Adaptive mesh refinement for the compressible Euler equations using AMReX

Gianfranco Grillo

Cavendish Laboratory, Department of Physics, J J Thomson Avenue, Cambridge. CB3 0HE

Abstract

We investigate the effects of adaptive mesh refinement by solving the compressible Euler equations of hydrodynamics within the framework of the AMReX software package [1]. We first run a number of one dimensional validation tests at different resolutions with and without mesh refinement and quantify convergence behavior in both cases. We proceed to extend the analysis to two dimensions and perform a two dimensional timing exercise to quantify the speed up in simulation time obtained via mesh refinement and parallelization. Our results illustrate the capabilities and limitations of the adaptive mesh refinement approach.

1. Introduction

A simplification of the more general Navier-Stokes equations, the compressible Euler equations provide an accurate description of the evolution of the flow of compressible materials such as gases and liquids in which the flow velocity approaches or exceeds the speed of sound, which is common in fields such as aeronautics and astrophysics [2], and in which heat exchange can be considered negligible. The coupled, nonlinear nature of the equations gives rise to highly nonlinear behavior and discontinuities that are often difficult to capture numerically, requiring large grid resolutions and therefore the usage of a considerable amount of computational resources. In an Eulerian framework, in which the computational grid is independent of the material inside it, a popular method of ameliorating this problem involves using a non-uniform grid resolution in order to properly capture those features of the flow that require high resolution while at the same time using a coarser grid to model the behavior of those parts of the fluid that vary more smoothly. This is known as Adaptive Mesh Refinement (AMR), and was originally introduced by Berger and Oliger [3], and extended by Berger and Colella [4], with the explicit purpose of better capturing discontinuities produced by hyperbolic systems of equations like the Euler equations at a lower computational cost. Since then, the approach has been applied in a number of different contexts, including accelerator design, additive manufacturing, astrophysics, cosmology, wind plant modeling, and electrodynamics, among others [1], which illustrates the power of the paradigm.

In this work, we demonstrate the usefulness of the adaptive mesh refinement approach in the context of solving the compressible Euler equations in one and two dimensions, specifically by using adaptive mesh refinement as implemented in the popular AMReX software package[1]. The numerical solution of the equations themselves is based on Godunov's method [5], and we use the HLLC Riemann solver first presented by Toro [6] in conjunction with the MUSCL-Hancock scheme [7] to construct a solver that is second order accurate in both space and time. The article is structured as follows. In Section 2, we provide a brief introduction to the compressible Euler equations and the numerical methods used to solve them, focusing on those implemented in the present work. In Section 3, we describe adaptive mesh refinement and give an overview of different available implementations, focusing more specifically on the approach used by the AMReX software. In Section 4, we explore the convergence properties of our numerical solver and show that a similar convergence pattern can be obtained with the appropriate use of AMR using lower base resolutions; we also show the results of a timing exercise demonstrating how AMR and parallelization can significantly speed up a simulation. Finally, in Section 5 we summarize our findings and discuss possible further work.

2. Euler Equations and Numerical Methods of Solution

2.1. Compressible Euler equations

The compressible Euler equations are a system of nonlinear hyperbolic conservation laws, a consequence of the laws of conservation of mass, momentum, and energy when applied to a fluid with negligible viscous stresses and thermal conductivity in the absence of external body forces. A full derivation of the equations can be found in standard textbooks on fluid mechanics such as [8, 2, 9]. In conservation form and independently of the geometry of the coordinate system, these equations read [9]

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0, \tag{1}$$

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v} + p\mathbf{I}) = 0, \tag{2}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot [(E + p)\,\mathbf{v}] = 0, \tag{3}$$

where ρ is the fluid's density, $\mathbf{v} = \left(v_x, v_y, v_z\right)^T$ is the fluid's velocity, p is the fluid's pressure, E is the fluid's total energy, and $\mathbf{I}$ is the identity matrix. The total energy can be expressed as a sum of the internal and kinetic energies

$$E = \rho\varepsilon + \frac{1}{2}\rho v^2. \tag{4}$$

The system of equations is closed with an additional equation relating pressure and internal energy, the equation of state. The equation of state is by no means unique and must be chosen taking into account the thermodynamical properties of the fluid whose behavior is being modeled; in this work we restrict ourselves to validation tests in which the fluid obeys the ideal gas equation of state, which is given by

$$p = \rho\varepsilon\,(\gamma - 1), \tag{5}$$

where γ is the adiabatic constant. Our tests in Section 4 use a value of $\gamma = 1.4$, which is the typical value used to model atmospheric air at temperatures between 10 K and 400 K and can be derived by considering the rotational degrees of freedom for molecular oxygen and nitrogen [2].

