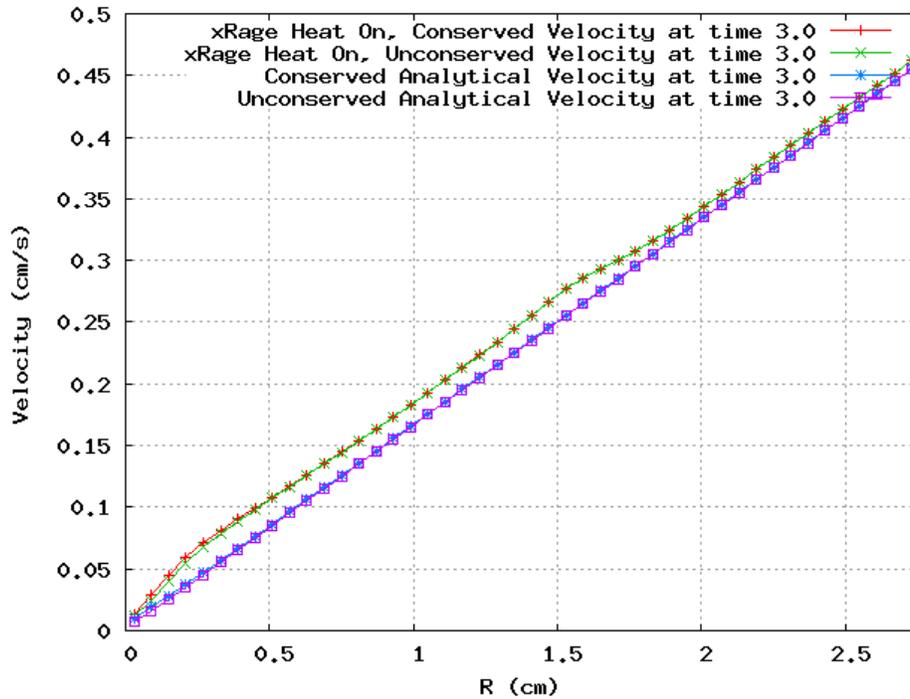


Figure 19: Viewgraph norm of the xRAGE velocity solution with and without conserved cell-averages with the heat conduction module implemented at  $t=3.0$  s

## Conserved, Heat Conduction On

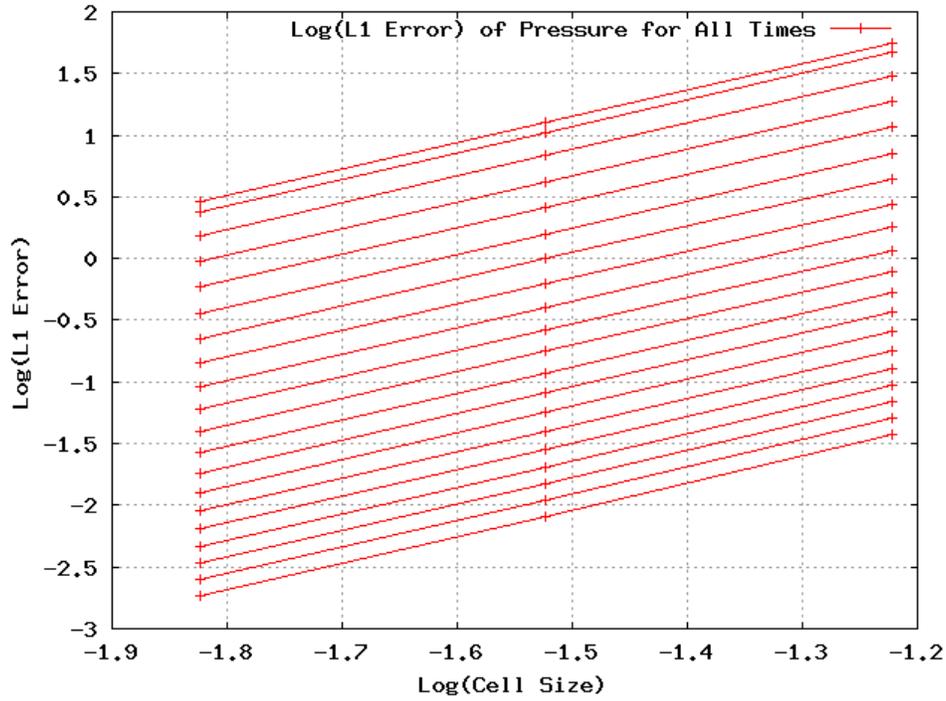


Figure 20: Log-log error plot for all times as a function of cell size for xRAGE pressure using the Coggeshall 9 solution with conserved values and the heat conduction module implemented

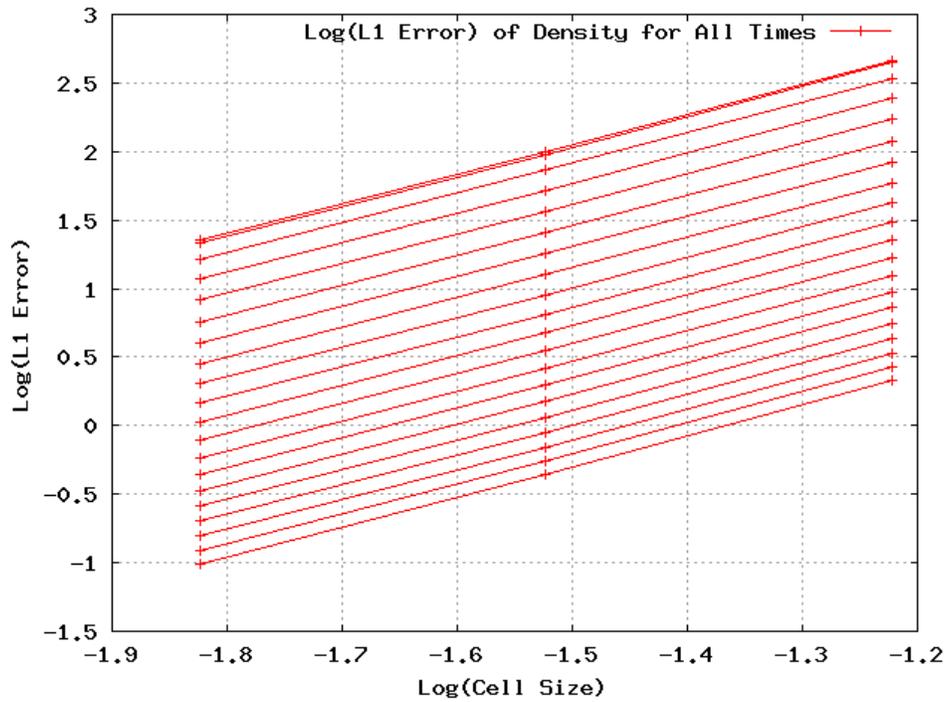


Figure 21: Log-log error plot for all times as a function of cell size for xRAGE density using the Coggeshall 9 solution with conserved values and the heat conduction module implemented

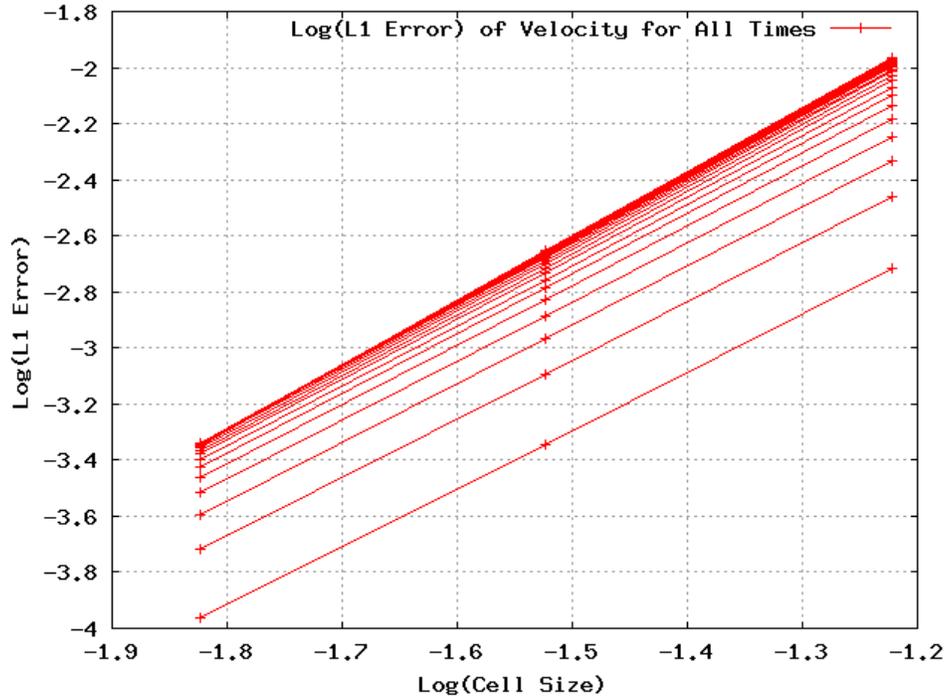


Figure 22: Log-log error plot for all times as a function of cell size for xRAGE velocity using the Coggeshall 9 solution with conserved values and the heat conduction module implemented

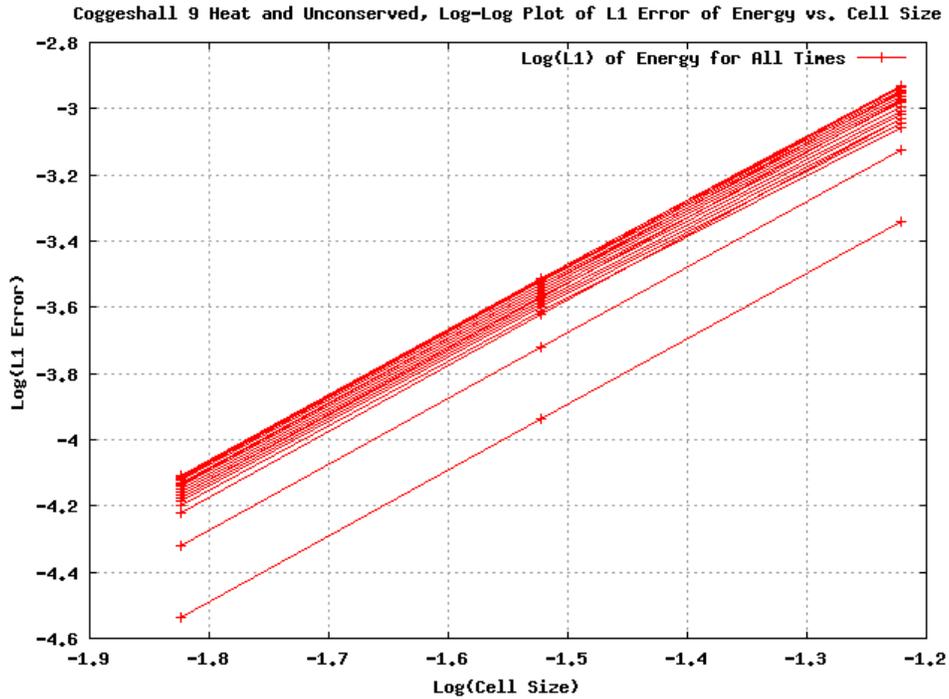


Figure 23: Log-log error plot for all times as a function of cell size for xRAGE specific internal energy using the Coggeshall 9 solution with conserved values and the heat conduction module implemented

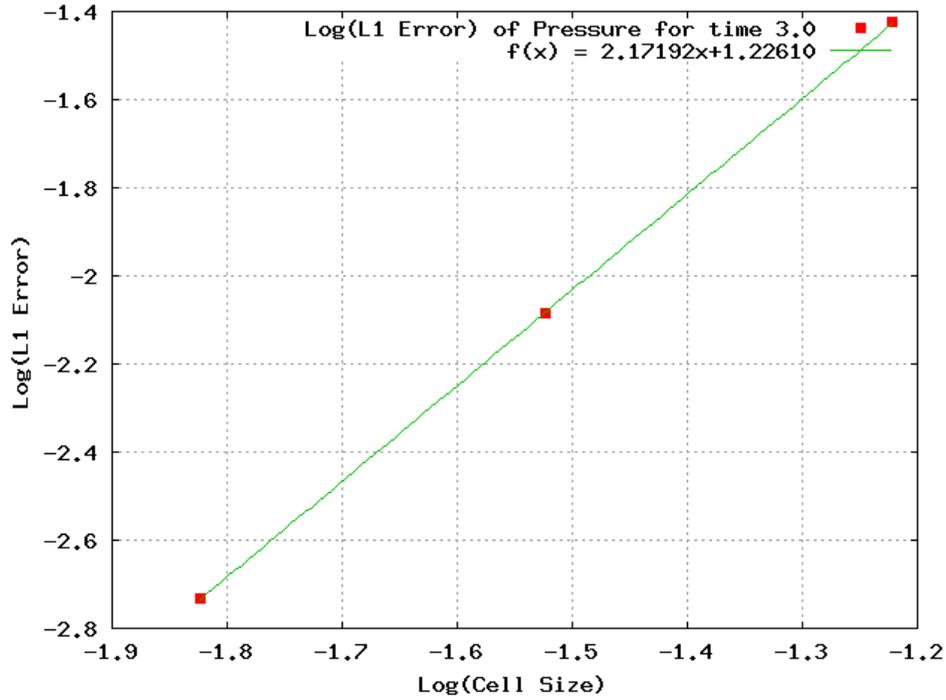


Figure 24: Log-log error plot with a fitted curve as a function of cell size for xRAGE pressure using the Coggeshall 9 solution with conserved values and the heat conduction module implemented

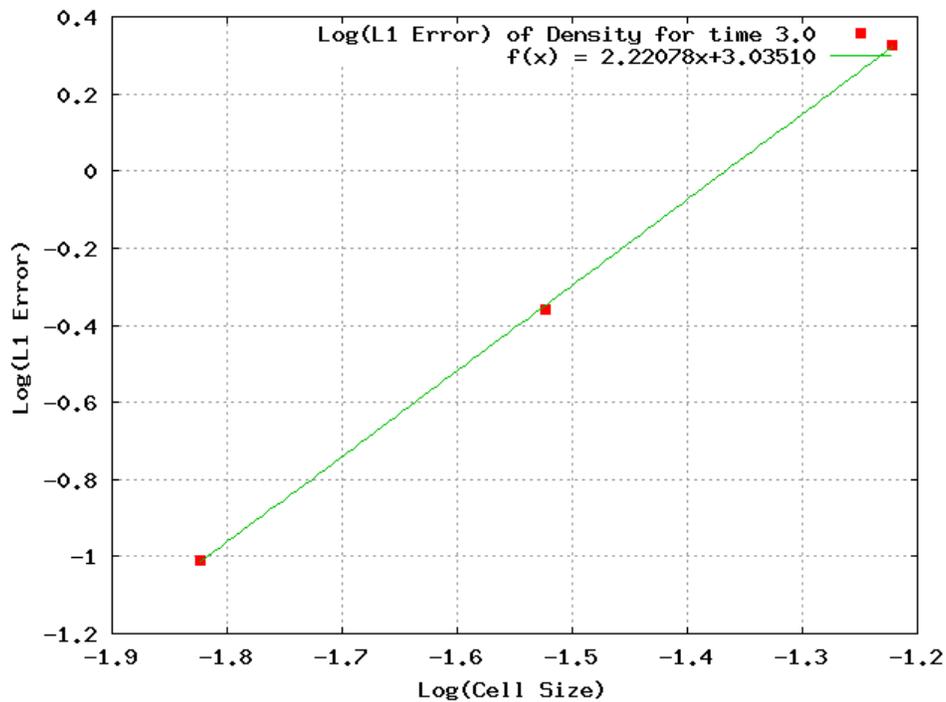


Figure 25: Log-log error plot with a fitted curve as a function of cell size for xRAGE density using the Coggeshall 9 solution with conserved values and the heat conduction module implemented

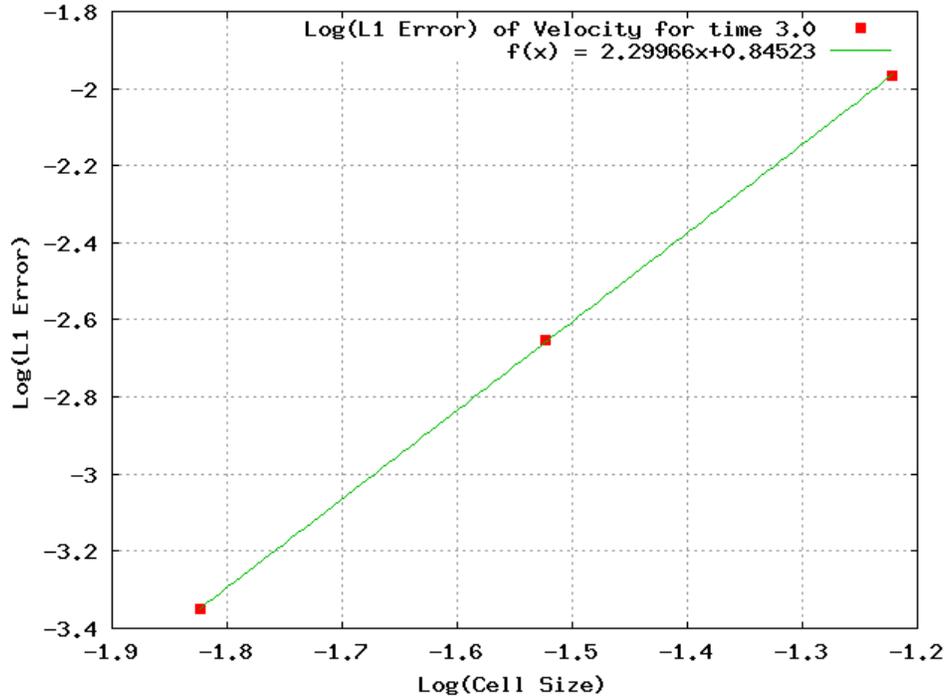


Figure 26: Log-log error plot with a fitted curve as a function of cell size for xRAGE velocity using the Coggeshall 9 solution with conserved values and the heat conduction module implemented

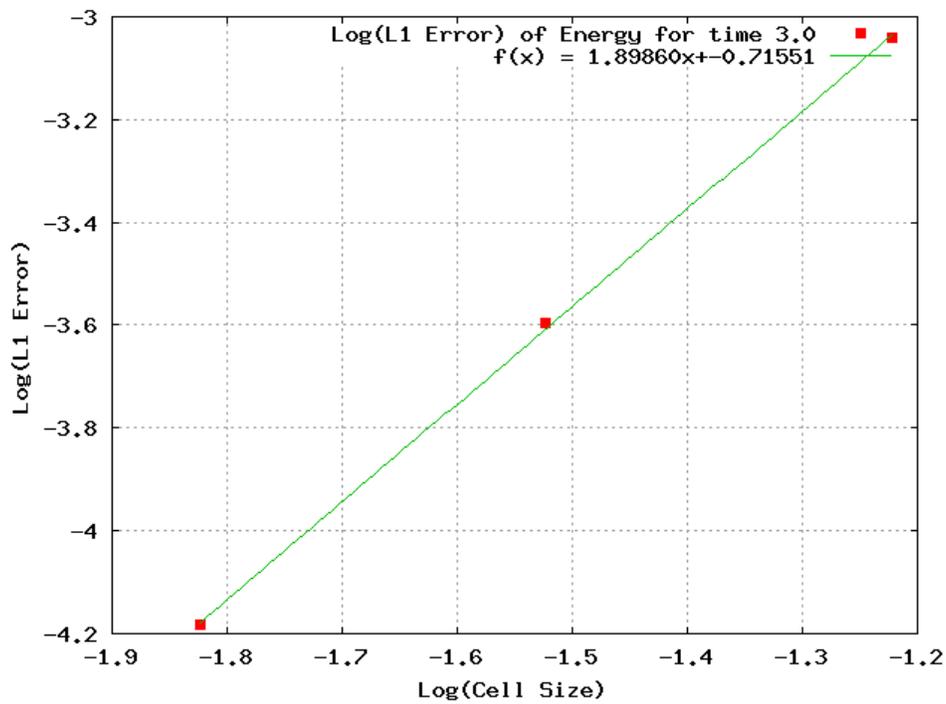


Figure 27: Log-log error plot with a fitted curve as a function of cell size for xRAGE specific internal energy using the Coggeshall 9 solution with conserved values and the heat conduction module implemented

## Non-Conserved, Heat Conduction On

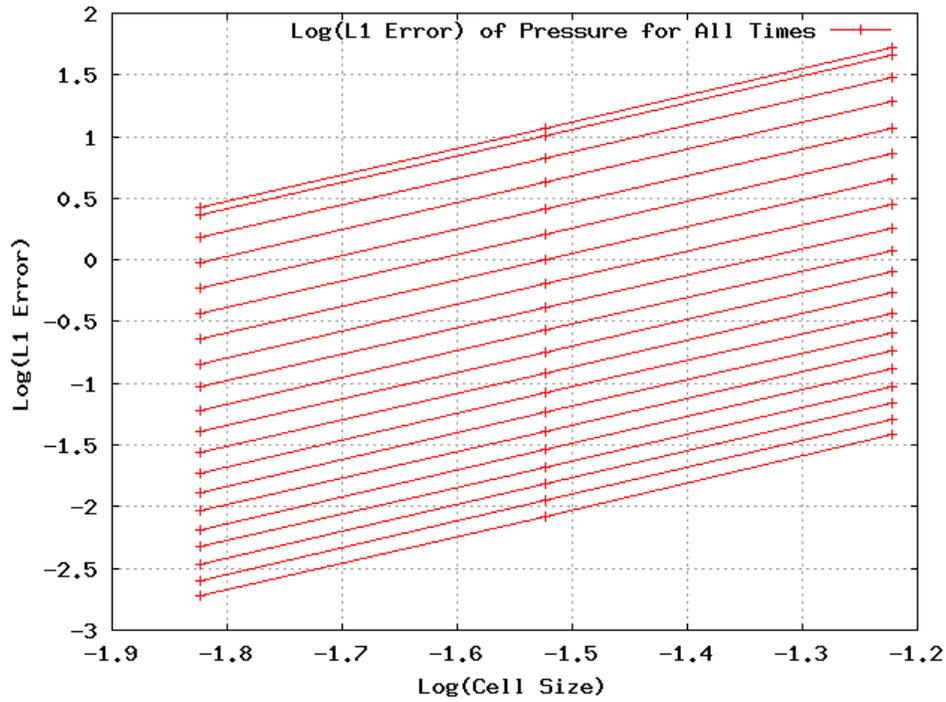


Figure 28: Log-log error plot for all times as a function of cell size for xRAGE pressure using the Coggeshall 9 solution with non-conserved values and the heat conduction module implemented

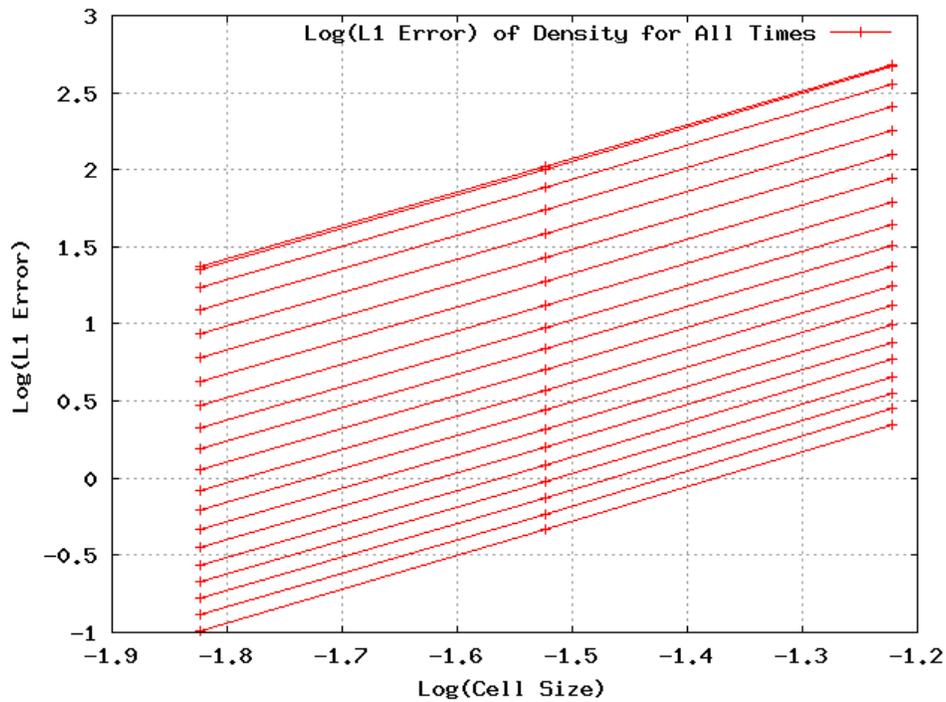


Figure 29: Log-log error plot for all times as a function of cell size for xRAGE density using the Coggeshall 9 solution with non-conserved values and the heat conduction module implemented

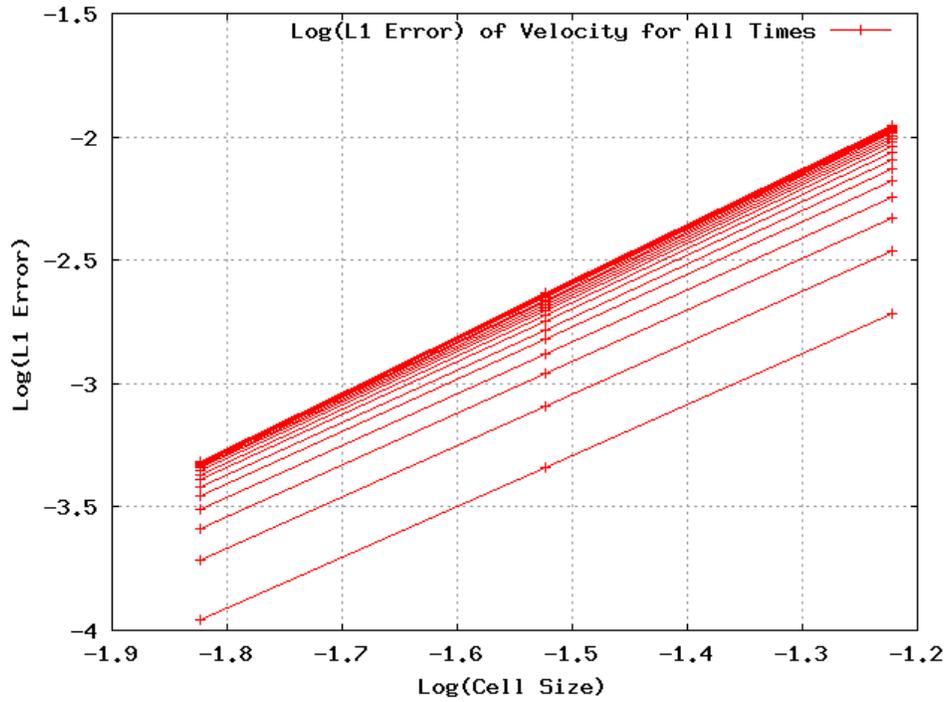


Figure 30: Log-log error plot for all times as a function of cell size for xRAGE velocity using the Coggeshall 9 solution with non-conserved values and the heat conduction module implemented

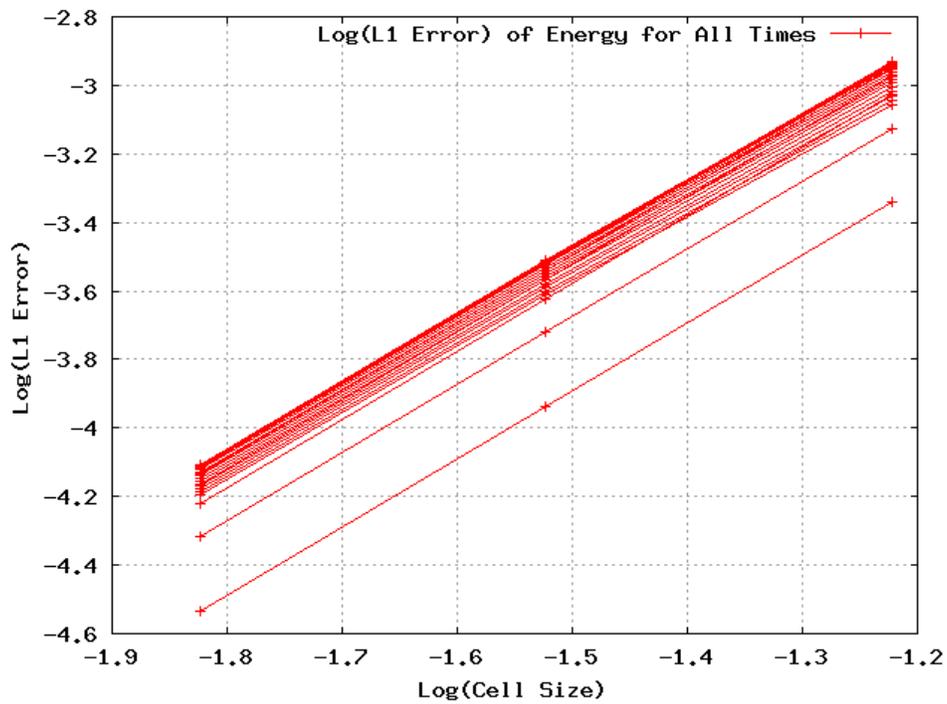


Figure 31: Log-log error plot for all times as a function of cell size for xRAGE specific internal energy using the Coggeshall 9 solution with non-conserved values and the heat conduction module implemented

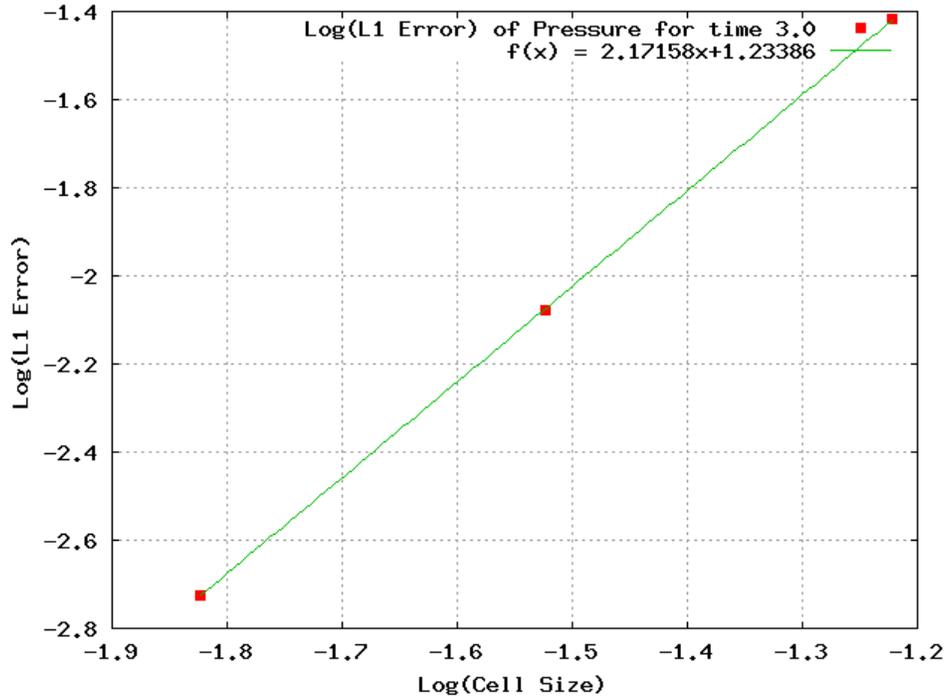


Figure 32: Log-log error plot with a fitted curve as a function of cell size for xRAGE pressure using the Coggeshall 9 solution with non-conserved values and the heat conduction module implemented

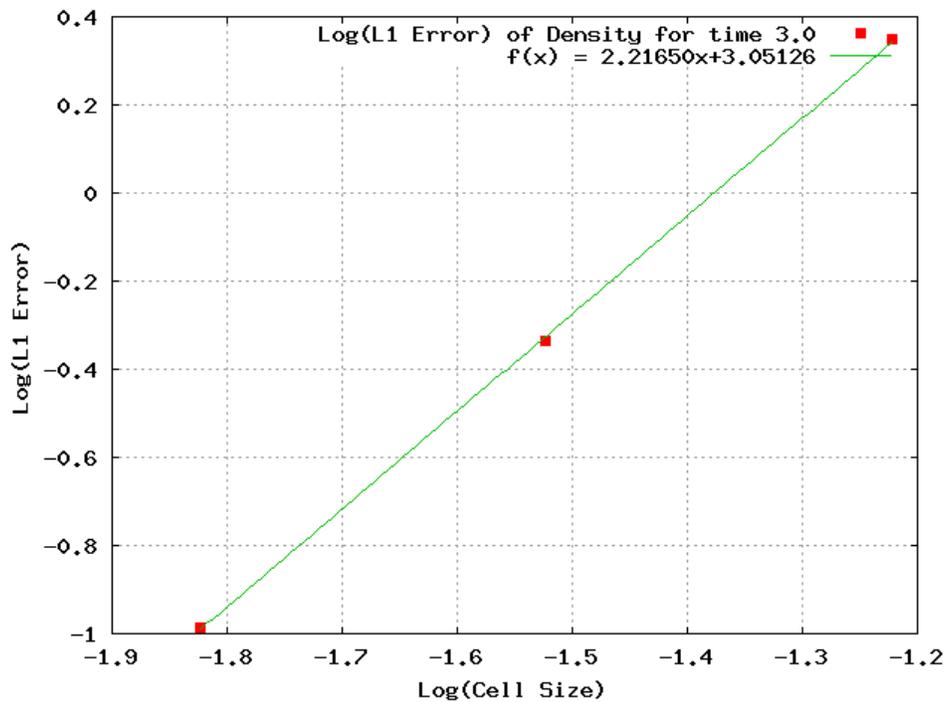


Figure 33: Log-log error plot with a fitted curve as a function of cell size for xRAGE density using the Coggeshall 9 solution with non-conserved values and the heat conduction module implemented

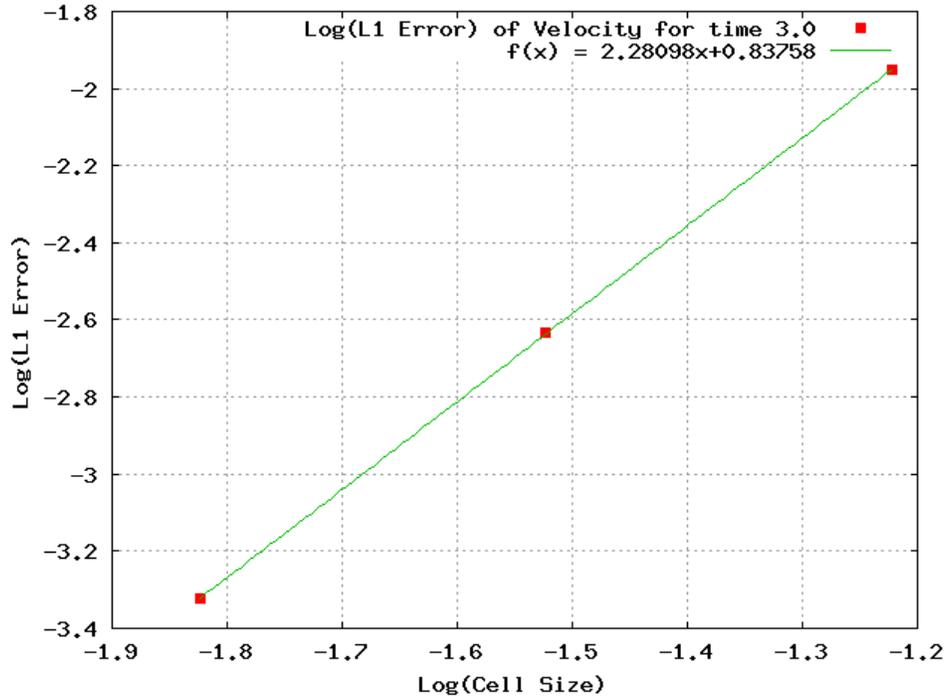


Figure 34: Log-log error plot with a fitted curve as a function of cell size for xRAGE velocity using the Coggeshall 9 solution with non-conserved values and the heat conduction module implemented

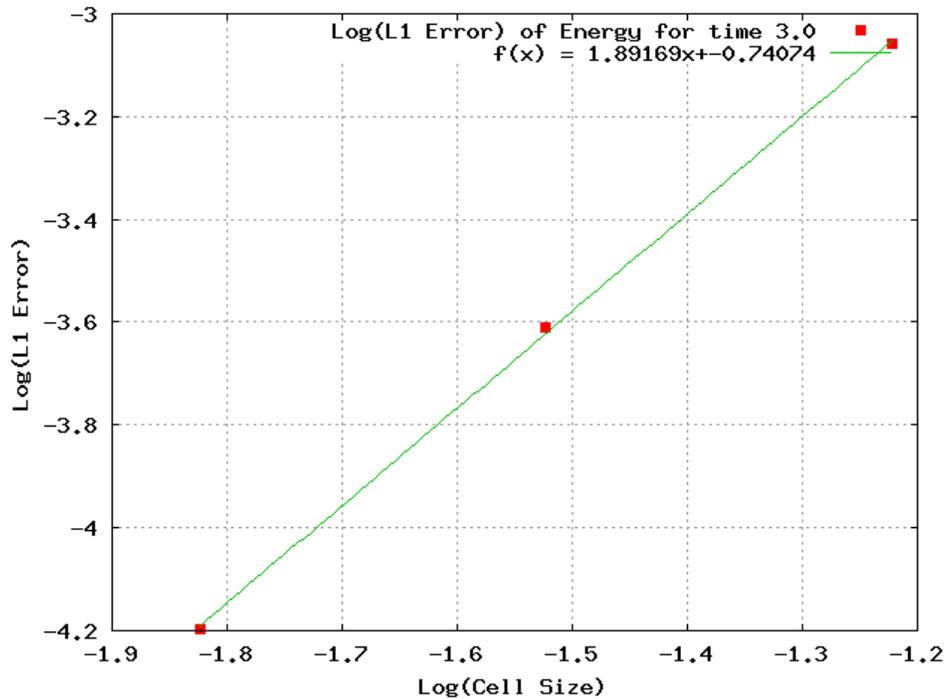


Figure 35: Log-log error plot with a fitted curve as a function of cell size for xRAGE specific internal energy using the Coggeshall 9 solution with non-conserved values and the heat conduction module implemented

### Conserved, Heat Conduction Off

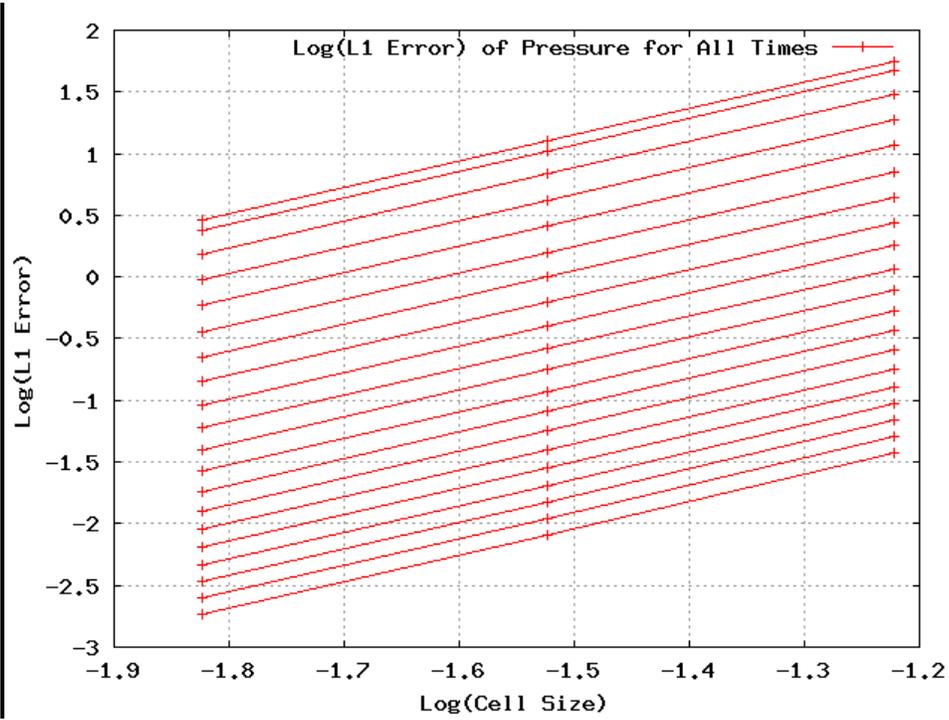


Figure 36: Log-log error plot for all times as a function of cell size for xRAGE pressure using the Coggeshall 9 solution with conserved values and the heat conduction module not implemented

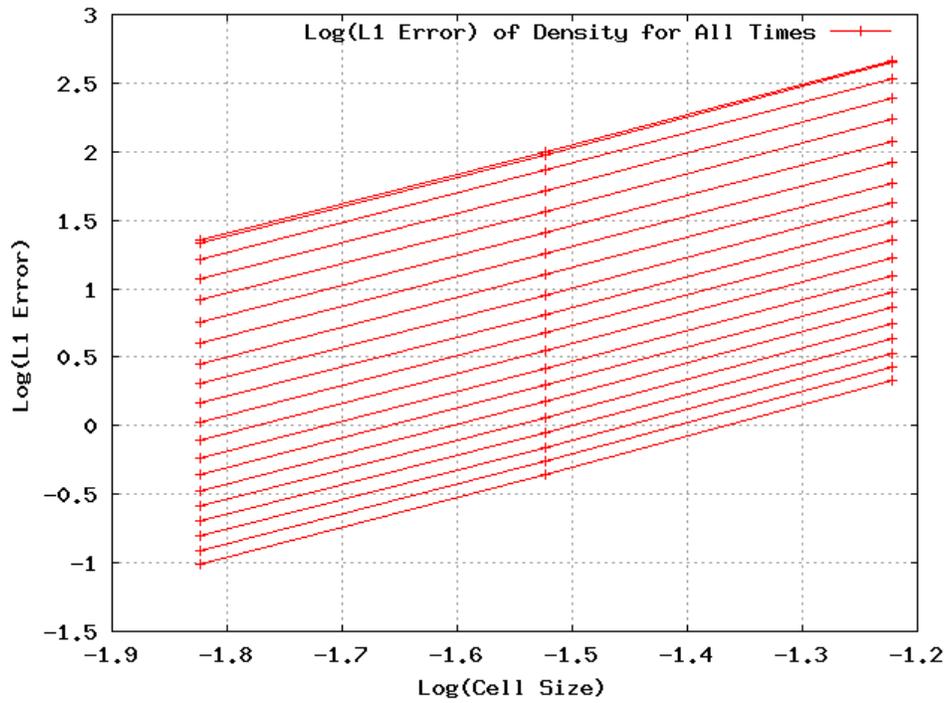


Figure 37: Log-log error plot for all times as a function of cell size for xRAGE density using the Coggeshall 9 solution with conserved values and the heat conduction module not implemented

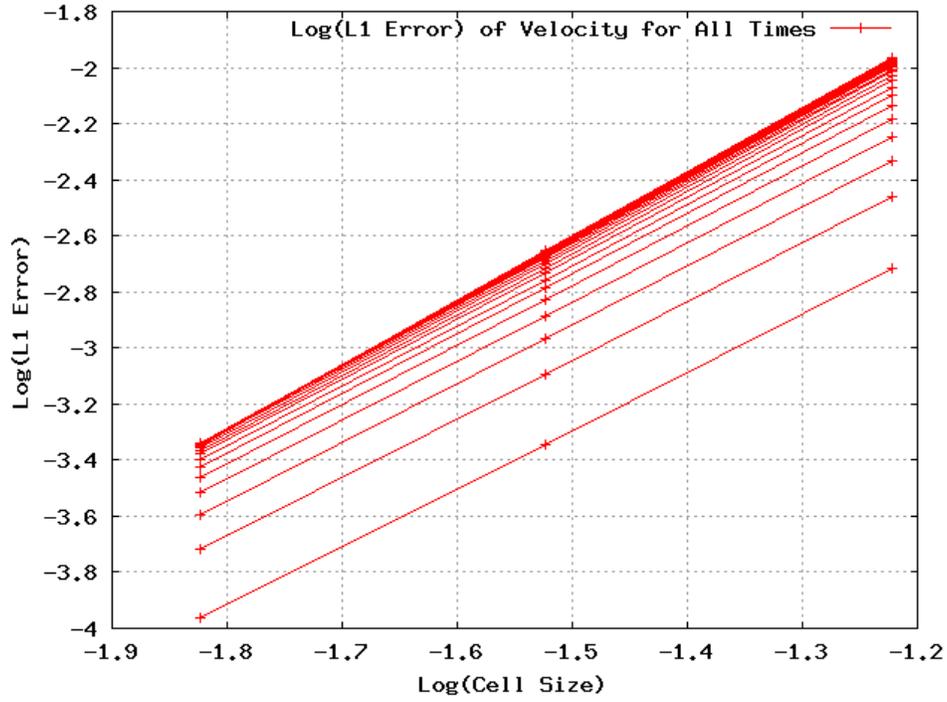


Figure 38: Log-log error plot for all times as a function of cell size for xRAGE velocity using the Coggeshall 9 solution with conserved values and the heat conduction module not implemented

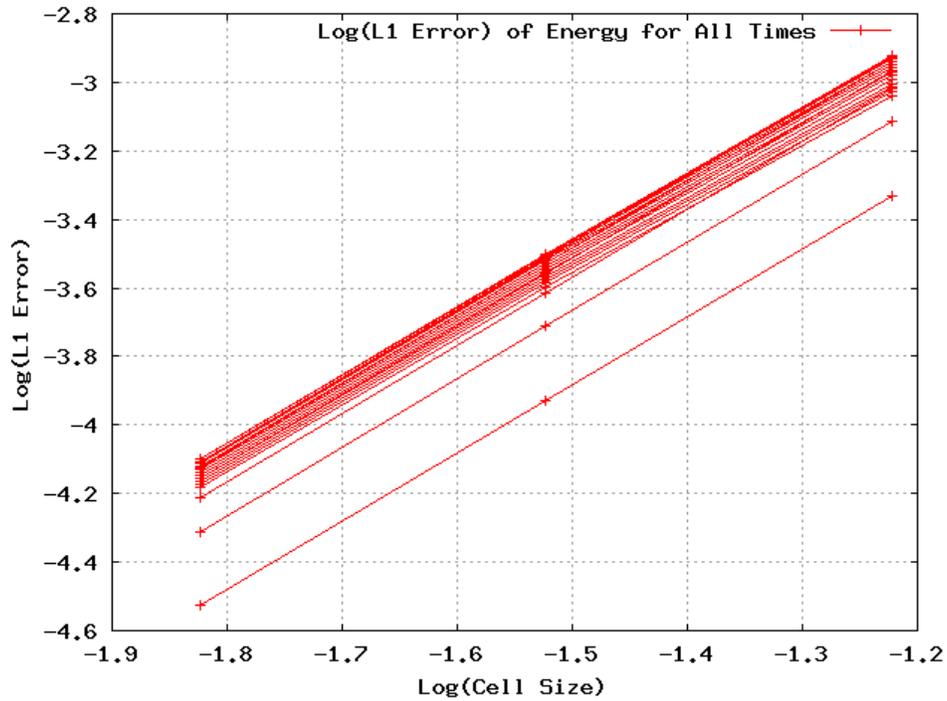


Figure 39: Log-log error plot for all times as a function of cell size for xRAGE specific internal energy using the Coggeshall 9 solution with conserved values and the heat conduction module not implemented

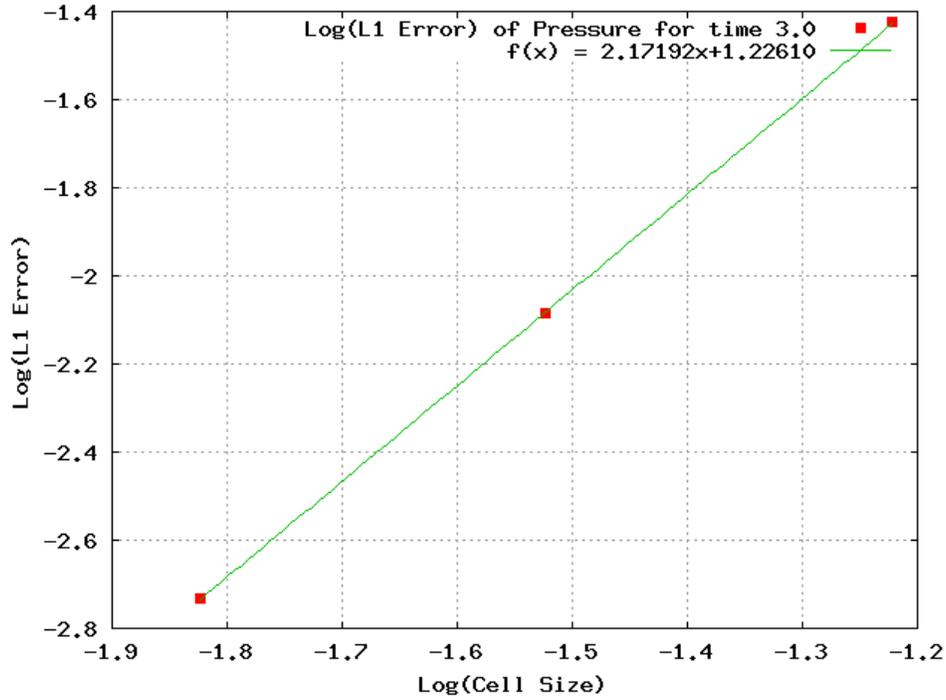


Figure 40: Log-log error plot with a fitted curve as a function of cell size for xRAGE pressure using the Coggeshall 9 solution with conserved values and the heat conduction module not implemented

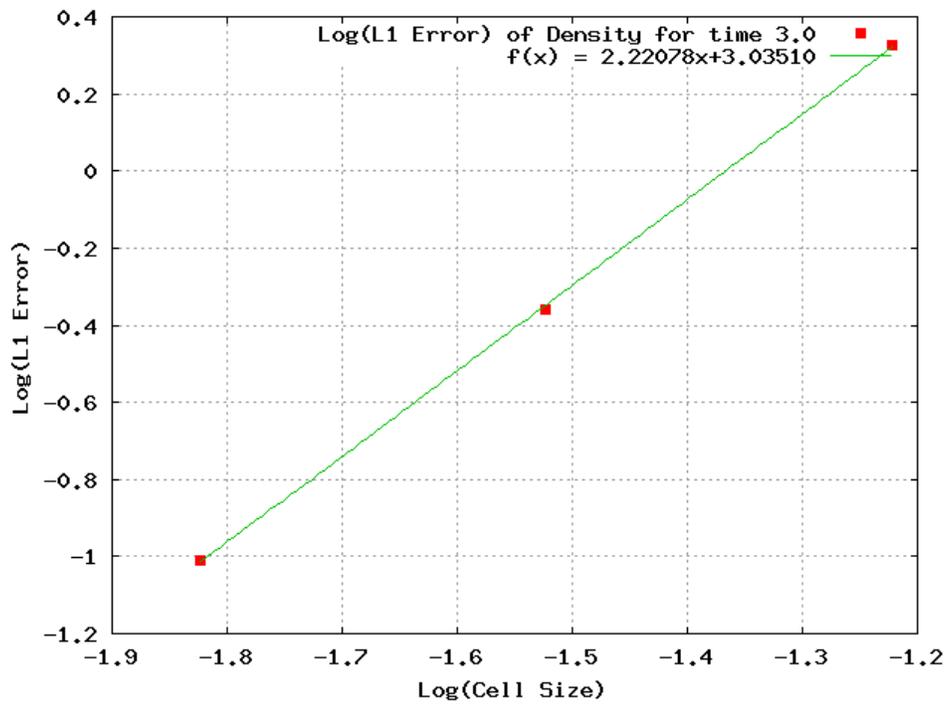


Figure 41: Log-log error plot with a fitted curve as a function of cell size for xRAGE density using the Coggeshall 9 solution with conserved values and the heat conduction module not implemented

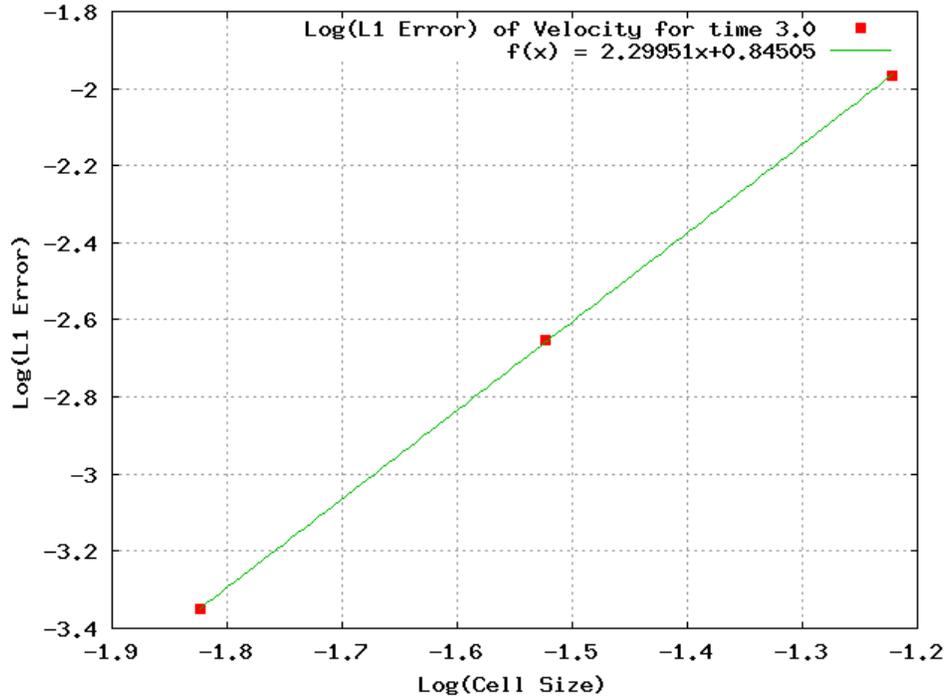


Figure 42: Log-log error plot with a fitted curve as a function of cell size for xRAGE velocity using the Coggeshall 9 solution with conserved values and the heat conduction module not implemented

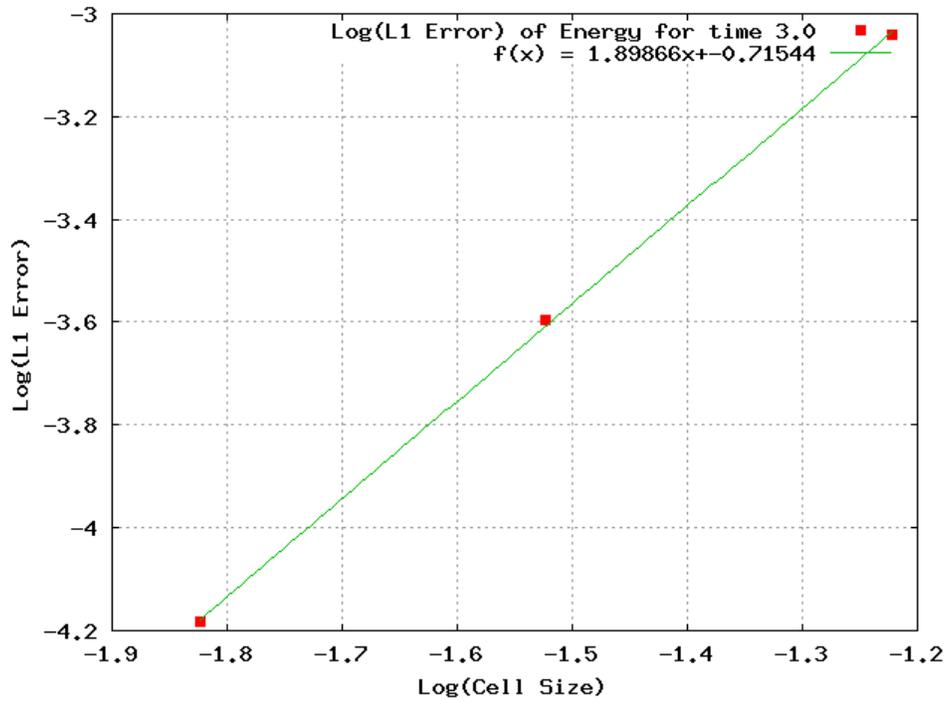


Figure 43: Log-log error plot with a fitted curve as a function of cell size for xRAGE energy using the Coggeshall 9 solution with conserved values and the heat conduction module not implemented

## Non-Conserved, Heat Conduction Off

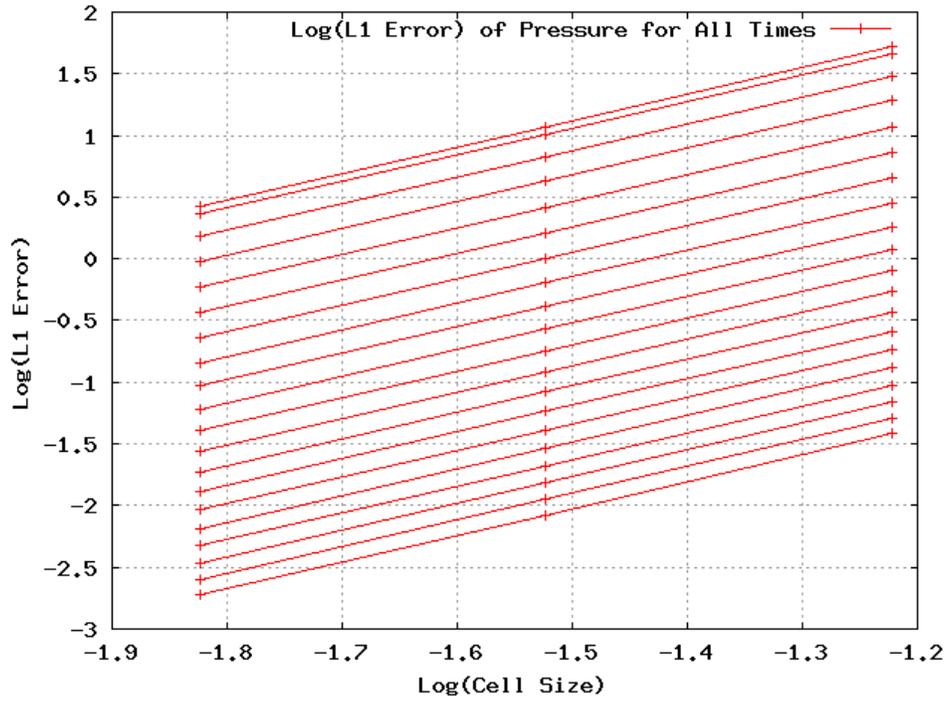


Figure 44: Log-log error plot for all times as a function of cell size for xRAGE pressure using the Coggeshall 9 solution with non-conserved values and the heat conduction module not implemented

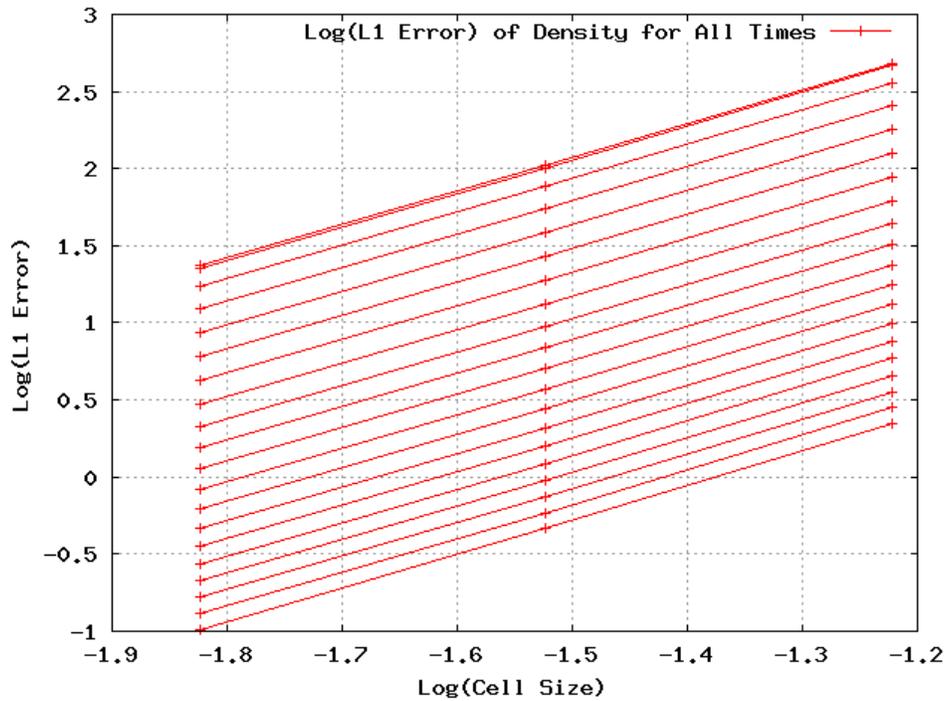


Figure 45: Log-log error plot for all times as a function of cell size for xRAGE density using the Coggeshall 9 solution with non-conserved values and the heat conduction module not implemented

