LA-UR-15-28038
Approved for public release; distribution is unlimited.

Title: 2015 Final Reports from the Los Alamos National Laboratory Computational Physics Student Summer Workshop

Author(s): Runnels, Scott Robert; Caldwell, Wendy; Brown, Barton Jed; Pederson, Clark; Brown, Justin; Burrill, Daniel; Feinblum, David; Hyde, David; Levick, Nathan; Lyngaas, Isaac; Maeng, Brad; Reed, Richard LeRoy; Sarno-Smith, Lois; Shohet, Gil; Skarda, Jinhie; Stevens, Josey; Zeppetello, Lucas; Grossman-Ponemon, Benjamin; Bottini, Joseph Larkin; Loudon, Tyson Shane; VanGessel, Francis Gilbert; et al.

Intended for: Web

Issued: 2015-10-15

2015 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

LA-UR-15-__-TBD-___

Included in this Report

Contents

Introduction

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 perform 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; historically interactions rely heavily 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 fifth year for the Summer Workshop and the fourth in a series of reports [1] [2] [3]. As before, the workshop's goals were achieved by attracting a select group of students recruited from across the U.S. and immersing them for ten weeks in lectures and interesting research projects. The lectures provided an overview of the computational physics topics of interest 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 LANL mentors on specific research projects associated with predefined topics. This year, the topics included turbulence modeling using the BHR model, multi-scale materials modeling, multi-phase compressible flow, warm dense matter modeling, Monte Carlo thermal radiation transport, studying production codes on Xeon Phi/Knights Corner, Lagrangian radiation hydrodynamics, computational physics using a domain-specific language, and advanced mesh-free methods for compressible mechanics. The students' growth was furthered by their participation on teams where their teammates were sometimes of a different academic year. It also developed their skills 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 sponsors this Summer Workshop by funding the workshop coordinator under the University Liaison budget and paying the lease for the workshop facility. Funding for the students' stipends come from a variety of programmatic sources. A large majority of them fall under various projects that are part of the ASC Program, but a few other programs also provide funding for some students. This year, there were twenty mentors, up slightly from last year. The mentor participation amongst different divisions included XCP, XTD, CCS, and T. This broad participation is welcomed and it is hoped that it continues in future years.

Students

Sixty-six students applied for admission to the workshop, all eligible U.S. citizens with the breakdown shown in the chart on the next page. The twenty-two who ultimately were selected and participated were from the following schools: ASU, Virginia Tech, Univ. Arizona, UC Santa Cruz, Univ. Vermont, UC Irvine, Stanford, Univ. New Mexico, Florida State Univ., Univ. Michigan, Kansas State, Univ. Michigan, Univ. Illinois Urbana-Champaign, Missouri Univ. of Sci. & Tech., Columbia, Univ. Illinois Urbana-Champaign, Univ. Texas at Austin, Univ. Maryland, and Univ. Washington

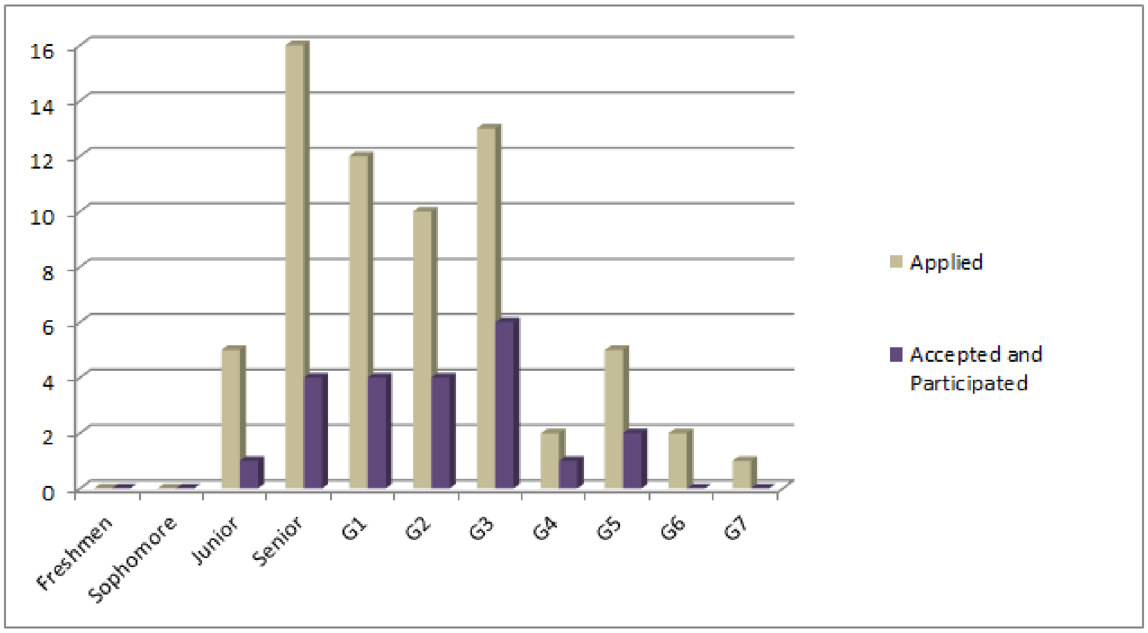


Figure 1: This figure shows the number of students who applied to the Summer Workshop and how many were accepted and participated, broken down by academic year. In this figure “G1” means “first-year graduate student” at the time of the workshop,.i.e., starting their first year of graduate school in the fall after the workshop. “G2” means “second-year graduate student” at the time of the workshop, i.e., starting their second year of graduate school in the fall after the workshop.

Lectures

In this fifth year of the Summer Workshop the effort begun last year, which was to more tightly integrate the lectures, was continued. The increased integration is part of a effort to transform the stand-alone lectures into a sequence exhibiting a more course-like feel. A foundational lecture at the beginning of the Summer Workshop, introducing the fundamentals of transport theory was introduced last year to provide a common basis upon which several other lectures could build. That lecture was provided again this year. Also the development of a one-dimensional hydrocode was performed in class; the resulting code provided a basis for other lecture materials and for exploratory studies that some of the students performed at the beginning stages of their projects. The approximately 27 hours of lectures, for which the students’ attendance was required, were augmented with other lectures and demonstrations for which the students’ attendance was optional. These lectures were provided to help students who were lacking certain skills develop them quickly to aide them during the summer. The lectures included a tutorial on C++ object-oriented programming, Python programming, and Unix.

The lectures were scheduled to be most frequent in the beginning of the workshop, when the students’ research was just getting started and they needed the most background information. Their frequency dropped significantly until there were no lectures at all in the latter weeks of the workshop so that the students could focus on their research. The lectures are summarized in the table that follows.

Required Lectures		
Title	**Hrs.**	**Lecturer**
Essentials of Transport Equations	2	S. Runnels
Introduction to Lagrange Hydro	1	S. Runnels
Intro to High-Performance Computing at LANL	1	R. Cunningham
Introduction to Hydro Terminology and Artificial Viscosity	1	S. Runnels
Live Demo: Development of a 1-D Gas Hydrocode	1	S. Runnels
Plasticity Modeling	1	S. Runnels
Live Demo: Adding Plasticity to a 1-D Hydrocode	1	S. Runnels
Introduction to Monte Carlo and MCNP	3	F. Brown
Introduction to Thermal Radiation Transport	1	T. Urbatsch
Warm Dense Matter Simulation	1	O. Certik
Radiation Hydrodynamics	1	S. Ramsey
Rocks in Space	1	C. Plesko
Introduction to Molecular Dynamics	1	C. Starrett
Multi-Material Equilibration Techniques	1	A. Harrison
HPC Performance Analysis	1	L. Vernon
Domain Specific Languages - Scout	1	C. Sweeney
Turbulence Modeling	2	D. Israel
Interface Reconstruction Methods	1	M. Shashkov
V & V and Uncertainty Quantification	1	J. Kamm & G. Weirs (Sandia)
Mesh Free Methods	1	G. Dilts
Introduction to Slidelines	1	N. Morgan
Survey of ALE Methods	1	N. Morgan
Compressing Colloids	1	D. McDermott (Wabash)
Optional Lectures		
Tutorial in C++ Programming	2	S. Runnels
Live Demo: Unix Tutorial	1	C. Sweeney
Introduction to Python	1	D. Israel
Python for Scientific Computing Applications	1	D. Israel
Introduction to Parallel Programming	1	R. Robey
Git	1	B. Runnels (UCCS)
Python for Scientific Computing Applications	1	D. Israel

References

[1] Scott R. Runnels (editor). Final report from the 2012 computational physics student summer workshop. Technical report, Los Alamos National Laboratory, 2012.

[2] Scott R. Runnels (editor). Final report from the 2013 computational physics student summer workshop. Technical report, Los Alamos National Laboratory, 2013.

[3] Scott R. Runnels (editor). Final report from the 2014 computational physics student summer workshop. Technical report, Los Alamos National Laboratory, 2014.

Introduction to the Technical Reports and the Teaming Arrangements

The Summer Workshop is primarily an educational endeavor with a healthy emphasis on research. Because of that, most of the chapters that follow represent actual research progress, some of which are worthy of conference or peer-reviewed publication. However, other chapters may simply represent the students' educational progress in a particular area. Because of that mixture, it is worth mentioning that the results and opinions expressed in these reports may or may not be representative of the ASC program's position the associated technical areas.

In this workshop, each student is paired with another student under one or more mentors, and in that arrangement the students are not necessarily at the same academic level or background. Developing a team-based approach to the research project is one of the secondary objectives of the workshop, but not the primary objective, which is the students' education. The technical reports that follow may or may not have a strong teaming arrangement behind them. For some projects, close teaming is the best choice, while for others it is more appropriate to allow the students to explore the project area at their own pace. These aspects are discussed in some of the projects' Introduction section.

While the students' reports are integrated in this report, each chapter is intended to essentially be a stand-alone document. Nomenclature may not be consistent between the chapters. Figures, equations, and concepts may be repeated.

Monte Carlo Thermal Radiation Transport: Discrete Diffusion Monte Carlo (DDMC) on Triangular Mesh

Team Members
Frank G. VanGessel and Richard L. Reed

Mentors
Mathew Cleveland, Allan Wollaber, Todd Urbatsch

Abstract

The study of thermal radiative transport is an important area of research at Los Alamos National Laboratory (LANL). The work presented in this report was completed as part of the Computational Physics Workshop hosted by LANL. Goals for the project were to apply Monte Carlo methods to solving the diffusion based radiative transport equations. These methods are specifically applied to an unstructured triangular mesh.

For this project, a new discretization of the transport equations was developed and implemented for the triangular mesh. The work was tested by implementing the discretization into a Python program, and several test problems were developed. Furthermore, the functionality was implemented into the existing C++ project Jayenne at LANL.

This work was the first part of a five year project that would fully implement the method for unstructured triangular meshes. Future work for this part of the project is to validate the results provided by the method against analytical results, and perform mesh comparisons using Jayenne between triangular and Cartesian meshes.

Introduction

This work focused on the solution of the strongly coupled thermal radiative transport equations in the diffusive regime, where simplifications to the transport equations yield a diffusion equation [1]. Solving the diffusion equations is significantly quicker than solving full transport equation in the diffusive regime. The focus for this project was in three distinct areas including: derivation and discretization of the diffusion equations, creating a python code for algorithm development, and implementing new capability into the existing Monte Carlo radiative transport code at Los Alamos National Laboratory (LANL).

In what follows, a brief background of the governing equations is provided, followed by an overview of the discretization. Following that, the Monte Carlo method is described, and finally, the results from the Python simulation are presented and discussed.

The ultimate goal of this project was to provide a proof of concept for the discretization technique outlined by Maire and Breil [2] as applied to a regular triangular mesh. This would suggest that the discretization is valid for solving the diffusion equation on non-orthogonal grids. Code written this summer will provide a foundation for developing a coupled radiation-hydrodynamics code for performing Lagrangian simulations on unstructured triangular grids.

Theoretical Background

In this section, a brief introduction to the underlying equations is provided. Implicit Monte Carlo (IMC) is a method used to solve the time dependent thermal radiative transport equations using Monte Carlo techniques to implicitly solve for each time step. Discrete Diffusion Monte Carlo (DDMC) is an approximation to the time dependent thermal radiative transport equations formed by incorporating Fick's law of diffusion, which is accurate for highly scattering regions. Furthermore, DDMC solves the equations by Monte Carlo methods, while treating the time variable continuously. All of the work and derivations presented here have been cast in the gray form, which is to say that all photons are assumed to be at uniform energy (or frequency). Each of the concepts and methods are extendable to the multigroup approximation, which is formulated to treat the particle energy (frequency).

Implicit Monte Carlo (IMC)

The continuous transport equations begin as

$$\overbrace{\frac{1}{c}\frac{\partial I}{\partial t}}^{\substack{\text{time-rate}\\\text{of change}}} + \overbrace{\Omega\cdot\nabla I}^{\substack{\text{streaming}\\\text{losses}}} + \overbrace{\sigma I}^{\text{absorption}} = \overbrace{\sigma B}^{\substack{\text{radiative}\\\text{emission}}} + \overbrace{\frac{Q_r}{4\pi}}^{\substack{\text{external}\\\text{source}}} , \tag{1}$$

where I is the intensity of the radiation, c is the speed of light, σ is the opacity, B is a function of temperature and represents the emission of photons from the material, Q_r is an external source. In this form, no material motion is assumed. This equation is coupled to the material energy balance equation presented as

$$\overbrace{C_V\frac{\partial T}{\partial t}}^{\substack{\text{time-rate}\\\text{of change}}} = \overbrace{\int_0^\infty\int_{4\pi}\sigma(I-B)d\Omega' d\nu'}^{\substack{\text{absorption/emission}\\\text{from material}}} + \overbrace{Q_m}^{\substack{\text{external}\\\text{source}}} , \tag{2}$$

where C_V is the heat capacity, and Q_m is an external source. Note that the radiative emission term in Eq. 1 appears as a source that also appears as an energy sink in Eq. 2. Similarly, the absorption term appears as a sink in Eq. 1 and a source in Eq. 2. Furthermore, each of the material properties (e.g., σ, C_V, etc.) can be functions of temperature, thus these two equations are highly coupled.

The solution to these equations is outside the scope of the present project, but one technique with which to proceed is to discretized the problem in time, and treat particles that are absorbed and re-emitted in a given time step as an effective scatter. This concept was described by Fleck and Cummings [3], and is represented by the Fleck factor, $f \in [0,1]$, where f may be thought of as the ratio of effective absorption events to total events, and is approximated by

$$f = \frac{1}{1 + \frac{4acT^3\sigma\Delta t}{C_v}} . \tag{3}$$

The Fleck factor will be discussed in more detail later in this report. The eventual extension to this work would link together IMC and DDMC solvers. The challenge in this case is to ensure that particles would successfully transition to the correct type of particle as it travels throughout the problem space. DDMC is only accurate in highly scattering regions, which is the region where IMC is most expensive. Thus a coupled solver would use DDMC as a way to speedup an IMC method.

Discrete Diffusion Monte Carlo (DDMC)

As previously mentioned, in domains where a large amount of scattering occurs, an IMC particle will have a mean free path which is much smaller than the dimension of the spatial cell in the problem domain. As a result, the IMC particle undergoes a large amount of effective scatters or absorptions before it leaves the spatial cell, this process is computationally expensive. If the Fleck factor, f, is very small then the majority of events that the photon will undergo will be isotropic effective scatters. In this regime it is valid to take the zeroth moment of the IMC transport equations to obtain the diffusion equation for radiative transport [1]. Doing so yields the following

$$\frac{1}{c}\frac{\partial E}{\partial t} + \frac{1}{c}\nabla F = f\sigma_t acT^4 - f\sigma_t E , \tag{4}$$

where E is the energy in the radiation field (related to I from Eq. 1), F is the radiation energy flux, a is the radiative constant, and T is the temperature.

The radiative flux F is related to the energy E through Fick's Law as

$$F = -cD\nabla E , \tag{5}$$

where the diffusion coefficient D may be temperature dependent and is inversely related to the opacity by

$$D = \frac{1}{3\sigma} . \tag{6}$$

Furthermore, the diffusion equation is strongly coupled to the material energy equation

$$\rho C_v \frac{\partial T}{\partial t} = f\sigma E - f\sigma acT^4 \tag{7}$$

where σ is the total opacity for the material. It is important to note that DDMC is only a valid approximation in highly scattering regions, or where the Fleck factor is small. Outside of this region, the IMC equations should be solved instead.

Discretization

In order to solve the DDMC equations, the Finite Volume technique was applied to a spatially discretized domain. specifically, a structured spatial grid of right triangles was chosen for the discretization. A sample of the mesh is presented in Fig. 1.

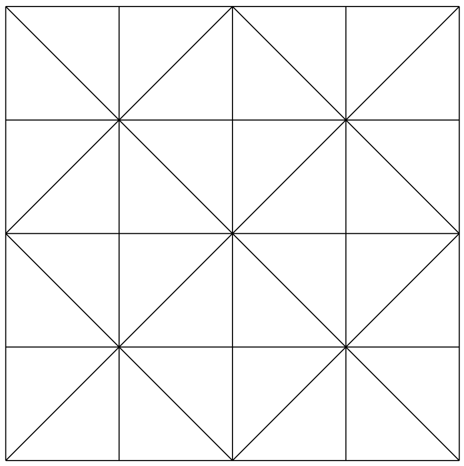


Figure 1: Schematic of spatial grid used to discretize problem

Integration over each spatial cell is trivial for all terms in Eq. 4 except for the gradient operator. The divergence formula may be applied to express the cell centered energy values in terms of unknown face energy values. A rigorous method for obtaining a discrete representation for the gradient operator was a large focus of our research project. To do so, a discretization formulated by Maire and Breil [2] was explored. These mathematicians obtained a discrete representation of the gradient operator for any two-dimensional unstructured mesh. The structured mesh is a subset of the unstructured mesh, thus fully applicable to the chosen triangular grid.

Maire and Breil's discretization method involved overlaying the existing mesh with a dual mesh. This dual mesh was created by connecting the center of each cell face to the cell centroid. This effectively divided the cell into subcells, the number of which being equal to the number of cell faces. The dual mesh overlay creates a system of subcells surrounding a node; for this mesh type there are two types of nodal subsystems and a schematic of each is provided in Fig. 2.

1

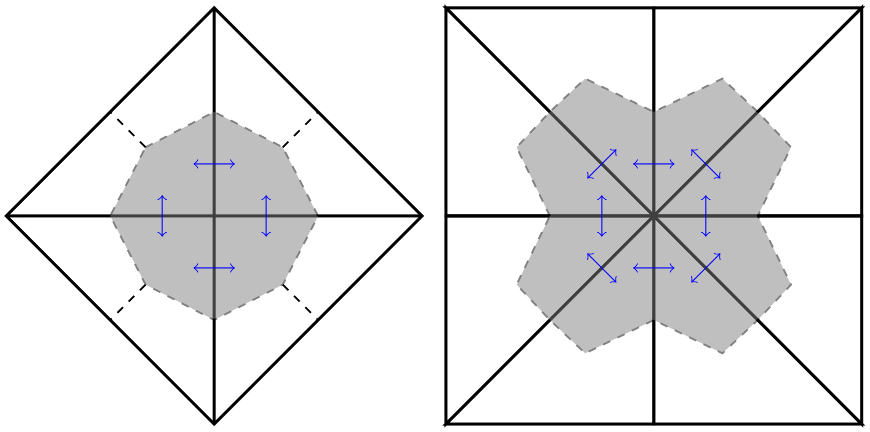


Figure 2: The two nodal subsystems that arise in our triangular mesh. Solid black lines represent the spatial mesh. Dashed lines represent subcells arising from the dual mesh. The grey area comprises the subcells that are part of the nodal subsystem. Blue arrows indicate fluxes across subcell faces.

In this figure the gray subcells surrounding a node represent a single nodal subsystem. To each subcell, we apply a variational principle. Next, a series of modifications to the resulting integral are made by employing the divergence formula and integration by parts. Finally, the system is closed by requiring that the energy fluxes be continuous across the sub-cell faces. In the figure, fluxes are represented by blue arrows. For full details on this discretization, the reader is referred to the original work [2].

This process allows one to obtain the face-centered energy values in terms of cell centered energy values, yielding a scheme in which all primary unknowns exist at the cell center. The manner in which face-centered values are obtained in terms of cell-centered values is through the solution of a linear system at each node. Then, the values are propagated back up to the global linear system and resolved purely in terms of cell-centered unknowns.

The discretization requires the assumptions that the flux F is constant over a subcell, the diffusion coefficient D is constant throughout a cell, and that none of the angles in the mesh are greater than $\pi/2$. With these assumptions, the DDMC equation in energy for cell i is converted to

$$\frac{1}{c}\frac{\partial E_i}{\partial t} + \left(f\sigma_i + \sum_j^{N_i} \sigma_{i,f_j} \right) E_i = f\sigma_i a_i c T_i^4 + \sum_j^{N_i} \sigma_{i,f_j} E_{i,f_i} , \tag{8}$$

where σ_i is the total opacity for cell i, N_i is the number of subfaces for cell i, E_{i,f_j} is the energy at the subface j of cell i, and σ_{i,f_j} is the effective leakage opacities from cell i to subface j on that cell. The effective leakage opacities are given by the following equation

$$\sigma_{i,f_j} = l_{i,j}^2 \frac{3D_i}{(\Delta V_i)^2} \left(1 - 2\frac{l_{i,j-1}}{l_{i,j}} \mu_{i,j} \right) , \tag{9}$$

where $l_{i,j}$ and $l_{i,j-1}$ are the lengths of the subfaces on cell i that bound the vertex connected to subface j, $\mu_{i,j}$ corresponds to the angle for cell i connected to subface j, and ΔV_i is the volume of cell i. To close Eq. 8, a continuity condition for the fluxes across a subface is employed to provide a relationship for E_{i,f_j}. After rearranging the continuity condition, a relationship for the energy at a half face is developed, which is given as,

$$\begin{aligned} E_{i,j} = \frac{1}{\frac{D_i}{V_i}+\frac{D_{i-1}}{V_{i-1}}} \Bigg[& E_i \frac{D_i}{V_i}\left(1-\frac{l_{i,j+1}}{l_{i,j}}\mu_i\right) + E_{i-1}\frac{D_{i-1}}{V_{i-1}}\left(1-\frac{l_{i,j-1}}{l_{i,j}}\mu_{i-1}\right) \\ & E_{i-1,j-1}\frac{D_{i-1}}{V_{i-1}}\frac{l_{i,j-1}}{l_{i,j}}\mu_{i-1} + E_{i,j+1}\frac{D_i}{V_i}\frac{l_{i,j+1}}{l_{i,j}}\mu_i \Bigg], \end{aligned} \tag{10}$$

where each of the subscripts on E correspond to a location as outlined by the right portion of Fig. 4, i.e., $E_{i,j}$ is the energy at the subface j and cell i, $E_{i,j+1}$ is the energy at subface $j+1$ in cell i, E_{i-1} is the energy at the cell center of cell $i-1$, etc. This closure relationship is only valid for triangles as the volume for each subcell has been converted to volume of the entire cell.

Finally, note that this discretization results in a linear system ($Ax = b$) with desirable properties. The method ensures that the coefficient matrix A is both semi-positive definite as well as L_2 stable. The result of semi-positive definiteness ensures that the probabilities interpreted from the matrix are bounded by zero and unity. In the next section we will delve more deeply into how we obtain probabilities from a linear system. In addition L_2 stability guarantees that for a boundary value problem, in the absence of sources, we obtain a radiation energy field which obeys the maximum principle. It is for these highly desirable properties, as well as applicability to unstructured grids, that we chose to implement this discretization.

Solution of the Linear System

Using a completely deterministic algorithm we would obtain a global system of equation of the form

$$Ax = b, \tag{11}$$

where A is an $n \times n$ matrix of coefficients with n being the number of phase-space (space, angle, frequency) cells. A is dependent on the grid geometry as well as material properties. x is the unknown cell centered energies (or temperatures) and b are sources. However, a Monte Carlo solution of the system using probabilities interpreted from the linear system is sought. A Monte Carlo approach allows one to solve transport (IMC) and diffusion (DDMC) regimes simultaneously, whereas deterministic methods would require a boundary condition between the two regimes which must be iterated over for self-consistency. In addition, a Monte Carlo solution technique allows use of a continuous time variable instead of discretizing in time; thus, when DDMC particles are re-emitted into a transport regime from the diffusion regime, the IMC particle will always know its time instead of requiring additional sampling [4].

The method for obtaining a probabilistic interpretation from the matrix of coefficients is detailed by Cleveland [5] and only a brief overview is presented here. A DDMC particle in cell i has the probability, P_{ij} of leaking to a cell j, where this probability is given by

$$P_{ij} = a_{ij}/a_{ii}, \tag{12}$$

where a_{ij} is the element in the ith row and jth column of the coefficient matrix A. By obtaining the probabilities in this way, it can be demonstrated that in the limit of simulating an infinite number of DDMC particles, the Monte Carlo simulation exactly obtains the cell centered energy values x from Eq 11.

While the stochastic inversion method described above is perfectly valid, an additional step is taken. As previously described in the discretization section, each node retains a linear system which resolves the face values purely in terms of cell centered values. Thus, an identical methodology is used to invert the nodal subsystem as was applied to the global system. The result of this will be many linear systems, where the global system will be used to build cumulative distribution functions (CDFs) at the cell centers. Sampling from these CDFs will then evolve a DDMC particle to one of the cell half faces.

In addition to the global system, there exists a set of nodal linear systems represented as

$$\bigcup_{n=1}^{\text{\# of nodes}} A_n x_n = b_n \,. \tag{13}$$

From each nodal subsystems, a CDF is constructed for each half face. These half face CDFs are then used to evolve a DDMC particle to another cell face or back to a cell center.

Note that due to the discretization applied, DDMC particles may now travel to face neighbors as well as nodal neighbors. To illustrate, consult Fig. 3 for possible final locations that a DDMC particle can move to in a single scattering event from a given cell. However, as a result of solving the continuity equations via Monte Carlo, one to one connectivity is maintained such that in order to move to a grey cell a particle must pass through a blue cell.

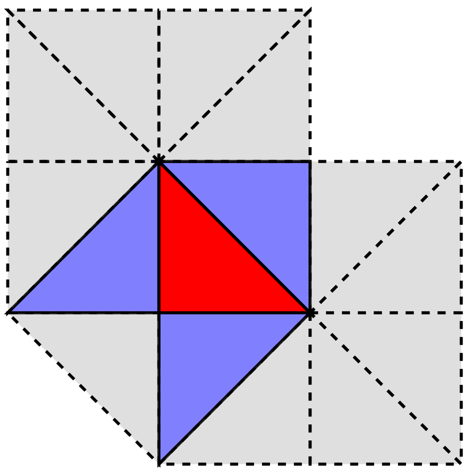


Figure 3: A DDMC particle initially located in the cell center of the red triangle can move not only to the face neighbors, highlighted in blue, but also to all nodal neighbors, highlighted in grey

Furthermore, in the case of an orthogonal, Cartesian mesh this scheme reduces to the five point stencil one would obtain via the familiar finite differencing scheme. In the orthogonal grid case transport only occurs between face neighbors.

Technical Approach

In this section, a brief summary of the Monte Carlo method is discussed. Much of the technique for solving the DDMC equations is comparable to any Monte Carlo approach used to simulate particles. The first step is to initialize several parameters (e.g. f) from the material properties. This includes determining how many source particles to simulate from a given cell. This is computed by defining

$$S_e = ca\Delta t \sigma_a T^4 \,, \tag{14}$$

where S_e is the source emission for a given cell, σ_a is the effective absorption opacity, and all other parameters were previously defined. The absorption opacity is given by

$$\sigma_a = f\sigma \,. \tag{15}$$

The next step is to determine the ideal weight of a particle. First compute the total emission source by summing the cell dependent S_e terms. Next, the ideal weight is computed by dividing the total emission source by the number of particles to simulate. Finally, the number of particles to simulate for a given cell is S_e for that cell divided by the ideal weight. If S_e for a cell is greater than zero, then ensure at least one particle is emitted from that cell. Finally, the initial weight for a particle born in cell i is determined by dividing S_e for cell i by the number of particles emitted within cell i. The remaining particle initialization parameter is time. All particles are uniformly distributed along a time step, i.e.,

$$t_p = \xi \Delta t \,, \tag{16}$$

where t_p is the particle time, and $\xi \in [0,1]$ is a random number.

Particles are always started at the cell centers (with exception of boundary conditions discussed later in this report). The first step in moving a particle is to compute the time to the next collision. In analog absorption, this function is given as

$$t_c = \frac{-\log(\xi)}{c\sigma_a + c\sum_{f_i}^{N_f} \sigma_{f_i}} \,, \tag{17}$$

where t_c is the time to the next collision, f_i is subface i, σ_{f_i} is the effective opacity (probability) of moving to subface i, and N_f is the number of subfaces for the cell (6 for the triangular mesh). This formula may be adjusted by implementing various variance reduction techniques (i.e., implicit capture), but these methods will not be discussed here.

Now check if $t_c + t_p \geq \Delta t$. If so, the next collision occurs in a future time step, and the particle is stored in census to be computed at the next time step. Otherwise, the next collision will occur, and a random number is drawn to determine if the particle is absorbed in the current cell. If so, the particle deposits its weight in the absorption tally for the current cell, and the particle is considered dead. If the event is not an absorption, the particle moves to a half face for the current cell by sampling from the face CDF as discussed previously.

Once on a face, a particle no longer moves in time until it returns back to a cell center. This is due to the half face movement being a result of only the closure relationship. The second CDF is used to determine how the particle moves from the half face as previously discussed. This movement on half faces is unique to Monte Carlo methods and has been the focus of this work. Once the particle returns to a cell center, a new value for t_c is computed and the process continues until the particle is absorbed.

For cells with a subface on a problem boundary, the CDFs are adjusted to account if it is an albedo vacuum condition or a reflective condition. If it is reflective, the particle will move back to the current cell if it tries to move across a reflective face, while if vacuum, the particle has a chance to leak out of the problem and its weight (energy) escapes.

A Dirichlet boundary condition was also implemented, where the temperature at the boundary is set, and a number of particles is determined similarly to the previous process, however, all of the particles are sourced onto the subface on the boundary as opposed to the cell center. From that point, the particles are able to leak from the problem like the albedo vacuum condition or move deeper into the problem by sampling the appropriate CDFs.

Once all the particles for a given time step have either escaped from the problem, gone to census, or have been absorbed, the new temperature is computed for the problem as

$$T_{\text{new}} = \rho C_v \Delta t \Delta E \,, \tag{18}$$

where ρ is the density, and ΔE is computed as the difference between the S_e and the absorption tally for a given cell. From here, the temperature dependent parameters are updated, and the next time step begins. The only remaining note is that particles that went to census are pulled from census at $t_p = 0$ with the same position and weight as when entering the census, and thus jump directly to determining the time to next collision. Finally, the energy in the radiation field E is given by the total weight of the particles in census at the end of the time step.

Computational Implementation

In addition to working through the details of the discretization and implementing a proof a concept into python, the mesh capabilities were implemented into the existing Jayenne code. Jayenne is LANL's code for performing simulations of thermal radiative transport via Monte Carlo methods where the radiation source is tightly coupled to the material. It is based on the IMC method developed by Fleck and Cummings [3]. Jayenne is highly parallelized to take full advantage of the high powered computing facilities at LANL.

Currently there does exist a DDMC solver in the Jayene code, however it is only applicable to orthogonal, Cartesian (or RZ) meshes. However, the discretization and solution technique used for solving the DDMC equations are general to an unstructured polygonal mesh, and is novel to this field. Therefore, the majority of capabilities for solving DDMC on triangles did not exist in the Jayenne code, and were implemented over the course of this work. The major capabilities added into the existing Jayenne code were a triangular mesh builder, a DDMC transporter for the triangular discretization, and a DDMC Builder for the discretization.

Furthermore, a triangular mesh class had to be implemented into the existing Jayenne code. As a reminder of the type of triangular mesh used refer to Fig. 1. The mesh object for other mesh types already contained capabilities for determining several geometric properties of the mesh. However, additional functionally was required for triangles including:

- Nodal Connectivity: A list of cells connected to each node
- Half Face Lengths: Length of the half faces of each cell
- Interior Angle: Interior angle at each vertex in the cell
- Face Normal: Normal vector of each face of a cell

These geometric quantities explicitly arise in the discretization utilized.

The next task was to implement a DDMC transporter object with increased capability. The DDMC transporter is the object which transports a DDMC particle through the diffusion regime. Previously, DDMC particles only existed at a cell center due to the orthogonal nature of the meshes used. However, due to the choice to use a stochastic method to invert the nodal subsystems, a DDMC particle may now exist at a cell face. Therefore the transporter must now be capable of transporting a DDMC particle from the cell center to a cell face, from a cell face to a cell center, and from a cell face to another cell face. A schematic illustration of these transport events is detailed in Fig. 4.

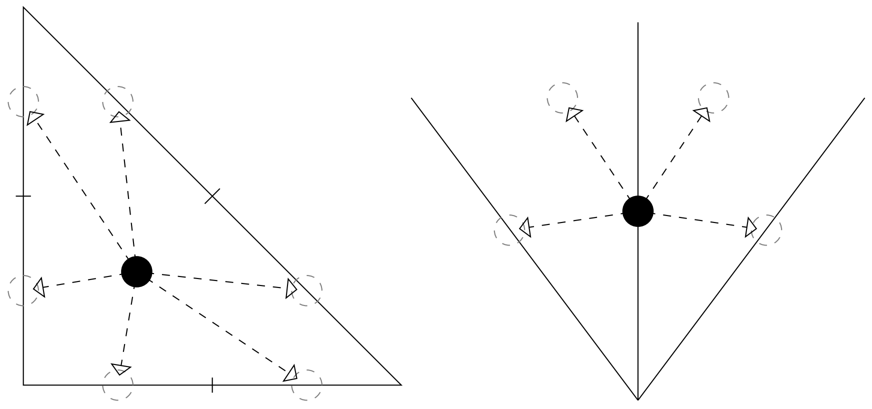


Figure 4: Left: Transport events from cell center to cell half faces. Right: Transport events from cell half face to either cell center or cell half face

Inherently linked with scattering events, which are implemented through the transporter, is the sampling of such an event. These events are sampled from a cumulative distribution function (CDF). The CDFs, as discussed previously, are constructed from the terms in the coefficient matrix of the global linear system as well as the nodal subsystems. As such they are inherently linked to the discretization used. With a new discretization, the DDMC Builder must be modified to build the necessary CDFs. This results in modifying the existing cell center CDFs to give the probability of a DDMC particle scattering to each cell half face. In addition, an entirely new capability was implemented for building the CDFs at each cell half face. These half face CDFs give the probability of a DDMC particle to move to a cell center or another cell half face.

With all these capabilities implemented, Jayenne now has the ability to perform DDMC simulations on a structured triangular meshes.

Test Problems

In order to test the implementation and provide a proof of concept in Python, several simple, 2-D, time dependent test problems were developed. Each of these test problems is summarized in Tab. 1. The first problem (RRRR) is to use reflective boundary conditions on all boundaries with no external source, thus representing an infinite homogeneous medium. It is expected that the temperature remain constant for all time for this problem.

The second problem (VVVV) is similar to the first, expect vacuum conditions are used on all boundaries. Without an external source, it is expected that the temperature of the system will decrease with increasing time as energy is leaking out of the system. The center should maintain a higher temperature than the cells on the boundary.

The third problem (VVRR) is to use reflective conditions on the $y = 0$ and $y = L_y$ boundaries and vacuum conditions on the $x = 0$ and $x = L_x$ boundaries. This problem will effectively replicate a 1-D simulation with vacuum conditions, and similar trends to problem 2 are expected. A variation of this problem (RRVV) is to flip the boundary conditions for the x and y bounds.

The fourth test problem (TwoT) is to use all reflective boundary conditions, but to initialize the problem at two different temperatures. The upper half of the problem is initialized at a lower temperature than the lower half of the problem. It is expected that the temperature will equilibrate to some mean temperature (dependent on material properties) as the problem approaches steady state.

A variation on the previous problem is the final problem (TwoTS). Instead of initializing the two temperatures on the upper/lower sections, the dividing line is initialized on the diagonal from $(0, l_y)$ to $(l_x, 0)$. Above the diagonal is a high temperature, and below is the low temperature. Similarly to problem four, the temperature should equilibrate as the simulation time approaches infinity.

Each of these problems were designed to test the proof-of-concept for this derivation in Python. At this time, we do not have an analytical solution with which to compare these test problems, but analytical and benchmark solutions are being developed for this purpose. The current implementation of the underlying Monte Carlo algorithm is implemented on a Cartesian mesh, and thus will be readily compared to the triangular solution once the triangular mesh class is fully implemented.

Each of the described test problems were solved with the following physical parameter definitions. Heat capacity C_v set to 0.5. Reference opacity σ_0 set to 10000. Total opacity σ assumed as $\sigma = \sigma_0/T^3$. Density ρ set to 1.0. Radiative constant a set to 1×10^{-6}. Length in x direction L_x set to 1.0. Length in y direction L_y set to 1.0. Furthermore, the number of rectangular cells in both x and y directions were set to 30, where each rectangular cell was split into two to create the triangular cells. Finally, the number of particles per time step was set to 5000.

Table 1: Summary of test problems

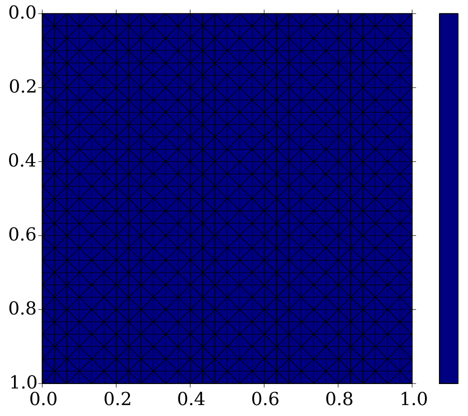
Abbreviation	Problem description
RRRR	All reflective BC, $T(t = 0) = 10$
VVRR	Vacuum BC on x, Reflective BC on y, $T(t = 0) = 10$
RRVV	Reflective BC on x, Vacuum BC on y, $T(t = 0) = 10$
VVVV	All vacuum BC, $T(t = 0) = 10$
TwoT	All reflective BC, upper $T(t = 0) = 5$, lower $T(t = 0) = 15$
TwoTS	All reflective BC, upper left $T(t = 0) = 15$, lower right $T(t = 0) = 5$

Simulation Results

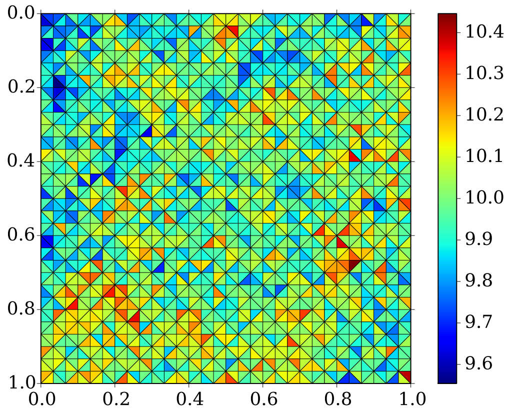
This section presents the results from the proof of concept implementation in Python. Each of the results presented in this section were computed using the previously defined parameters. Furthermore, each simulation was run to a maximum time of 200 s using a time step of 1 s.

Four figures are presented for each of the test problems. The first two show a two dimensional plot of the temperature for each cell at the initial time and the final time. The second two figures show temperatures that have been spatially averaged and presented as a function of space, e.g., averaging the cells in the y direction to reduce the simulation to a one dimensional simulation in x. The second two figures also present the averaged data for several different time steps.

Figure 5 shows the initial and final temperature distribution for the fully reflective test problem (RRRR). It is expected that the temperature would remain constant because there are no external sources, and no leakage. Within statistical variance, the temperature does remain constant at 10, as expected. An additional means of observing time dependent temperature is presented in Fig. 6, where the temperature remains roughly constant as a function of time both in the x and y directions.



(a) Initial time, $t = 0$



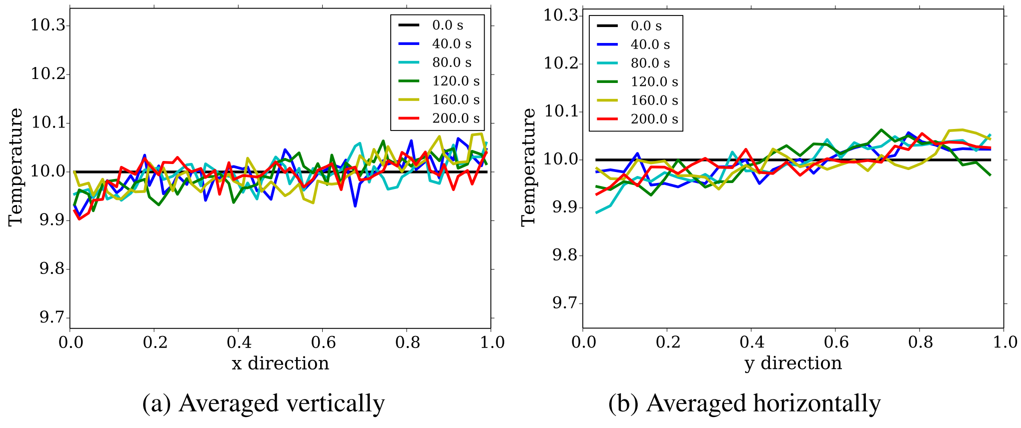
(b) Final time, $t = 200$

Figure 5: (RRRR) All reflective boundary conditions, initialized at $T = 10$.

The test problem VVRR is expected to vary only with x position, and to maintain a higher temperature on the centerline that on the x boundaries. This behavior is observed in Fig. 7. Since leakage is possible in this problem, the maximum temperature is expected to decrease as a function of time as is shown in Fig. 8. The left plot in the figure a parabolic temperature profile that decreases with increasing time, as expected, while the right plot shows a statistically flat profile that decreases with increasing time, as expected.

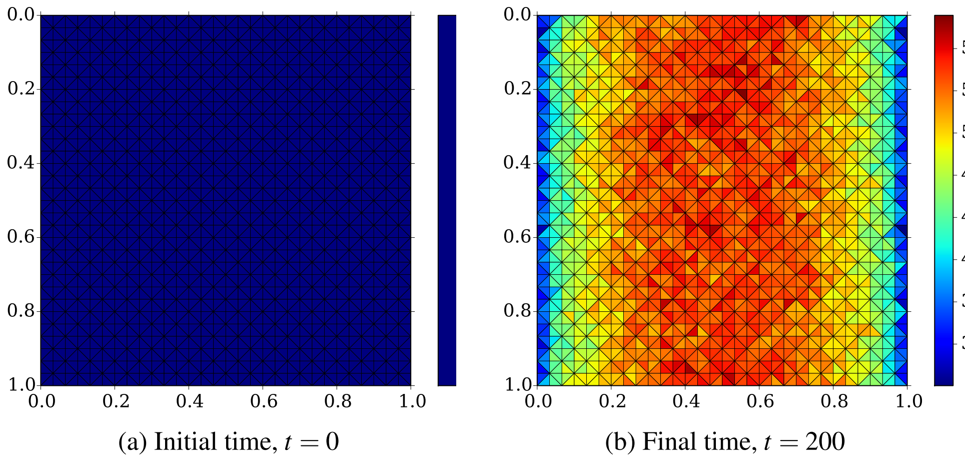
The same trends are expected for problem RRVV, but reversed as compared to problem VVRR. Figure 9 confirms expectations. Similarly, Fig. 10 shows a statistically flat profile for the x direction, and a parabolic profile in the y direction, as expected. Were the simulation run to a much higher time, the temperature would decrease to a minimum of zero everywhere.

With all vacuum conditions as in problem VVVV, the expectation is a higher temperature in the center, with low temperatures around the boundary. The maximum temperature should de-



(a) Averaged vertically (b) Averaged horizontally

Figure 6: (RRRR) All reflective boundary conditions, initialized at $T = 10$.



(a) Initial time, $t = 0$ (b) Final time, $t = 200$

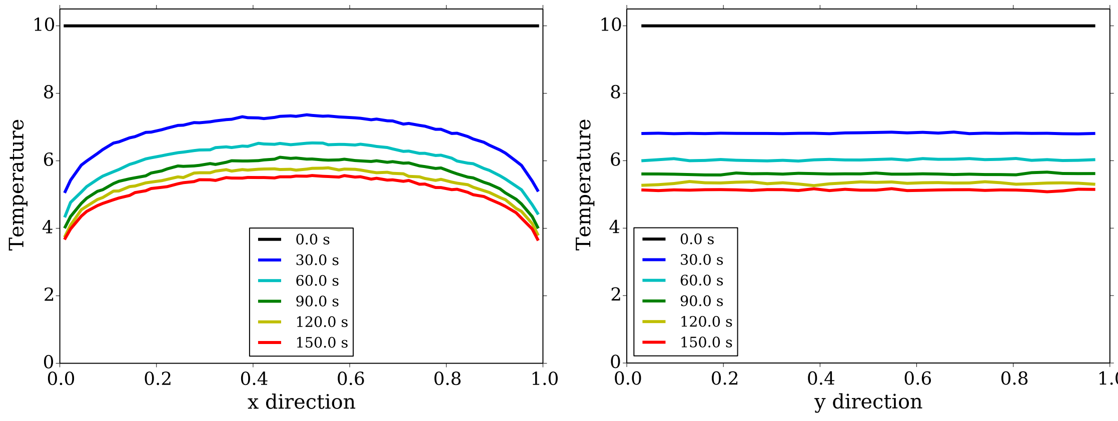
Figure 7: (VVRR) Reflective y boundaries, albedo vacuum x boundaries, initialized at $T = 10$.

crease with increasing time, which is observed in Fig. 11. The simulation is roughly symmetric in the x and y directions as observed in Fig. 12.

The first of the two temperature problems (TwoT) was expected to equilibrate at an average temperature of 10 as the material properties were constant throughout the test problem. Figure 13 confirms expectations, while Fig. 14 shows the equilibration rate. At the given inputs, it took less than 40% through the simulation to equilibrate and come to a relatively flat temperature profile.

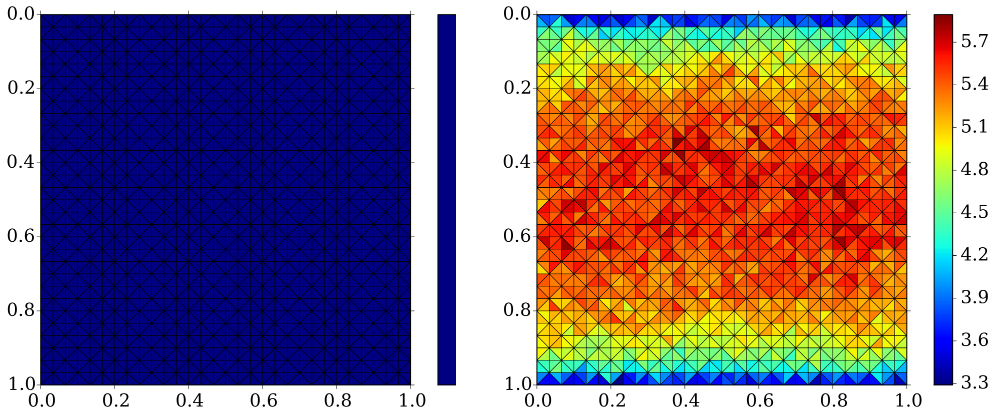
The same trends were expected for the slanted two temperature problem (TwoTS) as TwoT. Once again, expectations were confirmed by Fig. 15. As the dividing line was on the diagonal for this problem, averaging in the x and y directions provides comparable results, and the temperature equilibrates to the mean temperature by approximately 80 s, as shown in Fig. 16.

Note that on several of the test problems, some skewness to the results can be observed. This is especially apparent in problem RRRR, as observed in Fig. 6. There the upper left quadrant is slightly cooler than the lower right quadrant, which is suggestive of an underlying bug in the Python script. This bug has been identified and new results are being generated. In



(a) Averaged vertically (b) Averaged horizontally

Figure 8: (VVRR) Reflective y boundaries, albedo vacuum x boundaries, initialized at $T = 10$.



(a) Initial time, $t = 0$ (b) Final time, $t = 200$

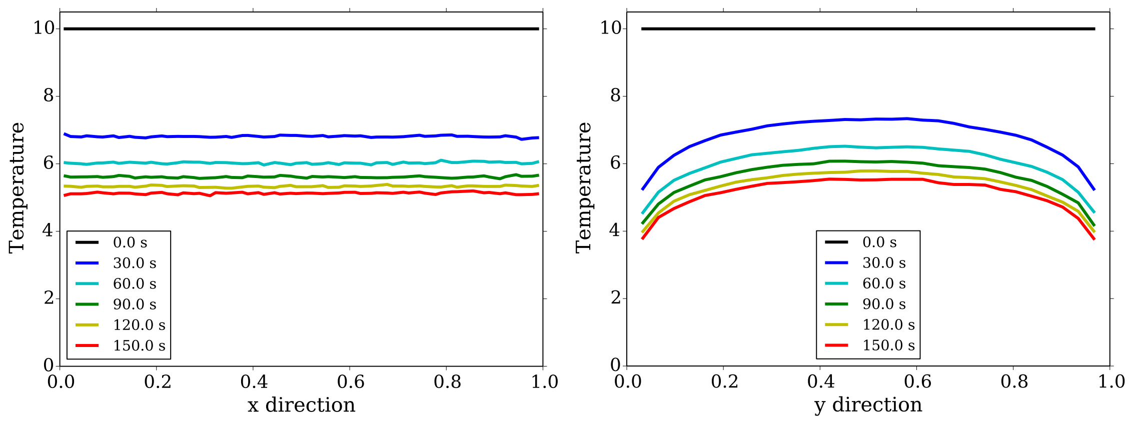
Figure 9: (RRVV) Reflective x boundaries, albedo vacuum y boundaries, initialized at $T = 10$.

this report, only the results from RRRR, TwoT, and TwoTS are affected by this error.

Conclusion

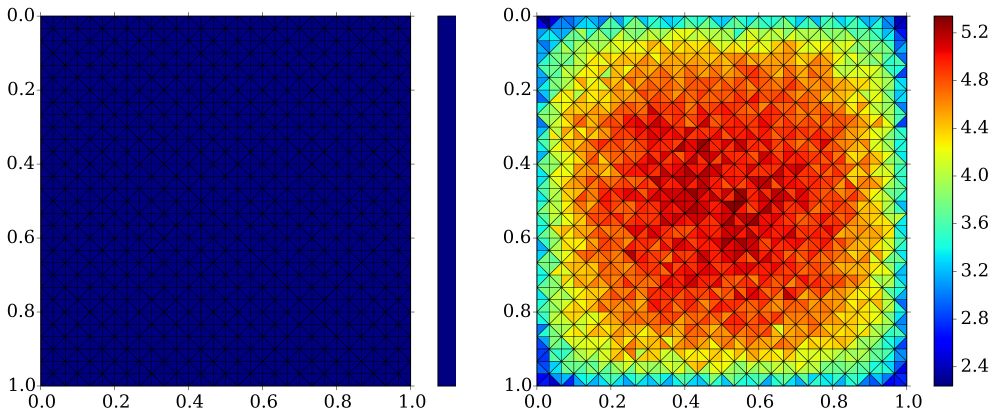
We have demonstrated a proof of concept for solving DDMC on a structured triangular mesh in Python as discussed in the previous section. The fundamental conclusion is that the Maire and Breil discretization is applicable to solving the DDMC equations for a structured triangular mesh, and should be applicable to unstructured triangular meshes. This proof of concept was demonstrated through a python implementation.

Furthermore, a capability for performing the same DDMC calculation now exists in the Jayenne Project. The remaining effort for this work is to test the functionality in Jayenne against analytical solutions as well as compare the test problems described in this report to corresponding problems designed with a 2D Cartesian mesh. Furthermore, timing studies are appropriate for comparison between the triangular and Cartesian meshes, and will be com-



(a) Averaged vertically (b) Averaged horizontally

Figure 10: (RRVV) Reflective x boundaries, albedo vacuum y boundaries, initialized at $T = 10$.



(a) Initial time, $t = 0$ (b) Final time, $t = 200$

Figure 11: (VVVV) All albedo vacuum boundary conditions, initialized at $T = 10$.

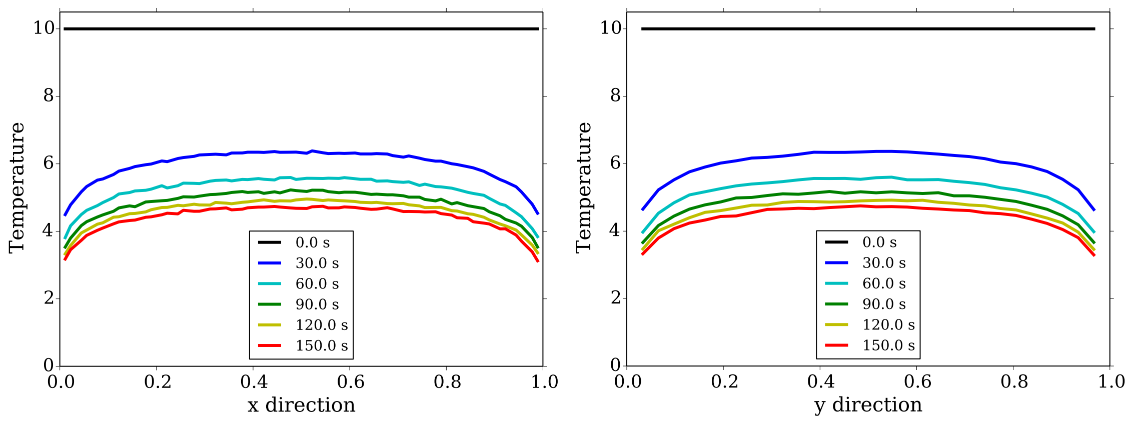
pleted in further study.

The work presented in this section is the first part of a five year project that will fully implement the discrete diffusion Monte Carlo on unstructured triangular meshes. It will eventually be linked to the existing implicit Monte Carlo routines to solve fully time depended, energy dependent, 3-D transport problems.

This work was completed as part of the Computational Physics Workshop 2015 using resources provided by Los Alamos National Laboratory.

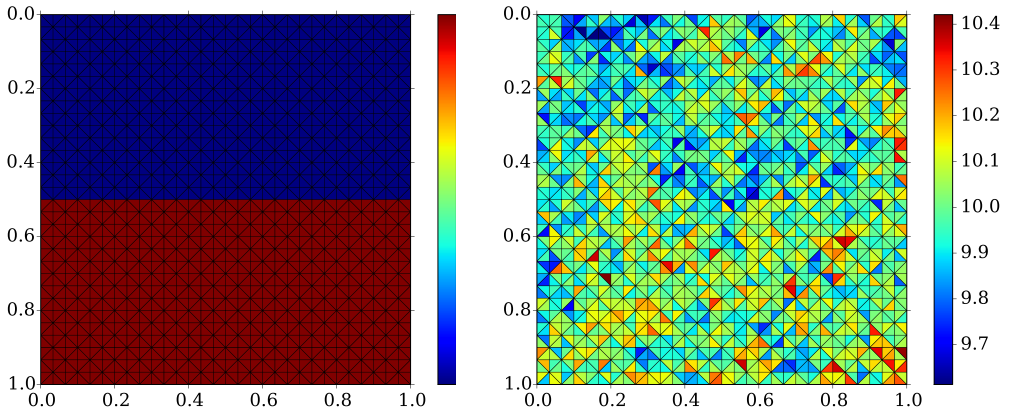
References

[1] N. Gentile. Implicit monte carlo diffusion an acceleration method for monte carlo time-dependent radiative transfer simulations. *Journal of Computational Physics*, 172(2):543–571, 2001.



(a) Averaged vertically (b) Averaged horizontally

Figure 12: (VVVV) All albedo vacuum boundary conditions, initialized at $T = 10$.



(a) Initial time, $t = 0$ (b) Final time, $t = 200$

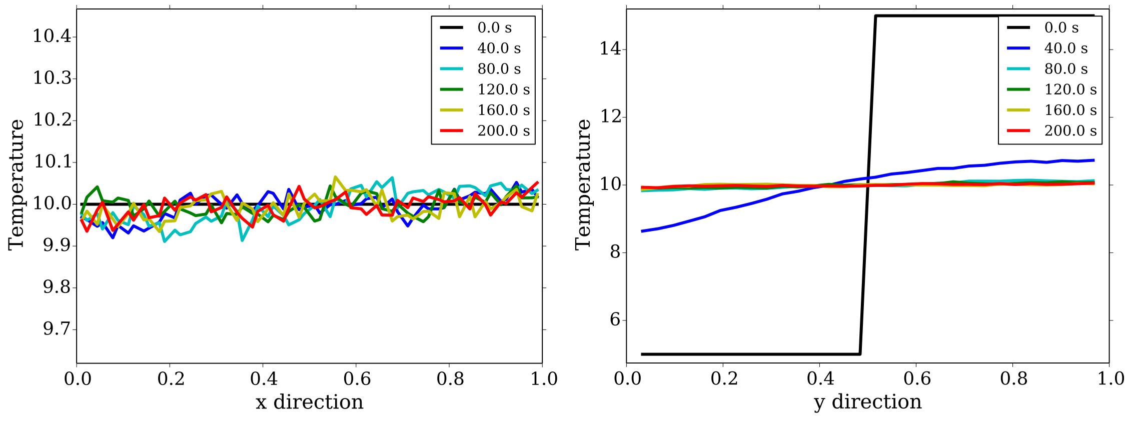
Figure 13: (TwoT) All reflective boundary conditions, upper initialized at $T = 5$ and lower initialized at $T = 15$.

[2] Pierre-Henri Maire and Jérôme Breil. A nominally second-order accurate finite volume cell-centered scheme for anisotropic diffusion on two-dimensional unstructured grids. *Journal of Computational Physics*, 231(5):2259–2299, 2012.

[3] Jr J. A. Fleck and J. D. Cummings. An implicit monte carlo scheme for calculating time and frequency dependent nonlinear ratiation transport. *Journal of Computational Physics*, 8:313–242, 1971.

[4] J. D. Densmore, R. J. Urbatsch, T. M. Evans, and M. W. Buksas. A hybrid transport-diffusion method for monte carlo radiative-transfer simulations. *Journal of Computational Physics*, 222(2):485–503, 2007.

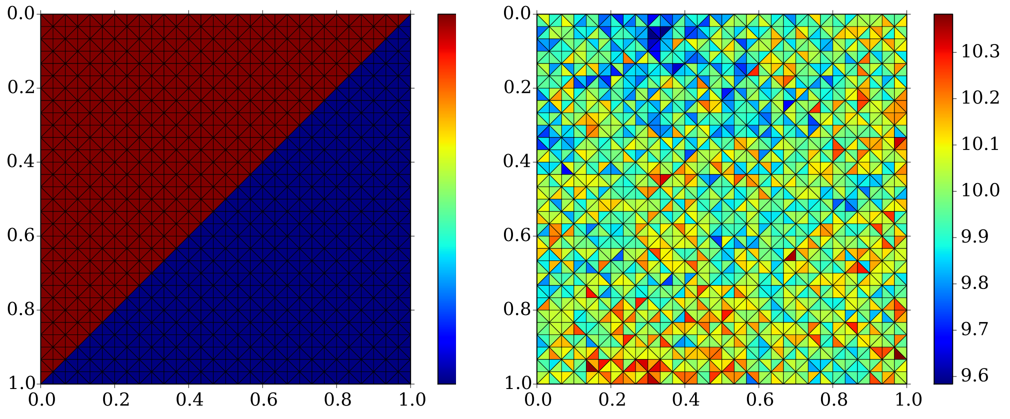
[5] Todd S. Palmer Mathew A. Cleveland, Nick A. Gentile. An extension of implicit monte carlo diffusion: Multigroup and the difference formulation. *Journal of Computational*



(a) Averaged vertically (b) Averaged horizontally

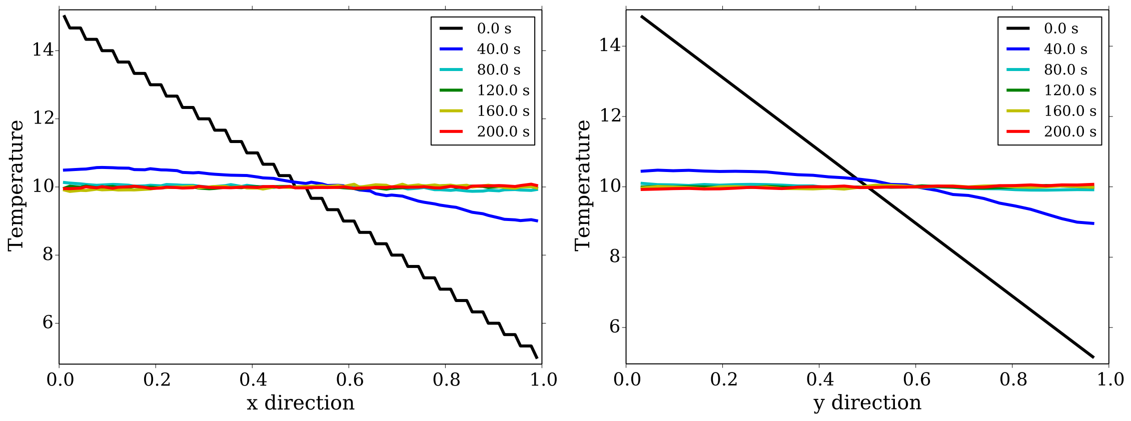
Figure 14: (TwoT) All reflective boundary conditions, upper initialized at $T = 5$ and lower initialized at $T = 15$.

Physics, 229(16):5707–5723, 2010.



(a) Initial time, $t = 0$ (b) Final time, $t = 200$

Figure 15: (TwoTS) All reflective boundary conditions, upper left initialized at $T = 15$ and lower right initialized at $T = 5$.



(a) Averaged vertically (b) Averaged horizontally

Figure 16: (TwoTS) All reflective boundary conditions, upper left initialized at $T = 15$ and lower right initialized at $T = 5$.

Production Codes on Xeon Phi Knights Corner

Team Members
Isaac Lyngaas and Josey Stevens

Mentor
Louis Vernon

Abstract

With increases in processing power coming from increased numbers of cores instead of improvements in per core speed, efficient parallel approaches must be taken to achieve performance on todays supercomputing systems. This becomes increasingly important as new parallel hardware and special vector units bring parallelism to levels not reached before. Here we explore the usage of efficient parallel algorithms on two production codes used in Institutional Computing (IC) at Los Alamos National Labs: CPIC and MPAS-O.

Introduction

Moore’s Law, Dennard Scaling, and Multi-Core Processors

In 1965 Gordon Moore published a paper in which he described (predicted) that the number of transitors that can be manufactured on a silcon wafer would double every year. In 1975 this prediction was revised to every two years. This scaling became known as Moore’s Law. [4]

In similar fashion, in 1974 Robert Dennard co-authored a paper stating that as transistors get smaller their power density remains constant. This manifests in a way such that the power use of the processor is proportional to the area of the processor die. [1]

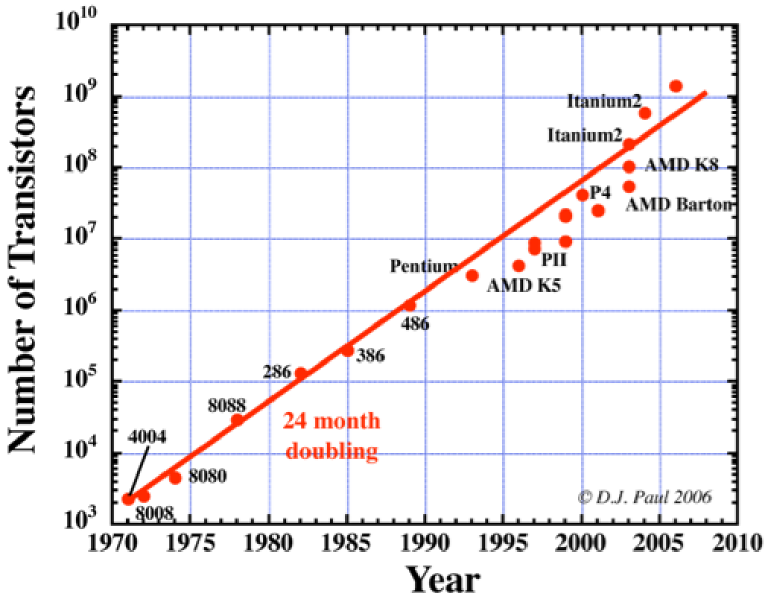


Figure 1: A graph of the transistor count per CPU as a function of time. The linear data in the logscale shows the strong correlation to Moore’s law.
[6]

Dennard scaling combined with Moore’s law has led to a situation in which processing power has increased steadily year afterr year. Unfortunately, in 2005 Dennard scaling began to fail and power use has not scaled proportionally as transistor size becomes smaller. This has led to a situation in which increases in per-core performance has began to stall.

In order to continue the increase in performance, processor manufacturers have moved to integrating many processor cores onto one chip to create multi-core processors. While multi-core processors have continued to increase the total performance of a processor, this performance gain has not made its way to serial programs. In order for one program to take advantage of multi-core processors (and clusters) parallelism must be exploited.

Memory Bandwidth Bottleneck

In order to fully utilize a processor's performance the processor must be able to access the data on which it is doing work at the same rate that it can opperate on the data. For many years memory bandwidth (the rate at which data can be transfered from RAM to the processor) was greater than the processor speed; however, computational performance has grown much faster than memory bandwidth.

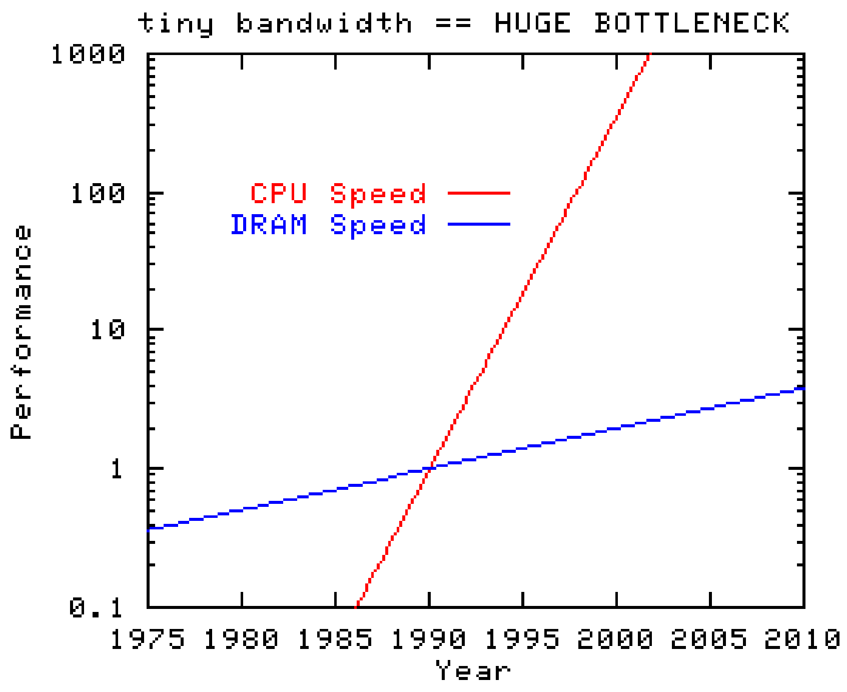


Figure 2: A graph of the scaling of both CPU and RAM performance over time. Notice how large the gap between CPU and RAM performance has grown.
[3]

In an effort to limit the impact of low relative memory bandwidth, manufacturers incorporated cache into the processors. Cache is low capacity, high bandwidth memory that exists closer to the processor. If a program is structured such that a small amount of data is accessed frequently, that data will be stored in cache and access to that data will not be the limiting factor.

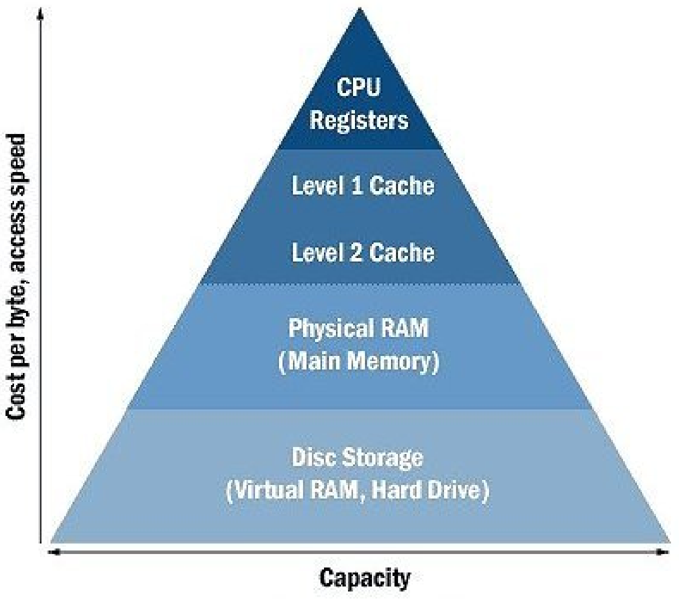


Figure 3: A graphic representing the memory hierarchy inside of a modern computer. [8]

Parallel Computing

As mentioned above, in order to utilize multi-core processors and computing clusters, parallel programs must be written. While there are countless parallel frameworks and languages, here we will be discussing two of the most popular parallel forms, MPI and OpenMP.

MPI

The majority of High Performance Computing (HPC) codes use the Message Passing Interface (MPI) to achieve parallelism. MPI is a distributed memory parallel construct. Distributed memory means that each *process* (each individual parallel thread of execution) has a private portion of RAM seperate from all other processes even if they share physical RAM modules (e.g. are on the same node). Since each process has its own private memory space, MPI lends itself to parallelism both on a single node (all cores share the same memory address space) or on a cluster (many different nodes connected through a network).

MPI is a Shared Program Multiple Data (SPMD) parallel construct. This means that each process exicutes the exact same code; however, each process has a unique ID (integer) that can be used as a parameter to differentiate the runtime. Since each process has its own private memory, if one process is to get data from another process, the information must be explicitly communicated. MPI employs sychronous communication, the simplest example of which is; process 1 calls MPI_SEND() to process 2 and process 2 calls MPI_RECEIVE() from 1. If (for example) process 1 calls MPI_SEND() before process 2 calls MPI_RECEIVE(), then process 1 waits until process 2 calls MPI_RECEIVE(), then the communication is completed and both processes continue execution.

OpenMP

Open Multi-Processing (OpenMP) is a very popular parallel construct that is less utilized in scientific computing but has been adopted in comercial applications. OpenMP is not typically the primary form of parallelism in scientific computing because scientific computing generally relies heavily on distributed computing clusters, while OpenMP only enables intra-node parallelism. Unlike MPI which creates processes, OpenMP creates threads, while each process has a private memory address space, OpenMP threads share address spaces. Data does not have to be communicated since all threads share and can access the same set of memory. Threads, unlike MPI processes, are not connected to only one processing core, and threads are not necessarily persistent for the entire execution, unlike MPI processes. Threads can be created dynamically thoughout a program. In contrast to MPI which is strictly SPMD, OpenMP can operate as SPMD but can also operate as Multiple Instruction Multiple Data (MIMD) in the form of intrinsics that "automatically" parallelize loops.

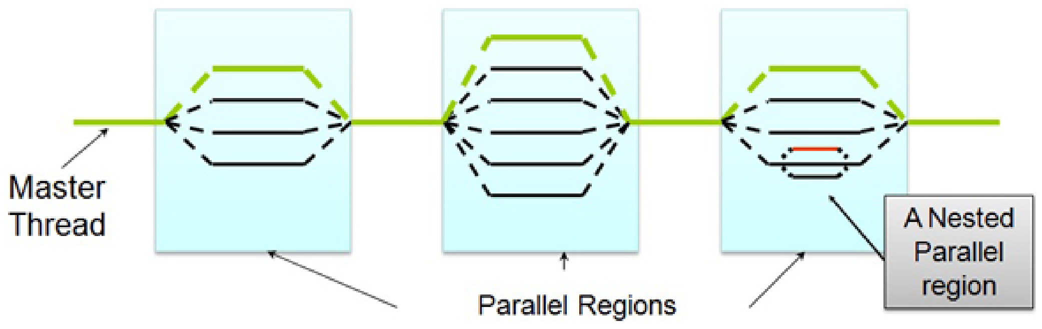


Figure 4: A graphic showing how a program utilizing openMP could dynamically spawn threads at runtime.

[8]

Issues with MPI

While MPI is a very powerful parallel construct, problems can arise in the implementation of MPI in production codes. One of the largest issues occurs during node to node communication. Node to node communication relies on a network connection (ethernet or infiniband), and can take considerably longer than any other (basic) operation in computing. The time added in MPI communication (and all other MPI calls) is often called MPI overhead. If a program requires a relatively large amount of communication then MPI overhead will comprise a considerable amount of the runtime. MPI overhead almost always increases as the number of MPI processes increases, which means that there is diminishing returns as additional processes are used. In practice, most large scale scientific codes are ultimately limited due to MPI overhead.

Another issue with MPI arises from its sychronous communication patterns. A problem arises when one process (or a few processes) takes considerably longer than other processes to reach the MPI_SEND() or MPI_RECEIVE(). As mentioned above, all other processes must block before they do any more work; when a significant portion of execution time is spent waiting for synchronous MPI communication, this is called load-imbalance. Depending on the program, load-imbalance can heavily impact the time of execution.

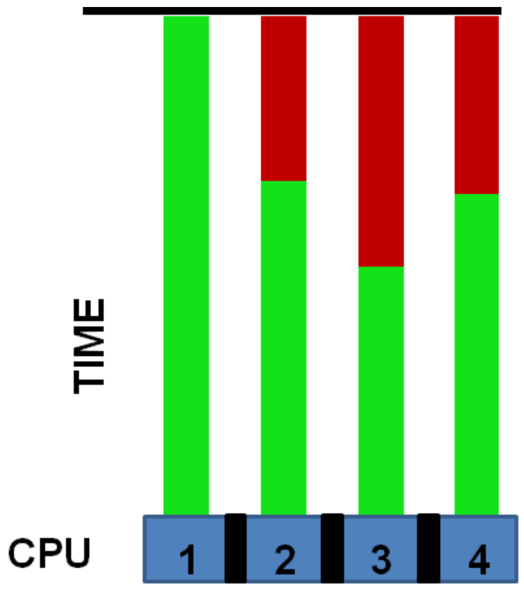


Figure 5: A graphic representing load-imbalance. Green is the time that a core is computing, red is the time the core is idle.

Issues with OpenMP

OpenMP also suffers from overhead; however, since there is no communication time (since all threads share the same memory) most of this overhead comes from spawning (including memory allocation) the threads. Also, parallel implementations of algorithms may induce overhead. One common issue that can occur with OpenMP that does not happen with MPI is the *race condition*. A race condition occurs when two threads attempt to perform an operation on the same data at the same time. For example (during a reduction operation), if two threads both read the value of x(stored in memory), one thread does x=x+1 and the other does x=x+2, then they both try to save the value, whichever value is written into data last is the persistent one. That is, the new value of x will only be x+1 or x+2 not x+3. The programmer has to be very aware of race conditions and must be vigilant in preventing them. A less intuitive but prevalent concern with OpenMP is *False Sharing*. False sharing occurs when two or more threads are acting on independent data elements that live within the same cache-line. If one thread updates an element in the cache-line, the cache is poisoned for the other threads and must be synchronized in order to retain coherency.

Parallel Computing Hardware

With the break down of Dennard scaling at smaller transistor sizes, hardware companies have had to get creative in order to develop new processors that are able to do more operations per second than traditional CPU processors, due to power and thermal constraints. Intel's solution for overcoming the limitations of traditional CPU processors is the Xeon Phi. The idea behind the Xeon Phi is to use cores with lower clock speeds (and therefore lower current leakage) in order to dramatically increase the core density achievable on a single processor. If all of these cores are able to run at the same time, then peak floating point performance can be much higher than a traditional CPU processor. Some other features added to the Xeon Phi for added parallel computation are four hardware threads and two extended vector units per core. Table 1 shows the specifications of the current generation of Xeon Phi the Knight's Corner (KNC), the next generation of Xeon Phi the Knight's Landing (KNL), and a node with two current Haswell CPU processors. Specifications for the KNL are based on pre-released information from Intel.

Table 1: Hardware Specifications

	Haswell Node	KNC	KNL
Cores	32	60-61	74
Clock Speed(GHz)	2.30	1.053	1.X GHz
GFLOPs	.588	1.011	> 3
RAM	128 GB DDR4	8-16 GB DDR5	16 GB MCDRAM + 92 GB DDR4
Vectorization Unit	256 bit	512 bit	512 bit
Processor Type	Host Processor	Co-Processor	Host/Co Processor

With the KNL, Intel is attempting to improve on some of the problems of the KNC through incorporating faster, better performing cores. The KNL will have much more memory than the KNC which will allow the execution of larger problems. The memory constraints of the KNC is a major problem with some codes as will be shown later in this report. Another problem with the KNC is that it is a co-processor, and does not support the standard x86-ISA, meaning it does not support the host operating system, compilers, etc. Building existing MPI/OpenMP code will be much easier on the KNL, with no cross-compilation required.

Vectorization

One advantage Xeon Phis have over traditional CPUs is enhanced vector units. These extended vector units, up to eight times speed up with double precision and sixteen times speed up with single precision is possible in the perfect case. As such it is important to identify as much vectorization as possible to approach peak performance.

Vector units essentially give an extra level of parallelism to a code. Vectorization introduces parallelism that is seperate from the parallelism gained from using multiple ranks with MPI or multiple threads with OpenMP. With each core of having at least one vector unit, each MPI rank or OpenMP thread used is able to incorporate vectorization as long as the number of ranks and threads is not overprescribed.

Vectorization for the most part is done through auto-vectorization behind the scenes at compile time. The compiler is usually smart enough to add vectorization that could help the performance of the program and not add vectorization when it could affect the correctness of the code. A programmer does need to be aware of the things not to do that could break automatic vectorization. There are occasions when auto-vectorization by the compiler does not happen because the compiler does not know that it would be safe to vectorize. This happens because the languages themselves do not expose constructs to convey this information to the compiler. In these cases, vectorization can be obtained through explicit compiler directives in the code.

Profiling

To determine if a program is performant one must do a detailed analysis of that program. We accomplished this by profiling the frequency and execution time of function calls and certain CPU instructions.

The first use of profiling should be to determine in which functions the program is spending the most time. This is important because these functions are where one should to identify primary issues in performance. Basic profiling can be used for this task; we used GPROF (a compiler provided instrumentation based profiler) to generate call-trees which visually demonstrate the time spent in specific functions. Instrumentation based profilers insert instructions that wrap calls within your program to collect execution data.

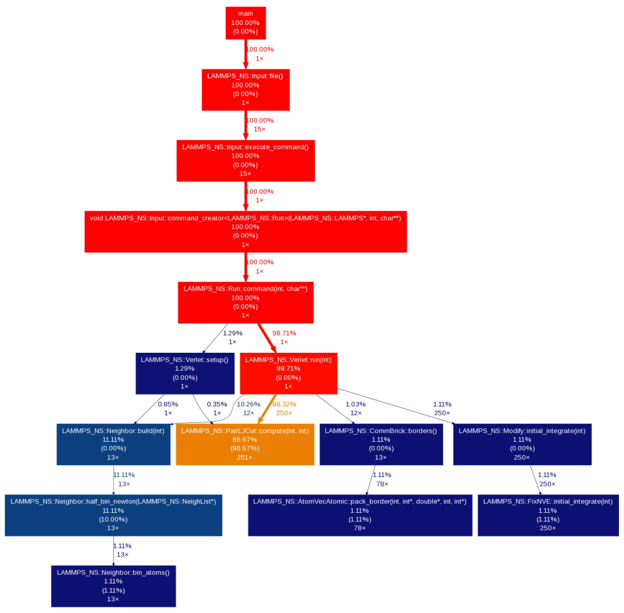

main
100.00%
(0.00%)
100.00%
1×
LAMMPS_NS::Input::file()
100.00%
(0.00%)
1×
100.00%
15×
LAMMPS_NS::Input::execute_command()
100.00%
(0.00%)
15×
100.00%
1×
void LAMMPS_NS::Input::command_creator<LAMMPS_NS::Run>(LAMMPS_NS::LAMMPS*, int, char**)
100.00%
(0.00%)
1×
100.00%
1×
LAMMPS_NS::Run::command(int, char**)
100.00%
(0.00%)
1×
1.29%
1×
98.71%
1×
LAMMPS_NS::Verlet::setup()
1.29%
(0.00%)
1×
LAMMPS_NS::Verlet::run(int)
98.71%
(0.00%)
1×
0.85%
1×
0.35%
1×
10.26%
12×
86.32%
250×
1.03%
12×
1.11%
250×
LAMMPS_NS::Neighbor::build(int)
11.11%
(0.00%)
13×
LAMMPS_NS::PairLJCut::compute(int, int)
86.67%
(86.67%)
251×
LAMMPS_NS::CommBrick::borders()
1.11%
(0.00%)
13×
LAMMPS_NS::Modify::initial_integrate(int)
1.11%
(0.00%)
250×
11.11%
13×
1.11%
78×
1.11%
250×
LAMMPS_NS::Neighbor::half_bin_newton(LAMMPS_NS::NeighList*)
11.11%
(10.00%)
13×
LAMMPS_NS::AtomVecAtomic::pack_border(int, int*, double*, int, int*)
1.11%
(1.11%)
78×
LAMMPS_NS::FixNVE::initial_integrate(int)
1.11%
(1.11%)
250×
1.11%
13×
LAMMPS_NS::Neighbor::bin_atoms()
1.11%
(1.11%)
13×

Figure 6: A sample call tree. This tree shows how functions call each other and how much of the execution time is spent in each of these functions.

To gain further information such as amount of time spent in MPI overhead, OpenMP overhead, memory usage, and vector operation more advanced profiling tools must be used. For this we used a sample based profiling tool called Allinea Map. Sample based profilers probe the program in regular intervals using operating system interrupts to gather their timing data. Map is a powerful tool capable of displaying all of the above data in an an easy to digest graphical form. It can also give a line by line analysis of the execution time and mode.

We used these profilers to identify where it is most efficient to focus our time on achieving an increase in performance. If a lack of vectorization utilization is determined then information about why vectorization is not occuring must be gathered elsewhere.

Optimization Reports

Optimization reports allow us to obtain information about vectorization at compile time. Optimization reports are essential to understing and maximizing the amount of vectorization in a code. These optimization reports give information on all of the loops inside of a program and most importantly include reasons why the compiler could not vectorize a loop, along with the potential speedup vectorization could provide. These optimization reports are essential for identifying and diagnosing vectorization issues.

CPIC

Curvilinear Particle in Cell (CPIC) is a plasma simulation code that is used to determine charge buildup on satellites. Particle in Cell plasma codes differ from Molecular Dynamic codes in that instead of explicitly calculating the forces between all (relevant) particles and updating positions based on them, the particle positions are used to update the field on a mesh. The particles have an 'exact' position on the mesh and the field at that point is used to integrate forward in time. CPIC is written in FORTRAN with MPI Everywhere for parallelism. To parallelize the program, the mesh is spatially decomposed across the MPI processes. If a particle moves from the mesh region controled by one MPI process to another it must be communicated to the new process(core).

CPIC is intended to scale to large problems, therefore optimization should be done to ensure efficient computing resource usage. While profiling CPIC considerable load-imbalance was detected. The cause of this load-imbalance comes from the nature of of the physics of charge buildup on satellites. If you imagine electrons and ions being sourced from the sun coming incident to the satellite, the particles would (almost) all be incoming from one direction. In simulation this is done by having a source of particles on the faces of the mesh corresponding to the direction from which the particles would be sourced. As we are interested in capturing high flux solar events, the system is no where near a steady state at the beginning of the simulation. This fact combined with the small integrtation timesteps required for accurate propogation results in a disproportionate particle distribution at one end of the system. To make the problem somewhat worse, the electrons and ions have the same energy, and since the ions are heavier, they propogate though the system much slower than the electrons. This means even once the electrons are propogated thoughout the system, there remains an imbalance of ions in the mesh for a much larger portion of time.

In an attempt to reduce the load imbalance in CPIC we use OpenMP to implement a work stealing algorithm. Work stealing occurs when one MPI process offloads work onto its 'neighboring cores'. OpenMP thread pools can not span multiple nodes and for performance reasons, neighboring cores describes cores on the same socket (same multi-core processor).

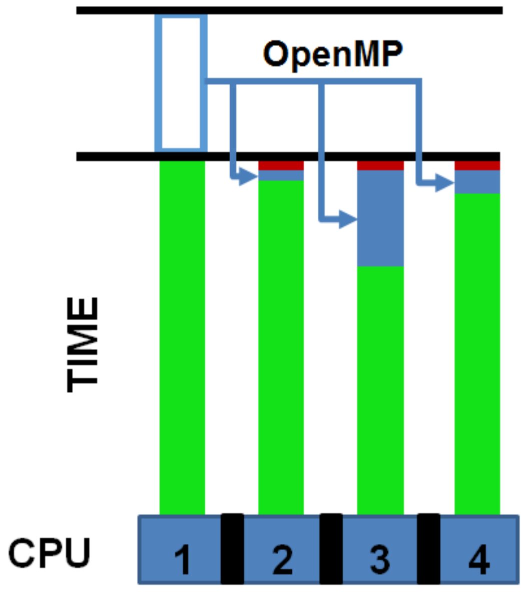


Figure 7: A graphic representing work stealing. Green is the time that a processor is computing, red is the time the process is idle. The hollow blue box is the work from the slow process that gets split onto the other processes (represented by the solid blue boxes).

The implementation of work stealing was done in the portion of the code that updates the positions of particles and was performed dynamically and intelligently (problem dependent). Ranks that had more parciles on their mesh region (these are the processes that would run slower) get more OpenMP threads. The number of threads that each process gets is determined dynaimically at runtime based on the number of particles that process has. To decide the number of particles that determine how many threads the process gets, we must use physical insights into the physics of the problem. For the implentation we have used this defined by the amount of injected particles, time step size, and ratio of mass between the electrons and ions.

The implementation of of work stealing has shown promise. It is important to recognize that work stealing is only able to achieve performance gains if the expensive processes do not exist on the same socket. For this reason the work stealing implementation only achieves performance gains in instances with proper spatial decomposion across sockets. With this in mind we show the results of a case in which proper decomposition has been achieved.

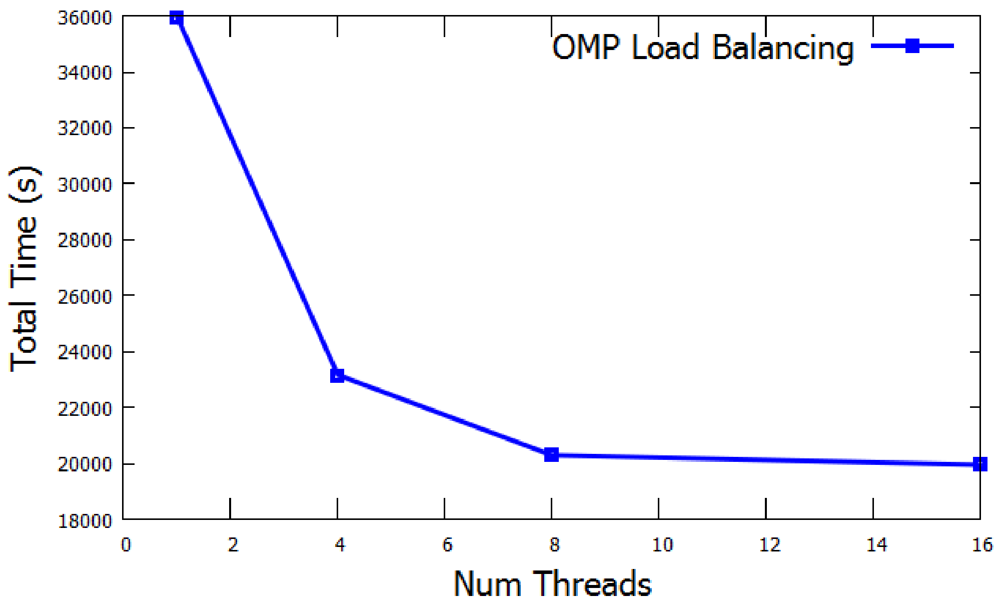


Figure 8: A graph showing the impact of the number of threads spawned (for expensive processes) on the time spent in load-imbalanced regions.

We can see that the load-imbalance has been improved largely by the work stealing implementation. The predominate issue with this implemenation of work stealing in CPIC is that it requires a knowledge of the physics of the problem at hand. This means the user must have more input into the problem. Another problem is that the user (or programmer) must be aware of the architecture of the machies they are using to ensure proper decomposition. Unfortunately, the way the decomposition happens is also subject to change in the version of MPI that is used, and therefore intelligent process mapping is required.

There is also room for improvement in load-balancing within CPIC that was beyond the scope of this project. These improvements would require a larger rework of the logic of the program. It is also possible to widen the range of situations in which the the program yields improvements from work stealing, but again this requires further changes to the logic of CPIC.

MPAS-O

MPAS-O, the Model for Prediction Across Scales - Ocean, is an ocean modelling code developed here at LANL that is intended for use in large scale simulations of the ocean for long scale climate and weather models. We are interested in looking at MPAS-O to get an idea of how it will perform on the KNL and to discover some performance gains using profiling and optimization reports. It was chosen to be studied because it is a highly used IC used on the supercomputers at LANL.

MPAS-O uses a Voronoi mesh to discretize the computational domain which in most cases is the entire ocean of the world. Finite volume methods are then used on this mesh to solve the differential equations of interest (refer to [5] for more details on the implementation). MPAS-O is implemented in Fortran and uses MPI to exploit parallel computing. In order to utilize MPI, the Voronoi mesh created is partitioned between the number of MPI ranks specified by the user. Computation can then be done simultaneously on each partition of the mesh. There are some known drawbacks to using MPI with MPAS-O due to load-balancing issues that come from the Voronoi mesh being unstructured [7]. These load balancing issues will be evident in the performance analysis section.

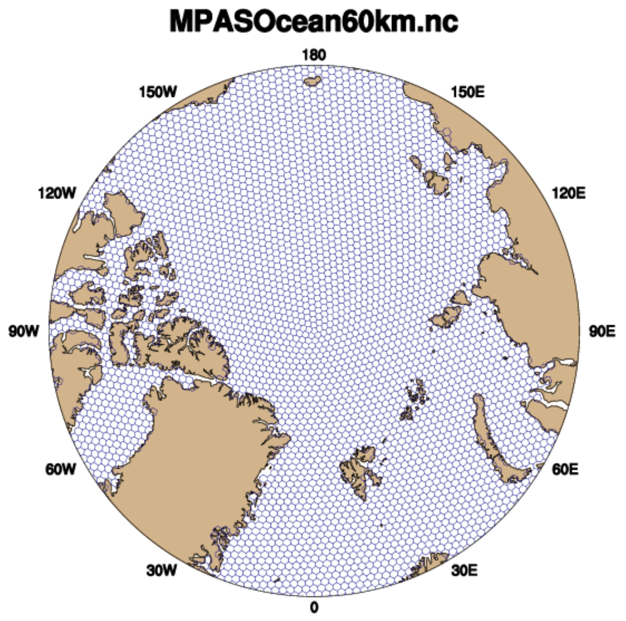


Figure 9: A visualization of a Voronoi Mesh over the Ocean
[2]

Building for the Xeon Phi

A feature of the Xeon Phi that is very convenient is that code written for use with CPUs can be used to run on a Xeon Phi. This gives the Xeon Phi a distinct advantage over other *accelerated* computing hardware such as GPUs that would require a significant time commitment to write a code the could efficiently use them. One would not expect original CPU code to work especially well with the Xeon Phi immediately, however if the code is fine-tuned for the Xeon Phi it has the potential to perform well.

While the Xeon Phi provides the convenience of using existing code written for the CPU, it is not necessarily always easy to build existing software libraries for the KNC. The problem we ran into building MPAS-O for the KNC was the large number of dependencies that are required. These problems stem from the KNC being a co-processor which makes cross-compiling for them much more challenging. Fortunately the KNL will support native compilation. [Instructions for building MPAS-O on KNC have been developed and are available at http://ic-wiki.lanl.gov/ic-help:ic-knights:mpas-o:build_mic.]

Performance Analysis

When considering the performance of MPAS-O, we were interested in seeing how an implementation on the KNC compares to that same implementation on a node containing current

CPU processors (refer to Table 1 for specifications of the hardware being used). Depending on how the results of this comparison, we can then recommend the type of hardware for running simulations with MPAS-O.

Figure 10 shows scaling results for both a CPU and KNC implmentation running the largest test problem possible (due to KNC memory constraints),an entire ocean problem with 240 km spacing between data points. The scaling plots for the CPU implementation show that it does scale up to 32 MPI ranks meaning that it performs best when using all 32 cores of the node which is what we would hope to see. The scaling plots for the KNC implementation show that the fastest implementation is between 30 and 60 MPI ranks, so performance degres as we approach the maximum amount of MPI ranks able to be used effectively with a KNC which is 240. For this problem, the fastest CPU implementation took around 200 seconds while the fastest KNC implementation took around 2800 seconds. In its current state, we recommend that MPAS-O be ran with CPUs rather than a Xeon Phi.

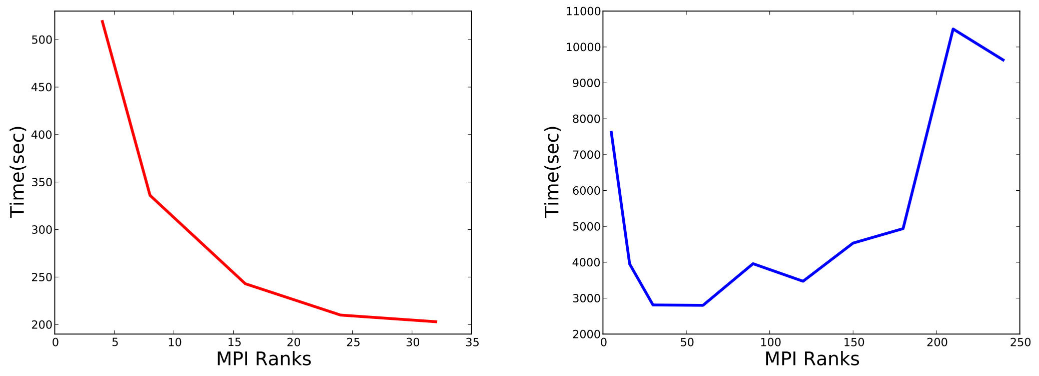


Figure 10: Scaling plots for the world ocean test problem with 240 km spacing. CPU timings are on the left and KNC timings are on the right.

Performance Issues

The performance issues with MPAS-O on the KNC stem from a combination of two different sources: overdecomposition and load-imbalance. Overdecomposition means that the amount of computatiion in the problem was not large enough to be split across all of the cores, and therefore we cannot effectively use all of the computing offered by the KNC. Overdecomposition becomes a particularly big factor in our tests since there is a limited amount of memory available on the KNC. This ultimately limits the performance capability on the KNC. Load-imbalance is another issue when using MPAS-O. Load-imbalance has a significant impact on MPAS-O because the unstrutuctured mesh is very difficult to partition, resulting in an uneven distribution of work across MPI ranks [7]. Figure 11 illustrates the problem that arises from load-imbalance. This plot shows for the CPU implementation how much time is spent in MPI communication and wait time. As the number of MPI ranks increases, the percentage of time being spent in MPI communication becomes dominant. This is particularly detrimental on the KNC which needs more MPI ranks than the CPU in order to utilize all of its available computing power.

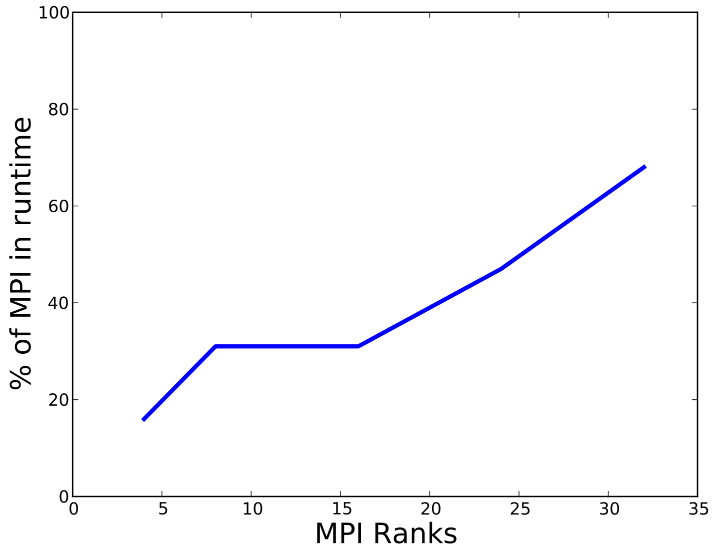


Figure 11: Total percentage of time spent in MPI for the world ocean test problem with 240 km spacing on a CPU node.

Performance Improvements

In addition to comparing the performance of MPAS-O on the KNC versus on a node with traditional CPUs, we explored the potential for performance improvement within the code. Through profiling the CPU implementation, we are able to identify regions where vectorization was not utilized. Through close examination of the optimization reports, we were able to identify computationally expensive loops that were not being vectorized and yet had potential for safe vectorization within minimal refactoring. Through the addition of explicit compiler directives to the code, we could instruct the compiler to ignore (false) potential interdependencies. This resulted in improved vectorization and we saw the performance improvements for the CPU implementation in Figure 12.

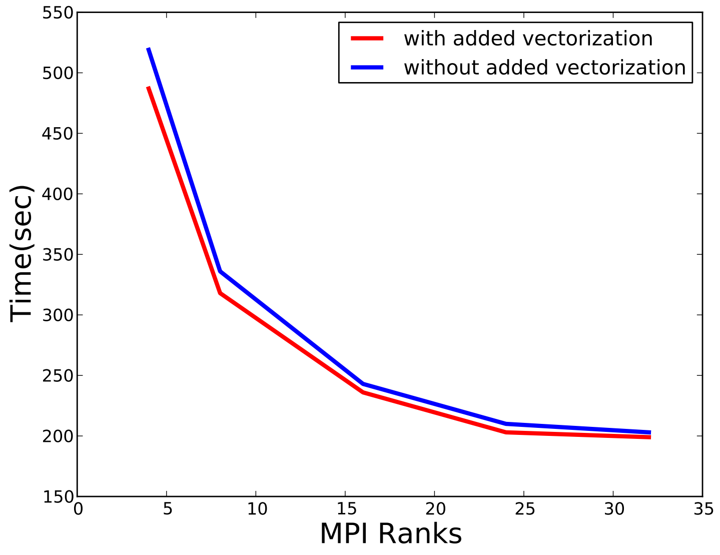


Figure 12: Performance gained from improving vectorization.

Conclusions

CPIC

The implementation of OpenMP work stealing was successful at eliminating much of the load-imbalance within CPIC. The issue with this implemention is that it requires both knowledge of the physics of the system you are studying (which is not always easy to provide) and a knowledge of the architecture of the computing system being used; the problem itself must be carefully decomposed in order to utilize the work stealing implementation.

Future work could focus on reducing these issues. By appropriately determining the division of particles across MPI ranks, and the careful distribution of those ranks across the physical hardware, appropriate cut-offs for work stealing could be dynamically determined. It is theoretically possible to dynamically setup the decomposition such that more cases would lend themselves to benefit from the work stealing implementation. It is also possible to apply the work stealing implementation to other regions within the code, beyond the particle mover.

Work stealing is method that is applicable to almost all MPI programs that suffer from significant load-imbalance. The dynamic and intellegent implementation used here was more efficient than a standard ’all in’ attempt in which every process gets threads. In order to implement this work stealing approach into another problem, all that needs to be developed is the appropriate way to determine which processes are most expensive and the appropriate implementation of the work stealing algorithm.

MPAS-O

Although scaling on the KNC with MPAS-O was not successful, all hope is not lost for using Xeon Phis with MPAS-O. With the KNL, the larger memory footprint means larger problems can run providing more opportunity to scale up to the full capacity of hardware and could make it more competitive with CPU implementations.

Improvements were found for MPAS-O in our test problem by improving vectorization within the code. However, these speedups were relatively small and would most likely not be seen in large problems due to MPAS-O being communication bound. In order to obtain significant performance improvements within MPAS-O, it would be necessary to improve the partitioning of the work in order to provide better load-balancing. This could possibly be done through a hybrid MPI/OpenMP code.

References

[1] R.H. Dennard, V.L. Rideout, E. Bassous, and A.R. LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *Solid-State Circuits, IEEE Journal of*, 9(5):256–268, Oct 1974.

[2] https://www.ncl.ucar.edu/Applications/mpas.shtml.

[3] John D. McCalpin. Memory bandwidth and machine balance in current high performance computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pages 19–25, December 1995.

[4] G.E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, Jan 1998.

[5] Todd Ringler, Mark Petersen, Robert L Higdon, Doug Jacobsen, Philip W Jones, and Mathew Maltrud. A multi-resolution approach to global ocean modeling. *Ocean Modelling*, 69:211–232, 2013.

[6] D Rudiger. Moores_law.pdf, August 2015.

[7] Abhinav Sarje, Sukhyun Song, Douglas Jacobsen, Kevin Huck, Jeffrey Hollingsworth, Allen Malony, Samuel Williams, and Leonid Oliker. Parallel performance optimizations on unstructured mesh-based simulations. *Procedia Computer Science*, 51:2016–2025, 2015.

[8] Dr. Eric Stotzer, August 2015.

Material interface evolution and semi-analytic radial piston test problems

Team Members
Bart Brown and Clark Pederson

Mentor
Nathaniel Morgan

Abstract

In this research we propose a novel test problem for the verification of material interface evolution in a vortex coupled to hydrodynamics and a new method for setting up a Sedov verification test. In the first section we discuss the new vortex test problem. Vortical flow coupled to hydrodynamics is important in a range of applications including inertial confinement fusion where the mix volume is a function of interfacial area. It is difficult to quantify the accuracy of the method using the current test problems due to the lack of semi-analytic solutions on this class of a problem. For this reason, we consider particles moving in a 2D steady state velocity field to which a semi-analytic solution of the particle trajectories exists. We perform convergence studies using a constant of motion in the Taylor-Green velocity field. The domain in which we consider the problem is the square domain. Using the FLAG hydrocode with cell-centered hydrodynamics (CCH) and corner gradient reconstruction (CGR) methods we numerically simulate a system on the same domain populated with gamma law gasses on a uniform quadrilateral mesh. Here we consider the case of two and three materials, where the initial interfaces are unmixed. The materials are arranged similar to a conventional triple point problem for the three material case. Convergence of the velocity, pressure and density fields is studied numerically. We discuss the two, and three material cases using a arbitrary Lagrange-Eulerian (ALE) method. Second order convergence was observed in the Lagrangian limit and first-order convergence with ALE. Finally we discuss a new technique, mesh intersection based ALE (xALE), where we observe third-order convergence in the underlying fields for one material.

In the second section we discuss the modified Sedov bast problem. The Sedov blast wave is of great utility as a verification problem for codes using Lagrangian hydrodynamics. Nevertheless, the typical implementation represents the energy point source as a energized zone of finite volume This approximation can be avoided by directly finding the effects of the energy source as a boundary condition. This proposed method transforms the Sedov problem into a radial piston problem with a time-varying velocity. A portion of the mesh adjacent to the energy source is removed and the boundaries of this hole are forced

with the exact velocities from the Sedov solution. This verification test is implemented on two types of meshes and convergence is shown. The results from the typical initial condition method and the new boundary condition method are compared.

Verification of Material Interface Evolution in a Vortex with Hydrodynamics

Introduction

The simulation of material evolution in a vortex fully coupled to hydrodynamics is important in many applications such as inertial confinement fusion (ICF) where the mix volume is a function of the interfacial area. There has been some disagreement between experimentalists and theorists in regards to the plastic-fuel interface evolution. The verification of vortical motion in hydrocodes could shed light on this issue. We propose a novel new test problem that builds on a 2D steady state velocity field. Our goal is to quantify the accuracy of material-material interface evolution in a vortex coupled to hydrodynamics against a semi-analytic solution.

We discuss the method of manufactured solutions and use the technique to force the system into a steady state with the appropriate velocity field. There is some freedom in choosing the source terms and initial conditions. In this work, we make some simplifying restrictions to the state. We show that while the Taylor-Green field can be supported with an energy source, the Rider-Kothe field requires a momentum source, which is not compatible with the FLAG hydrocode at this time.

The semi-analytic solution uses a 4th-order Runge-Kutta integrator to solve the particle trajectories. We find a constant of motion for the Taylor-Green velocity field which we use to verify the semi-analytic solution. The period of the particle located at the triple point is calculated numerically as 2.56 microseconds. We primarily use arbitrary Lagrange-Eulerian (ALE) methods but also explore the field convergence using the Lagrange approach. The ALE method uses an Eulerian remap with swept face advection. We also perform single material calculations using a new mesh intersection based method (xALE). The new method couples naturally across cell corners and conserves total energy[7]. The hydrocode calculations are compared to the interface evolution extracted from the particle trajectories given by the semi-analytic solution. The ALE hydrocode calculations are graphically compared to the semi-analytic solutions for various mesh resolutions with two and three materials. We go on to perform convergence studies on the material-material interface.

Governing Equations

The 2D unsteady Euler equations govern the system and are written below in the flux-conservative form

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho v_i) = 0 \tag{1}$$

$$\frac{\partial}{\partial t}(\rho v_i) + \frac{\partial}{\partial x_j}(\rho v_i v_j + p\delta_{ij}) = 0 \tag{2}$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_i}[(v_i(\rho E + p))] = S_E \tag{3}$$

Where ρ is the density, v is the velocity, $E = v_i v_i/2 + e$ is the specific total energy, e is the specific internal energy and p the pressure. In this research we assume a constant density and a divergence-free velocity field. The system is closed by the gamma law gas equation of state

$$p = \rho e(\gamma - 1) \tag{4}$$

Where γ is the ratio of specific heats.

The method of manufactured solutions may be used to force the flow in the system to match a velocity field of interest; the Taylor-Green and the Rider-Kothe velocity fields

$$\vec{v}_{TG} = \sin(\pi x)\cos(\pi y)\hat{\imath} - \cos(\pi x)\sin(\pi y)\hat{\jmath} \tag{5}$$

$$\vec{v}_{RK} = -\sin^2(\pi x)\sin(2\pi y)\hat{\imath} + \sin^2(\pi y)\sin(2\pi x)\hat{\jmath} \tag{6}$$

The unknowns in the steady state system are the pressure p, the total energy E, the internal energy e, and the energy source term S_E. The mass Eqn.(1) is satisfied identically for a divergence free velocity field with constant density. The remaining equations together with the gamma law equation of state and the definition of total energy close the system, allowing unique pressure, internal energy, and source energy terms to be calculated.

Test Code

The numerical results for the vortex problem including the convergence studies were generated with the FLAG hydrocode [4][2] using the cell-centered hydrodynamics method (CCH)[3] and corner gradient reconstruction (CGR)[8] with Young's interface reconstruction on a uniform quadrilateral mesh. Most of our calculations are done using the arbitrary Lagrange-Eulerian (ALE) formulation in the Eulerian limit with swept face ALE advection[19]. In addition, a mesh intersection based ALE method is studied.
The global L_1 error in each field variable, F, was calculated as

$$||\varepsilon|| = \frac{\sum |F_{calc} - F_{theory}|}{N} \tag{7}$$

The L_1 error in the interface length was calculated as

$$||\varepsilon|| = \frac{dx \sum |F_{calc} - F_{theory}|}{F_{calc}} \tag{8}$$

The convergence rate was calculated as the slope in the loglog plot of $||\varepsilon||$ versus the zone size dx.

Forcing Steady State Vorticular Flow

We use the method of manufactured solutions to calculate the energy source term and initial fields which support the velocity field.

We follow similar steps as in [20]. Here, we demand a constant density, $\rho = 1$, and allow only for an energy source. In this research we consider only materials with a gamma law equation of state. The internal energy is then uniquely determined from the equation of state and the pressure. The total energy is determined by the internal energy and the velocity field.

The mass Eqn.(1) vanishes for a divergence free velocity field and the momentum Eqn.(2) reduces to

$$\frac{\partial}{\partial x_j}(\rho v_i v_j + p\delta_{ij}) = 0$$
$$\rho(\frac{\partial}{\partial x_j}v_i)v_j + \rho v_i(\frac{\partial}{\partial x_j}v_j) + \frac{\partial}{\partial x_i}p = 0.$$

Thus we can write the spatial derivative of the pressure as a function of the velocity.

$$\frac{\partial}{\partial x_i}p = -\rho v_j \frac{\partial}{\partial x_j}v_i \tag{9}$$

Therefore, if the system is valid under our assumption (no momentum sources), the pressure must satisfy the following relation in 2D

$$p = -\rho\frac{1}{2}v_1^2 + \rho\int v_2\frac{\partial}{\partial x_2}v_1 dx_1 + g(x_2) \tag{10}$$
$$= -\rho\frac{1}{2}v_2^2 + \rho\int v_1\frac{\partial}{\partial x_1}v_2 dx_2 + f(x_1) \tag{11}$$

where f and g are arbitrary functions of x_1 and x_2 respectively. It is easily shown that the relation is satisfied for the Taylor-Green velocity field, Eqn.(5).

$$p_{TG} = -\rho\frac{1}{2}\sin^2(\pi x_1) + g(x_2) \tag{12}$$
$$= -\rho\frac{1}{2}\sin^2(\pi x_2) + f(x_1) \tag{13}$$

Taking the arbitrary functions to be zero, $f = g = 0$, p_{TG} reduces Eqn.(11) to

$$p_{TG} = -\rho\frac{1}{2}(\sin^2(\pi x_1) + \sin^2(\pi x_2)) \tag{14}$$

However, the Rider-Kothe velocity field, Eqn.(6), is not compatible with Eqn.(11)

$$p_1 = -\rho\sin^4(\pi x_1)\sin^2(\pi x_2)(2\cos^2(\pi x_2)$$
$$-\cos(2\pi x_2)) + g(x_2)$$
$$p_2 = -\rho\sin^4(\pi x_2)\sin^2(\pi x_1)(2\cos^2(\pi x_1)$$
$$-\cos(2\pi x_1)) + f(x_1)$$

We see that p_1 cannot equal p_2 for any choice of $f(x_1)$ or $g(x_2)$. Thus, a momentum source term is necessary to support a steady state system with this velocity.

The internal energy for the system with the Taylor-Green velocity field is determined by the gamma law equation of state, Eqn.(4), and the pressure p_{TG}, Eqn.(14).

$$e_{TG} = \frac{1}{4(\gamma-1)}(\cos(2\pi x) + \cos(2\pi y)) \tag{15}$$

For $\gamma = \frac{5}{3}$

$$e_{TG} = \frac{3}{8}(\cos(2\pi x) + \cos(2\pi y)). \tag{16}$$

Under the conditions of a steady state divergence-free velocity field with a gamma law gas, the energy equation can be written as

$$\begin{aligned}
\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_i}[v_i(\rho E + p)] &= S_E \\
\frac{\partial}{\partial x_i}[v_i(\rho E + p)] &= S_E \\
v_i \rho \frac{\partial}{\partial x_i} E + v_i \frac{\partial}{\partial x_i} p &= S_E
\end{aligned}$$

The spatial derivative of the pressure, Eqn.(9), is substituted in

$$v_i(\rho(\frac{\partial}{\partial x_i}E) - \rho v_j \frac{\partial}{\partial x_j} v_i) = S_E \tag{17}$$

and, using the definition of total specific energy

$$\begin{aligned}
\frac{\partial}{\partial x_i}E &= v_j \frac{\partial}{\partial x_i} v_j + \frac{1}{\rho(\gamma-1)} \frac{\partial}{\partial x_i} p \\
&= v_j \frac{\partial}{\partial x_i} v_j - v_j \frac{1}{(\gamma-1)} \frac{\partial}{\partial x_j} v_i.
\end{aligned}$$

$$E = \sum_i \frac{1}{2} v_i^2 + e = \sum_i \frac{1}{2} v_i^2 + \frac{p}{\rho(\gamma-1)} \tag{18}$$

Employing Eqn.(9) again

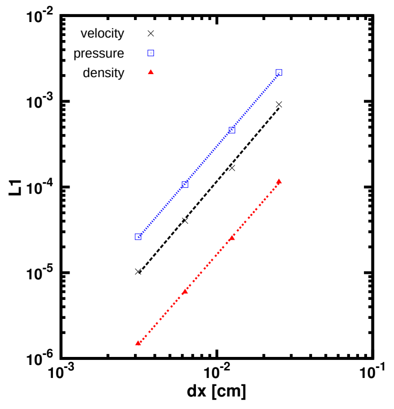
$$\begin{aligned}
S_E &= v_i(\rho v_j \frac{\partial}{\partial x_i} v_j - v_j \frac{\rho}{(\gamma-1)} \frac{\partial}{\partial x_j} v_i - \rho v_j \frac{\partial}{\partial x_j} v_i) \\
S_E &= -v_i v_j \frac{\rho}{(\gamma-1)} \frac{\partial}{\partial x_j} v_i
\end{aligned}$$

Therefore, the required source term is

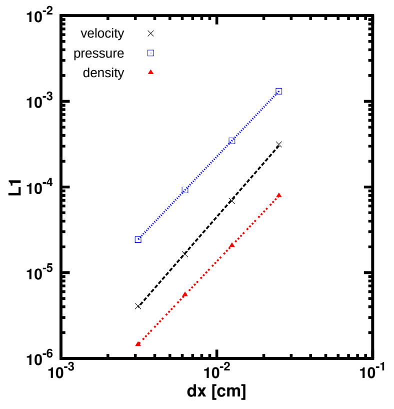
$$S_E = \frac{1}{4} \frac{\rho \pi}{(\gamma-1)} [\cos(3\pi x)\cos(\pi y) - \cos(\pi x)\cos(3\pi y)] \tag{19}$$

The above solution was evolved in the hydrocode using $\gamma = 5/3$, and $\rho = 1$ with four increasing mesh resolutions (40x40, 80x80, 160x160, 320x320) for approximately seven periods of the vortex.

The pure Lagrangian calculations show second order convergence in the velocity, pressure, and density fields with two and three materials. However, the simulation could not evolve the interface past one microsecond in the highest mesh resolution (320x320) due to mesh tangling. Therefore, all of the interface length studies were done using ALE in the Eulerian limit where we observe first order convergence in the fields for one, two and three materials. Usually one would expect second order convergence when using CCH-CGR. The lower order convergence could be caused by the swept face advection necessary in the Eulerian remap step.



(a)



(b)

Figure 1: Second-order convergence consistent with CCH-CGR is observed in the fields for the pure Lagrangian method. All data points were measured at only one time (t=0.3 microseconds). For the two material calculations (a) the velocity converges with power 2.15 ($r^2 = 0.997$), the pressure converges with 2.12 ($r^2 = 0.999$), and the density with 2.09 ($r^2 = 0.999$). Likewise, in the three material calculations (b) the velocity converges with power 2.09 ($r^2 = 0.999$), the pressure converges with 1.92 ($r^2 = 0.999$), and the density with 1.92 ($r^2 = 0.999$).

Semi-Analytic Solution

Particle trajectories in the Taylor Green velocity field cannot be solved analytically, therefore, a semi-analytic solution is necessary for the verification of the numerical results of the interface. We use a 4th-order Runge-Kutta method to solve for a particle's motion in the steady state velocity field. Convergence studies were performed on the solution using the constant of the motion $sin(\pi x_1)sin(\pi x_2)$.

$$\begin{aligned}
&\frac{d}{dt}sin(\pi x_1)sin(\pi x_2) = \\
&\pi cos(\pi x_1)sin(\pi x_2)v_1 + \pi sin(\pi x_1)cos(\pi x_2)v_2 = \\
&\pi cos(\pi x_1)sin(\pi x_2)sin(\pi x_1)cos(\pi x_2) - \\
&\pi sin(\pi x_1)cos(\pi x_2)cos(\pi x_1)sin(\pi x_2) = 0
\end{aligned}$$

Thus, $sin(\pi x_1)sin(\pi x_2)$ is a constant of motion and one can immediately see that particle trajectories must be closed orbits. We study the convergence of the semi-analytic solution by the absolute error in the constant, C, shown in Eqn. (20).

$$sin(\pi x_1)sin(\pi x_2) = C \tag{20}$$

Using Eqn.(7) we show that the semi-analytic solution converges to the actual closed orbits described by Eqn.(20). The trajectory of a single particle initially placed at the triple point in

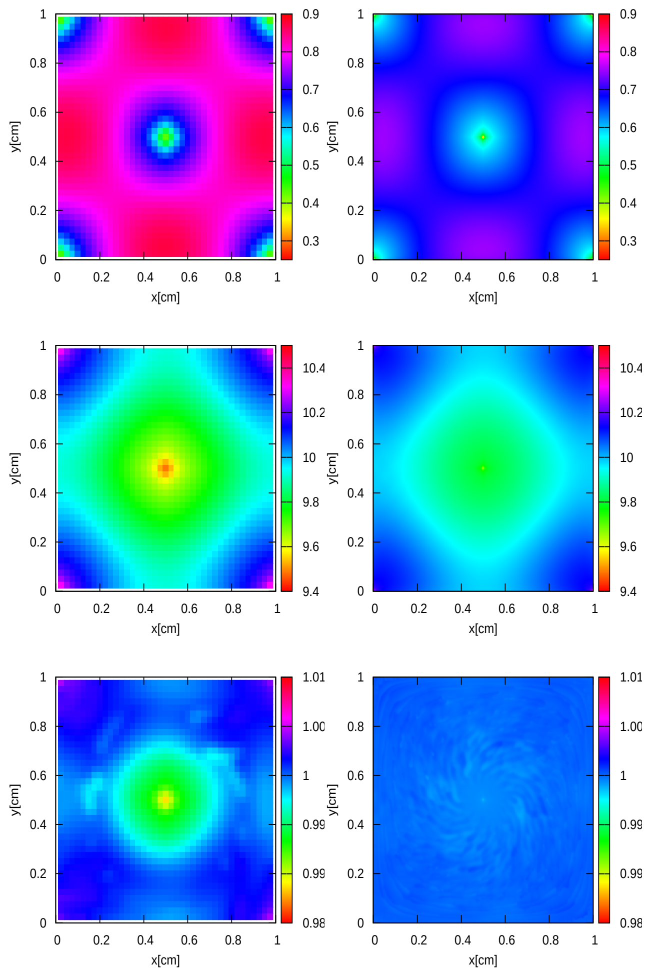


Figure 2: Selected images from the two material ALE simulation show significant distortion after seven periods in the 40x40 mesh (first column) compared to the 320x320 mesh (second column). The velocity (first row) has visibly grown in magnitude. The center of the pressure field (second row) has been expanded into a diamond like shape, indicating mesh imprinting. The density (third row) has dropped in the center of the domain however, visual variations are expected due to the precision of the color pallet.

the three material calculation is studied using the semi-analytic solution for varying time step, dt.

The orbit of the particle converges at fourth order in dt, shown in Fig 5. The period of the particle located at the triple point for the three material case $(1/2, 4/5)$ was measured numerically to be approximately 2.56 microseconds. Many of our results are based on this unit of time and we will refer to this as the period of the vortex from now on.

In order to verify the interface length of the two and three material calculations we use the semi-analytic solution to measure the trajectories of particles placed at the interface of materials. The semi-analytic solution is solved on the unit square centered at $(1/2, 1/2)$. In the two material case particles are distributed along the interface $y = 4/5$ and also along the line

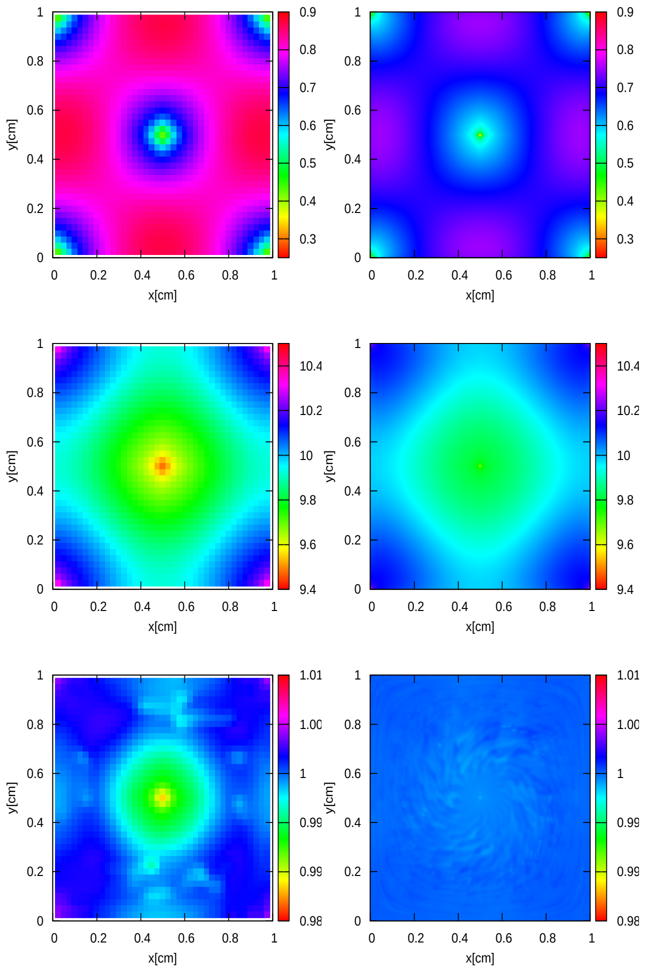


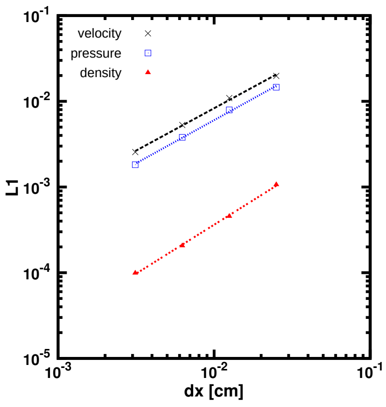
Figure 3: Similar observations may be made as in Fig 2. The velocity (first row) of the 40x40 mesh (first column) has been increased in magnitude compared to the 320x320 mesh (second column). The pressure field (second row) has been diffused and artificially squared. Only small variations are present in the density field (third row).

which bisects the upper division of the domain for the three material case. All convergence studies use the semi-analytic solution with dt chosen to be 0.00003.

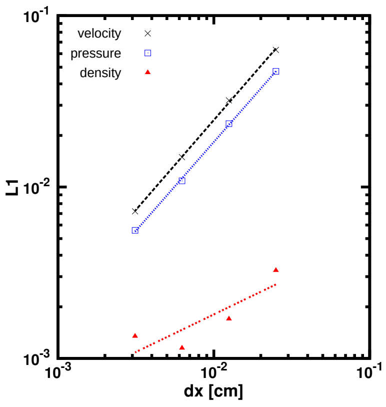
The particles are spatially evolved through time according to the Taylor-Green velocity field using a 4th order Runge-Kutta method. The interface length as a function of time is then recovered from the solution by taking the sum of the distances between successive particles. The distance is calculated by the L_2 norm of the vector from particle i to particle $i+1$

$$\text{Interface Length(t)} = \sum_{i=1}^{N} ||x(\vec{t})_{i+1} - x(\vec{t})_i||_2 \tag{21}$$

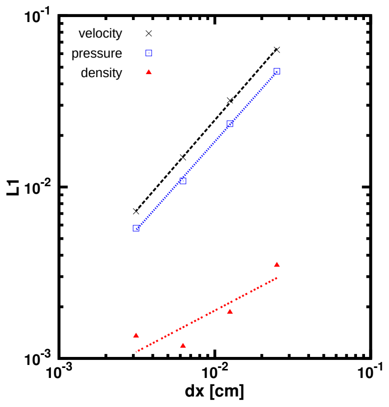
where $x(\vec{t})_i$ is the location of the i_{th} particle.



(a)



(b)



(c)

Figure 4: The fields converge slower with the swept face ALE approach than with a pure Lagrangian method. The one material calculations (a) reveal first-order convergence in the fields. The velocity converges with power 0.99 ($r^2 = 0.998$), the pressure with power 1.01 ($r^2 = 0.998$) and the density with 1.14 ($r^2 = 0.999$). The two and three material calculations, (a) and (b), agree with first-order convergence in the fields. For the two material calculations (a), the velocity converges with power 1.05 ($r^2 = 0.999$), the pressure with power 1.02 ($r^2 = 0.999$) and the density with 0.44 ($r^2 = 0.732$). For the Three material calculations (b), the velocity converges with power 1.05 ($r^2 = 0.999$), the pressure with power 1.02 ($r^2 = 0.999$) and the density with power 0.48 ($r^2 = 0.779$).

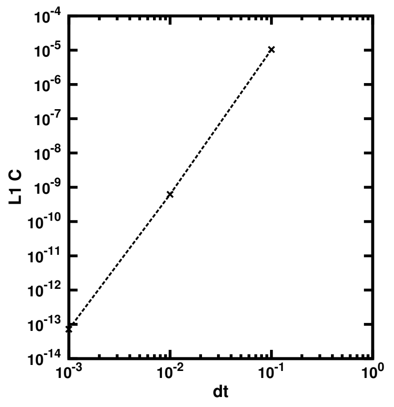


Figure 5: The constant of motion $sin(\pi x_1)sin(\pi x_2) = C$ in the semi-analytic solution converges semi-to the analytic constant of $C = sin(4\pi/5)$ with a dt of order 10^{-13}.

Interfacial Length

The material-material interface is measured as it evolves through time and compared to the semi-analytic solution. The mesh dimensions used in the two and three material hydrocode calculations are 40x40, 80x80, 160x160 and 320x320. Figures 7 and 8 show the material evolution for periods 1,3,5, and 7 of the vortex for two and three materials respectively. The

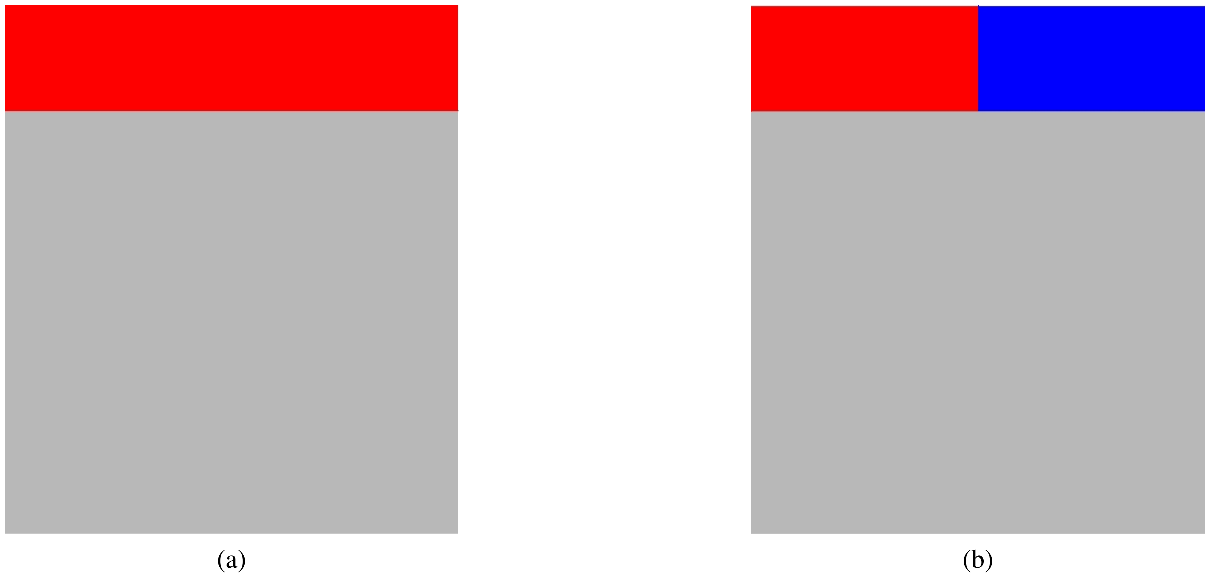


Figure 6: The initial material interface is located at $y = 4/5$ in the two material calculations (a) and also at $x = 1/2$ for $4/5 \leq y \leq 1$ in the three material calculations (b). All materials are ideal gasses and in the three material calculations we refer to the top left material as material one and the top right material as material two.

calculations with the lower mesh resolutions visibly deteriorate after just a few periods unlike the 160x160 and 320x320 meshes.

Large deviations from the semi-analytic solutions are seen in the higher mesh resolutions (Fig.9) even though the domain compares favorably to the semi-analytic solution as shown in Figs.7 and 8. This is due to the break down of the materials near the boundaries, though difficult to see. As the system is evolved through time the materials continue to wind about each other under the influence of the vortex, causing the filaments to grow ever thinner especially near the boundaries. This is responsible for the eventual break up of the material as the thickness of the filament becomes comparable to the zone size, and can no longer be modeled accurately in the flow. The effect is most pronounced near the boundaries of the domain where the filament becomes thinnest. The hydrocode is unable to model the interface as the material is dragged close to the boundaries resulting in a significant error in the interface length. This effect explains the large error seen in Fig.9 for the 320x320 mesh resolution. In order to accurately model the filaments one would need a mesh with ever increasing resolution near the boundaries as the problem evolves through time. This deterioration is not a fault of the test problem but rather, it illuminates a break down in the hydrocode as the problem becomes more demanding.