The simulations in this work are restricted to one and two dimensions, which leads to a simplification of the evolution equations because the z-derivative (and in the 1D case, the y-derivative) vanish. For numerical purposes, it is useful to write the system given by equations 1-4 as a single vector equation in conservation form. For the two dimensional case, and in Cartesian coordinates, the general conservation form is

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{u})}{\partial x} + \frac{\partial \mathbf{g}(\mathbf{u})}{\partial y} = 0, \tag{6}$$

The vector of conserved variables has four components, $\mathbf{u} = \left(\rho, \rho v_x, \rho v_y, E\right)^T$. The corresponding flux vectors are given by

$$\mathbf{f}(\mathbf{u}) = \begin{bmatrix} \rho v_x \\ \rho v_x^2 + p \\ \rho v_x v_y \\ (E + p)\,v_x \end{bmatrix}, \tag{7}$$

$$\mathbf{g}(\mathbf{u}) = \begin{bmatrix} \rho v_y \\ \rho v_x v_y \\ \rho v_y^2 + p \\ (E + p)\,v_y \end{bmatrix}. \tag{8}$$

2.2. Eigenstructure of the equations

One way of verifying that the conservative system 6 is hyperbolic is by writing it in terms of the primitive variable vector $\mathbf{w} = \left(\rho, v_x, v_y, p\right)^T$ and in quasilinear form,

$$\frac{\partial \mathbf{w}}{\partial t} + \mathbf{A}(\mathbf{w})\frac{\partial \mathbf{w}}{\partial x} + \mathbf{B}(\mathbf{w})\frac{\partial \mathbf{w}}{\partial y} = 0, \tag{9}$$

after which we verify that the matrices $\mathbf{A}$ and $\mathbf{B}$ are diagonizable with real eigenvalues. Starting from equation 6, we can use the relationships in equations 4-5 to solve for the variables in $\mathbf{w}$, and we find that the matrices are given by

$$\mathbf{A}(\mathbf{w}) = \begin{bmatrix} v_x & \rho & 0 & 0 \\ 0 & v_x & 0 & 1/\rho \\ 0 & 0 & v_x & 0 \\ 0 & \rho c_s^2 & 0 & v_x \end{bmatrix}, \tag{10}$$

$$\mathbf{B}(\mathbf{w}) = \begin{bmatrix} v_y & \rho & 0 & 0 \\ 0 & v_y & 0 & 0 \\ 0 & 0 & v_y & 1/\rho \\ 0 & 0 & \rho c_s^2 & v_y \end{bmatrix}, \tag{11}$$

where $c_s = \sqrt{\frac{\gamma p}{\rho}}$ is the sound speed. Both matrices have four eigenvalues; for matrix $\mathbf{A}$, we have

$$\lambda_1 = v_x - c_s,\ \lambda_2 = \lambda_3 = v_x,\ \lambda_4 = v_x + c_s. \tag{12}$$

Matrix $\mathbf{B}'$s eigenvalues are identical, except v_x is replaced with v_y. Since $\rho > 0$ and $p > 0$, c_s is real, and all four eigenvalues for both matrices are real; this means that the system 6 is hyperbolic. However, since $\lambda_2 = \lambda_3$, the system is not strictly hyperbolic. Each eigenvalue is

associated with a wave that moves at a speed equal to the eigenvalue. For the Euler equations in more than one dimension, we can distinguish four types of waves: rarefactions, shocks, contact discontinuities, and shear waves. Rarefactions and shocks are associated with λ_1 and λ_4, whereas contact discontinuities and shear waves are associated with λ_2 and λ_3 and travel at the same speed.

2.3. Numerical Approaches to Solving the Euler Equations

There are many different numerical frameworks for solving partial differential equations. The most common and widely used are finite difference methods, finite element methods, and finite volume methods [10][1]. These methods differ in the way they discretize the underlying PDE. In the finite difference case, data is defined at points, whereas in the finite element framework data is defined at nodes and in the finite volume approach data is defined within volumes, with function values representing an integral average over a volume centered around a particular point. The algorithms for the simulations in the present work are based on a finite volume approach, which is particularly widely used in dealing with hyperbolic PDEs in conservation form.