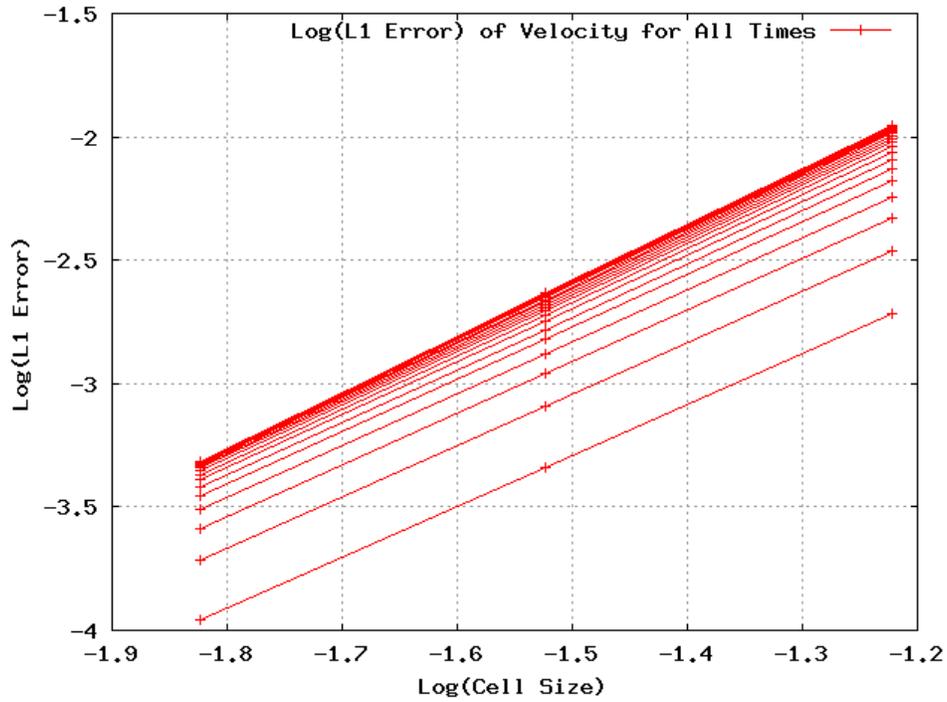


Figure 46: Log-log error plot for all times as a function of cell size for xRAGE velocity using the Coggeshall 9 solution with non-conserved values and the heat conduction module not implemented

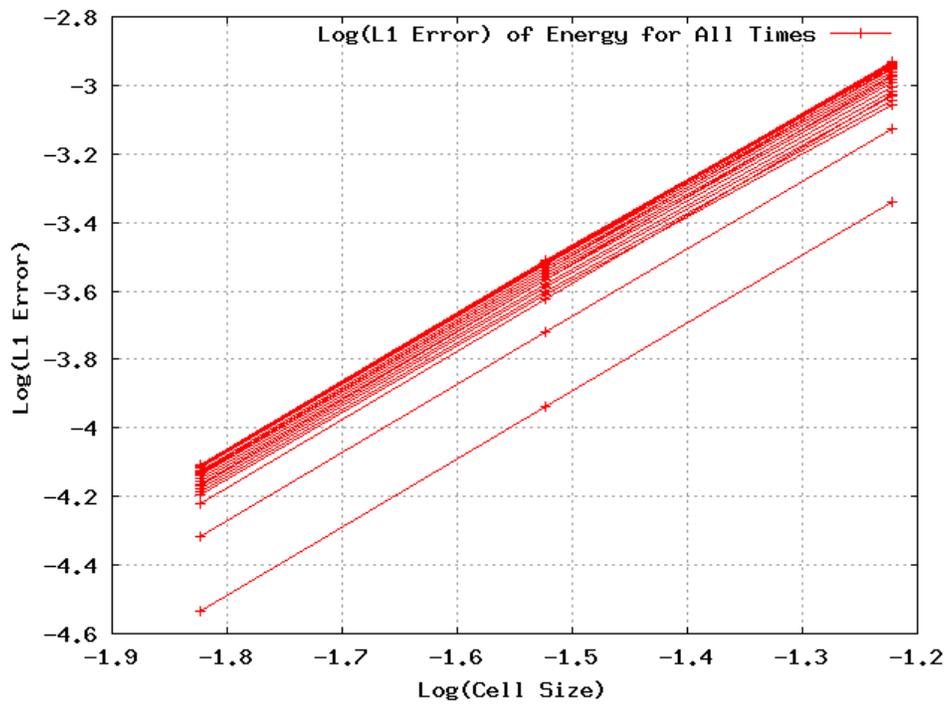


Figure 47: Log-log error plot for all times as a function of cell size for xRAGE specific internal energy using the Coggeshall 9 solution with non-conserved values and the heat conduction module not implemented

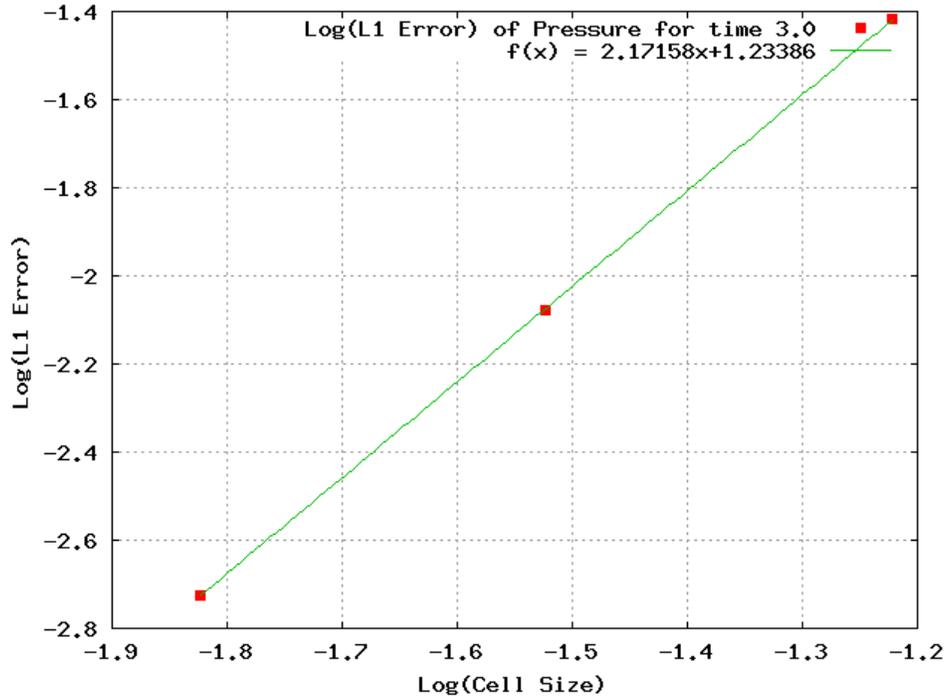


Figure 48: Log-log error plot with a fitted curve as a function of cell size for xRAGE pressure using the Coggeshall 9 solution with non-conserved values and the heat conduction module not implemented

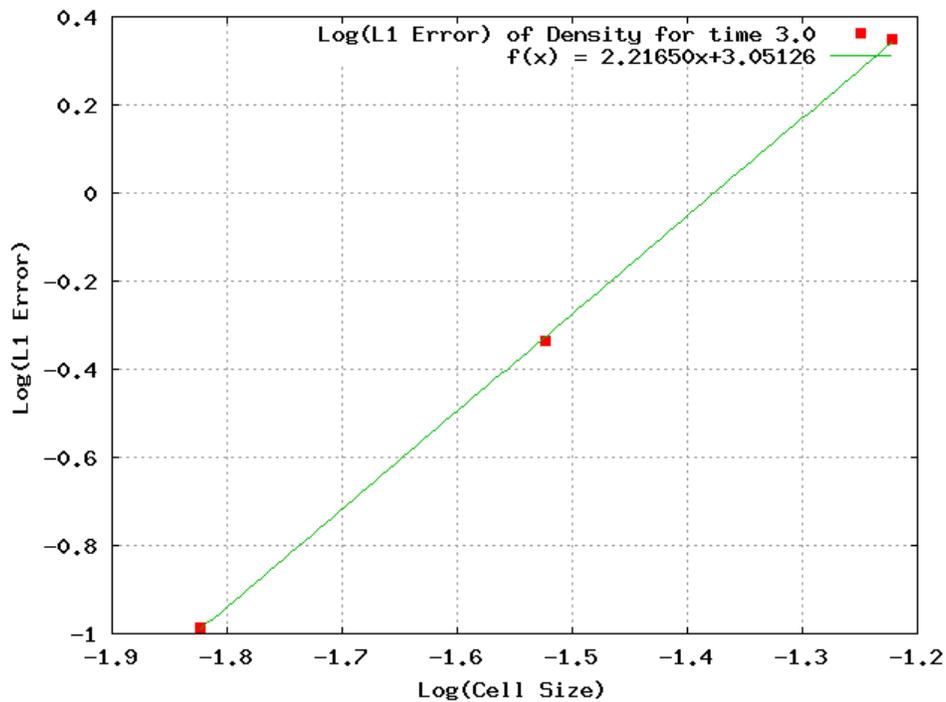


Figure 49: Log-log error plot with a fitted curve as a function of cell size for xRAGE density using the Coggeshall 9 solution with non-conserved values and the heat conduction module not implemented

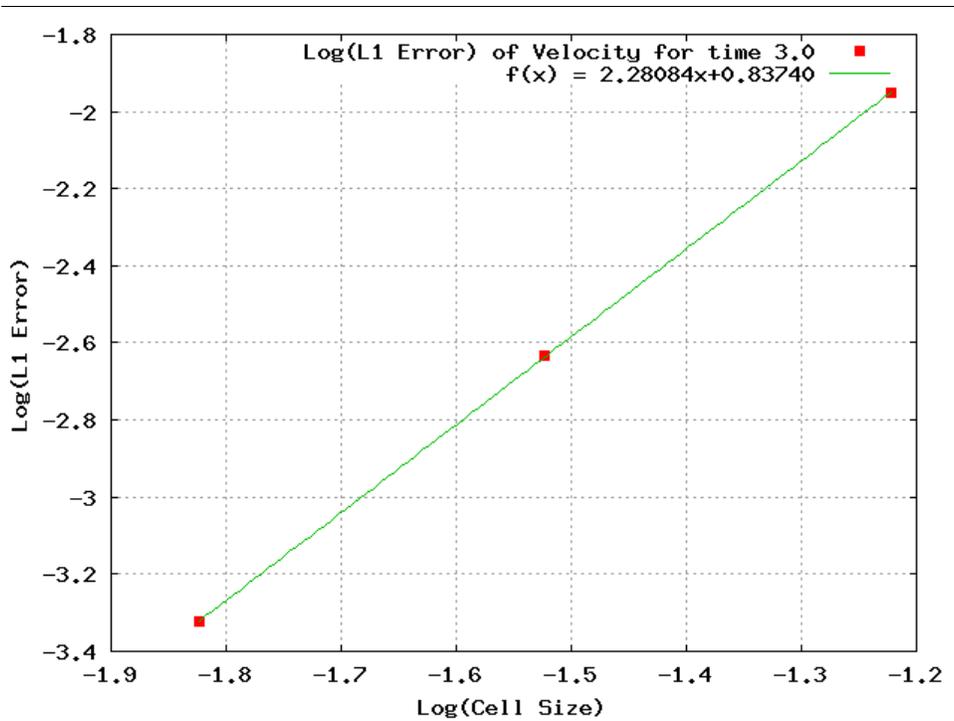


Figure 50: Log-log error plot with a fitted curve as a function of cell size for xRAGE velocity using the Coggeshall 9 solution with non-conserved values and the heat conduction module not implemented

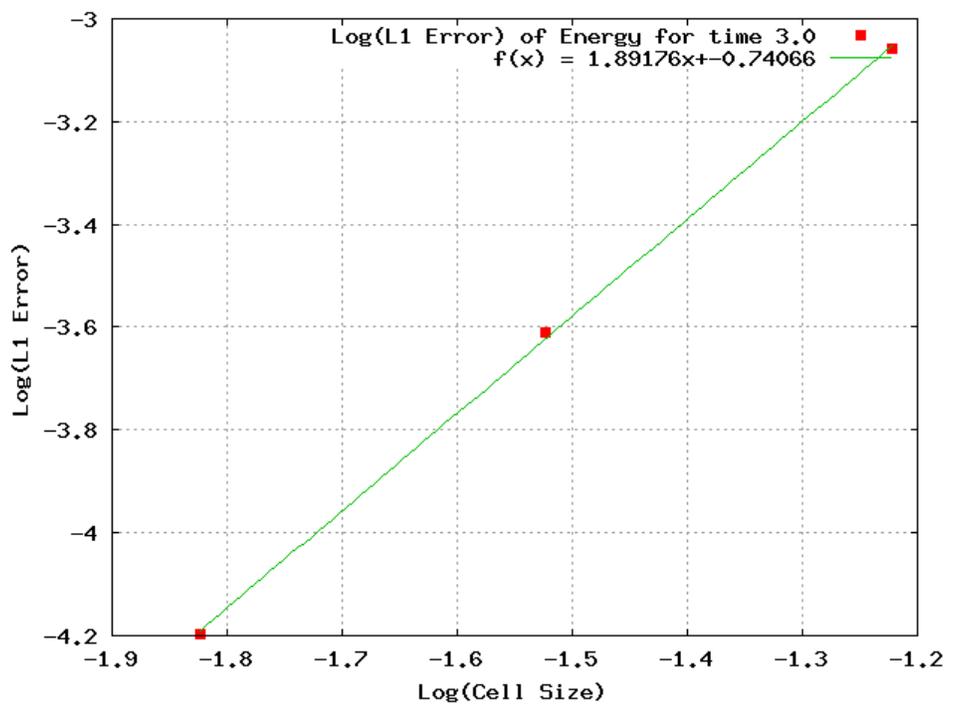


Figure 51: Log-log error plot with a fitted curve as a function of cell size for xRAGE specific internal energy using the Coggeshall 9 solution with non-conserved values and the heat conduction module not implemented

## E: Coggeshall Solution 11

### Viewgraph Norms for Conserved Cell-Averages with and without the Heat Conduction Module Implemented

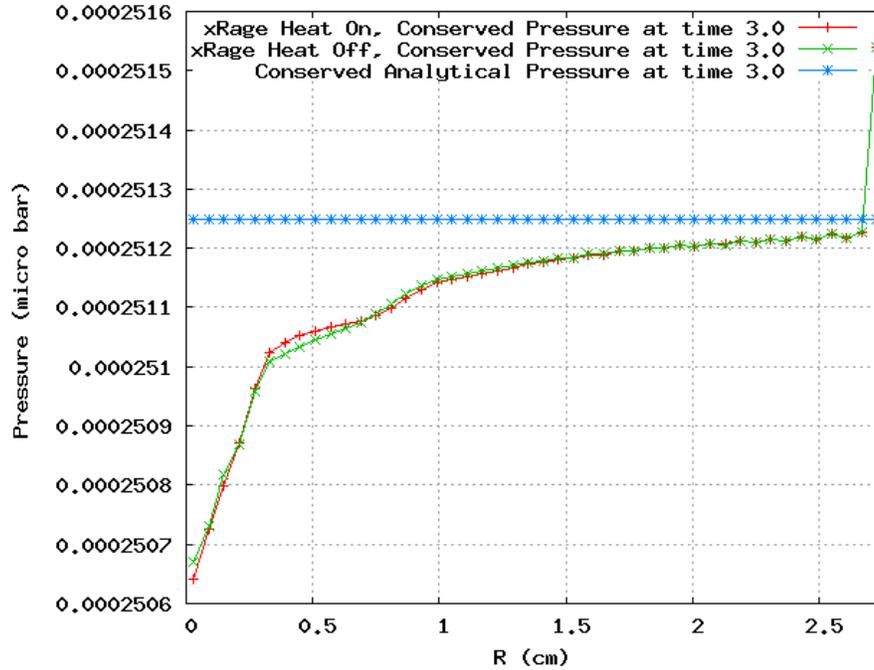


Figure 52: Viewgraph norm of the xRAGE pressure solution with and without the heat conduction module implemented for conserved cell-average values at  $t=3.0$  s

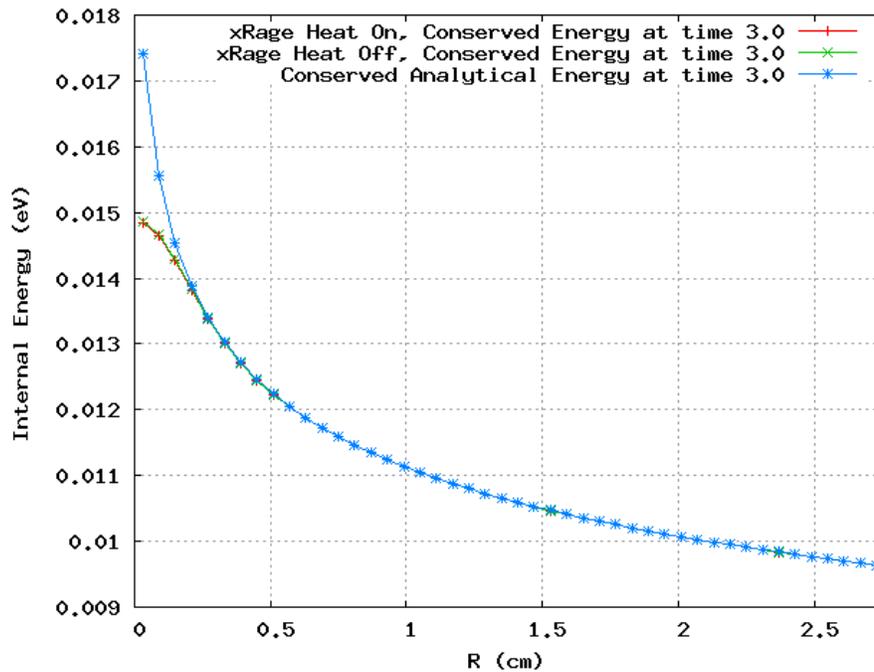


Figure 53: Viewgraph norm of the xRAGE specific internal energy solution with and without the heat conduction module implemented for conserved cell-average values at  $t=3.0$  s

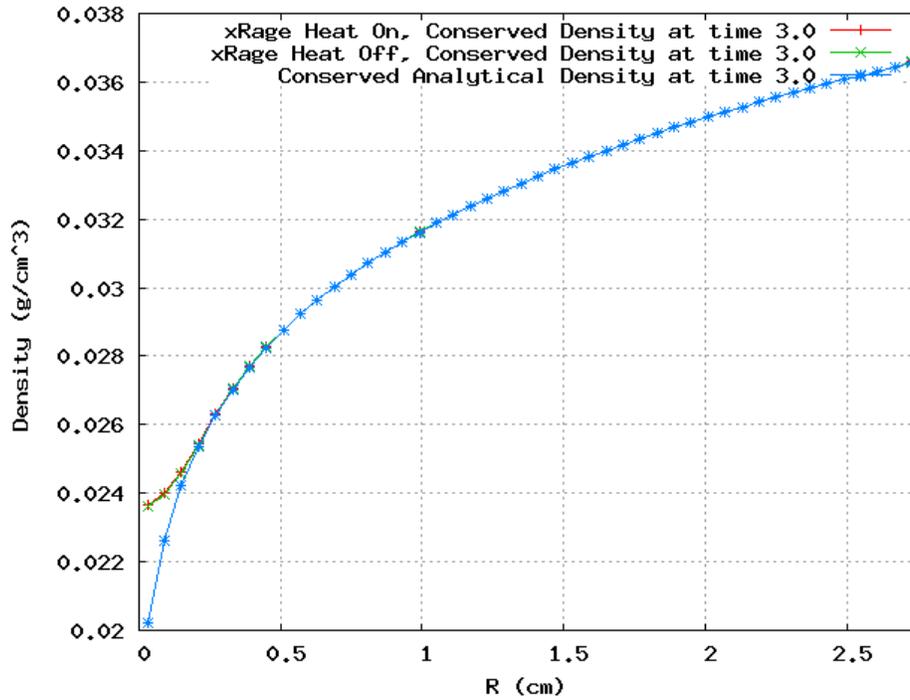


Figure 54: Viewgraph norm of the xRAGE density solution with and without the heat conduction module implemented for conserved cell-average values at  $t=3.0$  s

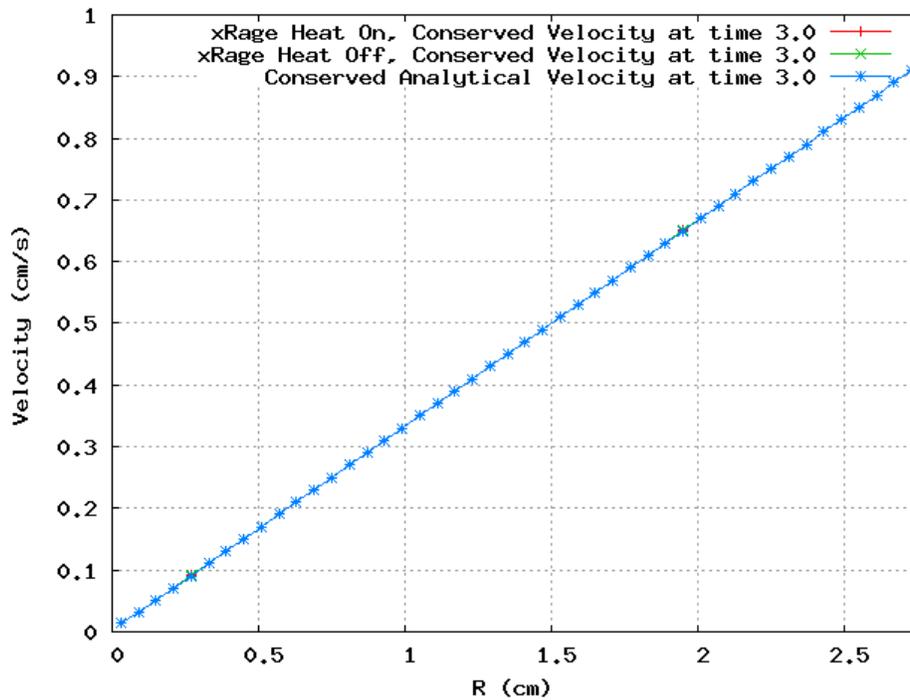


Figure 55: Viewgraph norm of the xRAGE velocity solution with and without the heat conduction module implemented for conserved cell-average values at  $t=3.0$  s

## Viewgraph Norms for Unconserved Cell-Averages with and without the Heat Conduction Module Implemented

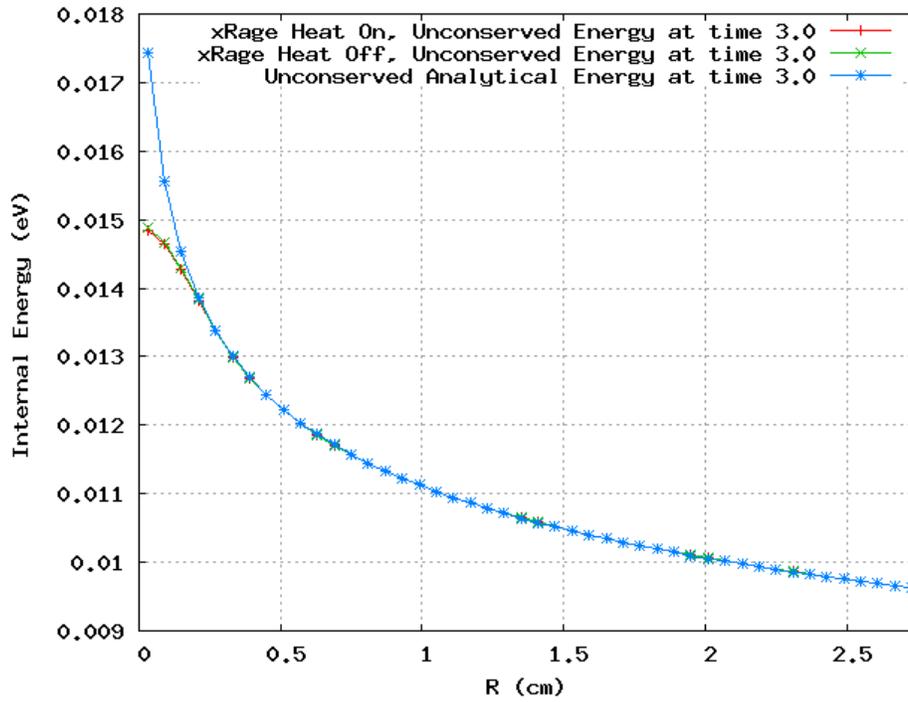


Figure 56: Viewgraph norm of the xRAGE velocity solution with and without the heat conduction module implemented for unconserved cell-average values at  $t=3.0$  s

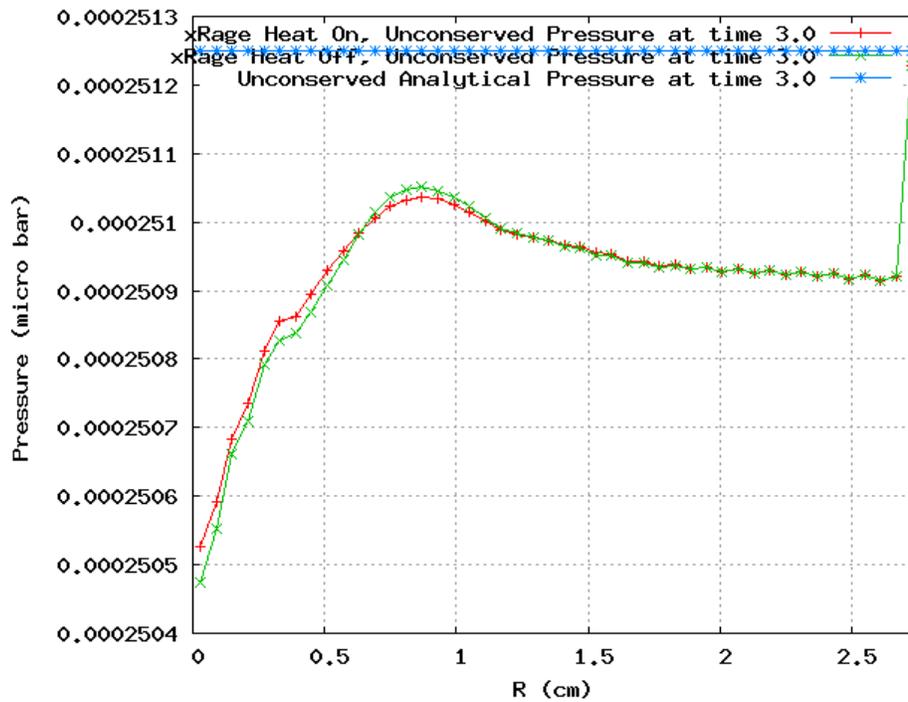


Figure 57: Viewgraph norm of the xRAGE pressure solution with and without the heat conduction module implemented for unconserved cell-average values at  $t=3.0$  s

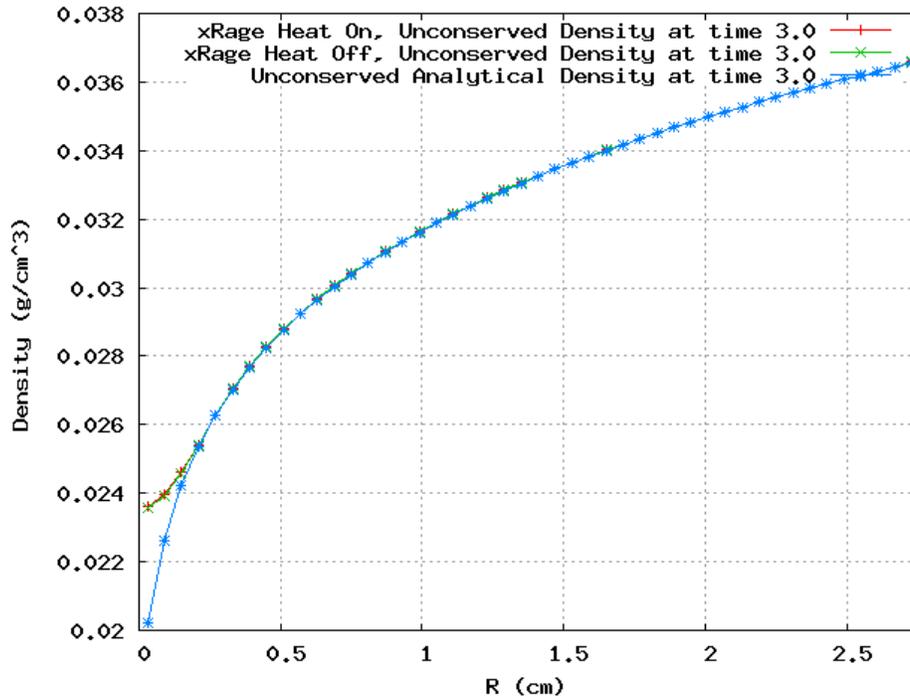


Figure 58: Viewgraph norm of the xRAGE density solution with and without the heat conduction module implemented for unconserved cell-average values at  $t=3.0$  s

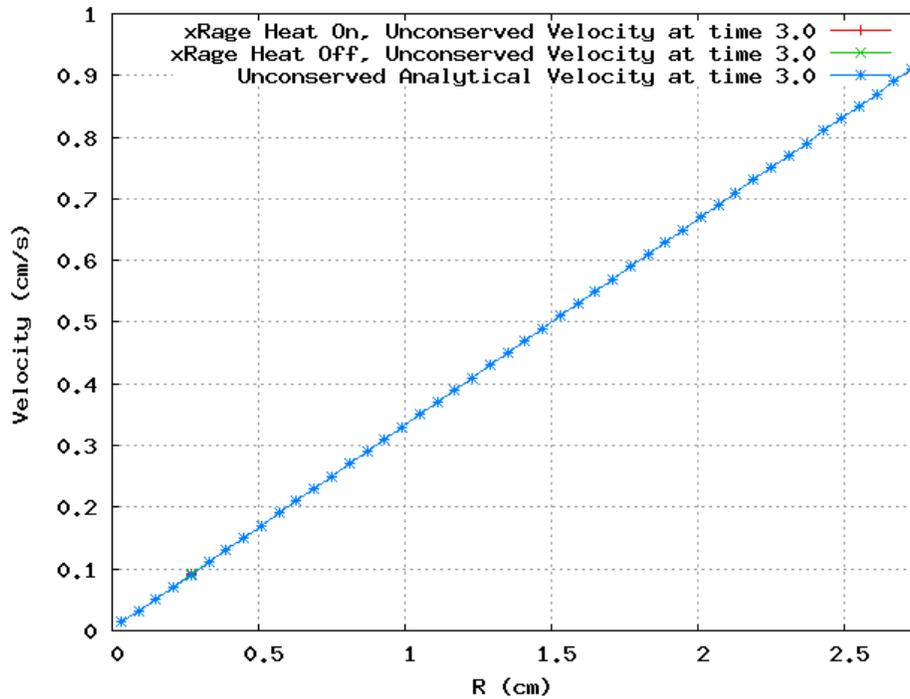


Figure 59: Viewgraph norm of the xRAGE velocity solution with and without the heat conduction module implemented for unconserved cell-average values at  $t=3.0$  s

## Viewgraph Norms with the Heat Conduction Module Not Implemented with and without Conserved Cell-Averages

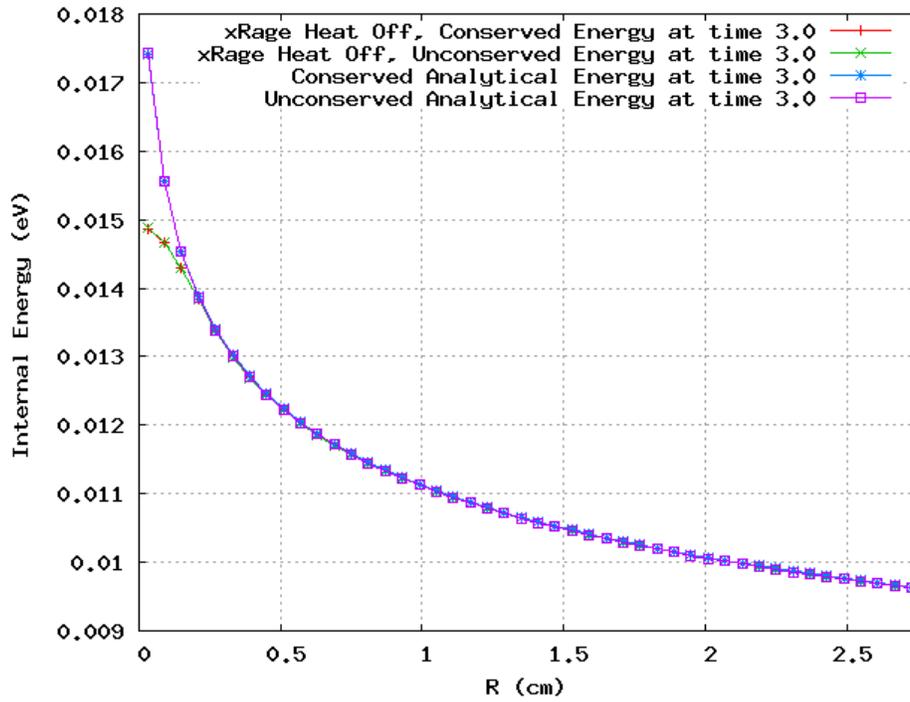


Figure 60: Viewgraph norm of the xRAGE specific internal energy solution with and without conserved cell-averages with the heat conduction module not implemented at  $t=3.0$  s

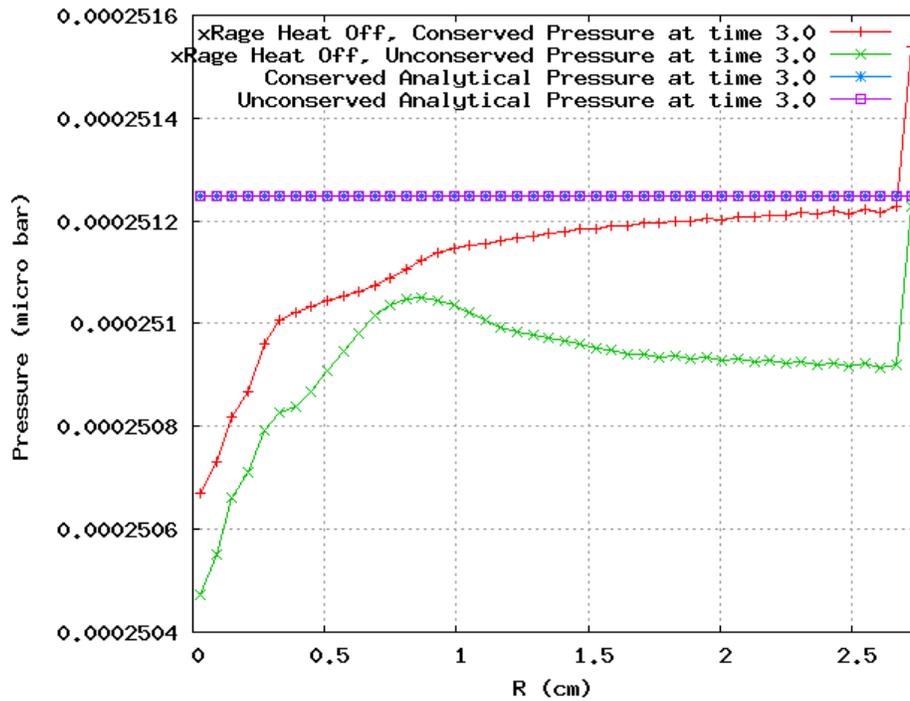


Figure 61: Viewgraph norm of the xRAGE pressure solution with and without conserved cell-averages with the heat conduction module not implemented at  $t=3.0$  s

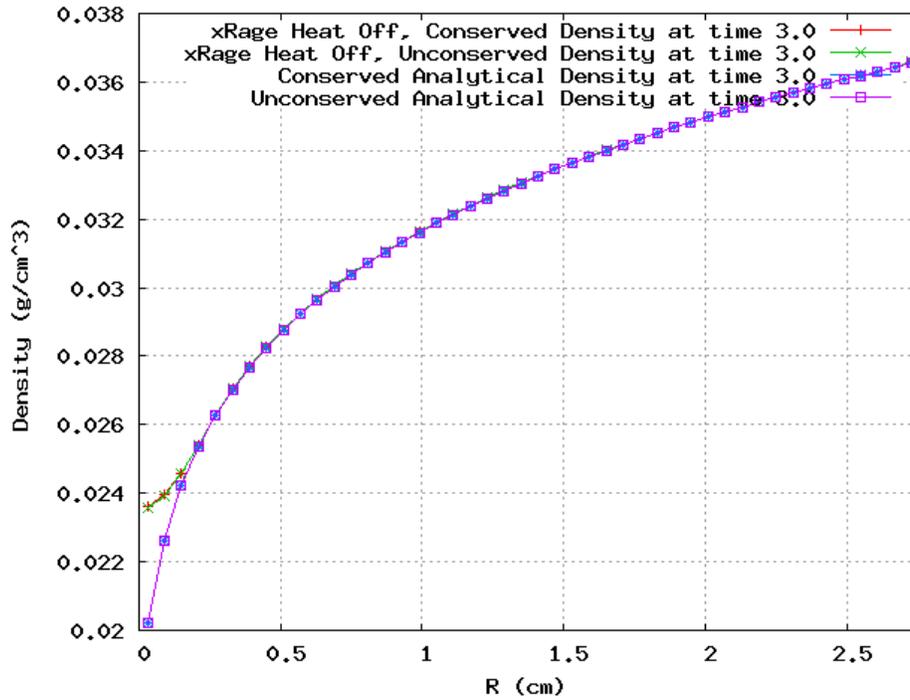


Figure 62: Viewgraph norm of the xRAGE density solution with and without conserved cell-averages with the heat conduction module not implemented at  $t=3.0$  s

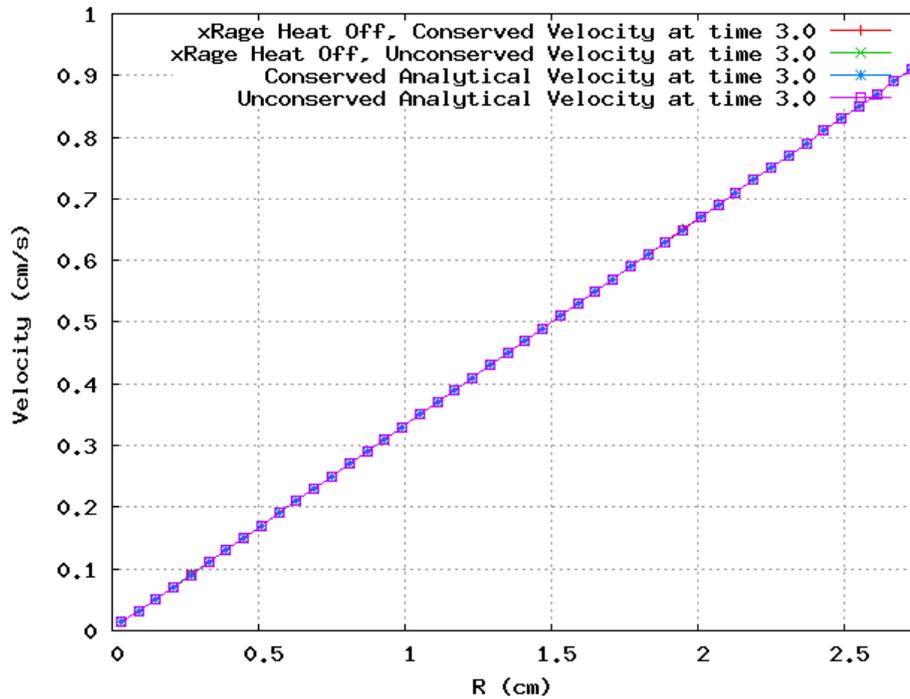


Figure 63: Viewgraph norm of the xRAGE velocity solution with and without conserved cell-averages with the heat conduction module not implemented at  $t=3.0$  s

## Viewgraph Norms with the Heat Conduction Module Implemented with and without Conserved Cell-Averages

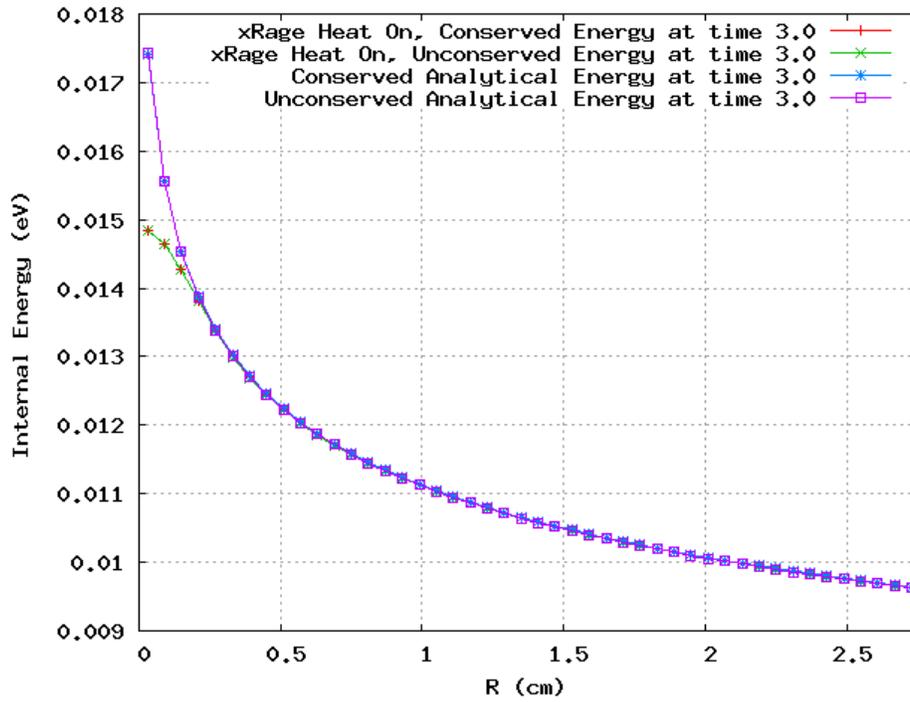


Figure 64: Viewgraph norm of the xRAGE specific internal energy solution with and without conserved cell-averages with the heat conduction module implemented at  $t=3.0$  s

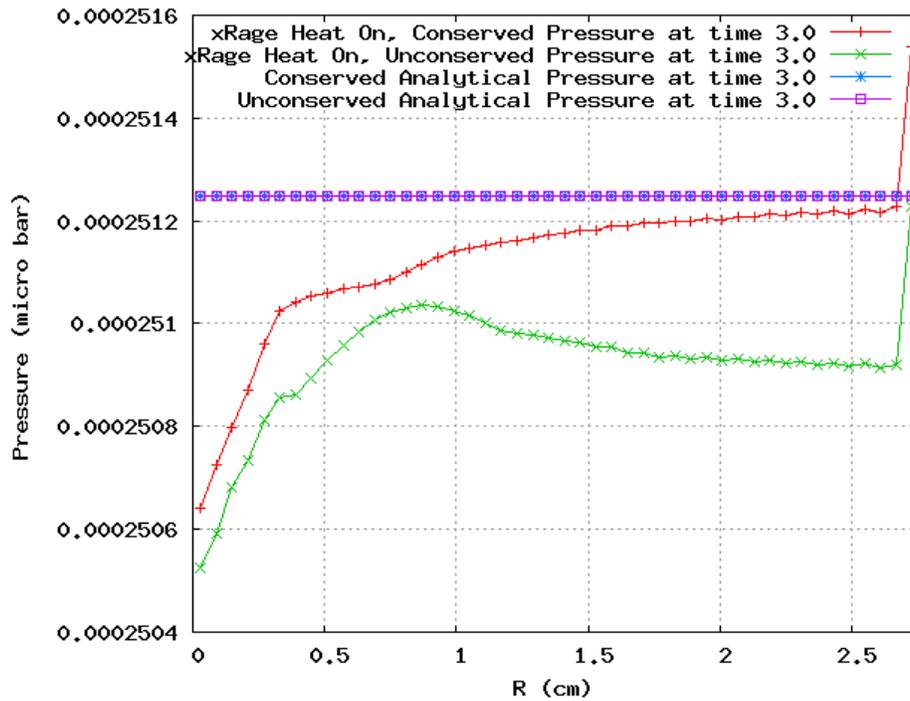


Figure 65: Viewgraph norm of the xRAGE pressure solution with and without conserved cell-averages with the heat conduction module implemented at  $t=3.0$  s

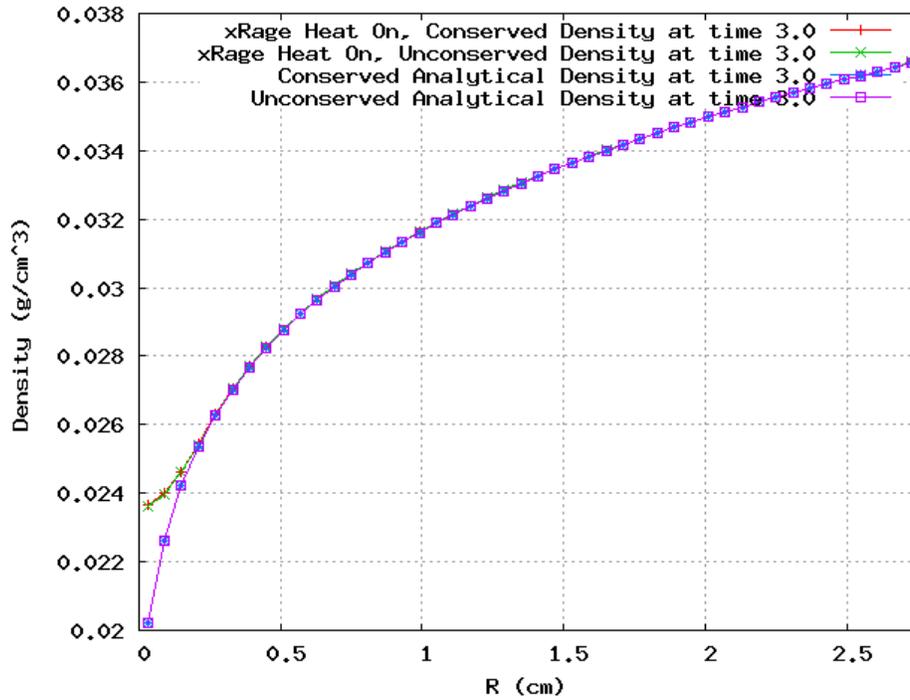


Figure 66: Viewgraph norm of the xRAGE density solution with and without conserved cell-averages with the heat conduction module implemented at  $t=3.0$  s

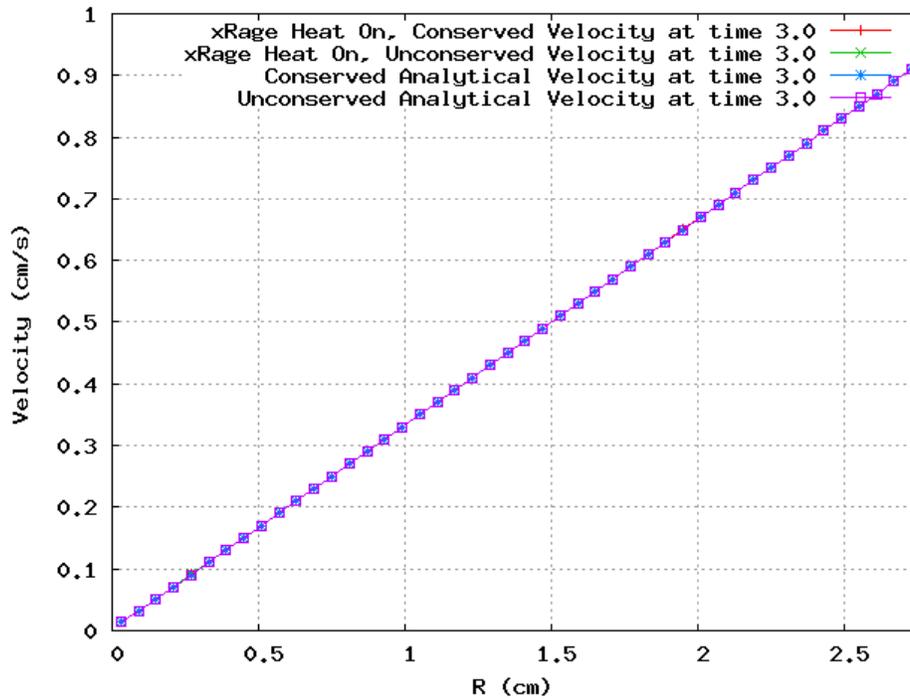


Figure 67: Viewgraph norm of the xRAGE velocity solution with and without conserved cell-averages with the heat conduction module implemented at  $t=3.0$  s

## Conserved, Heat Conduction On

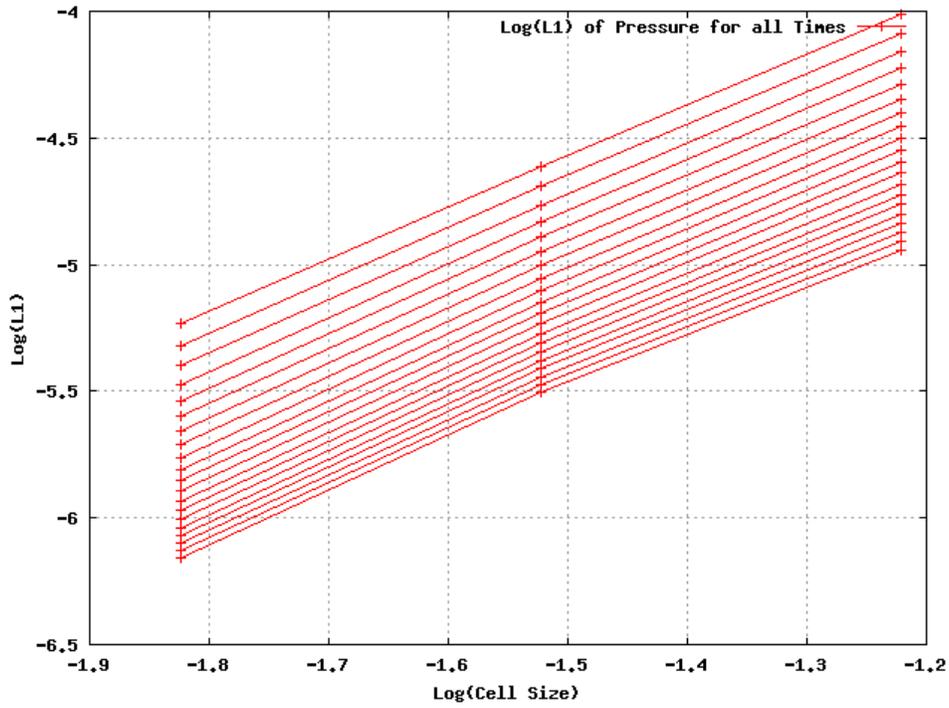


Figure 68: Log-log error plot for all times as a function of cell size for xRAGE pressure using the Coggeshall 11 solution with conserved values and the heat conduction module implemented

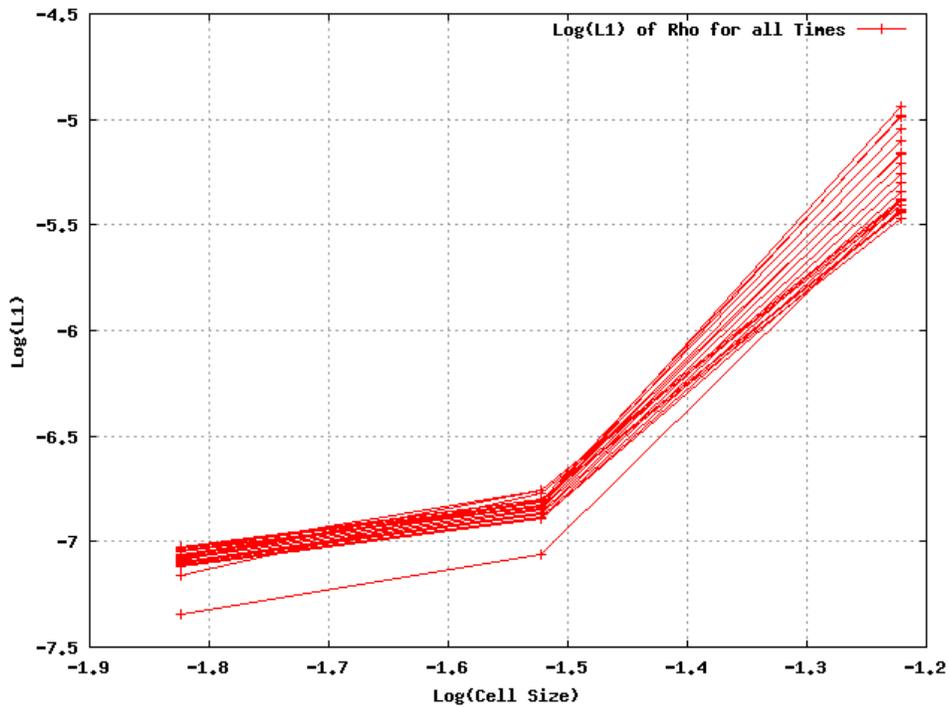


Figure 69: Log-log error plot for all times as a function of cell size for xRAGE density using the Coggeshall 11 solution with conserved values and the heat conduction module implemented

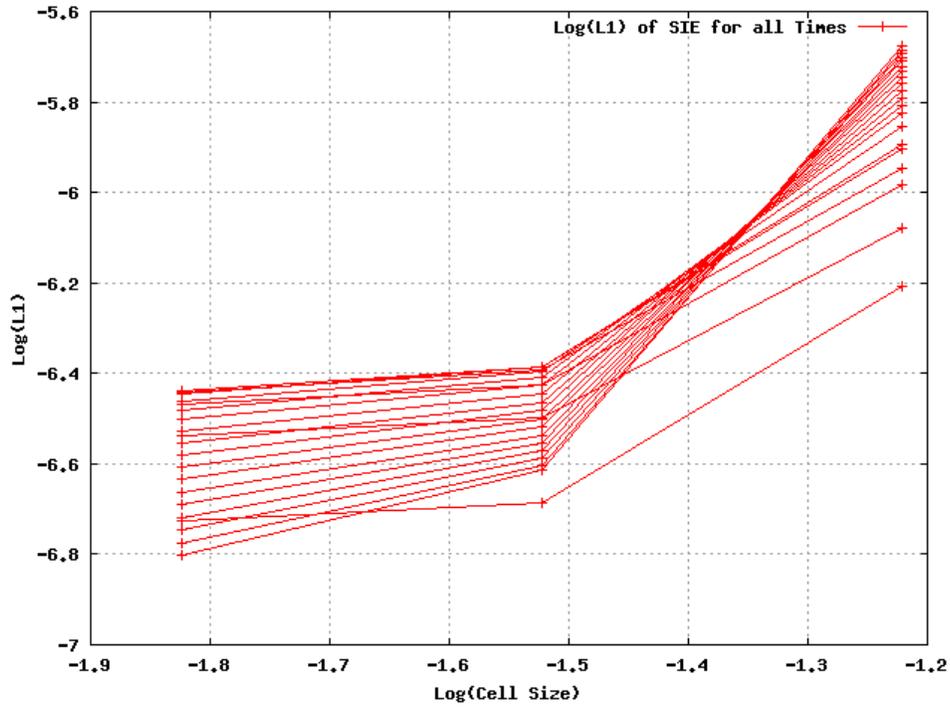


Figure 70: Log-log error plot for all times as a function of cell size for xRAGE specific internal energy using the Coggeshall 11 solution with conserved values and the heat conduction module implemented

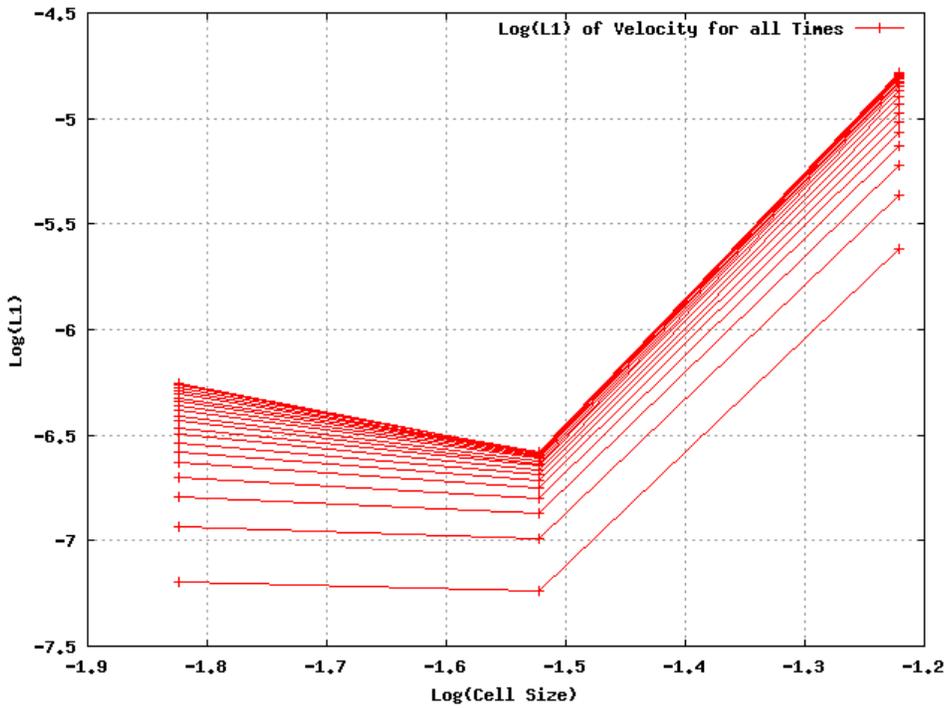


Figure 71: Log-log error plot for all times as a function of cell size for xRAGE velocity using the Coggeshall 11 solution with conserved values and the heat conduction module implemented

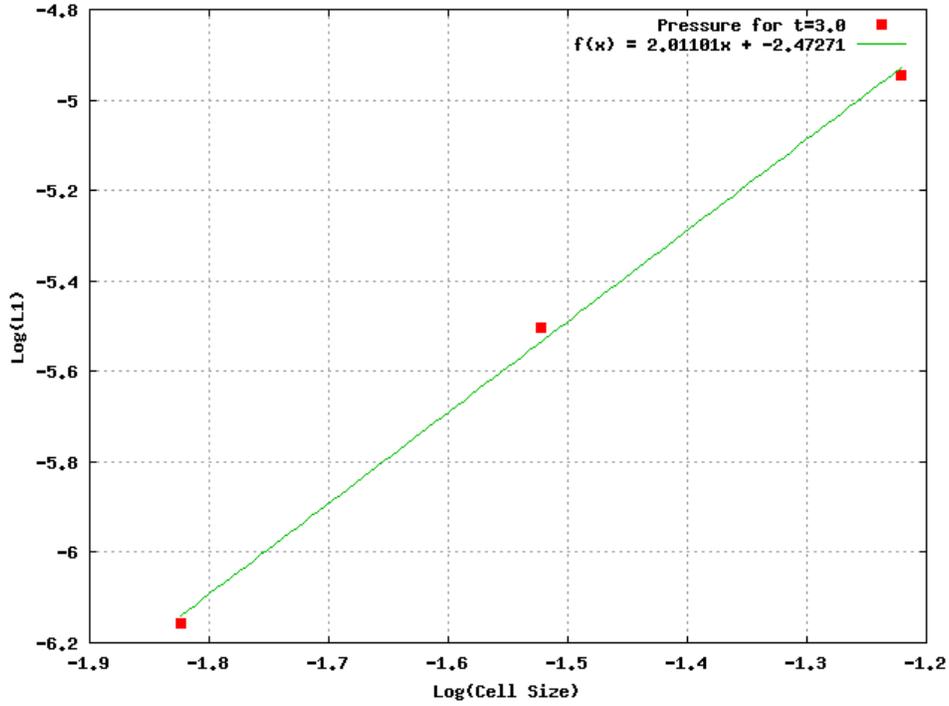


Figure 72: Log-log error plot with a fitted curve as a function of cell size for xRAGE pressure using the Coggeshall 11 solution with conserved values and the heat conduction module implemented

