Draft version November 12, 2024
Typeset using LaTeX default style in AASTeX631

A Study on the Performance of Approximate Riemann Solvers, Integrators, and Parallelization in the Simulation Code Athena++

Kian M. Hayes[1]

[1] *Benedictine College*
1020 N 2nd Street
Atchison, KS 66002, USA

ABSTRACT

The performance of several popular Riemann solvers and time integrators in the hydrodynamics simulation code Athena++ is studied. The results of which show that the Local-Lax-Friedrich's (LLF) solver has the fastest time to completion but prominent dissipation due to the nature of this solver and its exclusion of solutions to the Riemann problem that feature shocks and other discontinuities. For integrators, we find that the VL2 integrator performs best with the lowest time to completion for all test problems. We confirm as proposed in Stone et al. (2020) that the HLLD and HLLC solvers and VL2 integrator are most suitable for general simulations involving shocks and other discontinuities in MHD and hydrodynamic problems respectively. Lastly, the parallelization capabilities of Athena++ are also tested in which we find an exponential decaying relation between the number of processes and run time of the simulation.

Keywords: Computational astrophysics, magneto-hydrodynamics, fluid dynamics, computational methods

1. INTRODUCTION

In modern computational astrophysics, studying ever-increasing complex fluid systems has become more intensive as the need for higher resolution places a growing workload on computer clusters. More efficient algorithms and software have therefore become a necessity in the field. One such code, Athena++ (Stone et al. 2020), is a leading magnetohydrodynamics (MHD) code used widely for problems ranging from protoplanetary disks as in Zhu et al. (2015), stellar evolution as in Antoni & Quataert (2022), the role of magnetic-fields in the evolution of stars and disks as in Takasao et al. (2022) and Xu & Kunz (2021), galaxy evolution in Oku et al. (2022), to cosmology in Higashi et al. (2024). The code works to solve the governing equations of fluid dynamics through a Godonuv scheme and features a plethora of Riemann solvers and time integrators. With various choices for running a simulation, finding the least computationally expensive and most accurate runtime parameters becomes a priority. We are then prompted to study the performance of these parameters. With parallelization being an important computational method, we also wanted to study the parallelization performance of the code. In this study, we specifically sought to answer the following:

a What is the relative and absolute performance of Riemann Solvers in Athena++?

b How do the various time integrator techniques in Athena++ compare against each other? Do they affect accuracy and what is their computational cost?

c What is the parallelization performance of Athena++?

We ultimately address how to get the best performance out of Athena++ for hydrodynamic and MHD problems. while simultaneously comparing the performance of Riemann Solvers and integrators. This paper is structured in the following way: in sections 2 - 6 we delve into the background of fluid dynamics and its governing equations, the Godonuv method and thereby the Riemann problem and the implementation of Riemann solvers in Athena++. We then include a description of the four test problems used in this study to test the various Riemann solvers and time integrators, the results of which are featured in Section 8. We then end with concluding remarks about this study and the most effective parameters for Athena++ simulations given our analysis.

2. FLUID DYNAMICS AND GOVERNING EQUATIONS

For the following study, numerical methods are employed to solve equations of fluid dynamics. This section outlines these governing equations for both hydrodynamics and MHD. The simulation codes make use of both Euler and Navier-Stokes Equations for inviscid and viscid flow respectively which are later adapted for the inclusion of electric and magnetic fields for MHD.

2.1. *Euler's Equations*

The equations for fluid dynamics are partial differential equations in space and time which are solved iteratively for numerical problems. Namely, for inviscid flow they are Euler's equations. In one dimension, these are written as:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} = 0 \tag{1}$$

$$\frac{\partial (\rho u)}{\partial t} + \frac{\partial (\rho u^2 + p)}{\partial x} = 0 \tag{2}$$

$$\frac{\partial E}{\partial t} + \frac{\partial [u(E+p)]}{\partial x} = 0 \tag{3}$$

where e is the total internal energy of the system, E is the total energy per unit volume written as

$$E = \rho(\frac{1}{2}V^2 + e) \tag{4}$$

and V is the specific kinetic energy written as

$$\frac{1}{2}V^2 = \frac{1}{2}V \cdot V = \frac{1}{2}(u^2 + v^2 + w^2). \tag{5}$$

These equations describe the conservation of mass, momentum, and energy in fluid systems. They can be written more compactly by defining U as a column vector of conserved variables and F as the flux vector

$$U_t + F(U)_x = 0 \tag{6}$$

where U and F are column vectors written in matrix notation as

$$U = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}, F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix} \tag{7}$$

This is common notation and will be used for the remainder of the paper when referring to conserved variables and flux.

2.2. *Conservation Laws in MHD*

For ideal MHD, the conservative equations are written as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \tag{8}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v} - \mathbf{B}\mathbf{B}) + \nabla P^* = 0 \tag{9}$$

$$\frac{\partial B}{\partial t} + \nabla \cdot (\mathbf{v}\mathbf{B} - \mathbf{B}\mathbf{v}) = 0 \tag{10}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot ((E + P^*)\mathbf{v} - \mathbf{B}(\mathbf{B} \cdot \mathbf{v})) = 0 \tag{11}$$

Where P^* is a diagonal tensor for gas pressure with components $P^* = P + B^2/2$, E is the energy density, also written as

$$E = \frac{P}{\gamma - 1} + \frac{1}{2}\rho v^2 + \frac{B^2}{2} \tag{12}$$

and $B^2 = \mathbf{B} \cdot \mathbf{B}$. These governing equations don't have easy analytical solutions which begs for the use of computational methods. There are several numerical methods to solve these equations for various dynamical systems but a common framework utilized in modern codes including Athena++ is known as the Godunov scheme. The following section outlines this numerical scheme and how it is employed in Athena++.

3. THE GODUNOV METHOD AND RIEMANN PROBLEM

The Godunov Scheme, first proposed in Godunov & Bohachevsky (1959), is the numerical scheme utilized in many modern simulation codes including Athena++ that iteratively solves the governing equations in Section 2 in a mesh grid. The mathematical derivation starts by considering a cell in the grid where the primitive variables are integrated over the cell length

$$\frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_i+1/2} U(x,t)\, dx. \tag{13}$$

The value for U at the time step $n+1$ is then calculated

$$U_i^{n+1} = U_i^n - \frac{\tau}{\Delta x}(F_{i+1/2}^n - F_{i-1/2}^n) \tag{14}$$

Where $F_{i+1/2}^n$ or $F_{i-1/2}^n$ is the flux at that respective cell interface. However, due to the discrete nature of the mesh grid, a discontinuity occurs at each cell interface which is known as the *Riemann problem*. The problem is an initial boundary value problem described mathematically as

$$\begin{aligned} \text{PDE: } & U_t + F(U)_x = 0 \\ \text{Initial Value: } & U(x,0) = U^{(0)}(x) \\ \text{Boundary Value: } & U(0,t) = U_1(t), U(L,t) = U_r(t) \end{aligned} \tag{15}$$

The structure of this is represented graphically in Figure 1. Given that this is fundamentally a step function, the problem is then what value are we to take at this interface location $x_{i+1/2}$ for the primitive variables U and therefore the flux F. Thus, to solve this problem is to *find a value for the flux* at grid cell interfaces, referred to as $F_{i+1/2}$ here on, which consequently leads to the calculation of U. Finding a solution to this problem is the fundamental step of the Godunov method. It is required that $F_{i+1/2}$ be known but the problem must be solved at every interface in the grid making it a calculation that takes up a large amount of computational power in the simulation. For example, a two-dimensional simulation with 100 by 100 grid cells in the x and y directions would require the Riemann problem to be solved 10,000 times every time step. This goes even further when considering a three-dimensional simulation. Therefore, finding efficient algorithms to make this calculation is essential. There are many methods to solve this problem both exactly and approximately which are described in the following section, much of which can be found extensively in Toro (2009).