Within the finite volume framework, and specifically in the context of the numerical solution of the compressible Euler equations, there are two broad categories of numerical schemes: centred schemes and Riemann problem-based schemes. In one dimension and in Cartesian coordinates, both approaches discretize the computational domain into a finite number of computational cells of width Δx, and the conserved variable vector defined at the center of cell i at time t, $\mathbf{u}(x,t) = \mathbf{u}_i^n$ is advanced to $\mathbf{u}(x, t+\Delta t) = \mathbf{u}_i^{n+1}$ via the formula

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{f}^n_{i+\frac{1}{2}} - \mathbf{f}^n_{i-\frac{1}{2}}\right). \tag{13}$$

Where the centred and Riemann problem based schemes differ is in determining the intercell fluxes at the right (+) and left (-) cell boundaries, $\mathbf{f}^n_{i\pm\frac{1}{2}}$. Broadly speaking, centred schemes derive the flux via a suitable discretization of the derivatives in the original PDE, whereas Riemann problem based methods calculate the flux by solving a Riemann problem at intercell boundaries. Although centred schemes tend to be somewhat easier to implement, the inherent non-linearity of Riemann problem based methods tend to allow them to better capture sharp discontinuities. In this work, we solve the Euler equations using a Riemann problem based approach.

[1] Other methods such as the lattice Boltzmann approach [11] are also popular in the context of fluid dynamics.

2.4. The Riemann Problem

A Riemann problem is an initial value problem consisting of one or more equations in conservation form with initial conditions given by two constant states separated by a single discontinuity. Without loss of generality, we can consider the one dimensional Riemann problem for the Euler equations along a slice in the x-direction; for multidimensional problems we can incorporate the additional dimensions by solving additional Riemann problems along those directions, as we explain below. Along the x-direction, equation 6 becomes

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{u})}{\partial x} = 0 \tag{14}$$

with $\mathbf{f}(\mathbf{u})$ given by equation 7. Then the Riemann problem consists in finding a solution to the above equation for $t > 0$, $x \in [-\infty, \infty]$ when the initial conditions at $t = 0$ are given by

$$\mathbf{u}(x,0) = \mathbf{u}^0(x) = \begin{cases} \mathbf{u}_L, & x < 0 \\ \mathbf{u}_R, & x > 0 \end{cases}, \tag{15}$$

with $\mathbf{u}_L$, $\mathbf{u}_R$ constants and $\mathbf{u}_L \neq \mathbf{u}_R$.

2.5. Godunov's Scheme

The first and simplest Riemann problem based scheme was proposed more than 60 years ago by Godunov [5]. In the one-dimensional, Cartesian version of the scheme, the intercell fluxes $\mathbf{f}^n_{i\pm\frac{1}{2}}$ in equation 13 are given by

$$\mathbf{f}_{i\pm\frac{1}{2}} = \mathbf{f}\left(\mathbf{u}^n_{i\pm\frac{1}{2}}\right), \tag{16}$$

with $\mathbf{u}^n_{i\pm\frac{1}{2}}$ the solution along the ray $x/t = 0$ of the Riemann problem with initial conditions $\mathbf{u}_i^n$ and $\mathbf{u}^n_{i\pm1}$ centered at the appropriate cell interface, and $\mathbf{f}$ given by equation 7.

In simple terms, the basic Godunov scheme finds a solution to the hyperbolic PDE by solving a Riemann problem at each cell interface in the domain in order to compute intercell fluxes, and assumes that the value of the conserved variables on each cell is piecewise constant inside the cell. The timestep of the simulation is restricted by the Courant-Friedrichs-Lewy condition,

$$\Delta t = \frac{c_{cfl}\Delta x}{a^n_{max}}, c_{cfl} \leq 1, \tag{17}$$

where a^n_{max} is the maximum wave speed in the domain at time t^n and c_{cfl} is the dimensionless CFL constant. In

the case of the one dimensional Euler equations, a^n_{max} is given by

$$a^n_{max} = \max_i \left(\left| v_{x,i} \right| + c_{s,i} \right). \tag{18}$$

In practice, simulations usually have $c_{cfl} < 1$ in order to ensure that the scheme is stable even if the expression for a^n_{max} is not guaranteed to be completely accurate. Our solver uses a value of $c_{cfl} = 0.8$.