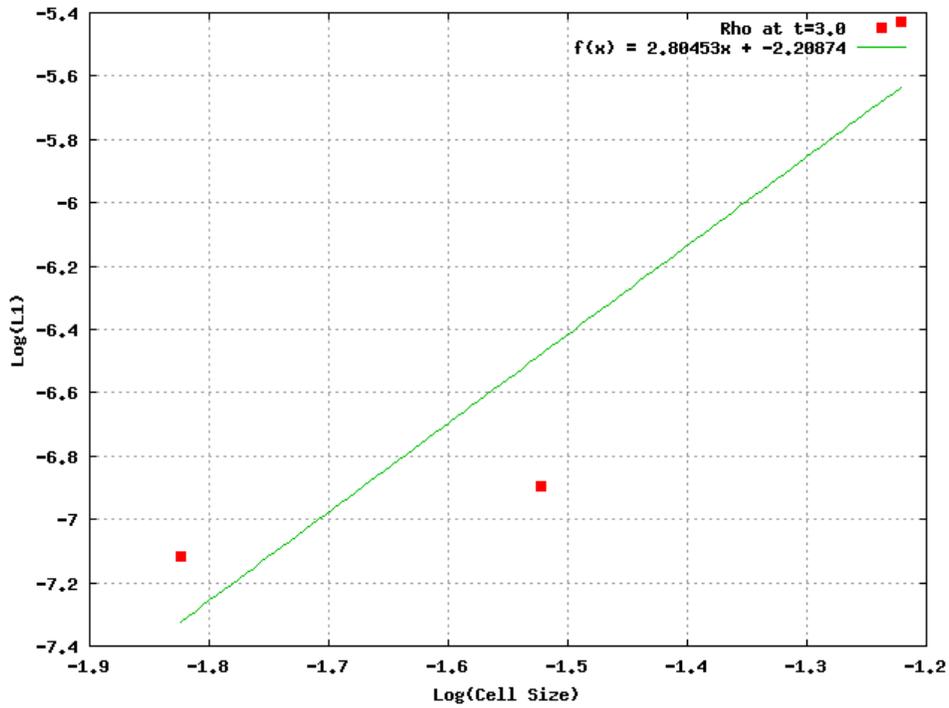


Figure 73: Log-log error plot with a fitted curve as a function of cell size for xRAGE density using the Coggeshall 11 solution with conserved values and the heat conduction module implemented

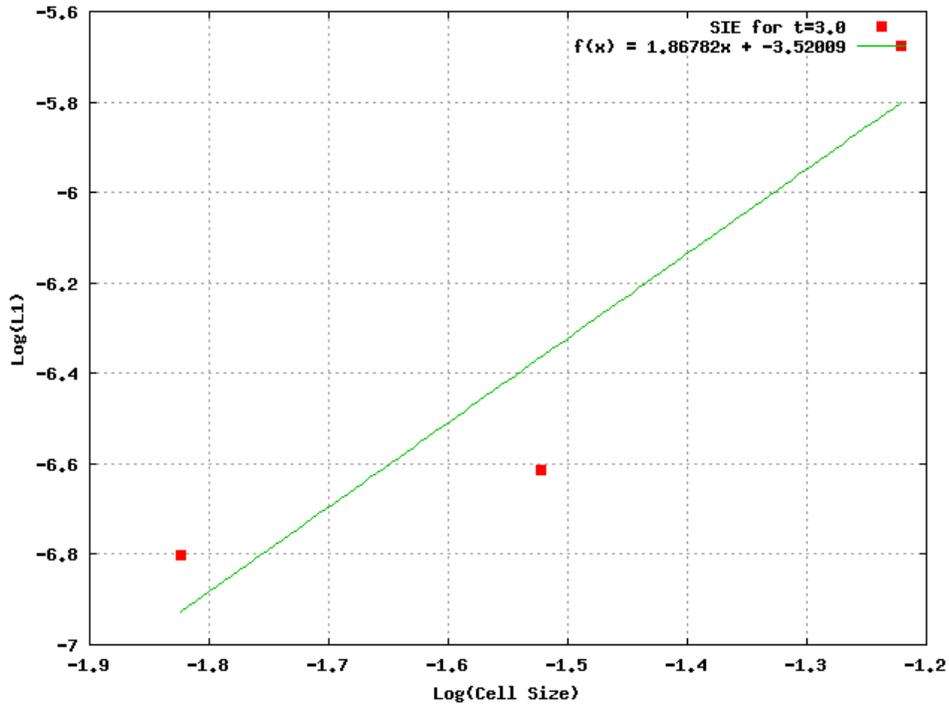


Figure 74: Log-log error plot with a fitted curve as a function of cell size for xRAGE specific internal energy using the Coggeshall 11 solution with conserved values and the heat conduction module implemented

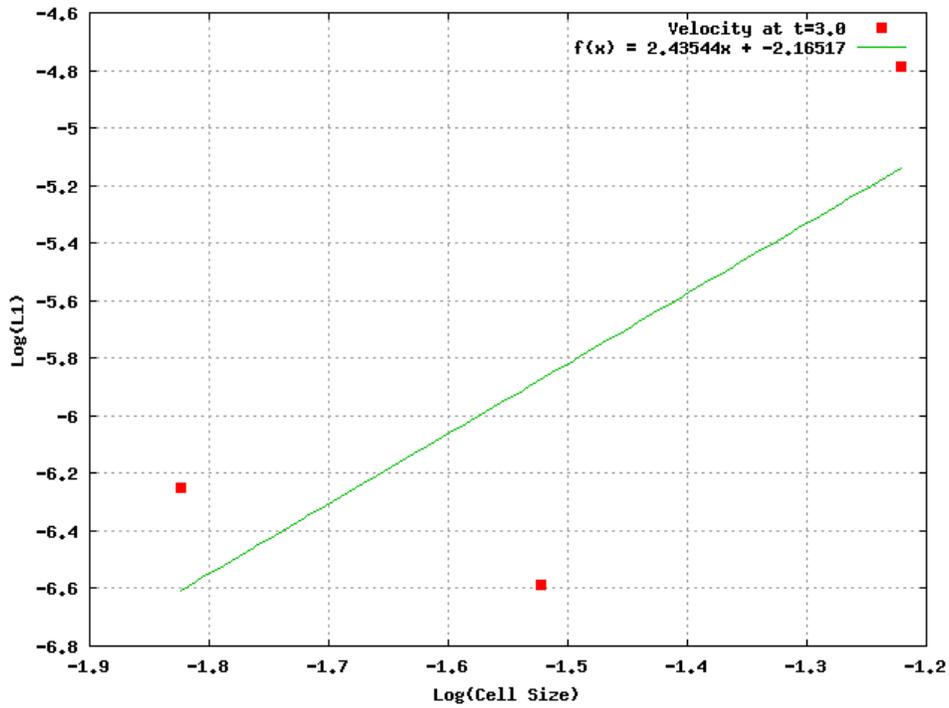


Figure 75: Log-log error plot with a fitted curve as a function of cell size for xRAGE velocity using the Coggeshall 11 solution with conserved values and the heat conduction module implemented

## Non-conserved, Heat Conduction On

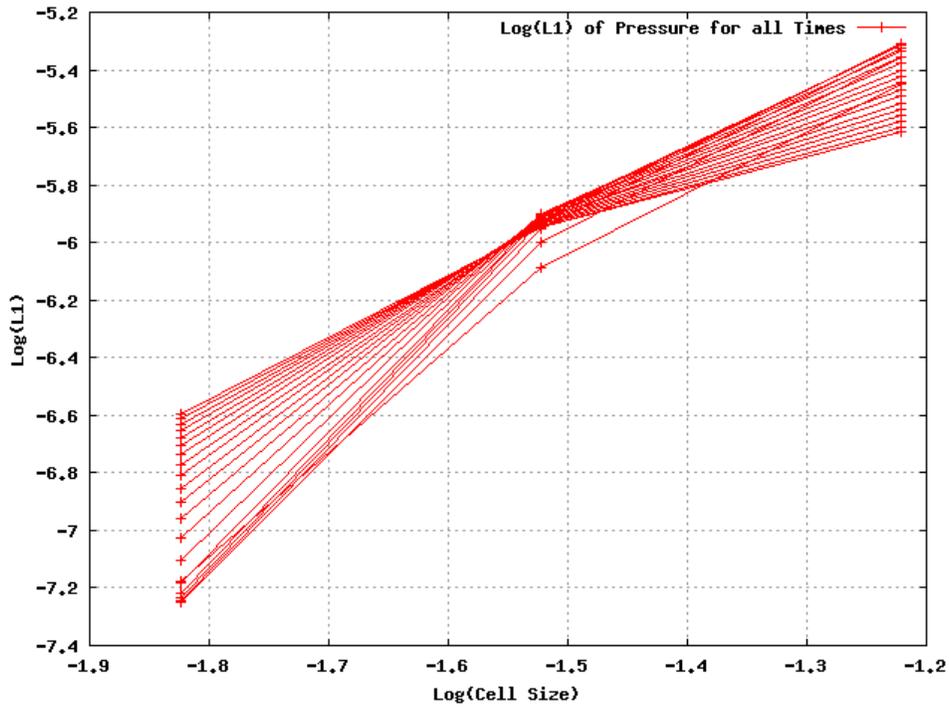


Figure 76: Log-log error plot for all times as a function of cell size for xRAGE pressure using the Coggeshall 11 solution with non-conserved values and the heat conduction module implemented

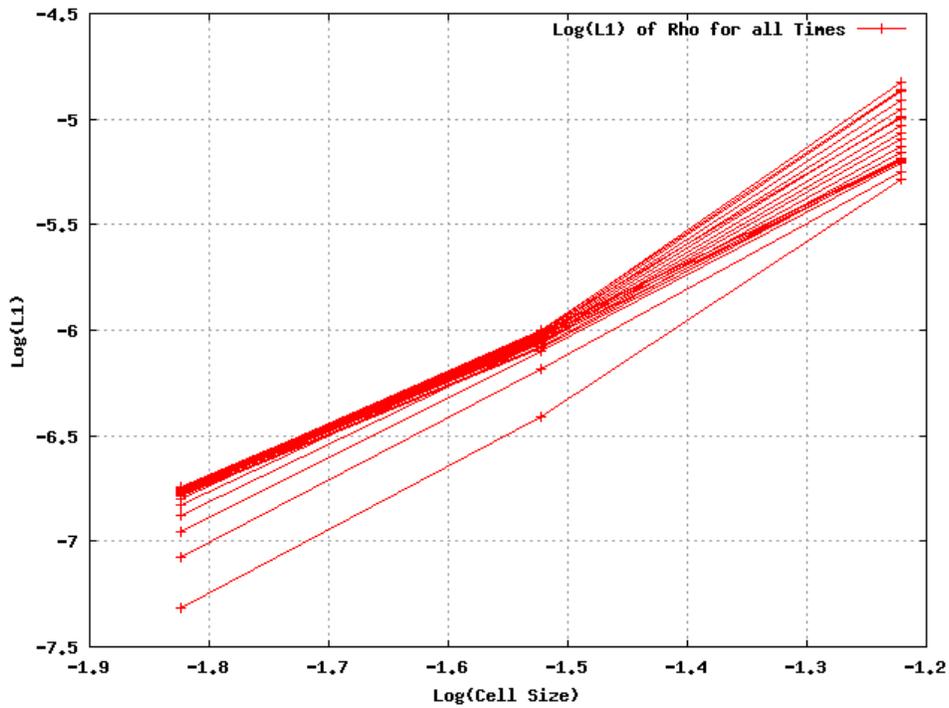


Figure 77: Log-log error plot for all times as a function of cell size for xRAGE density using the Coggeshall 11 solution with non-conserved values and the heat conduction module implemented

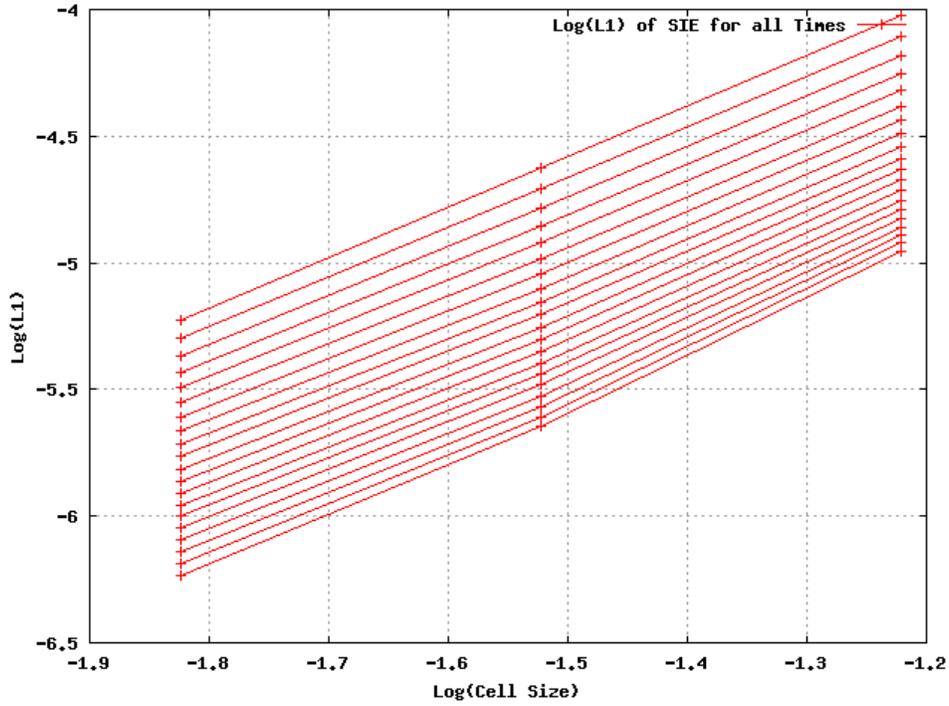


Figure 78: Log-log error plot for all times as a function of cell size for xRAGE specific internal energy using the Coggeshall 11 solution with non-conserved values and the heat conduction module implemented

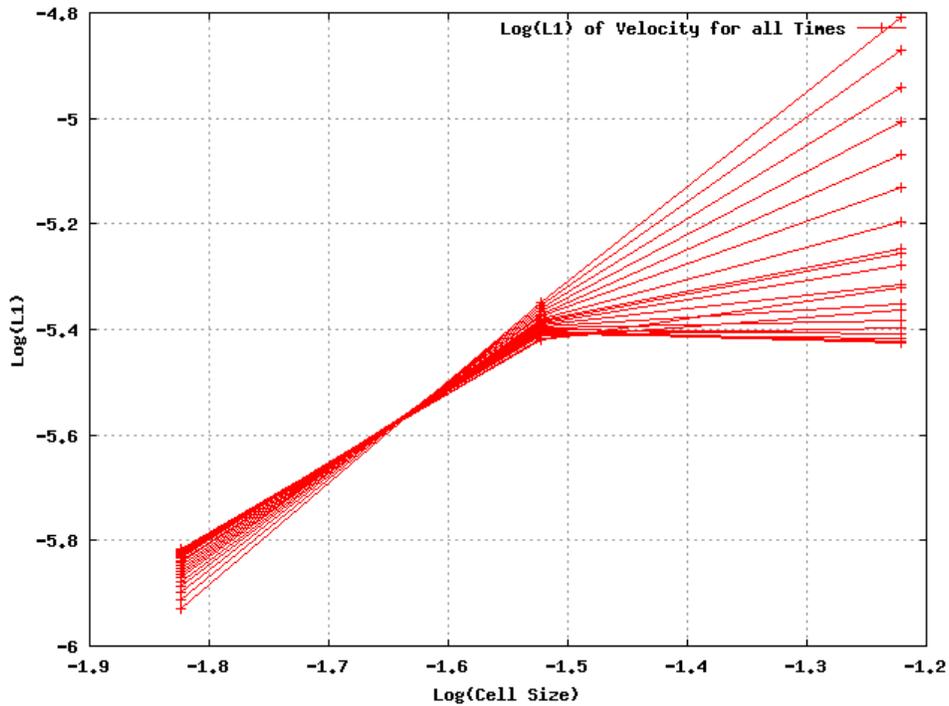


Figure 79: Log-log error plot for all times as a function of cell size for xRAGE velocity using the Coggeshall 11 solution with non-conserved values and the heat conduction module implemented

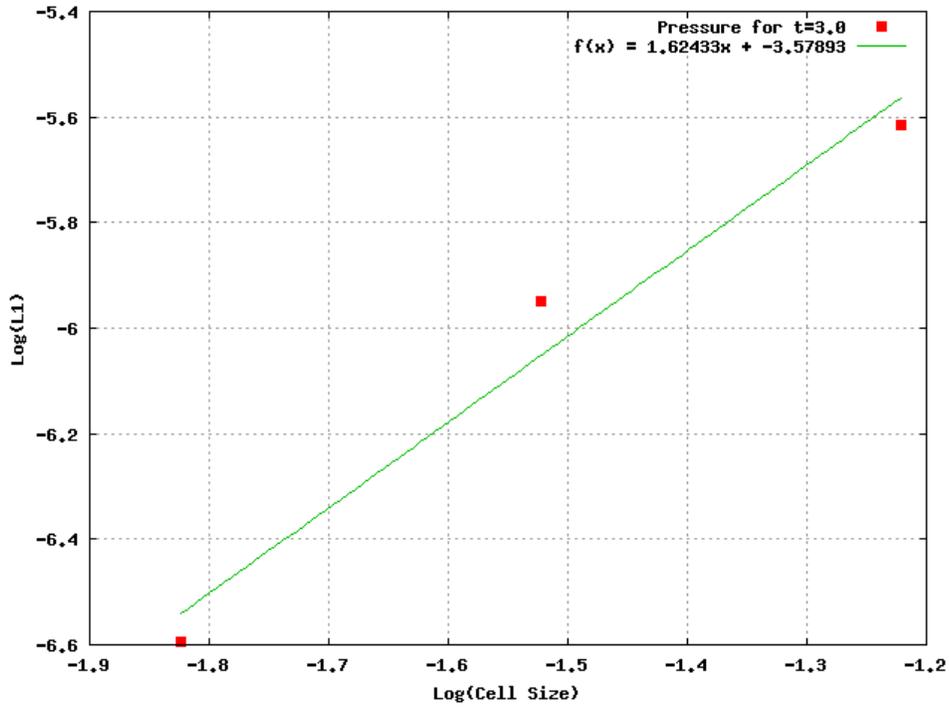


Figure 80: Log-log error plot with a fitted curve as a function of cell size for xRAGE pressure using the Coggeshall 11 solution with non-conserved values and the heat conduction module implemented

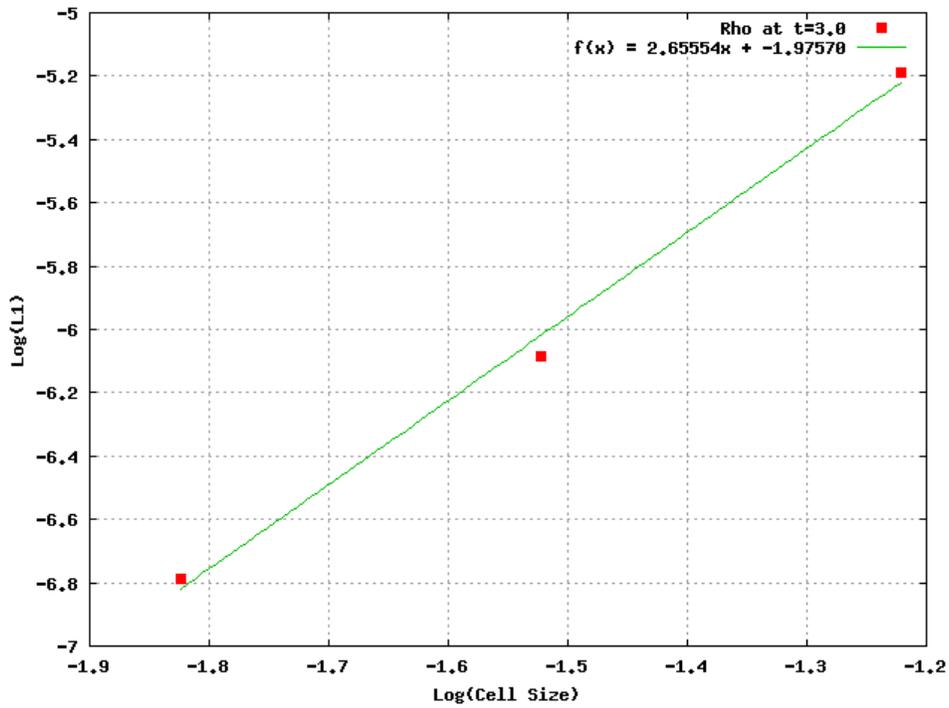


Figure 81: Log-log error plot with a fitted curve as a function of cell size for xRAGE density using the Coggeshall 11 solution with non-conserved values and the heat conduction module implemented

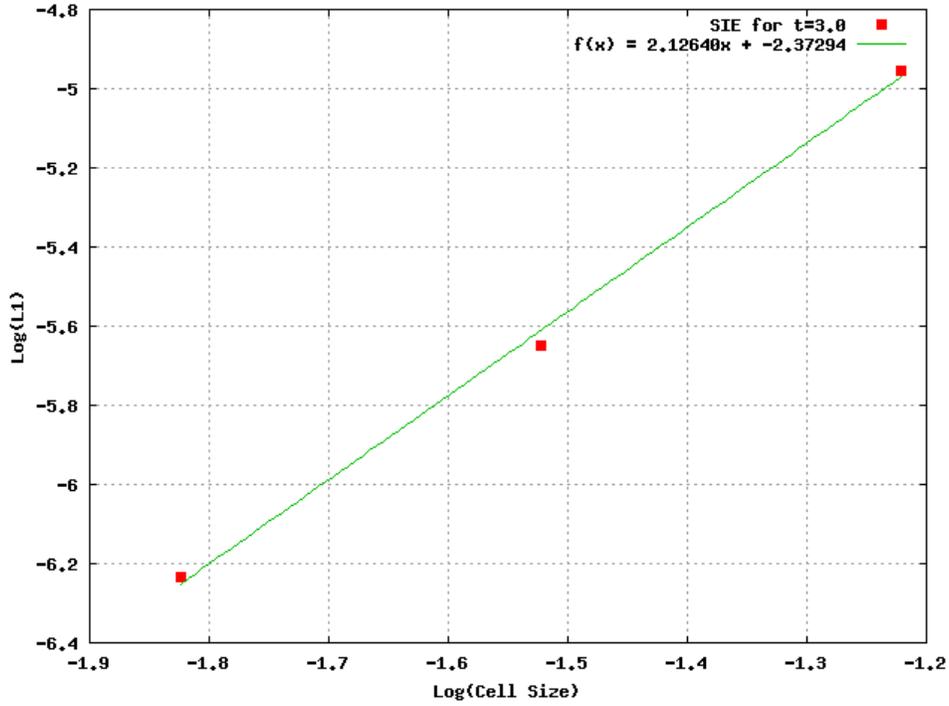


Figure 82: Log-log error plot with a fitted curve as a function of cell size for xRAGE specific internal energy using the Coggeshall 11 solution with non-conserved values and the heat conduction module implemented

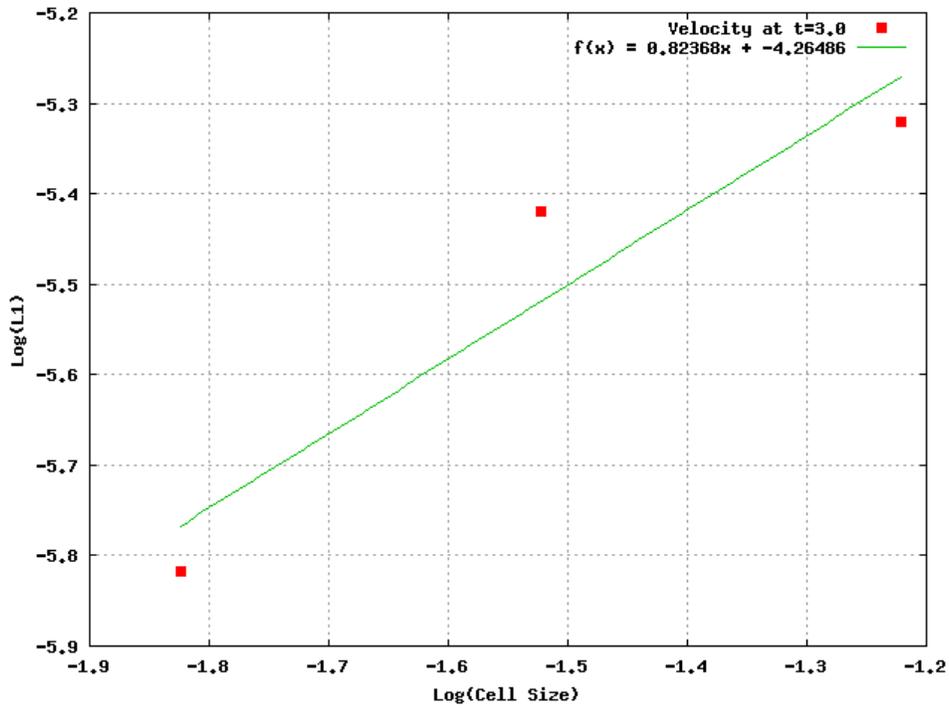


Figure 83: Log-log error plot with a fitted curve as a function of cell size for xRAGE velocity using the Coggeshall 11 solution with non-conserved values and the heat conduction module implemented

### Conserved, Heat Conduction Off

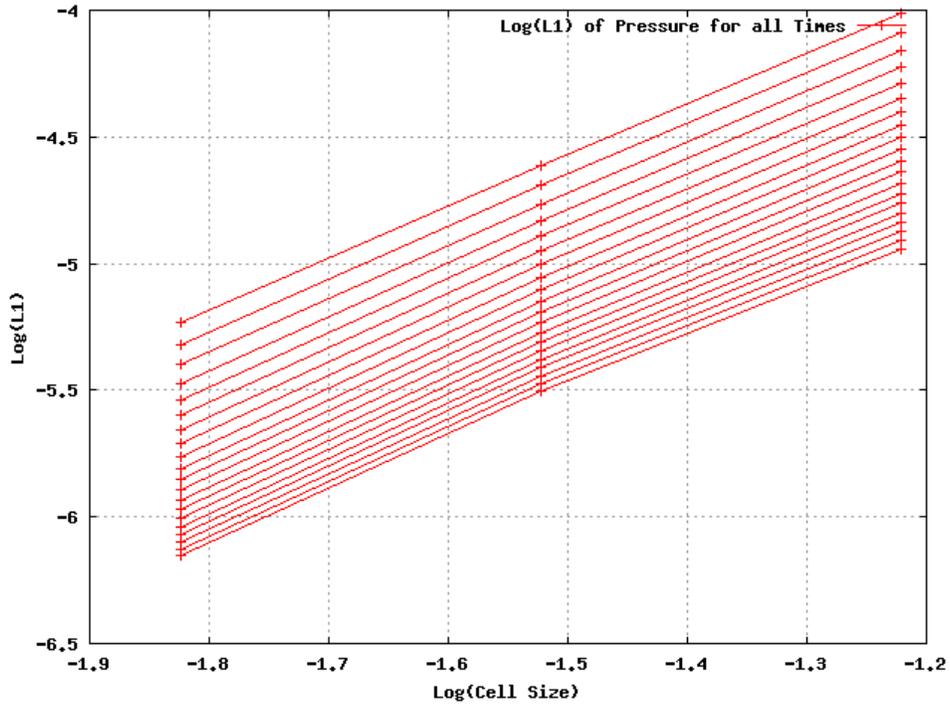


Figure 84: Log-log error plot for all times as a function of cell size for xRAGE pressure using the Coggeshall 11 solution with conserved values and the heat conduction module not implemented

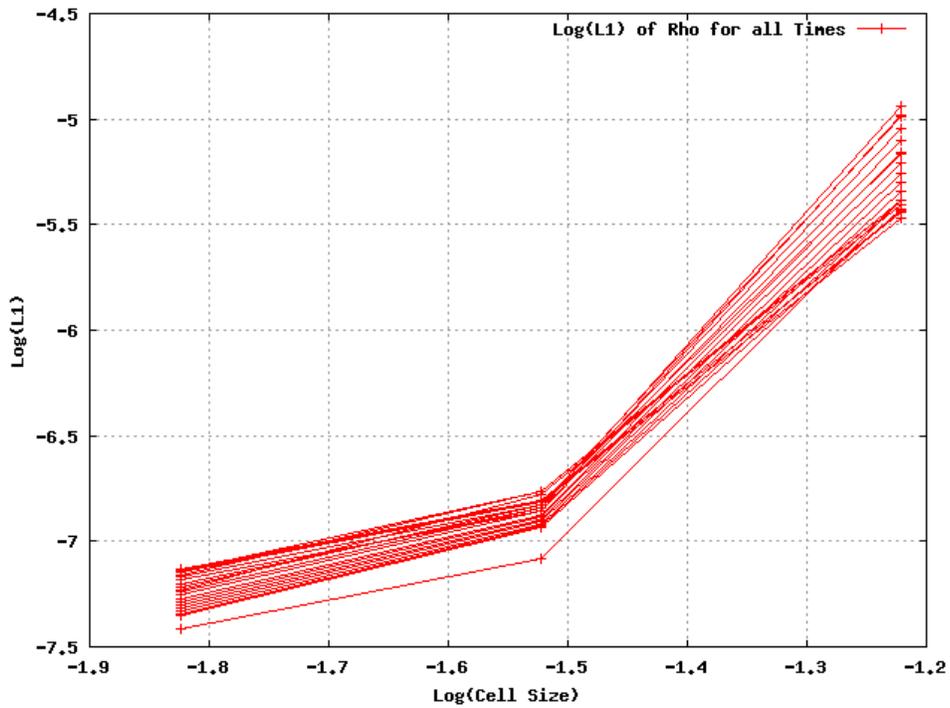


Figure 85: Log-log error plot for all times as a function of cell size for xRAGE density using the Coggeshall 11 solution with conserved values and the heat conduction module not implemented

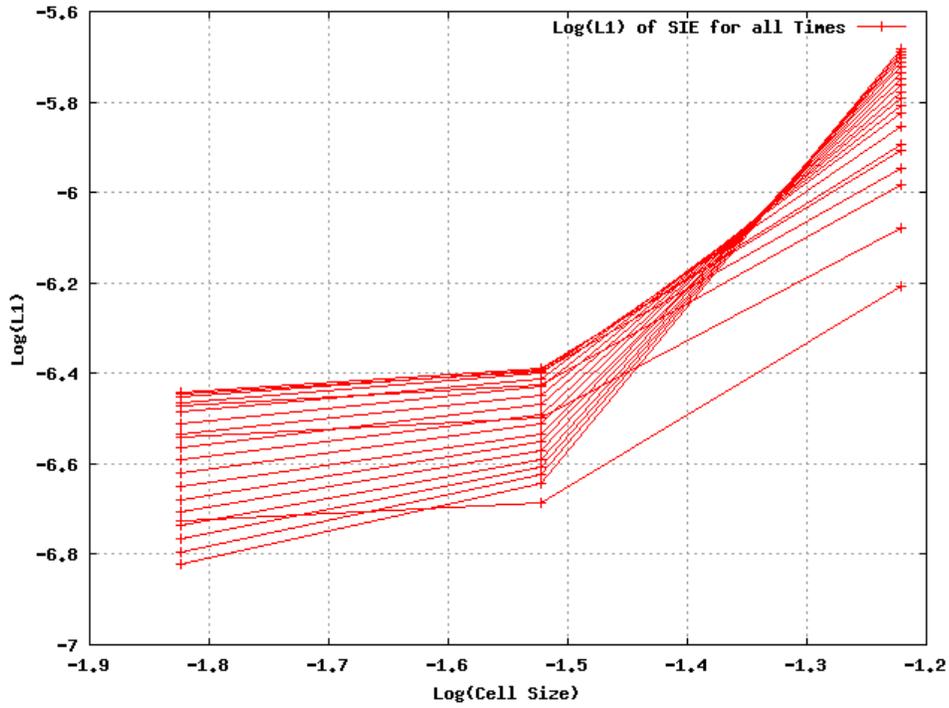


Figure 86: Log-log error plot for all times as a function of cell size for xRAGE specific internal energy using the Coggeshall 11 solution with conserved values and the heat conduction module not implemented

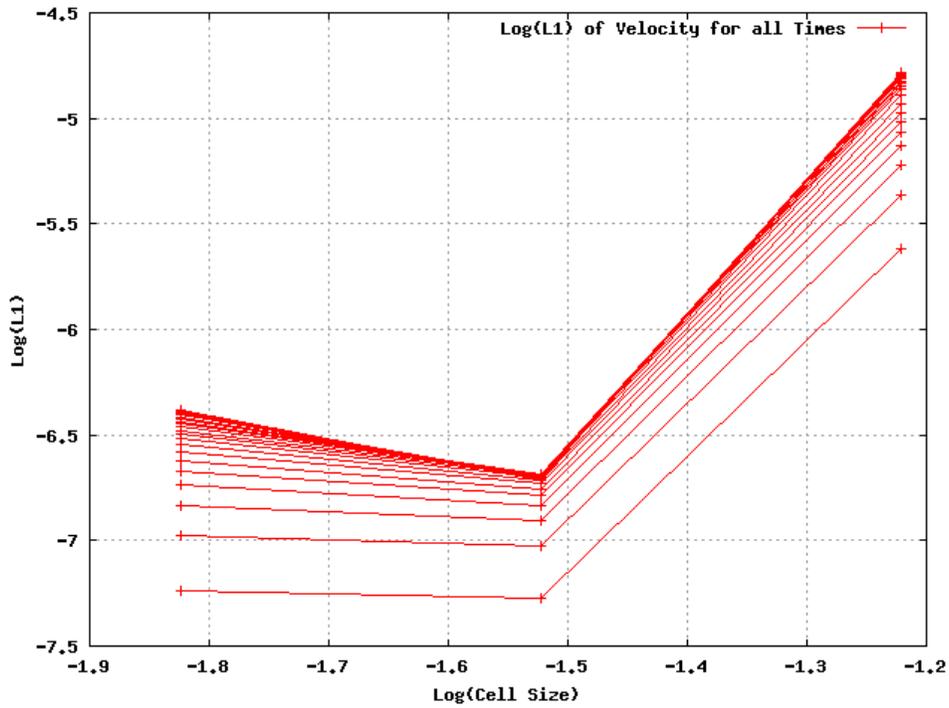


Figure 87: Log-log error plot for all times as a function of cell size for xRAGE velocity using the Coggeshall 11 solution with conserved values and the heat conduction module not implemented

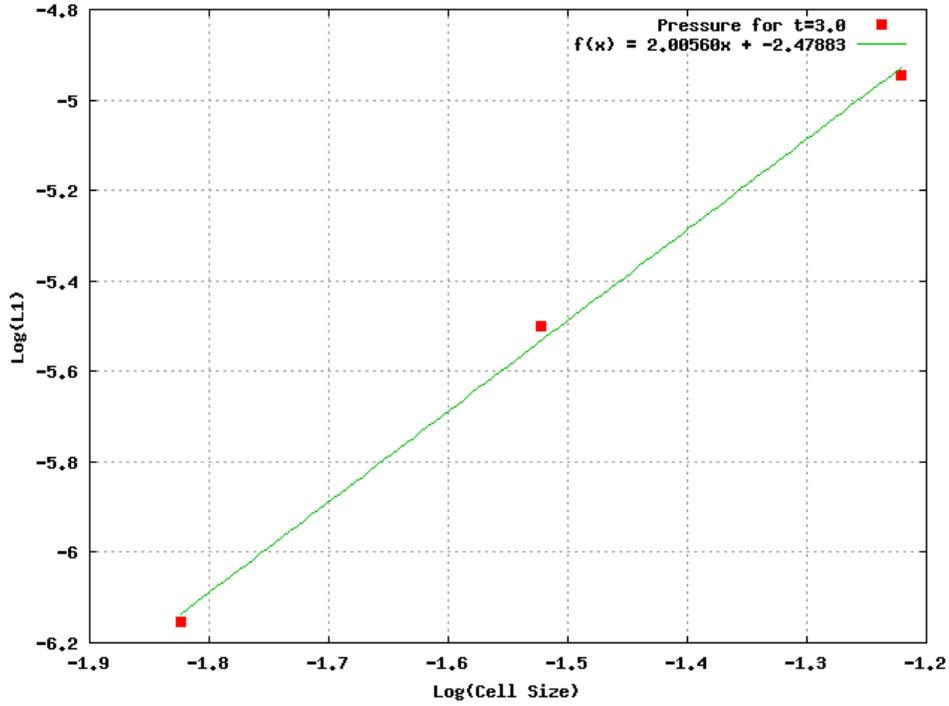


Figure 88: Log-log error plot with a fitted curve as a function of cell size for xRAGE pressure using the Coggeshall 11 solution with conserved values and the heat conduction module not implemented

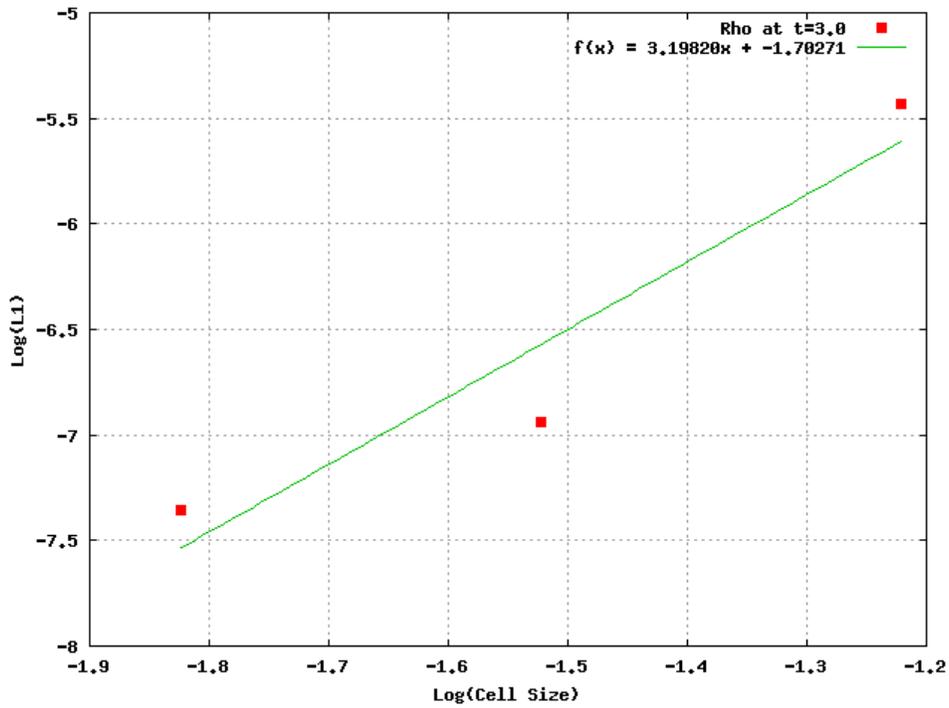


Figure 89: Log-log error plot with a fitted curve as a function of cell size for xRAGE density using the Coggeshall 11 solution with conserved values and the heat conduction module not implemented

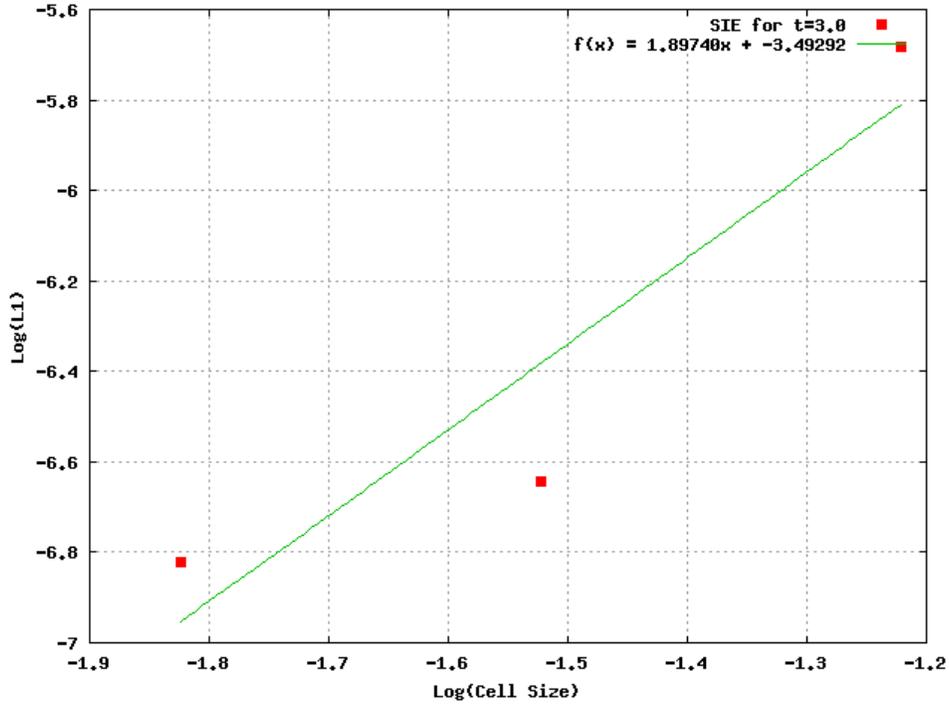


Figure 90: Log-log error plot with a fitted curve as a function of cell size for xRAGE specific internal energy using the Coggeshall 11 solution with conserved values and the heat conduction module not implemented

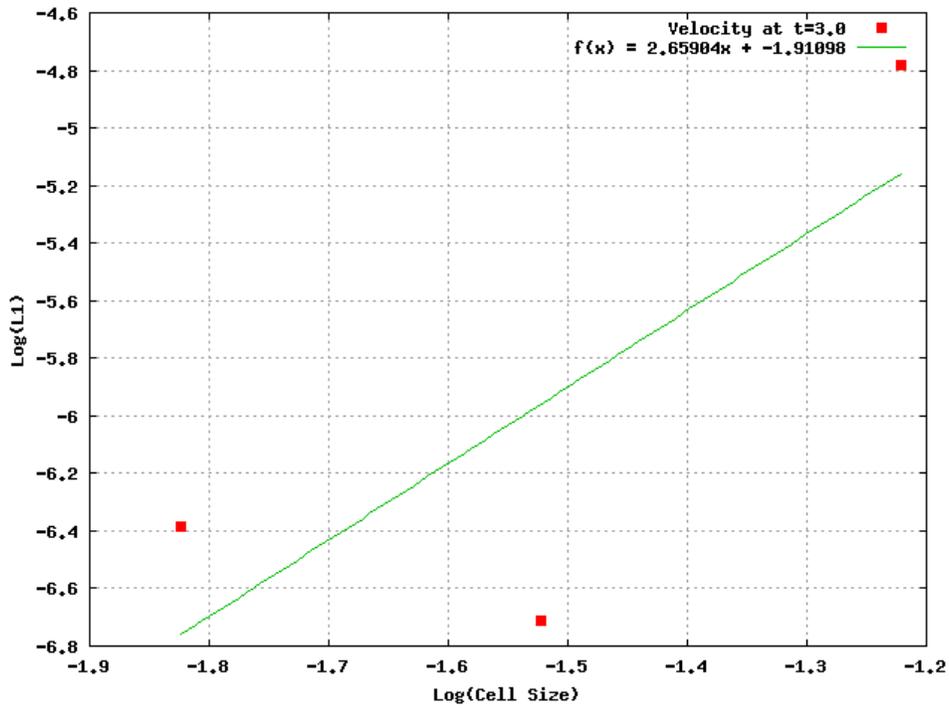


Figure 91: Log-log error plot with a fitted curve as a function of cell size for xRAGE velocity using the Coggeshall 11 solution with conserved values and the heat conduction module not implemented

## Non-conserved, Heat Conduction Off

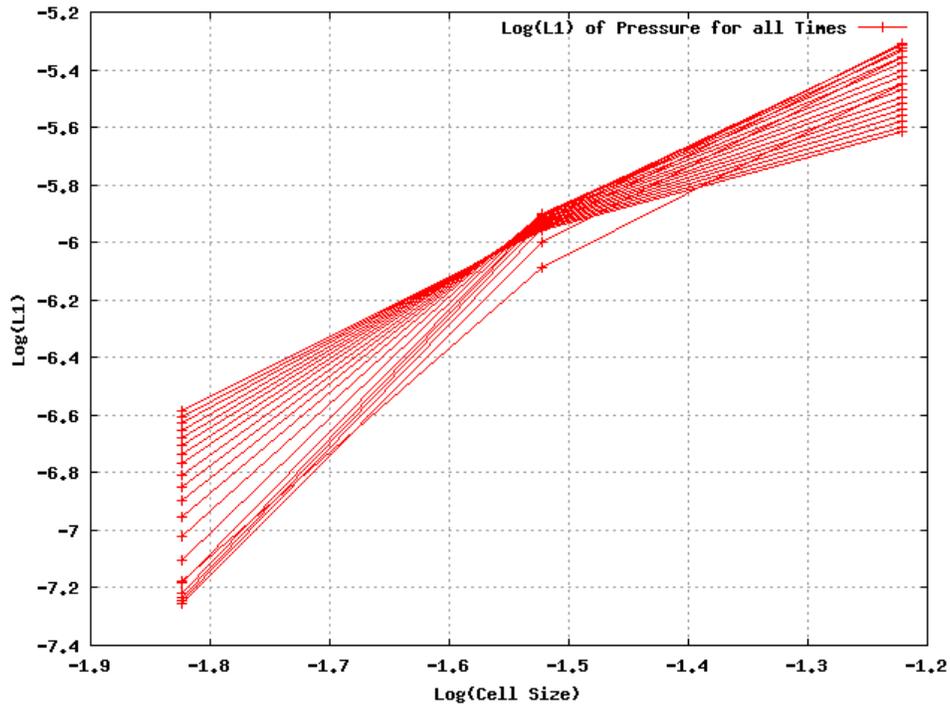


Figure 92: Log-log error plot for all times as a function of cell size for xRAGE pressure using the Coggeshall 11 solution with non-conserved values and the heat conduction module not implemented

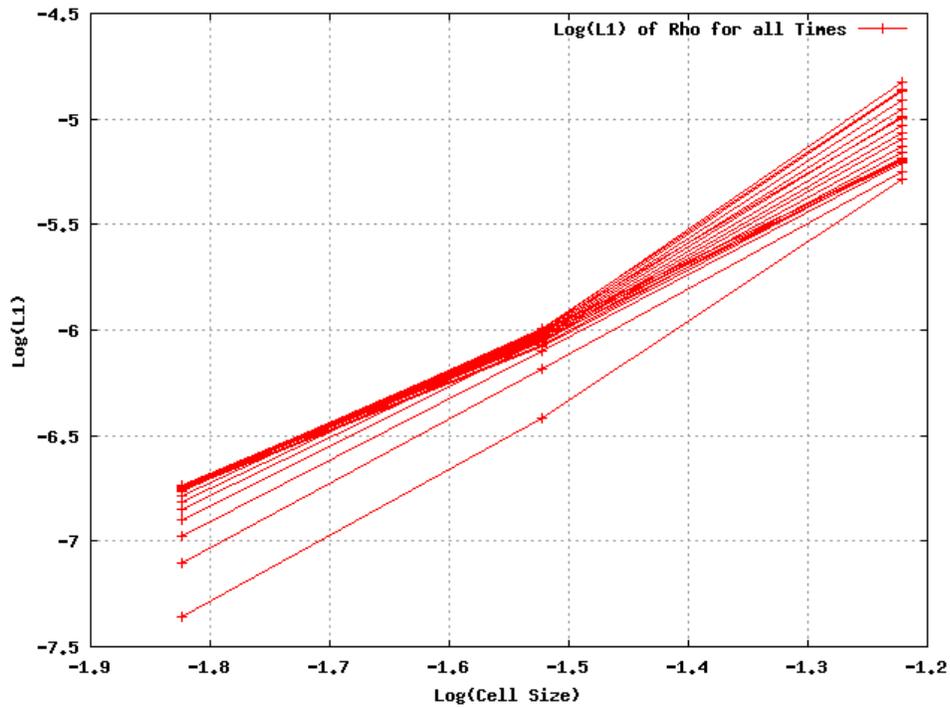


Figure 93: Log-log error plot for all times as a function of cell size for xRAGE density using the Coggeshall 11 solution with non-conserved values and the heat conduction module not implemented

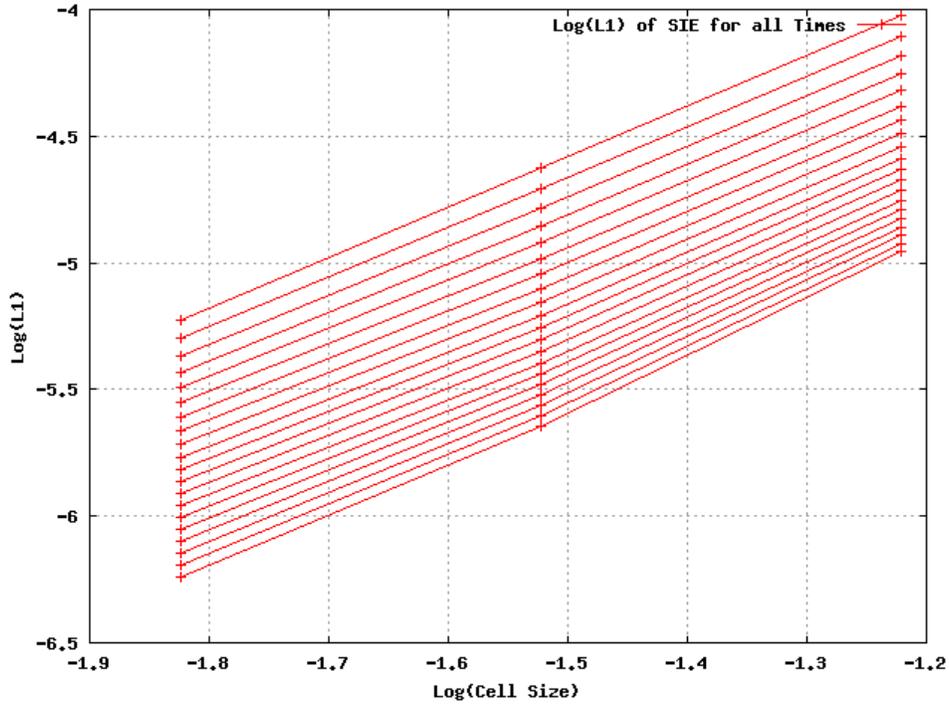


Figure 94: Log-log error plot for all times as a function of cell size for xRAGE specific internal energy using the Coggeshall 11 solution with non-conserved values and the heat conduction module not implemented

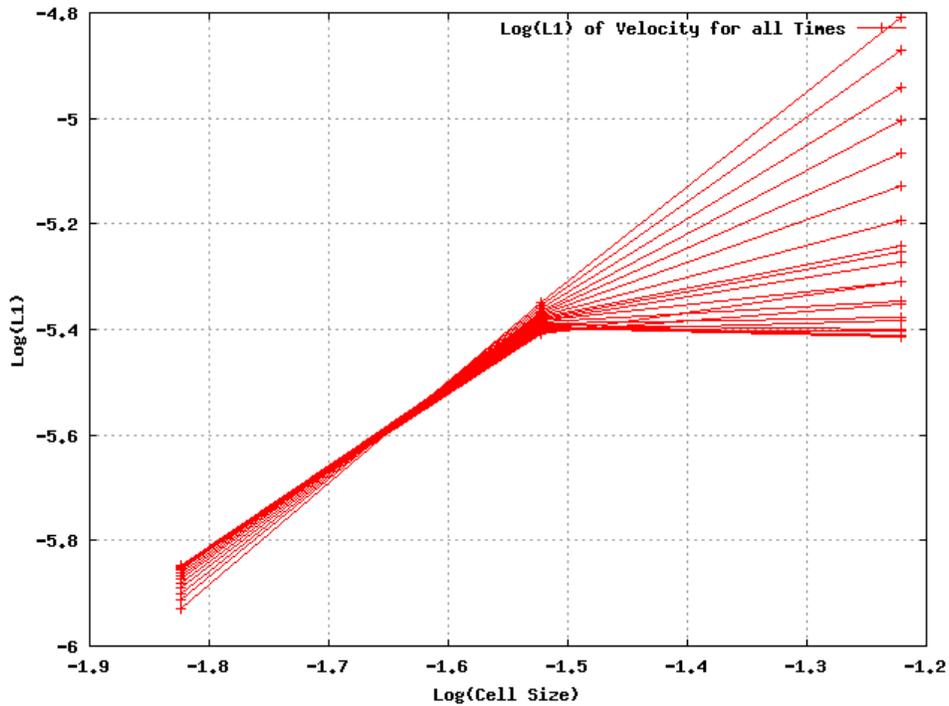


Figure 95: Log-log error plot for all times as a function of cell size for xRAGE velocity using the Coggeshall 11 solution with non-conserved values and the heat conduction module not implemented

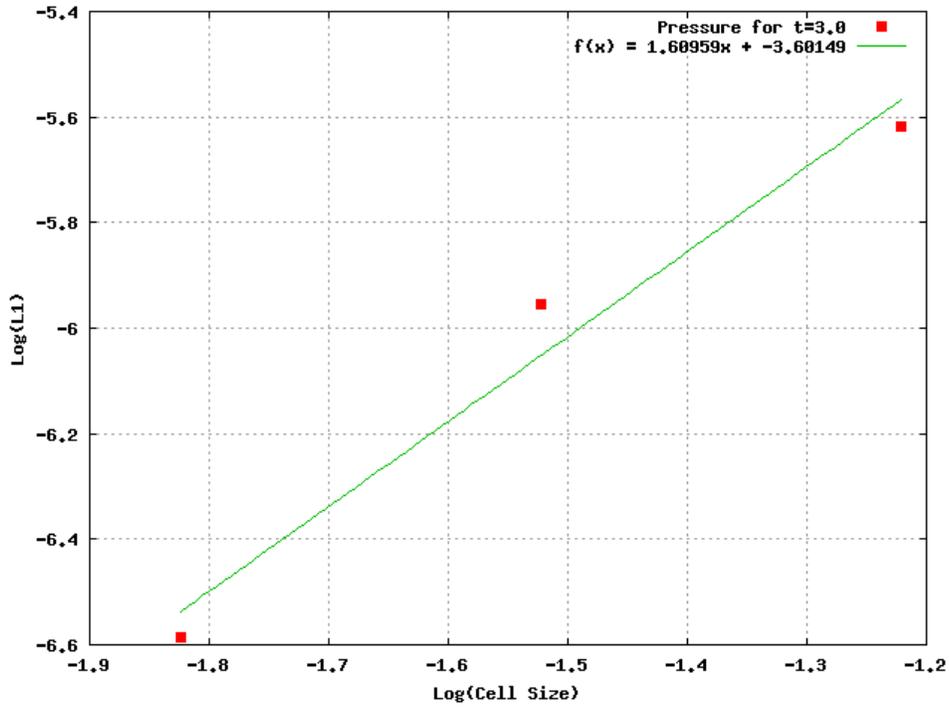


Figure 96: Log-log error plot with a fitted curve as a function of cell size for xRAGE pressure using the Coggeshall 11 solution with non-conserved values and the heat conduction module not implemented

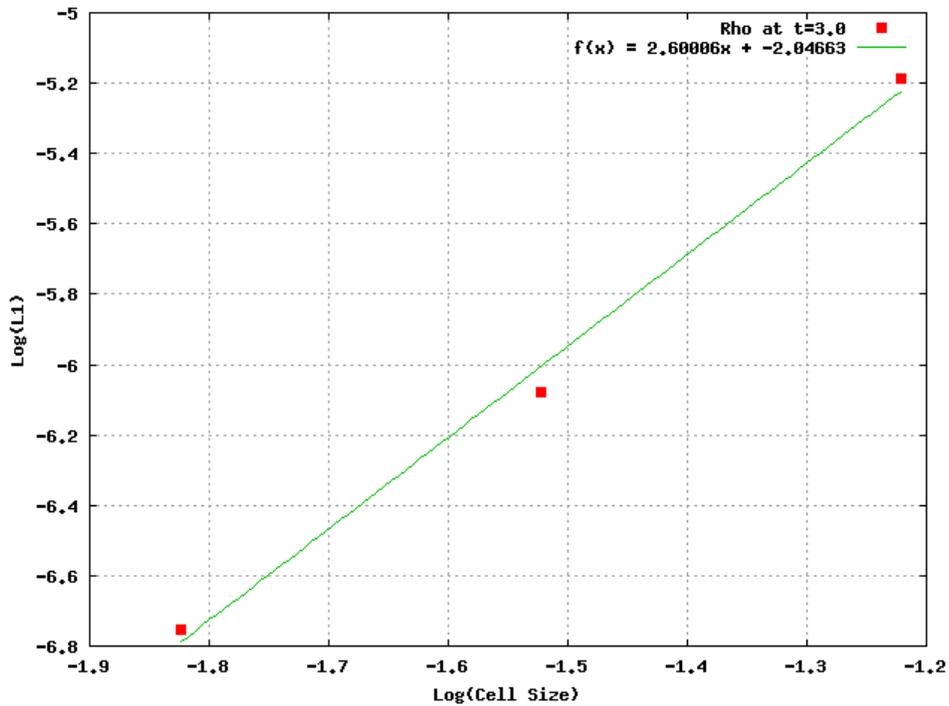


Figure 97: Log-log error plot with a fitted curve as a function of cell size for xRAGE density using the Coggeshall 11 solution with non-conserved values and the heat conduction module not implemented

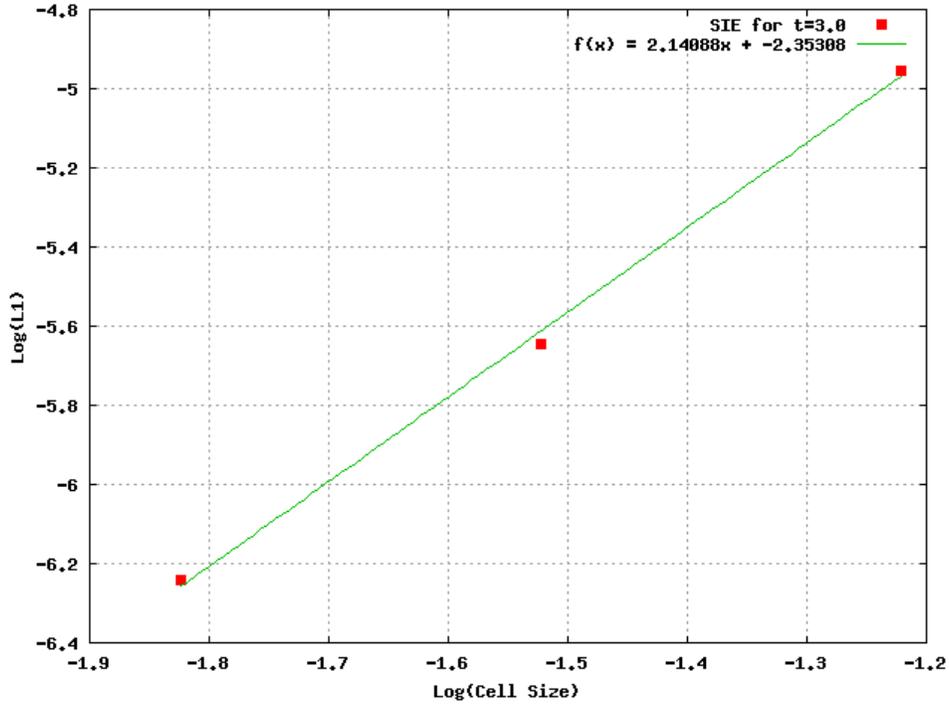


Figure 98: Log-log error plot with a fitted curve as a function of cell size for xRAGE specific internal energy using the Coggeshall 11 solution with non-conserved values and the heat conduction module not implemented

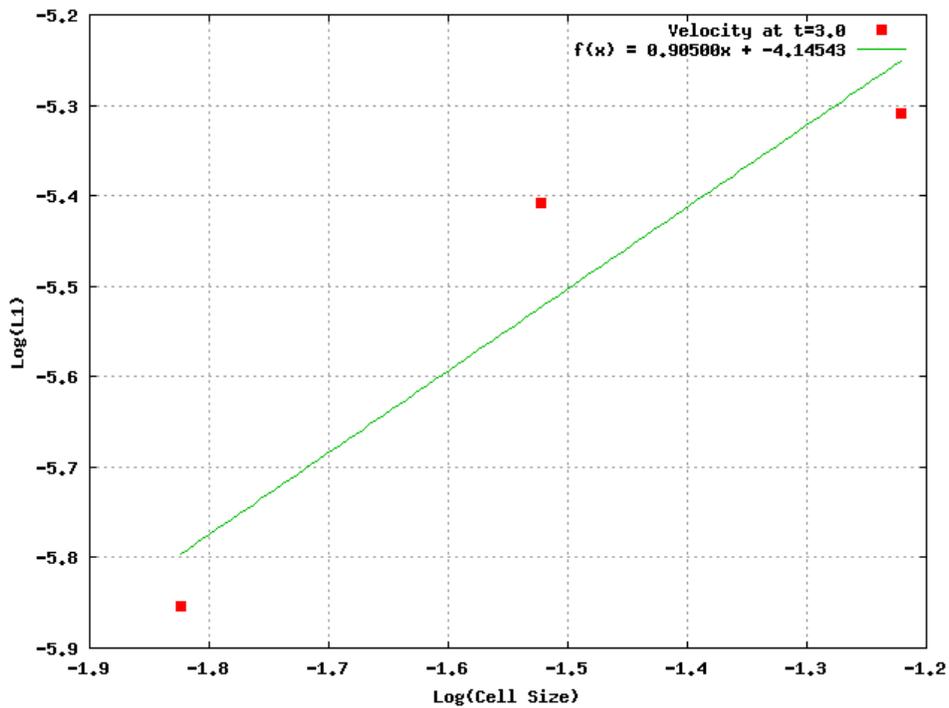


Figure 99: Log-log error plot with a fitted curve as a function of cell size for xRAGE velocity using the Coggeshall 11 solution with non-conserved values and the heat conduction module not implemented

**EMP Simulations**  
**(Heidi Tierney, mentor)**

# Electromagnetic Pulse Simulations (EMP)-Swarm Electron Time Delay in Thunderstorm Environment

Elise Pusateri and Evan Snyder  
Mentor: Heidi Tierney

Summer 2012 LANL Computational Physics Workshop

## 1 Introduction

Our work in the 2012 Computational Physics Workshop included the development of a swarm model for use in both lightning physics and the larger electromagnetic pulse (EMP) project. This model is designed to analyze the evolution of a swarm of electrons that is created by photoelectron ionization of air and Townsend impact ionization at a given air density. Here, the results are presented for a mid-thunderstorm altitude of 5 km. The code uses an adaptive timestep and solves a system of coupled differential equations for electric field, electron temperature, swarm electron number density, and drift velocity. For lightning, the purpose of this work is two fold. First, we are interested in the amount of time it takes for the newly generated electrons equilibrate to the ambient electric field and pressure to reach a characteristic electron temperature that defines a swarm. In this context a swarm means that the electrons are in thermodynamic equilibrium with themselves, but not necessarily with the background. Many models treat the conduction or swarm electrons as responding instantaneously to the environment. In reality, the electron swarm has a time dependent characteristic temperature. Second, in runaway breakdown theory, the ionization produced by photoelectrons is often overlooked. Here we calculate this contribution to the swarm electron density and discuss the possible implications for lightning initiation.