4. RIEMANN SOLVERS

A Riemann solver is necessary to find the flux at cell interfaces in the Godunov scheme. Numerous algorithms solve this problem, making different approximations to find a solution. In this section, the different solvers utilized in this study are described starting with the Roe solver.

4.1. *Roe*

Originally proposed in Roe (1981), the Roe method makes use of a linearized equation to find a solution for the flux at the cell interface $F_{i+1/2}$. It has gone through many iterations since its first introduction but here the original approach by Roe will be outlined. Roe introduces the Jacobian Matrix denoted as

$$A(U) = \frac{\partial F}{\partial U} \tag{16}$$

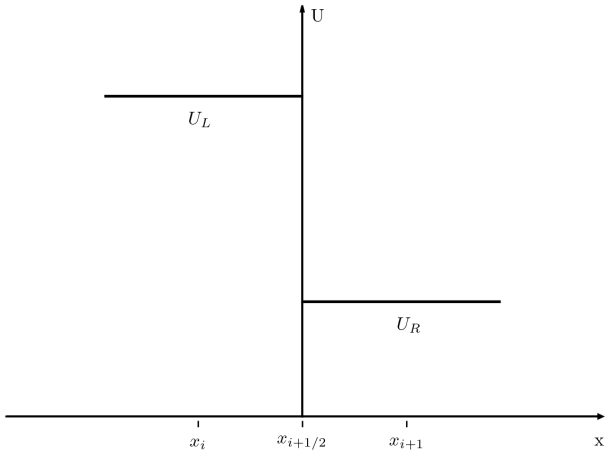

Figure 1. A graphical representation of the Riemann Problem

which allows for (6) to be written as

$$U_t + A(U)U_x = 0 \tag{17}$$

in which Roe replaces the matrix $A(U)$ with a constant one

$$\tilde{A} = \tilde{A}(U_L, U_R). \tag{18}$$

This allows for (17) to be written in a linear form with

$$U_t + \tilde{A}U_x = 0 \tag{19}$$

$$U(x,0) = \begin{cases} U_L, x \leq 0 \\ U_R, x \geq 0 \end{cases}$$

This is now simply an eigenvalue-eigenvector equation in which solving for the eigenvalues $\tilde{\lambda}_i$ of the Jacobian matrix $\tilde{A}$ allows for the Riemann problem to be solved exactly with

$$F_{i+1/2} = \frac{1}{2}(F_L + F_R) - \frac{1}{2}\sum_{i=1}^{m} \tilde{\alpha}_i |\tilde{\lambda}_i| \tilde{K}^{(i)} \tag{20}$$

where m, $\tilde{\alpha_i}$, $\tilde{\lambda}_i$, and $\tilde{K}^{(i)}$ are respectively the number of conservation laws to be solved, wave strength, the eigenvalues of $\tilde{A}$, and the right eigenvectors. The method for obtaining these values is outlined further in Toro (2009).

4.2. *LLF*

The local Lax-Friedrichs (LLF) Riemann solver is a simple solver that assumes that the wave speeds on either side of a cell interface are equal and opposite in that

$$-S_1 = S_2 = a \tag{21}$$

The wave speed is chosen to be no smaller than the maximum wave speed given as a function of the flux where

$$a = max(|\lambda^1(U_L)|, |v_x - c_s|, |v_x + c_s|, \lambda^M(U_R)) \tag{22}$$

This maximum wave speed we use in finding the interface flux with

$$F^{LLF} = \frac{1}{2}(F_R + F_L) - \frac{S_{max}}{2}(U_R - U_L) \tag{23}$$

4.3. *HLLE and HLLC*

The HLLE solver was one of the initial algorithms for solving the Riemann Problem, introduced in Harten et al. (1983). The basic formulation of the HLLE solver is finding $F_{i+1/2}$ (as described in Section 3) by utilizing the left and right wave speeds to approximate the state of the column vector U^{HLLE} at the interface. The structure of this solution can be found in Figure 2. The state at the interface is said to be between the left and right states, but all solutions are brought together to approximate U^{HLLE} which is then used to solve for $F_{i+1/2}$ using

$$F^{HLLE} = F_L + S_L(U^{HLLE} - U_L) \tag{24}$$

or

$$F^{HLLE} = F_r + S_R(U^{HLLE} - U_R) \tag{25}$$

We can combine these two equations by applying Rankine–Hugoniot conditions across the left and right waves respectively to get

$$F^{HLLE} = \frac{S_R F_L - S_L F_R + S_L S_R (U_R - U_L)}{S_R - S_L} \tag{26}$$

Using this, the intercell flux $F^{HLLE}_{i+\frac{1}{2}}$ then can be written as

$$F^{HLLE}_{i+\frac{1}{2}} = \begin{cases} F_L & 0 \leq S_L \\ \frac{S_R F_L - S_L F_R + S_L S_R (U_R - U_L)}{S_R - S_L} & S_L \leq 0 \leq S_R \\ F_R & 0 \leq S_R \end{cases} \tag{27}$$

Where the intercell flux solution is determined conditionally by the wave speed of the left and right cells. This equation for flux can then be employed to complete the Godonuv method. Further details on this method and how it compares to its linearized counterpart can be found in Einfeldt et al. (1991). However, this method is more common to be a basis for others, as it introduces into the system what is commonly referred to as "dissipation" of the fluid due to the exclusion of solutions that have discontinuities. These discontinuities are necessary to accurately model a fluid system so their inclusion in the solution of the Riemann problem are paramount. Toro et al. (1994) introduces the HLLC solver to remedy this. The HLLC solver builds upon the HLLE solver by reincorporating so-called *intermediate waves* or *contact waves*. These intermediate waves encompass the discontinuities of shocks and rarefactions. The new solution structure can be found in Figure 3, also known as a *Riemann fan*. In the HLLC solver, flux is defined conditionally as

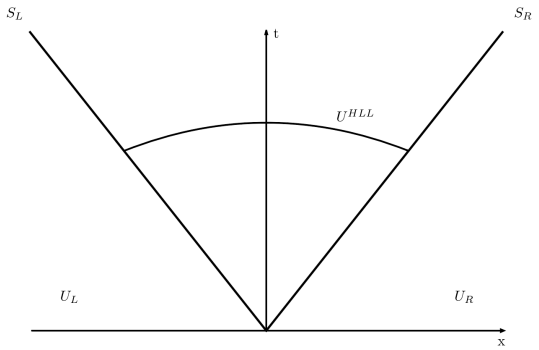

Figure 2. A Riemann fan that shows the solution structure of the HLLE solver. The solver seeks to find a solution between the S_L and S_R waves. This is also a common way to represent the solutions to a Riemann problem. A diagonal line represents wave speeds present in the simulation at a specific cell.