This scheme can be easily extended to two dimensions by solving an additional Riemann problem, now in the y-direction. This can be done in a dimensionally unsplit or a dimensionally split manner; our solver uses the dimensionally split approach because it does not lead to further restrictions on c_{cfl} [12, 9]. We first perform an update exclusively in the x-direction using a two dimensional version of equation 13, obtaining the fluxes by solving the Riemann problems to at the left and right interfaces of each cell. For each cell, this results in an intermediate state $\bar{\mathbf{u}}^n_{i,j}$,

$$\bar{\mathbf{u}}^n_{i,j} = \mathbf{u}^n_{i,j} - \frac{\Delta t}{\Delta x} \left(\mathbf{f}^n_{i+\frac{1}{2},j} - \mathbf{f}^n_{i-\frac{1}{2},j} \right), \tag{19}$$

where $\mathbf{f}^n_{i\pm\frac{1}{2},j}$ is defined in an analogous way as equation 16. The intermediate state is now updated in the y-direction by solving the Riemann problems centered at the top and bottom interfaces of each cell to obtain the states $\bar{\mathbf{u}}^n_{i,j\pm\frac{1}{2}}$ and calculating fluxes via $\mathbf{g}\left(\bar{\mathbf{u}}^n_{i,j\pm\frac{1}{2}}\right)$, where $\mathbf{g}$ is given by equation 8. This gives us the advanced state $\mathbf{u}^{n+1}_{i,j}$,

$$\mathbf{u}^{n+1}_{i,j} = \bar{\mathbf{u}}^n_{i,j} - \frac{\Delta t}{\Delta y} \left[\mathbf{g}\left(\bar{\mathbf{u}}^n_{i,j+\frac{1}{2}}\right) - \mathbf{g}\left(\bar{\mathbf{u}}^n_{i,j-\frac{1}{2}}\right) \right]. \tag{20}$$

The CFL condition is identical to that given in equation 17 except Δx is replaced by $\min(\Delta x, \Delta y)$ and $v_x \rightarrow |\mathbf{v}| = \sqrt{v_x^2 + v_y^2}$ in the expression for a^n_{max} 18, which now also incorporates the additional index j. Our solver uses $\Delta x = \Delta y$ and deals with rectangular domains by using a different number of cells along the x-direction versus the y-direction.The assumption that states $\mathbf{u}^n_i$ are piecewise constant within cells restricts the basic Godunov method to first order accuracy in space, but several extensions exist that result in higher order schemes that still follow the same basic approach. In this work, we achieve second order accuracy in space and time by following the MUSCL-Hancock approach [13], which works by linearly reconstructing the states within the cells. A more detailed description of this scheme can be found in [9].

2.6. The HLLC Solver

The Riemann problem for the ideal gas compressible Euler equations can be solved exactly after numerically obtaining a value for the intermediate state pressure using an iterative root finding procedure; this value can be obtained to arbitrary precision, which is why this type of solution is called an exact solution despite the fact that it cannot be written in closed form. In the early days of computational fluid dynamics, the iterative nature of this scheme meant that it was too computationally expensive to use it for practical purposes, and this is true even today in some contexts like very high resolution simulations, simulations performed in real time, and solutions of more complex systems of equations like those of magnetohydrodynamics or the Euler equations with more complicated equations of state [10]. Therefore, much work has been done in developing approximate numerical Riemann solvers; in this work we implement one such solver, the HLLC solver [6].

Unlike the earlier HLL solver[14], which employs a single intermediate state and can be overly diffusive, the HLLC solver allows for two intermediate states separated by a contact discontinuity, decreasing the level of diffusion and enabling it to better capture isolated contact discontinuities. The scheme exploits the continuity of velocity and pressure across a contact discontinuity, the continuity of tangential velocities across shocks, and the Rankine-Hugionot conditions in order to obtain algebraic expressions for the components of the intermediate star states $\mathbf{u}^*_R$ and $\mathbf{u}^*_L$. This, coupled with appropriate estimates for the speeds of the left and right moving discontinuities, S_L and S_R, can be used to compute a flux $\mathbf{f}^{HLLC}$ across the interface separating the two initial states $\mathbf{u}_R$ and $\mathbf{u}_L$ of the form

$$\mathbf{f}^{HLLC} = \begin{cases} \mathbf{f}_L, & 0 \leq S_L \\ \mathbf{f}^*_L & S_L < 0 \leq v^*_x \\ \mathbf{f}^*_R & v^*_x < 0 \leq S_R \\ \mathbf{f}_R & S_R < 0 \end{cases}, \tag{21}$$

assuming that we are solving the Riemann problem along the x-direction, and where $\mathbf{f}_K = \mathbf{f}(\mathbf{u}_K)$, $K = L, R$. As derived in detail by Toro [6, 9], expressions for the intermediate fluxes $\mathbf{f}^*_K$ and the intermediate state velocity v^*_x follow from the Rankine-Hugionot conditions and the continuity of pressure across the contact discontinuity,

$$\mathbf{f}^*_K = \mathbf{f}_K + S_K \left(\mathbf{u}^*_K - \mathbf{u}_K \right), \tag{22}$$