We observe between first and second-order convergence in the interface length for the two material calculations (Fig.10). It is not clear why the interface length converges at a higher order than the velocity, pressure, and density fields. For the triple point (three material) calculation, material one (top left) converges at second order in interface length and material two converges at first-order (Fig.10). The interface length of material two converges significantly

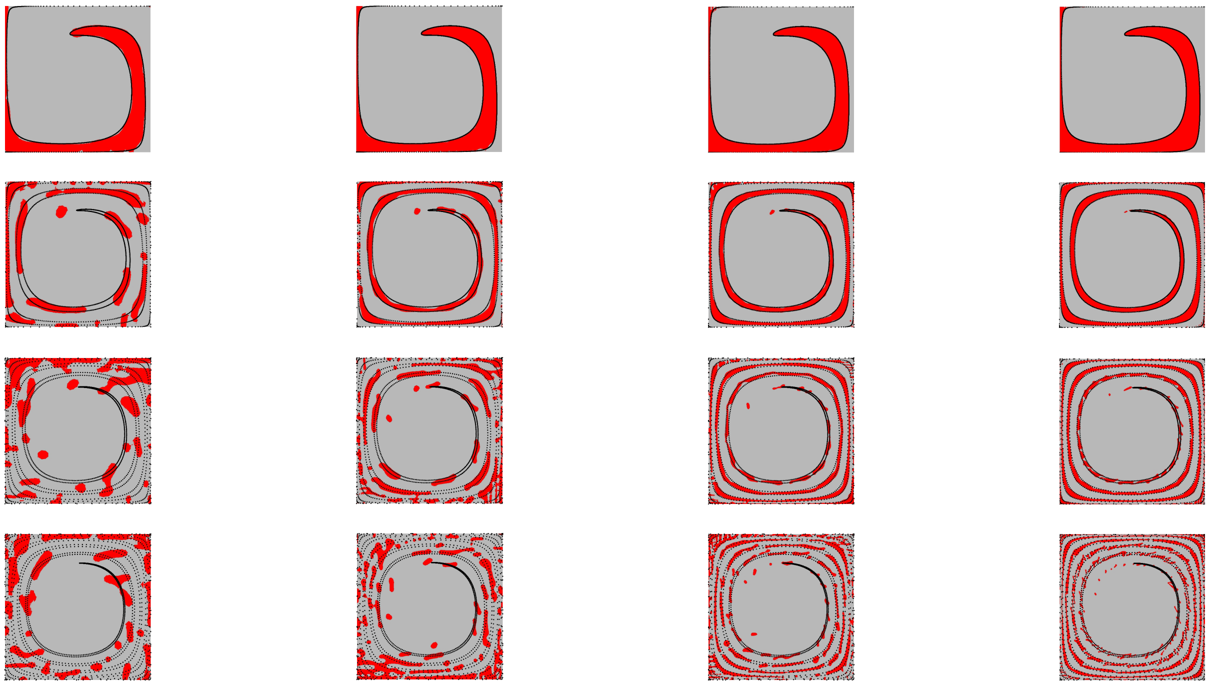


Figure 7: The two material calculations are evolved through time on different mesh sizes 40x40, 80x80, 160x160, and 320x320 matching the columns from left to right. The rows are the odd periods 1,2,3,5,7 of the vortical motion starting from the top at the first period. The semi-analytic solution is overlaid on each time step as black dots.

slower than the interface length of material one. This is caused by a detail of the hydrocode which gives preference to the first material, transferring errors in interface position to the second and third material.

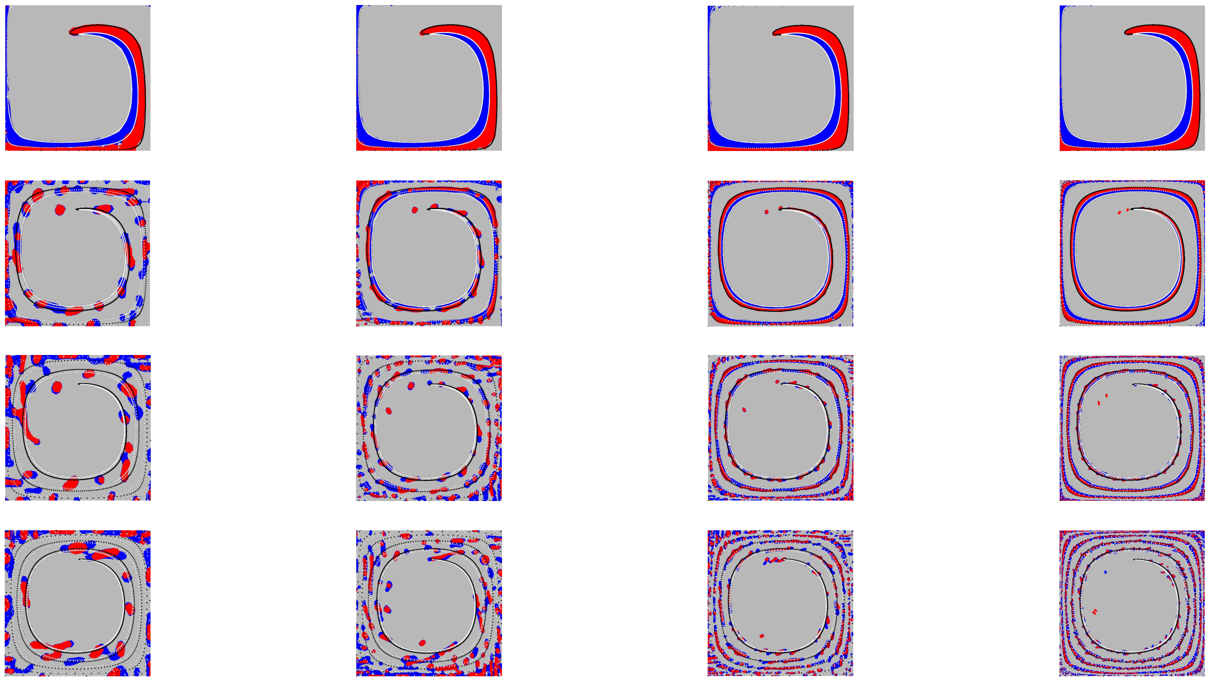
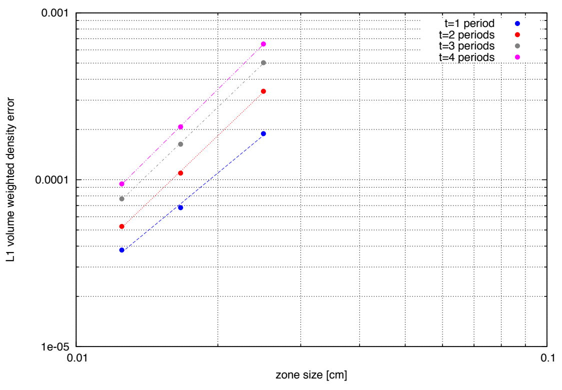


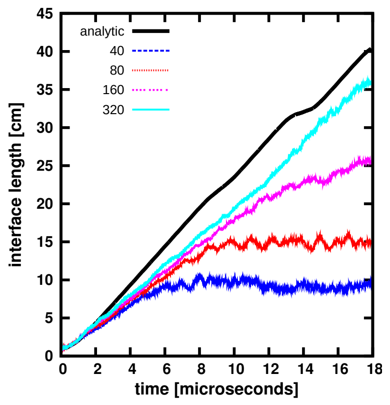
Figure 8: Likewise, the three material calculations are evolved through time in the mesh sizes 40x40, 80x80, 160x160, and 320x320 matching the columns from left to right. The rows are again the odd periods 1,2,3,5,7 of the vortical motion starting from the top at the first period. The semi-analytic solution is overlaid on each time step as black dots for the interface of material one and white dots for material two.

xALE

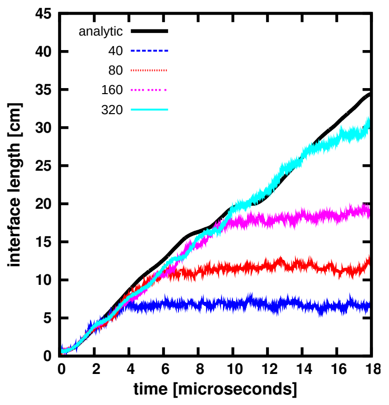


(a)

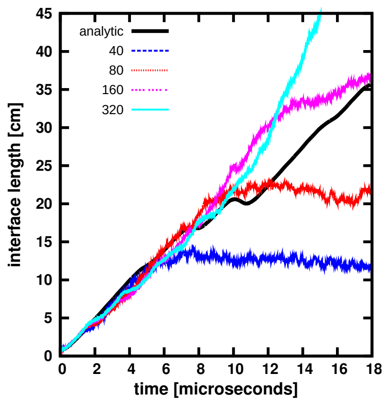
Figure 11: xALE field convergence plots: the density field



(a)

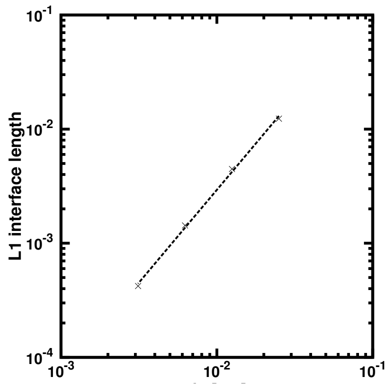


(b)

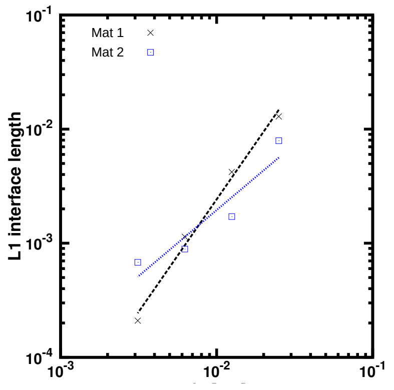


(c)

Figure 9: The material interface is plot versus seven periods of the vortex in the four mesh sizes and compared to the semi-analytic solution. The two material interface (a) converges to the semi-analytic solution as mesh resolution increases and the interface lengths of different resolutions do not cross each other. The interface length of material one in the three material calculations (b) also converges to the semi-analytic solution and the interface lengths of different mesh sizes do not cross. However, the associated interface lengths of material two in the three material calculations (c) does cross over the analytic solution. The highest resolution calculation even seems to diverge from the semi-analytic solution near 12 microseconds.

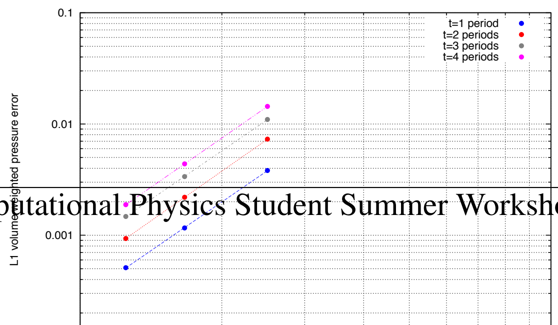


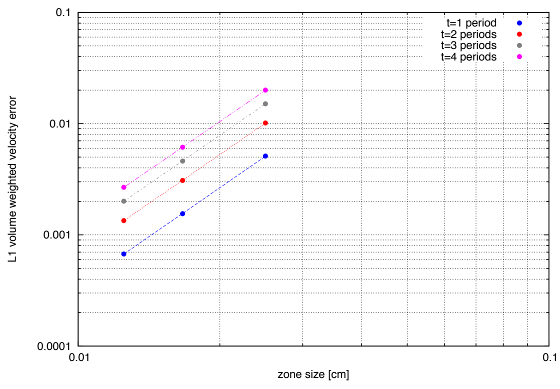
(a)



(b)

Figure 10: The interface length converges at near second order using the ALE remap method. The two material case (a) converges with respect to zone size (dx) with power 1.63 ($r^2 = 0.998$). The interface length associated with material one (top left material) in the three material calculations (b) converges with power 1.97 ($r^2 = 0.991$) and material two (top right material) with power 1.16 ($r^2 = 0.890$)





(a)

Figure 13: xALE field convergence plots: the velocity field)

While swept face ALE methods do not suffer from mesh tangling to the same extent as Lagrange methods thanks to the remap step, they do not couple well across cell corners and, in this case, does not conserve total energy. A modified ALE method which uses mesh-mesh intersection, has recently been introduced. The method uses an intersection based remap, xALE[7], which naturally couples across cell corners and conserves total energy. The single material calculations were rerun using this method. We observed third-order convergence in the velocity, pressure, and density fields as shown in Figs.11 - 13. This is a very promising result, and hints that higher-order convergence in the interface length might be obtained using xALE.

The Sedov Blast Wave as a Radial Piston Verification Test

Introduction

Analytical and computational explorations of explosion modelling have been an important area of research for many decades. Von Neumann [18], Taylor [17], and Sedov[16] all independently found a set of self-similar equations describing an idealized form of an explosion. Their simplified models use an infinitely small point source of energy as a seed. As long as the energy is large enough, this creates a strong shock that propagates outwards. Their solutions describe the self-similar equations for the evolution of state variables as the shock progresses. Because Sedov's solution was the most rigorous of his time, these self-similar equations are referred to as "Sedov solutions" in this paper. Sedov solutions can be used to understand many real-world phenomenon involving blast waves, including the evolution of supernovas [9], detonation of high explosives, and expanding laser plasmas[13].

With the constant increase in computing capabilities, an increasing number of companies and researchers are depending on hydrodynamics codes. Verification tests allows the developers of these programs to determine if the simulations converge to the analytical solutions. Convergence tests can be performed to ensure that the code will continue to increase in accuracy as resolution increases. The Sedov blast wave is of great value as a verification problem because of its semi-analytical solution for all relevant state variables. A researcher can test his or her hydrocode by modelling a Sedov blast wave and then computing the precise error in the code for various mesh resolutions and various hydrodynamics models. Sedov verification tests are useful for for testing the ability of the approach to deal with features such as shock waves, radial propagation of energy, and large variations in the energy.

The traditional set-up of a hydrocode simulation for the Sedov blast wave corresponds to an initial condition problem. In this setup, thee energy from the point source is uniformly distributed in the zone nearest the origin. This introduces a secondary source of error into the test; this error is due to the imperfect modeling of the point source. It has generally been assumed that convergence rates are still maintained since both the error from the improper representation of point source and the discretization error go to zero in the continuum limit. Nevertheless, it has been hypothesized the errors in some hydrocode tests can be attributed to the error associated with [12].

This paper introduces a new method for modeling the point source that eliminates this secondary error. In order to avoid using an energized cell of finite size, the computational domain was modified to remove the zone(s) adjacent to the origin. The effects of the energized point source can then be applied to the exposed vertices as boundary conditions. As the energy propagates outwards, the vertices on this boundary are driven by a velocity boundary condition corresponding to the semi-analytic solution. While the initial condition method would only approximate these velocities near the origin, this method applies them exactly. This makes the Sedov problem equivalent to a radial piston problem in which a shock wave emanates from the origin due to an applied velocity boundary condition.

This method represents a significant departure from the typical methodology in at least two regards. First, the typical setup as an initial condition problem is transformed into a boundary condition problem. This proposed test verifies the ability of a hydrocode to convert applied velocities into internal energy, similar to the Noh problem [15]. Second, this approach eliminates the errors resulting from approximating the energy point source as an energy source of finite volume. This allows one to determine whether errors seen in a typical Sedov test, where the Sedov problem is run as an initial condition, are due to this secondary source of error. One can then analyze how well any particular hydrocode performs independently of the setup errors.

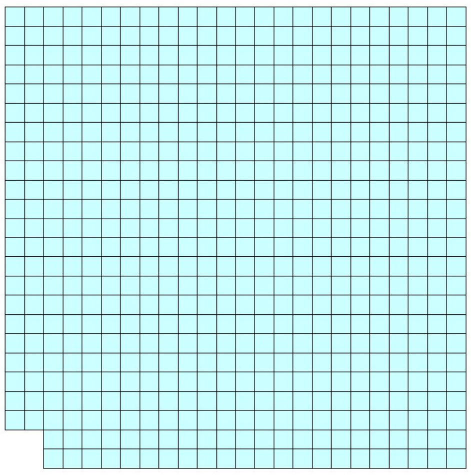
Test Problem Methodology

Once the element containing the origin of the blast wave has been removed, the appropriate boundary conditions must be applied to the exposed nodes. These values can be found using the semi-analytic Sedov solution; this methodology is summarized here and explained in greater detail in the following subsections. Sedov's original solution results in a semi-analytic solution which gives a nondimensional radius, λ and a nondimensional velocity, f. Using this similarity solution, one can then find the velocity of a particle at any position and at any time. The kinematic ODE relating position and velocity must be solved to find the vertex velocities as a function of time. The end result of this is a set of velocity boundary conditions that can be used in a Lagrangian hydrodynamics code. This approach was tested in the FLAG hydrocode to model the Sedov blast wave without the use of a finite-size energy source.

Mesh Modifications

Two separate types of computational domains are described in this paper. The first type corresponds to a “classical” or initial condition Sedov problem, where the blast wave is initiated by an energy source of finite dimensions. The second corresponds to the “radial piston” or boundary condition Sedov problem, where a continuous set of cells adjacent to the origin are removed.

For the velocity BC Sedov problem, the exact same computational domain is used with only one modification. Two examples of this modified computational domain can be seen in Fig. 14. A portion of the mesh next to the origin is removed; this region can be of various sizes and shapes. The aforementioned velocity boundary conditions are applied to the vertices exposed by removing the cells adjacent to the origin. The initial density and computational discretization are otherwise identical to the classical Sedov solution.


(a) Box Mesh

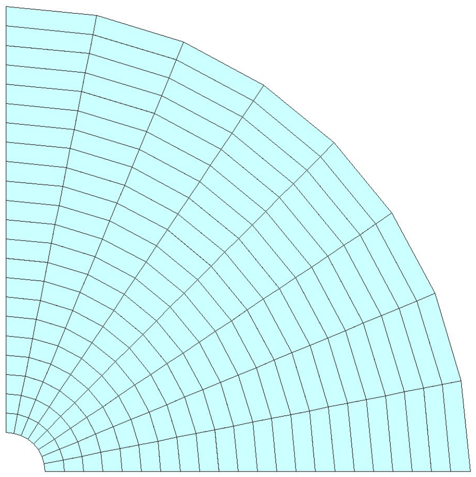

(b) Radial Mesh

Figure 14: RZ meshes used in this work

Solution to Sedov’s Equations

Kamm explained a robust method for solving Sedov’s equations[11], which is summarized here. The Sedov solution assumes that the fluid of interest is compressible, inviscid, and governed by the gamma-law equation of state. The most difficult portion of the calculation is relating the integral of energy over the domain to the state variables inside the shock region. Once this integral relation is solved, the similarity solution can be found using set of algebraic equations. These algebraic equations can be used to find nondimensional values, among which are f and λ, which are defined as:

$$f = v/v_s \tag{22}$$

$$\lambda = r/r_s \tag{23}$$

where r is the radial distance from the origin of the blast to any defined point, r_s is the radius of the shock wave, v is the radial material velocity of the same defined point, and v_s represent the radial material velocity immediately following the shock. Other similarity variables also exist for density and pressure. These similarity variables are coupled, meaning each value of f corresponds to exactly one value of λ. Because the similarity solution involves nondimensional measures of radius and time, a function relating f and λ is valid for all positions and all times. Selecting a specific position corresponds to selecting a specific value of f or λ. Since λ is the radial position, it is easiest to use this as the independent variable and f as a dependent variable. Once f and λ are known for a given point, all the relevant properties (density, velocity, pressure, or internal energy) can be computed by multiplying the similarity variables by the post-shock conditions. The calculation of these similarity variables was performed using the code presented in Reference [11].

Generation of Velocity Boundary Constraints

Once the relation between radius and velocity is determined, a table of boundary conditions must be generated for the hydrocode. The velocity of a node can be represented as a function of the radial distance and time, or $v = g(r,t)$, where g is a function. Due to the Lagrangian nature of the solution, this node represents a particle that must be followed throughout the time domain. This can be done by substituting the nondimensional velocity into the elementary ODE relating position, velocity, and time:

$$dr/dt = v(r,t) = f(r,t)v_s(t) \tag{24}$$

where f is the nondimensional ratio of velocities for a given point and time and v_s is the radial material velocity immediately following the shock. From this kinematic ODE the velocities and positions of individual node points can be solved. In order to include these velocities in the hydrodynamics setup, a table of velocity BC values was created. The same time steps were used for both the ODE solver and the table of velocity BCs in order to maintain simplicity and uniformity. As one can see from Fig. 15, the ODE resembles stiff problem due to the sudden change in velocities as the shock wave passes. Several different numerical ODE solvers were considered, varying from the basic Euler forward method to the 4th order Runge-Kutta method. More complicated ODE solvers designed for stiff problems were not considered to be necessary.

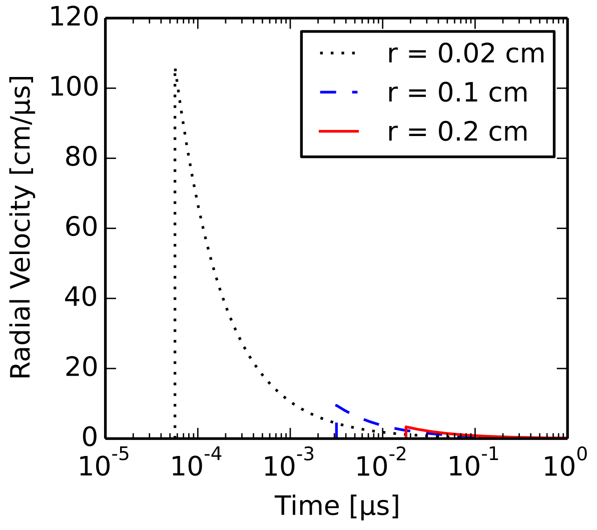


Figure 15: A comparison of material velocities for particles at three different initial positions, as a function of time. The velocity jump for a position of $r = 0.02$ cm is about two orders of magnitude higher than the jump for $r = 0.2$ cm.

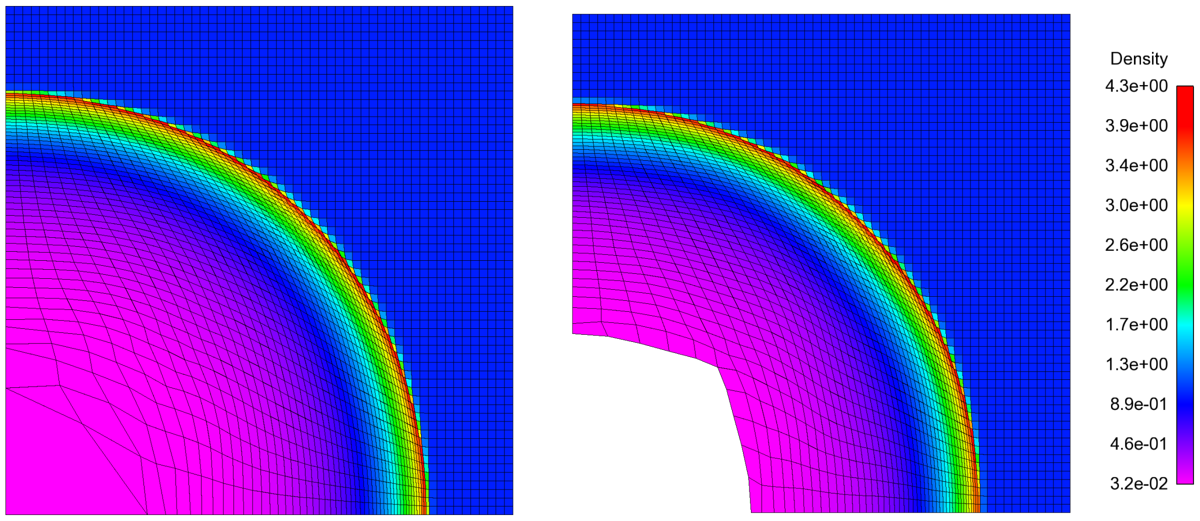
The greatest success was found using a method similar to that proposed by Bogacki and Shampine (hereafter referred to as BS23)[1]. This method compares a second and third-order Runge-Kutta method to estimate the error with each new time step. If the error is above a specified threshold, the step size is reduced until error falls below the specified threshold. This method allows adaptive temporal refinement to be employed and places explicit limitations on truncation errors.

Summary of Necessary Steps

The following is a summary of the steps taken to run a verification test using the proposed BC method. An example of typical results is shown in Fig. 16, where the density fields and meshes of the IC and BC method are shown side by side.

1. Determine the state variables to be used (blast energy, inital density distribution, initial pressure, and gamma).
2. Remove a section of the mesh adjacent to the origin of the blast wave.
3. Determine the radial distance from the origin to each vertex on the exposed boundary.
4. Solve the the semi-analytic equations described by Kamm to find the similarity solution relating f and λ.
5. For each vertex, determine the starting time when the shock will hit.
6. For each vertex, use the initial position, starting time, post-shock conditions, and solution for f and λ to solve the ODE for r, v, and t.

7. Write these values of radial velocity as boundary conditions in the hydrocode input.
8. Run the verification test.
9. Compare the hydrocode solution to semi-analytic solution.



(a) Initial Condition Method (b) Boundary Condition Method

Figure 16: The density and mesh are shown for the IC and BC approaches

Test Implementation

The following subsections show an example implementation of the verification test proposed in this paper. The setup parameters are described and the error in each of the implementation steps is quantified. Convergence results and plots of the error are included in the next section.

Description of FLAG Hydrocode

A proof of concept of this method was implemented on a research Lagrangian hydrodynamics code FLAG[5,6]. The conservation equations were solved using staggered grid hydrodynamics (SGH), which decomposes the problem variables into two interconnected meshes. The conservation of momentum is solved on a control volume around a node, imparting a specific velocity to this node. The density, pressure, internal energy, and other state variables are defined at the cell centers. Similarly, the conservation of energy equation is solved on the cell boundary. The two meshes correspond to the set of nodes, which correspond to velocities, and the set of cell centers, which correspond to the other state variables.

In order to better capture the changes in state variables across the shock, the Godunov-like approach suggested in[14] was used. This method employs the use of an approximate Riemann solver. This approach is second order accurate on smooth flows and 1st order accurate at shocks. One advantage of this method is that it is very robust against mesh instabilities such as hourglass modes and chevron modes.

For the test problem calculations, a CFL limiting value of 0.025 was introduced to limit the time steps. In addition, a maximum time step of 0.0001 µs was employed to ensure that properties would remained uniform. An initial time step of 1E-7 µs was specified to ensure that the time steps would not be too large initially. Finally, a maximum ratio of 1.01 between time steps was imposed to ensure that the solution maintained small time steps initially.

Computational Domain and State Variables

Though a Sedov blast wave can be planar, cylindrical, or spherical, the problems in this paper are strictly spherical blast waves. The computational domain is reduced through a boundary condition and by assuming RZ axisymmetric coordinates. Under these two reductions, the computational domain can be represented by a 2D set of coordinates, though each cell is actually an axisymmetric RZ cell. In order to fully examine the Sedov verification test, two types of meshes were examined for both the initial condition (IC) and the boundary condition (BC) method. Fig. 14 shows the two types of axisymmetric meshes, which are termed “box” and “radial.” These terms refer specifically to the mesh topology. Though the removed portion can be any arbitrary shape, in this paper square hole is used for the box mesh and a circular hole is used for the radial mesh. Three different hole radii were considered: 0.02 cm, 0.1 cm, and 0.2 cm. The total computational domain was set to be 1.2 cm x 1.2 cm.

The ratio of specific heats (gamma) was set to 5/3, but other values of gamma could be used. The initial density was set to 1.0 g/cc everywhere, and the extensive source internal energy was set to 0.493390 Mbar g/cc. This energy was chosen so that the shock would be located at a radius of 1.0 cm at a time of 1.0 µs. The specific source internal energy used for the IC method is calculated by dividing by the mass in the source zone. The source energy must be scaled to account for the symmetry conditions in the z-direction.

Quantification of Errors in the Sedov Solution

Though the generation of boundary constraints using the method described in this paper is derived from Sedov’s exact semi-analytic solution, the implementation requires several numerical steps. As the technique presented in this paper is largely a verification technique, its utility is limited if the errors created by any of these numerical steps are greater than the errors created by grid size or the hydrodynamics code. For this reason, the errors of each step in this process are quantified in the following subsections. It will be seen that each of these steps involves errors which are smaller than the errors seen in the convergence study.

In order to obtain the velocity of points at specific positions and times, the Sedov nondimensional solutions need to be included in the program. This inclusion was implemented by performing a piecewise linear spline interpolation on a range of 10000 Sedov values. Several other possibilities also exist for finding arbitrary values of the Sedov nondimensional solution, but only one method is considered here. Errors were calculated by comparing the interpolation with known values of the Sedov solution. 10000 exact values of the nondimensional variables (not matching those used for the interpolation) were used to find the relative and the absolute error. Representative statistics for the error are shown in Tab. 1. The piecewise linear interpolation using 10000 data points was considered to be valid because absolute errors remained below 6.329E-7 and were usually on the order of 7.273E-6. Due to the high number of data points in

Table 1: **Relative and absolute errors from the piecewise linear interpolation for exact solutions to Sedov's equations**

Type	Relative Error	Absolute Error
Mean	1.545 E−6	7.273 E−8
Median	8.862 E−8	4.035 E−8
Maximum	2.110 E−2	6.329 E−7

the interpolation process, no significant advantage was seen in using a cubic interpolation over a linear interpolation.

Verification of Convergence of ODE Solver

In order to solve the ordinary differential equation 24, Euler and Runge-Kutta methods were compared on several time stepping-schemes. These methods were compared using a hole size of 0.02 cm, since this size has the sharpest discontinuity of all the hole sizes considered (see Fig 15). The relative error of these schemes is shown in Fig. 18. The relative error incurred by the ODE solver was found to be highly dependent on the type of time steps used. This can easily be seen by comparing the differences between logarithmic temporal spacing and the quadratic temporal spacing for the classical fourth-order Runge-Kutta method. From Fig. 18 it can be seen that the BS23 method has the lowest error. Though this method only employs a 3rd-order Runge-Kutta solution, the adaptive method employed gives it a lower relative error than any of the 4th order Runge-Kutta schemes. Based on the results shown, a table with 2000 values or more is recommended for the velocity BC table. The numerical generation of this table adds a maximum possible relative error on the order of 1E-8 to the solution. This error falls well below the relative error of the hydrocode itself for reasonable mesh resolutions.

A second test was also employed to ensure the accuracy of the time-stepping of the boundary conditions. The script used to generate the velocity boundary conditions was applied to a uniformly spaced set of points between 0.01 cm and 1.2 cm. These points can then be advanced to their position at time t = 1 µs. These points can represent the spacing of a simple 1D Lagrangian mesh. The volume associated with these mesh zones can be calculated as the volume of a spherical shell whose thickness is the distance between one point and the next. Since the mesh is Lagrangian and each cell retains the same mass for all time, the final density at t = 1 µs can be calculated by dividing the initial cell mass by the final cell volume. This method does not use the momentum equations and relies solely on the solution of the kinematic ODE 24, so it is a good test of whether these velocity BCs can accurately predict the final density distribution. These calculated densities and velocities are plotted versus the analytic solution in Fig. 17. From the figure, it can be seen that the BS23 method can lead to the exact density profile. The only "bad" point is the zone closest to $r = 1.0$ cm, which is a zone that bridges the shock wave. Since the properties are very different on both sides, the average properties in this zone do not match the exact properties at the zone's center. This sort of behavior is to be expected.

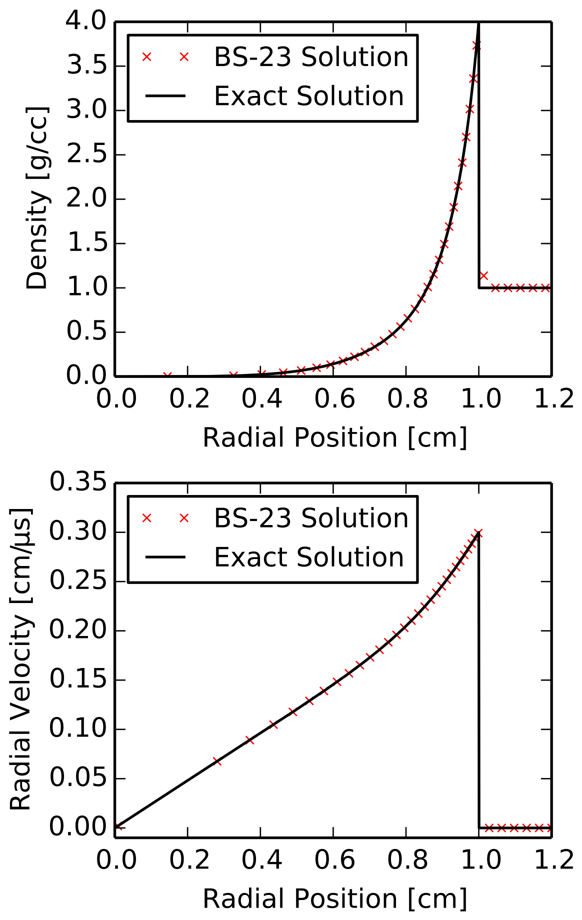


Figure 17: Comparison of density and velocity profiles generated by the BS23 solver with the analytic solution

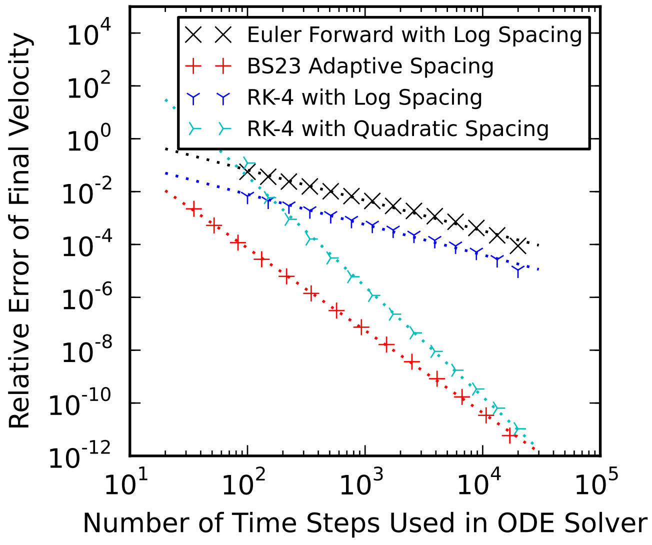


Figure 18: Convergence rates of the four ODE solvers described: The Euler forward with a logarithmic spacing, fourth-order Runge-Kutta with logarithmic spacing, fourth-order Runge-Kutta with quadratic spacing, and the BS23 adaptive time-stepping method

Convergence of New Method

The BC method proposed in this paper was observed to converge to the exact solution for density, pressure, velocity, and internal energy. This can be seen in Fig. 19. In order to verify that the code convergences to the exact solution as the computational resolution goes to zero, an error regression test was performed. This was done on both the radial and box mesh layouts. Convergence was calculated using an L1 error norm weighted by the volume of each zone. The L1 error norm for a quantity of interest, U, is calculated using the formula:

$$L_U^1 = \frac{\sum\limits_{p\in\mathscr{R}} \left(V_p |U_p - U_{exact}|\right)}{\sum(V_p)} \tag{25}$$

where V_p is the volume of the zone considered and $\mathscr{R}$ is the problem domain. This is the same error normalization used to thoroughly investigate the error involved in CCH and SGH methods in [10]. All of the convergence plots use this error normalization.

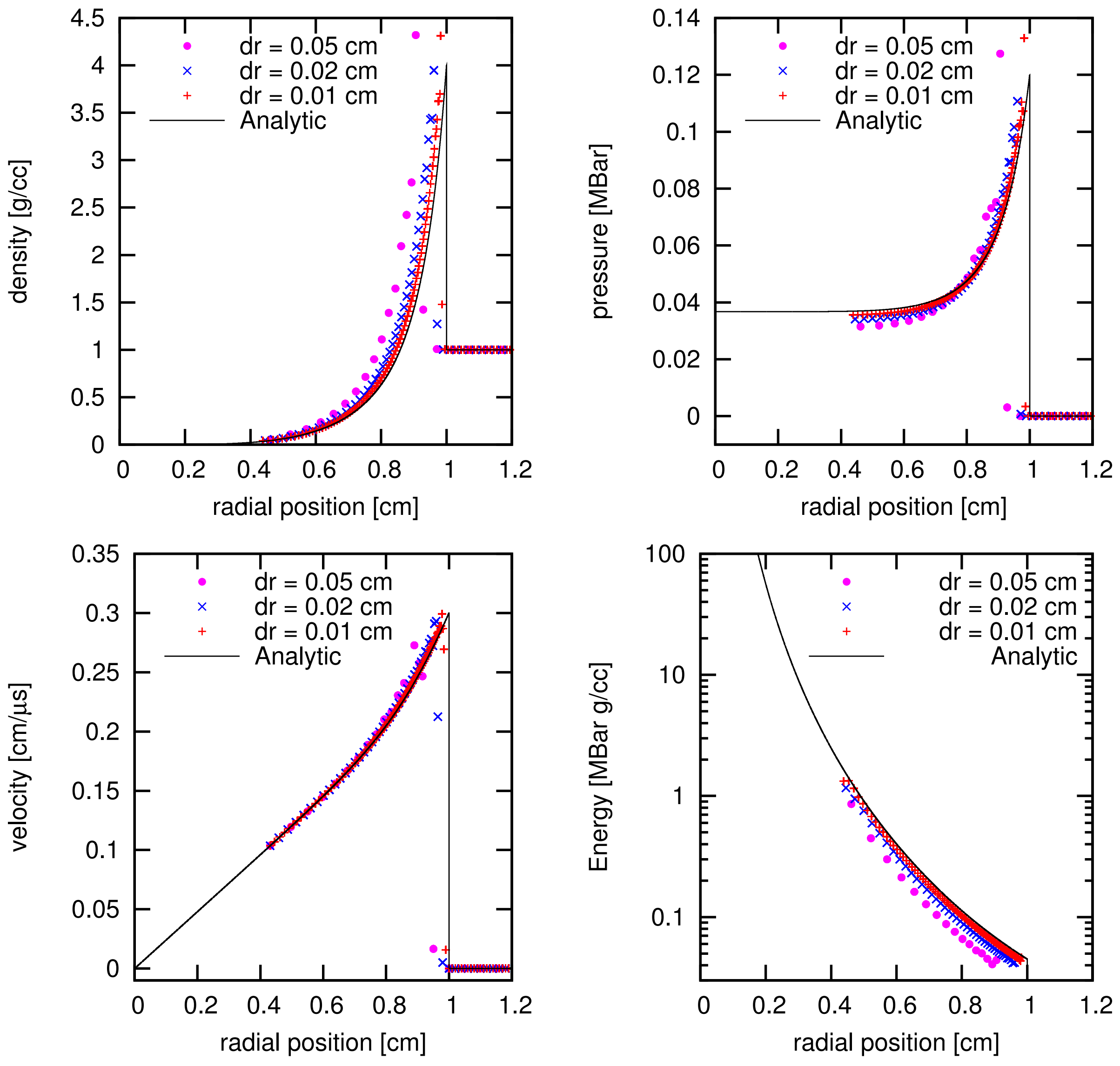
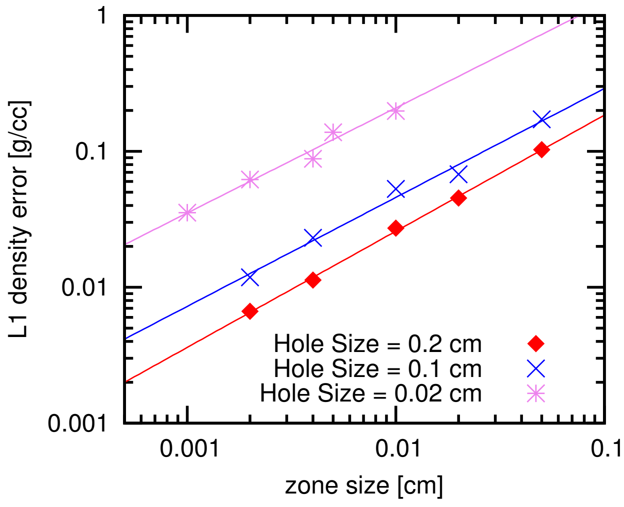


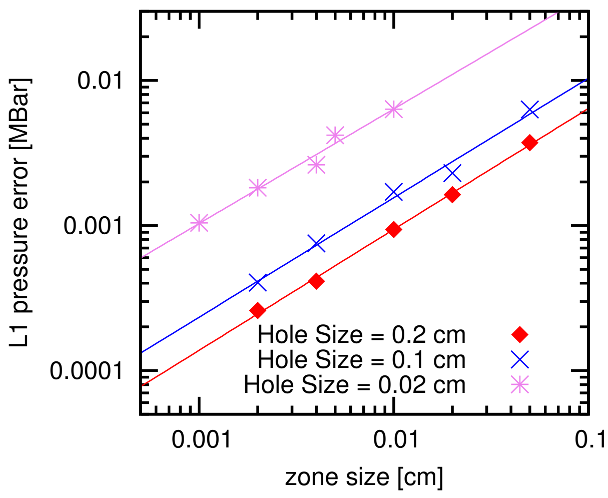
Figure 19: Plots of density, pressure, velocity, and energy for various mesh resolutions using a radial mesh.

As seen in Fig. 20, the errors shrink monotonically as the zone size decreases. Convergence rates and the correlation coefficients can be seen in Tab. 2 and 3. The correlation coefficients for the two larger holes are all above 0.97, signifying a predictable convergence with no extraneous grid-dependent errors. The errors in the final solution grow significantly larger when the size of the removed section (the “hole size”) is very small. This difference is attributed to the large magnitudes of velocity and energy near the origin at very small times. The relative scales in the velocity for various initial radii can be seen in Fig. 15. As the blast wave spreads out from the origin, the speed of the air behind the shock slows from approximately 100 cm/µs at $r = 0.02$ cm to approximately 4 cm/µs at $r = 0.2$ cm. Even small relative errors near the origin can lead to errors 1-2 orders of magnitude larger than errors far from the origin.

It can also be seen that for radial mesh with a hole size of $r = 0.02$ cm the error did not decrease significantly from the coarsest mesh resolution to the second coarsest resolution. This was believed to be a pathological error; the mesh seems to have a minimum threshold resolution required for the solution to be properly resolved. Once it passes this threshold (appearing just beyond 1/2 of the hole size) the error begins to decrease at about the same convergence rate as the other hole sizes, with correlation coefficients of 0.995 and 0.996 for the density and pressure errors, respectively. Including these lower resolutions makes their correlation coefficients go to 0.935 and 0.950 for the density and pressure errors. This lower correlation coefficient signifies that a power-law does not completely describe the convergence rate for the smallest holes at lower mesh resolutions.

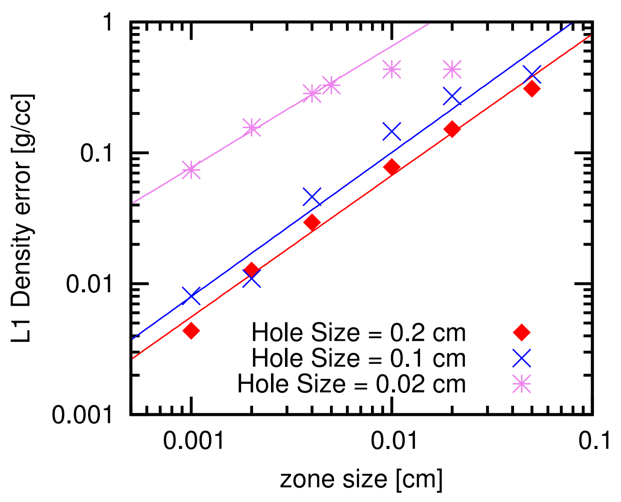


(a)

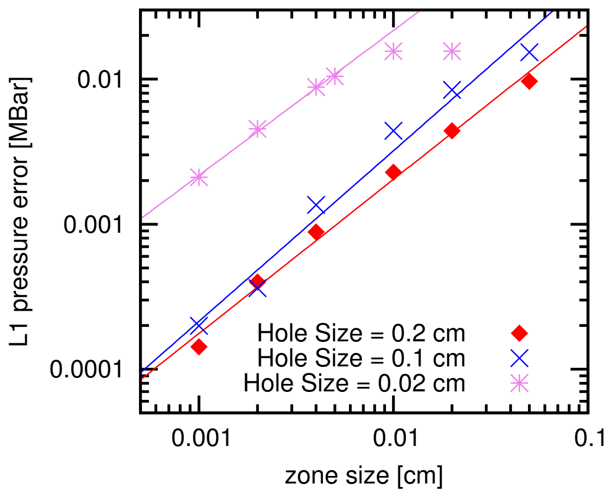


(b)

Figure 20: Convergence plots are shown for the a) density and b) pressure fields for the BC method using a box mesh. For the density, the order of convergence was 0.77, 0.80, and 0.85 for radii of 0.02 cm, 0.1 cm, and 0.2 cm, respectively. For pressure, the orders of convergence were 0.79, 0.82, and 0.83, respectively.



(a)



(b)

Figure 21: Convergence plots are shown for the a) density and b) pressure fields for the BC method using a radial mesh. For the density, the order of convergence was 0.93, 1.1, and 1.08 for radii of 0.02 cm, 0.1 cm, and 0.2 cm, respectively. For pressure, the orders of convergence were 1.0, 1.18, and 1.06, respectively.

Comparison of Radial and Box Meshes

As seen by either comparing the plots in Fig. 20 and Fig. 21 or looking at Fig. 23, the BC method generates larger numerical errors for radial meshes rather than box meshes. This is believed to be due to a dependence upon the orientation of the zones with respect to the applied velocities. Figure 22 shows the absolute errors in the box mesh as a function of both the radial position and the angle, where θ is the angle in degrees between the cell centers (for density, pressure, or energy) or the vertices (for velocity) and the horizontal symmetry plane.

Table 2: Comparison of the L1 density error regression between various hole sizes

	Box Mesh		Radial Mesh	
Hole Size	Order	R^2	Order	R^2
0.02 cm	0.754	0.989	0.930	0.993
0.10 cm	0.801	0.993	1.10	0.978
0.20 cm	0.854	0.999	1.08	0.994

Table 3: Comparison of the L1 pressure error regression between various hole sizes

	Box Mesh		Radial Mesh	
Hole Size	Order	R^2	Order	R^2
0.02 cm	0.789	0.989	1.00	0.996
0.10 cm	0.824	0.995	1.17	0.988
0.20 cm	0.834	0.999	1.06	0.999

According to this definition, $\theta = 0°$ corresponds to the horizontal axis and $\theta = 90°$ corresponds to the vertical axis. From this plot, it can be seen that there is a definite dependence upon θ. When the sides of the zones are aligned with the radial velocities, the error is highest. When the sides of the zones are positioned at 45° angles to the radial velocities, the errors are lowest. The higher error in the radial mesh is attributed to this angular dependency.

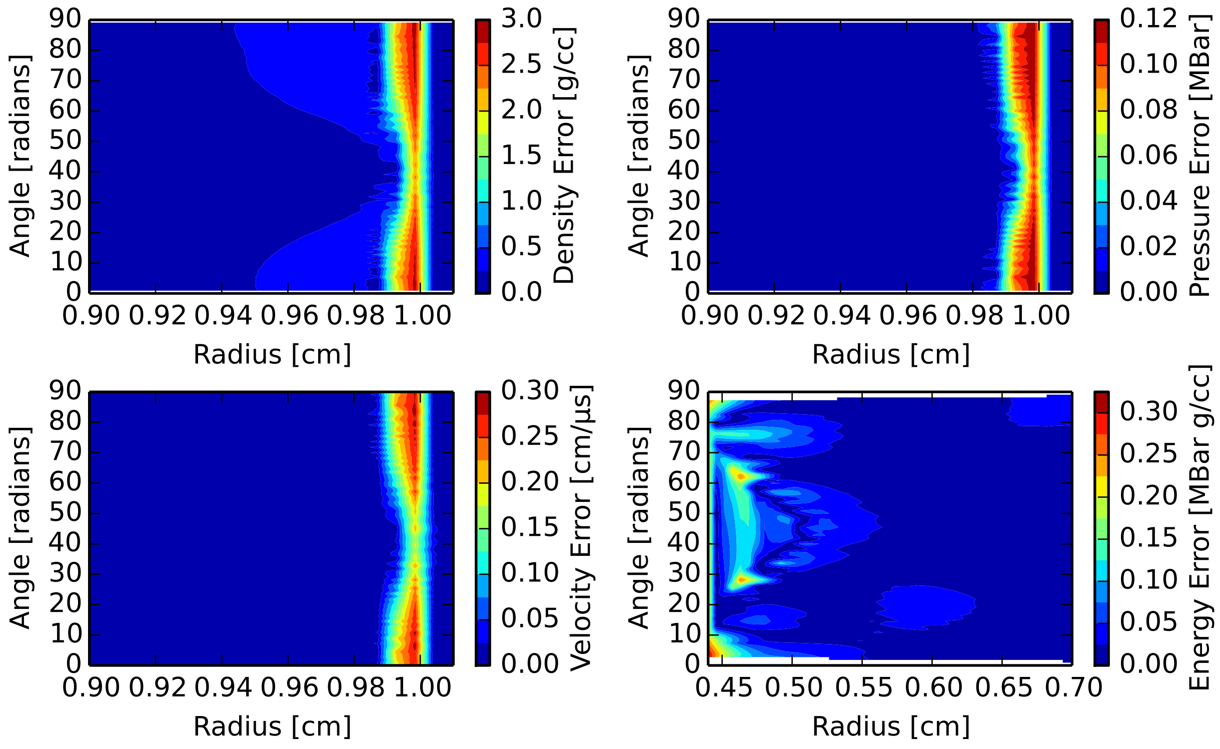


Figure 22: Absolute value of the absolute errors in the Box Mesh Setup for dr = 0.02 as a function of radius and angle

Comparison with the BC Method

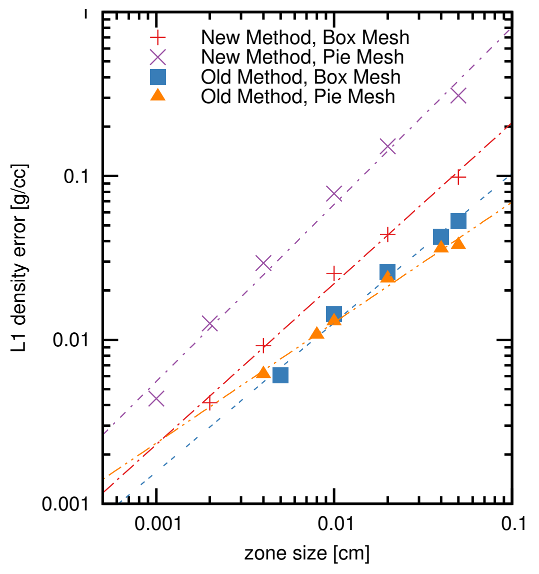
A plot of the deformed mesh and the density distribution is shown for both the IC and the BC methods in Fig. 16 Figure 23 is a comparison of the error convergence plots for both the IC and the BC condition methods discussed in this paper. The BC data for this figure corresponds to the minimum error seen at a hole size of 0.2 cm. For the IC method, convergence is very similar for a box and a radial mesh, which both converge at a rate close to 1, the expected convergence rate of the FLAG hydrocode[10, 14]. The BC method converges at approximately the same rate, with the radial mesh converging at a rates close to 1.0. However, the numerical errors are higher for this method, with the box mesh having a slightly higher error in all state variables and the radial mesh having a significantly higher error in all state variables.

Table 4: **Comparison of the L1 density error regression between IC and BC method**

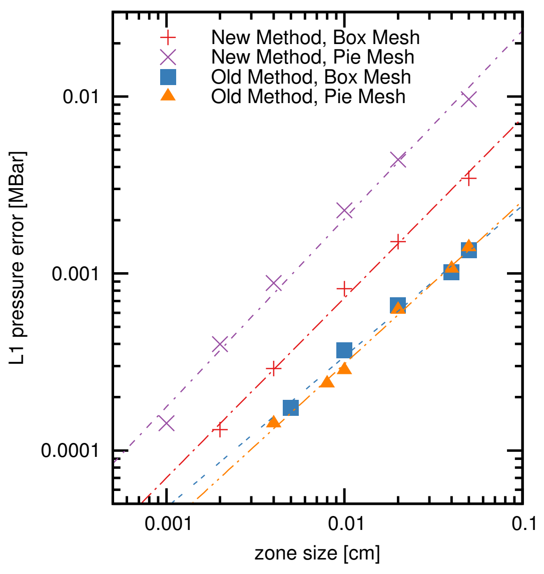
Setup	Order	R^2
IC, Box Mesh	0.911	0.994
IC, Radial Mesh	0.735	0.995
BC, Box Mesh	0.917	0.997
BC, Radial Mesh	1.08	0.994

Table 5: **Comparison of the L1 pressure error regression between IC and BC method**

Setup	Order	R^2
IC, Box Mesh	0.853	0.995
IC, Radial Mesh	0.920	0.997
BC, Box Mesh	1.02	0.998
BC, Radial Mesh	1.06	0.996



(a)



(b)

Figure 23: Comparison of the initial condition and boundary condition methods on a box and radial mesh. For the order of convergence and correlation coefficients, please see Tab. 4 and 5.

This error can be better understood by observing where the error is higher for the BC method. Figure 24 shows plots of the error in density, pressure, velocity, and internal energy at their respective radii from the origin of the blast. This figure represents the error only for radial meshes. The cusps on this plot correspond to places where the numerical solution crossed the exact solution. From this figure it can be observed that the errors in both density and energy are equal for both the IC and the BC approaches at the boundary to which the conditions are applied. This is further proof that the boundary conditions applied are precise. However, the hydrocode then differs in how this energy and density propagate throughout the mesh. This is a known challenge with Lagrangian SGH methods[14]. In verification tests, velocity-driven test problems (such as the Noh test) result in significantly higher errors than internal-energy-driven test problems. (such as the IC Sedov)[14]. It is believed that this difficulty is due to errors in converting kinetic energy to internal energies. Large errors also occur at the shock front for all cases; this is due to difficulties in capturing discontinuities in the solution[14].

Also shown is a similar comparison of the error for the IC method and the BC method at various radii for the box mesh. These data are shown in Fig. 25. It should be observed that the general trends are the same, but the error has a much wider variance due to radial asymmetry. The connection between Fig. 22 and Fig. 25 should be noted. The wider distribution of errors for the box mesh is due to the dependency on the angle θ. For any given radius, the maximum error on the box mesh is close to the average error on the pie mesh. The wider distribution of errors on the box mesh includes many errors which are lower than the equivalent error for the radial mesh. This comparison of error distributions on the box and radial mesh further explains the lower error seen on the box mesh.

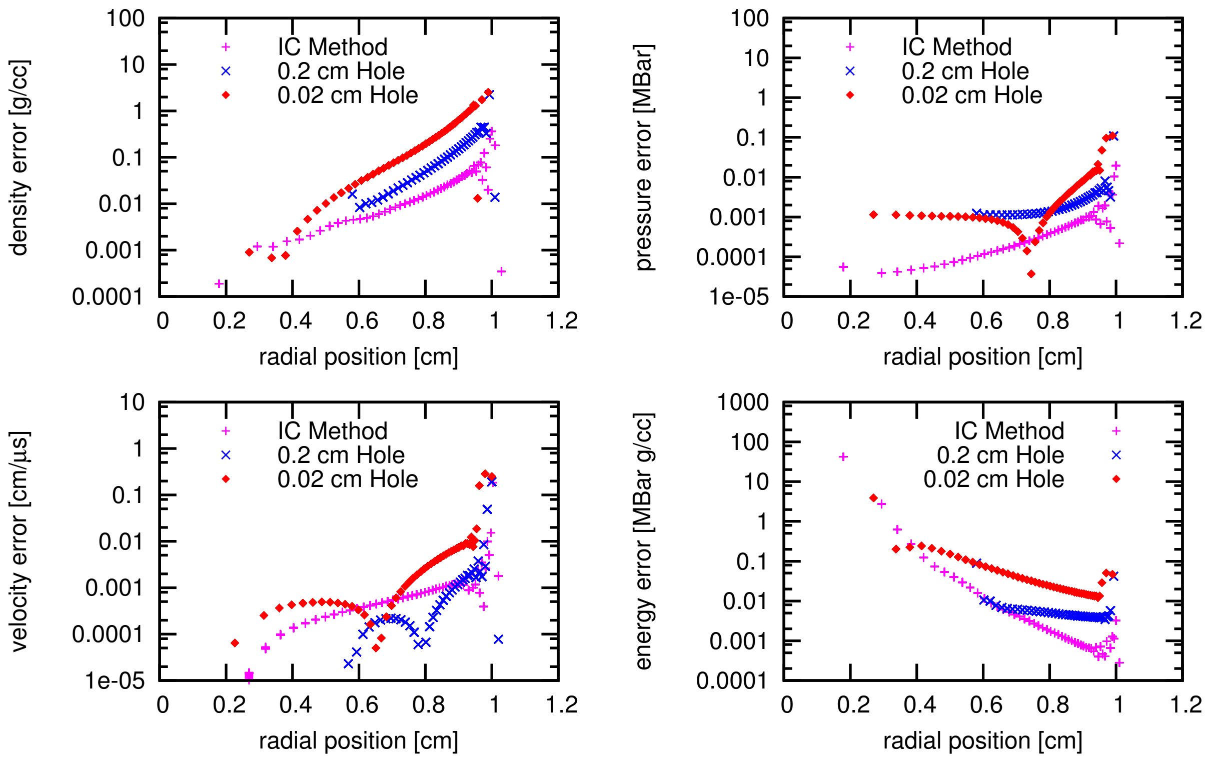


Figure 24: Absolute values of the absolute errors are compared for the IC method and the BC method using the radial mesh, as a function of the radial distance from the origin. For the BC method, two separate hole sizes, r = 0.02 cm and r = 0.2 cm, are considered.

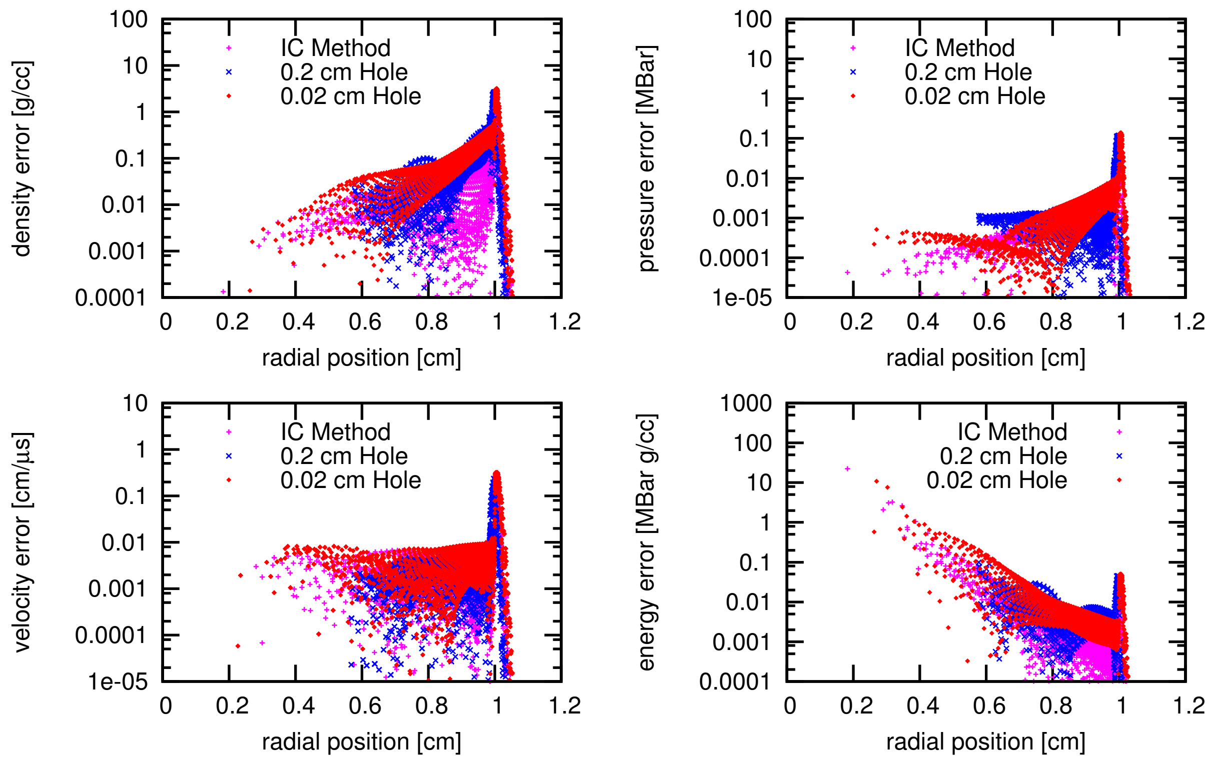


Figure 25: Absolute values of the absolute errors are compared for the IC method and the BC method using the box mesh, as a function of the radial distance from the origin. For the BC method, two separate hole sizes, r = 0.02 cm and r = 0.2 cm, are considered.

Conclusions

We proposed a new vortical test problem coupled to hydrodynamics where the particle trajectories and material-material interface evolution are known semi-analytically. Using the method of manufactured solutions we derived the energy source and initial conditions which support the 2D steady state Taylor-Green velocity field in the hydrocode. We showed that a momentum source is necessary to manufacture a system with the Rider-Kothe velocity field. The hydrocode calculations were performed on various mesh resolutions and compared to the analytic solutions of the fields and the semi-analytic material interface position and length. Second-order convergence was observed with a pure Lagrange method while calculations with swept face ALE achieved first-order convergence even in the single material case but near second-order convergence in the interface length. Significant variations were observed in the interface length associated with the second material in the triple point (three material calculation) due perhaps to a detail of the hydrocode which causes it to prefer the interface location of one material over another. Finally, third-order convergence was observed using the new xALE method in the underlying fields, however, the technique was only run in the one material calculation. The addition of materials did not have a significant effect in the convergence of the other higher order methods. Future research could determine if the convergence in xALE holds for two and three material calculations and investigate the convergence of the material-material interface length.

In this research was saw that the hydrocode converged to the semi-analytic solution using the ALE method at approximately first-order for the two and three material calculations. Fairly uniform convergence was observed in the two material calculations and first material interface in the three material calculations. Significant variations were observed in the interface length associated with the second material. This is believed to be due to the preference given to conserving quantities associated with the first material in the hydrocode. Simulations could easily be run in the future to verify this claim by observing a reversal of the interface length results corresponding to a change in the conservation preferences of the hydrocode.

The new ALE technique xALE was briefly discussed. Third order convergence was observed in the underlying fields with respect to the zone size, whoever, the technique could only be run in the one material calculation due to computational restrictions. We plan to simulate the two and three material cases in the future as time allows and we expect to see third order convergence up to three materials and a higher convergence in the interface length than was observed using the ALE method.

A new approach for conducting a Sedov verification test was presented. This method models the Sedov blast wave as a radial piston problem of varying velocity. This corresponds to a shift from an initial condition problem, where energy is deposited in a cell, to a boundary condition problem, where the boundaries are moved. Since these applied boundary conditions can exactly match the velocity of the Sedov analytical solution, this new approach allows an exact model of the energy as a point source. This method was successfully implemented and analyzed on an SGH hydrocode with a second-order approximate Riemann solver. Convergence rates were observed to be approximately the same between the classical initial condition method and the proposed boundary condition method. Error was slightly higher for a box mesh discretization and significantly higher for a radial mesh discretization. Since the boundary conditions applied correspond to the exact analytical velocity of these points, these errors were considered to be artifacts of the SGH solver. This implementation demonstrates the ability of this new test to illuminate errors due to velocity boundary conditions, similar to the Noh test problem. This implementation proves that this test is valid, and the same techniques presented in this paper can be applied to a wide range of other verification tests to introduce additional boundary condition tests.

Acknowledgements

We gratefully acknowledge the support of the Advanced Scientific Computing (ASC) program at Los Alamos National Laboratory. Thanks to Jim Kamm, Kim Molvig, Don Burton and Bob Greene for all their help with this research. We also acknowledge the efforts of Scott Runnels in organizing the LANL Computational Physics Summer Workshop, under which the research for this paper was conducted.

References

[1] P Bogacki and Lf Shampine. A 3(2) Pair of Runge-Kutta Formulas. *Appl Math Lett*, 2(4):321–325, 1989.

[2] D Burton. Lagrangian hydrodynamics in the flag code. Technical report, Technical Report LA-UR-07-7547, Available from the Author, 2007.

[3] DE Burton, TC Carney, NR Morgan, SK Sambasivan, and MJ Shashkov. A cell-centered lagrangian godunov-like method for solid dynamics. *Computers & Fluids*, 83:33–47, 2013.

[4] Donald E Burton. Multidimensional discretization of conservation laws for unstructured polyhedral grids. In *2nd International Workshop on Analytical Methods and Process Optimization in Fluid and Gas Mechanics*, 1994.

[5] Donald E. Burton. Multidimensional Discretization of Conservation Laws for Unstructured Polyhedral Grids. In *2nd International Workshop on Analytical Methods and Process Optimization in Fluid and Gas Mechanics*, VNIIEF, Arzamas-16, Russia, 1994.

[6] Donald E Burton. Lagrangian Hydrodynamics in the FLAG Code. Number NOVEMBER 2007, page 51, 2008.

[7] Donald E Burton, Mark A Kenamond, Nathaniel R Morgan, Theodore C Carney, and Mikhail J Shashkov. An intersection based ale scheme (xale) for cell centered hydrodynamics (cch). Technical Report LA-UR-13-26756, Los Alamos National Laboratory, August 2013. MultiMat 2013: International Conference on Numerical Methods for Multi-Material Fluid Flows, San Francisco, CA September 2-6, 2013.

[8] DE Burtona, NR Morgana, TC Carneya, and MA Kenamonda. Reduction of dissipation in lagrange cell-centered hydrodynamics (cch) through corner gradient reconstruction (cgr). Technical report, Los Alamos National Laboratory (LANL), 2014.

[9] R. A. Chevalier. The hydrodynamics of type II supernovae. *Journal of Astrophysics*, pages 207–872, 1976.

[10] S. Doebling. Impact of numerical treatment of shocks on verification assessment of the FLAG Lagrangian hydrodynamics code using the Sedov problem. Technical report, Los Alamos National Laboratory, Los Alamos, NM, 2014.

[11] James R. Kamm and F. X. Timmes. On efficient generation of numerically robust Sedov solutions. Technical report, Los Alamos National Laboratory, Los Alamos, NM, 2007.

[12] Jr Kamm. Evaluation of the Sedov-von Neumann-Taylor Blast Wave Solution. Technical report, Los Alamos National Laboratory, Los Alamos, NM, 2000.

[13] T. A. Leonard and F.J. Mayer. Helium blastwave measurements of laser-heated microshell targets. *Journal of Applied Physics*, 46(8):3562–3565, 1975.

[14] Nathaniel R. Morgan, Konstantin N. Lipnikov, Donald E. Burton, and Mark A. Kenamond. A Lagrangian staggered grid Godunov-like approach for hydrodynamics. *Journal of Computational Physics*, 259:568–597, 2014.

[15] William F Noh. Errors for calculations of strong shocks using an artificial viscosity and an artificial heat flux. *Journal of Computational Physics*, 72(1):78–120, 1987.

[16] L. I. Sedov. *Similarity and dimensional analysis in mechanics*. Academic Press, New York, N.Y., 1959.

[17] Geoffrey Taylor. The formation of a blast wave by a very intense explosion. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, pages 159–174, 1950.

[18] John von Neumann. The point source solution. In AJ Taub, editor, *John von Neumann: Collected Works*, pages 219–237. Permagon Press, Elmsford, N.Y., 6 edition, 1963.

[19] J Waltz. Ale applications with flag. *Numerical Methods for Multi-Material Fluids and Structures. Pavia, Italy*, 2009.

[20] J Waltz, TR Canfield, NR Morgan, LD Risinger, and JG Wohlbier. Manufactured solutions for the three-dimensional euler equations with relevance to inertial confinement fusion. *Journal of Computational Physics*, 267:196–209, 2014.

Determining the Efficacy of the Menikoff-Kober Crush-Out Model for Lunar Crater Formation

Team Members
Wendy K. Caldwell and Lucas Vargas Zeppetello

Mentors
Catherine S. Plesko, Richard Strelitz, and Scott Runnels

Abstract

Impact cratering is the dominant geological process on the moon. Because the surface of the moon is covered by a layer of regolith - inorganic particles of varying compositions and sizes - shock waves will propagate through the material differently than they would through a solid material. Thus, porosity of the regolith is a factor in how these craters form. Ralph Menikoff and Edward Kober proposed a model for crushing out the pores in a porous material. This model takes into account the porosity of a material and attempts to model how shock waves propagate through the porous material, forming a crater [5]. In this paper, we test the effectiveness of this model by running hydrocode simulations and comparing the results to experimental data. After implementing the Menikoff-Kober model in FLAG and using the hydrocode simulation to find the shock particle velocity, we were able to conclude that the Menikoff-Kober model gives a reasonable estimate for the shock particle velocity of a porous lunar regolith simulant material.

Introduction

Impact gardening is the dominant geological process on the moon. The resulting shock waves caused the lunar regolith to have varying densities and porosities in relation to the depth of the regolith [3]. The porous nature of the regolith poses a problem in modeling how these craters are formed. Shock waves behave differently in solid materials than in porous materials. Thus, an accurate porosity model is necessary in order to simulate formation of lunar craters.

Another key ingredient in modeling lunar crater formation is regolith simulant. Because the moon lacks water, the regolith has not experienced the kind of erosion and smoothing of particles that we see on Earth. Thus, the regolith consists of particles with spiked edges and increased internal friction [2]. Most of these regolith particles began with diameters greater than 1 cm and are now less than 1 mm in diameter due to disruption by impact gardening. However, large particles also exist in the regolith. Some of these particles have undergone multiple instances of breaking up and reforming as a result of the bombardment. These particles are from a variety of materials and origins [4].

Performing experiments designed to imitate the surface of the moon requires the use of a regolith simulant. Based upon samples from the Apollo missions, various lunar regolith simulants exist for use in experiments. The Johnson Space Center has developed several of these simulants. JSC simulants were created using samples from the Apollo 14 mare site. They are developed to have minearology, chemistry, and texture similar to that of mature lunar mare regolith. JSC-1A, one of these simulants developed at the Johnson Space Center, also incorporates solar wind particles [11].

In this paper, we consider the Menikoff-Kober crush-out model. We run multiple hydrocode simulations using this model, and we compare the results to experimental data obtained from impact experiments using JSC-1A. By doing this, we are able to verify that the Menikoff-Kober model provides reasonable values for particle velocity after a shock wave has propogated through a porous regolith simulant.

The 1-D Shock Tube

Before examining the four-material experiment, we first need a basic understanding of shock waves. To accomplish this, we consider the familiar shock tube problem, operating in only one dimension. We seek to verify the results produced by the staggered-grid hydrocode provided to us by our mentors [9].

Analytical Results

We first identify the initial conditions specified in the code.

Table 1: Initial Conditions and Parameters Values of the 1-D Shock Tube

Parameter	Definition	Value
ρ_0	Initial density	0.125
γ	Specific heat ratio	1.4

We then utilize the equations provided in *Physics of Shock Waves and High-Temperature Hydrodynamic Phenomena* to determine unknown values of interest[12]. From (4.1) in the aforementioned text[12], we have

$$\begin{aligned} \rho_1 &= \frac{\gamma+1}{\gamma-1}\rho_0 && (1) \\ p_1 &= \frac{2}{\gamma+1}\rho_0 D^2 && (2) \\ u_1 &= \frac{2}{\gamma+1}D && (3) \end{aligned}$$

We will first perform the calculations for the low density gas. Plugging ρ_0 into (1), we have

$$\begin{aligned} \rho_1 &= \frac{1.4+1}{1.4-1}(0.125) \\ &= \frac{3}{4} \\ \rho_1 &= 0.75 \end{aligned}$$

We now solve for D, the wave propogation speed, using the $p_1 = 0.3$ estimate from the code and Equation (2) from the text[12].

$$\begin{aligned} 0.3 &= \frac{2}{1.4+1}(0.125)D^2 \\ \Longrightarrow D^2 &= \frac{72}{25} \\ \Longrightarrow D &= \frac{6\sqrt{2}}{5} \approx 1.697 && (4) \end{aligned}$$

Using the result from (4) and Equation (3), we have

$$\begin{aligned} u_1 &= \frac{2}{1.4+1}\left(\frac{6\sqrt{2}}{5}\right) \\ u_1 &= \sqrt{2} \end{aligned}$$

Numerical Results

In this section, we will display the numerical results of the one-dimensional shock tube model. We used the aforementioned hydrocode as a basis, and we made some slight modifications to this code [10]. The resulting plots show the relationships of position and pressure. The green line in each image illustrates the analytical solution of the position of the shock. Note that as time progresses, the analytical solution and the numerical solution differ by a greater amount than at the previous time step.

Figure 1: Cell Pressure as a Function of Position with Wave Front Velocity, t=1

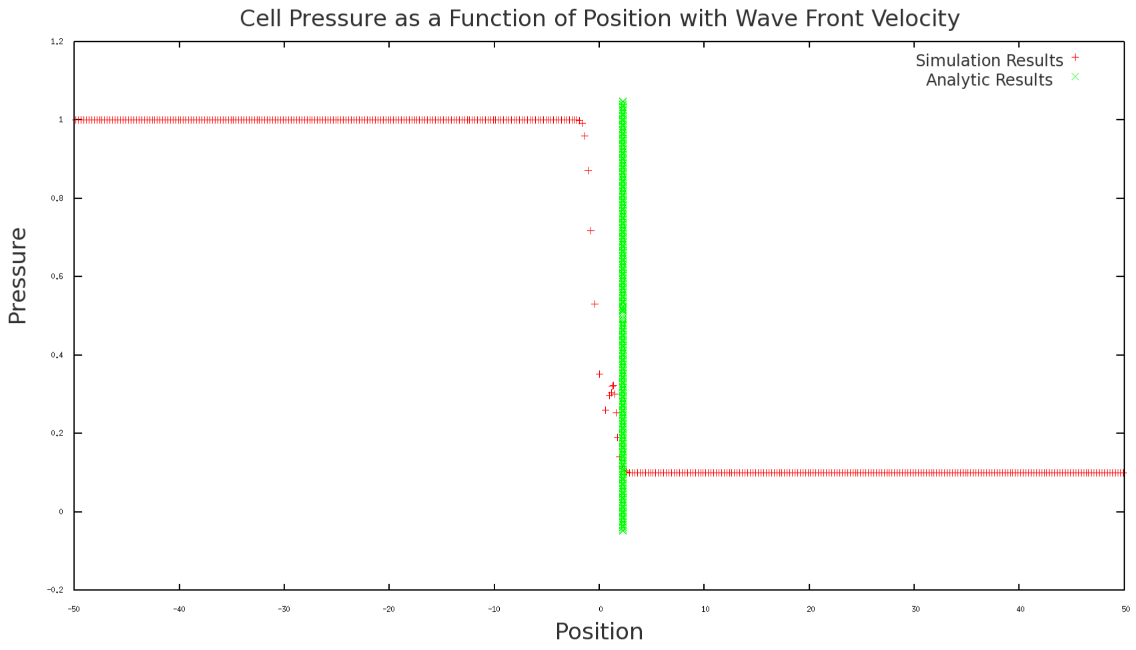


Figure 2: Cell Pressure as a Function of Position with Wave Front Velocity, t=2

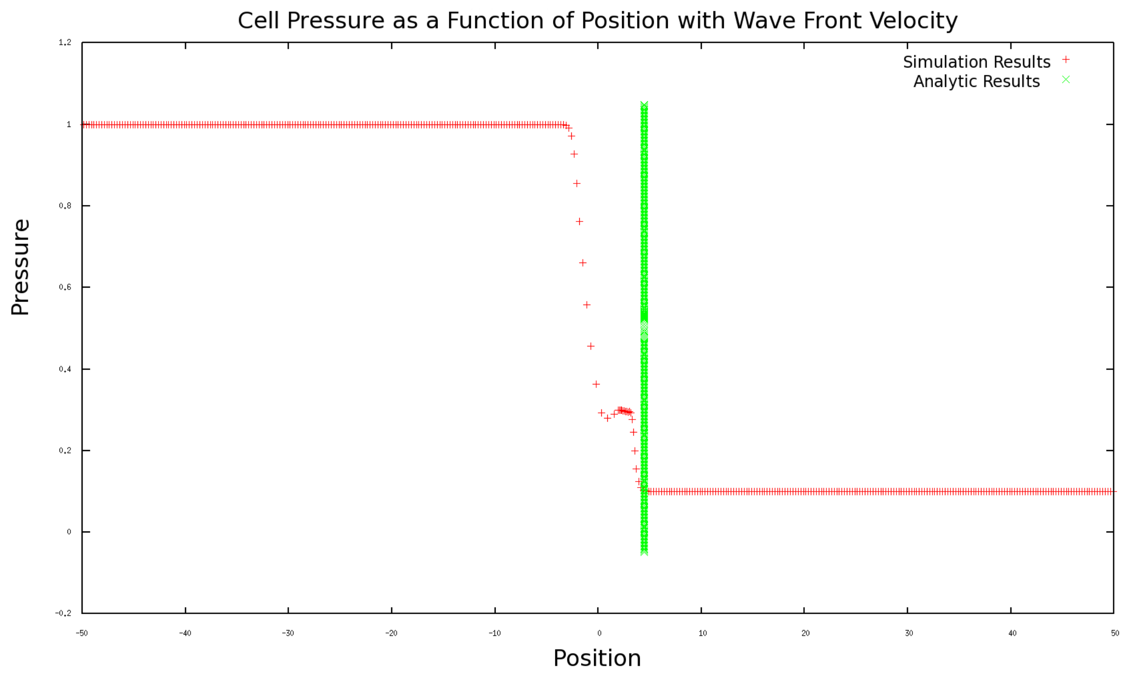


Figure 3: Cell Pressure as a Function of Position with Wave Front Velocity, t=3

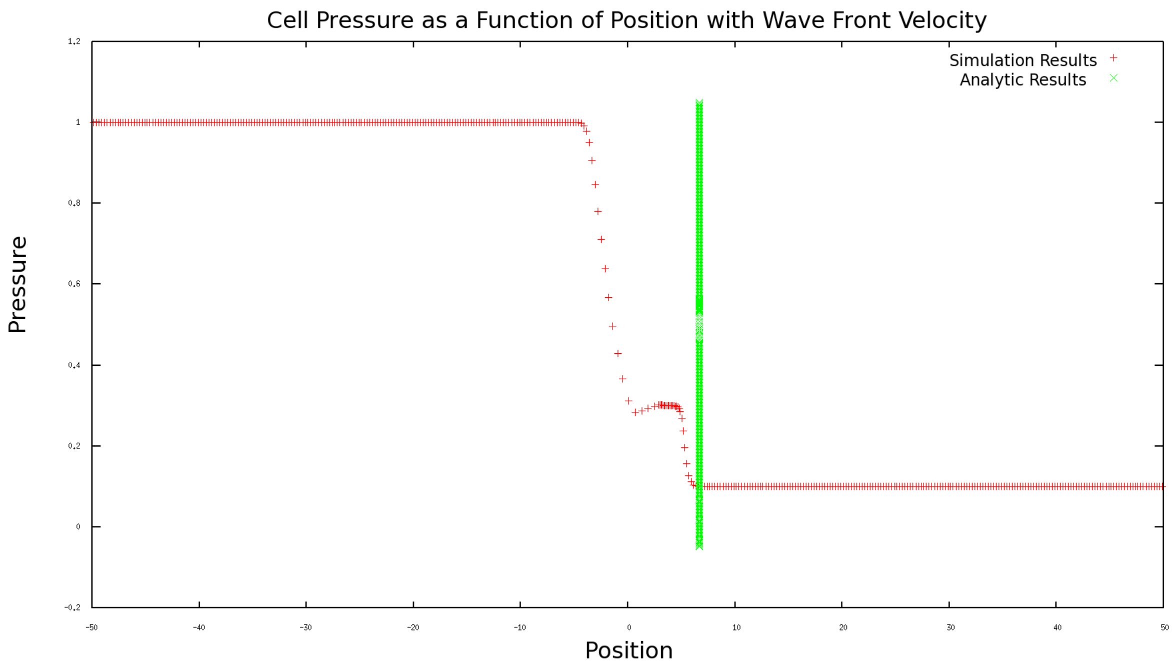


Figure 4: Cell Pressure as a Function of Position with Wave Front Velocity, t=4

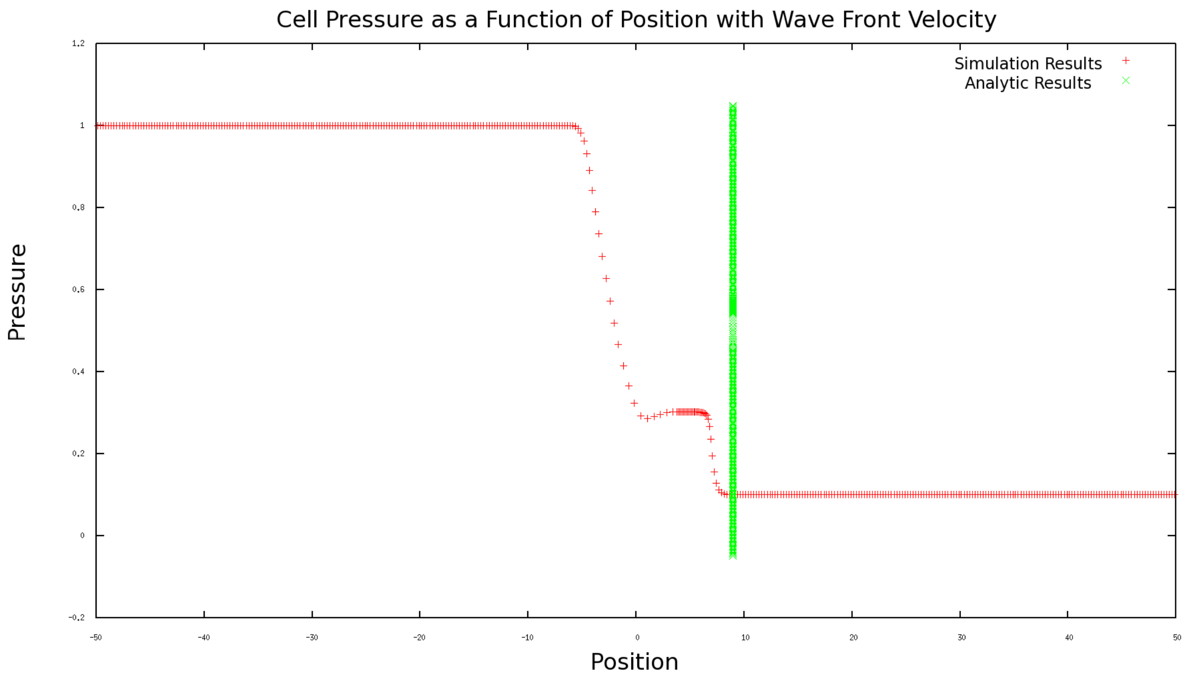
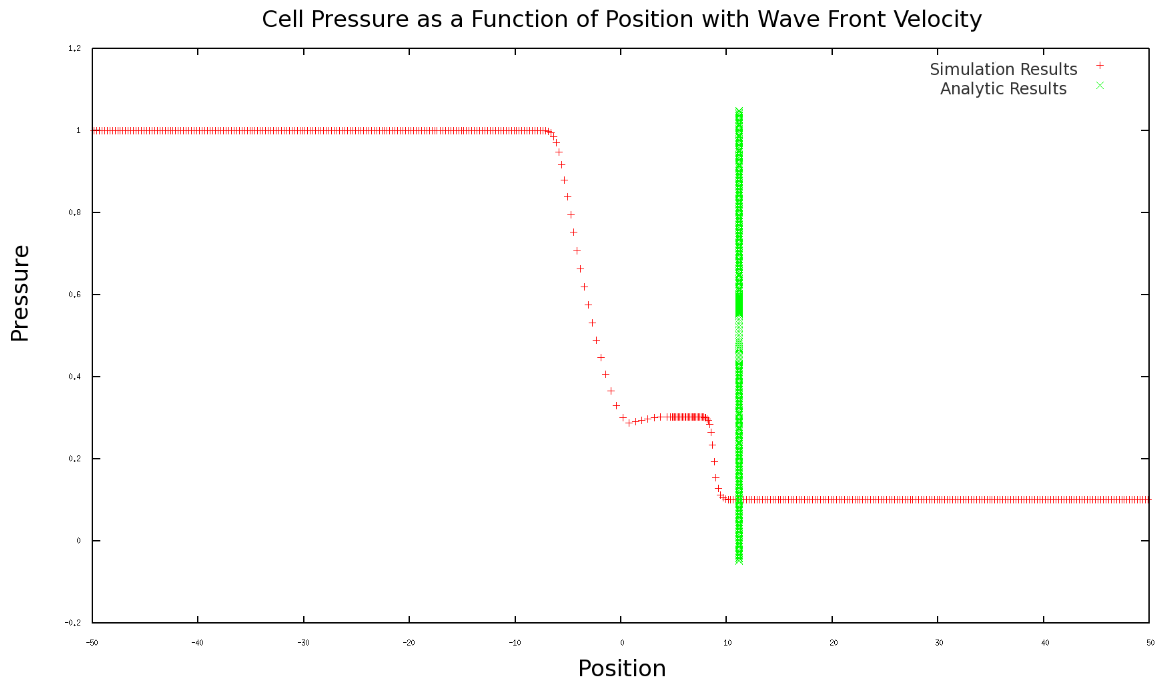


Figure 5: Cell Pressure as a Function of Position with Wave Front Velocity, t=5



Limitations

The wave front position predicted from first principles is plotted in the density figures in this section. While agreement between the 1-D hydrocode and the derivation in Zel'dovich and Raizer [12] is fairly good, the hydrocode's wave front lags behind the predicted value as time progresses. This may be due to artifical viscosity.

Additionally, the hydrocode [9] computes the c values needed to complete further analysis. The function that returns sound speed does not include density, although density is initialized as one of the input variables of the sound speed function. We also note that, computing the sound speed as in the code, the resulting units are not the expected units. We believe there may be an omission of the density term in the sound speed calculation.

Roxane Simulations

1-D Shock Tube

In the one-dimensional shock tube problem, we consider two gases of varying pressures separated by a thin membrane. When the membrane is broken, we observe a shock wave propagating through the gases. We modeled the Sod problem in Roxane [9]. The resulting figures are displayed below, with the time given in microseconds. The lines represent the density in the shock tube, which changes as the shock wave moves through the materials.

Figure 6: Shock Tube, $t = 0\ \mu$s

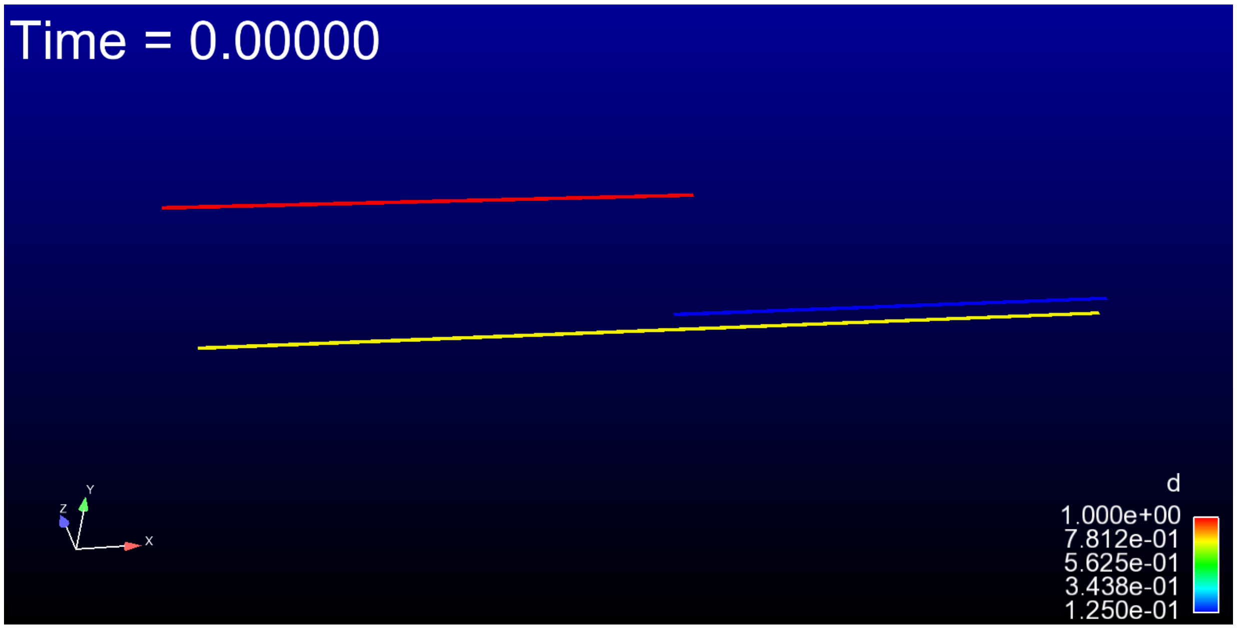


Figure 7: Shock Tube, $t = 3.05267\ \mu$s

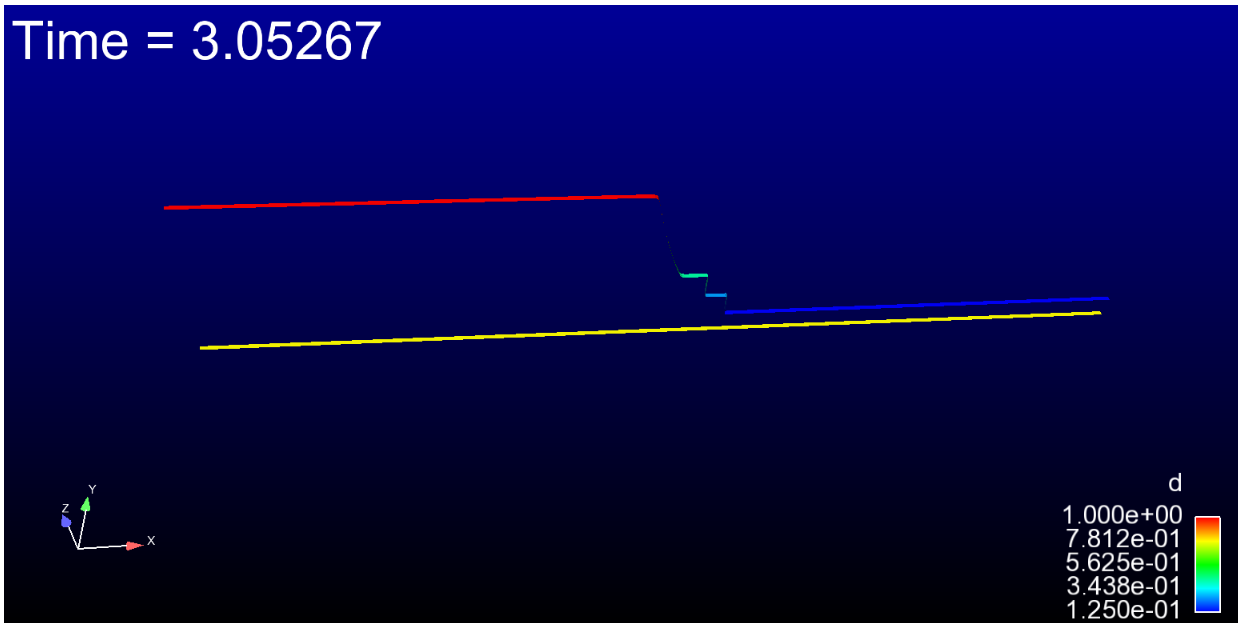


Figure 8: Shock Tube, $t = 6.28806\ \mu$s

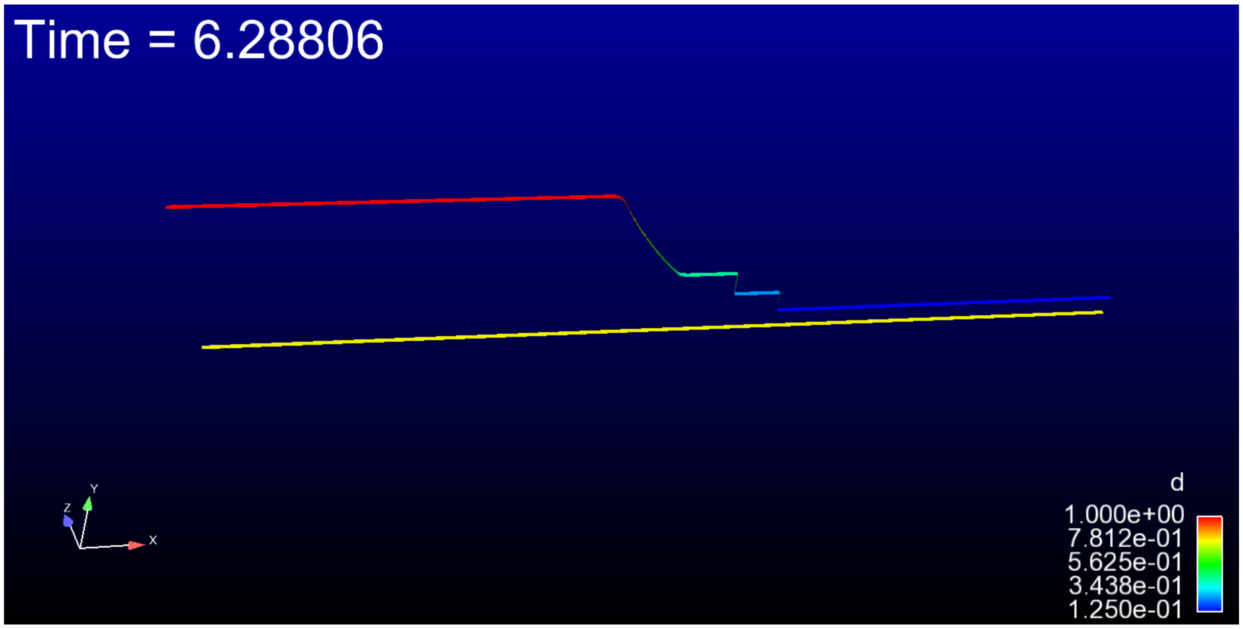


Figure 9: Shock Tube, $t = 9.52247$ μs

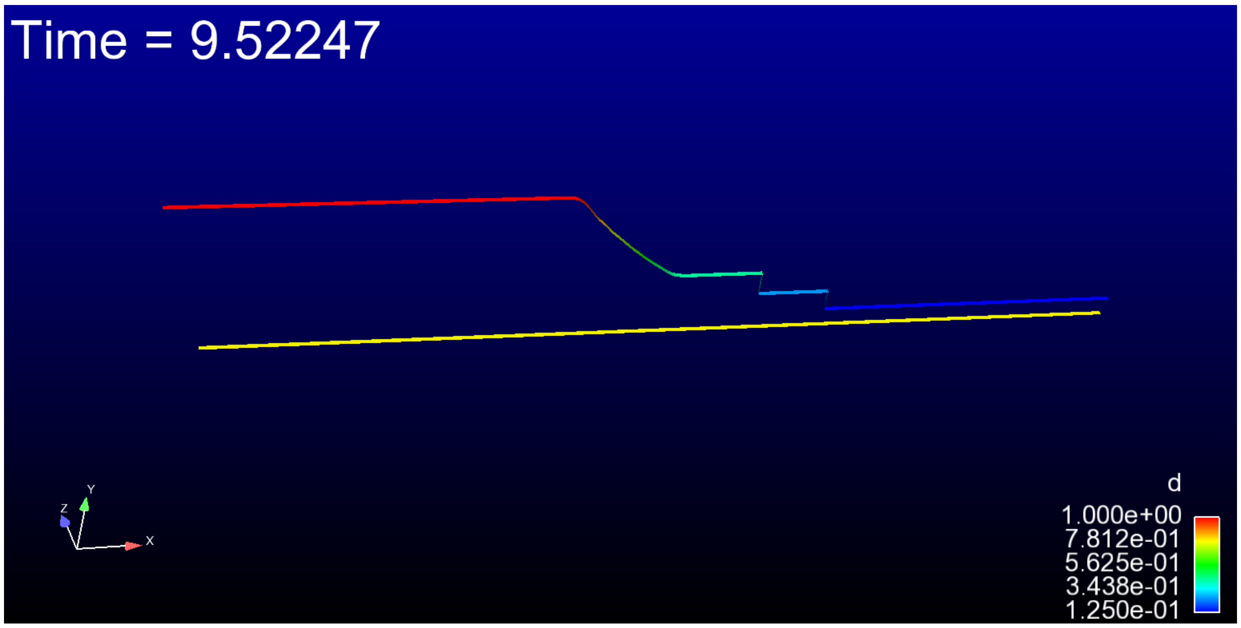
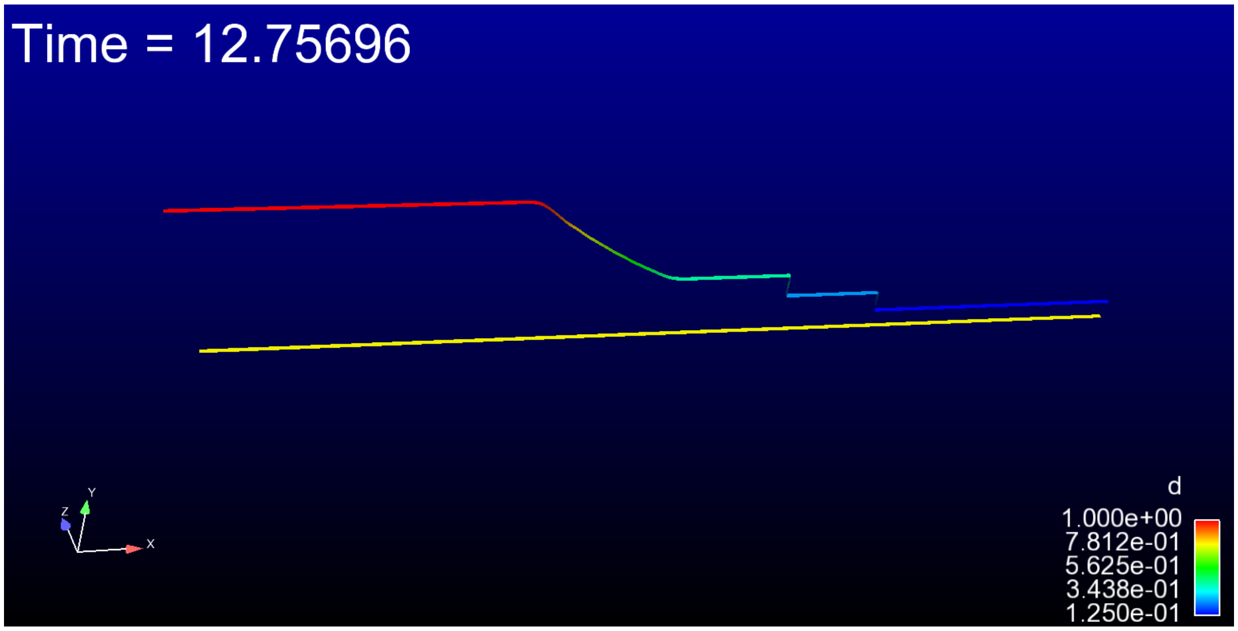


Figure 10: Shock Tube, $t = 12.75696$ μs



Impact Analysis

Introduction to the problem

In the following examples, we assume planar, normal, and parallel impacts, which can be be described by the conservation equations:

$$\begin{aligned} \rho_0 U_s &= \rho(U_s - U_p) \\ P - P_0 &= \rho U_s U_p \\ P U_p &= \frac{1}{2}\rho_0 U_s U_p^2 + \rho U_s (E - E_0) \end{aligned} \tag{5}$$

In these equations, U_s is the shock velocity, and U_p the velocity of the particles within the shock wave. We now turn to an impact situation where the projectile impacts the target with initial velocity V. After this impact, two resultant shock waves are created, one which propagates into the target material and one that propagates in the reverse direction (opposite the initial velocity). The particle speed in this back propagation shock wave must be identical to the particle speed within the shock wave that propagates into the target material; if it were otherwise regions of super high density or voids would be formed. In addition, we know that when the compression is complete, and the interface between the two materials stops the motion associated with initial compression, the pressures in the two materials will be equal. Thus:

$$\begin{gathered} V - U_{pi} = U_{pj} \\ P_{i(eq)} = P_{j(eq)} \end{gathered} \tag{6}$$

Where the region denoted i represents the projectile and the region denoted j represents the target.

Finding expressions for pressures created by planar impacts

To begin, we start with the principle of momentum conservation (5) and set $P_0 = 0$ to obtain:

$$\begin{aligned} P_i &= \rho_{0i} U_{si} U_{pi} \\ P_j &= \rho_{0j} U_{sj} U_{pj} \end{aligned}$$

Using the known equation of state for both materials, $U_s = C + SU_p$, where C and S are experimentally defined constants that depend on the material, we can write for the two materials:

$$P_i = \rho_{0i}(C_i + S_i U_{pi}) U_{pi} \tag{7}$$

$$P_j = \rho_{0j}(C_j + S_j U_{pj}) U_{pj} \tag{8}$$

Now using (6) to express U_{pi} in terms of V and U_{pj}, we obtain:

$$P_i = \rho_{0i} C_i (V - U_{pj}) + \rho_{0i} S_i (V - U_{pj})^2$$

We then set $P_i = P_j$ to obtain a quadratic equation for U_{pj}:

$$\rho_{0j}(C_j + S_j U_{pj}) U_{pj} = \rho_{0i} C_i (V - U_{pj}) + \rho_{0i} S_i (V - U_{pj})^2$$

Rearranging terms, we obtain a quadratic equation purely in terms of U_{pj}:

$$U_{pj}^2(\rho_{0j}S_j - \rho_{0i}S_i) + U_{pj}(\rho_{0j}C_j + \rho_{0i}C_i + 2\rho_{0i}S_iV) - \rho_{0i}(C_iV - S_iV^2) = 0 \tag{9}$$

Using the quadratic formula to obtain the roots of this equation, we can plug our U_{pj} values back into the momentum conservation equations (7) and (8) to find the pressure in the two shock zones of compressed material.

Examples

In this section, we will examine some examples related to shocks in order to better understand how solids behave immediately following a collision.

Example 4.1, *Dynamic Behavior of Materials*

We wish to calculate the pressure generated by the impact of a copper projectile against a copper target at 500 m/s. Using Table 5.1 in the text[6], we determine the value of C_0 to be 3.94 km/s. Because our impact rate is given in m/s, we convert to obtain the C_0 value used in this problem[6]:

$$C_0 = 3.94 \times 10^3 \quad \text{m/s}$$

Again using the table provided in the text[6], we determine S for this problem. Because both of our materials are copper in this case,

$$S_1 = S_2 = S = 1.49$$

Using the same table[6], and again because both materials are the same, we find the density:

$$\rho_{01} = \rho_{02} = \rho_0 = 8.93^*$$

* *Both Tables 4.1 and 5.1 in the text [6] list the ρ value for copper to be 8.93. The worked example in the text uses 8.92. We are proceeding with the value listed in the two tables.*

Because we have two materials that are the same, we can utilize Equation (4.20) (10) in the text[6] to obtain our particle velocity, U_p, by using the given velocity:

$$\begin{aligned} U_p &= \frac{1}{2}V \qquad (10)\\ &= \frac{1}{2}500 \quad \text{m/s} \\ U_p &= 250 \quad \text{m/s} \end{aligned}$$

We have now determined the necessary values to solve for pressure using Equation (4.17) from the text (11) [6]:

$$\begin{aligned} P_2 &= \rho_{02}(C_2 + S_2U_{p2})U_{p2} \qquad (11)\\ P &= \rho_0(C + SU_p)U_p \\ &= 8.93 \times 10^3(3.94 \times 10^3 + 1.49(250))250 \\ &\approx 9.6 \times 10^9 \quad \text{N/m}^2 \\ &= 9.6 \quad \text{GPa} \qquad (12) \end{aligned}$$

Note that (12) differs from the text's answer of 8.4 GPa [6]. This is due to a calculation error in the text.

Example 4.2, ***Dynamic Behavior of Materials***

We seek the pressures in both the target and projectile of a tungsten carbide projectile impacting a steel target at a velocity of 1200 m/s. From the text, we are given the following [6]:

Table 2: U_s, ρ, S, and C for Example 4.2

Material	U_s	ρ	S	C
Tungsten carbide	$4.920 + 1.339U_p$	15 g/cm^3	1.339	4.92
Steel	$3.57 + 1.92U_p - 0.068U_p^2$	7.85 g/cm^3	1.92	3.57

$$P_1 = \rho_1(C_1 + S_1U_{p1})U_{p1} \tag{13}$$

Using Equation (4.16) (13) and Equation (4.17) (11) from the text[6], along with substituting $U_{p1} = V - U_{p2}$, we obtain the following quadratic equation:

$$\underbrace{(\rho_1S_1 - \rho_2S_2)}_{A}U_{p2}^2 - \underbrace{(\rho_1C_1 + 2\rho_1S_1V + \rho_2C_2)}_{B}U_{p2} + \underbrace{(\rho_1C_1V - \rho_1S_1V^2)}_{C} = 0$$

We now use the information in Table 2 to find A (14), B (15), and C (16), which we will use in the quadratic formula to solve for U_{p2}.

$$\begin{aligned} A &= 15(1.339) - 7.85(1.92) = 5.013 && (14)\\ B &= -15(4.92) - 2(15)(1.339)(1.2) - 7.85(3.57) = -150.0285 && (15)\\ C &= 15(4.92)(1.2) + 15(1.339)(1.2)^2 = 117.4824 && (16)\end{aligned}$$

$$\begin{aligned} U_{p2} &= \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \\ &= \frac{150.0285 \pm \sqrt{(-150.0285)^2 - 4(5.013)(117.4824)}}{2(5.013)} \\ U_{p2} &\approx 0.805 \text{ km/s} && (17)\\ U_{p2} &\approx 29.02 \text{ km/s}\end{aligned}$$

Because the particle velocity must be lower than our impact velocity of 1.2 km/s, we use (17) as U_{p2} in (11) to calculate the target pressure:

$$P_2 = 7.85(3.57 + 1.92 \cdot 0.805)0.805 \approx \boxed{32.3 \text{ GPa}}$$

Using the substitution of $U_{p1} = V - U_{p2}$ in (13) [6], we calculate the projectile pressure:

$$\begin{aligned} U_{p1} &= 1.2 - 0.805 = 0.395 \\ P_1 &= 15(4.92 + 1.339 \cdot 0.395)0.395 \approx \boxed{32.3 \text{ GPa}} \end{aligned}$$

Aluminum on Copper Impact Problem

This problem deals with an aluminum projectile striking a copper target at $V = 1682$ m/s. We first draw up the following table for the relevant physical constants of the problem [6].

Table 3: C, S, and ρ Values for Al-Cu Impact

Parameter	Al 2024 Value	Cu Value
C	$5.33{\times}10^3$ m/s	$3.94{\times}10^3$ m/s
S	1.34	1.49
ρ_0	$2.79{\times}10^3$ kg/m^3	$8.93{\times}10^3$ kg/m^3

From (9), we can use the quadratic equation to solve for U_{pj}, where the elements in the quadratic equation a, b, and c are given by:

$$\begin{aligned} a &= \rho_{0j}S_j - \rho_{0i}S_i \\ b &= \rho_{0j}C_j + 2\rho_{0i}S_iV + \rho_{0i}C_i \\ c &= -\rho_{0i}(C_iV + S_iV^2) \end{aligned}$$

Substituting our values from the table above, we find:

$$\begin{aligned} a &= 9.567 \times 10^3 \text{ kg/m}^3 \\ b &= 6.263 \times 10^6 \text{ kg/m}^2\text{s} \\ c &= 3.543 \times 10^{10}\text{kg/m s}^2 \end{aligned}$$

We use substitute these values into the quadratic equation to arrive at the value of $U_{pj} = 5.242 \times 10^2$ m/s. As one of our initial conditions for this problem was $V = U_{pi} + U_{pj}$, we can easily obtain the value of $U_{pi} = 1.1578 \times 10^3$ m/s.

We then use the equation of state $U_s = C + SU_p$ to find the value of shock velocities as follows:

$$\begin{aligned} U_{si} &= 5.33 \times 10^3 + (1.34)1.1578 \times 10^3 \\ &= 6.88 \times 10^3 \text{ m/s} \\ U_{sj} &= 3.94 \times 10^3 + (1.49)5.242 \times 10^2 \\ &= 4.72 \times 10^3 \text{ m/s} \end{aligned}$$

Getting these shock velocities enables us to obtain the values of pressure in the two shocked regions by using (5). The final values of interest for the shocked regions of the projectile and the target are shown in Table 4.

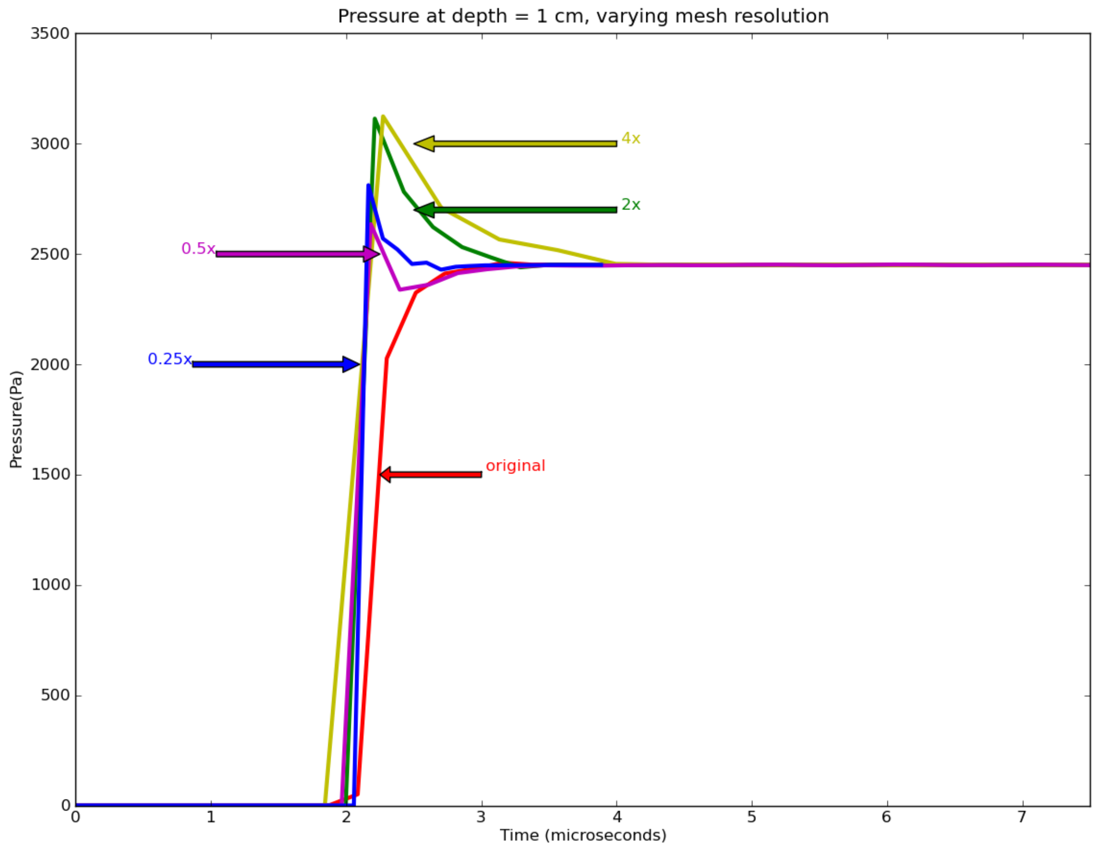
Table 4: Velocity and Pressure for Al-Cu Impact

Parameter	Al Value	Cu Value
Particle Velocity	1.1578×10^{3} m/s	5.242×10^{2} m/s
Shock Velocity	$\times10^{3}$ m/s	4.72×10^{3} m/s
Shock Pressure	22.224 GPa	22.095 GPa

Mesh Resolution Study

To aid in the verification of Roxane, we produced a resolution study for our solid aluminum on copper simulation. Starting from the original mesh, we began by multiplying the number of cells in the mesh by 2, 4, and 8 to improve resolution in the hydrocode. Then to see how the model performed under decreased resolution, we divided the number of cells in the original mesh by 2, 4, and 8 to investigate convergence under reduced resolution. The figure below shows the results of this resolution study and demonstrates good convergence, as all versions of the mesh arrive at the same value for shocked pressure of approximately 24 GPa. Roxane converges to a value that is approximately 2 GPa above the value for pressure in the shocked region arrived at by our analytic solution demonstrated in the previous section.

Figure 11: Mesh Resolution Study in Roxane



Simulations Using Different Equations of State

The simulations of the Menikoff-Kober model are intend to mimic the experiments performed on regolith simulant impact. The experimental setup consisted of an aluminum projectile 13 mm thick impacting a target composed of three materials: a 2 mm layer of copper, a 44 mm regolith sample, and an 11 mm PMMA backing. For the simulations, we utilized two different equations of state for the regolith sample while using consistent equations of state for the other three materials. For each simulant equation of state, we ran simulations both without porosity and basalt with porosity, in order to detect the effect of the porosity on the model. The results of the simulations are discussed in this section.

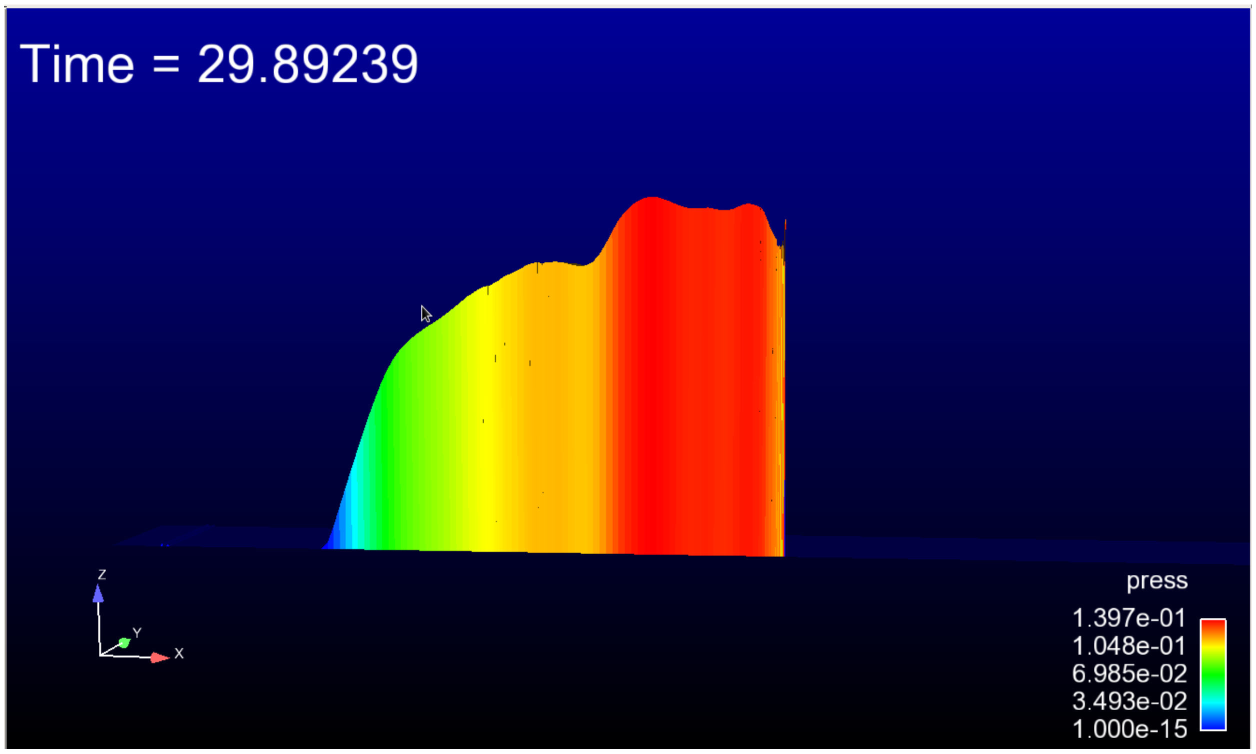
Simulant Materials, Solid

We ran simulations using the equations of state for materials similar to the regolith surrogate used in the experiment. For these initial simulations, we used only solid materials. This was in order to ensure the input file was working properly and giving reasonable results. We did not compare these results to experimental data because the experimental data was taken from an experiment that used a porous substance with a lower sound speed.

Basalt

For the first simulation, we used the SESAME equation of state for basalt. The reason basalt was chosen was due to the nature of many regolith simulants, which are composed in part of volcanic ash [8]. More than 90% of all volcanic rocks contain basalt, and many of the moon rocks collected during the Apollo missions are composed of basalt [7].

Figure 12: Pressure at the end of the solid basalt simulation



Dry Sand

To accurately model the flyer plate experiment in a one-dimensional Roxane simulation, we first need a choice of material to act as our lunar regolith simulant. The lunar regolith is composed of approximately 60% pyroxenes, 30% plagioclase, and 10% brown glass and opaques [1]. This mixture has an intrinsic density of 3.10 g/cm^3 and different experimental procedures have reduced its initial density to model the porous behavior of the lunar surface. This value for initial density of the simulant is given at 1.80 g/cm^3, which gives an initial porosity of 58.065% [1].

In the following simulation, the sample material in the Roxane simulation is dry sand (Sesame Database #7100). Dry sand is composed of 70% silica, a naturally occurring molecule that is materially similar to the basalt regolith material found on the lunar surface. The other 30% of dry sand is composed of a variety of compounds (water, Carbon Dioxide), some of which are not found on the lunar surface, which could lead to some disagreements between experiment and simulation. However, the porosity that is exhibited by terrestrial dry sand and lunar regolith material makes it a viable option for these simulations. In addition, the material characteristics of shocked dry sand have been experimentally treated by Sandia National lab, with the theoretical Hugoniots in good agreement with their experimental results. We first tested the experimental setup of the flyer plate experiments without porosity to see the qualitative characteristics of the shocked material before including the porosity model in our simulation.

Figure 13: Forward and backward propogations of shock wave

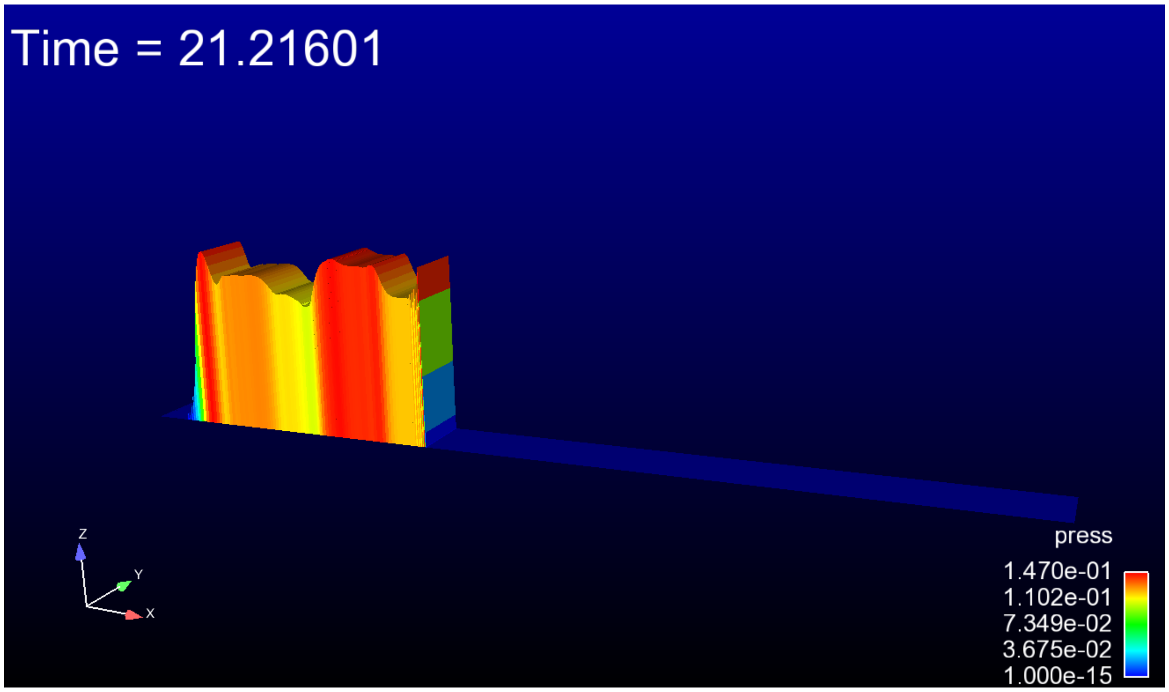
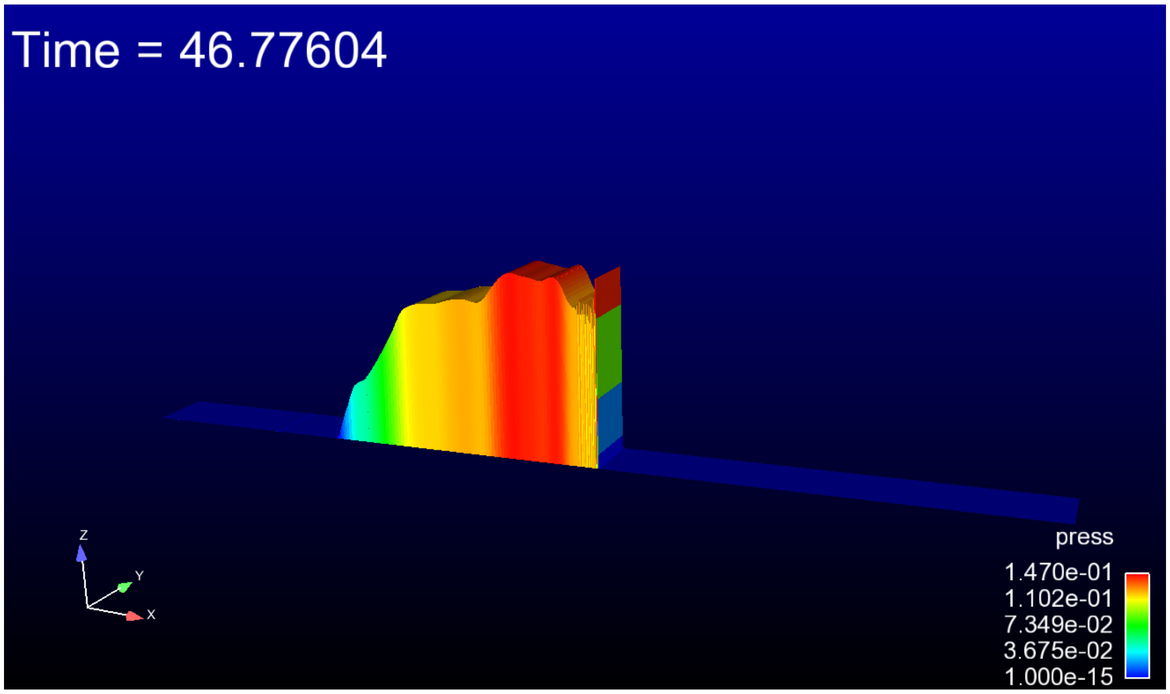


Figure 14: Pressure at end of dry sand simulation, illustrating the transfer of energy from the projectile to the simulant material through the shock wave



Sesame EOS Simulations, with Porosity

We implemented porosity in the simulant material after checking that the models for the solid materials were well behaved. We initialized the porosity to be consistent with the density of the lunar regolith. Each of our canditate materials had slighly different table densities, so we varied Φ to give a porous density for each material that matched the regolith simulant.

Basalt

When attempting to run the simulation with porosity, we encountered various problems. However, before we encountered these issues, we were able to obtain data for 3 time steps. These preliminary results are displayed in this section. The solid simulation at the same time step is shown for comparison.

Figure 15: Basalt, 100% solid, t=0.02240 μs

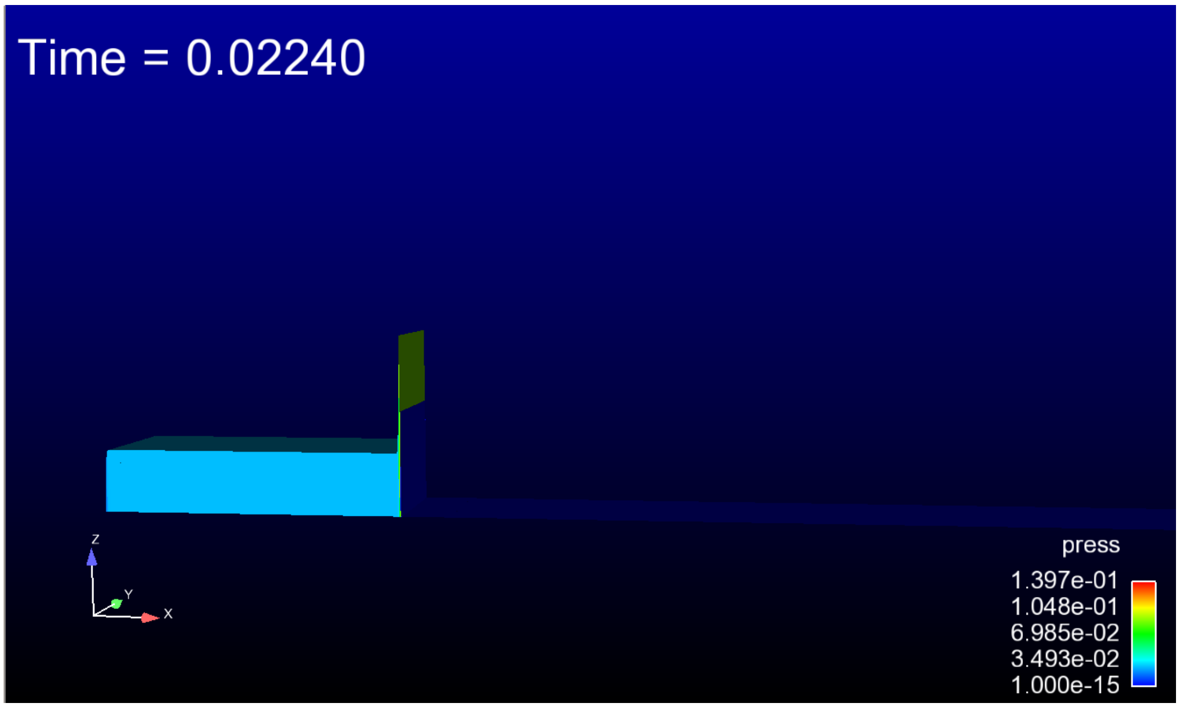
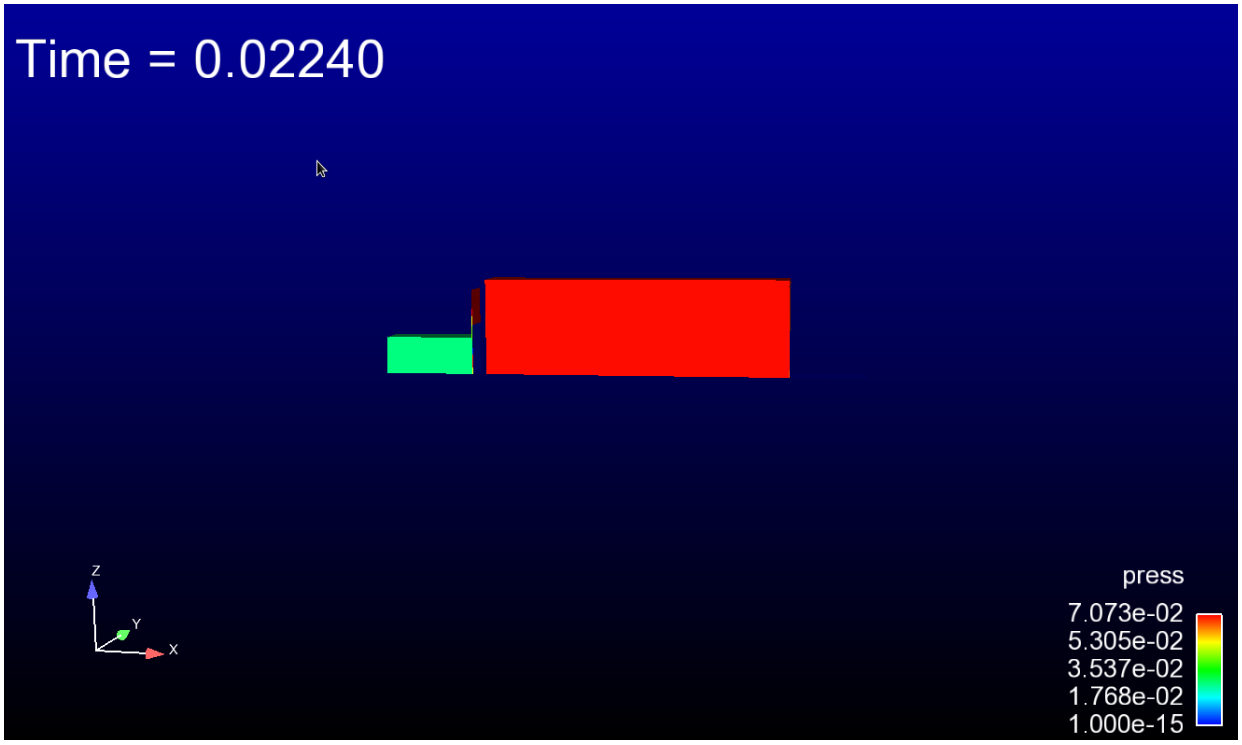


Figure 16: Basalt, 80% solid, t=0.02240 μs



These figures highlight a possible problem with the material state initialization in the porosity simulation. Note that when a porous material is used, the backing material initializes to a higher pressure relative to the impactor than in the solid material simulation. We consulted the code team about this possible issue, but we were unable to resolve the problem in the time allotted.

FLAG Simulations

Roxane vs. FLAG

After we ran Roxane simulations on the solid materials, our next step was to include porosity. The Menikoff-Kober model was an option in both Roxane and FLAG. However, the Roxane implementation did not run to completion. The FLAG implementation was sufficient for testing the Menikoff-Kober model.

Specifying the input deck

We set up the input decks for FLAG with four materials: aluminum (flyer), copper (buffer), sample material (simulant), and polycarbonate (backing). We specified these materials in the input deck with their Sesame IDs. We used an initial time step of 0.01 μs. We selected the initial temperature to be room temperature (273 K), and we utilized the Sesame tables to determine the initial density. We varied the initial porosity parameter (Φ_0) of the simulant material in our simulations. We set $r_{s0} = \rho$ for each simulant material, and we set $p_c = 1.0$.

Table 5: Material Properties and Initial Velocities

Material	Sesame ID	ρ	V_0
Aluminum	3720	2.7	1.682×10^{-1}
Copper	3333	8.93	0
Basalt	7530	2.868	0
Dry Sand	7100	2.6	0
Polycarbonate	7740	1.196	0

Sample Material: Garnet

As a test of explicit porosity modeling through the Menikoff-Kober model versus implicit porosity modeling through the equation of state, we ran the porosity model in FLAG with two substances as the simulant: garnet and garnet sand. For the garnet simulation, we set the porosity for the model. For the garnet sand simulation, we did not, because the associated Sesame equation of state already accounts for this. Table 6 shows the Sesame IDs, densities, and Φ_0 value (for garnet).