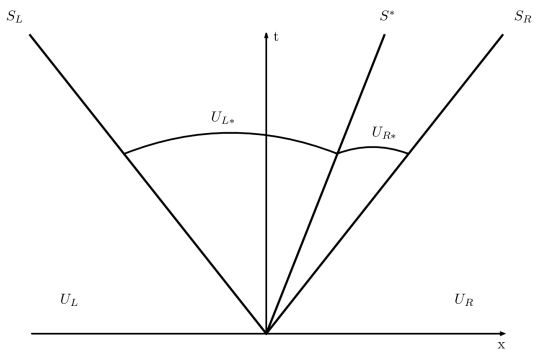

Figure 3. A Riemann fan showing the solution structure of the HLLC solver. Now we are considering an intermediate wave speed between S_L and S_R. The final flux is determined conditionally based on these wave speeds to capture discontinuities like shocks and rarefactions.

$$F_{i+\frac{1}{2}}^{HLLC} = \begin{cases} F_L & 0 \leq S_L \\ F_L^* & S_L \leq 0 \leq S_* \\ F_R^* & S_* \leq 0 \leq S_R \\ F_R & 0 \leq S_R \end{cases} \tag{28}$$

4.4. *HLLD*

The Harten-Lax-van Leer discontinuities solver is a Riemann solver solely for ideal MHD, first proposed in Miyoshi & Kusano (2005). Similarly to the Roe solver, we consider intermediate states between S_L and S_R except the HLLD solver allows for five different states for U in the Riemann fan instead of just one. Across these five states, the pressure P_t^* and normal wave speed S_M are assumed to be constant with

$$P_t^* = \frac{(S_R - u_R)\rho_R P_{tL} - (S_L - u_L)\rho_L P_{tR} + \rho_L \rho_R (S_R - u_R)(S_L - u_L)(u_R - u_L)}{(S_R - u_R)\rho_R - (S_L - u_L)\rho_L} \tag{29}$$

$$S_M = \frac{(S_R - u_R)\rho_R u_R - (S_L - u_L)\rho_L u_L - (P_{tR} - P_{tL}}{(S_R - u_R)\rho_R - (S_L - u_L)\rho_L} \tag{30}$$

By calculating these constants, the numerical flux is calculated using the five intermediate states in the Riemann fan with

$$F^{HLLD} = \begin{cases} F_L & S_L \geq 0 \\ F_L^* & S_L \leq 0 \leq S_{*L} \\ F_L^{**} & S_{L*} \leq 0 \leq S_M \\ F_R^{**} & S_M \leq 0 \leq S_{R*} \\ F_R^* & S_{R*} \leq 0 \leq S_R \\ F_R & S_R \geq 0 \end{cases} \tag{31}$$

4.5. *LHLLD/LHLLC*

The low-dissipation HLLD solver (LHLLC) was first proposed in Minoshima & Miyoshi (2021) which takes a variation on how the pressure and normal wave speed are calculated. Instead of equations (29) and (30), correction factors are added to the equations, making them

$$P_t^* = \frac{(S_R - u_R)\rho_R P_{tL} - (S_L - u_L)\rho_L P_{tR} + \phi\rho_L \rho_R (S_R - u_R)(S_L - u_L)(u_R - u_L)}{(S_R - u_R)\rho_R - (S_L - u_L)\rho_L} \tag{32}$$

$$S_M = \frac{(S_R - u_R)\rho_R u_R - (S_L - u_L)\rho_L u_L - \theta(P_{tR} - P_{tL}}{(S_R - u_R)\rho_R - (S_L - u_L)\rho_L} \tag{33}$$

These are then used in solving the algebraic system of wave states to obtain a value for the interface flux with

$$F^{LHLLD} = F^{HLLD} - \frac{(\phi - 1)\bar{\rho} c_f \Delta x}{2}\left[0, \frac{\Delta u}{\Delta x}, 0, 0, 0, 0, (1 + M)\frac{\Delta(u^2/2}{\Delta x}\right]^T \tag{34}$$

For greater detail, we suggest reviewing Minoshima & Miyoshi (2021).

5. TIME INTEGRATORS

Time integrators in hydrodynamic simulations serve the purpose of advancing the simulation through time. After we solve for primitive variables at a certain time step, the time integrator will move our simulation from a time step n to a time step $n + 1$. In numerical methods, numerous approximations can be made to make this calculation from one time step to another and Athena++ includes several. In this study we seek to compare these different methods and how they affect a simulation. The simplest method is that of Euler. Consider the initial value problem

$$\frac{dy}{dt} = f(y, t) \tag{35}$$

$$y(t_0) = y_0 \tag{36}$$

If we want to know the value of y at t_{n+1} (y_{n+1}) then we can say

$$y_{n+1} = y_n + \Delta t f(t_n, y_n) \tag{37}$$

Where Δt is the increment from t_n to t_{n+1}. However, this method is typically inaccurate for non-smooth functions, and given that we have highly discontinuous functions in the solutions of hydrodynamic equations, higher-order methods are required. This then leads to the formulation of the Strong stability preserving Runge-Kutta (SSPRK) methods, namely the two-stage second-order, third-stage third-order, and fifth-stage fourth-order methods (Ketcheson 2010):

SSPRK (2,2):

$$u^{(1)} = u^n + \Delta t F(u^n) \tag{38}$$

$$u^{n+1} = \frac{1}{2}u^n + \frac{1}{2}u^{(1)} + \frac{1}{2}\Delta t F(u^n)$$

SSPRK (3,3):

$$u^{(1)} = u^n + \Delta t F(u^n) \tag{39}$$

$$u^{(2)} = \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t F(u^{(1)})$$

$$u^{n+1} = \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t F(u^{(2)})$$

SSPRK(5,4)

$$u^{(1)} = u^n + 0.391\Delta t F(u^n) \tag{40}$$

$$u^{(2)} = 0.444u^n + 0.555u^{(1)} + 0.368\Delta t F(u^{(1)})$$

$$u^{(3)} = 0.620u^n + 0.380u^{(2)} + 0.252\Delta t F(u^{(2)})$$

$$u^{(4)} = 0.178u^n + 0.822u^{(3)} + 0.545\Delta t F(u^{(3)})$$
$$u^{n+1} = 0.517u^{(2)} + 0.096u^{(3)} + 0.064\Delta t F(u^{(3)}) + 0.387u^{(4)} + 0.226\Delta t F(u^{(4)})$$

