Oregon Undergraduate Research Journal
9.1 (2015) ISSN: 2160-617X
ourj.uoregon.edu

GPU-Imogen: An Astrophysical Hydrodynamic Code Built for Graphics Processing Units

Isaac Brown, Tom Wolken, Physics and Math*

ABSTRACT

We describe Graphics Processing Unit-Imogen (GPU-Imogen), an astrophysical hydrodynamics computer code, developed by Erik Keever and Scott Ernst. GPU-Imogen uses the Harten-Lax-van Leer-Contact (HLLC) fluid scheme to simulate the compressible Euler equations (equations governing a fluid with no viscosity or heat conduction). The fluid scheme is performed on the GPU, with the possibility of parallelization to multiple GPUs per node and/or GPU clusters. We describe the fluid scheme and GPU parallelization to understand the robustness and efficiency of the code. Benchmark tests of one-, two-, and three-dimensional problems are provided to verify robustness. User friendly features of GPU-Imogen are also explored. GPU-Imogen is shown to be a strong choice for an astrophysical hydrodynamics code.

INTRODUCTION

1.1 BACKGROUND

GPU-Imogen was written for use in numerical studies of astrophysical fluid flows such as spiraling gas around a star. Numerical methods can involve either serial or parallel computation. Serial computation is when one calculation is carried out at a time. In some problems, like fluid dynamics and graphics processing, it becomes more efficient to break the problem into many smaller parts that can each be solved with the same scheme simultaneously. This is one example of a parallel computation. GPU-Imogen's fluid scheme is parallelizable, so it performs the bulk of its computations in parallel on a single GPU, or on any number of GPUs working in parallel with each other. Parallelization makes GPU-Imogen highly efficient, which is particularly important for computational astrophysics, where high resolution experiments can take days to run.

* Tom Wolken is a fourth year physics and math double major from Portland, OR. He is a math tutor and an undergraduate researcher in in Dr. Imamura's astrophysics lab. He hopes to further his education by pursuing a Ph.D. in astrophysics. Always having a passion for outer space, he is currently interested in the computational aspect of astrophysics. Please direct correspondence to thomas.wolken@gmail.com.

Isaac Brown is a fourth year physics and math double major from Springfield, OR. He has worked as lab manager in the physics teaching labs for two years, spent several terms as a physics undergraduate teaching assistant, and currently works in the math drop-in center at the Math Library. He enjoys watching bad movies and staying inside on sunny days. Before becoming a physics major, Isaac studied vocal performance at the UO school of music, and enjoys playing the piano and singing. Please direct correspondence to brwnisaac@yahoo.com.

Two subfields of computational astrophysics are computational fluid dynamics (CFD) and magnetohydrodynamics (MHD). GPU-Imogen is currently a CFD code. In these fields, computer schemes can be developed to approximate the dynamics of fluids under conditions that are difficult to achieve in a laboratory. For example, the plasma in the core of the sun can reach pressures of 100 billion atmospheres, and temperatures of 15 million degrees Kelvin. Both CFD and MHD codes are currently analyzing important unsolved questions in astrophysics.[1,2,3,4] MHD codes are inherently more complicated than CFD codes since they must incorporate the additional physics of magnetism. Even though MHD codes account for more physics, there is still a need for specialized CFD codes in astrophysical research. Implementing a robust CFD code is arguably simpler than implementing a robust MHD code, so if magnetic effects can be ignored then CFD codes are advantageous. Examples of astrophysical CFD codes are Art, Enzo, and Heracles,[5,6,7] while examples of astrophysical MHD codes are Athena, Nirvana, and Zeus-MP.[8,9,10] The name "Imogen" in GPU-Imogen follows the trend of using Greek mythological names for astrophysical codes.

1.2 HISTORY OF GPU-IMOGEN

The code that is now GPU-Imogen is an evolution of a code first developed by Scott Ernst, which was originally named Imogen. Imogen ran only on central processing units (CPUs) and used a different fluid scheme presented by Jin and Xin.[11] Imogen was an MHD code that Ernst used to investigate magnetic accretion shocks in his thesis.[12] Beginning in 2009, Erik Keever modified Imogen into a GPU-based parallel code, GPU-Imogen. Up until 2013, GPU-Imogen kept the scheme used in the original Imogen. In 2013, Keever presented this version of GPU-Imogen at a conference on supercomputing.[13] In late 2013, Keever replaced the original fluid scheme with a new fluid scheme developed by Toro et al named Harten-Lax-van Leer-Contact (HLLC).[14] This scheme was chosen for its known robustness and efficiency in CFD. We worked under the direction of Keever to first understand and then use GPU-Imogen. This paper is our description of tests that we ran and information that we compiled on the most recent version of GPU-Imogen. The source code for GPU-Imogen is available to download for free online at https://github.com/imogenproject/GPUImogen.

1.3 FEATURES OF GPU-IMOGEN

A key feature of GPU-Imogen is that the fluid scheme is performed on the GPUs (and is hidden from the user), while the initial conditions for the experiments are generated using a user friendly language called MATLAB. The fluid scheme can be thought of as the laws of physics for our program since they are unchanging and do not need to be tampered with by the user. On the other hand the setup of each experiment needs to be accessible to the user so it is written in MATLAB. Once GPU-Imogen runs the initial conditions, it sends them to be evaluated on GPUs automatically. It then prints images and data files at specified intervals, making analysis straightforward. GPU-Imogen also contains pre-built functions, allowing for automatically generated supersonic shocks, static cells (cells that act as a solid wall), and potential fields (such as gravity). This makes it even easier for users to quickly generate a new experiment.

2. THEORY

2.1 BASIC EQUATIONS FOR CFGD

GPU-Imogen implements a fluid scheme to solve the compressible Euler equations. In conservative and Cartesian form, the Euler equations are:

$$\frac{\partial \rho}{\partial t} = -\frac{\partial}{\partial x_i}\rho v_i \qquad (1)$$

$$\frac{\partial(\rho v_i)}{\partial t} = -\frac{\partial}{\partial x_j}(T_{ij} + \mathbb{I}P) \qquad (2)$$

$$\frac{\partial E}{\partial t} = -\frac{\partial}{\partial x_i} v_i(E + P) \qquad (3)$$

for mass density ρ, fluid bulk velocity v_i, momentum stress tensor $T_{ij} = \rho v_i v_j$, identity tensor $\mathbb{I}$, scalar thermal pressure P, and total energy density $E = \frac{1}{2}\rho v^2 + \epsilon$. Where ϵ is specific internal energy. These equations represent the conservation of mass, momentum, and energy respectively. The equation for pressure is $P(\epsilon) = (\gamma - 1)\epsilon$. Here, the term γ is the adiabatic index. Physical values of γ lie within $1 < \gamma \leq 5/3$.

2.2 RIEMANN PROBLEMS AND SOLVERS

A Riemann problem is a fluid dynamics problem where the initial values for the problem are piecewise constant with one discontinuity. One example of a Riemann problem is shown in Figure 1a. Riemann problems occur naturally in CFD, where scientists are often interested in studying shock waves and other phenomenon that are inherently discontinuous. Also, the discreteness of grids used in CFD leads to discontinuities at cell boundaries. Scientists such as Godunov have derived exact solutions to one-dimensional Riemann problems.[15] The evolution of the exact solution to our Riemann problem pictured in Figure 1a is shown in Figure 1b. In Figure 1b, region one is the not-yet affected high density region. Region two is known as the rarefaction wave. The discontinuity between region three and four is the contact discontinuity. The discontinuity between region four and five is the shock front, and region five is the not-yet affected low density region.

Unfortunately the exact solutions to one dimensional Riemann problems do not carry over to the two-dimensional case, so approximate Riemann solvers are used. In this way, GPU-Imogen's fluid scheme uses an approximate Riemann solver.