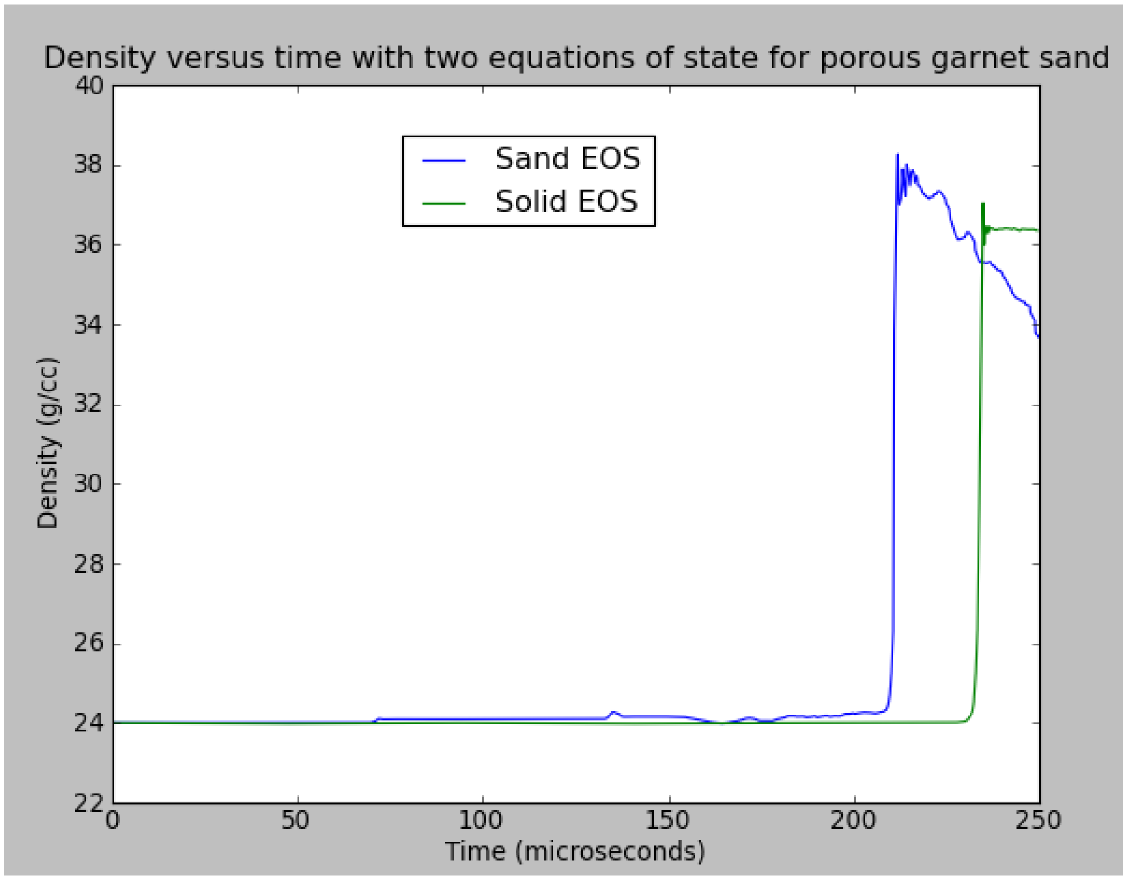
Table 6: Garnet Material Properities and Φ_0

Material	Sesame ID	ρ	Φ_0
Garnet	7760	4.05	0.5926
Garnet Sand	7761	2.4	N/A

Solid Garnet, Garnet Sand Comparison

After running the hydrocode with both solid garnet (using Menikoff-Kober) and garnet sand (using the EOS for porosity rather than Menikoff-Kober), we obtained different results for what should have been the same material. In each simulation, we used the same geometry as was used in the actual experiment, altering only the simulant material. Additionally, the impact velocities used in each simulation were identical to one another and to the impact velocity of the experiment.

Figure 17: Densities of Solid Garnet with Porosity and Garnet Sand after Pressure Adjustments



This figure illustrates that making these pressure adjustments does reduce the initial drop in density of the solid garnet with porosity. However, the densities do not allign, as one would expect two identical materials to do. The shock times are different for solid garnet with porosity and garnet sand. The garnet sand simulation had an earlier shock time than the solid garnet with Menikoff-Kober porosity simulation. The shock arrival time differed by approximately 35-40 μs. Additionally, the peak densities for the two simulations were inconsistent. The garnet sand simulation had a peak density about 2 g/cc higher than that of the simulation using solid garnet with Menikoff-Kober porosity. Consequently, the U_s value for the garnet sand simulation, which has implicit porosity from the equation of state tables that is derived from experiment, is higher than the U_s value for the solid garnet with modeled porosity.

EOS problems

Initially, we selected basalt and dry sand as our simulant materials. Upon running simulations, we discovered that we needed to provide specific internal energy instead of temperature for each material. The goal of this was to ensure that either each material initialized with zero pressure or, if that was not possible, to ensure that the regolith simulant and polycarbonate initialized with the same pressure. This was to avoid spurious pressure waves on the mesh.

Dry Sand

The dry sand material alternated between two pressure values, -4.19098×10^{-2} and -6.61653×10^{-2}. We were unable to initalize the polycarbonate pressure to either of these two values. After trying multiple positive and negative energy values, varying across multiple orders of magnitude, we concluded that there was an error in the calculation of this value. Because we were unable to initialize the specific internal energies in such a way as to avoid spurious pressure waves in the backing material, we could not move forward with dry sand as our simulant material.

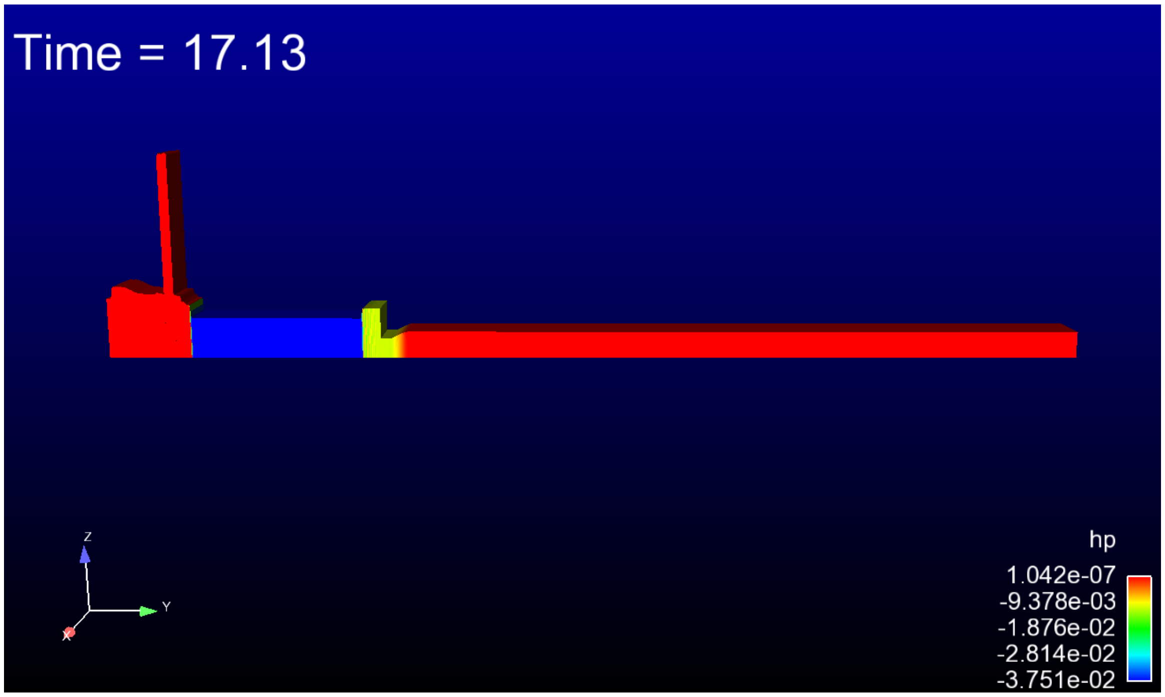
Westerly Granite

Next, we tried using westerly granite, another igneous rock with density of 2.627. We experienced the same issue with this material, with pressure limited regardless of energy. Again, we concluded that there was an error.

Nevada Alluvium

Our next material of interest was Nevada alluvium, a kind of river silt with density 2.35. We anticipated this material to be somewhat similar to the sand. However, we were again met with the problem of being unable to change the initial pressure. Once again, we had found an error. The following figures show the shock wave propagation at three different times using Nevada alluvium as the simulant. The energy has been specified to -1.45783×10^{-5}.

Figure 18: Nevada alluvium, $t = 150\mu$s



Sample Material: Mica

Next, we tried mica. The density of mica is 2.7, which is similar to the densities of the other materials we had tried. Because mica is a silicate mineral, we selected this material, expecting some similarities between it and sand. With mica, we were able to designate an energy such that the pressure initialized to zero, as desired. Thus, because we did not experience the same problems with mica as we did with the other materials, we opted to use mica for our other simulant material. The following images show velocity and density at the back of the mica.

Figure 19: Mica: velocity over time

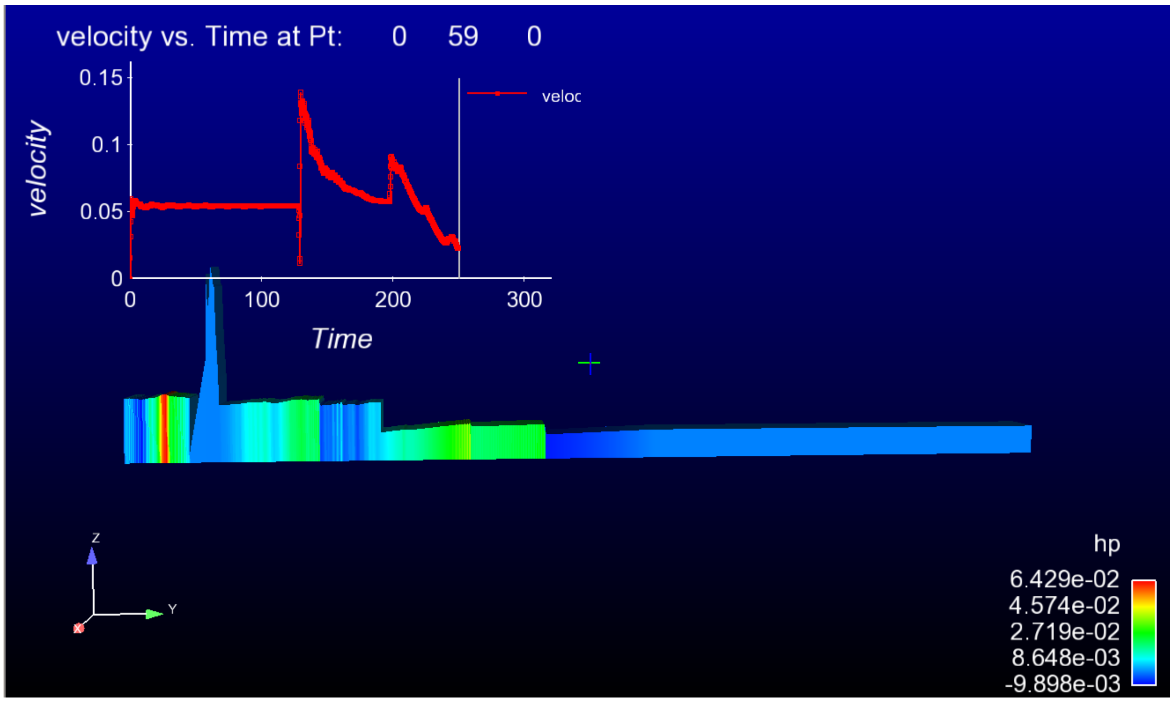


Figure 20: Mica: density over time

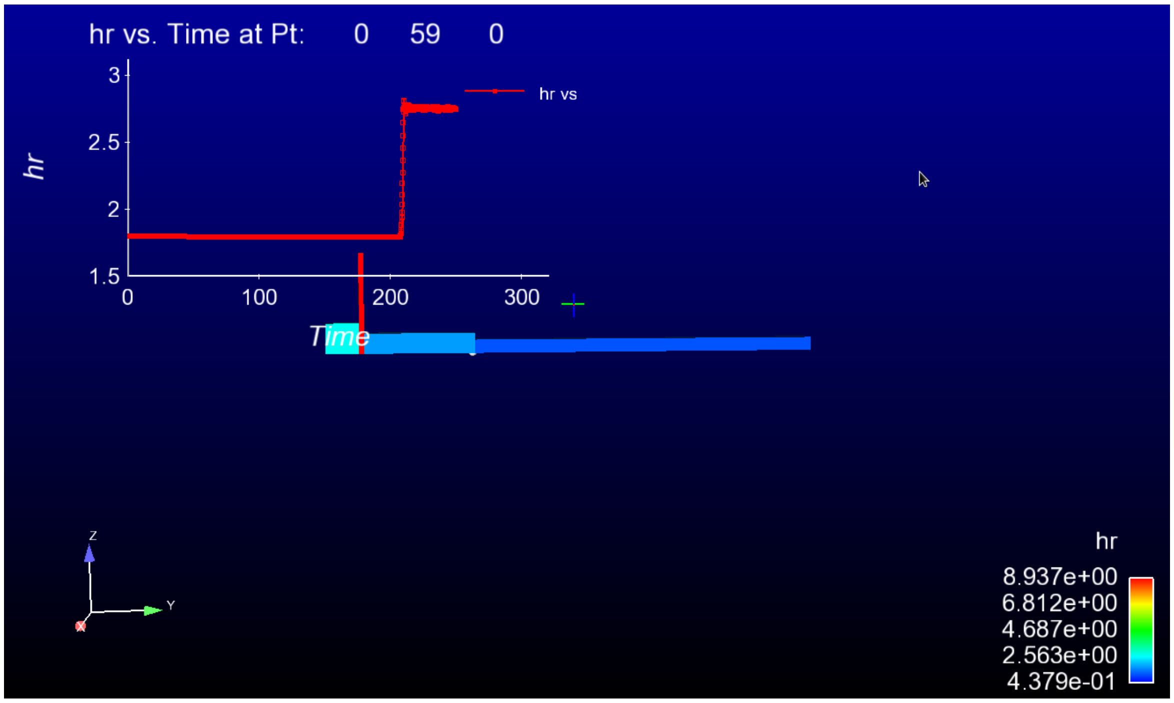
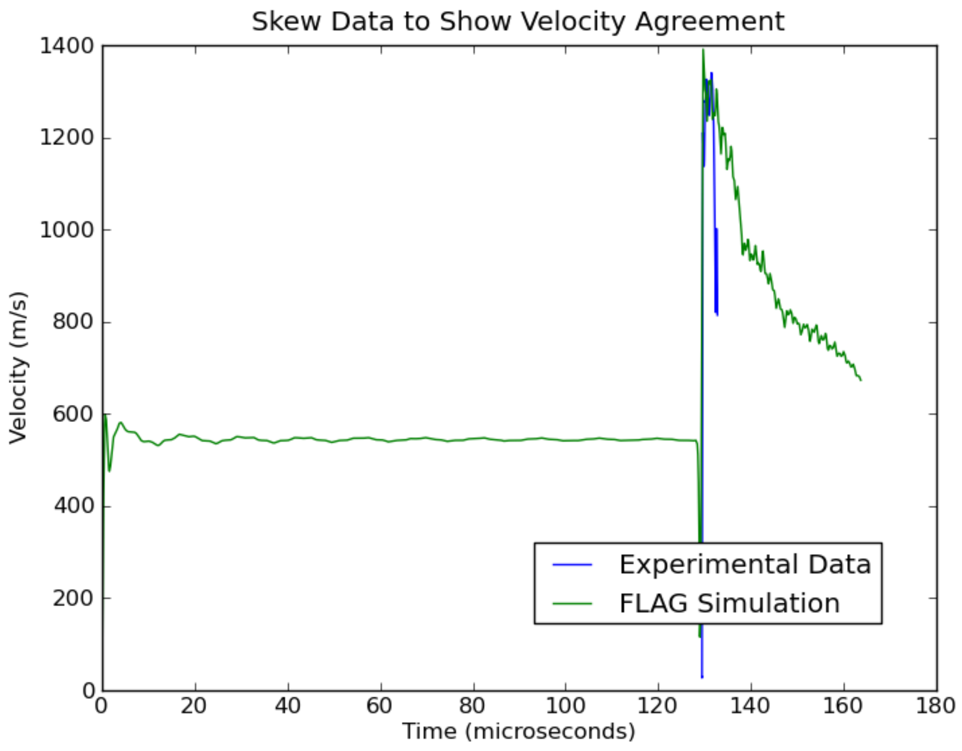


Figure 21: Mica Simulation vs. Experimental Data



The shock speed through the mica was 4.5 km/s, while the shock speed through the simulant (JSC-1A) from 5.536 km/s.

Sample Material: Basalt

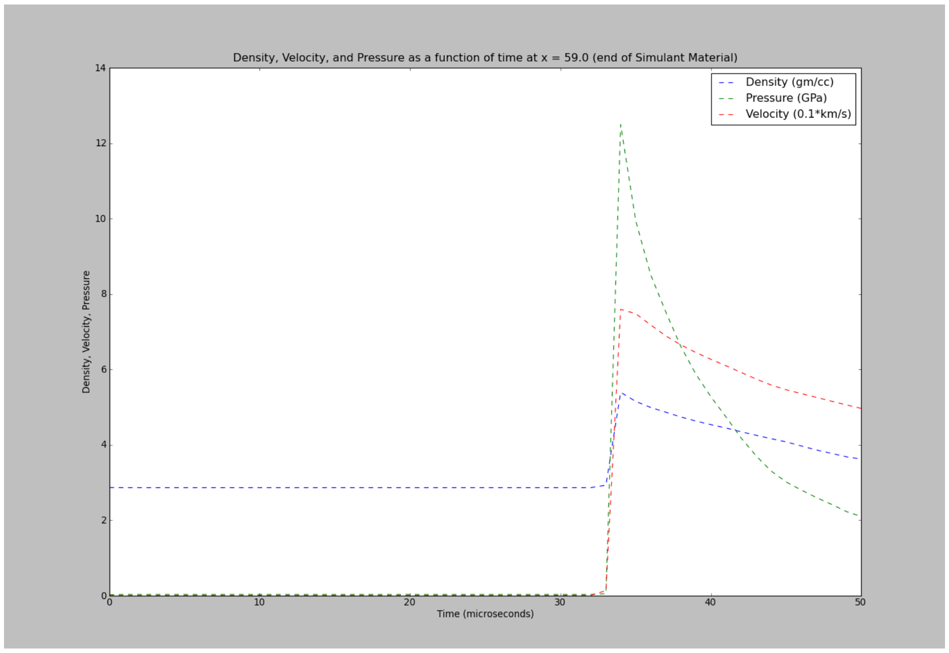
We ran FLAG with porosity, using basalt as the simulant material. We used four different values for Φ_0. Using Ensight92, we found U_p for each Φ_0 value, which we then used to calculate U_s. The results are displayed in Table 7. Basalt was chosen as the primary material for modeling the lunar simulant JSC-1A. The choice was motivated by the lunar regolith composition and the composition of the simulant itself, which is a basaltic ash with a high glass content. While basalt is not a well-defined material, meaning that any equation of state is an approximation made to some ideal basaltic composition, the sesame file was well behaved in FLAG, and we were able to equalize the pressures within the sample to avoid pressure wave propagation which threatened to upset the simulation results.

Table 7: Basalt Simulation Results

Φ_0	U_s (km/s)	U_p (km/s)
0.9	6.047	0.873
0.8	6.043	0.867
0.7	6.021	0.867
0.6276	6.028	0.867

We plotted density, pressure, and volume against time for the end of the basalt region. Figure 22 displays these results.

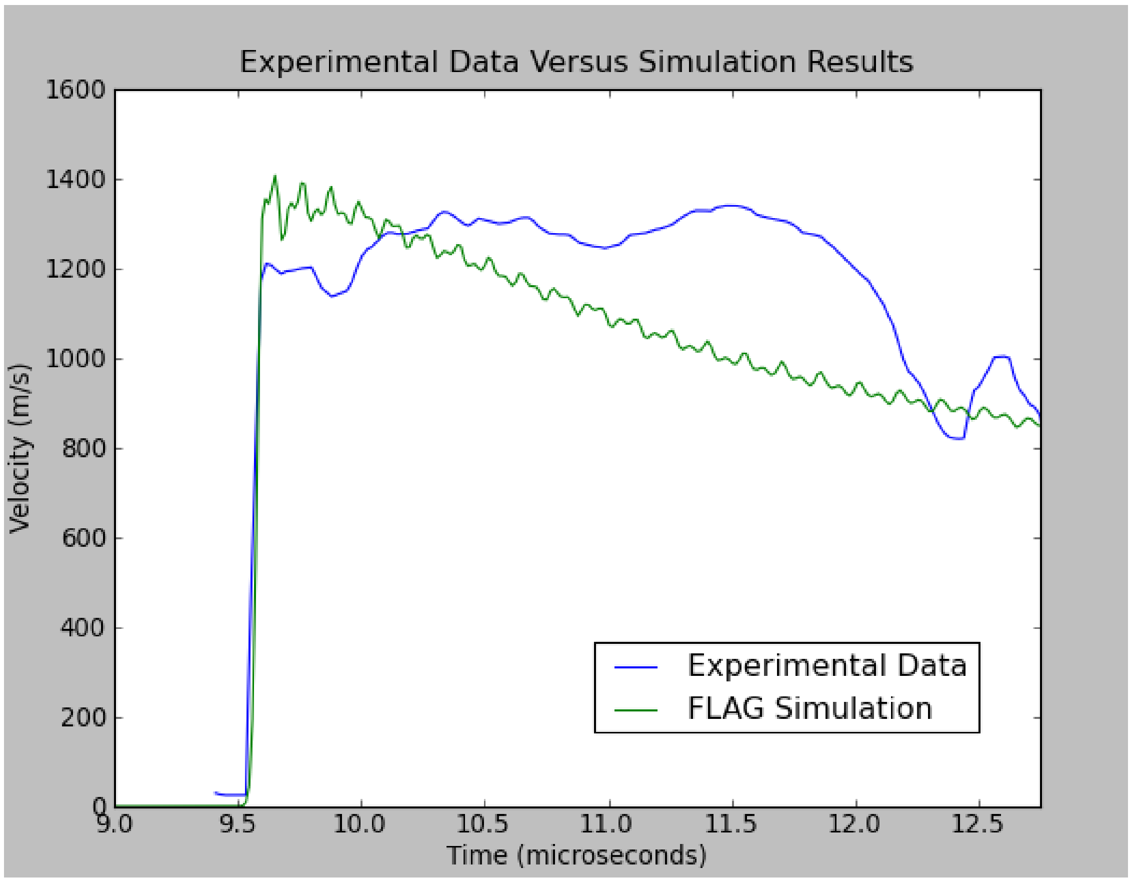
Figure 22: Density, Pressure of Basalt



Comparison to Experimental Data

Lasers measure the particle velocity at three different points in the experimental assembly, one where the flyer collides with the copper buffer plate, one at the buffer simulant barrier, and another where the JSC-1A and PMMA meet at the back end of the simulant. We compared the experimental data to a FLAG simulation and obtained the plot below.

Figure 23: Basalt Simulation vs. Experimental Data



This figure is significant because it shows that when the equations of state are compatible with the Menikoff-Kober model, the model gives accurate predictions of particle velocity for shock waves propagating through porous materials. The plot above shows an idealized scheme where the shock speeds are virtually identical. However, we estimate from the experimental data that the shock speed through JSC-1A is approximately 5.536 km/s, while our model predicted a shock speed of 4.608 km/s. This represents a non-trivial error in shock speed but only results in a 16 microsecond time lag between the experiment's shock front and that of the FLAG simulation. This discrepancy could be partially explained by the missing pressure term in the hydrocode calculations that will be discussed in the conclusions. However, we have demonstrated the model's performance on particle velocity predictions.

Conclusions

Through this study, we have verified that the Menikoff-Kober porosity crush-out model can be effective for predicting shock properties in solid material hydrocodes. The verification of this model was predicated on FLAG's implementation of the Menikoff-Kober model being compatible with the sesame tabular equations of state. However, this was not a foregone conclusion, as we experienced multiple problems with the porosity models agreeing with these tabular equations of state.

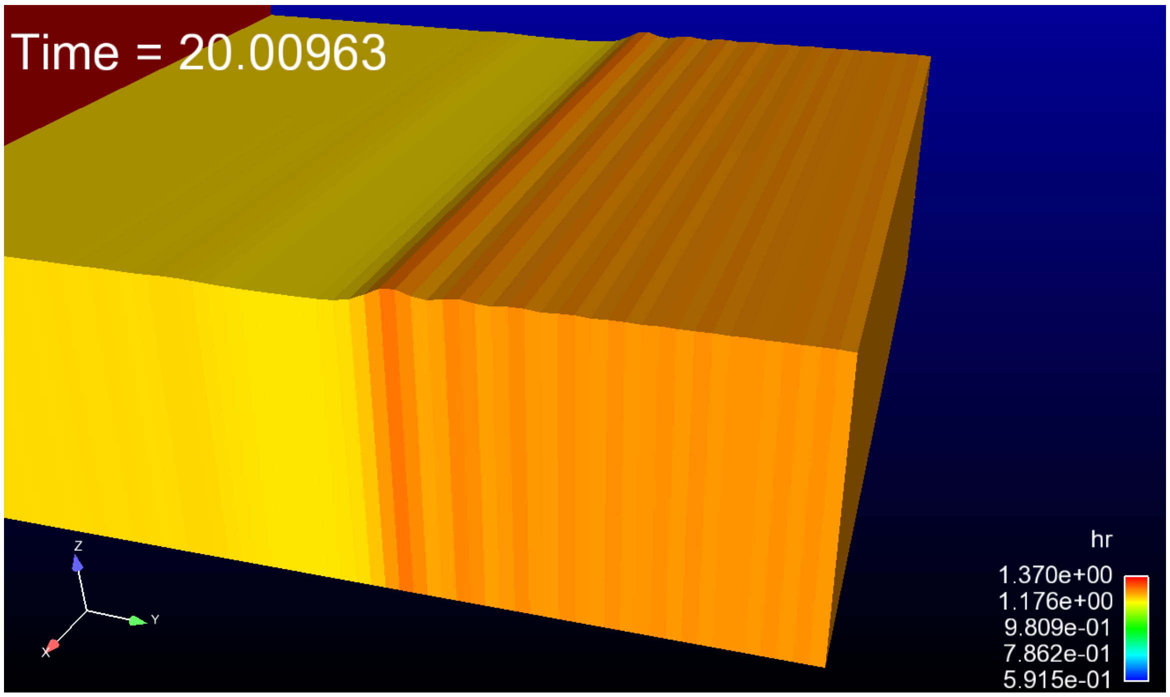
Roxane

In Roxane, we were unable to run a simulation with the Menikoff-Kober model working alongside the tabular equations of state in the sesame database. We have multiple theories as to why these two models are not currently compatible. The first has to do with generating NaNs within the solver, which could be a result of division by zero. However, we feel it is more likely like that specific volume or specific energy, two state variables that are taken directly from the tables in the sesame database are being set to "NaN" within the solver and are then being advected through the rest of the hydrocode.

FLAG

In FLAG, this problem was still present, but was more subtle as we were able to run the Menikoff-Kober model to simulate some porous materials. While all materials were able to initialize in FLAG, for some materials, the initialization pressures could not be adjusted to eliminate pressure waves, of the kind shown in the figure below.

Figure 24: Abnormal Pressure Waves in FLAG



These waves have small amplitudes compared to the overall material density, however they were extremely problematic for our comparisons to experimental data, as they originate from the material interfaces, which is exactly where our data is taken from. While we were able to equalize the pressure discontinuity and eliminate these waves for porous basalt, many other materials exhibited a range of initialization pressures, independent of their specific internal energy, which made eliminating these pressure waves impossible. This represents another instance of endemic problems with the Menikoff-Kober model's compatibility with the tabular equations of state in the database.

Future Work

The future work needed to address the effectiveness of the Menikoff-Kober model centers on both theoretical work and code development. On the theoretical front, we are worried that simplifications in the thermodynamics of the Menikoff-Kober model could result in errors when compared to experimental data. Specifically, the model ignores V dependence in ϕ in parts of its derivation, and it would be worthwhile to include the extra term that comes from not neglecting this dependence to investigate possibly divergent trends in model behavior.

The code development aspect of future work needs to be geared at making both the implementation in both models (FLAG and Roxane) compatible with the tabular equations of state for solid materials in the database. This issue is considerably more pressing than the theoretical work, as without simulations, we have no indication on whether or not the model is useful. Our basalt simulation is a good first proof of concept, but more robust tests are needed to verify the Menikoff-Kober more completely.

Theoretical Problems With Menikoff-Kober Implementation

The Menikoff-Kober model operates in a thermodynamicly consistent manner by adding a potential to the Helmholtz free energy in the following manner:

$$\Psi(V,T,\phi) = \Psi_s(V,T) + B(\phi)$$

where Φ is the full equation for Helmholtz free energy, Φ_s is the free energy equation for the purely solid version of the material, and $B(\phi)$ is a potential associated with the crushing out behavior of a porous material. The paper by Menikoff and Kober is aimed purely at the derivation of this potential function $B(\phi)$, but in hydrocode development, we must be aware of state variable values so that we can advect them through the code. To that end, we use the following thermodynamic relation to find the pressure from the newly defined Helmholtz free energy:

$$\begin{aligned} P &= -\frac{\partial \Psi}{\partial V}|_T \\ P &= -\frac{\partial}{\partial V}[\Psi_s(V,T) + B(\phi)]|_T \end{aligned}$$

Both Φ_s and $B(\phi)$ are functions of V, so we must not ignore any term in this differential. Turning the crank, we obtain:

$$P = -\frac{\partial \Psi_s}{\partial V_s}\frac{\partial V_s}{\partial V} - \frac{\partial B}{\partial \phi}\frac{\partial \phi}{\partial V}$$

From our definition of the bulk porosity as $\phi = \frac{V_s}{V}$, and remembering our thermodynamic relation for pressure, we obtain the following three relations:

$$\begin{aligned}
\frac{\partial V_s}{\partial V} &= \phi \\
\frac{\partial \phi}{\partial V} &= \frac{-V_s}{V^2} \\
\frac{\partial \Psi_s}{\partial V_s} &= -P_s
\end{aligned}$$

Substituting these relations into the original differential equation for pressure of the porous material, we now have an equation with only one differential term:

$$P = P_s\phi + \frac{\partial B}{\partial \phi}\frac{V_s}{V^2}$$

This gives us a way to determine pressure in a porous material that is in keeping with the thermodynamic consistency of the Menikoff-Kober model. However, the second term in the above equation is not found in the current implementations of FLAG or Roxane, which we believe may be responsible for some of the problems with calculations of shock speed, particle velocity, and the initialization pressure waves discussed above. We think that this term cannot be tacitly assumed to be trivial without possibly compromising the thermodynamic consistency of any hydrocode running a Menikoff-Kober porosity model.

References

[1] Thomas J Ahrens and David M Cole. Shock compression and adiabatic release of lunar fines from apollo 17. In *Lunar and Planetary Science Conference Proceedings*, volume 5, pages 2333–2345, 1974.

[2] Grant Heiken, David Vaniman, and Bevan M French. *Lunar sourcebook: A user's guide to the Moon*. CUP Archive, 1991.

[3] WN Houston, JK Mitchell, and WD Carrier III. Lunar soil density and porosity. In *Lunar and Planetary Science Conference Proceedings*, volume 5, pages 2361–2364, 1974.

[4] Y Langevin and JR Arnold. The evolution of the lunar regolith. *Annual Review of Earth and Planetary Sciences*, 5:449–489, 1977.

[5] Ralph Menikoff and Edward Kober. Equation of state and hugoniot locus. *Shock Compression of Condensed Matter-*, page 2, 1999.

[6] Marc A Meyers. *Dynamic behavior of materials*. John Wiley & Sons, 2007.

[7] Barry O'Connor. Geology rocks & minerals, 2005.

[8] C.S. Plesko et al. Shock compaction and release in jsc lunar and martian regolith simulates, 2011.

[9] Scott Runnels. *A 1-D staggered grid Lagrange hydrocode for Summer Workshop training and research*, 2014.

[10] Scott R. Runnels. *Theory, Implementation, and Use of Roxane (DRAFT)*, 2015.

[11] Karsten Seiferlin, Pascale Ehrenfreund, James Garry, Kurt Gunderson, E Hütter, Günter Kargl, Alessandro Maturilli, and Jonathan Peter Merrison. Simulating martian regolith in the laboratory. *Planetary and Space Science*, 56(15):2009–2025, 2008.

[12] Ya B Zel'dovich and Yu P Raizer. Physics of shock waves and high-temperature hydrodynamic phenomena. *New York: Academic Press, 1966, edited by Hayes, WD; Probstein, Ronald F.*, 1, 1966.

Implementation Of Computational Physics Methods In The Domain-Specific Language Scout

Team Members
Joseph Bottini and Sriram Nagaraj

Mentors
Christine Sweeney and Scott Runnels

Abstract

There is an acute need in recent years to restructure the way scientific codes are written. Traditional scientific codes are written in a linear, assembly-line fashion where the physical models, algorithms, data structures, parallelism, visualization etc. are disparate building blocks. Challenging this approach now is the advent of multiple new architectures requiring code specialization. In view of this, the promise of a *domain-specific language* (DSL) for computational physics applications is particularly appealing. A DSL provides a unified framework in which code development from the physical model to visualization is fully integrated and in which the data structures involved are specific to the domain. Finally, scientists and users need only to focus on the actual modeling and algorithm development while the programming details are handled behind the scenes.

The main aim of the present work is to critically exercise and analyze the capabilities of Scout, a promising computational physics DSL developed at Los Alamos National Laboratory (LANL). In this report, we describe the Scout programming language and our implementation of various computational methods in Scout such as the staggered-grid hydro (SGH) method, the finite element method (FEM), the finite difference (FD) method as well as a linear algebra framework. Our results indicate that Scout provides a robust framework for implementing computational physics algorithms which was verified by close agreement with results from traditional general purpose languages.

keywords – Computational Physics, Domain-Specific Language, Staggered-Grid Hydro, Finite Elements, Linear Algebra, High Performance Computing

Introduction

Many computational physicists, scientists and engineers today face the need to swiftly and easily design *parallel, large-scale* and *architecture-independent* code specific to their respective domain. Unfortunately, their efforts are often plagued by the following issues.

First, many scientists work with large legacy codes that were developed by host institutions over the course of decades. Very often, these codes are written in formats that were not originally designed to be parallelized. The task of recoding a large code base into a more modern format is a daunting one, especially if the code must be parallel.

Second, most scientific code is written in fairly generic languages that are also used in other domains such as sales, management, etc. This means that scientists need to first develop a broad, domain-specific infrastructure before being able to write scientific code. This infrastructure development includes constructing relevant data types, specific methods, and linking with scientific libraries. This "pre-coding" stage can be as taxing, if not more taxing, than coding the actual scientific algorithms.

Third, even if a domain-specific infrastructure were available, it may be architecture-dependent, and changes in machine architecture or hardware updates result in the need to recode or even reengineer the original code.

All of the issues listed above put a tremendous strain on the scientist to catalog and update code rather than spend the time to improve upon the science and physics behind the code. It would thus be of great value if the responsibility of managing the hardware, compiler issues, and parallelism were taken care of independently of the science and algorithm design.

Need for DSLs

In order to address these concerns, effort has been put into developing *inherently parallel, architecture-independent domain-specific languages (DSLs)*. While clearly it is optimal to have the triad of parallelism, architecture-independence, and domain-specificity in the same structure, there still is a large amount of freedom in the actual software design of such DSLs. We now briefly mention some DSLs with a short summary of each:

- Julia: Julia is a high-level, high-performance dynamic programming language for technical computing. Julia has a sophisticated compiler and distributed parallel execution with an extensive mathematical function library [1].

- Liszt: Liszt is a domain-specific language for constructing mesh-based PDE solvers. It is based on a topological classification of mesh elements (vertices, edges, faces, etc.). Liszt exploits the data parallelism in the mesh data type composed of mesh elements and targets three programming models: MPI, CUDA and pthreads [5].

- Halide: Halide focuses on the optimization of the image processing pipeline. It is applicable across different hardware architectures including multicores (with SIMD) and heterogeneous CPU+GPU execution [10].

While one may argue that instead of a DSL, one can simply take a general purpose language and combine it with an application specific API that provides a DSL-like functionality, there are a number of advantages of a dedicated DSL for computational physics.

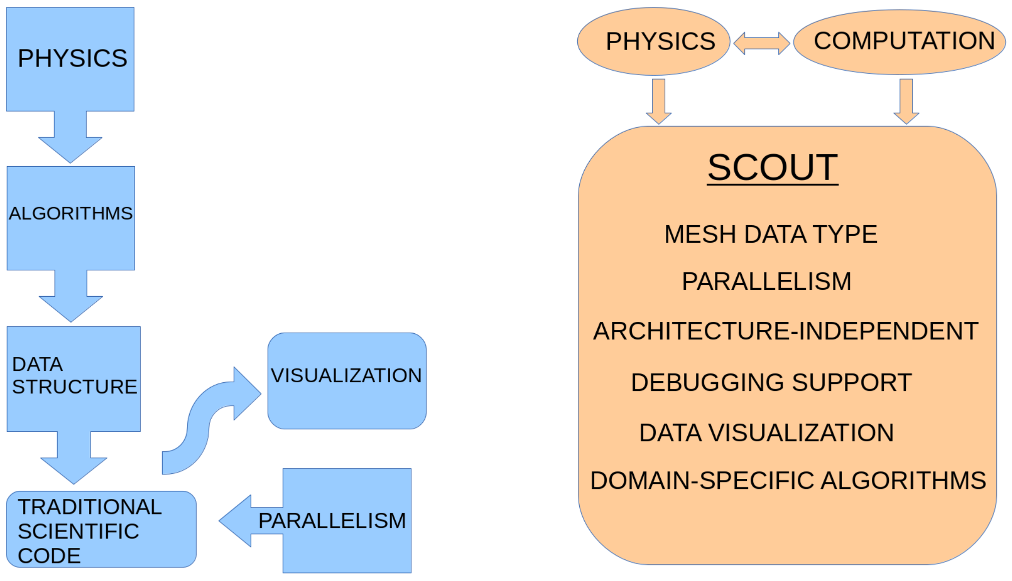


Figure 1: Comparison of traditional vs. Scout based scientific code generation

First, such a DSL offers appropriate notation specific to the computational physics domain. Second, domain-specific abstractions can be optimized by the compiler in some cases. A compiler for a general-purpose language would not recognize domain-specific opportunities for optimization. A DSL compiler can incorporate programming model runtime systems for parallelism and other runtime libraries to support capabilities such as visualization and domain-specific functionality. Finally, a DSL compiler can collect domain-specific information that makes it possible to debug domain-specific data structures such as the mesh.

The Scout DSL

Our original aim in the Computational Physics Student Summer Workshop was to explore the use of the DSL *Scout*. Scout is an architecture-independent, parallel, mesh-based DSL made for computational physics and science applications.

Scout is not a stand-alone DSL but rather a conservative extension to the C language. It is mesh-based in the sense that meshes are the fundamental data structures used in the language. Scout strikes a balance between existing implementation infrastructures and a stand-alone language and/or a general programming language. Scout is built on the Clang/LLVM compiler infrastructure and uses the Legion programming model to incorporate parallelism to allow for both data and task parallelism. We will delve deeper into the specific details of Scout constructs and working in the following section.

Specific Goals of the Project

Our main goals were to exercise the use of Scout as a computational physics DSL by implementing a variety of computational methods in it. These methods include the staggered-grid hydro (SGH) method, the finite element method (FEM) and the finite difference (FD) method for the Poisson problem. These methods are algorithmically distinct and thus offer insight into the current capabilities as well as limitations of Scout from different perspectives. Midway through the project, we also found the need to develop a Scout-based linear algebra solver which we implemented on a few examples.

Another key goal of the summer project was to identify and address the implementation issues with the current version of Scout. While doing so, we found it necessary to regularly report the technical difficulties we experienced to the development team. The outcome of these interactions was very productive; not only did we help the development team understand the needs and expectations of active Scout users from a computational science perspective, we were also able submit timely bug reports and instances of compiler crashes that aided the overall resolution of deeper problems in the fundamental constructs within Scout. Furthermore, we were able to provide an active list of future considerations that would enhance the usability of Scout.

Paper Organization

The rest of this report is organized as follows. After this preliminary introduction, we discuss the specifics of Scout in detail in the second section. We then discuss our in-Scout implementation of various computational methods in the subsequent sections. In the following two sections, we provide an introduction to the staggered-grid hydro method (SGH) and go into the details of the Scout implementation of SGH in two spatial dimensions along with a remapping procedure. Next, we provide a self-contained summary of the finite element method (FEM) and its implementation in Scout for the one-dimensional Poisson problem. We then discuss the need for a Scout-based, on-mesh linear algebra solver and provide details of our Jacobi iterative solver for the one- and two-dimensional Poisson problem discretized using finite differences. Finally, we provide a brief summary of our work as well as current limitations of Scout and future extensions to Scout which would render it a more versatile computational physics DSL and close with references.

Scout Overview and Functionality

In this section, we will provide a broad overview of the DSL Scout. We will focus on the fundamental constructs of Scout and explain the mechanisms that allow for Scout's parallelism and architecture-independence. We provide a systematic overview of the basic features of Scout and, where appropriate, code segments with explanations which allow one to understand how the constructs are used in practice.

Salient Features of Scout

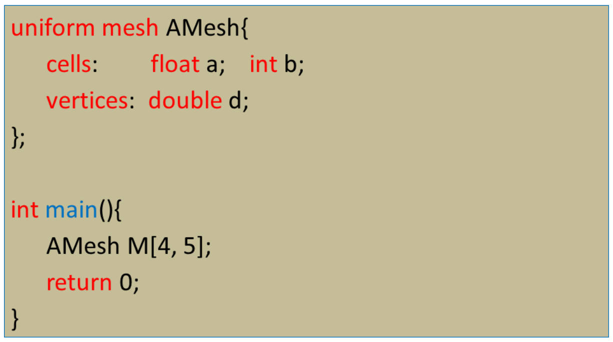
Scout is an open source [9], C-based language and consists of conservative extensions to C, including additional fundamental data types and language constructs. Scout consists of the following fundamental aspects:

- Mesh Data Types
- Parallelism (Data and Task)

- Data Visualization and Plotting
- Supports Domain-Specific Expression of Algorithms
- Architecture-Independence
- Debugging Support

Mesh Data Types

As was previously mentioned, Scout uses meshes as the fundamental data type in computations. Meshes are defined by topological entities (vertices, edges, faces, and cells) that hold independent data called *fields*. The mesh can be defined by any subset of these entities; thus, a mesh can contain, for example, cell fields and face fields while another mesh can contain vertex fields and edge fields. Algorithms can be written that access and/or modify the data contained in the fields through loops that are inherently parallel. By default, one specifies the size and dimension of a mesh in terms of the number of cells in each dimension. Figure 2 shows an example of how to declare a mesh and corresponding mesh properties.



```
uniform mesh AMesh{
    cells:      float a;    int b;
    vertices:  double d;
};

int main(){
    AMesh M[4, 5];
    return 0;
}
```

Figure 2: Code for Mesh and Field Declaration

The type `AMesh` is a uniform mesh which allows fields to be stored on cells and vertices. Currently, the types of the fields can only be numeric data types; Scout does not allow C-style pointers to avoid overly complicated data layout decisions and to promote parallelism [9]. The mesh `M` is a two-dimensional mesh of type `AMesh` with 4 cells in the *x*-direction and 5 cells in the *y*-direction. Scout currently supports uniform meshes from one- to three-dimensions.

Parallelism (Data and Task)

Scout provides for both data and task parallelism. Data parallelism is facilitated by means of the `forall` loop construct which processes a set of locations (cells, vertices, etc.) independent of one another. The programmer cannot make any assumptions regarding the order of execution across the different locations, e.g. a cell that "comes later" in the mesh may be accessed first. An example of a `forall` construct is shown in Figure 3 using the mesh `AMesh` defined in Figure 2.

```
int main() {
    AMesh M[5, 5];
    forall cells c in M {
        c.a = 5.;
        c.b = positionx();
    }
    return 0;
}
```

Figure 3: Forall Loop on a Mesh

Both cell fields are initialized for every cell in the mesh `M`. The cell `c` is referred to as the ‘active cell’. The `forall` loop is contructed to run on each cell independetly so each initialization can be run in parallel with one another. Within this code segment, the built-in function `positionx()` is also called which returns the global index of the x-position of the active cell. In this case, since there are five cells in the x-direction on the mesh M, the function will return 0-4 depending on which cell `c` refers to.

Within a `forall` loop, it is often necessary to access fields of neighboring topological entities (cells, vertices, etc.). Scout facilitates this access through a number of methods. Figure 4 shows a few of these means of accessing fields from surrounding entities.

```
int main() {
    AMesh M[5, 5];
    forall vertices v in M {
        double sum = 0.;
        forall cells c in v {
            sum += c.a * c.b;
        }
        sum += cshift(v.d, 0, 1)
        v.d = sum;
    }
    return 0;
}
```

Figure 4: Field Access Within Forall Construct

The nested `forall` loop is used to access fields from dissimilar topological entities, e.g. cells from a vertex or edges from a cell. The nested `forall` accesses all the relevant cells, vertices, etc. for the called entity. In the example, all cells on the vertex are called, and since the mesh is two-dimensional, the nested `forall` loop accesses the surrounding four cells corresponding to a particular vertex. If it is desired to access a field of a neighboring vertex from a cell, the built-in function `vfield()` can be called. The function takes a field and a number of indices to indicate what from direction to access the vertex field. The syntax is similar to the built-in function `cshift()`.

The `cshift()` function is used to access fields from similar topological entities, e.g. cells from a cell or vertices from a vertex. The `cshift()` function takes a few parameters: the field to be accessed and a number of indices, equal in number to the dimension of the mesh being operated on. In the example, the `cshift()` call accesses the 'd' field from the vertex one index to the north of the considered vertex. No movement is taken in the x-direction since the second parameter passed, the x-index, is 0.

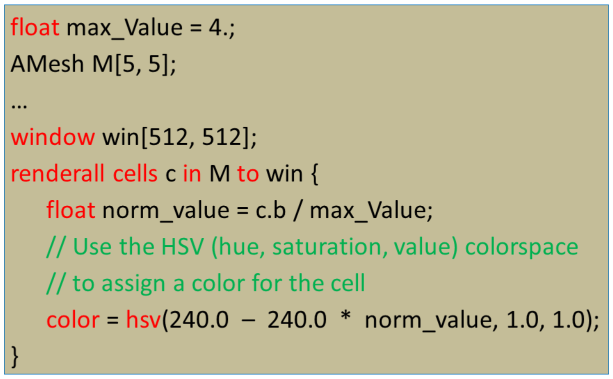
In order to facilitate parallelism within these `forall` constructs, some restrictions must be made. First, within any single forall loop, the mesh fields must be either read-only or write-only, but not both. If they were both readable and writeable, and neighboring fields were accessed, the end result would depend on the order of execution. Second, the constructs for accessing neighboring fields (nested `forall` loops, `cshift()`, `vfield()`) are always read-only.

We now address task parallelism. Tasks in Scout are represented by C-style function declarations prefixed by the keyword `task`. Scout demands `task` functions be side-effect free in that they must return the same result when provided with the same inputs, cannot modify or depend on hidden and/or global state, cannot contain statically declared variables, and cannot access input/output devices/streams [9]. Also, any C-language types used as parameters to tasks must be passed by value. With these constraints, Scout allows for task parallelism.

In addition, the Scout compiler runtime library uses the Legion programming model to create portable parallel and distributed programs. Legion is an open-source, task-based data-centric programming model and runtime system that has a variety of advantages that make it appropriate for use within Scout. It uses data dependencies to schedule tasks asynchronously and minimize data copying [9].

Data Visualization

Data visualization in Scout is done using a structure called the `renderall` construct. The `renderall` construct allows for an in-situ visualization of mesh fields which works in a way that is analagous to the `forall` construct. Also, the `renderall` construct requires a value (the "color" value) which represents a four-component floating point value (three for the RGB red, green, blue color space and the fourth representing the opacity/transparency value). Figure 5 shows how to visualize the data initialized in the previous code segment.



```
float max_Value = 4.;
AMesh M[5, 5];
...
window win[512, 512];
renderall cells c in M to win {
    float norm_value = c.b / max_Value;
    // Use the HSV (hue, saturation, value) colorspace
    // to assign a color for the cell
    color = hsv(240.0 – 240.0 * norm_value, 1.0, 1.0);
}
```

Figure 5: Code for Visualization in Scout

To render the data to an image, first the image is declared: `win`, which is 512×512 pixels. The `norm_value` is the normalized value, representing what fraction of the maximum value the data represents. In the case of the code segment, the field `b` will vary from $0-4$, so the maximum value is 4. To set the color, the keyword `color` is used and is initialized using the built-in function `hsv()`. The first argument will vary between 0 and 240, setting the color between red and blue, respectively. In this way, the data can be displayed as the code is run so that an accurate visualization can be achieved.

Domain Specific Algorithms

The fundamental mesh data types in Scout allow for domain specific algorithms. In particular, one can cast computational methods such as finite differences in the language of meshes. A simple 1D problem of the form $u' = \lambda u$ in the interval $[0,1]$ provides a good example. Here, one would define a uniform 1D mesh that contains N cells. Each cell would contain a value corresponding to the function value at that particular cell location. If one uses a forward Euler scheme, one can then arrive at a difference equation of the form $u_{n+1} = (1+h\lambda)u_n$ where h is the step size. One then applies this scheme on the mesh by updating each cell field according to the difference equation until convergence. Thus, Scout allows for a domain-specific way of writing computational physics algorithms

In addition, Scout has the capability to support built-in functions specific to meshes and computational physics. For example, the remapping of a mesh can be implemented as a built-in function so that the user need not to implement it.

Architecture-Independence

By building on the LLVM compiler infrastructure, Scout achieves architecture-independence since the LLVM-IR is targeted towards several architectures. In addition, Scout's use of the run-time system Legion allows it to be more architecture-independent as well since Legion tasks and memory hierarchy can be mapped onto different architectures in optimal ways.

Debugging Support

Scout is built on top of the Clang and LLVM components, so Scout is able to leverage the LLDB infrastructure to implement a robust debugger. LLDB is a debugger similar to GDB but for Clang/LLVM. Scout extensions are implemented with metadata that is used later for debugging. This metadata enables a domain-aware debugger. LLDB uses Clang as a library, so Scout constructs are available there. For example, `forall` constructs and `renderall` constructs can be used in the debugger.

SGH with Remap Overview

The staggered-grid hydro (SGH) method is a computational physics algorithm commonly used at LANL. It is most frequently used to model high velocity material deformation, but it can be applied to a variety of problems. The staggered-grid hydro method gets its name from the nature of the algorithm. Fields are stored on both cells and vertices; however, each vertex also has an associated staggered cell which is centered at the respective vertex and is bound by the centers of the surrounding cells. Figure 6 shows an example of a staggered cell in blue.

The staggered cell is used in each time step and is a fundamental aspect of the algorithm. The ‘hydro’ in ‘staggered-grid hydro’ refers to the high velocity material deformation problems for which SGH was made. At high strain rates such as in hypervelocity deformation, solid materials behave more like liquids than solids, so historically ‘hydro’ was associated with these codes [7].

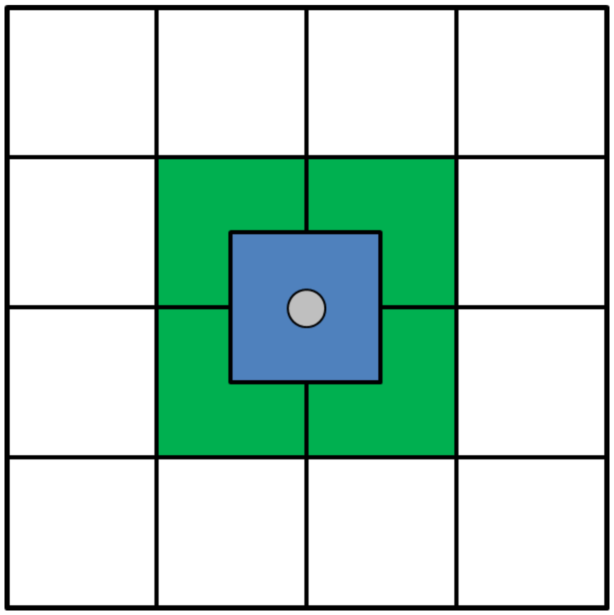


Figure 6: Staggered Cell on a 2D Mesh

One of the problems associated with staggered-grid hydro when run in pure Lagrange mode is that of mesh tangling. As the staggered-grid method progesses, vertices can move in such a way that causes cells to overlap. If this occurs, physical parameters can become irregular and inaccurate, and as the method continues the problem only gets worse [8]. One way to address this issue is to remap parameters from the deformed mesh to a more regular mesh, thus spacing the vertices more evenly and avoiding the problem of overlapping and tangling. The remap process can be computationally expensive; however, it avoids the issue of mesh tangling. Commonly, the remap algorithm is only implemented periodically or just to highly deformed sections of the mesh.

Reasons for Choosing SGH with Remap

The staggered-grid hydro method was chosen to be implemented in the DSL Scout for a few reasons. First, SGH can be easily implemented in parallel; the update for individual cells does not depend on the update of other cells, and likewise for vertices. Second, fields are stored on both cells and vertices, which Scout supports. However, it is necessary to test the communication between these two structures. The update for cell properties depends on the surrounding vertices, and the update for vertex properties depends on the surrounding cells. Since Scout easily facilitates this access, it is a good method to demonstrate how the DSL can simplify implementations. Lastly, in running the SGH algorithm, mesh tangling should be avoided, so a remap algorithm was implemented as well. While the Lagrange implementation tests the communication primarily from cell to vertex or vertex to cell, the remap algorithm tests communication primarily from cell to cell, or vertex to vertex. The promise of parallelism and the storing of fields on multiple mesh types were the underlying motivations for choosing the non-trivial SGH with remap implementation.

Physics of Staggered-Grid Algorithm

As mentioned earlier, fields are stored on both cells and vertices in the SGH method. The cells hold the physical fields of density, pressure, and specific internal energy, while the vertices hold the kinematic fields of position and velocity. The SGH method, like many other computational physics algorithms, starts with conservation equations for mass, momentum, and energy. These are all viewed here from the Lagrangian perspective.

The conservation equation for mass is given by,

$$\frac{D}{Dt}\int_V \rho dV = 0 \tag{1}$$

where ρ is the density of the material and $\frac{D}{Dt}$ is the material time derivative. This condition is upheld by asserting that the mass in each cell is held constant with each Lagrange time step. Therefore, the time derivative is inherently zero and the condition is satisfied.

Next, the conservation equation for momentum is given by,

$$\frac{D}{Dt}\int_V \rho \mathbf{u} dV = \int_S p\hat{\mathbf{n}} dS \tag{2}$$

where ρ is the density of the material, $\mathbf{u}$ is the velocity, p is the pressure acting on the surface, and $\hat{\mathbf{n}}$ is the unit normal vector from the external pressure. This condition is applied to each staggered cell since velocity fields are associated with the vertices. The net force due to surface pressure is calculated on each staggered cell, and the net force induces a change in momentum according to the above equation. The time derivative is handled using small time steps and using a forward Euler method.

Lastly, the conservation equation for energy is given by,

$$\frac{D}{Dt}\int_V \rho(\frac{1}{2}\mathbf{u}\cdot\mathbf{u}+e)dV = \int_S p(\mathbf{u}\cdot\hat{\mathbf{n}})dS \tag{3}$$

where ρ is the density of the material, $\mathbf{u}$ is the velocity, e is the specific internal energy, p is the external pressure, and $\hat{\mathbf{n}}$ is the unit normal vector from the pressure. In compatible hydro codes, the energy equation is applied to the internal energy, and the kinetic energy is handled implictly by the update to vertex velocities [8]. The equation is applied to each cell and considers the work done on/by the surrounding staggered cells to update the specific internal energy on each cell.

The SGH method is carried out in discrete time steps. With each time step, a sequence of calculations is performed. First, to avoid unwanted oscillations, the artificial viscosity is calculated for each cell which contributes to the cell pressure. The artificial viscosity is calculated from spatial gradients in the velocity fields and causes additional energy transfer from kinetic energy to internal energy. Next, the total force is calculated on the staggered cell associated with each vertex from the surrounding cells, and the change in staggered cell momentum is calculated according to the conservation of momentum equation above. From the change in momentum, the new velocity of the vertices can be calculated, which in turn can be used to calculate the new positions of the vertices.

From this point, all that must be calculated is the new cell quantities. The cell density is calculated according to the conservation of mass; mass must be held constant within each cell. In compatible hydro, the energy is split: internal energy on the cell, and kinetic energy on the vertices. The work done by the cell on the vertices is used to evolve the specific internal energy on the cell. Finally, the cell pressure is updated by applying an equation of state. In the SGH example problems considered, the equation of state is given by,

$$p = \gamma \rho e \tag{4}$$

where p is the pressure, γ is the adiabatic index ($\frac{5}{3}$ for an ideal gas), ρ is the density, and e is the specific internal energy. After all the cell and vertex fields are updated, the procedure for the next time step can be executed, unless a remap of the mesh is executed first.

Physics of Remap Algorithm

The remap algorithm serves to map the fields from a deformed mesh onto a new, uniform mesh. Figure 7 shows the remap of a deformed mesh to a regular mesh. The remap method is implemented by applying the conservation of the same three quantities as in the SGH implementation, namely, mass, momentum, and energy. It must be emphasized that while the conserved quantities are *extensive* properties, the fields associated with the cells and vertices are *intensive* properties. So, the intensive properties must be converted to extensive properties in order to perform the remap.

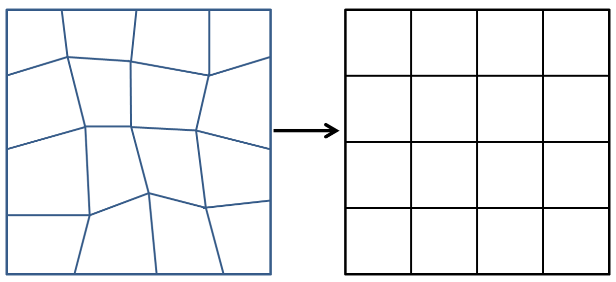


Figure 7: Remap of Deformed Mesh

The extensive properties can be calculated by simply multiplying the corresponding intensive property either by the area or the mass of the cell or staggered cell. After the extensive properties have been calculated, they must be mapped from the deformed cell to the regular cell. This is done by considering the area of overlap between a deformed cell and the regular cell and mapping that fraction of the total deformed area of the extensive properties to the regular cell. Figure 8 shows the remapping process for a particular cell.

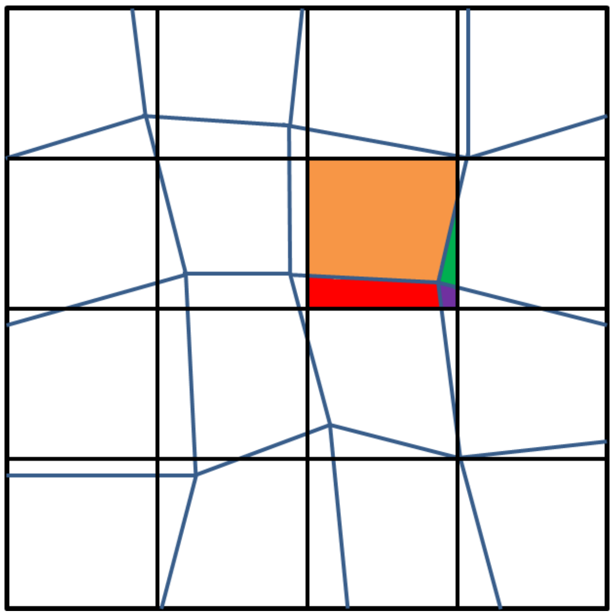


Figure 8: Remap of Deformed Mesh

The mass and energy are mapped from deformed cell to regular cell while the momentum is mapped from deformed staggered cell to regular staggered cell. After the extensive properties are calculated, the intensive properties can be calculated by dividing by the mass or area of the cell. Lastly, the pressure can be recalculated using the same equation of state in the SGH method.

The remap procedure can be executed after every Lagrange step or it can be performed periodically. Similarly, the remap can potentially be applied to only highly deformed sections of the mesh. Unlike the Lagrange method, the remap step has no associated time step with it; the update is purely spatial. The remap can be applied as liberally or conservatively as the user wishes; however, the remap should be applied so that tangling does not occur for the time step and the grid size considered. If tangling can be avoided, the SGH method can be applied for as long as necessary to produce the desired results.

SGH Implementation in Scout

The staggered-grid hydro algorithm was implemented in Scout for two test cases: a 1D piston problem and a 1D Sod tube problem on a 2D mesh. The problems are shown in Figure 9.

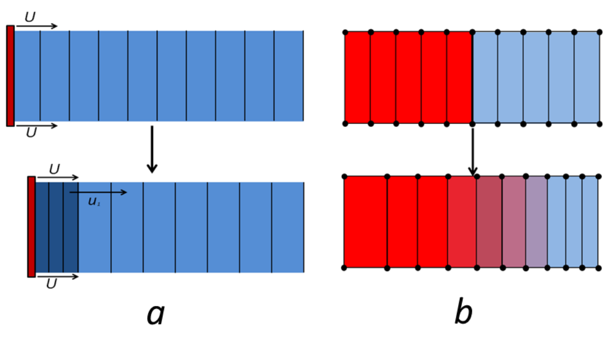


Figure 9: Considered Problems, a) 1D Piston Problem, b) 1D Sod Problem on 2D Mesh

The 1D piston problem was considered simply to become familiar with the SGH with remap algorithm and to become comfortable coding in Scout. The main focus is the Sod problem on the 2D mesh. A traditional implementation was provided for the Sod problem as a reference for the expected results from the Scout implementation. Details on the problem, the implementation and the results for the 2D problem are discussed in this section.

1D Sod Problem on a 2D Mesh

The Sod problem starts with a gas at two different states separated by a thin membrane in a fixed domain. The gas on the left is at a high pressure and high density, whereas the gas on the left is at a low pressure and low density. Dirichlet boundary conditions are applied to the positions at both the x- and y-boundaries that allow for horizontal sliding. At time $t = 0$, the membrane is removed, and the gases at the two states are allowed to interact. A shock wave propagates from the center to the right boundary of the domain, followed by an equilibration to a final state. Since there is symmetry in the vertical direction, it is expected that the fields should only vary in the horizontal direction, making the results only one-dimensional. However, since the mesh is two-dimensional, the cells must interact with the four surrounding vertices, and the vertices must interact with the four surrounding cells. The code in Scout was verified by comparing the generated results to the results of a traditional implementation at the same time step and grid size.

Implementation of Lagrange Step

The Lagrange step algorithm is very similar between the Scout implementation and the traditional implementation. Density, pressure and specific internal energy are stored on the cells, while position and velocity are stored on the vertices. In addition, several other parameters are stored on the cells and vertices to aid with calculations, such as cell artificial viscosity, cell area, force on staggered cells, and staggered cell mass. Figure 10 shows the mesh declaration used in the SGH implementation in Scout.

function

```
uniform mesh sghMesh {
  // Extensive properties have capitals; intensive are all lower case
  cells:
    // Old properties
    double c_x_old;   double c_y_old;   // Cell position field
    double c_p_old;   double c_r_old;   // Cell pressure and density fields
    double c_e_old;   double c_q_old;   // Cell S.I.E. and q fields
    double c_M_old;   double c_E_old;   // Cell mass and energy fields
    double c_A_old;                     // Cell area field

    // New properties
    double c_x_new;   double c_y_new;   // Cell position field
    double c_p_new;   double c_r_new;   // Cell pressure and density fields
    double c_e_new;   double c_q_new;   // Cell S.I.E. and q fields
    double c_M_new;   double c_E_new;   // Cell mass and energy fields
    double c_A_new;                     // Cell area field

  vertices:
    // Old properties
    double v_x_old;   double v_y_old;   // Vertex position field
    double v_u_old;   double v_v_old;   // Vertex velocity fields
    double v_f_x_old; double v_f_y_old; // Vertex force fields
    double v_U_old;   double v_V_old;   // Staggered momentum fields
    double v_M_old;                     // Staggered mass field

    // New properties
    double v_x_new;   double v_y_new;   // Vertex position field
    double v_u_new;   double v_v_new;   // Vertex velocity fields
    double v_f_x_new; double v_f_y_new; // Vertex force fields
    double v_U_new;   double v_V_new;   // Staggered momentum fields
    double v_M_new;                     // Staggered mass field
}; // end sghMesh declaration
```

Figure 10: SGH Mesh Declaration

There exist two of each field for both cells and vertices: an *old* variable and a *new* variable. This is necessitated by the underlying parallelism and the restrictions on fields within the `forall` loops. Since fields cannot be both accessed and modified within the same `forall` construct and since some field updates, such as velocity, depend on previous field values, it is necessary to declare two of each variable. The presence of two fields is also beneficial in the remap method and in the half Lagrange step.

Using the SGH algorithm, a sequence of calculations are performed to update the fields in the following order:

- The artifical viscosity for each cell
- The total force on each staggered cell
- The new velocity for each vertex
- The new position for each vertex
- The new density for each cell
- The new specific internal energy for each cell
- The new pressure for each cell

To improve accuracy, a half time step is performed to calculate a new pressure, then this pressure is applied for the whole Lagrange time step. This is done in both implementations to maintain consistency.

In Scout, the implementation follows a unique pattern. First, two fields are declared for every physical parameter on each cell and vertex, as shown previously in Figure 10. Then, `forall` loops are used for three general purposes: initialization of original parameters, calculation of new fields, and reassignment or update from new fields to old fields. The initialization is straightforward and similar to a traditional implementation. The calculation of new fields is unique to Scout; an example is provided for such a calculation in Figure 11.

```
/* sgh_velocity()
 * Updates the two components of velocity for every vertex on the mesh.
 * Uses v_f_x_new and v_f_y_new to update the velocity components.
 * This method assumes the staggered mass v.v_M_old on the vertices
 * is correct.
 */
task void sgh_velocity(sghMesh *M, double my_dt)
{
// Goal: Initialize v_u_new and v_v_new for all vertices
  forall vertices v in *M
  {
    v.v_u_new = v.v_u_old + (v.v_f_x_new / v.v_M_old) * my_dt;
    v.v_v_new = v.v_v_old + (v.v_f_y_new / v.v_M_old) * my_dt;
  }// end forall vertices loop
}// end sgh_velocity() method
```

Figure 11: Velocity Calculation Method Within Lagrange Step

The velocity on each vertex is calculated by first finding the acceleration, equal to the force divided by the mass, and multiplying it by the time step. The acceleration is assumed to be constant within the time step, so by applying the fundamental kinematic equation, the new velocity can be obtained. By storing two fields on the mesh, the velocity can be accessed through the 'old' variable, and they can be initialized onto the 'new' variable. In Figure 11, the fields `v_u_old` and `v_v_old` are accessed, and the fields `v_u_new` and `v_v_new` are initialized. This does not violate the restrictions placed on the `forall` loops since the fields are unique.

Once all the calculations are performed, it is necessary to reassign the 'new' variables back to the 'old' ones. Figure 12 shows an example of an update method in the Scout implementation.

```
/* updateAll()
 * Initializes old cell and vertex properties from new properties
 * Properties include:
 *   - Cells:    Position, pressure, density, SIE, q, mass, internal energy, area
 *   - Vertices: Position, velocity, force, momentum, mass, area
 */
task void updateAll(sghMesh *M)
{
  forall cells c in *M
  {
    c.c_x_old = c.c_x_new;
    c.c_y_old = c.c_y_new;
    c.c_p_old = c.c_p_new;
    c.c_r_old = c.c_r_new;
    c.c_e_old = c.c_e_new;
    c.c_q_old = c.c_q_new;
    c.c_M_old = c.c_M_new;
    c.c_E_old = c.c_E_new;
    c.c_A_old = c.c_A_new;
  }// end forall cells loop

  forall vertices v in *M
  {
    v.v_x_old   = v.v_x_new;
    v.v_y_old   = v.v_y_new;
    v.v_u_old   = v.v_u_new;
    v.v_v_old   = v.v_v_new;
    v.v_f_x_old = v.v_f_x_new;
    v.v_f_y_old = v.v_f_y_new;
    v.v_U_old   = v.v_U_new;
    v.v_V_old   = v.v_V_new;
    v.v_M_old   = v.v_M_new;
  }// end forall vertices loop
}// end updateAll() method
```

Figure 12: Update on All Mesh Fields

With the update method, the roles are reversed: the fields are accessed through the ‘new’ variables, and the fields are assigned to the ‘old’ variables. This way, the calculated fields can be stored back onto the ‘old’ fields, and the ‘new’ fields can take new values from the calculation for the next time step.

Aside from how the code was written, the traditional implementation and the Scout implementation are very similar. However, there are slight differences between the two implementations in a few aspects of the Lagrange step. First, the calculation of the staggered mass, which is necessary for the velocity update, is different between the two implementations. In the traditional implementation, the mesh is always assumed to be regular, so the staggered mass takes $\frac{1}{4}$ of the mass from each of the surrounding cells. The implementation in Scout calculates the area of intersection between the staggered cell and each regular cell, calculates the fraction of the total cell area that intersection represented, and maps that fraction of the mass to the staggered cell. The Scout implementation is more accurate, but in the case of low mesh deformation, the approximation used in the traditional implementation is sufficiently accurate.

The other difference between the Scout and traditional implementations stems from the calculation of force from each surrounding cell. The traditional code assumes a regular mesh when calculating force, whereas the Scout code does not make such an assumption and calculates the force for any non-tangled, deformed mesh. Again, the Scout implementation is more accurate, but in the case of low mesh deformation, the discrepancy is not large.

Implementation of Remap Step

Two implementations for the remap step were performed. For both, the remap is executed after every full Lagrange step to ensure tangling did not occur. One implementation in the traditional code uses a sampling of points to determine the overlapping area fraction which is then applied to the extensive quantities. The Scout implementation on the other hand applies the Sutherland-Hodgman algorithm [6] to find a sequence of points that defines the intersecting polygon vertices. From the points, the overlapping area is calculated and compared to the deformed cell area to find the overlap fraction. Likewise, this fraction is applied to the extensive quantities to map from the old mesh to the new mesh.

The Sutherland-Hodgman method is exact, while the sampling is approximate. However, the sampling is easily extended to three dimensions. With enough sampling points, they both should produce similar, accurate results. In both implementations, the momentum of the staggered cells is first mapped to the deformed cells, then back to the regular staggered cells. This was done in the Scout implementation due to considerations with the deformed staggered cells. The Sutherland-Hodgman algorithm must be modified to handle cells which are concave polygons, which in turn may produce multiple areas of intersection. Mapping from deformed staggered cells to actual cells reduces the risk of inaccurate intersection areas through the Sutherland-Hodgman method. It was also chosen to maintain consistency between the two codes to produce more consistent results.

Once the extensive properties are mapped from the deformed mesh to the regular mesh, the intensive properties of the regular mesh can be determined. Density, velocity, and specific internal energy are all determined from the extensive properties mass, momentum, and energy, respectively. The cell pressure is then determined using the new density and energy fields and applying the equation of state. The implementation in Scout for the remap step has the same application of the `forall` constructs with calculation and updates using the ‘old’ and ‘new’ fields.

Results of Sod Problem

The results of the two SGH codes for the Sod problem are shown in Figures 13, 14, 15, and 16. The plots shown were generated after 10 time steps with $dt = 0.001$ for a grid size of 100 cells in the x-direction, and 100 cells in the y-direction. However, since the problem is one-dimensional, the fields vary only in the horizontal direction.

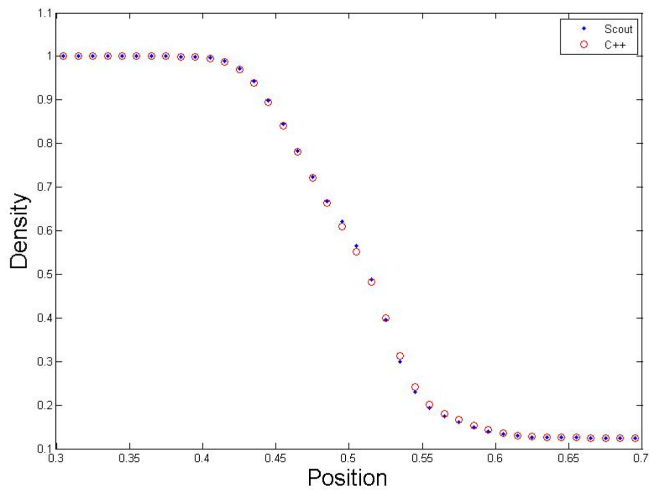


Figure 13: Density Plot for Traditional and Scout codes on Sod Problem

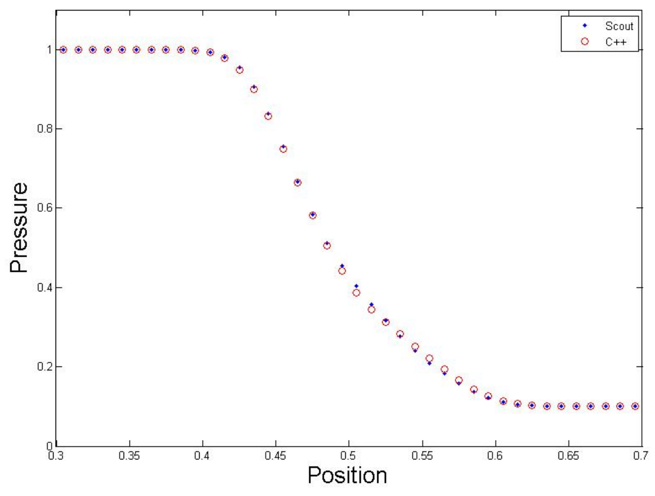


Figure 14: Pressure Plot for Traditional and Scout codes on Sod Problem

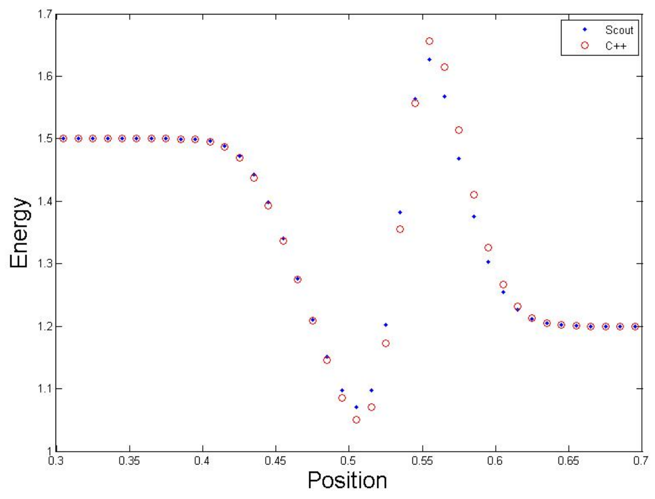


Figure 15: Energy Plot for Traditional and Scout codes on Sod Problem

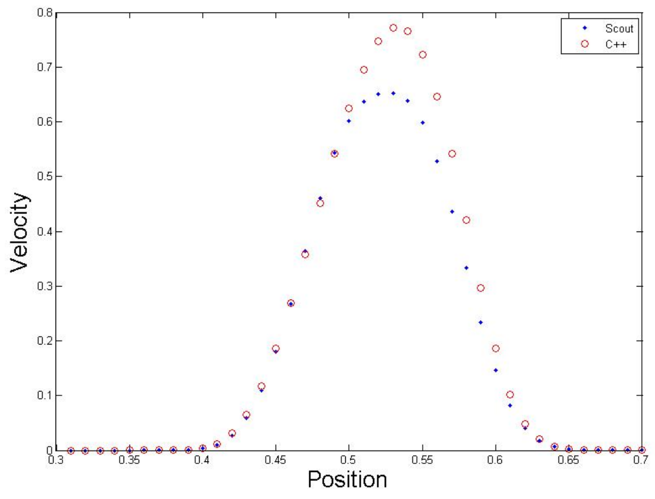


Figure 16: Velocity Plot for Traditional and Scout codes on Sod Problem

The density and pressure plots show the two initial states at the right and left ends of the plot, with a gradual step between them. The energy plot shows a shock wave propogating from left to right with a high energy peak and a low energy peak. The velocity curve is the least physical, but after ten time steps, there is a curve which has a maximum at the wave front.

These results are what we expect from this shock problem. The results for the two implementations are similar, but there are discrepancies which are most clearly seen in the energy and velocity plots. The discrepancy between the results of the two codes can be attributed to the differences in implementation. The different remap methods cause the velocity to differ noticeably between the two codes, and the curvature of the energy curve along with the scale of the plot exacerbate the discrepancy for the energy plot. However, the two codes produce reasonably consistent results. These plots demonstrate that the implementation in Scout is indeed accurate, and that the implementation was ultimately successful. Scout is fully capable of running a 2D SGH code.

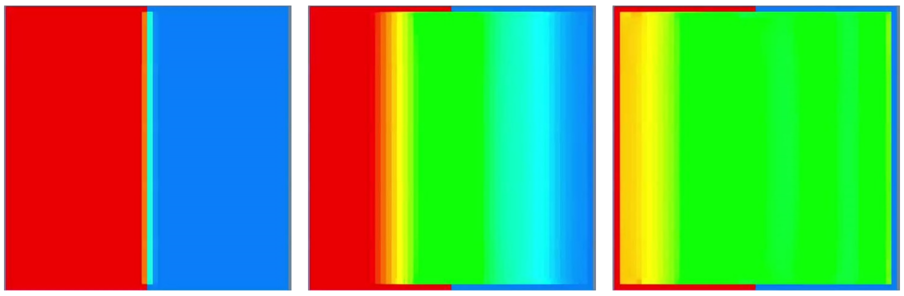


Figure 17: Visualizations for Density after a) 1, b) 50, and c) 100 time steps

In addition to the plots, visualizations were generated in Scout, shown in Figures 17 and 18. Figure 17 shows the density of the cells at a few points in time. Red cells represent a high density, and blue cells represent a low density. The change in density propogates through the domain as time progresses, eventually leading to a homogenization of the state of the gas.

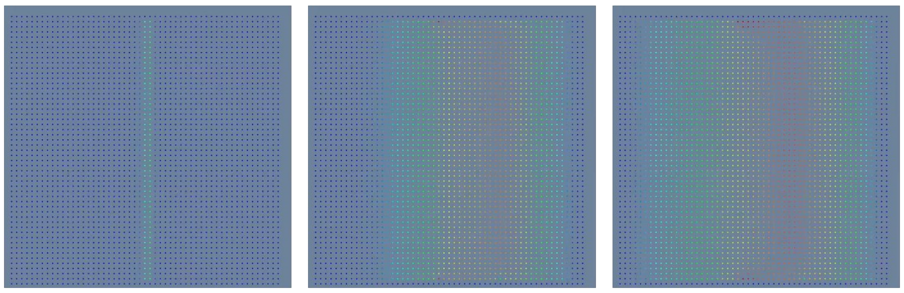


Figure 18: Visualizations for Velocity after a)1, b) 50, and c) 100 time steps

Figure 18 shows the velocity of the vertices, displayed as points in the domain. The red points represent high velocity vertices, and the blue points represent low velocity vertices. The problem begins in a stationary state, so all the vertices are blue. As time progresses, the red velocity peak follows the wave front, and has a distribution similar to that seen in the velocity plot in Figure 16. Between the later two visualizations, it is observed that the area of non-zero velocity extends further into the high pressure region as time progesses, implying a higher degree of mixing.

There are some important features to point out in addition to the trends observed in the visualizations. First, there appears to be multi-dimensional variation in the density and velocity fields, most noticably in the velocity visualizations. The velocity peak is further along toward the middle of the domain than at the centers. This is not expected since the problem is one-dimensional in nature. One possibility for this phenomenon is that the application of the boundary conditions is faulty, or that the half-staggered cells on the boundaries are not being handled properly. However, the general shape and behavior of the visualization is consistent with what is expected. In addition to this, it is important to point out the static cells and vertices on the boundary of the domain. There were problems encountered with the exterior vertices, so ghost cells were implemented around the domain whose cells were initialized, but never updated. Therefore, each visualization has an extra layer of points or cells on each side of the domain to handle the problems encountered with the exterior vertices.

FEM Implementation in Scout

We now describe the implementation of the 1D finite element method in Scout. As a preliminary treatment, we first give a general overview of the finite element method.

Historical Background of FEM

The finite element method (FEM) was historically developed for solving problems arising in civil engineering in the 1950's with the pioneering work of Boris Galerkin. The method was successful for a variety of reasons. First, the accuracy of the method was quite high, especially for linear elliptic problems. Second, the construction of the basis functions used in the method was quite inexpensive and a few polynomials up to degree 2 or 3 were sufficient for the method to deliver very accurate results. Finally, for a large class of problems, the method was very stable to perturbations.

The method was soon found applicable to a wider set of applications including fluid dynamics, electromagnetism and more recently, computer graphics. At the same time, a number of advances on the theoretical side of the method gave rise to adaptive FEM techniques which include the h, p and the hp adaptive FEM methods [4]. Changing the standard FEM formulation to a non-symmetric setting gave rise to Petrov-Galerkin methods. Moreover, the use of discontinuous functions for the approximation scheme resulted in various FEM-like methods including the discontinous Galerkin (DG), hybrid discontinous Galerkin (HDG) and discontinous Petrov-Galerkin (DPG) methods [3], [2], [11].

The increase in numerical methods fathered by the original FEM also created a niche for large, fast, parallel linear solvers since the FEM typically results in a large, sparse matrix that needs to be inverted. Various methodologies that try to use the accuracy of the FEM method coupled with efficient solvers have also been developed, most importantly the multigrid method and domain decomposition method.

The main goal of this part of the summer project was to see how much of the FEM infrastructure could be currently developed in Scout. In particular, we focussed on the 1D Poisson equation, since it is amenable to a simple solution, yet has all the major components that go into a traditional FEM code.

Variational Formulations

We now briefly describe the mathematics behind the FEM using the 1D Poisson equation as an example.

The FEM, unlike the Finite Difference method, works not with the original (or strong form) of a PDE, but rather a different formulation called a *weak* or *variational* formulation. As this is best described by example, we consider the case of the 1D Poisson equation with Dirichlet boundary conditions on the unit interval $[0,1]$:

$$-u'' = f \tag{5}$$

with

$$u(0) = u(1) = 0. \tag{6}$$

We refer to the above equation as the *strong formulation*. We then multiply the equation by a "test" function v that also vanishes on the endpoints and integrate both sides:

$$\int_0^1 -u''v = \int_0^1 fv, \tag{7}$$

and finally perform an integration by parts on the left hand side, which, along with the assumption that v vanishes on the endpoints, results in:

$$\int_0^1 u'v' = \int_0^1 fv. \tag{8}$$

We call the above equation the *variational formulation*. We then look for a differentiable function u vanishing on both ends of $[0,1]$ satisfies the variational formulation for *all* test functions v that also vanish on the both endpoints. It can be shown that not only do we have a unique solution u to this question, but also, the *same* u satisfies the strong formulation. Thus, in order to solve the strong formulation, one can simply compute the solution to the variational formulation.

Polynomial Approximation

In order to obtain a numerical (approximate) solution to the variational formulation, it is customary to approximate the desired solution u by polynomials upto a fixed order p. We denote the space of polynomials of degree up to p as $\mathscr{P}^p$. Since we are interested in a function that vanishes at the endpoints of $[0,1]$, we restrict our approximation to the subspace $\mathscr{P}_0^p$ of polynomials that vanish at the endpoints. We denote a (fixed) basis of $\mathscr{P}_0^p$ as e_i with $i = 1,\dots,N$.

We thus can write the approximate solution $\hat{u} = \sum_{i=i}^N a_i e_i$ where the coefficients a_i uniquely determine $\hat{u}$. We now pick for the test functions v each of the e_i in succession. In order to keep the notation simple, we write

$$\int_0^1 fgdx = (f,g)_{L^2}, \tag{9}$$

since the right hand side of the above equation is the L^2 inner product. Our testing process with the N basis functions e_i yields the linear system $Ax = b$, where

$$A_{ij} = (e_i', e_j')_{L^2} \tag{10}$$

is a square matrix A consisting of pair-wise inner products of the derivatives of the basis functions. The matrix A is traditionally referred to as the *stiffness matrix* of the system, in keeping with the civil engineering origins of the method.

Moreover:

$$b_i = (f, e_i)_{L^2} \tag{11}$$

is a vector containing the inner products of the original right hand side function f of the strong formulation with each of the basis vectors e_i. This vector is referred to as the *load vector*.

Finally, the unknown vector x contains the coefficients we are solving for, namely, the a_i in the expansion $\hat{u} = \sum_{i=i}^N a_i e_i$.

After solving the system for the a_i, we simply reconstruct the approximate solution by taking linear combinations of the a_i with the e_i. This, in essence, is a simple description of the classical FEM on a single element.

Multiple Elements and the Master Element

In practical implementations, we are not interested in only the interval $[0,1]$ but rather a more general interval $[a,b]$. In order to reformulate the classical FEM to general intervals, we simply subdivide $[a,b]$ into N subintervals (“elements”) and perform the usual FEM on each element. We would then get N stiffness matrices and N load vectors which are the *element* stiffness matrices and load vectors respectively. These element matrices and vectors are then patched together into a global stiffness matrix and global load vector which is the final system that is solved.

However, the important point is that integration is done over the “master” element $(0,1)$ and we simply scale the result to the interval $(hj, h(j+1))$ by the map $x \to a+(b-a)x, j = 0,\dots,N$ where $h = \frac{(b-a)}{N}$ is the element width of each sub-interval for a uniform mesh. The assembly of the global stiffness matrix and global load vector is done using a connectivity matrix that encodes information about the intervals and how to patch up functions over entire domain from functions defined only on one element. The connectivity matrix can be thought of as a local-to-global numbering system which tells us where inside the global matrix/vector the contribution of each element matrix/vector resides.

Finally, the global system is solved for the global coefficients and we use the connectivity matrix again to go from the global to local basis functions in order to do the reconstruction.

Scout implementation

We now come to the Scout implementation of the FEM. We restricted ourselves to the $p=2$ case (i.e., quadratic elements) in 1D. We solved the Poisson equation with Dirichlet conditions on the interval $[0,L]$ where L is the length of the interval. In Scout, we implemented the FEM with the following steps:

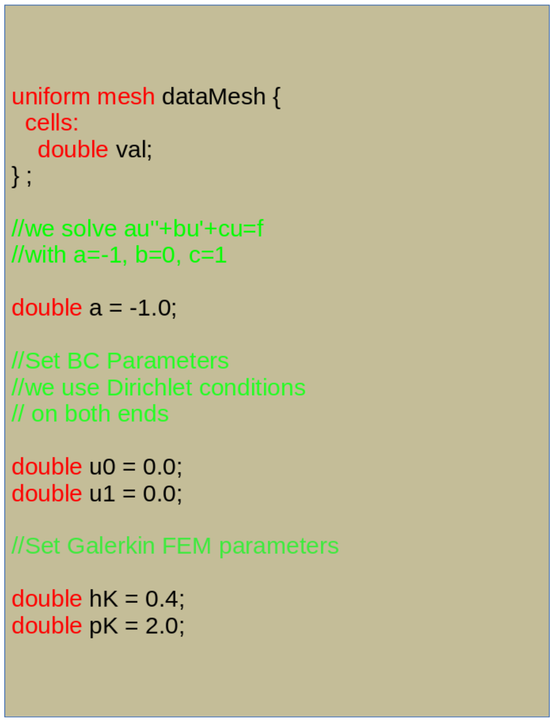
- Initialize $2p+1$ Gauss-Legendre quadrature nodes and weights corresponding to the master element $(0,1)$ on two Scout meshes. We store the quadrature nodes and weights as the cell fields of the corresponding meshes.

- Next, initialize the quadratic shape functions and their derivatives into respective Scout meshes, again using only cell fields.

- Next, compute the element stiffness matrices and element load vectors corresponding to the Poisson problem using the stored values of the quadrature nodes and weights along with the shape function values.

- Since the number of elements is decided beforehand, we pre-compute the connectivity array that gives us the local-to-global element numbering.

- Finally, assemble the global stiffness matrix and load vector and solve the system using the external GSL (GNU Scientific Library) solver after implementing the Dirichlet boundary conditions. We were motivated to use a GSL solver since, as of now, we do not have full linear algebra capabilities within Scout. We will come back to this particular aspect of the solver in the subsequent section.



```
uniform mesh dataMesh {
  cells:
    double val;
} ;

//we solve au''+bu'+cu=f
//with a=-1, b=0, c=1

double a = -1.0;

//Set BC Parameters
//we use Dirichlet conditions
// on both ends

double u0 = 0.0;
double u1 = 0.0;

//Set Galerkin FEM parameters

double hK = 0.4;
double pK = 2.0;
```

Figure 19: Basic FEM mesh declaration

We note that the global stiffness matrix and load vector are stored as double arrays and then converted into GSL objects before we solve for the global coefficients (also called *degrees of freedom*).

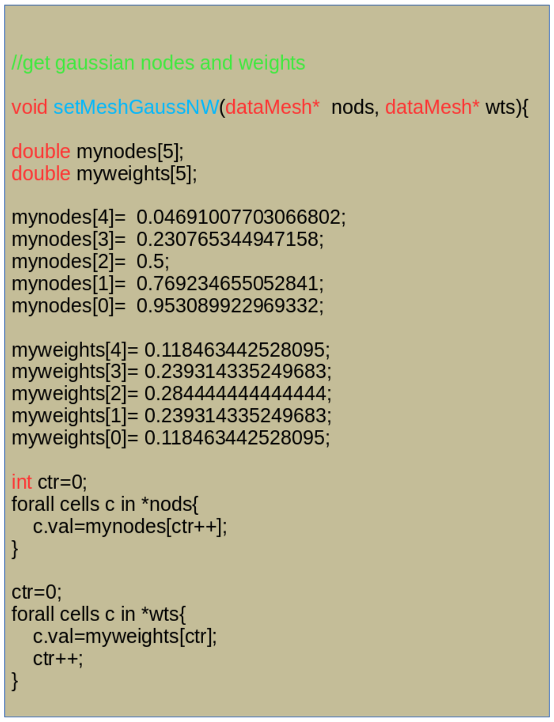
FEM Results

We plot the results of our simulation on the interval $(0,2)$. We first plot the shape (basis) functions used and then the reconstructed solution using 5 elements.

We see that there are 11 basis functions with $p = 2$ quadratic elements and with 5 elements in the interval $[0,2]$. The reconstructed solution obeys the Dirichlet boundary conditions on both ends of the interval.

Linear Algebra Library in Scout

As we saw in the FEM section, a critical step in the solution for the global degrees of freedom was the solution of a linear system $Ax = b$. This is not an isolated situation with the FEM. In fact, at the end of many useful numerical methods, we are left with the task of solving a matrix equation $Ax = b$. Moreover, the matrix A to be inverted is usually structured, and in many cases is sparse, i.e., contains only a few non-zeros. While it would be acceptable for a Scout implementation of a generic method to assemble the system to be solved and send the system to an external linear algebra package (such as LAPACK or GSL) to be solved, it would



```
//get gaussian nodes and weights

void setMeshGaussNW(dataMesh*  nods, dataMesh* wts){

double mynodes[5];
double myweights[5];

mynodes[4]=  0.04691007703066802;
mynodes[3]=  0.230765344947158;
mynodes[2]=  0.5;
mynodes[1]=  0.769234655052841;
mynodes[0]=  0.953089922969332;

myweights[4]= 0.118463442528095;
myweights[3]= 0.239314335249683;
myweights[2]= 0.284444444444444;
myweights[1]= 0.239314335249683;
myweights[0]= 0.118463442528095;

int ctr=0;
forall cells c in *nods{
   c.val=mynodes[ctr++];
}

ctr=0;
forall cells c in *wts{
   c.val=myweights[ctr];
   ctr++;
}
```

Figure 20: Initializing Gaussian nodes and weights on the mesh

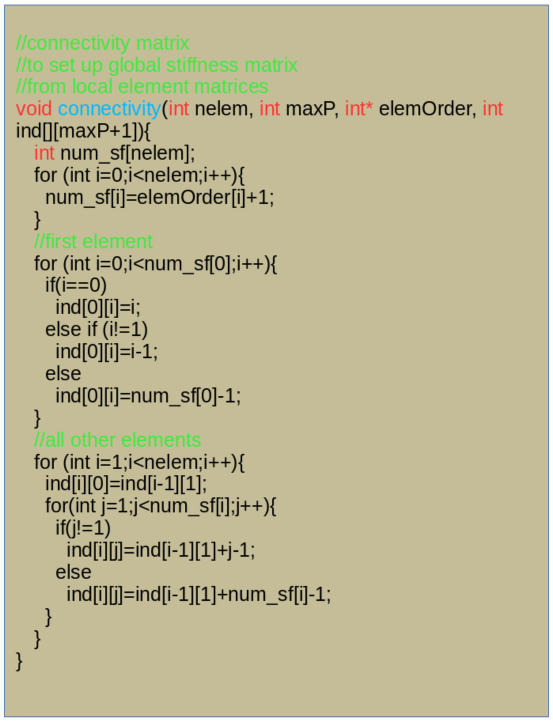
be more efficient and elegant to have an in-Scout *on-mesh* solver.

More specifically, one can consider a sparse $N \times N$ matrix with $k << N$ non-zeros on each row to be stored as a N length 1D mesh with k fields in each cell containing the k non-zeros. We would also store the relationship between the cells, namely, we would encode in each cell the information about the interaction of the corresponding row of the matrix with the unknown elements of the vector. For instance, a row with only three consecutive non-zeros would correspond to a cell that stored the three non-zero values of the row as well as the location in the unknown vector where the three nonzeros would multiply the elements of the vector. We refer to such a distributed mesh that stores a k sparse $N \times N$ matrix as a “solver mesh”.

Once this structure is set up, one can run an iterative procedure *on the solver mesh* directly. Indeed, one can iteratively update the mesh quantities using the matrix entries stored on the mesh until a convergence criteria is satisfied. Thus, Scout would solve the linear system locally on each cell using the sparse row associated with the cell.

This is the basic ideology behind the development of an *on-mesh linear solver* for Scout. A few advantages of such an approach are:

- An in-Scout solver allows for a complete exploitation of the relation between the matrix structure and mesh geometry. In other words, since the mesh geometry heavily influences the matrix structure, having an on-mesh solver allows us to tailor the solver specifically for the given mesh. This could significantly increase the speed of the solver.



```
//connectivity matrix
//to set up global stiffness matrix
//from local element matrices
void connectivity(int nelem, int maxP, int* elemOrder, int
ind[][maxP+1]){
   int num_sf[nelem];
   for (int i=0;i<nelem;i++){
     num_sf[i]=elemOrder[i]+1;
   }
   //first element
   for (int i=0;i<num_sf[0];i++){
     if(i==0)
       ind[0][i]=i;
     else if (i!=1)
       ind[0][i]=i-1;
     else
       ind[0][i]=num_sf[0]-1;
   }
   //all other elements
   for (int i=1;i<nelem;i++){
     ind[i][0]=ind[i-1][1];
     for(int j=1;j<num_sf[i];j++){
       if(j!=1)
         ind[i][j]=ind[i-1][1]+j-1;
       else
         ind[i][j]=ind[i-1][1]+num_sf[i]-1;
     }
   }
}
```

Figure 21: Connectivity array

- By having a Scout based solver, one can reduce dependencies on an external library and thereby have more control over the accuracy of the final solution.
- The use of Scout's parallel constructs will allow for a *parallel, architecture-independent* solver on the mesh and opens up the possibility of problem-specific parallel iterative solvers.
- Most importantly, the scientist would only be involved in specifying the relationship between the cells of the solver mesh, i.e., only how the matrix interacts with the unknown vector. Scout can then take over and complete the linear algebra behind the scenes with implicit matrix-vector products and vector updates. The scientist is fully aware of Scout's functioning but does not get involved directly with the details of the solving process. This saves the scientist the additional task of coming up with a solution procedure for the linear system generated by an algorithm that was coded.

In order to test our ideas, we decided to consider simple 1D and 2D finite difference schemes for the Poisson equation with Dirichlet conditions. The resulting linear system is sparse. In 1D, the computational stencil consists of three nonzeros (three point stencil) while in 2D, we get a five point stencil. In view of the inherent parallelism of Scout, we decided to utilize a Jacobi iteratitive method which yields a parallel solution method. We now describe our in-Scout Jacobi solver.

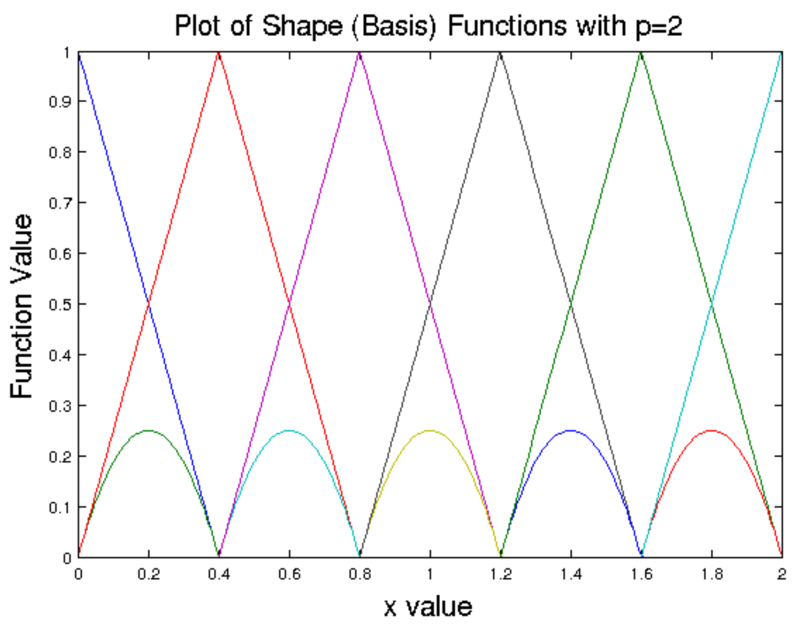


Figure 22: Plot of the basis functions with $p = 2$ quadratic elements

in-Scout Jacobi Solver

We first describe the Jacobi method and then show how we implemented it in Scout for the finite difference scheme.

The Jacobi method is an iterative procedure to solve the matrix equation $Ax = b$ for the unknown vector x. It belongs to a class of iterative methods called stationary methods which convert the equation $Ax = b$ into a fixed point iteration of the form $x = Gx + c$. For a general stationary method, one picks an initial guess x_0 and iteratively applies the mapping $x_{k+1} = Gx_k + c$. The squence of iterates x_k will converge to the unique solution of the original equation $Ax = b$ if the iteration matrix G has spectral radius less than unity, i.e., if the magnitude of the largest eigenvalue of G is less than 1, we can guarantee the convergence of the sequence x_k to the solution of $Ax = b$.

In the case of the Jacobi method, one first splits the matrix A into the form

$$A = D + M + N, \tag{12}$$

where D is a diagonal matrix consisting of the diagonal entries of A, M is the (strict) lower triangular part of A and N the (strict) upper triangular part of A. The expression $Ax = b$ then becomes

$$Dx + Mx + Nx = b, \tag{13}$$

which, assuming that all the diagonal entries of D are non-zero, can be written as

$$x = -D^{-1}(M + N)x + D^{-1}b. \tag{14}$$

If we call $G = -D^{-1}(M + N)$ and $c = D^{-1}b$, then the Jacobi method can be written as the fixed point iteration $x = Gx + c$, and one can apply the map $Gx + c$ to the sequence of iterates starting from an arbitrary guess x_0.

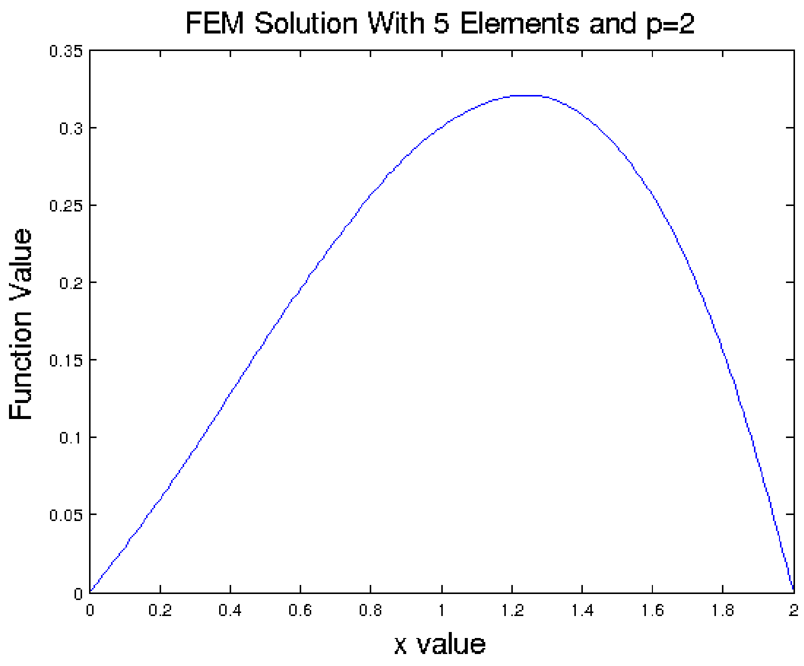


Figure 23: Plot of the reconstructed solution

The question of convergence still exists, i.e., how do we know that the spectral radius of $G < 1$? In order to address this, one usually considers a *preconditioner*, which is a matrix P which multiplies the expression $Ax = b$ on both sides thereby modifying the spectral properties of the original problem to result in a convergent method. In other words, we solve $PAx = Pb$ which, by a careful choice of the matrix P, has spectral properties that ensure convergence. After obtaining the solution $\hat{x}$ of $PAx = Pb$, one then obtains the solution x^* of $Ax = b$ by the map $P^{-1}\hat{x}$. It is assumed that P is designed in such a way that the inversion of P is very cheap, i.e., requires very little work.

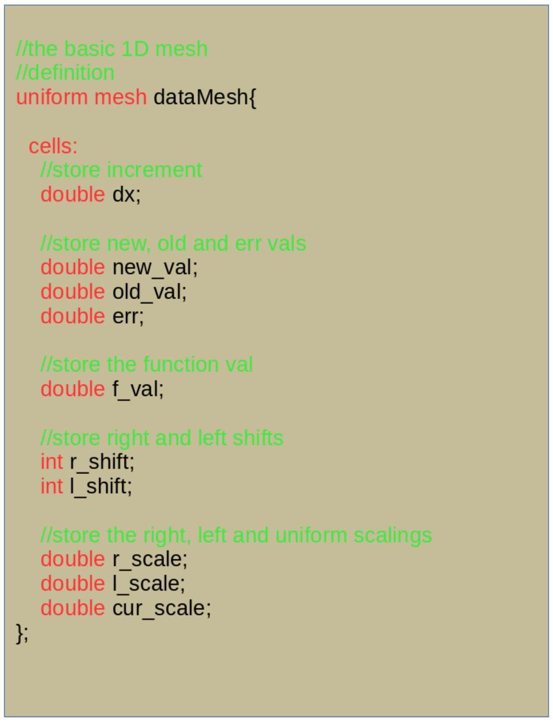
We note in passing that the Jacobi method is inherently parallel, i.e., updating entries between iterations can be done on each component of the current iterate vector independently of the other components.

The Jacobi Method in Scout

We now describe the Jacobi method as implemented in Scout. To keep the problem simple, we considered the 1D and 2D Poisson problem discretized using finite differences. We note that the Scout team has already implemented a finite difference scheme for the heat diffusion problem [9]. However, our approach is different and we emphasize the role of the on-mesh iterative solver.

1D Implementation We first consider the 1D case in the interval $[0,1]$:

$$-u'' = f, \tag{15}$$



```
//the basic 1D mesh
//definition
uniform mesh dataMesh{

  cells:
    //store increment
    double dx;

    //store new, old and err vals
    double new_val;
    double old_val;
    double err;

    //store the function val
    double f_val;

    //store right and left shifts
    int r_shift;
    int l_shift;

    //store the right, left and uniform scalings
    double r_scale;
    double l_scale;
    double cur_scale;
};
```

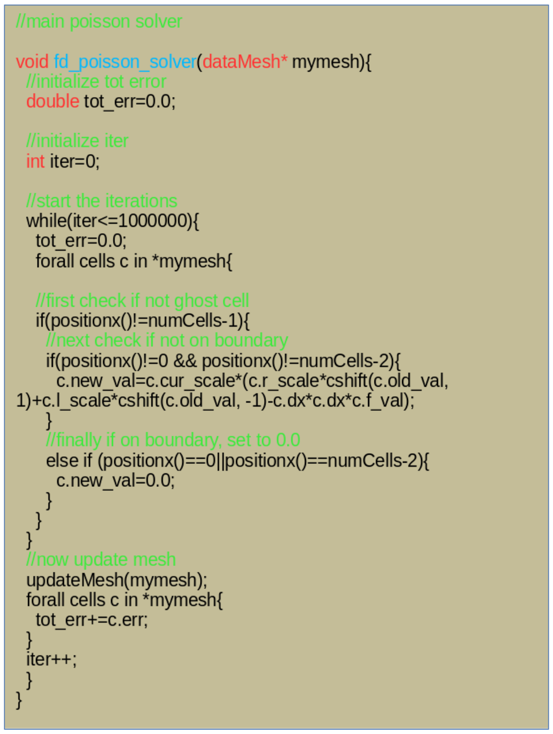
Figure 24: Basic 1D solver mesh declaration

and $u(0) = u(1) = 0$. We first discretize the interval $[0,1]$ into N intervals consisting of $N+1$ nodes: $0 = x_1 < x_2 < \ldots < x_{N+1}$ and $\Delta x = \frac{1}{N}$. The standard finite differencing scheme results in the following iterative procedure:

$$u_i^{n+1} = \frac{u_{i+1}^n + u_{i-1}^n - (\Delta x)^2 f_i}{2}, \tag{16}$$

where the superscripts indicate iteration indices and the subscripts indicate the position along the interval $[0,1]$ where the function evaluation happens. In other words, g_i^n is the n-th iterate of $g(x_i)$ for any function $g(x)$. Note that since $u(0) = u(1) = 0$, we know that $u_1 = u_{N+1} = 0$ and we need to solve for the intermediate values u_i for $i = 2, \ldots, N$. This is actually our Jacobi scheme, namely, we repeatedly apply $\frac{u_{i+1}^n + u_{i-1}^n - (\Delta x)^2 f_i}{2}$ to an initial vector u_0.

In order to implement this in Scout, we first define a mesh of $N+1$ cells with each cell containing an old and new function value. In addition, each cell contains the scaling values $\frac{1}{2}$ and Δx. Finally, we store the shifting indices $+1, -1$ on each cell. The Jacobi iteration then initializes a guess, and uses the *forall* construct to repeatedly update the new function value of cell i, $i = 1, \ldots, N+1$ by accessing the neighboring cell fields specified by the shifting indices $+1, -1$ with $+1$ indicating the next cell and -1 the previous cell. Next, the neighboring cell fields are scaled by the scaling value $\frac{1}{2}$ and added together along with the (non-iterating) value $-(\Delta x)^2 f_i$. Every iteration enforces the boundary condition $u_1 = u_{N+1} = 0$. After the pre-specified convergence tolerance is reached, the iteration ends.



```
//main poisson solver

void fd_poisson_solver(dataMesh* mymesh){
  //initialize tot error
  double tot_err=0.0;

  //initialize iter
  int iter=0;

  //start the iterations
  while(iter<=1000000){
    tot_err=0.0;
    forall cells c in *mymesh{

      //first check if not ghost cell
      if(positionx()!=numCells-1){
        //next check if not on boundary
        if(positionx()!=0 && positionx()!=numCells-2){
          c.new_val=c.cur_scale*(c.r_scale*cshift(c.old_val,
1)+c.l_scale*cshift(c.old_val, -1)-c.dx*c.dx*c.f_val);
        }
        //finally if on boundary, set to 0.0
        else if (positionx()==0||positionx()==numCells-2){
          c.new_val=0.0;
        }
      }
    }
  //now update mesh
  updateMesh(mymesh);
  forall cells c in *mymesh{
    tot_err+=c.err;
  }
  iter++;
  }
}
```

Figure 25: Main 1D Jacobi solver

2D Implementation The 2D implementation is algorithmically identical, except with the obvious changes in the iteration procedure. We again solve

$$-\Delta u = f, \tag{17}$$

on the square $[0,1]\times[0,1]$ with zero Dirichlet boundary conditions on the boundary. We discretize the square into N^2 cells and index the position of the $(i,j)-$th node by subscripts with $i,j = 1,\dots,N+1$. The Jacobi iteration scheme becomes:

$$u_{i,j}^{n+1} = \frac{(u_{i+1,j}^n + u_{i-1,j}^n)(\Delta y)^2 + (u_{i,j+1}^n + u_{i,j-1}^n)(\Delta x)^2 - (\Delta x)^2(\Delta y)^2 f_{i,j}}{2((\Delta x)^2 + (\Delta y)^2)}, \tag{18}$$

where the superscripts again indicate iteration indices and the subscripts indicate the value of the function at the $(i,j)-$th position of the grid.

As in the 1D case, we implemented the Jacobi iteration on a Scout mesh. We first defined a 2D mesh whose cells hold the old and new function value at the $(i,j)-$th node along with the shifting indices $+1,-1$ and scaling values $(\Delta x)^2, (\Delta y)^2$ and $2((\Delta x)^2 + (\Delta y)^2$. We then initialize a guess and run the Jacobi iteration using the *forall* construct using only the stored values on each cell. Once the convergence criteria is reached, we break the iteration.

Discussion of in-Scout Solver

The in-Scout solver we implemented has several advantages. First, we do not explicitly store the iteration matrix. Instead, we store only the non-zeros that contribute to the iteration. Sec-

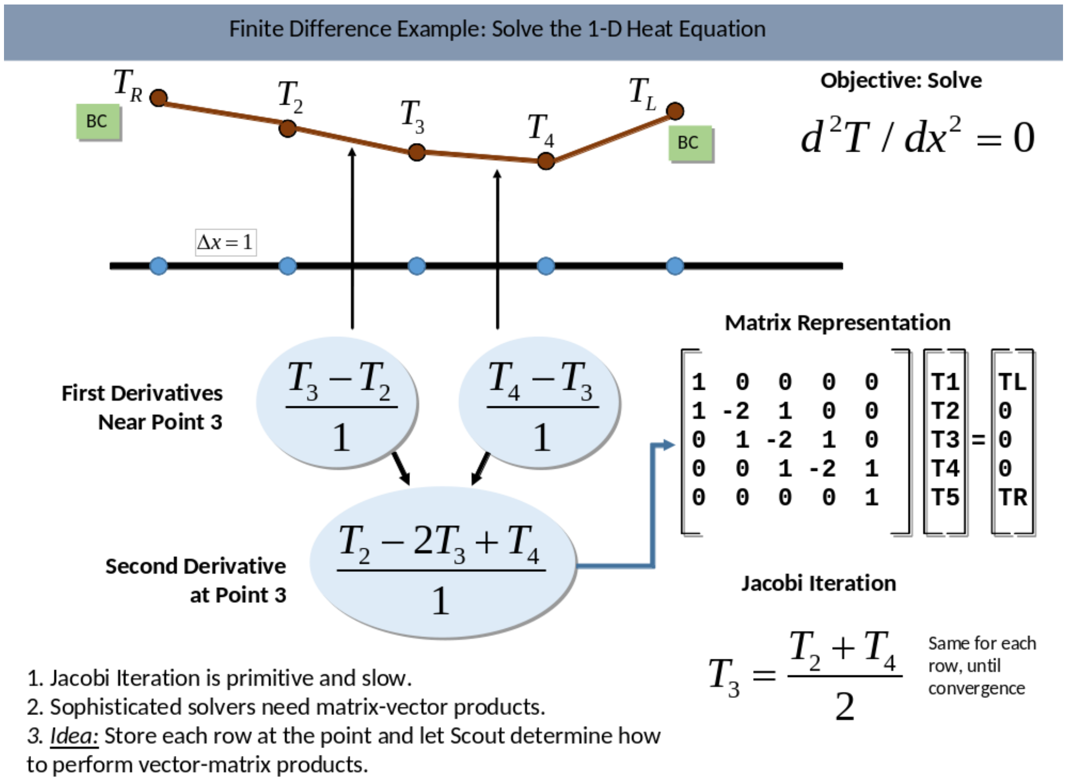

Finite Difference Example: Solve the 1-D Heat Equation
T_R
T_2
T_3
T_4
T_L
BC
BC
Objective: Solve
$d^2T / dx^2 = 0$
$\Delta x = 1$
First Derivatives Near Point 3
$\frac{T_3 - T_2}{1}$
$\frac{T_4 - T_3}{1}$
Second Derivative at Point 3
$\frac{T_2 - 2T_3 + T_4}{1}$
Matrix Representation
$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} T1 \\ T2 \\ T3 \\ T4 \\ T5 \end{bmatrix} = \begin{bmatrix} TL \\ 0 \\ 0 \\ 0 \\ TR \end{bmatrix}$
Jacobi Iteration
$T_3 = \frac{T_2 + T_4}{2}$
Same for each row, until convergence
1. Jacobi Iteration is primitive and slow.
2. Sophisticated solvers need matrix-vector products.
3. Idea: Store each row at the point and let Scout determine how to perform vector-matrix products.

Figure 26: Schematic of the linear solver

ond, we store the connectivity information between nodes using the shifting indices $+1, -1$ which means that we access only the required neighbors for each update and do not access the entire mesh values. Finally, the entire iterative procedure is completely parallel, thereby increasing the convergence speed. This is due both to the fact that the **forall** construct implements cell accessing in a parallel manner as well as the Jacobi scheme itself being a parallel method.

Future of in-Scout Linear Algebra

Our experimentation with the Jacobi scheme has led to a new view of doing linear algebra in Scout. Indeed, most useful computational methods involve solving the matrix equation $Ax = b$ at some point of time. Thus, the scientist using Scout would presumably prefer if the linear solving could be relegated to Scout, which would do the necessary solving in the background. From this point of view, the scientist should have to specify the inter-node relationship along with the non-zero matrix entries contributing to the iteration. Scout would then automatically create the necessary data structures and inter-node communication and run a user-specified scheme and return the solution. While ambitious, this view of Scout would enhance not only its applicability, but also is a natural way of tapping into its already existing data structures and parallelism. Moreover, it is conceivable that such an approach to in-Scout linear algebra would facilitate the use of various computational methods including finite volume, mimetic, hydro and finite element approaches. These ideas are therefore some goals to be explored in the near

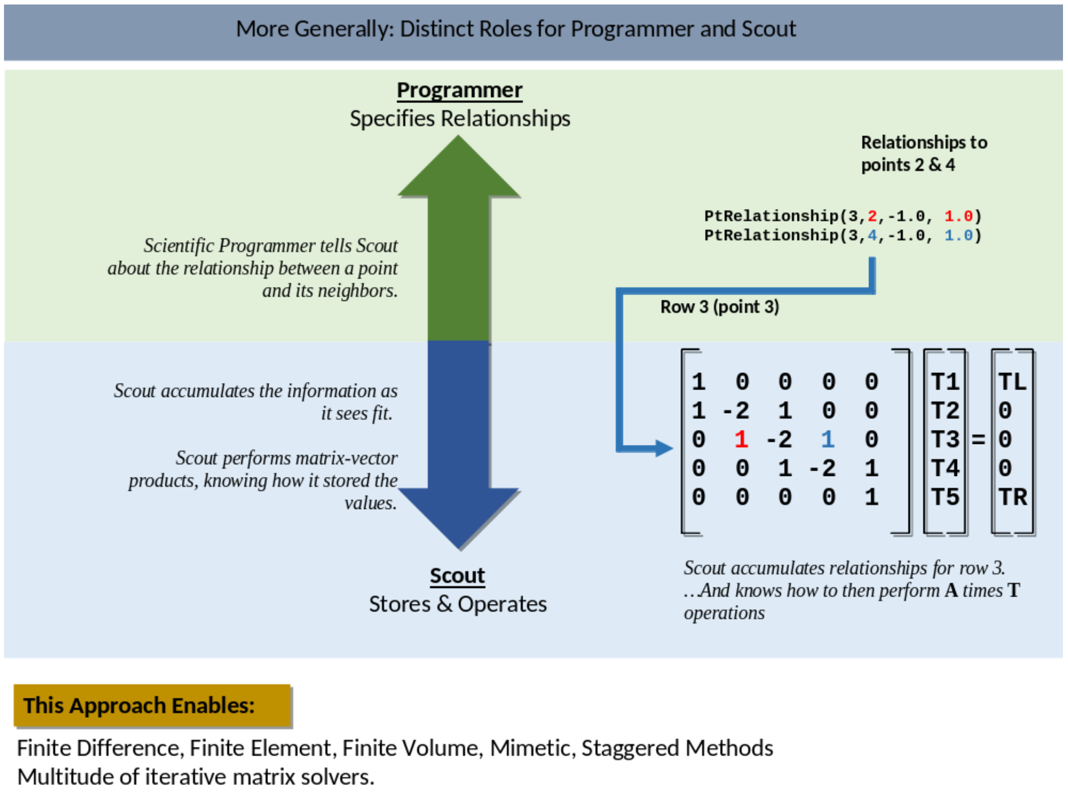

More Generally: Distinct Roles for Programmer and Scout
Programmer
Specifies Relationships
Relationships to points 2 & 4
PtRelationship(3,2,-1.0, 1.0)
PtRelationship(3,4,-1.0, 1.0)
Scientific Programmer tells Scout about the relationship between a point and its neighbors.
Row 3 (point 3)
Scout accumulates the information as it sees fit.
Scout performs matrix-vector products, knowing how it stored the values.
1 0 0 0 0
1 -2 1 0 0
0 1 -2 1 0
0 0 1 -2 1
0 0 0 0 1
T1 T2 T3 T4 T5
=
TL 0 0 0 TR
Scout
Stores & Operates
Scout accumulates relationships for row 3. ...And knows how to then perform A times T operations
This Approach Enables:
Finite Difference, Finite Element, Finite Volume, Mimetic, Staggered Methods
Multitude of iterative matrix solvers.

Figure 27: A big-picture view of how linear algebra can be handled in Scout

future.

Summary and Conclusion

In this summer project, we have demonstrated the feasibility of Scout as a promising computational physics DSL. As we saw, Scout is a parallel, mesh-based DSL which is designed as a conservative extension to the C-programming language. Scout addresses both data and task parallelism and is built on the Clang/LLMV compiler. Scout also uses the run-time Legion programming model to allow for parallelism.

We successfully tested the current capabilities of Scout by implementing a host of distinct computational methods such as the staggered-grid hydro (SGH) method, finite element method (FEM) and finite difference (FD) method. During the course of the project, we also found great promise in considering in-Scout linear algebra capabilities. Towards this end, we were able to develop a simple Jacobi solver that was fully Scout based and where all the linear matrix-vector operations were distributed on the mesh. In addition, we were able to communicate effectively with the Scout compiler team in the CCS division at LANL to help progress Scout's capabilities.

Advantages of Scout-based Code

Our experience with implementing scientific code in Scout has made us aware the many advantages that Scout provides. First, the mesh data type is a very simple yet versatile coding platform. For example, the ability to use different topological entities (vertices, cells etc.) to store and access data made the SGH implementation transparent. Moreover, the translation of algorithms into code was relatively smooth due to the mesh data type. Second, data access between topological entities was easily accomplished using the `cshift()` operator. This, in conjunction with the `forall` construct allowed us to write inherently parallel algorithms.

Challenges Surmounted

Our results indicate that Scout has great applicability in various computational scenarios. However, we also ran into some technical issues within the Scout framework that required us to change our programming style:

- In keeping with the need for parallel algorithms, we needed to re-work the sequential nature of some of the algorithms we implemented. In particular, we adopted the pattern of initializing separate loops for updating fields associated with different topological entities.
- Since Scout is designed to allow for data and task parallelism, explicit indexing of topological entities (vertices, cells, etc.) within a mesh is not readily accomplished. However, in the implementation of the SGH method, our algorithm necessitated the accumulation of forces and work on vertices which required explicit handling of local indices. In order to overcome this issue, we worked in conjunction with the Scout development team to allow for local enumeration of vertices within a cell.
- Another issue we had to overcome was related to circular looping within a cell. Currently, the `cshift()` operator when applied to vertex fields on boundary vertices loops

around the mesh and returns fields corresponding to the vertex fields on the *other boundary* of the mesh. For instance, on a 1D mesh, applying `cshift()` to the rightmost vertex would return fields corresponding to the *first* vertex, i.e., the leftmost vertex. In order to circumvent this issue, we included a layer of ghost cells on the mesh boundary and we would never access or modify the quantities on the ghost cells. Currently, a solution the Scout team is pursuing is to add a "circular" or "regular" qualifier to the mesh definition that would be implicitly understood by the `cshift()` and `forall` constructs and lead to the correct handling of boundary data.

- Finally, during the process of writing Scout programs, we found a number of instances where the Scout compiler would fail. These instances were duly brought to the attention of the Scout development team as bug reports and were resolved.

Future Directions

While developing scientific code in Scout, we found a number of avenues where Scout can be extended. Some of these future directions are quite pressing while others are interesting points worth considering in the long term.

- Currently, all computations with Scout are done on a single mesh. However, there are situations where quantities stored on two different meshes may need to be combined together in non-trivial ways. For instance, computing the correlation between data stored on two meshes would require point-wise multiplication of the two meshes and a global summation of the result. In order to facilitate this, we would require mesh-mesh interactions.
- The Scout team is in the process of developing an ALE mesh that keeps track of vertex coordinates for deformed meshes and has a built-in `remap()` operator. Also, they are exploring the efficiency versus parallelism tradeoff between using the `forall` construct with explicit indexing and a new `vfield()` construct.
- While we considered only one and two spatial dimensions for our code, Scout is designed to handle upto three dimensions. This implies a need for more extensive face and cell capabilities and the ability to modify vertex/edge/face/cell fields independently of each other even if they are geometrically linked.
- A long term goal would be to consider block structured meshes. These meshes offer more flexibility that uniform meshes as well as the ability to handle complicated domain geoemtries. Moreover, in many computational methods, one needs to consider successively finer meshes where only certain cells are refined while other are not. These adaptive mesh refinement capabilities would be an added bonus to Scout's mesh data structure.

Acknowledgements

Funding for the Scout project is by the Office of Advanced Scientific Computing Research, Office of Science, Program Manager, Lucy Nowell. We thank our mentors Dr. Scott Runnels and

Christine Sweeney for their constant encouragement, support, and invaluable help throughout this summer program. We also thank the Scout Team Leader, Pat McCormick, and Scout Team Members, Nick Moss, Dean Prichard, and Christine Sweeney, who were all very responsive to our needs during the code development phase. Finally, we greatly appreciate the opportunity given to us by Los Alamos National Laboratory (LANL) by means of the computational physics summer workshop under whose auspicies this work has been made possible.

References

[1] Jeff Bezanzon, Stefan Karpinski, Viral Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. In *Lang.NEXT*, 2012.

[2] Gopalakrishnan J. Cockburn, B. and R. Lazarov. Unified hybridization of discontinuous galerkin, mixed, and continuous galerkin methods for second order elliptic problems. *SIAM J. Numer. Anal*, 47(2):1058–1064, 2009.

[3] L. Demkowicz and J. Gopalakrishnan. A primal dpg method without a first-order reformulation. *Computers and Mathematics with Applications*, 66(6):1058–1064, 2013.

[4] L.F. Demkowicz. *Computing with Hp-Adaptive Finite Elements, Vol. 1: One and Two Dimensional Elliptic and Maxwell Problems*. Chapman and Hall/CRC, 2006.

[5] Zach DeVito, Niels Joubert, Francisco Palacios, Stephen Oakley, Montserrat Medina, Mike Barrientos, Erich Elsen, Frank Ham, Alex Aiken, Karthik Duraisamy, Eric Darve, Juan Alonso, and Pat Hanrahan. Liszt: a domain specific language for building portable mesh-based PDE solvers. In *Conference on High Performance Computing Networking, Storage and Analysis, SC 2011, Seattle, WA, USA, November 12-18, 2011*, pages 9:1–9:12, 2011.

[6] D. Hearn. *Computer Graphics, C Version (2nd Edition)*. TBS, 1997.

[7] L.G. Margolin and Mikhail Shashkov. Second-order sign-preserving conservative interpolation (remapping) on general grids. *Journal of Computational Physics*, 184(1):266–298, 2003.

[8] L.G. Margolin and Mikhail Shashkov. Remapping, recovery and repair on a staggered grid. *Computer Methods in Applied Mechanics and Engineering*, 193(39–41):4139–4155, 2004.

[9] Patrick McCormick, Christine Sweeney, Nick Moss, Dean Prichard, Samuel K. Gutierrez, Kei Davis, and Jamaludin Mohd-Yusof. Exploring the construction of a domain-aware toolchain for high-performance computing. In *Proceedings of the Fourth International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing*, WOLFHPC '14, pages 1–10, Piscataway, NJ, USA, 2014. IEEE Press.

[10] Jonathan Ragan-Kelley, Connelly Barnes, Andrew Adams, Sylvain Paris, Frédo Durand, and Saman Amarasinghe. Halide: A language and compiler for optimizing parallelism,

locality, and recomputation in image processing pipelines. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*, June 2013.

[11] B. Riviere. *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation.* Society for Industrial and Applied Mathematics, 2014.

Advanced Mesh-Free Methods for Mechanics

Team Members
Benjamin Grossman-Ponemon and Nathan Levick

Mentor
Gary Dilts

Abstract

A novel approach to the mesh-free solution of differential equations based on statistical estimation and regression was proposed and studied. A mesh-free solver was successfully implemented, numerous test cases were carried out, and the properties of the method were investigated. Extensions of the underlying statistical method were researched for the possibility of the recovery of spherical symmetry.

Introduction

At the core of most modern computational methods are meshes. Defined as sets of points or vertices connected by edges, meshes form the basis of classical numerical methods such as finite differences, finite volumes, and finite elements as well as more specialized methods such as staggered-grid hydrodynamics.

There are are traditionally two approaches to mesh-based methods. In the Eulerian approach, the mesh is viewed as a stationary discretization of space through which the material or object of interest moves. Benefits of a stationary mesh include the ease of implementation of adaptive refinement and other techniques to resolve fine details in the simulation, yet, since the objects move through a fixed mesh, material boundaries are not always well-represented, potentially causing boundary diffusion as the system evolves. The alternative approach, the Lagrangian viewpoint, moves the mesh discretization with the material. Here, the mesh conforms to the material as it deforms allowing for accurate boundary representation. However, in the presence of large deformations, there is a risk of mesh tangling, where the mesh changes in such a way that it flattens or even inverts cells. In an attempt to overcome the weaknesses of pure Lagrangian or Eulerian methods, Arbitrary Lagrangian Eulerian (ALE) schemes combine aspects of the two standard views together. For example, in a Lagrangian simulation, it may be advantageous to remap the solution from a highly distorted mesh onto a more uniform one. Consequently, in marrying the two approaches, such schemes may rectify the weaknesses of one pure method at the risk of introducing the other pure method's inherent weaknesses.

In the endeavor to simulate natural physical processes computationally, it is worth noting that meshes are a contrived concept, not naturally arising but used out of convenience. Therefore, by rethinking the discretization process itself, "mesh-free" possibilities emerge. In this way, focusing exclusively on freely moving vertices or particles allows many mesh-based difficulties to simply drop away along with the connecting edges, e.g. tangling, extreme deformations, and adaptive refinement [9]. Of course, even with the promise of mesh-free methods, new consequences arise as the loss of the mesh brings the loss of an intuitive discrete calculus. This requires the development of a calculus on unconnected points. Nonetheless, mesh-free methods present an attractive and effective alternative to meshes.

For the Computational Physics Student Summer Workshop at Los Alamos National Laboratory, the authors investigated the prospects of an advanced method in statistics and data estimation as the foundation of a novel approach to mesh-free. It is here that the authors chose to investigate separate topics: Benjamin Grossman-Ponemon researched the applicability of this method as a mesh-free solver of differential equations and built a proof-of-concept. Meanwhile, Nathan Levick studied an extension to the underlying statistical method which may have implications in recovering symmetry.

Background

A Brief History of Mesh-free Methods

Mesh-free methods originated in 1977, independently developed by Gingold and Monaghan [8] and Lucy [13]. Termed Smoothed Particle Hydrodynamics (SPH) by the former, the method centers around the use of an approximate delta function transform to estimate a function by a weighted sum of known function values. Transforming differential equations in this fashion results in a linear system, in turn solved for unknown function values at data points. After the calculation of respective fields, the particles are then advected. In this way, SPH is a Lagrangian method. The creators of SPH, however, saw their method more as Monte Carlo simulation, i.e. the points that solved the equations were only one instance of an infinite number of trials.

Initially applied to astrophysics problems by its creators and others, SPH found use in a variety of other fields including high-velocity impact problems [11], shock dynamics, and other problems in solid and fluid mechanics such as multi-phase, quasi-incompressible, and porous media flows [10]. In its original form, there exists several issues which have prevented smoothed particle hydrodynamics from becoming as accepted among computational physicists as other traditional mesh-based methods. These include the infamous tension instability [1], the result of particles moving too far apart from each other, and, importantly, a lack of convergence as the smoothing lengths and inter-particle distances are reduced [3].

Since the development of SPH, numerous mesh-free methods have been developed [9]. There are probabilistic tools like molecular dynamics and direct simulation Monte Carlo methods. Methods based on the Galerkin weak formulation such as the element-free Galerkin method, reproducing kernel particle method, h-p cloud method, and finite point method came out of the finite element community. Additionally, subsequent improvements have been made directly to SPH such as replacing its traditional interpolant with moving least-squares (MLS) to create moving least squares particle hydrodynamics (MLSPH) [2]. However, in all these methods, boundary conditions remain an issue, requiring special techniques to incorporate them into the mesh-free framework.

Statistical Methods for Data Estimation

The statistics community has a long history of deriving information from points. Such powerful foundational tools, e.g. data estimation and smoothing, have recently transcended disciplinary lines to applications not only with mesh-free methods but also in machine learning, image analysis, and computer vision. Notably Rosenblatt [14] in 1956 laid the groundwork for the theory of kernel density estimation (KDE), a generalization of the histogram that attempts to best recreate an underlying distribution from a collection of discrete data samples. In the late 1970s, interestingly around the time when SPH was developed, KDE was extended into the local regression estimator (LRE), a technique that estimates functional model correlations between random variables [12]. Further, as demonstrated by Dilts [5], LRE, in a special case, is equivalent MLS and provides a promising framework to a novel mesh-free method.

Local Regression Estimation

To build a successful mesh-free method, it is necessary to have a means of performing calculations, such as derivatives, on points. Estimation of function values and derivatives are ubiquitous in the world of statistics, and it is from here that the local regression estimator is used.