These methods take approximations around the point of interest to calculate the next value u^{n+1}. Lastly, there's also a predictor-corrector method that combines the methods of van Leer (VL) and constrained transport (CT) known as the second-order van Leer integrator (VL2) proposed in Stone & Gardiner (2009). A predictor-corrector method differs from the Runge-Kutta method in that we use the half-time step in order to get a value at $n+1$. Take the discretization of the conservative advection equation:

$$\frac{\partial U}{\partial t} = -\frac{1}{\Delta x}(F(U)_{i+1/2} - F(U)_{i-1/2}) \tag{41}$$

This equation updates in time through the boundary of the zone. Therefore, we can also say

$$\frac{\partial U}{\partial t} = \frac{(U^{n+1} - U^n)}{\Delta t} \tag{42}$$

To get second-order accuracy of U^{n+1} we must evaluate (41) at the half-time step $n+1/2$. After doing this and equating (41) and (42)

$$\frac{U^{n+1} - U^n}{\Delta t} = -\frac{F(U)^{n+1/2}_{i+1/2} - F(U)^{n+1/2}_{i-1/2}}{\Delta x} \tag{43}$$

By solving for U^{n+1}, we get

$$U^{n+1} = U^n - \frac{\Delta t}{\Delta x}(F(U)^{n+1/2}_{i+1/2}) \tag{44}$$

What values are we to take for these interface fluxes though? This is where the predictor part of the method comes in. We then say that

$$F(U)^{n+1/2}_{i+1/2} = F(U^{n+1/2}_{i+1/2}) \tag{45}$$

To acquire this $U^{n+1/2}_{i+1/2}$, we Taylor expand the data in the cell to the interface and update the variable U^n which is the corrector step of this method. Further details on this method can be found in Londrillo & Zanna (2000) and Zingale (2017). This is the basis of the VL2 integrator. The main feature of this method is its use of first-order fluxes which are used to calculate second-order interface states with spacial reconstruction. Based on these higher-order calculations, the value for conserved variables at $t = t_{n+1}$ is determined. Namely, in three dimensions, the conserved variable matrix at t_{n+1}, U^{n+1} is:

$$U^{n+1}_{i,j,k} = U^n_{i,j,k} - \frac{\Delta t}{\Delta x}(F^{n+1/2}_{i+1/2,j,k} - F^{n+1/2}_{i-1/2,j,k}) - \frac{\Delta t}{\Delta y}(G^{n+1/2}_{i+1/2,j,k} - G^{n+1/2}_{i-1/2,j,k}) - \frac{\Delta t}{\Delta z}(H^{n+1/2}_{i+1/2,j,k} - H^{n+1/2}_{i-1/2,j,k}) \tag{46}$$

6. NUMERICAL METHOD AND ATHENA++

Athena++ (Stone et al. 2020) is an open-source simulation code built for solving the governing equations of fluid dynamics in the Godonuv scheme using SMR/AMR and comes with a suite of test problems including those described in the above section. It was chosen for its ease of use and widely available support for the various topics studied in this paper.

To run Athena++, an executable is created after running configuration commands and the MAKE file. This step is where the compilation occurs which allows for changing the test problem and Riemann solver. This is also where the compilation command can be altered to have Athena++ compiled with MPI. Unless otherwise stated, all simulations conducted in this study were ran on the Stony Brook University ELECTRA system, compiled with MVAPICH Version 4.1.2 and with 16 MPI processes, the most processes available on the ELECTRA system. After which, the executable is ran, taking in an input file where specific simulation parameters can be changed. Specifically, this is where the integrator can be changed along with properties regarding the limits, mesh grid, and initial conditions. Details regarding the runtime parameters and input file for each test conducted in this study can be found in tables in the Appendix.

7. DESCRIPTION OF TEST PROBLEMS

The test problems employed by this study include two hydrodynamic tests and two MHD test. Namely, the Rayleigh-Taylor Instability (Liska & Wendroff 2003a), Liska & Wendroff Implosion (Liska & Wendroff 2003b), Lecoanet Kelvin-Helmholtz Instability (Lecoanet et al. 2016), and MHD Spherical Blast (Gardiner & Stone 2005) problems. These were chosen to fully study the hydrodynamic and MHD capabilities of Athena++ and their scalability. To make a problem more or less computationally intensive is a matter of changing the simulation time, boundary size, or number of cells. All of which can be edited easily in the respective Athena input file. These problems also serve as standard tests for fluid discontinuities in most simulation codes.

Rayleigh-Taylor Instability: The Rayleigh-Taylor Instability is a common phenomena in fluids in which a fluid of higher density interacts with a fluid of comparatively lower density. It is often a source of mixing and turbulence for many astrophysical systems and is common test problem due to its discontinuous nature. The simulation setup is based on that which is found in Liska & Wendroff (2003a). The initial interface of the fluid is at y = 0 and has an initial perturbation in the y-direction with a gravitational force of $g = 0.1$ added in the momentum equation.

Implosion: The second hydro test problem used in this study is the 2D implosion test as outlined in Liska & Wendroff (2003b). The main feature of this problem is a fluid with a diamond-shaped interface at the center of the boundaries with a density ρ_i and a separate density ρ_e surrounded by this inner density. The effect of this initial condition is a collapse of the fluid toward the center of the box, causing an implosion. The test in this study is of the variation as seen in Stone et al. (2020) where the first quadrant of the aforementioned box is the boundary rather than the full diamond as seen in the original paper.

Lecoanet Kelvin-Helmholtz Instability: The Kelvin-Helmholtz Instability is another common phenomena in fluids that occurs when there is velocity shear between two fluids with different speeds. The instability also provides conditions for testing highly discontinuous systems in simulation codes. The instability has a cascading effect in fluids making it a prominent source of mixing and turbulence. The particular variation of this instability used in this study is one described in Lecoanet et al. (2016) which has the following initial conditions:

$$
\begin{aligned}
\rho &= 1 + \frac{\Delta\rho}{\rho_0}\left(\frac{1}{2}\tanh\left(\frac{z-z_1}{a}\right) - \tanh\left(\frac{z-z_2}{a}\right)\right) \\
u_x &= u_{\text{flow}}\left(\tanh\left(\frac{z-z_1}{a}\right) - \tanh\left(\frac{z-z_2}{a}\right) - 1\right) \\
u_z &= A\sin(2\pi x)\left(\exp\left(-\frac{(z-z_1)^2}{\sigma^2}\right) + \exp\left(-\frac{(z-z_2)^2}{\sigma^2}\right)\right) \\
P &= P_0 \\
c &= \frac{1}{2}\left(\tanh\left(\frac{z-z_2}{a}\right) - \tanh\left(\frac{z-z_1}{a}\right) + 2\right)
\end{aligned}
\tag{47}
$$

Respective constants reflect that which is found in Lecoanet et al. (2016) with $a = 0.05$, $\sigma = 0.2$, $u_{flow} = 1$, and $P_0 = 10$ so that the flow is subsonic with a Mach number $M \approx 0.25$ in regions with $\rho = 1$ and $M \approx 0.35$ in regions with $\rho = 2$. Lastly, the amplitude of the initial velocity perturbation is $A = 0.01$.