A theory of lightning initiation involves high energy seed electrons (about 1 MeV) that come about from cosmic rays.[5] If these high energy electrons enter a region of the atmosphere that has a sufficient electric field over a distance of about 1 km or more, measurable enhancements in background radiation take place. The high energy electrons interact with surrounding air molecules to create Bremsstrahlung (X-ray) radiation, positive and negative ions, conduction electrons [9], and possibly further high energy electrons through the runaway breakdown process. X-ray production is highly correlated with the strength of the large scale thunderstorm electric field.[3] Figure 1 shows this measured X-ray intensity. X-ray intensity lasted for a longer time period (about one minute) compared to the time periods of any lightning flash ( $< 1\text{sec}$ ). *McCarthy et al.*, [1985] discusses two scenarios in which X-ray measurements a factor of one thousand above background levels can be produced. One possibility is that the energetic electrons are gaining enough energy from the thunderstorm electric field to compensate for and enhance Bremsstrahlung losses. The other possibility is that runaway breakdown is taking place and the newly generated high energy electrons are also gaining energy from the field and emitting Bremsstrahlung X-rays. Our model only requires the X-ray measurements to inform us about the photoelectron source rather than detailed knowledge of the processes that create the X-ray enhancements prior to a lightning strike.

The photoelectron number density per second is created by the x ray flux. The photoelectrons then experience a frictional drag force as they traverse through a thundercloud, which is proportional to  $Z\ln(E_0)$ , where  $Z$  is the atomic number of the molecule that the electron interacts with, and  $E_0$  is the initial energy of the electron. This process varies with energy and atmospheric height as can be seen in Figure 2.[2] Figure 2 also shows two electric field strengths, including the breakdown threshold electric field for laboratory sparks and a typical thunderstorm field. The dynamical friction force is a minimum for electrons of energy near 1 MeV. The high-energy electron distribution tends to equilibrate to this value since those that gain energy from the field will then lose more energy from the friction force. Those electrons that fall below the minimum energy to gain more energy from the E field than they lose in creating ion pairs will gradually add to the low energy, conduction, electron group.

If the electrons experience a high enough electric field (whether it be external or self-consistent), then the electrons can gain more energy from the field than they lose from the frictional and radiative forces. When this

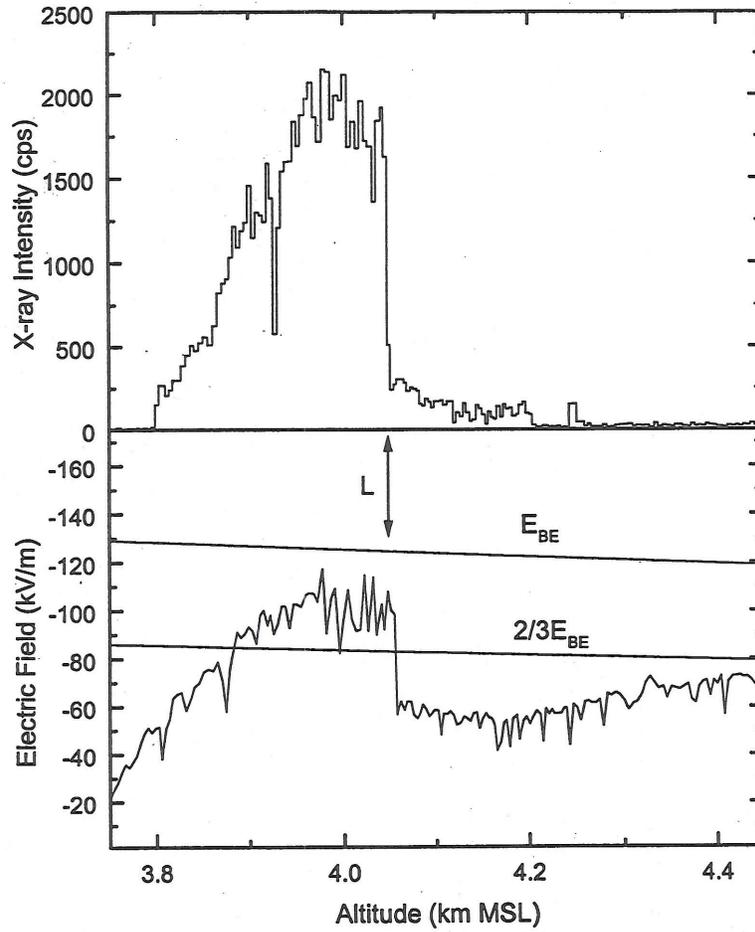


Figure 1: X-ray intensity observed in a measurement by *Eack et al* [1996]. It should be noted that the low X-ray intensity near 3.9 km has a spectrum similar to that of the background that was observed before and after the event.[4]

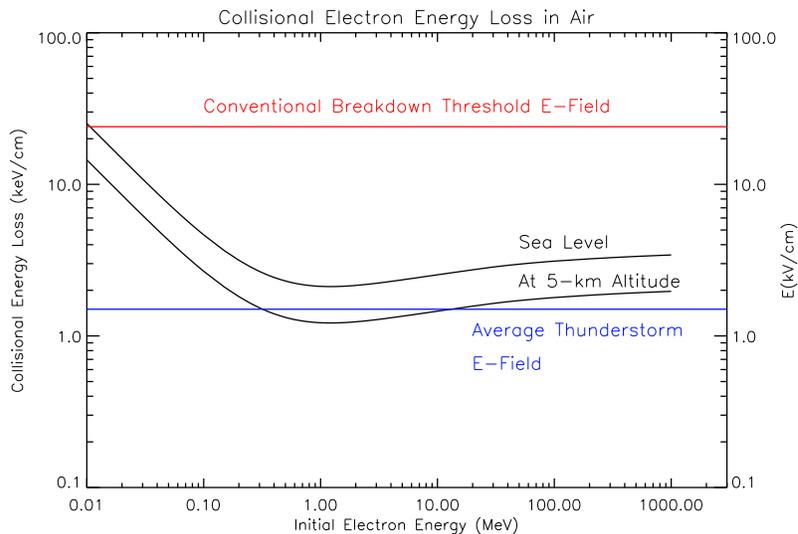


Figure 2: Collisional energy loss and electric field versus fast electron energy in air.

occurs, Bremsstrahlung radiation is increased and more secondary electrons can be ejected. As an example, radiative energy losses and collision energy losses versus electron energy with no electric field present can be seen in Figure 3.[2] This phenomenon is referred to as Runaway Breakdown.

Runaway breakdown will stop when the thunderstorm electric field is depleted. This can happen because all the new electrons that join the swarm increase the conductivity of the thunderstorm region in which runaway breakdown is taking place or because a lightning flash depletes the thunderstorm charge. The conduction electrons also attach to molecules, reducing the conductivity. We account for attachment in this paper.

In this paper, we explore four coupled ODE's formed by swarm theory and the effects they have on lightning initiation and propagation in the low energy regime from about (1-10) keV. Swarm theory assigns a mean temperature to the electrons and relates their drift velocity to the electric field and the energy transfer and momentum transfer collision frequencies.[7] Of primary interest are the conduction electrons and the resultant conduction current. The primary current comes from Compton electrons and photoelectrons created in the processes described above. We acknowledge the effects that high energy Compton electrons have on the associated equations, but

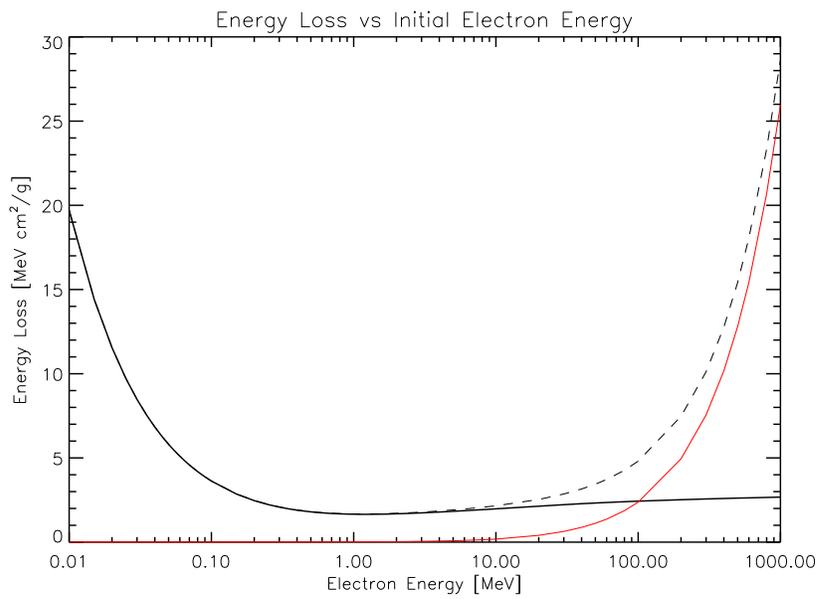


Figure 3: Energy loss versus fast electron energy, The loss mechanism are collisional (black line), radiative (red line), and total (dashed black line). There is no electric field included.

we ignore them for the purpose of this paper for several reasons. Mainly, we are testing the importance of the role of the photoelectrons in producing conduction electrons in a thunderstorm environment. This contribution is overlooked in runaway breakdown theory. These photoelectrons are also assumed to be monoenergetic at 1 keV.

In addition, we also make the assumption that background photoelectron flux is isotropic at 5 km altitude. Our evaluation is limited in this assumption but further experimental research in this area would greatly help to validate our model.

## 2 Analysis

In this section, we review the four, coupled, ordinary differential equations to be modeled using a variant of a 4th order Runge-Kutta method called the Rosenbrock method. Kap and Rentrop produced the first practical implementation of these methods, so these methods are also called Kaps-Rentrop methods.[10] These are used to solve a stiff set of equations, which refers to the case where there are very different scales of the independent variable on which the dependent variable is changing. An adaptive timestep is employed, which is calculated based on an error between test solutions of high and low order. This is an embedded approach. The rapidly changing timescale is represented physically in a lightning situation.

Our first equation describes the change in the electric field,  $E$ , as a function of time. We have

$$\frac{dE}{dt} = -4\pi c \left[ J_p + \frac{e}{c} N_e v_d \right] \quad (1)$$

where  $J_p$  is the primary current density, which contains "free" Compton and photoelectrons. More importantly, for this analysis,  $N_e$  is the number of low energy electrons created from the ionization of the air by the primary photoelectrons and  $v_d$  is the average drift velocity of the low energy electrons in the thunderstorm electric field. Note that the low energy electrons have their motion characterized by two components. One is a relatively slow drift in the applied electric field and the other is a random thermal motion that is characterized by the electron temperature. The primary electrons lose about 86 eV per ionization event, causing secondary electrons with sufficient energy to produce additional ionization. On average, one ion pair is created for each

34 eV of energy lost by the primary electrons.

We assume that  $J_p$  is uniformly distributed with respect to the volume of interest so equation (1) reduces to

$$\frac{dE}{dt} = -4\pi e N_e v_d. \quad (2)$$

Both  $N_e$  and  $v_d$  are functions of time so, to determine these variables, electron swarm theory is applied, which assigns a mean temperature to the electrons and relates their drift velocity to the electric field and various collision frequencies. [7] The solution of Equation 2 is an exponentially decaying electric field where the decay time constant is inversely proportional to the conductivity of the medium.

This includes, first, the rate of change of electron number density, which is a function of source terms and an attachment loss term defined by:

$$\frac{dN_e}{dt} = S_1(t) + C_1(U)N_e - \eta N_e \quad (3)$$

Here,  $S_1(t)$  [*electrons/cm<sup>3</sup>sec*] is the rate of production of swarm electrons,  $C_1(U)$  [*sec<sup>-1</sup>*] is the cascading rate, and  $\eta$  [*sec<sup>-1</sup>*] is the electron attachment rate.

$S_1(t)$  is defined as

$$S_1(t) = \sigma v N_{air} N_p \quad (4)$$

where  $N_{air}$  is the number density of air particles in the atmosphere at the height of the lightning strike (5km).  $N_p$  is the number density of primary photoelectrons which ranges from  $(5 - 8) \times 10^4 \text{cm}^{-3}$  [6] and is approximated as  $(6.5) \times 10^4 \text{cm}^{-3}$  here.  $\sigma v$  is the rate coefficient for the ionization of air by fast electrons. This is represented graphically in Figure 4 as a function of primary electron energy.

The primary electron energy in this analysis is that of the photoelectrons which is on the order of 1 keV. This means that we use a rate coefficient of about  $8 \times 10^{-8} \frac{\text{cm}^3}{\text{sec}}$ .

$C_1$  is the rate at which energetic swarm electrons create further electrons by ionization's and is just the drift velocity times the Townsend Coefficient. This implies that  $C_1(U)N_e$  is the rate at which low energy electrons produce further electrons by cascading. [7] Townsend impact ionization is the ionization in a swarm of low energy electrons. The characteristic energy is on the

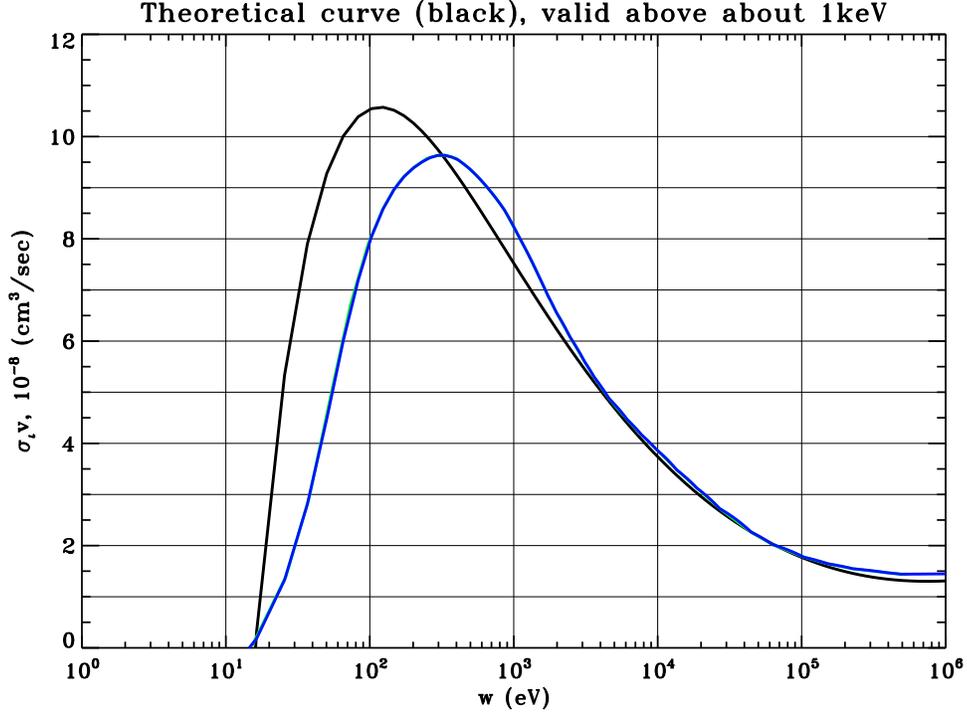


Figure 4: Rate coefficient for the creation of ion pairs,  $\sigma v$ , as a function of fast electron energy,  $w$  [eV]. [7]

order of 1 eV. As the electron temperature of the swarm increases, the electrons in the high energy tail of the distribution are more able to free more electrons from the surrounding air and the rate increases. The fit for the behavior of this phenomena with respect to energy can be seen below.

$$\frac{C_1}{N_{air}} = \frac{3.3 \times 10^{-14} U^{8.7}}{(1 + 5.87 U^{5.5})(1 + 1.29 \times 10^{-3})} + 3.26 \times 10^{-7} e^{-44.08/U} \quad (5)$$

This is also displayed pictorially in Figure 5.

The other important aspect of Equation 3 is the two and three body attachment rate of free electrons,  $\eta$ . This is dependent on the water vapor content which will can be relatively high (3-4 percent) inside a cloud at 5km. This attachment rate versus altitude can be seen in Figure 6.

The two and three body attachment is also dependent on the time varying electric field seen in Equation 2 and is taken into account in the stiff code.

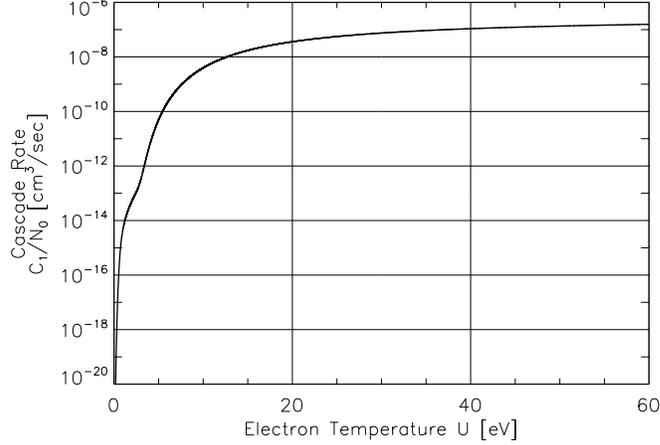


Figure 5: Cascading rate,  $C_1$ , as a function of Internal Swarm Energy,  $U$ . [7]

The differential energy change formula, derived from energy conservation, is

$$\frac{dU}{dt} = \frac{2}{3} \frac{e}{1.6 \times 10^{-12}} v_d E - v_w [U - U_0] + S_2(t) \quad (6)$$

where  $U_0$  [eV] is the ambient air temperature ( $\frac{1}{40}$  eV). While we are not yet taking into consideration the heating of the background medium, it can be seen that as the air temperature increases to the electron temperature, this loss term goes to zero.  $S_2$  [eV/sec] is the rate at which energy is added to the swarm.  $S_2$  is defined as

$$S_2(t) = \frac{2}{3} \left( \frac{\bar{w}}{N_e} S_1 - w_c C_1 \right) \quad (7)$$

$\bar{w}$  is the average energy of the ionization electrons produced by the photoelectrons, which is about 50 eV, and  $w_c$  is the ionization potential of air, which is about 14 eV.

The energy transfer collision frequency,  $\nu_w$ , comes into play when we consider Equation 6 which encompasses the energy term. This collision frequency is based on experimental data and is a function of the swarm internal energy  $U$ . The expression used here is [7]

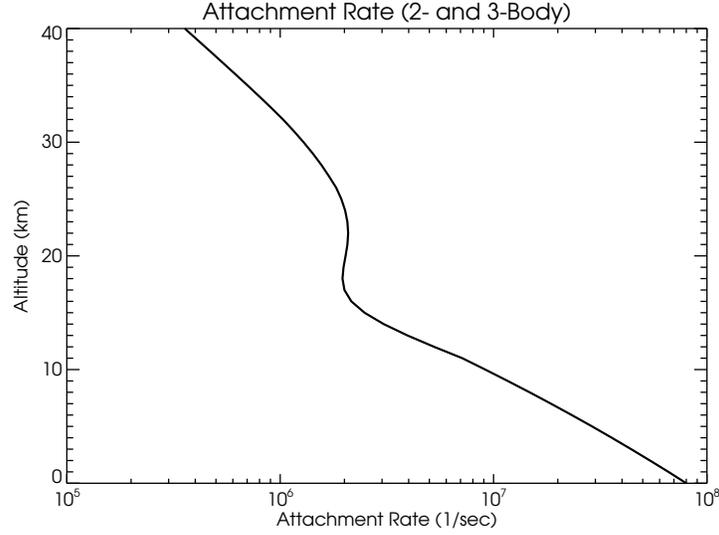


Figure 6: Attachment Rate as a function of altitude. The electric fields here is constant at 538 kV/m, which is the initial Electric field used in this analysis.

$$\frac{\nu_w}{N_0} = 1.0x10^{-11} + \frac{A_1U^{3.22}}{(1 + A_2U^{9.15})^{.307}} + \frac{B_1U^{5.22}}{(1 + B_2U^{4.74})^{.965}} \quad (8)$$

The next coupled PDE comes about by applying Newton's law relating rate of change of momentum to the applied force. This is the change in the swarm drift velocity in time.[7] The drift velocity will increase because of the applied electric field and will decrease in time as the electrons experience collisions. The collisions that reduce the drift velocity are characterized by a momentum transfer collision frequency.

$$\frac{dv_d}{dt} = \frac{e}{m}E - \nu_m(U)v_d \quad (9)$$

Here we have another collision frequency, the momentum transfer collision frequency,  $\nu_m$ , which is defined empirically as [7]

$$\frac{\nu_m}{N_0} = \frac{1.25x10^{-7}U^{.935}}{(1 + 1.26U^{1.67})^{.373}} \quad (10)$$

Both the energy and momentum transfer collision frequencies are plotted in Figures 7 and 8, respectively, as a function of U.

To summarize, the differential equations to be evaluated here are Equations 2, 3, 6, 9. Finding the parameters in these equations helps us to find

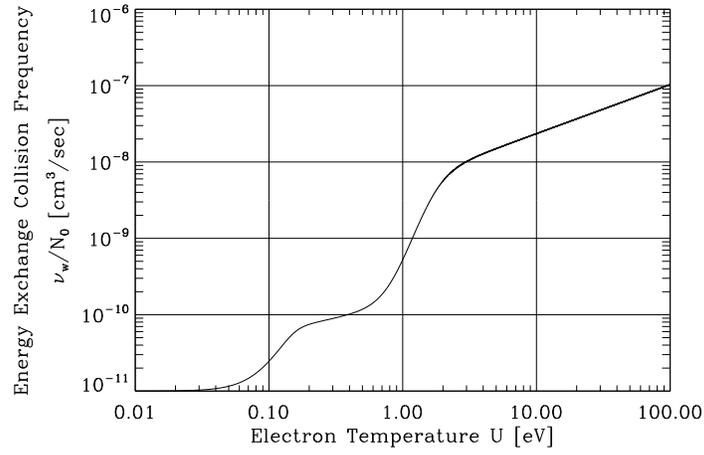


Figure 7: The energy transfer collision frequency,  $\nu_w$ , as a function of Internal Swarm Energy, U.[7]

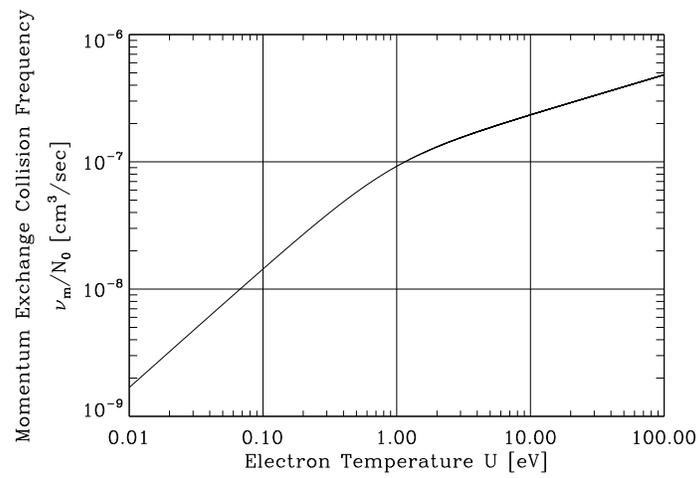


Figure 8: The momentum exchange collision frequency,  $\nu_m$ , as a function of Internal Swarm Energy, U.[7]

other values of interest including the conductivity and the number of positive and negative ions created due to the creation of conduction electrons. The number of positive ions is equal to the number of conduction electrons,  $N_e$ , and the number of negative ions is proportional to the attachment rate,  $\eta$ .

The conductivity [ $sec^{-1}$ ] is defined as

$$\sigma = e\mu_e N_e = \frac{e^2 N_e}{m_e \nu_m} \quad (11)$$

$\mu_e$  [ $\frac{cm^2}{statV*sec}$ ] is the electron mobility and is defined as

$$\mu_e = \frac{e}{m_e \nu_m} \quad (12)$$

Calculating these parameters can better help us to understand how electron swarm effects thunderstorms by showing how it changes the surrounding in this type of environment. The results from this analysis are summarized in the following sections.

### 3 Results

In this section, the results found by using the stiff equation solver[10] are presented graphically. First we have  $E$ ,  $N_e$ ,  $U$ , and  $v_d$ , as a function of time.

From this graph, it is clear that the drift velocity is heavily dependent on the electric field, and will decrease to zero. This is because the number of low density electrons is increasing by 6 orders of magnitude, which will eventually decrease the electric field to zero.

Also, the internal energy initially increases. Assuming that the photoelectrons are at a constant 1 keV, this means that the ratio of  $S_1$ , or primary photoelectrons, to  $N_e$  is higher than the energy loss rate of cascading electrons. Since the electric field in this region of increase is constant, this causes a local minimum in  $v_d$  where this local maximum is in  $U$ . Eventually  $N_e$  will overtake the  $S_1$  since, in our case,  $S_1$  is constant.

In order to validate these results, we compare them to the results of [Alietal, 1985] which can be seen in Figure10.

Our results are very close to the experimental results but are systematically underestimated, meaning some of our assumption must be inaccurate. When the electric field decays, the Ali plot is not applicable. It should be noted that we use an electric field of 14.47 [V/cm/Torr] or 538 [kV/m/atm]

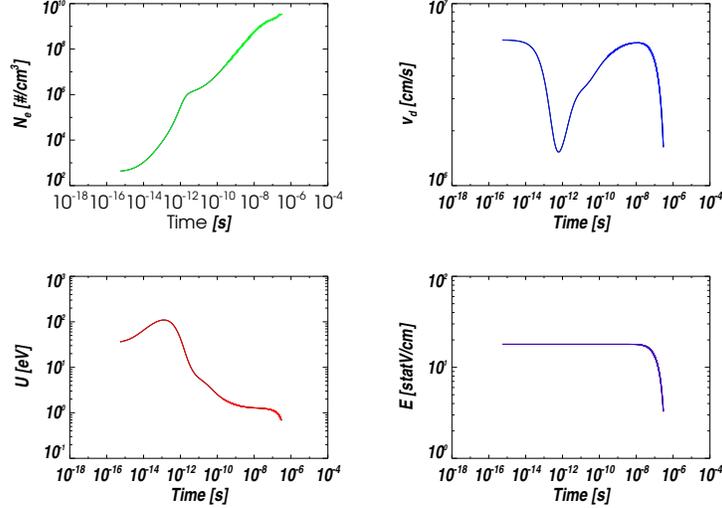


Figure 9: Results from solving the coupled Equations 2, 3, 6, 9. These variables are plotted with respect to time [seconds].

for our analysis of the other parameters, which was found previously by *Tierney et al.* [2005]. As stated above, it is important in our study to know the equilibrium times for the electric field. Table 1.

Another area of interest is the conductivity which is found by using Equation 11. This is shown in Figure 11.

The conductivity is dependent primarily on the  $N_e$  as can be seen by comparing Figure 11 with the  $N_e$  versus *time* plot in Figure 9. The value for  $\sigma$  does also depend on  $\nu_m$  but, since  $U$  has a much smaller change with respect to time compared to  $N_e$ ,  $\nu_m$  does not vary as greatly with time. Our

Electric Field Strength [statV/cm]	Equilibrium Time [ns]
8.97	0.25
17.946	0.1
35.89	0.08
53.837	0.05

Table 1: Equilibrium times [ns] for various electric field strengths found by implementing our stiff code

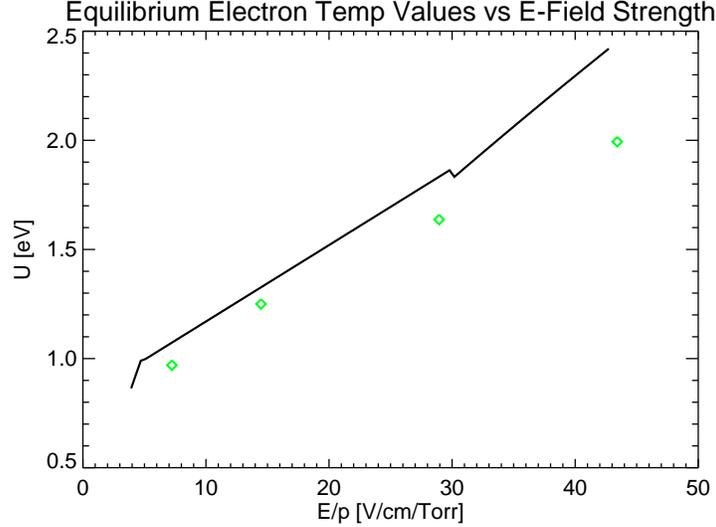


Figure 10: The internal energy of the swarm with respect to applied electric field.[1] The green diamonds represent the  $U$  that is obtained when various electric field strengths are used. The electric field in for the majority of this analysis is 14.47 [V/cm/Torr].

results indicate that the conductivity increases by eight orders of magnitude in a microsecond using our assumptions and initial conditions. Over most of this time period, the electric field is flat, therefore the conduction current,  $j$ , will depend on  $\sigma$  ( $j = \sigma E$ ) until the electric field starts its rapid decent.

The final parameter of concern is the number of negative ions,  $N^-$ , which are created by 2 and 3-body attachment. This can be seen in Figure 12.

This result is very interesting because it means that the surrounding air changes in composition in a thunderstorm environment. This allows the electric field to decrease during a lightning event since both the conductivity and the environment produce a better condition for lightning initiation.

Since the energy of the photoelectrons was assumed to be a constant 1 keV, it is of interest to see what happens to the parameters when we increase the energy to 10 keV and 100 keV. Figure 13 shows the effect on  $E$ ,  $N_e$ ,  $v_d$ , and  $U$ . Here, the values at 1 keV are represented by a green line, the values at 10 keV are represented by a red line, and the values at 100 keV are represented by a blue line.

Since increasing the energy decreases the ion pair creation rate coefficient by about one-half what is was at 1 keV (See Figure 4) then we would expect  $N_e$  to decrease by a factor of one-half at 10 keV. (See Equations 3 and 4) this is what we see when referring to Figure 13.

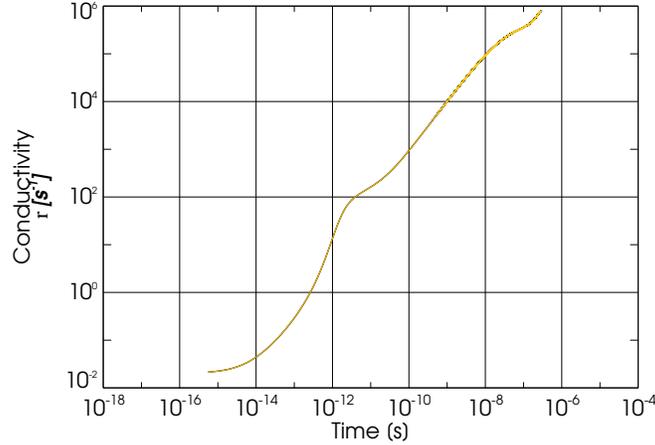


Figure 11: The conductivity versus time found from the results of solving the coupled Equations 2, 3, 6, 9.

Although the change of energy only seems to effect the results slightly, it is interesting to note that the electric field does start decaying at a later time, and that the conduction electrons maintain their internal energy and drift velocity longer for both cases.

In referring to Figures 14 and 15, we can see how changing the photoelectron energy affects the conductivity and number of negative ions created.

## 4 Conclusions

We have developed a model for investigating the time evolution of an electron swarm in the Earth's atmosphere. We found that for photoelectric and Townsend sources, the electron swarm will reach an equilibrium value on the order of  $.025 - 0.3 \times 10^{-9}$  seconds (decreasing with increasing electric field) for thunderstorm conditions. Our other finding is that the electron density can only attain a value of about  $10^{10}$  per  $cm^3$  before rapidly decreasing the ambient electric field. This is in agreement with the value found by [11].

The time lag of ionization is of interest for EMP theory since the decay of an EMP is governed by the conductivity of the background medium. Getting the correct EMP amplitude is important for our verification and validation

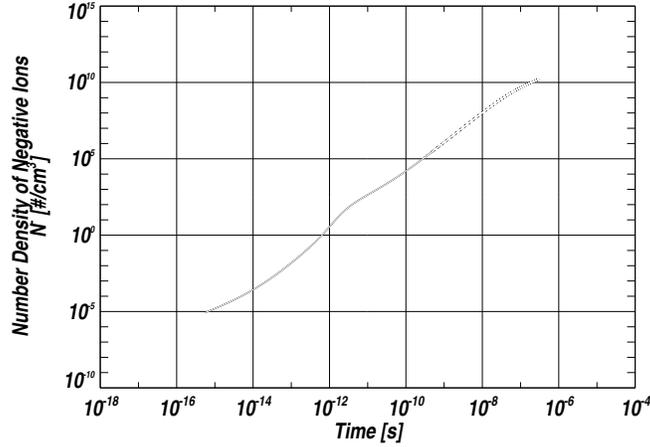


Figure 12: The number of negative ions produced versus time.

effort. The detailed characteristics of how swarms and electric fields evolve is of importance to lightning initiation. Here we have only considered an energetic source of electrons in a relatively cold background medium. If heating of the air takes place, such as in a return stroke process of lightning, the chemistry of the background becomes very complex and vitally important.

We have considered the creation of negative ions as a means of decreasing the ionization potential of air. Our experience with laboratory discharges shows that thunderstorm electric fields are always too low by a factor of about ten to breakdown air. One way to overcome this is to change the medium that the discharge is developing in. We investigated whether the creation of negative ions in air that is impacted by photoelectrons could result in a significant change of the characteristic ionization potential of air. The ionization potential for neutral air is about 13.6eV, while the ionization potential for a negative ion is on the order of 1eV. Our results show that the maximum number density of negative ions that can be produced is about  $10^{10}$  per  $cm^3$ . The density of air at 5 km is on the order of  $10^{19}$  per  $cm^3$ . Thus the ionization fraction is too low to have a significant effect on the characteristic ionization potential of air. The limiting factor is the conductivity of air that is produced. This conductivity will act to screen and reduce the local electric field. When the electric field is gone, runaway breakdown processes

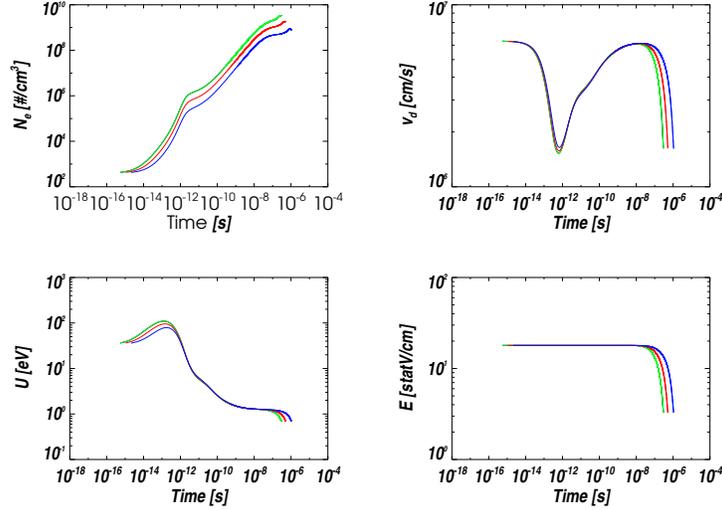


Figure 13: Results from solving the coupled Equations 2, 3, 6, 9. These variables are plotted with respect to time [seconds]. Here, the values at 1 keV are represented by a green line, the values at 10 keV are represented by a red line, and the values at 100 keV are represented by a blue line.

cannot occur. However, if a runaway avalanche was taking place and then the electric field was suddenly gone, all of the energy would be deposited quickly in the form of low energy electrons. Depending on the size of the region of deposition and the amount of energy contained in the avalanche, a vary large number of low energy electrons would be produced that could then attach to air and have a more significant effect.

The results gained from this analysis may have a substantial impact on how we view lightning initiation. It is clear that the background environment changes a lot, and the conductivity is still increasing even though the electric field is decreasing. Therefore, there is still a reason for lightning event when the electric field is below the breakeven electric field.

In order to improve this analysis, there would need the be improvements on the approximations made, such as that the photoelectrons are also assumed to have discrete energies and that background photoelectron flux is isotropic at 5 km altitude. There is going to be a spectrum of photoelectron energies since bremsstrahlung radiation has a spectrum for probabilities of energies, rather than just a discrete value. Also, the x rays creating the

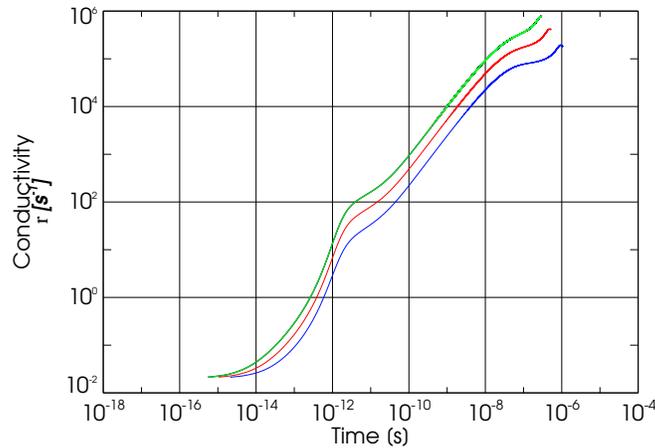


Figure 14: The conductivity versus time for various photoelectron energies. Here, the values at 1 keV are represented by a green line, the values at 10 keV are represented by a red line, and the values at 100 keV are represented by a blue line.

photoelectrons are of importance. Their presence would likely photo-detach any electrons that were able to attach to air. This could be handled in a more self-consistent manner than was possible in the ten-week duration of the workshop.

Overall, we have created an important capability for the EMP project. With further improvements to the swarm model that was developed we can better understand EMP and lightning phenomena.

## References

- [1] Ali, A.W., Intense and Short Pulse Electric Field (DC and Microwave) Air Breakdown Parameters, pp. 30, Naval Research Laboratory, Washington DC, 1986.
- [2] Berger, Martin J., and Stephen M. Seltzer. *TABLES OF ENERGY LOSSES AND RANGES OF ELECTRONS AND POSITRONS*. Rep. National Aeronautics and Space Administration, 1964.

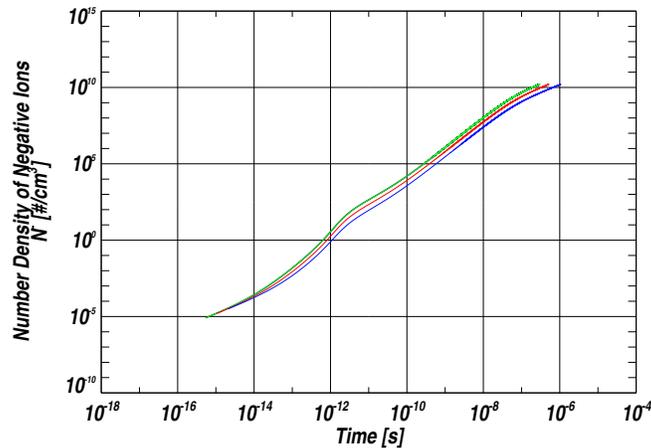


Figure 15: The number of negative ions produced versus time for various photoelectron energies. Here, the values at 1 keV are represented by a green line, the values at 10 keV are represented by a red line, and the values at 100 keV are represented by a blue line.

- [3] Eack, Kenneth B., William H. Beasley, W. David Rust, Thomas C. Marshall, and Maribeth Stolzenburg. "Initial Results from Simultaneous Observation of X Rays and Electric Fields in a Thunderstorm." *Journal of Geophysical Research* 101 (1996): 29637-9640.
- [4] Eack, K. B., Observations of x-rays produced by strong electric fields in thunderstorms, PhD Thesis, Norman, OK, 1997.
- [5] Gurevich, A. V., and G. M. Milikh. "Generation of X-rays Due to Multiple Runaway Breakdown inside Thunderclouds." *Physics Letters A* 262 (1999): 457-63. Elsevier Science B.V.
- [6] Gurevich, Aleksandr V., Kirill P. Zybin Runaway breakdown and electric discharges in thunderstorms *Phys.-Usp.* 44 1119, 2001.
- [7] Higgins, D.F., C.L. Longmire, and A.A. O'Deil. *A METHOD FOR ESTIMATING THE X-RAY PRODUCED ELECTROMAGNETIC PULSE OBSERVED IN THE SOURCE REGION OF A HIGH ALTITUDE BURST*. Rep. The Defense Nuclear Agency, Feb. 1973.

- [8] Marshall, Thomas C., Michael P. McCarthy, and W. David Rust. "Electric Field Magnitudes and lightning Initiation in Thunderstorms." *Journal of Geophysical Research* 100 (1995): 7097+.
- [9] McCarthy, M.P. and G.K. Parks (1985), Further observations of X-rays inside thunderstorms, *Geophys. Res. Lett*, 12(6), 393-396.
- [10] Press, William H., Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in Fortran: The Art of Scientific Computing*. 2nd ed. New York: Press Syndicate of the University of Cambridge, 1992.
- [11] Tierney, Heidi E., R. A. Roussel-Dupre, E.M.D. Symbalisty, and W.H. Beasley. "Radio Frequency Emissions From a Runaway Electron Avalanche Model Compared with Intense, Transient Signals from Thunderstorms." *Journal of Geophysical Research* 110 (2005): 7097+.

**Grain Boundary Formation  
Simulation**  
**(Cynthia Reichhardt, mentor)**

# Pattern Formation in a 2D Colloid System

Danielle McDermott and Jeffrey Amelang

Advisor Cynthia Reichhardt

August 15, 2012

## 1 Abstract

Using numerical simulations we study pattern formation in a collection of colloid particles interacting with a square pinning lattice. As the density of colloids is varied from 4 colloids per pin to 5, the ground state configuration changes from a triangular lattice to a square lattice. The intermediate fillings between 4 and 5 display a range of patterns: the lower density patterns have distinctive grain boundaries separating ordered regions. These grains form drops, stripes, and an unusual crystalline lattice at the 4.5 filling. As the density increases just beyond 4.5, it no longer makes sense to discuss individual grain boundaries and a quasi-ordered tiling occurs. As the density approaches 5, the ground states contain patches of both the 4-fold 5th filling and the quasi-ordered states.

For fillings below 4.5 the location and properties of the pins were altered with the intent of controlling the location of grain formation. By altering pinning sites, certain patterns/densities proved to be more stable than others. Furthermore, by selectively moving or removing pins, we were able to consistently control the location of drop and stripe formation, thus making these patterns relevant for device fabrication.

While the static anneals of the various densities already contain a rich variety of behavior, we put a potential on the system and observed the current response of bubbles, stripes, and patches to drive. Only a few of the fillings (aside from crystal 4, 4.5, and 5.0) had a depinning value for the individual colloids. The grain boundary depinning varied greatly in the bubble systems since it largely depends on the vertical/horizontal orientation of the grain. The striped systems showed a fairly consistent behavior of collectively traveling stripes for low drive values, followed by grain depinning/breaking. We witnessed hysteresis when lowering the current back to zero in the bubbles and stripes. The fillings above 4.5 showed a classic plastic regime and no hysteresis.

A primary focus of the summer project was to develop a code which could identify and characterize the grain boundaries in the system. The culmination of the summer work was approximately 5K lines of code which calculates the voronoi tessellation in 2D, finds grain boundaries, creates particle trails in driven systems, and creates energy maps. Furthermore the code analyzes the following grain boundary properties: identifies them as bubbles, stripes, or neither, finds the aspect ratio of the bubbles, calculates the stripe angle with respect to the x-axis, and tracks the grain boundary velocity.

## 2 Introduction

The study of pattern formation reveals universal tendencies of particles to form ordered and quasi-ordered states. We use a two-dimensional system of colloids interacting with a square pinning lattice to study the kinds of patterns that may be formed in many Condensed Matter systems of interest including monolayers of atoms or molecules on a crystal surface and superconducting vortices interacting with pinning sites.

Repulsively interacting colloid particles which are isolated in a 2D system will form into a triangular lattice in order to minimize their interaction energies. If these particles instead are placed in a region containing a rigid substrate lattice, they will be forced to arrange into commensurate and incommensurate patterns due to interactions with the lattice. The commensurability of the lattice depends on the density of colloids in the system.

The patterns formed by the colloids will depend strongly on the ratio of colloids to pins. Given a precise ratio of 4:1, the colloids neatly populate the pins and form a triangular lattice [1]. A ratio of 5:1 will yield a 4-fold pattern of squares rotated by  $27^\circ$  [1] with respect to the pinning lattice.

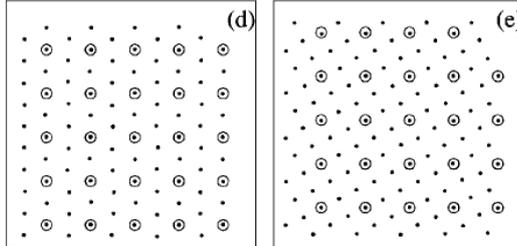


Figure 1: The 4th and 5th fillings of the vortex system studied by Reichhardt et. al. [1]

The intermediate fillings have not previously been modeled and the resulting square lattice and stripes resemble patterns seen by Curran et al in their SHPM imaging for vortex lattices in  $\text{Sr}_2\text{RuO}_4$  single crystals [2]. Many questions remain regarding the superconducting SRO system that could potentially be studied using complimentary systems such as 2D colloids.

### 3 Model and Simulation

We perform simulated annealing of a 2D system of colloids by numerically integrating the overdamped equation of motion:

$$\eta \mathbf{v}_i = \mathbf{f}_i = \mathbf{f}_i^{cc} + \mathbf{f}_i^{cp} + \mathbf{f}_i^T$$

The total force per unit length acting on colloid  $i$  is  $\mathbf{f}_i$ . The colloid-colloid interaction  $f_i^{cc} = -\sum_{j \neq i}^{N_c} \nabla V(r_{ij})$  is described by the Yukawa potential:

$$V(r_{ij}) = \frac{E_0}{r_{ij}} \exp(-\kappa r_{ij}) \quad (1)$$

Where  $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ , the scaling factor  $E_0$  is a screened Coulomb factor:  $Z^{*2}/(4\pi\epsilon a_0)$  with effective charge  $Z^*$  and a solvent dielectric constant of  $\epsilon$ . The exponential describes the soft cutoff of the Yukawa potential with  $1/\kappa$  as the screening length.

The pinning force  $\mathbf{f}_i^{cp}$  is described by:

$$V_p(r_{ik}^{(P)}) = -\left(\frac{f_p}{2r_p}\right)(r_{ik}^{(P)} - r_p)^2 \Theta(r_{ik}^{(P)} - r_p)$$

where  $\Theta$  is the Heaviside step function,  $f_p$  is the maximum pinning force,  $N_p$  is the number of pinning sites, and  $r_{ik} = |\mathbf{r}_i - \mathbf{r}_k^{(p)}|$ . Temperature is added as a stochastic term with properties:

$$\langle f_i^T(t) \rangle = 0$$

and

$$\langle f_i^T(t) f_j^T(t') \rangle = 2\eta k_B T \delta_{ij} \delta(t - t')$$

### 4 Colloid Ground States on a Square Pinning Lattice

The square pinning lattice was randomly populated with colloid particles for particle densities, or fillings, ranging from  $N_c/N_p = 4$  to 5. The annealed ground states are found by slowly cooling a system from a high

temperature. The 4th and 5th fillings are known from previous simulations [1] to form crystalline lattices. Because the 4th filling is triangular and the 5th filling is square, the intermediate fillings were expected to illustrate how one ordered lattice could reform into another.

The pinning lattice constant is  $a_0 = 3.6$  and system sizes are  $L \times L = 36 \times 36$  (100 pins),  $72 \times 72$  (400 pins), and  $108 \times 108$  (900 pins). Because the  $72 \times 72$  system was composed of  $N_p = 400$  pins ( $20a_0 \times 20a_0$ ), an ordered state with lattice parameter  $2a_0$  emerged for the filling ( $N_c/N_p$ )=4.5. This ordering was not observed in the other sizes since the crystal could not fit in the  $10 \times 10$  or  $30 \times 30$  lattices. All figures and data displayed will be drawn from the  $72 \times 72$  systems, but the general trends of this paper were observed in all three sizes.

## 4.1 Bubble Properties

As expected the colloid densities just above 4:1 form a triangular pattern with small imperfections. Once more colloids are added, more imperfections take the shape of drops: a degenerate triangular lattice rotated with respect to the original surrounded by a grain boundary.

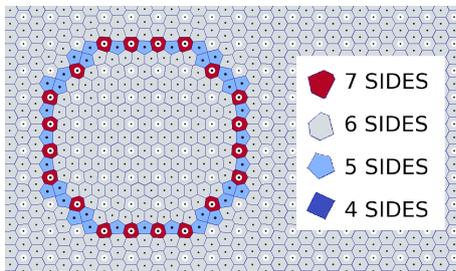


Figure 2: The bubble systems contain two regions of 6 sided colloids separated by a grain boundary, where one is rotated with respect to the other by  $90^\circ$ .

Above a filling of 4.03, it becomes most favorable for a single grain boundary to form, and the drop grows until it reaches an unstable size (which is 4.08 in the  $72 \times 72$  system). Like drops, the fillings between 4.08 and 4.095 feature triangular regions separated by grain boundaries. But unlike the drops, these systems prove unstable to modified pinning and applied currents.

## 4.2 Stripe Properties

Once a certain density is reached (4.095 in the  $72 \times 72$  system), the grain boundaries become stripes dividing ordered regions. As the density of the particles is increased, the stripes grow closer together until they are dense enough that identifying individual stripes and grain boundaries is no longer reasonable.

Unexpectedly, the 4.5 filling is a crystal state with lattice parameter  $2a_0$ . Unlike the 4th and 5th filling, the particles do not all see the same symmetries, but rather have Voronoi regions with sides of 4, 5, 6, and 7. These different faces form a tiled pattern with 2-fold symmetry.

## 4.3 Tiling and Patches

While the fillings surrounding 4.5 might be considered densely striped, it becomes difficult to consider the grain boundaries as distinct. Fillings above 4.7 are composed of patches of 4-fold coordinated colloids (as the 5th filling) and quasi-ordered pattern of 5, 6, and 7 sided particles (as the 4.6 filling).

Beyond 4.7 a clear transition toward the 5th filling begins. The quasi-ordered tiling competes with 4-fold square lattice in patches.

# 5 Characterizing the Phase transitions of the Ground states

A major goal of this work is to study a solid-solid phase transition. As the density of the colloids increases, the nature of the ground state of the colloids changes considerably. The Voronoi cells which do have

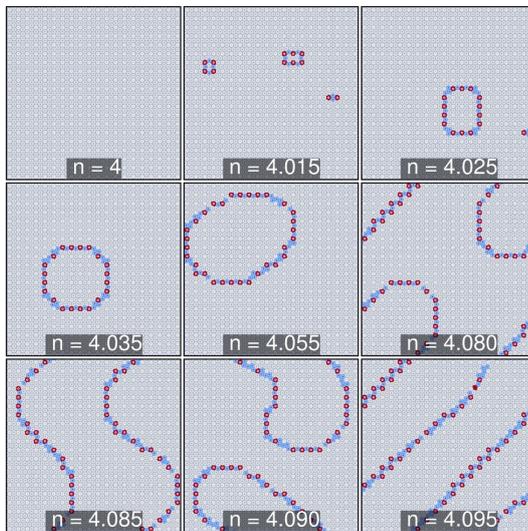


Figure 3: Below 4.1 the colloids form drops, one wide stripe, and final approach ordered stripes. The figure shows the voronoi tessellation of the particle positions, colored by number of sides: 5-sided polygons are blue, 6-sided polygons are white, and 7-sided polygons are red.

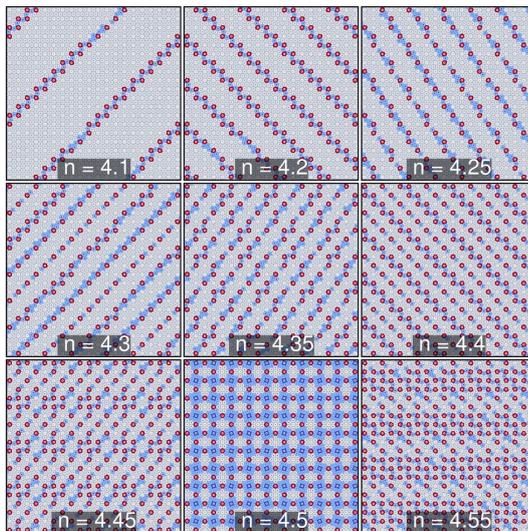


Figure 4: Above 4.1 the colloids form ordered stripes. At 4.5 they form an ordered crystal with lattice constant  $2a_0$ . Fillings about 4.5 begin to resemble tilings and then become patches of that tiling and the 5th ordered filling.

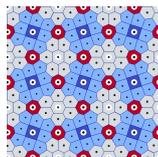


Figure 5: A 6x6 section of the ordered filling 4.5 which has a lattice constant of  $2a_0$

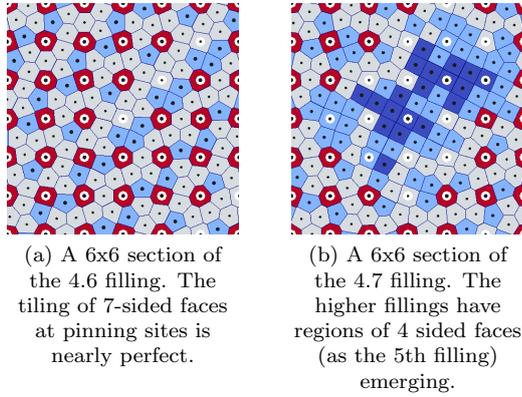


Figure 6: The transition from tiled to patches

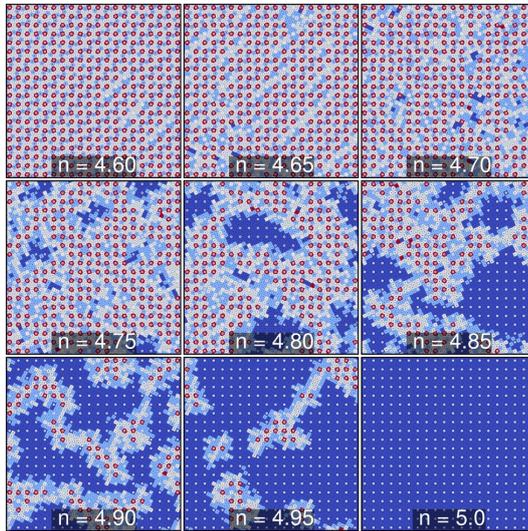


Figure 7: Fillings about 4.5 begin to resemble tilings and then become patches of that tiling and the 5th ordered filling.

6 sides are considered to be defected cells. Below the ordered 4.5 filling, the defected cells increase linearly. At the 4.5 filling, there is an unusually high number of 5-sided cells. Beyond the 4.5 filling, the systems still appear visually to contain stripes, but the tiling pattern of 7-sided cells at the pinning sites is clear. Once the density reaches 4.7 and beyond, the number of 4-sided cells increases rapidly.

## 6 System Deformation with Modified Pinning

By turning off pins in key locations, we were able to control the well-ordered stripe systems. It was clear that the systems less than 4.2 filling are less stable than the more densely striped systems. While the 4.3 and 4.4 fillings were able to adjust to meet the defected pin regions, the lesser fillings would deform and break in the presence of defects in the lattice.

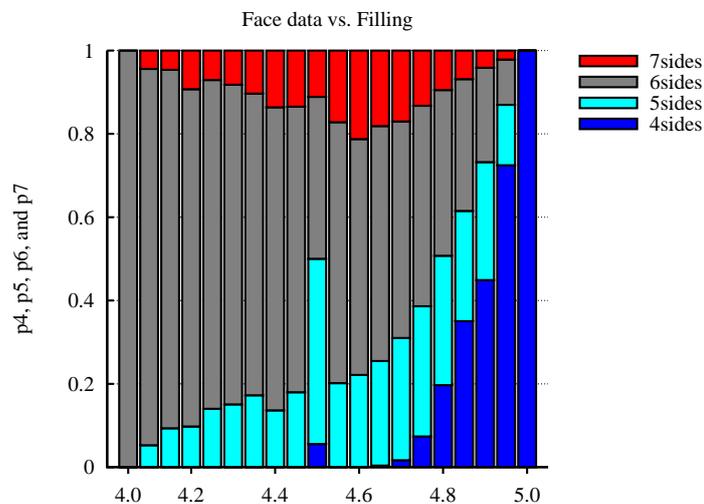


Figure 8: The Voronoi tessellation for each filling reveals the percentage of colloid containing cells with 4, 5, 6, and 7 sides.