Mathematical Preliminaries

Suppose that there exists a set of points $\{y_j\}_{j=1}^N \subset \mathbb{R}^d$ and at each point y_j there is data $u_j \in \mathbb{R}^m$. $\{y_j\}_{j=1}^N$ is called the set of *data points*. Let $\{x_i\}_{i=1}^M \subset \mathbb{R}^d$, at which the data is to be estimated, be the set of *evaluation points*

Let $w : \mathbb{R}^d \otimes \mathbb{R}^d \to \mathbb{R}$ be a *weight function*. The form of the function is not specified, only that $w(x,y)$ has compact support and $w(x,y) \geq 0$ for all $(x,y) \in \mathbb{R}^d \otimes \mathbb{R}^d$. The purpose of the weight function is to assign weight to the correlation between the data at points x and y. To this end, it is customary to choose the weight function to be a kernel, $w(x,y) = CK(|y-x|/h)$, where C is chosen so that $\int w(x,y)\,\mathrm{d}x = 1$. The dimensions of the support is specified by a smoothing length $h \in \mathbb{R}^d$, which is often labeled as a third parameter of the weight function. While many such kernels exist, the standard in mesh-free methods is based on the cubic B-spline, defined below in one dimension [2]:

$$B_4(x) = \begin{cases} 1 - 1.5|x|^2 + 0.75|x|^3 & |x| \leq 1 \\ 0.25(2-|x|)^3 & 1 < |x| \leq 2 \\ 0 & |x| > 2 \end{cases} \tag{1}$$

For example, a weight function with a rectangular support would be:

$$w(x,y,h) = \nu_d \prod_{\alpha=1}^{d} \frac{1}{h_\alpha} B_4\left(\frac{|y_\alpha - x_\alpha|}{h_\alpha}\right) \tag{2}$$

where ν_d is a normalization constant.

Local Regression

Given the set of data on the data points, the purpose of local regression is to estimate the data on the evaluation points. As the name implies, at each evaluation point, a local Taylor series polynomial approximation of the data is constructed. The Taylor series polynomial is represented as an inner product of the shifted basis, $p(x_i, y_j)$ and the unknown coefficients $\beta(x_i)$. For example, in one dimension:

$$\beta(x_i)p(x_i,y_j) = \beta_0(x_i) + \beta_1(x_i)(y_j - x_i) + \frac{1}{2!}\beta_2(x_i)(y_j - x_i)^2 + \dots \tag{3}$$

As a remark, the shifted basis can be constructed from a *reproducing basis* $b(x)$ and a *jet matrix* $J(x)$. For example, in one dimension, the reproducing basis is

$$b(x) = \left[1, x, x^2/2!, \dots\right]^T \tag{4}$$

while the jet matrix is

$$J(x) = \left[b, \frac{\mathrm{d}b}{\mathrm{d}x}, \frac{\mathrm{d}^2 b}{\mathrm{d}x^2}, \dots\right] \tag{5}$$

With these, the shifted basis is then

$$p(x_i, y_j) = J^{-1}(x_i) b(y_j) \tag{6}$$

The construction of shifted bases is not limited to monomial reproducing bases and Taylor series jets. For example, trigonometric bases may be used [4] and the columns of the jet can be linear operators other than derivatives. However, the jet matrix must be invertible. This will be explained in greater detail later in the report.

The unknown coefficients are found by comparing this polynomial against nearby data, producing the weighted least-squares problem for each evaluation point x_i:

$$\text{minimize (w.r.t. } \beta(x_i)) \qquad R(x_i) = \sum_{j=1}^{N} (u_j - \beta(x_i) p(x_i, y_j))^2 w(x_i, y_j, h_j) \tag{7}$$

There exists an analytical solution to this problem. Setting

$$\frac{\partial R(x_i)}{\partial \beta(x_i)} = 0$$

the minimizer can be found:

$$\beta(x_i) = \sum_{j=1}^{N} u_j \psi_j(x_i)^T \tag{8}$$

where

$$\psi_j(x_i) = P(x_i)^{-1} p(x_i, y_j) w(x_i, y_j, h_j) \tag{9}$$

$$P(x_i) = \sum_{j=1}^{N} p(x_i, y_j) p(x_i, y_j)^T w(x_i, y_j, h_j) \tag{10}$$

ψ_j is a vector of shape functions for data point j, where each row corresponds to shape function for each derivative. P is the moment matrix. It is worth noting that as the smoothing length decreases (i.e. the distance between points decreases), the moment matrix approaches singular. This can make inverting P difficult. To account for this, dropping function arguments and subscripts, the system is pre- and post-conditioned:

$$P\psi = pw$$

The conditioned system is then:

$$\tilde{P}\tilde{\psi} = \tilde{p}w$$

where

$$\begin{aligned} \tilde{P} &= H^{-1} P H^{-1} \\ \tilde{\psi} &= H\psi \\ \tilde{p} &= H^{-1} p \end{aligned}$$

and the preconditioner is

$$H = \mathrm{diag}(b(h))$$

Since the smoothing length h is usually chosen to be proportional to the average distance between points, preconditioning can drastically improve the condition of the moment matrix.

The solution of the local regression problem is the vector of coefficients $\beta(x_i)$. As these are the coefficients of the Taylor series polynomial, Equation (3), they also represent estimates of the function value and its derivatives. For example, in the one dimensional case:

$$\beta(x_i) = \left[\widehat{u}\big|_{x=x_i}, \left.\widehat{\frac{\mathrm{d}u}{\mathrm{d}x}}\right|_{x=x_i}, \left.\widehat{\frac{\mathrm{d}^2u}{\mathrm{d}x^2}}\right|_{x=x_i}, \ldots \right] \tag{11}$$

where $\hat{u}$ represents the estimate of u, etc. Properties of this estimate will be analyzed in the sequel.

Convergence Results

The convergence of the local regression estimate $\beta(x_i)$ has been studied extensively in the statistics literature. Originally formulated in [15], the salient points, as in [6,7], are reproduced here for one-dimensional local regression.

Suppose that the data u is estimated with a polynomial of order n. Furthermore, suppose that the number of data points N and smoothing length h are chosen such that $Nh \to \infty$. For the estimate of the kth derivative, if $n-k$ is odd, then the bias at a point x, or the expected value of the error at x over all possible sets of data points, should converge like

$$Bias(x) \sim h^{n-k+1} \tag{12}$$

Furthermore, if N and h are chosen such that $Nh^3 \to \infty$, then for $n-k$ even, it is expected

$$Bias(x) \sim h^{n-k+2} \tag{13}$$

What this means is that if the number of points and smoothing length are selected properly, then the estimate should converge at least at the Taylor series convergence rate. Even more attractive, if $Nh^3 \to \infty$, then $n-k$ even estimates get a boost in the convergence rate!

It should be noted that the condition $Nh^3 \to \infty$ corresponds to the case where, for data points distributed with uniform density, the number of neighbors goes to infinity. If the data points are distributed uniformly in a box of side length L and a uniform smoothing length h is chosen, then the number of neighbors should trend as follows:

$$B \sim \left(\frac{N}{L^d}\right) h^d$$

In order for the condition $Nh^3 \to \infty$ to be satisfied,

$$h > \frac{1}{N^{1/3}}$$

Hence

$$B > \frac{1}{L^d} N^{1-d/3}$$

If $d < 3$, then $B \to \infty$ as $N \to \infty$. For $d = 3$, the number of neighbors will remain constant. Since the solution of the LRE minimization problem depends on the number of neighbors, more neighbors means larger computation times.

Meanwhile, if only $Nh \to \infty$ is satisfied, then, in the example above, the smoothing length can be chosen as

$$h = LN^{-1/d}$$

In this case, the number of neighbors will go as

$$B \sim 1$$

or the number of neighbors will be constant as $N \to \infty$.

Reproducing Property of LRE

In addition to the above convergence property, the local regression estimate $\beta(x_i)$ possesses an important feature: it is reproducing [5]. That is to say, suppose that the data to be estimated can be expressed as a linear combination of the reproducing basis:

$$u(x) = \lambda^T b(x) \tag{14}$$

where λ is a vector of coefficients, then the estimate will satisfy

$$\beta(x) = \lambda^T J(x) \tag{15}$$

i.e., the estimate will *exactly* reproduce the jet of the basis, which, for Taylor series, means the function value and its derivatives. This property has great implications in the possibility of enriching the reproducing basis to recover underlying symmetries or behavior of the data. For example, enriching b with the radius will allow the local regression estimate to exactly reproduce any functions of the form $f(r) = kr$, where k is a constant.

Tuned Regression Estimation

In the prequel, it was shown that local regression estimation is a convergent and reproducing method to estimate function values and derivatives on a collection of points. In this section, it shall be shown how a convergent mesh-free method may be built from local regression.

Constrained Local Regression

Suppose a priori it is known that data used in local regression should satisfy certain relations, such as differential equations or boundary conditions. These conditions can be reframed as constraints on the data and its derivatives.

For example, suppose that the measured data must satisfy a Poisson equation:

$$\Delta u = f$$

In one dimension, this is

$$u_{xx} = f$$

If the data were to be estimated using a quadratic basis $b = [1, x, x^2/2!]^T$, then from Equation (11) the local regression estimate would be:

$$\beta = [\widehat{u}, \widehat{u_x}, \widehat{u_{xx}}]$$

The differential equation can then be recast as a constraint on β:

$$\beta e_2 - f = 0$$

The constraints need not be global. For example, a problem with Dirichlet and Neumann boundary conditions will have points on one boundary or on the other. Thus, the set of constraints are expressed:

$$\mathscr{D}(x, \beta(x)) = 0 \tag{16}$$

With these constraints, a new estimate at an evaluation point x_i may be found from the minimization problem:

$$\begin{array}{ll} \text{minimize (w.r.t. } \beta(x_i)) & \sum_{j=1}^{N} (u_j - \beta(x_i)p(x_i, y_j))^2 w(x_i, y_j, h_j) \\ \text{subject to} & \mathscr{D}(x_i, \beta(x_i)) = 0 \end{array} \tag{17}$$

Because this approach modifies or "tunes" the estimate, it is referred to as tuned regression [5].

If a basis higher than the order of the constraint is used, then derivatives of the differential equation can be used. If the Poisson problem above were to be estimated with a cubic basis $b = [1, x, x^2/2!, x^3/3!]^T$, then the additional constraint

$$u_{xxx} = f_x$$

could be used to improve the accuracy of the estimate. In this case, the constraints acting on the estimate would be:

$$\begin{cases} \beta e_2 - f = 0 \\ \beta e_3 - f_x = 0 \end{cases}$$

Tuned Regression for the Solution of Differential Equations

In the previous discussion, both local regression and tuned regression have relied on the presence of data and data points when producing the estimate. However, when solving differential equations, the solution is not known a priori!

To handle the lack of data when solving a differential equation, one makes use of the aforementioned constraints. By constraining the estimate with a differential equation and appropriate boundary conditions, it should satisfy the equation, even if the data do not. However, because the data may be inaccurate, the estimate may not solve the problem globally. On the other hand, if the minimization problem were to find data u_j given an estimate $\beta(x_i)$, the result would be the data set which produces the estimate. If these two problems were performed simultaneously, then one should be able to solve for an estimate which satisfies the differential equation and boundary conditions all while choosing the correct data to produce that estimate. This forms the basis of a mesh-free method.

As a note, the set of data points and estimation points are no longer distinct, $\{x_i\}_{i=1}^{M} = \{y_j\}_{j=1}^{N}$ and $M = N$. At each point an estimate is constructed and at each point a corresponding data value is found. Thus, the residual to be minimized is that over all points $\{x_i\}_{i=1}^{N}$:

$$\begin{array}{ll} \text{minimize (w.r.t. } \{\beta(x_i)\}_{i=1}^{N},\ \{u_j\}_{j=1}^{N}) & \sum_{i=1}^{N}\left[\sum_{j=1}^{N}(u_j-\beta(x_i)p(x_i,x_j))^2 w(x_i,x_j,h_j)\right] \\ \text{subject to} & \mathscr{D}(x_i,\beta(x_i)) \quad i=1,\ldots,N \end{array} \tag{18}$$

On the Enforcement of Constraints

In tuned regression, whether as an estimator or as a mesh-free method, there exist various approaches to the enforcement of the constraints $\mathscr{D}$. Detailed here are the direct substitution and Lagrange multiplier methods.

To enforce the constraints directly, a reduced set of variables γ is constructed so that

$$\beta(x) = \mathscr{E}(\gamma(x)) \tag{19}$$

If different constraints are applied at different points (e.g. boundary conditions on boundary nodes), then the set of variables γ will vary at each point, along with the map $\mathscr{E}$. As an illustration of the direct substitution, consider the Poisson equation in two dimensions:

$$u_{xx} + u_{yy} = f$$

If the two-dimensional, quadratic basis is $b = [1, x, y, x^2/2, xy, y^2/2]^T$, then the constraint on the estimate is

$$\beta e_3 + \beta e_5 - f = 0$$

A reduced set may then be constructed by eliminating βe_5:

$$\gamma = [\beta e_0, \beta e_1, \beta e_2, \beta e_3, \beta e_4]$$

and the mapping $\beta = \mathscr{E}(\gamma)$ is:

$$\beta = [\gamma e_0, \gamma e_1, \gamma e_2, \gamma e_3, \gamma e_4, f - \gamma e_3]$$

The advantage of this approach is that the size of the system is then reduced by the number of constraints. However, as illustrated in the previous example, the choice of reduced set is arbitrary, and it is unknown how the choice may affect the solution. Furthermore, algebraic equations may not always have an analytical solution, meaning that for complicated equations finding γ may be a nontrivial task.

With Lagrange multipliers, the constraints are added to the residual by an unknown multiplier. The new system is then minimized over the estimate β, the data u and the multiplier λ:

$$\begin{aligned}&\text{minimize (w.r.t. } \{\beta(x_i)\}_{i=1}^{N},\ \{u_j\}_{j=1}^{N},\ \{\lambda(x_i)\}_{i=1}^{N})\\ &\sum_{i=1}^{N}\left[\sum_{j=1}^{N}(u_j-\beta(x_i)p(x_i,x_j))^2 w(x_i,x_j,h_j)+\lambda(x_i)^T\mathscr{D}(x_i,\beta(x_i))\right]\end{aligned} \tag{20}$$

The use of Lagrange multipliers eliminates the arbitrary choice of reduced variable set; however, the number of unknowns is increased. Furthermore, the residual loses convexity. All results in this paper were computed using Lagrange multipliers.

Known Data

While the data are allowed to vary in the method described above, there may be situations where data are known. For example, in a time-dependent problem, the solution at a time step may be derived from the solution at a previous step. In this case, there are two approaches to handle this previous data. The solution at the previous time step $\beta(t-\Delta t)$ can be used as a constraint on the solution at the current time step, $\beta(t)$. Alternatively, the data or solution from the previous time step can be used as data for the current. Due to the short nature of the Summer Workshop, time-dependent problems were not investigated with this method, though previous work in [5] does show that this is a viable method of time integration.

A Remark on Optimal Data

If Equation (18) (or Equation (20, where $\{x_i\}_{i=1}^{M}=\{y_j\}_{j=1}^{M}$) is differentiated with respect to u_j and the result set to zero, then:

$$\frac{\partial R}{\partial u_j}=2\sum_{i=1}^{M}(u_j-\beta(x_i)p(x_i,y_j))w(x_i,y_j,h_j)=0$$

Solving for the optimal data u_j:

$$u_j=\frac{\sum_{i=1}^{M}\beta(x_i)p(x_i,y_j)w(x_i,y_j,h_j)}{\sum_{i=1}^{M}w(x_i,y_j,h_j)} \tag{21}$$

This is nothing more than the Nadaraya-Watson estimate (or, equivalently, zeroth order local regression) of the local Taylor series polynomial [5] over the evaluation points! Since the constraints are independent of u_j, this result holds no matter which enforcement approach is used. As solving for optimal data u_j is redundant (because βe_0 provides an estimate of the function value), this result may be used for eliminating u_j from the minimization entirely!

Error Metrics

As described above, tuned regression for the solution of elliptic differential equations provides a solution only at the evaluation points, unlike, e.g., the finite element where shape functions reconstruct the solution between nodes, new error metrics need to be introduced.

The theorems from statistics regarding the convergence of local regression make use of the bias, or the expected value of the absolute error at a point over all possible samples of points. To account for the statistical nature of the bias, three metrics were considered: maximum, mean and root mean square error. Let u be the true solution and β the estimate. Then

$$\|e\|_\infty = \max_{1\leq i\leq N} |u(x_i) - \beta(x_i)| \tag{22}$$

$$\|e\|_1 = \frac{1}{N}\sum_{i=1}^{N} |u(x_i) - \beta(x_i)| \tag{23}$$

$$\|e\|_2 = \left[\frac{1}{N}\sum_{i=1}^{N} |u(x_i) - \beta(x_i)|^2\right]^{1/2} \tag{24}$$

The above notation is similar to that of the norms in the functional spaces L_∞, L_1 and L_2, and, in fact, there is some similarity. For example, for the L_2 norm:

$$\|e\|_{L^2(\Omega)} = \left[\int_\Omega |e|^2\,\mathrm{d}V\right]^{1/2}$$

A box of side-length h can approximate a piece of volume

$$\approx \left[\sum_{i=1}^{N} h^d |e(x_i)|^2\right]^{1/2}$$

If $h \sim N^{-1/d}$, then

$$= \left[\frac{1}{N}\sum_{i=1}^{N} |e(x_i)|^2\right]^{1/2}$$
$$= \|e\|_2$$

However, if the smoothing lengths are not chosen in this way, then there is a discrepancy in the two norms. As the ∞-norm is equivalent to the bias, we expect similar convergence rates for all three norms.

Tuned Regression Example Problems and Results

Tuned regression shows promise as a tool to solve differential equations. To test the method, a C++ implementation was built from the ground up. The implementation used the Lagrange multiplier approach to enforcing the constraints. All simulations were performed on a cluster of Intel Xeon E5-4650 2.70 GHz processors with a total of 64 cores.

Testing Local Regression Capabilities - Reproducing Property with Data

As local regression is simply tuned regression with zero constraints and prescribed data, preliminary tests of the method involved recovering the desirable properties of local regression.

The first test involved recovering the reproducing property of local regression. Random points were distributed on the unit square $\Omega = [0,1] \times [0,1]$ and the smoothing length was chosen to be $h \sim N^{-1/2}$. The function

$$u(x,y) = 4x^2 + 3xy + 2y^2 + x + y \tag{25}$$

was prescribed as data at every point and no constraints were enforced. Shown below are the convergence results of this test.

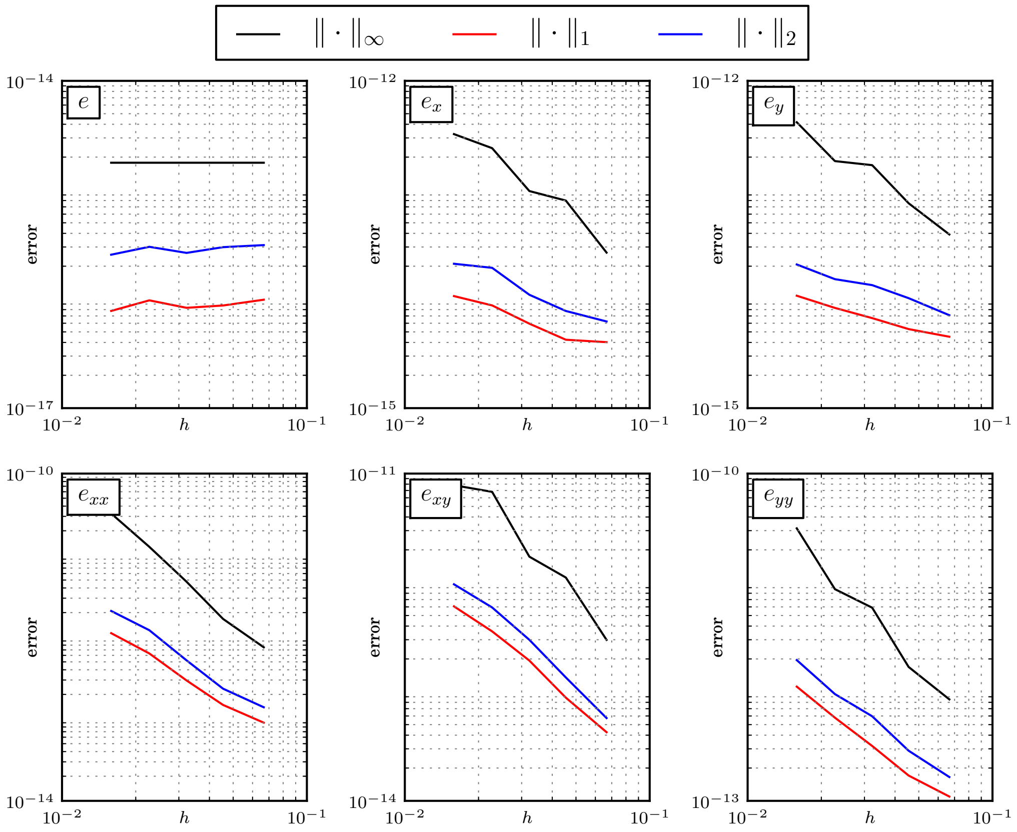


Figure 1: Error for the Reproducing Problem with Data

The data is successfully reproduced to within tolerance. As expected, mean error is the smallest, while maximum error is the largest. Interestingly, higher derivatives show greater errors, even increasing with smaller smoothing lengths, while the zeroth derivative estimate is nearly constant for all smoothing lengths.

Testing Local Regression Capabilities - Reproducing Property with Constraints

As a second test, the estimate was constrained through the Poisson problem: Let $\Omega = (0,1) \times (0,1)$. Find $u(x,y) \in C^2(\Omega)$ such that for all $(x,y) \in \Omega$

$$\Delta u(x,y) = 12 \tag{26}$$

with boundary conditions

$$u(x,0) = 4x^2 + x \tag{27}$$
$$u(x,1) = 4x^2 + 4x + 3 \tag{28}$$
$$u_x(0,1) = 3y + 1 \tag{29}$$
$$u_x(0,1) = 3y + 9 \tag{30}$$

The solution to this problem is given by Equation (25). The purpose of this test was to observe whether the reproducing property could be recovered without prescribing data. Using the same points and smoothing lengths as the previous test, the convergence results are shown following:

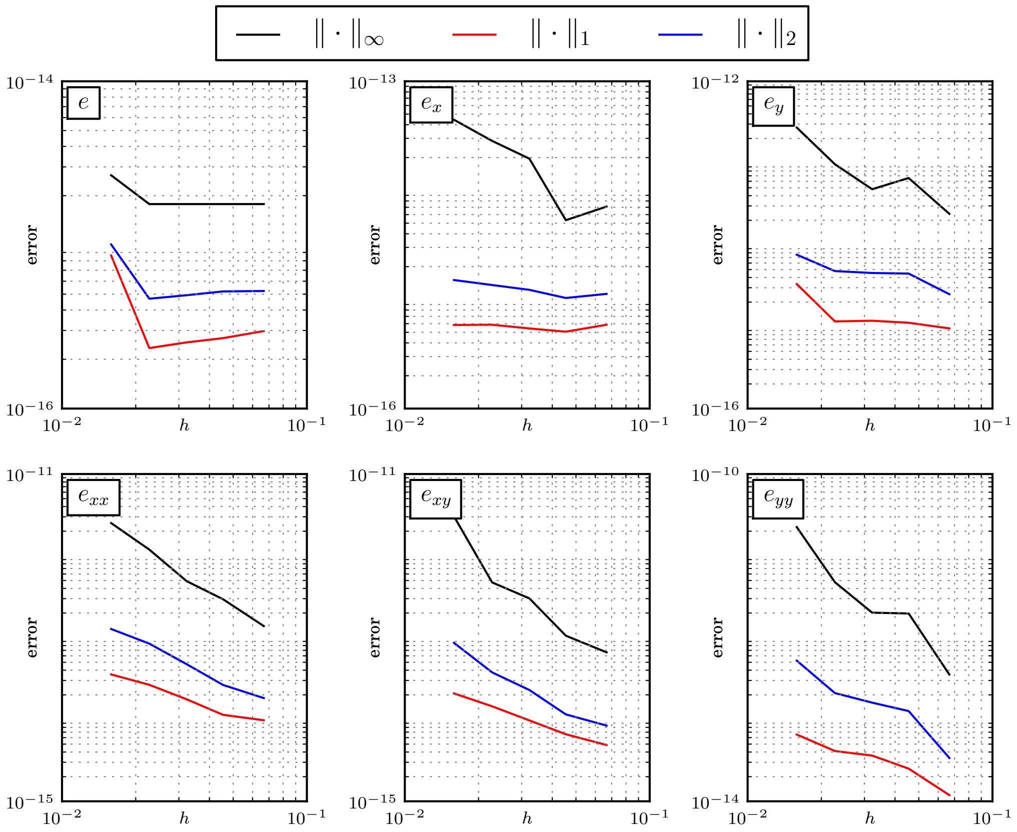


Figure 2: Error for the Reproducing Problem with Constraints

The error in each derivative of the estimate is very small and close to machine precision. Thus, as a mesh-free method, tuned regression is capable of achieving the reproducing property! Again, the zeroth derivative estimate is nearly constant across smoothing lengths, while higher derivatives exhibit larger errors.

Testing Local Regression Capabilities - Convergence Property $Nh \to \infty$

It was also investigated whether the code could recover the convergence rates for local regression. Convergence of the code was tested with the two-dimensional data

$$u(x,y) = \sin(\pi x)\sin(\pi y) \tag{31}$$

prescribed at random points in the unit square $\Omega = [0,1] \times [0,1]$. The presence of sines in the data ensures that the estimate does not benefit from the reproducing property of local regression.

In this example, the smoothing length was chosen as $h \sim N^{-1/2}$. This guarantees that the number of neighbors is roughly constant as the number of points increases. From Equation (12), the estimate is expected to get a convergence rate of h^{n-k+1}. Linear, quadratic, and cubic bases were used to study the convergence rates. Shown below are the convergence plots for a linear basis,

$$b = [1, x, y]^T \tag{32}$$

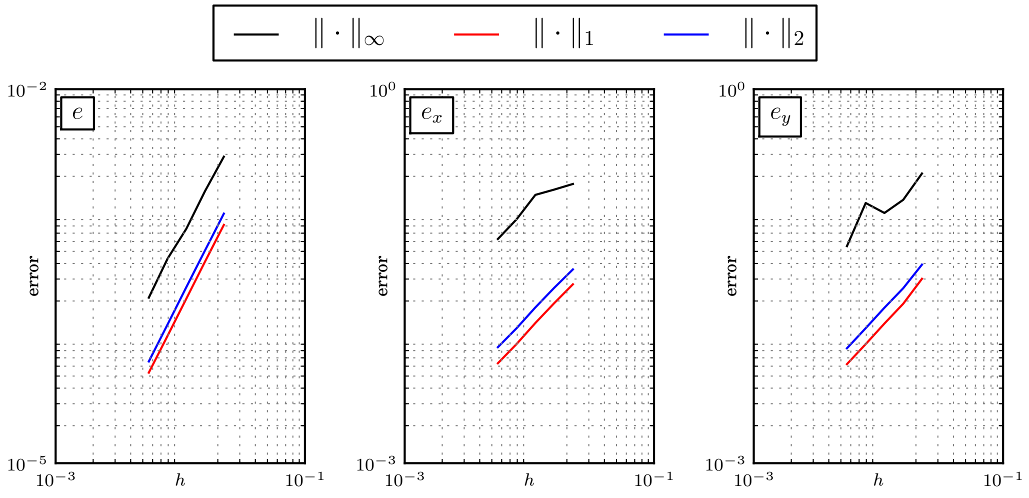


Figure 3: Error for local regression with linear basis with $Nh \to \infty$

The convergence rates are summarized in the following table:

	$\|\cdot\|_\infty$	$\|\cdot\|_1$	$\|\cdot\|_2$
e	1.9	2.0	2.0
e_x	0.7	1.1	1.0
e_y	0.8	1.1	1.1

Table 1: Convergence rates for local regression with linear basis with $Nh \to \infty$

The expected convergence rates for the linear basis are [2, 1, 1], which is exactly what is seen above. The error plots are shown below for a quadratic basis

$$b = [1, x, y, x^2/2, xy, y^2/2]^T \tag{33}$$

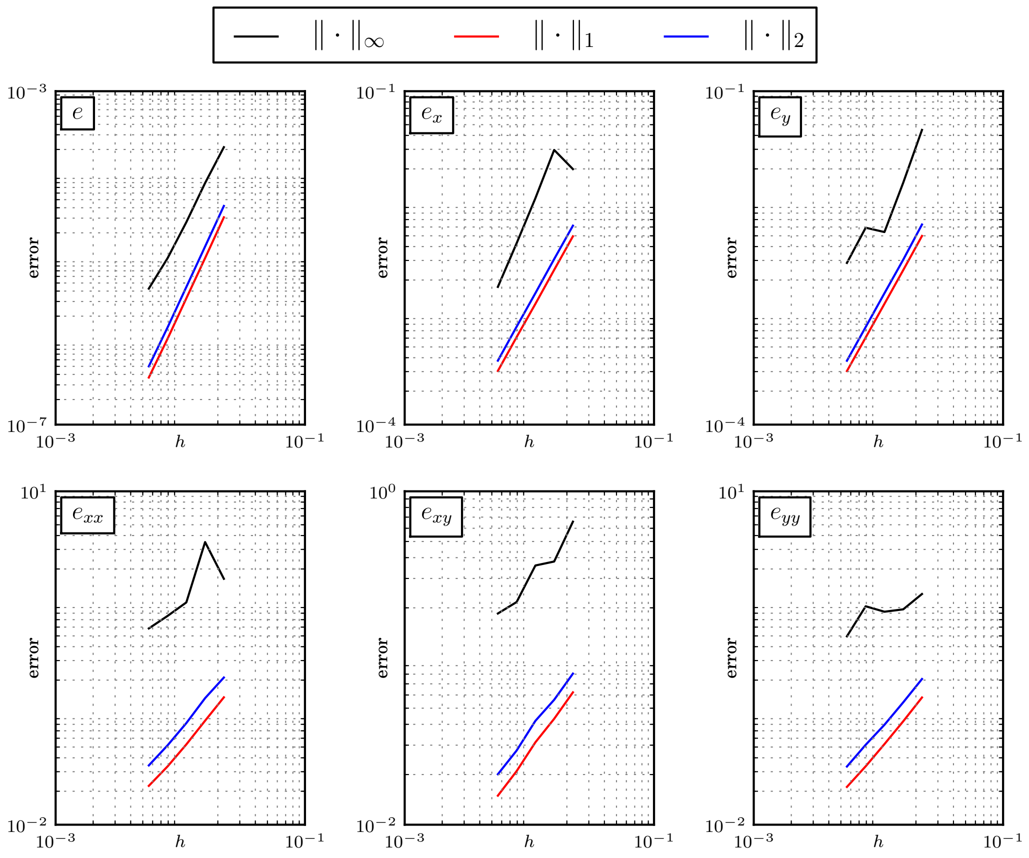


Figure 4: Error for local regression with quadratic basis with $Nh \to \infty$

The convergence rates are summarized in the table below:

	$\|\cdot\|_\infty$	$\|\cdot\|_1$	$\|\cdot\|_2$
e	2.9	3.2	3.2
e_x	2.0	2.0	2.0
e_y	1.9	2.0	2.0
e_{xx}	1.0	1.3	1.3
e_{xy}	0.9	1.0	1.0
e_{yy}	0.5	1.3	1.3

Table 2: Convergence rates for local regression with quadratic basis with $Nh \to \infty$

Expected convergence rates for the quadratic basis are [3, 2, 2, 1, 1, 1]. Similar rates are seen for the mean and RMS norms, but there is a small deviation in the convergence rate for $\|e_{yy}\|_\infty$. Finally, results for the cubic basis

$$b = [1, x, y, x^2/2, xy, y^2/2, x^3/6, x^2y/2, xy^2/2, y^3/6]^T \tag{34}$$

are shown below:

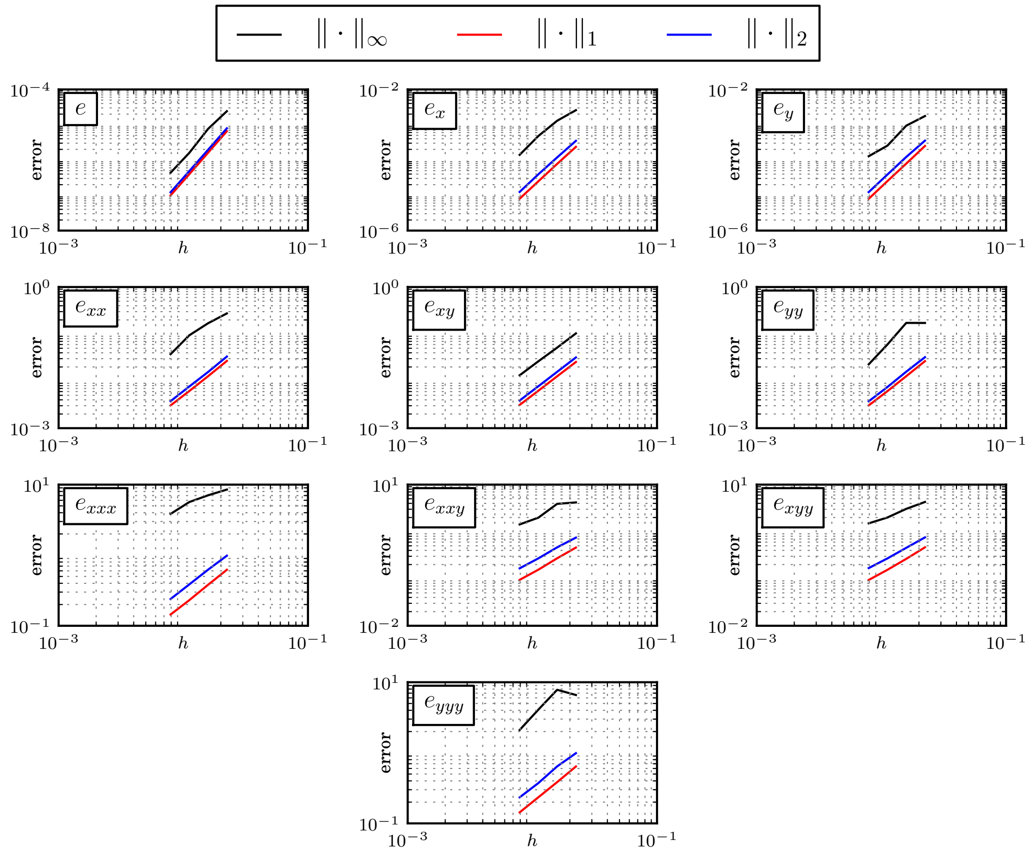


Figure 5: Error for local regression with cubic basis with $Nh \to \infty$

Convergence rates are summarized below:

	$\|\cdot\|_\infty$	$\|\cdot\|_1$	$\|\cdot\|_2$
e	3.9	4.0	4.0
e_x	2.8	3.3	3.2
e_y	2.6	3.3	3.2
e_{xx}	1.9	2.1	2.1
e_{xy}	2.0	2.0	2.0
e_{yy}	2.1	2.1	2.1
e_{xxx}	0.8	1.4	1.4
e_{xxy}	1.1	1.5	1.5
e_{xyy}	1.0	1.5	1.5
e_{yyy}	1.2	1.4	1.4

Table 3: Convergence rates for local regression with cubic basis with $Nh \to \infty$

For a cubic basis, the expected convergence rates are $[4,3,3,2,2,2,1,1,1,1]$, which is exactly as observed. In all three cases, the optimal convergence rates for local regression with $Nh \longrightarrow \infty$ were verified.

Testing Local Regression Capabilities - Convergence Property $Nh^3 \longrightarrow \infty$

In a final test of local regression, the smoothing length was chosen as $h \sim N^{-1/4}$. With this choice of smoothing length, $Nh^3 \longrightarrow \infty$. Because this condition is now satisfied, Equation (13) predicts a convergence rate of h^{n-k+2} for $n-k$ even. Below are results for the convergence of the linear basis, Equation (32):

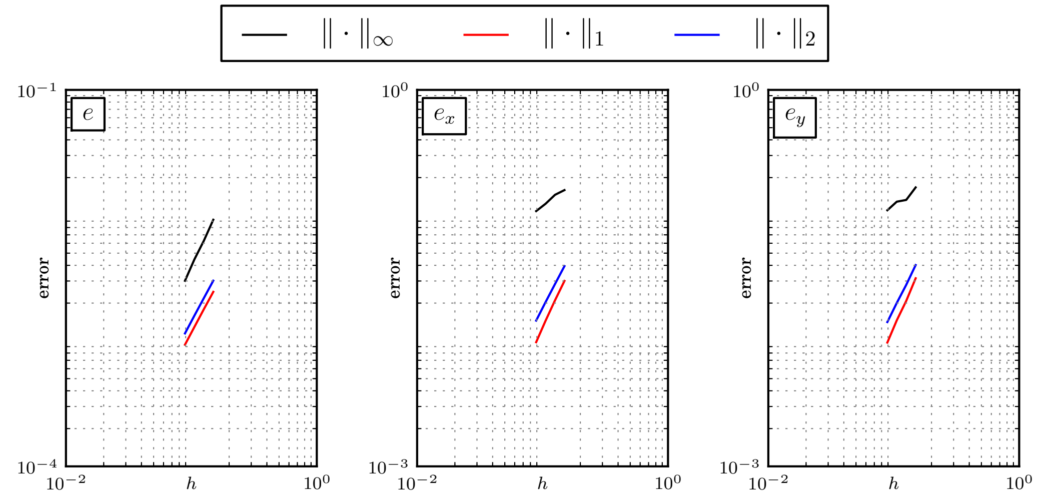


Figure 6: Error for local regression with linear basis with $Nh^3 \longrightarrow \infty$

The convergence rates are summarized in the following table:

	$\|\cdot\|_\infty$	$\|\cdot\|_1$	$\|\cdot\|_2$
e	2.1	1.9	1.9
e_x	0.8	2.2	1.9
e_y	0.7	2.2	2.0

Table 4: Convergence rates for local regression with linear basis with $Nh^3 \longrightarrow \infty$

For first derivative estimates, $(n-k=0)$ is even, which means that a second order convergence rate is expected. This is observed in the mean and RMS mean estimates. However, there is still first order convergence in the maximum norm. Shown below are the convergence results for the quadratic basis, Equation (33):

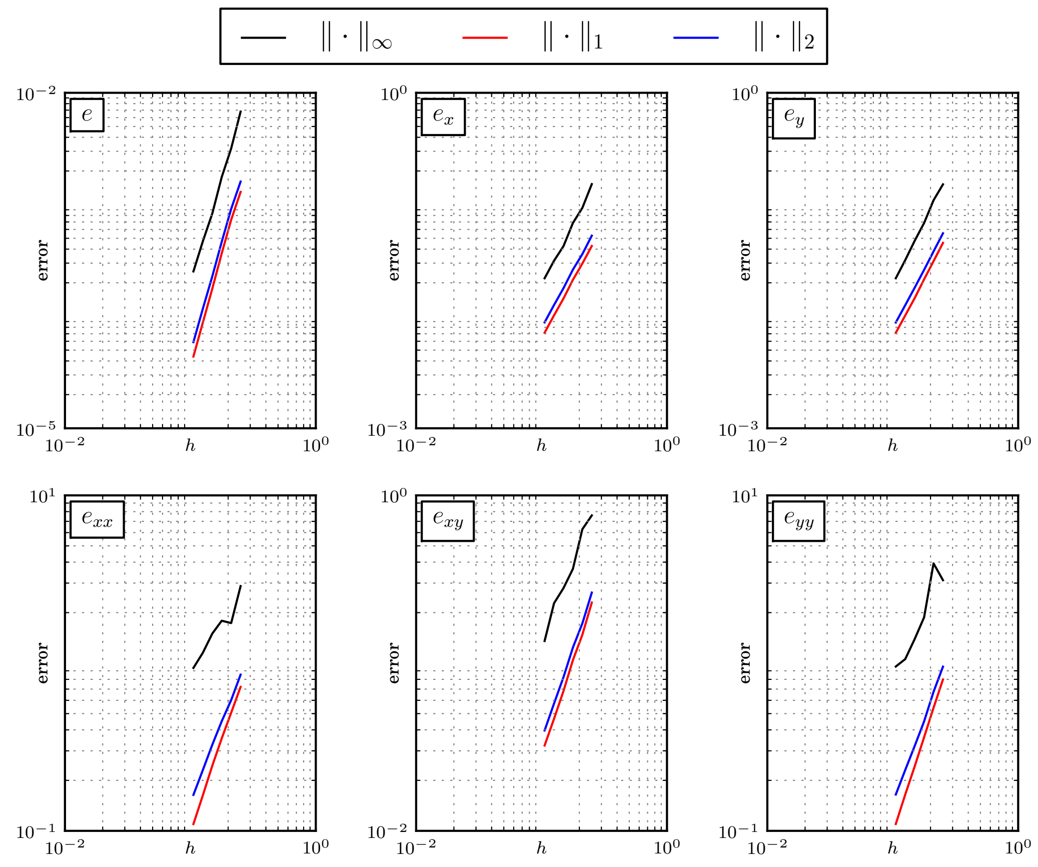


Figure 7: Error for local regression with quadratic basis with $Nh^3 \to \infty$

The convergence rates are summarized in the table below:

	$\|\cdot\|_\infty$	$\|\cdot\|_1$	$\|\cdot\|_2$
e	3.8	4.0	3.9
e_x	2.2	2.1	2.1
e_y	2.3	2.1	2.1
e_{xx}	1.2	2.2	1.9
e_{xy}	2.0	2.3	2.2
e_{yy}	1.7	2.3	2.0

Table 5: Convergence rates for local regression with quadratic basis with $Nh^3 \to \infty$

In the case of a quadratic basis, the zeroth derivative ($n-k=2$) and second derivative estimates ($n-k=0$) should converge at fourth and second order rates, respectively. This is observed for the zeroth order estimate. However, the maximum norm shows some suboptimality for the convergence rate of $\|e_{xx}\|_\infty$. Finally, convergence results for a cubic basis, Equation (34), are shown below:

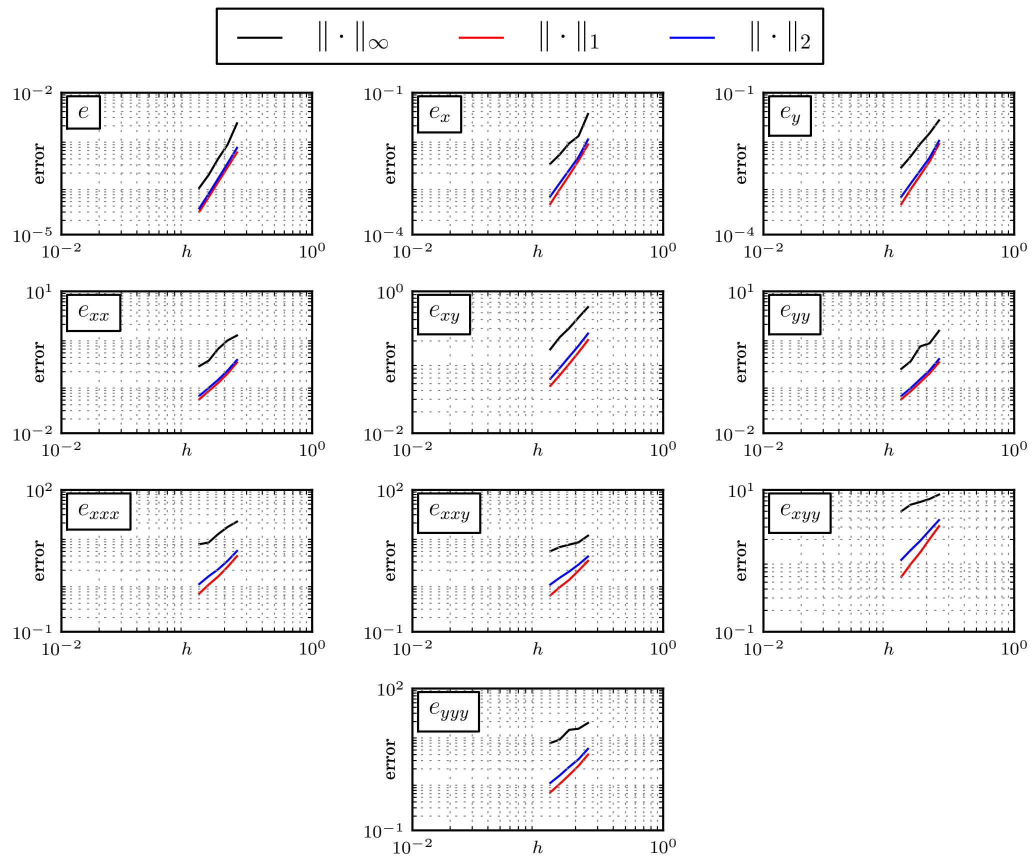


Figure 8: Error for local regression with cubic basis with $Nh^3 \to \infty$

Convergence rates are summarized below:

	$\|\cdot\|_\infty$	$\|\cdot\|_1$	$\|\cdot\|_2$
e	4.5	4.2	4.3
e_x	3.3	4.2	3.9
e_y	3.3	4.2	3.9
e_{xx}	2.3	2.6	2.5
e_{xy}	2.0	2.1	2.1
e_{yy}	2.6	2.6	2.6
e_{xxx}	1.7	2.6	2.3
e_{xxy}	1.0	2.4	2.0
e_{xyy}	0.7	2.4	1.9
e_{yyy}	1.4	2.6	2.4

Table 6: Convergence rates for local regression with cubic basis with $Nh^3 \to \infty$

For a cubic basis, first derivative ($n-k=2$) and third derivative ($n-k=0$) estimates should converge with fourth and second order rates, respectively. This is observed for the mean and RMS norms, but not for the maximum norm, where $\|e_x\|_\infty$ and $\|e_y\|_\infty$ converge at closer to third order while $\|e_{xxy}\|_\infty$ and $\|e_{xyy}\|$ converge at near first order. In general, however, the mean and RMS error do exhibit the extra convergence rates for $n-k$ even estimates.

A Model Problem - The Poisson Equation

It has been demonstrated that the method is capable of acting as a local regression estimator and recovering the convergence rates detailed previously. With this in mind, the Poisson equation was chosen as a model problem. It is a second-order partial differential equation, which allows for the use of Neumann and Dirichlet boundary conditions. Furthermore, with manufactured solutions, the analytical solution is easily obtained to compare against the method.

For the purposes of this paper, the following problem was considered. Let $\Omega = (0,1) \times (0,1)$. Find $u(x,y) \in C^2(\Omega)$ such that

$$\Delta u(x,y) = -2\pi^2 \sin(\pi x)\sin(\pi y) \tag{35}$$

holds for all $(x,y) \in \Omega$ and the following hold on the boundary:

$$u(x,0) = 0 \tag{36}$$

$$u(x,1) = 0 \tag{37}$$

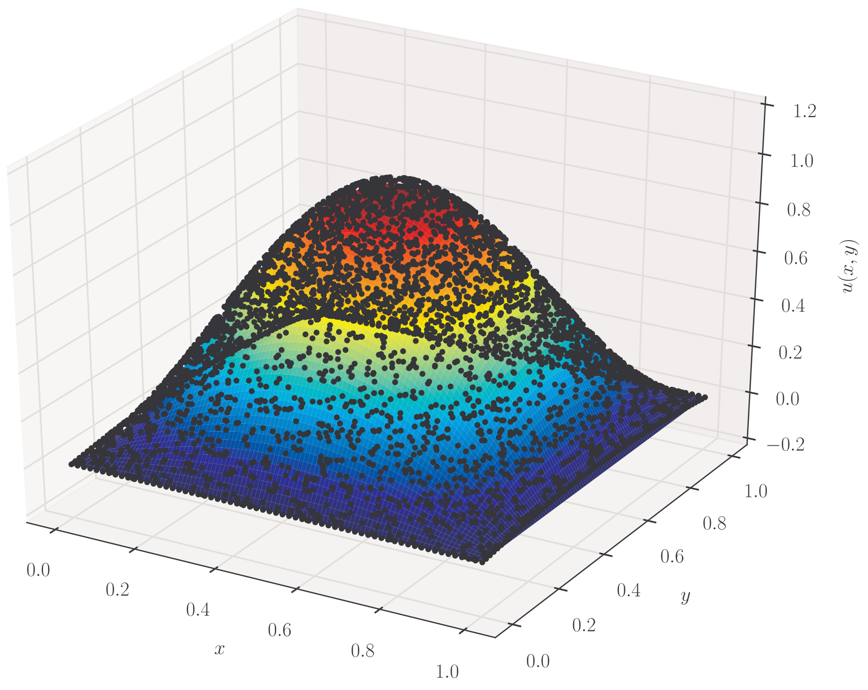
$$u_x(0,y) = \pi \sin(\pi y) \tag{38}$$

$$u_x(1,y) = \pi \sin(\pi y) \tag{39}$$

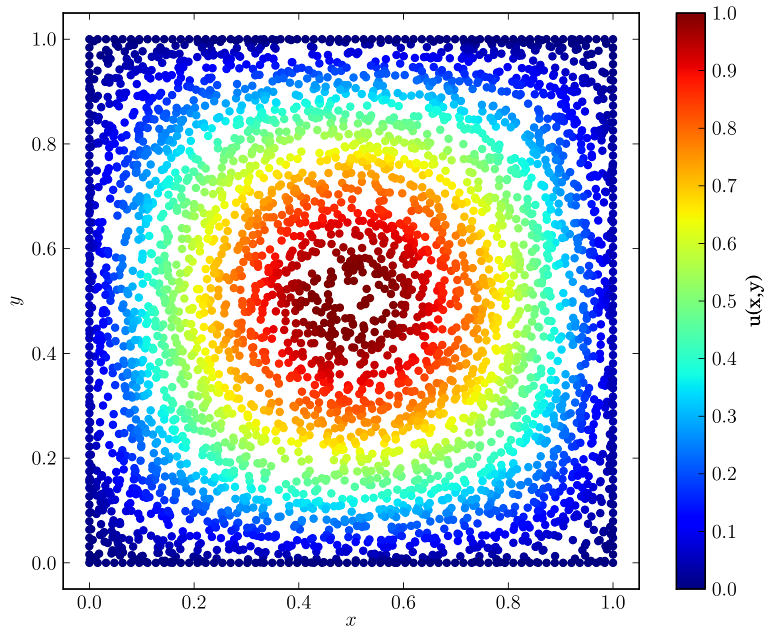
This problem was chosen because it uses both Dirichlet and Neumann boundary conditions. Furthermore, the solution

$$u(x,y) = \sin(\pi x)\sin(\pi y) \tag{40}$$

involves sines, meaning that the solution will not be reproducible by local regression, as the sine function cannot be expressed as a finite-length polynomial. Shown below is a scatter plot of the solution with approximately 4000 data points. Tuned regression was performed with a quadratic basis, Equation (33).



(a)



(b)

Figure 9: (a) 3D plot of solution to model problem with 4000 points (black points) and analytical solution. (b) Solution to the model problem with 4000 points

There is good agreement between the analytical solution and the tuned regression solution.

Of note is the discrepancy on the boundary $x = 1$. This corresponds to one of the Neumann boundaries, where the function value itself is not constrained. Shown below are the convergence results for the model problem, where the number of points and smoothing length are chosen so that $Nh^3 \to \infty$.

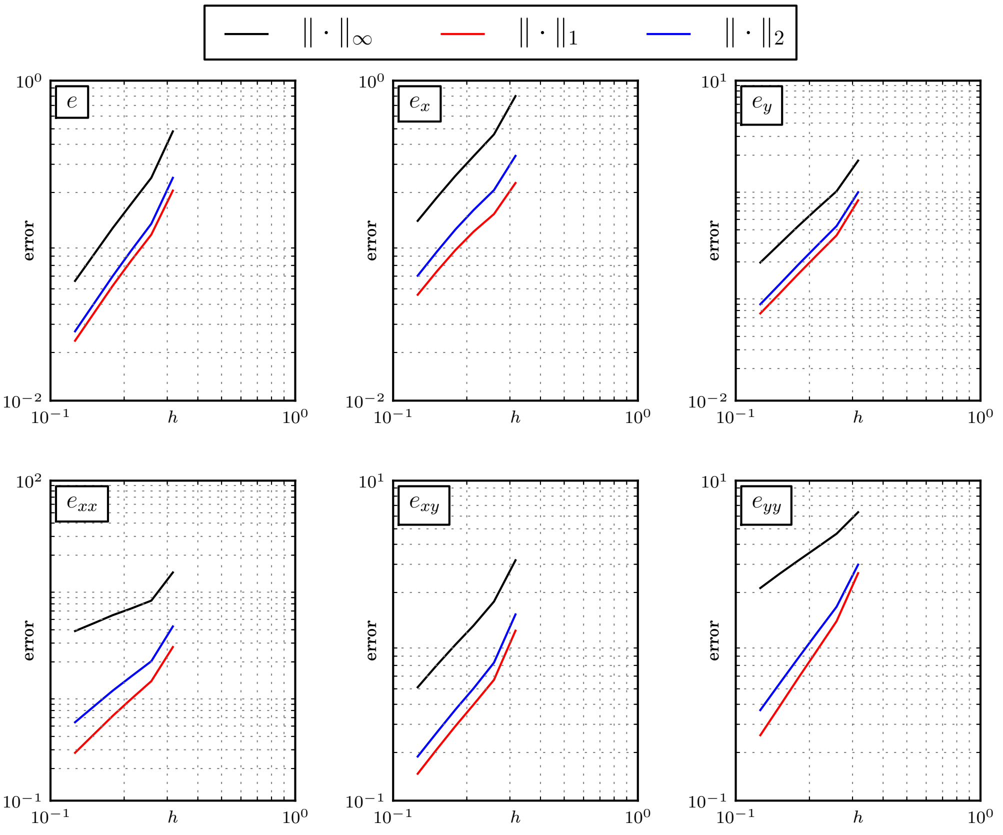


Figure 10: Error for the model problem with $Nh^3 \to \infty$

Shown below are the convergence rates for the model problem errors.

	$\|\cdot\|_\infty$	$\|\cdot\|_1$	$\|\cdot\|_2$
e	2.3	2.3	2.3
e_x	1.9	1.7	1.8
e_y	2.3	2.6	2.6
e_{xx}	1.3	2.4	2.1
e_{xy}	1.9	2.1	2.1
e_{yy}	1.2	2.5	2.2

Table 7: Convergence rates for the model problem with $Nh^3 \to \infty$

The model problem shows convergence at nearly optimal rates for first derivatives. However, there is a lower convergence rate for the error in the zeroth derivative estimate. Furthermore, there is some suboptimality in the convergence rates of $||e_{xx}||_\infty$ and $||e_{yy}||_\infty$. This could be due to the fact that the derivatives are constrained by the Poisson equation. It could also result from the optimal data. Because the data which is being smoothed is not the true solution, but rather is a zeroth order smoothing of the estimate, there could be some discrepancies. However, this requires further study.

Mechanics - Linear Elastostatic Beam Bending

To test the method on a more complex system, the problem of the elastostatic deformation of a beam was chosen. The problem is to find a displacement field $\mathbf{u}(x,y) \in C^2(\Omega)$, where $\Omega = (0,L) \times (-a,a)$ which satisfies

$$\sigma_{ij,j} = 0 \tag{41}$$

for all $(x,y) \in \Omega$, along with the boundary conditions

$$\begin{align} \mathbf{u}(0,y) &= \mathbf{0} \tag{42} \\ \mathbf{u}(L,y) &= [0,\bar{u}] \tag{43} \\ \sigma_{ij}(x,a)n_j &= 0 \tag{44} \\ \sigma_{ij}(x,-a)n_j &= 0 \tag{45} \end{align}$$

where σ is the Cauchy stress tensor and $\mathbf{n}$ is the outward unit normal. The beam was taken to be linear elastic:

$$\sigma_{ij} = \lambda \varepsilon_{kk}\delta_{ij} + 2\mu\varepsilon_{ij} \tag{46}$$

where λ and μ are the Lamé parameters. For all simulations, the following parameters were used:

$$\begin{align} L &= 1.0 \\ a &= 0.1 \\ \lambda &= 1.0 \\ \mu &= 1.0 \\ \bar{u} &= 0.01 \end{align}$$

although the vertical displacement was arbitrary as the solution for any displacement can be found by scaling (the principle of superposition). Shown below is the deformed configuration of the beam with 8000 points and the deformation scaled by a factor of 5.

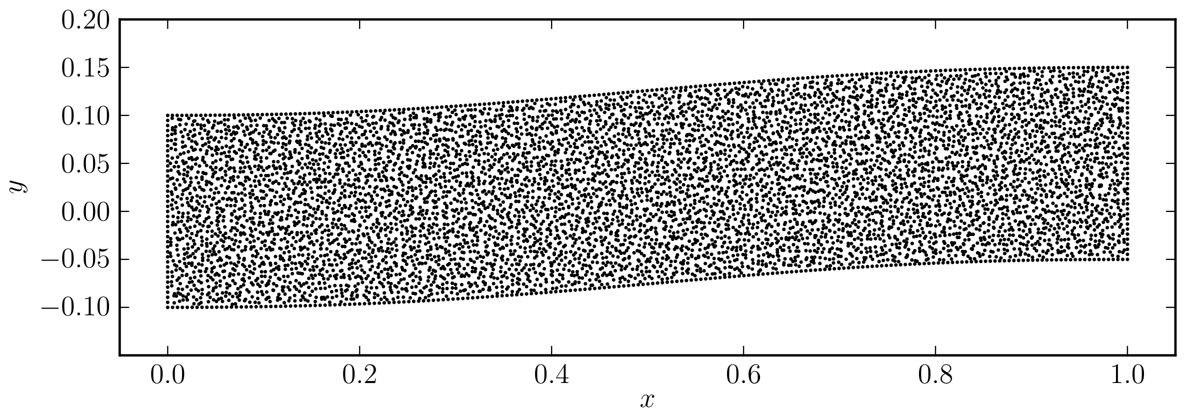


Figure 11: Deformed configuration of elastic beam

With tuned regression, the elastostatics problem was successfully solved. The displacement field, and derivatives up to second order, were estimated. From these and Hooke's Law, Equation (46), the stress field was reconstructed, shown below:

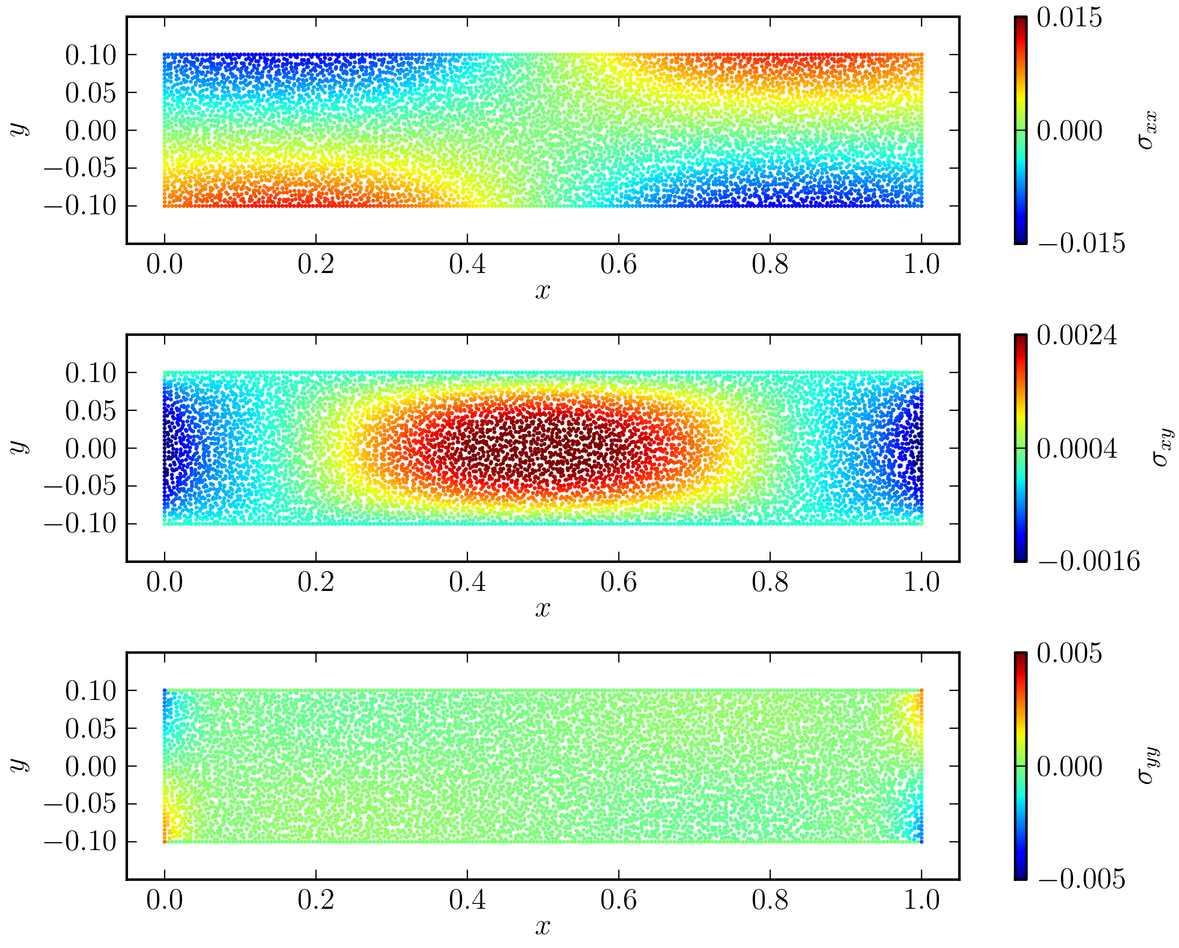


Figure 12: (Top) σ_{xx}, (Middle) σ_{xy}, (Bottom) σ_{yy}

The Spherized Basis

The Problem

In hypervelocity hydrodynamic mechanics problems, radially symmetric functions play an important role, e.g. modeling rapid expansion or contraction of a material. Like other conservative properties, the associated numerical representation of these functions should preserve the accuracy of its radial symmetry. Nevertheless, when using Euclidean coordinates this is not necessarily the case; however, this is the default coordinate system for many particle methods. Achieving such computation radial accuracy requires converting the model equations to spherical or cylindrical coordinates and, in turn, adjusting code implementation of the method. In an attempt to avoid conversion to a radially based coordinate system, it is surmised that an enrichment of the standard Taylor series basis can simplify the route to the numerical conservation of radial symmetry.

Non-Taylor Series Basis

It has already been shown that reproducing bases other than the Taylor series can be used with LRE [5]. For example, using the standard jet, Equation (5), bases such as

$$[1,\sin(x),\cos(x)]^T, \quad [1,e^x,e^{-x}]^T \tag{47}$$

result in shifted bases

$$\begin{bmatrix} 1 \\ \sin(y_i-x) \\ 1-\cos(y_i-x) \end{bmatrix}, \quad \begin{bmatrix} 1 \\ \frac{1}{2}e^{-(y_i-x)}(-1+e^{2(y_i-x)}) \\ \frac{1}{2}e^{-(y_i-x)}(-1+e^{(y_i-x)})^2 \end{bmatrix}. \tag{48}$$

However, $r(x)=\sqrt{\sum_\alpha^d x_\alpha{}^2}$ for $x\in\mathbb{R}^d$ is a function which is linearly independent of the Euclidean coordinates. As such, the traditional Taylor series bases can simply be enriched by appending r resulting in a *spherized basis*. In two dimensions, the linear basis is now given by

$$b_{sp}(x)=[1,x_1,x_2,r]^T. \tag{49}$$

Note that theoretically the reproduction of r is now possible, helping to ensure accuracy of spherically symmetric functions. As before, a nonsingular jet matrix $J_{b_{sp}}$ must be derived in order to solve for a shifted basis. For $J_{b_{sp}}$ to be square, another linear operator applied to the basis must be also be appended. While other functions are potentially viable options, the *spherized jet*

$$J_{b_{sp}}(x)=\left[b,\frac{db}{dx_1},\frac{db}{dx_2},\Delta b\right] \tag{50}$$

provides a simple and effective choice. To not result in a degenerate jet, the additional operator must be selected carefully, e.g. if $\frac{db}{dr}$ or $\frac{d^2b}{dx_1x_2}$ is chosen as the operator in the last column, $J_{b_{sp}}$ will be singular or undefined on axes, respectively. The process is continued by expanding Equation (50) and finding the jet inverse,

$$J_{b_{sp}}(x)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ x_1 & 1 & 0 & 0 \\ x_2 & 0 & 1 & 0 \\ r & \frac{x_1}{r} & \frac{x_2}{r} & \frac{1}{r} \end{bmatrix}, \quad J_{b_{sp}}(x)^{-1}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ -x_1 & 1 & 0 & 0 \\ -x_2 & 0 & 1 & 0 \\ 0 & -x_1 & -x_2 & r \end{bmatrix} \tag{51}$$

and then solving

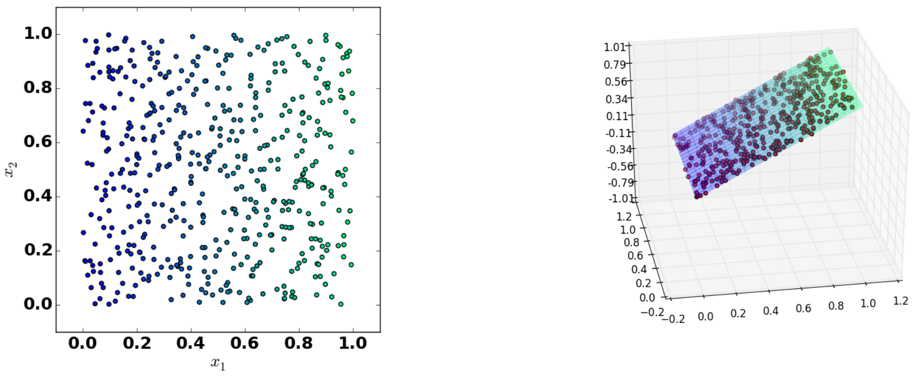
$$p_i(x) = J_{b_{sp}}(x)^{-1} b_{sp}(y_i), \tag{52}$$

which gives the shifted basis

$$p_i(x) = [1, y_{1i} - x_1, y_{2i} - x_2, -(x_1 y_{1i} + x_2 y_{2i}) + r(x) r(y_i)]^T. \tag{53}$$

Code Implementation

In order to test the reproduction and convergence implications of a spherized basis, an LRE program was implemented in Python to both fully comprehend the process and subsequently build upon that foundation. Using randomly generated two dimensional data and evaluation points over the domain $\Omega = [0,1]^2$, circular neighborhood support, cubic spline weight function, Equation (1), and the previously described conditioning, the code can perform from zeroth to fourth order Taylor series regressions, $n \in [0,4]$, on a single field function u.

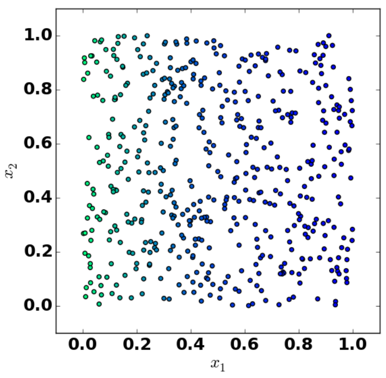


(a) Heat Map (b) 3D Map

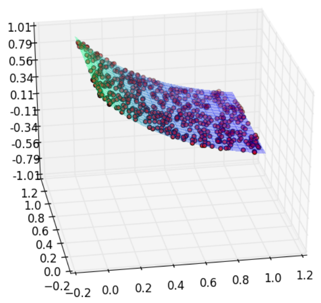
Reproduction Test: Approximation of Field Function, $\hat{u}$

Figure 13: Third order Taylor series basis reproduction test for function $u(x) = x_1$ with Python LRE implementation using 500 evaluation points, 100 data points, and 0.3 smoothing length, **(a)** maps $\hat{u}$ to color and **(b)** 3D comparison of $\hat{u}$ and the continuous field curve. The ℓ_∞-error $= 6.22391027605\text{e-}13$.

To demonstrate the effectiveness of the code, a reproduction test, Figure 13, shows that since the field function $u(x)$ is a linear combination of the Taylor series monomial basis functions, machine precision accuracy is achieved. Next, a test for a non-reproducing function can be observed in Figure 14. Using a third order two dimensional ($n = 3$, $d = 2$) approximation, a fourth order convergence for the zeroth derivative agrees with the expected theoretical result from Equation (12).



(a) Heat Map

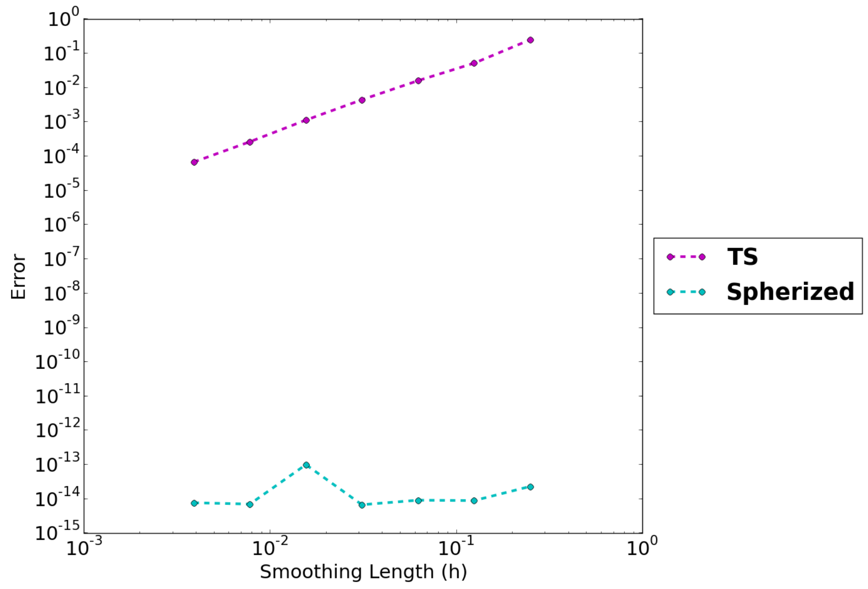


(b) 3D Map

Non-Reproducing Test: Approximation of Field Function, $\hat{u}$

Figure 14: Third order Taylor series basis test for function $u(x) = e^{-2x}$ with Python LRE implementation using 500 evaluation points, 500 data points, and 0.3 smoothing length. The ℓ_∞-error $= 0.000376385078803$.

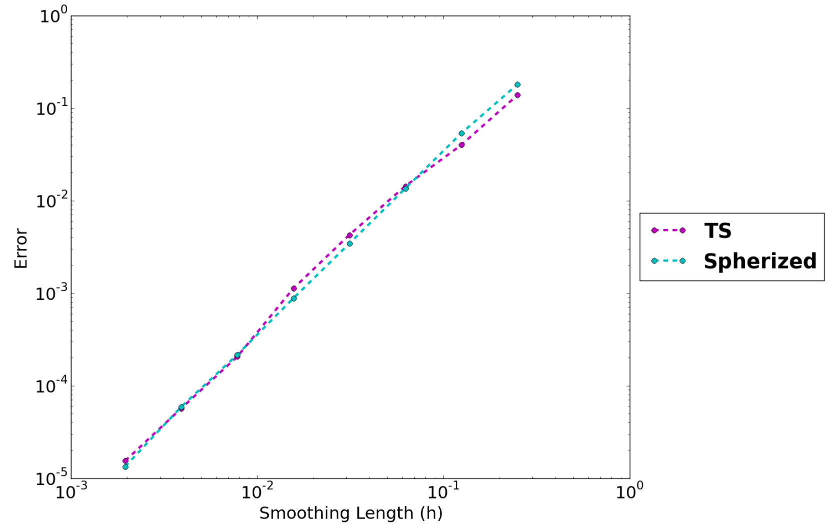
The spherized basis demonstrates success over the Taylor series basis by reproducing $u(x) = r$. In Figure 15, it is shown that the spherized basis achieves machine precision error while the comparable first order Taylor series, without r appended, only achieves second order convergence per the theoretical results.



Error Comparison for Reproduction: Spherized vs Taylor Series Basis

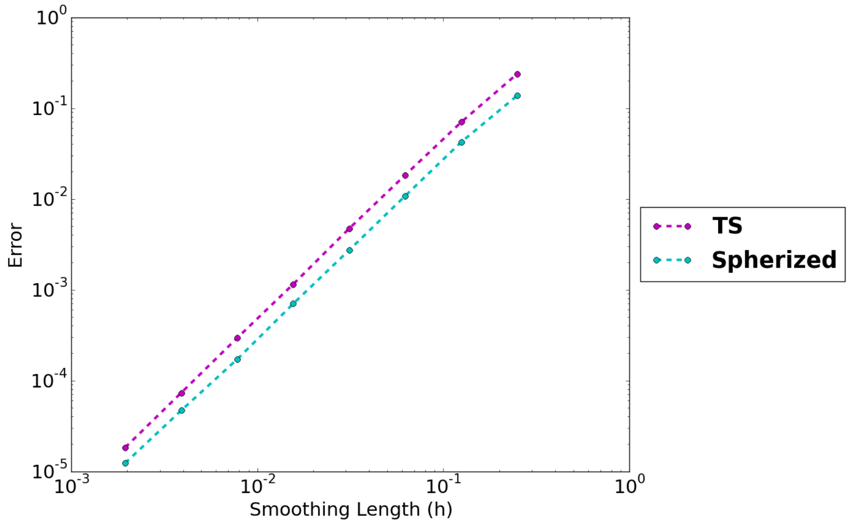
Figure 15: Comparison of first order spherized and Taylor series basis ℓ_2-error for function $u(x) = r$ with Python LRE implementation over two decades of smoothing length over 250 random evaluation points shows reproduction for spherized.

Comparing the evaluation of a radial, non-reproducing function using both first order Taylor series and spherized bases reveals that while both have order two convergence, the spherized performs slightly better, Figure 16 and Figure 17.



Error Comparison Non-Reproducing Radial Field #1: Exponential

Figure 16: First order spherized and Taylor series basis test for function $u(x) = e^{-2r}$ with Python LRE implementation over two decades of smoothing length over 300 random evaluation points. The convergence rates are nearly identical for both spherized and Taylor series bases.



Error Comparison Non-Reproducing Radial Field #2: Bessel

Figure 17: First order spherized and Taylor series basis test for function $u(x) = J_0(-2r_i)$ over two decades of smoothing length over 300 random evaluation points. Spherized converges with the same order but has smaller error.

Overall, it can be seen that not only does the spherized basis work in its own right, it has favorable performance attributes compared to the standard Taylor series basis for radially symmetric functions.

Conclusions and Future Work

Tuned Regression for the Solution of Differential Equations

Local regression offers great promise as the basis for a novel mesh-free solver of differential equations. A working C++ implementation was created and has been shown to reproduce the desirable properties of local regression and to solve several time-independent examples. Tuned regression as a mesh-free method offers numerous benefits over existing mesh-free approaches by including the direct incorporation of boundary conditions (which are automatically and exactly satisfied at all points on the boundary) and the flexibility of LRE to handle non-Taylor series approximations.

The use of tuned regression to solve differential equations is new. This means that there is much analysis to be done and strengths and weaknesses to be discovered. The sub-optimal convergence rates for certain derivatives in the model problem are troubling. It is uncertain whether this is a shortcoming of the method or whether it is an issue with the implementation. When solving for the minimizer with Lagrange multipliers, the resulting matrix system can be poorly conditioned. Preconditioning the system helps, though it is possible that there are numerical errors produced in the solution step.

An immediate extension of this work is to tackle time-dependent problems. The heat equation is a prototypical problem, though elastodynamics or Euler equations/Navier-Stokes appear within reach. Time-dependent problems raise new issues, including appropriate time-step selection and numerical stability. Furthermore, questions are raised about how to advect the particles and update the smoothing lengths. For Navier-Stokes or the Euler equations, the flexibility of LRE offers the possibility of using the method to capture shocks. However, this requires alternate weight functions which may depend on the current state of the system.

Finally, as discussed previously, the method solves for optimal data u_j. However, this seems redundant, given that an estimate of the function values βe_0 is one of the outputs of local regression. The optimal solution to the minimization with respect to the data is a Nadaraya-Watson estimate of the local regression estimate. However, it is possible that other a priori guesses for the data may yield better results. For example, one could regress directly against the local regression estimate βe_0. It remains to be seen how the choice of u_j affects the accuracy or convergence rate of the solution.

Spherized Bases

Local regression estimation is a powerful tool within the process of using mesh-free methods to solve mechanics problems. It is highly flexible, thus requiring its code execution to contain many parameters and design choices. Within the Python implementation presented here there is substantial room for imbuing more robustness in this regard. For example, in order to better solve material fracture mechanics, where particles lose neighbors as the material is pulled apart, a faceted/polygonal support shape can provide more information via its edges compared to a simple circular or ellipsoid shape. Hence, adding other support shapes like faceted, rectilinear, and ellipsoid is an important extension for solving more types of problems more accurately. As a result, there would be an additional need of the associated routines for finding the neighbors within each shape and the opportunity to optimize the neighbor searching algorithm for computational time and space. Another place for improvement related to the support is dynamically encoding smoothing length for optimal number of neighbors. As this implementation is intended only as an educational tool, it is worth noting that at some point the Python language is not suitable or capable enough in comparison to other lower-level languages to support the level of performance required for advanced research and scientific development. Further, while it is evident that a spherized basis performs successfully and more accurately for radial functions, only a cursory amount of investigation and analysis was conducted. More work is needed to find the full potential of a spherized basis including higher order basis, testing other spherized options, for instance with the inclusion of θ, more convergence analysis to discover the underlying mathematical trends, and ultimately using the spherized basis in the mesh-free numerical solution of differential equations. Finally, with the promise of a spherized basis, it is also worth asking what other non-Taylor Series basis can be concocted to help advance the applicability of mesh-free methods.

References

[1] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering*, 139(14):3 – 47, 1996.

[2] Gary A Dilts. Moving-least-squares-particle hydrodynamics - i. consistency and stability. *International Journal for Numerical Methods in Engineering*, 44(8):1115–1155, 1999.

[3] Gary A Dilts. Moving least-squares particle hydrodynamics ii: conservation and boundaries. *International Journal for Numerical Methods in Engineering*, 48(10):1503–1524, 2000.

[4] Gary A Dilts. A convergent method for shock hydrodynamics. Technical report LA-UR-07-6844, Los Alamos National Laboratory, October 2007.

[5] Gary A Dilts, Aamer Haque, and John Wallin. Tuned local regression estimators for the numerical solution of differential equations. In *Meshfree Methods for Partial Differential Equations*, pages 87–104. Springer, 2003.

[6] Jianqing Fan, I Gijbels, Tien-Chung Hu, and Li-Shan Huang. *An asymptotic study of variable bandwidth selection for local polynomial regression with application to density estimation*. Department of Statistics [University of North Carolina at Chapel Hill], 1993.

[7] Jianqing Fan and Irene Gijbels. *Local polynomial modelling and its applications: monographs on statistics and applied probability 66*, volume 66. CRC Press, 1996.

[8] Robert A Gingold and Joseph J Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

[9] Shaofan Li and Wing Kam Liu. *Meshfree particle methods*. Springer Science & Business Media, 2007.

[10] Gui-Rong Liu and Moubin B Liu. *Smoothed particle hydrodynamics: a meshfree particle method*. World Scientific, 2003.

[11] Richard M Lloyd. Physics of direct hit and near miss warhead technology. *Progress in astronautics and aeronautics*, 2001.

[12] Clive Loader. *Local regression and likelihood*. Springer Science & Business Media, 2006.

[13] Leon B Lucy. A numerical approach to the testing of the fission hypothesis. *The astronomical journal*, 82:1013–1024, 1977.

[14] Murray Rosenblatt et al. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.

[15] David Ruppert and Matthew P Wand. Multivariate locally weighted least squares regression. *The annals of statistics*, pages 1346–1370, 1994.

Turbulence Modeling with BHR in OpenFOAM

Team Members
Justin Brown and Lois Sarno-Smith

Mentors
Daniel Israel and Nick Denissen

Abstract

We have implemented a set of variable density turbulence equations using the BHR model into the open source computational fluid dynamics code OpenFOAM. We have compared the results from axisymmetric calculations with OpenFOAM to experimental results from the Turbulent Mixing Tunnel and with OpenFOAM results using the k-ε turbulence model. We find some qualitative agreement between these results but note that a full study will be needed before

Introduction

In many contexts, modeling all the physical scales of a hydrodynamical problem remains impossible with present computers. In this light, algorithms for modeling turbulence in a computationally efficient manner have arisen; however, these algorithms often must be tailored to the problems of interest. Here, we focus on one model for incompressible systems with two miscible components of different densities.

In geophysical and astrophysical systems, fluids with multiple components of different densities can play substantial roles. Certainly, the effects of variable density turbulence are critical to modeling the results of inertial confinement fusion, astrophysical jets, and fuel injection systems, to name a few. In order to attempt a Reynolds-Averaged Navier Stokes (RANS) formulation of such problems, we need to include additional terms that arise due to these density changes, even in a Boussinesq limit. The BHR equations were originally derived to address these concerns.[2] These equations have undergone substantial changes over the past two decades.[6–8]

In order to complete the closures introduced in the above citations, the models are calibrated to fluid dynamics problems with known solutions. For example, it is common to calibrate the parameters according to the Rayleigh-Taylor, Richtmyer-Meshkov, and Kelvin-Helmholtz problems.[1] Once the calbration is complete, we can validate the models against physical experiments of variable density turbulence.

In this report, it is our goal to attempt to reproduce the experimental results of the turbulent mixing tunnel experiments.[5] This experimental setup is that of a vertical subsonic wind tunnel filled with air traveling downward into which a jet of heavier material (either air with acetone or sulfur hexaflouride) is injected from the top. Simultaneous two-dimensional velocity and density fields can be measured at four measurement windows along the jet, and it is with these observations that we can compare the results of our turbulence models.

Governing Equations

Exact Equations

The governing equations for an incompressible, two-phase fluid are as follows:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0, \tag{1}$$

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_j u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \rho g_i + \frac{\partial \tau_{ij}}{\partial x_j}, \tag{2}$$

$$\frac{\partial \rho c_i}{\partial t} + \frac{\partial \rho u_j c_i}{\partial x_i} = \frac{\partial}{\partial x_i} \rho D \frac{\partial c_i}{\partial x_i}, \tag{3}$$

where ρ is the density, u_i is the i-component of the velocity, p is the pressure, g_i is the i-component of the gravitational acceleration, τ_{ij} is the stress tensor, c_i is the mass fraction of the ith species, and D is the molecular diffusion coefficient. Einstein notation is used to simplify the expression. For this system, we will assume that there are only two species, c_1 and c_2 such that $c_1 + c_2 = 1$. Because each individual phase is itself incompressible, we associate a constant, uniform density with each phase, ρ_1 and ρ_2, and describe them jointly using Amagat's Law

$$\frac{1}{\rho} = \frac{c_1}{\rho_1} + \frac{c_2}{\rho_2} = c_1\left(\frac{1}{\rho_1} - \frac{1}{\rho_2}\right) + \frac{1}{\rho_2}. \tag{4}$$

We can define a volume fraction α_i of the ith species as $\overline{\rho}\tilde{c}_i = \rho_1\overline{\alpha_i}$ such that

$$\rho = \rho_1\alpha_1 + \rho_2\alpha_2. \tag{5}$$

We will be evolving α_i rather that c_i for numerical convenience.

By combining Equation 4 with Equations 1 and 3, we arrive at a velocity constraint to ensure that the flow remain incompressible:

$$\frac{\partial u_i}{\partial x_i} = -\frac{\partial}{\partial x_i}\frac{D}{\rho}\frac{\partial \rho}{\partial x_i}. \tag{6}$$

Turbulence Equations

To construct the turbulent form of the above equations, we decompose the equations into their average and turbulent parts, using either a traditional average,

$$\rho = \overline{\rho} + \rho', \tag{7}$$

where $\overline{\rho'} = 0$, or using a Favre average,

$$u_i = \tilde{u}_i + u_i'', \tag{8}$$

where $\tilde{u}_i = \overline{\rho u_i}/\overline{\rho}$. Either decomposition is formally correct, but in variable density systems, it is analytically convenient to write all decompositions save density and pressure in the Favre sense.

A full derivation of the turbulent equations is beyond the scope of this report, so the authors recommend the work of Besnard et al. (1992).[2] The exact form of Equations 1–3 are as follows:

$$\frac{\partial\overline{\rho}}{\partial t} + \frac{\partial\overline{\rho}\tilde{u}_i}{\partial x_i} = 0, \tag{9}$$

$$\frac{\partial\overline{\rho}\tilde{u}_i}{\partial t} + \frac{\partial\overline{\rho}\tilde{u}_j\tilde{u}_i}{\partial x_j} + \frac{\partial R_{ij}}{\partial x_j} = -\frac{\partial\overline{p}}{\partial x_i} + \overline{\rho}g_i + \frac{\partial\overline{\tau_{ij}}}{\partial x_j}, \tag{10}$$

$$\frac{\partial\overline{\rho}\tilde{c}_i}{\partial t} + \frac{\partial\overline{\rho}\tilde{u}_j\tilde{c}_i}{\partial x_j} + \frac{\partial\overline{\rho u_j'' c_i''}}{\partial x_j} = \frac{\partial}{\partial x_i}\overline{\rho}D\frac{\partial\tilde{c}_i}{\partial x_i} + \frac{\partial}{\partial x_i}D\overline{\rho\frac{\partial c_i''}{\partial x_i}}, \tag{11}$$

where $R_{ij} = \overline{\rho u_i'' u_j''}$ is the Reynolds stress tensor. To evaluate this system, we will need to develop closures for any averages of the turbulent terms that remain.

Noting that we can convert from c_i to α_i by setting $\overline{\rho}\tilde{c}_i = \rho_1\overline{\alpha_i}$, we can express the species equation as

$$\frac{\partial\overline{\alpha_i}}{\partial t} + \frac{\partial\tilde{u}_j\overline{\alpha_i}}{\partial x_j} + \frac{1}{\rho_1}\frac{\partial\overline{\rho u_j'' c_i''}}{\partial x_j} = \frac{\partial}{\partial x_i}\overline{\rho}D\frac{\partial}{\partial x_i}\frac{\overline{\alpha_i}}{\overline{\rho}} + \frac{1}{\rho_1}\frac{\partial}{\partial x_i}D\overline{\rho\frac{\partial c_i''}{\partial x_i}} \tag{12}$$

We can also derive a velocity constraint by the same method as above, assuming that Equation 4 holds for the averages:

$$\frac{\partial \tilde{u}_i}{\partial x_i} = -\frac{\partial}{\partial x_i}\frac{D}{\overline{\rho}}\frac{\partial \overline{\rho}}{\partial x_i} - \left(\frac{1}{\rho_1} - \frac{1}{\rho_2}\right)\left(\frac{\partial \overline{\rho u_i'' c_i''}}{\partial x_i} - \frac{\partial}{\partial x_i}\overline{D\rho\frac{\partial c_i''}{\partial x_i}}\right). \tag{13}$$

This equation replaces Equation 9, just like its counterpart from the exact equations; however, this equation also requires closures for the turbulent averages

Reynolds Stress and Turbulent Species Diffusion

The Reynolds stress is modeled using a gradient diffusion approximation:

$$R_{ij} = -\frac{2}{3}\overline{\rho}k + \overline{\rho}\nu_{\mathrm{t}}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial \tilde{u}_k}{\partial x_k}\right), \tag{14}$$

where $\nu_{\mathrm{t}} = S\sqrt{k}$, S is the turbulent length scale, and k is the kinetic energy associated with the turbulence. To truly calculate this quantity, we need to generate equations for S and k.

To close Equations 11 and 13, we let

$$\frac{\partial}{\partial x_i}\overline{D\rho\frac{\partial c_i''}{\partial x_i}} - \frac{\partial \overline{\rho u_j'' c_i''}}{\partial x_j} = \frac{\partial}{\partial x_i}\overline{\rho}\frac{\nu_{\mathrm{t}}}{\sigma_\alpha}\frac{\partial \tilde{c}_i}{\partial x_i} = \rho_1\frac{\partial}{\partial x_i}\overline{\rho}\frac{\nu_{\mathrm{t}}}{\sigma_\alpha}\frac{\partial}{\partial x_i}\frac{\overline{\alpha_i}}{\overline{\rho}}, \tag{15}$$

where σ_α is a parameter of the models.[2] Thus, Equation 11 becomes (if we rewrite the mass fraction in terms of the volume fraction)

$$\frac{\partial \overline{\alpha_i}}{\partial t} + \frac{\partial \tilde{u}_j\overline{\alpha_i}}{\partial x_j} = \frac{\partial}{\partial x_i}\overline{\rho}\left(D + \frac{\nu_{\mathrm{t}}}{\sigma_\alpha}\right)\frac{\partial}{\partial x_i}\frac{\overline{\alpha_i}}{\overline{\rho}}, \tag{16}$$

and Equation 13 becomes

$$\frac{\partial \tilde{u}_i}{\partial x_i} = -\frac{\partial}{\partial x_i}\frac{(D + \sigma_\alpha \nu_{\mathrm{t}})}{\overline{\rho}}\frac{\partial \overline{\rho}}{\partial x_i}. \tag{17}$$

Kinetic Energy Equation

The kinetic energy equation is derived by taking the second moment of Equation 10 and combining it with Equation 2. Again, the derivation of the exact form has been done.[2] The closures have also been supplied elsewhere.[1] Thus, we only provide the results here:

$$\frac{\partial \overline{\rho}k}{\partial t} + \frac{\partial \overline{\rho}\tilde{u}_i k}{\partial x_i} = a_i\frac{\partial p}{\partial x_i} - \frac{R_{ij}}{\overline{\rho}}\frac{\partial \tilde{u}_j}{\partial x_i} + \frac{\partial}{\partial x_i}\overline{\rho}\frac{\nu_{\mathrm{t}}}{\sigma_k}\frac{\partial k}{\partial x_i} - \overline{\rho}\frac{k^{\frac{3}{2}}}{S}, \tag{18}$$

where $a_i = -\overline{u_i''}$ is the mass flux. To close this equation, we now need to add another equation for a_i.

Turbulent Length Scale Equation

The turbulent length scale equation is derived by taking the form of Equation 18 and applying it to the turbulent energy dissipation, ε. The turbulent length scale has been defined as $S = k^{3/2}/\varepsilon$ and rework the turbulent energy dissipation equation into the following:

$$\frac{\partial \overline{\rho} S}{\partial t} + \frac{\partial \overline{\rho} \tilde{u}_i S}{\partial x_i} = \frac{\partial}{\partial x_i} \overline{\rho} \frac{\nu_t}{\sigma_\varepsilon} \frac{\partial S}{\partial x_i} + \frac{S}{k} \left[\left(\frac{3}{2} - C_4 \right) a_i \frac{\partial \overline{p}}{\partial x_i} - \left(\frac{3}{2} - C_1 \right) R_{ij} \frac{\partial u_j}{\partial x_i} \right], \tag{19}$$

where σ_ε, C_1, and C_4 are parameters of the model.[1]

Mass Flux Equation

The exact equation for the turbulent mass flux is also derived by looking at second moments.[2] We present their closed version here for completeness:

$$\frac{\partial \overline{\rho} a_i}{\partial t} + \frac{\partial \overline{\rho} \tilde{u}_j a_i}{\partial x_j} = b \frac{\partial p}{\partial x_i} - \frac{R_{ij}}{\overline{\rho}} \frac{\partial \overline{\rho}}{\partial x_j} + \overline{\rho} \frac{\partial a_i a_j}{\partial x_j} - \overline{\rho} a_j \frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial}{\partial x_j} \overline{\rho} \frac{\nu_t}{\sigma_a} \frac{\partial a_i}{\partial x_j} - C_{a,1} \overline{\rho} \frac{\sqrt{k}}{S} a_i, \tag{20}$$

where $b = \overline{\rho} \overline{1/\rho} - 1$ is the correlation of the turbulent perturbations of density and specific volume. A fourth closure equation can be derived for this correlation. Here, we choose to use the algebraic model from Banerjee et al. (2010):[1]

$$b = C_b \frac{\overline{\alpha_1}\, \overline{\alpha_2} \left(\rho_1 - \rho_2 \right)^2}{\rho_1 \rho_2}, \tag{21}$$

where C_b is a parameter of the model, which we set to 0.1.

This completely closes the system.

Verification

With some effort, a self-similar solution to these equations can be derived. This form of verification has been used frequently in the field, for example, see the work of Bradbury et al. (1965).[3] The goal of this form of analysis is to find a solution for which a change of dependent variables produces a static solution. For simplicity, we will reduce the equations to a one-dimensional system:

$$\frac{\partial \rho k}{\partial t} - \frac{\partial}{\partial x_1} \rho \frac{\nu_t}{\sigma_k} \frac{\partial}{\partial x_1} k = a_i \frac{\partial}{\partial x_1} p - \rho \frac{k}{S} \sqrt{k} \tag{22}$$

$$\frac{\partial \rho S}{\partial t} - \frac{\partial}{\partial x_1} \frac{\rho \nu_t}{\sigma_\varepsilon} \frac{\partial}{\partial x_1} S = \frac{S}{k} \left(\frac{3}{2} - C_4 \right) a_1 \frac{\partial}{\partial x_1} p - \left(\frac{3}{2} - C_2 \right) \rho \sqrt{k} \tag{23}$$

$$\frac{\partial \rho a_i}{\partial t} - \frac{\partial}{\partial x_1} \frac{\rho \nu_t}{\sigma_a} \frac{\partial}{\partial x_1} a_i = b \frac{\partial}{\partial x_1} p - C_{a,1} \rho \frac{\sqrt{k}}{S} a_i \tag{24}$$

$$\frac{\partial \rho \alpha_i}{\partial t} - \frac{\partial}{\partial x_1} \rho \frac{\nu_t}{\sigma_\alpha} \frac{\partial}{\partial x_1} \frac{\alpha_i}{\rho} = 0 \tag{25}$$

If we assume a self-similar solution for the quantities in the above equations as follows,

$$z = \alpha\eta \mathrm{At}|g_1|t^2, \tag{26}$$

$$k = \hat{k}\left(\mathrm{At}|g_1|t\right)^2\left(1-\eta^2\right), \tag{27}$$

$$S = \hat{S}\left(\mathrm{At}|g_1|t^2\right)\sqrt{1-\eta^2}, \tag{28}$$

$$a_1 = \hat{a}_i \mathrm{At}|g_1|t\left(1-\eta^2\right), \tag{29}$$

$$\alpha_i = \frac{1}{2} + \hat{\alpha}_i\eta, \tag{30}$$

we find that we can achieve a self-similar solution if we match powers of η to ensure that the solution holds for all space.

We assume the fluid is Boussinesq and hydrostatic, so $\frac{\partial p}{\partial x_1} = \rho g_i$ and we divide each term by ρ. The species equation becomes

$$\alpha^2 = \frac{C_\mu \hat{S}\sqrt{\hat{k}}}{\sigma_\alpha}. \tag{31}$$

The kinetic energy equation then becomes

$$2\hat{k}\left(1+\eta^2\right) = -\frac{2}{\sigma_\alpha}\left(1-3\eta^2\right)\hat{k} + \frac{\hat{a}_1 g_1}{|g_1|}\left(1-\eta^2\right) - \frac{\hat{k}^{\frac{3}{2}}}{\hat{S}}, \tag{32}$$

from which we can conclude, by matching powers of η, that $\sigma_\alpha = 1$ and $\hat{a}_1 = \frac{4\hat{k}\hat{S}+\hat{k}^{\frac{3}{2}}}{S g_1/|g_1|}$.

With these substitutions, the S equation reduces to

$$-2\hat{S} = \frac{1}{\sigma_\varepsilon}\left(1-2\eta^2\right) + C_2\sqrt{\hat{k}}\left(1-\eta^2\right) - C_4\left(4\hat{S}+\sqrt{\hat{k}}\right)\left(1-\eta^2\right) + 2\hat{S}\left(1-3\eta^2\right), \tag{33}$$

by matching powers of η we conclude that $\sigma_\varepsilon = 1/2$ and $\hat{S} = -\left(C_4 - C_2\right)\frac{\sqrt{\hat{k}}}{2(2C_4-1)}$.

The last equation then is just

$$\hat{k} = \frac{\left(C_2 - C_4\right)^2}{2\left(2C_2-1\right)\left(3C_2+4C_4C_{a,1}-3C_4-2C_{a,1}\right)}, \tag{34}$$

as long as $\sigma_a = 1$.

These expressions can then be solved for the traditional values of the turbulence coefficients.[7] We find that the value of $\hat{k}$, $\hat{S}$, $\hat{a}_i$, and α must be 0.051, 0.941, -0.214, and 0.243, respectively.

Testing OpenFOAM against the Analytic Solution

We have sought to compare our numerical results against this analytic solution for verification purposes of our implementation of the BHR model. We have run this one-dimensional test with time steps in factors of ten from 0.1 seconds to 10^{-4} seconds and with a number of gridpoints across the 2.0 meter domain of 25 to 1600 in factors of two. We find that our results compare favorably with the analytical results as we converge in both time and space, as can be seen in Figure 1, where we plot the L1 relative error. We have discovered a small issue with these in that the centroid of the solutions, which should remain fixed in space, appears to shift slightly. In the attached figure, we have corrected the centroid of the simulation results to match the analytic centroid to illustrate that the shape of the solution is converging, even though this need to correct the offset does concern us. As we converge in time and space, the offset itself converges to a value of 10^{-3}, which is smaller than any resolution we have run thus far.

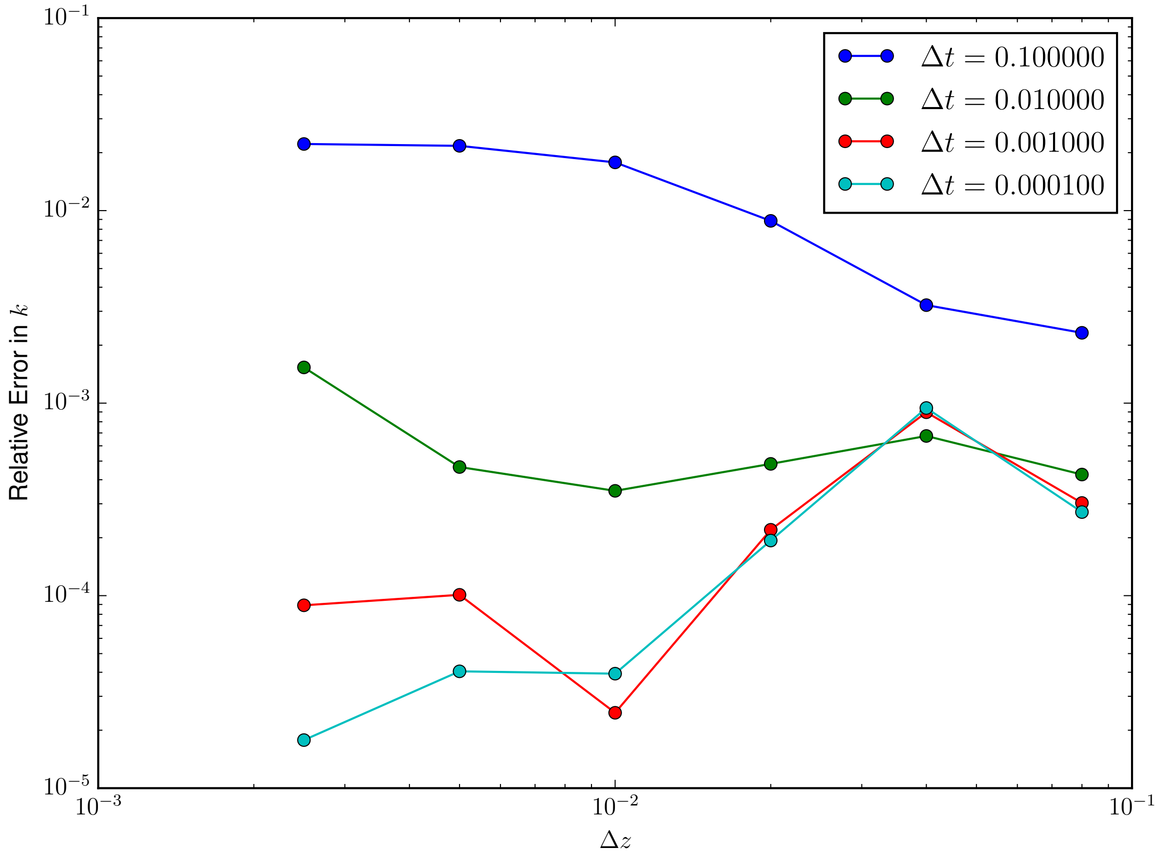


Figure 1: The L1 relative error of several Rayleigh-Taylor simulations, described in Section . We find that the simulations do converge with increasing temporal (color) and spatial (abscissa) resolution. In this figure, we have corrected the centroid of the simulations to match that of the analytic solution in order to compare results. This offset is less than a grid cell, but we are somewhat concerned about the need to correct this. The non-monotonicity of this plot is also concerning.

Simulation Setup

We have attempted to reproduce the experiment tubulent mixing tunnel experiments by using several different numerical configurations. We use a jet diameter of 1.0 centimeter in a chamber

0.5 meters wide and 1.0 meter long.[5] The Atwood number matches that of the air-acetone experiment (0.11), and we use the velocities they observe in their experiments (26 meters per second in the jet and 1.4 meters per second in the freestream). We use near-zero values for the turbulent quantities in the freestream and initialize the jet with 20.0 square meters per square second and a turbulent length scale of 3.0 centimeters to try to match the experimental results to some degree. We run this in two geometries with cell widths of 1.0 centimeter for a plane-parallel setup and an axisymmetric setup.

We run the jet both with our implemented BHR equations and with the standard k-ε model already present in OpenFOAM in an axisymmetric geometry. We are aware that a correction should be applied to the turbulence parameters for the axisymmetric geometry to slow the rate of spreading, but we have not yet added these, so we seek only a qualitative comparison.[4] We adopt a set of turbulence parameters that have been gathered from comparison with self-similar solutions.[1]

By analyzing the trends of the profiles along the centerline of the jet, we can see that our implementation of the BHR equations appears to be more diffusive than the built-in k-ε equations. In particular, this can be seen by looking at the velocity profile at the first grid cell: both simulations have the same initial velocity, but we see that the BHR velocity decays by almost a fifth of its initial value instantly. This is likely due in large part to the much larger value of k near the jet inlet, but the exact reason for this initially large value of k is still unknown. The later behavior does match our implementation better, but it's unclear how meaningful this is since the initial jet disagrees substantially with the experimental results. To better match the inlet values will require much more simulations with initial conditions and possibly additional calibrations of the BHR equation parameters to better reproduce cases for which the density gradient is mostly perpendicular to the flow of gravity.

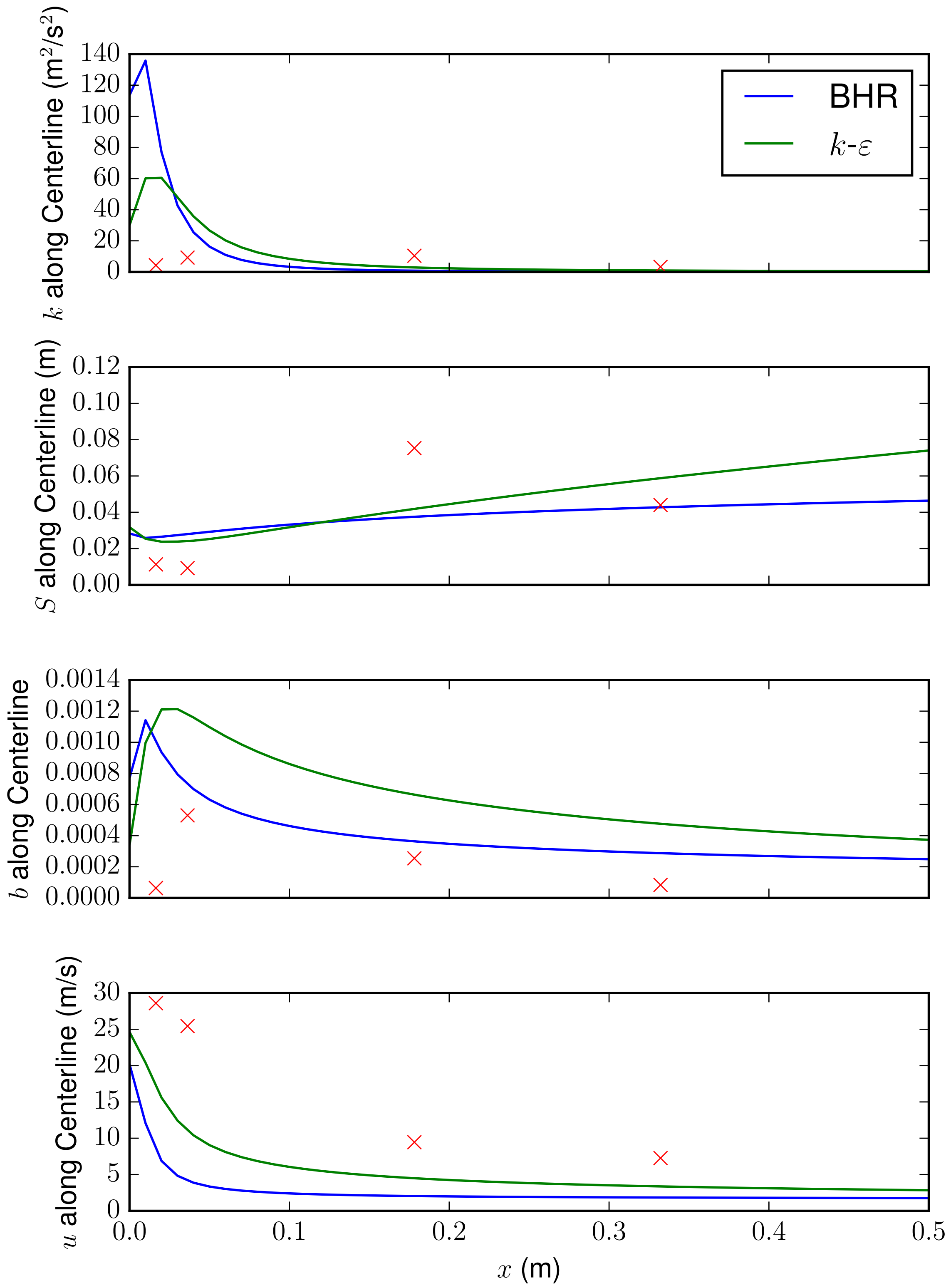


Figure 2: We show here the profiles measured along the jet center of *k*, *S*, *b*, and *u*. The results from the BHR equations are in blue, those from *k*-*ε* are in green, and the experimental values are in red crosses. The jet progresses in the positive *x* direction.

Simulation Results

In our comparison of OpenFOAM k-ε and BHR turbulence models to the experimental data results, it is important to note that we are using the axi-symmetric case. Therefore, we expect our results to be at least some degree close to the experimental outcomes.

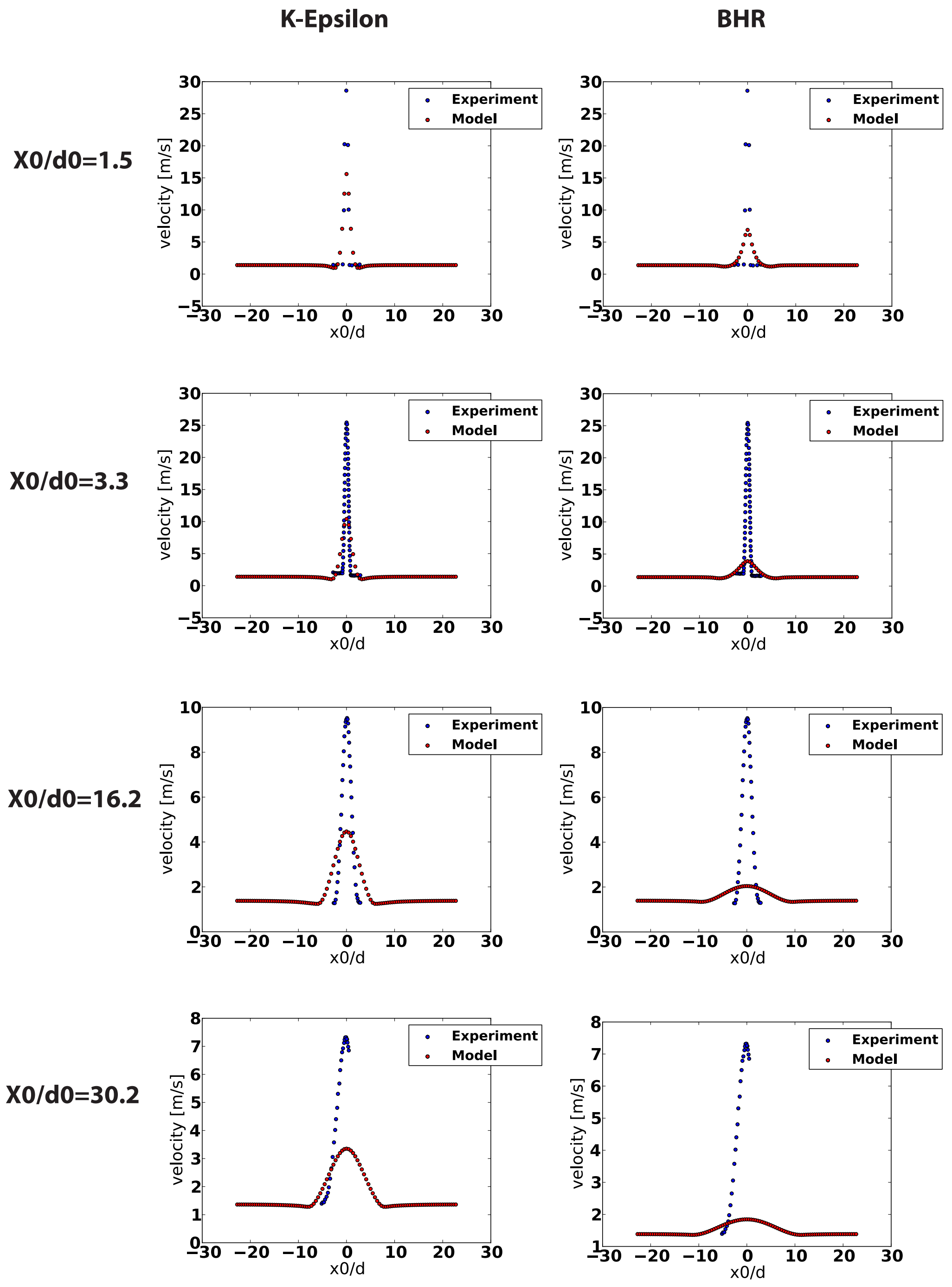


Figure 3: Velocity comparison of BHR and k-ε models in OpenFOAM to experimental results from the TMT at 4 specified locations.

Figure 3 shows the velocity output from OpenFOAM using the k-ε and BHR turbulence models compared to experimental results from the TMT. At x0/d0=1.5, the differences in the models versus the experiment are quite large, with the experimental values near zero and the Models have a non-zero peak at x0/d0=0; however, the experimental jet profile appears somewhat artificially peaked. At x0/d0=3.3, the experiment and model results for both BHR and k-ε are much closer; both the K-epsilon and BHR models capture the max velocity of the flow well. Further downstream at x0/d0=16.2, small velocity differences between the turbulence models begin to appear. The BHR turbulence module produces a wider velocity distribution around the jet with smoother wings far from the jet's center. The k-ε model, on the other hand, has a narrower velocity distribution, which is closer to the experimental results.

In Figure 3, it is important to note that for BHR and the k-ε models, the spreading rate of the velocity is too fast. The centerline velocity decay rate is too fast also for the two turbulence models. For the BHR model, we hypothesize that this is from the additional production mechanism from the buoyancy term.

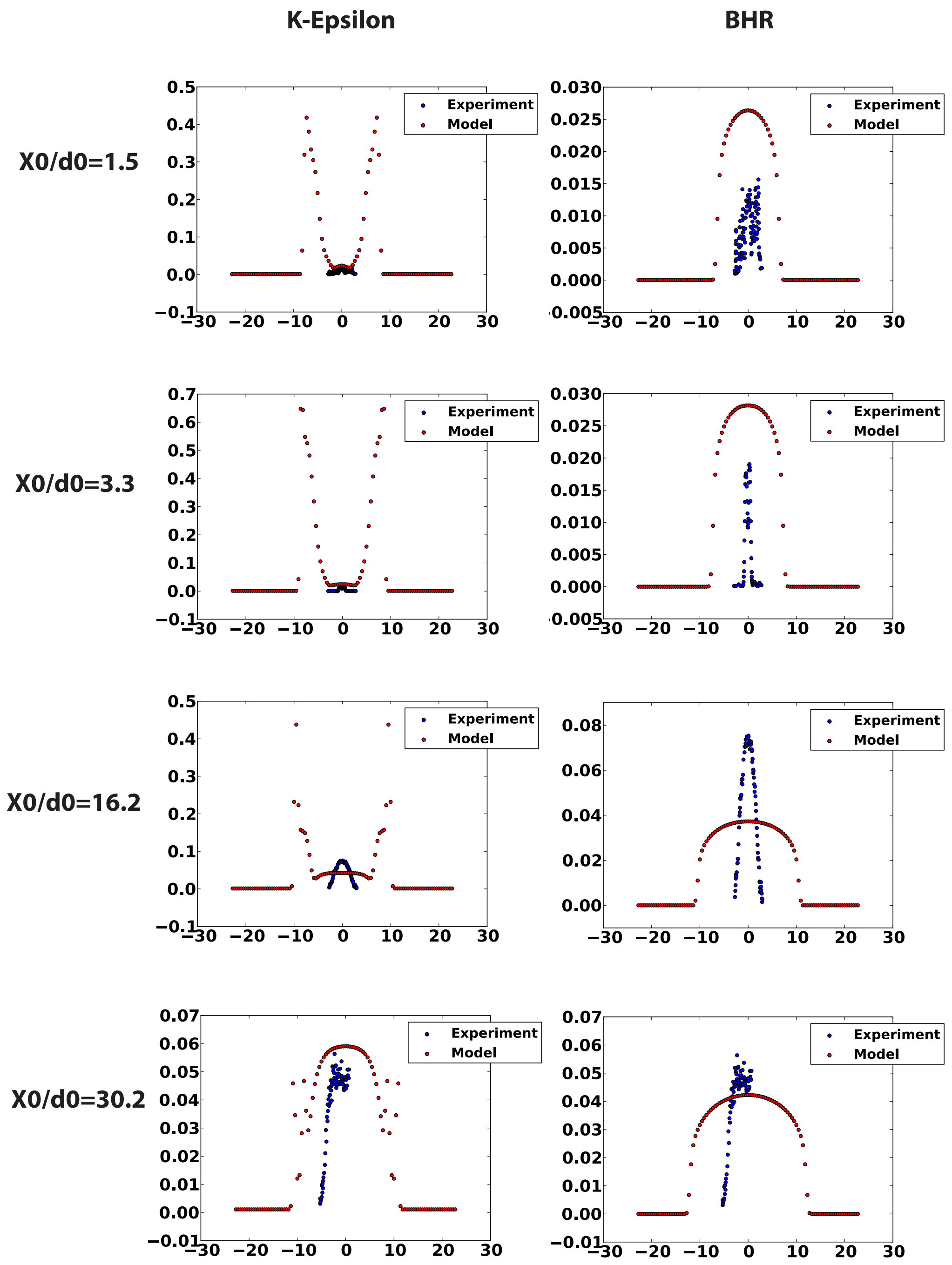


Figure 4: Turbulent length scale (S) comparison of BHR and k-ε models in OpenFOAM to experimental results from the TMT at 4 specified locations.

Figure 4 shows the turbulent length scale output from the OpenFOAM k-ε and BHR turbulence models. At x0/d0=1.5, BHR performs much better than the k-ε model. The BHR output peaks around the jet center, but is approximately a factor of 1.5 off from the experiment. On the other hand, the k-ε model produces turbulent length scales that appear morphologically different from the experimental values. At x0/d0=3.3, the result is similar; k-ε has off center peaks that are an order of magnitude greater than the experimental data.

At x0/d0=16.2, k-ε performs better, producing output on the same order of magnitude as the experimental turbulent length scales. However, the shape for the k-ε modeled S is not correct. BHR turbulence does a better job, but the model's distribution is still too wide and flat. Lastly, at x0/d0=30.2, the experimental data does not particularly match either of the turbulent length scales produced by either model.

For S in Figure 4, the shape for both of the turbulence models is not quite right. The k-ε and BHR models have too wide and flat of distributions compared to the experimental results. In the k-ε model, the peaked wings are concerning at first appearance, but they are likely an artifact of the k-ε boundaries. Thus, for our analysis, they should be considered inconsequential and not a cause for alarm.

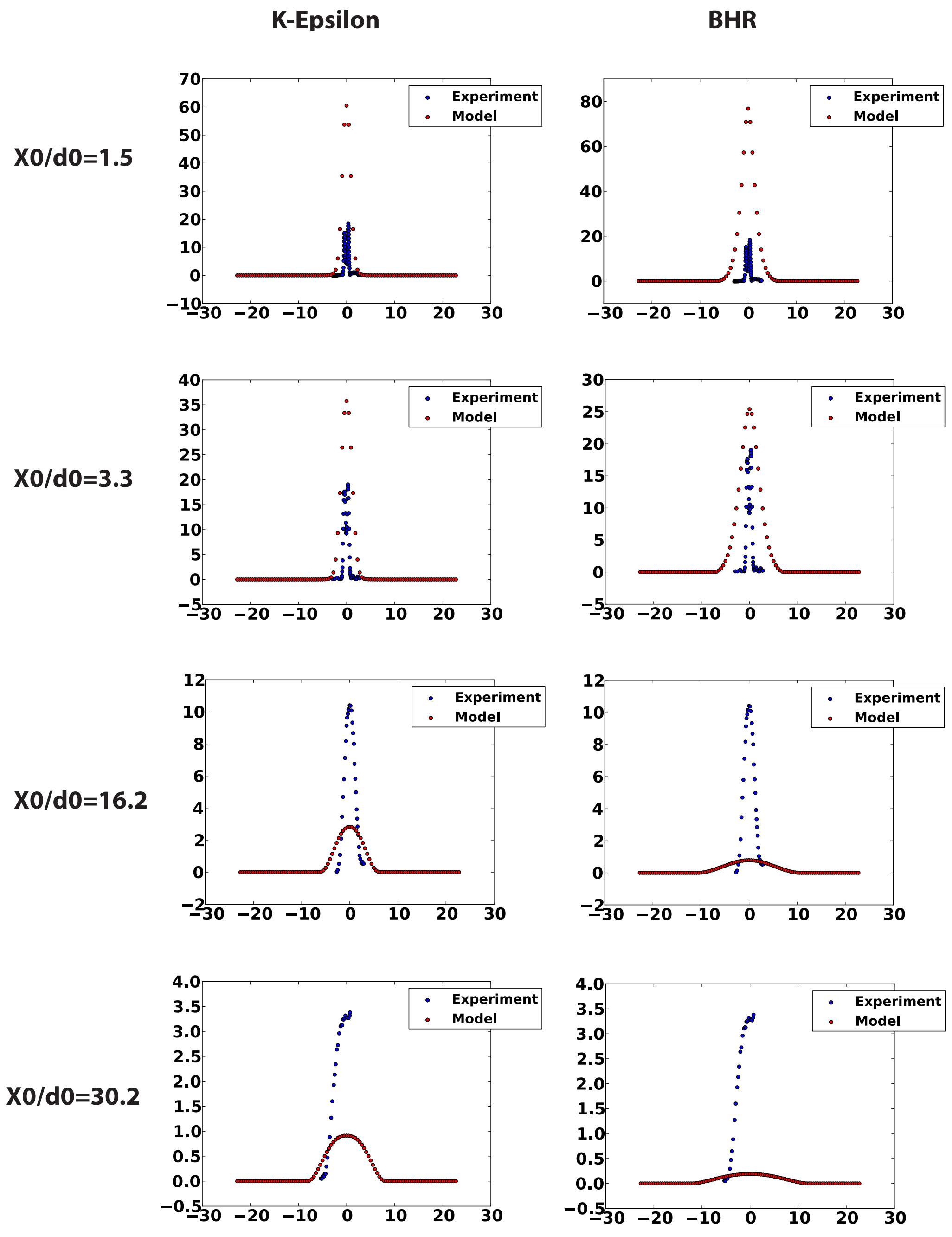


Figure 5: Turbulent kinetic energy (k) comparison of BHR and k-ε models in OpenFOAM to experimental results from the TMT at 4 specified locations.

Figure 5 displays the turbulent kinetic energy output from OpenFOAM with the k-ε model and the BHR turbulence model. At x0/d0=1.5, k-ε and BHR are both approximately an order of magnitude off in peak amplitude from the experimentally determined k. Both models appear qualitatively similar to the experimental results in shape except near the first two stations, where the profile has two peaks. This is likely because both models appear somewhat too diffusive to reproduce these results. Overall, the BHR and k-ε distributions look reasonable and we can assess that the models are doing a sufficient job of depicting behavior of turbulence kinetic energy in the experiment. However, the spreading rate is too high for both of the models.

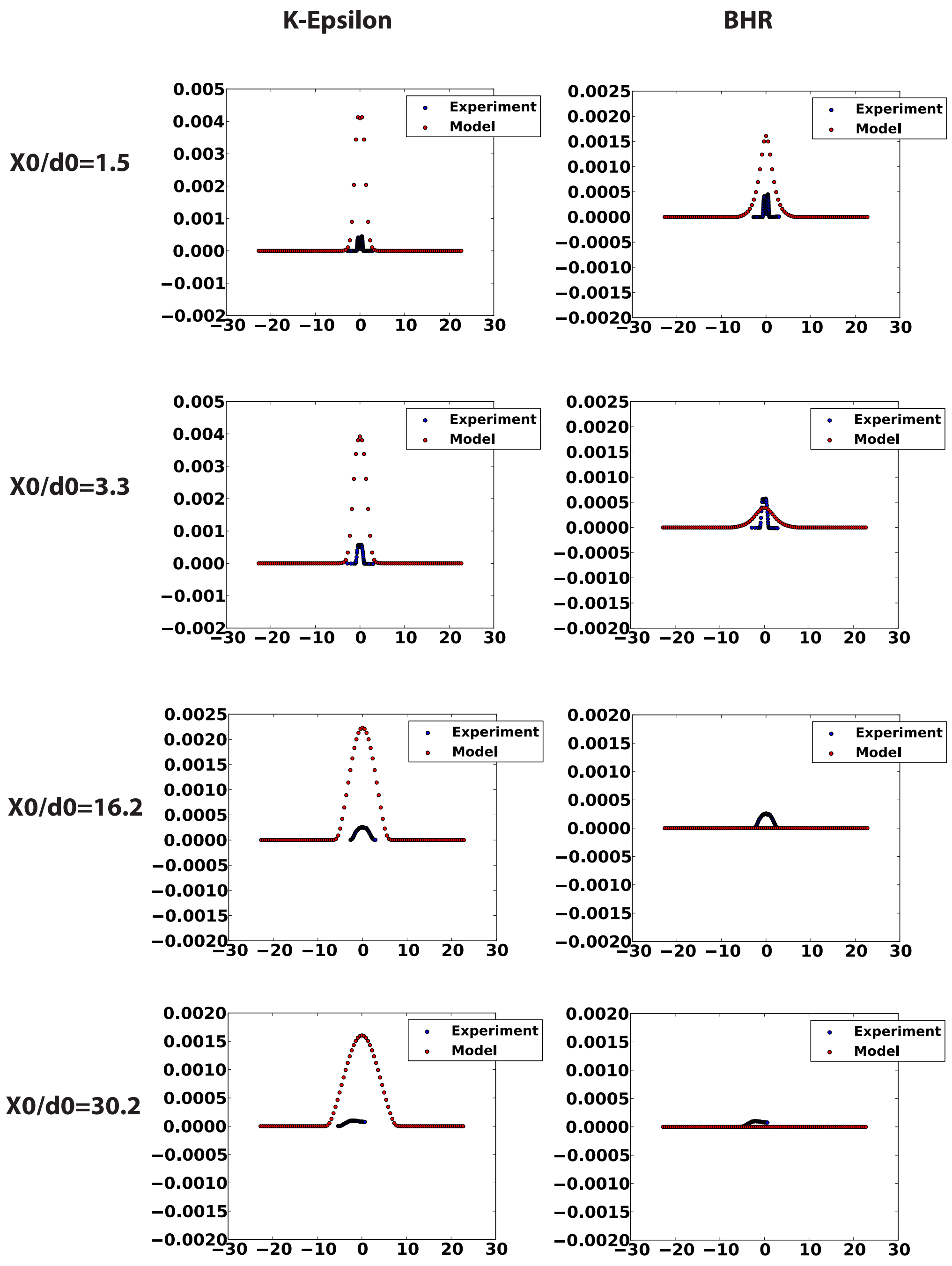


Figure 6: Degree of mixing of density (b) comparison of BHR and k-ϵ models in OpenFOAM to experimental results from the TMT at 4 specified locations.

The comparison of k-ϵ and BHR OpenFOAM output to experimental data of the degree of density mixing is shown in Figure 6. For this comparison, the primary things to note are that once again, the spreading rate of both of the models is too fast. In the case of the k-ϵ, the equations have no b explicitly, so we must derive b from an algebraic model as a function of the volume fraction, alpha. Therefore, we can assess that in the case of the k-ε model, alpha is controlling the decay rate of b. The BHR b decays much faster than the k-ε's b, which is something we should take into consideration for future work.

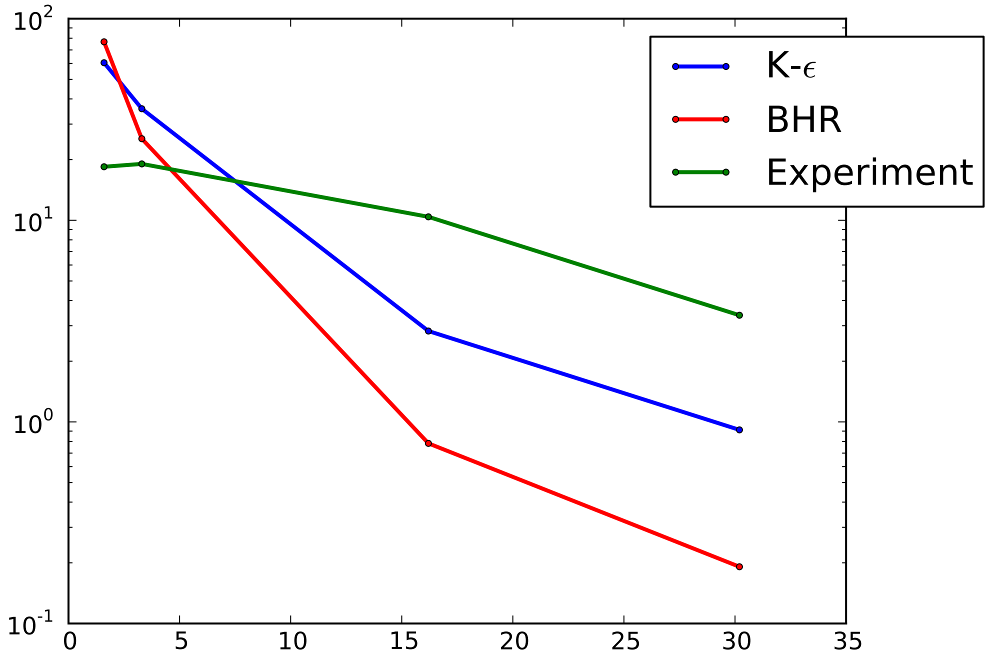


Figure 7: k-ε and BHR maximum turbulent kinetic energy (k) compared to experimental k maximums at x0/d0=1.5,3.3,16.2,30.2

Figure 7 shows the maximum turbulent kinetic energy at all four measurement locations specified by the experiment. The maximum k value shows how the turbulent kinetic energy evolves with time. In Figure 7, BHR and k-ε turbulence models produce k values that evolve together at different distances from the jet base. At first, the k-ε model produces a maximum turbulent kinetic energy slightly higher than that of the BHR model. However, further downstream from the jet, the k-ε model drops below that the BHR model. Both models underestimate the experiment by approximately a factor of five or so past x0/d0=16.2. It is also important to note that both models show a rapid decay in maximum turbulent kinetic energy whereas the experimental results show a much more gradual decline. The nearest point between the k maximum of the model is x0/d0=3.3.

Conclusions

For our project, we added BHR turbulence to OpenFOAM and compare the results with K-Epsilon turbulence in OpenFOAM and experimental results from the Turbulent Mixing Tunnel (TMT). Upon implementating the BHR model into OpenFOAM, we found that the analytical results converged in both time and space as expected, with a minimum error of 10^{-4}. Upon comparing OpenFOAM BHR and K-Epsilon turbulence models to experimental results, we found BHR generally performed better, particularly in modeling the turbulence kinetic energy and of the flow velocity. However, the degree of mixing of density (b) was over two orders of magnitude off due to the nature of using an analytic equation to solve for b instead of the full transport equations.

Since this is the product of a short summer project, we have merely scratched the surface of what still needs to be done with this work. Firstly, we should use the aforementioned full transport equations to describe b instead of an analytic equation. Also, we need to perform additional testing and verfication to ensure that the BHR turbulent model is behaving as it should within OpenFOAM. Lastly, a more thorough in depth comparison of experimental and simulated data should be run for completeness. After these tasks have been completed, using OpenFOAM BHR turbulence as a test bed for larger turbulence models, such as FLAG at LANL, may be an option.

References

[1] Arindam Banerjee, Robert a. Gore, and Malcolm J. Andrews. Development and validation of a turbulent-mix model for variable-density and compressible flows. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 82(4):1–14, 2010.

[2] Didier Besnard, Francis H. Harlow, Rick M. Rauenzahn, and Charles Zemach. Turbulence Transport Equations for Variable-Density Turbulence and Their Relationship to Two-Field Models. Technical report, 1992.

[3] L. J. S. Bradbury. The structure of a self-preserving turbulent plane jet. *Journal of Fluid Mechanics*, 23(01):31, 1965.

[4] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, Cambridge, 2000.

[5] Kathy Prestridge, John Charonko, and Nicolas Garcia. The Los Alamos National Laboratory Turbulent Mixing Tunnel. *Stockpile Stewardship Quarterly*, 4(4):9–11, 2014.

[6] Donald Leon Sandoval. *The dynamics of variable-density turbulence*. PhD thesis, 1995.