MHD Blast: The last MHD test problem conducted in this study is the MHD Blast problem as found in Gardiner & Stone (2005) and Londrillo & Zanna (2000). It considers the explosion of a centrally over-pressurized region into a low-pressure, low-ambient medium. Exact initial conditions such as the ratio between ambient and the central pressures, radius of the inner sphere, and the magnetic field strength can be found in the Appendix under Table 1. These initial conditions result in the propagation of two strong shocks that are aligned to the present magnetic field. The strength of these shocks makes it a useful test of Riemann solvers and their ability to resolve such discontinuities in an MHD system.

8. RESULTS

The following section entails the results from the Riemann solvers, integrators, and processes tests conducted in this study. We discuss mainly the qualitative difference between simulations for the Riemann solvers and integrators tests, along with reasoning for apparent differences. For these two tests specifically, we report the performance of the Riemann solvers and integrators by their time to completion for each test problem. This serves as a quantitative comparison between the various solvers or integrators that were studied.

8.1. *Solvers Test*

The solvers test was conducted to study the performance of each Riemann solver described in section 4. The four test problems described in Section 7 were run with identical parameters except for changing the Riemann solver which is done when running the configuration script for Athena++. The test shows low computational power is used for the LLF and HLLE solvers compared to others as seen in 4 and **??**, but a large amount of dissipation. This dissipation is most apparent in Figures 6 and 7. In the figures, an indication of this dissipation can be found around the edges of the fluid. In the LLF and HLLE panels, we see that these edges appear to be "blurred". Comparatively, the HLLC, LHLLC, and Roe solvers have sharp edges around the fluid. Such behavior is expected due to the nature of the respective solvers and their respective formulations. Interestingly, different structures can be seen at the same time step fo the HLLC, LHLLC, and Roe solvers. This is especially apparent in Figure 5. A jet-like structure appears to come from the fluid in the corner that has more speed than the HLLC and Roe solvers. This phenomenon does not appear for the MHD Blast problem.

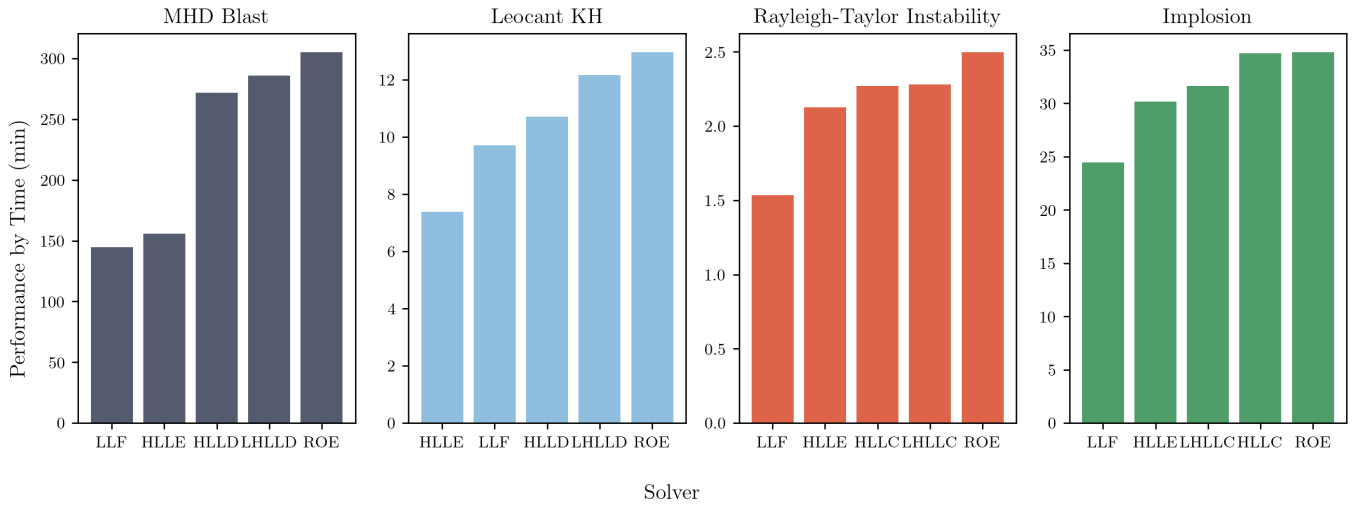

Figure 4. Bar graph showing the computation time for both Hydro and MHD test problems with the different Riemann solvers sorted from least to greatest for each plot. Notably, the LLF takes the least amount of time across all test problems except for the KH instability test, while the Roe is the longest for each.

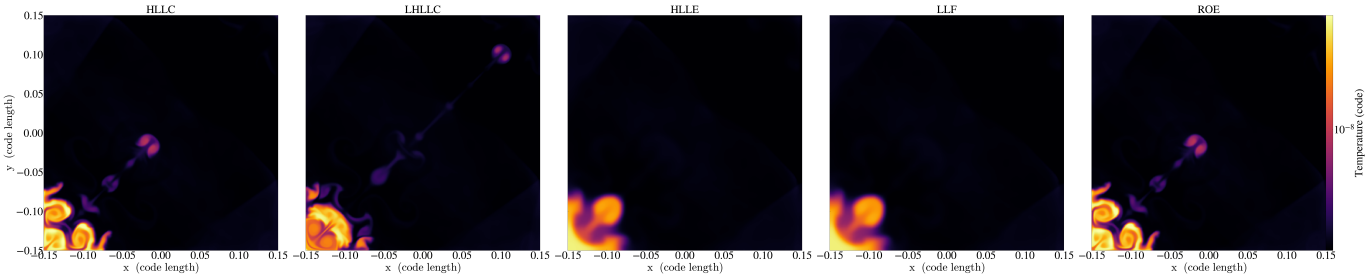

Figure 5. The Implosion problem for Solvers test at a simulation time of $t = 5$. From left to right we present different simulations that use the HLLC, LHLLC, HLLE, LLF, and Roe solvers. Strong dissipation can be seen in the HLLE and LLF solvers.