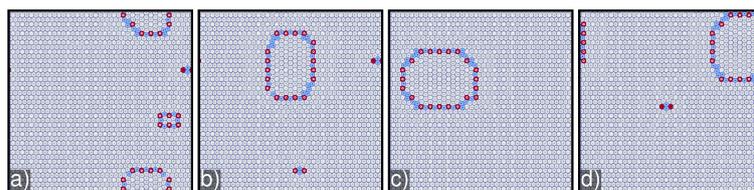


Figure 9: While turning off pins does not necessarily capture the bubble, it does sometimes work. (c) The bubble was pinning between two pins that had been turned off.

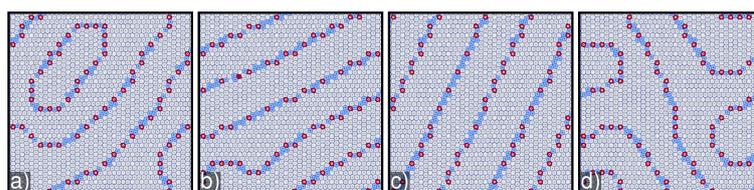


Figure 10: It is clear that the 4.15 filling breaks easily when exposed to deformations in the pinning lattice

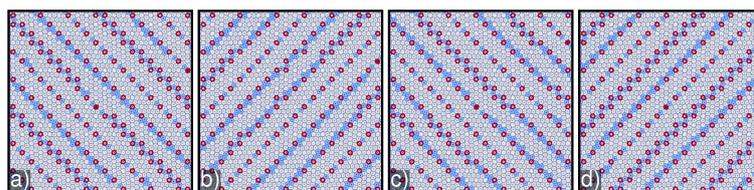


Figure 11: The 4.3 filling is clearly robust under deformed pinning. The direction of striping can be controlled by selecting how/where the pins are turned off

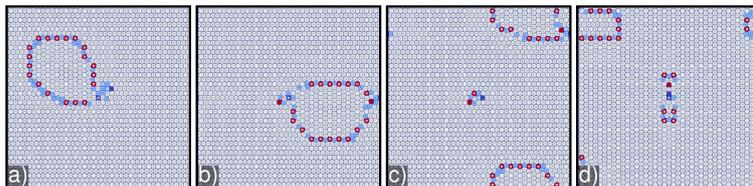


Figure 12: While the bubble can be more consistently pinned by moving pins, it increases the number of defects in the system

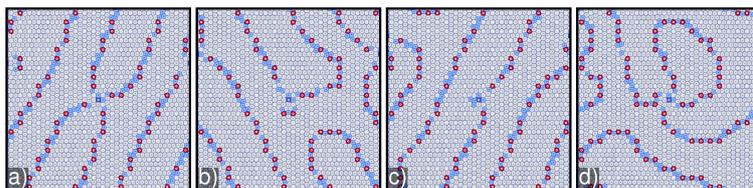


Figure 13: It is clear that the 4.15 filling breaks easily when exposed to deformations in the pinning lattice

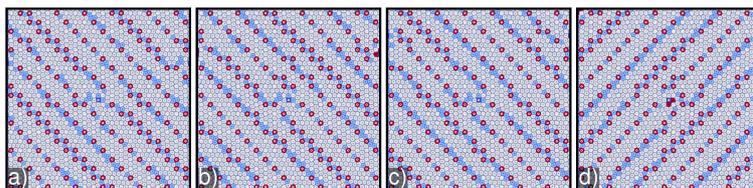


Figure 14: Moving pins in the 4.3 filling introduces considerably more defects than merely turning the pins off

## 7 Identifying and Characterizing Grain Boundaries

In figures 3 and 4 (which highlight the defected particles) it is apparent that there are grain boundaries, or structures of dislocated particles that separate different grains (illustrated in figure 15). These grains are groups of particles that have 6-sided voronoi polygons and are either horizontally or vertically oriented. For a given filling, whether the horizontal or vertical grain was more plentiful was observed to be random and is not considered significant.

The colored cells in figures 3 and 4 show particles that have a number of sides different than 6 and collectively demarcate the grains with grain boundaries. In this section, we'll first discuss how grain boundaries were programmatically detected, after which we'll look at their geometric and energetic properties.

### 7.1 Identifying Grain Boundaries

Particles whose corresponding voronoi polygon had a number of sides different than 6 were considered defected and were used to form the grain boundaries. From the pool of defected particles, one was chosen as the beginning of a new grain boundary. Successive grain boundary members were chosen as the closest particle to the end of the grain boundary that was within a tier of neighbors (second, third, etc) as well as the neighbors required to reach that particle. If there were defected particles remaining in the pool, another grain boundary was formed in the same way.

The presented algorithm performs well for closed grain boundaries, but can identify open grain boundaries as several pieces. To fix this, individual grain boundaries are then concatenated if their ends satisfy the proximity criterion. Lastly, the resultant grain boundaries are smoothed by switching the order of the inner

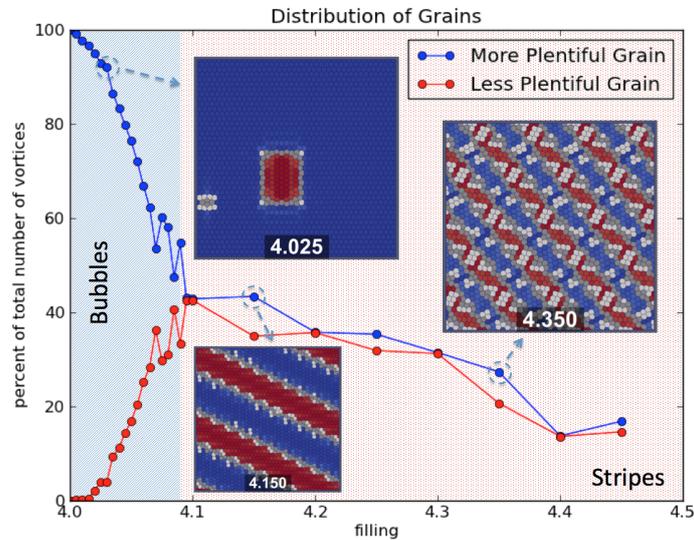


Figure 15: Distribution of six sided Grains versus Filling. Particles that aren't oriented strongly enough to be horizontal or vertical are colored white, and grain boundaries are colored gray.

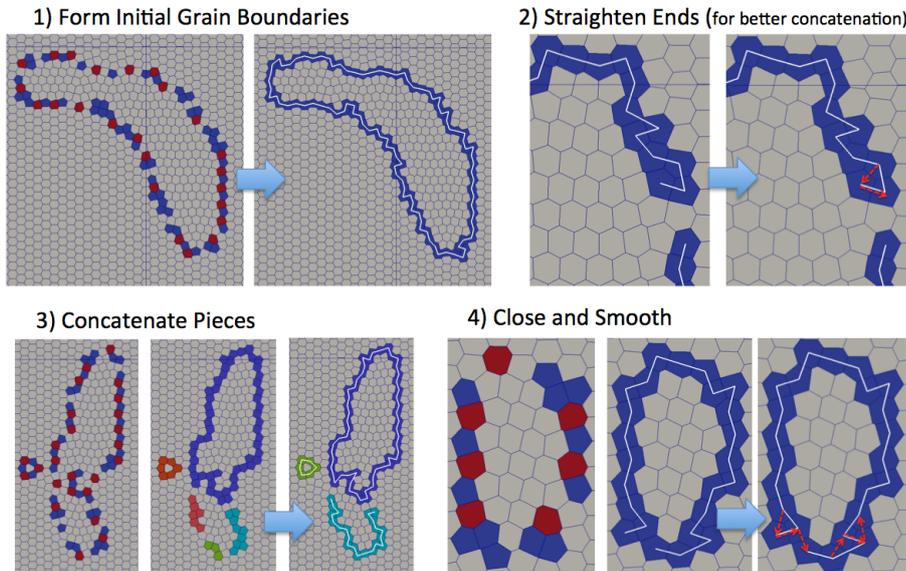


Figure 16: Illustration of the steps to Identify Grain Boundaries. Corrected paths are shown in red dashed arrows.

two grain boundary members of each set of four if the sum of inscribed angles can be reduced. Finally, applicable boundaries are closed. The process is illustrated in figure 16.

For closed grain boundaries, bubbles can be programmatically differentiated from stripes by traversing the grain boundary starting from one particle and determining in which “periodic copy” of the domain that particle is again found. For bubbles, that “periodic copy” is the domain in which the traversal was started.

Beyond a filling of 4.45, there are no longer discernable grain boundary structures and the algorithm is not applied.

## 7.2 Grain Boundary Geometric Properties

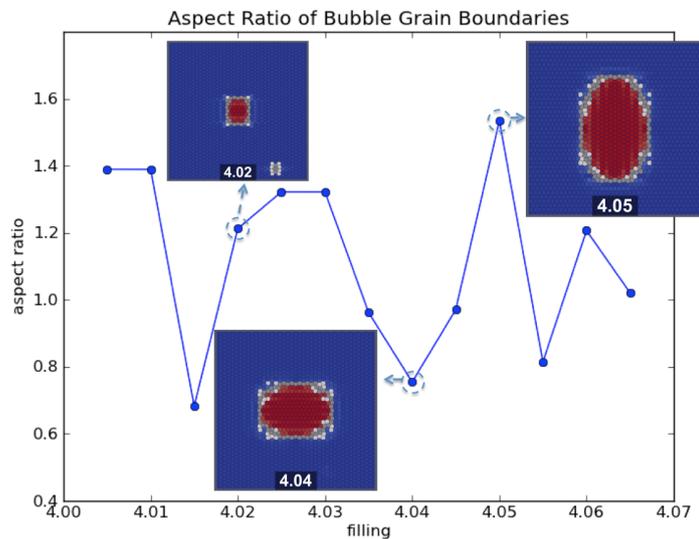


Figure 17: Bubble Aspect Ratios (coloring indicates grain orientation)

Unsurprisingly, the number of defected atoms and the total length of all grain boundaries grow linearly with filling for both bubbles and stripes. The aspect ratio of the bubble grain boundaries is shown in figure 17. Cynthia, is there anything to say about this? TODO: verify that the aspect ratio is the same after wiggling the parameters of the anneal.

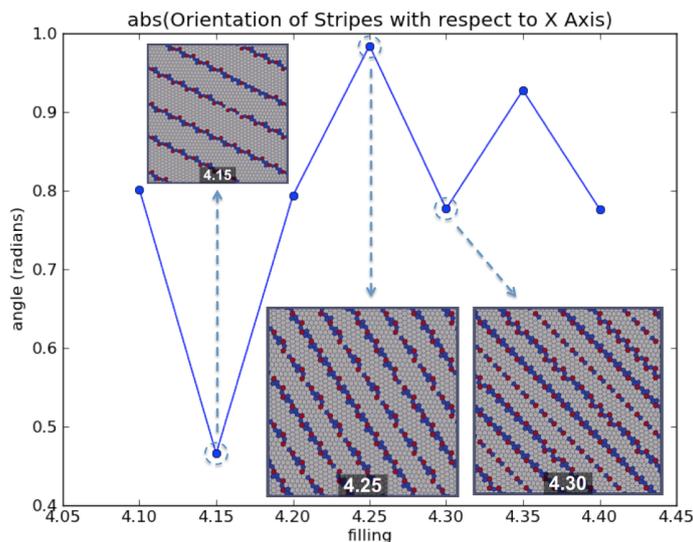


Figure 18: Stripe Orientations (The voronoi tessellation is colored by the number of sides of each cell)

All stripe systems were found to anneal to parallel lines, though for some fillings a few of the lines were joined by small tendrils. The angle that each system formed with respect to the x axis is shown in figure 18. No shallow or steep angles were observed in the annealed systems. Cynthia, is there anything to say about this? TODO: verify that the angle for a given filling is the same after wiggling the parameters of the anneal.

Maybe: The magnitude of the orientation was robust with regards to the parameters of the simulation, but the sign was observed to be random and is not considered significant.

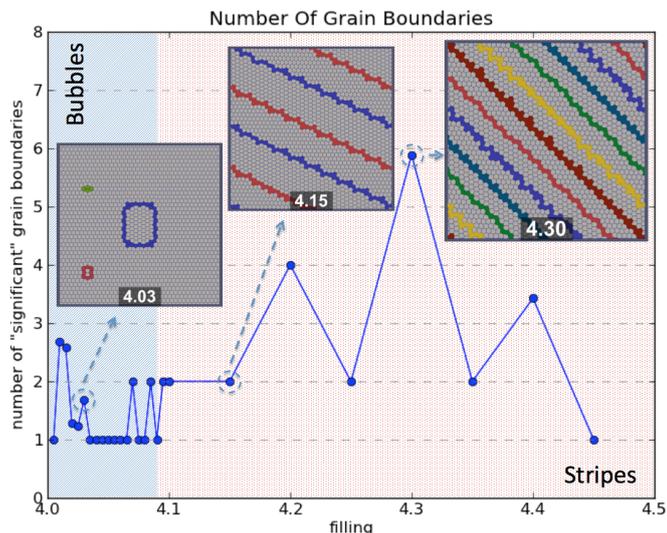


Figure 19: Number of Grain Boundaries (The voronoi tessellation shows the grain boundaries, each colored by their index)

The number of “significant” grain boundaries, which tries to ignore grain boundaries far smaller than the largest one at that filling, is shown in figure 19. At very low fillings, the defected particles more resemble dislocations than grains and they do not clump together, so there are several. Once there is a sufficient number of defects, a single grain boundary forms. When the bubble grows large enough to interact with itself across the periodic boundary conditions, a single stripe that winds the domain several times forms. Eventually, multiple independent stripes form instead of one large stripe.

### 7.3 Grain Boundary Energetic Properties

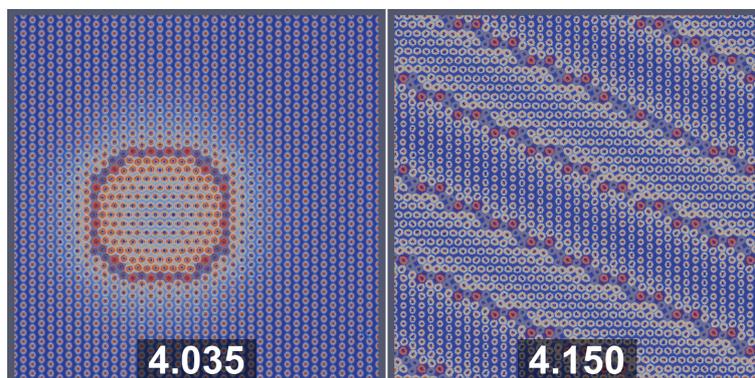


Figure 20: Energy Fields (The voronoi cells of defected particles are drawn semi-transparently)

Using the interaction potential (equation 1), the interaction energy with respect to all the particles at any given point can be computed, as shown by the energy fields in figure 20. By computing the energies only at particle locations, grain boundaries can be compared by their energetic structure and properties.

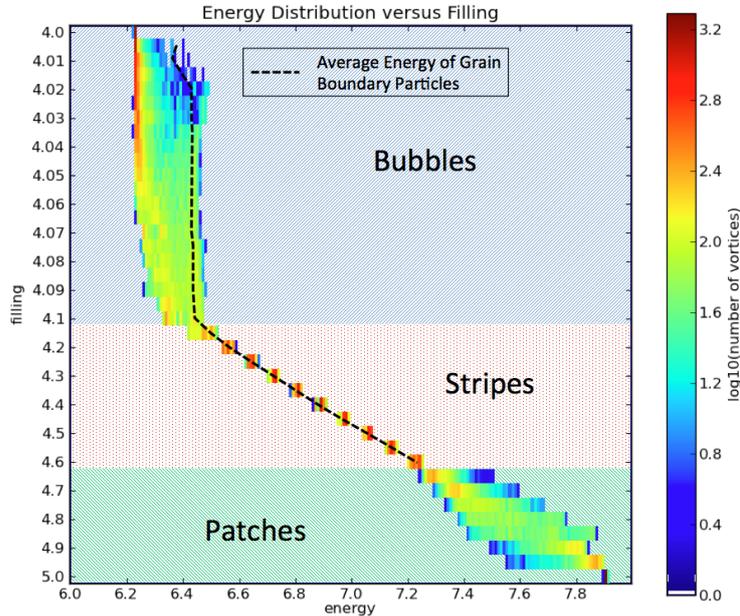


Figure 21: Energy Distribution (note the irregular spacing on the vertical axis and the fact that the filling goes up to 5)

Figure 21 shows the distribution of energy of the vortices at each filling. The total energy of the system of particles grows slightly faster than linearly with filling, which is unsurprising because the energy per vortex only transitions from 6.3 at a filling of 4 to 7.9 at a filling of 5.

The energy landscape is significantly different between the bubble, stripe, and patch phases. In the bubble phase, the span of energies seen is mostly the same for all bubble sizes. The average energy of particles in the grain boundary is constant, which shows that the grain boundaries are not becoming more energetic as the filling is increased, they just become longer. In the stripe phase, the energy span is different at each filling and the spread is much smaller. In the patch phase, the energy span becomes wide again similar to the bubble phase, but the extrema do change from one filling to the next.

The energetic structure of the grain boundaries also changes significantly in the bubble, stripe, and patch phases, as shown in figure 22. In the bubble phase (first panel), the grain boundary encircles the high energy particles and there is a clear segregation of high and low energy particles. In the beginning of the stripe phase (second panel), the grain boundary contains the majority of the high energy particles and the low energy particles form their own stripes as far from the high energy stripes as possible.

However, at stripes formed from higher fillings (third panel), the grain boundaries contain both the highest and the lowest energy particles, which form into dimers. The dimers can be observed beginning to form on the edges of the stripes in the 4.25 filling. In the patch phase (fourth panel), there are no longer grain boundaries and the regions of high and low energy particles grow in clumps.

## 8 Driven Systems

We apply a linearly increasing current  $\mathbf{f}_d = f_d \hat{x}$  to our annealed systems to measure the depinning threshold of the interstitial particles. We observed dynamical reordering in some systems. This reordering is quantified by measuring the current of the system:

$$v_x = \frac{1}{N_c} \sum_i^{N_c} \mathbf{v}_i \cdot \hat{x}$$

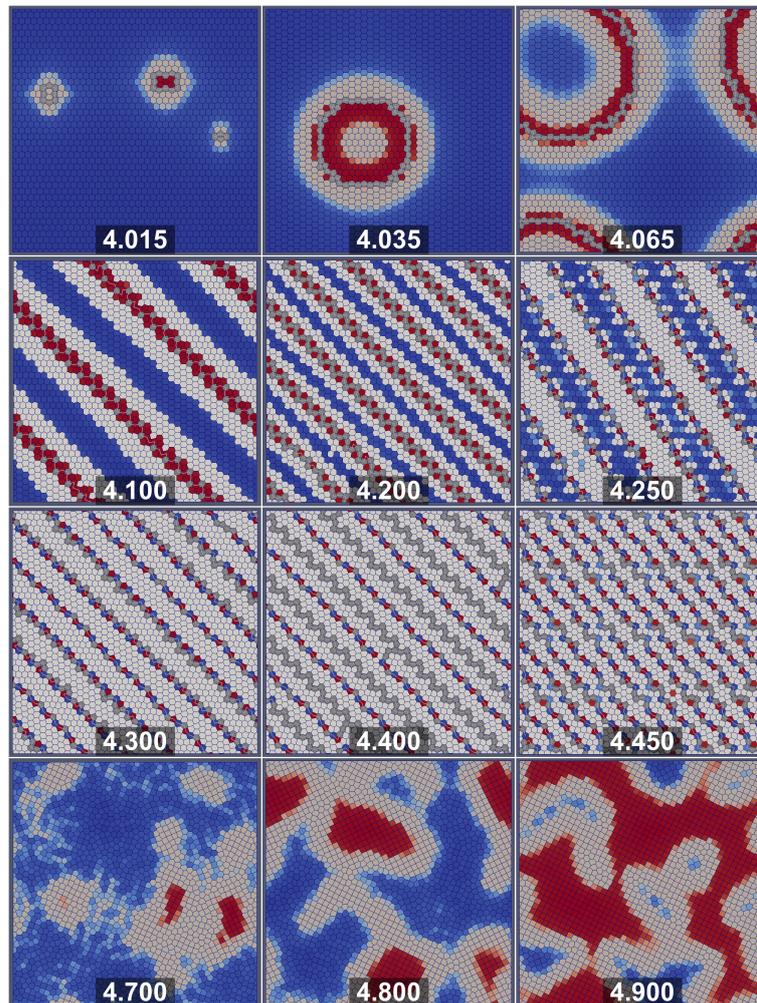


Figure 22: The energetic structure of grain boundaries at characteristic fillings. The voronoi cells are colored by the energy of the corresponding particle

## References

- [1] C. Reichhardt, C. J. Olson, and Franco Nori. Commensurate and incommensurate vortex states in superconductors with periodic pinning arrays. *Phys. Rev. B*, 57:7937–7943, Apr 1998.
- [2] P. J. Curran, V. V. Khotkevych, S. J. Bending, A. S. Gibbs, S. L. Lee, and A. P. Mackenzie. Vortex imaging and vortex lattice transitions in superconducting  $\text{Sr}_2\text{RuO}_4$  single crystals. *Phys. Rev. B*, 84:104507, Sep 2011.

**Turbulent Mixing**  
**(Rob Gore, mentor)**

# Turbulent modeling of plane mixing layer

Jeffrey Smith

Computational Physics Summer Workshop 2012

August 13, 2012

## Abstract

This paper employs the BHR-3 turbulence model to simulate constant and variable density plane mixing layer. The modeled form of the Reynolds stress, turbulent mass flux, and density-specific volume covariance are developed by Besnard et al. [4]. Motivated by limitations in modeling Rayleigh Taylor instability, a two length scale model developed by J. Schwartzkopf et al. are used to independently tune diffusion and dissipation rates. Coefficients for the pressure-strain model are proposed to match the experimental Reynold stress distribution of turbulent kinetic energy for a constant density plane mixing layer of air. The coefficients established for a constant density plane mixing layer are applied to a variable density plane mixing layer of nitrogen and helium, the large density difference across the mixing layer is shown to have little influence on the growth of the mixing layer.

## 1 Introduction

Modeling high Reynolds number turbulent flows by single point closure methods continues to receive interest by many disciplines of research, the Reynolds Averaged Navier Stokes (RANS) method has the potential to characterize complex turbulent flows [7]. The transport terms of the Reynolds stress equation are model with the approach of Daly and Harlow [5]. The pressure-strain terms are modeled with the LRR-IP model [1]. Higher order terms such as turbulent mass flux and density specific volume covariance are modeled according to Besnard 1992 [4].

A limitation of single point closure methods used with RANS equations is the lack of ability to capture all the length and time scale which are needed to fully characterize turbulent flows. Conventional methods use a single dissipation equation to model the average length and time scales thought

to best capture the scale needed to model transport and dissipation of the turbulent flow. An approach to overcome this limitation is to introduce an additional dissipation equation which applies another length and time scale to uniquely model both the transport and dissipation scale. Such multi-scale models have been proposed in the past (Hanjilac, Launder and Schiestel 1980). Motivated by the inability for a single scale model to predict Rayleigh Taylor Instability and homogeneous variable density decay, J. Schwarkopf et al. suggest the use of a two length/time scale model which aims to better characterize complex turbulent flow. In this work, one and two length/time scale models are applied to constant and variable density plane mixing layer with the motivation to predict: growth of mixing layer, self-similar Reynolds stress amplitudes, and distribution of Reynolds stresses which are referenced against the experimental data of Bell and Metha [2], Brown and Roshko [3], and Direct Numerical Simulation data of Rogers and Moser [8].

## 2 Governing equations

The governing equations of fluid flow (Navier-Stokes equations) are averaged and the result is a new set of differential equations that are solved for the mean flow field variables. The approach is to decompose the instantaneous flow field variables into the average and fluctuating components

$$p(x_i, t) = \overline{p(x_i)} + p'(x_i, t) \quad (1)$$

$$\rho(x_i, t) = \overline{\rho(x_i)} + \rho'(x_i, t) \quad (2)$$

$$u_i(x_i, t) = \overline{u_i(x_i)} + u'_i(x_i, t) \quad (3)$$

the bar indicates the time average and the prime indicates the magnitude of the fluctuating component at time  $t$ , by definition the time average of the fluctuating component  $\phi'$  is zero

$$\overline{\phi'} = \lim_{t \rightarrow +\infty} \frac{1}{t} \int_0^t \phi' dx = 0 \quad (4)$$

For variable density it is convenient to decompose the quantities into a mass weighted average and fluctuation

$$u = \tilde{u} + u''; \quad \tilde{u} = \frac{\overline{\rho u}}{\bar{\rho}} \quad (5)$$

then

$$\tilde{u} = \bar{u} + \frac{\overline{\rho' u'}}{\bar{\rho}} \quad (6)$$

and

$$u'' = u' - \frac{\overline{\rho' u'}}{\bar{\rho}} \quad (7)$$

from these definitions

$$\overline{u''} \neq 0 \quad (8)$$

$$\overline{\rho u''} = 0 \quad (9)$$

$\tilde{u}$  known as Favre averaging. Equations 1, 2, and 5 are substituted in the continuity and momentum equation and the equations themselves are averages. The results is a new set of equations where the variables to be solved are the mean quantities. The averaged equations are

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_k}{\partial x_k} = 0 \quad (10)$$

$$\frac{\partial \bar{\rho} \tilde{u}_i}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_i \tilde{u}_k}{\partial x_k} = -\frac{\partial \bar{P}}{\partial x_i} + \frac{\partial \bar{\tau}_{ij}}{\partial x_j} - \frac{\partial}{\partial x_j} \overline{\rho u_i'' u_j''} \quad (11)$$

the last term in the averaged momentum equation is the Reynolds stress tensor which introduces six additional equations. The governing equation are now unclosed and therefore, the Reynold stress tensor requires a modeling approach.

## 2.1 Lenght Scale

The length scale is defined by the turbulent kinetic energy which is half the trace of the Reynold stress tensor  $K = 1/2 R_{ii}$ , and a modified dissipation equation originally proposed by Hanjalic and Lauder [6]

$$\begin{aligned} \frac{\partial \epsilon}{\partial t} + \frac{\partial \tilde{u}_k \epsilon}{\partial x_k} = & -C_{\epsilon 1} \frac{\epsilon}{K} \tilde{R}_{ij} \frac{\partial \tilde{u}_i}{\partial x_j} - C_{\epsilon 2} \frac{\epsilon^2}{K} + \frac{\partial}{\partial x_k} \left( C_{\epsilon} \frac{K}{\epsilon} \tilde{R}_{ij} \frac{\partial \epsilon}{\partial x_j} \right) \\ & + C_{\epsilon 3} \frac{\epsilon}{K} a_j \frac{\partial \bar{P}}{\partial x_j} \end{aligned} \quad (12)$$

the last term of equation 12 is added for variable density flows [9], the length scale  $S$  is defined as

$$S = \frac{K^{\frac{3}{2}}}{\epsilon} \quad (13)$$

and time scale

$$\tau = \frac{S}{\sqrt{K}} \quad (14)$$

Two length scales has the ability to capture the dominate scales of diffusion and dissipation, independently. Two sets of coefficients are needed for each length scale.

$$\begin{aligned} \frac{\partial(\bar{\rho}S_{diff})}{\partial t} + \frac{\partial\tilde{u}_k S}{\partial x_k} = & \underbrace{-\frac{S_{diff}}{K} \left(\frac{3}{2} - C_1\right) \bar{\rho}\tilde{R}_{ij}\tilde{u}_{i,j} - \left(\frac{3}{2} - C_2\right) \bar{\rho}\sqrt{K}}_{production} \\ & + \underbrace{\frac{S_{diff}}{K} \left(\frac{3}{2} - C_4\right) a_j \bar{P}_{,j} - C_3 \bar{\rho} S_{diff} \tilde{u}_{j,j}}_{production} + \underbrace{C_s \left(\frac{S_{diff}}{\sqrt{K}} \bar{\rho} \tilde{R}_{kn} S_{diff,n}\right)_k}_{Diffusion} \end{aligned} \quad (15)$$

$$\tau_{diff} = \frac{S_{diff}}{\sqrt{K}} \quad (16)$$

$$\begin{aligned} \frac{\partial(\bar{\rho}S_{diss})}{\partial t} + \frac{\partial\tilde{u}_k S}{\partial x_k} = & \underbrace{-\frac{S_{diss}}{K} \left(\frac{3}{2} - C_{1v}\right) \bar{\rho}\tilde{R}_{ij}\tilde{u}_{i,j} - \left(\frac{3}{2} - C_{2v}\right) \bar{\rho}\sqrt{K}}_{production} \\ & + \underbrace{\frac{S_{diss}}{K} \left(\frac{3}{2} - C_v A\right) a_j \bar{P}_{,j} - C_{3v} \bar{\rho} S_{diss} \tilde{u}_{j,j}}_{production} + \underbrace{C_s \left(\frac{S_{diss}}{\sqrt{K}} \bar{\rho} \tilde{R}_{kn} S_{diss,n}\right)_k}_{Dissipation} \end{aligned} \quad (17)$$

$$\tau_{diss} = \frac{S_{diss}}{\sqrt{K}} \quad (18)$$

## 2.2 Reynolds stress tranport equation

The set of differential equations for the transport of the Reynolds stress is [9]

$$\begin{aligned}
\frac{\partial}{\partial t}(\bar{\rho}\tilde{R}_{ij}) + \frac{\partial}{\partial x_k}(\bar{\rho}\tilde{u}_k\tilde{R}_{ij}) &= \underbrace{a_i \frac{\partial \bar{P}}{\partial x_j} + a_j \frac{\partial \bar{P}}{\partial x_i} - \bar{\rho}\tilde{R}_{ik} \frac{\partial \tilde{u}_j}{\partial x_k} - \bar{\rho}\tilde{R}_{jk} \frac{\partial \tilde{u}_i}{\partial x_k}}_{\text{Production}} - a_i \frac{\partial \tilde{\tau}_{jk}}{\partial x_k} - a_j \frac{\partial \tilde{\tau}_{ik}}{\partial x_k} \\
&\quad - \underbrace{\frac{\partial}{\partial x_k} \overline{\rho u_i'' u_j'' u_k''} + \frac{\partial}{\partial x_k} \left( \overline{u_i'' \tau_{jk}''} + \overline{u_j'' \tau_{ik}''} \right)}_{\text{Transport}} - \frac{\partial}{\partial x_j} \overline{u_i'' P'} - \frac{\partial}{\partial x_i} \overline{u_j'' P'} \\
&\quad + \underbrace{\bar{P}' \left( \frac{\partial u_i''}{\partial x_j} + \frac{\partial u_j''}{\partial x_i} \right)}_{\text{PressureStrain}} - \underbrace{\overline{\tau_{jk}''} \frac{\partial u_i''}{\partial x_k} - \overline{\tau_{ik}''} \frac{\partial u_j''}{\partial x_k}}_{\text{Dissipation}} \quad (19)
\end{aligned}$$

The mean variable to be solved are  $\tilde{u}$ ,  $\bar{\rho}$ ,  $\bar{P}$ , and  $\tilde{R}_{ij}$ . The terms that need modeled are transport, pressure strain, and dissipation. The mean viscous stress tensor terms are discarded at high Reynold numbers. The transport terms are modeled as originally put forth by Daly and Harlow 1970 [5]. The dissipation and pressure-strain terms are modeled according to Launder 1975 [1].

The modeled form of the exact Reynolds stress transport equation is [9]

$$\begin{aligned}
\frac{\partial(\bar{\rho}\tilde{R}_{ij})}{\partial t} + \frac{\partial}{\partial x_k}(\bar{\rho}\tilde{u}_k\tilde{R}_{ij}) &= \underbrace{a_i \frac{\partial \bar{P}}{\partial x_j} + a_j \frac{\partial \bar{P}}{\partial x_i} - \bar{\rho}\tilde{R}_{ik} \frac{\partial \tilde{u}_j}{\partial x_k} - \bar{\rho}\tilde{R}_{jk} \frac{\partial \tilde{u}_i}{\partial x_k}}_{\text{Production}} \\
&\quad + \underbrace{C_{r3} \frac{\partial}{\partial x_k} \left( \tau_{diff} \bar{\rho} \tilde{R}_{km} \frac{\partial \tilde{R}_{ij}}{\partial x_m} \right)}_{\text{Transport}} - \underbrace{C_{r4} \bar{\rho} \frac{1}{\tau_{diss}} \left( \tilde{R}_{ij} - 1/3 \tilde{R}_{kk} \delta_{ij} \right)}_{\text{Slow Return to Isotropy}} \quad (20) \\
&\quad + \underbrace{\bar{\rho} C_{r2} \left( \tilde{R}_{ik} \frac{\partial \tilde{u}_j}{\partial x_k} + \tilde{R}_{jk} \frac{\partial \tilde{u}_i}{\partial x_k} \right) - C_{r2} 2/3 \bar{\rho} \tilde{R}_{mk} \frac{\partial \tilde{u}_m}{\partial x_k} \delta_{ij} - C_{r1} \left( a_i \frac{\partial \bar{P}}{\partial x_j} + a_j \frac{\partial \bar{P}}{\partial x_i} \right)}_{\text{Rapid Return to Isotropy}} \\
&\quad + \underbrace{C_{r1} 2/3 a_k \frac{\partial \bar{P}}{\partial x_k} \delta_{ij}}_{\text{Rapid Return to Isotropy}} - \underbrace{\bar{\rho} 2/3 \frac{K}{\tau_{diss}} \delta_{ij}}_{\text{Dissipation}}
\end{aligned}$$

### 2.3 Turbulent mass flux

The term  $\overline{\rho' u'}$  known as turbulent mass flux is nonzero and appears in the averaged governing equations. The transport equation for  $a_i = \frac{\overline{\rho' u'}}{\bar{\rho}}$  can be

derived from the Navier-Stokes equations the model for is [4] [9]

$$\begin{aligned}
\frac{\partial \bar{\rho} a_i}{\partial t} + \frac{\partial}{\partial x_k} (\bar{\rho} \tilde{u}_k a_i) &= \frac{\partial}{\partial x_i} b(1 - C_{ap}) \bar{P} - \frac{\partial}{\partial x_k} (1 - C_{ar}) \tilde{R}_{ik} \bar{\rho} \\
- \frac{\partial}{\partial x_k} (1 - C_{au}) \bar{\rho} a_k \tilde{u}_i + - \frac{\partial}{\partial x_k} \bar{\rho} a_k a_i + C_a \frac{\partial}{\partial x_k} \bar{\rho} \tau_{diff} \tilde{R}_{kn} & \\
- C_{a1} \bar{\rho} \frac{1}{\tau_{diss}} b \frac{\partial a_i}{\partial x_n} &
\end{aligned} \tag{21}$$

## 2.4 Variable-density volume covariance

## 3 Results and Discussion

Instability of a plane mixing layer was simulated on xRage. A temporally evolving shear layer was established by two flows in parallel with equal velocities and opposite direction.

The ideal Reynolds stress transport model coefficients are found empirical based on simplified experiments which aim to isolate each terms and determine their contribution. For single length scale constant density plane mixing layer, the normal Reynolds stress components in the direction of flow can be reduced to

$$\begin{aligned}
\frac{\partial (\bar{\rho} \tilde{R}_{11})}{\partial t} + \frac{\partial}{\partial x_k} (\bar{\rho} \tilde{u}_k \tilde{R}_{11}) &= \bar{\rho} (C_{r2} - 1) \left( \tilde{R}_{1k} \frac{\partial \tilde{u}_1}{\partial x_k} - \tilde{R}_{1k} \frac{\partial \tilde{u}_1}{\partial x_k} \right) \\
+ C_{r3} \frac{\partial}{\partial x_k} \left( \tau \bar{\rho} \tilde{R}_{1m} \frac{\partial \tilde{R}_{11}}{\partial x_1} \right) - C_{r2} \frac{2}{3} \bar{\rho} \tilde{R}_{1k} \frac{\partial \tilde{u}_m}{\partial x_k} & \\
- C_{r4} \bar{\rho} \frac{1}{\tau} \tilde{R}_{11} - \bar{\rho} \frac{2}{3} \frac{K}{\tau} &
\end{aligned} \tag{22}$$

with length scale

$$\begin{aligned}
\frac{\partial (\bar{\rho} S)}{\partial t} + \frac{\partial \tilde{u}_k S}{\partial x_k} &= - \frac{S}{K} \left( \frac{3}{2} - C_1 \right) \bar{\rho} \tilde{R}_{ij} \frac{\partial \tilde{u}_i}{\partial x_j} - \left( \frac{3}{2} - C_2 \right) \bar{\rho} \sqrt{K} \\
- C_3 \bar{\rho} S \frac{\partial \tilde{u}_j}{\partial x_j} + C_s \left( \frac{S}{\sqrt{K}} \bar{\rho} \tilde{R}_{kn} S_{,n} \right)_k &
\end{aligned} \tag{23}$$

and time scale

$$\tau = \frac{S}{\sqrt{K}}$$

the coefficients from the pressure-strain model  $C_{r2}$  and  $C_{r4}$  indirectly affect the production and dissipation, respectively. Since our first intent

is to investigate the growth rate of the mixing layer and production of the normal Reynold stresses the coefficient set by the (LRR-IP) model of  $C_{r2} = 0.6$ ,  $C_{r4} = 1.8$  are used. As seen from the equation above the other contributors to the normal Reynold Stress are  $C_{r3}$  and the time scale, the time scale is determined by the modeled form of the dissipation equation or length scale equation. The suggested coefficients of the dissipation equation are  $C_1 = 1.44$ ,  $C_2 = 1.92$  and  $C_3 = 0.15$  (Launder 1990). More recent experiments of homogeneous decaying turbulence sets the value of  $C_2$  closer to 1.77 ( Mohammed and Laur) this value is used throughout this study.

### 3.1 Growth of mixing layer

One important characterization in free shear is the diffusion of momentum perpendicular to the free stream flow; the boundary of this momentum layer is defined in several ways by other investigators.

In this work the characteristic width of the mixing layer in a temporally evolving shear flow is defined by  $\pm 10\%$  of the mean velocity profile

$$\bar{U}(t, y(t)_l) = U_L + 0.1\Delta U \quad (24)$$

$$\bar{U}(t, y(t)_h) = U_h - 0.1\Delta U \quad (25)$$

$$\delta(t) = y(t)_h - y(t)_l \quad (26)$$

where  $U_l$  and  $U_h$  are the low and high flow velocity,  $\Delta U = U_h - U_l$ , and  $y$  is the cross-stream spatial coordinate.

It is desired for universality to have the center of the mixing domain referenced to zero, regardless of simulation setup or method of physical measurements. The abscissas of the mean velocity profile is shifted by  $\bar{y} = \frac{1}{2}(y(t)_h + y(t)_l)$  in the positive direction. The spatial cross-stream coordinate of the mean velocity profiles are non-dimensionalized by  $\delta(t)$ .

$$\eta = \frac{y - \bar{y}}{\delta} \quad (27)$$

The growth rate of the self-similar temporally evolving mixing layer is found from [9].

$$\beta = \frac{1}{\Delta U} \frac{\Delta \delta(t)}{\Delta t} \quad (28)$$

This simplified model with the coefficients described above is compared to the experimental results of Bell and Metha [2] and DNS data of Rogers and Mosers [8]. Profiles of the mean velocity and Reynold stresses  $R_{ij}/\Delta U^2$  are

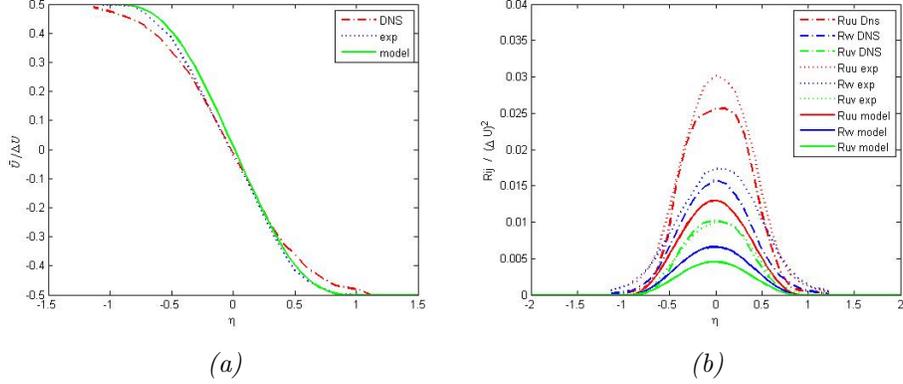


Figure 1: (a) Scaled mean velocity profile compared to experimental results of Bell and Metha and DNS data of Rogers and Mosers. (b) Scaled Reynolds stresses  $R_{uu}$ ,  $R_{vv}$ , and  $R_{uv}$

plotted against  $\eta$  in figure 1a and 1b. While the scaled velocity profile has developed to agree with experimental results, the Reynold stresses are clearly being under-predicted, additionally, the growth rate of the mixing layer for figure 1a and 1b is  $\beta = 0.025$  which is well below the the experimental growth rate  $\beta = 0.065$  and DNS growth rates  $\beta = 0.062$ .

A principal observation of the plane mixing layer is a self-similar state is obtained after an initial transient time or distance. Figure 2b shows the evolution the  $R_{uu}$ ; shortly after  $t = 0.0002$  seconds  $R_{uu}$  becomes nearly constant, the sharp rise in  $R_{uu}$  at  $t > 0.001$  is due to the mixing layer encountering the width of the simulation domain. The evolution of the Reynolds stress in the stream-wise direction has reached a self similar state at  $2 \times 10^{-4} < t < 10 \times 10^{-4}$  seconds. The growth rate of the mixing layer is found during the steady state.

The value of  $C_1$  in equation 23 is lowered to 0.7 to decrease the dissipation, or alternatively, increase the growth rate. The scaled velocity profile and Reynolds stresses are shown in figure 3a and 3b. The growth rate has increased to 0.067, comparable to experiment and the scaled Reynolds stresses obtained at self-similarity closely match experiment.

Increasing the length scale to compensate for under-predicated growth rates has inadvertently affected the transport of turbulent kinetic energy as seen by the widened profile of the Reynold stresses in figure 3b. The transport term in equation 20 is directly proportional to the time scale by lowering  $C_1$  to decrease dissipation the transport term in equation  $C_{r3} \frac{\partial}{\partial x_k} \left( \tau \bar{\rho} \tilde{R}_{km} \frac{\partial \tilde{R}_{ij}}{\partial x_m} \right)$

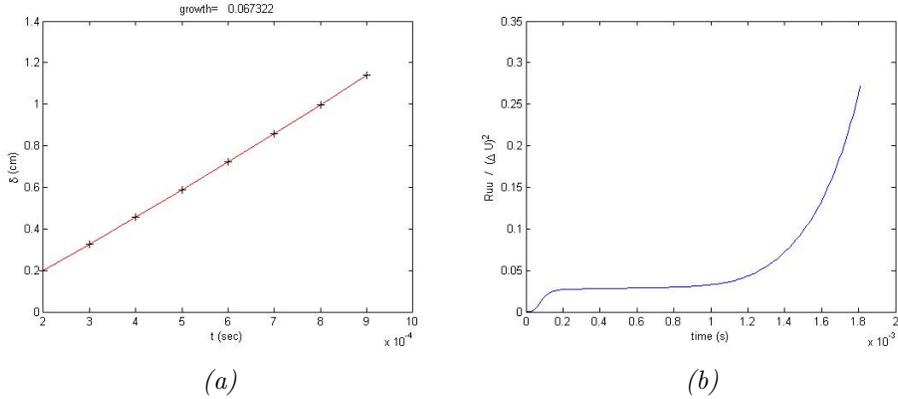


Figure 2: Self-similar growth of mixing layer (a) the characteristic mixing width is plotted against time after an initial transient period (b) the initial transient period is shown  $t < 0.002s$ , the rapid growth of the Reynolds stress is due to the mixing layer encountering the boundary of the domain.

is increased and results in the widened Reynolds stress profile.

In the interest of matching the transport of Reynolds stress using a single length scale the coefficient  $C_{r3}$  and  $C_\mu$  could be adjusted to compensate for the increased time scale.

### 3.2 Two Length Scales

In order to capture the effects of diffusion and dissipation John Schwartzkopf suggested the use of two length scales. One will impact the modeled transport term  $C_{r3} \frac{\partial}{\partial x_k} \left( \tau \bar{\rho} \tilde{R}_{km} \frac{\partial \tilde{R}_{ij}}{\partial x_m} \right)$  and the other will impact the modeled dissipation term  $-\bar{\rho} \frac{2}{3} \frac{K}{\tau}$ .  $C_1$  and  $C_{1v}$  can be calibrated to match the dissipation and diffusion of the experimental data of Bell and Metha [2]. The transport of kinetic energy across the domain is decreased by letting  $C_1 = 1.2$  and comparable growth rates are obtained with  $C_{1v} = 0.9$ . The growth rate ( $\beta = 0.065$ ) obtained with the two length scale model is comparable to that obtained with the single length scale. In figure ?? the profile of  $R_{uu}$  closely matches experiment in diffusion and amplitude, the slight shift of the of the experimental data toward the low speed stream is a common observation when  $U_h/U_l \neq 1$  [8].

Figure 6 shows the evolution of the length scale equation at the center of the mixing regions for one and two length scale models. The single length scale is from figure 3 which matched the experimental and DNS growth rates

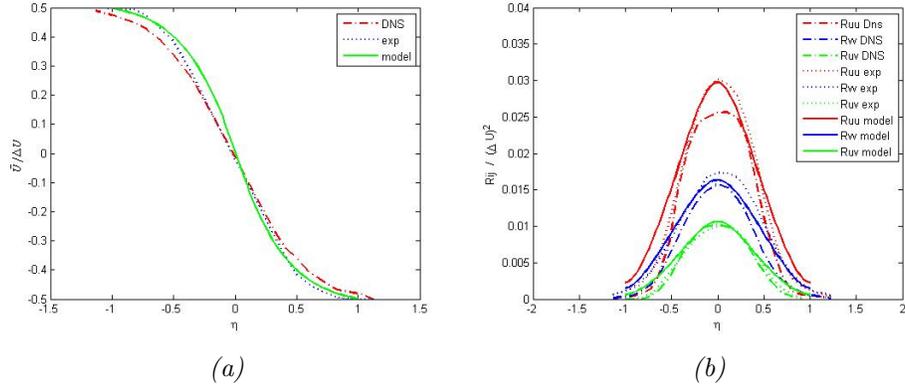


Figure 3: Single length scale model with increased time scale to predict experimental and DNS growth rates (a) Mean velocity profile. (b) Reynold stress profile.

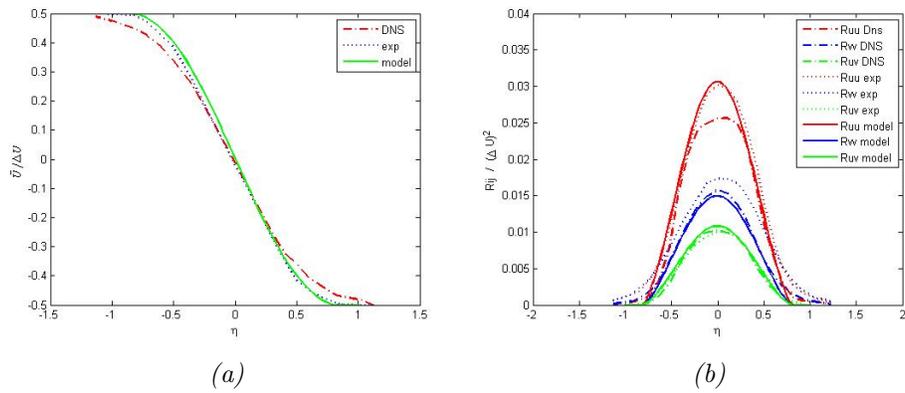


Figure 4: Two length scale model . (a) Mean velocity profile. (b) Reynolds stress profile.

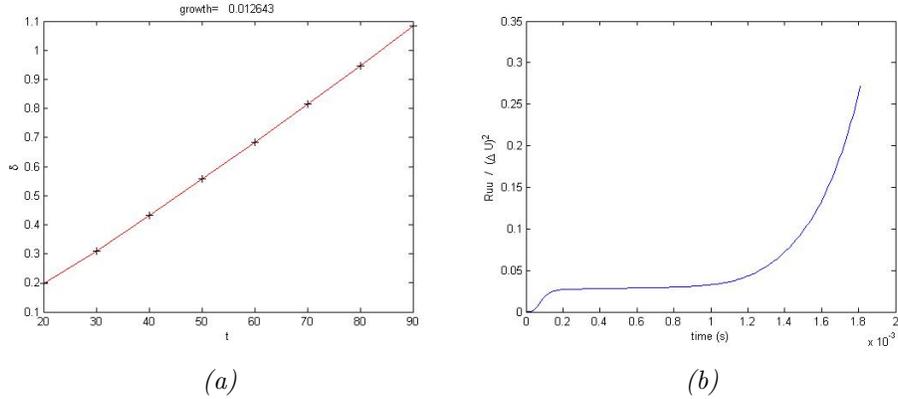


Figure 5: Self-similar growth of mixing layer with two length scale model (a) the characteristic mixing width plotted against time. (b) Evolution of Reynolds stress

with a reduced length scale coefficient  $C_1 = 0.7$ , the two length scale model of figure 4 also matches the growth rates. The length scale increases as the flow evolves, the dissipation scale for the two length scale model and single length scale closely follow one another which would be expected to produce similar growth rates. In the effort to reduce the transport of Reynolds stress the diffusion length scale is shown to be less than the dissipation scales for the two length scale model.

### 3.3 Ideal coefficients for the pressure strain model

The pressure-strain term  $\bar{P}' \left( \frac{\partial u_i''}{\partial x_j} + \frac{\partial u_j''}{\partial x_i} \right)$  of equation 19 makes no contribution to the production or dissipation of the kinetic energy since its trace is zero; it serves to redistribute energy among the components of the Reynolds stress. The coefficients proposed by the LRR-IP model are  $C_{r2} = 0.6$  and  $C_{r4} = 1.8$  and are used in 4. The length scale coefficients were calibrated to match the stream-wise Reynolds stress  $R_{uu}$ , the distribution of Reynolds stresses, particularly the cross-stream component in figure ?? does not match experiment. The value of  $C_{r4}$  is set to 2.6 to increase the energy of  $R_{vv}$ , alternatively, the value of  $C_{r2}$  is set to 0.7, the results are compared in figure 7a and 12b, there is little distinction among them. In fact, any combination  $C_{r2}$  and  $C_{r4}$  within a range of  $0 < C_{r2} < 1$  that satisfy

$$C_{r2} = -0.125C_{r4} + 0.925 \quad (29)$$

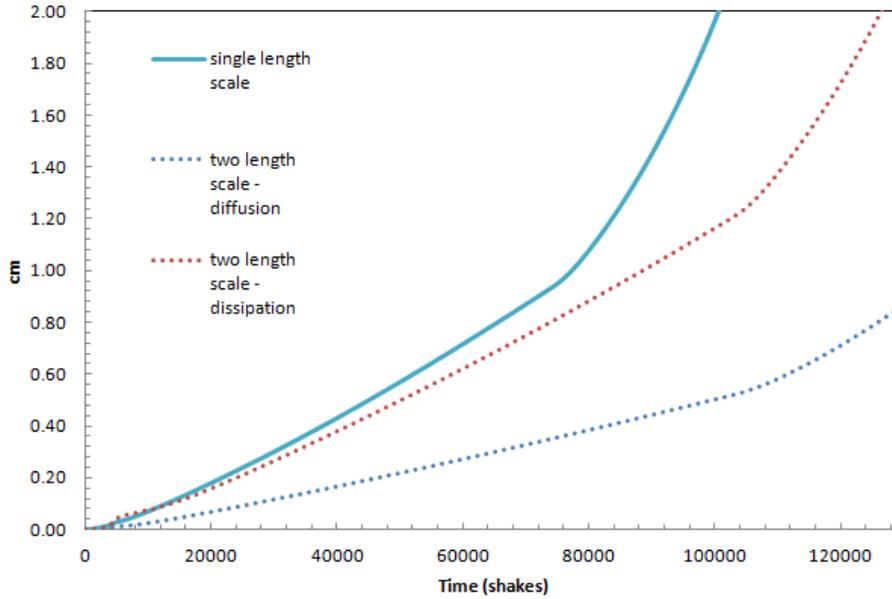


Figure 6: Evolution of length scale equation for one and two length scale models

$C_1$	$C_{1v}$	$C_{r2}$	$C_{r4}$	$C_2$	$C_{2v}$
1.2	0.9	0.6	2.8	1.77	1.77

Table 1: Coefficients used to predict the mixing layer growth rates, Reynolds stress amplitude and distribution of constant density plane mixing layer, experimental results of Bell and Metha are used for this calibration.

produce results nearly identical Reynold stress distribution to those in figure 7a or figure 12b. The combination of pressure-strain coefficients proposed by other investigators and those that satisfy equation 29 is compared in figure 8.

### 3.4 Grid dependence

Multiple simulations were carried out with different grid sizes with a stream-wise domain of 2cm and periodic in the x-direction. The magnitudes of  $R_{uu}$  are shown to reach a self-similar state before the mixing width reaches the boundary of the domain (figure9. The characteristic width of the momentum layer does not exceed 1cm, see figure 5a. The results in this study use a grid size of 0.01cm.

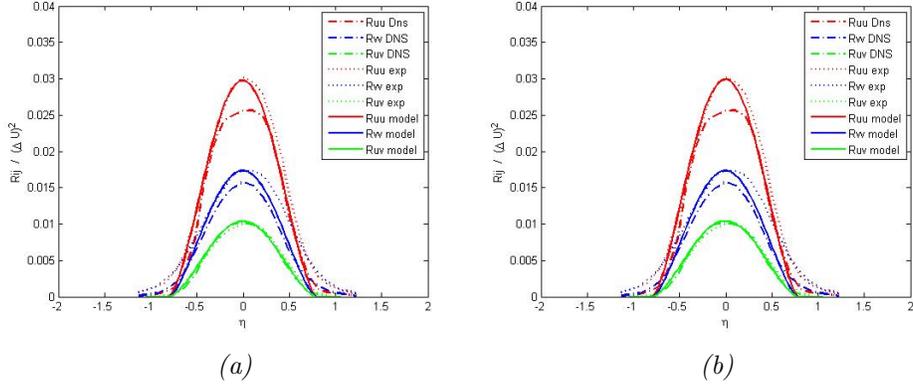


Figure 7: The influence of the pressure-strain terms  $C_{r2}$  and  $C_{r4}$  on the distribution of Reynolds stresses,  $C_1 = 1.2, C_{1v} = 0.9, \beta = 0.067$  (a)  $C_{r2} = 0.6, C_{r4} = 2.6$ . (b)  $C_{r2} = 0.7, C_{r4} = 1.8$

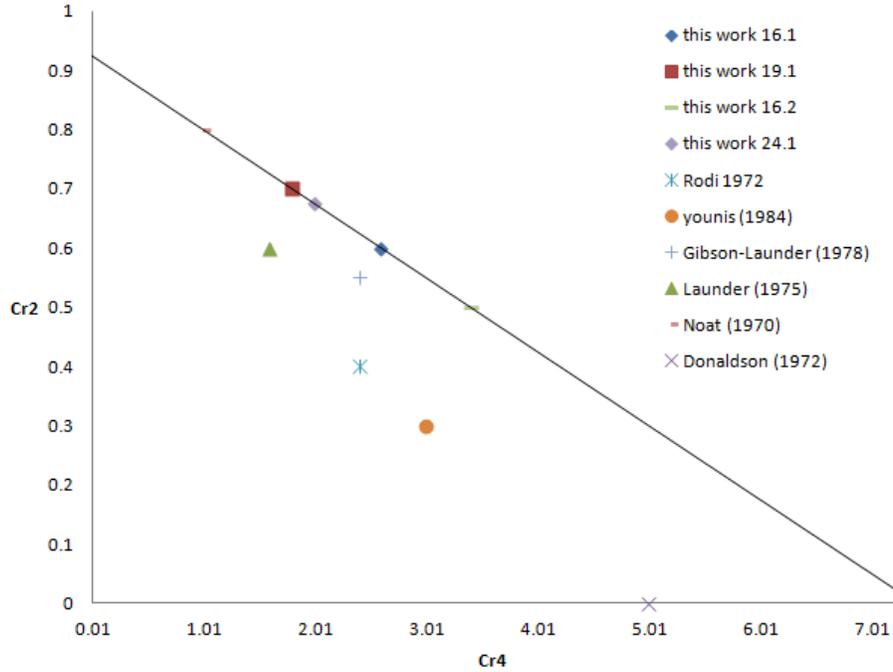


Figure 8: Chosen values of pressure-strain coefficients  $C_{r2}$  and  $C_{r4}$  by other investigators [7], compared to a range of values which give identical Reynolds stress distribution.

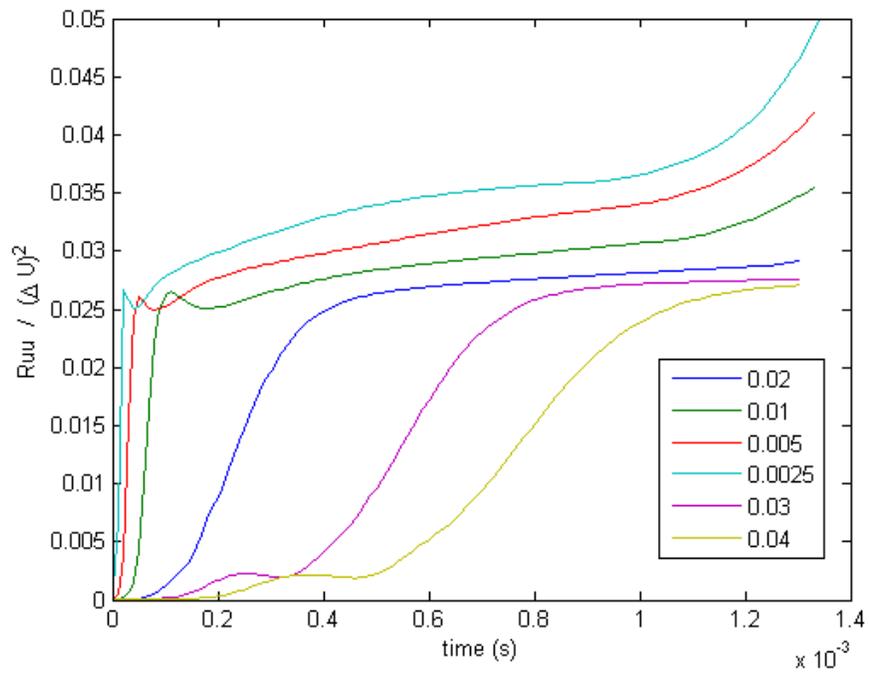


Figure 9: Evolution of  $R_{uu}$  as a function of grid size, the legend has dimensions (cm).

$C_1$	$C_{1v}$	$C_{r2}$	$C_{r4}$	$C_2$	$C_{2v}$	$C_{a1}$	$C_{b2}$	$C_{4v}$	$C_{au}$	$C_{ar}$
1.2	0.9	0.6	2.8	1.77	1.77	2.8	1.8	1.36	1.0	0.3

Table 2: Coefficients used for variable density case.

### 3.5 Variable-density shear

A temporally evolving free shear layer between two different gases (nitrogen and helium) was simulated and compared to the experimental data of Brown and Roshko 1974 [3]. The simulation is carried out at a pressure of  $7atm$  and a velocity difference of  $2000cm/s$  at room temperature.

The molecular weight ( $M$ ) of helium and nitrogen is  $4.003g/mole$  and  $28.02g/mole$ , respectively. The density is found from the ideal gas law  $\rho = \frac{PM}{R_u T}$ ,  $\rho_{He} = 1.22 \times 10^{-3}g/cm^3$ ,  $\rho_{N_2} = 8.45 \times 10^{-3}g/cm^3$ , which gives a density ratio of 7.