[7] John D. Schwarzkopf, Daniel Livescu, Robert a. Gore, Rick M. Rauenzahn, and J. Raymond Ristorcelli. Application of a second-moment closure model to mixing processes involving multicomponent miscible fluids. *Journal of Turbulence*, 12(December 2011):N49, 2011.

[8] K. Stalsberg-Zarling and R. A. Gore. The BHR2 turbulence model: incompressible isotropic decay, RayleighTaylor, KelvinHelmholtz and homogeneous variable density turbulence, Los Alamos National Laboratory. *LA-UR*, 2011.

Plasma Modeling with the Heterogeneous Multiscale Method

Team Members
Jacob Price and Gil Shohet

Mentors
Michael Murillo and Jeff Haack

Abstract

Many phenomena can be described at different levels of detail by varying the choice of physical model. High-fidelity microscopic models generally come with an increased computational cost compared to low-fidelity macroscopic models. It is often the case that we desire the detail of the microscopic model on macroscopic scales. The heterogeneous multiscale method (HMM) provides a computational and analytical link between disparate physical models. It can be used to hybridize physical models in a way that improves the accuracy of the macroscale model while avoiding much of the computational expense of the microscale model. Accurately modeling plasmas generally requires expensive microscale models, so exploiting scale disparities has the potential to provide significant computational benefits. Given an accurate relaxation parameter, the Bhatnagar-Gross-Krook (BGK) approximation is an effective and efficient kinetic model for plasmas that are near equilibrium. This parameter, however, is difficult to know *a priori*. Molecular dynamics offers a fully detailed model for ionic motion, but is computationally intensive. In this report, we present a proof of concept of HMM as a modeling method for plasmas. Simulations using the hybrid kinetic-molecular dynamic model are both more accurate than the kinetic model alone, and orders of magnitude more efficient than the molecular dynamics model alone.

Introduction

Multiscale modeling

Most mathematical and scientific problems contain phenomena with widely varying length and time scales of interest. Multiscale modeling aims to take advantage of this disparity to understand complex behavior at the necessary level of detail [11]. Multiphysics problems are a prototypical example of problems that are well-addressed by a multiscale approach. There is a hierarchy of physical models that allow systems to be modeled with varying degrees of detail, illustrated in Figure 1. Quantum mechanics (QM) provides a complete description of physical interactions, but is often computationally and analytically intractable for all but the simplest systems. From quantum mechanics, one can derive laws of molecular dynamics (MD) as a more tractable, but less detailed physical description than the wavefunctions of quantum mechanics. These laws of motion are governed by interparticle potentials. From molecular dynamics, kinetic theory (KT) can be derived as a statistical description of molecular motion. Here, collisional cross-sections, an abstract quantity that averages away the behavior of individual particles, governs the evolution of the system. An even coarser model, hydrodynamics, can be derived either directly from molecular dynamics or from kinetic theory. Here, equations of state, which contain less detail than cross-sections, determine system behavior. With each step up the hierarchy, physical detail is generally sacrificed for greater computational efficiency.

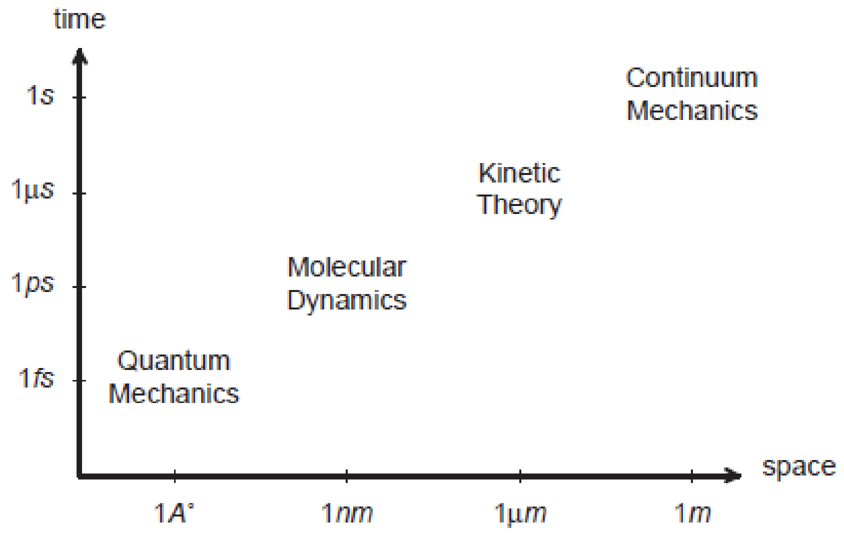


Figure 1: Multiscale model hierarchy [12]. As we progress up the hierarchy, the sizes of the spatial and temporal domains we can compute reasonably with a model increase, but the level of detail decreases. HMM seeks to construct a hybrid method that joins two models at different levels in the hierarchy.

Heterogeneous multiscale method

Unlike many multiscale models that use macroscale models to accelerate fully microscale simulations, the heterogeneous multiscale method (HMM) takes a top-down approach to multiphysics modeling [12]. Multiple levels of detail are combined such that the models are mathematically consistent. HMM focuses on the coarsest (macroscale) model because it can be efficiently computed over much larger time scales than the more detailed (microscale) model. The goal is to minimize the computational time spent in the fine regime, using the microscale model only when it is critical to the accuracy of the solution. The two regimes are connected by a *compression* operator, which compresses the necessary information in the microscale system into a consistent macroscopic description of the same system, and a *reconstruction* operator, which generates a microscale system that matches the properties of the macroscale [12]. The result is a hybrid simulation method that retains much of the computational speed of the coarse model with much of the accuracy of the fine model. This is essential when facing problems that have a large spatial or temporal region of interest, but which also include regions or aspects of interest that require microscopic detail. A full microscopic simulation would be prohibitively expensive, and a full macroscopic simulation would overlook crucial physical behavior in the domain. HMM is ideally suited to these types of problems.

Most HMM problems belong to one of two categories, which are outlined in Figure 2. *Type A* problems use the microscale model in isolated regions, such as material defects, interfaces, or shocks, where the macroscale model is insufficient to model the system. *Type B* problems exploit the microscale to provide an informed estimate of parameters that are necessary to close the macroscale system. A range of example applications of *Type A* and *Type B* problems is provided in [12]. See [6] for detailed solutions of *Type A* and *Type B* problems using a combination of hydrodynamics and molecular dynamics. The problem we are solving is of *Type B*; we use molecular dynamics simulations on the microscale to periodically provide an estimate of the relaxation parameter in our macroscale kinetic theory model. This general strategy is illustrated in Figure 3.

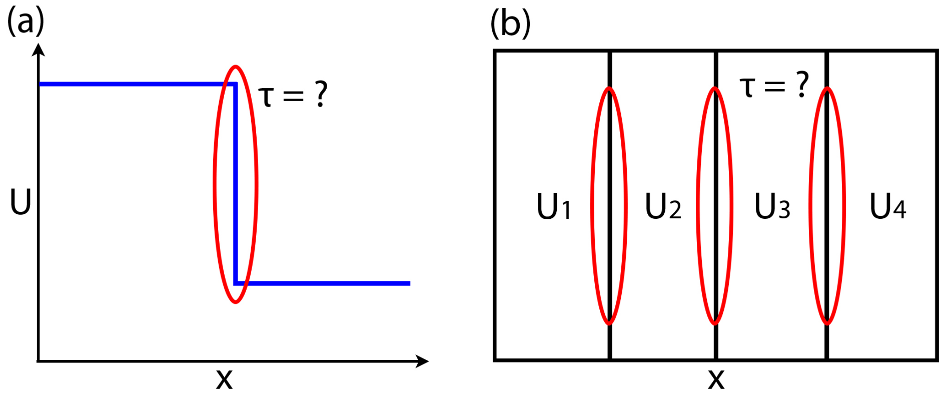


Figure 2: Let τ be some quantity, such as the stres tensor, that is required to compute fluxes. (a): *Type A* problem. There is an isolated discontinuity, for example a shock, in some quantity U. In the rest of the domain constitutive relations can be used to compute τ, but in the discontinuity region, a microscale model is used since constitutive relations are insufficient. (b): *Type B* problem. There is no sufficient constiutive model for τ, so τ is computed at all cell interfaces using a constrained microscale model.

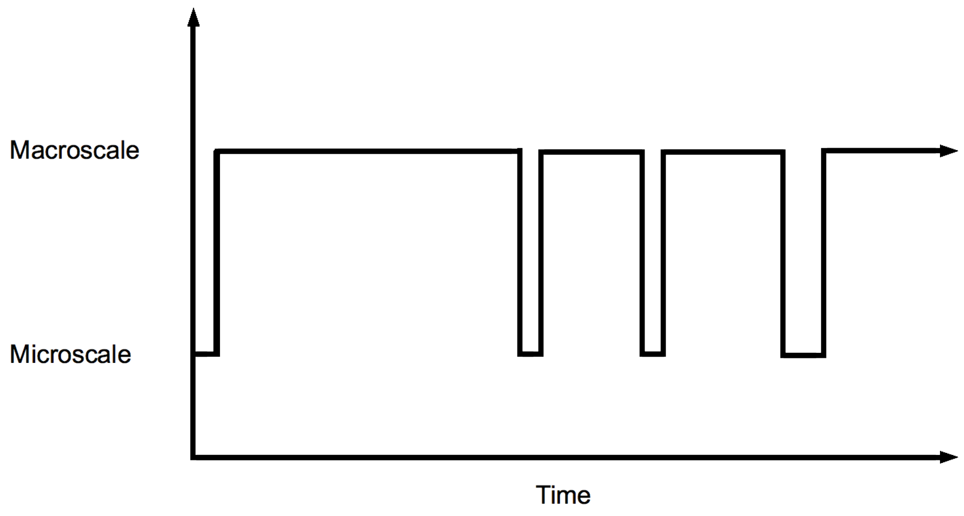


Figure 3: Illustration of the computational time spent in different computational regimes during an HMM simulation. In our system, the microscale is molecular dynamics simulation and the macroscale is kinetic theory simulation. Lines moving vertically upward correspond to the compression operator, lines moving vertically downward correspond to the reconstruction operator.

Plasmas

Plasmas are electrically neutral systems of electrons and ions in which each ion interacts with many neighbors, making collective effects critical [10]. Because opposite charges attract, each positive ion tends to be surrounded by a cloud of negatively charged background electrons which effectively "screen" the electrostatic field [2]. The length scale of this near-neighbor interaction is represented by the *Debye length* λ. This causes the potential around ions to fall rapidly, limiting the relevant interaction range between charged particles to a few Debye lengths. Examples of plasmas include astrophysical phenomena, such as stars and the interplanetary medium, terrestrial phenomena, such as lightning and aurorae, and technological constructs, such as plasma televisions and the materials used in fusion reactor research. Plasmas can exist at a wide range of temperatures and can exhibit a wide degree of coupling. We focus on plasmas that are moderately coupled, between the fully kinetic and fully hydrodynamic regimes.

Motivation

In this report we develop a hybrid multiphysics framework for modeling plasmas using the heterogeneous multiscale method (HMM). To model our system we combine the Bhatnagar-Gross-Krook (BGK) appoximation to the Boltzmann equation with molecular dynamics to model the time-evolution of a plasma system. This proof-of-principle model is designed to be as general and mathemetically rigorous as possible. Our goal is not to model any specific system, but rather to develop the framework to model a class of problems involving weakly to moderately coupled plamas. We choose this regime because very weakly coupled plasmas can be better modeled as ideal gases and highly coupled plasmas can be better modeled using hydrodynamic relations. In the moderately-coupled regime, kinetic effects are significant. This formulation is novel in two ways: to our knowledge this is the first HMM formulation to combine molecular dynamics with kinetic theory, and also the first attempt to model plasmas using HMM.

Model Formulation

Our model system is plasma in which the ions are confined to a two-dimensional plane on a periodic domain, shown in Figure 4. Periodic boundary conditions effectively approximate the behavior of a much larger domain, since ion interactions are screened by the background electrons. We seek to demonstrate multiscale plasma modeling as a proof-of-principle. The theoretical results are easily generalized to a three-dimensional system. However, the link between the kinetic and MD models does not depend on the dimensionality of the system, so we confine ourselves to a two-dimensional simulation for computational efficiency. We choose a screened potential because this avoids the additional complexity and computational cost of explicitly modeling electrons. This is a sufficient description of physical systems in which we are only interested in ion behavior and where the time scale of plasma interactions is much slower than the timescale of electron motion.

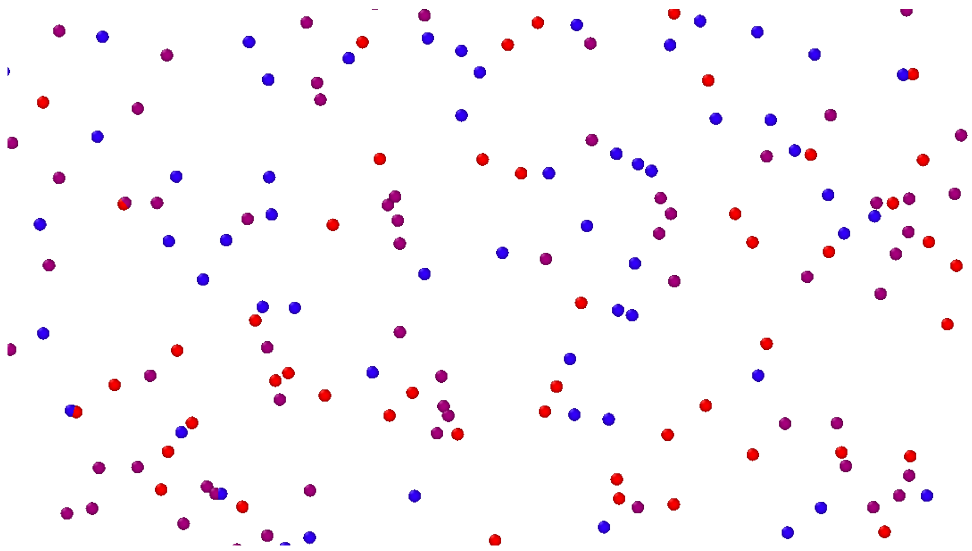


Figure 4: Ions are confined to a two-dimensional plane with periodic boundary conditions. Electrons are only modeled implicitly through the screened potential. There may be multiple ion species, denoted by the different colors.

There are physical examples of plasmas that are confined to a plane, such as *dusty plasmas*, which consist of large (millimeter to nanometer), highly charged particles. They can be found in the mesosphere of the earth, industrial processing, and laboratory experiments. In experiments, they can be confined to a plane by a balance of gravitational force and vertical electric force generated by the laboratory. Electrons, which weigh siginificantly less than the ions, are free to move in three dimensions, but the motion of the ions is confined to a flat plane [8].

To use HMM to simulate this system, we select the Bhatnagar-Gross-Krook (BGK) approximation of the Boltzmann equation as our coarse (macroscale) kinetic theory model and molecular dynamics as our fine (microscale) model. An advantage of the BGK model is that the quantities we must derive from the molecular dynamics simulations, the relaxation times τ_{kl}, are low-dimensional, depending only on position. This is in contrast to the collisional cross-sections needed to connect MD to a Boltzmann kinetic formulation or the two-particle correlation functions f_{kl} needed to connect MD to the BBGKY hierarchy, both of which are significantly more difficult to compute from MD. The quantity of interest in the BGK model is the distribution of each ion species. For ion species k, we represent this distribution by $f_k(\mathbf{r},\mathbf{v})$, which gives the expected number of particles of species k at position $\mathbf{r}=(x,y)$ with velocity $\mathbf{v}=(v_x,v_y)$. f_k is normalized such that integrating f_k over a region A in phase space gives the expected number of particles in A at time t. The partial differential equation governing this distribution is

$$\begin{aligned}
&\frac{\partial f_k}{\partial t}+\mathbf{v}\cdot\nabla_{\mathbf{r}}f_k+\frac{Z_k e}{m_k}\mathbf{E}(\mathbf{r},t)\cdot\nabla_{\mathbf{v}}f_k=\sum_l \frac{f_{kl}^{eq}-f_k}{\tau_{kl}}\\
&\mathbf{E}(\mathbf{r},t)=-\int\frac{1}{4\pi\varepsilon_0}\rho(\mathbf{r}',t)\nabla_{\mathbf{r}}\left(\frac{e^{-|\mathbf{r}-\mathbf{r}'|/\lambda}}{|\mathbf{r}-\mathbf{r}'|}\right)d\mathbf{r}'\\
&f_k(\mathbf{r},\mathbf{v},0)=f_k(\mathbf{r},\mathbf{v})\\
&\quad f_k(\mathbf{r},\mathbf{v},t)|_{x=0}=f_k(\mathbf{r},\mathbf{v},t)|_{x=L_x},\quad f_k(\mathbf{r},\mathbf{v},t)|_{y=0}=f_k(\mathbf{r},\mathbf{v},t)|_{y=L_y}
\end{aligned}\tag{1}$$

Here, Z_k is the charge of the ion and m_k is its mass. $\mathbf{E}(\mathbf{r})$ is the electric field at position $\mathbf{r}$ due to all the ions of all species in the system. e is the unit charge, ε_0 is the vacuum permittivity, ρ is the charge density due to every ion species, and λ is the Debye length describing the degree of electron screening. f_{kl}^{eq} is the equilibrium distribution of the mixture of ion species k and l, and τ_{kl} is the time scale for relaxation towards this equilibrium distribution. f_{kl}^{eq} is constructed such that the conservation laws are satisfied. τ_{kl} is the required input quanitity from MD to the BGK model.

In the MD model, we have N ions that are individually tracked according to the Hamiltonian equations

$$\begin{aligned}
\frac{\partial \mathbf{r}_i}{\partial t}&=\frac{1}{m_i}\nabla_{\mathbf{v}_i}H\\
\frac{\partial \mathbf{v}_i}{\partial t}&=-\frac{1}{m_i}\nabla_{\mathbf{r}_i}H\\
H(\{\mathbf{r}_k\}_{k=1}^N,\{\mathbf{v}_k\}_{k=1}^N)&=\sum_\alpha\left[\frac{m_\alpha|\mathbf{v}_\alpha|^2}{2}+\sum_{\alpha<j}U_{\alpha j}(\mathbf{r}_\alpha,\mathbf{r}_j)\right].
\end{aligned}\tag{2}$$

Consider Ω different species with their own charges and masses. Here $\mathbf{r}_i$ and $\mathbf{v}_i$ are the position and velocity of ion i. H is the Hamiltonian of the system, consisting of the sum of the kinetic energy and the potential energy. Ions evolve according to these simple equations in a periodic two-dimensional box. Note that, though motion is confined to this two dimensional box, representing the interaction with a screened Yukawa potential requires the system to actually be three-dimensional. Indeed, although not modeled explicitly, the electrons are able to move freely in 3D space; only the ions are confined to 2D.

The potential energy $U_{ij}(\mathbf{r}_i,\mathbf{r}_j)$ represents the potential energy of particle i due to particle j, screened by the electrons. It depends upon the identity of the two particles, as well as their positions. Let the indices of the particles that are of species one be expressed as $S_1 = \{1,\dots,N_1\}$ where N_1 is the number of ions of species one. Similarly, the indices of species k are $S_k = \{1+\sum_{l=1}^{k-1} N_l,\dots,\sum_{l=1}^{k} N_l\}$. Interparticle potentials can thus be written as $U_{ij}(\mathbf{r}_i,\mathbf{r}_j) = U_{\{kl\}}(\mathbf{r}_i,\mathbf{r}_j)$ where $i \in S_k$ and $j \in S_l$. We leave this potential general when deriving the kinetic equations, such that this derivation is generalizable to any desired potential. We only require that the potential satisfy the constraint that $\sum_i \sum_{i<j} U_{ij}(\mathbf{r}_i,\mathbf{r}_j)$ be fixed under a permutation of particle indices. That is,

$$\sum_i \sum_{i<j} U_{ij}(\mathbf{r}_i,\mathbf{r}_j) = \sum_{p(i)} \sum_{p(i)<p(j)} U_{p(i),p(j)}(\mathbf{r}_{p(i)},\mathbf{r}_{p(j)})$$

for all permutations $\{1,2,\dots\} \to \{p(1),p(2),\dots\}$. This simply implies that the total potential energy is independent of the choice of indices, which is a natural requirement.

Let ions of species k have mass m_k and charge Z_k. Using the Yukawa potential to simulate electron screening, the potential energy between two ions at position $\mathbf{r}$ and $\mathbf{r}'$ with charges Z_k and Z_l respectively is

$$U_{\{kl\}}(\mathbf{r},\mathbf{r}') = \frac{Z_k Z_l e^2}{4\pi\varepsilon_0} \frac{e^{-|\mathbf{r}-\mathbf{r}'|/\lambda}}{|\mathbf{r}-\mathbf{r}'|}. \tag{3}$$

BGK and MD models have a vast history of analytical and computational investigation. The HMM model requires us to connect them in a rigorous and explicit manner.

Multiscale derivation

Our system is described in full detail when we evolve every ion in the molecular dynamics simulation according to the Hamiltonian equations (2). We will derive, from this starting point, the BGK kinetic theory formulation of the problem (1). We begin by constructing a Klimontovich distribution [4] $\mathcal{N}_k$ for each species k that stores the location $\mathbf{r}_i$ and velocity $\mathbf{v}_i$ of every ion of that species at a given time t:

$$\mathcal{N}_k(\mathbf{r},\mathbf{v},\{\mathbf{r}_\alpha\}_{\alpha=1}^{N},\{\mathbf{v}_\alpha\}_{\alpha=1}^{N},t) = \sum_{i\in S_k} \delta\left(\mathbf{r}-\mathbf{r}_i(t)\right)\delta\left(\mathbf{v}-\mathbf{v}_i(t)\right). \tag{4}$$

$\delta(x)$ is the Dirac delta function. We are interested in the time evolution of the Klimontovich distribution. Thus, an aside relating to distributions and their derivatives is warranted.

Distributional derivatives

Consider the set of test functions, $D(\mathbb{R})$ that are infinitely differentiable and have compact support. A distribution is a linear mapping $T : D(\mathbb{R}) \to \mathbb{R}$. By convention, the operation of a distribution T on a target function f is written $\langle T, f\rangle$, and is defined as the integral over $\mathbb{R}$ of the product between $T(x)$ and $f(x)$. The Dirac delta "function" can thus be written $\langle \delta, f\rangle = f(0)$.

The derivative of a distribution is defined as $\langle T', f\rangle = -\langle T, f'\rangle$. This is a consequence of integration by parts:

$$\langle T', f\rangle = \int_{-\infty}^{\infty} \delta'(x) f(x)\,dx = [f(x)\delta(x)]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} \delta(x) f'(x)\,dx = -\langle T, f'\rangle.$$

We discount the first term in the integration by parts because our test functions are considered equal to zero outside of a bounded set. Though we will not explicitly compute the derivative of $\delta(x)$, it suffices to show that it exists and is well-defined. The above definition of the distributional derivative also allows for a seamless use of the product and chain rules in their standard forms in the presence of distributions. Our problem utilizes two-dimensional Dirac delta functions, defined as

$$\delta(\mathbf{r}) = \delta(r_x)\delta(r_y).$$

The gradient of this function similarly exists:

$$\langle \nabla_{\mathbf{r}}\delta(\mathbf{r}), f\rangle = -\langle \delta(\mathbf{r}), \nabla_{\mathbf{r}} f\rangle.$$

Gradients of two-dimensional delta functions are thus well defined.

Derivation of BGK from MD

Returning our attention to the time evolution of the indicator function, the Klimontovich distribution $\mathscr{N}_k$, we have

$$\begin{aligned}
\frac{\partial \mathscr{N}_k}{\partial t} &= \sum_{i\in S_k} \frac{\partial}{\partial t}[\delta(\mathbf{r}-\mathbf{r}_i(t))\delta(\mathbf{v}-\mathbf{v}_i(t))] \\
&= \sum_{i\in S_k} \delta(\mathbf{v}-\mathbf{v}_i(t))\frac{\partial}{\partial t}\delta(\mathbf{r}-\mathbf{r}_i(t)) + \delta(\mathbf{r}-\mathbf{r}_i(t))\frac{\partial}{\partial t}\delta(\mathbf{v}-\mathbf{v}_i(t)) \\
&= \sum_{i\in S_k} \Bigg\{ \delta(\mathbf{v}-\mathbf{v}_i(t))\frac{\partial(\mathbf{r}-\mathbf{r}_i(t))}{\partial t}\cdot\nabla_{(\mathbf{r}-\mathbf{r}_i)}[\delta(\mathbf{r}-\mathbf{r}_i(t))] \\
&\qquad + \delta(\mathbf{r}-\mathbf{r}_i(t))\frac{\partial(\mathbf{v}-\mathbf{v}_i(t))}{\partial t}\cdot\nabla_{(\mathbf{v}-\mathbf{v}_i)}[\delta(\mathbf{v}-\mathbf{v}_i(t))] \Bigg\} \\
&= \sum_{i\in S_k} \Bigg\{ -\frac{\partial \mathbf{r}_i(t)}{\partial t}\delta(\mathbf{v}-\mathbf{v}_i(t))\cdot\nabla_{\mathbf{r}}[\delta(\mathbf{r}-\mathbf{r}_i(t))] \\
&\qquad - \frac{\partial \mathbf{v}_i(t)}{\partial t}\delta(\mathbf{r}-\mathbf{r}_i(t))\cdot\nabla_{\mathbf{v}}[\delta(\mathbf{v}-\mathbf{v}_i(t))] \Bigg\}.
\end{aligned}$$

We now use the definition of the Hamiltonian. We focus our attention on particle i:

$$\frac{\partial \mathbf{r}_i}{\partial t} = \frac{1}{m_i}\nabla_{\mathbf{v}_i} H = \frac{1}{m_i}\nabla_{\mathbf{v}_i}\left(\sum_{\alpha}\left[\frac{m_\alpha|\mathbf{v}_\alpha|^2}{2} + \sum_{\alpha<j} U_{\alpha j}(\mathbf{r}_\alpha, \mathbf{r}_j)\right]\right) = \mathbf{v}_i. \tag{5}$$

In order to consider the potential, first we permute the indices such that particle i and particle

1 switch places. That is, $p(i) = 1$. Furthermore, let particle i be a member of species k, so

$$\begin{aligned}
\frac{\partial \mathbf{v}_i}{\partial t} = -\frac{1}{m_i}\nabla_{\mathbf{r}_i} H &= -\frac{1}{m_i}\nabla_{\mathbf{r}_i}\left(\sum_{\alpha}\left[\frac{m_\alpha|\mathbf{v}_\alpha|^2}{2} + \sum_{\alpha<j} U_{\alpha j}(\mathbf{r}_\alpha, \mathbf{r}_j)\right]\right) \\
&= -\frac{1}{m_i}\nabla_{\mathbf{r}_{p(i)}}\left[\sum_{p(\alpha)}\sum_{p(\alpha)<p(j)} U_{\alpha j}(\mathbf{r}_\alpha, \mathbf{r}_j)\right] \\
&= -\frac{1}{m_i}\nabla_{\mathbf{r}_{p(i)}}\left[\sum_{p(j)\neq 1} U_{ij}(\mathbf{r}_{p(i)}, \mathbf{r}_{p(j)})\right] \\
&= -\frac{1}{m_i}\nabla_{\mathbf{r}_{p(i)}}\left[\sum_{l}\sum_{p(j)\in S_l, p(j)\neq 1} U_{\{kl\}}(\mathbf{r}_{p(i)}, \mathbf{r}_{p(j)})\right].
\end{aligned} \tag{6}$$

Abusing notation, let us redefine the indices such that $i = p(i)$. Using (5) and (6), we find

$$\begin{aligned}
\frac{\partial \mathcal{N}_k}{\partial t} &= \sum_{i\in S_k}\left[-\frac{\partial \mathbf{r}_i(t)}{\partial t}\delta(\mathbf{v}-\mathbf{v}_i(t))\cdot\nabla_{\mathbf{r}}\delta(\mathbf{r}-\mathbf{r}_i) - \frac{\partial \mathbf{v}_i(t)}{\partial t}\delta(\mathbf{r}-\mathbf{r}_i(t))\cdot\nabla_{\mathbf{v}}[\delta(\mathbf{v}-\mathbf{v}_i(t))]\right] \\
&= \sum_{i\in S_k}\Bigg[-\mathbf{v}_i\delta(\mathbf{v}-\mathbf{v}_i(t))\cdot\nabla_{\mathbf{r}}\delta(\mathbf{r}-\mathbf{r}_i(t)) \\
&\quad + \frac{1}{m_i}\delta(\mathbf{r}-\mathbf{r}_i(t))\nabla_{\mathbf{r}_i}\left[\sum_{l}\sum_{j\in S_l, j\neq 1} U_{\{kl\}}(\mathbf{r}_i, \mathbf{r}_j)\right]\cdot\nabla_{\mathbf{v}}\delta(\mathbf{v}-\mathbf{v}_i(t))\Bigg] \\
&= \sum_{i\in S_k}\Bigg[-\mathbf{v}_i\delta(\mathbf{v}-\mathbf{v}_i(t))\cdot\nabla_{\mathbf{r}}\delta(r-\mathbf{r}_i(t)) \\
&\quad + \frac{1}{m_k}\delta(\mathbf{r}-\mathbf{r}_i(t))\nabla_{\mathbf{r}_i}\left[\sum_{l}\sum_{j\in S_l, j\neq 1} U_{\{kl\}}(\mathbf{r}_i, \mathbf{r}_j)\right]\cdot\nabla_{\mathbf{v}}\delta(\mathbf{v}-\mathbf{v}_i(t))\Bigg].
\end{aligned}$$

Notice that $\mathbf{v}_i\delta(\mathbf{v}-\mathbf{v}_i(t))$ is only nonzero when $\mathbf{v} = \mathbf{v}_i$, such that

$$\mathbf{v}_i\delta(\mathbf{v}-\mathbf{v}_i(t)) = \mathbf{v}\delta(\mathbf{v}-\mathbf{v}_i(t)).$$

Similarly, $\nabla_{\mathbf{r}_i}U_{\{kl\}}(\mathbf{r}_i, \mathbf{r}_j)$ depends on $\mathbf{r}_i$, so it is only nonzero when $\mathbf{r} = \mathbf{r}_i$, at which point

$$\nabla_{\mathbf{r}_i}U_{\{kl\}}(\mathbf{r}_i, \mathbf{r}_j)\delta(\mathbf{r}-\mathbf{r}_i(t)) = \nabla_{\mathbf{r}}U_{\{kl\}}(\mathbf{r}, \mathbf{r}_j)\delta(\mathbf{r}-\mathbf{r}_i(t)).$$

Making these substitutions,

$$
\begin{aligned}
\frac{\partial \mathcal{N}_k}{\partial t} =& \sum_{i\in S_k}\Bigg[-\mathbf{v}_i\delta(\mathbf{v}-\mathbf{v}_i(t))\cdot\nabla_{\mathbf{r}}\delta(\mathbf{r}-\mathbf{r}_i(t))\\
&+\frac{1}{m_k}\delta(\mathbf{r}-\mathbf{r}_i(t))\nabla_{\mathbf{r}_i}\Bigg[\sum_l\sum_{j\in S_l, j\neq 1}U_{\{kl\}}(\mathbf{r}_i,\mathbf{r}_j)\Bigg]\cdot\nabla_{\mathbf{v}}\delta(\mathbf{v}-\mathbf{v}_i(t))\Bigg]\\
=&-\mathbf{v}\cdot\nabla_{\mathbf{r}}\sum_{i\in S_k}\delta(\mathbf{r}-\mathbf{r}_i(t))\delta(\mathbf{v}-\mathbf{v}_i(t))\\
&+\frac{1}{m_k}\nabla_{\mathbf{r}}\Bigg[\sum_l\sum_{j\in S_l}U_{\{kl\}}(\mathbf{r},\mathbf{r}_j)\Bigg]\cdot\nabla_{\mathbf{v}}\sum_{i\in S_k}\delta(\mathbf{r}-\mathbf{r}_i(t))\delta(\mathbf{v}-\mathbf{v}_i(t))\Bigg]\\
=&-\mathbf{v}\cdot\nabla_{\mathbf{r}}\mathcal{N}_k+\frac{1}{m_k}\nabla_{\mathbf{r}}\sum_l\Bigg[\sum_{j\in S_l}U_{\{kl\}}(\mathbf{r},\mathbf{r}_j)\Bigg]\cdot\nabla_{\mathbf{v}}\mathcal{N}_k.
\end{aligned}
$$

We can think of the term $\sum_l\left[\sum_{j\in S_l}U_{\{kl\}}(\mathbf{r},\mathbf{r}_j)\right]$ as the potential at the point $\mathbf{r}$ due to every ion. We now take the ensemble average of both sides of this equation, defined as $f_k(\mathbf{r},\mathbf{v},t) = E[\mathcal{N}_k]$. This ensemble average is taken with respect to all equivalent initial choices for $\mathbf{r}_i(0)$ and $\mathbf{v}_i(0)$. Thus, this expected value only acts on $\mathbf{r}_i$ and $\mathbf{v}_i$. Using this, we can show

$$
\begin{aligned}
E\left[\frac{\partial \mathcal{N}_k}{\partial t}\right] =& E\Bigg[-\mathbf{v}\cdot\nabla_{\mathbf{r}}\mathcal{N}_k+\frac{1}{m_k}\nabla_{\mathbf{r}}\sum_l\Bigg[\sum_{j\in S_l}U_{\{kl\}}(\mathbf{r},\mathbf{r}_j)\Bigg]\cdot\nabla_{\mathbf{v}}\mathcal{N}_k\Bigg]\\
\frac{\partial E[\mathcal{N}_k]}{\partial t} =& -E\left[\mathbf{v}\cdot\nabla_{\mathbf{r}}\mathcal{N}_k\right]+\frac{1}{m_k}E\Bigg[\nabla_{\mathbf{r}}\sum_l\Bigg[\sum_{j\in S_l}U_{\{kl\}}(\mathbf{r},\mathbf{r}_j)\Bigg]\cdot\nabla_{\mathbf{v}}\mathcal{N}_k\Bigg]\\
\frac{\partial f_k}{\partial t} =& -\mathbf{v}\cdot\nabla_{\mathbf{r}}E[\mathcal{N}_k]+\frac{1}{m_k}E\Bigg[\nabla_{\mathbf{r}}\sum_l\Bigg[\sum_{j\in S_l}U_{\{kl\}}(\mathbf{r},\mathbf{r}_j)\Bigg]\cdot\nabla_{\mathbf{v}}\mathcal{N}_k\Bigg]\\
\frac{\partial f_k}{\partial t} =& -\mathbf{v}\cdot\nabla_{\mathbf{r}}f_k+\frac{1}{m_k}E\Bigg[\nabla_{\mathbf{r}}\sum_l\Bigg[\sum_{j\in S_l}U_{\{kl\}}(\mathbf{r},\mathbf{r}_j)\Bigg]\cdot\nabla_{\mathbf{v}}\mathcal{N}_k\Bigg].
\end{aligned}
$$

Because $\mathbf{v}$ and the $\nabla_{\mathbf{r}}$ gradient do not depend on any of the ensemble variables, we can simplify the first term. We cannot do the same with the second term because the potential field is heavily dependent on the initial positions of the ions and their subsequent trajectories.

Consider multiplying the potential function by several delta functions and integrating:

$$
\begin{aligned}
\sum_l\sum_{j\in S_l}U_{\{kl\}}(\mathbf{r},\mathbf{r}_j) =& \sum_l\sum_{j\in S_l}\iint U_{\{kl\}}(\mathbf{r},\mathbf{r}')\delta(\mathbf{r}'-\mathbf{r}_j)\delta(\mathbf{v}'-\mathbf{v}_j)\,d\mathbf{r}'d\mathbf{v}'\\
=& \sum_l\iint U_{\{kl\}}(\mathbf{r},\mathbf{r}')\sum_{j\in S_l}\delta(\mathbf{r}'-\mathbf{r}_j)\delta(\mathbf{v}'-\mathbf{v}_j)\,d\mathbf{r}'d\mathbf{v}'\\
=& \sum_l\iint U_{\{kl\}}(\mathbf{r},\mathbf{r}')\mathcal{N}_l(\mathbf{r}',\mathbf{v}',t)\,d\mathbf{r}'d\mathbf{v}'.
\end{aligned}
$$

Plugging this into our remaining expected value expression, we have

$$\begin{aligned}
E\left[\nabla_{\mathbf{r}}\sum_{l}\left[\sum_{j\in S_l}U_{\{kl\}}(\mathbf{r},\mathbf{r}_j)\right]\cdot\nabla_{\mathbf{v}}\mathscr{N}_k\right] =&E\left[\nabla_{\mathbf{r}}\sum_{l}\left[\iint U_{\{kl\}}(\mathbf{r},\mathbf{r}')\mathscr{N}_l(\mathbf{r}',\mathbf{v}',t)\,d\mathbf{r}'d\mathbf{v}'\right]\cdot\nabla_{\mathbf{v}}\mathscr{N}_k(\mathbf{r},\mathbf{v},t)\right]\\
=&E\left[\iint\nabla_{\mathbf{r}}\sum_{l}\left[U_{\{kl\}}(\mathbf{r},\mathbf{r}')\mathscr{N}_l(\mathbf{r}',\mathbf{v}',t)\right]d\mathbf{r}'d\mathbf{v}'\cdot\nabla_{\mathbf{v}}\mathscr{N}_k(\mathbf{r},\mathbf{v},t)\right]\\
=&E\left[\iint\sum_{l}\left[\nabla_{\mathbf{r}}\left(U_{\{kl\}}(\mathbf{r},\mathbf{r}')\right)\cdot\nabla_{\mathbf{v}}\mathscr{N}_l(\mathbf{r}',\mathbf{v}',t)\mathscr{N}_k(\mathbf{r},\mathbf{v},t)\right]d\mathbf{r}'d\mathbf{v}'\right]\\
=&\iint\sum_{l}\left[\nabla_{\mathbf{r}}\left(U_{\{kl\}}(\mathbf{r},\mathbf{r}')\right)\cdot\nabla_{\mathbf{v}}E\left[\mathscr{N}_k(\mathbf{r},\mathbf{v},t)\mathscr{N}_l(\mathbf{r}',\mathbf{v}',t)\right]\right]d\mathbf{r}'d\mathbf{v}'.
\end{aligned}$$

We have once again simplified the expected value to contain only the terms that depend on the ion positions and velocities. Let us consider this term more carefully:

$$\mathscr{N}_k(\mathbf{r},\mathbf{v},t)\mathscr{N}_l(\mathbf{r}',\mathbf{v}',t)=\left(\sum_{i\in S_k}\delta(\mathbf{r}-\mathbf{r}_i(t))\delta(\mathbf{v}-\mathbf{v}_i(t))\right)\left(\sum_{j\in S_l}\delta(\mathbf{r}'-\mathbf{r}_j(t))\delta(\mathbf{v}'-\mathbf{v}_j(t))\right).$$

This expression is only nonzero when there is a particle of species k at $(\mathbf{r},\mathbf{v})$ and a particle of species l at $(\mathbf{r}',\mathbf{v}')$. Thus, it can be expressed as

$$\mathscr{N}_{kl}(\mathbf{r},\mathbf{v},\mathbf{r}',\mathbf{v}',\{\mathbf{r}_k\}_{k=1}^{n},\{\mathbf{v}_k\}_{k=1}^{n},t)\;=\sum_{i\in S_k,j\in S_l}\delta(\mathbf{r}-\mathbf{r}_i(t))\delta(\mathbf{v}-\mathbf{v}_i(t))\delta(\mathbf{r}'-\mathbf{r}_j(t))\delta(\mathbf{v}'-\mathbf{v}_j(t)).$$

$\mathscr{N}_{kl}$, which exists for every pair of species, is a generalization of $\mathscr{N}_i$ to two particles that may be of different species. Its expected value is the two-particle correlation function $f_{kl}(\mathbf{r},\mathbf{v},\mathbf{r}',\mathbf{v}',t)$. At this point, we have derived the BBGKY hierarchy. If f_{kl} could be computed in MD, we could couple these two models. One advantage of this would be that the BBGKY hierarchy involves no assumptions and is an exact statistical description of the system. However, f_{kl} is high-dimensional and difficult to compute from MD. Therefore, we make some simplifying assumptions. Because f_{kl} is the two-particle correlation function, we write it without loss of generality as

$$f_{kl}(\mathbf{r},\mathbf{v},\mathbf{r}',\mathbf{v}',t)=f_k(\mathbf{r},\mathbf{v},t)f_l(\mathbf{r}',\mathbf{v}',t)+C_{kl} \tag{7}$$

where C_{kl} is a complicated, unknown remainder function. Substituting this into our differential equation, we have

$$
\begin{aligned}
E\left[\nabla_{\mathbf{r}}\sum_l\left[\sum_{j\in S_l}U_{\{kl\}}(\mathbf{r},\mathbf{r}_j)\right]\cdot\nabla_{\mathbf{v}}\mathscr{N}_k\right]& \\
&=\iint\sum_l\left[\nabla_{\mathbf{r}}\left(U_{\{kl\}}(\mathbf{r},\mathbf{r}')\right)\cdot\nabla_{\mathbf{v}}f_{kl}(\mathbf{r},\mathbf{v},\mathbf{r}',\mathbf{v}',t)\right]d\mathbf{r}'d\mathbf{v}'\\
&=\iint\sum_l\left[\nabla_{\mathbf{r}}\left(U_{\{kl\}}(\mathbf{r},\mathbf{r}')\right)\cdot\nabla_{\mathbf{v}}\left[f_k(\mathbf{r},\mathbf{v},t)f_l(\mathbf{r}',\mathbf{v}',t)+C_{kl}\right]\right]d\mathbf{r}'d\mathbf{v}'\\
&=\nabla_{\mathbf{r}}\sum_l\left(\iint U_{\{kl\}}(\mathbf{r},\mathbf{r}')f_l(\mathbf{r}',\mathbf{v}',t)\,d\mathbf{r}'d\mathbf{v}'\right)\cdot\nabla_{\mathbf{v}}f_k+C'_{kl}\\
&=\nabla_{\mathbf{r}}\sum_l\left(\int U_{\{kl\}}(\mathbf{r},\mathbf{r}')\left(\int f_k(\mathbf{r}',\mathbf{v}',t)\,d\mathbf{v}'\right)d\mathbf{r}'\right)\cdot\nabla_{\mathbf{v}}f_k+C'_{kl}\\
&=\nabla_{\mathbf{r}}\sum_l\left(\int U_{\{kl\}}(\mathbf{r},\mathbf{r}')n_l(\mathbf{r}',t)\,d\mathbf{r}'\right)\cdot\nabla_{\mathbf{v}}f_k+C'_{kl}.
\end{aligned}
$$

C'_{kl} is an additional unknown function quantifying the collisional properties between species k and species l. We here make use of the fact that the particle density of ion k, $n_k(\mathbf{r},t)$, is defined as the velocity integral of f_k.

Up until this point, we have left the potential function undefined. For different problems, a different molecular dynamics potential function will lead to a different multiscale kinetic theory formulation. We here use the Yukawa potential between ions (3), representing Coulomb interactions screened by background electrons. Let Z_ke be the electric charge of ion species k, and λ be the Debye length dictating the degree of electron screening. Then,

$$U_{\{kl\}}(\mathbf{r},\mathbf{r}')=\frac{Z_kZ_le^2}{4\pi\varepsilon_0|\mathbf{r}-\mathbf{r}'|}e^{-|\mathbf{r}-\mathbf{r}'|/\lambda}.$$

Plugging this in, we get

$$
\begin{aligned}
\nabla_{\mathbf{r}}\sum_l\left(\int U_{\{kl\}}(\mathbf{r},\mathbf{r}')n_l(\mathbf{r}',t)\,d\mathbf{r}'\right)&=\nabla_{\mathbf{r}}\sum_l\left(\int\frac{Z_kZ_le^2}{4\pi\varepsilon_0|\mathbf{r}-\mathbf{r}'|}e^{-|\mathbf{r}-\mathbf{r}'|/\lambda}n_l(\mathbf{r}',t)\,d\mathbf{r}'\right)\\
&=Z_ke\nabla_{\mathbf{r}}\int\frac{\rho(\mathbf{r}',t)}{4\pi\varepsilon_0|\mathbf{r}-\mathbf{r}'|}e^{-|\mathbf{r}-\mathbf{r}'|/\lambda}\,d\mathbf{r}'\\
&=Z_ke\int\frac{1}{4\pi\varepsilon_0}\rho(\mathbf{r}',t)\nabla_{\mathbf{r}}\left(\frac{e^{-|\mathbf{r}-\mathbf{r}'|/\lambda}}{|\mathbf{r}-\mathbf{r}'|}\right)d\mathbf{r}' \qquad (8)\\
&=-Z_ke\mathbf{E}(\mathbf{r},t),
\end{aligned}
$$

where we have defined the total charge density to be

$$\rho(\mathbf{r},t)=\sum_l Z_ln_l(\mathbf{r},t). \qquad (9)$$

In three dimensions, the electric potential of a charge distribution governed by the Yukawa potential satisfies the *screened Poisson equation*:

$$(\triangle-\lambda^2)\phi(\mathbf{r},t)=-\frac{1}{\varepsilon_0}\rho(\mathbf{r},t) \qquad (10)$$

and the electric field can be computed by

$$\mathbf{E}(\mathbf{r},t) = -\nabla_{\mathbf{r}}\phi(\mathbf{r},t). \tag{11}$$

This is an efficient and computationally robust way to compute the electric field in three dimensions. However, in two dimensions, there exists no such simple ordinary differential equation for the electric potential, and we must compute the electric field directly by evaluating the integral (8), which is difficult to evaluate due to the short range of the Yukawa potential [1]. This is a drawback *only* of the simplified two-dimensional model, and does not pose any issues in a fully physical implementation.

Regardless, the partial differential equation governing the evolution of f_k is

$$\frac{\partial f_k}{\partial t} + \mathbf{v}\cdot\nabla_{\mathbf{r}} f_k + \frac{Z_k e}{m_k}\mathbf{E}(\mathbf{r},t)\cdot\nabla_{\mathbf{v}} f_k = \sum_l C'_{kl}.$$

Each species distribution evolves according to its respective PDE. Only the unknown C'_{kl} terms remain. They are exceptionally complicated functions relating to the ion correlations. They are typically called the “collisional” terms since they describe particle-particle interactions beyond the electric potential interaction. The Bhatnagar-Gross-Krook (BGK) approximation of these terms is

$$C'_{kl} = \frac{f^{eq}_{kl}(\mathbf{r},\mathbf{v},t) - f_k(\mathbf{r},\mathbf{v},t)}{\tau_{kl}(\mathbf{r})},$$

where f^{kl}_{eq} is the known Maxwellian equilibrium distribution for the mixture of species k and l and $\tau_{kl}(\mathbf{r})$ is a relaxation parameter. Calculation of τ_{kl} is typically handled in an ad hoc manner. We instead derive a means of computing these parameters from an MD simulation. Thus, the kinetic partial differential equation is

$$\begin{aligned}
&\frac{\partial f_k}{\partial t} + \mathbf{v}\cdot\nabla_{\mathbf{r}} f_k + \frac{Z_k e}{m_k}\mathbf{E}(\mathbf{r},t)\cdot\nabla_{\mathbf{v}} f_k = \sum_l \frac{f^{eq}_{kl} - f_k}{\tau_{kl}}\\
&\mathbf{E}(\mathbf{r},t) = -\int \frac{1}{4\pi\varepsilon_0}\rho(\mathbf{r}',t)\nabla_{\mathbf{r}}\left(\frac{e^{-|\mathbf{r}-\mathbf{r}'|/\lambda}}{|\mathbf{r}-\mathbf{r}'|}\right) d\mathbf{r}'\\
&f_k(\mathbf{r},\mathbf{v},0) = f_k(\mathbf{r},\mathbf{v})\\
&\quad f_k(\mathbf{r},\mathbf{v},t)|_{x=0} = f_k(\mathbf{r},\mathbf{v},t)|_{x=L_x}, \quad f_k(\mathbf{r},\mathbf{v},t)|_{y=0} = f_k(\mathbf{r},\mathbf{v},t)|_{y=L_y}
\end{aligned}$$

as described in (1). In the next section, we demonstrate how to derive the relaxation times τ_{kl} from a molecular dynamics simulation.

Derivation of compression operator

The key steps of HMM are compression and reconstruction. The reconstruction operator is discussed at length in the following section. Here we derive a complete compression operator to instantiate a BGK simulation from the MD simulation with the proper parameters.

First, in order to initialize BGK from MD, we must construct an MD discretized version of f_k, which will serve as the initial condition in the BGK simulation. Let us call this function

g_k. Drawing from [7], consider the Klimontovich distribution $\mathscr{N}_k$, and integrate it over a small region in phase space:

$$g_k(\mathbf{r}_n, \mathbf{v}_m, t) = \int_{\mathbf{r}_n}^{\mathbf{r}_n+\Delta\mathbf{r}} \int_{\mathbf{v}_m}^{\mathbf{v}_m+\Delta\mathbf{v}} \mathscr{N}_k(\mathbf{r}, \mathbf{v}, \{\mathbf{r}_\alpha\}_{\alpha=1}^N, \{\mathbf{v}_\alpha\}_{\alpha=1}^N, t)\, d\mathbf{r}\, d\mathbf{v}$$

where $g_k(\mathbf{r}_n, \mathbf{v}_m, t)$ is the number of ions of species k that fall in the phase space region defined by $[\mathbf{r}_n, \mathbf{r}_n + \Delta\mathbf{r}] \times [\mathbf{v}_m, \mathbf{v}_m + \Delta\mathbf{v}]$. $f_k(\mathbf{r}, \mathbf{v}, t)$ is defined as the expected particle density of species k at time t at position $\mathbf{r}$ with velocity $\mathbf{v}$. Thus, if we take $\Delta\mathbf{r} \to 0$ and $\Delta\mathbf{v} \to 0$, $E[g(\mathbf{r}_n, \mathbf{v}_m, t)] \to f(\mathbf{r}_n, \mathbf{v}_m, t)$ by definition. g_k is therefore a discretized, approximate version of f_k.

Now, in order to fully initialize BGK, we must develop a means of extracting an approximation of the parameters τ_{kl} from the MD simulation. We will assume τ_{kl} is spatially dependent, but not velocity dependent. Critically, the relaxation times τ_{kl} have no formulae as a function of $\{\mathbf{r}_i\}_{i=1}^N$ and $\{\mathbf{v}_i\}_{i=1}^N$. We must compute τ_{kl} through some auxiliary function or functions.

The BGK equation is constructed to inherit useful features of the Boltzmann equation. One of the most important of these features is the existence of an H-theorem [9]. Namely, we can define the quantity H as

$$H(t) = \iint f(\mathbf{r}, \mathbf{v}, t) \log(f(\mathbf{r}, \mathbf{v}, t))\, d\mathbf{r}\, d\mathbf{v}, \tag{12}$$

and this function will be monotonically decreasing. H is a critical quantity in statistical mechanics, and it is often compared to entropy. Consequently, we would like to select τ_{kl} such that the derivative of H (i.e. the entropy production rate) in the BGK equation matches the computed rates of change of H from the MD simulation. This rate can be decomposed into contributions from all species pair interactions at all spatial coordinates, so we select each $\tau_{kl}(\mathbf{r})$ such that the contribution to the entropy production rate from the $\{k, l\}$ interaction at that point is the same in MD and BGK.

Our computation of H in MD draws from [7]. Consider now the distribution of all species together, f, which is approximated in MD by g:

$$\sum_k f_k(\mathbf{r}, \mathbf{v}, t) = f(\mathbf{r}, \mathbf{v}, t) \approx E[g(\mathbf{r}, \mathbf{v}, t)] = \sum_k g_k(\mathbf{r}, \mathbf{v}, t) \tag{13}$$

Let us discretize $\mathbf{r}$ with spacing Δr and $\mathbf{v}$ with spacing Δv in all directions. Then H can be computed from our MD simulations by:

$$\begin{aligned} H^{MD} &= \iint f(\mathbf{r}, \mathbf{v}, t) \log(f(\mathbf{r}, \mathbf{v}, t))\, d\mathbf{r}\, d\mathbf{v} \\ &\approx \iint E[g(\mathbf{r}, \mathbf{v}, t)] \log(E[g(\mathbf{r}, \mathbf{v}, t)])\, d\mathbf{r}\, d\mathbf{v} \\ &\approx \sum_n \sum_m E[g(\mathbf{r}_n, \mathbf{v}_m, t)] \log(E[g(\mathbf{r}_n, \mathbf{v}_m, t)])(\Delta r \Delta v)^d \end{aligned}$$

where d is the dimensionality of the problem and $\sum_n \sum_m$ is interpreted as the sum over all positions and velocities up to some cutoff velocity. For example, in three dimensions, this would become a six-sum over the three spatial and velocity dimensions. We now decompose g

into the different species components and consider the time derivative of H^{MD}:

$$\begin{aligned}
\frac{dH^{MD}}{dt} \approx& \frac{\partial}{\partial t}\left(\sum_n\sum_m E[g(\mathbf{r}_n,\mathbf{v}_m,t)]\log(E[g(\mathbf{r}_n,\mathbf{v}_m,t)])(\Delta r\Delta v)^d\right)\\
=&\sum_n\sum_m\left(\frac{\partial E[g]}{\partial t}(\log(E[g])+1)\right)(\Delta r\Delta v)^d\\
=&\sum_k\sum_n\left(\sum_m\left[\frac{\partial E[g_k]}{\partial t}(\log(E[g])+1)\right](\Delta r\Delta v)^d\right)
\end{aligned}$$

The quantity inside the parentheses can be computed from the molecular dynamics simulation, with the ensemble averages being approximated as time averages. It has also been decomposed into terms corresponding to the entropy production of each species k at each position $\mathbf{r}_n$.

Consider now the time derivative of the BGK form of H:

$$\begin{aligned}
\frac{\partial H^{BGK}}{\partial t} =& \frac{\partial}{\partial t}\iint f\log(f)\,d\mathbf{r}\,d\mathbf{v}\\
=&\iint \frac{\partial f}{\partial t}(\log(f)-1)\,d\mathbf{r}\,d\mathbf{v}\\
=&\sum_k\iint\frac{\partial f_k}{\partial t}(\log(f)-1)\,d\mathbf{r}\,d\mathbf{v}\\
=&\sum_k\iint\sum_l\frac{f_{kl}^{eq}-f_k}{\tau_{kl}}(\log(f)-1)\,d\mathbf{r}\,d\mathbf{v}\\
&-\iint\mathbf{v}\cdot\nabla_{\mathbf{r}}f(\log(f)-1)\,d\mathbf{r}\,d\mathbf{v}\\
&-\iint\frac{Z_k e}{m_k}\mathbf{E}(\mathbf{r},t)\cdot\nabla_{\mathbf{v}}f(\log(f)-1)\,d\mathbf{r}\,d\mathbf{v}.
\end{aligned}$$

Consider the third integral. We use the divergence theorem and the fact that f vanishes at large $|\mathbf{v}|$ to show

$$\begin{aligned}
\iint\frac{Z_k e}{m_k}\mathbf{E}(\mathbf{r},t)\cdot\nabla_{\mathbf{v}}f(\log(f)-1)\,d\mathbf{r}\,d\mathbf{v} &= \int\frac{Z_k e}{m_k}\mathbf{E}(\mathbf{r},t)\cdot\int\nabla_{\mathbf{v}}(f\log(f))\,d\mathbf{v}\,d\mathbf{r}\\
&=\int\frac{Z_k e}{m_k}\mathbf{E}(\mathbf{r},t)\cdot\left(\oint f\log(f)d\mathbf{v}\right)d\mathbf{r}\\
&=0
\end{aligned}$$

Thus,

$$\begin{aligned}
\frac{dH^{BGK}}{dt} = \sum_k\Bigg(\sum_l\iint&\frac{f_{kl}^{eq}(\mathbf{r},\mathbf{v},t)-f_k(\mathbf{r},\mathbf{v},t)}{\tau_{kl}(\mathbf{r})}(\log(f)-1)\,d\mathbf{r}\,d\mathbf{v}\\
&-\iint\mathbf{v}\cdot\nabla_{\mathbf{r}}f(\log(f)-1)\,d\mathbf{r}\,d\mathbf{v}\Bigg).
\end{aligned}$$

Consider one part of the first integral:

$$-\int\frac{1}{\tau_{kl}(\mathbf{r})}\int f_{kl}^{eq}-f_k\,d\mathbf{v}\,d\mathbf{r}=0.$$

This is the case because the velocity integral of f_k is the number density of ions k at $\mathbf{r}$ and the equilibrium distribution f_{kl}^{eq} is chosen in each case to conserve this quantity. Note also that if we integrate over a small region in the spatial domain, the second integral becomes:

$$\begin{aligned}\iint \mathbf{v}\cdot\nabla_{\mathbf{r}} f(\log(f)-1)\,d\mathbf{r}\,d\mathbf{v} &= \int \mathbf{v}\cdot\int \nabla_{\mathbf{r}}(f\log(f))\,d\mathbf{r}\,d\mathbf{v}\\ &= \sum_k \iint \mathbf{v}\cdot\mathbf{F}_k\,d\mathbf{r}\,d\mathbf{v}\end{aligned}$$

where $\mathbf{F}_k = \nabla_{\mathbf{r}}(f_k\log(f))$. In order to compute the predicted rate of change of H in BGK from an MD simulation, consider discretizing phase space in the same manner as used for H^{MD}:

$$\begin{aligned}\frac{dH^{BGK}}{dt} \approx &(\Delta r\Delta v)^d \sum_k\sum_l\sum_n \frac{1}{\tau_{kl}(\mathbf{r}_n)}\left(\sum_m (f_{kl}^{eq}(\mathbf{r}_n,\mathbf{v}_m,t)-f_k(\mathbf{r}_n,\mathbf{v}_m,t))\log(f(\mathbf{r}_n,\mathbf{v}_m,t)\right)\\ &(\Delta r\Delta v)^d \sum_k\sum_n\sum_m \mathbf{F}_k(\mathbf{r}_n,\mathbf{v}_m,t)\cdot\mathbf{v}_m\end{aligned}$$

Now let us compare these entropy production rates and isolate the entropy production due to species k at position $\mathbf{r}_n$:

$$\begin{aligned}\frac{dH^{MD}}{dt} &= \frac{dH^{BGK}}{dt}\\ \sum_k\sum_n\left(\sum_m\left[\frac{\partial E[g_k]}{\partial t}(\log(E[g])+1)\right]\right) &= \sum_k\sum_l\sum_n \frac{1}{\tau_{kl}(\mathbf{r}_n)}\left(\sum_m (f_{kl}^{eq}(\mathbf{r}_n,\mathbf{v}_m,t)-f_k(\mathbf{r}_n,\mathbf{v}_m,t))\log(f(\mathbf{r}_n,\mathbf{v}_m,t)\right)\\ &\quad \sum_k\sum_n\sum_m \mathbf{F}_k(\mathbf{r}_n,\mathbf{v}_m,t)\cdot\mathbf{v}_m\\ \sum_m\left[\frac{\partial E[g_k]}{\partial t}(\log(E[g])+1)\right] &= \sum_l \frac{1}{\tau_{kl}(\mathbf{r}_n)}\sum_m (f_{kl}^{eq}-E[g_k])\log(E[g])\\ &\quad +\sum_m \mathbf{F}_k(\mathbf{r}_n,\mathbf{v}_m,t)\cdot\mathbf{v}_m.\end{aligned} \tag{14}$$

The equilibrium distributions f_{kl}^{eq} are derived analytically from moments of f_k and f_l, which can be approximated from $E[g_k]$ and $E[g_l]$. All other quantities can be computed as time averages from an MD simulation.

Note that in the single species case, there is only a single τ, and this equation allows us to solve for it explicitly at each position. In the case where there are $\Omega > 1$ species, the above formulation gives us Ω equations at each position, but we have Ω^2 unknown quantities τ_{kl} at each position. We need an additional $\Omega(\Omega-1)$ equations to compute every τ_{kl} in the multispecies case.

A critical feature of the BGK model is conservation of kinetic energy over the whole domain. That is, in the MD and KT notations we have

$$\sum_k\sum_{i\in S_k}\frac{1}{2}m_k|\mathbf{v}_i|^2 = \text{constant}$$

$$\sum_k\iint \frac{1}{2}m_k|\mathbf{v}|^2 f_k\,d\mathbf{v}\,d\mathbf{r} = \text{constant}$$

where k denotes the particle species and i denotes an individual particle. The kinetic energy density K_k of species k at any point $\mathbf{r}$ is defined

$$K_k = \int \frac{1}{2} m_k |\mathbf{v}|^2 f \, d\mathbf{v}. \tag{15}$$

Although the overall kinetic energy is constant in the domain, kinetic energy is transported throughout the domain and exchanged between particle species as the system moves towards equilibrium. We close our system of equations for τ_{kl} by matching the rates of kinetic energy transfer between different species in the BGK and MD formulations.

To compute the rate of change of K_k, we start by multiplying the BGK partial differential equation (1) by $\frac{1}{2} m_k |\mathbf{v}|^2$ and integrating over $\mathbf{v}$ to get

$$\int \frac{1}{2} m_k |\mathbf{v}|^2 \frac{\partial f_k}{\partial t} \, d\mathbf{v} + \int \frac{1}{2} m_k |\mathbf{v}|^2 \mathbf{v} \cdot \nabla_{\mathbf{r}} f_k \, d\mathbf{v} + \int \frac{1}{2} m_k |\mathbf{v}|^2 \frac{Z_k e}{m_k} \mathbf{E} \cdot \nabla_{\mathbf{v}} f_k \, d\mathbf{v}$$
$$= \int \frac{1}{2} m_k |\mathbf{v}|^2 \sum_l \frac{f_{kl}^{eq} - f_k}{\tau_{kl}} \, d\mathbf{v}.$$

If we rearrange terms within the integrals and note that

$$|\mathbf{v}|^2 \nabla_{\mathbf{v}} f_k = \nabla_{\mathbf{v}} (|\mathbf{v}|^2 f_k) - 2\mathbf{v} f_k,$$

then we can rewrite this as

$$\frac{\partial}{\partial t} \int \frac{1}{2} m_k |\mathbf{v}|^2 f_k \, d\mathbf{v} + \nabla_{\mathbf{r}} \cdot \int \frac{1}{2} m_k |\mathbf{v}|^2 \mathbf{v} f_k \, d\mathbf{v} + Z_k e \mathbf{E} \cdot \left(\int \nabla_{\mathbf{v}} \left(\frac{1}{2} |\mathbf{v}|^2 f_k \right) d\mathbf{v} - \int \mathbf{v} f_k \, d\mathbf{v} \right)$$
$$= \sum_l \frac{\int \frac{1}{2} m_k |\mathbf{v}|^2 f_{kl}^{eq} \, d\mathbf{v} - \int \frac{1}{2} m_k |\mathbf{v}|^2 f_k \mathbf{v}}{\tau_{kl}}.$$

The term $\int \nabla_{\mathbf{v}} \left(\frac{1}{2} |\mathbf{v}|^2 f \right) d\mathbf{v}$ vanishes due to the divergence theorem provided f_k decays faster than $|\mathbf{v}|^{-2}$ as $|\mathbf{v}| \to \infty$. This is a reasonable assumption of any physical system. Noting the momentum transport density of species k is

$$m_k n_k \mathbf{u}_k = \int m_k \mathbf{v} f_k \, d\mathbf{v}$$

and the energy transport density $\mathbf{Q}$ of species k is defined

$$\mathbf{Q}_k = \int \frac{1}{2} m_k |\mathbf{v}|^2 \mathbf{v} f_k \, d\mathbf{v},$$

we find

$$\frac{\partial K_k}{\partial t} + \nabla_{\mathbf{r}} \cdot \mathbf{Q}_k - Z_k e \mathbf{E} \cdot n_k \mathbf{u}_k = \sum_l \frac{K_{kl}^{eq} - K_k}{\tau_{kl}} \tag{16}$$

where K_{kl}^{eq} is the kinetic energy density from the equilibrium distribution.

We must decompose the the above equation into the contributions from each species. $\frac{\partial K_k^l}{\partial t}$ denotes the contribution to $\frac{\partial K_k}{\partial t}$ due to particles of species l. Since the electric field is a result of a linear combination of contributions from each species, we can also define the contribution to the electric field of species l as $\mathbf{E}_l$.

We are interested in τ_{kl} for localized regions, so there may be a net flux of energy into any given spatial region due to $\mathbf{Q}_k$. We argue that this contribution to $\frac{dK_k}{dt}$ only arises from ions of species k. This is because this energy flux is due to ions of species k entering or leaving a region carrying kinetic energy with them. That is, particles of species $l \neq k$ entering and exiting the region do not contribute to the kinetic energy density of species k in that region. We also note that for $k = l$, we have $K_{kk}^{eq} = K_k$ by definition. Therefore, if we decompose (16) by species and subtract out the intra-species $l = k$ term, we find

$$\sum_{l \neq k} \left(\frac{\partial K_k^l}{\partial t} - Z_k e \mathbf{E}_l \cdot \mathbf{u}_k \right) = \sum_{l \neq k} \frac{K_{kl}^{eq} - K_k}{\tau_{kl}} \tag{17}$$

for the cross-species contributions. In MD, the kinetic energy density of species k in a given region A is given by

$$K_k^{MD} = \frac{1}{(\Delta r)^d} \sum_{\substack{i \in S_k, \\ \mathbf{r}_i \in A}} \frac{1}{2} m_k |\mathbf{v}_i|^2.$$

Taking the time derivative, we have

$$\frac{\partial K_k^{MD}}{\partial t} = \frac{1}{(\Delta r)^d} \sum_{i \in S_k, r \in A} m_k \mathbf{v}_i \cdot \frac{\partial \mathbf{v}_i}{\partial t} = \frac{1}{(\Delta r)^d} \sum_{\substack{i \in S_k, \\ \mathbf{r}_i \in A}} \mathbf{v}_i \cdot \mathbf{f}_i$$

where $\mathbf{f}_i$ is the net force on particle i from all other particles. This force can be decomposed into the contributions from each ion species. We again note that intra-species forces do not change the total kinetic energy of a given species, so we can write

$$\frac{\partial K_k^l}{\partial t} = \frac{1}{(\Delta r)^d} \sum_{j \in S_l} \sum_{\substack{i \in S_k, \\ \mathbf{r}_i \in A}} \mathbf{v}_i \cdot \mathbf{f}_{ij} \tag{18}$$

for the cross-species terms. The electric field due to species l can be computed by solving the screened Poisson equation with the charge density of species l:

$$\left(\triangle - \frac{1}{\lambda^2} \right) \phi_l = -\frac{1}{\varepsilon_0} \rho_l, \qquad \mathbf{E}_l = -\nabla_{\mathbf{r}} \phi_l$$

The quantity $\rho_l = \int f_k \, d\mathbf{v}$ can be approximated in MD by using $f_k \approx E[g_k]$ as before. At a given position $\mathbf{r}_n$, every term other than τ_{kl} in (17) can now be computed in MD. By pairing like species terms at specific positions, we can solve for every interspecies τ_{kl} at every position:

$$\tau_{kl}(\mathbf{r}_n) = \left(K_{kl}^{eq} - K_k \right) \Bigg/ \left(\frac{1}{(\Delta r)^d} \sum_{j \in S_l} \sum_{\substack{i \in S_k \\ \mathbf{r}_i \in [\mathbf{r}_n + \Delta \mathbf{r}]}} (\mathbf{v}_i \cdot \mathbf{f}_{ij}) - Z_k e \mathbf{E}_l \cdot n_k \mathbf{u}_k \right) \tag{19}$$

Once these cross-species τ_{kl} are known, they can be plugged into (14), and we can solve for the remaining τ_{kk} at each position:

$$\tau_{kk}(\mathbf{r}_n) = \left(\sum_m (f_{kk}^{eq} - E[g_k]) \log(E[g])\right) \Bigg/ \left(\sum_m \left[\frac{\partial E[g_k]}{\partial t}(\log(E[g])+1) - \mathbf{F}_k \cdot \mathbf{v}_m - \sum_{k \neq l} \frac{1}{\tau_{kl}(\mathbf{r}_n)} \sum_m (f_{kl}^{eq} - E[g_k]) \log(E[g])\right]\right). \tag{20}$$

Using (19) and (20), we can compute both the inter- and intraspecies $\tau_{kl}(\mathbf{r}_n)$ entirely in terms of MD variables. Once this is completed, the f_k can be used as initial conditions for a BGK simulation with τ_{kl} supplied as spatially-dependent parameters in the evolution. Ideally, this BGK simulation will be more accurate than one with τ_{kl} chosen in an ad hoc manner, and will be much faster to evolve than the full MD simulation.

Symmetry simplification

We focus on plasma problems that contain an additional symmetry. Our problems are initialized to be symmetric in y. In order to test the effectiveness of HMM, the fine model evolves in full two-dimensional motion, but the coarse model only has one spatial dimension and two velocity dimensions. In this way, the coarse model capitalizes on symmetry and is significantly less detailed than the fine model. Agreement between the two is indicative of the effectiveness of the approach. To this end, we assume the BGK model initial condition is independent of y. In practice, this means averaging out any y dependence in the MD model when we wish to compare the two.

We here demonstrate that an initial condition for every f_k that is independent of y remains independent of y for all time when evolved by the BGK model. We begin by explicitly writing the x and y dependence of every term in the evolution equation:

$$\frac{\partial f_k}{\partial t} + v_x \frac{\partial f_k}{\partial x} + v_y \frac{\partial f_k}{\partial y} + \frac{Z_k e}{m_k} E_x \frac{\partial f_k}{\partial v_x} + \frac{Z_k e}{m_k} E_y \frac{\partial f_k}{\partial v_y} = \sum_l \frac{f_{kl}^{eq} - f_k}{\tau_{kl}}$$

where E_x and E_y are the x and y components of the electric field, respectively. Consider an initial condition $f_i(x,y,v_x,v_y,0) = f_{i0}(x,v_x,v_y)$ for all species i. The first time step of size h for the distribution of species k is

$$f_k(x,y,v_x,v_y,h) = -hv_x \frac{\partial f_{k0}}{\partial x} - \frac{hZ_k e}{m_k} E_{x0} \frac{\partial f_{k0}}{\partial v_x} - \frac{hZ_k e}{m_k} E_{y0} \frac{\partial f_{k0}}{\partial v_y} + \sum_l \frac{hf_{kl}^{eq} - hf_{k0}}{\tau_{kl}}.$$

The derivative of f_{k0} with respect to y is zero by definition. We now consider the electric potential. Note that, because the charge density $\rho = \sum_l \int Z_l e f_l \, d\mathbf{v}$, if every f_l is independent of y, ρ will be as well. Thus, $\rho(x,y,0) = \rho_0(x)$. Let us look closely at the initial electric field and make the substitution $\mathbf{r}'' = \mathbf{r} - \mathbf{r}'$:

$$\begin{aligned}
E_0(x,y) &= -\int \frac{1}{4\pi\varepsilon_0}\rho_0(x)\cdot\nabla_{\mathbf{r}}\left(\frac{e^{-|\mathbf{r}-\mathbf{r}'|/\lambda}}{|\mathbf{r}-\mathbf{r}'|}\right) d\mathbf{r}' \\
&= -\int \frac{1}{4\pi\varepsilon_0}\rho_0(x-x'')\cdot\nabla_{\mathbf{r}''}\left(\frac{e^{|\mathbf{r}''|/\lambda}}{|\mathbf{r}''|}\right) d\mathbf{r}''
\end{aligned}$$

From this, we clearly see that E_0 does not depend on y. Looking at each component (and abusing notation such that $\mathbf{r}'' = \mathbf{r}'$ again):

$$\begin{aligned}
E_{x0}(x) &= -\frac{1}{4\pi\varepsilon_0}\iint \rho_0(x-x')\frac{\partial}{\partial x}\left(\frac{e^{-|\mathbf{r}'}/\lambda}{|\mathbf{r}'|}\right) dy'\,dx' \\
&= -\frac{1}{4\pi\varepsilon_0}\int x'\rho_0(x-x')\int \frac{e^{-|\mathbf{r}'|/\lambda}}{|\mathbf{r}'|^2}\left(\frac{1}{|\mathbf{r}'|}-\frac{1}{\lambda}\right) dy'\,dx' \\
E_{y0}(x) &= -\frac{1}{4\pi\varepsilon_0}\int \rho_0(x-x')\int \frac{\partial}{\partial y}\left(\frac{e^{-|\mathbf{r}'|/\lambda}}{|\mathbf{r}'|}\right) dy'\,dx' \\
&= -\frac{1}{4\pi\varepsilon_0}\int \rho_0(x-x')\left(\left.\frac{e^{-|\mathbf{r}'|/\lambda}}{|\mathbf{r}'|}\right|_{y'=-\infty}^{y'=\infty}\right) dx' \\
&= 0.
\end{aligned}$$

We see that E_{x0} is independent of y and $E_{y0} = 0$. Therefore, the value of f_k at $t = h$ is given by

$$f_k(x,y,v_x,v_y,h) = -hv_x\frac{\partial f_{k0}}{\partial x} - \frac{hZ_k e}{m_k}E_{x0}(x)\frac{\partial f_{k0}}{\partial v_x} + \sum_l \frac{hf_{kl}^{eq} - hf_{k0}}{\tau_{kl}}$$

Because all terms on the right hand side are independent of y, f_k at time h is independent of y. We can then take another time step, using the same arguments, to find that f_k remains independent of y as time evolves. By taking $h \to 0$, we show that an initial condition that is independent of y will remain so provided f_k is sufficiently well behaved. Because the system is being continually driven towards a sum of Maxwellian distributions, the necessary regularity conditions are forced upon the system by the collisional terms. Thus, the evolution equation becomes:

$$\begin{aligned}
&\frac{\partial f_k}{\partial t} + v_x\frac{\partial f_k}{\partial x} + \frac{Z_k e}{m_k}E(x)\frac{\partial f_k}{\partial v_x} = \sum_l \frac{f_{kl}^{eq} - f_k}{\tau_{kl}} \\
&E(x) = -\frac{1}{4\pi\varepsilon_0}\int x'\rho(x-x')\int \frac{e^{-|\mathbf{r}'|/\lambda}}{|\mathbf{r}'|^2}\left(\frac{1}{|\mathbf{r}'|}-\frac{1}{\lambda}\right) dy'\,dx' \\
&f_k(x,\mathbf{v},0) = f_{k0}(x,\mathbf{v}), \quad f_k(0,\mathbf{v},t) = f_k(L_x,\mathbf{v},t).
\end{aligned} \tag{21}$$

Simulation Details

Kinetic simulation

The evolution equation for the kinetic regime is (21). In order to compute f_k, we uniformly discretize the x, v_x, and v_y domains with step sizes Δx and Δv. The velocity domain is doubly infinite, so we must select a cutoff velocity L_v. This is reasonable in practice, as we observe very few particles with extremely high velocities in the MD simulation.

We first calculate the moments of the distribution f_k to express the number density n_k, bulk velocity $\mathbf{u}_k$, and temperature T_k of each species at a particular point x and time t:

$$n_k(x,t) = \int f_k(x,\mathbf{v},t)\,d\mathbf{v} \tag{22}$$

$$\mathbf{u}_k(x,t) = \frac{1}{n_k(x,t)} \int \mathbf{v} f_k(x,\mathbf{v},t)\,d\mathbf{v} \tag{23}$$

$$T_k(x,t) = \frac{m_k}{2n_k(x,t)} \int |\mathbf{v}-\mathbf{u}_k|^2 f_k(x,\mathbf{v},t)\,d\mathbf{v}. \tag{24}$$

We compute these integrals using the trapezoidal rule. We use the screened Poisson equation to compute the electric potential. We use the standard, periodic, second order finite difference representation for the Poisson operator, then linearly solve for ϕ. The derivative of ϕ is computed using a second order centered difference scheme. In two dimensions, this corresponds to a pair potential that is Yukawa to first order. An exact solution requires us to develop a methodology to compute the electric field integral accurately, or to simulate our system in three dimensions, in which case the screened Poisson equation is exact.

The collisional term is calculated directly, using a mixed Maxwellian equilibrium distribution with the proper mixture temperature for every pair of species. We then advance f one time step, using an operator splitting second order finite volume method with the minmod limiter for the advection, electric, and collisional terms.

Molecular dynamics simulation

The molecular dynamics simulation is governed by simple Hamiltonian dynamics. We evolve it using the symplectic velocity Verlet algorithm to guarantee energy conservation:

$$\begin{aligned}
\mathbf{v}_i\left(t+\frac{\Delta t}{2}\right) &= \mathbf{v}_i(t) + \frac{\Delta t}{2}\frac{\mathbf{F}_i(t)}{m_i} \\
\mathbf{r}_i(t+\Delta t) &= \mathbf{r}_i(t) + \Delta t\,\mathbf{v}_i\left(t+\frac{\Delta t}{2}\right) \\
\mathbf{v}_i(t+\Delta t) &= \mathbf{v}_i\left(t+\frac{\Delta t}{2}\right) + \frac{\Delta t}{2}\frac{\mathbf{F}_i(t+\Delta t)}{m_i}
\end{aligned}$$

where $\mathbf{r}_i$ and $\mathbf{v}_i$ are the two-dimensional position and velocity of particle i, and $\mathbf{F}_i$ is the force exerted on particle i due to all other particles, using the Yukawa potential to simulate electron screening. The Yukawa potential and force of particle j on particle i are given by

$$V_{ij} = \frac{Z_i Z_j e^2}{4\pi\varepsilon_0 |\mathbf{r}_i - \mathbf{r}_j|} e^{-\frac{|\mathbf{r}_i - \mathbf{r}_j|}{\lambda}}$$

$$\mathbf{F}_{ij} = -\nabla_{\mathbf{r}_i} V_{ij} = V_{ij}\left(\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} + \frac{1}{\lambda}\right)\frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|}$$

where V_{ij} is the electrostatic potential at the position of ion i due to ion j, and $\mathbf{F}_{ij}$ is the force exerted on ion i by ion j.

The potential energy, kinetic energy, and total energy are computed by

$$\mathrm{PE} = \sum_i \sum_{i<j} V_{ij} = \frac{1}{2}\sum_{i \neq j} V_{ij}$$

$$\mathrm{KE} = \sum_i \frac{1}{2} m_i |\mathbf{v}_i|^2$$

$$\mathrm{TE} = \mathrm{KE} + \mathrm{PE}.$$

The particle density, bulk velocity, and local temperature of species k in a region A are calculated by

$$n_k = \frac{N_k}{A}$$

$$\mathbf{u}_k = \frac{1}{N_k}\sum_{i \in S_k} \mathbf{v}_i$$

$$T_k = \frac{1}{N_{DoF}} \sum_{i \in S_k} m_k (\mathbf{v}_i - \mathbf{u}_k)^2$$

where N_k is the number of particles of species k in region A and N_{DoF} is the number of translational of degrees of freedom, which for a 2D system is

$$N_{DoF} = \begin{cases} 2N_k & \text{no bulk velocity prescribed,} \\ 2N_k - 2 & \text{bulk velocity prescribed.} \end{cases}$$

Our regions of interest are centered cells of width Δx (the discretization of x in the kinetic regime) stretching from 0 to L_y in the y direction.

When we place ions throughout the domain initially, particle correlations are unknown. As the particles move to a natural correlation, the temperature of the system will change. In order to counteract this effect, we use Langevin dynamics to drive the system towards a distribution with the desired particle density and temperature properties [3]. The velocity Verlet equations during this phase are modified, using the prescribed temperature, T:

$$\mathbf{v}_i\left(t + \frac{\Delta t}{2}\right) = \mathbf{v}_i(t) + \frac{\Delta t}{2}\frac{\mathbf{F}_i(t)}{m_i} - \frac{\gamma_i \Delta t}{2}\mathbf{v}_i(t) + \frac{\Delta t}{2m_i}\sigma\,\eta$$

$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \Delta t\,\mathbf{v}_i\left(t + \frac{1}{2}\Delta t\right)$$

$$\mathbf{v}_i(t + \Delta t) = \left(1 + \frac{\gamma_i \Delta t}{2}\right)^{-1}\left(\mathbf{v}_i\left(t + \frac{\Delta t}{2}\right) + \frac{\Delta t}{2}\frac{\mathbf{F}_i(t + \Delta t)}{m_i} + \frac{\Delta t}{2m_i}\sigma\,\eta\right).$$

where γ_i is the friction coefficient for ion i, σ is the desired velocity variance, which is related to the local temperature of the cell, and η a random variable drawn from the standard normal distribution. The thermostat functions by balancing the frictional γ_i with the velocity variance. The friction is connected to the variance through the fluctuation-dissipation relation [5], such that the variance of the normal distribution is

$$\sigma^2 = 2\gamma_i m_i T \delta(t - t').$$

The discretization of the delta function as $1/\Delta t$. In our Verlet algorithm, the time step is $\Delta t/2$, so the discretized version becomes

$$\frac{\Delta t}{2m_i}\sigma\eta = \frac{\Delta t}{2m_i}\sqrt{\frac{2\gamma_i m_i T}{\Delta t/2}}\eta = \sqrt{\frac{\gamma_i \Delta t T}{m_i}}\eta.$$

Over several hundred time steps, the system finds proper interparticle correlations consistent with the desired temperature.

Compression and reconstruction

Transitioning from the coarse to fine representations of the system and vice versa are nontrivial tasks that comprise the core of the HMM philosophy. We must consistently transition between regimes and compute τ_{kl} from the MD simulations.

Suppose at time t, we have a distribution $f_k(x, \mathbf{v}, t)$ from the kinetic simulation. At this time, we wish to update τ_{kl}, and so must sample this distribution to place ions in the two-dimensional MD domain. However, this distribution contains no knowledge of the particle correlations, complicating the task of placing particles in the MD domain. Naive particle placement necessarily lead to temperature changes in the initial steps of the MD simulation due to particles placed too near or far from one another, driving the distribution away from the desired initialization.

We begin by initializing the MD distribution *near* the distribution f_k. To do this, we compute the spatially dependent density $n_k(x)$ and temperature $T_k(x)$. Dividing the MD domain into the same number of cells as x as grid points in the kinetic simulation, we place particles in each cell to match the density in that cell. We initialize them using the Halton sequence, which yields a more correlated distribution than pseudorandom selections, allowing faster equilibration. We then run an equilibration phase, driving each cell toward the correct $T_k(x)$ with Langevin dynamics. At the right and left side of each cell, we enforce reflective boundary conditions, such that each ion must stay within the cell in which it was originally placed. The particles *do*, however, electrostatically interact with the ions in other cells. This model configuration is shown in Figure 5. The result is that the particles in each cell are correlated consistently with the prescribed density and temperature. The inter-cell interactions should smooth the coarse distribution into something consistent on the fine scale.

Once equilibrated, we throw away the velocities of each particle, and draw new velocities for every particle in cell x_i from the velocity distribution at x_i, $f_k(x_i, \mathbf{v}, t)$. Because this distribution is very discrete in the kinetic formulation, we linearly interpolate it before sampling. This method is able to accurately capture in MD at least the first four moments of the f_k distribution

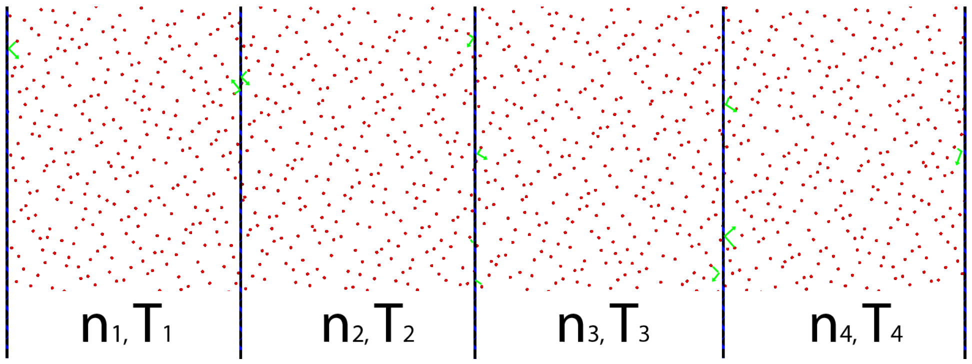


Figure 5: Configuration of MD simulation during equilibration phase.

with sufficient particles. Once this is done, the particles have correlated spatially and are distributed according to the correct velocity distribution. This is considered the "initial condition" for an MD simulation.

The MD simulation is run for a period of time, and the positions and velocities are recorded for use in compression. Each f_k is approximated by $E[g_k]$. That is, we record the number of ions of species k that have x positions in $[x, x+\Delta x]$ with velocities in $[\mathbf{v}, v+\Delta\mathbf{v}]$ at each timestep. The average of this is $E[g_k]$, and $E[g] = \sum_k E[g_k]$. The moments of $E[g_k]$ can be computed and used to define the f_{kl}^{eq} Maxwellian distributions.

Using the time series for $E[g_k]$, we can compute $\frac{\partial E[g_k]}{\partial t}$ using a second order centered difference scheme averaged over many timesteps. The gradient $\mathbf{F}_k = \nabla_{\mathbf{r}}(f_k \log(f))$ can be computed by using a centered difference spatial derivative on $E[g_k]\log(E[g])$. In the single species case, we have now computed every quantity needed to calculate $\tau(\mathbf{r}_n)$ from (14).

In the multispecies case, we now use the $E[g_k]$ to compute ρ_k, and solve the Poisson equation using a linear solve with the standard centered difference Poisson operator to compute each $\mathbf{E}_k$. The densities and bulk velocities n_k and $\mathbf{u}_k$ can be computed from the moments of $E[g_k]$. Finally, the expected interparticle forces $\mathbf{f}_{ij}$ and kinetic energies K_k are computed each timestep already, and so can be recorded. The equilibrium kinetic energies K_{kl}^{eq} are again determined by the equilibrium distributions.

From this, we can now compute every cross-species τ_{kl} using (19), and we then use these in (20) to compute the intraspecies τ_{kk}. Note that the $(\Delta r)^d$ is actually equal to $(\Delta x)L_y$ because we are averaging out the y dimension. This is the case for every spatial integral.

The simulation is run for a period of time. Positions and velocities are recorded, and used to compute the expected values needed to compute $\tau_{kl}(x)$ and $f_k(x, \mathbf{v}, t)$. Once the statistics on these quantities are sufficiently accurate, the MD simulation ends and the $\tau_{kl}(x)$ and $f_k(x, \mathbf{v}, t)$ are used to initialize another BGK simulation period.

We continue to compute $\frac{dH^{BGK}}{dt}$ during our BGK simulation. Once this changes sufficiently from its initial value, we consider the τ_{kl} values "stale" and reinitialize an MD simulation to update them.

Parametrization

We wish to study plasmas that meet certain collisionality criteria, such that a kinetic perspective is warranted. For this reason, it is useful to rewrite the governing equations in a parameterized, dimensionless form. We first choose our length scale as the ion-circle radius a. We then define a dimensionless coupling parameter Γ that describes the ratio of the potential energy to the kinetic energy of the system, a dimensionless screening parameter κ that describes the extent to which the background electrons screen the electrostatic field of the ions, and a plasma frequency ω_p that defines an important timescale. These parameters are defined as

$$a^2 = \frac{1}{\pi n} \tag{25}$$

$$\Gamma = \frac{Z^2 e^2}{4\pi\varepsilon_0 a T} \tag{26}$$

$$\kappa = \frac{a}{\lambda} \tag{27}$$

$$\omega_p^2 = \frac{Z^2 e^2 n}{2\varepsilon_0 a m}. \tag{28}$$

Since the properties of the system vary in space and time, it is convenient to define reference parameters a_0, ω_0, and m_0. We define a_0 based on the average density n_{avg}, so that $\pi a_0^2 n_{avg} = 1$. Therefore, on average there is a distance of a_0 between neighboring ions. It is also helpful to define ω_0 with $Z = 1$ for notational simplicity. The reference mass m_0 may be arbitrary. One choice is to use the mass of the lightest particle species of interest. The fundamental scaled variables, denoted by the tilde, are defined as:

Length:	$\tilde{x} = \dfrac{x}{a_0}$
Time:	$\tilde{t} = \omega_0 t$
Mass:	$\tilde{m} = \dfrac{m}{m_0}.$

From these we can construct the scaled versions of all derived quantities:

Velocity:	$\tilde{\mathbf{v}} = \dfrac{\mathbf{v}}{a_0\omega_0}$
Force:	$\tilde{F} = \dfrac{F}{a_0\omega_0^2 m_0}$
Energy:	$\tilde{U} = \dfrac{U}{a_0^2\omega_0^2 m_0}$
Distribution:	$\tilde{f} = \dfrac{f}{a_0^4\omega_0^2}$
Electric Potential:	$\tilde{\phi} = \dfrac{\phi e}{a_0^2\omega_0^2 m_0}$
Density:	$\tilde{n} = n a_0^2$
Temperature:	$\tilde{T} = \dfrac{T}{a_0^2\omega_0^2 m_0}.$

The coupling parameter and plasma frequency in any given region of the domain can then be written in terms of the dimensionless parameters:

$$\Gamma = \frac{Z^2\sqrt{\pi\tilde{n}}}{2\tilde{T}} \tag{29}$$

$$\omega_p^2 = \frac{Z^2(\pi\tilde{n})^{3/2}}{\tilde{m}}\omega_0^2. \tag{30}$$

In the weakly to moderately coupled regime that we are studying, Γ should be approximately $O(0.1-1)$. If the system is too weakly collisional, it will behave as an ideal gas, where collisions are not significant. If the system is very strongly collisional, then collisional effects dominate and the system is best be modeled with hydrodynamic equations. We wish to select a regime in which the kinetic scale is the most relevant. This formulation also places a constraint on the average dimensionless density of the system such that $\tilde{n}_{avg} = \frac{1}{\pi}$ so that a_0 corresponds to the average ion-circle radius in the domain. For our computations, we use a screening parameter $\kappa = 1$.

The dimensionless kinetic equations are nearly identical to the dimensional version, with the exception of the Poisson equation. The dimensionless distribution function, $\tilde{f}_k$, evolves according to

$$\frac{\partial \tilde{f}_k}{\partial \tilde{t}} + \tilde{v}_{\tilde{x}}\frac{\partial \tilde{f}_k}{\partial \tilde{x}} + \frac{\tilde{Z}_k}{\tilde{m}_k}\tilde{E}(\tilde{x})\frac{\partial \tilde{f}_k}{\partial \tilde{v}_{\tilde{x}}} = \sum_l \frac{\tilde{f}_{kl}^{eq} - \tilde{f}_k}{\tilde{\tau}_{kl}}$$

$$\tilde{E}(x) = -\frac{1}{2}\int \tilde{x}'\tilde{\rho}(\tilde{x}-\tilde{x}')\int \frac{e^{-\kappa|\tilde{\mathbf{r}}'|}}{|\tilde{\mathbf{r}}'|^2}\left(\frac{1}{|\tilde{\mathbf{r}}'|} - \kappa\right) d\tilde{y}'\, d\tilde{x}'$$

where $\tilde{\rho} = \sum_l Z_l \tilde{n}_l$. The moments of the distribution function become

$$\tilde{n}_k = \int \tilde{f}_k \, d\tilde{\mathbf{v}}$$

$$\tilde{\mathbf{u}}_k = \frac{1}{\tilde{n}_k}\int \tilde{\mathbf{v}}\tilde{f}_k \, d\tilde{\mathbf{v}}$$

$$\tilde{T}_k = \frac{\tilde{m}_k}{2\tilde{n}_k}\int |\tilde{\mathbf{v}} - \tilde{\mathbf{u}}_k|^2 \tilde{f}_k \, d\tilde{\mathbf{v}}.$$

As with the kinetic equations, the dimensionless MD equations are essentially unchanged. Only the potential term differs. The velocity Verlet algorithm becomes:

$$\tilde{\mathbf{v}}_i\left(\tilde{t} + \frac{\Delta\tilde{t}}{2}\right) = \tilde{\mathbf{v}}_i(\tilde{t}) + \frac{\Delta\tilde{t}}{2}\frac{\tilde{\mathbf{F}}_i(\tilde{t})}{\tilde{m}_i}$$

$$\tilde{\mathbf{r}}_i(\tilde{t} + \Delta\tilde{t}) = \tilde{\mathbf{r}}_i(\tilde{t}) + \Delta\tilde{t}\,\tilde{\mathbf{v}}_i\left(\tilde{t} + \frac{\Delta\tilde{t}}{2}\right)$$

$$\tilde{\mathbf{v}}_i(\tilde{t} + \Delta\tilde{t}) = \tilde{\mathbf{v}}_i\left(\tilde{t} + \frac{\Delta\tilde{t}}{2}\right) + \frac{\Delta\tilde{t}}{2}\frac{\tilde{\mathbf{F}}_i(\tilde{t} + \Delta\tilde{t})}{\tilde{m}_i}.$$

During the equilibration phase, the Langevin velocity Verlet algorithm becomes

$$\begin{aligned}
\tilde{\mathbf{v}}_i\left(\tilde{t}+\frac{\Delta\tilde{t}}{2}\right) &= \tilde{\mathbf{v}}_i(\tilde{t}) + \frac{\Delta\tilde{t}}{2}\frac{\tilde{\mathbf{F}}_i(\tilde{t})}{\tilde{m}_i} - \frac{\tilde{\gamma}_i\Delta\tilde{t}}{2}\left(\tilde{\mathbf{v}}_i(\tilde{t})\right) + \sqrt{\frac{\tilde{\gamma}_i\Delta\tilde{t}\,\tilde{T}}{\tilde{m}_i}}\eta \\
\tilde{\mathbf{r}}_i(\tilde{t}+\Delta\tilde{t}) &= \tilde{\mathbf{r}}_i(\tilde{t}) + \Delta\tilde{t}\,\tilde{\mathbf{v}}_i\left(\tilde{t}+\frac{\Delta\tilde{t}}{2}\right) \\
\tilde{\mathbf{v}}_i(\tilde{t}+\Delta\tilde{t}) &= \left(1+\frac{\tilde{\gamma}_i\Delta\tilde{t}}{2}\right)^{-1}\left(\tilde{\mathbf{v}}_i\left(\tilde{t}+\frac{\Delta\tilde{t}}{2}\right) + \frac{\Delta\tilde{t}}{2}\frac{\tilde{\mathbf{F}}_i(\tilde{t}+\Delta\tilde{t})}{\tilde{m}_i} + \sqrt{\frac{\tilde{\gamma}_i\Delta\tilde{t}\tilde{T}}{\tilde{m}_i}}\eta\right).
\end{aligned}$$

The Yukawa potential and force equations become

$$\begin{aligned}
\tilde{V}_{ij} &= \frac{\tilde{Z}_i\tilde{Z}_j}{2|\tilde{\mathbf{r}}_i-\tilde{\mathbf{r}}_j|}e^{-\kappa|\tilde{\mathbf{r}}_i-\tilde{\mathbf{r}}_j|} \\
\tilde{F}_{ij} &= \tilde{V}_{ij}\left(\frac{1}{|\tilde{\mathbf{r}}_i-\tilde{\mathbf{r}}_j|}+\kappa\right)\frac{\tilde{\mathbf{r}}_i-\tilde{\mathbf{r}}_j}{|\tilde{\mathbf{r}}_i-\tilde{\mathbf{r}}_j|}.
\end{aligned}$$

The dimensionless energies and moments are

$$\begin{aligned}
\tilde{\text{PE}} &= \frac{1}{2}\sum_{i\neq j}\tilde{V}_{ij} \\
\tilde{\text{KE}} &= \frac{1}{2}\sum_{i}\tilde{m}_i|\tilde{\mathbf{v}}_i|^2 \\
\tilde{\text{TE}} &= \tilde{\text{PE}}+\tilde{\text{KE}} \\
\tilde{n}_k &= \frac{N_k}{\tilde{A}} \\
\tilde{\mathbf{u}}_k &= \frac{1}{N_k}\sum_{i\in S_k}\tilde{\mathbf{v}}_i \\
\tilde{T}_k &= \frac{1}{N_{DoF}}\sum_{i\in S_k}\tilde{m}_k\left|\tilde{\mathbf{v}}_i-\tilde{\mathbf{u}}_k\right|^2.
\end{aligned}$$

The moments are time-averaged quantities, since thermal noise affects each of them to some degree.

Results and Discussion

We have the theoretical basis for linking the MD and BGK models into a consistent HMM solver, but the programmatic link between the two was incomplete at the time of writing. To provide an initial comparison, both moels were used to solve an identical interface problem. In this problem of interest, a single ion species was initialized in an isothermal domain, with zero bulk velocity, such that the center region has a density three times that of the outer region. The simulation was then allowed to evolve for a time of $120\omega_0^{-1}$. The simulation was stopped at this end time due to time constraints on the MD. In the 1D-2V BGK solver, the spatial domain was discretized into 32 cells. The model was then evolved assuming a screened Poisson equation for the electric field, and using a naive first-order estimate of the relaxation timescale, $\tau = \omega_0^{-1}$. The hydrodynamic moments in the corresponding spatial regions were tracked throughout the 2D-2V MD simulation, which used 32,000 particles, to facilitate comparison. The conditions at the beginning and end of the simulation are shown in Figures 6 and 7 with axes labeled in the nondimensional units.

From Figure 7, we observe that the MD and BGK simulations yield qualitatively and even quantitatively similar results, even with a very naively calculated τ.The noise that appears in the MD result is present mainly due to the small number of particles in each cell used to compute moments and the fact that time-averaging was not performed. In practice, a moving time averaging window would smooth the MD hydrodynamic moments. At the end time, the high density region in the BGK has expanded further than the high density region in the MD, suggesting the timescale associated with the BGK model is slightly faster than that for the MD in these conditions. This demonstrates the advantages we will gain when the HMM implementation allows us to inform τ through MD. The screened Poisson equation is not valid in 2D and only useful as a first-order approximation. This may also have played a role in the discrepancy.

The BGK simulation completed in under a minute, compared with over 24 hours for the MD simulation. The large speedup gained from the BGK model compared with MD, combined with the close results even with a first-order approximation to τ, suggests that very good results might be obtained with a better estimate of the collisional relaxation timescale using the method outlined above. We are optimistic that this implementation will provide stark benefits to plasma modeling in the near future.

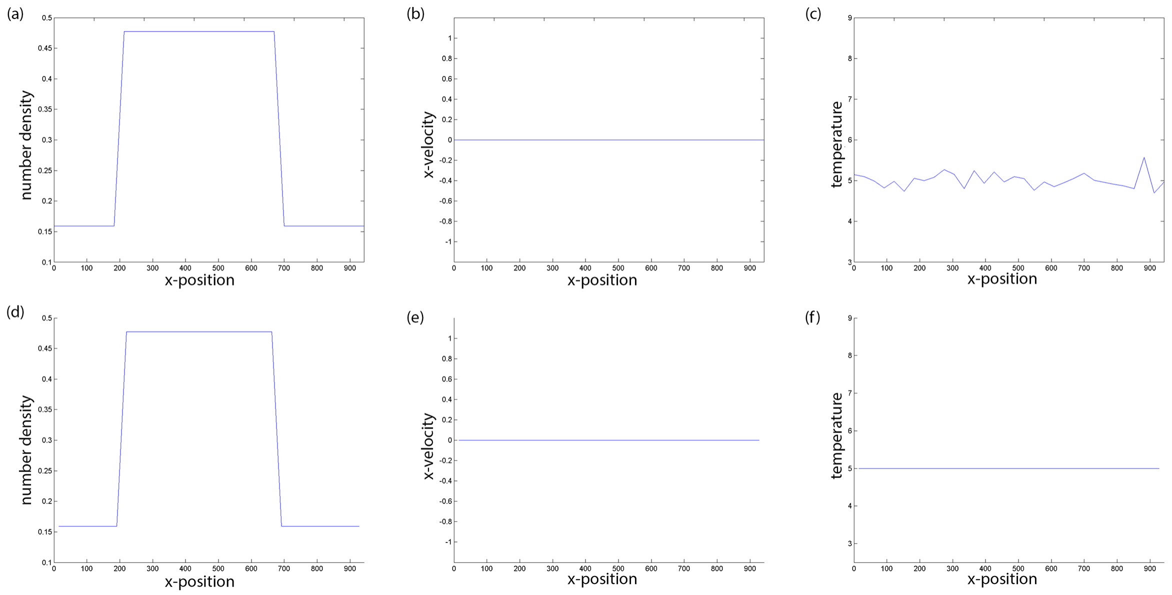


Figure 6: Initial conditions for the MD (top) and BGK (bottom) simulations. (a)-(c): zeroth through second moments computed from MD. (d)-(f): zeroth through second moments computed from BGK.

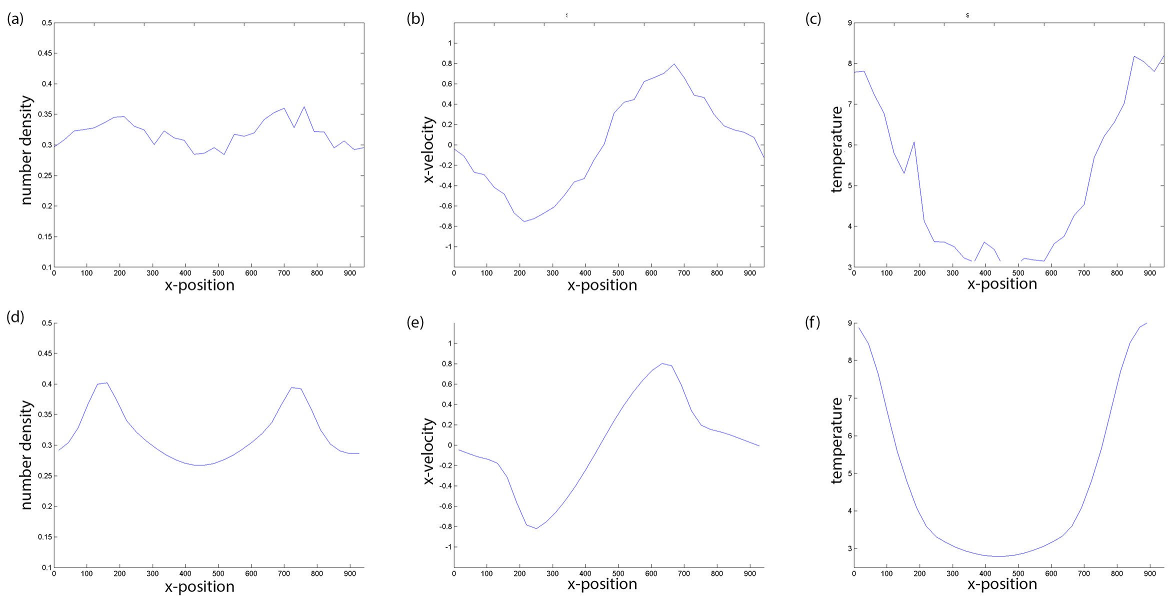


Figure 7: State at $t = 120\omega_0^{-1}$ for the MD (top) and BGK (bottom) simulations. (a)-(c): zeroth through second moments computed from MD. (d)-(f): zeroth through second moments computed from BGK.

Conclusions

The heterogeneous multiscale method is an exceptionally powerful tool for solving problems that contain multiple physical regimes. HMM provides a rigorous, well-defined means to build hybrid models that combine the accuracy of a fine-grained model with the computational efficiency of a coarse-grained model. This allows detailed and efficient exploration of multiscale phenomena. These phenomena include mesoscale systems that concurrently display macroscopic and microscopic behavior and systems with defects such as cracks or shocks.

In this report, we have demonstrated its power in simulating plasma using kinetic theory and molecular dynamics. Both models, given the same initial condition, evolve in nearly identical manners. Only the time scale of the kinetic model is incorrect. Once the models are coupled through τ_{kl}, the kinetic model will accurately capture the behavior of the full microscale model with a computational speedup factor of hundreds to thousands, depending on the problem.

This proof-of-principle paves the way for additional improvements in the method. First of all, the method is currently implemented in Matlab. Converting this to another language would increase the efficiency even further and allow three-dimensional simulations. Second, our method is eminently paralellizable. The τ_{kl} are computed as ensemble averages. The accuracy of this could be improved by running many MD simulations in parallel, greatly decreasing statistical noise by making use of the Law of Large Numbers. One can also foresee running MD simulations and BGK simulations concurrently, updating τ_{kl} continuously, rather than periodically. The potential for additional speedups is significant and enticing.

Furthermore, in this report we have created a fully consistent multiscale connection between molecular dynamics and kinetic theory. This methodology can be adapted with no change to the formulation to study systems with different potentials, systems that incorporate electron motion, and systems of three dimensions. It allows for n species with different charges and masses, and even interparticle potentials that differ more significantly than simply being scaled by the charge. Finally, we can modify our assumptions to connect MD to other kinetic models. For example, if we do not approximate f_{kl}, but instead compute it from MD, we can couple MD to the BBGKY hierarchy. We could also compute collisional cross-sections in order to couple MD with the Boltzmann equation.

These results are significant, as to our knowledge there does not exist an HMM model that couples molecular dynamics and kinetic theory. Our derivation here allows the coupling of MD to any number of kinetic theory formulations. Additionally, HMM has not, to our knowledge, been used to model plasmas. It is the opinion of the authors that HMM represents an important part of the future of computational physics. The applications include plasma modeling, mesoscale modeling, mathematical biology, high-accuracy hydrodynamic models, and countless more. This report is a definitive proof-of-concept of the applicability of HMM to plasma modeling, and further demonstrates the viability of HMM as a computational tool.

Acknowledgments

We would like to thank our mentors, Michael Murillo and Jeff Haack, for working with us on this project. Their project proposal was fascinating, and our many fantastic conversations with them led to countless breakthroughs. We also express our thanks to Jeff Haack and Mathieu Marciante, who provided the base codes for BGK and MD, respectively. Without these to build upon, we would have been unable to fully explore HMM in our brief time here. Thanks is also due to the Nambé group, which graciously allowed us to attend their quarterly meeting and present our results to plasma physicists from around the world.

Finally, we would like to thank the other students, administration, and lecturers of the Computational Physics Student Summer Workshop. Together, they gave us an enlightening, enriching, and enjoyable summer of research, and we hope we have established lasting relationships with the lab and our student colleagues.

References

[1] Pravin Bhat, Brian Curless, Michael Cohen, and Larry Zitnick. *Fourier Analysis of the 2D Screened Poisson Equation for Gradient Domain Problems*. 2008.

[2] Francis F. Chen. *Introduction to Plasma Physics and Controlled Fusion*.

[3] Philippe H Hünenberger. Thermostat algorithms for molecular dynamics simulations. In *Advanced computer simulation*, pages 105–149. Springer, 2005.

[4] U. L. Klimontovich and A Dobroslavsky. *The kinetic theory of electromagnetic processes*, volume 10. Springer-Verlag Berlin, 1983.

[5] Rep Kubo. The fluctuation-dissipation theorem. *Reports on progress in physics*, 29(1):255, 1966.

[6] Weiqing Ren and E Weinan. Heterogeneous multiscale method for the modeling of complex fluids and micro-fluidics. *Journal of Computational Physics*, 204(1):1–26, 2005.

[7] Víctor Romero-Rochin and Enrique González-Tovar. Comments on some aspects of boltzmannh theorem using reversible molecular dynamics. *Journal of statistical physics*, 89(3-4):735–749, 1997.

[8] Padma K Shukla and AA Mamun. *Introduction to dusty plasma physics*. CRC Press, 2001.

[9] Henning Struchtrup. *Macroscopic transport equations for rarefied gas flows*. Springer, 2005.

[10] Peter Andrew Sturrock. *Plasma Physics: An Introduction to the Theory of Astrophysical, Geophysical and Laboratory Plasmas*. Cambridge University Press, 1994.

[11] E Weinan. *Principles of multiscale modeling*. Cambridge University Press, 2011.

[12] E Weinan, Bjorn Engquist, Xiantao Li, Weiqing Ren, and Eric Vanden-Eijnden. Heterogeneous multiscale methods: a review. *Commun. Comput. Phys*, 2(3):367–450, 2007.

Survey of Multi-Material Closure Models in 1D Lagrangian Hydrodynamics

Team Members
David A. B. Hyde and J. Brad Maeng

Mentors
Joe Schmidt, Chris Malone, David Nicholaeff

Abstract

Accurately treating the coupled sub-cell thermodynamics of computational cells containing multiple materials is an inevitable problem in hydrodynamics simulations, whether due to initial configurations or evolutions of the materials and computational mesh. When solving the hydrodynamics equations within a multi-material cell, we make the assumption of a single velocity field for the entire computational domain, which necessitates the addition of a closure model to attempt to resolve the behavior of the multi-material cells constituents. In conjunction with a 1D Lagrangian hydrodynamics code, we present a variety of both the popular as well as more recently proposed multi-material closure models and survey their performances across a spectrum of examples. We consider standard verification tests as well as practical examples using combinations of fluid, solid, and composite constituents within multi-material mixtures. Our survey provides insights into the advantages and disadvantages of various multi-material closure models in different problem configurations.

Introduction

Mesh-based hydrodynamics simulation is traditionally divided into two schools of thought, Lagrangian and Eulerian. Both approaches have a number of strengths as well as shortcomings (see e.g. [6]), and both have found widespread use in hydrocodes over the past several decades ([5] gives a brief history and analysis). Arbitrary Lagrangian-Eulerian (ALE) schemes aim to combine the benefits of these two methodologies, though they also are vulnerable to inheriting the weaknesses of both schemes ([2] discusses Lagrangian, Eulerian, and ALE methods in hydrocodes). Simulations involving multiple materials are commonly performed using any of these three classes of methods.

Cells containing multiple materials may appear in Lagrangian, Eulerian, and ALE hydrodynamics simulations. In a Lagrangian simulation, one might imagine initializing a two-dimensional fluid domain with a circular region of one material surrounded by a region of another material. If using a Cartesian mesh, there are inevitably multi-material cells at the interface of the two regions, regardless of the inclusion of techniques like adaptive mesh refinement (AMR). In an Eulerian simulation, the movement of fluids throughout the fixed computational domain again makes it inevitable that multi-material cells will occur at some point during the simulation. In ALE methods (as a trivial example, a Lagrange-plus-remap scheme), a combination of the initial configuration and the evolution of the domain and materials will lead to the presence multi-material cells. Regardless of the underlying scheme, there are coupled interactions between the constituent materials of multi-material cells that are generally ignored by hydrodynamics algorithms.

Resolving the dynamics of constituent materials within a multi-material cell is a topic of practical import. In any simulation, the modeler makes a choice of the finest resolution to be used in their simulation. Thus, even with AMR, there is some limit where physics cannot be captured due to ignoring sub-cell dynamics. Models for sub-cell dynamics attempt to capture all the coupled physics that occurs below the level of mesh resolution, thus providing more accurate simulation results (especially along multi-material interfaces) at potentially coarser mesh resolutions. However, sub-cell dynamics cannot be approximated for free.

In order to approximate the physics occurring below mesh resolution, additional variables are introduced for multi-material cells. These variables may include the volume fractions of the constituent materials (what fraction of the cell each constituent occupies) or the location of the multi-material interface. We assume that a single velocity field is available for the computational domain, which is a common assumption for multi-material cell modeling. Due to these considerations, when we solve the hydrodynamics equations governing the fluid simulation, we are left with more unknowns than equations – an unclosed system.

Consider, for example, a one-dimensional simulation with a two-material cell. In the cell, there are five hydrodynamics equations: there is a set of three conservation equations for each material, with the two momentum conservation equations combining due to having a single velocity field. There are two equations of state – one for each material – and finally, there is a constraint on the volume fractions that they sum to 1 (we ignore tracking the interface location for the time being). The variables in our system include pressure, density, specific internal energy, and volume fraction for each material, as well as the single velocity vector. Adding these numbers, we find we are left with eight equations for nine unknowns. To provide the additional information needed to solve our system and to attempt to resolve the physics within multi-material cells, we use multi-material closure models.

Multi-material closure models are methods that close the hydrodynamics equations under the addition of extra variables to track the physical quantities of interest within multi-material cells. There is a wide variety of multi-material closure models that has burgeoned over the past several decades (see [7, 11] for comparisons of several models). These models are based on different assumptions about the underlying physics and configurations of multi-material cells. Additionally, the models use different mathematical methods to approximate the evolution of constituents in the multi-material cells. With the great diversity in multi-material closure models, it remains an open question as to what is the most appropriate multi-material closure model to employ in a given simulation.

In this work, we survey a number of multi-material closure models, from older methods that have been used in production codes for decades to newer methods that may become prevalent in next-generation codes. We implement a one-dimensional predictor-corrector staggered-grid Lagrangian hydrodynamics code as the testbed for our computational experimentation. In conjunction with this code, we implement an assortment of multi-material closure models from the literature in an attempt to cover a broad swathe of theoretical approaches to multi-material modeling. Using our closure-model–equipped hydrocode, we run a number of test cases using combinations of gas, liquid, solid, and composite materials. After defining criteria for evaluating them, we analyze how the considered multi-material models perform against this gamut of problems (in addition to verifying that our implementations agree with expectations from literature and mathematics). We conclude by offering general recommendations for which multi-material closure models are most appropriate for which situations and by suggesting avenues for extension and use of our survey. Throughout, we offer insight into the assumptions behind the multi-material closure models we consider (and, consequently, their expected or surprising success and failure modes).

Governing Equations

The Conservation Laws

Also known as the Euler equations in differential form are

$$\partial_t \rho + \nabla \cdot \rho \mathbf{u} = 0 \tag{1a}$$

$$\partial_t \rho \mathbf{u} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p = 0 \tag{1b}$$

$$\partial_t E + \nabla \cdot (\mathbf{u}(E + p)) = 0 \tag{1c}$$

where the total energy is defined as $E = \rho(e + \frac{1}{2}u^2)$.

Lagrangian Hydrodynamics Equations

Now we reformulate the conservation laws in Eulerian frame, Equation (1), to Lagrangian frame.

Conservation of Mass

$$\partial_t \rho + \nabla \cdot \rho \mathbf{u} = 0 \tag{2a}$$

$$\partial_t \rho + \rho \nabla \cdot \mathbf{u} + \mathbf{u} \cdot \nabla \rho = 0 \tag{2b}$$

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{u} = 0 \tag{2c}$$

Since $\rho = \frac{M}{V}$,

$$\frac{D\frac{1}{V}}{Dt} + \frac{1}{V} \nabla \cdot \mathbf{u} = 0 \tag{3}$$

Consider

$$\frac{DV\frac{1}{V}}{Dt} = 0 = \frac{1}{V}\frac{DV}{Dt} + V\frac{D\frac{1}{V}}{Dt} \Rightarrow \frac{D\frac{1}{V}}{Dt} = -\frac{1}{V^2}\frac{DV}{Dt} \tag{4}$$

$$\boxed{\frac{1}{V}\frac{DV}{Dt} = \nabla \cdot \mathbf{u}} \tag{5}$$

This describes the fundamental Lagrangian representation of fluid flow. The discretized equation is

$$\frac{1}{V_i}\frac{DV_i}{Dt} = (\nabla \cdot \mathbf{u})_i. \tag{6}$$

It is utilized in two ways. First, given the velocity field $\mathbf{u}_i$ in discretized space, the evolution of volume $V_i(t)$ can be obtained. Second, given a prescription of volume $V_i(t)$ as a function of some coordinate $\mathbf{R}_j$, so that $V_i(t) = V_i(\mathbf{R}_1(t), \mathbf{R}_2(t), \ldots)$, the divergence of the velocity field, $(\nabla \cdot \mathbf{u})_i$, in discretized form is obtained.

Conservation of Momentum

$$\partial_t \rho \mathbf{u} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p = 0 \tag{7a}$$

$$\mathbf{u}\partial_t \rho + \rho \partial_t \mathbf{u} + \mathbf{u}(\nabla \cdot \rho \mathbf{u}) + (\rho \mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = 0 \tag{7b}$$

Cancelling out continuity terms, we arrive at the momentum equation.

$$\boxed{\rho \frac{D\mathbf{u}}{Dt} = -\nabla p} \tag{8}$$

Specific Internal Energy

$$\partial_t E + \nabla \cdot (\mathbf{u}(E + p)) = 0 \tag{9a}$$

$$\rho \partial_t \left(e + \frac{\mathbf{u}^2}{2}\right) + \underbrace{\left(e + \frac{\mathbf{u}^2}{2}\right)\partial_t \rho + \left(e + \frac{\mathbf{u}^2}{2}\right)(\nabla \cdot \rho \mathbf{u})}_{(E/\rho)(\partial_t \rho + \nabla \cdot \rho \mathbf{u}) = 0} + (\rho \mathbf{u} \cdot \nabla)\left(e + \frac{\mathbf{u}^2}{2}\right) + \nabla \cdot \rho \mathbf{u} = 0 \tag{9b}$$

$$\rho \frac{D\left(e + \frac{\mathbf{u}^2}{2}\right)}{Dt} + \nabla \cdot \rho \mathbf{u} = 0 \tag{9c}$$

Now subtracting the product of momentum equation and **u**, we have the specific internal energy equation.

$$\boxed{\rho \frac{De}{Dt} = -p\nabla \cdot \mathbf{u}} \tag{10}$$

Total Energy Equation

The total energy equation can be formulated by adding the product of momentum equation and **u** and the specific internal energy equation, and integrating over domain D.

$$\int_D \left(\frac{\rho}{2} \frac{Du^2}{Dt} + \rho \frac{De}{Dt} \right) dV = -\int_D (\mathbf{u} \cdot \nabla p + p\nabla \cdot \mathbf{u})\, dV = -\oint_{\partial D} p\mathbf{u} \cdot d\mathbf{S} \tag{11}$$

Compatibility - Semi-Discrete Form

When discretizing a continuum system of conservation laws of fluid dynamics, one should try to incorporate as many mathematical properties of the continuum system on which it relies. The numerical error observed in the total energy associated with the physical model results from any inconsistencies that exist in discrete form of the continuum system. The mathematical relations that must be obeyed to achieve this are the discrete analog of the vector identities that involve the dependent variables of the physical system. Numerical algorithms constructed in this manner are said to be *compatible*, meaning the forms of the discrete terms that compose them are specified with the physics in mind [3].

Semi-discrete form of Lagrangian hydrodynamic equations

Consider the momentum equation Equation (8) and integrate it over a volume element D_p defined at vertex point p.

$$m_p \frac{D\mathbf{u}_p}{Dt} = -\int_{D_p} \nabla p dV = -\oint_{\partial D_p} p d\mathbf{S} = \sum_z \mathbf{f}_{zp} \tag{12}$$

where z is the zone or cell index and $\mathbf{f}_{zp}$ is the *force* vector, $\mathbf{f}_{zp} = p_z \cdot D_p$, where D_p is the control volume centered at point p. By definition, the force vectors point out of the control volume.

Define the total energy in zone z as

$$E_z = m_z e_z + \sum_p m_{pz} \mathbf{u}_p^2/2. \tag{13}$$

Now, take the total derivative with respect to time and substitute the momentum equation from Equation (12)

$$\frac{DE_z}{Dt} = m_z \frac{De_z}{Dt} + \sum_p \frac{m_{pz}\mathbf{u}_p}{m_p} \cdot \sum_{z'}^{p} \mathbf{f}_{z'p}. \tag{14}$$

Summing over all zones

$$\frac{D}{Dt}\left(\sum_z E_z \right) = \sum_z m_z \frac{De_z}{Dt} + \sum_z \sum_p \frac{m_{pz}\mathbf{u}_p}{m_p} \cdot \sum_{z'}^{p} \mathbf{f}_{z'p}. \tag{15}$$

The total mass is $\sum_z m_{pz} = m_p$. Using this fact in Equation (13) and Equation (15) gives the result for conservation of total energy over all zones as

$$\frac{D}{Dt}\left(\sum_z m_z e_z + \sum_p m_{pz}\mathbf{u}_p^2/2\right) = \sum_z m_z \frac{De_z}{Dt} + \sum_p \sum_z \mathbf{f}_{zp} \cdot \mathbf{u}_p \tag{16}$$

$$= \sum_z \left(m_z \frac{De_z}{Dt} + \sum_p \mathbf{f}_{pz} \cdot \mathbf{u}_p\right) + \sum_i \mathbf{f}_{bd,i} \cdot \mathbf{u}_{bd,i}. \tag{17}$$

Now if the sum over zones in the second form of Equation (17) is set to zero for each zone z, then

$$m_z \frac{De_z}{Dt} = -\sum_p \mathbf{f}_{pz} \cdot \mathbf{u}_p \equiv -\int_D p\nabla \cdot \mathbf{u} dV. \tag{18}$$

Equation (18) is the integral form of the internal energy equation in Equation (10) with $\mathbf{f}_{pz}$, pressure forces acting on points p in zone z. Therefore, compatibility is naturally obtained for control volume differencing in Cartesian geometry for arbitrary number of dimensions.

The left hand side of Equation (17), the total energy over the entire domain at time t is defined as

$$E(t) = \sum_z m_z e_z + \sum_p m_p \mathbf{u}_p^2/2. \tag{19}$$

The total energy integrated over time is

$$E(t) = E(0) + \sum_{m=1}^{n} \Delta t_m \sum_i \mathbf{f}_{bd,i} \cdot \mathbf{u}_{bd,i}^{m+1/2}. \tag{20}$$

This is discrete analog of Equation (11) and implies that only work done by the external forces at the boundary of the computational domain can increase or decrease the total energy of the system.

Compatibility - Fully Discrete Form

The time variation of the total energy equation Equation (20) results

$$\sum_z m_z \Delta e_z + \sum_p m_p \mathbf{u}_p^{n+1/2} \cdot \Delta \mathbf{u}_p = \Delta t \sum_i \mathbf{f}_{bd,i}^{\sigma} \cdot \mathbf{u}_{bd,i}^{n+1/2}. \tag{21}$$

The time centered velocity $\mathbf{u}_p^{n+1/2} \equiv (\mathbf{u}_p^{n+1} + \mathbf{u}_p^n)/2$ follows directly from the time variation of the kinetic energy defined at the points since $\mathbf{u}_p^{n+1/2} \cdot \Delta\mathbf{u}_p = [(\mathbf{u}_p^2)^{n+1} - (\mathbf{u}_p^2)^n]/2$ where $\Delta\mathbf{u}_p \equiv \mathbf{u}_p^{n+1} - \mathbf{u}_p^n$, $\Delta e_z \equiv e_z^{n+1} - e_z^n$ and superscript σ is some intermediate time level.

The time discrete form of momentum equation, Equation (12), is

$$m_p \Delta\mathbf{u}_p = m_p \frac{\mathbf{u}_p^{n+1} - \mathbf{u}_p^n}{\Delta t} = \sum_z \mathbf{f}_{zp}^{\sigma} \Delta t \tag{22}$$

where again superscript σ is some intermediate time level between n and $n+1$.

Since both the internal energy and the kinetic energy must be defined at the same time level so that the total energy is at a single time level, it follows that we must use and even time integration scheme.

Substituting Equation (22) into Equation (21) gives

$$\sum_z m_z \Delta e_z + \sum_p \mathbf{u}_p^{n+1/2} \cdot \sum_z \mathbf{f}_{zp}^{\sigma} \Delta t = \Delta t \sum_i \mathbf{f}_{bd,i}^{\sigma} \cdot \mathbf{u}_{bd,i}^{n+1/2}. \tag{23}$$

The crucial step is the interchange in the order of the double discrete summation on the left hand side of this equation. It is equivalent to a discrete integration by parts. Regrouping

$$\sum_z \left[m_z \Delta e_z + \sum_p \mathbf{u}_p^{n+1/2} \cdot \mathbf{f}_{pz}^{\sigma} \Delta t \right] = \Delta t \sum_i \mathbf{f}_{bd,i}^{\sigma} \cdot \mathbf{u}_{bd,i}^{n+1/2}. \tag{24}$$

The final step is to satisfy Equation (24) in the strong form by setting the quantity in brackets zero for all z. This yields the same equation as Equation (18) but with additional conclusion that $\mathbf{u}_p = \mathbf{u}_p^{n+1/2}$. Finally, the fully discrete specific internal evolution equation, Equation (18), is

$$m_z \Delta e_z = m_z \frac{e_z^{n+1} - e_z^n}{\Delta t} = -\sum_p \mathbf{f}_{pz}^{\sigma} \cdot \mathbf{u}_p^{n+1/2} \Delta t. \tag{25}$$

where again superscript σ is some intermediate time level between n and $n+1$. Notice that the half time step velocity, $\mathbf{u}_p^{n+1/2}$, must be used for compatibility.

Multi-Material Closure Models

In this section, the multi-material closure models considered in this research are investigated.

Staggered Grid Hydrodynamics Lagrangian Code

One dimensional staggered grid hydrodynamics (SGH) hydrocode was developed to investigate various multi-material closure models. There are many variations of the SGH Lagrangian hydrocodes available in various literatures. The one presented in this report is one variation of the widely known predictor-corrector SGH code, [4].

In SGH codes, the thermodynamic quantities such as density, ρ, specific internal energy, e, and pressure, p, are stored at the cell center. The vertices are initialized with the velocity $\mathbf{u}$. A generic algorithm for a SGH code is adopted from [4] and presented in the following section.

Algorithm

- The predictor step:

  1. For each cell/zone, z:

     (a) Compute divergence of velocity field: $(\nabla \cdot \mathbf{u})^n = \frac{u^n_{p+1} - u^n_p}{x_{p+1} - x_p}$.

     (b) Compute volume ratio: $d^{n+1/2} = \left(1 + (\nabla \cdot \mathbf{u})^n \frac{\Delta t}{2}\right)$.

     (c) Compute mid-time volume: $V^{n+1/2} = V^n d^{n+1/2}$.

     (d) Compute artificial viscosity: q^n.

     (e) For all species, i, in z compute component properties. In case of pure material, skip this.

     (f) Compute common density: $\rho^{n+1/2} = \sum_i \alpha_i^{n+1/2} \rho_i^{n+1/2}$

     (g) Compute common internal energy: $(\rho e)^{n+1/2} = \sum_i \alpha_i^{n+1/2} \rho_i^{n+1/2} e_i^{n+1/2}$.

     (h) Compute common pressure: $p^{n+1/2} = \sum_i \alpha_i p_i^{n+1/2}$.

- The corrector step:

  1. For each vertex, p:

     (a) Compute velocity: $\mathbf{u}^{n+1} = \mathbf{u}^n - \frac{\nabla\left(p^{n+1/2} + q^{n+1/2}\right)\Delta t}{\rho^n}$.

     (b) Compute mid-time velocity: $\mathbf{u}^{n+1/2} = \frac{\mathbf{u}^{n+1} + \mathbf{u}^n}{2}$. Implied conservation of energy.

  2. For each zone, z:

     (a) Compute divergence of velocity field: $(\nabla \cdot \mathbf{u})^{n+1/2} = \frac{u^{n+1/2}_{z+1} - u^{n+1/2}_z}{x_{p+1} - x_p}$.

     (b) Compute volume ratio: $d^{n+1} = \left(1 + (\nabla \cdot \mathbf{u})^{n+1/2} \Delta t\right)$.

     (c) Compute mid-time volume: $V^{n+1} = V^n d^{n+1}$.

     (d) Compute artificial viscosity: $q^{n+1/2}$.

(e) For all species, i, in z compute component properties. In case of pure material, skip this.

(f) Compute common density: $\rho^{n+1} = \sum_i \alpha_i^{n+1} \rho_i^{n+1}$.

(g) Compute common internal energy: $(\rho e)^{n+1} = \sum_i \alpha_i^{n+1} \rho_i^{n+1} e_i^{n+1}$.

(h) Compute common pressure: $p^{n+1} = \sum_i \alpha_i p_i^{n+1}$.

Equal Compressibility Model

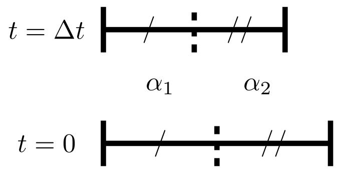


Figure 1: Equal compressibility model

The equal compressibility model is one of the simplest one-step multi-material closure models, [4]. In Figure 1, an illustration of the equal compressibility model is shown. For simplicity, we will only consider two material configurations. At time $t = 0$, the multi-material cell is initialized with material volume fractions

$$\alpha_1^n = \frac{V_1^n}{V^n} \quad \text{and} \tag{26}$$

$$\alpha_2^n = \frac{V_2^n}{V^n}. \tag{27}$$

The equal compressibility closure model assumes that the compressibility of each material stays constant throughout the closure model procedure. For this reason, the volume fractions of each material stay constant by maintaining the ratio of component volumes the same before and after the closure model procedure.

$$\alpha_1^{n+1} = \frac{V_1^{n+1}}{V^{n+1}} \quad \text{and} \tag{28}$$

$$\alpha_2^{n+1} = \frac{V_2^{n+1}}{V^{n+1}}. \tag{29}$$

Corrector Stage

The algorithm for the equal compressibility model for the corrector stage is described. The predictor stage is analogous to the corrector stage. We take the volume fractions of each material, i,

$$\alpha_i^{n+1} = \alpha_i^n. \tag{30}$$

Then, the density and the specific internal energy values of the multi-material components can be computed

$$V_i^{n+1} = V^{n+1}\alpha_i^{n+1} \tag{31}$$

$$\rho_i^{n+1} = m_i^{n+1}/V_i^{n+1} \tag{32}$$

$$e_i^{n+1} = e_i^n - \left[\frac{p_i^{n+1/2} + q_i^{n+1/2}}{\rho_i^n}\right](\nabla\cdot\mathbf{u})^{n+1/2}\Delta t. \tag{33}$$

where $(\nabla\cdot\mathbf{u})^{n+1/2}$ is the divergence of velocity at time step $n+1/2$. Lastly, the pressures of components can be computed from the equations of state

$$p_i^{n+1} = \mathscr{P}_i\left(\rho_i^{n+1}, e_i^{n+1}\right). \tag{34}$$

Equal Velocity Increment Model

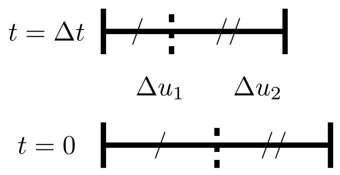


Figure 2: Equal velocity increment model

The equal velocity increment model shares the same philosophy as the equal compressibility model. The velocity increments of individual material in the multi-material cell are approximated then kept constant throughout the model procedure, [11]. In Figure 2, a simple illustration of the equal compressibility model is shown. For simplicity, we will only consider two material configurations.