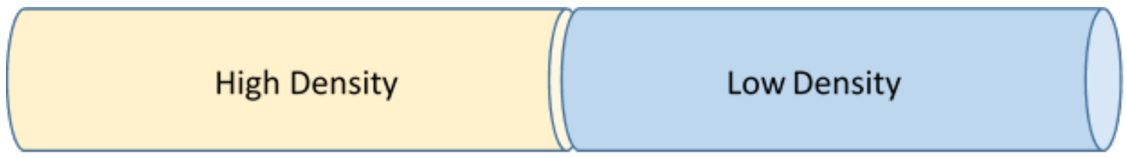

Figure 1a: Two regions of differing density in contact. This specific Riemann problem is often referred to as Sod's shock tube.

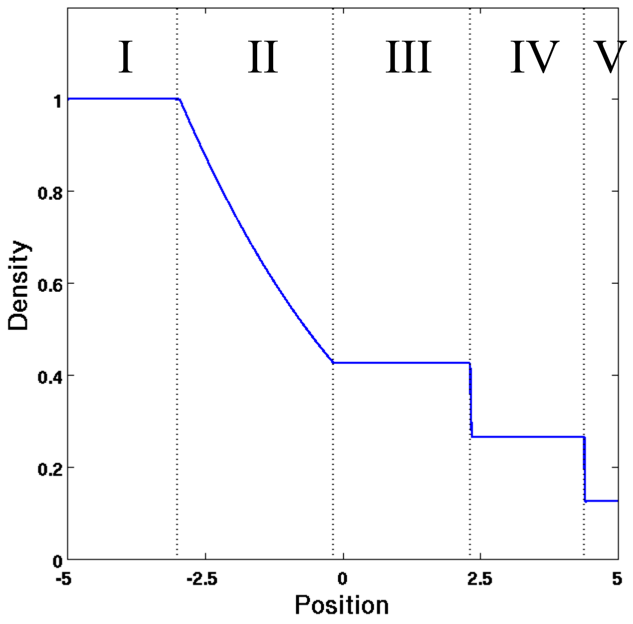

Figure 1b: The exact, time evolved, solution to Sod's shock tube.

2.3 DISCRETIZATION

In theory, fluids are continuous and infinitesimally fine. In practice, computer algorithms must approximate the continuous flow of fluid by discretizing the space of the problem. GPU-Imogen does this by hashing the problem domain into equally sized cells. In one dimension, these cells are line segments. In two-dimensions they are squares, and in three dimensions they are cubes. This type of discretization is known as a finite volume Cartesian grid. The continuous spatial coordinates are (x, y, z) and the cell represented by (i, j, k) is centered at (x_i, y_j, z_k).

Space is not the only thing to be discretized, as time is also broken into a finite number of steps. We follow the notation that the current time step is denoted in superscript by "n." The length of a given time step is then $\Delta t^n = t^{n+1} - t^n$. Notice how the length of a given time step depends on n (from here on we drop the n superscript on Δt). This is because the time step needs to be small enough to ensure that the fastest wave does not cover more than one discrete cell within the time step. To ensure this, we use the Courant–Friedrichs–Lewy (CFL) condition: $\Delta x < \Delta t(|v_i| + C_s)$ where Δx, Δt, v_i, and C_s represent the change in space, time step, bulk fluid velocity of the ith cell, and the speed of sound in the fluid.

2.4 ONE-DIMENSIONAL FLUID SCHEME

For completeness, the details of GPU-Imogen's fluid scheme in one dimension is introduced. In one dimension the conservation form of the Euler equations (1) – (3) can be represented as

$$\frac{\partial \boldsymbol{q}}{\partial t} + \frac{\partial \boldsymbol{f}}{\partial x} = 0 \qquad (4)$$

where the vectors $\boldsymbol{q}$ and $\boldsymbol{f}$ stand for the conserved quantities and their fluxes as

$$\boldsymbol{q} = < \rho, \rho v_x, \rho v_y, \rho v_z, E >$$

$$\boldsymbol{f} = < \rho v_x, \rho v_x^2 + P, \rho v_x v_y, \rho v_x v_z, (E+P) v_x >$$

where ρ, v_x, v_y, v_z, E, and P stand for the mass density, Cartesian components of velocity, energy, and thermal pressure respectively. Equation (4) can be recast as an integral over one discrete cell spatially and a time step (Δt) temporally to obtain

$$\boldsymbol{q}_i^{n+1} = \boldsymbol{q}_i^n - \frac{\Delta t}{\Delta x}\left(\boldsymbol{f}_{i+\frac{1}{2}}^{n+\frac{1}{2}} - \boldsymbol{f}_{i-\frac{1}{2}}^{n+\frac{1}{2}}\right) \qquad (5)$$

where

$$\boldsymbol{q}_i^n = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \boldsymbol{q}\,(x, t^n)dx \qquad (6)$$

is a vector of volume averaged conserved quantities, and

$$\boldsymbol{f}_{i-\frac{1}{2}}^{n+\frac{1}{2}} = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \boldsymbol{f}\left(x_{i-\frac{1}{2}}, t\right) dt \qquad (7)$$

are time averaged fluxes at the location $x_{i-1/2}$. Being able to compute (5) means that we can calculate the values for our cell centered conserved quantities at the next time step. Starting from (6) a piecewise linear reconstruction scheme is used to compute $\boldsymbol{q}_{L,i-1/2}$, and $\boldsymbol{q}_{R,i-1/2}$ which represent the conserved quantities on the left and right side of the interface between cells i-1 and i. Figure 2 provides a visual. The reconstruction scheme is outlined in,[8] and is shown to be second order accurate and total variation diminishing by Sweby.[16] Slope limiters are used in the reconstruction scheme to ensure that no new extreme values are created. An approximate Riemann solver is used to compute (7). The approximate Riemann solver for GPU-Imogen is introduced in section 2.5.

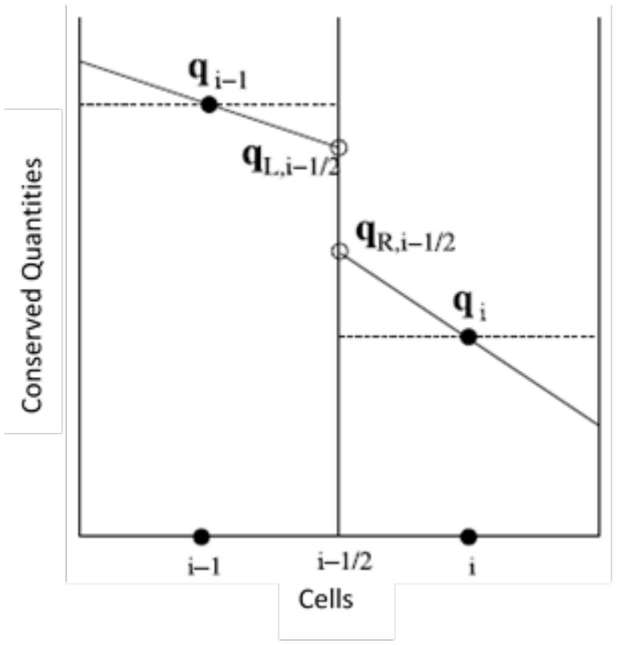

Figure 2: A piecewise linear reconstruction of conserved quantities from the center of each cell ($\boldsymbol{q}_{i-1}$, $\boldsymbol{q}_i$) to the left and right sides of the cell interface ($\boldsymbol{q}_{L,i-1/2}$, $\boldsymbol{q}_{R,i-1/2}$). Notice how the conserved quantities are discontinuous at the cell boundary. This discontinuity defines a Riemann problem. Figure credit Stone.[8]

We will now outline the steps for the one dimensional fluid scheme:

Step 1: From $\boldsymbol{q}_i^n$, calculate $\boldsymbol{q}_{L,i-1/2}$, and $\boldsymbol{q}_{R,i-1/2}$ at every cell boundary by using the piecewise linear reconstruction scheme.

Step 2: Compute the fluxes (7) by using an approximate Riemann solver (solver is introduced in section 2.5).

Step 3: Use (5) to update the new cell centered conserved quantities.

Step 4: Advance the time from t^n to t^{n+1} by the relation $t^{n+1} = \Delta t + t^n$. Use the CFL condition $\Delta x < \Delta t(|v_i| + C_s)$ to calculate an appropriately small time step Δt for the next cycle.

Step 5: Repeat steps 1-4 until the final time is reached.