The density profile is scaled similar to the velocity profile, the cross-stream spatial coordinate  $y$  is normalized by a characteristic mixing width of the density profile, found from  $\pm 10\%$  of the mean density

$$\bar{\rho}(x, y_\alpha(x), 0) = \rho_l + \alpha(\rho_h - \rho_l), \text{ for } \alpha = 0.9, 0.1 \quad (30)$$

where  $\rho_l$  and  $\rho_h$  are the low and high densities. The characteristic width of the shear layer is then

$$\delta_\rho = y_{0.9} - y_{0.1} \quad (31)$$

the cross-stream position is scaled by the characteristic width and adjusted laterally by the average of the position that defines the boundary of the mixing width  $\bar{y} = (y_{0.9} + y_{0.1})/2$

$$\eta_\rho = \frac{y - \bar{y}}{\delta_\rho} \quad (32)$$

The mean density is normalized by the density of nitrogen

Figure 10 compares the mean velocity profile with the constant density experimental and DNS data previously shown in figure 3a. The flow of nitrogen is on the left half of the figures and helium on the right.

Modified variable density coefficients shown in table 2 are used in the nitrogen/helium mixing layer and compared to standard coefficients and experimental results in figure 11. The value of the standard coefficients and their origins that contribute the multi-species variable density mixing can be found in reference [9].

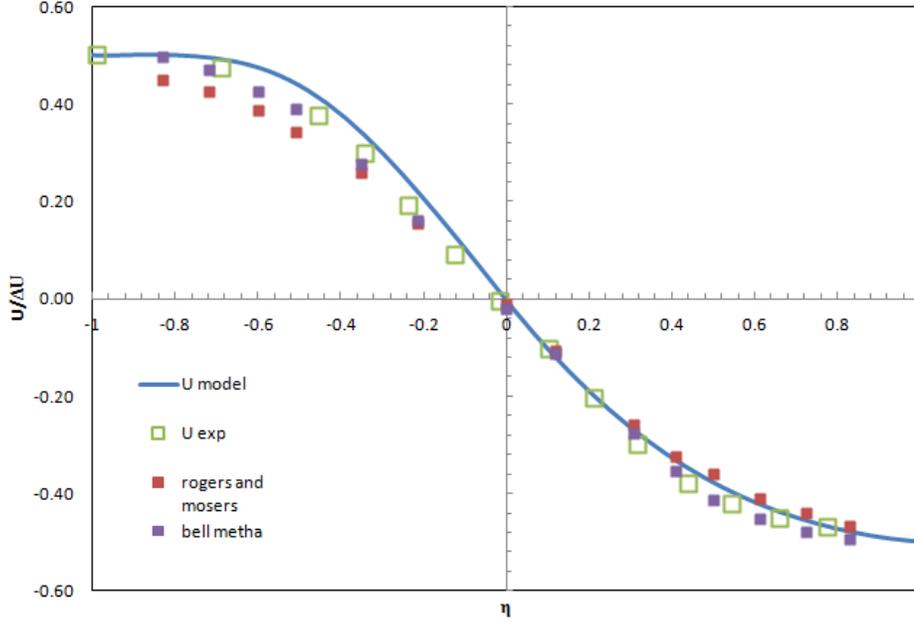


Figure 10: Mean velocity profile of helium and nitrogen mixing layer.

Experiment	DNS	Model
0.062	0.065	0.065

Table 3: Mixing layer growth rates.

Figure 12 show the development of the mixing region based on a characteristic width of the mean velocity and density profile. The growth of the mean velocity profile is  $\beta = 0.065$  almost equal to the constant density case  $\beta = 0.067$ . In agreement with the results of Brown and Roshko, a density ratio of 7 across the mixing layer has little effect of the growth of the mixing region. Also, Brown and Roshko observe that the mixing region of the density profile is almost twice as large as the mixing region of the velocity profile, the growth rate of the density profile based on a characteristic width is  $\beta_\rho = 0.077$ , the model predicts the density mix width will be about 15% larger than the velocity mix width.

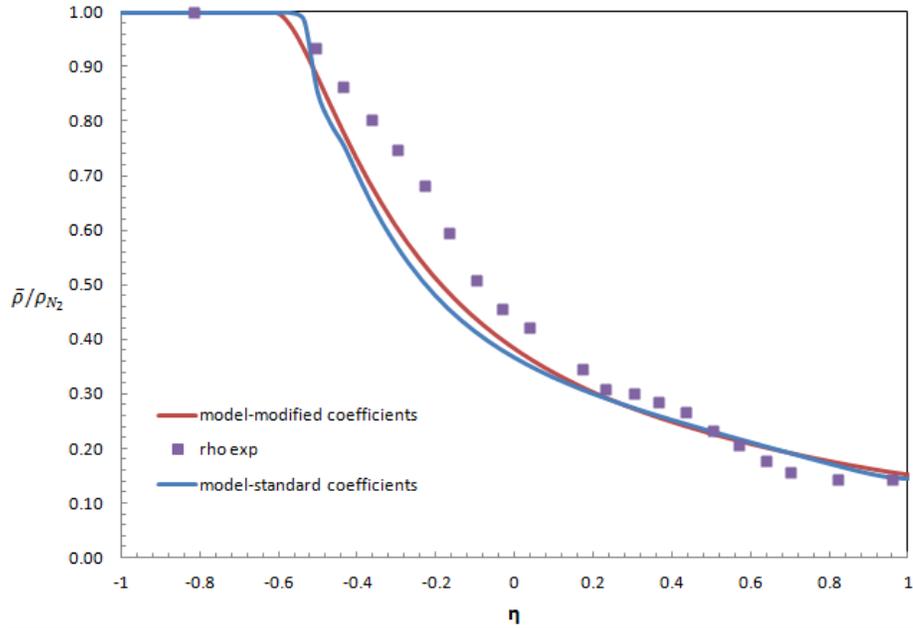


Figure 11: Mean density profile of helium and nitrogen mixing layer with standard and modified variable density coefficients compared to experimental results of Brown and Roshko. Standard coefficients can be found in [9]

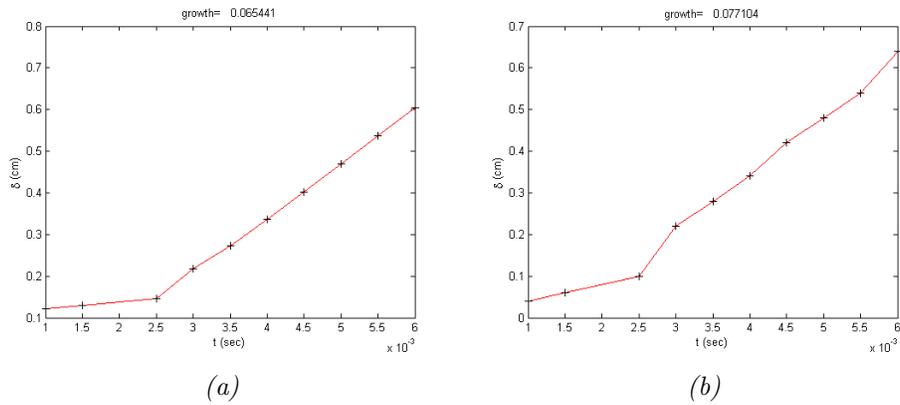


Figure 12: Growth of the temporal mixing region for nitrogen/helium plane mixing layer (a) growth rate based on mean velocity profile. (b) growth rate based on mean density profile.

## 4 Conclusions

Turbulent mixing of a plane mixing layer for constant and variable-density flows has been simulated by second order closure model of the RANS equations and compared to experimental and DNS data. Although the pressure strain coefficients suggested by Launder do not match the anisotropy of Reynold stresses obtained from experimental data, it seems modeling the diffusion and dissipation terms has the greater source of error in growth rates and Reynold stress amplitudes. Satisfactorily growth rates can be achieved with one and two length scales

While there has been many suggested coefficients of the pressure-strain model (figure8) to match the anisotropy of free shear flows the model discussed in this paper requires the Rotta constant and return to isotropy constant to satisfy equation 29.

## References

- [1] G.J. Reece B.E. Launder and W. Rodi. Progress in the development of a reynolds-stress turbulence closure. *J. Fluid Mech.*, volume =, 1975.
- [2] J.H. Bell and R.D. Mehta. Application of a second-moment closure model to mixing processes involving multi-component miscible fluids. *AIAA J*, 28:2034, 1990.
- [3] G.L. Brown and A. Roshko. On density effects and large structure in turbulent mixing layers. *J. Fluid Mech.*, volume =, 1974.
- [4] R.M. Rauenzahn D. Besnard, F.H. Harlow and Zemach. Turbulence transport equations for variable-density turbulence and their relationship to two-field models. LA-UR-12303:LANL, 1992.
- [5] B.D. Daly and F.H. Harlow. Transport equations in turbulence. *Phys. Fluids*, 13:2634, 1970.
- [6] K. Hanjalic and B.E. Launder. A reynolds stress model of turbulence and its application to thin shear flows. *J. Fluid Mech.*, 52:609–638, 1972.
- [7] B.E. Launder. An introduction to single point closure methology. *Found in Simulation and Modeling of Turbulent Flows*, Chap. 6:243–310, 1996.
- [8] M.M. Rogers and R.D. Moser. Direct simulation of a self-similar turbulent mixing layer. *Phys. Fluids*, 6:903, 1990.

- [9] J.D. Schwarzkopf. Application of a second-moment closure model to mixing processes involving multi-component miscible fluids. 2012.

# RANS modeling of RTI and HVDT with BHR3

Ben Trettel<sup>1</sup> August 15, 2012

## Abstract

The BHR3 turbulence model was improved to include two different scales as suggested by Livescu et al. [Liv+09, §4.4.6]: one for turbulent transport and the other for turbulent dissipation. Additionally, destruction terms modeled analogously to production terms were added to the turbulent mass-weighted velocity equation. New model coefficients were developed for this model. The first change was to use  $C_2 = 1.77$  for the isotropic turbulence decay coefficient rather than the  $k$ - $\epsilon$  model's 1.92, which is outside of the experimentally measured values [ML90; KF09].

The new model coefficients were developed to accurately model a wide range of experimental and numerical results: constant and variable density Kelvin-Helmholtz instabilities, Rayleigh-Taylor instabilities, and homogeneous variable density turbulence (HVDT) [LR07]. My work focused on the buoyancy-driven flows: Rayleigh-Taylor instabilities and HVDT.

## 1 Rayleigh-Taylor instability background

A Rayleigh-Taylor instability occurs when a light fluid is under a heavy fluid. If a force is applied that leads to an acceleration, the fluids will start to mix. Small disturbances (e.g. density fluctuations as shown in figure 1) that are always present in real flows will initiate the Rayleigh-Taylor instability. This instability eventually goes turbulent as can be seen in figure 2.

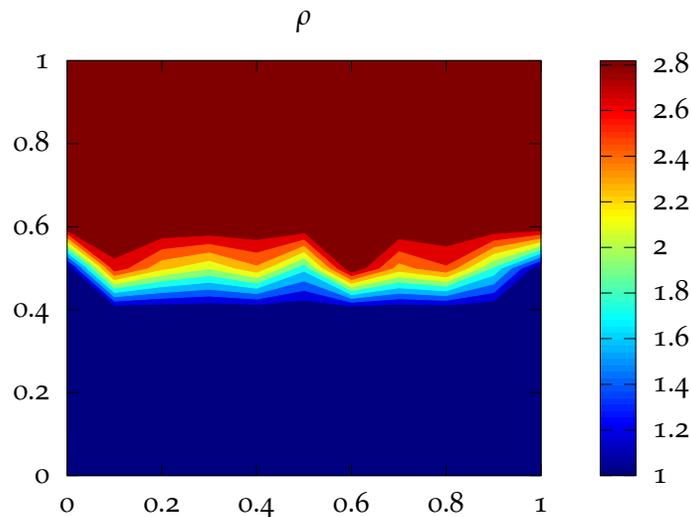


Figure 1: Initial conditions of a hypothetical Rayleigh-Taylor instability.

Figure 2 shows a spatially developing mixing layer, i.e., the two fluids start on the right separated by a splitter plate but flow to the left, leaving the splitter place, which allows them to mix while moving. Each vertical slice of fluid mixes as they move. This configuration is statistically steady — the time-averaged solution is steady.

The simulations described in this work develop temporally. A heavy fluid is placed above a lighter fluid, and the two fluids mix in time.

The Rayleigh-Taylor instability is controlled by the mass density difference of the two fluids. This density

<sup>1</sup> Dept. of Fire Protection Engineering, University of Maryland, College Park. Email: ben.trettel@gmail.com



Figure 2: View of a spatially developing Rayleigh-Taylor instability from experiments by Banerjee, Kraft, and Andrews [BKA10, p. 133] for  $A = 0.47$ . The white fluid is lighter than the dark fluid. The fluids are moving to the left at 1.65 m/s.

difference is non-dimensionalized to form the Atwood number, defined as

$$A = \frac{\rho_H - \rho_L}{\rho_H + \rho_L}, \quad (1.1)$$

where  $\rho_H$  is the density of the heavy fluid and  $\rho_L$  is the density of the lighter fluid.

A key quantity of interest in the Rayleigh-Taylor instability is the width of the mixing layer and how fast it grows. For large times, this width can be shown to be [BKA10, p. 128]

$$h(t) = \alpha A g t^2 \quad (1.2)$$

where  $\alpha$  is the growth rate,  $g$  is the gravitational acceleration due to gravity, and  $t$  is time.

Drazin and Reid [DR04] detail the theory behind a variety of different hydrodynamic instabilities, including for Rayleigh-Taylor instabilities.

## 2 Model description and development

The equations governing the flow of single-phase mixtures of compressible fluids are [Sch+11; PV05]

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j} (\rho u_j) = 0, \quad (2.1)$$

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial}{\partial x_j} (\rho u_i u_j + P \delta_{ij}) = \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i, \quad (2.2)$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x_j} (\rho u_j E + P u_j) = \frac{\partial}{\partial x_j} (\tau_{ij} u_i - q_i), \quad (2.3)$$

$$\frac{\partial \rho c_n}{\partial t} + \frac{\partial}{\partial x_j} (\rho (u_j + V_{n,j}) c_n) = 0. \quad (2.4)$$

where

$\rho$  is the mass density,

$u_j$  is the  $j$ -th component of the fluid velocity,

$P$  is the thermodynamic pressure,

$\tau$  is the shear stress tensor,

$g_i$  is the acceleration due to gravity in the  $i$ -th direction,

$E$  is the total energy (sensible and kinetic — see Poinso and Veynante [PV05, p. 21]) per unit mass,

$q_i$  is the total heat flux,

$c_n$  is the mass fraction for species  $n$ , and

$V_{n,j}$  is the  $j$ -th component of the diffusion velocity.

To fully close this system additional equations are needed including equations of state for thermodynamic pressure and internal energy and equations modeling the viscous stress, heat flux, and mass diffusion.

For Reynolds averaging, quantities are decomposed into mean ( $\bar{u}$ ) and fluctuating ( $u'$ ) components [Popoo, p. 83]:

$$u = \bar{u} + u' . \quad (2.5)$$

The type of averaging used includes time, space, and ensemble averaging. Pope [Popoo, ch. 3] details the definitions of these statistics among others.

For variable-density and compressible flows, mass weighted averages called Favre averages are preferred to avoid additional unclosed terms that need to be modeled. A Favre averaged quantity is defined as

$$\tilde{u} \equiv \frac{\bar{\rho} u}{\bar{\rho}} \quad (2.6)$$

and the quantity  $u$  is split into mean and fluctuating components

$$u = \tilde{u} + u'' . \quad (2.7)$$

After averaging the fluid equations are Schwarzkopf et al. [Sch+11, sec. 2]

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{u}_j) = 0 , \quad (2.8)$$

$$\frac{\partial}{\partial t} (\bar{\rho} \tilde{u}_i) + \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{u}_i \tilde{u}_j + \bar{P} \delta_{ij} + \bar{\rho} \tilde{R}_{ij}) = \bar{\rho} g_i , \quad (2.9)$$

$$\begin{aligned} \frac{\partial}{\partial t} (\bar{\rho} \tilde{E}) + \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{u}_j \tilde{E}) = & - \frac{\partial}{\partial x_j} (\bar{P} \tilde{u}_j) - \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{u}_i \tilde{R}_{ij}) - \frac{\partial}{\partial x_j} (\bar{P}' u'_j) \\ & - \frac{\partial}{\partial x_j} (\overline{\rho E'' u_j''}) - \frac{1}{2} \frac{\partial}{\partial x_j} (\overline{\rho u_i'' u_i'' u_j''}) - \frac{\partial \bar{q}_j}{\partial x_j} , \end{aligned} \quad (2.10)$$

$$\frac{\partial}{\partial t} (\bar{\rho} \tilde{c}_n) + \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{u}_j \tilde{c}_n) = - \frac{\partial}{\partial x_k} (\overline{\rho u_k'' c_n''}) \quad (2.11)$$

where  $\tilde{R}_{ij} \equiv \overline{\rho u_i'' u_j''} / \bar{\rho}$  are the components of the Favre averaged Reynolds stress tensor. The equations have many “unclosed terms” that must be modeled including the Reynolds stress. The BHR<sub>3</sub> model closes these equations.

A derivation of the Reynolds stress transport equation for single-phase compressible flows can be found in Cebeci and Smith [CS74, Ch. 2].

## 2.1 BHR3 model development

Schwarzkopf et al. [Sch+11, sec. 3] reviews the development of the single scale version of the BHR3 turbulence model. The primary changes since that work are the inclusion of two turbulence scales (one for turbulent transport and the other for turbulent dissipation) as opposed to one and adding destruction terms modeled analogously to production terms in the turbulent mass-weighted velocity ( $a_i$ ) equation.

The use of two scales is motivated by turbulence's multiscale physics. The energy spectrum of turbulence [Pop00, ch. 6] includes many scales ranging from the scales of the dimensions of the domain studied down to the smallest scales of the flow, the Kolmogorov scales. The smallest length scales dissipate energy under the "energy cascade" viewpoint that Pope describes. Turbulent transport is associated with the larger physical scales as the geometry usually dictates the resolved (i.e., large scale) transport.

Additionally, Livescu et al. [Liv+09, §4.4.6] describes why buoyancy-driven flows have at least two distinct length scales. To summarize, the increase in energy of the large scales from buoyancy takes time to be felt by the dissipation scales. Thus, at least these two distinct scales are necessary to accurately model buoyancy-driven flows.

The BHR3 model includes two quantities to model the mixing: the density-specific-volume covariance,  $b \equiv -\overline{\rho'(1/\rho)'}$ , and the turbulent mass-weighted velocity,  $a_i \equiv -\overline{u''} = \overline{\rho' u_i} / \bar{\rho}$ . The density-specific-volume covariance is a measure of the mixing. In a fully mixed region  $b$  is zero. The turbulent mass-weighted velocity represents the coupling of density and velocity and is zero for single-phase incompressible flows. The turbulent mass-weighted velocity is also related to the turbulent mass flux,  $\bar{\rho} a_i$ .

The new model for destruction of  $a_i$  is simple — the production and destruction of  $a_i$  are related and an "effective" production can be found by simply assuming that some of the destruction of  $a_i$  is proportional to the production of  $a_i$ .

The next few sections list the full BHR3 equations. Descriptions of several terms are underbraced.

## 2.2 Length and time scales

$$\begin{aligned} \frac{\partial \bar{\rho} S_{\text{diff}}}{\partial t} + \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{u}_j S_{\text{diff}}) &= \underbrace{-\frac{S_{\text{diff}}}{K} \left( \frac{3}{2} - C_1 \right) \bar{\rho} \tilde{R}_{ij} \frac{\partial \tilde{u}_i}{\partial x_j} - \left( \frac{3}{2} - C_2 \right) \bar{\rho} \sqrt{K} + \frac{S_{\text{diff}}}{K} \left( \frac{3}{2} - C_4 \right) a_j \frac{\partial \bar{P}}{\partial x_j} - C_3 \bar{\rho} S_{\text{diff}} \frac{\partial \tilde{u}_j}{\partial x_j}}_{\text{net production}} \\ &+ \underbrace{C_s \frac{\partial}{\partial x_k} \left( \frac{S_{\text{diff}}}{\sqrt{K}} \bar{\rho} \tilde{R}_{kn} \frac{\partial S_{\text{diff}}}{\partial x_n} \right)}_{\text{diffusion}} \end{aligned} \quad (2.12)$$

$$\begin{aligned} \frac{\partial \bar{\rho} S_{\text{diss}}}{\partial t} + \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{u}_j S_{\text{diss}}) &= \underbrace{-\frac{S_{\text{diss}}}{K} \left( \frac{3}{2} - C_{1v} \right) \bar{\rho} \tilde{R}_{ij} \frac{\partial \tilde{u}_i}{\partial x_j} - \left( \frac{3}{2} - C_{2v} \right) \bar{\rho} \sqrt{K} + \frac{S_{\text{diss}}}{K} \left( \frac{3}{2} - C_{4v} \right) a_j \frac{\partial \bar{P}}{\partial x_j} - C_{3v} \bar{\rho} S_{\text{diss}} \frac{\partial \tilde{u}_j}{\partial x_j}}_{\text{net production}} \\ &+ \underbrace{C_s \frac{\partial}{\partial x_k} \left( \frac{S_{\text{diss}}}{\sqrt{K}} \bar{\rho} \tilde{R}_{kn} \frac{\partial S_{\text{diss}}}{\partial x_n} \right)}_{\text{diffusion}} \end{aligned} \quad (2.13)$$

$$\tau_{\text{diss}} \equiv \frac{S_{\text{diss}}}{\sqrt{K}} \quad , \quad \tau_{\text{diff}} \equiv \frac{S_{\text{diff}}}{\sqrt{K}} \quad (2.14)$$

### 2.3 Reynolds stress

$$\begin{aligned}
\frac{\partial (\bar{\rho} \tilde{R}_{ij})}{\partial t} + \frac{\partial}{\partial x_k} (\bar{\rho} \tilde{u}_k \tilde{R}_{ij}) &= \underbrace{a_i \frac{\partial \bar{P}}{\partial x_j} + a_j \frac{\partial \bar{P}}{\partial x_i} - \bar{\rho} \tilde{R}_{ik} \frac{\partial \tilde{u}_j}{\partial x_k} - \bar{\rho} \tilde{R}_{jk} \frac{\partial \tilde{u}_i}{\partial x_k}}_{\text{net production}} + \underbrace{C_{r3} \frac{\partial}{\partial x_k} \left( \tau_{\text{diff}} \bar{\rho} \tilde{R}_{km} \frac{\partial \tilde{R}_{ij}}{\partial x_m} \right)}_{\text{diffusion}} - \underbrace{\bar{\rho} \frac{2}{3} \frac{K}{\tau_{\text{diss}}} \delta_{ij}}_{\text{destruction}} \\
&+ \underbrace{\bar{\rho} C_{r2} \left( \tilde{R}_{ik} \frac{\partial \tilde{u}_j}{\partial x_k} + \tilde{R}_{jk} \frac{\partial \tilde{u}_i}{\partial x_k} \right) - C_{r2} \frac{2}{3} \bar{\rho} \tilde{R}_{mk} \frac{\partial \tilde{u}_m}{\partial x_k} \delta_{ij} - C_{r1} \left( a_i \frac{\partial \bar{P}}{\partial x_j} + a_j \frac{\partial \bar{P}}{\partial x_i} \right) + C_{r1} \frac{2}{3} a_k \frac{\partial \bar{P}}{\partial x_k} \delta_{ij}}_{\text{rapid return to isotropy}} \\
&+ \underbrace{C_{r4} \bar{\rho} \frac{1}{\tau_{\text{diss}}} \left( \tilde{R}_{ij} - \frac{1}{3} \tilde{R}_{kk} \delta_{ij} \right)}_{\text{slow return to isotropy}} \tag{2.15}
\end{aligned}$$

### 2.4 Turbulent mass-weighted velocity

$$\begin{aligned}
\frac{\partial \bar{\rho} a_i}{\partial t} + \frac{\partial}{\partial x_k} (\bar{\rho} \tilde{u}_k a_i) &= \underbrace{(1 - C_{\text{ap}}) b \frac{\partial \bar{P}}{\partial x_i} - (1 - C_{\text{ar}}) \tilde{R}_{ik} \frac{\partial \bar{\rho}}{\partial x_k} - (1 - C_{\text{au}}) \bar{\rho} a_k \frac{\partial \tilde{u}_i}{\partial x_k}}_{\text{net production}} + \underbrace{\bar{\rho} \frac{\partial}{\partial x_k} (a_k a_i)}_{\text{redistribution}} \\
&+ \underbrace{\bar{\rho} C_a \frac{\partial}{\partial x_k} \left( \tau_{\text{diff}} \tilde{R}_{kn} \frac{\partial a_i}{\partial x_n} \right)}_{\text{diffusion}} - \underbrace{C_{a1} \bar{\rho} \frac{1}{\tau_{\text{diss}}} a_i}_{\text{destruction}} \tag{2.16}
\end{aligned}$$

### 2.5 Density-specific-volume covariance

$$\frac{\partial \bar{\rho} b}{\partial t} + \frac{\partial}{\partial x_k} (\bar{\rho} b \tilde{u}_k) = \underbrace{-2(b+1) a_k \frac{\partial \bar{\rho}}{\partial x_k}}_{\text{production}} + \underbrace{2 \rho a_k \frac{\partial b}{\partial x_k}}_{\text{redistribution}} + \underbrace{\bar{\rho}^2 C_b \frac{\partial}{\partial x_k} \left( \frac{\tau_{\text{diff}}}{\bar{\rho}} \tilde{R}_{kn} \frac{\partial b}{\partial x_n} \right)}_{\text{diffusion}} - \underbrace{C_{b1} \bar{\rho} \frac{1}{\tau_{\text{diss}}} b}_{\text{destruction}} \tag{2.17}$$

### 2.6 Species mass fraction

$$\frac{\partial \bar{\rho} \tilde{c}_n}{\partial t} + \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{c}_n \tilde{u}_j) = \frac{\partial}{\partial x_k} \left( C_c \frac{S_{\text{diff}}}{\sqrt{K}} \bar{\rho} \tilde{R}_{km} \frac{\partial \tilde{c}_n}{\partial x_m} \right) \tag{2.18}$$

## 3 Modeling the Rayleigh-Taylor instability

LANL's code Rage [Git+08] was used to model Rayleigh-Taylor instabilities. The Atwood number used for these simulations was 0.5 and the results were compared against DNS data by Cabot and Cook [CC06] that was processed by Livescu et al. [Liv+09] and experiments by Banerjee, Kraft, and Andrews [BKA10].

### 3.1 Non-dimensionalization

The  $z$  location in the Rayleigh-Taylor mixing layer was non-dimensionalized by an integral mix width defined by [AS90]

$$h(t) \equiv 6 \int_{-\infty}^{\infty} f_v (1 - f_v) dz \quad , \quad f_v \equiv \frac{\rho - \rho_L}{\rho_H - \rho_L} . \tag{3.1}$$

case	DNS growth rate	exp. growth rate	RANS growth rate
Rayleigh-Taylor ( $A = 0.5$ )	0.0416	0.141	0.0447
Kelvin-Helmholtz	0.062	0.069	0.067

*Table 1:* Growth rates for the Rayleigh-Taylor ( $A = 0.5$ ) case and Kelvin-Helmholtz case for comparison. The RT DNS growth rate is from the data-set of Cabot and Cook [CC06] and the experimental growth rate is from Banerjee, Kraft, and Andrews [BKA10, p. 153] where the total growth rate is the sum of the two integral half mix-widths interpolated for  $A = 0.5$ . For KH, the DNS growth rate is from Rogers and Moser [RM94], the experimental growth rate is from Bell and Mehta [BM90], and the RANS BHR3 growth rate is from Smith [Smi12, §3.1].

The times were non-dimensionalized by [Sch+11, p. 25]

$$\tau \equiv \sqrt{\frac{h(t)}{A|g|}}, \quad (3.2)$$

the velocities by

$$\lambda \equiv \frac{h(t)}{\tau}, \quad (3.3)$$

and the energies by  $\lambda^2$ .

### 3.2 Quantifying the model's fit

I wanted a fast and easy way to approximately quantify how well the output from a run fit experimental or DNS data. I developed a metric that my post-processing script (which is in an appendix) could output. This metric was the sum of the errors in  $K$ ,  $b$ , and  $a_z$  of the peak values, integral, width (where, as detailed before, the ideal width is 1), and growth rate. The RANS model was compared solely against the DNS data due to the discrepancy in the growth rates between the experiments and DNS. The DNS data was smoothed a bit to reduce the effects of some small fluctuations in it that can not be represented in the RANS simulation.

The BHR3 model does not accurately predict the tails of the Rayleigh-Taylor instability as can be seen in figures 3 through 5. The tails of the RANS data are sharper than the tails of the DNS data. This problem occurs because the tails of the DNS Rayleigh-Taylor layer are not fully turbulent. BHR3 applies to the turbulent center of mixing layer, not the edges [Liv12]. Thus, BHR3 is not capable of complete accuracy in the tails. In the tails, the instability retains memory of the initial conditions, which affects the growth rate of the tails as explained below.

### 3.3 Growth rate

The growth rate of the Rayleigh-Taylor instability is only found in the self-similar regime, where the previously detailed non-dimensionalization collapses all times into a single curve.

The growth rate was calculated with the following formula:

$$\alpha = \left( \frac{\sqrt{h(t)} - \sqrt{h(t_0)}}{\sqrt{A|g|}(t - t_0)} \right)^2 \quad (3.4)$$

as Schwarzkopf et al. [Sch+11, p. 22] did.

The growth rate for the Rayleigh-Taylor experiments is found to be over 3 times as high as the DNS

growth rate as seen in table 1. Livescu [Liv12] attributes the difference in the growth rates between the experiments and simulations to the effects of the initial and boundary conditions. Low-wavenumber oscillations that exist in reality influence the growth rates. Because of the difficulty in prescribing initial and boundary conditions, the RANS growth rate was matched to the DNS growth rate in this work.

### 3.4 Results

For the model coefficients developed in this work, plots of turbulent kinetic energy ( $K$ ), density-specific-volume covariance ( $b$ ), and turbulent mass-weighted velocity ( $a_z$ ) are plotted in figures 3 through 5. Unlike in previous work [Sch+11], the fit of the BHR3 model and the DNS is excellent.

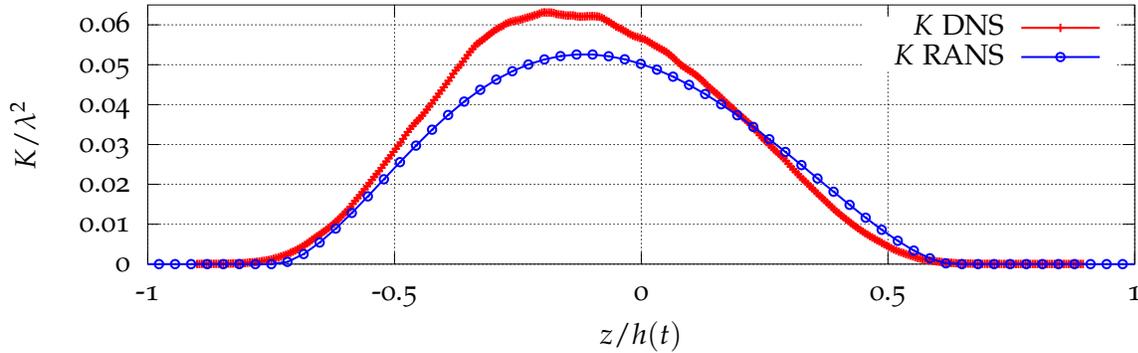


Figure 3: Comparison of DNS and RANS results for the turbulent kinetic energy,  $K$ , for  $A = 0.5$ .

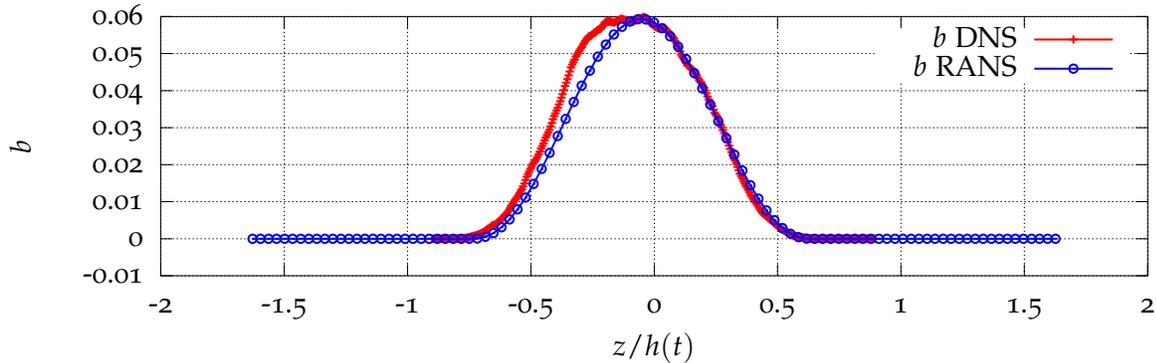


Figure 4: Comparison of DNS and RANS results for the density-specific-volume covariance,  $b$ , for  $A = 0.5$ .

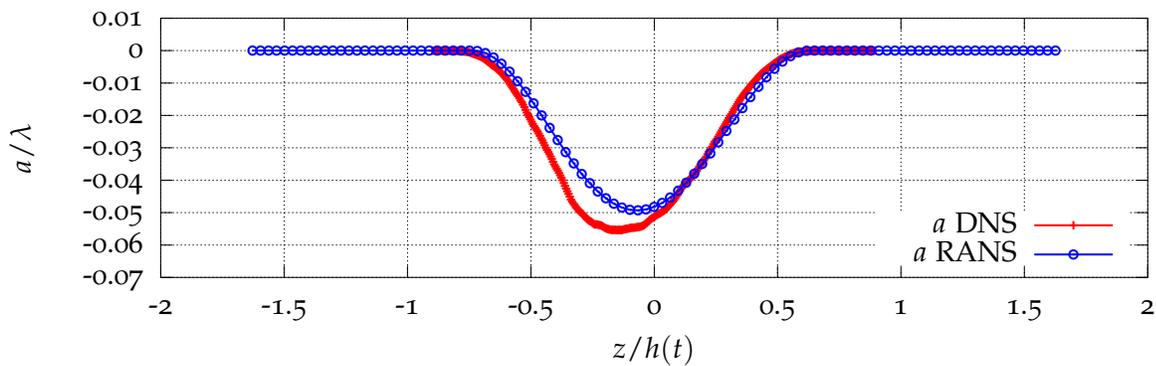


Figure 5: Comparison of DNS and RANS results for the turbulent mass-weighted velocity,  $a_z$ , for  $A = 0.5$ .

#### 4 Modeling homogeneous variable density turbulence (HVDT)

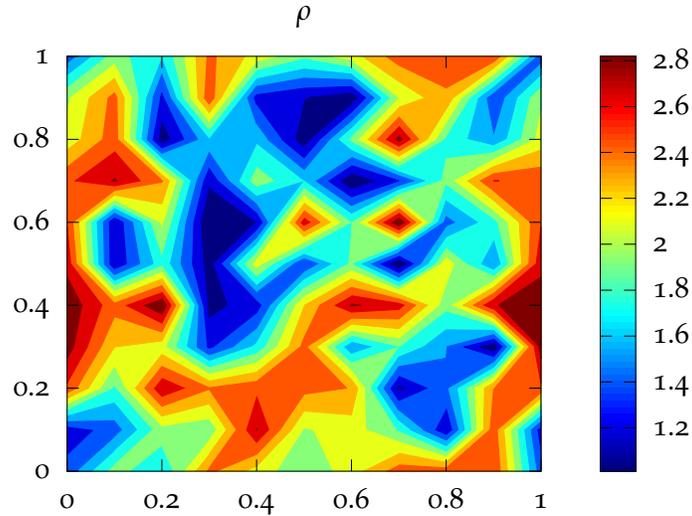


Figure 6: Illustrative example of the initial conditions for density in homogeneous variable density turbulence.

Homogeneous variable density turbulence (HVDT) is generated by modeling a homogeneous mixture of random density fluctuations in a periodic cube, e.g., as shown in figure 6. Livescu and Ristorcelli [LR07] modeled HVDT with DNS, and the results were used as another point of comparison for the development of model coefficients for the BHR<sub>3</sub> model.

The BHR<sub>3</sub> equations reduce to ODEs in this case. A Mathematica file was developed which integrated the BHR<sub>3</sub> equations in time, which I used to model HVDT. The results from the current model coefficients are shown in figure 7.

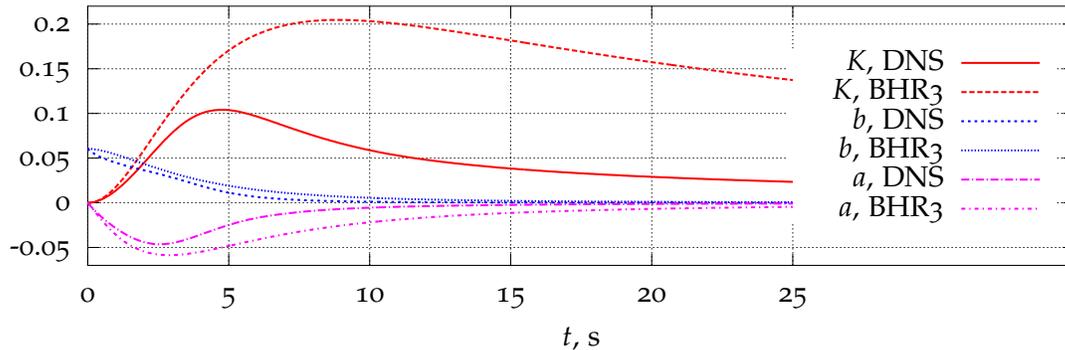


Figure 7: The BHR<sub>3</sub> model with current coefficients compared against DNS of homogeneous variable density turbulence.

equation	new coeff.	new value	justification	old coeff.	old value
$S_{\text{diff}}$	$C_1$	1.2	KH growth rates	$C_1$	1.2
$S_{\text{diff}}$	$C_2$	1.77	[ML90], [KF09]	$C_2$	1.92
$S_{\text{diff}}$	$C_4$	1.0	RT growth rates	$C_3$	1.2
$S_{\text{diff}}$	$C_3$	0		—	—
$S_{\text{diff}}$	$C_s$	4.2	[Sch+11, p. 19]	$C_s$	4.2
$S_{\text{diss}}$	$C_{1v}$	0.9	KH $R_{ij}, \alpha$	—	—
$S_{\text{diss}}$	$C_{2v}$	1.77	[ML90], [KF09]	—	—
$S_{\text{diss}}$	$C_{4v}$	1.31	HVDT DNS [LR07]	—	—
$S_{\text{diss}}$	$C_{3v}$	0		—	—
$S_{\text{diss}}$	$C_{sv}$	4.2		—	—
$R_{ij}$	$C_{r3}$	0.42	[Sch+11, p. 23]	$C_r$	0.42
$R_{ij}$	$C_{r4}$	2.6	KH $R_{ij}, \alpha$	$C_{r3}$	1.8
$R_{ij}$	$C_{r2}$	0.6	RDT [Pop00, p. 423]	$C_{r2}$	0.6
$R_{ij}$	$C_{r1}$	0.3		$C_{r1}$	0.3
$a_i$	$C_{\text{ap}}$	0.28	RT $K, a_i, b, \alpha$	—	—
$a_i$	$C_{\text{ar}}$	0		—	—
$a_i$	$C_{\text{au}}$	0		—	—
$a_i$	$C_a$	0.3	RT width	$C_a$	0.3
$a_i$	$C_{a1}$	2.8	RT $K, a_i, b, \alpha$	$C_{a1}$	3.2
$b$	$C_b$	0.3	RT width	$C_b$	0.3
$b$	$C_{b2}$	1.8	RT $K, a_i, b, \alpha$	$C_{b1}$	2
$c_n$	$C_c$	0.56	[Sch+11, p. 23]	$C_c$	0.56

Table 2: Model coefficients found to match the Rayleigh-Taylor instability and HVDT best. The old coefficients are from BHR3 before this work.

## 5 Model coefficients and their justification

The model coefficients developed in this work and a summary of their justification is in table 2.

### 5.1 Isotropic turbulence decay

The standard  $k$ - $\epsilon$  model [Pop00, p. 375-376] chooses  $C_2 = 1.92$  as the isotropic turbulence decay coefficient. The first step towards improving the coefficients was to choose  $C_2 = C_{2v} = 1.77$  as suggested by Mohamed and LaRue [ML90, p. 211, fig. 13] based on experimental data they compiled. More recent data summarized by Kurian and Fransson [KF09, p. 13, fig. 9(a)] suggests that  $C_2$  increases with the grid Reynolds number, however, this may be misleading as the lower grid Reynolds number experiments had significant anisotropy [KF09, p. 9, fig. 4(a)] below a grid Reynolds number of approximately  $2 \times 10^3$ .

The grid Reynolds number is defined as

$$\text{Re}_M \equiv \frac{U_0 M}{\nu} \quad (5.1)$$

where  $U_0$  is the free stream velocity,  $M$  is the mesh width, and  $\nu$  is the kinematic viscosity of the fluid. The effect of  $M$  can be thought of as the initial conditions' effect. The grid Reynolds number combines the initial conditions and the fluid velocity. The dependence on the grid Reynolds number for the isotropic turbulence decay experiments may be due to the effect of the initial conditions, fluid velocity, anisotropy, or possibly a cause that was not yet considered.

The spread in the data for grid Reynolds numbers greater than about  $2 \times 10^3$  suggests that  $1.6 \lesssim C_2 \lesssim 1.8$

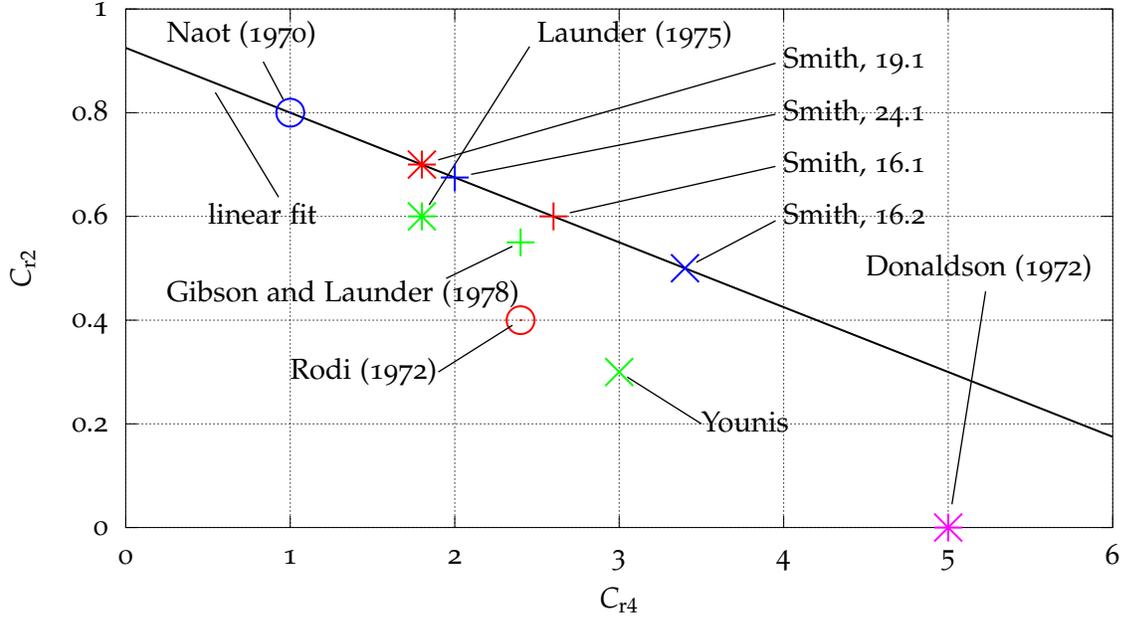


Figure 8: Proposed values of  $C_{r2}$  and  $C_{r4}$ . The BHR<sub>3</sub> model with the coefficients on the line return results which visually match Kelvin-Helmholtz experimental data [BM90] nearly exactly [Smi12, §3.3]. The line is  $C_{r2} = -0.23C_{r4} + 1$ . The numbered points refer to different configurations tested. Other citations are from [Lau96, p. 268].

with most of the data clustering around  $C_2 = 1.77$ . Thus, choosing  $C_2 = 1.77$  seems most consistent with the available experimental data.

Speziale and Bernard [SB92] prove theoretically that the isotropic decay exponent asymptotically approaches 2 as the Reynolds number increases. How high the Reynolds number needs to be for the coefficient to be near two is unclear based on the mentioned work.

## 5.2 Rapid distortion theory

Pope [Pop00, p. 423] notes that if  $C_{r2}$  is set to 0.6 then the isotropization-of-production model returns the correct response of rapidly distorted isotropic turbulence.

Launder [Lau96, p. 268] suggests that compatible values of  $C_{r2}$  and  $C_{r4}$  lie on a line. Smith [Smi12, §3.3] found a line that prescribes values of  $C_{r2}$  and  $C_{r4}$  as can be seen in figure 8. Using the BHR<sub>3</sub> model with coefficients found on this line visually appear to give results which excellently match the experimental data of Bell and Mehta [BM90]. The value of  $C_{r4}$  on this line when  $C_{r2} = 0.6$  is 2.6.

## 5.3 Kelvin-Helmholtz instability

The following coefficients fit the Kelvin-Helmholtz experiments and DNS best [Smi12, §3.3]:

$$C_1 = 1.2 \quad , \quad C_{1v} = 0.9 \quad , \quad C_{r4} = 2.6 \quad . \quad (5.2)$$

## 5.4 Homogeneous variable density turbulence (HVDT)

John Schwarzkopf and I generated a wide variety of different configurations which fit the HVDT DNS data reasonably well. “Reasonably well” means that the BHR<sub>3</sub>  $b$  was overlapping then DNS  $b$  (which is

easy to do), that the BHR3  $a_z$  is almost overlapping or overlapping the DNS  $a_z$ , and that the BHR3  $K$  has the same general trend as the DNS  $K$ . Perfect fit was not expected.

The coefficients with the best fit to the HVDT data were used as starting points for tweaking to fit the Rayleigh-Taylor DNS data better.

The DNS data and BHR3 model can be compared in figure 7.

The single time scale version of BHR3 had poor fit for  $K$  — the turbulent kinetic energy greatly overpredicted the DNS value. Here the RANS turbulent kinetic energy peaks with a slightly higher magnitude than the DNS data and decreases after the peak in a way qualitatively like the DNS data.

Another possible set of coefficients is similar to the coefficients in table 2, but with the following modifications:

$$C_{a1} = 1.5 \quad , \quad C_{b2} = 1.5 \quad , \quad C_{ap} = 0.40 \quad , \quad C_{4v} = 1.36 \quad , \quad C_4 = 1.12 \quad . \quad (5.3)$$

Future work should consider trying these coefficients. With these coefficients the overall fit of the Reynolds stresses, turbulent kinetic energy,  $b$ , and turbulent mass-weighted density are slightly worse than with the coefficients in table 2, but not significantly so. The growth rate for the  $A = 0.5$  case is slightly better ( $\alpha = 0.0431$ ) but this can be improved in the table's coefficients.

## 6 Conclusions

The two-scale model is a significant improvement in the development of BHR3. With two scales Rayleigh-Taylor instabilities, the Kelvin-Helmholtz instabilities, and homogeneous variable density turbulence can be modeled reasonably accurately. The BHR3 model has been validated against a wide variety of flows for a turbulence model, and our hope is that it can be used as a general-purpose turbulence model.

Additionally, the isotropic turbulence decay coefficient was changed so that BHR3 fits the latest experimental data.

### 6.1 Future directions

These coefficients can be applied to Richtmyer-Meshkov instabilities to further test their applicability in general. Discrepancies can be addressed either in additional or improved turbulence models or improved model coefficients.

### 6.2 Acknowledgments

I would like to thank Rob Gore and John Schwarzkopf for suggesting this research and their assistance in explaining the intricacies of turbulence. I would also like to thank Jeffrey Smith for working with me this summer and providing coefficients found by matching shear layer data. This work was performed during the 2012 Los Alamos National Lab Computational Physics Student Summer Workshop. The financial and logistical support the workshop provided is gratefully acknowledged.

## References

- [AS90] Malcolm J. Andrews and Dudley Brian Spalding. "A simple experiment to investigate two-dimensional mixing by Rayleigh-Taylor instability". In: *Physics of Fluids A: Fluid Dynamics* 2.6 (June 1990), pp. 922–927. ISSN: 08998213. DOI: doi:10.1063/1.857652. URL: [http://pof.aip.org/resource/1/pfadeb/v2/i6/p922\\_s1](http://pof.aip.org/resource/1/pfadeb/v2/i6/p922_s1). (Cit. on p. 5).
- [BKA10] Arindam Banerjee, Wayne N. Kraft, and Malcolm J. Andrews. "Detailed measurements of a statistically steady Rayleigh-Taylor mixing layer from small to high Atwood numbers". In: *Journal of Fluid Mechanics* 659.1 (2010), pp. 127–190. DOI: 10.1017/S0022112010002351. (Cit. on pp. 2, 5, 6).
- [BM90] James H. Bell and Rabindra D. Mehta. "Development of a two-stream mixing layer from tripped and untripped boundary layers". In: *AIAA Journal* 28.12 (1990), pp. 2034–2042. (Cit. on pp. 6, 10).
- [CC06] William H. Cabot and Andrew W. Cook. "Reynolds number effects on Rayleigh-Taylor instability with possible implications for type Ia supernovae". In: *Nature Physics* 2.8 (2006), pp. 562–568. ISSN: 1745-2473. DOI: 10.1038/nphys361. URL: <http://www.nature.com/nphys/journal/v2/n8/full/nphys361.html>. (Cit. on pp. 5, 6).
- [CS74] Tuncer Cebeci and A. M. O Smith. *Analysis of turbulent boundary layers*. Applied Mathematics and Mechanics 15. New York: Academic Press, June 1974. ISBN: 0121646505. (Cit. on p. 4).
- [DR04] P. G. Drazin and W. H. Reid. *Hydrodynamic Stability*. 2nd ed. Cambridge University Press, Sept. 2004. ISBN: 0521525411. (Cit. on p. 2).
- [Git+08] Michael Gittings et al. "The RAGE radiation-hydrodynamic code". In: *Computational Science & Discovery* 1.1 (Nov. 2008), p. 015005. ISSN: 1749-4699. DOI: 10.1088/1749-4699/1/1/015005. URL: <http://iopscience.iop.org/1749-4699/1/1/015005>. (Cit. on p. 5).
- [KF09] Thomas Kurian and Jens H. M. Fransson. "Grid-generated turbulence revisited". In: *Fluid Dynamics Research* 41.2 (Apr. 2009), p. 021403. ISSN: 0169-5983, 1873-7005. DOI: 10.1088/0169-5983/41/2/021403. URL: <http://iopscience.iop.org/0169-5983/41/2/021403>. (Cit. on pp. 1, 9).
- [Lau96] Brain E. Launder. "An introduction to single-point closure methodology". In: *Simulation and Modeling of Turbulent Flows*. Ed. by Thomas B. Gatski, M. Yousuff Hussaini, and John L. Lumley. Oxford University Press, July 1996, pp. 243–310. ISBN: 9780195106435. (Cit. on p. 10).
- [Liv+09] D. Livescu et al. "High-Reynolds number Rayleigh-Taylor turbulence". In: *Journal of Turbulence* (2009), N13. DOI: 10.1080/14685240902870448. URL: <http://www.tandfonline.com/doi/abs/10.1080/14685240902870448>. (Cit. on pp. 1, 4, 5).
- [Liv12] Daniel Livescu. Aug. 2012. (Cit. on pp. 6, 7).
- [LR07] Daniel Livescu and J. R. Ristorcelli. "Buoyancy-driven variable-density turbulence". In: *Journal of Fluid Mechanics* 591 (2007), pp. 43–71. DOI: 10.1017/S0022112007008270. URL: <http://public.lanl.gov/livescu/frames/jfm3.pdf>. (Cit. on pp. 1, 8, 9).
- [ML90] Mohsen S. Mohamed and John C. LaRue. "The decay power law in grid-generated turbulence". In: *Journal of Fluid Mechanics* 219 (1990), pp. 195–214. DOI: 10.1017/S0022112090002919. (Cit. on pp. 1, 9).
- [Pop00] Stephen B. Pope. *Turbulent Flows*. 1st ed. Cambridge University Press, Oct. 2000. ISBN: 0521598869. (Cit. on pp. 3, 4, 9, 10).
- [PV05] Thierry Poinso and Denis Veynante. *Theoretical and numerical combustion*. R. T. Edwards, Jan. 2005. ISBN: 9781930217102. (Cit. on pp. 2, 3).
- [RM94] Michael M. Rogers and Robert D. Moser. "Direct simulation of a self-similar turbulent mixing layer". In: *Physics of Fluids* 6.2 (Feb. 1994), pp. 903–923. ISSN: 10706631. DOI: doi:10.1063/1.868325. URL: [http://pof.aip.org/resource/1/phfle6/v6/i2/p903\\_s1](http://pof.aip.org/resource/1/phfle6/v6/i2/p903_s1). (Cit. on p. 6).

- [SB92] Charles G. Speziale and Peter S. Bernard. "The energy decay in self-preserving isotropic turbulence revisited". In: *Journal of Fluid Mechanics* 241 (1992), pp. 645–667. DOI: 10.1017/S0022112092002180. (Cit. on p. 10).
- [Sch+11] John D. Schwarzkopf et al. "Application of a second-moment closure model to mixing processes involving multicomponent miscible fluids". In: *Journal of Turbulence* (2011), N49. DOI: 10.1080/14685248.2011.633084. URL: <http://www.tandfonline.com/doi/abs/10.1080/14685248.2011.633084>. (Cit. on pp. 2–4, 6, 7, 9).
- [Smi12] Jeffrey Smith. *Turbulent modeling of plane mixing layer*. Tech. rep. Los Alamos National Lab, Aug. 2012. (Cit. on pp. 6, 10).