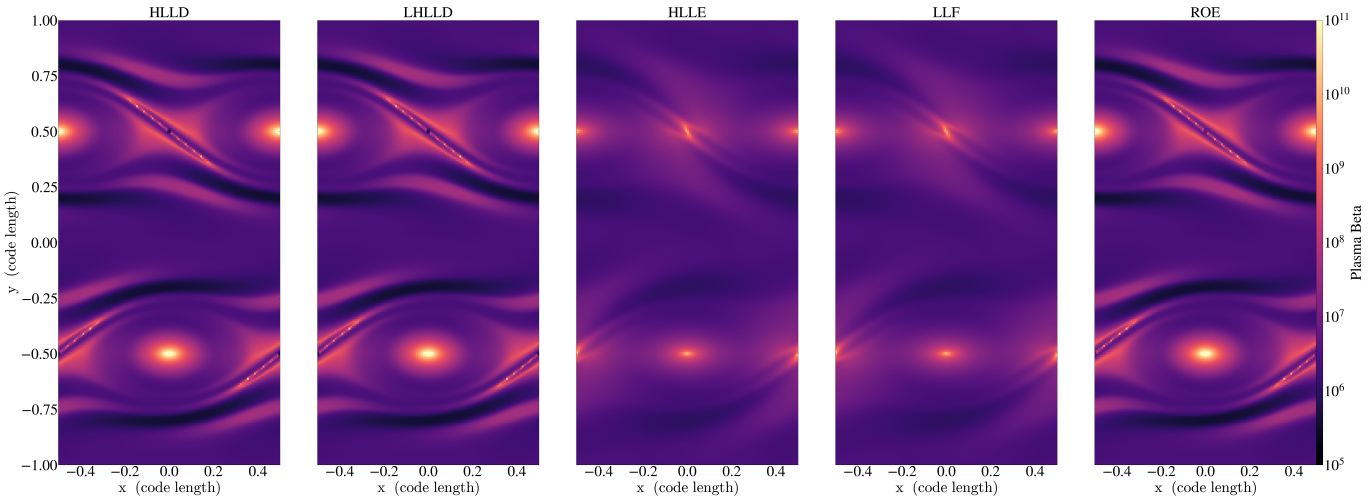

Figure 6. KH Instability Test problem for the solvers test at $t = 10$. These simulations again show the dissipation in HLLE and LLF.

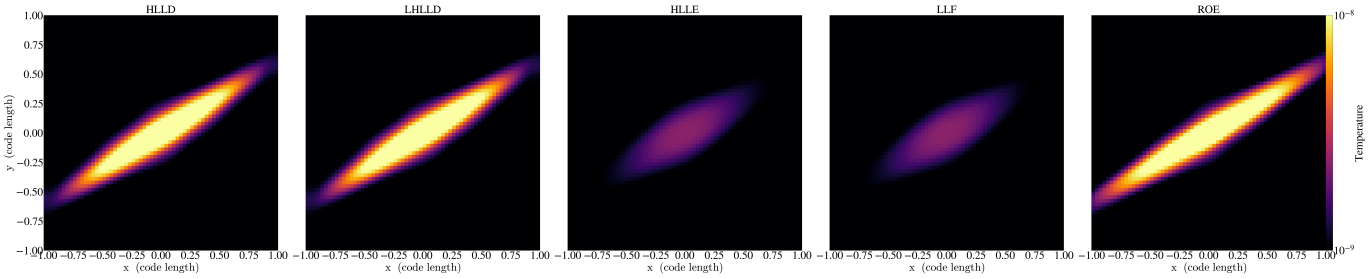

Figure 7. MHD Blast test problem for the solvers test at $t = 5$. Another case of dissipation for the HLLE and LLF solvers.

8.2. *Integrators Test*

In the integrators test, the two hydro test problems were used to compare the performance of integrators discussed in 5. We opted to implement the HLLC Riemann solver for all simulations conducted in this test. Figures 8 and **??** show the results of these simulations at $t = 5$ and $t = 10.4$ respectively. Structure differences between these integrators are not apparent although the VL2 integrator appears to show the largest difference between the SSPRK counterparts. With that, the performance by time bar graph in 9 shows that the VL2 has the best capabilities even with the small structural differences in the simulations. This confirms what is shown in Stone et al. (2020).

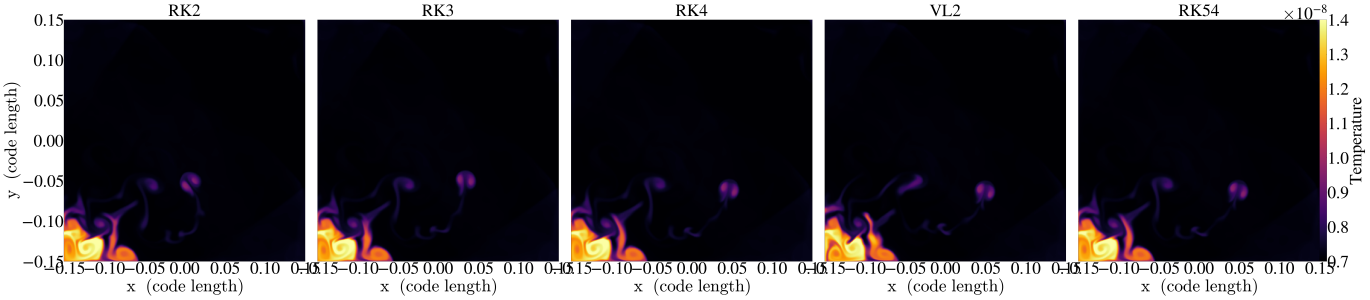

Figure 8. Caption

8.3. *Parallelization Test*

The last test done in this study was conducted to find the parallelization capabilities of Athena++. For this, we utilized the Rayleigh-Taylor Instability test problem at varying computational costs with Test 1 having the lowest