2.5 APPROXIMATE RIEMANN SOLVER FOR GPU-IMOGEN

The current approximate Riemann solver for GPU-Imogen was developed by Toro, Spruce, and Speares.[14] Toro et al had taken an existing approximate Riemann solver developed by Harten, Lax, and van Leer[17] (called HLL) and restored the missing contact wave back into the scheme. They called their new scheme HLLC (the C stands for contact). To evolve the system in time, the solver will have to approximate the solution to the Riemann problems at the boundaries seen in Figure 2. HLLC approximates the solution to the Riemann problem by splitting up the problem domain into four separate regions (U_L, U_L*, U_R*, and U_R) as seen in Figure 3. The horizontal axis of this figure represents position (x) and the vertical axis represents time (t). The middle of Figure 3 is the position where the initial discontinuity between cells occurs. The domains are separated by the lines S_L, S_M, and S_R. Respectively, these lines track the location of the fastest wave moving to the left, the contact discontinuity, and the fastest wave moving to the right. After approximating the Riemann problem, the fluxes in step two of the fluid scheme can be calculated. For an exact description of how these fluxes are calculated see Batten's paper.[18]

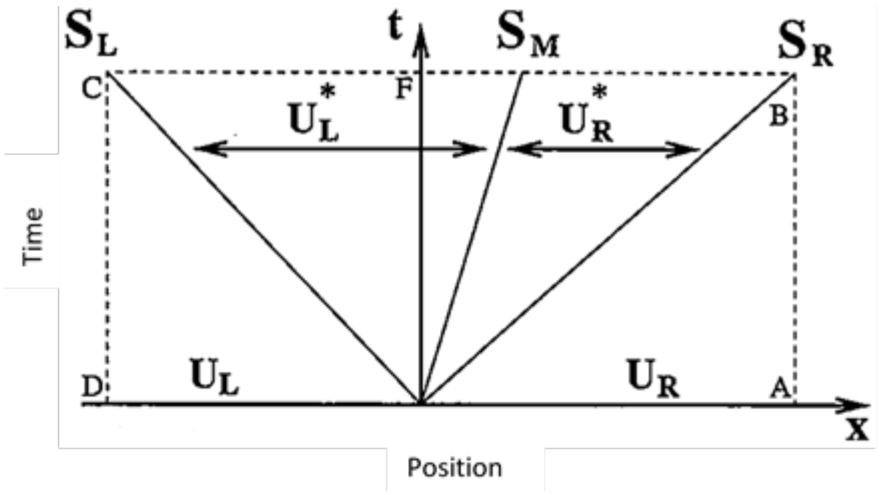

Figure 3: A visualization of how HLLC approximates a solution to the Riemann problem. Figure credit Batten.[18]

HLLC has been proven to be positivity conserving by Batten et al,[18] meaning that a problem with initially positive density will remain positive throughout the simulation. This is important because negative pressures are unphysical (they do not occur in nature) and they can cause codes to malfunction. Being positivity conserving is a property that not all approximate Riemann solvers have. For instance the Roe scheme[19] has been shown to produce negative pressure and fail when given certain initial conditions such as Einfeldt's strong rarefaction test.[20] Furthermore, Batten et al proves that by tracking the location of the contact discontinuity, HLLC has the ability to perfectly resolve stationary contact discontinuities.

2.6 DIMENSIONAL SPLITTING

We have described the fluid scheme in one dimension. In two dimensions, the one dimensional scheme is essentially run again independently in the second spatial dimension. In three dimensions, the process is split into three parts. This process is called directional splitting, or Strang splitting, as it was proposed by Strang in 1968.[21] Since the flux calculation is split amongst dimensions, there is a possibility that some small systematic error will favor one direction. We explore this effect in section 4.2a.

3. COMPUTER IMPLEMENTATION

3.1 COMPUTER ARCHITECTURE

User level files for GPU-Imogen are written in MATLAB, a commonly used programming language by scientists and engineers. This means that at user level, GPU-Imogen is easy to understand and modify. The fluid scheme is written in CUDA, a parallel language for GPUs. Figure 4 shows the architecture of GPU-Imogen. If the user wants to run a simulation on only one node (one computer) then the left route in Figure 4 is taken. The user has the option to use one or multiple GPUs contained on that node. If the user wants to use multiple nodes (such as a GPU cluster) then MPI or "message passing interface" is used for communication between nodes.

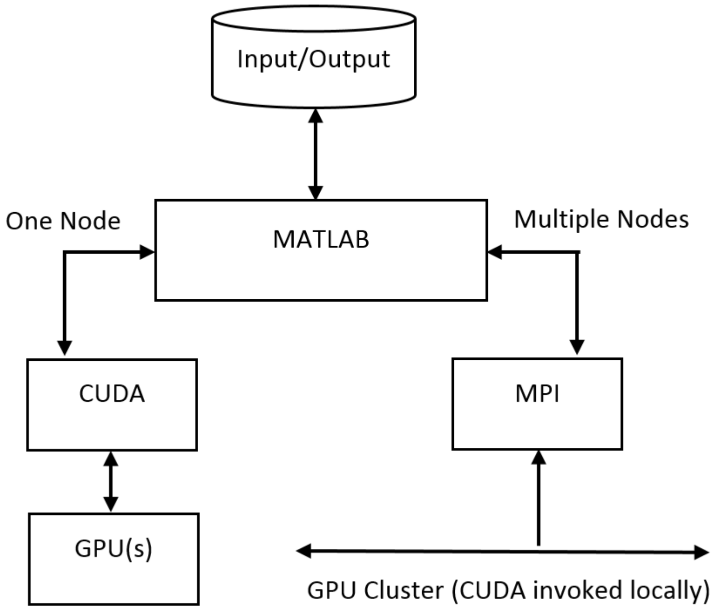

Figure 4: Architecture of GPU-Imogen.

3.2 PARALLEL SCALING

In parallel computing, weak scaling is the impact on efficiency when a problem is scaled up in size while the workload for each processor remains constant. For example, start with a problem of size X with Y processors, then double the size to 2X while also doubling the number of processors to 2Y. If the amount of time to solve the new problem if roughly the same as the amount of time to solve the original, then your problem exhibits flat weak scaling and would translate nicely to a massively parallel environment. Keever explored weak scaling for one- and two-dimensional problems on his poster[13] and found that GPU-Imogen exhibited roughly flat weak scaling for all tests. GPU-Imogen was tested on up to 72 GPUs simultaneously with the University of Oregon's supercomputer ACISS. With this flat weak scaling, GPU-Imogen can handle an arbitrarily large problem if given enough computing power.

4. BENCHMARK TESTS

4.1 ONE-DIMENSIONAL TESTS

4.1a Sod Shocktube

i. Background

The Sod shock tube problem was first proposed in 1978 by Gary Sod,[22] and has been regarded as a prototypical CFD test. This problem was designed to reveal the three different primitive waves in fluid dynamics, and to evaluate a code's ability to resolve each of these waves. These are the rarefaction wave, contact discontinuity, and shock wave.

ii. Initial Conditions

This test is defined by the following parameters: the left half of the tube has density $\rho=1$ and internal energy $e=1$, while the right half has $\rho = .125$ and $e = .1$.

iii. Time-Evolved Behavior

When released, the high pressure region on the left sends a shockwave to the right, which propagates through the low density region. The slower contact discontinuity also propagates to the right. The rarefaction wave recedes to the left.