## 7 Appendix A: Input deck

```
1
2 ! Modified from base case A=0.04 TAMU water channel data
3
4 !=====
5 ! CONTROL
6 !=====
7 secmax = 396000 ! How long run should go (in secs)
8 pname = "rt2s" ! all output files have this prefix
9
10 test_pname='rt11_1v' ! 2d R-T
11 userparms(1)=1. ! rho-lo
12 userparms(2)=3. ! rho-hi
13 userparms(3)=1.e4 ! rho*cv
14 userparms(4)=2000. ! tev0
15 userparms(5)=0.4 ! GAMMA-1
16
17 mincellpe = 0 ! For parallel runs these two entries determine the min and
18 maxcellpe = 0 ! max number of cells on each processor. 0 turns it off
19 ncmx = 50000000 ! Maximum number of cycles (timesteps) to run
20 secmax = 27000
21 tmax = 2.6 ! Maximum time in simulation time units
22 tedit = 5.e-1 ! How often to print status information and restart dumps
23 ncredit = 1000 ! If not 0 how many cycles between status and restart dumps
24 modcyc = 1000 ! Status dumps only
25
26 kread = -1 ! Does nothing at -1, other #'s determine cycle to restart
27 uselast = .true. ! If restart dumps exists in this director, use the last one
28 ! Note: if this happens some initialization parameters
29 ! set later are skipped.
30 shortmodcyc = 10 ! Prints a short status message every cycle
31
32
33 !=====
34 ! Hydro
35 !=====
36 numrho = 4
37 hydro_version = 2
38 interface_option = 0
39
40
41 ! TURBULENCE MIX
42 !S0_max = 1.0
43 do_mix = .true. ! Turn on/off BHR
44 mix_model = 'BHR-3'
45 mix_t_start=0. ! Determine when model is turned on
46 mix_t_end=10.e8 ! Determine when model is turned off
47 turb_scale_0 = 0.1000 ! Determine initial length scale
48 turb_ske_0 = 0.25 ! Determine initial value for ke
49 const_scale = .false. ! Use a regular L-transport equation
50 use_mixscale = .true.
51 mix_b_cutoff = 1.0e-20
52 mix_numrho = 4
53 use_mixvisc_scale = .true. ! uses both S eqns (must be false for BHR-2)
54 num_mixmat=2 ! Number of materials to be mixed
55 mixmat(1)=1,2 ! Allow both materials to mix
56 use_Rij_diff_coef=.true. ! .false. means it uses turbulent viscosity
```

```

57
58 !=====
59 ! Coefficients for students to alter
60 !=====
61 ! coefficients that affect variable density RT
62   mix_ca1 = 2.80 ! fit this
63   mix_cb2 = 1.80 ! fit this
64   mix_cap = 0.28 ! fit this
65   mix_c4v = 1.31 ! adjust to fit HVDT DNS data
66   mix_c4  = 1.00 ! adjust to get correct growth rate
67
68   mix_c_a = 0.30 ! adjust to get correct widths
69   mix_c_b = 0.30 ! adjust to get correct widths
70   mix_c_c = 0.56
71   mix_cr1 = 0.3
72   mix_cau = 0.0
73   mix_car = 0.0
74
75 ! coefficients that affect single phase KH
76   mix_c1  = 1.2
77   mix_c1v = 0.9 ! set from KH
78   mix_cr2 = 0.6 ! do not change, RDT, see Pope p. 423
79   mix_cr3 = 0.42
80
81 ! coefficients that affect both
82   mix_cr4 = 2.6 ! set from KH
83   mix_c2  = 1.77 ! do not change
84   mix_c2v = 1.77 ! do not change
85   mix_c_s = 4.2
86   mix_c_sv = 4.2
87
88 ! Additional coefs
89   mix_c3  = 0.0 ! leave at zero
90   mix_c3v = 0.0 ! leave at zero
91 !=====
92 ! PLOTS
93 !=====
94
95 !mix_plts = .true.
96 !mix_plts_ncedit = 250
97 !mix_plts_xmn = 0.0
98 !mix_plts_xmx = 0.25
99 !mix_plts_ymn = -80.0
100 !mix_plts_ymx = 80.0
101 !mix_plts_ncstart = 1
102 ! mix_plts_tmzdir = 'y'
103 ! mix_plts_tmzmat = 1
104 !mix_vfedge=0.01
105
106 ! pop
107 use_int_method_2=.false.
108 use_tracers=.false.
109 num_mixreg=2 ! Number of material regions to be mixed (can be greater
    than num_mixmat)
110 mixmat_order(1)=1,2 ! User must specify order of materials from r=0 to rmax
111 dopop=.true.
112 pop_dt=5.0e-2
113 dodmpxdt=.true.
114 !pop_cycle=100

```

```

115 npop_prob=1
116 pop_prob(1)='mass'
117 npop_matl=6
118 !npop_matl=1
119 pop_matl(1)='tev'
120 pop_matl(2)='rho'
121 pop_matl(3)='sie'
122 pop_matl(4)='mix_2'
123 pop_matl(5)='intpos_2_lo'
124 pop_matl(6)='intpos_2_hi'
125 npop_mesh=20
126 pop_mesh(1)='tev'
127 pop_mesh(2)='rho'
128 pop_mesh(3)='c01'
129 pop_mesh(4)='c02'
130 pop_mesh(5)='ten'
131 pop_mesh(6)='a_1'
132 pop_mesh(7)='mxb'
133 pop_mesh(8)='tsc'
134 pop_mesh(9)='ydt'
135 pop_mesh(10)='prs'
136 pop_mesh(11)='snd'
137 pop_mesh(12)='a_2'
138 !JDS added the following pop_mesh
139 pop_mesh(13)='Rnn'
140 pop_mesh(14)='RS1'
141 pop_mesh(15)='RS2'
142 pop_mesh(16)='RS3'
143 pop_mesh(17)='RS4'
144 pop_mesh(18)='RS5'
145 pop_mesh(19)='RS6'
146 pop_mesh(20)='tvs'
147
148
149 !=====
150 ! SETUP
151 !=====
152 ! numfine = 7           ! See RAGE page
153 dxset = 0.125         ! size of zones in x and y (must be equal)
154 dyset = 0.125
155 imxset = 2           ! How many coarsest zones in x- and y-direction
156 jmxset = 2400
157 xzero = 0.0         ! minimum x and y values
158 yzero = -150.
159
160 ! resolutions
161 !!!!! l0 = 0.25
162 ! l1 = 0.125         ! 1000 micron
163 ! 2   0.0625
164 ! 3   0.03125
165 ! 4   0.015625     ! 156 micron
166 ! 5   0.0078125
167 ! 6   0.00390625
168 ! 7   0.001953125 ! 19 micron
169
170 !sizemat(1) = 2*0.001953125 ! Base level of zones for this material
171 !sizebnd(1) = 2*0.001953125 ! How fine to go at interfaces with other materials
172 sizerho(1) = 2*0.5         ! adapt if delta-rho is bigger than this

```

```

173 fvolpct = 0.00005      ! Change in fractional volume for use in determining if near
      interf
174
175 !=====
176 ! MATERIALS
177 !=====
178 keos = 0                ! keos = 0 means analytic (gamma-law) EOS
179 nummat = 2              ! How many distinct materials are there?
180
181 ! ideal gases can have pressure and temperature equilibrium if
182 ! (g1-1)*rho1*cv1*T = (g2-1)*rho2*cv2*T; ie. rho*cv = const
183
184 matdef(1,1) = 0.0        ! I forget
185 matdef(16,1) = 0.4       ! gamma - 1.0
186 matdef(30,1) = 10000.0  ! C_v for material
187
188 matdef(1,2) = 0.0
189 matdef(16,2) = 0.4       ! gamma - 1.0
190 matdef(30,2) = 3333.33333
191
192
193 !tevbca=9.93875
194 !tevbcb=10.1225
195
196
197 !=====
198 ! REGIONS
199 !=====
200 numreg = 2
201
202 matreg(1) = 1            ! ambient, low-density material
203 rhoreg(1) = 1.0
204 tevreg(1) = 2000.
205
206 matreg(2) = 2            ! overlying higher-density material
207 rhoreg(2) = 3.0         !need to change matdef(30,2) also need to change rho-hi above
208 tevreg(2) = 2000.
209
210 typreg(2) = 13          ! region two is above the interface calculated
211 fitreg(2) = 1
212 loreg(2) = 2
213 hireg(2) = 2
214 coefreg(1,2) = 0.0000000 ! .01 Interface perturbation (amplitude)
215 coefreg(2,2) = 1.0      ! Box size (wavelength)
216 coefreg(3,2) = 0.0      ! No phase offset to yint= c1*cos(2*pi*x/c2)
217
218 ! reference instructions for doing the linear stability problem
219 ! To properly run a Rayleigh-Taylor incompressible problem, one has to set the
220 ! initial conditions to values that are self-similar. Thus both the interface
221 ! position and velocity should be perturbed, and v0 = gamma*z0, where gamma**2 = k*g*A
222 ! with k = (2*pi/L), g= gravity, and A=Atwood# = (rhoHi-rhoLo)/(rhoHi+rhoLo)
223 ! if box is size one, k = 2*pi. If g = 100/2pi =aprox= 15.915, then gamma**2 = 10*sqrt(
      A)
224 ! if Atwood = 0.36 (2.125:1), then gamma = 6.0
225 ! AND, one wants the interface perturbation to be small (imperceptible) wrt box size
226
227 ! lambda      k          g          A          gamma      a0          t          amr lev  yanaly  ycalc
228 ! -----      ----      -----      -----      -----      ----      ----      -----      -----      -----
229 ! 1.0          2pi       100/2pi     .36         6           .01         .038         .023

```

```

230 ! 1.0 2pi 10000/2pi .36 60 .025 .019 5 .103 .06-.08
231
232 ! This is the run in the repository.
233 ! 1.0 2pi 10000/2pi .36 60 .025 .019 3 .103
234
235 rtttype = 3 ! prestress and set velocities to self-similar initial conditions
236 rt_num=1
237 rt_coefa(1)=0.0 !0.00 ! v0 = gamma*z0, and coefreg(1,2) = 0.01
238 rt_coefb(1)=1.0 ! so k = 2*pi/1.0 . If box is not unit sized, rt_coefb=size
239
240
241 !=====
242 ! Gravity/ Prestress
243 !=====
244 grav_y = -1000.00 ! cm/s**2 ==> growthrate=60 /sec
245 do_grav_prestress = .false. ! neither 1d-x or 2-d spherical
246 !
247 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
248 !ncmax = 15 ! Maximum number of cycles (timesteps) to run
249 sizemat(1)=2*0.0625
250 sizebnd(1)=2*0.0625
251 !kread = 10 ! Does nothing at -1, other #'s determine cycle to restart

```

## 8 Appendix B: Post-processing script

```
1 clear all
2 close all
3 more off
4
5 plotz = true;
6
7 rho_min = 1;
8 rho_max = 3;
9
10 locspec = 0.01;
11
12 # 1 --- z/h(t)
13 # 2 --- tke
14 # 3 --- a
15 # 4 --- mxb
16 # 5 --- R11
17 # 6 --- R33
18 # 7 --- rho (does not exist for DNS?)
19
20 # read in the DNS data
21 load -ascii "dns_data.csv"
22
23 zoht_dns = dns_data(:, 1);
24 tke_dns = dns_data(:, 2);
25 a_dns = dns_data(:, 3);
26 mxb_dns = dns_data(:, 4);
27 R11_dns = dns_data(:, 5);
28 R33_dns = dns_data(:, 6);
29
30 tke_smo = supsmu(zoht_dns, tke_dns);
31 a_smo = supsmu(zoht_dns, a_dns);
32 mxb_smo = supsmu(zoht_dns, mxb_dns);
33 R11_smo = supsmu(zoht_dns, R11_dns);
34 R33_smo = supsmu(zoht_dns, R33_dns);
35
36 tke_smoindx = find(tke_smo > (locspec * (max(tke_smo) - min(tke_smo)) + min(tke_smo)));
37 mxb_smoindx = find(mxb_smo > (locspec * (max(mxb_smo) - min(mxb_smo)) + min(mxb_smo)));
38 a_smoindx = find(a_smo > (locspec * (max(a_smo) - min(a_smo)) + min(a_smo)));
39
40 tke_smoindxmin = min(tke_smoindx);
41 mxb_smoindxmin = min(mxb_smoindx);
42 a_smoindxmin = min(a_smoindx);
43
44 tke_smoindxmax = max(tke_smoindx);
45 mxb_smoindxmax = max(mxb_smoindx);
46 a_smoindxmax = max(a_smoindx);
47
48 if plotz == true
49     figure(1)
50     plot(zoht_dns, tke_dns, 'b')
51     hold on
52     plot(zoht_dns, tke_smo, 'r')
53
54     figure(2)
55     plot(zoht_dns, mxb_dns, 'b')
56     hold on
```

```

57  plot(zoht_dns, mxb_smo, 'r')
58
59  figure(3)
60  plot(zoht_dns, a_dns, 'b')
61  hold on
62  plot(zoht_dns, a_smo, 'r')
63
64  figure(4)
65  plot(zoht_dns, R11_dns, 'b')
66  hold on
67  plot(zoht_dns, R11_smo, 'r')
68
69  figure(5)
70  plot(zoht_dns, R33_dns, 'b')
71  hold on
72  plot(zoht_dns, R33_smo, 'r')
73  end
74
75  for i = [45]
76      filename = ['data' num2str(i) '.csv']
77
78      # read in the RAGE data
79      copyfile(filename, "tempcsv.csv");
80      load -ascii "tempcsv.csv"
81      unlink("tempcsv.csv");
82
83      z = tempcsv(:, 1);
84      tke = tempcsv(:, 2);
85      a = tempcsv(:, 3);
86      mxb = tempcsv(:, 4);
87      R11 = tempcsv(:, 5);
88      R33 = tempcsv(:, 6);
89      rho = tempcsv(:, 7);
90      rho_older = tempcsv(:, 8);
91
92      # scale the RAGE data
93      f_v = (rho - rho_min) / (rho_max - rho_min);
94
95      integrand = f_v .* (1 - f_v);
96
97      ht = 6 * trapz(z, integrand);
98
99      scale = ht^2 / (ht / 500);
100
101      zoht = z / ht;
102      tke = 1e16 * tke / scale;
103      a = a / sqrt(scale);
104      mxb = mxb;
105      R11 = 1e16 * R11 / scale;
106      R33 = 1e16 * R33 / scale;
107
108      #matout = [zoht tke a mxb R11 R33 rho rho_older];
109      #size(matout)
110
111      #unlink("output.csv");
112      #save -ascii "output.csv" matout
113
114      # earlier (t = 2.2 s) mix width
115      f_v = (rho_older - rho_min) / (rho_max - rho_min);

```

```

116
117     integrand = f_v .* (1 - f_v);
118
119     ht_older = 6 * trapz(z, integrand);
120
121     #t_new = 2.0;
122     #t_older = 1.7;
123     #t_new = 2.5;
124     #t_older = 2.2;
125
126     t_new = 2.1;
127     t_older = 2.0;
128
129     # calculate growth rate
130
131     alpha_dns = 0.0416; # for A = 0.5 (see p. 22 of Schwarzkopf, et al. (2011))
132     alpha = ((sqrt(ht) - sqrt(ht_older)) / (sqrt(500) * (t_new - t_older))) ^ 2;
133
134     errvec = [];
135
136     % peak-to-peak
137     errvec(1) = abs(max(abs(tke)) - max(abs(tke_smo))) / abs(max(abs(tke_smo)) + max(abs(
        tke)));
138     errvec(2) = abs(max(abs(mxb)) - max(abs(mxb_smo))) / abs(max(abs(mxb_smo)) + max(abs(
        mxb)));
139     errvec(3) = abs(max(abs(a)) - max(abs(a_smo))) / abs(max(abs(a_smo)) + max(abs(a)));
140
141     % integral
142     errvec(4) = abs(trapz(zoht, tke) - trapz(zoht_dns, tke_smo)) / abs(trapz(zoht, tke) +
        trapz(zoht_dns, tke_smo));
143     errvec(5) = abs(trapz(zoht, mxb) - trapz(zoht_dns, mxb_smo)) / abs(trapz(zoht, mxb) +
        trapz(zoht_dns, mxb_smo));
144     errvec(6) = abs(trapz(zoht, a) - trapz(zoht_dns, a_smo)) / abs(trapz(zoht, a) + trapz(
        zoht_dns, a_smo));
145
146     % calculate the indices of the transition locations
147     tkeindx = find(tke > (locspect * (max(tke) - min(tke)) + min(tke)));
148     mxbindx = find(mxb > (locspect * (max(mxb) - min(mxb)) + min(mxb)));
149     aindx = find(a > (locspect * (max(a) - min(a)) + min(a)));
150
151     tkeindxmin = min(tkeindx);
152     mxbindxmin = min(mxbindx);
153     aindxmin = min(aindx);
154
155     tkeindxmax = max(tkeindx);
156     mxbindxmax = max(mxbindx);
157     aindxmax = max(aindx);
158
159     % width
160     errvec(7) = abs(1 - (zoht(tkeindxmax) - zoht(tkeindxmin)));
161     errvec(8) = abs(1 - (zoht(mxbindxmax) - zoht(mxbindxmin)));
162     errvec(9) = abs(1 - (zoht(tkeindxmax) - zoht(tkeindxmin)));
163
164     % growth rate
165     errvec(10) = 5 * abs(alpha_dns - alpha) / (abs(alpha_dns) + abs(alpha));
166     alpha
167
168     [i, sum(errvec)]
169     %errvec

```

```

170
171     if plotz == true
172         figure(1)
173         plot(zoht, tke, '0')
174         figure(2)
175         plot(zoht, mxb, '0')
176         figure(3)
177         plot(zoht, a, '0')
178         figure(4)
179         plot(zoht, R11, '0')
180         figure(5)
181         plot(zoht, R33, '0')
182     end
183 end
184
185 if plotz == true
186     figure(1)
187     title('TKE')
188     xlabel('zh/h(t)')
189     ylabel('Kh/λ2')
190     #print -dpng K.png
191
192     figure(2)
193     title('b')
194     xlabel('zh/h(t)')
195     ylabel('b')
196     #print -dpng b.png
197
198     figure(3)
199     title('a')
200     xlabel('zh/h(t)')
201     ylabel('ah/λ')
202     #print -dpng a.png
203
204     figure(4)
205     title('R11')
206     xlabel('zh/h(t)')
207     ylabel('R11h/λ2')
208     #print -dpng R11.png
209
210     figure(5)
211     title('R33')
212     xlabel('zh/h(t)')
213     ylabel('R33h/λ2')
214     #print -dpng R33.png
215
216     hold off
217 end

```

**Development of an ICF Mix Code**

**(Erik Vold, mentor)**

# Development of ICF Mix Code

Jeremy A. Melvin, Sean T. Miller

August 15, 2012

## Abstract

Our project over the summer of 2012 was to develop an ICF mix code. The code we developed is a simplified arbitrary Lagrange Eulerian (ALE) method written from scratch in C++ and is designed to run on unstructured grids. We have performed initial verification of our code, comparing to analytical solutions of a Sod-like shock tube problem and the Sedov blast wave problem. The code was then extended to 2D and verified with respect to the 1D solutions. We also show the capability of our code to handle turbulent instabilities, such as Richtmyer-Meshkov (RM), which are commonplace within ICF experiments. Finally we present an ICF-like initial condition setup in our 1D Lagrange Hydro code.

## 1 Introduction

For the computational student summer workshop of 2012 our project was to develop a numerical method to accurately model inertial confinement fusion (ICF) implosions where the fuel and ablation material interface instabilities can occur. We were advised to construct a traditional hydro numerical method well established at the Los Alamos National Lab, called the Arbitrary Lagrangian Eulerian (ALE) method. This method is well known for modeling solids, and we were interested in utilizing it for modeling high energy density materials.

Inertial confinement fusion is one of the more simple forms of nuclear fusion. The concept is to compress a quantity of fusible material, such as a mixture of deuterium and tritium, to the point where the energy density and mass density allow ions to fuse. The most common compression methods in practice today are ablation implosion methods using lasers and ion beams. Fusion becomes viable when the compression of the fuel fulfills the Lawson criteria. The Lawson criteria estimates the required confinement time of a fusion device for energy to be produced and is a ratio of the energy content of a system to its power loss [2]:

$$n\tau_c = \frac{12k_bT}{E_{ch}\langle\sigma v\rangle} \quad (1)$$

where  $n$  is the number density of the fusing material,  $\tau_c$  is the confinement time,  $k_bT$  is the thermal energy of the system,  $E_{ch}$  is the energy added to the system by a single fusion event, and  $\langle\sigma v\rangle$  is an averaged quantity used to express the probability of a fusion event occurring.

For inertial confinement fusion, this relation can be simplified since the confinement time is known to be of the order of a characteristic length scale of the pellet divided by a characteristic ion flow speed in the medium:

$$\tau_c \geq \frac{r_{pellet}}{\sqrt{\frac{k_bT}{m_i}}} \quad (2)$$

where  $r_{pellet}$  is the radius of the pellet and  $m_i$  is the mass of the ion species. For a deuterium-tritium reaction the criteria can then be simplified to [2]:

$$\frac{M_{pellet}}{r_{pellet}^2} \geq 70 \left[ \frac{\text{kg}}{\text{m}^2} \right] \quad (3)$$

or

$$\rho_{fuel} r_{pellet} \geq 30 \left[ \frac{\text{kg}}{\text{m}^2} \right] \quad (4)$$

where  $M_{pellet}$  is the mass of the pellet,  $\rho_{fuel}$  is the compressed density of the deuterium-tritium mixture, and  $r_{pellet}$  is the compressed radius of the pellet. From this one can expect that a milligram mass pellet of a deuterium-tritium mixture must reach 1000 times its initial density, or in terms of the radius, a 10 fold compression [2].

The design and materials used to construct a pellet is dependent on the driver. For pellets undergoing laser ablation implosions there are usually at least two layers. The outside layer is the ablation layer where the laser deposits energy causing the implosive shock. Ablation layers are usually made of a high density plastic such as polyethylene or polystyrene doped with various chemicals to prevent preheating of the fuel during the implosion. The internal layer is made up of the fuel mixture. A thin layer of metal can also be found surrounding the pellet to inhibit the fusing plasma's outward expansion during detonation.

Laser ablation comes in two flavors, direct and indirect. The first and older method is known as direct drive ICF where a collection of lasers are aimed at a pellet of fuel and pulsed to directly drive the implosion. This method has lost momentum over time as beam alignment and consistent illumination of the target is difficult with modern technology. The second and more modern method is known as indirect drive. Here the pellet is placed inside an empty cavity, known as a hohlraum. The laser is then pulsed into the cavity where it interacts with the interior walls to produce x-rays. These x-rays are then directed toward the surface of the pellet to drive the implosion. While the x-ray production of the hohlraum is theoretically able to produce a smoother illumination of the pellet's surface, experiments have shown that it is not yet enough to fuse the material.

This leads to the main issue with ICF: instabilities. Instabilities of the Richtmyer-Meshkov type are generated from drive asymmetries and interface perturbations between the pellet layers, which grow when the medium is shocked and reshocked. The turbulence generated from these instabilities redirects much of the implosion's energy away from igniting the fuel. Modeling this turbulence is important to further research into viable ICF experiments.

This leads to the need for a proper ICF mix code that can analyze the different physical plasma effects generated during an implosion. The most influential effect include thermal diffusion and plasma-field transport effects.

The purpose of our project was to setup a code that would be able to model these turbulent effects in 1, 2 and 3D accurately for high energy density plasmas.

## 1.1 Governing Equations

The Euler equations are the most basic form of fluid equations that can be used to describe an inviscid flow implosion. The equations model the conservation of mass, momentum and energy within the fluid. The equations can take many useful forms, but for our purpose we are interested in the convective derivative form [3]:

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \vec{u} + S_\rho \quad (5)$$

$$\rho \frac{D\vec{u}}{Dt} = -\nabla p + S_u \quad (6)$$

$$\rho \frac{De}{Dt} = -p \nabla \cdot \vec{u} + S_e \quad (7)$$

where  $\rho$  is the density of the fluid,  $e$  is the specific internal energy of the fluid,  $\vec{u}$  is the flow velocity of the fluid, and  $p$  is the fluid pressure. The source terms  $S_i$  are placeholders for plugins to the model such as diffusion and laser ablation, which are set to 0 if unused. For example the laser ablation source term is given by:

$$S_e = \frac{P_{laser}}{M_s} \quad (8)$$

where  $P_{laser}$  is the laser power and  $M_s$  is the total mass of all elements receiving energy from the laser. Sources are added appropriately at the end of a time step. The operator  $\frac{D}{Dt}$  is known as the convective or Lagrangian derivative and can be expressed in terms of normal derivatives as  $\frac{D}{Dt} = \frac{\partial}{\partial t} + \vec{u} \cdot \nabla$ . It describes an evolution of the quantity in the frame of the moving fluid. The pressure  $p$  is given by the ideal gas equation of state:

$$p = (\gamma - 1)\rho e \quad (9)$$

where  $\gamma$  is the adiabatic index or the ratio of the specific heat at constant volume to the specific heat at constant pressure. For a monatomic gas, such as a high energy density plasmas,  $\gamma = \frac{5}{3}$ . In order to model two fluids one must add another equation for the mass fraction  $\psi$  where the densities of the separate fluids are given by  $\rho_1 = \psi\rho, \rho_2 = (1 - \psi)\rho$ . The mass fraction evolves convectively as:

$$\frac{D\psi}{Dt} = S_\psi \quad (10)$$

Given this mass fraction, under mixing conditions within an element one must calculate the correct value for  $\gamma$ . This can be done by taking a heat capacity average within the element [6]:

$$\gamma = \frac{\psi\rho c_{v_0}\gamma_0 + (1 - \psi)\rho c_{v_1}\gamma_1}{\psi\rho c_{v_0} + (1 - \psi)\rho c_{v_1}} \quad (11)$$

where  $c_{v_0}$  and  $c_{v_1}$  are the specific heat capacities at constant volume for fluids 0 and 1 respectively.

## 2 Numerical Method

The numerical method used in this project is a simplified version of an arbitrary Lagrangian-Eulerian (ALE) method. A Lagrangian solver is used to move the fluid around by manipulating the mesh, while an Eulerian reconstruction stage follows to smooth tangles caused by the Lagrangian Phase. The idea behind the method is to mix the ease and accuracy of the Lagrangian solver with the robustness of an Eulerian solver. In our experience over the summer, we found it can be difficult to mix these two stages, as they can interact in interesting ways.

We decided early on to run the ALE method on unstructured meshes since neither of the methods explicitly required a structured grid. The Lagrangian solver is based on a staggered grid meaning the density and specific internal energy terms are defined as constant within the element, while the velocities are nodal values. In 1D this is simple to work with, however moving to higher dimensions required knowing much more interconnectivity of the mesh in order to efficiently update the nodal and elemental values. While the end product is more general and versatile, the unstructured grid made debugging far more difficult and greatly slowed down development.

ALE methods are usually made up of three phases. First comes a Lagrangian phase that updates the element and node quantities convectively with the mesh. The mesh is then analyzed and reconstructed based on the desires of the designers in the remesh phase. The remap phase then moves the values created in the Lagrangian phase to the newly constructed mesh.

To help the reader comprehend the complexity of each phase, we spent a single week of the ten weeks given designing and coding the Lagrangian phase for 1 and 2 dimensions. The remesh stage was very

simplified and only took a day. The rest of the time was spent coding the remap phase and resolving the interactions between the different phases. While the ALE method appears simple at first glance, starting from a fully Eulerian finite element or finite volume method may have been easier to complete in the allotted time.

The CFL condition for stability with this solver is given on an elemental basis. The shortest distance across the element  $l$  is divided by the square root of the total specific energy. This is approximated by the following condition:

$$dt = C \frac{l}{\tilde{c}_s + \tilde{v}} \quad (12)$$

where  $C$  is the cfl limiter, usually set to  $C < 1$ . For an ideal gas the approximate sound speed within an element  $\lambda$  is given by  $\tilde{c}_{s,\lambda} = \sqrt{\gamma \frac{p_\lambda + q_\lambda}{\rho_\lambda}}$ , where  $q$  is the artificial viscosity described in Section 2.2.2.

The approximate velocity term is given by  $\tilde{v} = \sqrt{2ke_\lambda}$  where  $ke_\lambda$  is the specific kinetic energy within the element. The specific kinetic energy of an element is given by the sum of the kinetic energies of the control volumes within an element divided by the mass of the element. See Section 2.2.1 for more information about control volumes.

## 2.1 Unstructured Grid

The ALE solver we developed is based upon an unstructured grid. While the solver was created to import a mesh from gmsh or cubit, for now it generates a structured mesh and utilizes it as an unstructured mesh. There are three components to an unstructured mesh in this solver: elements, faces and nodes. Any unstructured mesh can be described by node positions and the nodal connectivity of the elements. Faces are utilized to increase the speed of the remap phase.

Only two geometric quantities, volume and surface area, are used in the calculations within this solver. In order to capture cylindrical and spherical geometries in 1D, one only needs to modify the calculations for the area and the volume. In 2D areas are found as the distance between two points multiplied by the thickness of the block or cylinder  $h$ . Volumes are calculated by finding the surface area of the polygon of interest and multiplying it by the thickness of the cylinder or box. The volume of any polygonal prism extruded from the  $x - y$  plane is given by:

$$V = \frac{h}{2} \sum_{i=0}^n x_i y_{i+1} - x_{i+1} y_i \quad (13)$$

where  $y_{n+1} = y_0$  and  $n$  is the number of nodes describing the face. Boundary conditions can be difficult to implement for unstructured grids, especially when it is also a staggered grid. For this solver we successfully constructed null-flux boundary conditions by fixing the normal component of the Lagrange phase accelerations on the boundary nodes to zero. No-slip, null-flux boundary conditions are done by setting the acceleration to zero at the boundary. Neither of these methods require ghost cells. We also created null-element ghost cells where the internal values for density and specific internal energy were set to zero. This last boundary condition acted as a boundary with vacuum space, which is useful for modeling the outward motion of the ablation layer during an ICF implosion.

## 2.2 Lagrange Phase

The Lagrange phase used in this solver is based on the Pember and Anderson predictor corrector method found in [3]. The following method is designed to model elemental values  $\lambda$  and nodal values  $\xi$ :

1. Construct the initial artificial viscosities  $q^n$  for the elements. See Section 2.2.2
2. Construct the initial accelerations  $\vec{a}^n$  (see Section 2.2.1), and update the predicted node positions  $\vec{x}^*$  and the predicted velocities  $\vec{u}^*$ :

$$\bar{a}_\xi^n = \frac{\int \nabla(p^n + q^n)dV}{\int \rho^n dV} \quad (14)$$

$$\bar{u}_\xi^* = \bar{u}_\xi^n + \Delta t \bar{a}_\xi^n \quad (15)$$

$$\bar{x}_\xi^* = \bar{x}_\xi^n + \frac{1}{2} \Delta t (\bar{u}_\xi^n + \bar{u}_\xi^*) \quad (16)$$

3. Construct the predicted element volumes  $V^*$ , specific internal energies  $e^*$ , densities  $\rho^*$  and pressures  $p^*$ :

$$\rho_\lambda^* = \rho_\lambda^n \frac{V_\lambda^n}{V_\lambda^*} \quad (17)$$

$$e_\lambda^* = e_\lambda^n - (p_\lambda^n + q_\lambda^n) \left( \frac{1}{\rho_\lambda^*} - \frac{1}{\rho_\lambda^n} \right) \quad (18)$$

$$p_\lambda^* = (\gamma_\lambda - 1) e_\lambda^* \rho_\lambda^* \quad (19)$$

4. Construct the predicted node accelerations  $\bar{a}^*$ , the new positions  $\bar{x}^{n+1}$  and the new velocities  $\bar{u}^{n+1}$ :

$$\bar{a}_\xi^* = \frac{\int \nabla(p^* + q^n)dV}{\int \rho^* dV} \quad (20)$$

$$\bar{u}_\xi^{n+1} = \bar{u}_\xi^n + \frac{1}{2} \Delta t (\bar{a}_\xi^n + \bar{a}_\xi^*) \quad (21)$$

$$\bar{x}_\xi^{n+1} = \bar{x}_\xi^n + \frac{1}{2} \Delta t (\bar{u}_\xi^n + \bar{u}_\xi^{n+1}) \quad (22)$$

5. Construct the new element volumes  $V^{n+1}$ , specific internal energies  $e^{n+1}$  and densities  $\rho^{n+1}$ :

$$\rho_\lambda^{n+1} = \rho_\lambda^n \frac{V_\lambda^n}{V_\lambda^{n+1}} \quad (23)$$

$$e_\lambda^{n+1} = e_\lambda^n - \left( \frac{1}{2} (p_\lambda^n + p_\lambda^*) + q_\lambda^n \right) \left( \frac{1}{\rho_\lambda^{n+1}} - \frac{1}{\rho_\lambda^n} \right) \quad (24)$$

$$p_\lambda^{n+1} = (\gamma_\lambda - 1) e_\lambda^{n+1} \rho_\lambda^{n+1} \quad (25)$$

$$\psi_\lambda^{n+1} = \psi^n \quad (26)$$

Currently there are no diffusion terms to modify the specific internal energy and mass fraction. There are two unknowns stated above. The first is the integral quantity used in calculating accelerations and the second is the artificial viscosity  $q$ . Both of these are best understood by first defining a control volume for the node.

### 2.2.1 Nodal Control Volumes and Accelerations

The nodal control volume represents a portion of space in which  $\vec{u}$  is considered constant. Figure 1 shows how the control volume may look in 1D and 2D. In the simple 2D case expressed in the figure, the control volume is made up of 4 separate quadrilateral regions constructed of the central node position  $n_c$  two face midpoints  $\bar{p}_i$  and an element centroid  $\bar{c}_i$ . On more complicated unstructured grids however, the control volume can contain as many of these sub-element quadrilaterals as required.

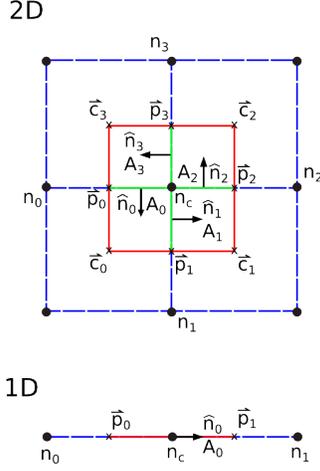


Figure 1: Nodal control volume used in code. Blue lines outline elements, red lines outline nodal control volume, green lines designate faces where forces are calculated,  $\vec{c}_i$  represent geometric centroids of elements,  $\vec{p}_i$  represent geometric midpoints between nodes.

The nodal acceleration calculation is the core of any Lagrange solver. Since the governing equations are inviscid at this time, there is no need to look into stress or viscosity tensors. The method we used can be shown in an example with respect to Figure 1 in 2D where we apply the divergence theorem to the acceleration integral quantity:

$$\vec{a}_\xi = \frac{\int \nabla(p^n + q^n)dV}{\int \rho^n dV} = \frac{\oint (p^n + q^n)\hat{n}ds}{\int \rho^n dV} \quad (27)$$

The integration is done over each section of the control volume separately on each face where there is a difference in  $p^n + q^n$ . Since the pressure and artificial viscosity are considered constant within each element, calculations are only done on faces between elements. In the case of Figure 1 this integral becomes:

$$\vec{a}_c = \frac{\sum_{i=0}^3 (p_{i-1} - p_i + q_{i-1} - q_i) A_i \hat{n}_i}{\sum_{i=0}^3 \rho_i V_i} \quad (28)$$

where  $p_{-1} = p_3$ ,  $A_i$  is the area of the control volume interface,  $V_i$  is the volume of the nodal control volume within each element, and  $\hat{n}_i$  is the counterclockwise norm of the interface.

## 2.2.2 Artificial Viscosity

Artificial viscosity is a means to convert kinetic energy into internal energy in order to add dissipation to a system. It acts in the same way that limiters act in purely Eulerian systems to enforce monotonicity in the solution. By adding the proper amount of dissipation, no new unwanted extrema and oscillations

are added to the system. There are a few conditions that must be fulfilled by the artificial viscosity which are explained very well in [1]. The most important of which is the artificial viscosity must only be non-zero in elements that are compressing, but only if the compression is non-uniform, e.g. normal to a shock front. The artificial viscosity used in this project is based on the edge centered artificial viscosity in [1]. To generate the artificial viscosity along an edge connecting two nodes one must first calculate the difference in velocity  $d\vec{v}_1$  and position  $d\vec{x}_1$  between the two nodes  $n_1$  and  $n_2$ :

$$d\vec{v}_1 = \vec{v}_2 - \vec{v}_1 \quad (29)$$

$$d\vec{x}_1 = \vec{x}_2 - \vec{x}_1 \quad (30)$$

$$dv_1 = |d\vec{v}_1| \quad (31)$$

$$d\hat{v}_1 = \frac{d\vec{v}}{dv_1} \quad (32)$$

$$dx_1 = |d\vec{x}_1| \quad (33)$$

$$d\hat{x}_1 = \frac{d\vec{x}}{dx_1} \quad (34)$$

The artificial viscosity only needs to be calculated if the edge is undergoing compression under the condition  $d\hat{x}_1 \cdot d\hat{v}_1 < 0$ . To test if there is a uniform compression we must construct a van leer limiter  $\phi_v$  by comparing this edge to those around it. If we have four nodes 0,1,2,3 interconnected with three edges then the van leer limiter is constructed in the following form:

$$r_0 = \frac{\frac{d\hat{v}_1 \cdot (\vec{v}_1 - \vec{v}_0)}{d\hat{x}_1 \cdot (\vec{x}_1 - \vec{x}_0)}}{\frac{dv_1}{dx_1}} \quad (35)$$

$$r_1 = \frac{\frac{d\hat{v}_1 \cdot (\vec{v}_3 - \vec{v}_2)}{d\hat{x}_1 \cdot (\vec{x}_3 - \vec{x}_2)}}{\frac{dv_1}{dx_1}} \quad (36)$$

$$\phi_v = \max(0, \min(\frac{r_0 + r_1}{2}, \min(2r_0, 2r_1))) \quad (37)$$

In our project it is assumed that the shocks are all traveling along the grid. This marks our main diversion from the method laid out in [1]. Instead of constructing the tensor form of the artificial viscosity at each edge and using it to update the nodal accelerations, we instead mass average the edge values for the artificial viscosities to create a cell centered artificial viscosity. Ideally, as expressed in [1] and [5], one must measure the compression along the shock direction within the element to generate a cell centered artificial viscosity. We tested both the tensor form and the element centered form, and found that as long as the shock is traveling along the grid, then the difference between the two solutions is minimal. In the future, this needs to be corrected such that the edge centered tensor form of the artificial viscosity is used to update both the node accelerations and the specific internal energy. In this project we used two different methods to calculate the scalar artificial viscosity quantity  $q$ . The first is the well known scalar monotonic artificial viscosity  $q_m$ , while the second is the more diffusive Kuropatenko viscosity  $q_k$  given in [1]:

$$q_m = \rho dv \left( \frac{1}{4}(\gamma + 1)dv(1 - \phi_v^2) + \frac{1}{2}c_s(1 - \phi_v) \right) \quad (38)$$

$$q_k = \rho dv(1 - \phi_v) \left( \frac{1}{4}(\gamma + 1)dv + \sqrt{\left( \frac{1}{4}(\gamma + 1)dv \right)^2 + c_s^2} \right) \quad (39)$$

Figure 2 shows the difference between  $q_m$  and  $q_k$  for a simple Sod shock tube problem. Also shown is what happens when artificial viscosity is not used, where it is apparent that the diffusion of the artificial viscosity kills off oscillations. The other figures within this paper all utilize the scalar monotonic artificial viscosity.

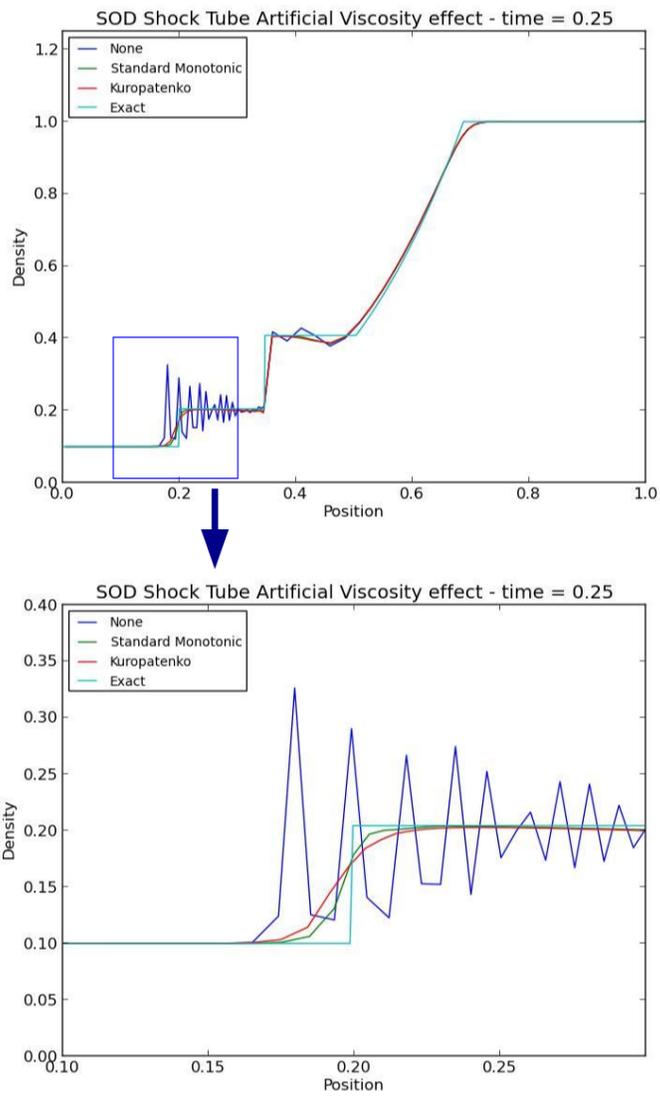


Figure 2: Comparison of the monotonic artificial viscosity, the Kuropatenko artificial viscosity, and no artificial viscosity to the exact solution with 100 elements.

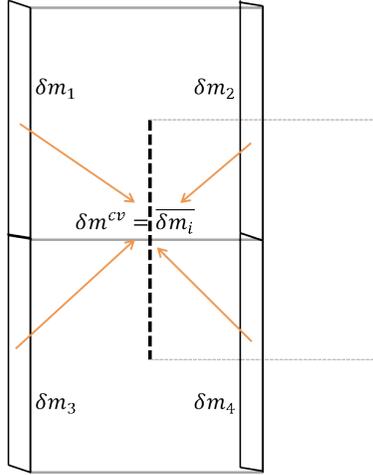


Figure 3: Control volume edge transport mass.

### 2.2.3 Remap Phase

In the Remap phase, we base our algorithm off of the one set forth in David Youngs' description of his TURMOIL3D ILES solver [6] and the Remap algorithm described in Pember and Anderson [3]. The method we use is a monotonically conservative advection based remap algorithm where you transport a state variable through a boundary face into the desired target cell. It uses a 3rd order Van Leer reconstruction of the face states, to determine the flux quantities [6].

The algorithm we use can be summarized as follows:

1. **Calculate the Cell Edge Transport Volumes ( $\delta V$ ).** A Transport Volume is defined as the volume created by connecting the nodes that define the face at the end of the Lagrangian step to their new positions under the original (or remeshed) grid. We take this volume to always be positive as it will be multiplied by a signed norm when its contribution to the cells that share the face's flux terms are calculated.
2. **Calculate the Cell Edge Transport Terms.** We flux across the cell faces the four quantities representing the four conserved cell centered values.
  - (a) Transport Mass:  $\delta m = \rho^f \delta V$
  - (b) Transport Partial Mass:  $\delta \psi = \psi^f \delta m$
  - (c) Transport Internal Energy:  $\delta e = e^f \delta m$
  - (d) Transport Kinetic Energy:  $\delta ke = ke^f \delta m$

The  $\Gamma^f$  terms above represent state values at the faces. The algorithm is independent of the method used to reconstruct the face states [3] and we choose here to use the 3rd order Van Leer reconstruction described in [6].

3. **Calculate the Control Volume Edge Transport Momentum.** We define the control volume edge transport mass to be the average cell edge transport mass for the four faces connected to the face in the perpendicular direction. See Figure 3 for a graphical depiction.

(a) Control Volume Transport Mass: 
$$\delta m^{cv} = \frac{1}{n} \sum_{faces} \delta m$$

(b) Transport Momentum: 
$$\delta u_i = u_i^f \delta m^{cv}$$

4. **Calculate the Cell Fluxes.**

A cell flux is the sum over the bounding faces of the cell of the transport term multiplied by a sign (+1 or -1). The purpose of the sign is to determine if it is an outward or inward flux and thus its contribution to the new cell value. We determine the sign through a vector projection of  $\vec{a}$  onto  $\vec{b}$  where  $\vec{a}$  is the vector pointing from the Lagrange grid's face center to the remap grid's face

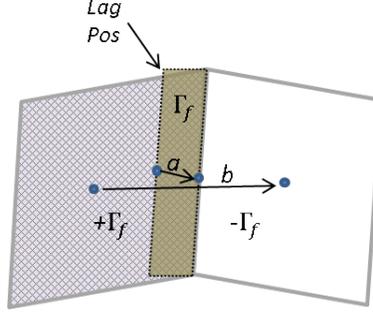


Figure 4: Cell flux contribution based on face positions.

center and  $\vec{b}$  is the vector pointing from the cell's centroid to the neighbor cell's centroid through the face in question. Figure 4 provides a graphical depiction of this. All we are concerned with is the sign of the projection and thus our sign can be calculated simply as  $\text{sign}(\vec{a} \cdot \vec{b})$ . We denote the total flux for a cell as  $F_m, F_{phi}, F_e$  and  $F_{ke}$  for mass, partial density, specific internal energy and specific kinetic energy respectively.

5. **Calculate the New Cell Values.** The new cell value is calculated via a simple advection equation. We take the Lagrangian conserved quantity and add the Cell Flux then divide by the necessary variables to turn the new remapped conserved quantity into the state value of interest. We use the remap of density as a typical example of this.

$$\rho_{remap} = \frac{V_{lag}\rho_{lag} + F_m}{V_{remap}} \quad (40)$$

6. **Calculate the Control Volume Fluxes.** This follows as a direct analog of 4 to the control volumes, with  $\vec{a}$  as the vector pointing from the Lagrange grid's face center to the remap grid's face center and  $\vec{b}$  as the vector pointing from the node's coordinate to the neighbor node's coordinate through the face in question.
7. **Calculate the New Node Values.** This follows as a direct analog of 5 with the control volumes associated with a node.

$$u_{i,remap} = \frac{cm_{lag}u_{i,lag} + F_{u_i}}{cm_{remap}} \quad (41)$$

Where we define  $cm$  to be the sum of the masses in each piece of the control volume for the given node.

8. **Enforce Total Energy Conservation\***. We denote this with an asterisk as we find that with strong shocks, enforcing full energy conservation causes large oscillations that can cause internal energy to break monotonicity and eventually become negative, thus crashing the algorithm. A thorough rethinking of this step from an analytical point of view, should allow for full energy conservation to be achieved even with strong shocks. For now, we instead define two types of energy conservation as follows:

- (a) **Full Conservation** In full conservation we enforce total energy ( $V\rho E = V\rho e + V\rho ke$ ) conservation by modifying the post-remap internal energy value based on the method described in [3]. We repeat the step here for completion.

$$kecalc_{remap} = \frac{1}{V_{remap}\rho_{remap}} \sum_{i \in nodes} \frac{1}{2}(\vec{u} \cdot \vec{u})cm_i \quad (42)$$

where  $cm_i$  = the mass of the piece of the control volume for the node that is contained within the element of interest and then

$$e_{remap} = e_{remap} + (ke_{remap} - kecalc_{remap}) \quad (43)$$

- (b) **Monotonicity Preserving Partial Conservation** In monotonicity preserving partial conservation of total energy, we follow the same method as above except that we cap the change in  $e_{remap}$  that can occur by ensuring that the change preserves the monotonicity of that element within its surrounding elements. Thus we identify the monotonic behavior of the element at the end of the Lagrangian stage and classify it into one of three cases.
- i. Element has the largest  $e_{lag}$  of all its neighbors:  
In this case we restrict the gain in energy of  $e_{remap}$  during the conservation step to not exceed the value of  $e_{lag}$ .
  - ii. Element has a bounded  $e_{lag}$  on either side by its neighbors:  
In this case we restrict the gain in energy of  $e_{remap}$  during the conservation step to be restricted between the max  $e_{remap}$  of its neighbors and the min  $e_{remap}$  of its neighbors, before they are adjusted during the conservation step.
  - iii. Element has the smallest  $e_{lag}$  of all its neighbors:  
In this case we restrict the loss in energy of  $e_{remap}$  during the conservation step to not fall below the value of  $e_{lag}$ .

Using this method, we find that the code is stable during strong shocks. In addition, we find that while from time step to time step energy is not conserved, it tends to fluctuate in both directions, usually within at most 1% – 2% of the total energy for the system.

## 2.2.4 Remesh Phase

The remesh phase used in this project was designed for remeshing ICF implosions where the shock travels primarily along the grid. The idea was to follow the Lagrangian updates to the node positions, but to smooth out the mesh by averaging  $x$  ( $r$ ) values and enforcing equally spaced  $y$  ( $\theta$ ) values. This would reserve the accuracy of the Lagrange phase while minimizing remap fluxes when running simulations on perturbed interfaces or with asymmetric drivers. This is the only section of the solver that requires a structured grid as a base to the unstructured grid in order to average the  $x$  ( $r$ ) values and properly space the  $y$  ( $\theta$ ) values. The remeshing stage can become slow for larger grids, and due to some constraints for the remap phase at this time, must be run every step if the option is turned on. This results in extra simulation time, but no remapping in steps before the shock passes through the discontinuity, preserving the ALE nature of the code.

# 3 Results

## 3.1 1D simulations

### 3.1.1 Shock Tube

The Sod shock tube problem in 1D is a very simple means of testing a hydro code. We ran a simple 1D shock tube with initial conditions:  $\gamma = 1.4$ ,  $\rho_{left} = .1$ ,  $\rho_{right} = 1$ ,  $x_0 = 0$ ,  $x_1 = 1$  with  $n = 100, 500, 1000$ . Figure 5 shows a comparison of the Lagrange phase only, while Figure 6 compares the Lagrange plus remap phase at time  $t = 0.25$ . Total energy conservation in the remap phase was used to generate these solutions.

As the figures show, the simulation seems to converge to a solution, and if we compare to an analytical solution using an  $L_1$  norm, as seen in Figure 7, we see that both the Lagrange only and Lagrange plus remap methods seem to converge at order 1. More analysis will need to be conducted to identify the cause of the first order convergence, since second order convergence would be expected in a predictor-corrector method.

### 3.1.2 Interacting Blast Wave

The interacting blast wave problem was our first foray into strongly shocked fluids. The initial conditions for this blast wave is given by:  $\gamma = 1.4$ ,  $\rho = 1$ ,  $p_l = 1000$ ,  $p_m = 0.001$ ,  $p_r = 100$ . The left interface is at  $x_l = 0.1$  and the right interface is at  $x_r = 0.9$ .

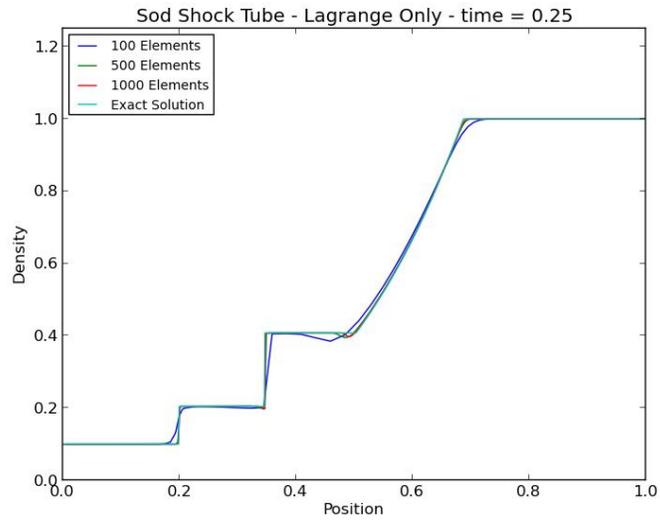


Figure 5: Comparison of 1D Sod shock tube solutions with Lagrange only.

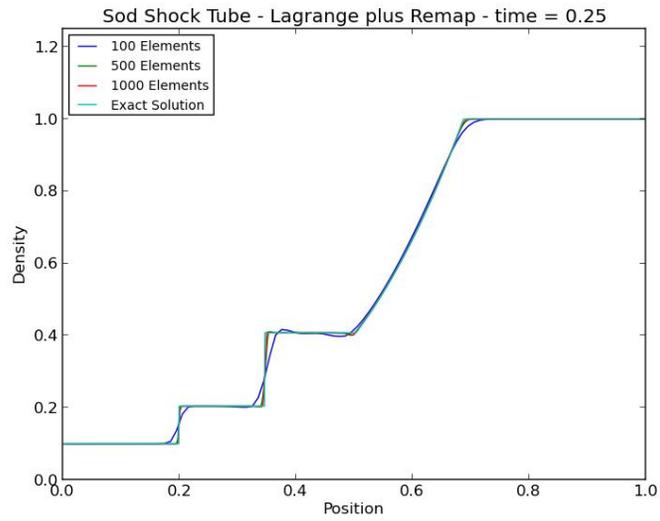


Figure 6: Comparison of 1D Sod shock tube solutions with Lagrange and remap phases.

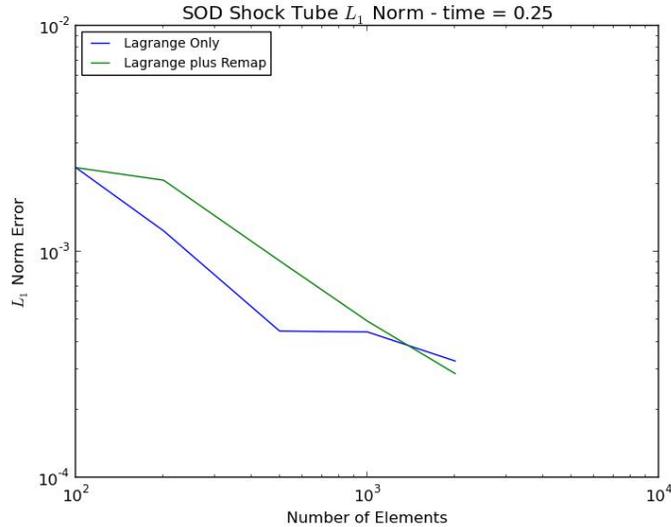


Figure 7: Verification of Sod shock tube problem in 1D for Lagrange only and Lagrange plus remap. Convergence rates appear to be of order 1.

Figure 8 and 9 shows the blast wave at time  $t = 0.038$  for  $n = 100, 500, 1000$  with  $\gamma = 1.4$  for Lagrange only and Lagrange plus remap respectively. Figure 10 shows the literature results from [3] which shows that the shocks are traveling and interacting appropriately. As our remap phase has issues with over compensating for total energy conservation, the remapped blast wave plots were run with partial energy conservation enabled.

### 3.1.3 Sedov Problem

The Sedov problem models a outward propagating wave of energy. The initial conditions for this problem is to set the specific energy in all elements to 0 and add enough specific internal energy in a single cell such that the total energy is  $E = 1$ . This means that the specific internal energy added to the element is dependent on the geometry and discretization of the grid. This problem was run in cylindrical and spherical geometry for  $n = 100, 500, 1000$  and  $\gamma = 1.4$ . The results at time  $t = 0.05$  are seen in Figures 11 and 12 for cylindrical and spherical geometries respectively.

## 3.2 2D Simulations

We use the 2D version of our ICF mix code to produce two-dimensional results in cartesian and cylindrical (polar) coordinates. In two-dimensions, we encounter new challenges that are not present in our 1D simulations. Most of these are resolved through careful extension of our 1D algorithm to multiple dimensions. We therefore discuss only the issue of potential mesh tangling, that arises when moving from 1D to 2D.

### 3.2.1 Mesh Tangling

In 1D simulations, we focus mainly on a strictly Lagrangian solver. This makes the most sense in 1D, as it is simplest algorithm and removes some of the numerical diffusion inherent to Eulerian methods. We present results for remapped simulations simply for verification purposes of the our remap

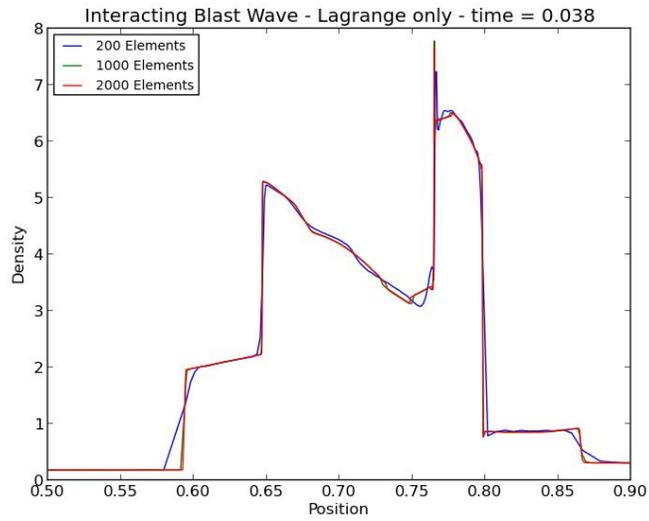


Figure 8: Comparison of 1D interacting blast wave with Lagrange only.

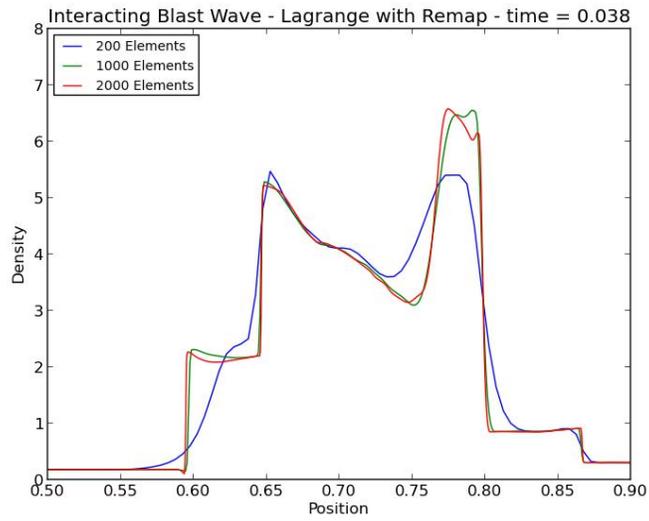


Figure 9: Comparison of 1D interacting blast wave with Lagrange plus remap.

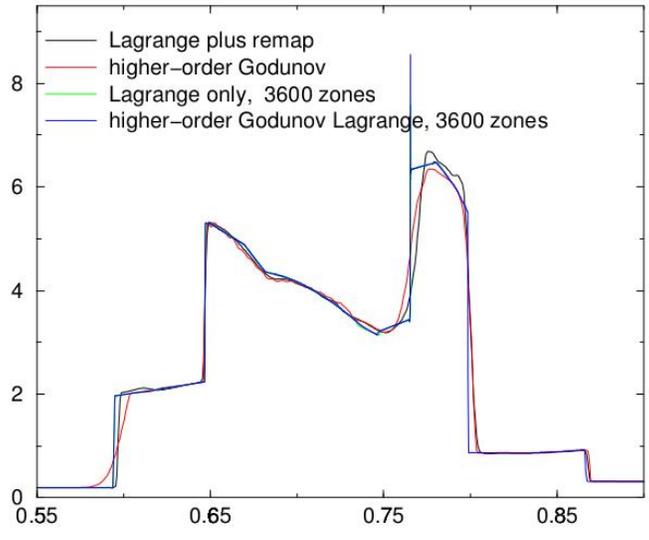


Figure 10: Literature values from [3] for interacting blast wave.

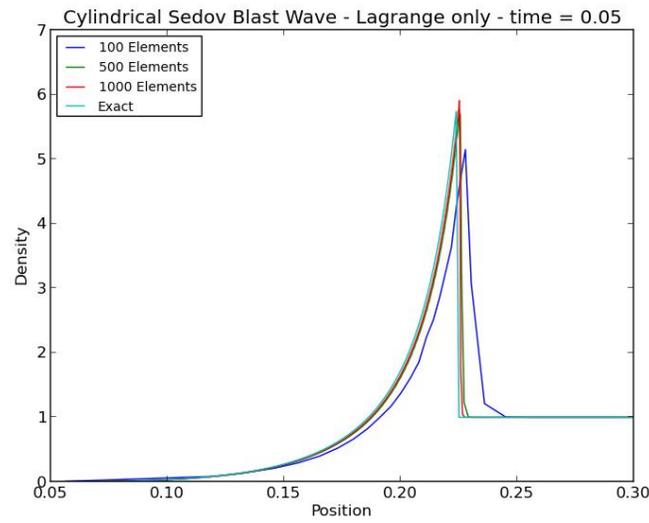


Figure 11: 1D Sedov problem for cylindrical geometry with Lagrange phase only compared to exact solution.

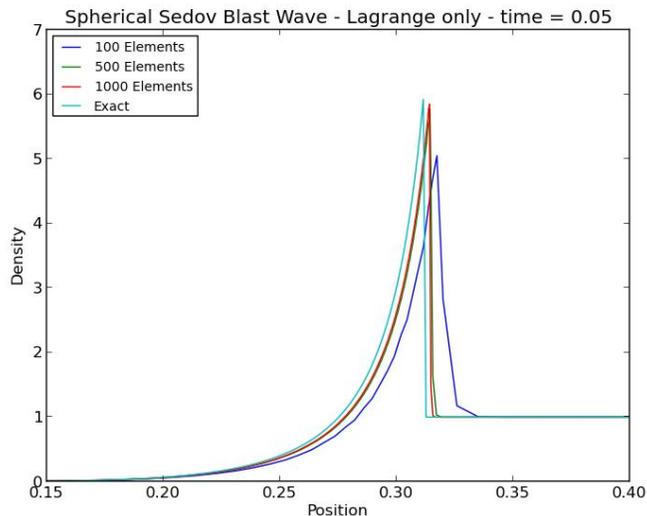


Figure 12: Literature values for 1D Sedov problem for spherical geometry with Lagrange phase only compared to exact solution.

algorithm. However, in 2D, with non symmetric initial conditions, such as those found in ICF experiments, using strictly Lagrangian methods results in severe mesh tangling, which causes the algorithm to fail. This is an expected phenomenon as the non-symmetric initial conditions lead to Richtmyer-Meshkov (RM) and Rayleigh-Taylor (RT) instabilities causing turbulent mixing. Therefore we utilize the remap and remesh algorithms to adjust the mesh at the end of the Lagrangian phase back to a more regular mesh. This allows us to avoid the issues associated with mesh tangling from a purely Lagrangian algorithm. We emphasize this point in Figure 13. Here we present a pressure balanced density discontinuity in 2D, with the more complicated addition of a perturbation in the interface. By passing a shock through the interface we generate an RM instability. In the top frame, we can see that the Lagrangian phase does indeed encounter mesh tangling and is not capable of completing the simulation. In the middle frame, we present the same initial conditions with the remap phase turned on. We show the mesh at the the same simulation time and see that we have no issues of mesh tangling and that the simulation can easily proceed past this point and complete. Finally, in the bottom frame, we show the same simulation at the same time, utilizing both the remap and remesh phases. The advantage of the remesh phase is to minimize the fluxes during the remap phase thus reducing numerical diffusion and increasing resolution around the shock interface, while also removing all potential tangling.

### 3.2.2 Verification of 2D Sod Shock Tube

We present the 2D extension of the 1D Sod Shock Tube problem presented in Section 3.1.1. This is a 2D problem which has symmetric initial conditions along the y-axis, thus should maintain perfect symmetry throughout the simulation and should mimic the 1D results. In Figure 14 we show both the 2D simulation at the  $t = 0.25$  for  $n_x = 2000, n_y = 20$  to emphasize the symmetry and a 1D projection of our 2D code compared to the exact solution. We see that we are able to correctly maintain the symmetry in the initial conditions as well as high accuracy.

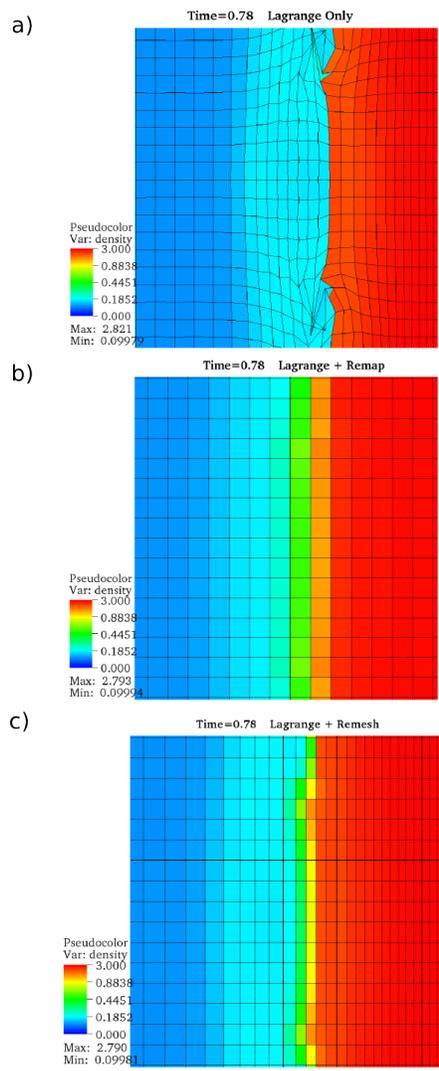


Figure 13: a) Example of a tangled mesh after a Lagrangian phase across a perturbation. b) Example a remap phase after the Lagrange phase back to the initial mesh. c) Example of remesh phase to an averaged mesh as described in Section 2.2.4.

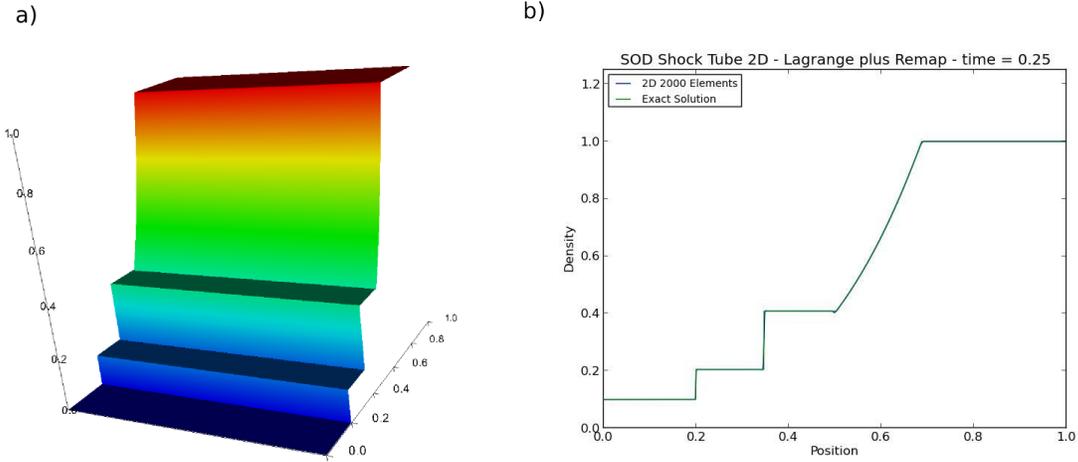


Figure 14: a) 2D sod shock wave with  $n_x = 2000, n_y = 20$  at time  $t = 0.25$ . b) Comparison of 2D solution projected on x axis to the exact solution.