Corrector Stage

The algorithm for the corrector stage is shown here. The predictor stage is analogous to the corrector stage. In the acoustic approximation for plane waves, for each material i the divergence of velocity field is approximated as,

$$\nabla\cdot\mathbf{u}_i = -\frac{\delta\rho_i}{\rho_i\Delta t} = -\frac{\delta u_i}{c_i\Delta t} \tag{35}$$

The above approximation along with the volume additivity requirement, we have for corrector stage

$$(\nabla\cdot\mathbf{u}_i)^{n+1/2} = \frac{c}{c_i}(\nabla\cdot\mathbf{u})^{n+1/2} \tag{36}$$

where the common speed of sound is defined as $c = \frac{1}{\sum_i \alpha_i / c_i}$. Then, the density and the specific internal energy values of the multi-material components can be computed

$$V^{n+1} = V_i^n \left(1 + (\nabla \cdot \mathbf{u}_i)^{n+1/2} \Delta t\right) \tag{37}$$

$$\rho_i^{n+1} = m_i^{n+1} / V_i^{n+1} \tag{38}$$

$$e_i^{n+1} = e_i^n - \left[\frac{p_i^{n+1/2} + q_i^{n+1/2}}{\rho_i^n}\right] (\nabla \cdot \mathbf{u}_i)^{n+1/2} \Delta t. \tag{39}$$

Lastly, the pressure values of individual components can be computed from the equations of state

$$p_i^{n+1} = \mathscr{P}_i \left(\rho_i^{n+1}, e_i^{n+1}\right). \tag{40}$$

Tipton's Pressure Relaxation Model

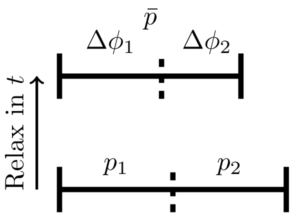


Figure 3: Tipton's pressure relaxation model

In Figure 3, an illustration of Tipton's pressure relaxation model is shown. The goal of Tipton's pressure relaxation closure model is to find volume changes for each material such that they sum to the total volume change of the multi-material cell and to find a common pressure which will be used in the momentum and material internal energy equations, as described in [1]. Tipton's pressure relaxation closure model consists of predictor and corrector stages.

Predictor Stage

In the predictor stage, Tipton's pressure relaxation closure model aims to relax pressures to the common pressure.

$$p_z^{n+1/2} = p_{z,i}^{n+1/2} + R_{z,i}^{n+1/2} \tag{41}$$

where z index is the cell/zone index and i is the material in the cell/zone. The relaxation term, $R_{z,i}^{n+1/2}$, emulates bulk viscosity, and is

$$R_{z,i}^{n+1/2} = -L_z^n \rho_{z,i}^n c_{z,i}^n \frac{1}{V_{z,i}^n} \frac{\Delta V_{z,i}^{n+1/2}}{\Delta t / 2}. \tag{42}$$

Assuming the pressure change in $\Delta t/2$ is isentropic ($\frac{dSz,i}{dt} = 0$), the pressure is

$$p_z^{n+1/2} = p_{z,i}^n - \underbrace{\rho_{z,i}^n (c_{z,i}^n)^2 \Delta V_{z,i}^{n+1/2} / V_{z,i}^n}_{\approx dp_{z,i}^n = \frac{dp_{z,i}}{d\rho_{z,i}}|_{S_{z,i}} d\rho_{z,i} + \frac{dp_{z,i}}{dS_{z,i}}|_{\rho_{z,i}} dS_{z,i}}. \tag{43}$$

Both terms in the right hand side of Equation (41) are linear functions of $\Delta V_{z,i}^{n+1/2}$. Relating the above to the total volume change of the cell, we arrive at the explicit solution.

$$\Delta V_{z,i}^{n+1/2} = \left(\frac{\alpha_{z,i}^n}{B_{z,i}^n}\bar{B}_z^n\right)\Delta V_z^{n+1/2} + \frac{V_{z,i}^n}{B_{z,i}^n}(p_{z,i}^n - \bar{p}_z^n) \tag{44}$$

$$p_z^{n+1/2} = \bar{p}_z^n - \bar{B}_z^n \Delta V_z^{n+1/2}/V_z^n \tag{45}$$

where

$$B_{z,i}^n = \rho_{z,i}^n (c_{z,i}^n)^2 \left[1 + L_z^n/(c_{z,i}^n \Delta t/2)\right] \tag{46}$$

$$\bar{B}_z^n = 1/(\sum_{i\in M(z)} \frac{\alpha_{z,i}^n}{B_{z,i}^n}) \tag{47}$$

$$\bar{p}_z^n = \sum_{i\in M(z)} \left(\frac{\alpha_{z,i}^n}{B_{z,i}^n} p_{z,i}^n\right) / \left(\sum_{i\in M(z)} \frac{\alpha_{z,i}^n}{B_{z,i}^n}\right). \tag{48}$$

Instead of working with the changes in material volumes, the approximate equations in terms of changes of volume fraction is used; definition of volume fraction and Equation (44) are used. Then, the preliminary change in volume fraction

$$\Delta\alpha_{z,i}^{n+1/2} = \alpha_{z,i}^{n+1/2} - \alpha_{z,i}^n \tag{49}$$

can be limited with respect to the current volume fraction.

$$\widetilde{\Delta\alpha}_{z,i}^{n+1/2} = \mathrm{sign}(\Delta\alpha_{z,i}^{n+1/2})\min(|\Delta\alpha_{z,i}^{n+1/2}|, \psi\alpha_{z,i}^n), \quad \psi = 1/4. \tag{50}$$

Renormalization is required if $\sum_{i\in M(z)} \widetilde{\Delta\alpha}_{z,i}^{n+1/2} \neq 0$ in order to make sure the renormalized volume fraction changes sum to 0. Skipping some details regarding limiting (for more details see [1]), the limited volume fraction changes are calculated as shown.

If $\widetilde{\Delta\alpha}_z^- > \widetilde{\Delta\alpha}_z^+$,

$$\widetilde{\widetilde{\Delta\alpha}}_{z,i}^{n+1/2} = \begin{cases} \frac{\Delta\alpha_z^+}{\Delta\alpha_z^-}\widetilde{\Delta\alpha}_{z,i}^{n+1/2} & \text{if} \quad \widetilde{\Delta\alpha}_{z,i}^{n+1/2} < 0, \\ \widetilde{\Delta\alpha}_{z,i}^{n+1/2} & \text{if} \quad \widetilde{\Delta\alpha}_{z,i}^{n+1/2} > 0 \end{cases} \tag{51}$$

If $\widetilde{\Delta\alpha}_z^- < \widetilde{\Delta\alpha}_z^+$,

$$\widetilde{\widetilde{\Delta\alpha}}_{z,i}^{n+1/2} = \begin{cases} \frac{\Delta\alpha_z^-}{\Delta\alpha_z^+}\widetilde{\Delta\alpha}_{z,i}^{n+1/2} & \text{if} \quad \widetilde{\Delta\alpha}_{z,i}^{n+1/2} > 0, \\ \widetilde{\Delta\alpha}_{z,i}^{n+1/2} & \text{if} \quad \widetilde{\Delta\alpha}_{z,i}^{n+1/2} < 0 \end{cases} \tag{52}$$

$\widetilde{\Delta\alpha}_z^{+/-}$ are absolute values of the sums of total change in volume fraction of the same sign $(+/-)$.

Corrector Stage

The common pressure we obtained from the predictor stage is used in the momentum equation.

$$m_p \frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \sum_{z \in Z(p)} \mathbf{f}_{zp}^{p^{n+1/2}} = \sum_{z \in Z(p)} p_z^{n+1/2} \mathbf{S}_{pz} \tag{53}$$

The node positions and volumes are updated in the usual way. The volume fraction to be used to model multi-material properties is extrapolated to full time step.

$$\Delta\alpha_{z,i}^{n+1} = 2\widetilde{\widetilde{\Delta\alpha}}_{z,i}^{n+1/2} \tag{54}$$

and

$$\alpha_{z,i}^{n+1} = \alpha_{z,i}^{n} + \Delta\alpha_{z,i}^{n+1}. \tag{55}$$

From this, the new material quantities are calculated.

$$V_{z,i}^{n+1} = \alpha_{z,i}^{n+1} V_z^{n+1} \tag{56}$$

$$\Delta V_{z,i}^{n+1} = V_{z,i}^{n+1} - V_{z,i}^{n} \tag{57}$$

$$\rho_{z,i}^{n+1} = m_{z,i}^{n+1} / V_{z,i}^{n+1}. \tag{58}$$

The compatible internal energy update is

$$m_{z,i}^{n+1} \frac{e_{z,i}^{n+1} - {}^{n}_{z,i}}{\Delta t} = - \left(\underbrace{\alpha_z^{n+1} \sum_{p \in P(z)} \mathbf{f}_{zp}^{p^{n+1/2}} \cdot \mathbf{u}_p^{n+1/2}}_{pdV \text{compatible with momentum}} + p_z^{n+1/2} \frac{\Delta\alpha_{z,i}^{n+1}}{\Delta t} (V_z^{n+1} + V_z^n)/2 \right). \tag{59}$$

Linearized Riemann Problem (LRP) Pressure Relaxation Model

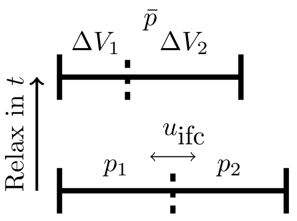


Figure 4: Riemann problem based model

In Figure 4, an illustration of general pressure relaxation model using Riemann problem solver is shown. In Linearized Riemann Problem pressure relaxation closure model, the linearized Riemann solver is utilized to initialize the volume fraction changes. This approach enforces that each material component satisfies its own *pdV* equation, therefore, the energy discrepancy must be redistributed by assuming that the corresponding pressure change in each component is equal. The details of this work is shown in [8].

Predictor Stage

For simplicity, we only consider two material configurations. The changes of material volumes are

$$\delta V_1 = (u_{intfc} - u_1)\delta t, \quad \delta V_2 = (u_2 - u_{intfc})\delta t \tag{60}$$

where u_{intfc} is the velocity of the interface between materials approximated by the lineaized Riemann solver. Assuming that the velocity in the multi-material cell varies linearly, we obtain the following expression for the interface velocity

$$u_{intfc} = \alpha_2 u_1 + \alpha_1 u_2 + \frac{p_1 - p_2}{\rho_1 c_1 + \rho_2 c_2}. \tag{61}$$

With above definition of u_{intfc}, we have

$$\delta V_1^{n+1/2} = \alpha_1 \delta V^{n+1/2} + \frac{\Delta p_{1,2}}{2\bar{\kappa}} \frac{\delta t}{2}, \tag{62}$$

and

$$\delta V_2^{n+1/2} = \alpha_2 \delta V^{n+1/2} + \frac{\Delta p_{1,2}}{2\bar{\kappa}} \frac{\delta t}{2}. \tag{63}$$

where $\Delta p_{1,2} = p_1 - p_2$, $\bar{\kappa} = 0.5(\kappa_1 + \kappa_2)$, $\kappa_i = \rho_i c_i$. The volumes of materials are

$$V_i^{n+1/2} = V_i^n + \delta V_i^{n+1/2} \quad \text{for } i = 1,2. \tag{64}$$

The pressures of materials are

$$p_i^{n+1/2} = p_i^n - \left((c_i^n)^2 \rho_i^n\right) \left(\delta V_i^{n+1/2} / V_i^n\right), \tag{65}$$

and the common pressure is

$$p^{n+1/2} = \sum_i \alpha_i^{n+1/2} p_i^{n+1/2}. \tag{66}$$

Corrector Stage

The corrector stage volumes are calculated similar to Equation (64) but with predictor values. The corrector specific internal energy update is divided into two parts: provisional and correction parts. The provisional specific internal energy update does not satisfy the total conservation of energy. The provisional specific internal energy is updated as follows

$$e_1^{n+1,*} = e_1^n - \alpha_i^{n+1/2}\left(p_1^{n+1/2} + q^n\right) dV_1^{n+1}/m_1 - p_1^{n+1/2} \frac{\Delta p_1^{n+1/2}}{2\bar{\kappa}_1^{n+1/2}} \frac{\delta t}{m_1} \tag{67}$$

$$e_2^{n+1,*} = e_2^n - \alpha_i^{n+1/2}\left(p_2^{n+1/2} + q^n\right) dV_2^{n+1}/m_2 + p_1^{n+1/2} \frac{\Delta p_2^{n+1/2}}{2\bar{\kappa}_2^{n+1/2}} \frac{\delta t}{m_2}. \tag{68}$$

To enforce the conservation of energy, we determine the internal energy discrepancy

$$\delta \mathscr{E}^{n+1/2} = \sum_i m_i d\tilde{e}_i^{n+1/2} = \frac{\left(p_1^{n+1/2} - p_2^{n+1/2}\right)^2}{\kappa_1^{n+1/2} + \kappa_2^{n+1/2}} \delta t \tag{69}$$

where $d\tilde{e}_i^{n+1/2}$ is the additional specific internal energy increment in material i to ensure the conservation of energy. To compute the value of the correction in the specific internal energy, $d\tilde{e}_i$, we make use the following relations

$$d\tilde{p} = (\partial p_1/\partial e_1)_{\rho_1} d\tilde{e}_1^{n+1}, \tag{70}$$

$$d\tilde{p} = (\partial p_2/\partial e_2)_{\rho_2} d\tilde{e}_2^{n+1}. \tag{71}$$

Equations (69), (70) and (71) can be solved and rearranged as follows

$$d\tilde{p}^{n+1} = \frac{\delta \mathscr{E}^{n+1/2}}{\left[m_1/(\partial p_1/\partial e_1)_{\rho_1}^{n+1}\right] + \left[m_2/(\partial p_2/\partial e_2)_{\rho_2}^{n+1}\right]} \tag{72}$$

From Equations (72), (70) and (71), the correction values to the specific internal energy, $d\tilde{e}_i^{n+1}$, can be evaluated. Finally, the consistent specific internal energy values are

$$e_i^{n+1} = e_i^{n+1,*} + d\tilde{e}_i^{n+1}, \tag{73}$$

and the pressure in each material is

$$p_i^{n+1} = \mathscr{P}_i(\rho_i^{n+1}, e_i^{n+1}). \tag{74}$$

Interface-Aware Sub-Scale Dyanmics (IASSD)

Another instance of a Riemann-problem–based pressure relaxation model is the Interface-Aware Sub-Scale Dynamics (IASSD) method proposed in [1]. This model is based on the idea of using interface location (obtained via an interface reconstruction algorithm) to solve a Riemann problem to compute velocities of constituent materials in a multi-material cell. Like the LRP model in the previous section, IASSD is described by Figure 4.

In a similar vein as flux-corrected transport (FCT), the authors of [1] then use a linear-inequality–constrained quadratic optimization problem to compute the volume changes of the constituents. The general idea of the IASSD optimization is to determine a set of flux limiters that are as close to unity as possible while still respecting a set of physical constraints.

Constrained Optimization Framework

The objective function for computing the IASSD limiters $\Psi_{i,k}$ is

$$\min_{\Psi_{i,k}} \left\{ \sum_i \left[\sum_{k \in M(i)} \left(1 - \Psi_{i,k}\right)^2 \right] \right\} \tag{75}$$

There are several constraints placed on this objective in order to maintain certain desirable (physical) properties. Each material i is subject to constraints:

1. To keep material volumes positive but less than the bulk volume:

$$V_i^{n+1,\text{bulk}} + \sum_{k \in M(i)} \Psi_{i,k} \delta V_{i,k}^{\max} \geq \kappa_{\text{bot}} V_i^{n+1,\text{bulk}} \tag{76}$$

where $0 < \kappa_{\text{bot}} \leq 1$ is a constant that keeps material volumes bounded away from zero.

2. To maintain positivity of internal energy:

$$\left(m_i E_i^n - p_i^n \Delta V_i^{n+1,\text{bulk}} + Q_i^n\right) \geq \sum_{k \in M(i)} \Psi_{i,k} p_{i,k}^* \Delta V_{i,k}^{\max} \tag{77}$$

where $\Delta V_{i,k}^{\max}$ is simply the largest computed $\Delta V_{i,k}$ over i and k.

3. To control the relaxation of pressures:

$$\frac{\kappa_i V_i^n}{\rho_i^n \left(c_i^n\right)^2} \left(\tilde{p}_i^{n+1,\text{bulk}} - \bar{p}\right) \geq \sum_{k \in M(i)} \Psi_{i,k} \delta V_{i,k}^{\max} \geq 0, \quad \text{if} \quad \tilde{p}_i^{n+1,\text{bulk}} \geq \bar{p} \tag{78}$$

$$\frac{\kappa_i V_i^n}{\rho_i^n \left(c_i^n\right)^2} \left(\tilde{p}_i^{n+1,\text{bulk}} - \bar{p}\right) \leq \sum_{k \in M(i)} \Psi_{i,k} \delta V_{i,k}^{\max} \leq 0, \quad \text{if} \quad \tilde{p}_i^{n+1,\text{bulk}} \leq \bar{p} \tag{79}$$

where κ_i is a constant chosen to control the rate of pressure equilibration.

We note that the authors in [1] simply choose constant values for κ_{bot} and κ_i. Additionally, they use a single κ_i for all i. The choice of these constants has direct impact on the IASSD algorithm; thus, we propose the theory behind the choice of these constants as a topic of highly relevant future research. One may hypothesize that using per–time-step and per-material values for these two constants may allow for improved results.

In the case of only two materials—such as the examples we consider in this paper—the IASSD optimization problem has an explicit solution:

$$\Psi_{i,k} = \min\left\{\Psi_{i,k}^V, \Psi_{i,k}^E, \Psi_{i,k}^p, 1\right\}, \tag{80}$$

$$\Psi_{i,k}^V = \begin{cases} \frac{(1-\kappa_{\text{bot}})V_2^{n+1,\text{bulk}}}{\delta V^{\max}} & \text{if } \delta V^{\max} > 0 \\ \frac{(1-\kappa_{\text{bot}})V_1^{n+1,\text{bulk}}}{|\delta V^{\max}|} & \text{if } \delta V^{\max} < 0 \end{cases} \tag{81}$$

$$\Psi_{i,k}^E = \begin{cases} \frac{E_1^{\text{bulk}}}{p^* \delta V^{\max}} & \text{if } p^* \delta V^{\max} > 0 \\ \frac{E_2^{\text{bulk}}}{|p^* \delta V^{\max}|} & \text{if } p^* \delta V^{\max} < 0 \end{cases} \tag{82}$$

$$\Psi_{i,k}^P = \begin{cases} 0 & \begin{cases} \text{if } p_1^{n+1,\text{bulk}} > p_2^{n+1,\text{bulk}} & \text{and } \delta V^{\max} < 0 \\ \text{or} & \\ \text{if } p_1^{n+1,\text{bulk}} < p_2^{n+1,\text{bulk}} & \text{and } \delta V^{\max} > 0 \end{cases} \\ 1 & \text{if } \delta V^{\max} = 0 \\ \min\left\{ \frac{\kappa_1 V_1^n}{\rho_1^n \left(c_1^n\right)^2} \frac{\left|\bar{p} - \tilde{p}_1^{n+1,\text{bulk}}\right|}{|\delta V^{\max}|}, \frac{\kappa_2 V_2^n}{\rho_2^n \left(c_2^n\right)^2} \frac{\left|\tilde{p}_2^{n+1,\text{bulk}} - \bar{p}\right|}{|\delta V^{\max}|} \right\} & \text{otherwise} \end{cases} \tag{83}$$

Predictor Stage

Like the other models considered in this paper, IASSD is based on a predictor-corrector approach. During the predictor and corrector phases, the host hydrocode is used to update the node locations and cell volumes of the domain. Beyond that, the predictor and corrector phases of the IASSD algorithm are explicitly below. In the algorithm, $S_{i,k}$ refers to the interfacial area between materials i and k, which is identically 1 in 1D. Additionally, we note that in both the predictor and corrector steps, the old volume fractions α_i^n are used (rather than, for example, time $(n+1,\text{pr})$ volume fractions) in order to enforce equal volumetric compressibility of the constituent materials. Finally, we note that $\vec{n}_{i,k}^n$ refers to the normal vector between the two materials i and k. In 1D, with two materials indexed 0 and 1, we note that $\vec{n}_{i,k}^n = k - i$.

$$\tilde{p}_z^n = \sum_{i\in M(z)} \alpha_{z,i}^n p_{z,i}^n \tag{84}$$

$$\Delta V_z^{n+1,\text{pr}} = V_z^{n+1,\text{pr}} - V_z^n \tag{85}$$

$$\delta V_{i,k}^{\max,n} = \frac{p_i^n - p_k^n}{\rho_i^n c_i^n + \rho_k^n c_k^n} S_{i,k}^n \Delta t \tag{86}$$

$$p_{i,k}^{*,n} = \mathscr{P}^*(\rho^n, c^n, p^n, \vec{u}^n) = \frac{\left(\rho_k^n c_k^n\right) p_i^n + \left(\rho_i^n c_i^n\right) p_k^n - \left(\rho_k^n c_k^n\right)\left(\rho_i^n c_i^n\right)\left(\vec{u}_k^n - \vec{u}_i^n\right)\cdot \vec{n}_{i,k}^n}{\rho_k^n c_k^n + \rho_i^n c_i^n} \tag{87}$$

$$\Psi_{i,k} = \text{CON\_OPT}\left(\Delta V_z^{n+1,\text{pr}}, \delta V_{i,k}^{\max,n}, p_{i,k}^{*,n}, \kappa_{\text{bot}}, \kappa_i\right) \tag{88}$$

$$V_{z,i}^{n+1,\text{pr}} = \alpha_{z,i}^n V_z^{n+1,\text{pr}} + \sum_{k\in M_z(i)} \Psi_{i,k}\delta V_{i,k}^{\max,n} \tag{89}$$

$$\rho_{z,i}^{n+1,\text{pr}} = m_{z,i}/V_{z,i}^{n+1,\text{pr}} \tag{90}$$

$$E_{z,i}^{n+1,\text{pr}} = E_{z,i}^n - \frac{1}{m_{z,i}}\Big(\alpha_{z,i}^n p_{z,i}^n \text{div}\vec{u}^{n+1,\text{pr}} V_{z,i}^n \Delta t \tag{91}$$

$$+ \sum_{k\in M_z(i)} \Psi_{i,k} p_{i,k}^{*,n} \delta V_{i,k}^{\max,n} \Delta t + Q_{z,i}^n \Delta t\Big) \tag{92}$$

$$p_{z,i}^{n+1,\text{pr}} = \text{EOS}\left(\rho_{z,i}^{n+1,\text{pr}}, E_{z,i}^{n+1,\text{pr}}\right) \tag{93}$$

Figure 5: The predictor stage of IASSD

Corrector Stage

$$\tilde{p}_z^{n+1} = \sum_{i\in M(z)} \alpha_{z,i}^n p_{z,i}^{n+1,\mathrm{pr}} \tag{94}$$

$$\Delta V_z^{n+1,\mathrm{pr}} = V_z^{n+1} - V_z^n \tag{95}$$

$$\delta V_{i,k}^{\max,n+1} = \frac{p_i^{n+1/2,\mathrm{pr}} - p_k^{n+1/2,\mathrm{pr}}}{\rho_i^{n+1/2,\mathrm{pr}} c_i^{n+1/2,\mathrm{pr}} + \rho_k^{n+1,\mathrm{pr}} c_k^{n+1/2,\mathrm{pr}}} S_{i,k}^{n+1} \Delta t \tag{96}$$

$$p_{i,k}^{*,n+1} = \mathscr{P}^*\left(\rho^{n+1/2,\mathrm{pr}}, c^{n+1/2,\mathrm{pr}}, p^{n+1/2,\mathrm{pr}}, \vec{u}^{n+1/2,\mathrm{pr}}\right) \tag{97}$$

$$= \frac{\left(\rho_k^{n+1/2,\mathrm{pr}} c_k^{n+1/2,\mathrm{pr}}\right) p_i^{n+1/2,\mathrm{pr}} + \left(\rho_i^{n+1/2,\mathrm{pr}} c_i^{n+1/2,\mathrm{pr}}\right) p_k^{n+1/2,\mathrm{pr}}}{\rho_k^{n+1/2,\mathrm{pr}} c_k^{n+1/2,\mathrm{pr}} + \rho_i^{n+1/2,\mathrm{pr}} c_i^{n+1/2,\mathrm{pr}}} \tag{98}$$

$$- \frac{\left(\rho_k^{n+1/2,\mathrm{pr}} c_k^{n+1/2,\mathrm{pr}}\right)\left(\rho_i^{n+1/2,\mathrm{pr}} c_i^{n+1/2,\mathrm{pr}}\right)\left(\vec{u}_k^{n+1/2,\mathrm{pr}} - \vec{u}_i^{n+1/2,\mathrm{pr}}\right)\cdot \vec{n}_{i,k}^{n+1/2,\mathrm{pr}}}{\rho_k^{n+1/2,\mathrm{pr}} c_k^{n+1/2,\mathrm{pr}} + \rho_i^{n+1/2,\mathrm{pr}} c_i^{n+1/2,\mathrm{pr}}} \tag{99}$$

$$\Psi_{i,k} = \mathrm{CON\_OPT}\left(\Delta V_z^{n+1}, \delta V_{i,k}^{\max,n+1}, p_{i,k}^{*,n+1}, \kappa_{\mathrm{bot}}, \kappa_i\right) \tag{100}$$

$$V_{z,i}^{n+1} = \alpha_{z,i}^n V_z^{n+1} + \sum_{k\in M_z(i)} \Psi_{i,k}\,\delta V_{i,k}^{\max,n+1} \tag{101}$$

$$\rho_{z,i}^{n+1} = m_{z,i}/V_{z,i}^{n+1} \tag{102}$$

$$E_{z,i}^{n+1} = E_{z,i}^n - \frac{1}{m_{z,i}}\Big(\alpha_{z,i}^n p_{z,i}^{n+1} \mathrm{div}\vec{u}^{n+1} V_{z,i}^{n+1}\Delta t \tag{103}$$

$$+ \sum_{k\in M_z(i)} \Psi_{i,k} p_{i,k}^{*,n+1} \delta V_{i,k}^{\max,n+1}\Delta t + Q_{z,i}^{n+1/2}\Delta t\Big) \tag{104}$$

$$p_{z,i}^{n+1} = \mathrm{EOS}\left(\rho_{z,i}^{n+1}, E_{z,i}^{n+1}\right) \tag{105}$$

$$\alpha_{z,i}^{n+1} = V_{z,i}^{n+1}/V_z^{n+1} \tag{106}$$

Figure 6: The corrector stage of IASSD

CON_OPT refers to the solution of the optimization problem Equations (75) to (78); for example, in 1D with two materials, it refers to applying the solution given in Equations (80) to (83). It is important to note that the variables at time $(n+1/2,\text{pr})$ are computed by using the relevant equations from the predictor stage with a time step of $\Delta t/2$, as opposed to some procedure like averaging the time n and time $(n+1,\text{pr})$ variables.

Technical Approach

Various technical approaches used in this report are discussed.

Multi-Material Configurations

In this section, a detailed explanation of how different multi-material closure models are compared against a range of realizable pure cell configuration results.

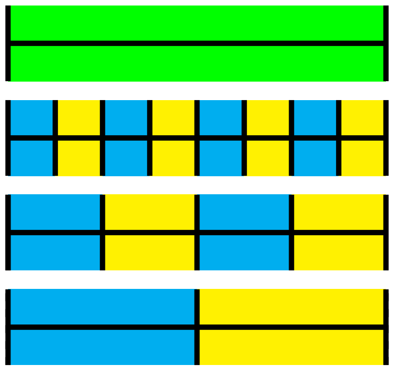


Figure 7: Sample multi-material configurations for $\alpha_i = 0.5$

Most multi-material closure models do not make explicit assumption about the configurations of the materials within the multi-material cell. This means that the sample multi-material configurations shown in Figure 7 for volume fraction of 0.5 are nearly indistinguishable for most multi-material closure models. A multi-material configuration where one material occupying left half and another material occupying the rest of the cell (bottom configuration) is exactly the same as any alternating configuration, or a composite configuration (top configuration), as long as the volume fractions of each material in the multi-material cell satisfies $\alpha_i = 0.5$.

Pure Cell Realizations

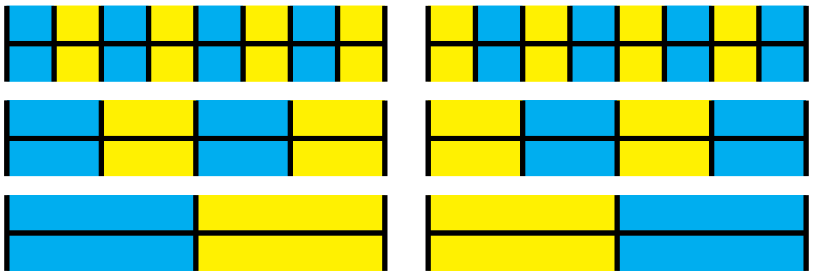


Figure 8: Realizable pure material configurations for $\alpha_i = 0.5$

In Figure 8, sample realizable pure material configurations for the volume fraction of 0.5 are shown. These pure material realizations of the multi-material configurations suggest some standards to compare the multi-material closure model results. And these pure material realizations make up of the range of plausibility for the multi-material closure model results, as will be shown in sanity check, modified shock tube and water air shock tube test problems.

Artificial Viscosity

The use of artificial viscosity for compressible flow simulations dates back to 1950. The form of artificial viscosity discovered by von Neumann and Richtmyer [10] augmented the pressure where compression was detected. The expression used to describe the nonlinear mechanism to suppress numerical oscillations near sharp discontinuities and compression shock waves is the following

$$q = c_2 \rho (\Delta \mathbf{u})^2. \tag{107}$$

where c_2 is a constant of unity.

The artificial viscosity used throughout this project is a modified version of the von Neumann and Richtmyer's original artificial viscosity, [7]. It is given by the addition of a linear term

$$q = \begin{cases} 0 & \text{if} \quad \Delta \mathbf{u} \geq 0, \\ -c_1 \rho c |\Delta \mathbf{u}| + c_2 \rho (\Delta \mathbf{u})^2 & \text{if} \quad \text{otherwise.} \end{cases} \tag{108}$$

where $c_1 = 1$ and $c_2 = 0.1$ are constants. c is the speed of sound in cell.

Boundary Conditions

The boundary condition used in this project is a transmissive boundary condition from which the immediate neighbors' boundary values are transmitted. Since SGH hydrocode locate the velocities at the vertex and all other variables in the centroid, one only needs to apply boundary conditions on the outer-most vertex points.

Results

The Sod Shock Tube Problem: Sanity Check

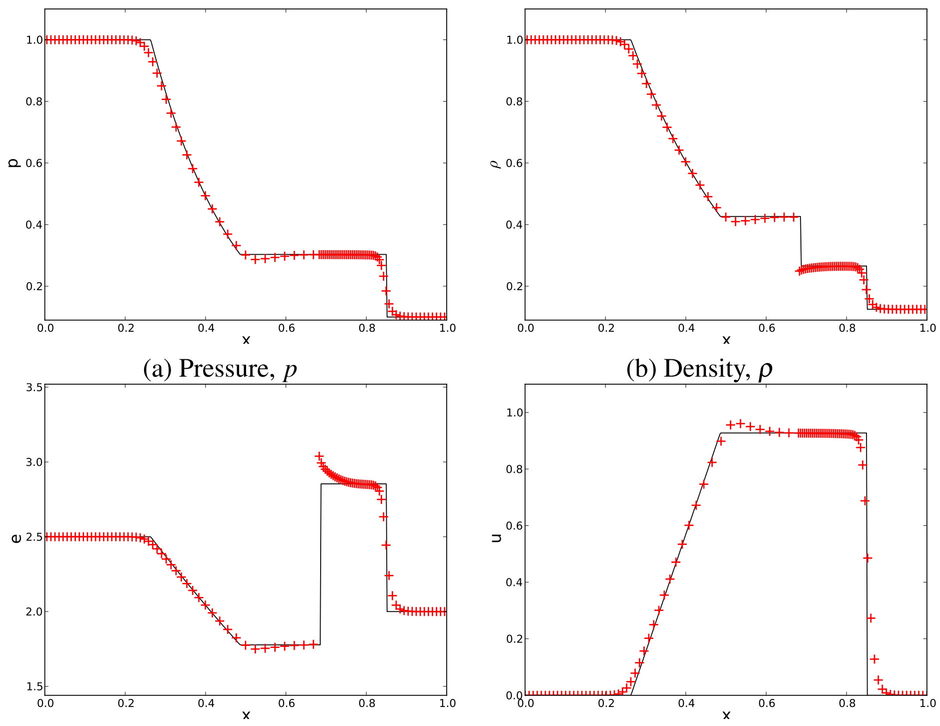
In the sanity check problem, multi-material closure model results are compared to the pure Lagrangian hydrodynamics results. The sanity check is a variation of simple one dimensional Sod shock tube problem with one pseudo multi-material cell at the center of the domain. The initial conditions for this problem are

$$(\gamma, \rho, e, p, u) = \begin{cases} (1.4, 1, 2.5, 1, 0) & \text{if} \quad 0 \le x < 0.5 \\ (1.4, 0.125, 2, 0.1, 0) & \text{if} \quad 0.5 \le x < 1.0. \end{cases} \tag{109}$$

The parameters for this problem are

- $x \in [0,1]$
- $N_{\text{cells}} = 99$
- $t_{\text{final}} = 0.2$
- $\text{CFL} = 0.1$
- One multi-material cell located at $x = 0.5$
- Multi-material sub-cells initialized with the volume fraction of $\alpha_i = 0.5$

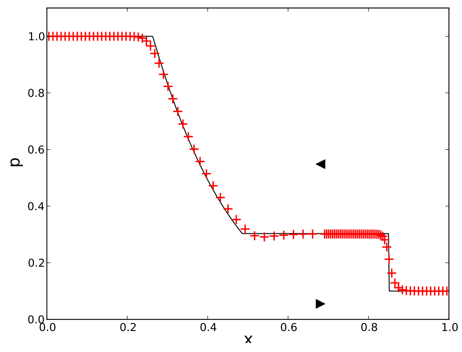
All results shown are at the final time. The black solid lines indicate the exact solutions. The symbols ◀ and ▶ indicate left and right component values in multi-material cell.



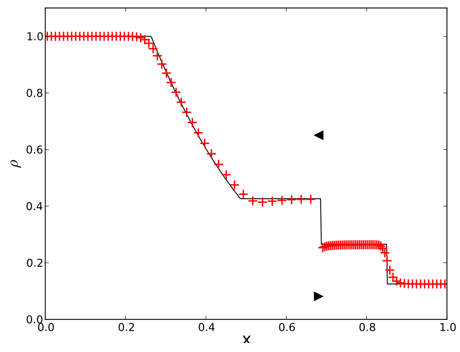
(a) Pressure, p (b) Density, ρ

(c) Specific internal energy, e (d) Velocity, u

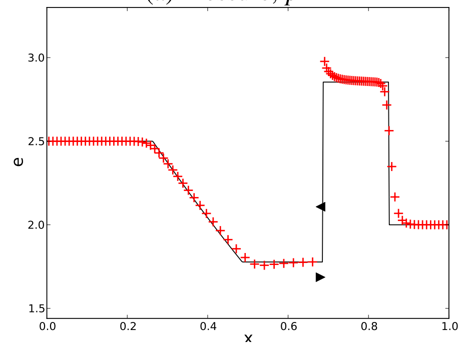
Figure 9: Pure Lagrangian solution



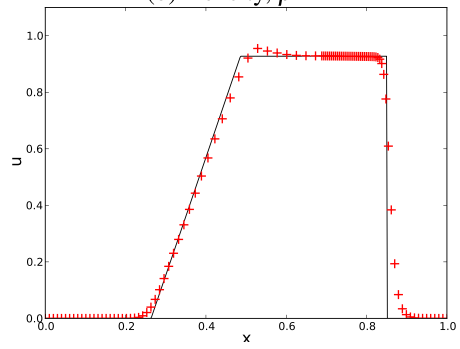
(a) Pressure, p



(b) Density, ρ

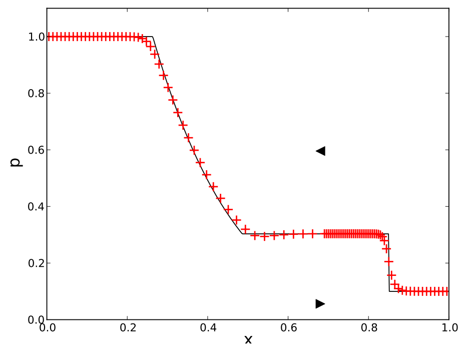


(c) Specific internal energy, e

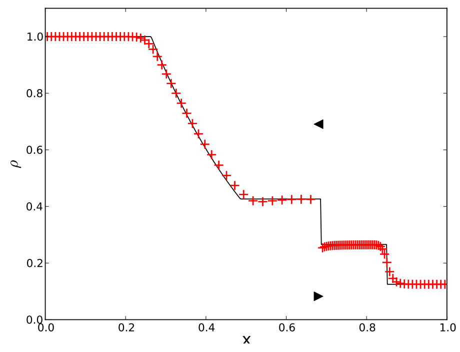


(d) Velocity, u

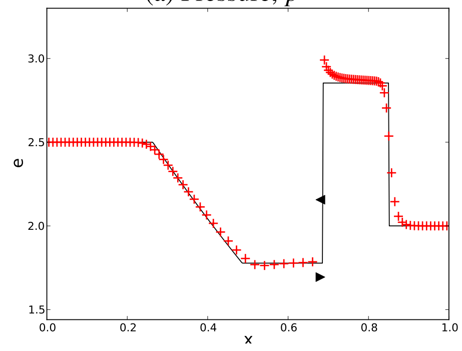
Figure 10: Equal compressibility model



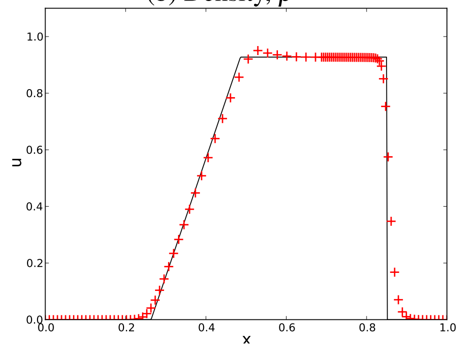
(a) Pressure, p



(b) Density, ρ

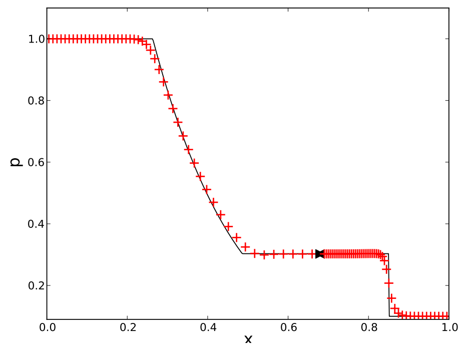


(c) Specific internal energy, e

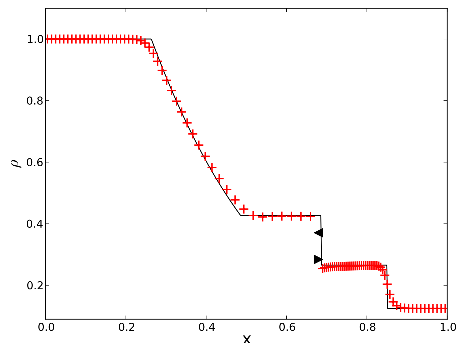


(d) Velocity, u

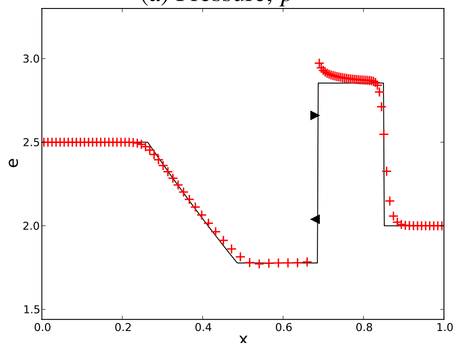
Figure 11: Equal velocity increment model



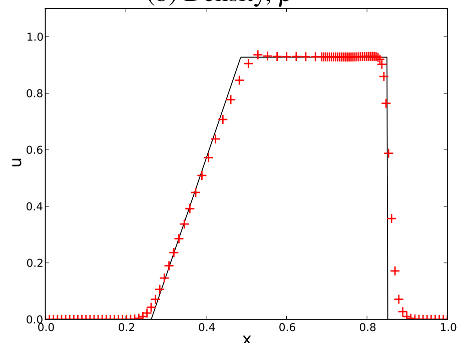
(a) Pressure, p



(b) Density, ρ

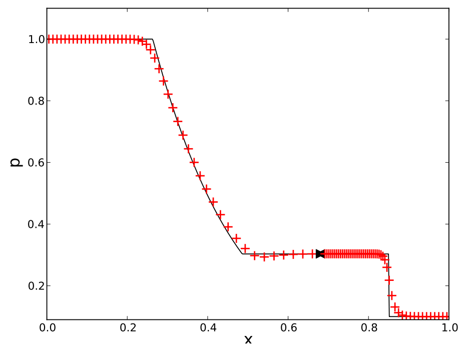


(c) Specific internal energy, e

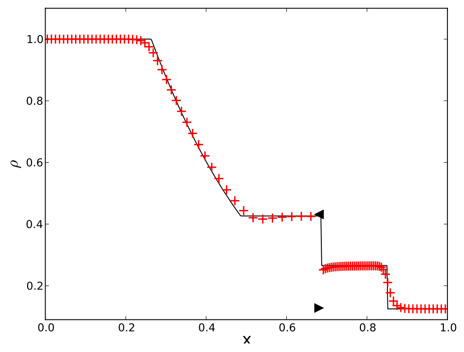


(d) Velocity, u

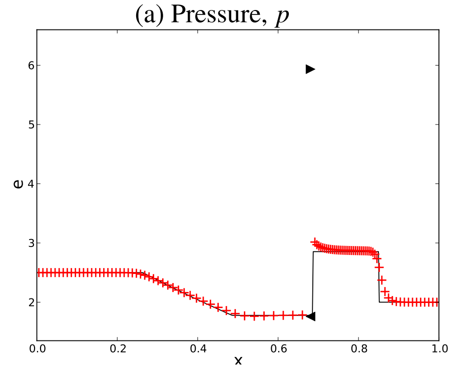
Figure 12: Tipton's pressure relaxation model



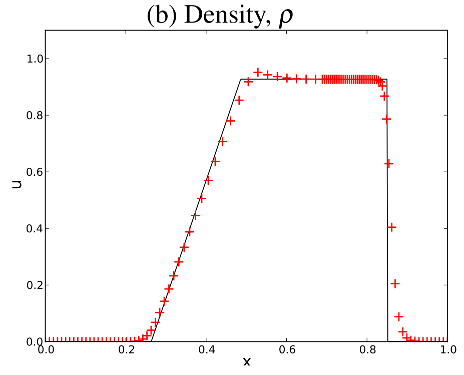
(a) Pressure, p



(b) Density, ρ

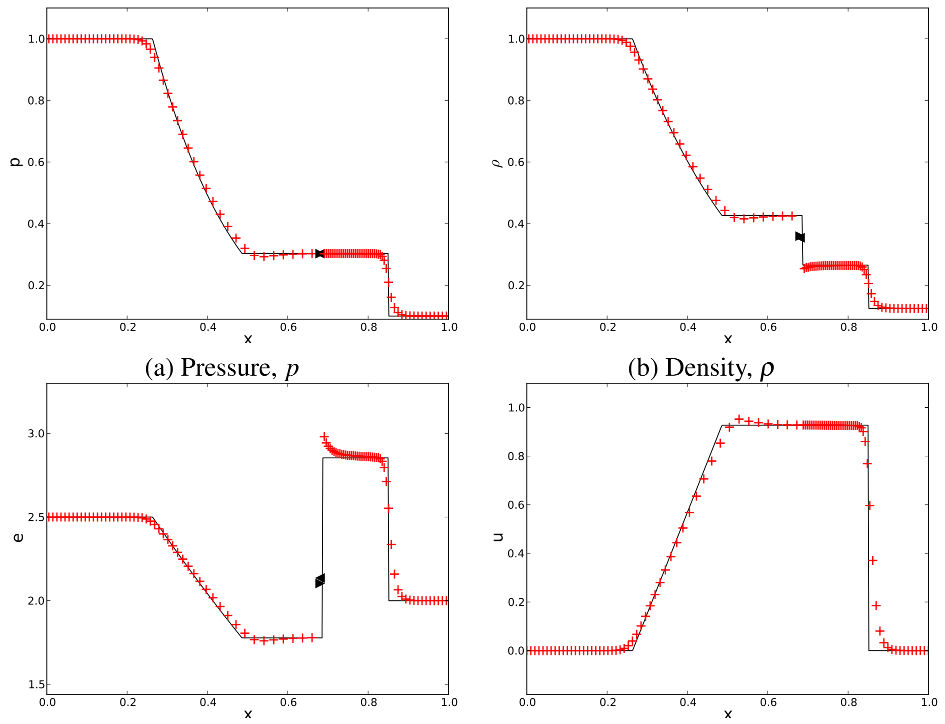


(c) Specific internal energy, e



(d) Velocity, u

Figure 13: Linearized Riemann Problem pressure relaxation model



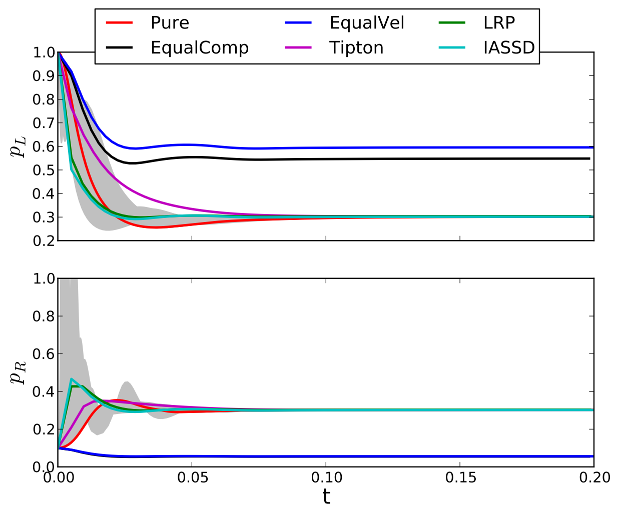
(a) Pressure, p (b) Density, ρ

(c) Specific internal energy, e (d) Velocity, u

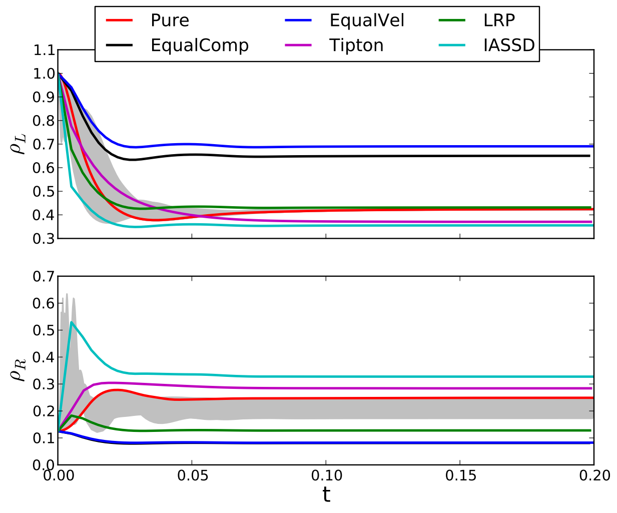
Figure 14: IASSD pressure relaxation model

In the Sod shock tube problem, the main purpose of this problem was to verify that all multi-material closure models behave in the correct manner. For instance, the pure Lagrangian results, which does not have any multi-material cell, follow the exact Sod shock tube result very closely in Figure 9. The equal compressibility and equal velocity increment model results exhibit the pressure non-equilibration for multi-material components in the multi-material cell in Figures 10 and 11. The density and the specific internal energy values also show varying results from the equilibrium values.

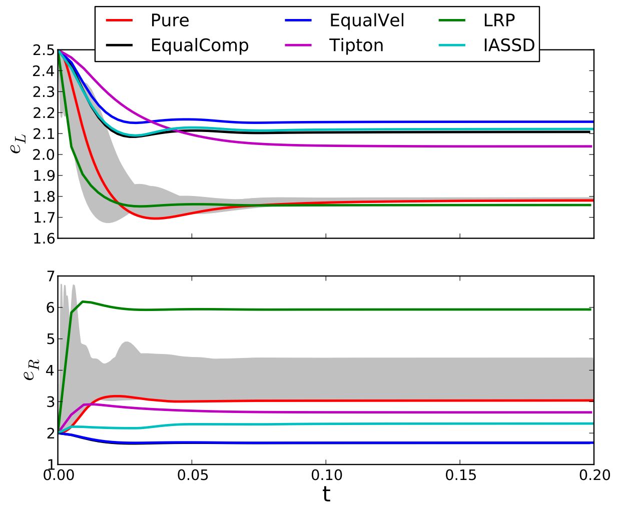
The pressure relaxation model results such as Tipton's pressure relaxation model, Linearized Riemann Problem model and IASSD model, do obtain the correct equilibrium pressure results. However, the density and the specific internal energy values do not reach the same equilibrated values as shown in Figures 12, 13 and 14. The pressure relaxation closure models are designed to obtain the same pressures of the components in the multi-material cells, not the density or the specific internal energy values. We can also notice that the density and the specific internal energy values are closely related to the pressure in the equation of state of materials. Any change in the specific internal energy must be accompanied by corresponding change in density to obtain the same pressure value. This is obvious from the equation of state, $p = \mathscr{P}(\rho, e)$, since pressure is a function of both density and specific internal energy. In Figure 13, one can notice that the large difference in the specific internal energy values give rise to large difference in the density values of two components, or vice versa.



(a) Pressure, p



(b) Density, ρ



(c) Specific internal energy, e

Figure 15: Time history of multi-material cell components for various closure models

In Figure 15, the multi-material closure model results are compared against a range of possible realizations of multiple material configurations in the multi-material cell. The time history of pressure, density and specific internal energies of left and right components are plotted along with the gray bands. The gray bands represent possible realizations of multiple material configurations in the multi-material cell.

The pressure relaxation closure models obtain the desired equilibrium pressures. However, the one-step closure models such as the equal compressibility and equal velocity increment model do not. Even within the list of the pressure relaxation closure models, one can find various relaxation times. The Riemann problem based relaxation models converge faster to the equilibrium values in general. The density and the specific internal energy values, however, are not well predicted by any of the closure models.

This test problem addresses the characteristics of individual closure models at the most basic level. Since the multi-material cell only consists of two pseudo materials, in other words the same material, it is a good test to check whether the closure models behave as expected or not.

The Modified Shock Tube Problem

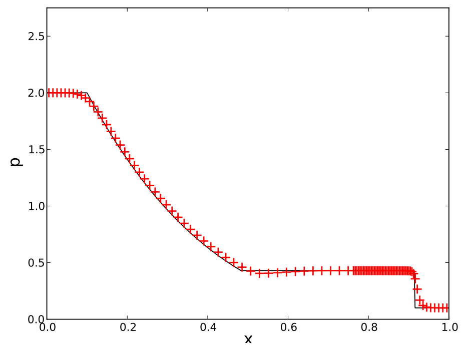
The modified shock tube problem consists of two distinct gamma gases. The multi-material cell is initialized with a mixture of the two distinct gases. The initial conditions for this problem are

$$(\gamma,\rho,e,p,u) = \begin{cases} (2.0,1,2,2,0) & \text{if} \quad 0 \le x < 0.5 \\ (1.4,0.125,2,0.1,0) & \text{if} \quad 0.5 \le x < 1.0. \end{cases} \tag{110}$$

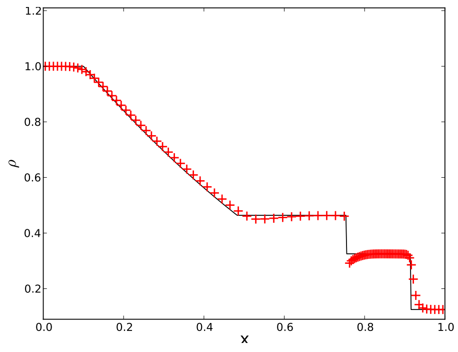
The parameters for this problem are

- $x \in [0,1]$
- $N_{\text{cells}} = 99$
- $t_{\text{final}} = 0.2$
- $\text{CFL} = 0.1$
- One multi-material cell located at $x = 0.5$
- Multi-material sub-cells initialized with the volume fraction of $\phi_i = 0.5$

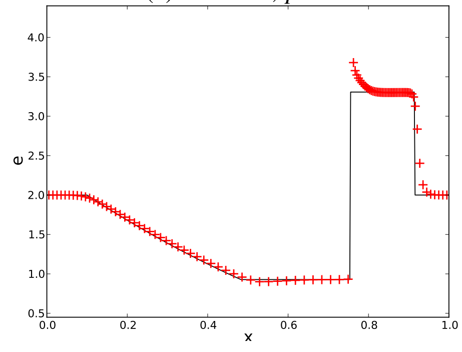
All results shown are at the final time. The black solid lines indicate the exact solutions. The symbols ◀ and ▶ indicate left and right component values in multi-material cell.



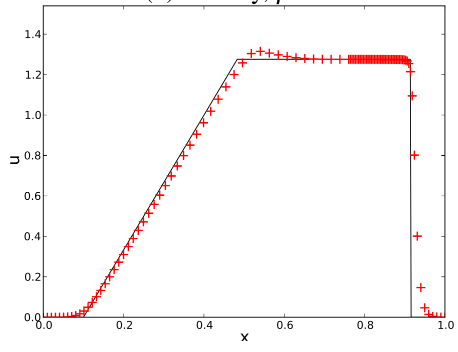
(a) Pressure, p



(b) Density, ρ

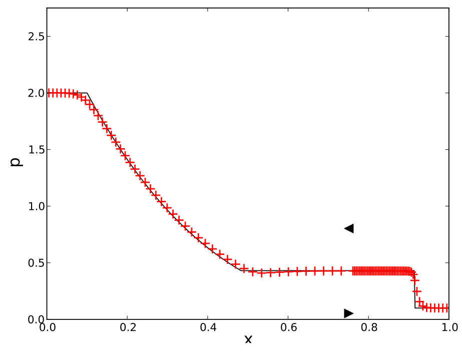


(c) Specific internal energy, e

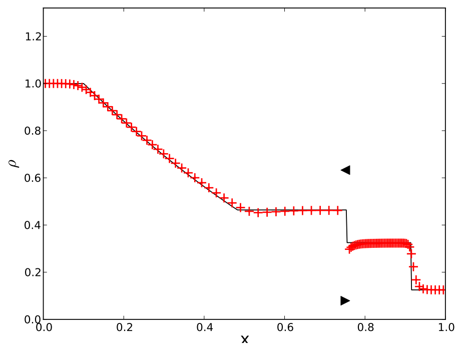


(d) Velocity, u

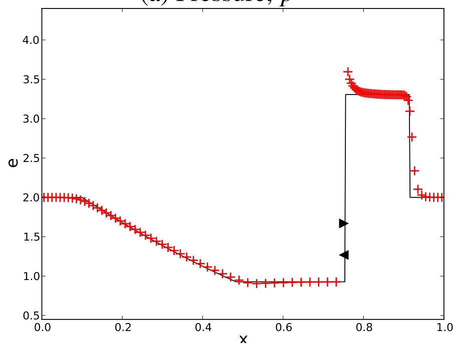
Figure 16: Pure Lagrangian solution



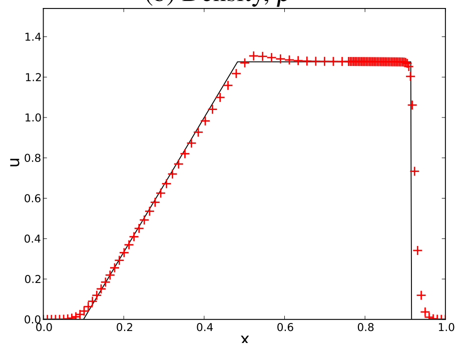
(a) Pressure, p



(b) Density, ρ

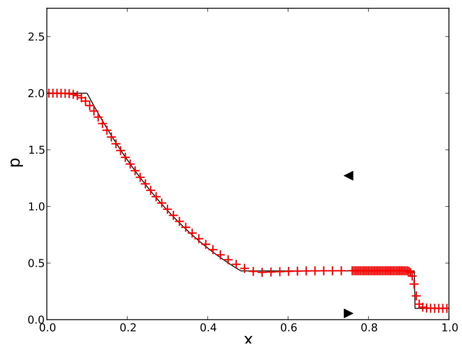


(c) Specific internal energy, e

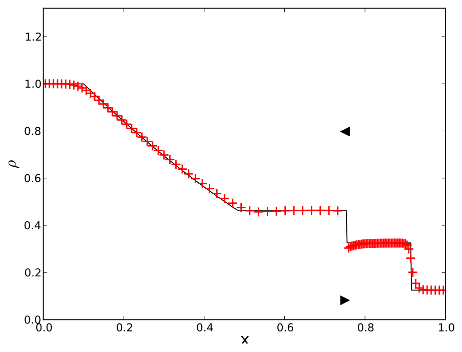


(d) Velocity, u

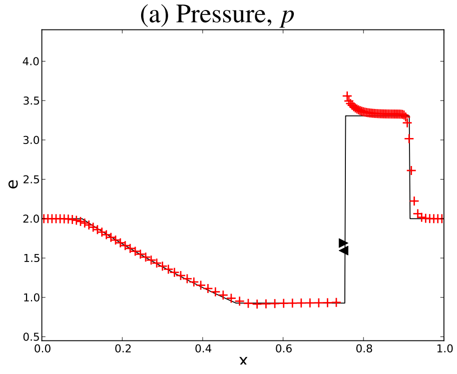
Figure 17: Equal compressibility model



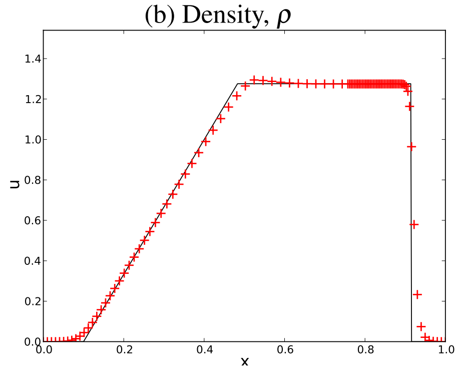
(a) Pressure, p



(b) Density, ρ

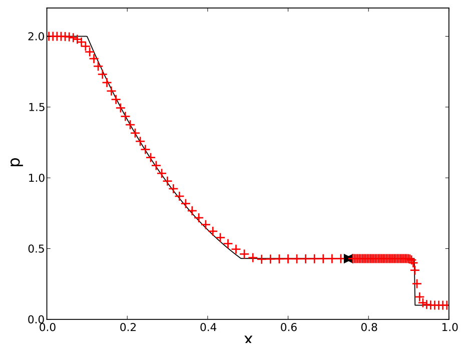


(c) Specific internal energy, e

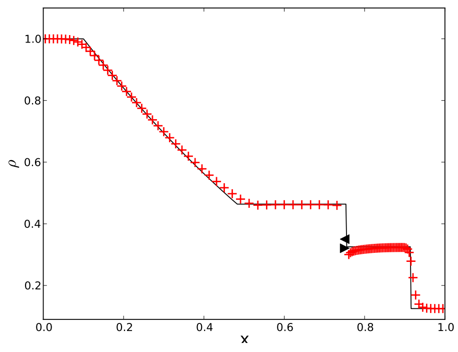


(d) Velocity, u

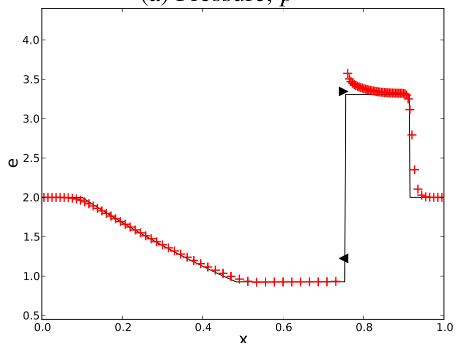
Figure 18: Equal velocity increment model



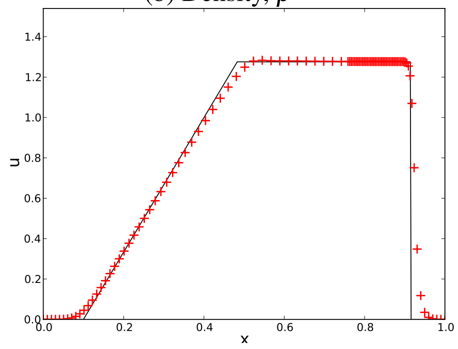
(a) Pressure, p



(b) Density, ρ

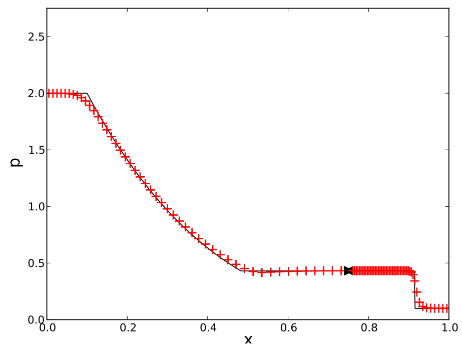


(c) Specific internal energy, e

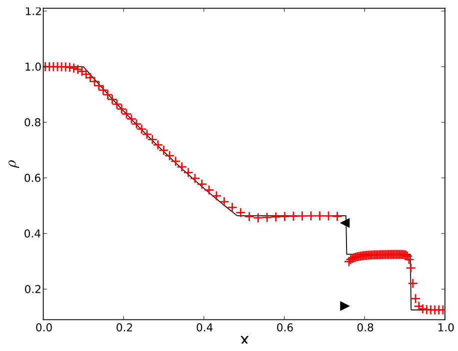


(d) Velocity, u

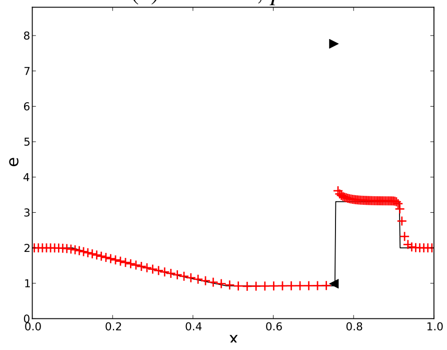
Figure 19: Tipton's pressure relaxation model



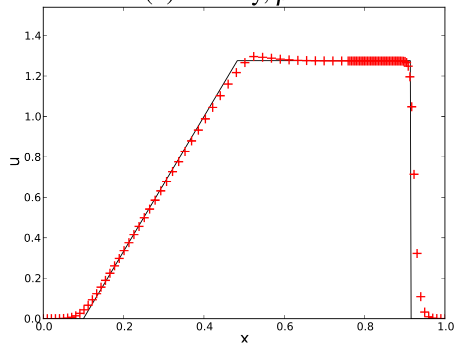
(a) Pressure, p



(b) Density, ρ

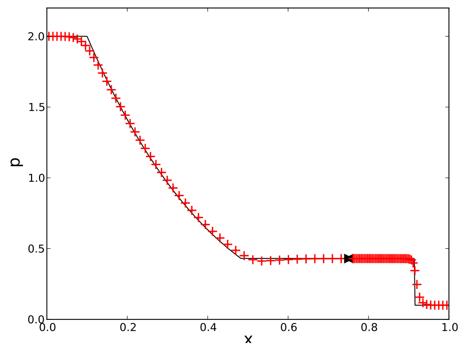


(c) Specific internal energy, e

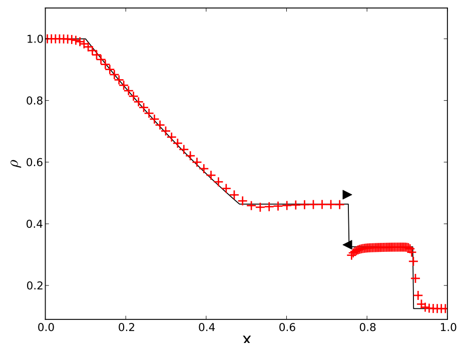


(d) Velocity, u

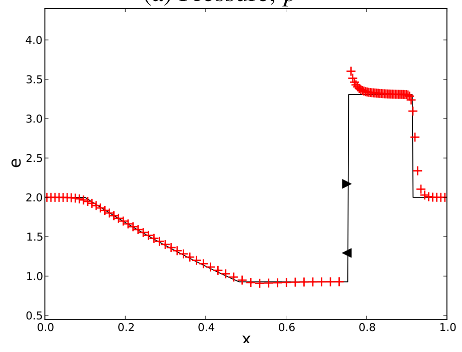
Figure 20: Linearized Riemann Problem pressure relaxation model



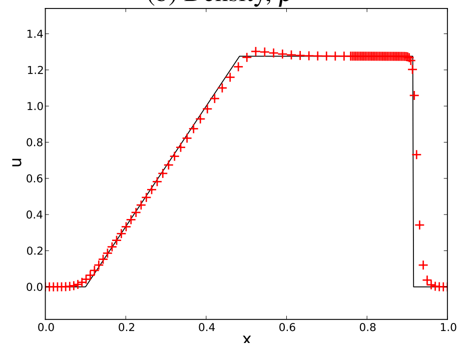
(a) Pressure, p



(b) Density, ρ



(c) Specific internal energy, e

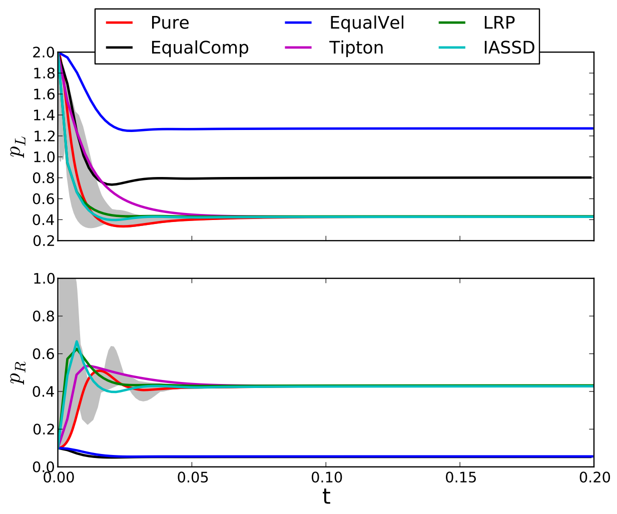


(d) Velocity, u

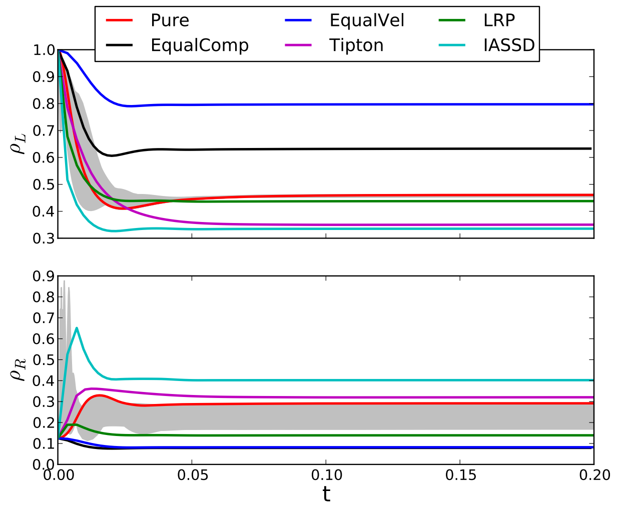
Figure 21: IASSD pressure relaxation model

In the modified shock tube problem, the main purpose is to show the multi-material closure model performances for slightly different gamma gases. The pure Lagrangian results, which does not have any multi-material cell, show very good agreement with the exact solution as shown in Figure 16. The equal compressibility and equal velocity increment model results, as expected, do not reach the equilibration values for the pressures for multi-material components in the multi-material cell as shown in Figures 10 and 11. The density and the specific internal energy values also show varying results from the equilibrium values.

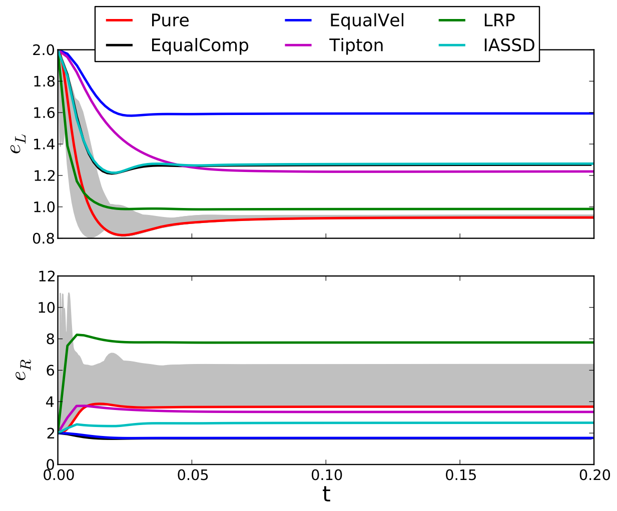
The pressure relaxation model results shown in Figures 12, 13 and 14 have similar results as the sanity test Sod shock problem.



(a) Pressure, *p*



(b) Density, *ρ*



(c) Specific internal energy, *e*

Figure 22: Time history of multi-material cell components for various closure models

In Figure 22, the multi-material closure model results are compared against a range of possible realizations of multiple material configurations in the multi-material cell. The time history of pressure, density and the specific internal energy of left and right components are plotted along with the gray bands. The gray bands represent possible realizations of multiple material configurations in the multi-material cell.

The pressure relaxation closure models obtain the desired equilibrium pressures. However, the one-step closure models such as the equal compressibility and equal velocity increment model do not. Even within the list of pressure relaxation closure models one can find various relaxation times. The Riemann problem based relaxation models converge faster to the equilibrium values in general. The density and the specific internal energy values, however, are not well predicted by any of the closure models. Overall, the modified shock tube results are quite similar to those of the standard Sod shock tube. This is expected since both of the test problems consists only of gamma gases with little difference in the gas constants.

It is conceivable that the pressure relaxation closure models are favored in this test problem due to the superior pressure equilibrium property. The one-step closure models will not be a good closure model choice if component pressures in the multi-material cell are to be utilized. Linearized Riemann Problem pressure relaxation model seems to show the best performance in all variables, compared to the plausible range of multi-material configurations, as shown in Figure 22. However, other pressure relaxation models also show acceptable results.

The Water Air Shock Tube Problem

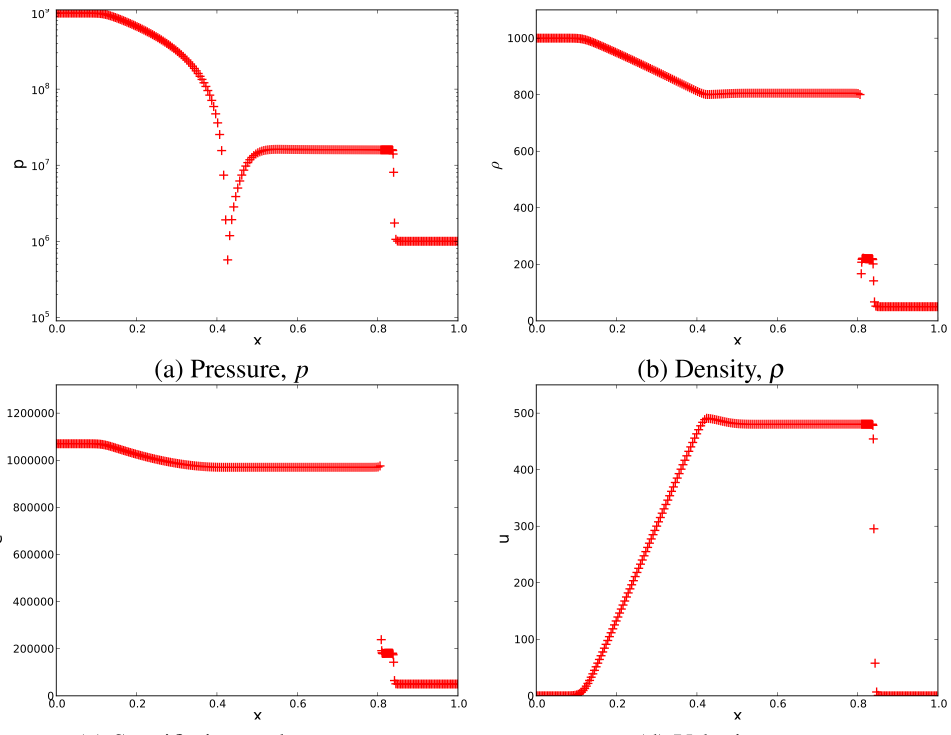
The water air shock tube problem consists of two distinct phases of materials. Water behaves like a stiffened gas, therefore, requires a stiffened-gas equation of state. The initial conditions for this problem are

$$(\gamma,\rho,e,p,u)=\begin{cases}(4.4,10^3,1.07\times10^6,10^9,0) & \text{if} \quad 0\le x<0.7\\ (1.4,50,5\times10^4,10^6,0) & \text{if} \quad 0.7\le x<1.0.\end{cases} \tag{111}$$

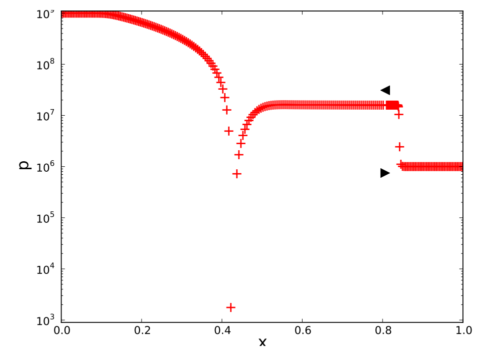
The parameters for this problem are

- $x\in[0,1]$
- $N_{\text{cells}}=249$
- $t_{\text{final}}=2.2\times10^{-4}$
- $\text{CFL}=0.1$
- The equation of state for water (stiffened-gas EOS): $(\gamma-1)\rho e-\gamma p_\infty$ where $p_\infty=6\times10^8$
- One multi-material cell located at $x=0.7$
- Multi-material sub-cells initialized with the volume fraction of $\phi_i=0.5$

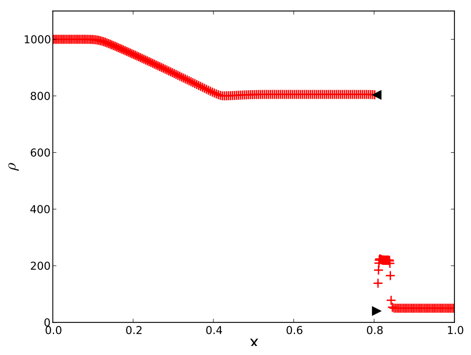
All results shown are at the final time. The symbols ◀ and ▶ indicate left and right component values in multi-material cell.



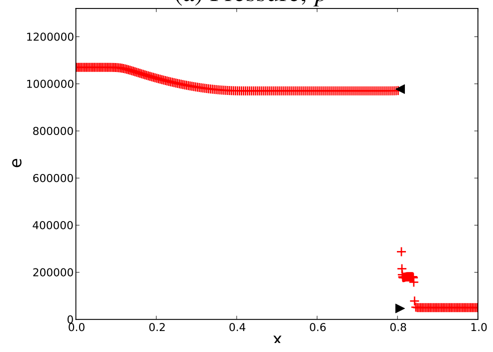
(a) Pressure, p (b) Density, ρ
(c) Specific internal energy, e (d) Velocity, u
Figure 23: Pure Lagrangian solution



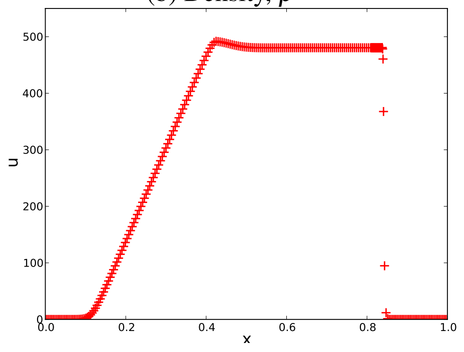
(a) Pressure, p



(b) Density, ρ

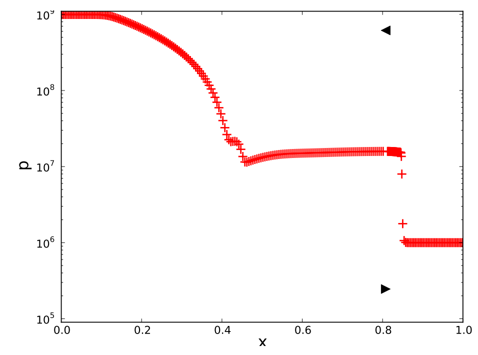


(c) Specific internal energy, e

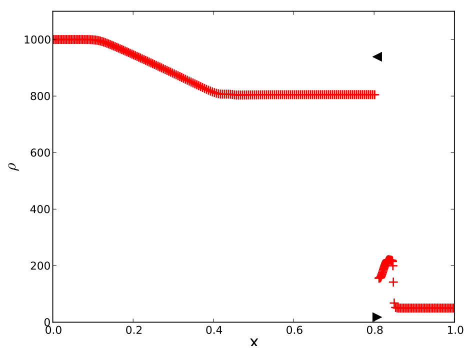


(d) Velocity, u

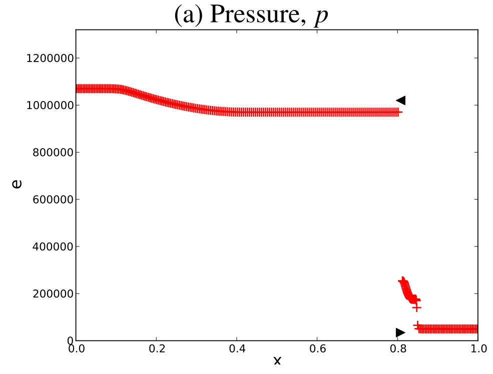
Figure 24: Equal compressibility model



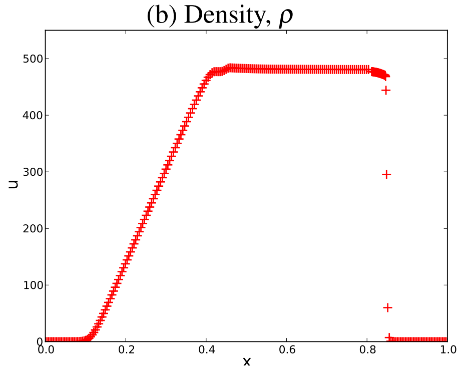
(a) Pressure, p



(b) Density, ρ

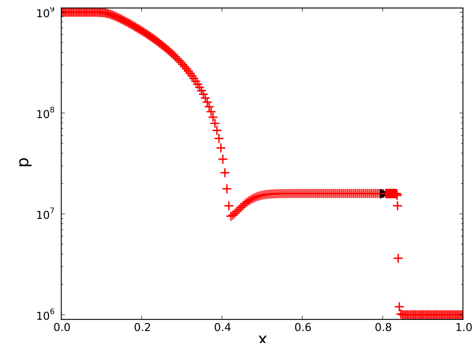


(c) Specific internal energy, e

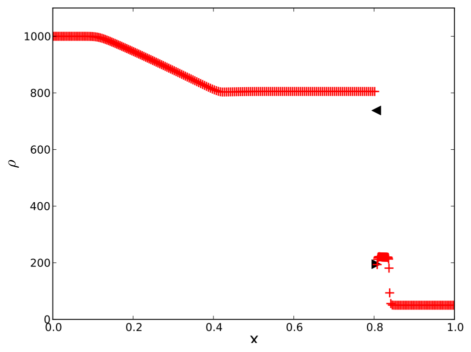


(d) Velocity, u

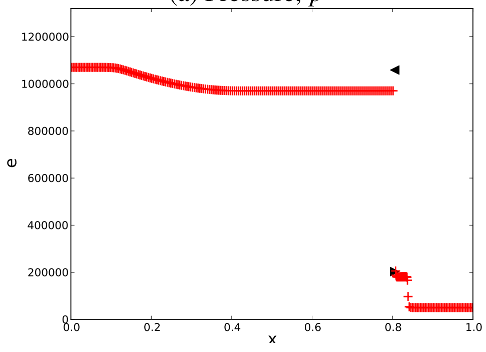
Figure 25: Equal velocity increment model



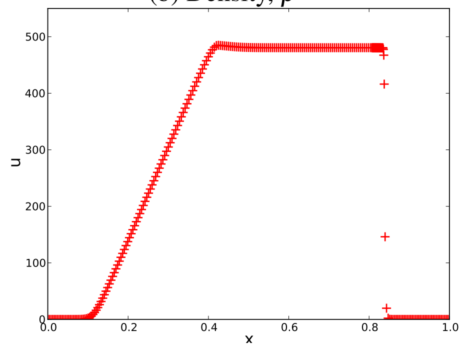
(a) Pressure, *p*



(b) Density, *ρ*

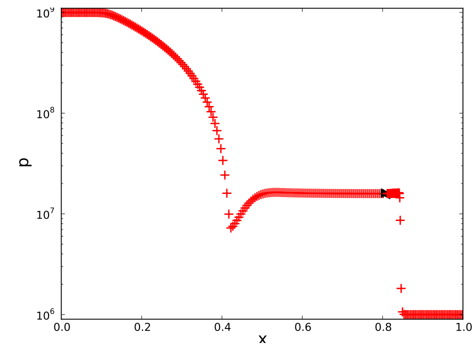


(c) Specific internal energy, *e*

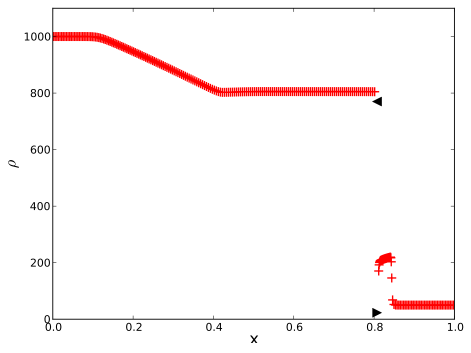


(d) Velocity, *u*

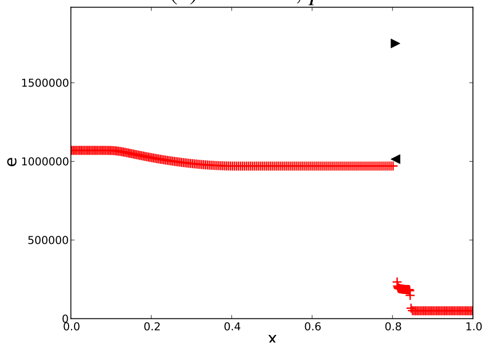
Figure 26: Tipton's pressure relaxation model



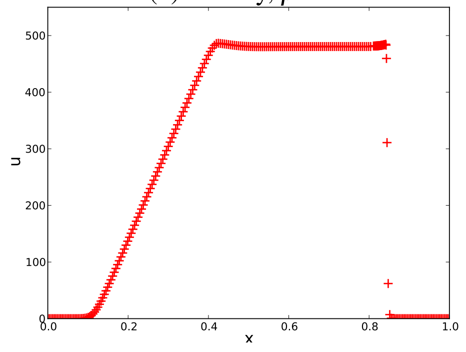
(a) Pressure, *p*



(b) Density, *ρ*

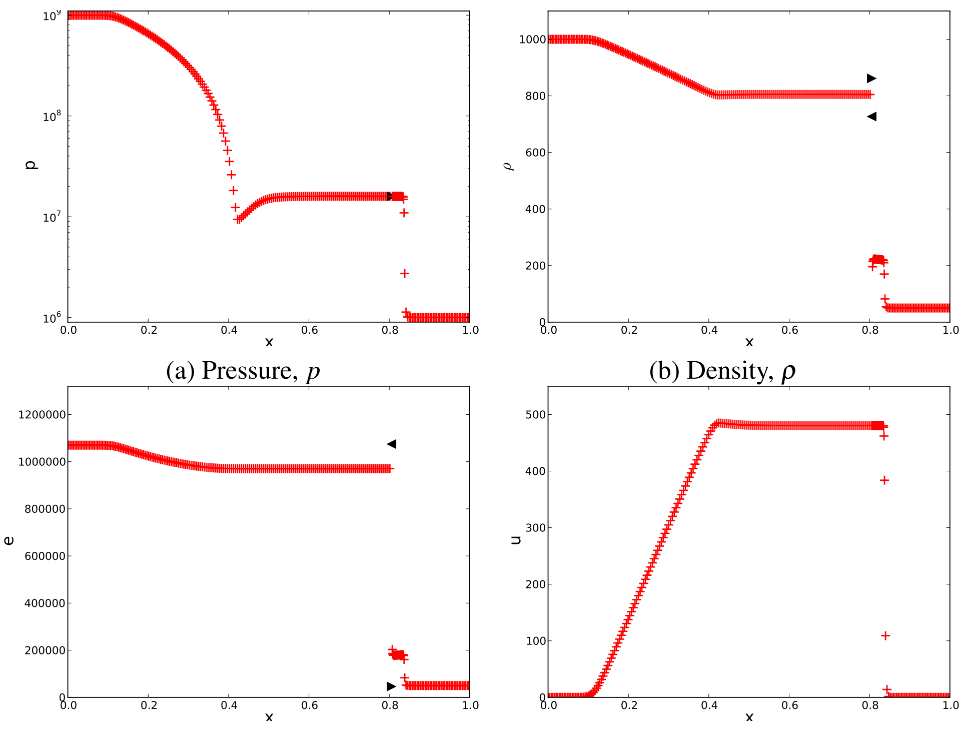


(c) Specific internal energy, *e*



(d) Velocity, *u*

Figure 27: Linearized Riemann Problem pressure relaxation model



(a) Pressure, p

(b) Density, ρ

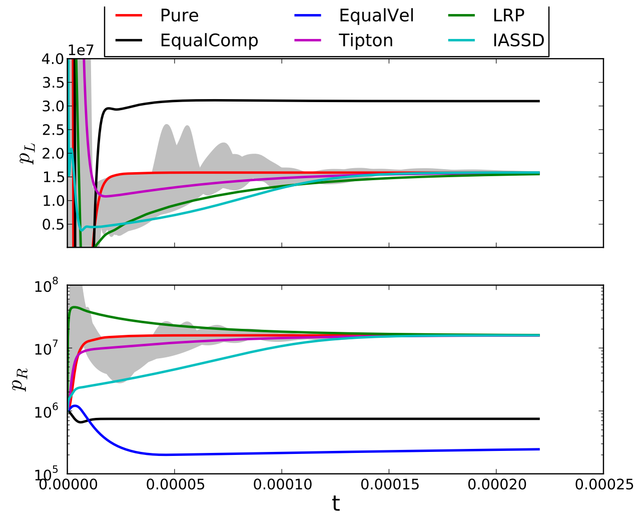
(c) Specific internal energy, e

(d) Velocity, u

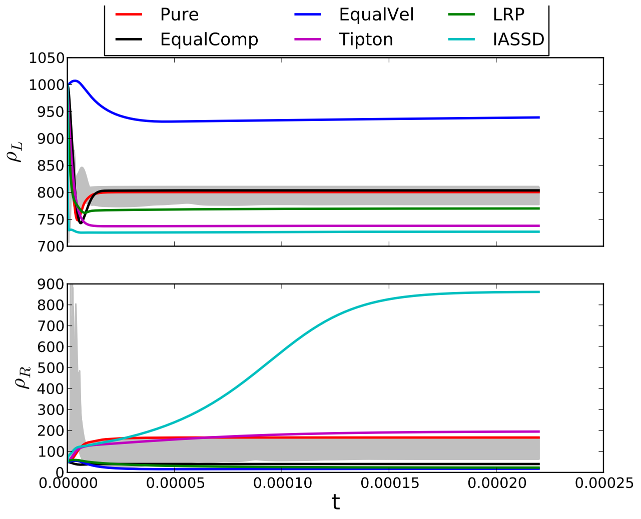
Figure 28: IASSD pressure relaxation model

The water air shock tube problem shows more interesting results. Two materials used in this test problem have different material properties and this gives rise to large jumps in pressure, density and specific internal energy values in the multi-material cell. Pressure values are scaled logarithmically in order to properly visualize. Both the pure Lagrangian and the equal compressibility show large drop in pressure at the beginning of the expansion wave, around $x = 0.4$. In fact, some of the pressure values are negative, hence not shown in the logarithmic scale, see Figure 24. Interestingly, the equal velocity increment model does not exhibit the non-physical negative pressure at the beginning of the expansion wave. Instead, a slight over-prediction of pressure at the beginning of the expansion wave is observed, Figure 25.

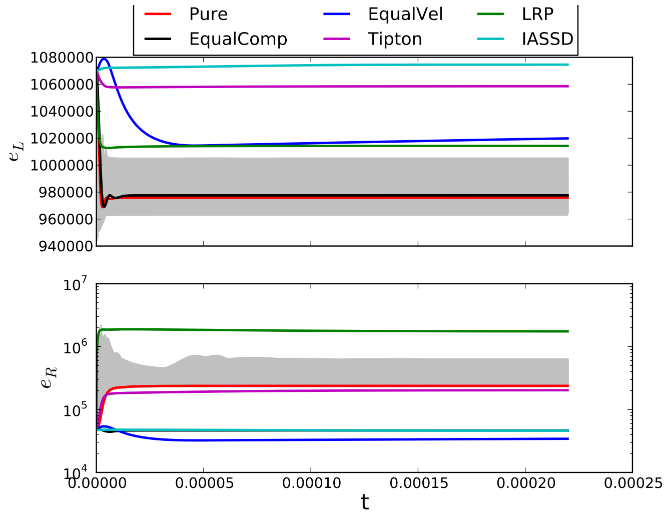
The pressure relaxation closure models all exhibit very similar pressure results. The major differences between different pressure relaxation closure models can be noticed in density and specific internal energy results. Tipton's model tends to more accurately model density and specific internal energy values of individual components with respect to the neighboring quantities, see Figure 26. Linearized Riemann Problem model results do not show any resemblance of the immediate neighboring properties, see Figure 27. However, IASSD model shows similar component density values but not specific internal energy values, as shown in Figure 28. In this test problem, Tipton's pressure relaxation closure model converges the fastest and IASSD model converges the slowest.



(a) Pressure, p



(b) Density, ρ



(c) Specific internal energy, e

Figure 29: Time history of multi-material cell components for various closure models

In Figure 29, the multi-material closure model results are compared against a range of possible realizations of multiple material configurations in the multi-material cell. Time history of pressure, density and specific internal energy values of left and right components are plotted along with the gray bands. The gray bands represent possible realizations of multiple material configurations in the multi-material cell. The pressures of equal compressibility and equal velocity increment closure models do not even come close to the accepted range.

Among all pressure relaxation models, IASSD closure model shows interesting density convergence. The right component of density steadily increases above the value of left component. This suggests that the density of air, located right, exceeds that of water, located left.

The Water-Air Shock Tube Problem – Bubbly interface

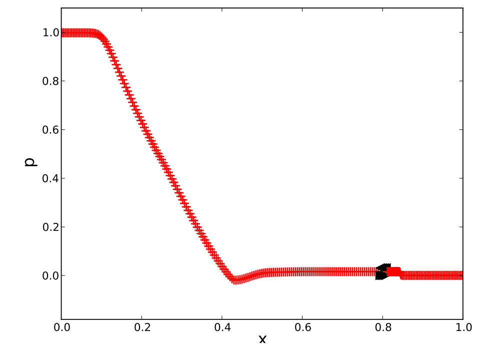
In this problem, several bubbly interfaces are initialized between two materials. The bubbly interface is realized by series of alternating water and air initial conditions. This problem shows how multi-material closure models perform when multiple multi-material cells exist in a series. The initial conditions for this problem are

$$(\gamma,\rho,e,p,u) = \begin{cases} (4.4, 10^3, 1.07\times10^6, 10^9, 0) & \text{if} \quad 0 \le x < 0.7-2\Delta x \\ (1.4, 50, 5\times10^4, 10^6, 0) & \text{if} \quad 0.7+2\Delta x \le x < 1.0 \end{cases} \tag{112}$$

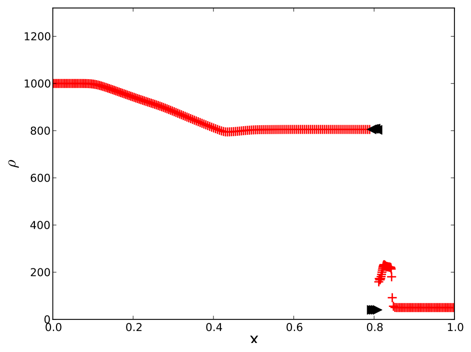
The parameters for this problem are

- $x \in [0,1]$
- $N_{\text{cells}} = 253$
- $t_{\text{final}} = 2.2\times10^{-4}$
- $\text{CFL} = 0.1$
- The equation of state for water (stiffened-gas EOS): $(\gamma-1)\rho e - \gamma p_\infty$ where $p_\infty = 6\times10^8$
- Four multi-material cells located at $x = 0.7, 0.7+\Delta x, 0.7+2\Delta x, 0.7+3\Delta x$
  - The material configurations within all multi-material cells is water and air, from left to right.
- Multi-material sub-cells initialized with the volume fraction of $\phi_i = 0.5$

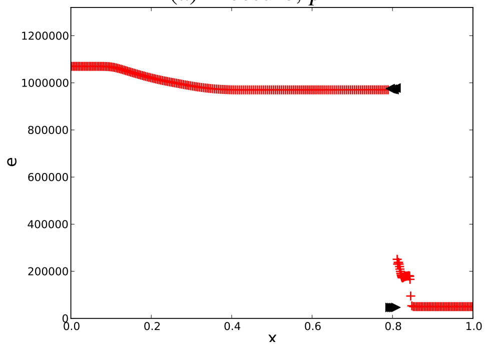
All results shown are at the final time. The symbols ◀ and ▶ indicate left and right component values in multi-material cell.



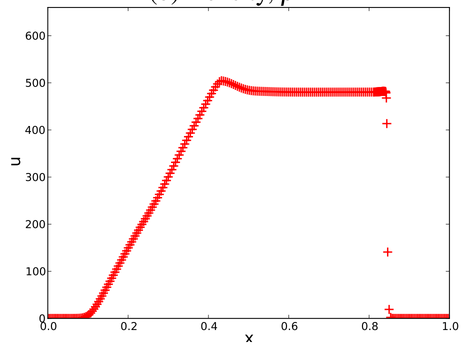
(a) Pressure, p



(b) Density, ρ

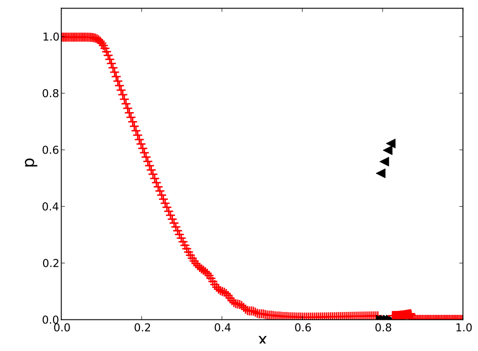


(c) Specific internal energy, e

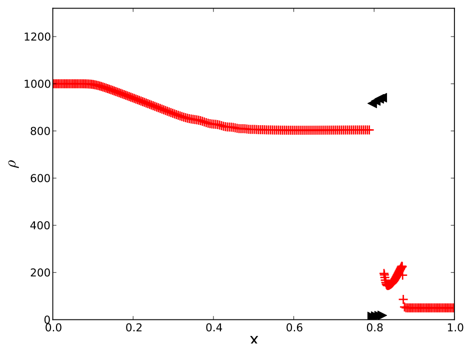


(d) Velocity, u

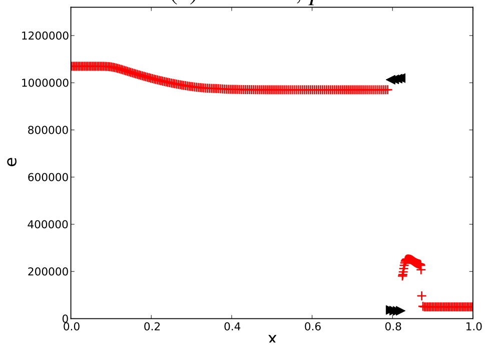
Figure 30: Equal compressibility model



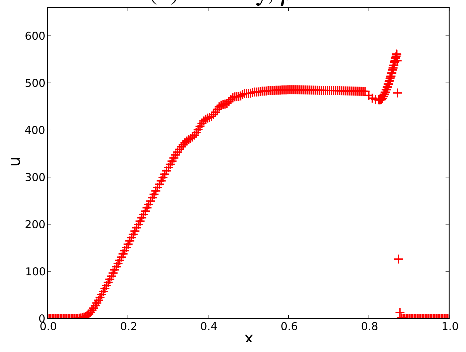
(a) Pressure, p



(b) Density, ρ

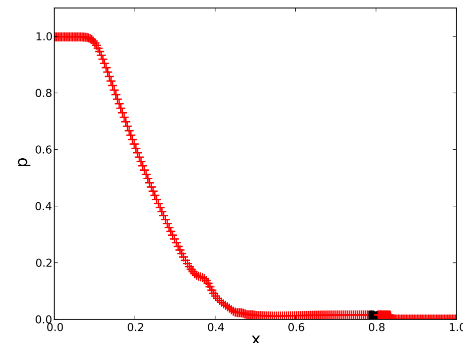


(c) Specific internal energy, e

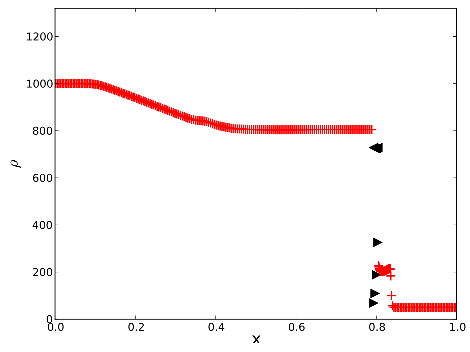


(d) Velocity, u

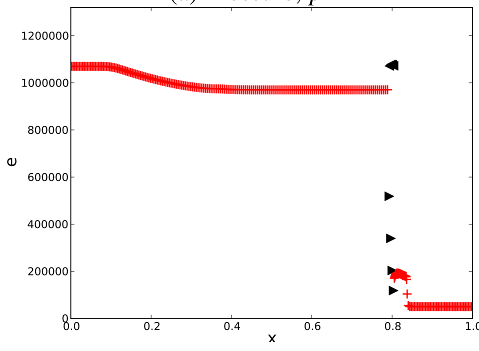
Figure 31: Equal velocity increment model



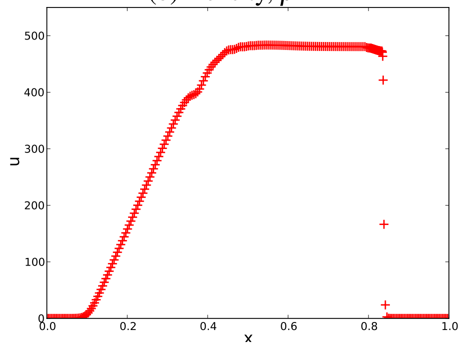
(a) Pressure, p



(b) Density, ρ

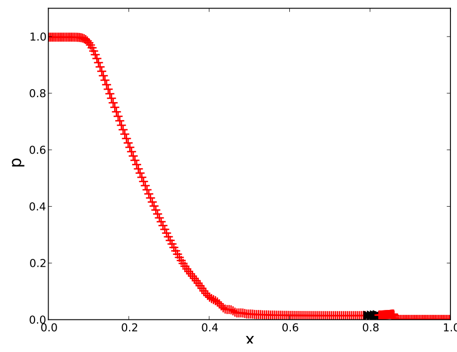


(c) Specific internal energy, e

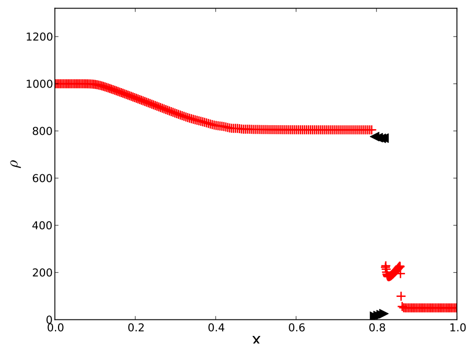


(d) Velocity, u

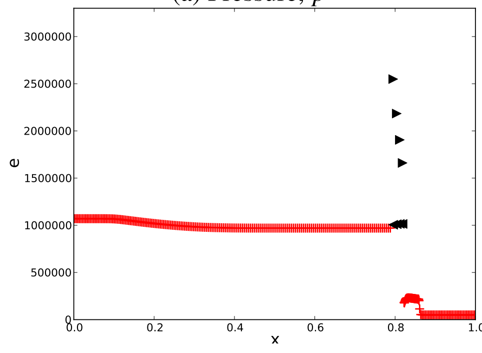
Figure 32: Tipton's pressure relaxation model



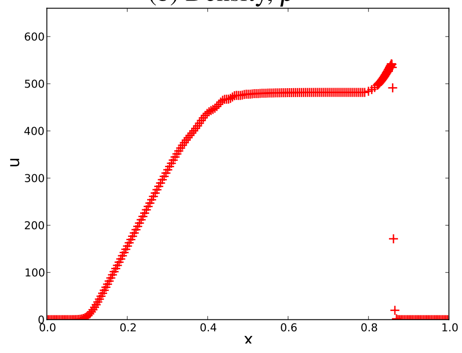
(a) Pressure, p



(b) Density, ρ

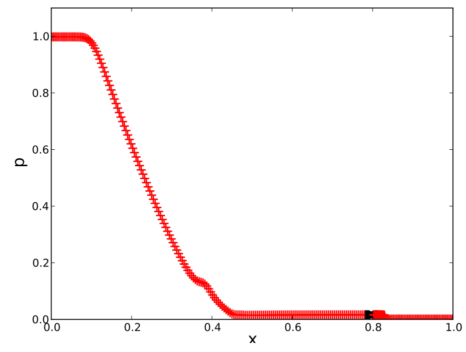


(c) Specific internal energy, e

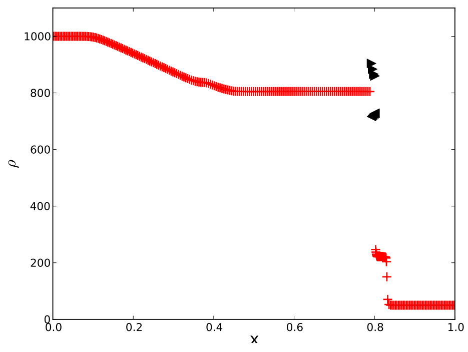


(d) Velocity, u

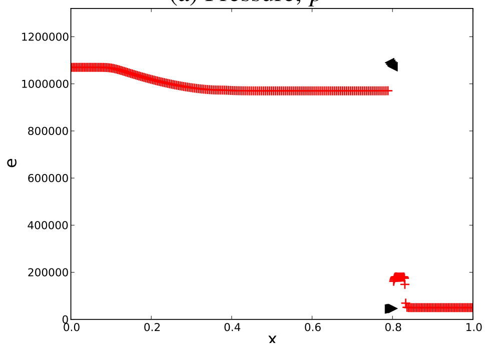
Figure 33: Linearized Riemann Problem pressure relaxation model



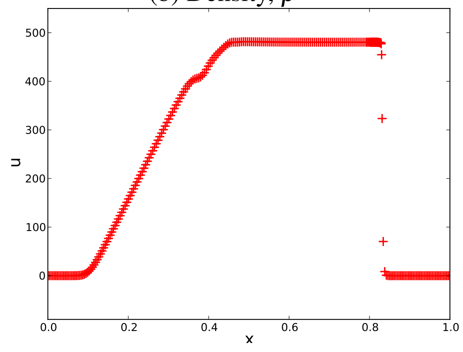
(a) Pressure, p



(b) Density, ρ



(c) Specific internal energy, e



(d) Velocity, u

Figure 34: IASSD pressure relaxation model

This test problem approximately models water and air separated by a finite thickness water air bubble interface. It can be recalled that most multi-material closure models do not make assumptions about the multi-material configuration within the cell. Some exceptions are the Riemann Problem based closure models such as Linearized Riemann Problem and IASSD models which require the interface as well as material orientation within the multi-material cell. The one-step closure model results vary greatly from one model to another. The differences are quite noticeable in pressure and velocity profiles.

The pressure relaxation closure model results do not show much variations in pressure value profiles. However, the density and the specific internal energy values are evaluated differently. Tipton's pressure relaxation closure model results scatters the density and specific internal energy values in the multi-material cells. On the other hand, IASSD model show somewhat congregated results for the density and specific internal energy values. It is interesting to note that almost all of the right values gather around other right values, and left values gather around other left values. Linearized Riemann Problem model shows intermediate results of Tipton's and IASSD results.

The Vacuum Problem

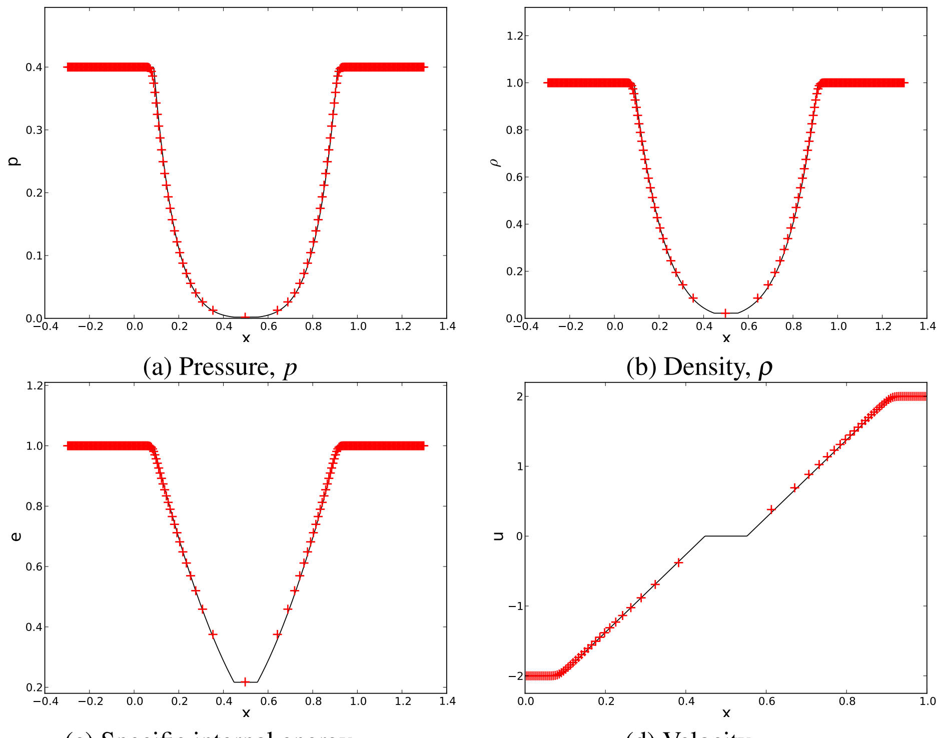
The vacuum problem consists only of air, a gamma gas. The multi-material cell undergoes a rapid expansion caused by the vertex velocities imposed on it. This problem tests how the multi-material closure model behaves under rapid expansion of velocity field. The initial conditions for this problem are

$$(\gamma,\rho,e,p,u) = \begin{cases} (1.4,1,1,0.4,-2) & \text{if} \quad 0 \le x < 0.5 \\ (1.4,1,1,0.4,2) & \text{if} \quad 0.5 \le x < 1.0 \end{cases} \tag{113}$$

The parameters for this problem are

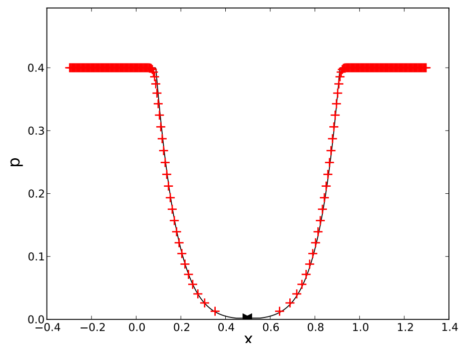
- $x \in [0,1]$
- $N_{\text{cells}} = 200$
- $t_{\text{final}} = 0.15$
- $\text{CFL} = 0.1$
- One multi-material cell located at $x = 0.5$
- Multi-material sub-cells initialized with the volume fraction of $\phi_i = 0.5$

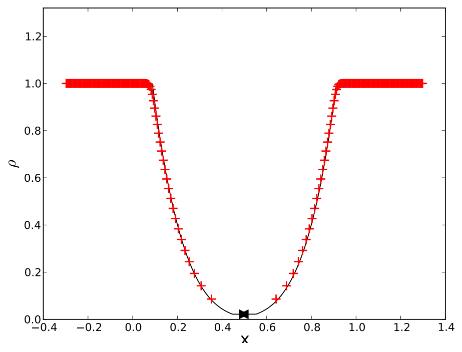
All results shown are at the final time. The black solid lines indicate the exact solutions. The symbols ◀ and ▶ indicate left and right component values in multi-material cell.

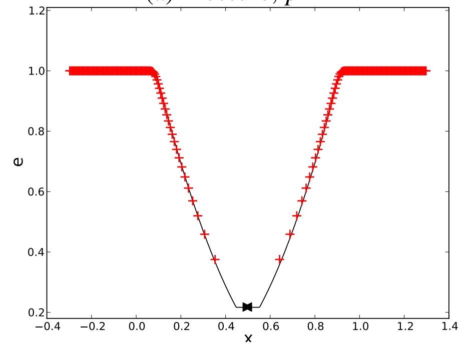

(a) Pressure, p (b) Density, ρ

(c) Specific internal energy, e (d) Velocity, u

Figure 35: Pure Lagrangian solution


(a) Pressure, p


(b) Density, ρ


(c) Specific internal energy, e

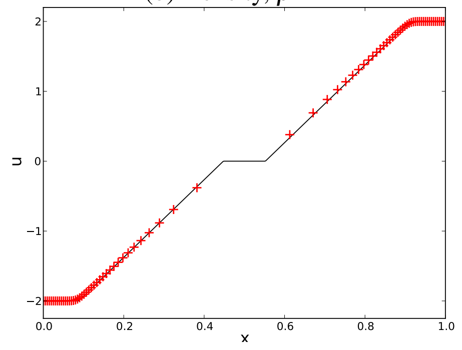
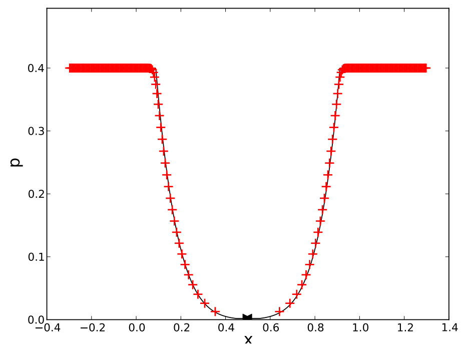
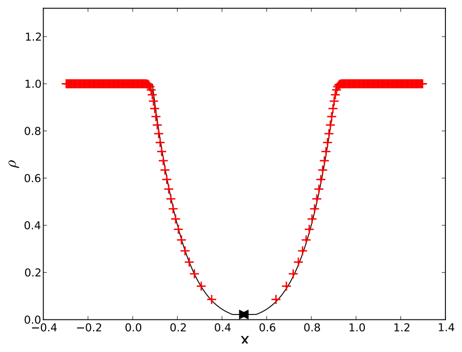
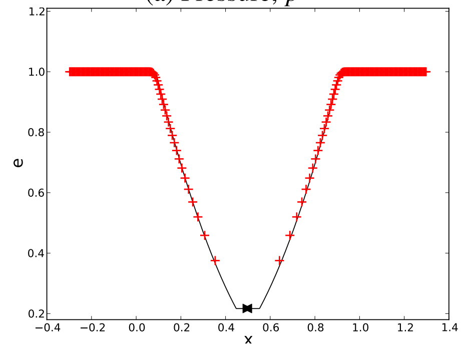

(d) Velocity, u

Figure 36: Equal compressibility model


(a) Pressure, p


(b) Density, ρ


(c) Specific internal energy, e

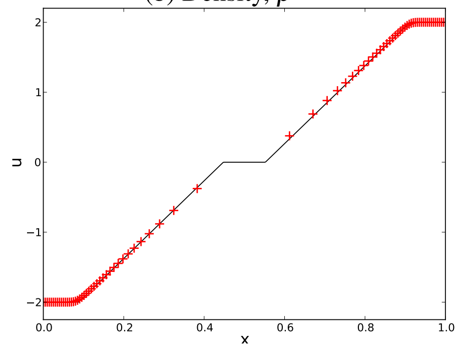
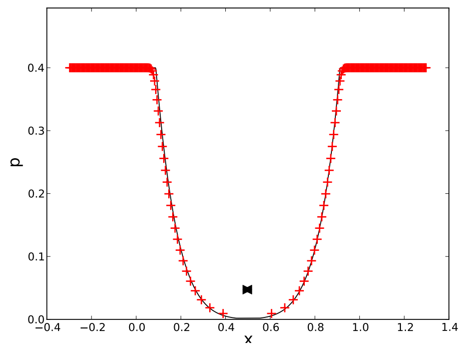
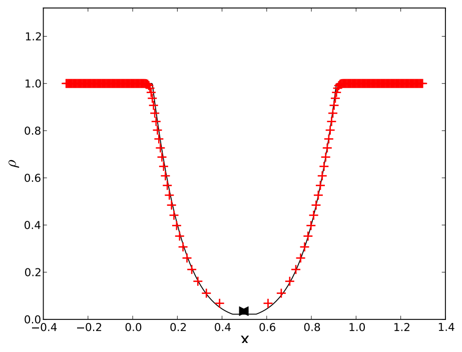

(d) Velocity, u

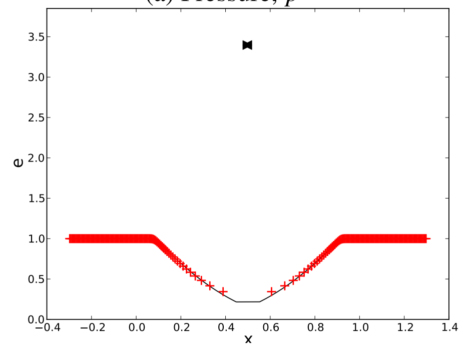
Figure 37: Equal velocity increment model



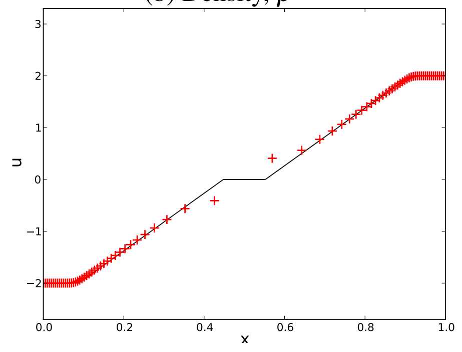
(a) Pressure, p



(b) Density, ρ

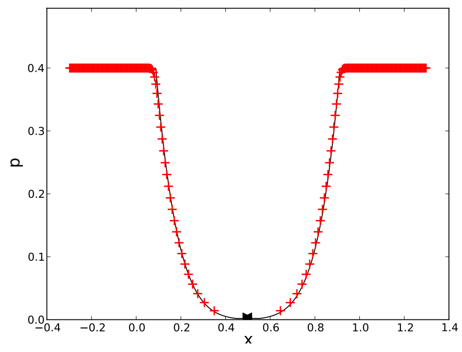


(c) Specific internal energy, e

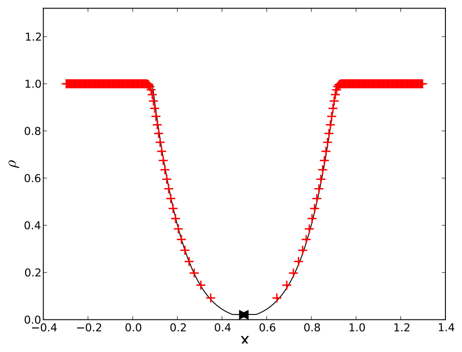


(d) Velocity, u

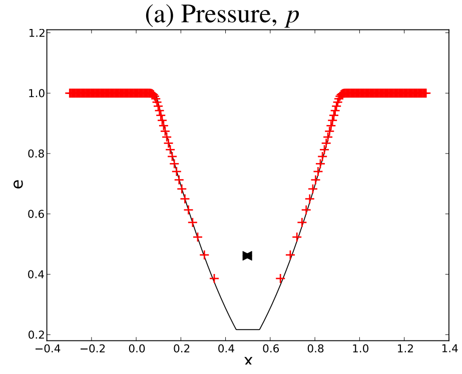
Figure 38: Tipton's pressure relaxation model



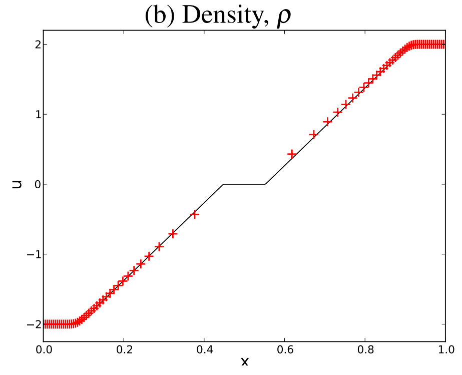
(a) Pressure, p



(b) Density, ρ

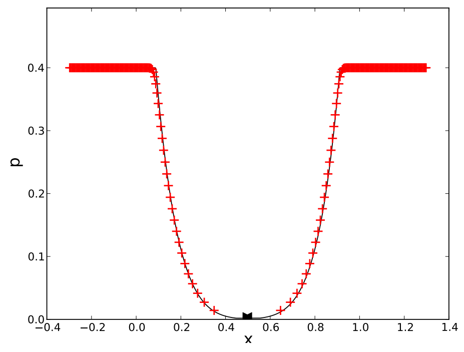


(c) Specific internal energy, e

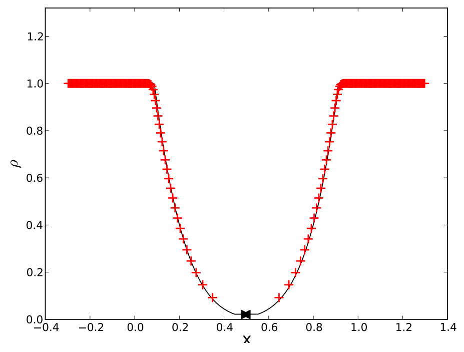


(d) Velocity, u

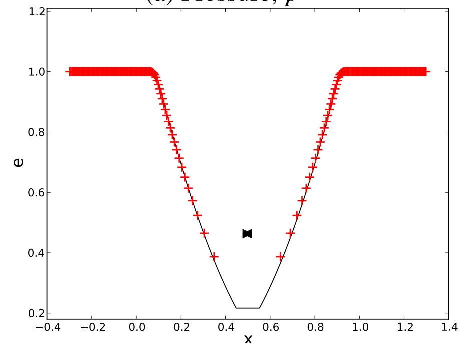
Figure 39: Linearized Riemann Problem pressure relaxation model



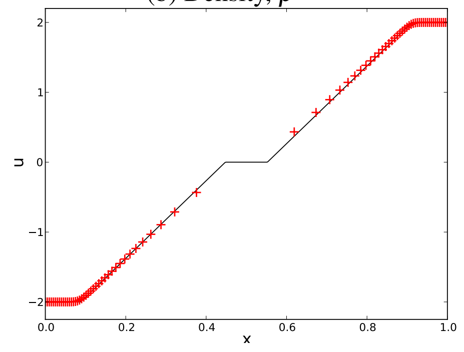
(a) Pressure, p



(b) Density, ρ



(c) Specific internal energy, e

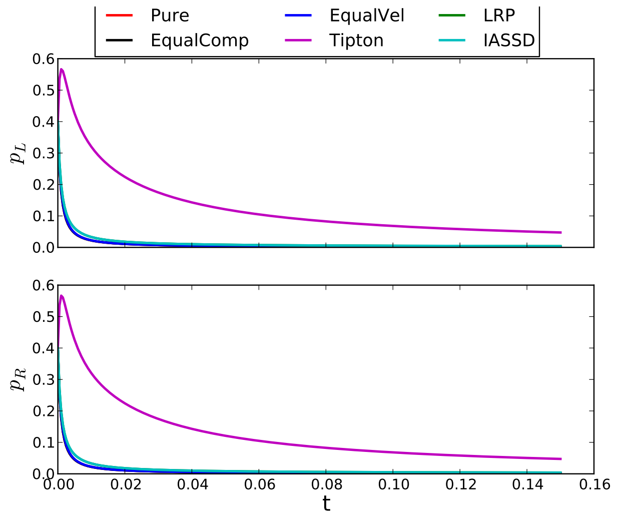


(d) Velocity, u

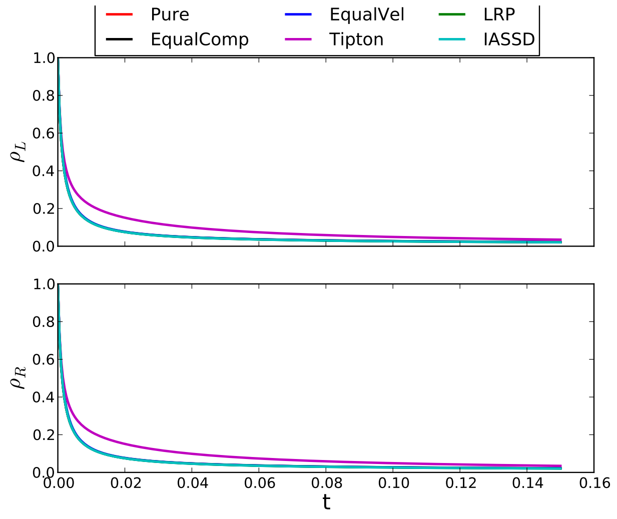
Figure 40: IASSD pressure relaxation model

One special observation for this test problem is that all one-step closure model results converge to the exact solutions. One possible explanation for this behavior is the initial condition. Unlike previous shock tube problems where left and right components of the multi-material cell did not have the same initial conditions, the vacuum problem has symmetric initial conditions for the left and right components.

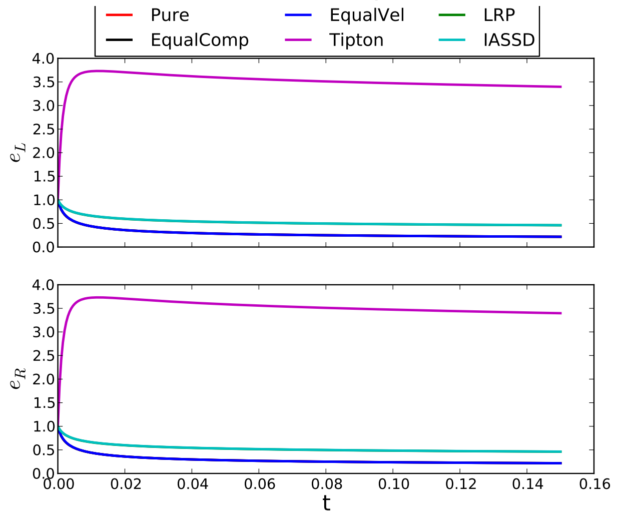
On the other hand, some pressure relaxation closure models do not converge to the exact solutions, especially in specific internal energy and density values. Tipton's closure model, Figure 38, seems to show the largest variation from the exact solution. This peculiar results are probably owing to the pressure relaxation mechanisms since non-pressure relaxation models did not show the similar results.



(a) Pressure, p



(b) Density, ρ



(c) Specific internal energy, e

Figure 41: Time history of multi-material cell components for various closure models

In Figure 41, the multi-material closure model results are compared against the pure Lagrangian results in the multi-material cell. Time history of pressure, density and specific internal energy values of left and right components are plotted. The pure Lagrangian results are not visible since they lie behind the equal compressibility and equal velocity increment model results.

The vacuum problem where a multi-material cell goes through rapid expansion reveals different characteristics of closure models. We discovered that simple one-step models perform better than more sophisticated pressure relaxation models. Tipton's pressure relaxation closure model showed the worst convergence history for all variables.

The Shock Contact Problem – Gas-Gas Interaction

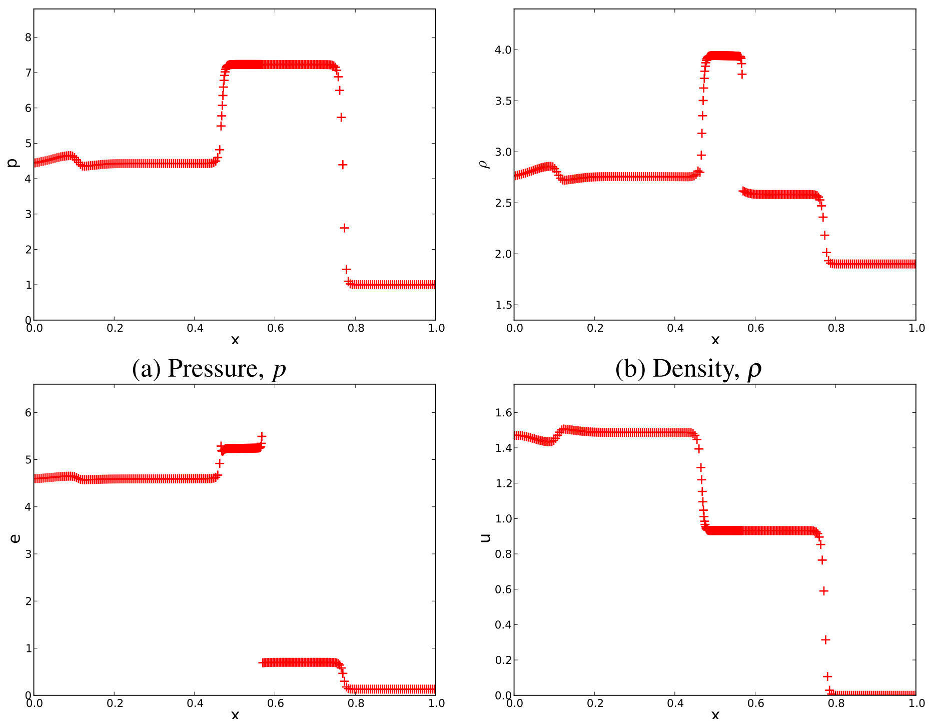
The shock contact problem tests the transmission and reflection of Mach 2 shock through an initially stationary contact surface located in the multi-material cell. The multi-material cell consists of two different gamma gases. The shock wave arrives at the interface at $t \doteq 0.172$. High precision initial condition data is given in [9]. The initial conditions for this problem are

$$(\gamma, \rho, e, p, u) = \begin{cases} (1.35, 2.76, 4.60, 4.45, 1.48) & \text{if} \quad -0.37 \le x < 0.1 \\ (1.35, 1.0, 2.86, 1.0, 0) & \text{if} \quad 0.1 \le x < 0.5 \\ (5.0, 1.9, 0.132, 1.0, 0) & \text{if} \quad 0.5 \le x < 1.0. \end{cases} \tag{114}$$

The parameters for this problem are

- $x \in [-0.37, 1]$
- $N_{\text{cells}} = 274$
- $t_{\text{final}} = 0.25$
- $\text{CFL} = 0.1$
- One multi-material cell located at $x = 0.5$
- Multi-material sub-cells initialized with the volume fraction of $\phi_i = 0.5$

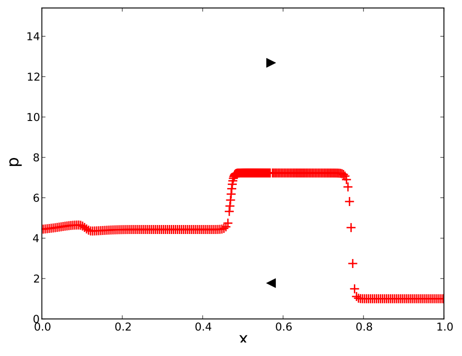
All results shown are at the final time.The symbols ◀ and ▶ indicate left and right component values in multi-material cell.