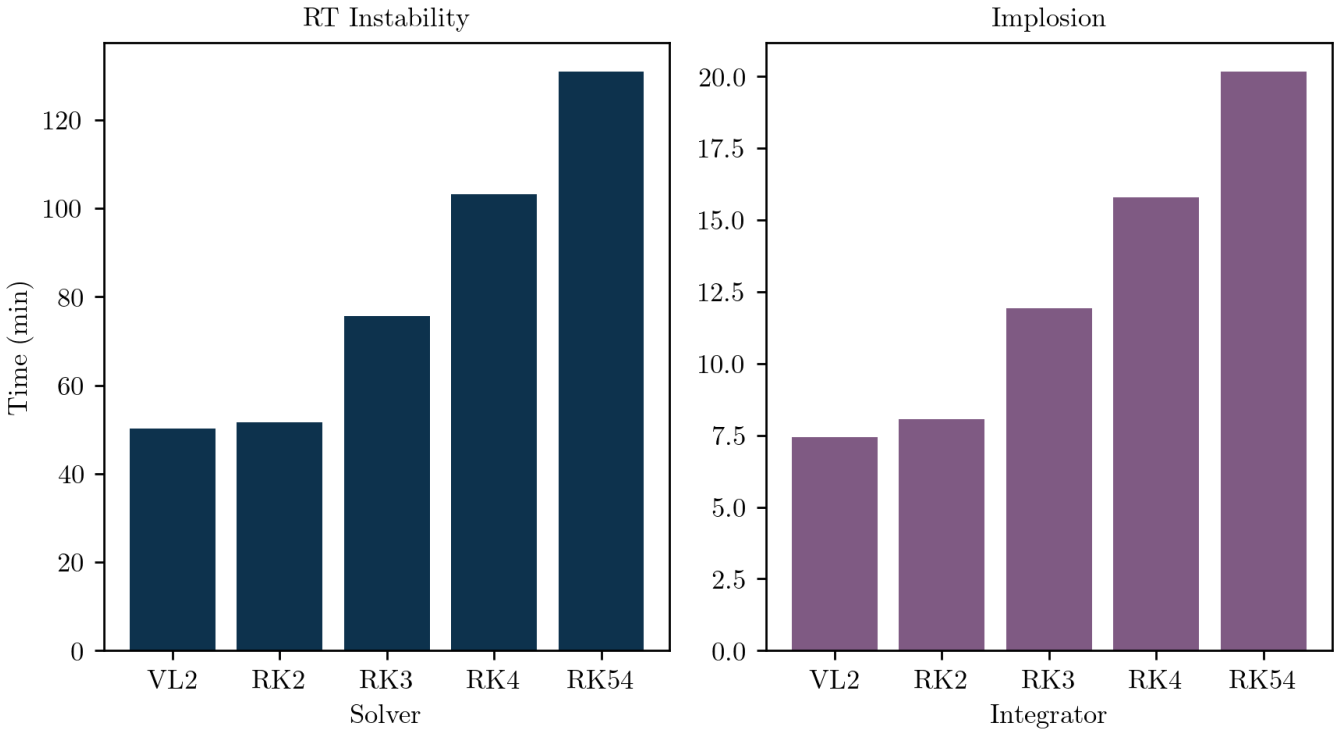

Figure 9. Caption

computational cost and Test 3 having the highest. This was done by increasing the resolution of the mesh grid or the simulation runtime. Specifics of these three tests can be found in Table REFERENCE TO TABLE. Across these three tests, a run of the simulation was done at varying MPI processes from one to 32. Results of these tests are found in Figure 10 in which we see an exponential decay relation between the number of processes and the time to completion. This is a typical relationship we see in tests that scale the number of processes on computational processes, so was expected.

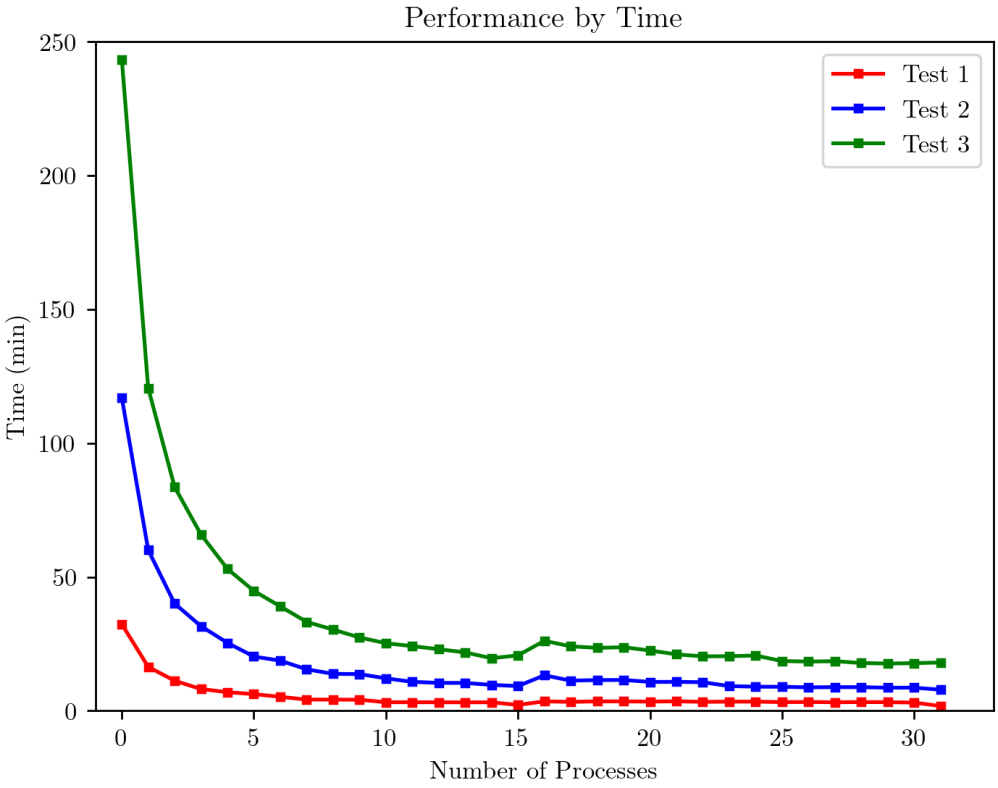

Figure 10. Data for Parallelization Test which shows decay relationship between the number of processes and the time to completion. The Rayleigh-Taylor Instability was used for this portion of the study. The runtime parameters of which can be found in the Appendix Table.

9. CONCLUSION

We've presented a study on the performance of Riemann solvers and integrators in the hydrodynamic code Athena++ along with its parallelization capabilities with MPI. The paper also serves as a review of the modern scheme for solving hydrodynamic and MHD problems in simulation codes in the Godnonuv method and consequently the Riemann problem. We broadly discussed the differences between Riemann solvers present in Athena++, namely the different approximations that are made in order to solve the Riemann problem. We direct those more interested in an in-depth discussion on these solvers to their respective papers and especially Toro (2009).

The study conducted in this paper consists of three tests; the solvers, integrators, and parallelization, all of which utilize different test problems that come with Athena++. The solver's test implements the discussed Riemann solvers across 4 different test problems that come with Athena++. We measured the performance of these Riemann solvers by inspecting their simulation results and the time it takes to complete the full calculation of the simulation. With that, we find that the LLF solver performs the best in the time to-completion metric, but features a large amount of dissipation which is expected with this particular solver. We then conclude that the HLLC solver for hydro problems and HLLD solver for MHD problems are better suited for capturing discontinuities such as shocks and rarefactions while balancing computational costs. When comparing these solvers to the exact method of Roe, we find that the HLLC and HLLD perform on a similar level while having less computational strain.

In the integrators study, we compare the various time integrators that are included in Athena++ in the two hydro problems discussed in Section 7. This includes the Strong Stability Preserving Runge-Kutta methods RK1, RK2, RK3, RK54, as well as a predictor-corrector method in VL2. Across these simulations, we find that there's hardly any variation in present fluid structures between the integrators. The main difference between these is their computational cost, in which the VL2 integrator performs the best which is expected given the low number of approximation calculations it makes whereas the SSPRK54 integrator uses 5 different calculations to arrive at its output. With that, the SSPRK54 integrator performs the worst.

Finally, our last test studies the parallelization capabilities of Athena++ in which 3 separate simulations were run at increasing MPI processes from one to 32. The 3 simulations had increasing computational costs to get a full scope of the code as calculations became more involved. With that, we find a typical exponential decay relation between the number of processes and the time to completion.

In the grand scheme of our study, we find the best combination of Riemann solver and integrator for both hydro and MHD problems when using Athena++. That being, the HLLC solver and VL2 integrator for hydro problems, and the HLLD solver and VL2 integrator for MHD problems.