### 3.2.3 RM Instability

We present an RM instability problem as run through our code. We first set up an initial perturbed density interface that is contained in pressure equilibrium. We then pass a shock through it, at which time an RM instability begins to grow. Bubbles (Light Density into Heavy Density) and Spikes (Heavy Density into Light Density) begin to form. The shock then bounces back and reshocks the instability growth, at which point the mixing starts to become chaotic. In Figure 15 we present an RM instability in Cartesian coordinates at various times during the simulation. We feel that this accurately mimics the proper growth behavior shown in other simulations and maintains proper symmetry for the majority of the time. The few anomalies in the growth behavior, we believe to be due to the coarse-grained definition of our initial perturbation. A more smoothly defined initial perturbation we believe will result in normal growth behavior. A more in depth study of our results would need to be conducted to determine if our growth rate agrees with that proposed by Richtmyer.

## 3.3 ICF

We present here a 2D hydro simulation with initial conditions that mimic an ICF implosion as prescribed in [4]. We use an internal mixture of deuterium-deuterium (DD) and an external species of Polyethylene (CH). The density of the outer plastic is  $\rho_{CH} = 931.7 \left[ \frac{\text{kg}}{\text{m}^3} \right]$  while the DD mixture has an initial density of  $\rho_{DD} = 3.344 \left[ \frac{\text{kg}}{\text{m}^3} \right]$ . The entire pellet is under pressure equilibrium at  $p = 2 \cdot 10^6$  [Pa]. We use a monatomic ideal gas  $\gamma = \frac{5}{3}$  and mimic the laser by adding energy into the last two cells of the domain smoothly over  $1 \cdot 10^{-9}$  [s]. The overall strength of the laser is set as  $P_{laser} = 20 \cdot 10^{12}$  [W] with an absorption factor of 20%, implying a total energy gain over the course of the laser activity of  $E_{laser} = 4 \cdot 10^3$  [J]. This creates a strong shock with a specific internal energy discontinuity of approximately  $10^7 : 1$ . This is run in a cylindrical 2D geometry with a domain size of  $0 \leq r \leq 0.425 \cdot 10^{-3}$  [m] and  $0 \leq \theta \leq \frac{\pi}{8}$ . All boundaries have reflective boundary conditions. In Figure ?? we show a 1D spherical ICF implosion as compared to a 1D lagrangian ICF implosion run in the HELIOS code [4]. We can see that even with a purely hydro algorithm, we can still obtain a compression factor of around 10 and that our results seem to mimic the timing that the HELIOS code shows. We believe that the addition of plasma physics models and other non-hydro requirements for ICF simulations will improve the accuracy of our ICF results.

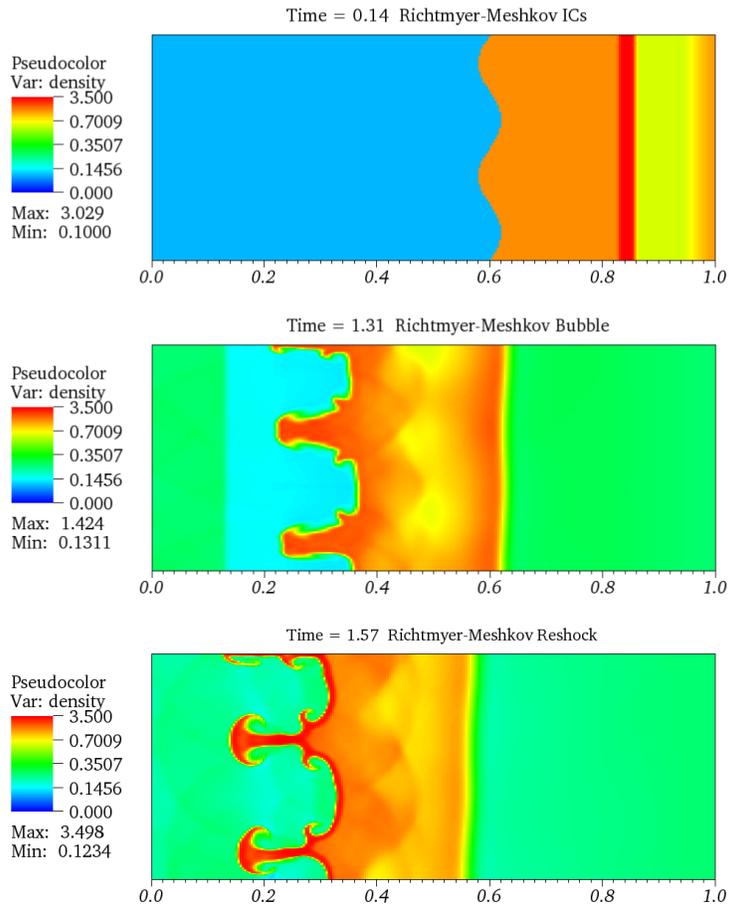


Figure 15: Top: Initial Conditions for an RM growth. Middle: Bubble and Spike development during the initially shocked interface. Bottom: Reshocked interface.

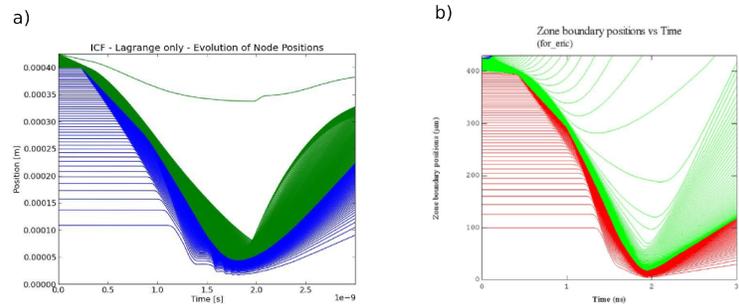


Figure 16: a) 1D spherical ICF Lagrange only. b) 1D spherical ICF from HELIOS code [4].

## 4 Conclusion

We began with a large potential scope of building an ICF mix code. Starting from scratch we built a staggered grid Lagrangian with remap hydro solver over an unstructured grid. We also built in a remesh step, which has the potential to expand our code into a full ALE algorithm. In addition, despite the added complication of an unstructured grid, we believe this gives our code a very generalized ability and thus can be extended to a lot of geometries outside of the typical ICF. We have performed initial verification of our code, comparing to analytical solutions of a Sod-like shock tube problem and a code comparison to the interacting blast wave problem described within [3]. We then extend our code to 2D and present verification as a comparison to the 1D problems described above. We also show the capability of our code to handle turbulent instabilities that are commonplace within ICF experiments and simulations. Finally, we show an ICF-like initial condition setup and run our hydro solver on this problem. We find that we have a solid hydro code base that is ready to be extended via plasma physics models and other ICF simulation requirements.

While we may have only briefly reached the ICF part of this project, we feel that our code and the algorithm described within this paper are a good jumping off point for plasma physics models, radiation, diffusion models and anything else that may be necessary to accomplish an accurate ICF simulation. In addition, there are still some undiagnosed bugs still within the hydro code we have written. In particular, we recommend looking into the artificial viscosity for multiple dimensions, the remesh algorithm and potential asymmetric numerical errors that occur within symmetric problems, as the main places of incompleteness in our Hydro code.

An observation we have made during this project are that a staggered-grid Lagrangian solver introduces a lot complication when it comes to conservation of energy. A quick search of the literature shows that there are many cell-centered Lagrangian methods out there and perhaps one of these would be a more successful implementation than our method. We do not disclose any particular references, as we have not done this research ourself. While unstructured grids are beneficial when dealing with complicated geometries, for this project, with its scope of only ICF mixing, this was probably an unnecessary step. While it is certainly beneficial now that it has been implemented, we figure the extra time spent on this aspect, prevented us from getting to a lot of the ICF plasma physics modeling. Our code has a good structure for allowing multi-threading and parallelization. This should be a priority if this code is going to be utilized further, as any realistic grid sizes will need a parallel option. As for total energy conservation, we believe this issue to be a byproduct of the staggered grid approach which leaves us with different control volumes for kinetic and internal energies. If the code is modified to have cell centered velocities or a lot of care is put into the conservation step, we believe that this issue is resolvable.

## Acknowledgments

We would like to thank our mentor Dr. Erik Vold for his advice and help in designing the solver. We would also like to thank Dr. Scott Runnels for setting up this workshop and running it so smoothly. Finally we would like to thank UNM and the Los Alamos Nation Lab for hosting the workshop.

## References

- [1] M. J. Shashkov E. J. Caramana and P. P. Whalen. Formulations of artificial viscosity for multi-dimensional shock wave computations. *Journal of Computational Physics*, 144:70–97, 1998.
- [2] Denis Keefe. Inertial confinement fusion. *Annu Rev. Nucl. Part. Sci.*, 32:391–441, 1982.
- [3] R. B. Pember and R. W. Anderson. A comparison of staggered-mesh lagrange plus remap and cell centered direct eulerian godunov schemes for eulerian shock hydrodynamics. *NECDC*, 2000.
- [4] Erik Vold and Leslie Welser-Sherrill. Momentum transport and associated scale lengths in an icf plasma. *Presented at: 2012 HED Plasma and Fluids*, 2011.

- [5] Mark L. Wilkins. Use of artificial viscosity in multidimensional fluid dynamic calculations. *Journal of Computational Physics*, 36:281–303, 1980.
- [6] David L. Youngs. The lagrangian remap method. *Implicit Large Eddy Simulation - Computing Turbulent Fluid dynamics*, pages 147–153, 2007.

**Verification of Shocks in Plasmas**  
**(Tom Masser, mentor)**

# Verification of Planar Shocks through Dense Plasma

William A. Hoey

10 August 2012

## Abstract

In this study, a set of two-temperature models for planar, steady-state shock waves in fully-ionized plasmas were addressed as potential verification cases for a key laboratory multiphysics code, xRage. The linear model of Masser, Wohlbier, and Lowrie [2011] and the nonlinear model of Jaffrin and Probstein [1964] were employed. Each model reduces to pair of coupled, autonomous ordinary differential equations, *i.e.* a dynamical system. The models are differentiated by their simplifications; most significantly, the linear model incorporates constant parameters for the electron-ion coupling term,  $\gamma_{ei}$ , and the thermal conductivity term  $\kappa_e$ . The nonlinear model allows for variable  $\gamma_{ei}$  and  $\kappa_e$  as functions of density and electron temperature, such that a unique solution exists for a given initial state. Integration codes for both models were written with adaptive-stepping versions of the explicit Euler, fourth-order Runge-Kutta, and fourth-order Adams-Bashforth numerical methods, and a wide variety of initial parameters were evaluated. Of particular interest were the boundaries in parameter space between discontinuous and continuous solutions; for the nonlinear model as a function of  $M_0$  alone, and for the linear model as a function of a dimensionless parameter  $R \propto \kappa_e \gamma_{ei}$  in addition to the ion charge state and initial Mach number. The relations employed for conductivity and coupling terms by the nonlinear reduced model and within xRage were addressed in depth. Ultimately, results were compared between the nonlinear and linear models, and to the evolved steady-states of inherently transient solutions within xRage. Expressly considered were the phase space diagrams from each model, as well as the plots of shock wave temperature versus displacement and the parameter space transitions from continuous to discontinuous solutions.

# Chapter 1

## Introduction

Studies of verification and validation are useful in measuring the performance of a piece of code, and in quantifying its success in satisfying its stated purposes. Here we consider the process of verifying **xRage**, a flagship laboratory hydrodynamics code, in a case of multi-temperature plasma physics. In verification, the solutions produced by **xRage** for a test case are compared against semi-analytic solutions of coupled, autonomous systems of ordinary differential equations that comprise reduced models of the code's three-temperature plasma physics package. The goal of this comparison was to determine whether **xRage** was properly solving its set of governing equations for such flows.

Two reduced models were examined: that of Jaffrin and Probstein[2] (released in 1964), and that of Masser, Wohlbier, and Lowrie[4] (released in 2011). These models originated from differing sets of governing partial differential equations, but followed a similar process of reduction to arrive at their final forms. That reduction process is discussed in the second chapter of this report.

The test problem addressed by **xRage** was the three-temperature Shafranov problem in one dimension. In this problem, either end of the domain is set to its corresponding equilibrium condition – the known upstream and downstream values – and a Heaviside step function produces a discontinuous transition between those equilibria at a point in the domain. There follows a relaxation in time to the steady-state solution.

The reduced models began as sets of partial differential equations in time and space, but were stripped of time dependence in an effort to transition into a Galilean reference frame moving at the shock velocity. This allowed their solution as ordinary differential equations for steady-state shock conditions. Therefore, the reduced models ought to produce representations of the equilibrium state to which the **xRage** solution will relax in time.

### 1.1 Models

- *1-D Galilean Steady-State: Jaffrin & Probstein [1964]. 'Nonlinear.'*
  - For Mach numbers below 2.0, this is two-layer model incorporating a large-scale thermal shock and a small-scale ion shock. In this analysis, the large-scale shock is of most significance, as its reductions are applicable across the vast majority of the domain. The ion shock model concerns the resolution of features across the shock face, on the order of a few mean free paths in thickness.
  - Each layer reduces to two ordinary differential equations, from the original set of five partial differential equations in electron and ion variables.
  - The electron-ion coupling coefficient and the electron thermal diffusivity,  $\gamma_{ei}$  and  $\kappa_e$  respectively, are treated as functions of density and electron temperature. This results in a unique solution for a given upstream (initial) state, specified by Mach number.
  - The flow's characteristic ion charge state,  $Z$ , is assumed to be 1.

- *1-D Galilean Steady-State: Masser, Wohlbier, & Lowrie [2011]. ‘Linear.’*
  - This is a one-layer model composed of two ordinary differential equations, likewise drawn from a governing set of PDEs. Ion temperatures across the shock face are taken as discontinuous.
  - The electron-ion coupling coefficient and the thermal diffusivity are taken as free parameters. They are linked by a dimensionless ratio of length scales,  $R$ , which is proportional to their product.
  - Furthermore, the model is expanded to incorporate ion charge states beyond that for monatomic hydrogen;  $Z$  is likewise a free parameter.
  - Therefore, solutions are dependent on the choice of  $Z$ ,  $\gamma_{ei}$ , and  $\kappa_e$  in addition to the upstream Mach number.
- *1-D Fully Transient: xRage*
  - This multiphysics package integrates its governing PDEs without the simplifications of the two reduced models. A wide range of materials, and thus ion charge states, are available for study. There are expressions of varying complexity available for representing both  $\gamma_{ei}$  and  $\kappa_e$ , including the choice of constant coefficients. Solutions are by default transient, and so comparisons to the reduced models are conducted after the shock has relaxed to its steady-state structure.
  - As this model is not formulated in dimensionless units, it is necessary to specify an initial temperature and initial density (or number density) in addition to the Mach number. Initial velocity is also required, but may be calculated from those parameters given the equations of state employed in the reduced models.

## 1.2 Solution of the Reduced Models

### 1.2.1 Masser, Wohlbier, and Lowrie [Linear]

A pair of coupled and autonomous equations, in dimensionless electron temperature and dimensionless density, were derived from the governing set of partial differential equations. That set was comprised of Euler conservation relations for mass and momentum, in addition to an energy equation with an electron conduction term, and an electron advection equation with coupling and conduction terms. The independent variable in each reduced ordinary differential equation was the scaled displacement. Integration was performed with explicit techniques, namely the Euler method and variable-stepping versions of the fourth order Runge-Kutta and Adams-Bashforth methods. Based on the potential for discontinuous solutions, the process of integration differed with the initial Mach number.

First, downstream conditions were calculated from the initial, upstream values. Critical Mach numbers were then calculated, bounding the region in parameter space in which either continuous or discontinuous solutions might occur – in which neither the presence nor absence of embedded hydrodynamic shocks could be guaranteed. Initial Mach numbers that were less than the lower bound of that region are known to produce continuous solutions in all cases, while Mach numbers above the high bound are likewise expected to produce discontinuous solutions in all cases. However, Mach numbers bounded by the critical values produce solutions of indeterminate form. As this model allows for the user input of its parameters, establishing an exceptionally high value for electron-ion coupling effectively binds the two constituent temperatures together and promotes continuity at high Mach numbers.

Low Mach number solutions, those guaranteed continuous, were simplest to solve. It was demonstrated in the 2011 paper that the initial, upstream equilibrium point was a saddle in phase space for all Mach numbers; the downstream point was shown, however, to be an attractor for Mach numbers below the lower-bound critical value. Integration in phase space must proceed outward from a saddle, as to integrate into a saddle point is to invite numerical error to rapidly dominate the solution.

## Chapter 2

# Derivation of a Reduced 2T Model

The derivations of the linear and nonlinear models are analogous, barring the treatment of variable  $\kappa_e$  and  $\gamma_{ei}$  terms in the nonlinear model. Here, we demonstrate the reduction of a set of governing equations for one-dimensional, fully-ionized, single-material flow in the linear case. A system of partial differential equations in time and space is proposed:

$$\partial_t \rho + \partial_x(\rho u) = 0, \quad (2.1)$$

$$\partial_t(\rho u) + \partial_x(\rho u^2 + P) = 0, \quad (2.2)$$

$$\partial_t(\rho E) + \partial_x[u(\rho E + P)] = \partial_x(\kappa_e \partial_x T_e), \quad (2.3)$$

$$\partial_t(\rho e_e) + \partial_x(\rho u e_e) + P_e \partial_x u = \gamma_{ei}(T_i - T_e) + \partial_x(\kappa_e \partial_x T_e). \quad (2.4)$$

Here,  $E$  denotes total specific material energy, as  $E = e + \frac{u^2}{2}$ . Represented in these equations are the familiar inviscid Euler expressions for conservation of mass and momentum, in addition to a form of the energy equation that admits an electron heat flux  $q_e = -\kappa_e \partial_x T_e$ . The final equation models electron advection; after application of fundamental thermodynamic relations, it resolves to the form

$$\rho T_e \frac{D(S_e)}{Dt} = \gamma_{ei}(T_i - T_e) + \partial_x(\kappa_e \partial_x T_e), \quad (2.5)$$

in which  $S_e$  represents the specific electron entropy. It is important to note that, contrary to the electron temperature,  $S_e$  may be discontinuous across shock surfaces. In the limit as  $\gamma_{ei}$  and  $q_e$  approach zero, we assume that the electron entropy jump also goes to zero.

### 2.1 Equation of State

Both the linear and nonlinear models assume an ideal gas equation of state:

$$P = \rho e(\gamma - 1), \quad (2.6)$$

$$e = e(\rho, T). \quad (2.7)$$

This allows us to make certain assumptions as to the relations between ion and electron properties. These relations are dependent on the ion charge state  $Z$ , a measure of average particle ionization. Note that, in the derivation of the nonlinear model,  $Z$  is taken to be 1; this value corresponds to monatomic hydrogen. Both the linear model and **xRage** allow for the specification of ion charge state. The ‘Z-split’ of the equation of state occurs as follows, and is replicated by default in **xRage**’s 3T package;

$$P_e = \frac{Z}{Z+1} P(\rho, T_e), \quad (2.8)$$

$$e_e = \frac{Z}{Z+1} e(\rho, T_e), \quad (2.9)$$

$$P_i = \frac{1}{Z+1} P(\rho, T_i), \quad (2.10)$$

$$e_i = \frac{1}{Z+1} e(\rho, T_i). \quad (2.11)$$

## 2.2 Shock Structure

Anticipating the formation of embedded shocks between the upstream and downstream equilibria, we apply a Galilean transformation to the system of governing equations, by which the shock speed is set to zero. A standard set of Rankine-Hugoniot conditions apply across the jump[3][1]:

$$(\rho u)' = 0 \quad (2.12)$$

$$(\rho u^2 + P)' = 0 \quad (2.13)$$

$$((\rho E + P)u)' = \left(\kappa_e \frac{dT_e}{dx}\right)' \quad (2.14)$$

An additional expression for jump conditions across the shock is required, since our model adds considerations of electron heat conduction and advection to the traditional equations. We can assume an isentropic jump in electron quantities at embedded hydrodynamic shocks.

Following the analysis of Zel'dovich and Raizer, the ideal gas law may be combined with the steady-state, Galilean invariant versions of the governing equations to yield expressions for the temperature and derivative of the electron temperature. Ultimately, the electron specific heat at constant volume is taken to be constant, and an equation relating ion and electron temperatures is derived:

$$T_i - T_e = (Z+1)(T - T_e). \quad (2.15)$$

The jump conditions, governing PDEs, and equation of state may now be combined and non-dimensionalized. An expression is required to evaluate the electron entropy term, and  $S_e = C_{v,e} \log [P_e \rho^{-\gamma}]$  is employed. The system is finally reduced to two coupled and autonomous ordinary differential equations:

$$\frac{d\Theta_e}{d\zeta} = (1-\eta)(\eta - \eta_1), \quad (2.16)$$

$$\frac{d\eta}{d\zeta} = -\eta R \frac{Z+1}{Z} \frac{\Theta_e - \Theta_e^{c2}}{\Theta_e - \Theta_e^1}. \quad (2.17)$$

The independent variable,  $\zeta$ , is proportional to displacement along the x-axis of the one-dimensional system. The dependent variable  $\eta$  represents the dimensionless ion density and velocity, as electron mass is neglected in this model; likewise,  $\Theta_e$  denotes the dimensionless electron temperature. The  $\Theta_e^{c1}$  and  $\Theta_e^2$  terms represent quadratic critical curves.

# Chapter 3

## Verification

### 3.1 Methods

- It can be shown that, for solutions to either reduced model without embedded hydrodynamic shocks, the upstream equilibrium point will always be a saddle point in phase space while the downstream equilibrium will be an attractor. Therefore, for continuous solutions, integration was performed from upstream to downstream equilibria to aid in numerical stability.
- For discontinuous solutions, or continuous solutions for which the initial Mach number is above a well-defined critical value, it is necessary to integrate outward from both the upstream and downstream equilibria, as each is a saddle point. In such cases, continuity in electron temperature is enforced via the jump conditions.
- It is of particular interest to examine solutions on the boundary between continuous and discontinuous behavior. For the nonlinear model of Jaffrin and Probst, this boundary was observed to occur for systems with initial Mach numbers of greater than 1.19. In the linear model, the capacity for modification of coupling and diffusivity parameters allows for substantial flexibility in this regard; for a given ion charge state, the curve in parameter space (the transition-critical  $R$  versus  $M_0$ ) is limited by requirements on the accessibility of the intersection between the left and right branches in phase space and the necessity that the slope through said intersection be real-valued.

# Chapter 4

## Further Calculations

### 4.1 Conductivity Models

The general format for a model of electron conductivity is as follows:

$$\kappa_e = \kappa_0 \frac{T_e^{\frac{5}{2}}}{\log \Lambda} \quad (4.1)$$

A dependence on electron temperature is shown explicitly. The logarithmic term in the denominator may be evaluated in one of several ways. A common treatment is to take its value as constant, since the expression varies weakly with temperature and density; however, **xRage** allows for a range of variable treatments of the term. The Coulomb log – and its counterpart Spitzer log, which may be alternately applied – will be discussed in depth further in this report.

#### 4.1.1 Zel’dovich and Raizer

The electron conductivity model of Zel’dovich and Raizer[5], introduced here for comparison with the treatments of the nonlinear model and **xRage**, is presented in the form

$$\kappa_e = \frac{\xi}{Z} \frac{k^{\frac{7}{2}}}{e^4 m_e^{\frac{1}{2}}} \frac{T_e^{\circ \frac{5}{2}}}{\log \Lambda}. \quad (4.2)$$

Note that this model shows a dependence on ion charge state,  $Z$ . The nonlinear model of Jaffrin and Probstein is formulated exclusively for the charge state of one, corresponding with monatomic hydrogen; therefore, this comparison model will be evaluated at that state.

We may first evaluate the unitless prefactor term  $\xi$ , following the discussion in the text:

$$\xi = \xi[Z] \quad (4.3)$$

$$\xi[1] = 0.950 \quad (4.4)$$

Continuing with the collocation and combination of known parameters, the following group is common to several of our models for conductivity and is numerically evaluated here:

$$\frac{k^{\frac{7}{2}}}{e^4 m_e^{\frac{1}{2}}} = 1.9249634 \cdot 10^{-5} \quad (4.5)$$

The units of this expression are  $[\frac{erg}{cm \cdot s \cdot K^{3.5}}]$ . It is important to note that **xRage** operates in electronvolt temperatures, so this coefficient will need to be restated for input into that code.

Combining the previous results, we yield a relation of the expected form;

$$\kappa_e = 1.8287153 \cdot 10^{-5} \frac{T_e^{\circ \frac{5}{2}}}{\log \Lambda}. \quad (4.6)$$

For comparison with, and input into, the **xRage** package, we multiply this expression by a factor of the Boltzmann constant. An equivalent electron conductivity model, keyed to temperature units of eV, is

$$\boxed{\kappa_e = 3.0785139 \cdot 10^9 \frac{T_e^{\frac{5}{2}}}{\log \Lambda}.} \quad (4.7)$$

### 4.1.2 Jaffrin and Probstein

The authors present the following expressions for, respectively, the electron and ion conductivities and the shear viscosity coefficient of a plasma of monatomic hydrogen:

$$\kappa_e = \frac{\kappa_{i2} \sigma^{\frac{5}{2}}}{\epsilon}, \quad (4.8)$$

$$\kappa_{i2} = \frac{45k}{16m_i} \mu_{i2}'' \quad (4.9)$$

$$\mu_{i2}'' = \frac{5}{6} \sqrt{\frac{m_i}{\pi}} \frac{(kT_i)^{\frac{5}{2}}}{e^4 \log \Lambda}. \quad (4.10)$$

Thus, we show the electron conductivity incorporated by the reduced model of Jaffrin and Probstein to be

$$\kappa_e = \frac{225}{96\pi^{\frac{1}{2}}} \frac{k^{\frac{7}{2}}}{e^4 m_e^{\frac{1}{2}}} \frac{T_e^{\circ \frac{5}{2}}}{\log \Lambda}. \quad (4.11)$$

In an analysis similar to that of the previous case, we evaluate the known parameters:

$$\kappa_e = (1.3223193) (1.8287153 \cdot 10^{-5}) \frac{T_e^{\circ \frac{5}{2}}}{\log \Lambda}, \quad (4.12)$$

$$\kappa_e = 2.5454163 \cdot 10^{-5} \frac{T_e^{\circ \frac{5}{2}}}{\log \Lambda}, \quad (4.13)$$

which remain in units of  $[\frac{erg}{cm \cdot s \cdot K^{3.5}}]$ . We conclude by converting this into the eV form:

$$\boxed{\kappa_e = 4.2850300 \cdot 10^9 \frac{T_e^{\frac{5}{2}}}{\log \Lambda}.} \quad (4.14)$$

### 4.1.3 Generalized Spitzer: xRage

The default model that calculates electron conduction within **xRage** employs a form of the generalized Spitzer coefficient as follows:

$$\kappa_e = \frac{8^{\frac{3}{2}}}{\pi} \frac{k^{\frac{7}{2}}}{e^4 m_e^{\frac{1}{2}}} \frac{1}{1 + 3.3/Z} \frac{T_e^{\frac{5}{2}}}{Z \log \Lambda}. \quad (4.15)$$

If we take, as we have in the previous cases,  $Z = 1$ , this model reduces to an expression with familiar terms,

$$\kappa_e = (0.9450216)(1.8287153 \cdot 10^{-5}) \frac{T_e^{\circ \frac{5}{2}}}{\log \Lambda}, \quad (4.16)$$

$$\kappa_e = 1.7281754 \cdot 10^{-5} \frac{T_e^{\circ \frac{5}{2}}}{\log \Lambda}, \quad (4.17)$$

in units of  $[\frac{erg}{cm \cdot s \cdot K^{3.5}}]$ . The result within **xRage** will be processed in units of eV, and ought to appear as follows:

$$\kappa_e = 2.9092620 \cdot 10^9 \frac{T_e^{\frac{5}{2}}}{\log \Lambda}. \quad (4.18)$$

#### 4.1.4 Simple Spitzer: xRage

The simple Spitzer conductivity model within **xRage** is a model equivalent to the general form for conductivity relations, (4.1), into which  $\kappa_0$  and  $\log \Lambda$  are entered by the user. The manual suggests a value of order  $[10^{-9}]$  be entered for the  $\kappa_0$  prefactor, given units of  $[\frac{erg}{cm \cdot s \cdot eV^{3.5}}]$  that are consistent with those presented in this section of the report. This would appear to be a clear typo, as all previous models have produced results of the order  $[10^9]$ . Therefore, the model with the corrected prefactor appended stands as

$$\kappa_e = 3.0041640 \cdot 10^9 \frac{T_e^{\frac{5}{2}}}{\log \Lambda}. \quad (4.19)$$

#### 4.1.5 Review and Conclusions

Comparison of a wide range of electron conductivity models from the literature demonstrates their many inherent similarities, and bolsters arguments for the validity of a representative calculation in **xRage** of the reduced form of Jaffrin and Probstein. As the simple Spitzer conductivity model reviewed above allows for the direct input of prefactor terms, it is trivial to flag that particular version of electron conductivity to be employed by **xRage** and input its  $\kappa_0$  as that calculated for the Jaffrin and Probstein model and listed above. It is worth noting that, while all four listed prefactor terms are on the same order of magnitude, it is the Jaffrin and Probstein term that is a slight outlier. All other prefactors are, in units of  $[\frac{erg}{cm \cdot s \cdot K^{3.5}}]$ , roughly equivalent to  $3 \cdot 10^9$ ; their model suggests a  $\kappa_0$  greater than  $4 \cdot 10^9$ . It is this dissimilarity that demands the employment of the simple Spitzer option in **xRage** for matching Jaffrin and Probstein results, as it has already been demonstrated that the default and generalized model produces results on par with the other conductivity models.

## 4.2 Electron-Ion Coupling

A general model for the electron-ion coupling is

$$\gamma_{ei} = \gamma_0 \rho^2 T_e^{-\frac{3}{2}} \log \Lambda. \quad (4.20)$$

Explicit dependence on electron temperature and density (alternately, number density) is shown. The logarithmic term, once more a form of either the Coulomb or Spitzer logarithm, presents notable difficulty in the representation of the reduced model in the full model of **xRage**. Jaffrin and Probstein make the assumption that that function depends only weakly on changes in density and temperature and therefore take it as a constant at the downstream value; **xRage** has no option available for a constant logarithmic term,

and therefore required direct modification to replicate the reduced model’s coupling term. At an initial Mach number of 1.70, an elevation in the term of some 30% was observed across the shock using a variable model. While this may be negligible in the sense of order of magnitude, as was previously assumed, it is worth noting that the logarithmic value peaked and declined quite rapidly about the shock; furthermore, alterations to transient solutions were observed after forcing the logarithm to remain constant. Their significance is not yet clear.

### 4.2.1 Jaffrin and Probstein

The equation for electron energy, as presented in (3.14) of the authors’ 1964 paper, contains the term

$$\sqrt{2} \frac{\epsilon \Delta_s (\sigma - \tau)}{M_1 l_1 \omega \omega_e \sigma^{\frac{3}{2}}}, \quad (4.21)$$

in which  $\epsilon$  is the square root of the ratio of electron mass to ion mass, and  $\Delta_s$  is the shock thickness.  $\sigma$  and  $\tau$  are the dimensionless electron and ion temperatures, respectively, scaled against the downstream equilibrium temperature  $T_1$ .  $\omega_e$  and  $\omega$  are the electron and ion velocities, respectively, and are likewise scaled against the downstream equilibrium velocity  $u_1$ .

We are concerned with this term because of its similarity to the implementation of electron-ion coupling in **xRage** and other models of 2T and 3T plasma flow. Note that there is present a relation  $(\sigma - \tau)$ , the difference in dimensionless electron and ion temperatures. Therefore, we can consider the collection of terms multiplying this relation to be a measure of the ‘coupling strength’ between electrons and ions in the solution. Replacing the shock thickness  $\Delta_s$  with its appropriate value for the relaxation layer with which we are concerned – that dominated by the dissipative mechanisms of electron thermal conduction and electron-ion energy interchange in the limit of small Debye length with respect to mean free path – we arrive at the expression

$$\gamma_{ei} \propto \sqrt{2} \frac{T_1}{M_1 \omega \omega_e \sigma^{\frac{3}{2}}}. \quad (4.22)$$

Given that the  $\omega$  terms in the denominator are equivalent to the inverse of the dimensionless number density terms, which are themselves proportional to the mass densities of the constituents, we have recovered an expression that contains the requisite  $[\rho^2 \cdot T_e^{-\frac{3}{2}}]$  of our general coupling form, (4.20) above.

In manipulating this expression into a fully dimensioned form, we may return to (2.17) in the 1964 paper of Jaffrin and Probstein, an earlier phase of the derivation. There we encounter the following relation:

$$\frac{4Ak}{\sqrt{\pi} c^3 (m_e + m_i)} (T_e - T_i), \quad (4.23)$$

which has the difference in constituent temperatures we expect to be linked to a coupling term.

That relation can be expanded into a form that resembles the general form of (4.20). Given the definitions for  $A$  and  $c$ ,

$$A = \frac{4\pi n_e n_i (m_e + m_i) e^4 \log \Lambda}{m_e m_i}, \quad (4.24)$$

$$c = \sqrt{2kT_e/m_e + 2kT_i/m_i}, \quad (4.25)$$

that relation expands to

$$\gamma_{ei} = \frac{16k\pi n_e n_i (m_e + m_i) e^4 \log \Lambda}{\sqrt{\pi} c^3 m_e m_i (m_e + m_i)}, \quad (4.26)$$

$$\gamma_{ei} = \frac{4\sqrt{2\pi} e^4 n_e n_i \log \Lambda}{m_e m_i} \frac{1}{\sqrt{k}(T_e/m_e + T_i/m_i)^{\frac{3}{2}}}. \quad (4.27)$$

Note that we can here make the reasonable assumption that the term  $(T_e/m_e + T_i/m_i)$  in the denominator ought to be dominated by the electron temperature term, as the electron mass is over three orders of magnitude larger than the corresponding ion mass. Having made this adjustment, we arrive a final expression

$$\gamma_{ei} = \frac{4\sqrt{2}\pi e^4 n_e n_i \log \Lambda}{m_e m_i} \left(\frac{m_e}{T_e}\right)^{\frac{3}{2}} \frac{1}{\sqrt{k}}. \quad (4.28)$$

### 4.2.2 ‘Optional Electron-Ion Coupling’: **xRage**

**xRage** incorporates, as a non-default option, a form of electron-ion coupling analogous to the model retrieved from an analysis of Jaffrin and Probst. With an ion charge state of 1, again equivalent to that for monatomic hydrogen, the version in the code simplifies to

$$\gamma_{ei} = \frac{4\sqrt{2}\pi e^4 n_e n_i \log \Lambda}{m_e m_i} \left(\frac{m_e}{T_e}\right)^{\frac{3}{2}}. \quad (4.29)$$

We immediately note a close agreement with the coupling term of the reduced model. However, there is a potentially troubling factor of the Boltzmann constant in the denominator of the latter coupling relation that is not present in the former. In an attempt to resolve this discrepancy, recall that the temperature units of Jaffrin and Probst are presumably Kelvin, while the units of **xRage** are typically  $eV$ . The Boltzmann constant,  $k$ , can be used to convert between those systems. As the  $\gamma_{ei}$  term must be multiplied against a temperature difference, the units of the outlier term  $-\left[\frac{\text{Temperature}}{\text{Energy}}\right]^{\frac{1}{2}}$ , and after substitution  $\left[\frac{K}{eV}\right]^{\frac{1}{2}}$  will properly convert the remaining temperature units, which will be of the form  $T_e^{o1-\frac{3}{2}}$  and thus  $\left[\frac{1}{K}\right]^{\frac{1}{2}}$

### 4.2.3 Review and Conclusions

Unlike in the case of conductivity, **xRage** has no ‘simple model’ functionality for coupling into which a prefactor and logarithmic term may be entered. However, this analysis has shown that there does exist a coupling model in **xRage** that ought to produce equivalent results to that employed by Jaffrin and Probst, for an equivalent logarithmic term. All other terms in the calculation of  $\gamma_{ei}$  through those two methods are directly comparable.

## 4.3 Coulomb and Spitzer Logarithms

The logarithmic term common to both the electron-ion coupling and electron conductivity relations is referred to as either the Coulomb logarithm or the Spitzer logarithm; it relates to the cross-section for ‘Coulomb collisions’ between particles. A wide range of expressions are available. Note that Jaffrin and Probst assume that the differences in the logarithmic term with density and temperature variance are negligible, and therefore employ a constant value taken at the downstream condition. A constant, user-specified input is available for the term in **xRage**’s simple Spitzer conductivity. No corresponding option exists for the electron-ion coupling, however. To replicate the Jaffrin and Probst model in **xRage**, it was therefore necessary to modify the code’s source to allow for a constant Coulomb logarithm.

# Chapter 5

## Conclusions

The linear model of Masser, Wohlbier, and Lowrie produced results very similar to those of **xRage**, and shows strong potential for service in future verification studies. A range of test cases at initial Mach numbers of 1.19, 1.40, and 1.70 demonstrated convergence; the ‘norm’ considered was that of maximum ion temperature. In order to reproduce the displacement scaling of the **xRage** results in the linear reduced model’s script, it was necessary to employ specific heats and constant electron conductivities equivalent to those expressed in the input deck. Following that, temperature profiles versus displacement were also convergent. Static grids and AMR (adaptive mesh) grids were employed in this study. AMR grids allowed for enhanced resolution about the shock face, but required significantly more wall time to produce steady-state solutions.

The nonlinear model of Jaffrin and Probststein did not produce results comparable to the model of **xRage**. Initially, the input deck was set to employ the default relations for coupling and conductivity, which were not in complete agreement with the relations of the nonlinear model. After an analysis of each parameter and a wide range of potential equations for its expression, the simple Spitzer conductivity model and the ‘Optional Electron-Ion Coupling’ model were selected for their demonstrated similarity with the parameters of Jaffrin and Probststein. The simple Spitzer prefactor term was calculated with reference to several published models, and ultimately set to be equal to that determined for the Jaffrin and Probststein conductivity. The Coulomb logarithm was calculated again with reference to the nonlinear model, and found to be on the order of 1 for the case of the atypically dense plasmas here studied.

Examining solutions with variable coefficients in **xRage**, it appears that the electron-ion coupling term is significantly stronger than that of the nonlinear model. Parameters relating to the conductivity and coupling were manually changed in the input deck to attempt a diagnostic of this effect, and it appeared that the product of the coupling term and conductivity were several times greater in **xRage**’s determination than in Jaffrin and Probststein’s. Neither term is explicitly present in the ordinary differential equations of that reduced model, and it is possible that the simplifications made in attaining that model altered the conductivity and coupling relations as they were initially stated.

The performance of verification tests of **xRage** against the linear model is promising. However, the current method for comparison between the nonlinear model and default **xRage**, with variable coupling and conductivity, requires further study. Instead of attempting to retrofit the source code to match a particular reduced model, it may prove more fruitful to add the functionality of variable parameters into a linear model, like that of Masser, Wohlbier, and Lowrie. In fact, initial tests of that model with a coupling parameter  $R$  that adjusts at each spacial step have produced results closer to those of **xRage** than the nonlinear model of Jaffrin and Probststein. However, the equations of that model must be derived from their governing form without assumptions of constant value before such a test can be comprehensive.

## Chapter 6

# Acknowledgments

I would like to thank my mentor, Thomas Masser, for his invaluable advice and assistance in directing this summer research project. I would also like to acknowledge the contributions of my research partner, Samuel Shaner, particularly as regards solutions to the linear reduced model. Finally, I must note my gratitude to Scott Runnels, the workshop manager, for his excellent direction of this program – and also the members of the LANL community who volunteered their time to present to, interact with, educate, and accommodate me and my fellow students.

# Bibliography

- [1] W. Fickett and W.C. Davis, *Detonation: Theory and experiment*, University of California Press, 1979.
- [2] M. Jaffrin and R. Probstein, *Structure of a plasma shock wave*, The Physics of Fluids (1964).
- [3] D. Mihalas and B. Mihalas, *Foundations of radiation hydrodynamics*, Oxford University Press, 1984.
- [4] J. Wohlbier T. Masser and R. Lowrie, *Shock wave structure for a fully ionized plasma*, Shock Waves (2011).
- [5] Y.B. Zeldovich and Y.P. Raizer, *Physics of shock waves and high-temperature hydrodynamic phenomena*, Academic Press, 1967.

# Verification Study of Planar Shocks in Dense Plasmas

Samuel Shaner<sup>1,2</sup> and Thomas Masser<sup>1</sup>

<sup>1</sup>*Los Alamos National Laboratory, Computational Physics and Methods (CCS-2)*

<sup>2</sup>*Massachusetts Institute of Technology, Department of Nuclear Science and Engineering*

## Abstract

*Planar shock waves in fully ionized plasmas can be characterized by their spatial electron and ion temperature profiles. These profiles are highly dependent on several properties of the ionized flow: density, mach number, ion atomic number, electron-ion coupling, and electron thermal conductivity. In this project, we are investigating the effect of these parameters on the boundary between continuous and discontinuous shock flows in dense plasmas. We are using two models to conduct this study: a steady-state two-temperature model developed by Masser et. al. and the three-temperature hydrodynamics model implemented in LANL's flagship multiphysics code, xRage. Through this ongoing analysis we hope to further refine the bounding conditions between continuous and discontinuous shock flows in dense plasmas and verify the numerical simulation for shocks in dense plasmas produced by xRage.*

## Introduction

Shock waves in plasmas are important for several applications including modeling of ICF pellet implosion and certain astrophysical systems. Experimental studies of shock waves are expensive and difficult or impossible to perform which necessitates the use of computational methods. Scientist at SAIC and Los Alamos National Laboratory have co-developed xRage, a generalized three-temperature, multi-material Eulerian hydrodynamics code, to model shocks in a wide range of materials. In the present study we focus on modeling planar shocks in dense plasmas using a simple two-temperature fully ionized plasma model and comparing the results to xRage.

The two-temperature model used in our paper is motivated by the earlier work done by several scientists in the 1950's and 1960's [1-4]. Zeldovich studied strong shock waves in air using a two-temperature electrically neutral plasma model [1]. Shafranov computed the shock profiles for a specific model of hydrogen with shocks of varying strength [2]. Imshennik analyzed a generalization of Shafranov's model to determine a critical shock strength between continuous (fully dispersed)

and discontinuous (embedded hydrodynamic shock) profiles [3]. Jaffrin and Probstein considered similar models with consideration of the effects of charge separation and electron viscosity [4]. In this study, we build on these models and focus only on the interactions of the electrons and ions within a shock moving through a fully ionized gas. Masser et. al. studied the parameter space of stationary shock waves in dense plasmas using this model and their paper is used as a basis for this study [5]. This model assumes that strong Coulomb interactions keep the electrons and ions rigidly coupled, so that the plasma remains electrically neutral. Additionally, it neglects all radiative effects and treat both the electron diffusivity and electron-ion coupling coefficients as constants. While the simplifications of this model make it invalid for studying real shocks, the solutions keep intact the important mathematical characteristics and allow us to use Eulerian numerical integration schemes to rapidly solve the equations and produce results.

## Model and Methods

The governing equations for our model consist of the Euler equations with heat transfer (1-3) along with the electron advection equation which serves to keep the electrons and ions coupled (4):

$$\partial_t \rho + \partial_x(\rho v) = 0 \quad (1)$$

$$\partial_t(\rho v) + \partial_x(\rho v^2 + p) = 0 \quad (2)$$

$$\partial_t(\rho E) + \partial_x(v(\rho E + p)) = \partial_x(\kappa_e \partial_x T_e) \quad (3)$$

$$\partial_t(\rho e_e) + \partial_x(\rho v e_e) + p_e \partial_x v = \gamma_{ei}(T_i - T_e) + \partial_x(\kappa_e \partial_x T_e) \quad (4)$$

Where  $\rho$  is the density,  $v$  is velocity,  $p$  is pressure,  $E$  is energy,  $e_e$  is specific electron energy,  $T$  is energy in eV,  $\kappa_e$  is the electron conductivity, and  $\gamma_{ei}$  is the electron-ion coupling term. To reduce the number of unknowns to the equal the number of equations, we apply several closure relations

### $\gamma$ -law equation of state

$$p = \rho e(\gamma - 1)$$

### Z split of specific heat capacities

$$e = C_v T, \quad \frac{C_{v,i}}{C_v} = \frac{1}{Z+1}, \quad \frac{C_{v,e}}{C_v} = \frac{Z}{Z+1}$$

We solve the governing equations by considering the steady-state case. The time derivatives drop out from the governing equations to yield a set of coupled ODEs:

$$\frac{d(\rho v)}{dx} = 0 \quad (5)$$

$$\frac{d(\rho v^2 + p)}{dx} = 0 \quad (6)$$

$$\frac{d[v(\rho E + p)]}{dx} = \frac{\kappa_e \frac{dT_e}{dx}}{dx} \quad (7)$$

$$\frac{d(\rho v e_e)}{dx} + p_e \frac{dv}{dx} = \gamma_{ei}(T_i - T_e) + \frac{\kappa_e \frac{dT_e}{dx}}{dx} \quad (8)$$

We can then apply the closure relations, combine equations, and nondimensionalize variables to yield a set of two autonomous, coupled ODEs:

$$\frac{2\kappa_e(\gamma - 1)}{\gamma + 1} \frac{d\Theta_e}{dx} = (1 - \eta)(\eta - \eta_1) \quad (9)$$

$$\frac{2\kappa_e(\gamma - 1)}{\gamma + 1} \frac{d\eta}{dx} = -\eta R \left( \frac{Z + 1}{Z} \right) \left( \frac{\Theta_e - \Theta_e^{c2}}{\Theta_e - \Theta_e^{c1}} \right) \quad (10)$$

Where the new terms represent nondimensional quantities:

$\eta$	–	velocity
$\Theta_e$	–	electron temperature
$R$	–	coupling parameter

The nonlinear system can produce a discontinuity (an embedded hydrodynamic shock) in the ion temperature as a function of position. This discontinuity is found by integrating a thermodynamically equivalent form of the governing equations around an infinitesimal domain surrounding the shock. The jump in entropy (and ion temperature) is motivated by the assumption of a zero electron entropy jump in the case of vanishing electron-ion coupling and electron heat conduction. The discontinuity condition can be simplified to the following transcendental equation:

$$\Theta_e(\gamma - 1) \frac{\gamma - 1}{\gamma + 1} \log \left( \frac{\eta_L}{\eta_R} \right) = (\eta_L - \eta_R) \left( \frac{1 + \eta_1}{2} - \frac{\eta_L + \eta_R}{2} \right) \quad (11)$$

In this study, we have written a matlab code to solve for the electron temperature profile over a shock by integrating from the steady state conditions. Additionally, we have written a parallel run script to execute the code at many points in the parameter space to determine the bounding curve that separates continuous

from discontinuous solutions in region 2 (green) of figure 1. Pending LA-UR approval, the source code and documentation will be put on Samuel Shaner's github account under the project title *shock\_analysis* (LINK). The solution procedure for generating temperature profiles of shocks is as follows:

1. Determine the upstream (initial) dimensionless velocity and ion temperature ( $\eta_1, \Theta_1$ ) given an input Mach number,  $M_0$ , and ionization number,  $Z$ .
2. Determine what region (see figure 1) the shock is in based on the input Mach number
3. If shock is in region 1 (green):

- Evaluate  $\frac{d\Theta_e}{d\eta}$  at  $\eta = 1$  (the upstream state).
- Determine  $\Theta_{e,\epsilon}, \Theta_e$  a short distance from the downstream state.
- Integrate in the  $+x$  direction from the state  $(\eta, \Theta_e)_\epsilon$  near the  $(1, \Theta_0)$ , the upstream state, to near  $(\eta_1, \Theta_1)$ , the downstream state.

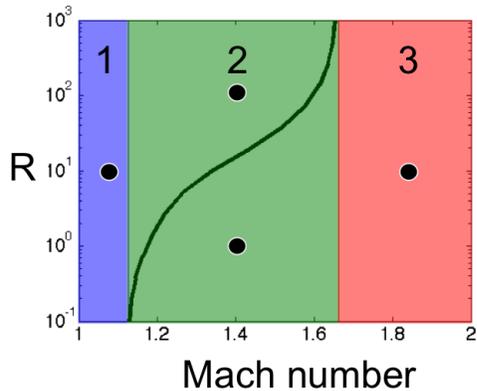
If shock is in region 2 or 3 (green or blue):

- Evaluate  $\frac{d\Theta_e}{d\eta}$  at  $\eta = 1$  (the upstream state).
- Determine  $\Theta_{e,\epsilon}, \Theta_e$  near both end states.
- Integrate in the  $+x$  direction from the state  $(\eta, \Theta_e)_\epsilon$  near the  $(1, \Theta_0)$ , the upstream state, until the  $\eta$  value corresponding to the intersection of  $\Theta_e^{c1}$  and  $\Theta_e^{c2}$  is reached.
- Integrate in the  $-x$  direction from a state  $(\eta_1, \Theta_1)$ , the downstream state, until the  $\eta$  value corresponding to the intersection of  $\Theta_e^{c1}$  and  $\Theta_e^{c2}$  is reached.
- If the solution curves intersect at the intersection of  $\Theta_e^{c1}$  and  $\Theta_e^{c2}$ , the solution is continuous. Otherwise, search through values of  $\Theta_e$  and corresponding  $\eta_R$  and  $\eta_L$  until the shock discontinuity condition is reached. Note the  $\eta_R$  and  $\eta_L$  values.
- Connect the two solution curves to create a continuous solution for  $\Theta_e$  vs.  $x$ .

4. Use the balance equation below to find the ion temperature profile.

$$\Theta_i - \Theta_e = (Z + 1) \left( (1 + \gamma M_0^2) \eta - \gamma M_0^2 \eta^2 - \Theta_e \right) \quad (12)$$

Where  $\gamma$  is the ratio of heat capacities (assumed to be 5/3).



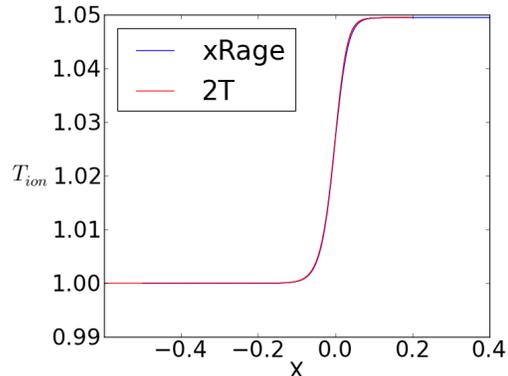
**Figure 1:** Solution parameter space for a  $H^1$  plasma. The blue region contains only continuous solutions. The green region includes both continuous solutions (above the black curve) and discontinuous solutions (below the black curve). The red region contains only discontinuous solutions. The black circles represent the shock conditions for shock temperature profiles in figures 2-5.

## Results

### Verification Study

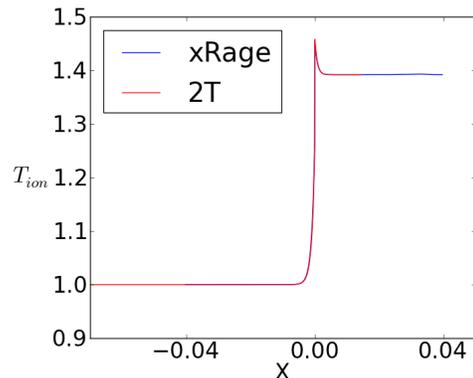
In the present study we considered contrived plasma shocks with constant electron conductivity, electron-ion coupling, and no radiative heat transfer. We present four shocks, one in each end region and two in the middle, green region of the parameter space (figure 1) to demonstrate our parameter space mapping and to verify the xRage solutions. To give some historical context, we present this parameter space plot and emphasize the different regions because they differ from the previous studies conducted with this model [1-4]. These older studies suggest that the parameter space only contains 2 regions, one being the blue region and the other being the combined green and red regions, where the continuity of the solution only depends on the Mach number. Upon further investigation we have found that there is an intermediate region (the green region) where the continuity of the solution depends on both Mach number and coupling parameter ( $R$ ). We present the ion temperature profiles for the four conditions shown in figure 1 below to demonstrate this finding. The ion temperature profiles are shown because they most clearly show discontinuous solutions. Temperature profiles were generated with both our two-temperature model and with xRage and by comparing the models we are verifying the solutions that xRage outputs.

We begin with the left most point at ( $Mach = 1.05, R = 10$ ). The shock at this condition is shown in Figure 2 below and we clearly see a continuous profile as suggested by our model and the previous models. Additionally, the xRage and 2T model temperature profiles line up almost exactly on top of each other. This confirms that xRage is producing the correct profile at this condition.



**Figure 2:** Ion temperature profiles for a shock with  $R = 10$  and  $M_0 = 1.05$ .

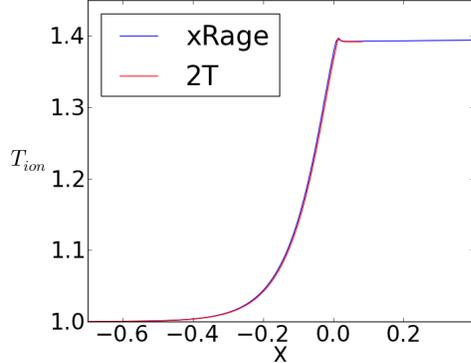
Now let's look at the cases the in green region where our model and the previous models disagree. Figure 3 shows the low  $R$  value point at ( $Mach = 1.4, R = 40$ ). The ion temperature profile for both rage and our 2T model show discontinuous solutions and line up closely.



**Figure 3:** Ion temperature profiles for a shock with  $R = 1$  and  $M_0 = 1.4$ .

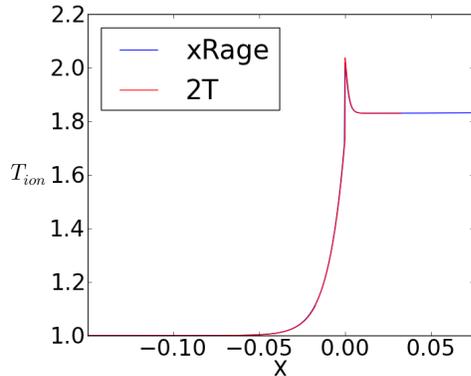
As discussed, the older models [1-4] predict that the continuity of the solution depends only on Mach number. Therefore, we would expect that raising the coupling parameter,  $R$ , at a constant Mach number of 1.4 would not affect the continuity/discontinuity of the solution. Figure 4 shows the ion temperature profile for a shock at ( $Mach = 1.4, R = 100$ ), which is above the continuous/discontinuous boundary curve in the green region. Both xRage and our 2T model show continu-

ous solutions and again agree quite well. This behavior speaks to the validity of our model and confirms that there is this intermediate region where the continuity of the temperature profiles depend on both Mach number and coupling parameter.



**Figure 4:** Ion temperature profiles for a shock with  $R = 100$  and  $M_0 = 1.4$ .

For good measure, we included an ion temperature plot at a high  $R$  value to confirm that both xRage and our 2T model produce discontinuous shocks. As seen in Figure 5 at the condition of ( $Mach = 1.8$ ,  $R = 10$ ) both rage and our 2T model produce discontinuous profiles and line up almost exactly on top of each other.

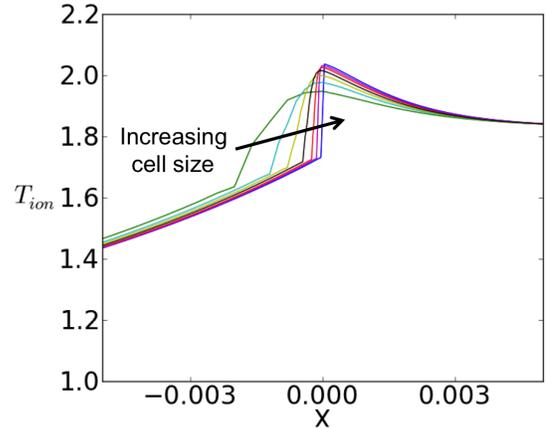


**Figure 5:** Ion temperature profiles for a shock with  $R = 10$  and  $M_0 = 1.8$ .

The four ion temperature profiles clearly show that both xRage and our 2T model produce shock temperature profiles that adhere to the continuity/discontinuity behavior predicted in our model and the profiles agree quite well. We have run many more trials in the parameter space and they all show the same behavior as long as xRage simulations are carried out with a large enough domain to fully capture the equilibrium solutions on either side and to long enough times to reach the steady state solution.

### *xRage Solution Convergence*

While our simple 2T model solves for the steady state case, xRage solves the time dependent problem. To make the xRage solution converge with the 2T solution, simulations must be run to long enough times and the cell width must be small enough. As seen in figures 3 and 5, some of the shocks can have very sharp temperature spikes near the shock discontinuity. This is inherently difficult to model when the cell size is finite and the shock may span only a few cells. Figure 6 below shows the converged ion temperature spikes for the  $R = 10$  and  $M_0 = 1.8$  case with cell sizes (in  $\mu\text{m}$ ) of: (0.4, 0.2, 0.1, 0.05, and 0.025) along with the 2T model solution shown in blue (the innermost curve). As the cell size decreases, we see that the shock width begins to decrease and the ion temperature spike gets sharper and closer to the 2T model value. This behavior was seen in all shocks with sharp discontinuities and demonstrates that users must keep a close eye on the cell width when simulating discontinuous shocks.



**Figure 6:** Zoomed in view of the ion temperature peak for the  $R = 10$  and  $M_0 = 1.8$  case with varying Eulerian cell sizes.

In addition to the cell width, the simulation time must be long enough so the temperature profile reaches the steady state solutions. While we have not performed a rigorous investigation of the convergence to steady state, we will mention that, in general, the simulation time to reach steady state follows a trend of:

$$time \propto \frac{R}{M_0} \quad (13)$$

## Conclusions and Future Work

In the present study, we have verified that xRage produces the correct temperature profiles for continuous and discontinuous shocks in dense plasmas with constant electron conductivities and electron-ion coupling parameters. Additionally, we have demonstrated that the parameter space for continuous/discontinuous solutions has three regions, instead of two as predicted by older models. The parameter space consists of a low Mach number region with only continuous solutions, a high Mach number region with only discontinuous solutions, and an intermediate Mach number region where the solution continuity depends on both Mach number and coupling parameter.

The verification study herein serves as an "eyeball norm" of the solutions. Future verification work on the shock solutions of xRage should go into more depth by performing a rigorous numerical analysis of the solutions that xRage produces. Additionally, our model should be extended to include more realistic models for electron conductivity and electron-ion coupling parameter as well as consideration of radiative heat transfer.

## Acknowledgements

I would like to thank Thomas Masser for providing direction and support on this project. I would also like to thank Scott Runnels for organizing the 2012 Computational Physics Summer Workshop. Helpful discussions with William Hoey are gratefully acknowledged. Funding was provided by the Advanced Scientific Computing Program at Los Alamos National Laboratory under charge code JPDJ.

## References

- [1] V. S Imshennik Shock wave structure in a dense high-temperature plasma *Soviet Physics JETP*, 15(1):167–174, 1962
- [2] M. Y. Jaffrin and R. F. Probstein Structure of a plasma shock wave *Physics of Fluids*, 7(10):1658–1674, 1964
- [3] V. D. Shafranov The structure of shock waves in a plasma *Soviet Physics JETP*, 5(6):1183–1188, 1957
- [4] Y. B. Zel’dovich Shock waves of large amplitude in air *Soviet Physics JETP*, 5(5):919–927, 1957
- [5] T. O. Masser, J. G. Wohlbiel, R. B. Lowrie Shock structure for a fully ionized plasma *Shock Waves*, (2011) 21:367-381