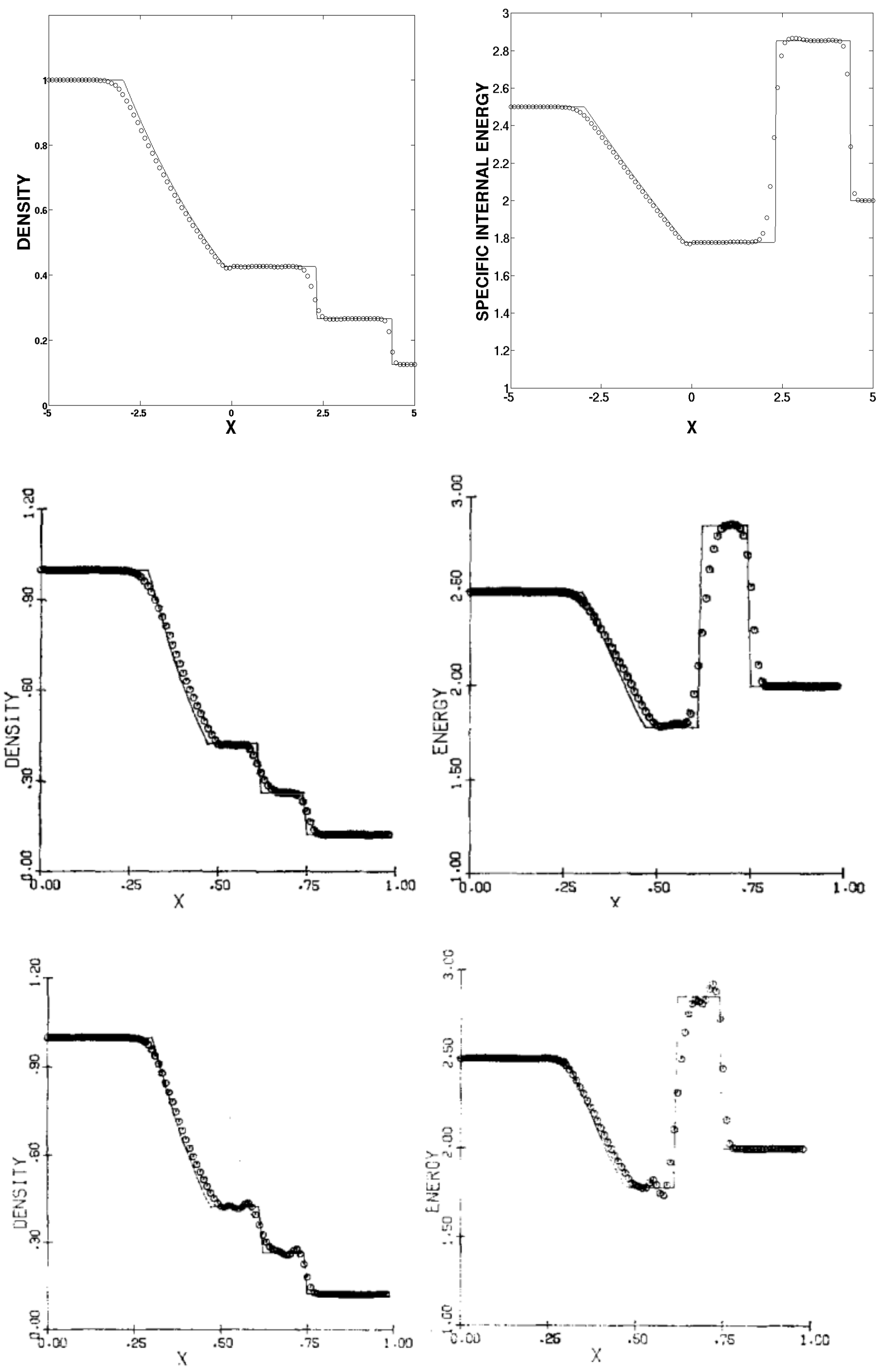

Figure 5: Sod Shock Tube at t = .25 for density (left) and specific internal energy (right). Spatial resolution of 100 cells. Analytic (exact) solutions are shown as solid lines, while each of the 100 computed cells are shown as circles. Schemes compared are GPU-Imogen (top), a 1st-order Godunov scheme (middle), and a second-order MacCormack scheme (bottom). MacCormack and Godunov figures credit Sod[22]

The contact discontinuity experiences diffusion, which has the effect of smearing out the contact. This is a non-physical artifact, because the velocity on either side of the contact is identical. If all the fluid around the contact is moving at the same speed, there should be no forces to induce mixing, so the contact should remain sharp. Notice how HLLC (the scheme used by GPU-Imogen) does not smear features as much as the first-order Godunov scheme. Also, HLLC does not suffer from the oscillatory overestimations at the contact and shock that MacCormack's second-order scheme introduces.

4.1b Einfeldt Strong Rarefaction test

i. Background

Einfeldt's Strong Rarefaction test evaluates a code's ability to maintain positivity (as defined in section 2.5) under extreme rarefactions. This happens when two slabs of fluid move away from each other, pulling a significant quantity of material out of some region, creating a low density region at the origin.

ii. Initial Conditions

To set up the test, we separate the grid into two equally sized regions on the right and left end of the grid. We define uniform density $\rho = 1$ and internal energy $e = 3$ everywhere, but with the fluid on the left moving to the left with momentum of -2, and the fluid on the right moving to the right with momentum of 2.

iii. Time-Evolved Behavior

When released, the region in the center of the grid experiences a strong density rarefaction. Many fluid codes will overcompensate in their computation of this rarefaction, as outlined in Batten et al.[18] In particular, many codes will, at some point, calculate a negative density within the rarefaction, which is a non-physical error, as mass density cannot have a negative value. However, HLL solvers (HLLC included) are guaranteed to maintain positivity under these conditions.[18] We compared the results from Einfeldt and Roe et al,[20] who used a similar first-order HLL solver called HLLE to evaluate this test, to our results using GPU-Imogen. The results are shown in Figure 6. The results confirm that GPU-Imogen handles the rarefaction test well, maintaining positivity as advertised.

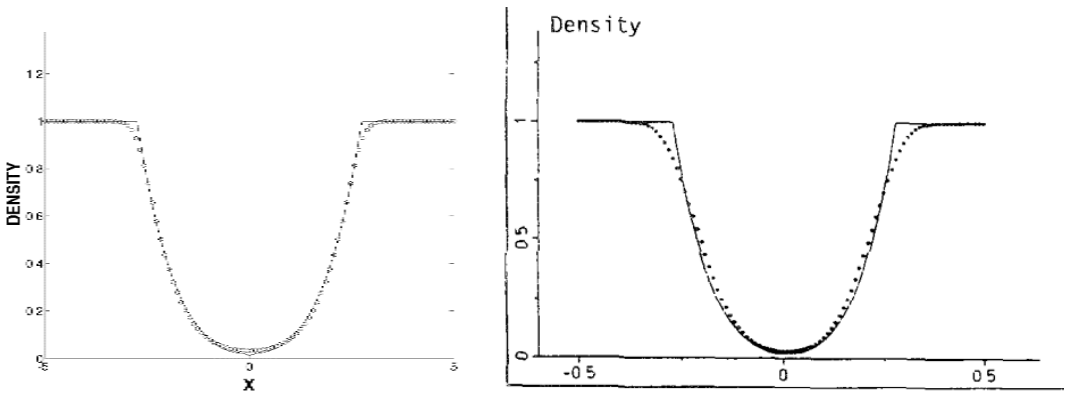

Figure 6: Numerical result of the HLLC-scheme (left) and the HLLE-scheme given in Einfeldt[20] (right) at time t = 0.1.

4.1c. Shu-Osher test

i. Background

This test evaluates a code's ability to capture both shocks and fine detail in smooth regions simultaneously. The particular details of the test were laid out by Shu, C and Osher, S.[23]

ii. Initial Conditions

For this test, we use a domain of (-5,5) for x. The left tenth of our grid (ie $-5<x<-4$) is a high pressure region moving to the right (density $\rho=3.857$; horizontal velocity $V_x= 2.629$, pressure $P = 10.333$), and the rest of the grid contains a stationary series of low-density sinusoidal waves. These sinusoidal fluctuations are described by

$$\rho=1 + 0.2 \sin(5 x);\ V_x=0;\ P=1,$$

in the region of $-4<x<5$.

iii. Time-Evolved Behavior

When released, the shockwave propagates into the density structures. These structures become distorted, attaining both higher frequency and higher amplitude. This frenetic region can prove problematic for certain codes. In particular, some of the extremes in this region can be lost. In Shu and Osher's 1989 paper,[23] they compared the numerical solutions of this test for several different solvers. Shu and Osher observe that higher order schemes are significantly better at capturing detail in the frenetic region, as seen in Figure 7. In Figure 8, we notice that GPU-Imogen is better at capturing all extrema in the frenetic region than the MUSCL scheme, despite both schemes being second order. We see that the accuracy and robustness of a code depends on more than its order, which prompts a discussion of convergence order, detailed in section 5.