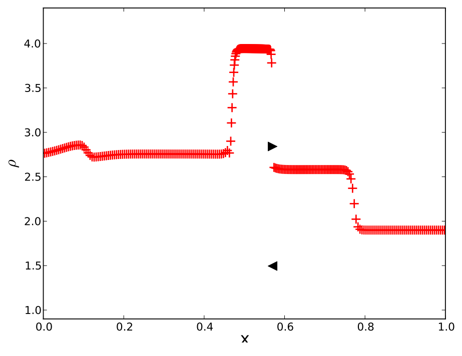
(a) Pressure, p (b) Density, ρ

(c) Specific internal energy, e (d) Velocity, u

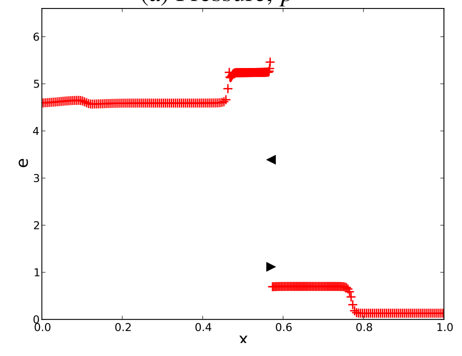
Figure 42: Pure Lagrangian solution



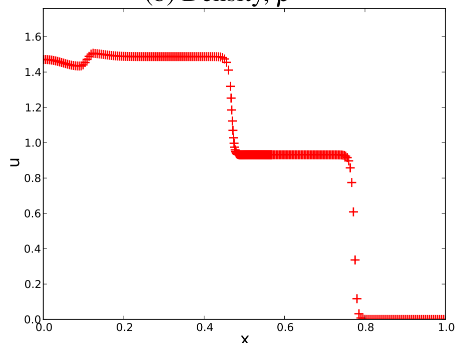
(a) Pressure, p



(b) Density, ρ

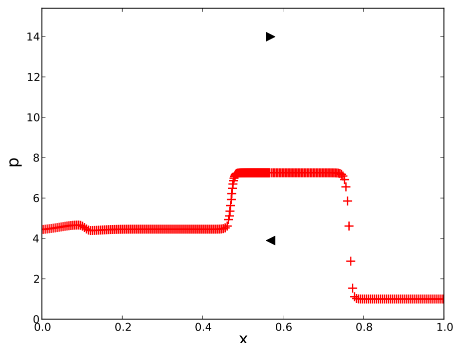


(c) Specific internal energy, e

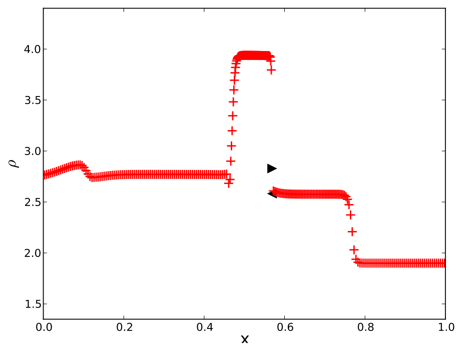


(d) Velocity, u

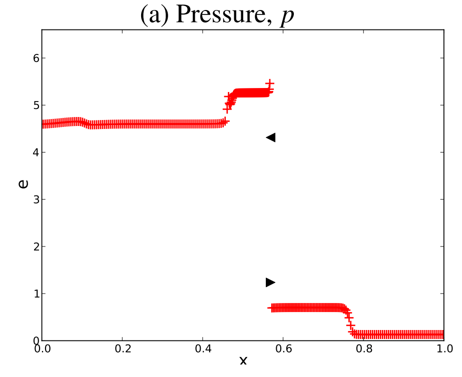
Figure 43: Equal compressibility model



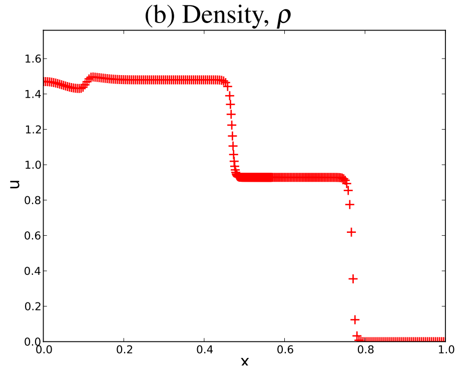
(a) Pressure, p



(b) Density, ρ

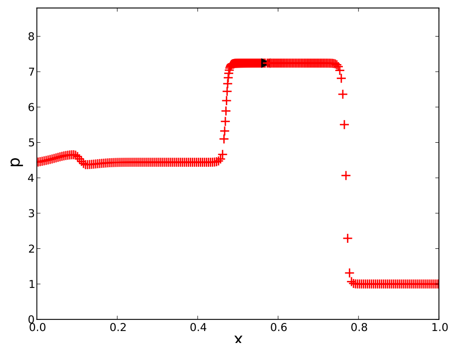


(c) Specific internal energy, e

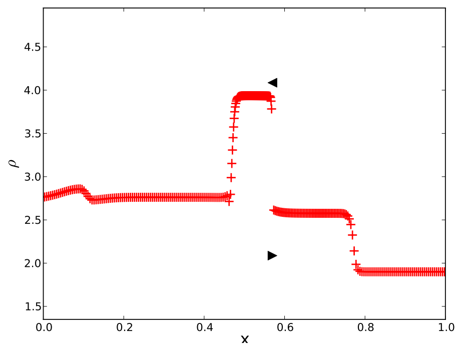


(d) Velocity, u

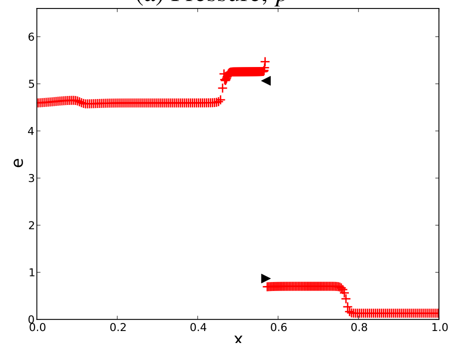
Figure 44: Equal velocity increment model



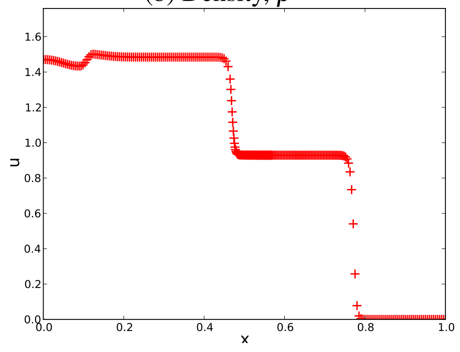
(a) Pressure, p



(b) Density, ρ

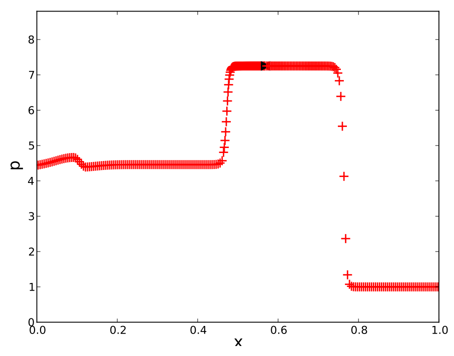


(c) Specific internal energy, e

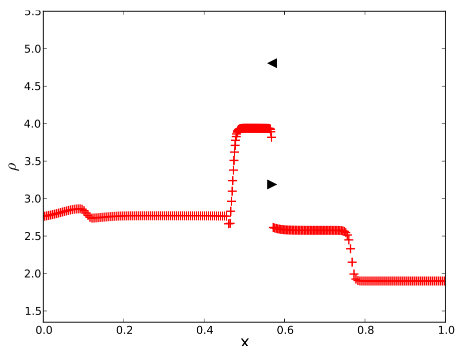


(d) Velocity, u

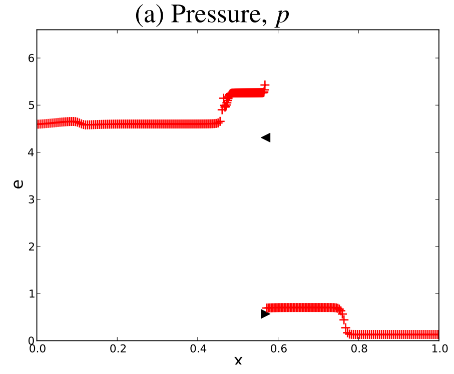
Figure 45: Tipton's pressure relaxation model



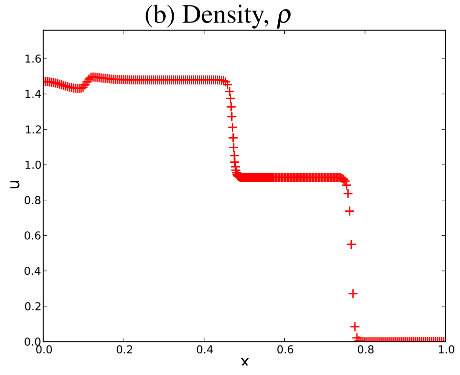
(a) Pressure, p



(b) Density, ρ

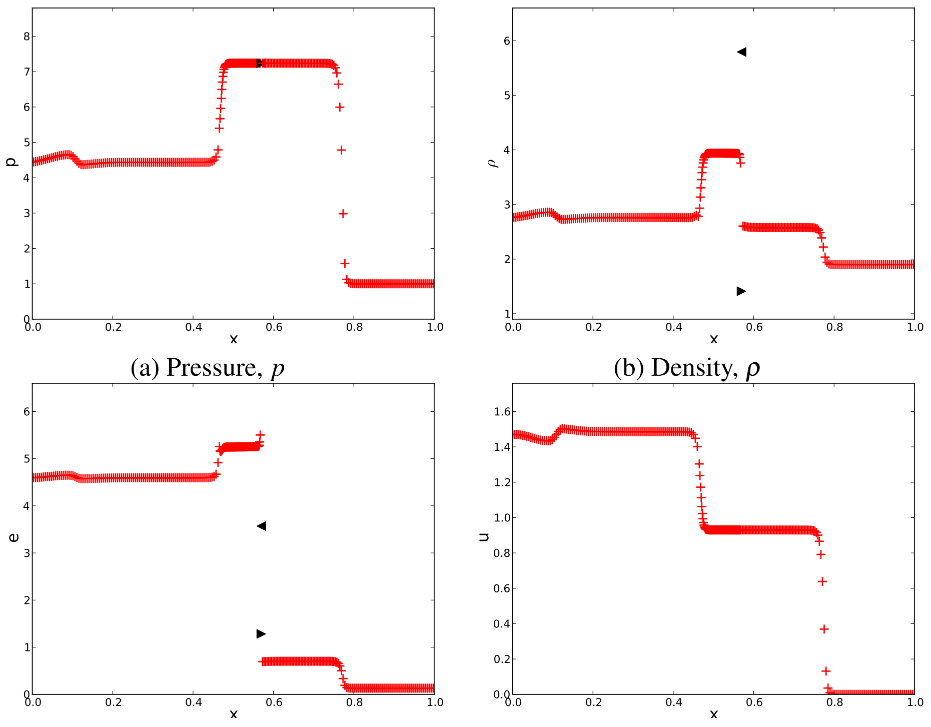


(c) Specific internal energy, e



(d) Velocity, u

Figure 46: Linearized Riemann Problem pressure relaxation model



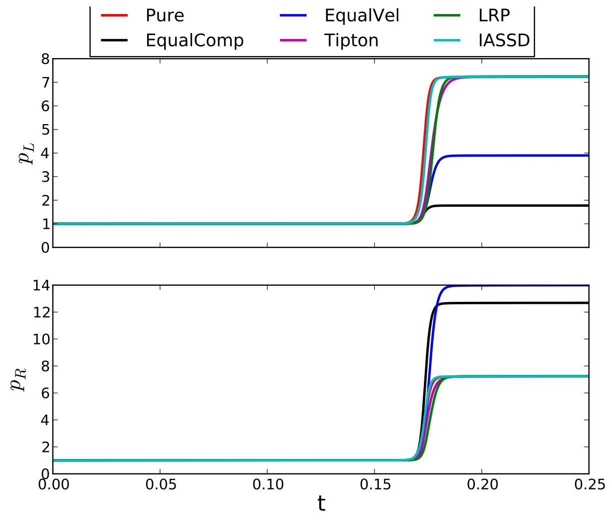
(a) Pressure, p (b) Density, ρ

(c) Specific internal energy, e (d) Velocity, u

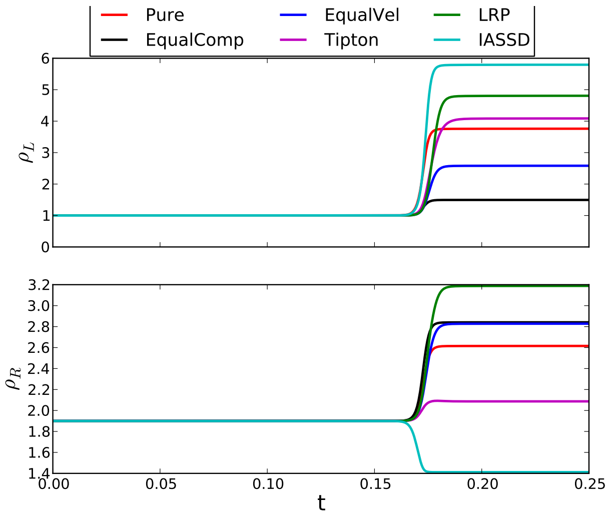
Figure 47: IASSD pressure relaxation model

At the final time $t = 0.25$, the results indicate that the multi-material cell has been influenced by an incoming shock at Mach 2.0 then again a transmitted and a reflected shock. More information about the problem can be found in [9]. All models succeed in capturing the transmission as well as the reflection of shock wave after the interaction at the contact surface. The equal compressibility and equal velocity increment models certainly lack in giving physical component pressures.

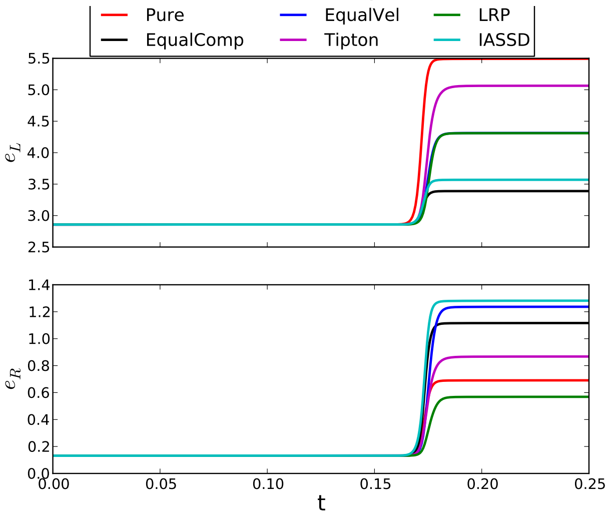
All pressure relaxation models correctly evaluate the component pressures. The difference, as we have observed over and over, comes down to how accurately one model computes the specific internal energy or the density of the components. All pressure relaxation closure models have over or undershoots in either the density or the specific internal energy values.



(a) Pressure, p



(b) Density, ρ



(c) Specific internal energy, e

Figure 48: Time history of multi-material cell components for various closure models

In Figure 48, the multi-material closure model results are compared to the pure Lagrangian results in the multi-material cell. It is easy to visualize that the pressure relaxation closure models have good agreements in the pressure of components, however, not in other variables. It is also easy to visualize the relaxation times compared to the pure Lagrangian results. IASSD model follows the pure Lagrangian component pressure results very closely.

The Shock Contact Problem – Gas-Solid Interaction

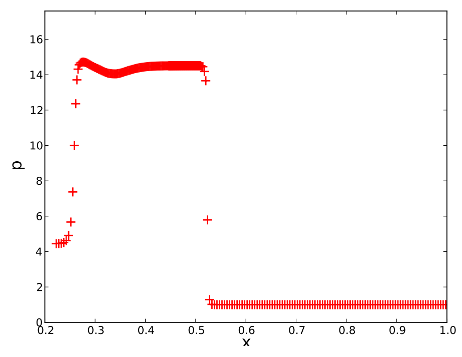
This problem is a variation of the shock contact problem involving two different phases of materials. The material located on the right side of the contact surface is a solid aluminium 2024 (AL 2024). The left material is still a gamma gas. Mach 2 shock initialized in the gas reaches the contact surface where two materials are mixed. This problem shows the transmission and reflection of shock waves in two different phases of materials. The shock wave arrives at the interface at $t \doteq 0.172$. This problem is a difficult one to model because not only there are two very different materials but also there is no strength model available for aluminium. The initial conditions for this problem are

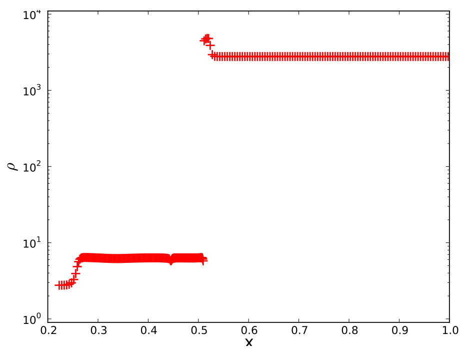
$$(\gamma,\rho,e,p,u) = \begin{cases} (1.35, 2.76, 4.60, 4.45, 1.48) & \text{if} \quad -0.37 \le x < 0.1 \\ (1.35, 1.0, 2.86, 1.0, 0) & \text{if} \quad 0.1 \le x < 0.5 \\ (5.0, 2.78\times 10^{3}, 1.799\times 10^{-4}, 1.0, 0) & \text{if} \quad 0.5 \le x < 1.0. \end{cases} \tag{115}$$

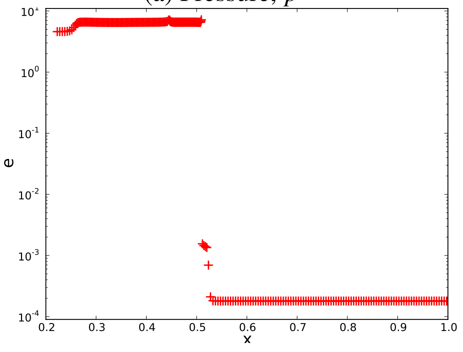
The parameters for this problem are

- $x \in [-0.37, 1]$
- $N_{\text{cells}} = 274$
- $t_{\text{final}} = 0.25$
- $\text{CFL} = 0.1$
- Aluminium 2024 equation of state: $p = 0.7906X + 1.3250X^2 + 2.130X^3 + 2.00\,(\rho e)$ where $X = \left(1 - \frac{\rho_0}{\rho}\right)$
- One multi-material cell located at $x = 0.5$
- Multi-material sub-cells initialized with the volume fraction of $\phi_i = 0.5$

All results shown are at the final time. The symbols ◀ and ▶ indicate left and right component values in the multi-material cell.


(a) Pressure, p


(b) Density, ρ


(c) Specific internal energy, e

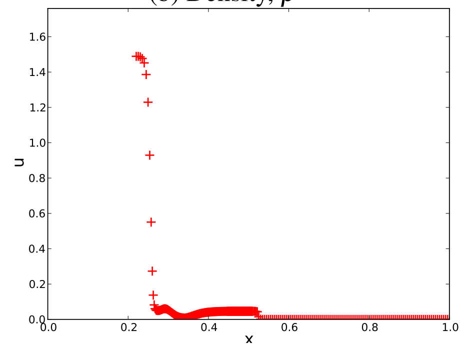
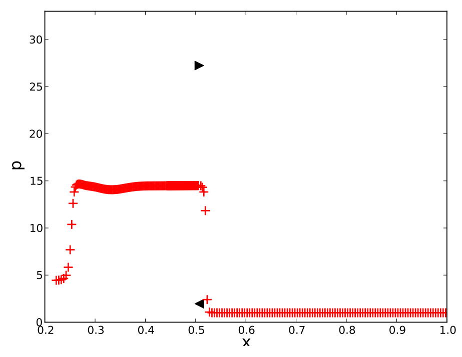
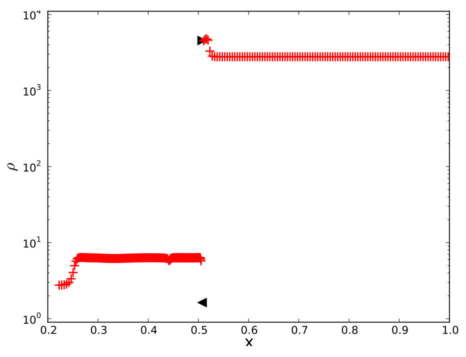
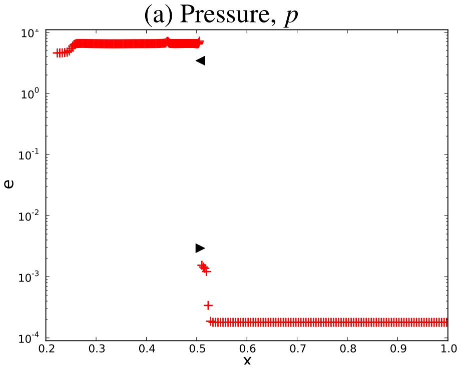

(d) Velocity, u

Figure 49: Pure Lagrangian solution


(a) Pressure, p


(b) Density, ρ


(c) Specific internal energy, e

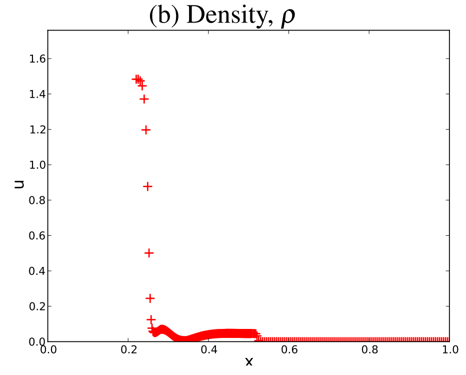
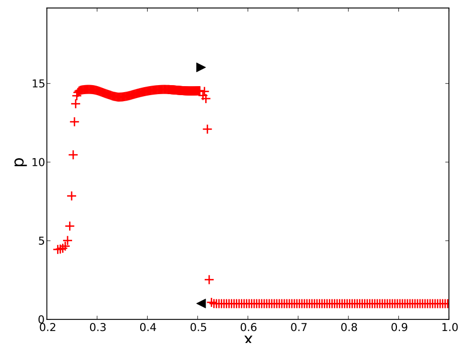
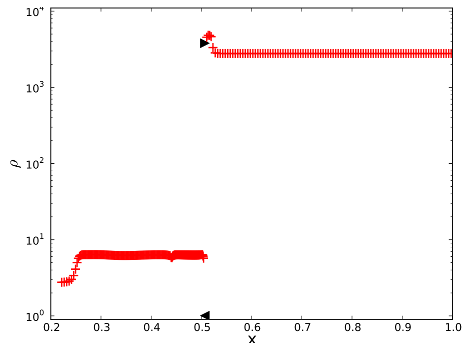

(d) Velocity, u

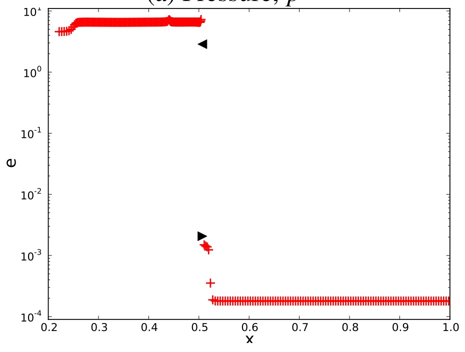
Figure 50: Equal compressibility model



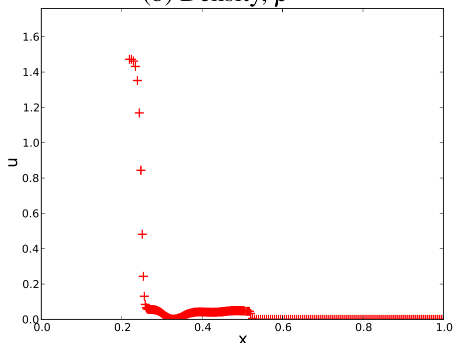
(a) Pressure, *p*



(b) Density, ρ

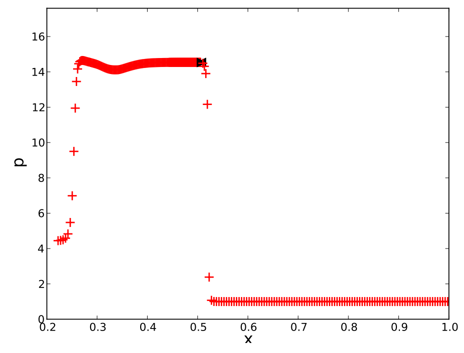


(c) Specific internal energy, *e*

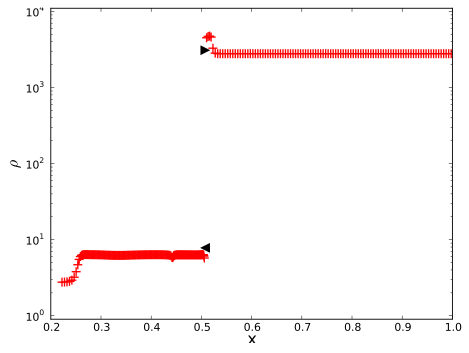


(d) Velocity, *u*

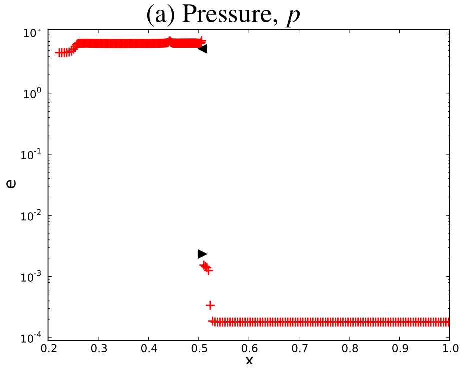
Figure 51: Equal velocity increment model



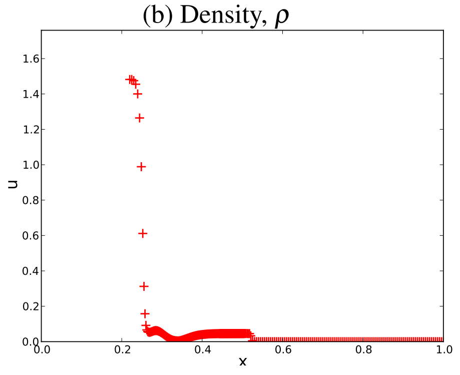
(a) Pressure, *p*



(b) Density, ρ

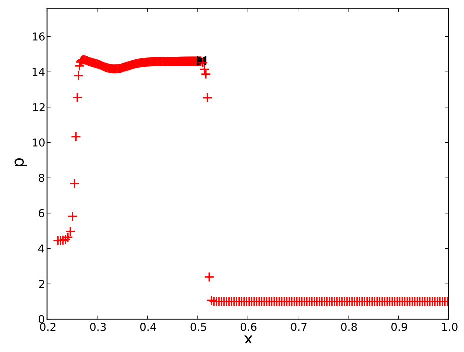


(c) Specific internal energy, *e*

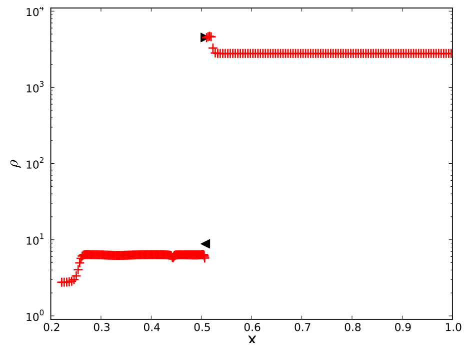


(d) Velocity, *u*

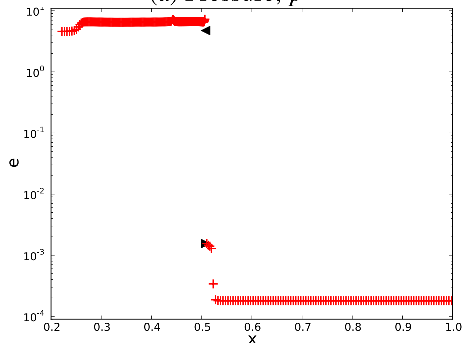
Figure 52: Tipton's pressure relaxation model



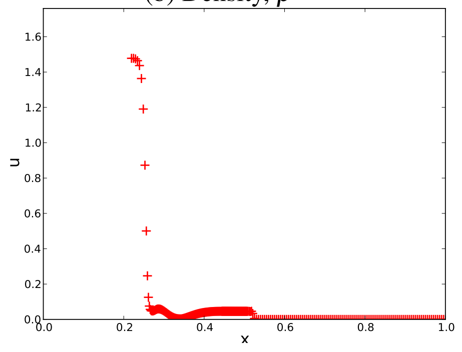
(a) Pressure, p



(b) Density, ρ

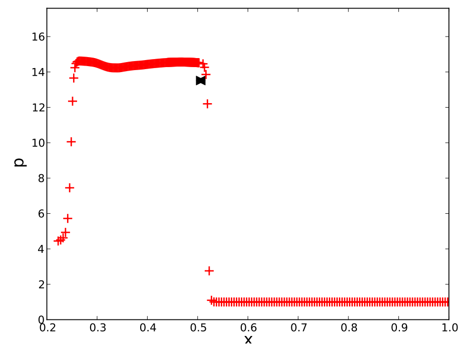


(c) Specific internal energy, e

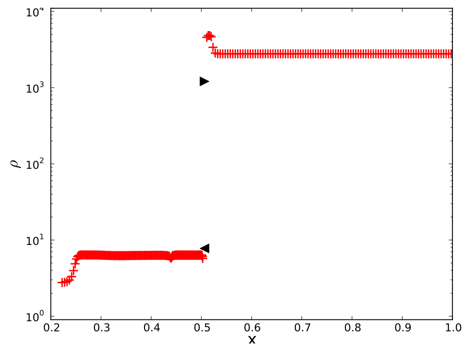


(d) Velocity, u

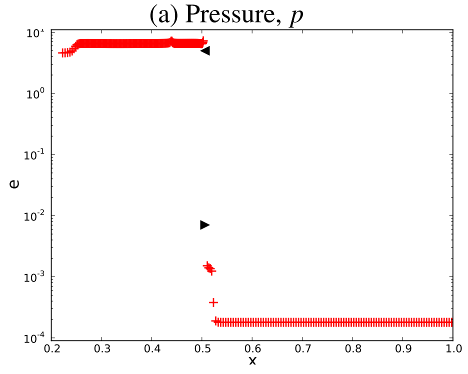
Figure 53: Linearized Riemann Problem pressure relaxation model



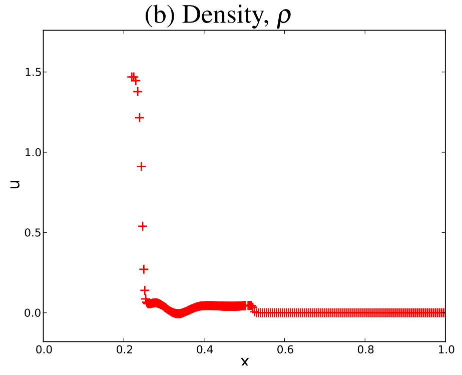
(a) Pressure, p



(b) Density, ρ



(c) Specific internal energy, e

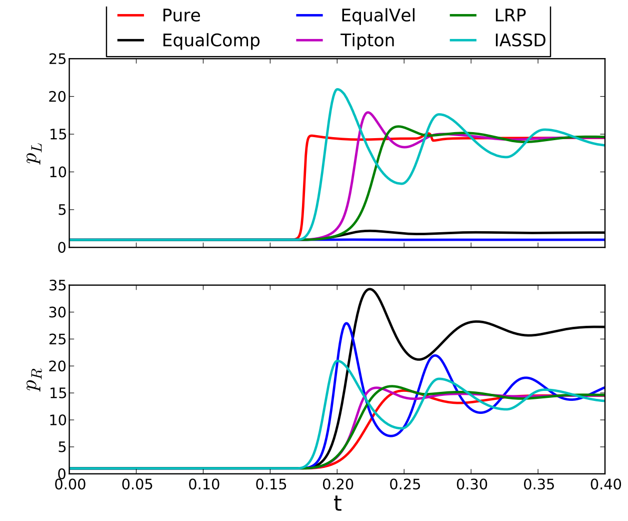


(d) Velocity, u

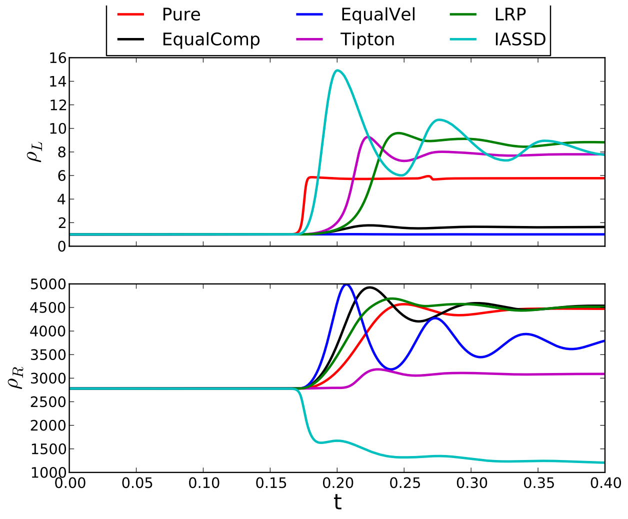
Figure 54: IASSD pressure relaxation model

Despite the lack of strength model needed to accurately model aluminium, the results suggest convincing outcome. Initially right-going Mach 2 shock wave collides with the multi-material cell. Then, some of the shock wave is transmitted into aluminium and the rest is reflected. It is difficult to exactly find what happened due to lack of the exact solution. However, this problem reaffirms our understandings of the types of closure models considered in this report.

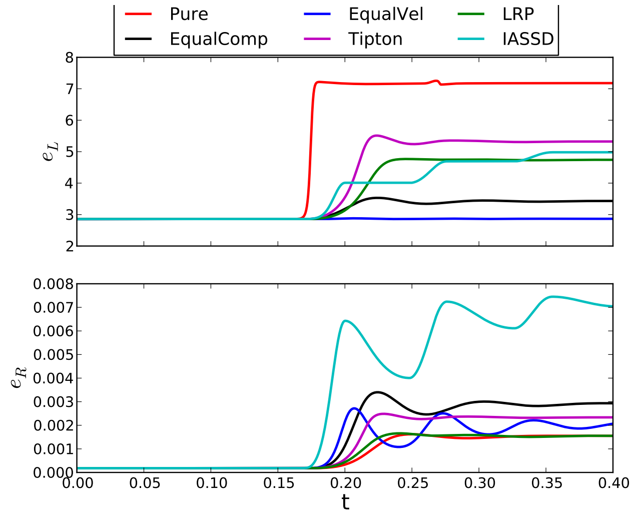
The results shown are at $t = 2.5$. By this time, the reflected shock is located at $x \approx 0.25$. The transmitted shock into aluminium is located at $x \approx 0.52$. The contact surface of two materials is also displaced slightly, now located at $x \approx 0.505$. This observation gives perspectives into the strength of reflected and transmitted shocks in different materials. In comparison, the contact surface location in the shock contact problem for gas-gas interaction is located at $x \approx 0.58$ at time $t = 0.25$. Noticeable change in the location of contact surface was observed after a long simulation run time.



(a) Pressure, p



(b) Density, ρ



(c) Specific internal energy, e

Figure 55: Time history of multi-material cell components for various closure models

In Figure 55, the multi-material closure model results are compared against the pure Lagrangian results in the multi-material cell. There is a large amount of oscillations in all properties in the multi-material cell. From this illustration, we can compare how the relaxation procedure varies from one model to another. IASSD model very quickly detects pressure variation, however, comes with very large oscillations. On the other extreme, the Linearized Riemann Problem closure model is slowest to detect the pressure variation, however, has the least amount of oscillations. Tipton's closure model lies somewhere in between. All these results seems to indicate the common fact that strength model for solids are required to counteract the oscillations produced by the closure models.

Compression Test—Sanity Check

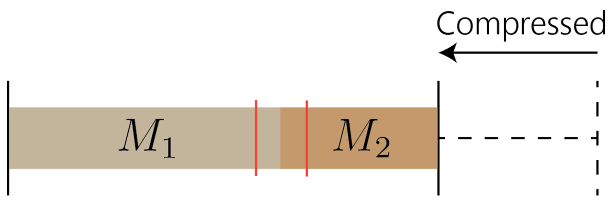


Figure 56: A cartoon illustration of a compression (shock unloading) test. Two materials occupy a domain with a multi-material cell (indicated by red lines) in the middle. The whole domain is initially compressed and is then allowed to release. (In the tests considered here, we use just one, multi-material cell as our domain.)

We are also interested in compression (shock unloading) tests, where a domain is compressed via a shock (instantaneously) to some higher pressure state and then allowed to release. When multi-material cells are present in the domain, the constituent materials interact with each other during the unloading process; therefore, the choice of closure model should affect the overall release path of the mixture. We show a cartoon of a compression problem in Figure 56.

To verify that our code and equations of state work for this type of example, we first run a "sanity check" with a pseudo–multi-material cell. Our example consists of a computational domain with a single cell. At rest, the cell is composed of two components of the same material, 2024 aluminum. The rest density of Al is taken to be $2.78\text{g}/\text{cm}^3$, and the rest pressure is zero. The rest volume of the computational cell is 1.0. Finally, we use a relative measure of specific internal energy for this test, where 0 indicates ambient conditions; hence, at rest, each component of Al has 0 specific internal energy.

We note that the equation of state used for aluminum is the same as for the gas-solid shock contact problem:

$$P = 0.7906X + 1.3250X^2 + 2.1300X^3 + 2.00\,(\rho e) \tag{116}$$

where 2.00 is the Grüneisen parameter Γ for the material and where $X = 1 - \rho_0/\rho$ (ρ_0 is the initial compressed density of the material).

We initialize the aluminum to a shocked state, and then track the release path of of the pressure as a function of relative volume for the constituent materials. The cell is shocked so that both components of Al have a pressure of 0.185Mbar. The density of each component therefore becomes $2.78/.8475 \approx 3.28\text{g}/\text{cm}^3$, and the specific internal energy of each component is approximately $3.97 \times 10^{-3}\text{Mbar cm}^3/\text{g}$.

We note that in this test, we use zero artificial viscosity. As discussed earlier in the paper, for boundary conditions, we fix the left endpoint of the domain and place a ghost cell to the right of our computational cell. The ghost cell is constantly at the (uncompressed) rest state of the aluminum.

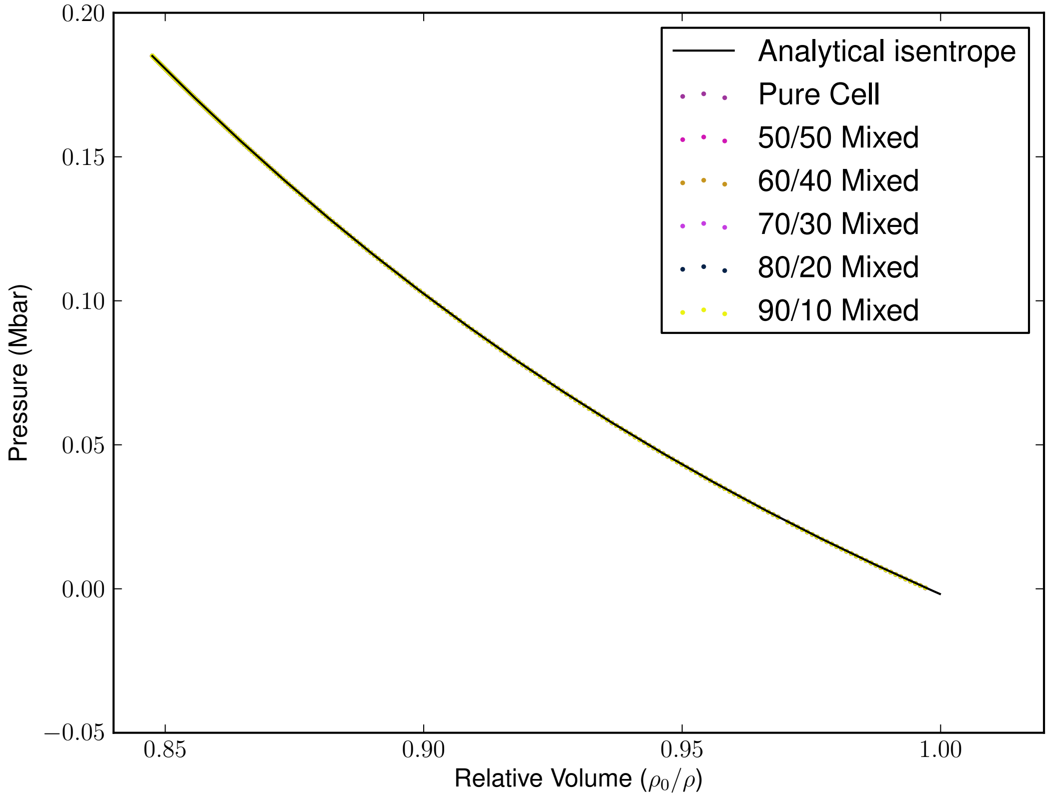


Figure 57: Results for the sanity check compression test. Shown is the average pressure in the cell (the mixture pressure) vs. the relative volume of the mixture (which should equilibrate to 1.0). Experimental data for different volume fractions of the two aluminum components are compared with the analytical isentrope for aluminum (given the initial conditions of our test). Our data match to the analytical solution to within 10×10^{-12}. The differences in the various volume fraction results are roughly on the order of machine precision (approximately 10×10^{-15}). The same results were obtained, up to roughly machine precision, regardless of the closure model used.

In Figure 57, we show our results for a variety of pseudo–multi-material cell configurations. For the two components of aluminum, we let them have volume fractions $(.5, .5)$, $(.6, .4)$, $(.7, .3)$, $(.8, .2)$, $(.9, .1)$, and $(1.0, 0)$. We compare these results to an analytical isentrope based on our initial conditions. We find essentially exact agreement between the theoretical values and our experimental results. The figure shows results for the equal compressibility model; however, all results were the same (within tolerance) regardless of which closure model we selected. There is no preferred closure model for this test; however, this test is intended as a validation of our code rather than as a comparison of closure models. As such, the results of this test are evidence that our host hydrocode, closure models, equations of state, and initial conditions are implemented correctly. A more practical compression test is discussed below.

Compression Test—Glass-Epoxy Shock Unloading

In this example, we compress a single-cell computational domain from volume 1 to volume .6 (see Figure 56 for a cartoon depiction of the compression problem). The cell, at its initial rest state, is composed of 40% glass and 60% 828 epoxy, whose equations of state are taken to be, respectively:

$$P = 0.8859X + 2.6040X^2 + 6.1980X^3 + 0.75\,(\rho e) \tag{117}$$

$$P = 0.0861X + 0.2114X^2 + 0.4421X^3 + 1.19\,(\rho e) \tag{118}$$

where 0.75 and 1.19 are the Grüneisen parameters Γ for the materials and where $X = 1 - \rho_0/\rho$ (ρ_0 are the initial compressed densities of the materials).

The rest densities of the glass and epoxy are $3.79\text{g}/\text{cm}^3$ and $1.19\text{g}/\text{cm}^3$, respectively. However, since glass and epoxy have different volumetric compressibilities, their densities, volume fractions, and other variables change disproportionately when the domain is compressed. In the compressed state (the initial conditions used for the simulation), glass and epoxy have volume fractions 0.56 and 0.44, densities $4.51\text{g}/\text{cm}^3$ and $2.70\text{g}/\text{cm}^3$, pressures 0.31Mbar and 0.31Mbar, and specific internal energies $\approx 2.25 \times 10^{-2}\text{Mbar cm}^3/\text{g}$ and $\approx 3.86 \times 10^{-2}\text{Mbar cm}^3/\text{g}$, respectively.

The same types of artificial viscosity and boundary conditions are used as in the sanity check compression test above: namely, there is zero artificial viscosity, the left endpoint of the cell is fixed, and there is a ghost cell to the right of the domain that is constantly maintained to be the uncompressed rest state of the mixture.

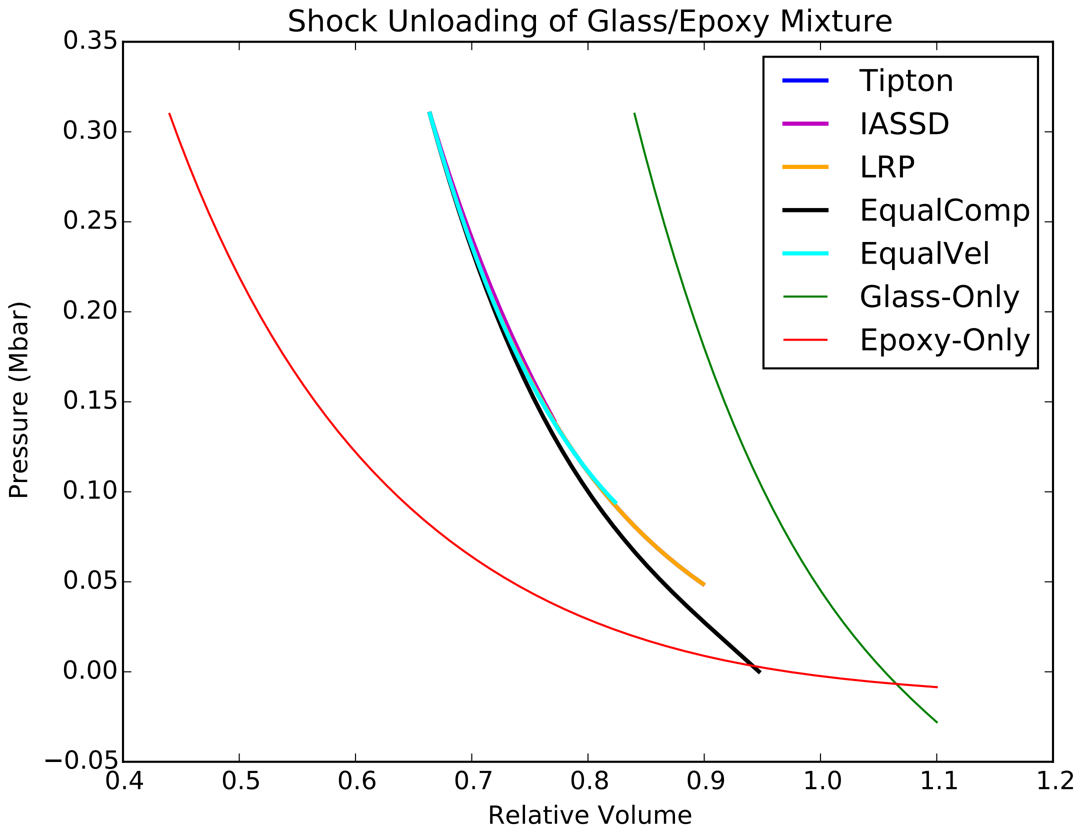


Figure 58: Results for the glass-epoxy shock unloading test. The pressure for the mixture in the multi-material cell (a 60-40 combination of epoxy and glass) is plotted against the relative volume of the computational cell. The red and green curves indicate the unloading curves if the cell only contained glass or epoxy; our results, using a combination of glass and epoxy, fall between these two curves. The simulation stops printing results when one of the constituent pressures becomes negative. In the figure, the results for Tipton's model are obscured directly behind the LRP model results.

Figure 58 shows the results of the test for various multi-material closure models. The pressure of the mixture is plotted against the relative volume of the mixture (the rest state of the mixture is at the point $(1,0)$). We expect that the pressures of each constituent should monotonically decrease from 0.31 to 0; therefore, we stop the simulation if either of the constituent material pressures becomes negative (hence, some of the result curves are longer than others). The red and green lines in the graph show, respectively, the isentropic unloading curves for an equivalent cell made of just the epoxy or just the glass material. We expect that since our cell contains a mixture of glass and epoxy, any plausible results for our mixture must fall between the red and green curves. Indeed, this is what we observe, except for the tail end of the results for the equal compressibility model.

The various closure models we tested demonstrate interesting behavior for this test problem. For the equal compressibility model, the constituent pressures monotonically decrease to

0 as expected, but the mixture equilibrates at a relative volume of less than 1. For the other closure models, either the glass or epoxy component attains negative pressure before the mixture as a whole equilibrates to zero pressure, which is an unexpected result, although a negative pressure by itself is not necessarily nonphysical. We note that the equal velocity increment model stops quickly relative to the other models, and as such, it is not a particularly appropriate model for this test. The LRP model, Tipton's model (almost exactly matching the LRP results), and IASSD all perform similarly. Even though these models achieve a negative pressure for a component earlier than equal compressibility, it appears that these models are on a closer trajectory for the mixture to reach $(1,0)$ than equal compressibility. Since this test problem is very sensitive to pressure, it makes sense that these three pressure-relaxation–based models all outperform the one-step models when it comes to better capturing the mixture pressure; however, further investigation is warranted into the trade-off between accurate mixture pressure and accurate component pressure that is exhibited by the varying model results.

Moving Plate Problem

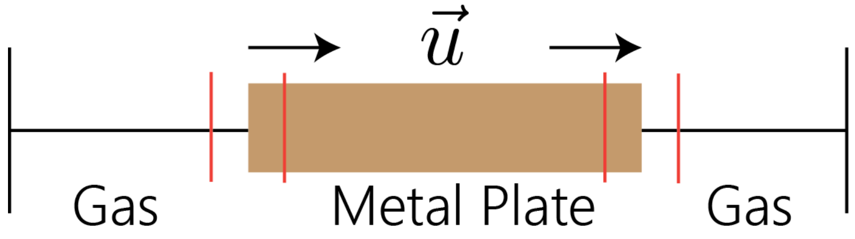


Figure 59: The setup for the moving plate test problem. Multi-material cells are indicated by red lines.

Another test we consider involves a solid plate flying through the computational domain. We initialize a domain with a solid plate surrounded on either side by a gas. The plate has a positive velocity relative to the velocity of the gas. There is a multi-material cell at each solid-gas interface. A cartoon of the setup for this problem is shown in Figure 59.

The test is run with 100 grid cells, a CFL number of 0.1, and a final time of 0.5. In both multi-material cells, the volume fractions are all 0.5. The metal plate is made out of 2024 aluminum, using the same cubic EOS seen earlier in the results (Equation (116)).

The initial thermodynamic state of the problem is:

$$(\gamma,\rho,e,p,u) = \begin{cases} (1.35, 1.0,\ 2.86, 1.0, 0.095) & \text{if} \quad 0 \le x \le 0.3 \\ (\text{N/A}, 2.78\times 10^3,\ 1.80\times 10^{-4}, 1.0, 0.1) & \text{if} \quad 0.3+\Delta x \le x \le 0.7 \\ (1.35, 1.0,\ 2.86, 1.0, 0.095) & \text{if} \quad 0.7+\Delta x \le x \le 1.0. \end{cases} \tag{119}$$

Note that the density used for aluminum is in kg/m^3 as opposed to g/cm^3 (the density units used in the compression tests).

The initial configuration of the problem is shown in Figure 60. Figures 61 to 65 show the results of the simulation for the various closure models we considered. All results shown are at the final time. The black solid lines indicate the pure Lagrangian solutions. The symbols ◀ and ▶ indicate left and right component values in multi-material cell.

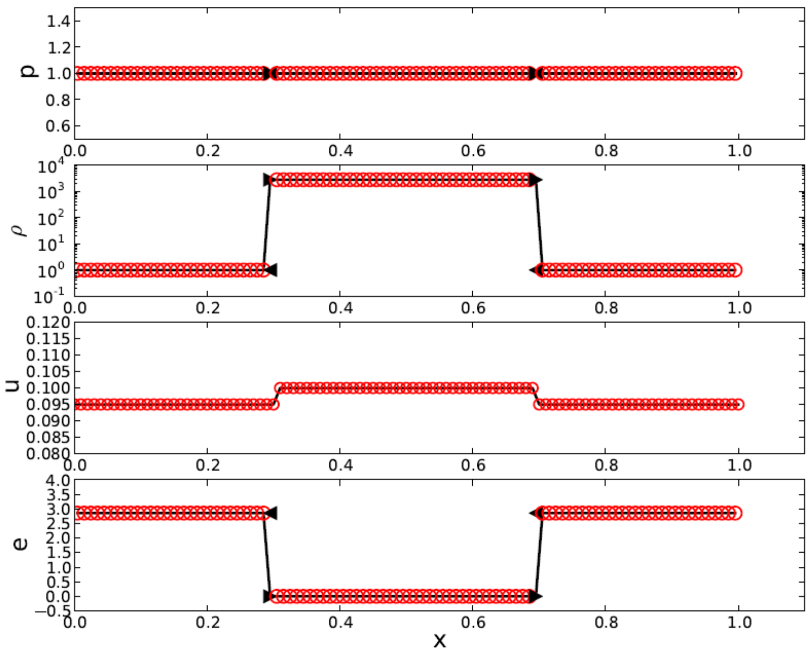


Figure 60: The initial state of the moving plate problem

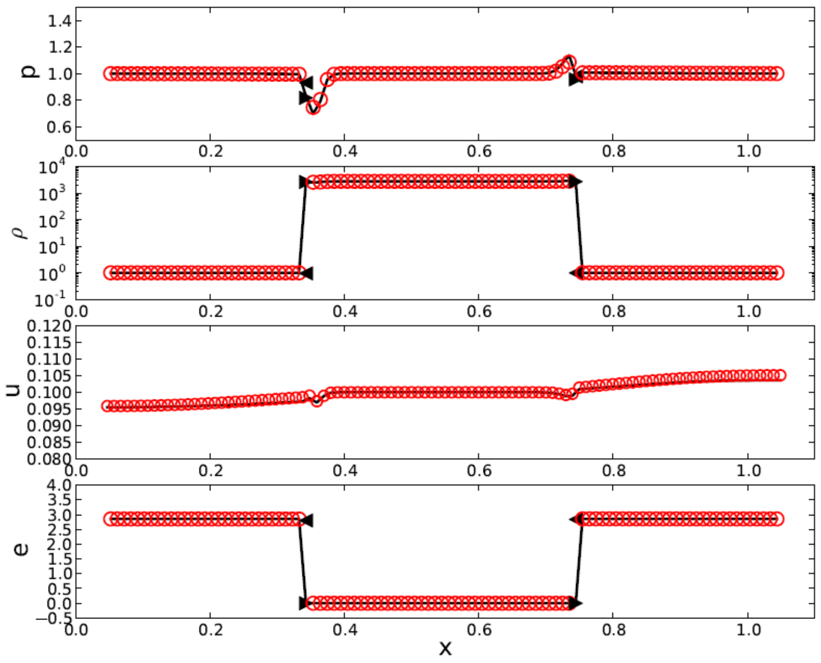


Figure 61: The final state of the moving plate problem for equal compressibility model

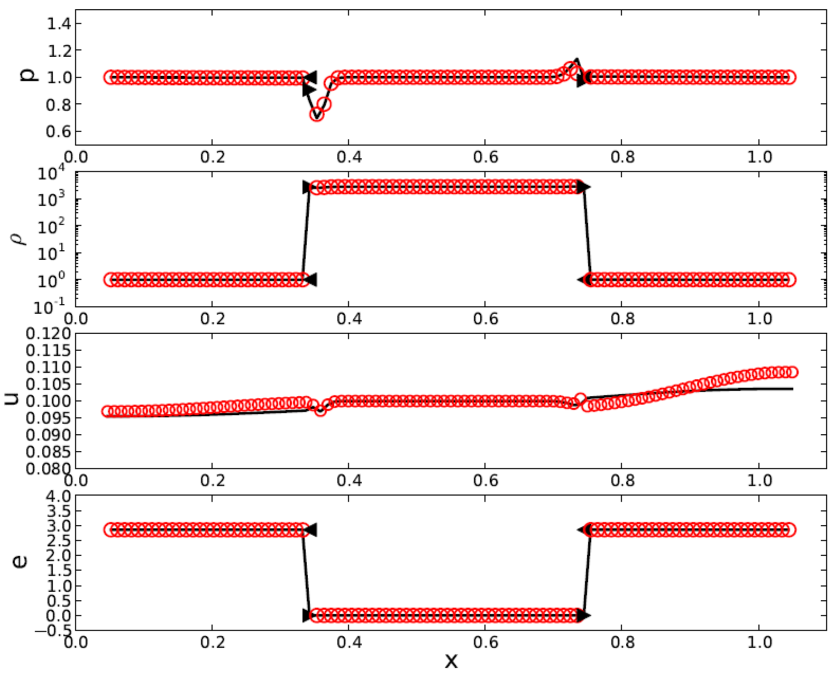


Figure 62: The final state of the moving plate problem for equal velocity increment model

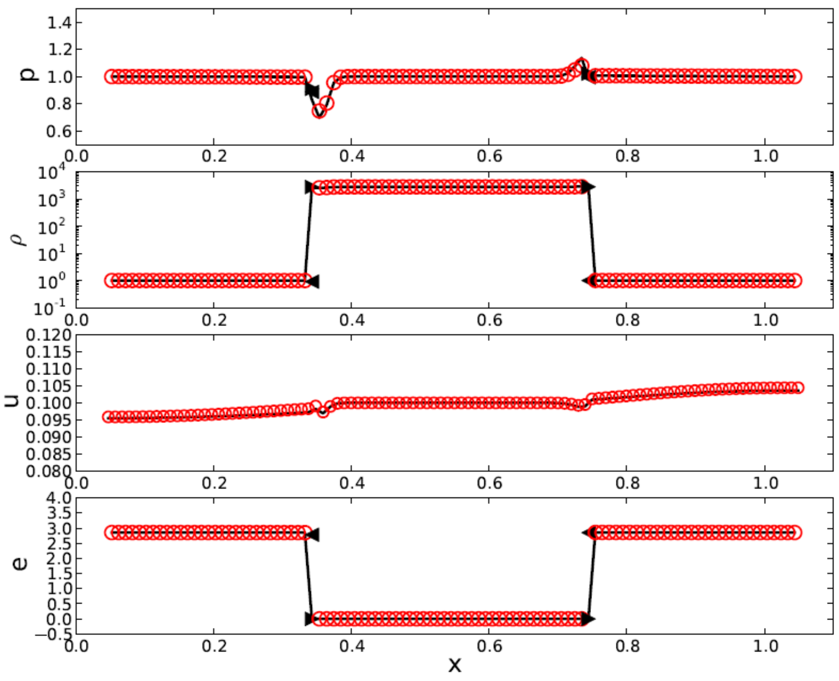


Figure 63: The final state of the moving plate problem for Tipton's model

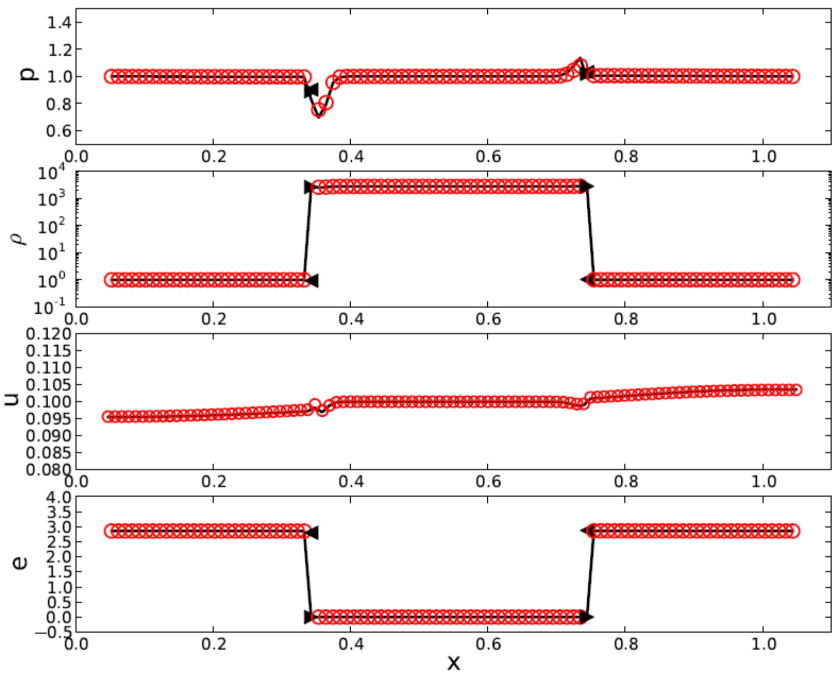


Figure 64: The final state of the moving plate problem for LRP model

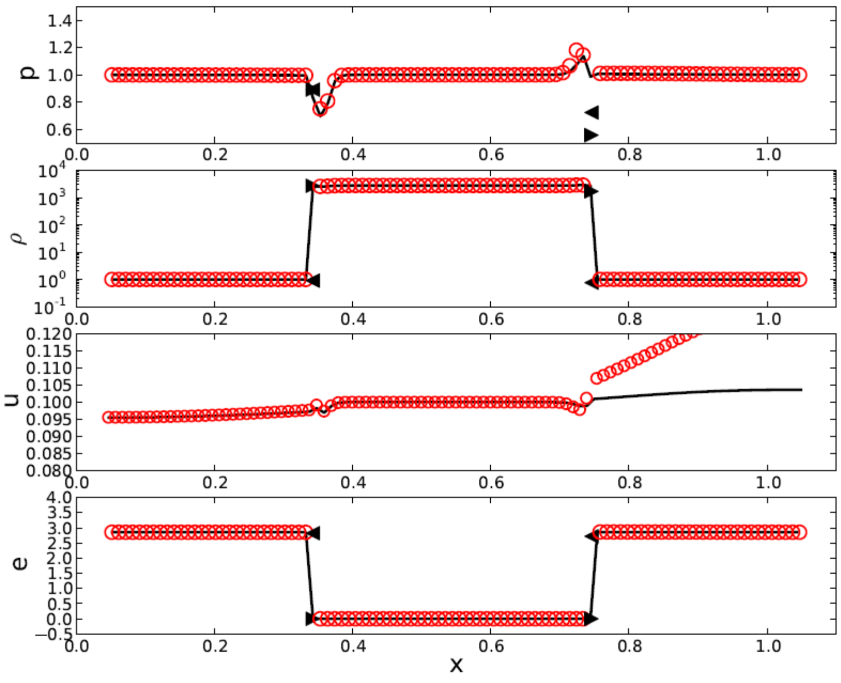


Figure 65: The final state of the moving plate problem for IASSD model

We make several observations about the results. The plate begins to move faster than the surrounding gas. This velocity differential between the plate and surrounding gas allows for the creation of right-going waves at the tip of the plate and left-going waves at the trailing edge of the plate. The presence of the waves are reflected on the velocity profiles at the final time step.

The one-step closure model results, especially in velocity, deviate from the pure Lagrangian results. However, pressure, density and specific internal energy values are well predicted as compared to the pure Lagrangian results.

The pressure relaxation model velocity profiles are well predicted. Tipton's closure model and Linearized Riemann Problem model results are very similar for all variables. The only difference was noticed in IASSD model where the leading edge pressure and velocity profiles are greatly mishandled. However, the trailing edge quantities are evaluated quite well.

Summary

The one-step closure models such as the equal compressibility model and the equal velocity increment model are very easy models to implement due to the simple assumptions made throughout the closure model procedure. As we have observed in many test problems, however, the one-step closure models do not give very reliable results of the multi-material component properties. As a result, they may not be the most attractive closure models to use in practical situations where certain physical properties must be considered. For instance, the equal compressibility assumption is not favorable when the compressibility properties of components differ significantly. The linearly varying velocity assumption made in the equal velocity increment model is simple, yet, may not provide accurate enough representation of the physical problem. Despite the inadequate results produced by the one-step closure models, these closure models theoretical basis and understanding for building future multi-material closure models.

Tipton's pressure relaxation closure model is one of the widely used multi-material closure models in production hydrocodes such as FLAG production code in Los Alamos National Laboratory. It is a pressure relaxation closure model which doesn't require the material interface location or the orientation. For this reason, it is a relatively simple method and can be extended to multiple material configurations with relative ease. Linearized Riemann Problem pressure relaxation model and IASSD model, on the other hand, require explicit information of the interface location as well as the material orientation. This is achieved by a more physically and mathematically sound concept of the Riemann problem. Solving the Riemann problem accurately represents the physical process of transport and evaluation of the material specific thermodynamic quantities. However, these sophisticated methods that require additional information to accurately determine thermodynamic quantities come with an increased computational cost. As observed in many results, the Riemann problem based pressure relaxation models are not considered superior to Tipton's closure model.

Some Recommendations of Multi-Material Closure Models

For many practical situations, the one-step closure models are not wise choices due to their unreliable performances. All pressure relaxation closure models, however, are better suited for more complex problems as shown in various test problems considered in this project.

Although Tipton's pressure relaxation closure model has been proven to be useful in various production hydrocode and hydrodynamics community, it lacks some sophistication that the modern multi-material closures are equipped with. However, Tipton's pressure relaxation closure model is versatile. It does not require the interface reconstruction step, which would normally be needed in the Riemann problem based closure models. This eliminates the need for an extra routine in the hydrocode if one were to develop a hydrocode from scratch. As problems get more complex, there is an essential need for implementing a more accurate multi-material closure models. It would be a great investment to utilize the modern multi-material closure models because they provide insights that are based on more accurate physics.

Conclusion and Future Work

Numerous test problems reveal advantages and disadvantages of each closure model choice in different circumstances. It is a daunting task to catalogue as many test problems as we have considered in this report. However, it becomes obvious that there are countless list of problems to be considered in order to even come close to understanding a real world problem. The aim was not set to understand the entire physics, but rather have some educated understanding of the closure models and their designs with respect to the physical problems they are being tested on. We provided various array of test problems which range from simple sanity test to more complex multi-phase flows.

This research was also meant for the pilot study for the future development in the multi-material closure models. The SGH hydrocode developed and used in this project was only limited to one dimensional problems in order to maximize time spent on investigating various multi-material closure models. However, this limitation of the project has only a slight impact on the final outcome given the fact that it is meant to give insights into the field of multi-material closure modelling.

There are still some work needed to make the project more complete. First, a strength mode needs to be implemented for proper modelling of solid materials. Even though the preliminary results indicate the solid material can be simulated without the strength model, one must consider more complete representation of solid materials.

Second, there are only limited number of exact solutions shown in the results. For some of the problems, the multi-material closure model results were only compared with the pure Lagrangian results. In order to properly assess the closure model results, the exact solutions must be included.

Third, higher-dimensional hydrodynamics code must be considered. One dimensional hydrodynamics code is a useful tool in development of the closure models. One dimensional mathematical formulations lead to the simplest proving ground for any theoretical development. There is, however, a large gap between theory and practice. Often times one dimensional theories are difficult to put into practice for higher-dimensional problems because of the inherent differences in physics and mathematics.

Lastly, as it was obvious in multiple test problems considered in this project, most of the pressure relaxation multi-material closure models show promising performance in computing the correct common pressures. However, the specific internal energy and density values were often times not well approximated, or even fall within the bounds of the immediate neighboring values. This is understandable since all the multi-material closure models considered in this project are independent of the neighboring material property. More importantly, none of the multi-material closure models disallow the conservation laws of the fluid dynamics. It seems, however, unsatisfactory for any multi-material closure models to have varying opinions on the specific internal energy and density computation values. Especially, this would mean inaccurate computation of component material properties which may be essential to capturing the inter-material dynamics. Future development of new class of closure models should consider the possibility of possessing some suggestions regarding more realistic and bounded material properties in the multi-material cell for more realistic physics.

References

[1] Andrew Barlow, Ryan N. Hill, and Mikhail J. Shashkov. Constrained optimization framework for interface-aware sub-scale dynamics closure models for multimaterial cells in Lagrangian and arbitrary Lagrangian-Eulerian hydrodynamics. Technical report LA-UR-13-26180, Los Alamos National Laboratory, August 2013.

[2] David J. Benson. Computational methods in Lagrangian and Eulerian hydrocodes. *Computer Methods in Applied Mechanics and Engineering*, 99(2-3):235–394, September 1992.

[3] E. J. Caramana, D. E. Burton, M. J. Shashkov, and P. P. Whalen. The construction of compatible hydrodynamics algorithms utilizing conservation of total energy. *Journal of Computational Physics*, 146:227–262, 1998.

[4] Rod W. Douglass and Alan K. Stagg. A vertex-staggered hydrodynamics model for compressible flows. Technical report LA-UR-07-6986, Los Alamos National Laboratory, 2007.

[5] Wallace E. Johnson and Charles E. Anderson Jr. History and application of hydrocodes in hypervelocity impact. *International Journal of Impact Engineering*, 5(1-4):423–439, 1987.

[6] Charles E. Anderson Jr. An overview of the theory of hydrocodes. *International Journal of Impact Engineering*, 5(1-4):33–59, 1987.

[7] J. R. Kamm, M. J. Shashkov, J. Fung, A. K. Harrison, and T. R. Canfield. A comparative study of various pressure relaxation closure models for one-dimensional two-material Lagrangian hydrodynamics. *International Journal for Numerical Methods in Fluids*, 65:1311–1324, 2011.

[8] J. R. Kamm, M. J. Shashkov, and W. J. Rider. A new pressure relaxation closure model for one-dimensional two-material lagrangian hydrodynamics. *EPJ Web of Conferences*, 10, 2010.

[9] James R. Kamm and Mikhail J. Shashkov. A pressure relaxation closure model for one-dimensional, two-material lagrangian hydrodynamics based on the Riemann Problem. *Communication of Computational Physics*, 7(5):927–976, May 2010.

[10] J. von Nuemann and R. D. Richtmyer. A method for the numerical calculation of hydrodynamic shocks. *Journal of Applied Physics*, 21:232–237, 1950.

[11] Yury V. Yanilkin, Evgeny A. Goncharov, Vadim Yu. Kolobyanin, Vitaly V. Sadchikov, James R. Kamm, Mikhail J. Shashkov, and William J. Rider. Multi-material pressure relaxation methods for Lagrangian hydrodynamics. *Computers & Fluids*, 83:137–143, 2013.

Warm Dense Matter Calculations using OFMD

Team Members
Tyson Loudon and Jinhie Skarda

Mentors
Ondrej Certik

Abstract

Orbital-free molecular dynamics is used to perform calculations for aluminum in the warm dense matter regime. The dynamic structure factor obtained from the orbital-free calculations are compared with the dynamic structure factor obtained from pseudo-atom molecular dynamics calculations. These comparisons are made at 5, 20, and 50 eV for wavenumbers that span the single particle, generalized hydrodynamic, and hydrodynamic regimes. While the dynamic structure factors for warm dense aluminum from the pseudo-atom molecular dynamics framework have previously been compared with those from the orbital-free framework, these comparisons could only be performed up to 5 eV due to limitations in the pseudopotential used in the orbital free calculations. The orbital-free calculations performed in this project, however, self consistently calculate the pseudopotential and are valid at the higher temperatures of 20 and 50 eV. Good agreement is found between the pseudo-atom molecular dynamics and orbital-free dynamic structure factors in the single particle regime at all three temperatures. In the generalized hydrodynamic and hydrodynamic regimes the ion-acoustic resonance for the orbital-free calculation is slightly higher and appears at a lower energy than that of the pseudo-atom molecular dynamics calculations, but the shape of the curves matches well.

Introduction

In this paper we will be modeling warm dense matter using orbital-free density functional theory. Warm dense matter is an intermediate state of matter between a solid and a plasma. It exists in the lower-temperature portion of the high energy density regime, under conditions where the assumptions of both condensed matter theory and ideal-plasma theory break down, and where quantum mechanics, particle correlations, and electric forces are all important [2]. Warm dense matter occurs in many instances including in the core of Jovian planets, in inertial confinement fusion, and in nuclear explosions.

There are several methods by which to model warm dense matter computationally. Born-Oppenheimer molecular dynamics is a computationally efficient way to model WDM. Born-Oppenheimer molecular dynamics uses classical mechanics to propagate ion trajectories and quantum mechanics to model the electrons. The two main methods which use the Born-Oppenheimer molecular dynamics are an orbital method called Kohn-Sham density functional theory and orbital free density functional theory. Kohn-Sham density functional theory is very accurate, but is computationally expensive. Since the number of orbitals grows with temperature, high temperature Kohn-Sham calculations are not computationally feasible. Orbital free methods, on the other hand, are still accurate, but not as computationally expensive and scale well with temperature.

Recently, another molecular dynamics framework called pseudo-atom molecular dynamics, was developed and described in [3]. Pseudo-atom molecular dynamics is classical molecular dynamics with a pairwise potential and is thus extremely fast and scaleable to higher temperatures. The accuracy of the model was assessed in [3] by comparing the dynamic structure factor, a parameter that describes the temporal and spatial correlations in a system, obtained from pseudo-atom molecular dynamics calculations with the dynamic structure factor obtained from the orbital free calculations in [8]. However, these comparisons could only be made up to 5 eV because the pseudo-potential used in the orbital free calculations in [8] was only good up to about 5 eV. Since our orbital free code uses a pseudopotential that is good to higher temperatures, we take advantage of the good temperature scaling of orbital free methods in order to compute the dynamic structure factor at 5, 20, and 50 eV and compare these results to those of the pseudo-atom molecular dynamics calculations.

In this report, we first describe orbital free density functional theory in more detail. We then present our convergence studies and results.

Orbital Free Density Functional Theory

The orbital-free electronic free energy, F_e, in the WDM is given by

$$F_e[n_e] = T[n_e] + U_{en}[n_e] + U_{ee}[n_e] + F_{xc}[n_e] \tag{1}$$

where $T[n_e]$ is the kinetic energy, $U_{en}[n_e]$ is the electron-nuclei energy, $U_{ee}[n_e]$ is the Hartree electron-electron energy, and $F_{xc}[n_e]$ is the exchange-correlation energy. Each of these four terms are functionals of the electron density n_e. We will want to solve (1) for n_e by minimizing the orbital-free electronic free energy functional. Throught this paper we will be using Hartree atomic units which sets the numerical value of the electron mass, m_e, the elementary charge, e, the reduced Planck's constant, $\hbar$, and Coulomb's constant, k_e, equal to unity. Each of the terms in (1) is discussed individually below.

Kinetic Energy

The orbital-free approximation to the kinetic energy functional is given by

$$T_e[n_e] = E_{\text{chem}} - \frac{2}{3}E_{\text{kin}} \tag{2}$$

where E_{chem} denotes the chemical energy and E_{kin} is the Thomas-Fermi-Dirac kinetic energy. From [1] and [6] we can express the the Thomas-Fermi-Dirac kinetic energy of the electrons at finite temperature as

$$E_{\text{kin}} = \frac{1}{2\pi^2}\int_{\Omega}\int_{0}^{\infty}\frac{p^4}{\mathrm{e}^{\beta(E-\mu)}+1}\,\mathrm{d}p\,\mathrm{d}\mathbf{x} \tag{3}$$

where μ is the chemical potential, p is the electron momentum, $\beta = (k_{\text{B}}T)^{-1}$ where k_{B} is Boltzmann's constant and T is the absolute temperature. Making the change of variables $u = \frac{p^2}{2}$ gives

$$E_{\text{kin}} = \frac{\sqrt{2}}{\pi^2}\int_{\Omega}\int_{0}^{\infty}\frac{u^{\frac{3}{2}}}{\mathrm{e}^{\beta(E-\mu)}+1}\,\mathrm{d}u\,\mathrm{d}\mathbf{x}\,. \tag{4}$$

Lastly, using the fact that $E = \frac{p^2}{2} + V(\mathbf{x})$ and making another change of variables $y = \beta u$ gives

$$E_{\text{kin}} = \frac{\sqrt{2}}{\pi^2\beta^{\frac{5}{2}}}\int_{\Omega} I_{\frac{3}{2}}(\eta(\mathbf{x}))\,\mathrm{d}\mathbf{x} \tag{5}$$

where $\eta(\mathbf{x}) = \beta(\mu - V(\mathbf{x}))$ and

$$I_{\alpha}(z) = \int_{0}^{\infty}\frac{t^{\alpha}}{\mathrm{e}^{t-z}+1}\,\mathrm{dt} \tag{6}$$

is called the Fermi-Dirac integral of order α. Now, with the Fermi-Dirac integral defined we can express E_{chem} as

$$E_{\text{chem}} = \frac{\sqrt{2}}{\pi^2\beta^{\frac{5}{2}}}\int_{\Omega} I_{\frac{1}{2}}(\eta(\mathbf{x}))\eta(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{7}$$

Lastly, we can use equations (5) and (7) to write the final expression for the kinetic energy as

$$T_e[n_e] = \frac{1}{\beta}\int_{\Omega} n_e(\mathbf{x})f(y(\mathbf{x}))\,\mathrm{d}\mathbf{x} \tag{8}$$

where $y(\mathbf{x}) = \frac{\pi^2}{\sqrt{2}}\beta^{\frac{3}{2}}n_e(\mathbf{x})$ and f(y) is a special function of one variable composed of a Fermi-Dirac integral of order $\frac{3}{2}$ and its inverse of order $\frac{1}{2}$:

$$f(y) = I_{\frac{1}{2}}^{-1}(y) - \frac{2}{3y}I_{\frac{3}{2}}\left(I_{\frac{1}{2}}^{-1}(y)\right). \tag{9}$$

Electron-Nuclei Energy

The electron-nuclei energy functional is given by

$$\begin{aligned} U_{en}[n_e] &= \iint \frac{n_e(\mathbf{x})n_n(\mathbf{x}')}{|\mathbf{x}-\mathbf{x}'|}\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{x}' \\ &= \int_{\Omega} n_e(\mathbf{x})V_{en}(\mathbf{x})\,\mathrm{d}\mathbf{x} \end{aligned} \tag{10}$$

where

$$V_{en}(\mathbf{x}) = \int_{\Omega} \frac{n_n(\mathbf{x}')}{|\mathbf{x}-\mathbf{x}'|}\,\mathrm{d}\mathbf{x}' \tag{11}$$

is the potential energy which describes the Coulomb interaction between an electron and the collection of atomic nuclei in the WDM. In equations (10) and (11) $n_n(\mathbf{x})$ is the nuclear charge density.

In order to evaluate the electron-nuclei energy functional (10) we need to know the Coulomb potential for the interaction between the electron and nuclei V_{en}. We can express (11) in another way by taking the laplacian of both sides of (11) and using the relations

$$\begin{aligned} \nabla^2\left(\frac{1}{|\mathbf{x}-\mathbf{x}'|}\right) &= -4\pi\delta(\mathbf{x}-\mathbf{x}') \\ n_n(\mathbf{x}) &= \int_{\Omega} n_n(\mathbf{x}')\delta(\mathbf{x}-\mathbf{x}')\,\mathrm{d}\mathbf{x}', \end{aligned}$$

where δ represents the Dirac delta function, to get the Poisson equation

$$\nabla^2 V_{en} = -4\pi n_n. \tag{12}$$

Thus, in order to evaluate (10) we will first need to solve (12) for V_{en}.

Hartree Energy

The Hartree energy functional, which describes the Coulomb interaction between electrons, is given by

$$\begin{aligned} U_{ee}[n] &= \frac{1}{2}\iint \frac{n_e(\mathbf{x})n_e(\mathbf{x}')}{|\mathbf{x}-\mathbf{x}'|}\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{x}' \\ &= \frac{1}{2}\int_{\Omega} n_e(\mathbf{x})V_{ee}(\mathbf{x})\,\mathrm{d}\mathbf{x} \end{aligned} \tag{13}$$

where

$$V_{ee}(\mathbf{x}) = \int_{\Omega} \frac{n_e(\mathbf{x}')}{|\mathbf{x}-\mathbf{x}'|}\,\mathrm{d}\mathbf{x}' \tag{14}$$

is the Hartree potential. This potential describes the Coulomb repulsion between an electron and the total electron density defined by all electrons in the WDM. In a similar manner to the previous section we can take the laplacian of both sides of (14) to get the Poisson equation

$$\nabla^2 V_{ee} = -4\pi n_e. \tag{15}$$

Thus, in order to evaluate (13) we first need to solve (15) for V_{ee}.

Exchange-Correlation Energy

The exact expression for F_{xc} is currently unknown. This term is defined to include all the quantum mechanical effects that are not included in the previously defined functionals. In this paper we will use the Perdew-Zunger local density approximation (LDA) to estimate the exchange-correlation energy. The Perdew-Zunger LDA is given by

$$F_{xc}[n_e] = \int_{\Omega} n_e(\mathbf{x})\varepsilon_{xc}^{LD}(n_e)\,\mathrm{d}\mathbf{x} \tag{16}$$

where $\varepsilon_{xc}^{LD}(n_e)$ is the energy density of a homogenous electron gas. We then decompose the energy density into two components

$$\varepsilon_{xc}^{LD}(n_e) = \varepsilon_{x}^{LD}(n_e) + \varepsilon_{c}^{LD}(n_e). \tag{17}$$

where $\varepsilon_{x}^{LD}(n_e)$ represents the electron gas exchange energy density and $\varepsilon_{c}^{LD}(n_e)$ represents the electron gas correlation energy density. The electron gas exchange energy density can be computed exactly and is given by

$$\varepsilon_{x}^{LD}(n_e) = -\frac{3}{4}\left(\frac{3}{\pi}n_e\right)^{\frac{1}{3}}. \tag{18}$$

However, an exact expression for the electron gas correlation energy density does not exist. But, an accurate fit was obtained by Vosko, Wilk, and Nusair in [7] and is given by

$$\begin{aligned}\varepsilon_{c}^{LD}(n_e) = \frac{A}{2}\Bigg[&\log\left(\frac{y^2}{Y(y)}\right) + \frac{2b}{Q}\arctan\left(\frac{Q}{2y+b}\right)\\ &-\frac{by_o}{Y(y_o)}\left(\log\left(\frac{(y-y_o)^2}{Y(y)}\right) + \frac{2(b+2y_o)}{Q}\arctan\left(\frac{Q}{2y+b}\right)\right)\Bigg]\end{aligned} \tag{19}$$

where $y=\sqrt{r_s}$, $Y(y)=y^2+by+c$, $Q=\sqrt{4c-b^2}$, $y_o=-0.10498$, $b=3.72744$, $c=12.9352$, $A=0.0621814$, and r_s is the mean distance between electrons

$$r_s = \left(\frac{3}{4\pi n_e}\right)^{1/3}.$$

Free Energy Minimization

In this section, we discuss the theory of minimizing the free energy functional. We will minimize the free energy in order to obtain the true ground state electron density guaranteed by the second Hohenberg-Kohn theorem. In this section we will adopt the notation $n(\mathbf{x}) \equiv n_e(\mathbf{x})$.

Calculus of Variations

We solve for the electron density, n, by minimize the free energy functional (1) subject to the constraint of electron conservation. The electron conservation constraint is given by

$$N = \int_{\Omega} n(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

where N is the total number of electrons. Define the functional

$$G[n] = \int_{\Omega} n(\mathbf{x})\,\mathrm{d}\mathbf{x} - N = 0 \tag{20}$$

to be the constraint for our minimization problem. Therefore, we want to minimize (1) subject to the constraint (20). This constrained optimization problem can be done by the method of Lagrange multipliers which gives

$$\frac{\delta F[n]}{\delta n} = \varepsilon \frac{\delta G[n]}{\delta n} \tag{21}$$

where ε is the Lagrange multiplier. Using the fact that the functional derivative of $G[n]$ is one, equation (21) becomes

$$\frac{\delta F[n]}{\delta n} = \varepsilon. \tag{22}$$

By solving the functional (22) we will recover the true electron density guaranteed by the second Hohenberg-Kohn Theorem. Using the expression for the orbital-free electron free energy given in (1) we need

$$\frac{\delta T[n]}{\delta n} + \frac{\delta U_{en}[n]}{\delta n} + \frac{\delta U_{ee}[n]}{\delta n} + \frac{\delta F_{xc}[n]}{\delta n} = \varepsilon \tag{23}$$

The functional derivatives of the four terms on the left hand side of (23) are given by

$$\begin{aligned}
\frac{\delta T[n]}{\delta n} &= \frac{1}{\beta}\left(f(y) + \frac{\pi^2}{\sqrt{2}}\beta^{\frac{3}{2}} f'(y) n(\mathbf{x})\right) \\
\frac{\delta U_{en}[n]}{\delta n} &= V_{en} \\
\frac{\delta U_{ee}[n]}{\delta n} &= V_{ee} \\
\frac{\delta F_{xc}[n]}{\delta n} &= \varepsilon_{xc}^{LD}(n) + n(\mathbf{x}) \frac{\mathrm{d}\varepsilon_{xc}^{LD}(n)}{\mathrm{d}n}.
\end{aligned} \tag{24}$$

Setting $\frac{\delta F_{xc}[n]}{\delta n} = V_{xc}$ and defining

$$H[n] = \frac{1}{\beta}\left(f(y) + \frac{\pi^2}{\sqrt{2}}\beta^{\frac{3}{2}}\right) + V_{ee} + V_{en} + V_{xc} \tag{25}$$

the goal becomes to solve (25) subject to the constraint (20). It can be shown that this is equivalent to minimizing the functional

$$W[n] = F_e[n] - \varepsilon G[n] \tag{26}$$

which we will accomplish in the next section via a conjugate gradient algorithm.

Conjugate Gradient Method

We will use the conjugate gradient method outlined in [5] in order to minimize the functional defined in (26). But, instead of minimizing the energy with respect to the electron density we will introduce a quantity, $\psi(\mathbf{x})$, given by

$$n(\mathbf{x}) = \psi(\mathbf{x})^2 \tag{27}$$

which can be regarded as an artificial orbital. The reason for introducing the artificial orbital is that the requirement that $n(\mathbf{x})$ be positive can be onerous in numerical implementations, but is trivial in the case of this new formulation. We will use tildes henceforth to denote dependence on ψ, e.g. $W[n] = W[\psi^2] = \tilde{W}[\psi]$. Now, we will describe the method used to minimize $\tilde{W}[\psi]$ iteratively.

As is typical in the nonlinear conjugate gradient method we first want to calculate the steepest descent vector. If we were minimizing a typical function $f(\mathbf{x})$, the direction of steepest descent would be given by $-\nabla f(\mathbf{x})$. But, since we are minimizing a functional, the steepest descent vector is given by $-\frac{\delta \tilde{W}}{\delta \psi}$. This is calculated to be

$$-\frac{\delta \tilde{W}}{\delta \psi} = 2(\varepsilon - \tilde{H}[\psi])\psi \tag{28}$$

So, if we define χ_k to be the steepest descent vector at iteration k we have

$$\chi_k = 2(\varepsilon - \tilde{H}[\psi_k])\psi_k. \tag{29}$$

Note that we need to have an initial guess for ψ before beginning this method. Next, the conjugate gradient vector, φ_k, is calculated as

$$\varphi_k = \chi_k + \frac{\langle \chi_k, \chi_k \rangle}{\langle \chi_{k-1}, \chi_{k-1} \rangle} \varphi_{k-1} \tag{30}$$

where $\langle \cdot, \cdot \rangle$ denotes the standard inner product on $L^2(\Omega)$. Then, to satisfy the normalization constraint (20), the conjugate gradient function is further orthogonalized to ψ_k and normalized to N by setting

$$\begin{aligned} \varphi'_k &= \left(1 - \frac{1}{N^2}\psi_k^2\right)\varphi_k \\ \varphi''_k &= \sqrt{\frac{N}{\langle \varphi'_k, \varphi'_k \rangle}}\,\varphi'_k. \end{aligned} \tag{31}$$

This is done so that $\langle \varphi''_k, \psi_k \rangle = 0$ and $\langle \varphi''_k, \varphi''_k \rangle = N$. Lastly, ψ is updated as

$$\psi_{k+1} = \psi_k \cos(\theta_k) + \varphi''_k \sin(\theta_k) \tag{32}$$

where θ_k is determined by minimizing the free energy $\tilde{F}_e[\psi_{k+1}]$ as a function of θ_k.

Dynamic Structure Factor

The main parameter which we are calculating is the dynamic structure factor. The dynamic structure factor is the main parameter of interest because it contains information about both the temporal and spatial correlations in the system, making it a very sensitive test of a model. From the dynamic structure factor, we can extract parameters like the sound speed, that are important inputs to plasma simulations. Additionally, the dynamic structure factor is experimentally accessible. Given that there are not many parameters that can be experimentally measured at the high temperatures and pressures of warm dense matter, coupled with the fact that the dynamic structure factor really probes the microphysics of the system, means that as more experiments come out this parameter will be an excellent test of warm dense matter models. We now discuss both the experimental and theoretical methods of computing the dynamic structure factor.

Experimental Method

Through inelastic neutron scattering we can experimentally measure the double differential cross section, $\frac{d^2\sigma}{d\Omega d\omega}$, which tells us the number of neutrons that will be scattered at a given incident angle, Ω, at a given energy, ω. Then, the dynamic structure factor, $S(k,\omega)$, can be determined from the double differential cross section through the relation

$$\frac{d^2\sigma}{d\Omega d\omega} = a^2 \left(\frac{E_f}{E_o}\right)^{1/2} S(k,\omega) \tag{33}$$

where a is the scattering length, and E_o and E_f are the initial and final energies of the scattered neutron respectively.

Theoretical Method

The theory of calculating the dynamic structure factor from [4] begins with the van-Hove function $G(\mathbf{r},t)$, which for a system of N particles with positions $r_k(t)$ for $k = 1 \dots N$ is defined as

$$G(\mathbf{r},t) = \frac{1}{N}\Big\langle \int \sum_{i=1}^{N}\sum_{j=1}^{N} \delta[\mathbf{r} - \mathbf{r}_j(t) + \mathbf{r}_i(0)] \Big\rangle \tag{34}$$

which can be rewritten successively as

$$\begin{aligned} G(\mathbf{r},t) &= \frac{1}{N}\Big\langle \int \sum_{i=1}^{N}\sum_{j=1}^{N} \delta[\mathbf{r}' + \mathbf{r} - \mathbf{r}_j(t)]\delta[\mathbf{r}' - \mathbf{r}_i(0)]d\mathbf{r}' \Big\rangle \\ &= \frac{1}{N}\Big\langle \int \rho(\mathbf{r}' + \mathbf{r}, t)\rho(\mathbf{r}', 0)d\mathbf{r}' \Big\rangle \\ &= \frac{1}{N}\Big\langle \rho(\mathbf{r},t)\rho(\mathbf{0},0) \Big\rangle \end{aligned} \tag{35}$$

The van Hove function therefore has the meaning of a density-density time correlation function. The physical interpretation of the the van Hove function is that $G(\mathbf{r},t)d\mathbf{r}$ is the number of particles j in a region $d\mathbf{r}$ around a point $\mathbf{r}$ at time t given that there was a particle i at the origin at time $t = 0$.

Rather than considering the density-density correlation in real space it is often more convenient to focus attention on the correlation function of the Fourier components $\rho_{\mathbf{k}}$:

$$F(\mathbf{k},t) = \frac{1}{N}\left\langle \rho_{\mathbf{k}}(t)\rho_{-\mathbf{k}}(0)\right\rangle \tag{36}$$

The function $F(\mathbf{k},t)$ is called the intermediate scattering function and is closely related to the cross-section measured in inelastic scattering experiment. It can be shown that $F(\mathbf{k},t)$ is the spatial Fourier transform of the van Hove function, i.e.

$$F(\mathbf{k},t) = \int G(\mathbf{r},t)\exp(-\mathrm{i}\mathbf{k}\cdot\mathbf{r})\mathrm{d}\mathbf{r} \tag{37}$$

Finally, the dynamic structure factor is arrived at by taking the temporal Fourier transform of the intermediate scattering function $F(\mathbf{k},t)$, i.e.

$$S(\mathbf{k},\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\mathbf{k},t)\exp(i\omega t)dt \tag{38}$$

Now, with the background theory defined, we present our convergence studies and results.

Convergence Studies

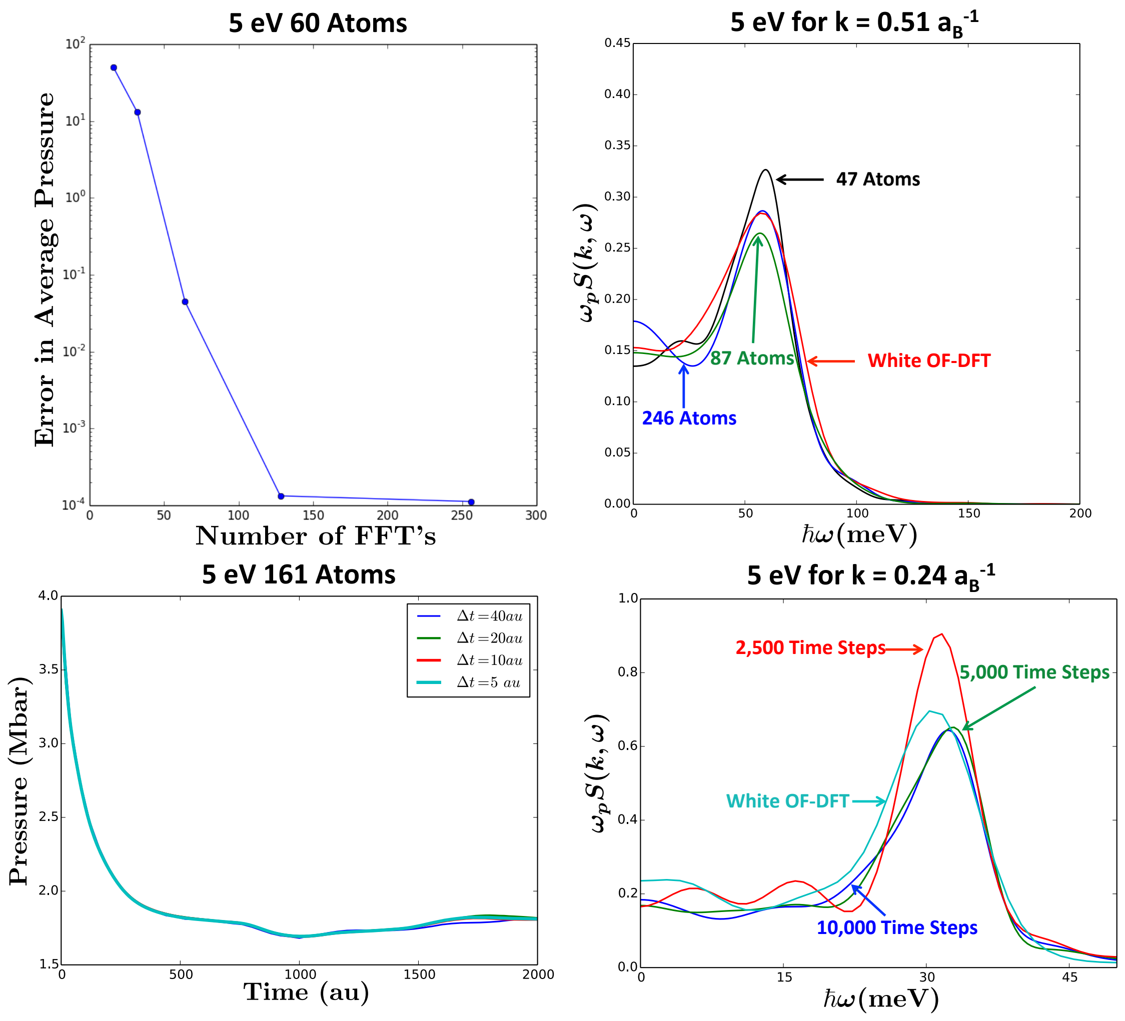


Figure 1: (Top left panel) FFT convergence plot for a system of 60 atoms at 5 eV. (Bottom left panel) Time step convergence plot for a system of 161 atoms at 5 eV. (Bottom right panel) Simulation length convergence plot for a system of 161 atoms at 5 eV. (Top right panel) Number of atoms convergence plot for $k = 0.51 a_B^{-1}$ at 5 eV.

In order to produce reliable results with the code, the calculations had to be converged with respect to the number of FFT's used to solve the Poisson equation, the time step, the simulation length, and the number of atoms. We checked for convergence with respect to the number of FFT's by studying how using more FFT's reduced the error in the average pressure. As can be seen in the top left panel of Fig. 1, using more FFT's significantly reduces the error up until a certain point, after which the error versus number of FFT's curve flattens out. Convergence with respect to the time step size was tested by running the same system with different time steps and plotting the pressure versus time as in the bottom left panel of Fig. 1 to ensure that, even with larger time step sizes, there were still enough points to accurately resolve the pressure curve fluctuations. We checked for convergence with respect to the simulation length by running a long simulation and comparing the dynamic structure factor obtained when calculated with subsets of the long simulation, as in the bottom right panel Fig. 1. Finally, we checked for convergence with respect to the number of particles, top right panel of Fig. 1, by running different size systems, computing the dynamic structure factor, and comparing to the orbital free study results in [8].

Results

We now present the results of our orbital free warm dense aluminum calculations. The calculations were done with aluminum at a density of 2.7 g/cm^3 and at temperatures of 5, 20, and 50 eV. We begin with comparisons between the dynamic structure factor from our orbital free calculuations and the dynamic structure factor from pseudo-atom molecular dynamics calculations. Then, we present our dispersion relation and sound speed calculations at 5, 20, and 50 eV.

Dynamic Structure Factor Results

Recently, the dynamic structure factor for warm dense aluminum calculated using pseudo-atom molecular dynamics in [3] was compared with the dynamic structure factor calculated in the orbital free study in [3] and discrepancies were found at 5 eV. Comparisons at temperatures higher than 5 eV were not possible because the pseudopotential used in [8] was only good up to 5 eV. Our orbital free code, however, computes the pseudopotential self-consistently using Thomas-Fermi-Dirac functional. This means it can run higher temperature calculations and, because it is orbital free, it can do so without incurring much additional computational cost. Below, we present comparisons between the dynamic structure factor from our orbital free calculations with the dynamic structure factor from the pseudo-atom molecular dynamics calculations at 5, 20, and 50 eV for three wavenumbers spanning the single particle, generalized hydrodynamic, and hydrodynamic regimes. These dynamic structure factor curves, particularly for the wavenumbers in the generalized hydrodynamic and hydrodynamic regimes, are a very sensitive test of agreement between models because the dynamic structure factor contains information about both the temporal and spatial correlations in the system.

Single Particle Regime ($k = 2a_B^{-1}$)

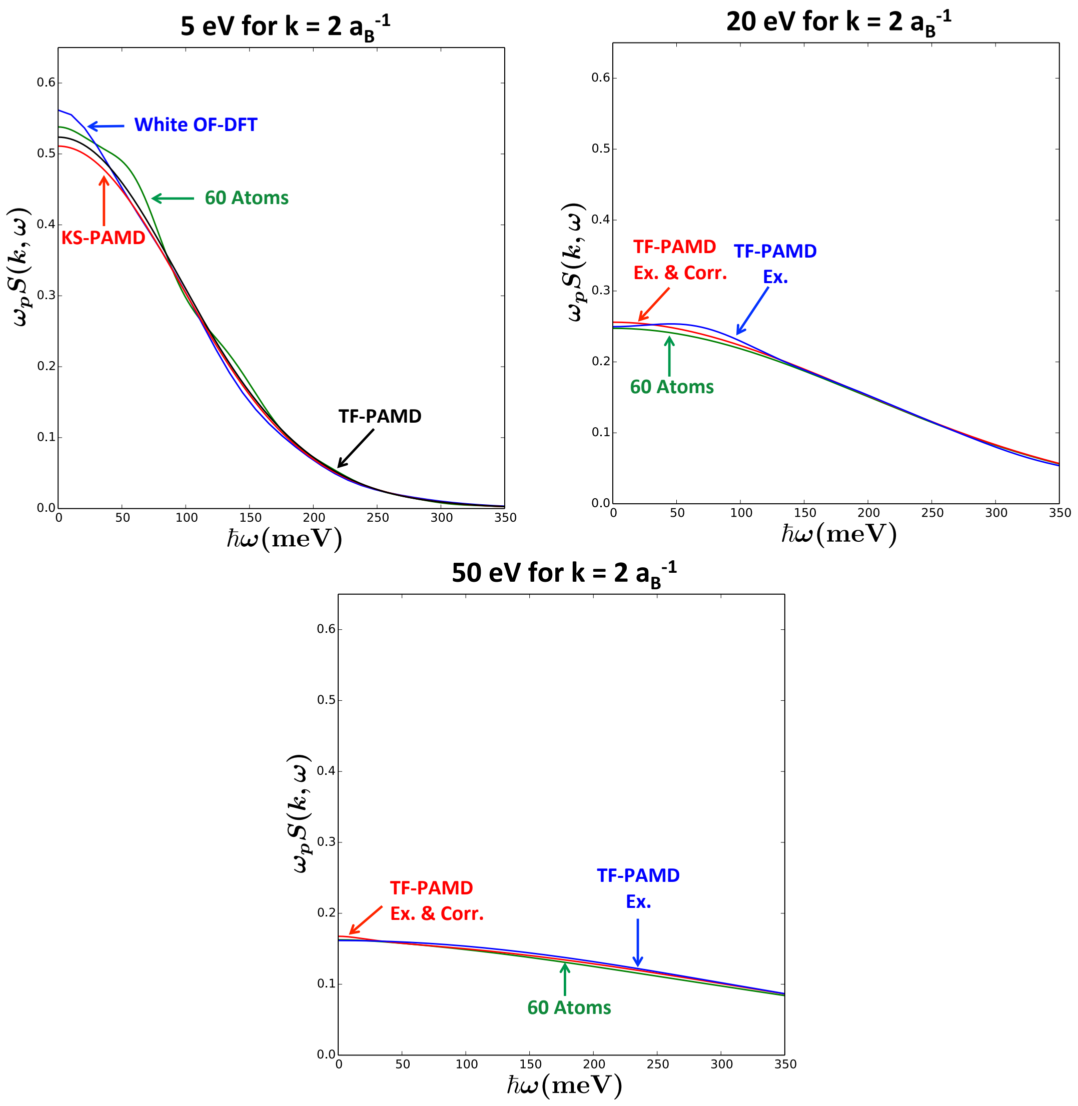


Figure 2: $k = 2a_B^{-1}$ dynamic structure factor for warm dense aluminum at 5 eV (top left panel), 20 eV (top right panel), and 50 eV (bottom panel).

We use a system of 60 atoms with 128 FFT's to compute the dynamic structure factor for $k = 2a_B^{-1}$, which falls in the single particle regime. In the single particle regime, microscopic interactions are important while macroscopic interactions matter far less. Since particles behave as though they are free and non-interacting, all models are expected to converge to a free electron gas result. Thus, we should see good agreement between our orbital free calculation results and the PAMD calculation results. Indeed, as expected, our results agree well with the PAMD calculation results at 5 eV, with better agreement at 20 eV, and the best agreement at 50 eV. As temperature increases, the dynamic structure factor curve gets lower and flatter. This is expected because at higher temperatures, the system becomes less ordered and there are fewer correlations in the ionic positions.

Generalized Hydrodynamic Regime ($k = 0.51a_B^{-1}$)

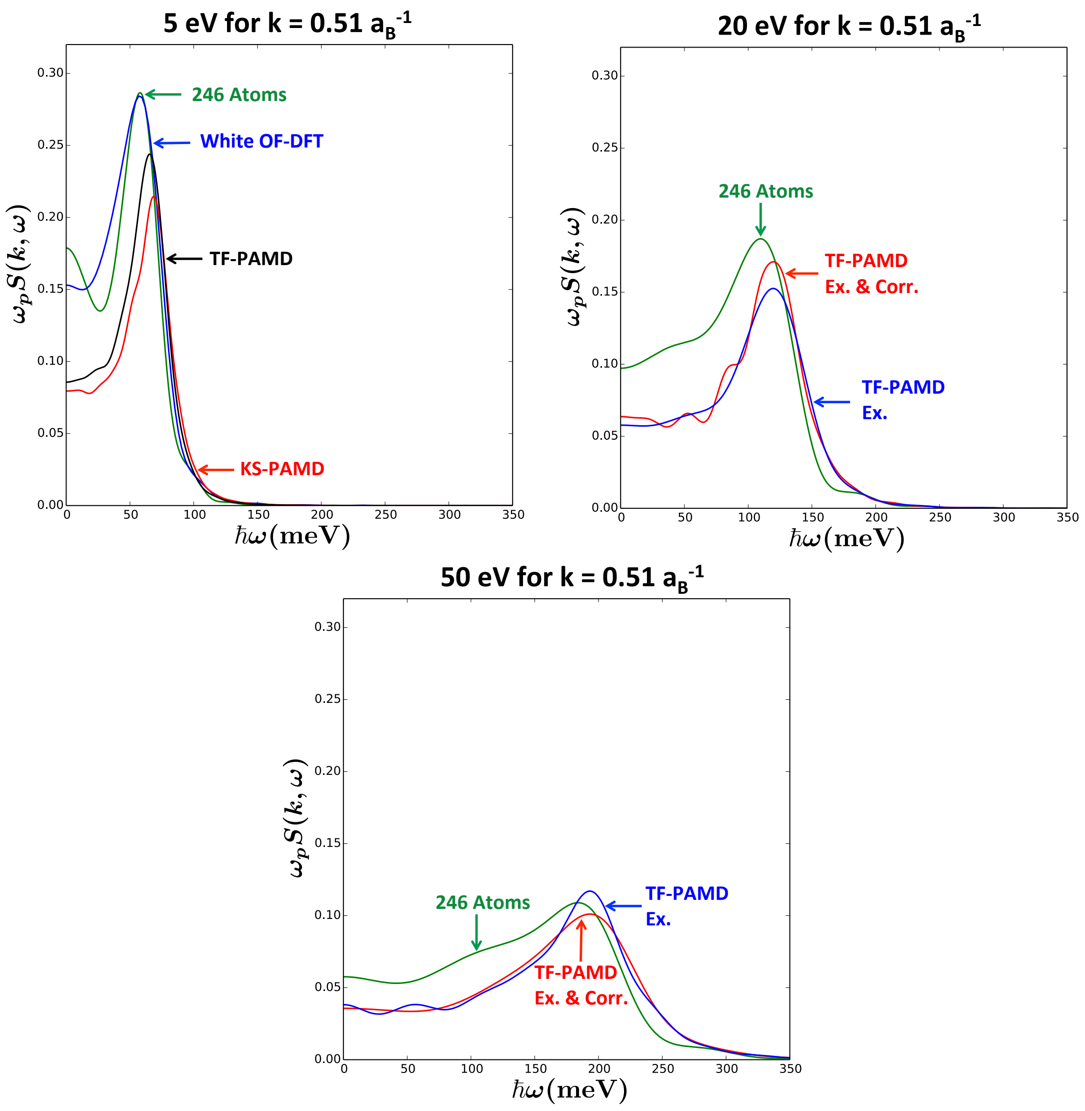


Figure 3: $k = 0.51a_B^{-1}$ dynamic structure factor for warm dense aluminum at 5 eV (top left panel), 20 eV (top right panel), and 50 eV (bottom panel).

We use a system of 246 atoms with 256 FFT's to compute the dynamic structure factor for $k = 0.51a_B^{-1}$, which falls in the generalized hydrodynamic regime. In the generalized hydrodynamic regime, both microscopic and macroscopic interactions become important. Here, the model matters more than it did in the single particle regime and differences between the models should start to manifest. At 5 eV, there is quite good agreement in the position and height of

the ion-acoustic resonance for our orbital free calculation with 246 atoms and White's orbital free calculation with 864 atoms. The ion-acoustic resonance for the orbital free calculations is higher and to the left of the ion-acoustic resonance for the PAMD calculations and the lower energy portions of the curves do not have good agreement.The higher energy portions of the dynamic structure factor curves, however, have quite good agreement.
At 20 eV, there is a similar trend where the ion-acoustic resonance for our orbital free calculation is higher and to the left of the ion-acoustic resonance for the PAMD calculations. Again, while there is disagreement in the lower energy portion of the curve, the higher energy portion sees much better agreement. Our orbital free calculations used a Thomas-Fermi-Dirac functional. When comparing our results to the PAMD calculations done using a Thomas-Fermi functional with only exchange and to the PAMD calculations done using a Thomas-Fermi functional with both exchange and correlation, which is the same thing as a Thomas-Fermi-Dirac potential, we find better agreement with the Thomas-Fermi PAMD calculation with both exchange and correlation.
Finally, at 50 eV, we once again find that the position of the ion-acoustic resonance for our orbital free calculations is shifted to the left of the ion-acoustic resonance for the PAMD calculations. There is still better agreement in the higher energy portion of the dynamic structure factor curve than in the lower energy portion. The Thomas-Fermi PAMD calculation with only exchange has a higher ion-acoustic resonance peak than that of the orbital free calculation and has a more pointed shape. The Thomas-Fermi PAMD calculation with both exchange and correlation, however, has a shape that better matches the orbital free calculation and is consistent with the trend we saw at lower temperatures where the ion-acoustic resonance peak is lower than that of the orbital free calculation. We see the best agreement in ion-acoustic peak height and position between the orbital free calculations and the PAMD calculations at 50 eV.

Hydrodynamic Regime ($k = 0.24 a_B^{-1}$)

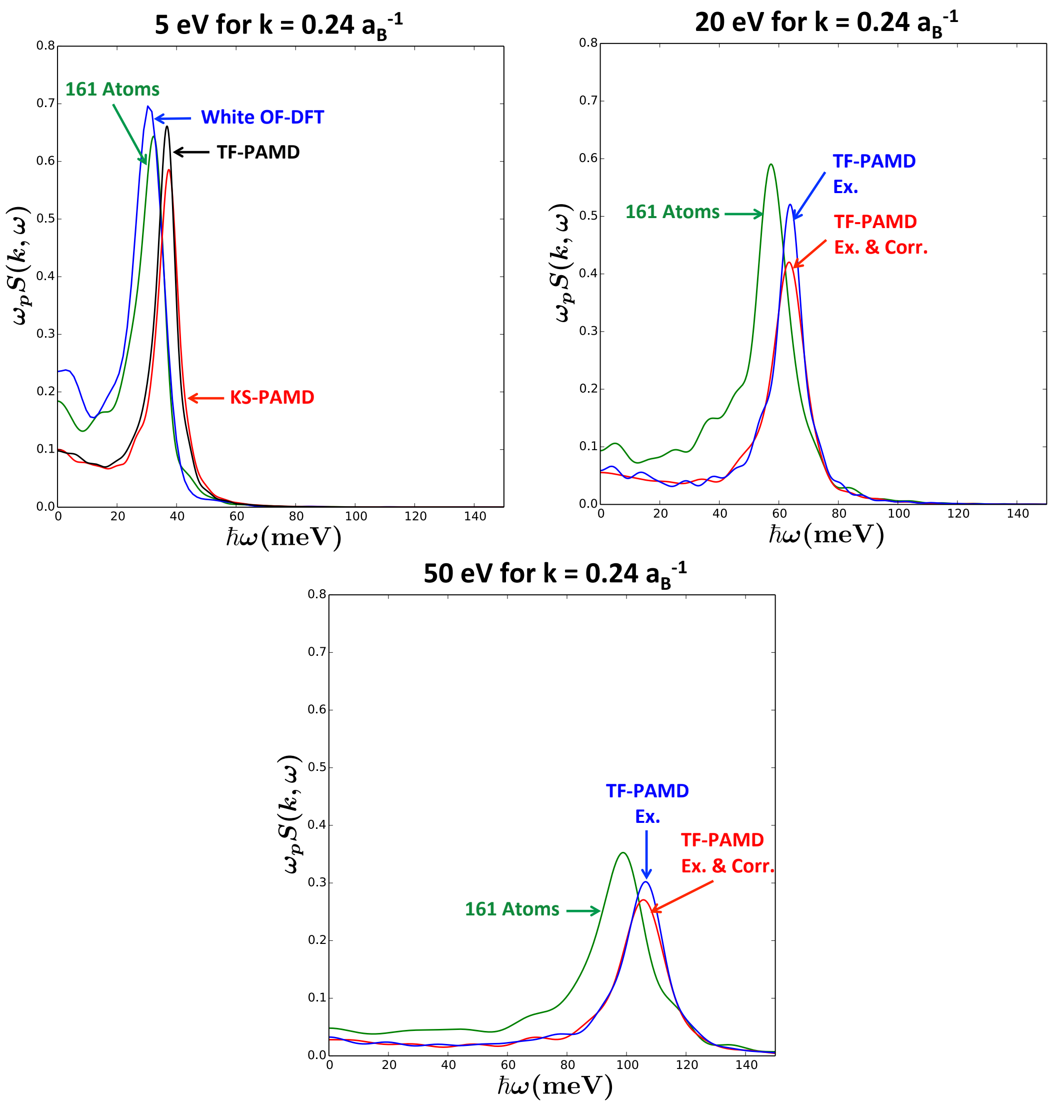


Figure 4: $k = 0.24 a_B^{-1}$ dynamic structure factor for warm dense aluminum at 5 eV (top left panel), 20 eV (top right panel), and 50 eV (bottom panel).

We use a system of 161 atoms with 256 FFT's to compute the dynamic structure factor for $k = 0.24 a_B^{-1}$, which falls in the hydrodynamic regime. In the hydrodynamic regime, the macroscopic interactions become far more important than the microscopic interactions. Here, at these small k values, the model is extremely important and comparisons between the dynamic structure factor curves are thus a very sensitive test of the differences between the models. At 5 eV, we see fairly good agreement between White's orbital free calculation with 864 atoms and our orbital free calculation with 161 atoms. The height of our calculation's ion-acoustic resonance is slightly lower than that of White's but there is good agreement on either side of the peak and we expect that if we run with more atoms, this slight discrepancy will be resolved. We again see that the orbital free calculation ion-acoustic resonance position is shifted slightly to the left of the PAMD calculation ion-acoustic resonance. Unlike the results in the generalized hydrodynamic regime, we do not see good agreement between the orbital free dynamic structure factor and the PAMD dynamic structure factor in any part of the curve.

At both 20 and 50 eV, the orbital free calculation result's ion-acoustic resonance is again to the left and higher than that of the PAMD calculations. Unlike the results in the generalized hydrodynamic regime, the Thomas-Fermi PAMD calculation with both exchange and correlation does not agree as well with our orbital free calculation as the Thomas-Fermi PAMD calculation with only exchange at both 20 and 50 eV.

Sound Speed Calculations

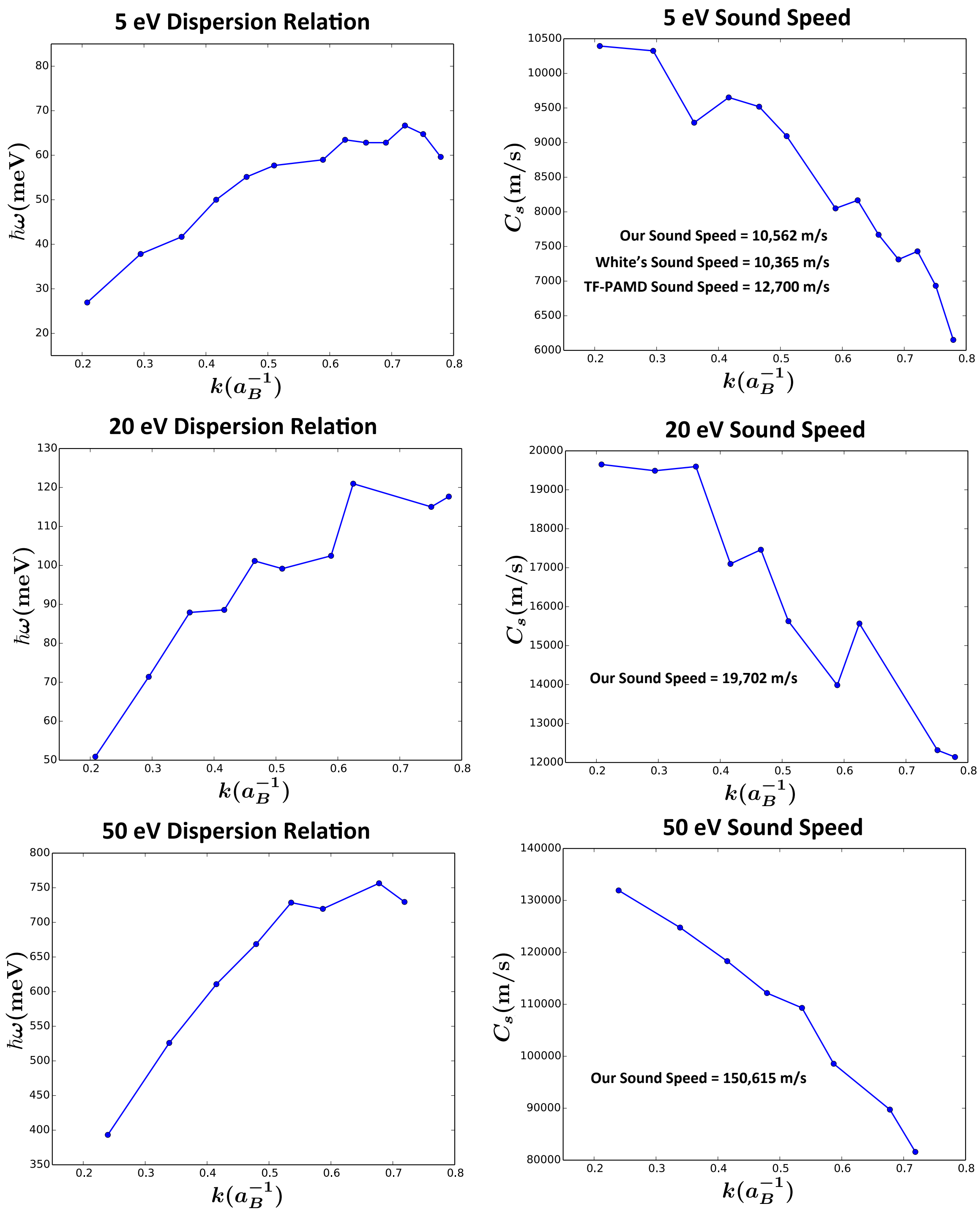


Figure 5: Dispersion relation for warm dense aluminum at 5 eV (top left panel), 20 eV (middle left panel), and 50 eV (bottom left panel). Sound speed plots for warm dense aluminum at 5 eV (top right panel), 20 eV (middle right panel), and 50 eV (bottom right panel).

We present the dispersion relation, which is the position of the ion-acoustic resonance as a function of wavenumber, for our 5 eV, 20 eV, and 50 eV orbital free calculations. From the dispersion relation, we are able to calculate the sound speed by extrapolating the curve given by $C_s = \omega/k$ versus k to $k = 0$. At 5 eV, we compare the sound speed obtained from our orbital free calculation to the sound speeds obtained from White's orbital free calculation and the PAMD calculation. We find much better agreement with White's orbital free calculation, with the PAMD calculation giving a higher sound speed. The 20 and 50 eV sound speeds from our orbital free calculations increase significantly as temperature increases, which is to be expected because particles already move quickly in high temperature systems and thus sound is able to propagate faster.

Future Work

In the future, more work must be done with regards to convergence studies, particularly with respect to the number of atoms. Convergence studies have been somewhat limited due to the high computational cost of running large systems. A significant amount of this computational cost comes from the process of minimizing the free energy. The current code uses nonlinear conjugate gradients where the initial guess is the density calculated in the previous iteration. It generally takes about 7 conjugate gradient iterations for each time step of the simulation. We have begun attempting to speed up this process by implementing an extended lagrangian technique in which an auxilliary density is propagated independent of the rest of the simulation according to equation (39).

$$n_{i+1} = 2n_i - n_{i-1} + \kappa(q_i - n_i) + \alpha \sum_{k=0}^{K} c_k n_{i-k}. \tag{39}$$

This auxiliary density is then used as the initial guess for the conjugate gradient method. Using this density provides a better guess and keeps the free energy from drifting even when the convergence tolerance is loosened. With this method, we are able to relax the convergence tolerance so this technique can greatly reduce the number of conjugate gradient iterations that must be done thereby giving a huge speed up in the rest of the code.

Finally, we are currently running more calculations at 100 eV and 500 eV to reach the linear response regime in which the PAMD and orbital free frameworks should produce exactly the same results. Then, we will go down in temperature to determine where the two models begin diverging.

Conclusions

We were able to run 5 eV, 20 eV and 50 eV orbital free molecular dynamics calculations for warm dense aluminum at 2.7 g/cm^3. At 5 eV, we have found good agreement between our orbital free warm dense aluminum calculations and those performed by White et al in 2013. In the single particle regime, ($k = 2a_B^{-1}$), we find good agreement between our orbital free calculations and PAMD results as expected. In both the hydrodynamic and generalized hydrodynamic regimes, we have found that the ion-acoustic resonance for our orbital free calculations is slightly higher and shifted to the left of the ion-acoustic resonance for the PAMD calculations. In all regimes, the agreement between the dynamic structure factor appears to get slightly better as the temperature increases.

Acknowledgements

We would like to thank Charlie Starrett, Lee Collins, Chris Ticknor, Daniel Livescu, and Robin Heinonen for their invaluable assistance with this project.

References

[1] R. P. Feynman, N. Metropolis, and E. Teller. Equations of state of elements based on the generalized fermi-thomas theory. *Phys. Rev.*, 75:1561–1573, May 1949.

[2] Ronald Redmer Samuel B. Trickey Frank Graziani, Michael P. Desjarlais. *Frontiers and Challenges in Warm Dense Matter*.

[3] N. M. Gill, R. A. Heinonen, C. E. Starrett, and D. Saumon. Ion-ion dynamic structure factor of warm dense mixtures. *Phys. Rev. E*, 91:063109, Jun 2015.

[4] J. P. Hansen and I. McDonald. *Theory of Simple Liquids with Applications to Soft Matter*. Academic, London, 2013.

[5] Hong Jiang and Weitao Yang. Conjugate-gradient optimization method for orbital-free density functional calculations. *The Journal of Chemical Physics*, 121(5):2030–2036, 2004.

[6] W. R. Johnson. Atoms at finite temperature. August 2000.

[7] S. H. Vosko, L. Wilk, and M. Nusair. Accurate spin-dependent electron liquid correlation energies for local spin density calculations: A critical analysis. *Canadian Journal of Physics*, 58(8):1200–1211, 1980.

[8] T. G. White, S. Richardson, B. J. B. Crowley, L. K. Pattison, J. W. O. Harris, and G. Gregori. Orbital-free density-functional theory simulations of the dynamic structure factor of warm dense aluminum. *Phys. Rev. Lett.*, 111:175002, Oct 2013.

Modeling Warm Dense Matter using Quantum-Mechanical Density Function Theory

Team Members
Daniel Burrill and David Feinblum

Mentors
Charlie Starrett and Marc Charest

Abstract

The Warm Dense matter regime has provided an excellent challenge to physical scientists due to its high temperatures and pressures. These conspire to make models of such systems computationally expensive and scientifically complex. Hydrocodes, which are used to model experiments at these conditions, rely on bulk properties like conductivity and resistivity to more accurately describe the physics occuring. In this paper, we utilize the Ziman-Evans formula to calculate the conductivity/resistivity of warm dense plasmas made of aluminum, hydrogen, beryllium, and mixtures.

Introduction

Most systems that feature physics of more than one type are difficult to model. In the case of warm dense matter (WDM), a state of matter which shares characteristics with both solids and plasmas, high temperatures and pressures conspire to elevate both quantum and classical effects to equivalent significances. High temperatures cause the Fermi-Dirac function to smear out, and this leads to partial occupation of the orbitals. Additionally, high pressures increase the amount of interaction ions and electrons experience, which further complicates modeling.

To capture this physics, quantum molecular dynamics (QMD) calculations can be used, but are computationally expensive. Further, there are issues with convergence for certain systems, particularly at high temperatures. In spite of these issues, QMD data is considered the gold standard when experimental data is not available. The ultimate use of this QMD data is in hydrocodes, where computationalists use tables of bulk properties as input. These bulk properties are vital for understanding and accurately simulating many systems which pass through the WDM regime, such as holoraums in inertial confinement fusion (ICF) experiments, inner-planetary cores, the earth's dynamo, and many others.

Since these properties need to be known at a large number of temperatures and densities, a cheaper method of calculating them is very attractive. Plasmas and solids have both been modeled successfully using density functional theory (DFT). Once the density is solved for, one may use it to generate a potential, which can then be used in a molecular dynamics (MD) simulation.

These two methods in tandem with one another have been used successfully in the past to simulate WDM. In this report, we will use DFT-MD to simulate warm dense systems and use the output of these simulations to calculate bulk properties.

Theory

Simulations within WDM systems have moved into the realm of combining density functional theory (DFT) with molecular dynamics (MD) in an effort to couple short calculation times with decent agreement to experiment. Properties calculated using these approaches depend heavily on the potentials used to model the ions within the systems due of DFT's reliance on external (ionic) potentials to generate orbitals. The two major types of models are the screened potentials which utilize pseudopotentials and the average atom (AA) model which describes ions within a set radius within a background of free electrons. Addtionally, the pseudoatom (PA) model is a more recent design [1–3] which builds off of the AA model but includes ion-ion correlations more effectively.

Such potentials form the basis of the theory of calculating bulk properties such as the structure factor [1, 2, 4–9], opacity [10, 11], thermal conductivity [12–16] and electrical resistivity [9, 11–23] within WDM. This report focusses on calculating the resistivity using the PA model within the framework of the Ziman-Evans formula. The use of the various potentials in deriving this formula is presented along with considerations of the ion-ion correlations contatined within the structure factor as well as the level of approximation to describe the scattering interaction. Following the derivation of the Ziman-Evans formula for mixtures, a discussion of the various potential models is presented.

Ziman-Evans Formula

In the mid-twentieth century growing interest in calculating the conductivities in solid metals prompted many investigations into the theory behind the role of electron scattering within ordered materials. However, the problem proved to be intractable at the time due to the unknown nature of the ion-electron interactions. Ziman's 1961 paper [24] offered an approach to the problem by considering the properties of liquid metals. In these systems the ion-ion correlations could be determined from experiment by measuring the structure factor while the ion-electron interations were known for a number of simple systems. He demonstrated that by calculating the resistivities in liquid metals, band gaps could be estimated for the solid phase.

Ziman's approach focussed on calculating the resistivity directly by considering weakly interacting pseudopotenials to govern ion-electron interactions where the Born approximation could be applied to simplify scattering processes. His derivation proceeds from calculating the conductivity within a gas where ordering of the ions is a minimum as shown in Eq. 1,

$$\sigma = \frac{ne^2\Lambda}{mv_F} \tag{1}$$

where n is the electron density, e is the electron charge, m is the electron mass, v_F is the Fermi velocity, and Λ is the mean free path. The scattering cross section, $\mathscr{F}$, is used to describe the mean free path of a scattering eletron as shown in Eq. 2,

$$\frac{1}{\Lambda} = 2\pi \sum_i \frac{N_i}{V} \int_0^{2\pi} \mathscr{F}_i(\theta) \sin\theta d\theta \tag{2}$$

where $\frac{N_i}{V}$ is the ion density per unit volume and θ is the scattering angle. Note that the sum over i accounts for all ion types in the system making this a general formula applicable to mixtures. At this point, the choice of a weakly interacting pseudopotential is utilized by applying the Born approximation as shown in Eq. 3,

$$\mathscr{F}(\theta) = \frac{1}{6\hbar v_F} \frac{n}{\mathscr{E}_F} \left| \frac{\mathscr{U}_K}{N} \right|^2 \tag{3}$$

where $\mathscr{E}_F$ is the Fermi energy and $\mathscr{U}_K$ is the potential in k-space. Eqs. 1-3 give the full description of the conductivity of a gas provided information about the Fermi energy, electron and ion densities, and potential are known. When compared to experimental values, Ziman found that the mean free paths calculated using this derivation agreed with experimental value qualitatively (quatitatively, some values differed from experiment by an order of magnitude). However, considering that creating the gasseous phase of most metals is a formidable problem experimentally, liquid or molten systems must be considered. While not perfectly ordered, the liquid phase of metals exhibits ion-ion correlation which, as stated previously, can be measured from the structure factor.

Ziman proposed that the structure factor, $a(K)$, enters the calculation through the potential and ultimately changes the mean free path as shown in Eq. 4,

$$\Lambda_{liq} = \frac{2\sqrt{2}}{3\pi} \frac{\mathscr{E}^{3/2}}{\langle a \rangle |\mathscr{U}|^2} \tag{4}$$

where he included the definition,

$$\langle a\rangle |\mathscr{U}|^2 = \frac{1}{4k_F^4}\int_0^{2k_f} |\mathscr{U}|^2 a(K)K^3 dK \tag{5}$$

Here we see that the mean free path is inversely related to the structure factor. For a completely disordered system (i.e. gas) the structure factor would be equal to one while order introduces an average value that is lower than one. In effect, ion-ion correlations within a system increase the mean free path of a scattering electron. The strength of this approach lies in the definition of the potential created by the ions. By assuming a weak scattering potential a number of small elements can be approximated which give qualitative agreement to experiment.

The significance of the choice of potential was further examined by Ashcroft and Lekner [9] where Ziman's formula for conductivity was utilized to calculate the resistivities of a series of liquid metals. Their approach focussed on determining the structure factor from a given model potential as well as testing a number of proposed potentials to determine which qualities are necessary to reproduce experimental values. They showed that potentials which account for screening and possess a more physical multi-well pseudopotential acheive qualitative agreement for a number of elements. However, since their analysis relied on Ziman's use of the Born approximation, it is difficult to tell whether the structure factor, potential, or Born approximation are the culprit where agreement is not found.

By 1973 Evans, Gyorffy, Szabo, and Ziman had proposed a solution to creating better potentials as well as disregarding the Born approximation in favor of a full scattering method known as the T-matrix [25]. Gyorffy and Szabo formulated the resisitivity within a material in terms of the average over the force operator as shown in Eq. 6,

$$R = \frac{-2\pi\hbar}{3e^2}\left(\frac{\Omega}{N_e}\right)^2 \lim_{\Omega\to\infty}\frac{1}{\Omega}\int dE f'(E)\left\langle \mathrm{Tr}\left\{F\delta(E-H)F\delta(E-H)\right\}\right\rangle \tag{6}$$

where Ω is the real space volume in question, $f'(E)$ is the derivative of the Fermi-Dirac distribution, and F is the force operator. In this case, the force operator is defined by the relation to the model potential, V,

$$\boldsymbol{F} = -\sum_i \nabla V\left(\boldsymbol{r}-\boldsymbol{R}_i\right) \tag{7}$$

From this, Evans was able to recover a version of Ziman's formula but with a general scattering term. Two major assumptions in the Evans model allowed for significant simplifications. The first is that the potential was described by a finite range, spherical, muffin-tin model about each atom. The second is that none of the potentials overlapped such that the interstitial region could be approximated as a screening background. This resulted in the Ziman-Evans formula as shown in Eq. 8,

$$R = \frac{48\pi^3\hbar}{e^2k_F^2}\frac{1}{\Omega_0}\int_0^1 d\left(\frac{q}{2k_F}\right)\left(\frac{q}{2k_F}\right)^3\left|t_{\boldsymbol{k},\boldsymbol{k}'}\right|^2 S(q) \tag{8}$$

where Ω_0 is the atomic volume defined by muffin-tin radius, q is the scattering momentum given by $\boldsymbol{q} = |\boldsymbol{k}-\boldsymbol{k}'|$, $S(q)$ is the structure factor (note the change in notation) evaluted at the scattered momentum, and $t_{\boldsymbol{k},\boldsymbol{k}'}$ is the T-matrix describing the full electron-ion scattering interaction. Note that this formula is only valid when the derivative of the Fermi-Dirac function approximates a delta function. In the context of WDM, this corresponds to very low temperature systems.

To capture the electron-ion interaction within the T-matrix approach, consideration must be given to what scattering takes place. From scattering theory we know that a scattering event depends on both the scattered particle as well as the potential that it is scattered from. However, rather than explicitly account for both of these in a formula, only the *effect* of scattering can be examined to describe the interaction. That is, if we look at the phase shift generated by the scattering potential as compared to the unscattered wave, we know an equivalent amount of information about the scattering process. The T-matrix expression is shown in Eq. 9,

$$t_{\boldsymbol{k},\boldsymbol{k}'}(E) = \frac{-1}{\sqrt{E}} \sum_l (2l+1) \exp i\eta_l \sin\eta_l P_l(\cos\theta_{\boldsymbol{k},\boldsymbol{k}'}) \tag{9}$$

where η is the phase shift and P_l are Legendre polynomials. Note that this expression is similar to the standard expression for the scattering differential cross section.

Converted into atomic units, Perrot and Dharma-Wardana present the form of the Ziman-Evans formula used in this paper. Eq. 10 shows the formulation which differs from Eq. 8 in two ways. First, the energy integral is recovered through not applying the approximation on the derivative of the Fermi-Dirac function, and secondly the scattering interaction is now encapsulated within the scattering cross section, $\sigma(\varepsilon)$, rather than explicitly through a T-matrix term. In the notation used here n_I^0 is the ion density and $\overline{n}_e^0$ is the average electron density.

$$R = -\frac{n_I^0}{3\pi \left(\overline{n}_e^0\right)^2} \int_0^\infty d\varepsilon \sigma(\varepsilon) \frac{\partial f}{\partial \varepsilon} \tag{10}$$

where the scattering cross section is given as,

$$\sigma(\varepsilon) = \int_0^{2p} q^3 \frac{\partial\sigma(\varepsilon,\theta)}{\partial\theta} S_{II}(q) dq \tag{11}$$

where q is the momentum transferred in the scattering process and S_{II} is the structure factor of atomic species I . And finally, the differential cross section is the T-matrix expression,

$$\frac{\partial\sigma(\varepsilon,\theta)}{\partial\theta} = \frac{1}{p^2} \left| \sum_{l=0}^{\infty} (2l+1) \sin\eta_l \exp i\eta_l P_l(\cos\theta) \right|^2 \tag{12}$$

Eqs. 10-12 are expressions only valid for systems with only one ionic species. An extension for multiple species is required to compute the resistivity within mixtures. The differential cross section is altered, as shown in Eq. 13, to accomodate a structure factor matrix, S_{mn} and the relative concentrations of each species, x_m.

$$\frac{\partial\sigma(\varepsilon,\theta)}{\partial\theta} S_{II}(q) = \sum_{m,n}^{N_s} \sqrt{x_m x_n} S_{mn}(q) \mathscr{F}_m(\varepsilon,\theta) \mathscr{F}_n^*(\varepsilon,\theta) \tag{13}$$

The equivalent T-matrix term occurs within the scattering amplitude, $\mathscr{F}$, and is given by the expression,

$$\mathscr{F}_m(\varepsilon,\theta) = \frac{1}{p}\sum_{l=0}^{\infty}(2l+1)\sin\eta_{lm}\exp i\eta_{lm}P_l(\cos\theta) \tag{14}$$

where we see that phase shifts are calculated from scattering due to each species' ionic potential. In the Born approximation Eq. 14 simplifies to,

$$\mathscr{F}_m(\varepsilon,\theta) = \frac{V_m(q)}{2\pi} \tag{15}$$

Potentials

Fundamental to any scattering problem is the scattering potential. As was seen in the previous section, the resistivity of any material is directly related to the scattering cross section of its ions. These scattering cross sections are determined through understanding the interactions between the scattered particle and the scattering center. In the case of resistivity we assume that the scattered particles are comprised of electrons in a continuum state while ions for the scattering centers. In the case of WDM we find that potentials often used in condensed matter physics no longer suffice due to the disordered nature of the system as well as the strong ionization due to high temperatures.