Software: yt Project for visualizations (Turk et al. 2011), Athena++ (Stone et al. 2020). The GitHub repository containing the Python and Bash scripts to conduct this study along with simulation videos for the various test problems can be found at https://github.com/kianhayes/summer2024.

APPENDIX

A. SOLVERS TEST SIMULATION PARAMETERS

Mesh	
nx1	64
nx2	64
nx3	64
Mesh Blocks/Limits	
nx1	16
nx2	16
nx3	16
xmax	1
xmin	-1
ymax	1
ymin	-1
zmax	1
zmin	-1
Time	
Simulation Time	1
CFL	0.3
Time integrator	VL2
Problem	
Ambient Pressure	0.1
Initial Pressure Ratio	100
Radius of Inner Sphere	0.1
B_0	1
Angle	30
Hydro	
gamma	1.67
Isothermal Sound Speed	0.4082

Table 1. MHD Blast problem

Mesh	
nx1	128
nx2	256
Mesh Blocks/Limits	
nx1	64
nx2	128
xmax	1
xmax	0.5
ymax	1
ymin	-1
Time	
Simulation Time	10
CFL	0.4
Time integrator	VL2
Problem	
Re	10^5
iprob	4
amp	0.01
b0	0.01
vflow	1
vboost	0
drho_rho0	0
nu_iso	2.00E-05
nu_scalar_iso	2.00E-05
kappa_iso	2.00E-05
Hydro	
gamma	1.4

Table 2. Kelvin-Helmholtz Instability

Mesh	
nx1	400
nx2	400
Mesh Blocks/Limits	
xmax	0
xmin	0.3
ymax	0
ymin	0.3
Time	
Simulation Time	5
CFL	0.4
Time integrator	VL2
Problem	
Inside Density	0.125
Inside Pressure	0.14
Outside Density	1
Outside Density	1
Hydro	
gamma	1.4

Table 3. Implosion problem

Mesh	
nx1	200
nx2	400
Mesh Blocks/Limits	
nx1	50
nx2	50
xmax	0.16666667
xmin	-0.16666667
ymax	0.3
ymin	-0.3
Time	
Simulation Time	30
CFL	0.4
Time integrator	VL2
Problem	
iprob	1
amp	0.01
drat	2.0
Hydro	
iso_sound_speed	1.0
gamma	1.4
grav_acc2	-0.1

Table 4. Rayleigh-Taylor Instability problem

B. INTEGRATORS TEST SIMULATION PARAMETERS

Mesh	
nx1	400
nx2	400
nx3	1
Mesh Blocks/Limits	
nx1	8
nx2	8
nx3	8
xmax	0.3
xmin	0
ymax	0
ymin	0.3
zmax	NA
zmin	NA
Time	
Simulation Time	5
CFL	0.4
Problem	
d_in	0.125
p_in	0.14
d_out	1
p_out	1
Hydro	
gamma	1.4

Table 5. Implosion problem

Mesh	
nx1	200
nx2	400
nx3	1
Mesh Blocks/Limits	
nx1	2
nx2	2
nx3	2
xmax	0.167
xmin	-0.167
ymax	0.3
ymin	-0.3
zmax	NA
zmin	NA
Time	
Simulation Time	30
CFL	0.4
Problem	
iprob	1
amp	0.01
drat	2
Hydro	
gamma	1.4
iso sound speed	1.00E+00
grav acc2	-1.00E-01

Table 6. Rayleigh-Taylor Instability problem

REFERENCES

Antoni, A., & Quataert, E. 2022, MNRAS, 511, 176, doi: 10.1093/mnras/stab3776

Einfeldt, B., Munz, C. D., Roe, P. L., & Sjögreen, B. 1991, Journal of Computational Physics, 92, 273, doi: 10.1016/0021-9991(91)90211-3

Gardiner, T. A., & Stone, J. M. 2005, Journal of Computational Physics, 205, 509, doi: 10.1016/j.jcp.2004.11.016

Godunov, S. K., & Bohachevsky, I. 1959, Matematičeskij sbornik, 47(89), 271. https://hal.science/hal-01620642

Harten, A., Lax, P., & van Leer, B. 1983, SIAM Rev, 25, 35

Higashi, S., Susa, H., Federrath, C., & Chiaki, G. 2024, ApJ, 962, 158, doi: 10.3847/1538-4357/ad2066

Ketcheson, D. I. 2010, Journal of Computational Physics, 229, 1763, doi: https://doi.org/10.1016/j.jcp.2009.11.006

Lecoanet, D., McCourt, M., Quataert, E., et al. 2016, Monthly Notices of the Royal Astronomical Society, 455, 4274, doi: 10.1093/mnras/stv2564

Liska, R., & Wendroff, B. 2003a, SIAM Journal on Scientific Computing, 25, 995, doi: 10.1137/S1064827502402120

—. 2003b, SIAM Journal on Scientific Computing, 25, 995, doi: 10.1137/S1064827502402120

Londrillo, P., & Zanna, L. D. 2000, The Astrophysical Journal, 530, 508, doi: 10.1086/308344

Minoshima, T., & Miyoshi, T. 2021, Journal of Computational Physics, 446, 110639, doi: https://doi.org/10.1016/j.jcp.2021.110639

Miyoshi, T., & Kusano, K. 2005, Journal of Computational Physics, 208, 315, doi: https://doi.org/10.1016/j.jcp.2005.02.017

Oku, Y., Tomida, K., Nagamine, K., Shimizu, I., & Cen, R. 2022, ApJS, 262, 9, doi: 10.3847/1538-4365/ac77ff

Roe, P. 1981, Journal of Computational Physics, 43, 357, doi: https://doi.org/10.1016/0021-9991(81)90128-5

Stone, J. M., & Gardiner, T. 2009, NewA, 14, 139, doi: 10.1016/j.newast.2008.06.003

Stone, J. M., Tomida, K., White, C. J., & Felker, K. G. 2020, ApJS, 249, 4, doi: 10.3847/1538-4365/ab929b

Takasao, S., Tomida, K., Iwasaki, K., & Suzuki, T. K. 2022, ApJ, 941, 73, doi: 10.3847/1538-4357/ac9eb1

Toro, E. F. 2009, Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction (Berlin, Heidelberg: Springer), doi: 10.1007/b79761

Toro, E. F., Spruce, M., & Speares, W. 1994, Shock Waves, 4, 25, doi: 10.1007/BF01414629

Turk, M. J., Smith, B. D., Oishi, J. S., et al. 2011, ApJS, 192, 9, doi: 10.1088/0067-0049/192/1/9

Xu, W., & Kunz, M. W. 2021, MNRAS, 508, 2142, doi: 10.1093/mnras/stab2715

Zhu, Z., Dong, R., Stone, J. M., & Rafikov, R. R. 2015, ApJ, 813, 88, doi: 10.1088/0004-637X/813/2/88

Zingale, M. 2017, Introduction to Computational Astrophysical Hydrodynamics (Open Astrophysics Bookshelf)