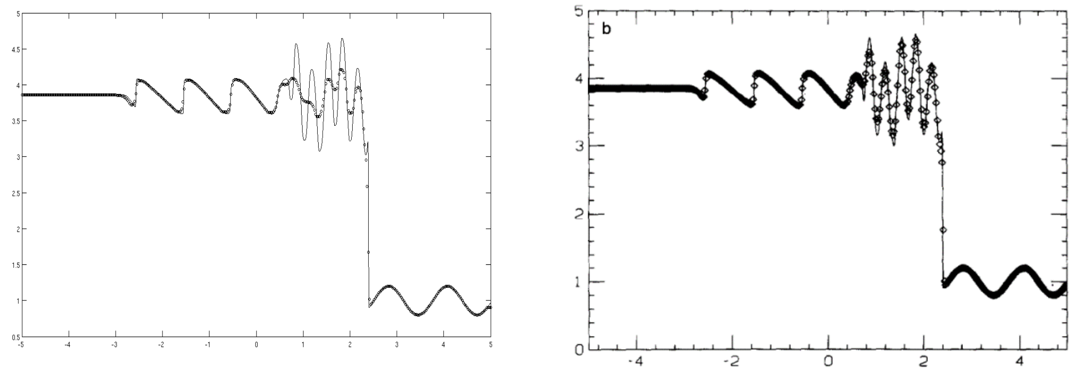

Figure 7: GPU-Imogen using 2nd order HLLC (left) compared with 3nd order ENO (right), both at 400 points. Note that the 3rd order code provides nearly perfect resolution of the extrema.

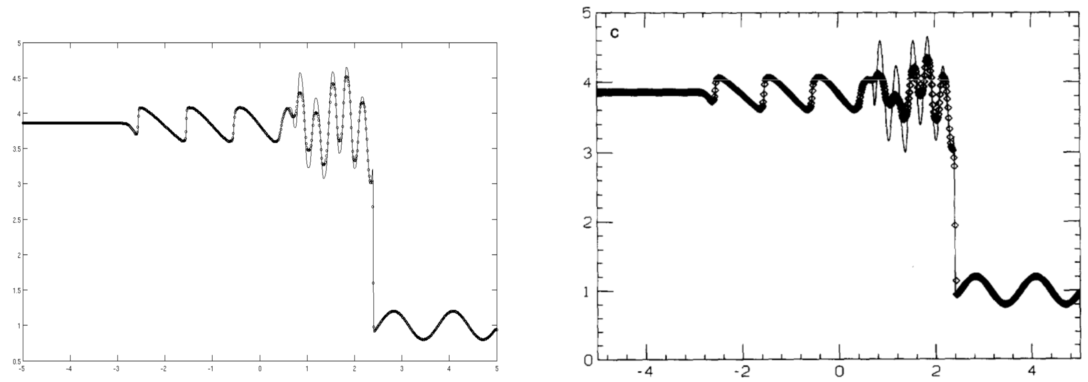

Figure 8: GPU-Imogen using 2nd order HLLC (left) compared with 2nd order MUSCL (right), both at 800 points.

4.2 TWO-DIMENSIONAL TESTS

4.2a Implosion Test

i. Background

This is a test often used to demonstrate the preservation of symmetry, or failure to preserve symmetry, along diagonal lines. HLLC uses a dimensional splitting method, as described in section 2.4. When fluid is moving along a diagonal path, its motion may first be updated left-to-right, then later updated top-to-bottom. Though this makes the calculations far simpler, it has the effect of disturbing the diagonal symmetry of a flow. To demonstrate this, we run the implosion test, first conceived by Hui, Li, and Li.[24]

ii. Initial Conditions

In Hui, Li, and Li's test, they placed a square diamond of low density and pressure in the center of an otherwise uniform square grid, creating a quadrilaterally symmetric flow upon release. We instead simulate only the upper right corner of Hui, Li, and Li's model, as previously done by Liska and Wendroff.[26]

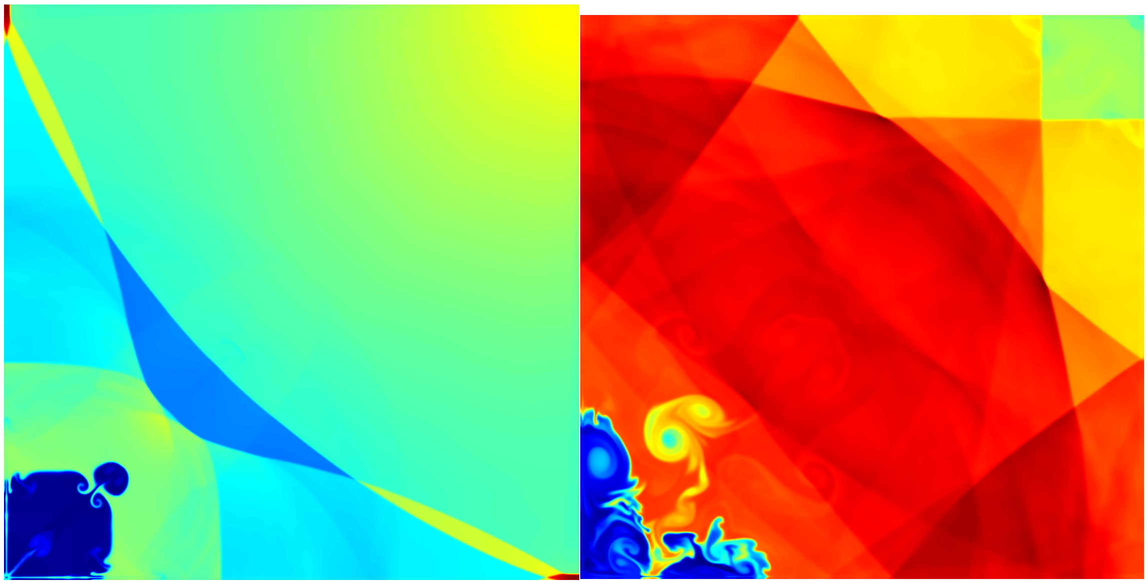

Figure 9: GPU-Imogen's simulation of the implosion test at t = 0.6 (left) and t = 3.9 (right). Resolution of 800x800 with density color plot shown. Note that, at the early stage, diagonal symmetry has not yet been visibly broken, but over time, errors accumulate, and the jet deflects away from its diagonal path.

iii. Time-Evolved Behavior

When released, a shock rushes into the low-density region in the bottom left corner, and reflects off of the left and bottom edges simultaneously. The high-pressure fluid flowing along the edges of the grid then meets in the corner, squeezing out a jet that travels along the diagonal from bottom left to upper right. As this jet travels along the diagonal, GPU-Imogen begins to accumulate errors that cause the jet to distort more and more over time. Liska and Wendroff compare the performance of several different solvers on the implosion test, some of which preserve symmetry and some of which fail to do so. We observe that GPU-Imogen performs similarly to the JT solver in terms of diffusivity, but fails to preserve symmetry to an extent comparable to VH1. Schemes that are directionally unsplit will preserve the symmetry along the diagonal in this test; however, these schemes can be much more complicated to implement. Every hydrodynamics program has strengths and weaknesses, so it is impractical to seek a 'perfect' code, making these symmetry errors tolerable in light of the strengths of HLLC.

4.2b Rayleigh-Taylor Instability Test

i. Background

The Rayleigh-Taylor Instability (RTI) is a phenomenon that can occur when a heavier fluid sits on top of a lighter fluid under the influence of gravity. In our case, we utilized GPU-Imogen's constant gravity field feature.

ii. Initial Conditions

For our test, we used a 400x800 grid. We gave the top half a density of 2, and the bottom half a density of 1, and placed a small sinusoidal momentum perturbation at the interface. This perturbation is defined by

y = .5 + .01*(1+cos(2pi*x)).

iii. Time-Evolved Behavior

When released, the heavy fluid will attempt to fall through the lighter fluid, but cannot do so uniformly, because the lighter fluid must be pushed out of the way. Small scale fluctuations on the boundary between fluids will become unstable and self-amplify over time.