Several models exist to describe the interactions between ions and electrons with extensions to account for more realistic physics. In this report we will focus on several models: the historically used *Average Atom* approach and the *Pseudoatom potential* [1–3]. Additionally, one of the first potentials, the screened potential will be discussed in the context of Ziman's formulation [24]. These models will be utilized in the Results section where clear differences emerge which can be explained though the approximations used within each potential.

Average Atom

The Average Atom (AA) model is built on the idea of an averaged 'jellium' surrounding a fixed radius atom. Within the atom a nucleus of charge Z is surrounded by an electron density, $n_e(\boldsymbol{r})$, which neutralizes the charge of the atom. Interactions between ions are not taken into account explicitly, but rather through the potential of the external medium.

Each atom is composed of a predefined sphere of radius R which is defined as the Wigner-Seitz radius calculated from the element type and mass density. The boundary at R marks the point at which the atom ends and the surrounding medium begins. In practice, the atomic potential acting on the electrons vanishes at this surface. Additionally, since ion-ion interactions are taken into account in an averaged fashion, the pair distribution function is a step funciton at this boundary as shown in Eq. 16

$$g(r) = H(r-R) \tag{16}$$

where H denotes the Heaviside function and r is a radial distance from the center of the atom.

To understand the electron distribution within the AA model we start with the Schödinger equation subject to an effective potential due to the nucleus and electron-electron interactions.

$$\left[\nabla^2 + V^{eff}\right]\psi_i = \varepsilon_i\psi_i \tag{17}$$

where ∇^2 is the kinetic energy operator, ψ_i are single particle wavefunctions, and ε_i are the corresponding eigenvalues. The V^{eff} term may be split into two parts,

$$V^{eff} = -\frac{Z}{r} + V^{ee} \tag{18}$$

where the first term on the right defines the ion-electron interaction and the second corresponds to electron-electron interaction

Due to the high temperatures of WDM systems special consideration must be applied to V^{ee} in Eq. 18. First, partial ionization of the atoms is accounted for by splitting the potential into bound and continuum state contributions. Then, the quantum effects are incorportated through an exchange-correlation term. This is shown in Eq. 19

$$V^{ee} = V^{bound} + V^{continuum} + V^{xc} \tag{19}$$

where Poisson's equation can be applied to determine the potentials due to the bound and continuum contributions,

$$\nabla^2 V(\boldsymbol{r}) = -4\pi n(\boldsymbol{r}) \tag{20}$$

where $n(\boldsymbol{r})$ is the electron density. To determine $n(\boldsymbol{r})$ we must turn to the solutions of Eq. 17. The eigenfunctions are expanded in a spherical basis [11],

$$\psi_i(\boldsymbol{r}) = \sum_{\alpha} \alpha_i \frac{P_i(r)}{r} Y_{l_i m_i}(\boldsymbol{r}) \chi_{\sigma i} \tag{21}$$

where $P(r)$ is the radial component of the wave function, Y_{lm} are the spherical harmonics, and χ_σ represents the spin of the particle. Then, the densities can be expressed as,

$$n_b(\boldsymbol{r}) = \frac{1}{2\pi r^2} \sum_{nl} (2l+1) f_{nl} P_{nl}^2(\boldsymbol{r}) \tag{22}$$

$$n_c(\boldsymbol{r}) = \frac{1}{2\pi r^2} \sum_{l} \int d\varepsilon (2l+1) f_{\varepsilon l} P_{\varepsilon l}^2(\boldsymbol{r}) \tag{23}$$

where f_{nl} is the Fermi-Dirac function representing the occupation of a given state nl.

Eqs. 17-22 represent a set of equations which can be solved self-consistently for the electron density (or, wave functions) within the AA model. The downside of this model is readily apparent in the way it accounts for ion-ion correlation. For transport properties in particular this correlation represents an important portion of the solution as the resistance term depends directly on it. Therefore, other models which take into account a more physical description of the correlation must be formulated.

Pseudoatom

The pseudoatom aproach builds on the formalism of AA but with two major differences. While the AA potential has been shown to agree with QMD in a number of cases, an improvement to the potential can be made by creating a better description of the interactions within WDM systems which can be accomplished with a pseudoatom.

A pseudoatom is generated first by considering the electron density from an AA potential. The heaviside pair distribution function remains in place as it was shown that including structure effects had little effect on the electron density generated. This potential is then modified by subtracting the resultant electron density from interactions within the system minus the nulcear interaction. That is, if $n_e(\boldsymbol{r})$ is the initial electron density and $n_e^{ext}(\boldsymbol{r})$ is the electron density due to external interactions the pseudoatom density, $n_e^{PA}(\boldsymbol{r})$, can be written as,

$$n_e^{PA}(\boldsymbol{r}) = n_e(\boldsymbol{r}) - n_e^{ext}(\boldsymbol{r}) \tag{24}$$

where $n_e^{ext}(\boldsymbol{r})$ is found by determining the density with the nucleus at the center of the atom removed from the calculation. This yields an atomic potential with the core and valence electrons described within the density.

Secondly, these potentials are then used to build the full potential of a WDM system. The potential is given as the superposition of pseudoatom densities,

$$n_e(\boldsymbol{r}) = \sum_i n_e^{PA}(|\boldsymbol{R_i} - \boldsymbol{r}|) \tag{25}$$

where i represents the index of a specific atom at location $\boldsymbol{R_i}$. Further, dynamic quantities can be calculated through pseudoatom molecular dynamics (PAMD) which relies on the pair interactions generated by these potentials.

Screened Potential

Pseudopotentials have a long history of use within chemistry and condensed matter numerical calculations. The underlying concept is to replace the core electrons of an atom with an effective potential which retains the physics of the original system while requiring less computational time. This gain in computational efficiency can be understood in terms of the kinetic energies of the electrons near the nucleus. Wavefunctions describing these high kinetic energy states possess a significant number of oscillations near the nucleus. If solution to these wavefunctions are determined in the plane-wave basis, for example, many terms must be included to account for these oscillations. By directly reducing the number of terms through a smoother pseudopotential fewer calculations must be performed.

Ziman's original formulation of the resistivity within liquid metals relied on such potentials within the nearly-free electron approximation [24]. This approximation treats the potentials as a perturbation term to the free electron potential which limits the scattering to weak interactions. Therefore, resistivities calculated with pseudopotentials will be examined within the Born approximation labelled as the *screening potential.*

Results

Born

In this section, we will report preliminary results for single-component plasmas of aluminum, hydrogen, and beryllium. For the first two elements, results are presented as conductivity (1/resistivity) and for aluminum, all data is resistivity.

Hydrogen

Hydrogen passes through the WDM regime on its way to ICF and inside planetary cores. As a result, its transport properties at significant temperature and pressure are of great interest to various fields. We begin this report by considering three different densities of hydrogen, all of which have relevant QMD data in the literature.

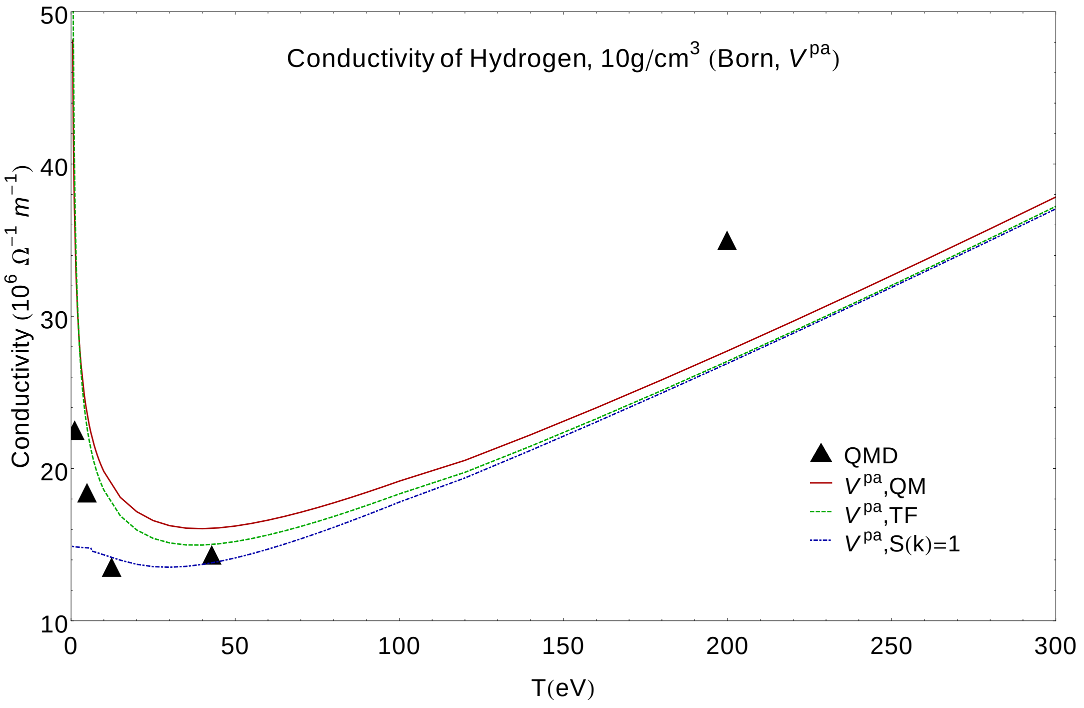


Figure 1: Pseudoatom conductivities of hydrogen at 10g/cm^3. The Born approximation with TF-DFT does a good job of replicating QMD data for temperatures below 50 eV. Beyond this temperature, agreement falls off.

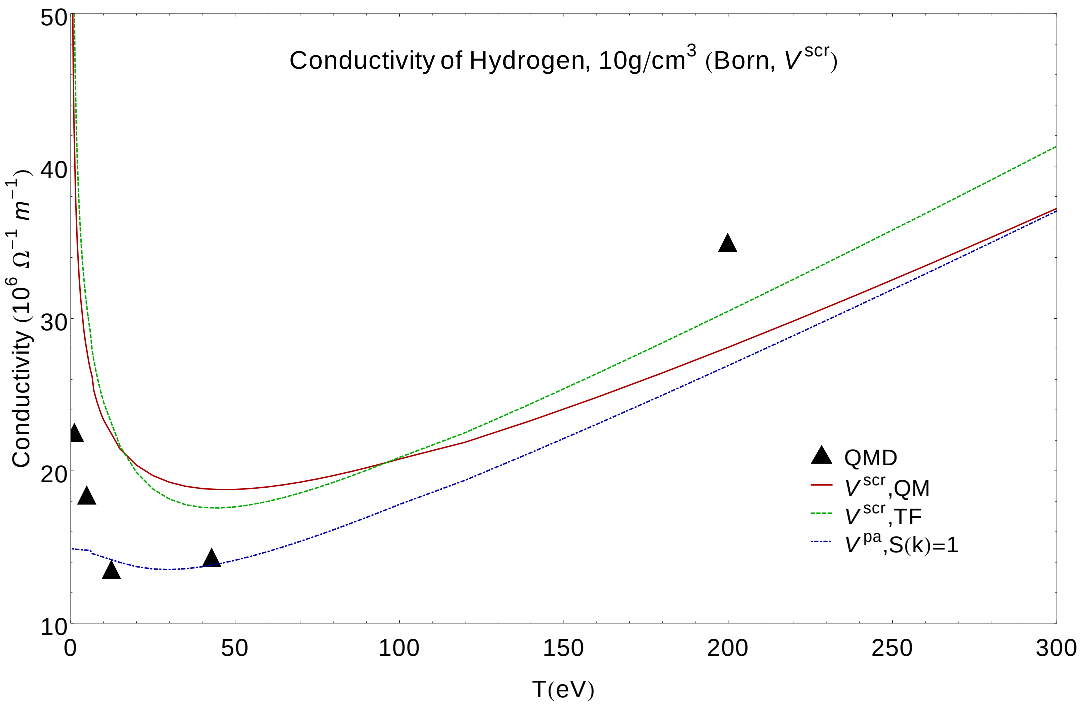


Figure 2: Pseudopotential conductivities of hydrogen at 10g/cm^3. Less agreement than the pseudoatom potential, but still quite good given the computational cheapness. We see substantial deviation at all temperatures though, and the $S(k) = 1$ test being close to QMD data is probably a coincidence more than anything else.

From Figs. 1 and 2, we can already see a few things worth mentioning. First, the pseudoatom potential seems to give the best results at this density and temperature range. We will see this trend continues as we look at other data. Additionally, in the $S(k) = 1$ tests, we see that the structure factor plays an important role in relation to the conductivity, particularly at low temperatures. This seems to make good sense, since setting the structure factor to one is equivalent to assuming an ideal gas. This would lead to less scattering and as Ziman et al. point out in their paper, less scattering leads to a lower conductivity [24].

We also see that, for a system as simple as hydrogen, the Born approximation seems to yield physically reasonable conductivities that agree with QMD data. Curiously it seems that, at this density, the Born approximation starts to fail as temperature increases beyond 50eV. However, ensuring that a QMD calculation is converged at high temperature can be difficult, so it is possible that the high temperature point has a large error. This is strange because the Born approximation, which assumes a weak scattering interaction, should become exact in the infinite temperature limit. It is possible that other effects, such as our choice in XC functional, could be affecting various portions of our integrand. If the structure factor and potentials are inaccurate, these inaccuracies will be made worse by a higer temperature simulation since the Fermi-Dirac function smoothes as temperature increases.

We also ran similar calculations on hydrogen at 80g/cm^3 and find that these results agree with the results at 10g/cm^3. We again observe that the pseudoatom potential follows the QMD data more closely. We also can now see another interesting trend which was hinted at by the data at lower density: it appears that Thomas-Fermi (TF) DFT is closer to QMD than the full Kohn-Sham (KS) DFT calculation. This is surprising in that a KS-DFT calculation should, in principle, contain more physical information about the system.

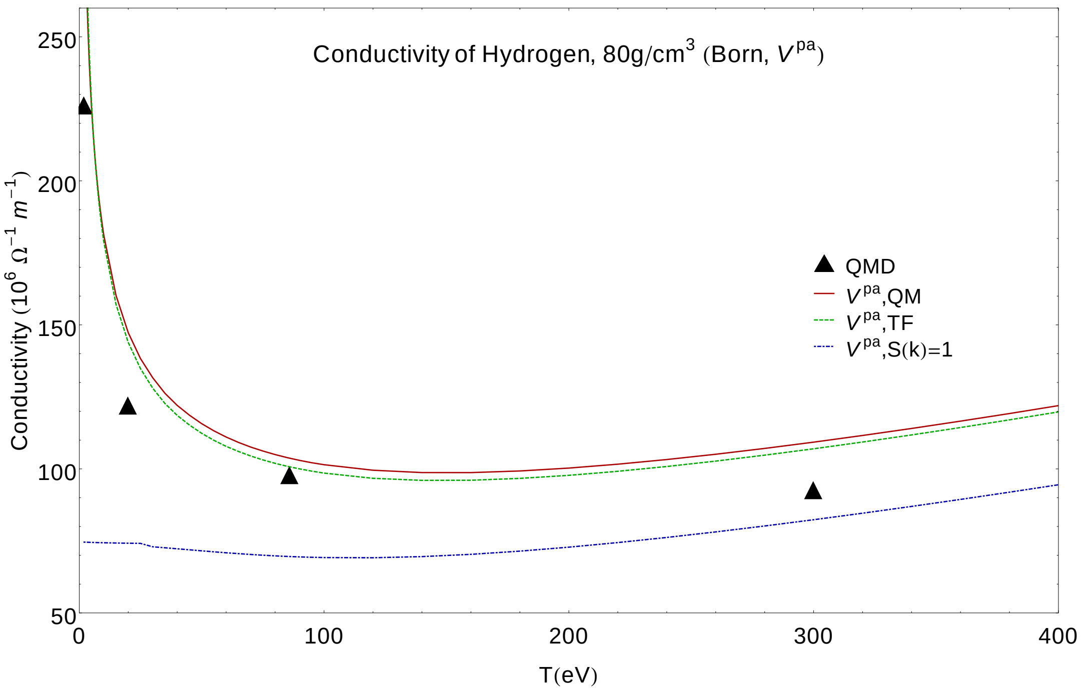


Figure 3: Pseudoatom conductivities of hydrogen at 80g/cm^3. Trends similarly to the previous density.

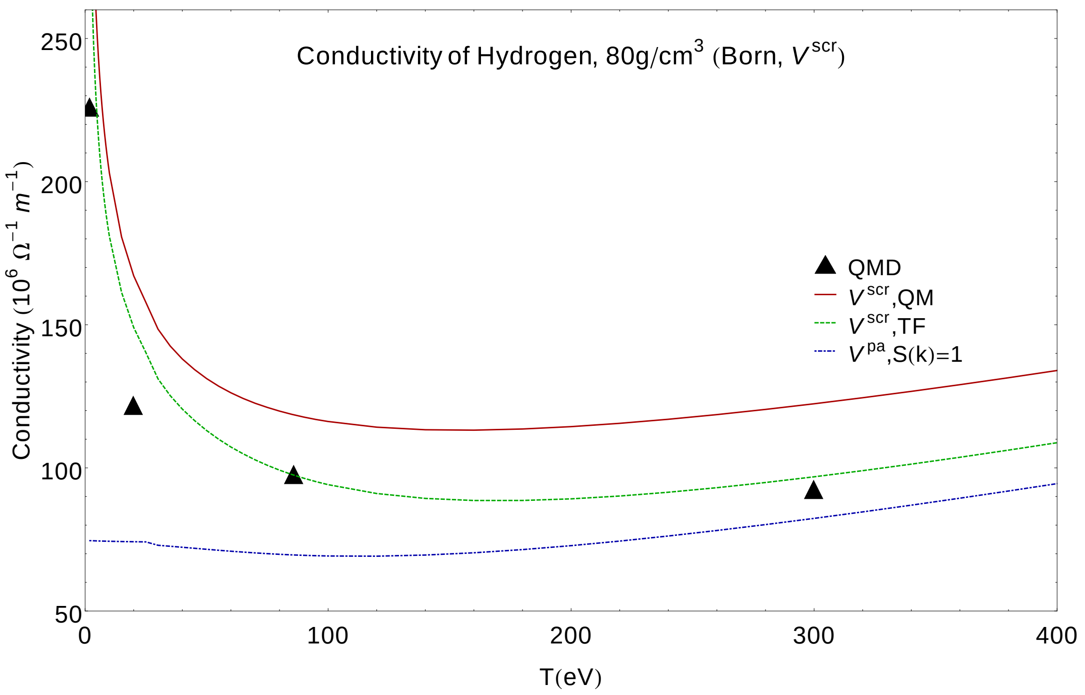


Figure 4: Pseudopotential conductivities of hydrogen at 80g/cm^3. Pseudoatom was better at this density as well.

Finally, we ran the same calculations for hydrogen at 160g/cm^3 and at this density, we continue to see agreement. The pseudopotential now does significantly worse than the pseudoatom potential and still, TF-DFT gives slightly better agreement than KS-DFT. This is likely due to cancellation of errors.

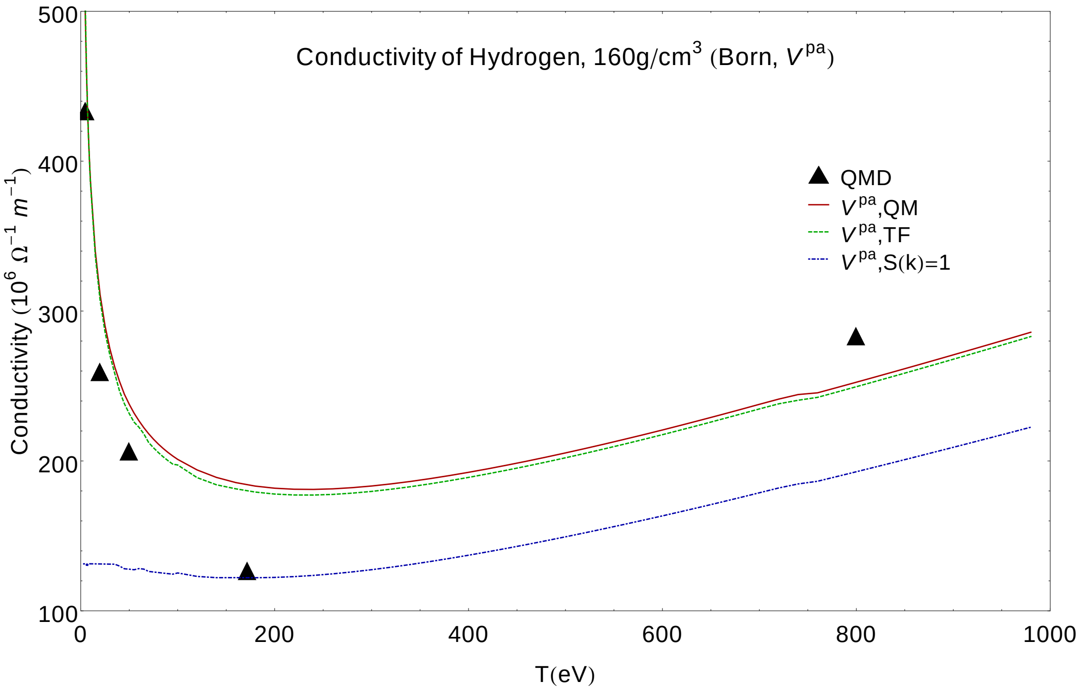


Figure 5: Pseudoatom conductivities of hydrogen at 160g/cm^3. At this density, we begin to see some numerical noise in the form of wobbles at both low and high temperatures.

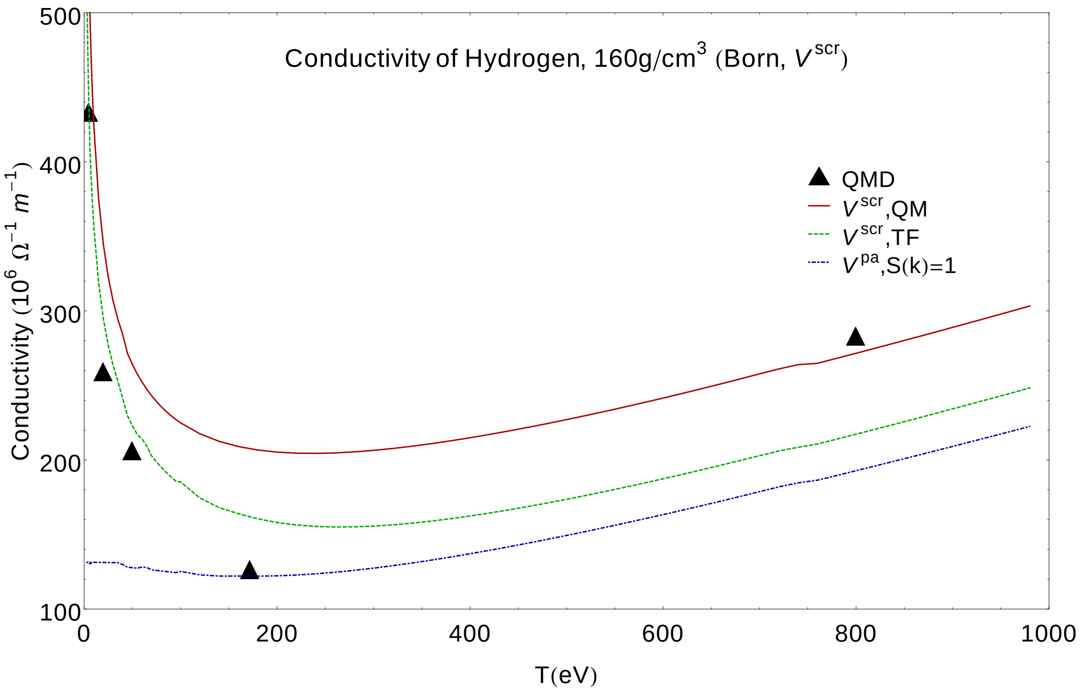


Figure 6: Pseudopotential conductivities of hydrogen at 160g/cm^3.

Beryllium

The calculations run on beryllium were slightly different than the calculations we performed for hydrogen. Instead of varying the density and scanning over a set range of temperatures, we first considered beryllium at a constant density of 10g/cm^3, and then considered beryllium at a constant temperature of 10 eV.

In addition to a constant temperature plot, we also tested a pseudized version of our pseudopotential with beryllium. By artificially removing oscillations in our potential caused by valence electrons, we hoped to improve the accuracy of this model but altering the potential only served to worsen the pseudopotential results.

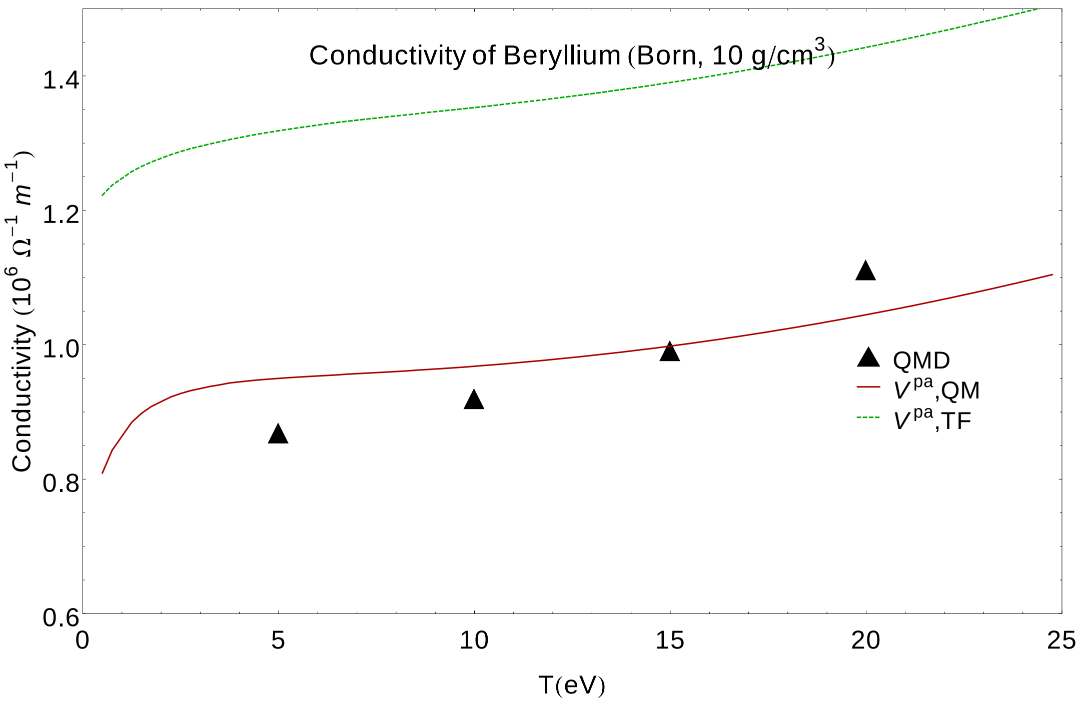


Figure 7: Pseudoatom conductivities of beryllium at 10g/cm^3. It's difficult to say whether or not the drop-off as $T \rightarrow 0$ is physical or not since we don't have QMD data below 5 eV, although we agree with T-Matrix. We now see that TF-DFT fails to get the correct magnitude for the conductivity.

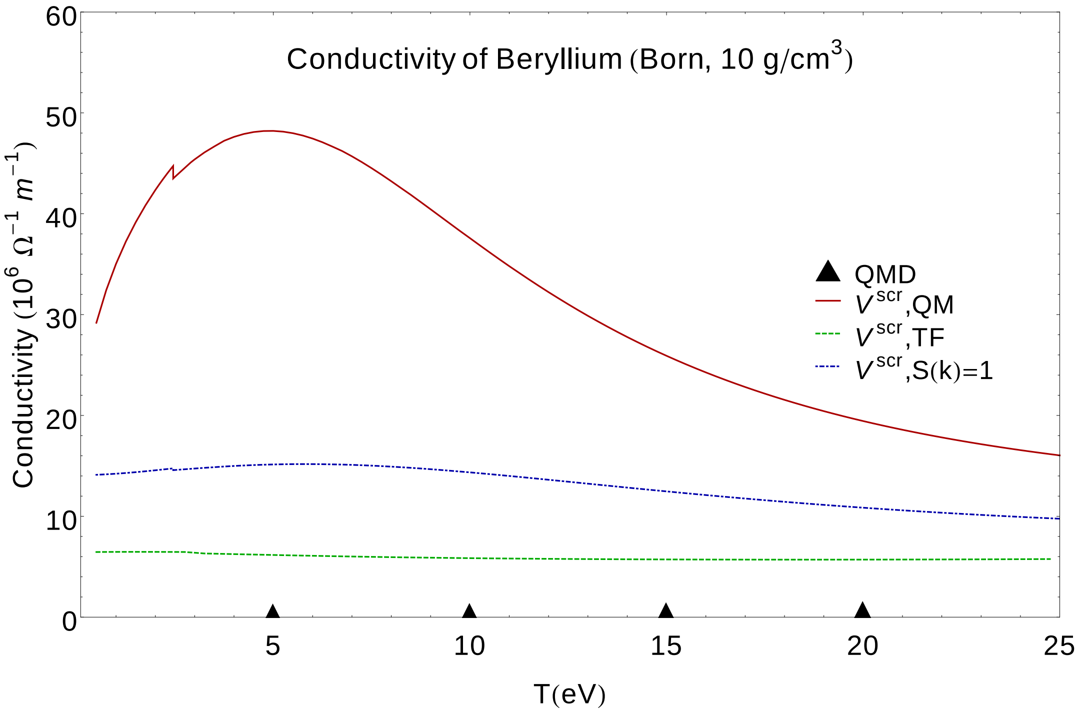


Figure 8: Pseudopotential conductivities of beryllium at 10g/cm^3. Note that the screened potential completely and totally misses the QMD behavior.

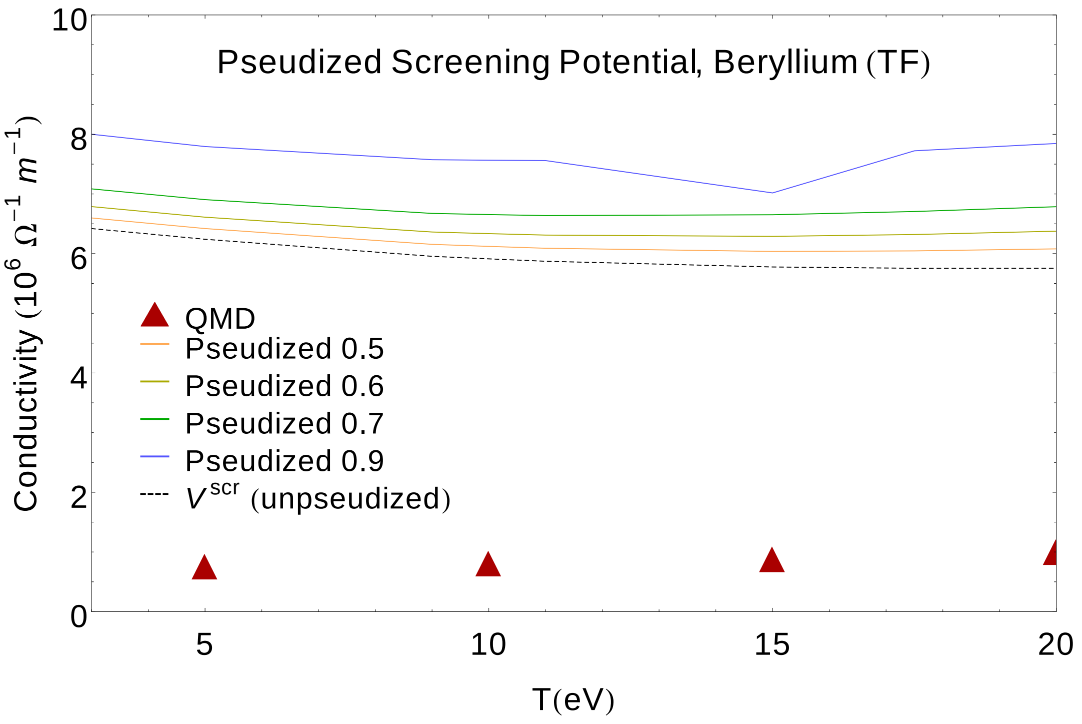


Figure 9: TF Pseudized conductivities of beryllium at 10g/cm^3.

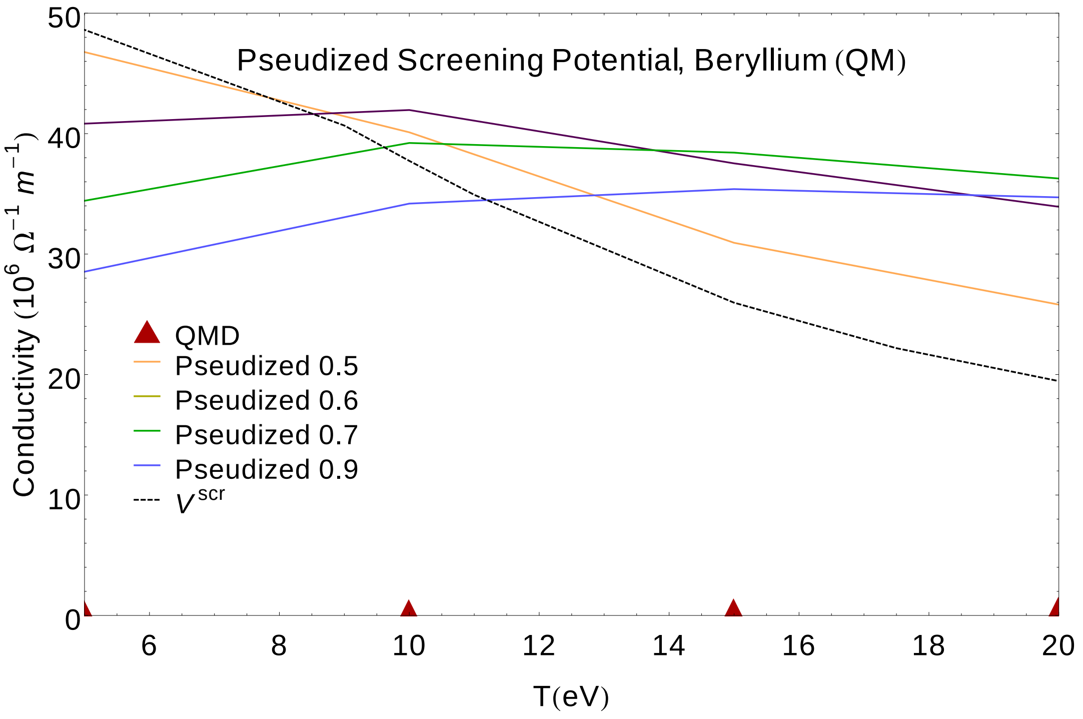


Figure 10: KS Pseudized conductivities of beryllium at 10g/cm^3.

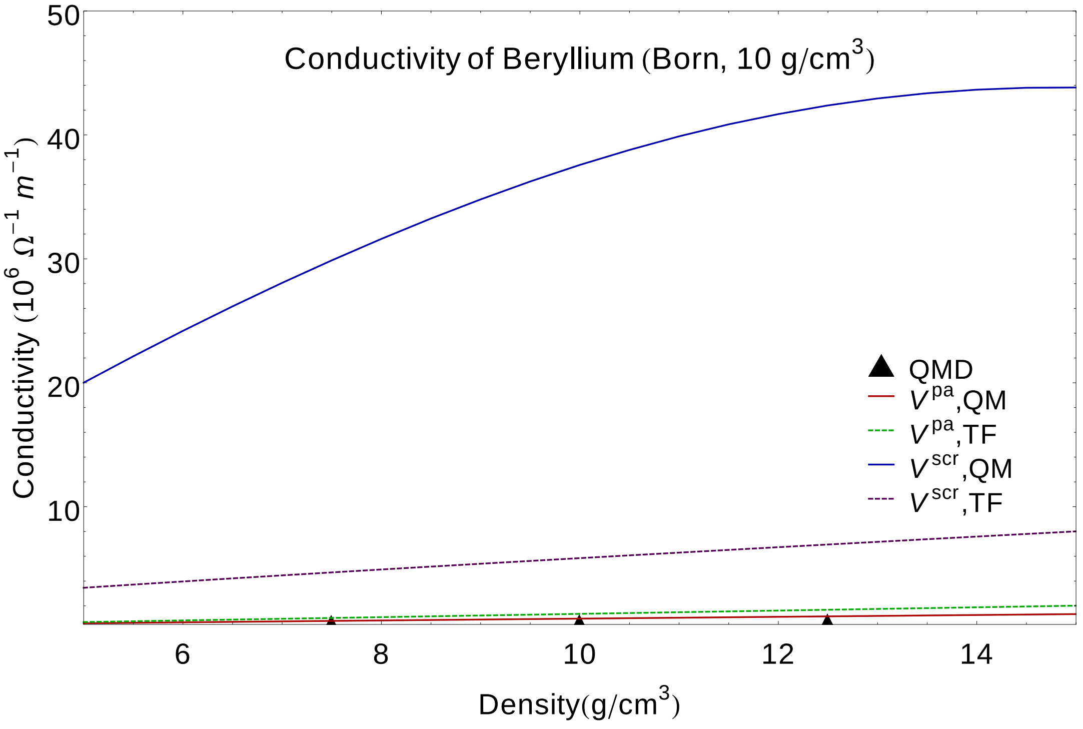


Figure 11: Conductivities of beryllium at 10eV. Again, we see that the pseudopotential fails and pseudoatom is decent.

In figure 7 , we see that the pseudoatom potential is, once again, providing data remarkably similar to QMD data points. We also see that, unlike hydrogen, the KS-DFT calculations are now doing far better than TF-DFT.

In the following figures, the pseudopotential provides a significant discrepancy to QMD data. We also observe some rather unphysical behavior in the KS-DFT conductivities. Pseudization of this potential doesn't help the situation and further suggests that the pseudopotential is just not a good model in tandem with the Born approximation.

In the final plot, we see that all of the previous trends apply as density is varied as opposed to temperature.

Aluminum

In the case of aluminum, we ran simulations similar to hydrogen. Again, a test involving the structure factor is carried out to see how much the Born approximation depends on it. We also pseudize the pseudopotential as with beryllium, though again, this only serves to worsen our data.

In addition to these runs, we also calculated the resistivity of aluminum at three and five times the normal solid density of aluminum.

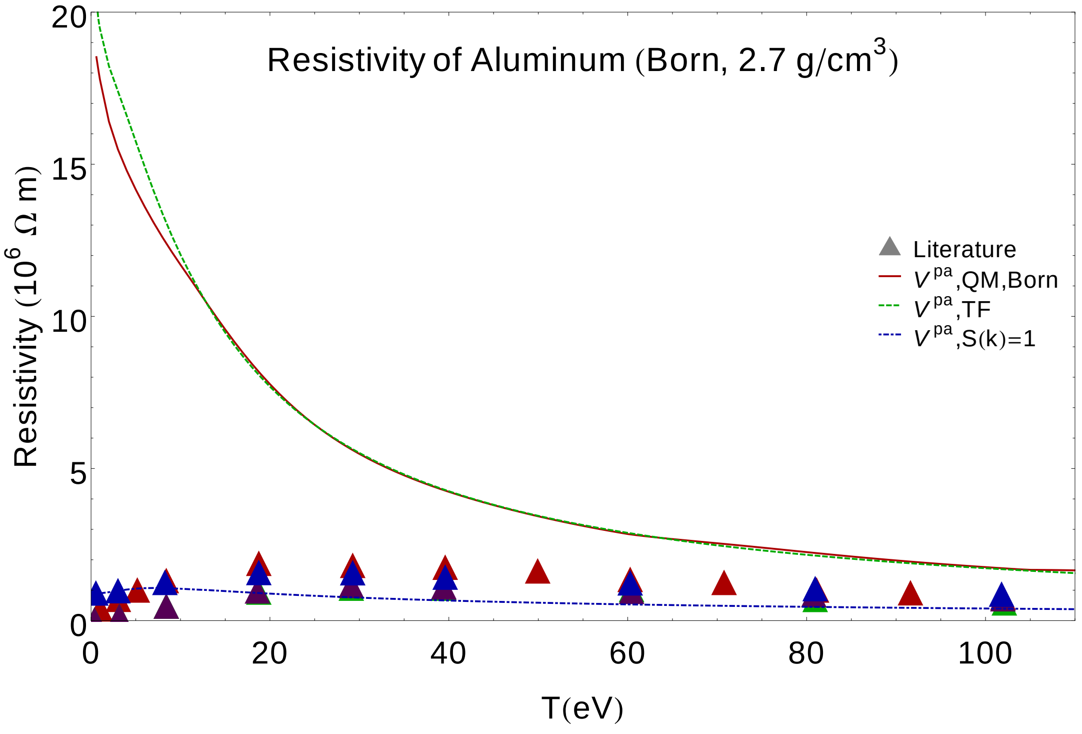


Figure 12: Pseudoatom conductivities of aluminum at 2.7g/cm^3. This plot demonstrates quite clearly that there are systems which simply cannot be modeled by the Born approximation. We see behavior at low temperature that disagrees both qualitatively and quantitatively with other literature calculations. We also see that, as $T \rightarrow \infty$, we do approach literature values.

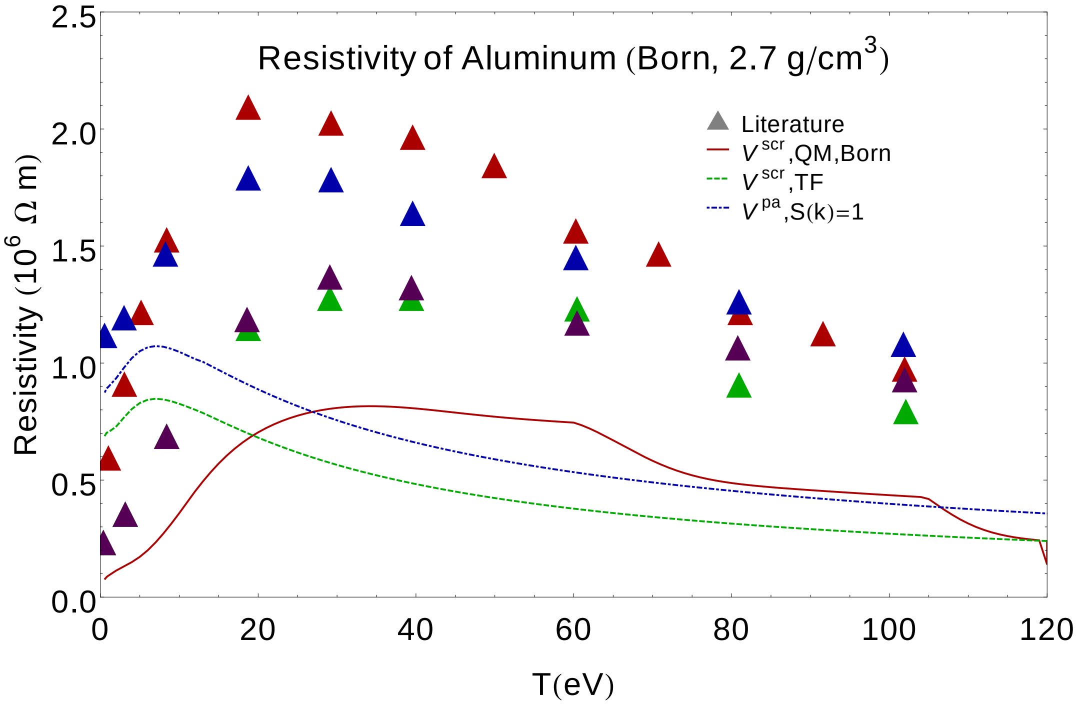


Figure 13: Pseudopotential conductivities of aluminum at 2.7g/cm^3. This result is misleading. It looks likes the pseudopotential is doing better than the pseudoatom potential but we believe this to be a cancellation of errors. We also see some really strange behaviors in the KS-DFT calculations.

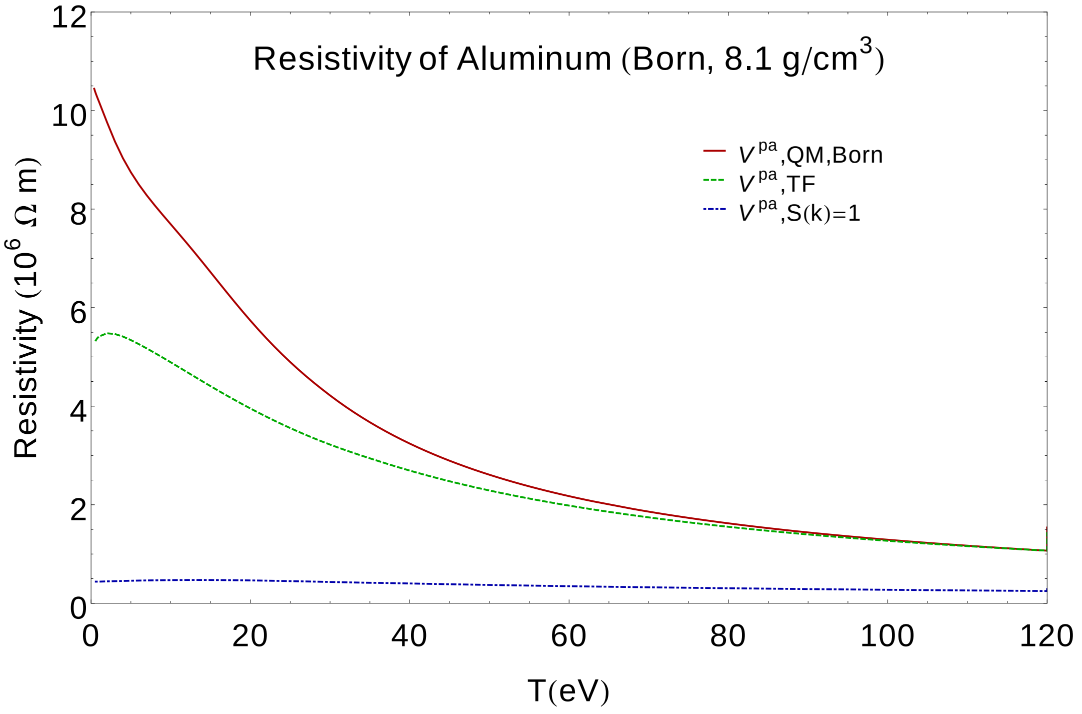


Figure 14: Pseudoatom conductivities of aluminum at 8.1g/cm^3.

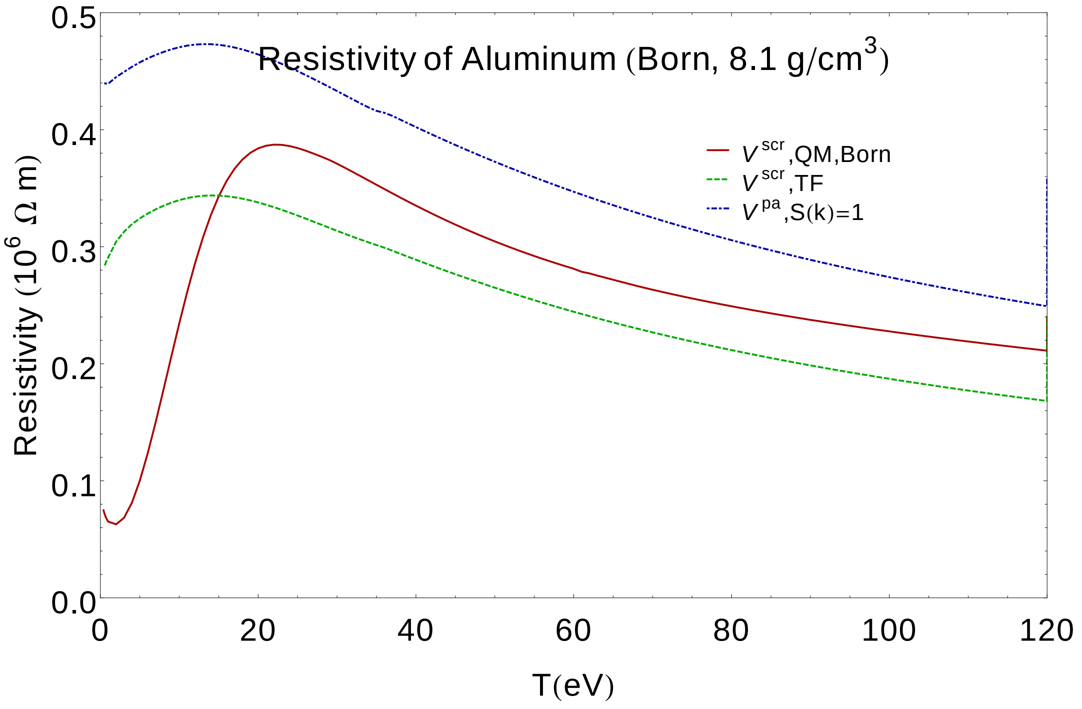


Figure 15: Pseudopotential conductivities of aluminum at 8.1g/cm^3.

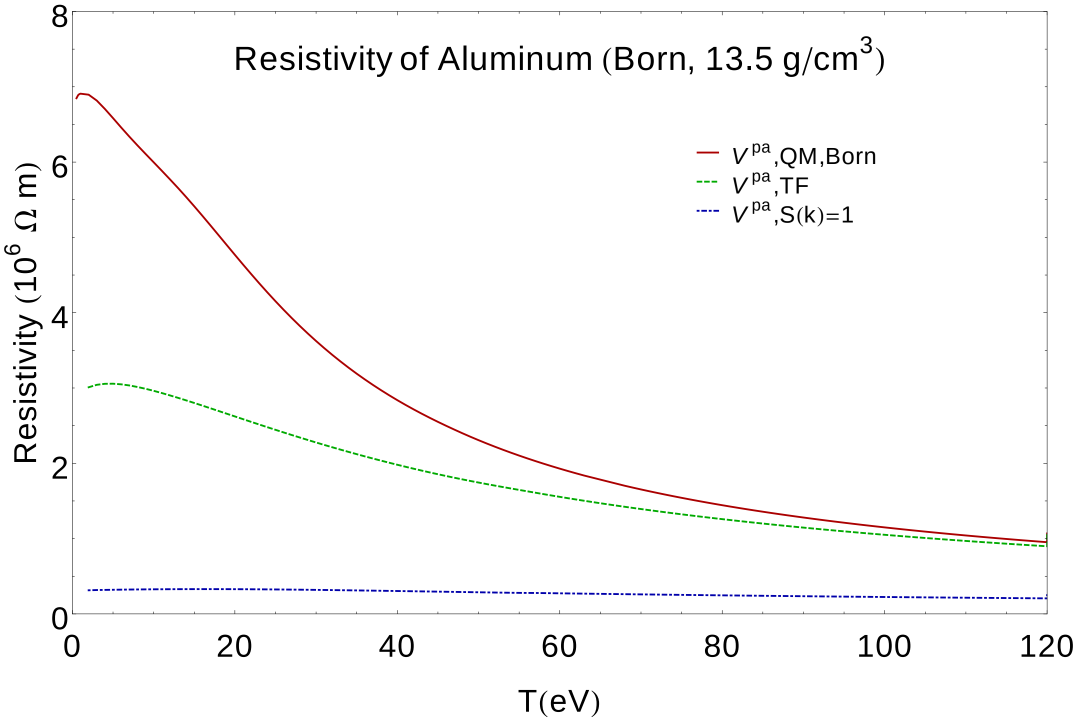


Figure 16: Pseudoatom Conductivities of aluminum at 13.5g/cm^3.

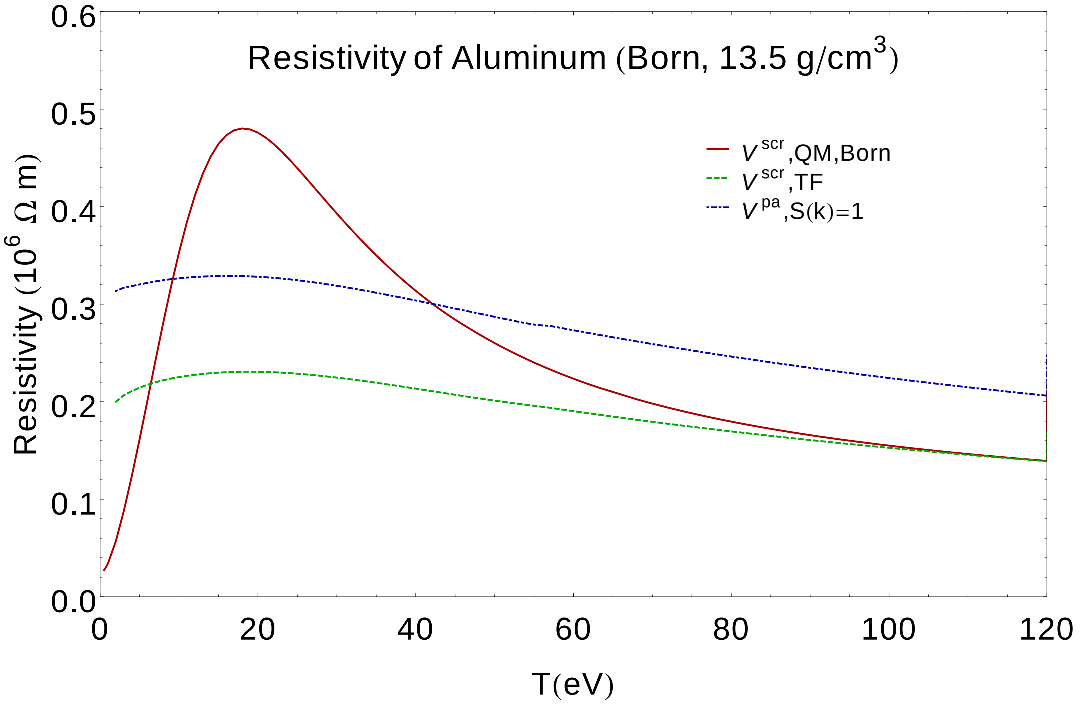


Figure 17: Pseudopotential Conductivities of aluminum at 13.5g/cm^3.

We see some very strange results for aluminum. First, the pseudoatom model utterly fails at lower temperatures for aluminum. This is most probably caused by the Born approximation since scattering within aluminum almost definitely involves higher-order scattering events which the Born approximation does not take into account. In addition to this, we see that the pseudopotential data is somehow quite close to other literature values. However, this is most likely due to a cancellation of errors.

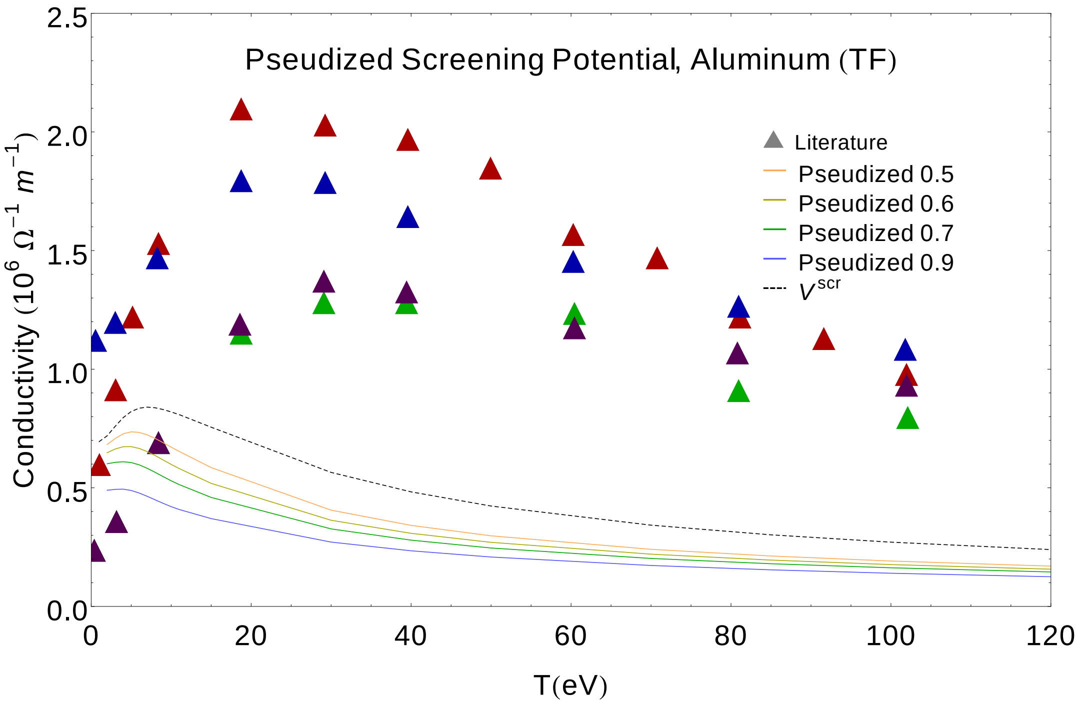


Figure 18: TF Pseudized Conductivities of aluminum at 2.7g/cm^3. As with beryllium, pseudization further decreases agreement with other calculations.

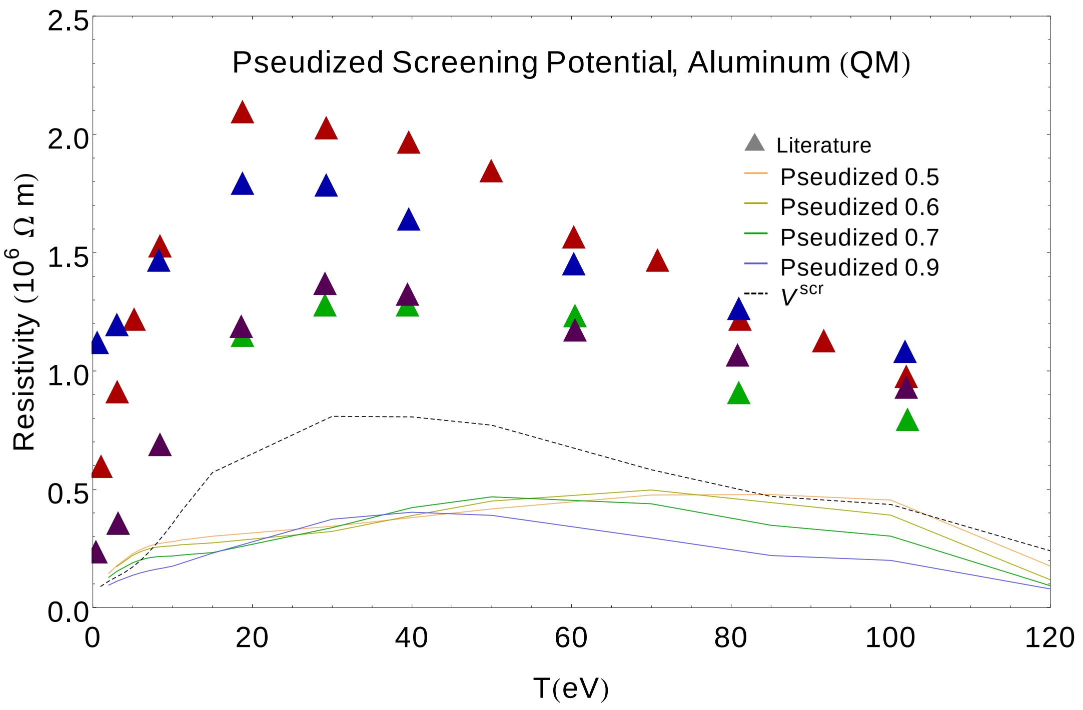


Figure 19: KS Pseudized Conductivities of aluminum at 2.7g/cm^3.

As mentioned previously, pseudization of our potential only serves to worsen our results. For aluminum, we not only see further deviation from the data present in literature, but pseudizing the potential also leads to a change in behavior of the resistivity of our system.

T-Matrix

Hydrogen

The hydrogen atom is the stereotypical benchmark element to use in warm dense matter calculations. There are numerous QMD calcualtions to compare to which makes benchmarking and testing code convenient.

V_{tcp}^{PA} vs. V_{aa}^{AA}

Figs. 20-22 present the calculated conductivites within the PA and AA models with their respective parameters compared to QMD data taken from literature at three different densities. Here we see that in general the PA model agrees much better with the QMD data while the AA model consistently overestimates the conductivities. At low temperatures both models predict a sharp rise in the conductivity consistent with QMD data, but quantitatively both models significantly overestimate this value at least for the $10g/cc$ case. Furthermore, at higher temperatures for the lowest density, the PA model tends to diverge from the trend while the $80g/cc$ and $160g/cc$ trends are in agreement at higher temperatures. The lack of agreement at these higher temperatures could be due to the system no longer residing in the high degeneracy regime indicating a transition into a low density plasma where simulations with QMD become difficult.

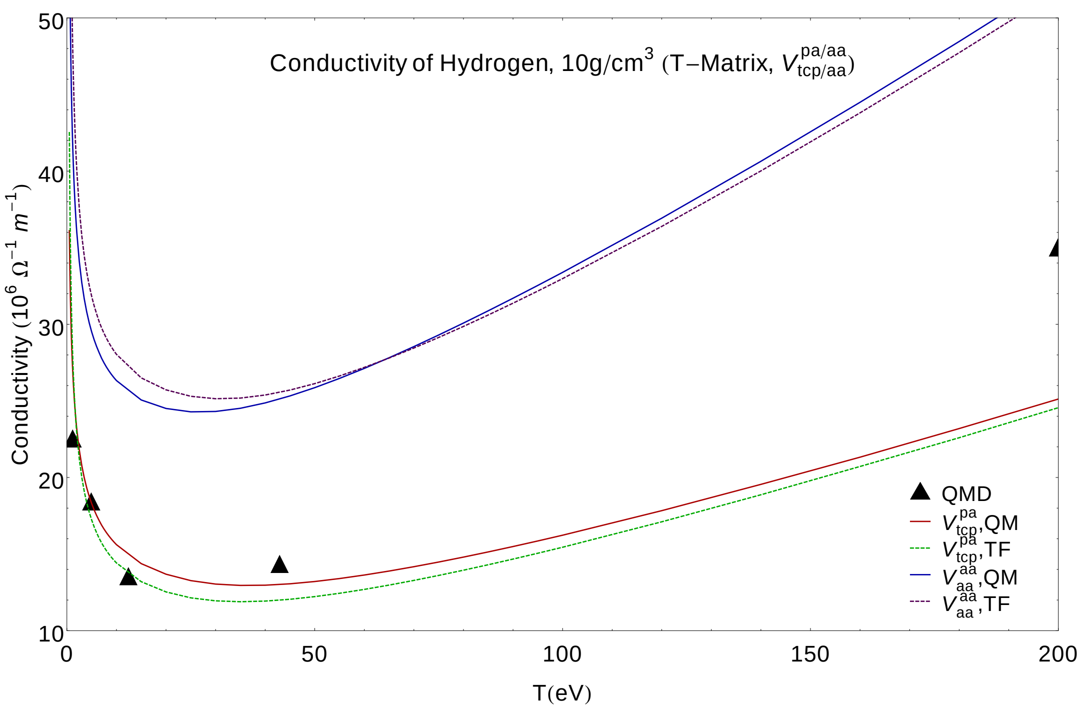


Figure 20: Hydrogen conductivity, $10g/cc$, V_{tcp}^{PA} vs. V_{AA}^{AA}. The PA calculations better follow the trend at lower temperatures, but both AA and PA trend differently from the QMD values at higher temperatures.

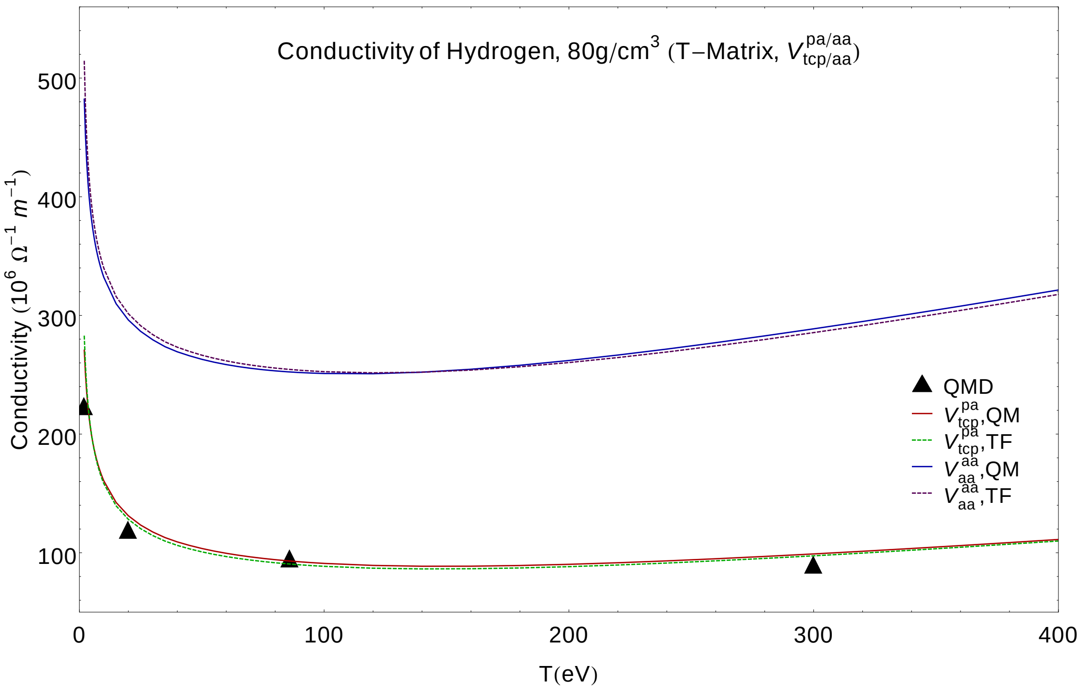


Figure 21: Hydrogen conductivity, $80g/cc$, V_{tcp}^{PA} vs. V_{AA}^{AA}. The PA calculations follow the QMD values very well compared with AA. Even at higher temperatures PA continues to follow the trend while AA clearly trends higher.

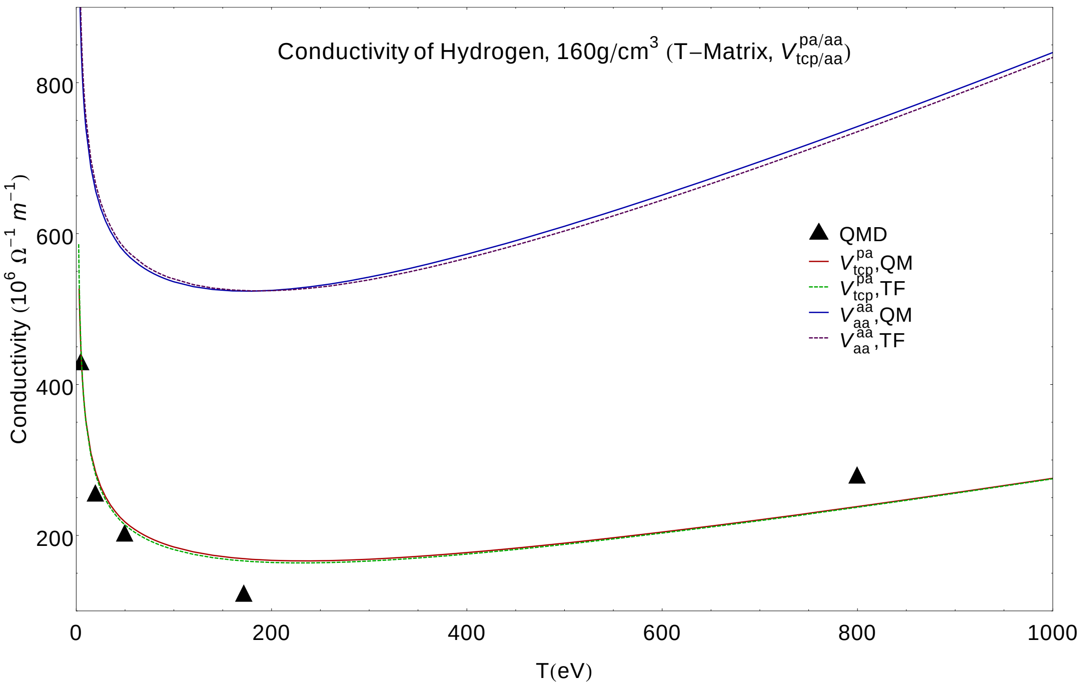


Figure 22: Hydrogen conductivity, $160g/cc$, V_{tcp}^{PA} vs. V_{AA}^{AA}. The PA calculations follow the QMD data decently. The turning point is off but the high temperature trend is recovered while AA continues to provide quantitatively incorrect values.

V_{aa}^{PA} vs. V_{aa}^{AA}

Historically, models of WDM systems have relied on simplicity and comparitive ease of analytical solutions to describe the physics in these systems. The one-component plasma model is one such approach where ions are treated as fully ionized residing within a background of fixed negative charge. The fixed background represented the plasma component and accounted for screening interactions between ions, but did not possess a dynamic component hence only the ions (one-component) were considered with any amount of physical argument. More sophisticated models emerged such as the average atom (related to one-component) and later the two-component models. The AA model draws from the one-component model by defining a background charge between atoms, but expands on it by defining a cutoff radius within which the atoms are treated with orbitals quantum mechanically (not fully ionized).

Presented in Figs. 23-25 are the results from the PA and AA models with the parameters chosen from the AA model. This helps to explain the relation of these parameters by showing how they reflect on the physics of the system. Two feature emerge: The first is that, when PA is coupled with AA parameters, the conductivities are lowered. Second, for the $10g/cc$ case the TF and QM values are in better agreement.

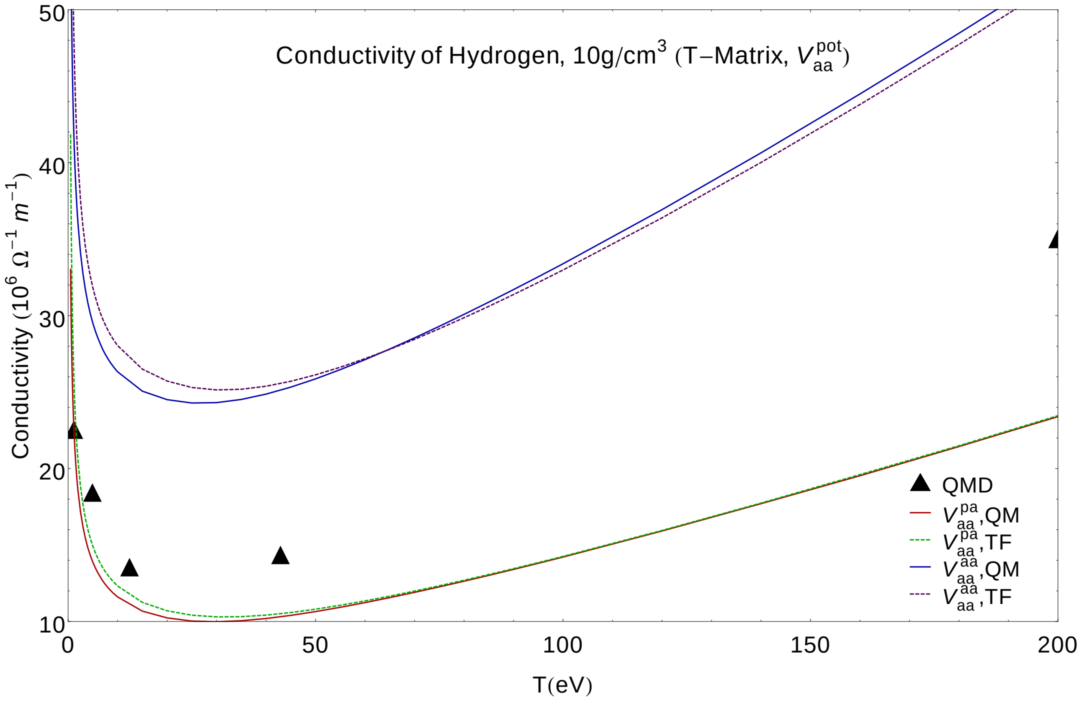


Figure 23: Hydrogen conductivity, $10g/cc$, V_{AA}^{PA} vs. V_{AA}^{AA}. The use of AA parameters in the PA calculation yield lower values than when using tcp.

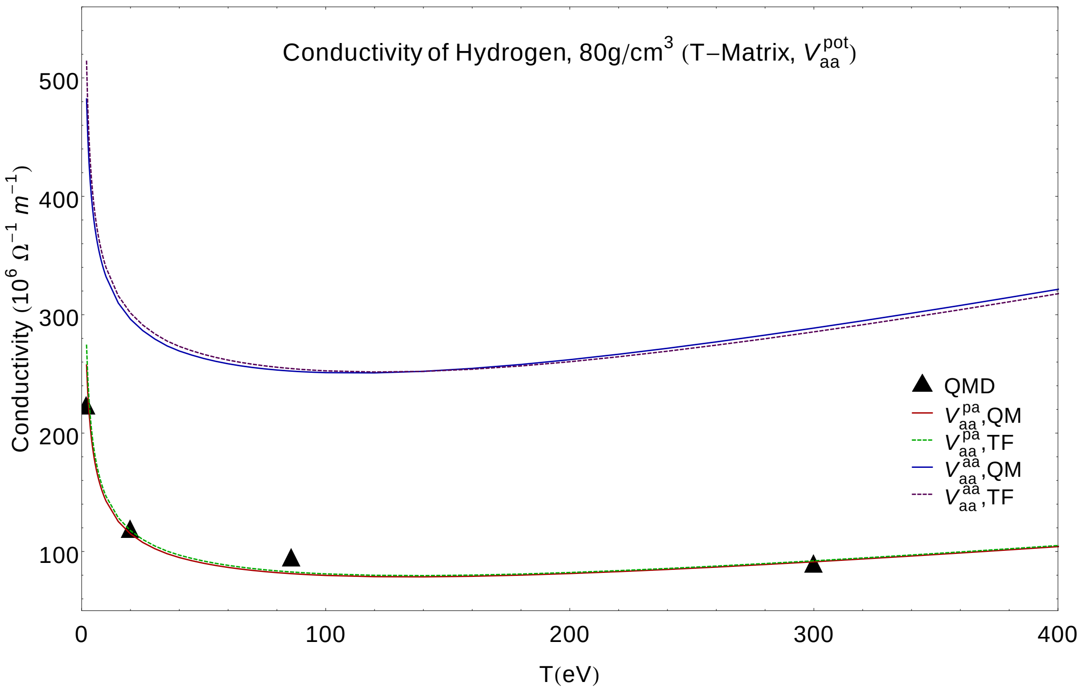


Figure 24: Hydrogen conductivity, $80g/cc$, V_{AA}^{PA} vs. V_{AA}^{AA}. The use of AA parameters in a PA calculation at this density does not greatly affect the results.

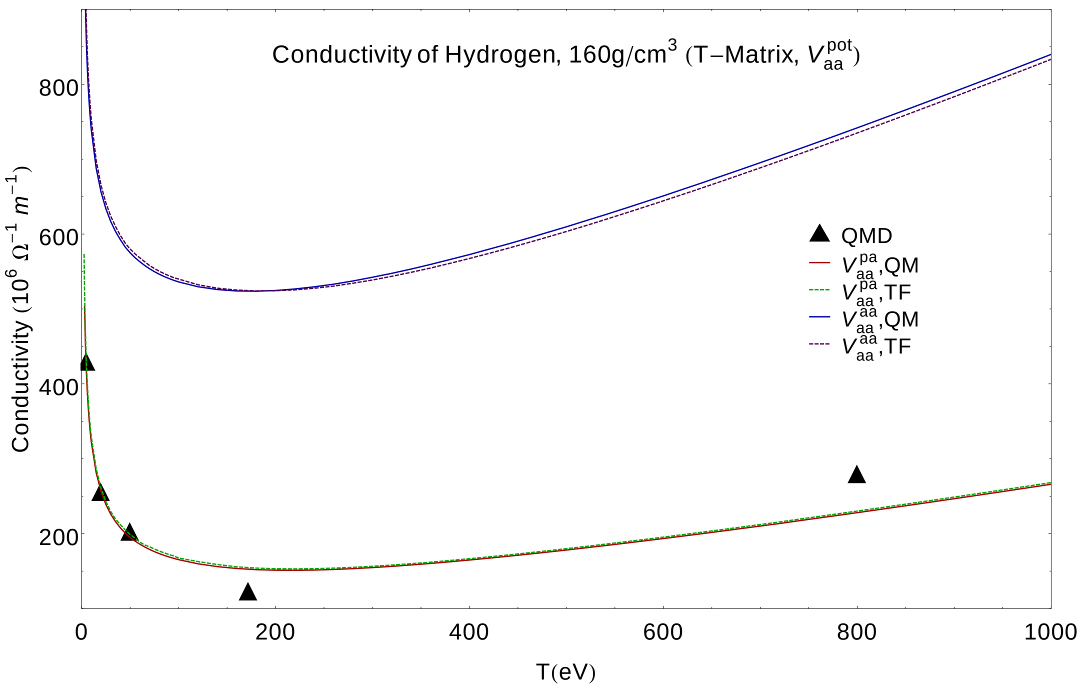


Figure 25: Hydrogen conductivity, $160g/cc$, V_{AA}^{PA} vs. V_{AA}^{AA}. The use of AA parameters in a PA calculation at this density does not greatly affect the results.

$S(k)$ input vs. $S(k) = 1$

Integral to understanding interactions within WDM is an accurate understanding of ion-ion correlations. These correlations drive the distributions of both ions and ionized electrons through their atomic potentials. A quatitiy of interest to describe such a phenomenon is the structure factor. This intrinsic property of a (numerically converged) system gives the distribution of ions from some chosen ion which effectively describes the presence of order as a series of peaks or disorder as a constant value. The role of the structure factor in calculating transport properties was outlined by Ziman as being crucial to accurately describing scattering processes. In Figs. 26-28 the difference between a calculation with the actual structure factor (ordered system)and one without the structure factor (disorder system) are shown. The expectation for such as test is to find the temperature-density regions where a disordered system may be considered. In all density regimea we see a relatively decent qualitative agreement at higher tempertures while the values are lower temperatures trend differently. This temperature related effect may be due to the H atoms prefering a structured state while a disordered state is favorable at higher temperatures.Ultimately, this suggests that the structure factor at lower energy scattering must be included to calculate comparable values to QMD.

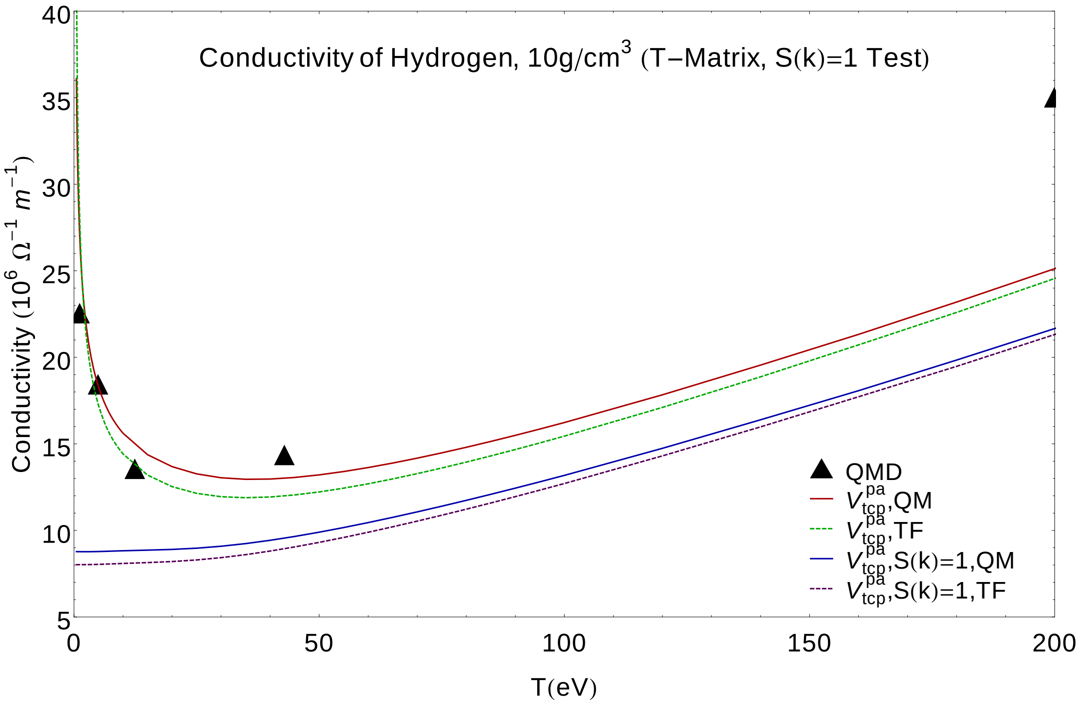


Figure 26: Hydrogen conductivity, $10g/cc$, V_{tcp}^{PA} with $S(k)$ from input vs. $S(k) = 1$. The structure factor is shown to be important especially at low temperatures where there are qualitative differences. Through all temperatures, the S(k)=1 calculation produced lower values.

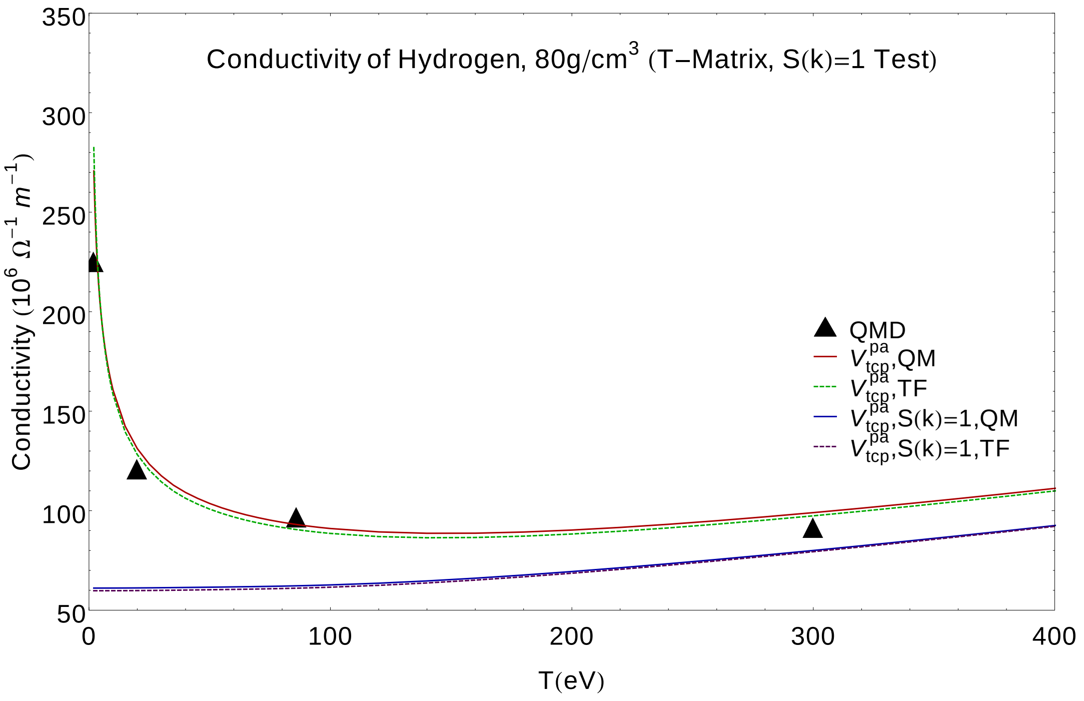


Figure 27: Hydrogen conductivity, $10g/cc$, V_{tcp}^{PA} with $S(k)$ from input vs. $S(k) = 1$. The structure factor is shown to be important especially at low temperatures where there are qualitative differences.

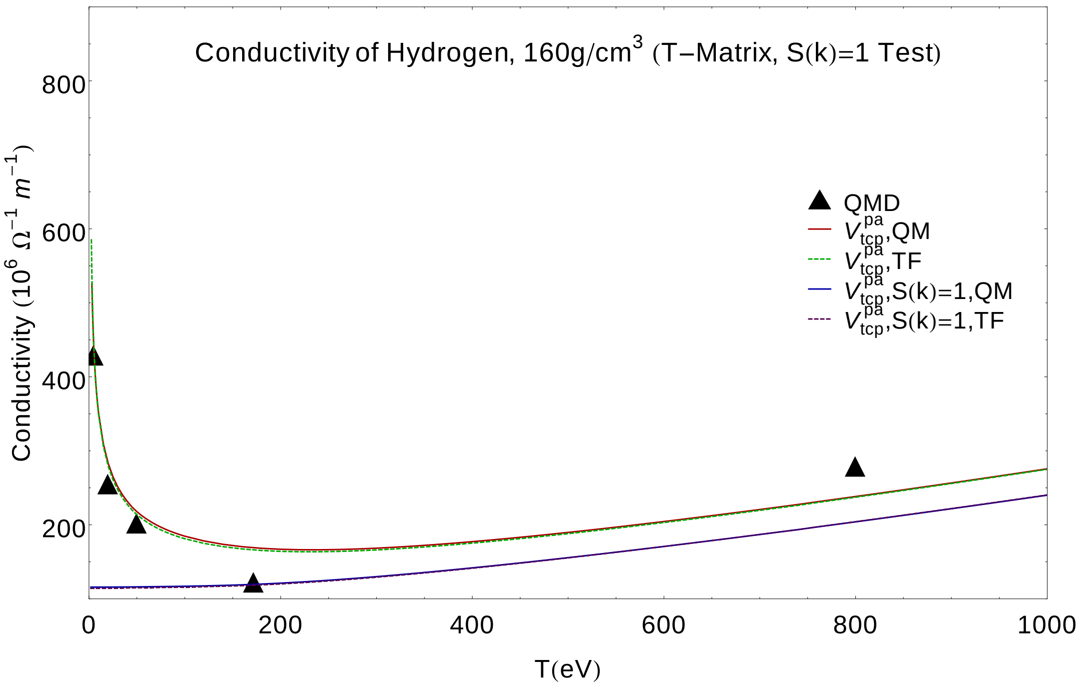


Figure 28: Hydrogen conductivity, $10g/cc$, V_{tcp}^{PA} with $S(k)$ from input vs. $S(k) = 1$. The structure factor is shown to be important especially at low temperatures where there are qualitative differences.

Beryllium

Constant Density

The following figures show the conductivities for Beryllium calculated with the PA and AA models at constant density (Fig. 29) and temperature (Fig. 30). For both figures, the conductivities calculated with the PA model give relatively good QM results when compared to the AA model. An important distinction between these plots and those of Hydrogen are that the QM and TF models no longer agree and for lower temperatures they are qualitatively different.

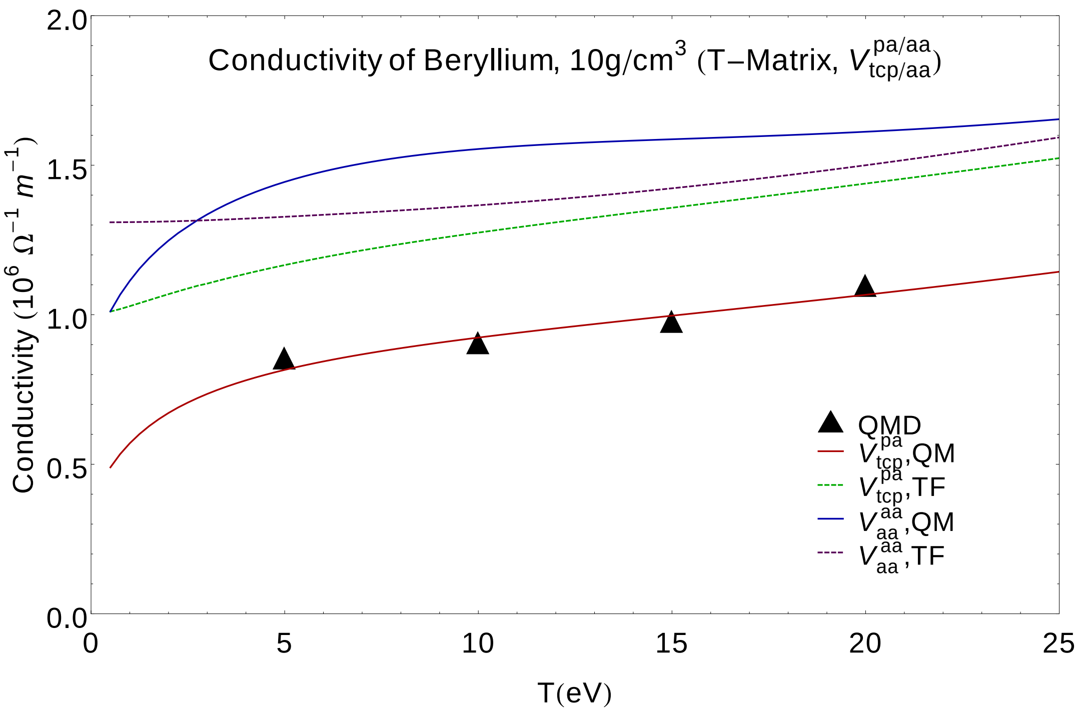


Figure 29: Beryllium conductivity, $10g/cc$, at constant density. PA calculations with KS are the most similar to QMD data. PA calculations with TF are qualitatively similar but yield larger values. AA calculations are all much higher than the QMD values with qualitatively similar values.

Constant temperature

The data collected for beryllium at constant temperature is shown in Fig. 30. Similar to the constant density values, Fig. 29 we see that the PA data correctly follows the QMD data whereas the AA model tends to overestimates the values. Addtionationally, TF calculations show similar trends to their KS counterparts but are higher for the PA case and slightly lower than KS with AA.

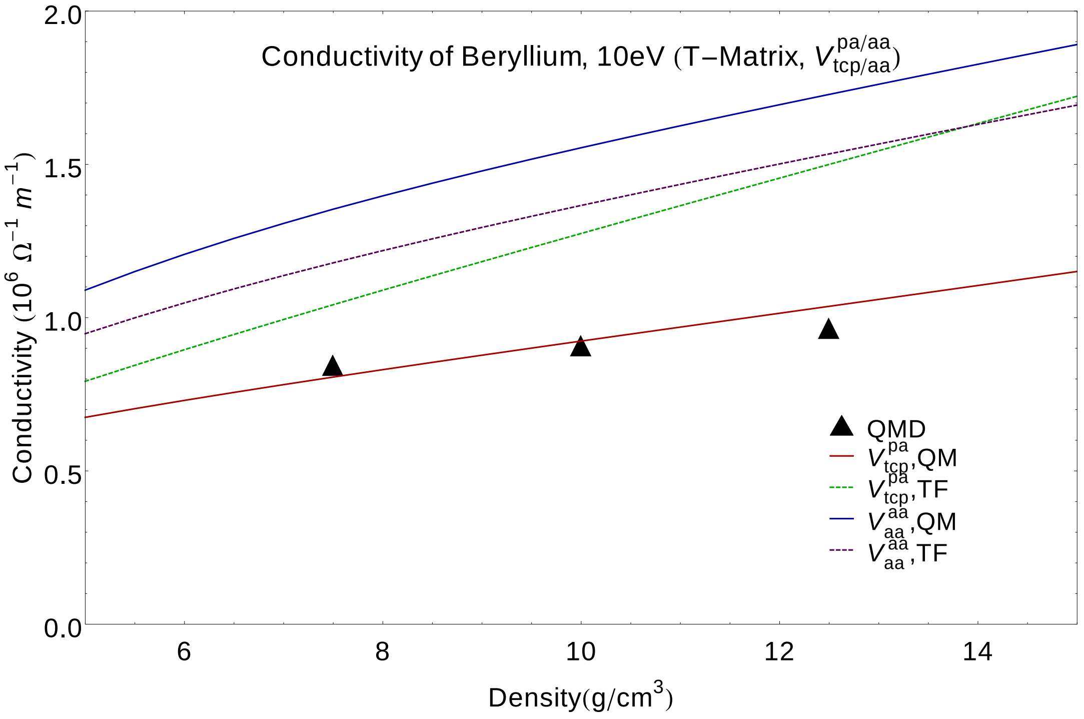


Figure 30: Beryllium conductivity, $10eV$, at constant temperature. PA calculations with KS are the most similar to QMD data. PA calculations with TF are qualitatively similar but yield larger values. AA calculations are all much higher than the QMD values with qualitatively similar values.

Aluminum

Similar to the hydrogen cases, shown here are several comparisons for aluminum. The comparisons examine the differences between the V^{PA} and V^{AA} potentials, the role of the model parameters, and finally how the structure factor impacts the calculations at lower temperatures. As a means of comparison, there exist only literature values calculated with V^{AA} or V^{scr} potentials without any QMD data. Therefore, we rely on the fact that the T-matrix approach with V^{PA} has been shown to be accurate with respect to QMD data in both hydrogen and beryllium.

V^{PA}_{tcp} vs. V^{AA}_{aa}

The resistivities for three different densities of aluminum are shown in Figs. 31-33. At the lowest density where there exists literature data to compare to we see the difference between V^{PA} and V^{AA}. As expected, the V^{AA} fit literature values well whereas the V^{PA} approach yields a quantitative agreement at higher temperatures but shows strong deviations at lower temperatures. Additionally, at least for the lower density the KS values are close to TF for both V^{PA} and V^{AA}. This trend does not continue at higher densities where the TF values diverge from KS for V^{PA}. The V^{PA} for all densities are consistently larger than their V^{AA} counterparts. Features at lower temperatures emerge which are due to the inclusion of the structure factor and will be discussed later.

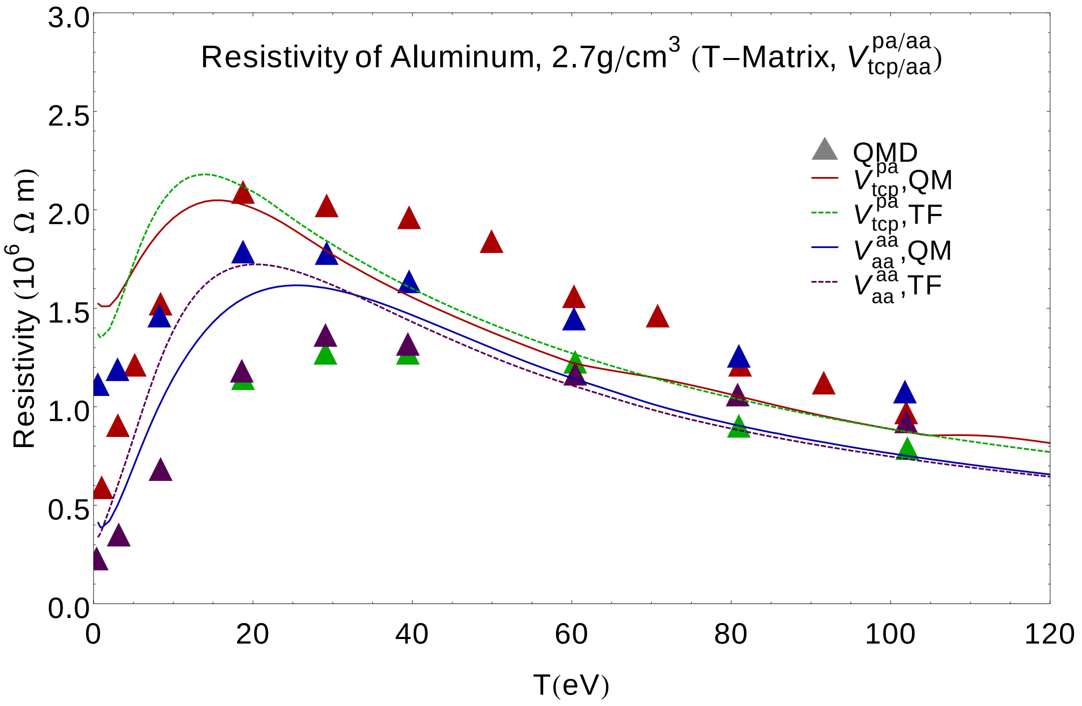


Figure 31: Aluminum resistivity, $2.7g/cc$, V_{tcp}^{PA} vs. V_{aa}^{AA}. AA calculations fit the literature AA values well. PA calculations are similar for high temperatures but a low temperature feature stands out. This is due to the structure factor difference between PA and AA.

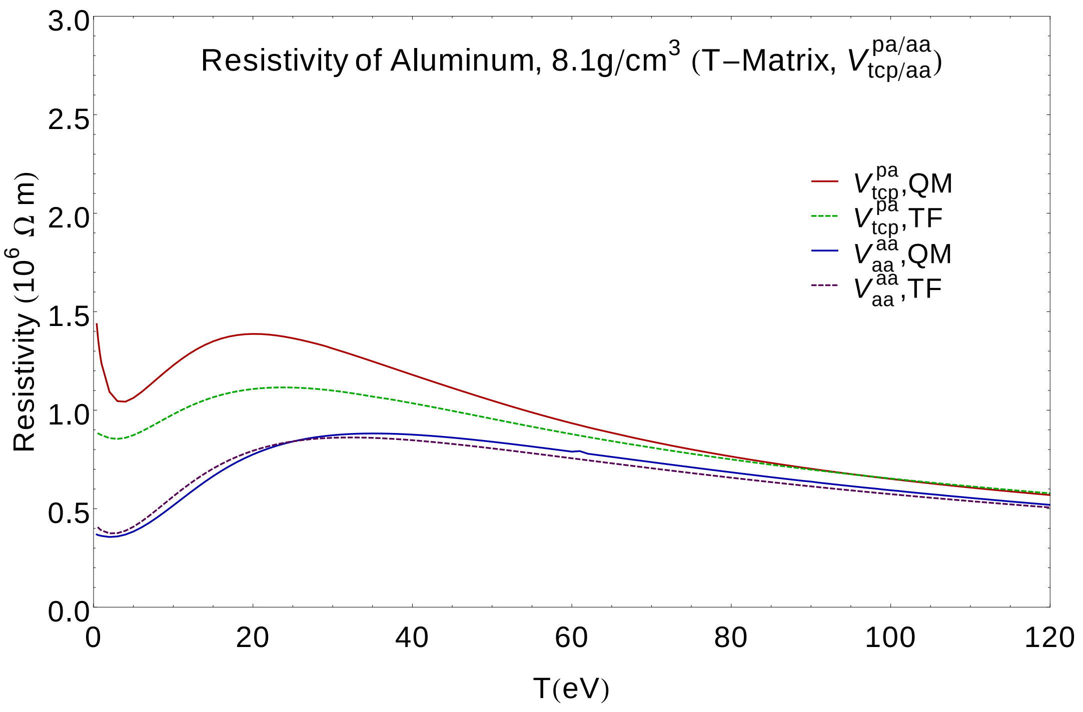


Figure 32: Aluminum resistivity, $8.1g/cc$, V_{tcp}^{PA} vs. V_{aa}^{AA}. There are low temperature features similar to the solid density case. However, the TF value is beginning to differ substantially from the KS for PA calculations.

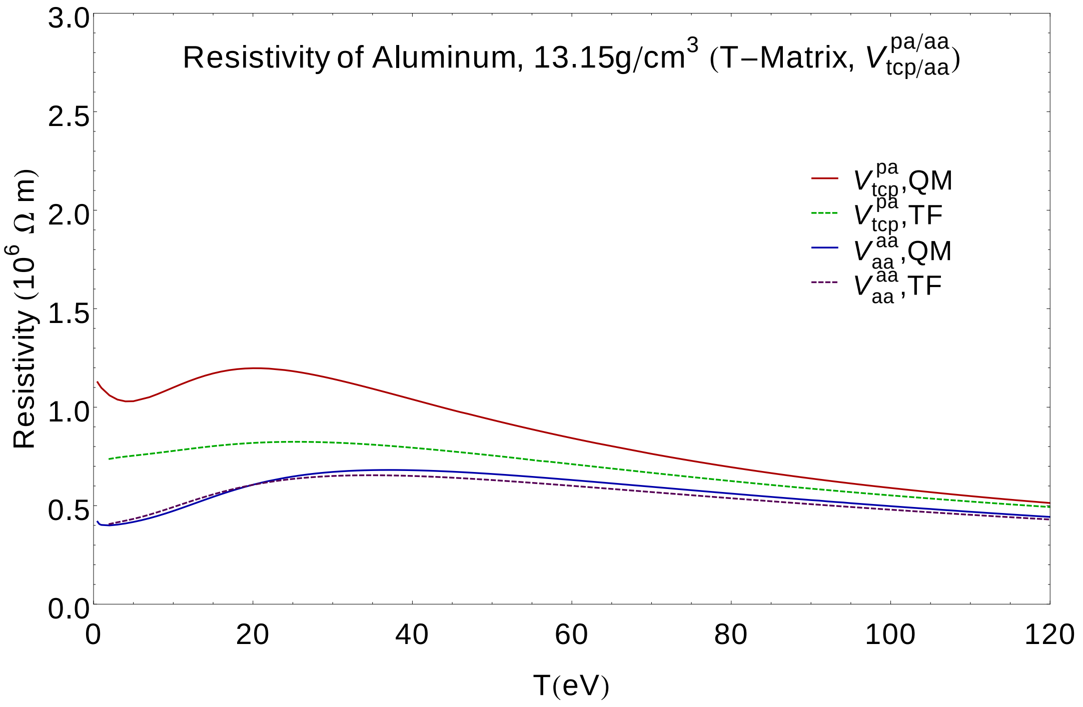


Figure 33: Aluminum resistivity, $13.5g/cc$, V^{PA}_{tcp} vs. V^{AA}_{aa}. There are low temperature features similar to the solid density case. However, the TF value is substantially different from the KS for PA calculations.

V^{PA}_{aa} vs. V^{PA}_{tcp}

To understand the difference that using different values for the electron density and chemical potential make, Fig. 34 shows the difference between a PA calculation using AA and tcp parameters. Two differences emerge: The first is that at lower temperatures the aa values push the resistivity higher while retaining the same general features. Since the overall behavior does not change, this must be a contribution of the PA model rather than the parameters chosen. The second feature is the inclusion of several ‘steps’ in the tcp resistivity compared to the aa values. This is due to a different method of treating ionization between the parameters.

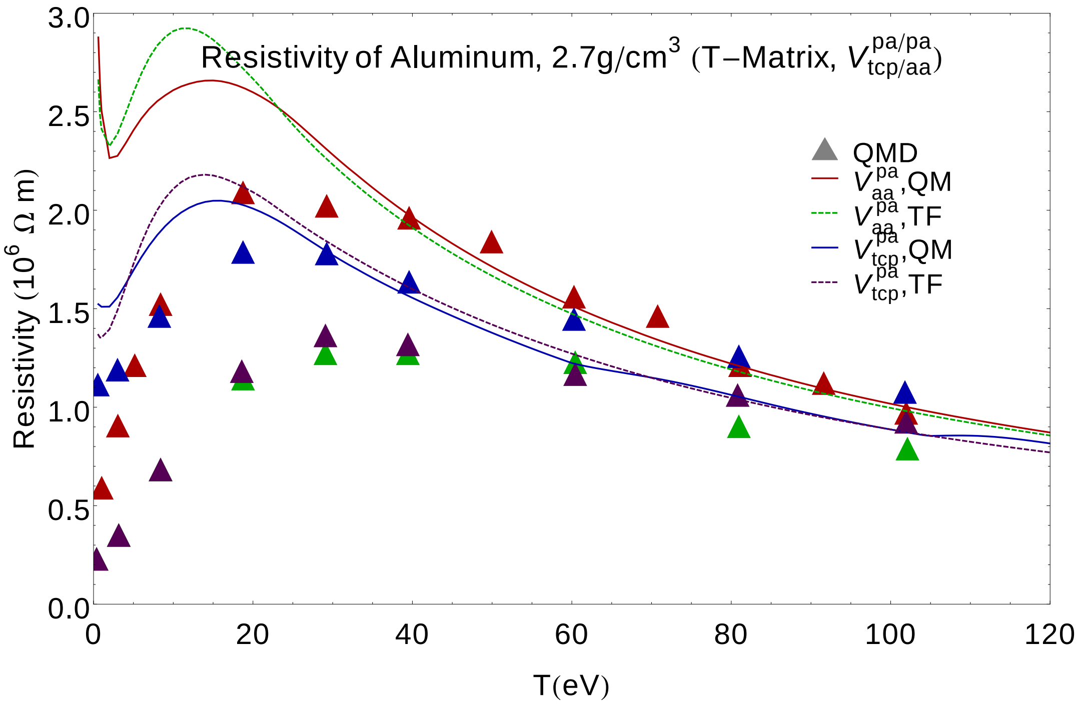


Figure 34: Aluminum $2.7g/cc$, V^{PA}_{aa} vs. V^{PA}_{tcp}. The use of AA parameters increases the resistivity while preserving low temperature features.

$S(k)$ **input vs.** $S(k) = 1$

The low temperature feature in all of the densities are shown to be due to the structure factor in Figs. 35-37. These figures show the comparison between the normal V^{PA} runs and V^{PA} with the structure factor set to one. Where the normal calculations show an uptick in the resistivity at the lower end of the temperatures studied, the $S(k) = 1$ values go to a local minimum. Therefore, the rise must be due to the ion-ion correlation changing near those temperatures.

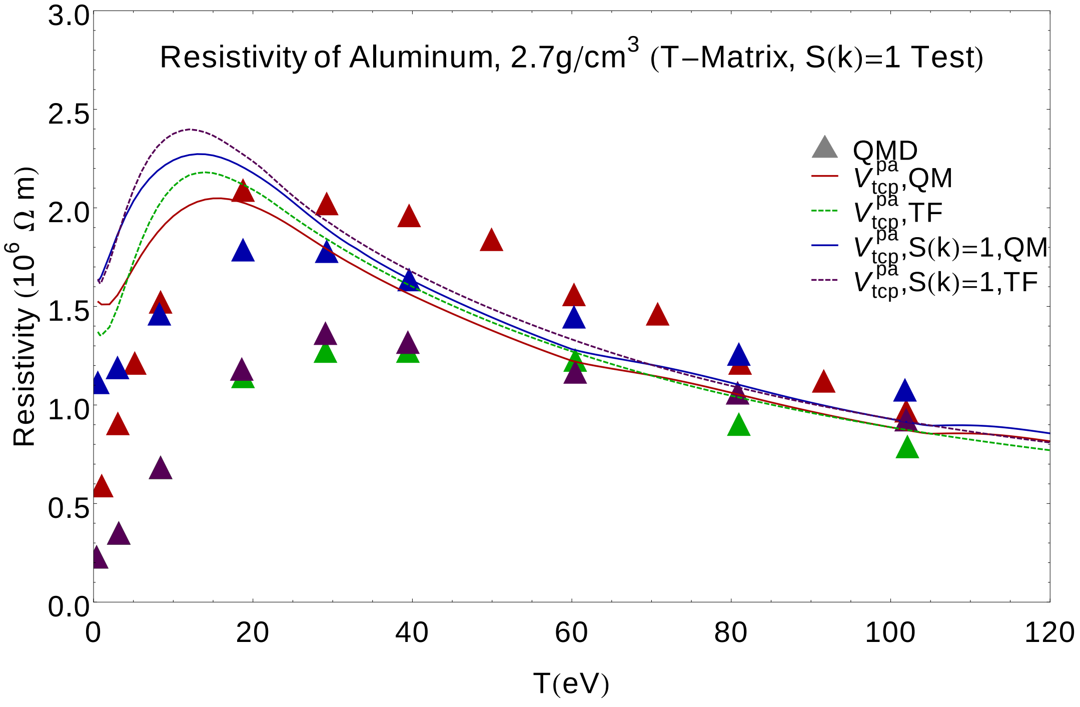


Figure 35: Aluminum resistivity, $2.7g/cc$, $S(k)$ input vs. $S(k) = 1$. The use of S(k)=1 shows that the low temperature features depend on the structure factor.

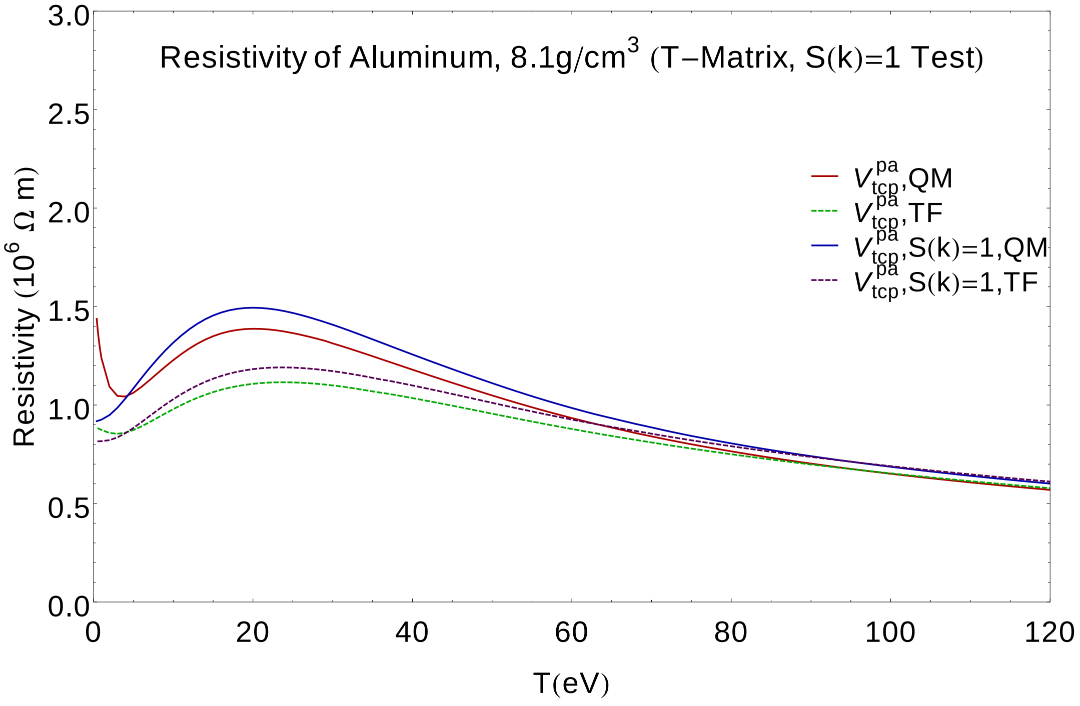


Figure 36: Aluminum resistivity, $8.1g/cc$, $S(k)$ input vs. $S(k) = 1$. The use of S(k)=1 shows that the low temperature features depend on the structure factor.

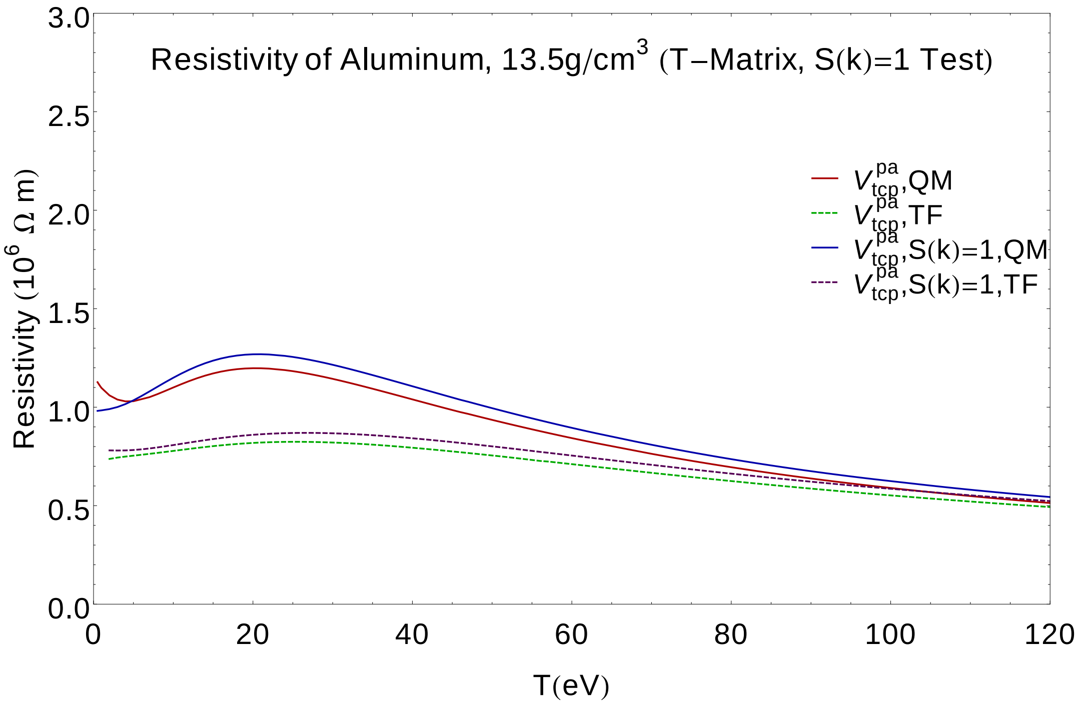


Figure 37: Aluminum resistivity, $13.5 g/cc$, $S(k)$ input vs. $S(k) = 1$. The use of S(k)=1 shows that the low temperature features depend on the structure factor.

Mixtures

There exists QMD data in the literature for a few systems of multi-component plasmas. In this section, we will look at data for three systems. Carbon and hydrogen (CH) in a ratio of $(7:9)$; beryllium, deuterium, and tritium (BeDT) in a ratio of $(1:1:1)$; and deterium and tritium in a ratio of $(1:1)$

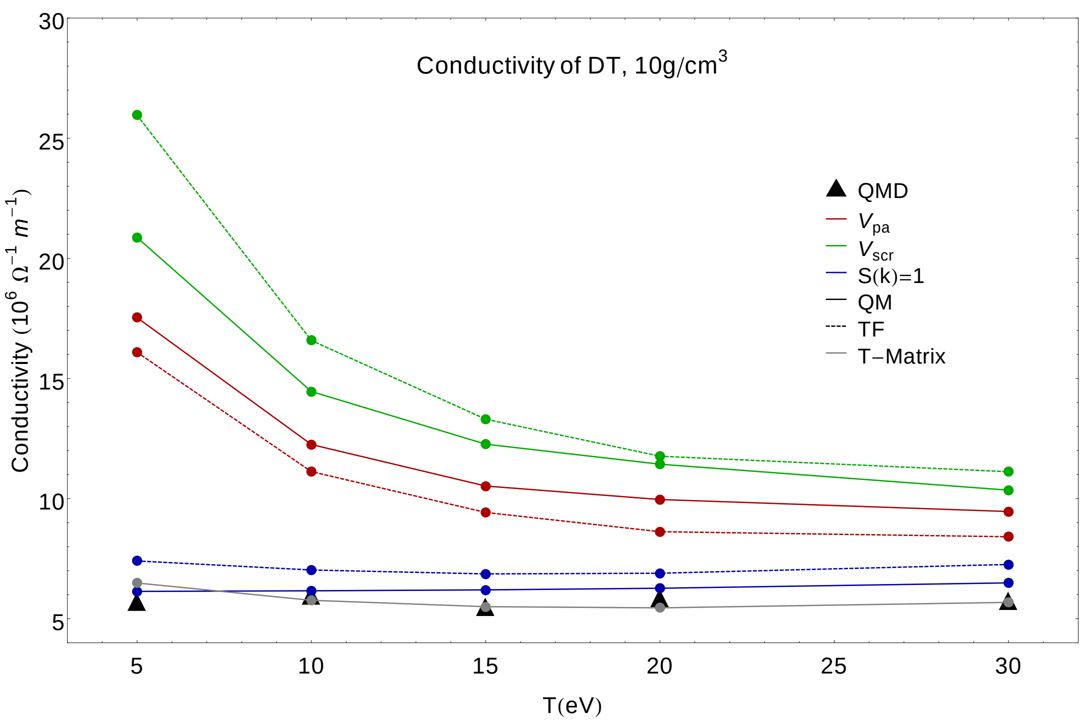


Figure 38: Deuterium and tritium at 10 g/cm^3 in a ratio of $(1 : 1)$. This plot demonstrates a trend that was present in all mixture calculations: the Born Approximation simply does not capture the physics present in mixtures.

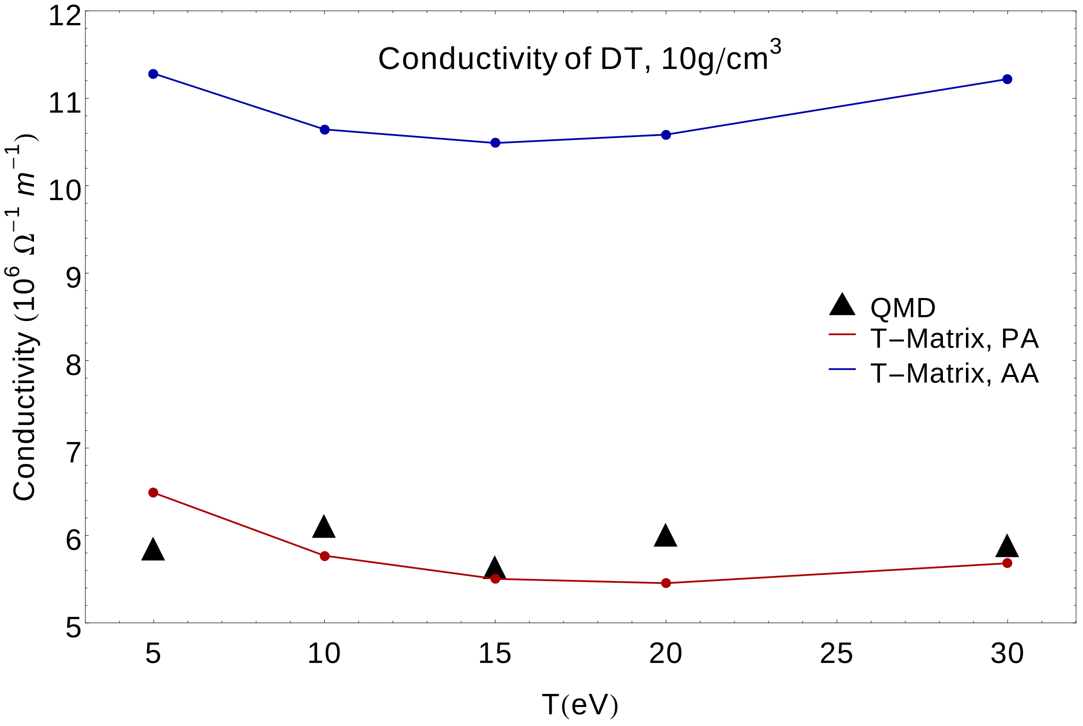


Figure 39: Deuterium and tritium at 10 g/cm^3 in a ratio of $(1:1)$. Here we take a closer look at the T-Matrix results. Both the AA and PA results are shown to confirm our earlier observations. AA consistently misses the QMD data quantitatively but does seem to trend correctly.

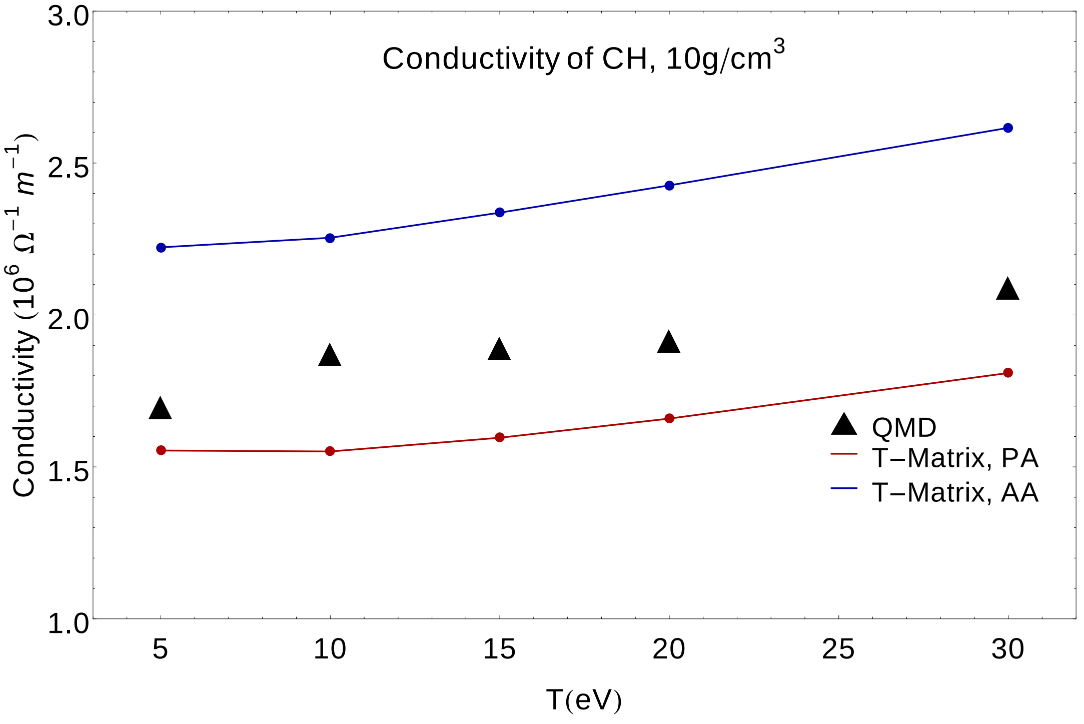


Figure 40: Carbon and hydrogen at 10 g/cm^3 in a ratio of $(1:1)$. Once again, the PAMD data more closely follows the QMD calculations.

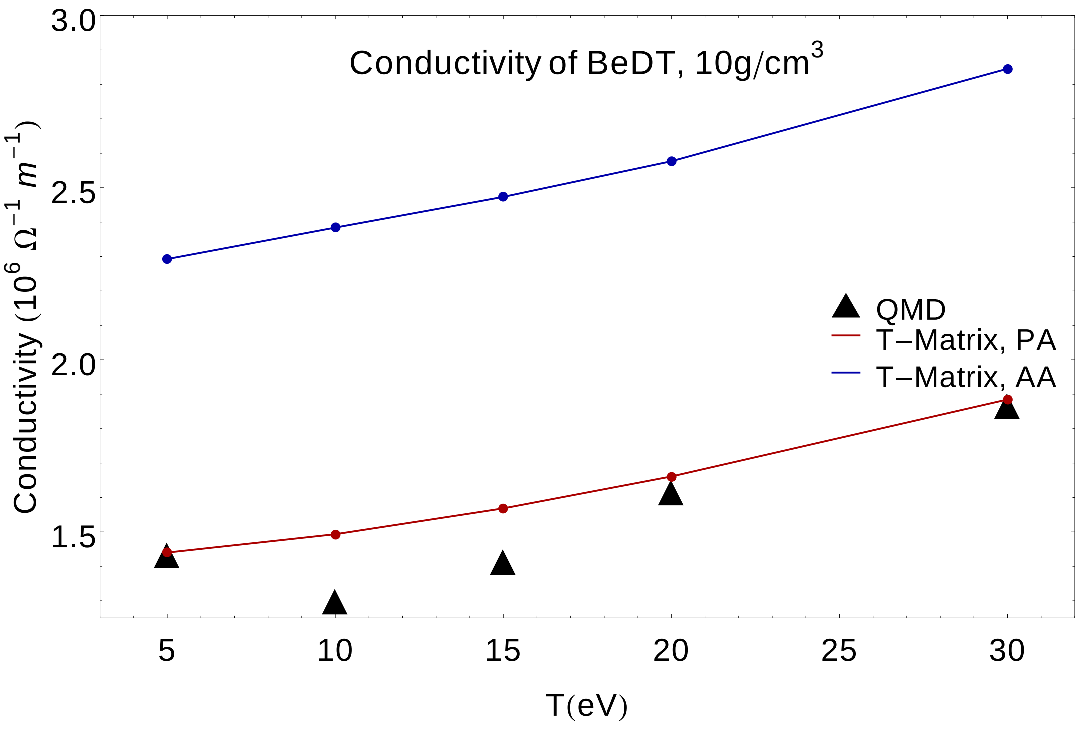


Figure 41: Beryllium, deuterium, and tritium at 10 g/cm^3 in a ratio of $(1:1:1)$.

The data from all three mixtures is very encouraging and agrees with our results for single component systems. The added complexity of more than one sort of atom seems to be too much for the Born approximation. Even for a mixture of deuterium and tritium, which shouldn't be significantly more complicated than pure hydrogen, the Born approximation only gets the right quantitative result if $S(k) = 1$ and this is most likely a cancellation of errors as opposed to an accurate modeling of the physics in our system. T-Matrix still does very well with the pseudoatom potential and, again, average atom overestimates the conductivity for all systems at all temperatures.

Conclusions

In spite of its crudeness, the Born Approximation in tandem with the Ziman-Evans formula does a surprisingly good job of qualitatively describing resistivities of various warm dense plasmas. That said, it is clear from our results that it should not be used for quantitative purposes. T-Matrix calculations do take more time to run, but the difference in time does not excuse the discrepancies between QMD data and the Born-based resistivities.

When chosing between a pseudoatom potential and a pseudopotential, the choice is very clear. Inclusion of the pseudopotential from our DFT calculations into the Ziman-Evans formula produces very nonphysical results, both qualitatively and quantitatively. Particularly, we see some fairly pathological dips and rises in resistivity as temperature increases. Since these features are not present in any other method we considered, it is most likely that the potential is the culprit.

For quantitative accuracy, a full T-Matrix calculation is almost compulsory. It is difficult to predict whether or not one is in the Born regime. Therefore, using full scattering is the best, especially when a system is complex.

In the literature, an average atom calculation is the standard method of calculating bulk properties when QMD is either unavailable or too expensive. We've shown in this report that this method can be improved by using a full pseudoatom potential instead of the more simple average atom potential. Since the computational cost is similar, PAMD is the clear choice.

References

[1] C. E. Starrett and D. Saumon. Electronic and ionic structures of warm and hot dense matter. *Physical Review E*, 87(1):1–14, 2013.

[2] C. E. Starrett and D. Saumon. A simple method for determining the ionic structure of warm dense matter. *High Energy Density Physics*, 10(1):35–42, 2014.

[3] C. E. Starrett, J. Daligault, and D. Saumon. Pseudoatom molecular dynamics. *Physical Review E*, 91(1):013104, 2015.

[4] H. D. Whitley, D. M. Sanchez, S. Hamel, A. A. Correa, and L. X. Benedict. Molecular Dynamics Simulations of Warm Dense Carbon. *Contributions to Plasma Physics*, 398(5), 2015.

[5] A. N. Souza, D. J. Perkins, C. E. Starrett, D. Saumon, and S. B. Hansen. Predictions of x-ray scattering spectra for warm dense matter. *Physical Review E*, 89(2):1–12, 2014.

[6] T. Ott and M. Bonitz. First-Principle Results for the Radial Pair Distribution Function in Strongly Coupled One-Component Plasmas. *Contributions to Plasma Physics*, 55(2-3):243–253, 2015.

[7] S. Mitake, S. Tanaka, X. Z. Yan, and S. Ichimaru. Theory of interparticle correlatinos in dense, high-temperature plasmas. II. Correlation functions. *Physical Review A*, 32(3):1775–1778, 1985.

[8] N. M. Gill, R. A. Heinonen, C. E. Starrett, and D. Saumon. Ion-ion dynamic structure factor of warm dense mixtures. *Physical Review E*, 91(6):063109, 2015.

[9] N. W. Ashcroft and J. Lekner. Structure and resistivity of liquid metals. *Physical Review*, 145(1):83–90, 1966.

[10] G. Faussurier, C. Blancard, and P. Cossé. Refractive index in warm and hot dense matter. *Physical Review E*, 91(5):36–39, 2015.

[11] W. R. Johnson. Low–frequency conductivity in the average–atom approximation. *High Energy Density Physics*, 5:61–67, 2009.

[12] R. Evans, D. Greenwood, P. Lloyd, and J. Ziman. The resistivity and thermopower of liquid mercury and its alloys, 1969.

[13] S. Tanaka and S. Ichimaru. Theory of interparticle correlatinos in dense, high-temperature plasmas. V. Electric and thermal conductivities. *Physical Review A*, 32(3):1785, 1985.

[14] C. E. Starrett, J. Clérouin, V. Recoules, J. D. Kress, L. A. Collins, and D. E. Hanson. Average atom transport properties for pure and mixed species in the hot and warm dense matter regimes. *Physics of Plasmas*, 19(10), 2012.

[15] V. Recoules and J. P. Crocombette. Ab initio determination of electrical and thermal conductivity of liquid aluminum. *Physical Review B*, 72(10):1–4, 2005.

[16] G. Faussurier, C. Blancard, P. Combis, and L. Videau. Electrical and thermal conductivities in dense plasmas. *Physics of Plasmas*, 21(9):092706, 2014.

[17] R. Evans, D. A. Greenwood, and P. Lloyd. Calculations of the transport properties of liquid transition metals. *Physics Letters*, 35(A):57–58, 1971.

[18] R. Evans, H. J. Güntherodt, H. U. Künzi, and A. Zimmermann. An extension of the Faber-Ziman formula to liquid alloys of transition metals. *Physics Letters A*, 38(3):151–152, 1972.

[19] T. E. Faber and J. M. Ziman. A theory of the electrical properties of liquid metals. *Philosophical Magazine*, 11(109):153–173, 1965.

[20] G. Faussurier and C. Blancard. Resistivity saturation in warm dense matter. *Physical Review E*, 91(1):1–4, 2015.

[21] A. Jain and R. Evans. Calculation of the electrical resistivity of liquid iron in the Earth's Iron Core. *Nature physical science*, 239(27):110–112, 1972.

[22] F. Perrot and M. Dharma-Wardana. Electrical resistivity of hot dense plasmas. *Physical Review A*, 36(1):238–246, 1987.

[23] V. K. Ratti and R. Evans. The resistivity and thermoelectric power of the liquid alkaline earth metals. *Journal of Physics F: Metal Physics*, 3(11):L238–L243, 2001.

[24] J. M. Ziman. A theory of the electrical properties of liquid metals. I: The monovalent metals. *Philosophical Magazine*, 6(68):1013–1034, 1961.

[25] R. Evans, B. L. Gyorfey, N. Szabo, and J. M. Ziman. On the resistivity of liquid transition metals. In S. Takeuchi, editor, *The properties of liquid metals*, pages 319–331. Taylor and Francis, London, 1973.