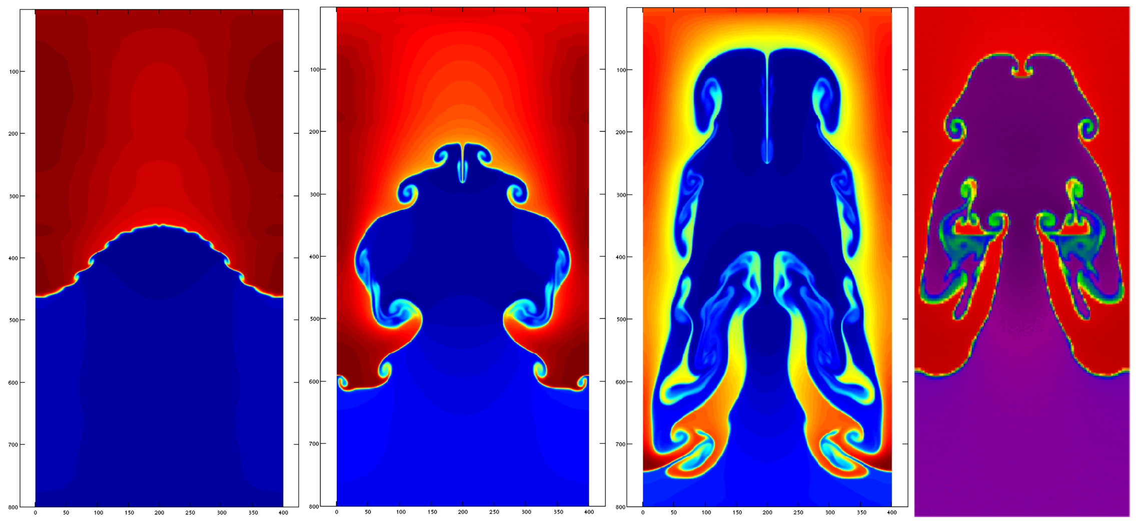

Figure 10: RTI at T = 2 (left), T = 3 (left middle) and T = 4 (right middle) at 400x800. Athena's solution to RTI at T=8.5 at 200x400 is on the right.

The RTI is a good indicator of the diffusivity of a scheme, since these two fluids are separated by a contact discontinuity. The sharpness of our stationary contacts is no surprise, but we see some diffusion in regions of local vorticity, or rotational motion. This is because HLLC is best at preserving stationary contacts. Within regions of vorticity, such as the spiral forming in panel 2 of Figure 10, contacts are in constant motion in multiple directions, so HLLC's contact preservation worsens. Overall, GPU-Imogen handles this RTI comparably to other top codes.

4.2c Richtmyer-Meshkov Instability Test

i. Background

The Richtmyer-Meshkov instability (RMI) occurs when a shock wave passes through a non-uniform fluid interface. Any small perturbations in the flatness of the interface will cause the shock to refract around it. Since the interface is not a straight line, but a curve, the shock will refract by different amounts at each point along the interface. Thus, the rotational motion at each point on the interface will be different. Specifically, the local rotation on either side of any peak on the interface will spin in opposite directions, amplifying the velocity of any fluid caught between those fields. This extrudes a jet from the peak, which moves in a direction opposite the shockwave's motion.

ii. Initial Conditions

For our test, we used a 400x800 grid with coordinates (0,1)x(0,2) where fluid with a density of 5 sits below the line y = .15. We added a single sinusoidal perturbation on the interface, and placed a subsonic incident wave at Mach 0.66 just above the perturbation.

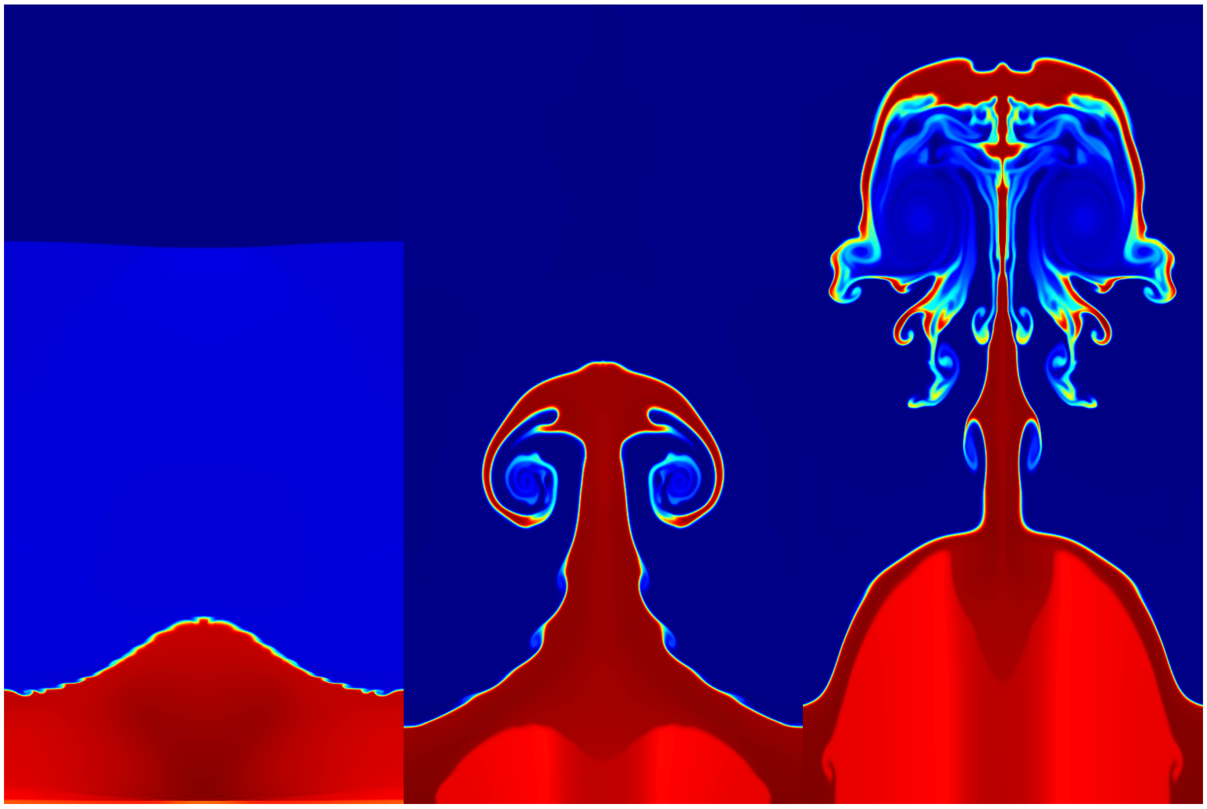

Figure 11: RMI at T = 1 (left), T = 8 (middle) and T=24 (right) on a 400x800 grid.

iii. Time-Evolved Behavior

When released, the shock wave immediately refracts around the perturbation, causing it to deform into a jet. RMI is an interesting test to run, because at sufficiently high resolutions, we can begin to observe other fluid phenomena occurring at the interface between the light and heavy fluids. On the stem of the jet in Figure 11, we see the beginning of a Kelvin-Helmholtz instability, forming due to the relative speed of the heavy fluid with respect to the light fluid at that interface. GPU-Imogen properly simulates this phenomenon.

4.3 THREE-DIMENSIONAL TESTS

4.3a Vortex Ring

i. Background

A vortex ring is a self-propagating toroidal vortex that travels along the normal to the plane of the ring. Examples of vortex rings include smoke rings made by human smokers, mushroom clouds formed by nuclear detonations, and bubble rings made by dolphins.

ii. Initial Conditions

To form our vortex ring, we created a thin cylinder of fluid moving through a stationary fluid along the normal to its circular plane. The cylinder does not need to have any other special properties besides this momentum. This replicates the conditions that would be present if an experimenter used a syringe to inject fluid into a stationary medium.

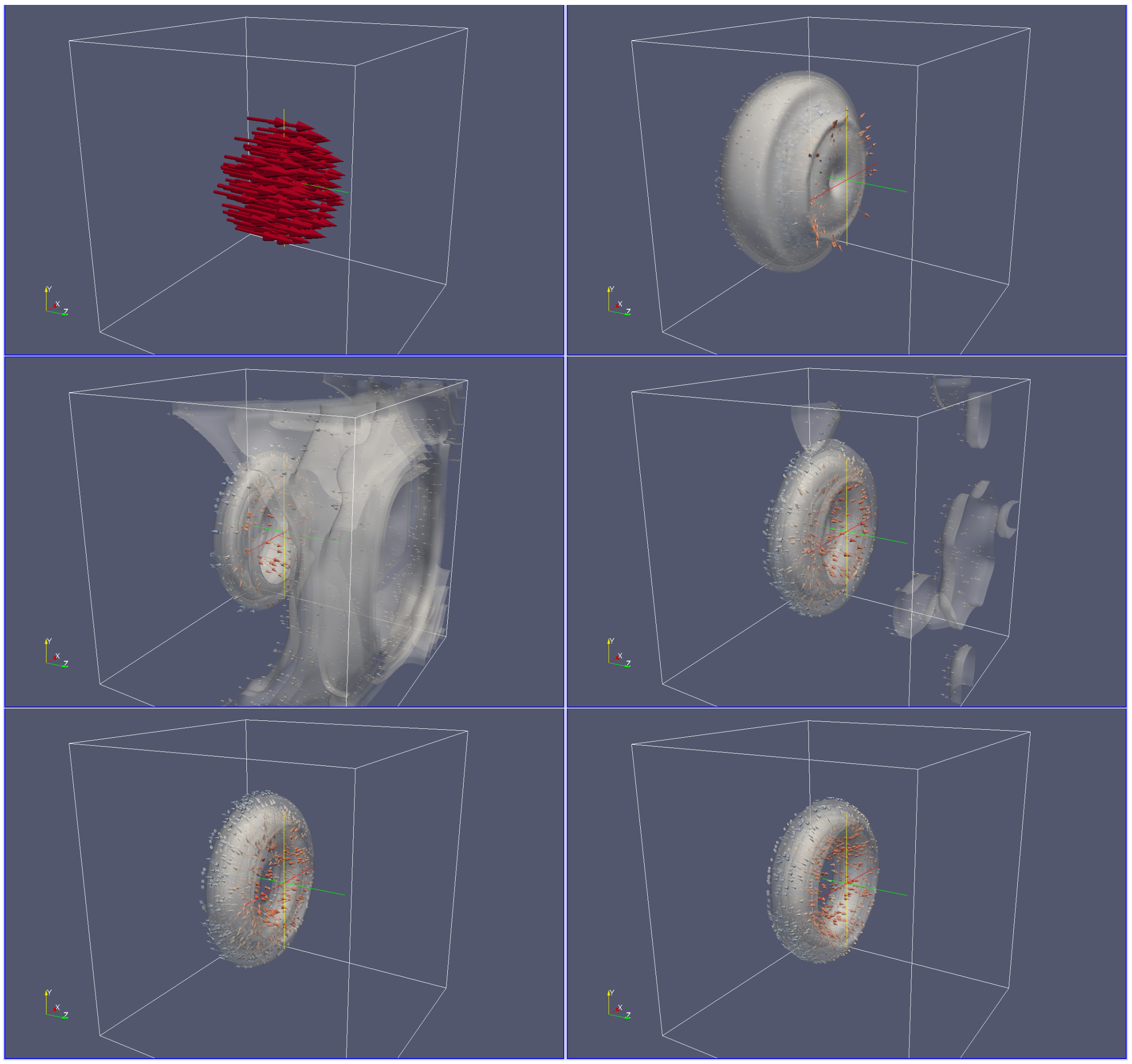

Figure 12: Formation of a vortex ring from a moving cylinder on a 100x100x100 grid. Frames taken at times T = 0, .12, .67, .94, 1.35, and 2.73. Shown are density isosurfaces and velocity arrows. Isosurfaces are drawn at density values of .978, .984, .990, and .997.

iii. Time-Evolved Behavior

As this cylinder travels, fluid at its surface experiences friction, causing its outer layers to peel away from front to back, creating vorticity (local spinning) at its surface. As it travels, this vorticity self-perpetuates, deforming the cylinder into a torus that maintains this rotational motion. If left undisturbed, this vortex will propagate in the direction of the cylinder's original momentum. This demonstrates GPU-Imogen's ability to simulate three dimensional problems.

5. ERROR ANALYSIS

GPU-Imogen features a full self-test routine that runs various tests to check that the fluid schemes are working properly. It also conducts error convergence analysis. For problems with derived (exact) solutions, such as the Sod Shock Tube and simple bulk fluid transport (advection), the self-test automatically runs tests at increasingly refined grid resolutions to examine exactly how quickly the approximate solution converges to the derived solution. We would refer to this speed as the convergence order, which should align with the order of the scheme. The piecewise linear reconstruction used by GPU-Imogen can be verified to be second-order accurate from this test.

GPU-Imogen measures error using L2 norms. These refer to the cumulative error across all cells, evaluated by

$$\mathrm{L2} = \frac{1}{\sqrt{\mathrm{N}}}\sqrt{\sum_{i}^{\mathrm{N}} \mathrm{E}(i)^2}$$

where

$$E(i) = F_{Approximate}(i) - F_{Exact}(i)$$

is the difference between the approximate solution and the exact derived solution at each cell index i. N refers to the total number of cells on the grid, often called the grid resolution. Normally, the L2 norm is preferred over the L1 norm as a measure of convergence, since compared to the L1 norm, it places less importance on the few outliers that have significantly more error than the rest of the grid.

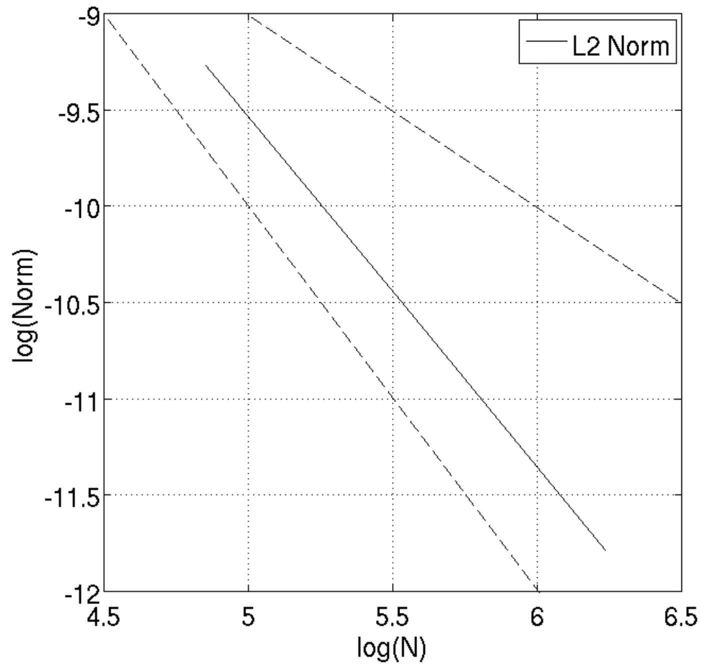

Figure 13: L2 norm for a simple, smooth fluid advection test. The dashed lines are reference lines with slopes of -1 and -2.

Figure 13 shows that the convergence order of GPU-Imogen's fluid scheme on the smooth advection problem is 2, as measured by the L2 norm. This confirms that the piecewise linear reconstruction scheme it uses is indeed second-order.

In some cases, advanced low-order schemes like HLLC can be more accurate per computation time than high-order schemes. This can be true in problems containing discontinuities, where all non-front-tracking codes are necessarily (formally at least) first order. To demonstrate this, we run convergence tests on the Sod Shock Tube, which contains several discontinuities.

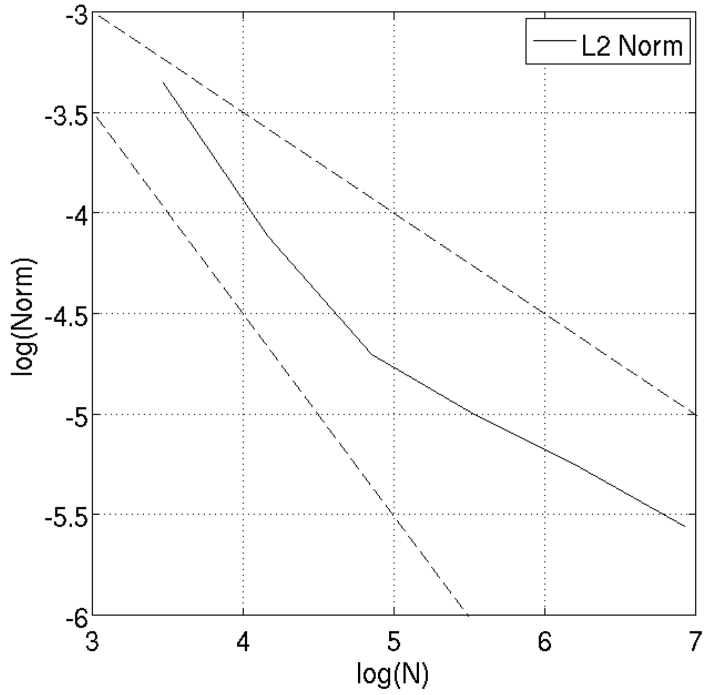

Figure 14: L2 norm for the Sod Shock tube problem. Dashed lines are reference lines with slopes of -0.5 and -1.

In Figure 14, note that the slope of the L2 line is piecewise. The kinks in this line show the critical resolution at which large features become fully resolved. Beyond those resolutions, the approximation converges at a slower rate. But more significantly, the convergence rate here never reaches second-order. This is because the error in the position of a discontinuity is first order, so any code more accurate than first-order will be wasting its effort on such problems, as the error in the discontinuity will disturb any accuracy that may have been maintained by the higher-order scheme. This would mean that, since higher-order schemes require more computation, one could increase efficiency in such situations by switching to an advanced lower-order scheme on a higher resolution.

It is fortunate that we have exact derived solutions to these particular problems, as it allows these types of quantitative analysis to give rigorous feedback on the performance of CFD codes. Without exact derived solutions, conversations about error and convergence become speculative. This makes tests like the Sod Shock tube and advection tests highly valuable to the CFD community.

6. CONCLUDING REMARKS

The point of this paper is to present the astrophysical hydrodynamic code GPU-Imogen, and to explore its features. We have shown through our benchmark tests that GPU-Imogen is robust and accurate. We have shown that HLLC is adept at capturing contact discontinuities, which allows us to resolve sharp interfaces on tests. We have also presented how GPU-Imogen is scalable to large GPU clusters, and how the use of MATLAB at surface level is user friendly. In conclusion, GPU-Imogen is a strong choice as an astrophysical CFD code.

7. FUTURE DIRECTIONS

We focused on running standard benchmark tests on GPU-Imogen to demonstrate its robustness. A next step is to run more subtle real world problems. One such problem is stellar accretion, where matter spirals in towards a star and is deposited at the surface. An eventual project would be to investigate ways to extend the existing HLLC scheme to include magnetism without having to scrap the scheme altogether. Researchers have been making breakthroughs in extending HLLC to include magnetism as recently as June of 2015.[26] This would allow us to maintain the robustness and efficiency of GPU-Imogen while broadening the scope of possible experiments to the field of MHD.

ENDNOTES

1. M. N. Lemaster, J. M. Stone, T. A. Gardiner, "Effect of the Coriolis Force on the Hydrodynamics of Colliding-Wind Binaries," *The Astrophysical Journal* 662, 582 (2007).

2. Y. Shen, J. M. Stone, T. A. Gardiner, "Three-dimensional Compressible Hydrodynamic Simulations of Vortices in Disks," *The Astrophysical Journal* 653, 513 (2006).

3. M. Shin, J. M. Stone, G. F. Snyder, "The Magnetohydrodynamics of Shock-Cloud Interaction in Three Dimensions," *The Astrophysical Journal* 680, 336 (2008).

4. R. Dong, J. M. Stone, "Buoyant Bubbles in Intracluster Gas: Effects of Magnetic Fields and Anisotropic Viscosity," *The Astrophysical Journal* 704, 1309 (2009).

5. A. V. Kravtsov, Ph. D Thesis, New Mexico State University, 2000.

6. B. W. O'Shea, "Introducing Enzo, an AMR Cosmology Application," *arXiv preprint astro-ph/0403044* (2004).

7. M. Gonzalez, E. Audit, P. Huynh, "HERACLES: a three-dimensional radiation hydrodynamics code," *A&A* 464, 429 (2007).

8. J. M. Stone, T. A. Gardiner, P. Teuben, J. F. Hawley, J. B. Simon, "Athena: a new code for astrophysical MHD," *The Astrophysical Journal Supplement Series* 178, 137 (2008).

9. U. Ziegler, "Self-gravitational adaptive mesh magnetohydrodynamics with the NIRVANA code," *A&A* 435, 385 (2005).

10. J. C. Hayes et al, "Simulating Radiating and Magnetized Flows in Multiple Dimensions with ZEUS-MP," *The Astrophysical Journal Supplement Series* 165, 188 (2006).

11. S. Jin, Z. Xin, "The relaxation schemes for systems of conservation laws in arbitrary space dimensions," *Communications on Pure and Applied Mathematics* 48, 235 (1995).

12. S. Ernst, Ph.D Thesis, University of Oregon, 2011.

13. J. N. Imamura, E. Keever, "Imogen: a parallel 3D fluid and MHD code for GPUs," *Proceedings of the 27th international ACM conference on International conference on supercomputing (ICS '13)*, 479 (2013).

14. E. F. Toro, M. Spruce, W. Speare, "Restoration of the contact surface in the HLL-Riemann solver," *Shock Waves* 4, 25 (1994).

15. S. K. Godunov, "A difference scheme for numerical computation of discontinuous solution of hyperbolic equation," *Mathematicheskii Sbornik* 47, 271 (1959).

16. P. K. Sweby, "High resolution schemes using flux-limiters for hyperbolic conservation laws," *SIAM Journal on Numerical Analysis* 21, 995 (1984).

17. A. Harten, P. D. Lax, B. van Leer, "On upstream differencing and Godunov-type schemes for hyperbolic conservation laws," *SIAM Review* 25, 35 (1983).

18. P. Batten, N. Clarke, C. Lambert, D. M. Causon, "On the Choice of Wavespeeds for the HLLC Riemann Solver," *SIAM Journal on Scientific Computing* 18, 1553 (2006).

19. P. L. Roe, "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics* 43, 357 (1981).

20. Einfeldt et al, "On Godunov-Type Methods near Low Densities," *Journal of Computational Physics* 92, 273 (1991).

21. G. Strang, "On the construction and comparison of different splitting schemes," *SIAM Journal on Numerical Analysis* 5, 506 (1968).

22. G. A. Sod, "A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws," *Journal of Computational Physics* 27, 1 (1978).

23. C. Shu, S. Osher, "Efficient Implementation of Essentially Non-Oscillatory Shock-Capturing Schemes, II," *Journal of Computational Physics* 83, 32 (1989).

24. W. Hui, P. Li, Z. Li, "A unified coordinate system for solving the two-dimensional euler equations," *Journal of Computational Physics* 153, 596 (1999).

25. R. Liska, B. Wendroff, "Comparison of Several Difference Schemes on 1D and 2D Test Problems for the Euler Equations," *SIAM Journal on Scientific Computing archive* 25, 995 (2003).

26. X. Guo, "An extended HLLC Riemann solver for the magneto-hydrodynamics including strong internal magnetic field," *Journal of Computational Physics* 290, 352 (2015).

ACKNOWLEDGEMENTS

This research would not have been possible without the continual mentorship, dedication and availability of Erik Keever. We also want to thank Professor James Imamura for the use of his lab and mentorship.