$$v^*_x = \frac{p_L - p_R + \rho_R v_{x,R} \left(S_R - v_{x,R}\right) - \rho_L v_{x,L} \left(S_L - v_{x,L}\right)}{\rho_R \left(S_R - v_{x,R}\right) - \rho_L \left(S_L - v_{x,L}\right)}, \tag{23}$$

with the intermediate states $\mathbf{u}_K^*$ given by

$$\mathbf{u}_K^* = \rho_K \left(\frac{S_K - v_{x,K}}{S_K - v_x^*} \right) \begin{bmatrix} 1 \\ v_x^* \\ v_{y,K} \\ \frac{E_K}{\rho_K} + (v_x^* - v_{x,K}) \left[v_x^* + \frac{p_K}{\rho_K (S_K - v_{x,K})} \right] \end{bmatrix}, \quad (24)$$

also a consequence of the Rankine-Hugionot conditions and the beahvior of variables across waves.

The final piece in the HLLC puzzle is finding an appropriate estimate for S_L and S_R. As detailed in [9], there are multiple options available, and this is an area of active research [15]. Our solver uses a modification of one set of estimates suggested by Davis [16],

$$\begin{aligned} S_R &= \max \left(\left| v_{x,L} \right| + c_{s,L}, \left| v_{x,R} \right| + c_{s,R} \right), \\ S_L &= -S_R, \end{aligned} \quad (25)$$

which we find to be robust, as it works properly for all our test cases.

The same scheme above can be used for solving the Riemann problem posed by cells along the y-direction, except now $v_x \rightarrow v_y$, $v_x^* \rightarrow v_y^*$ in all the expressions given, the second and third components in $\mathbf{u}_K^*$ are swapped, and $\mathbf{f}_K$ is now $\mathbf{g}_K$, ie. the flux is calculated according to equation 8 instead of equation 7.

3. Adaptive Mesh Refinement

3.1. Mesh Refinement

It is often the case in numerical simulations that some of the structures and patterns produced by the underlying equations need to be sampled more frequently in space compared to other patterns in order to accurately model their evolution. For a regular Cartesian grid with fixed cell sizes, this means that in practice the resolution of the grid will be determined by those features of the solution that need to be sampled more frequently in order to to be captured correctly.

A common approach to tackling this issue is known as mesh refinement; numerical techniques that fall under this broad category introduce modifications to regular structured grids or get rid of them altogether in order to achieve different levels of resolution at different parts of the computational domain. Examples of the mesh refinement approach include unigrids, mesh stretching, curvilinear grids, unstructured meshes, quad-trees, and hierarchical Adaptive Mesh Refinement (AMR)[17]. Our algorithm uses AMR, which we describe in more detail.

3.2. Adaptive Mesh Refinement and AMReX

Adaptive mesh refinement, introduced by Berger and Oliger [3] and extended by Berger and Colella [4], is a method to dynamically alter the resolution of portions of the computational domain in a simulation as a function of the evolution of the simulation itself. An important aspect of AMR is that it operates in a Cartesian framework and uses logically rectangular grids. Starting with a coarse base grid, the algorithm employs a dynamic tagging procedure in order to flag those regions of the domain that need to be resolved in more detail, according to some criteria supplied by the user. It then proceeds to construct a finer grid that encompasses the flagged regions, at a resolution that is a multiple of the base grid's resolution. The extent of these higher resolution grids is in practice determined by an iterative algorithm [18] and two user provided inputs: the blocking factor, which sets the minimum size of the meshes, and a parameter that sets the maximum grid size.

The framework allows for recursively repeating the above procedure to reach arbitrarily high resolutions by adding additional levels of refinement, and grants the user a great degree of flexibility to tailor the methodology depending on the problem at hand. One of the user's most important responsibilities is determining the criteria for refininement. For example, if the numerical structures of interest are known to be located within a boundary, the user can implement a signed distance function that evolves together with the simulation, effectively tracking the region of the domain that needs to be refined at all times.

A simpler approach which is particularly well suited for problems in fluid dynamics is for the tagging procedure to be based on the change of one of the quantities in $\mathbf{u}$ or $\mathbf{w}$; as we have seen in previous sections, the compressible Euler equations tend to generate sharp discontinuities in these quantities, and it is at these discontinuities that higher resolutions are often needed. A common choice of refinement variable is the density ρ, because it jumps across both shocks and contact discontinuities. Thus for the tests in this work, our tagging procedure is density-based: our algorithm tags cells in the domain according to the formula

$$\frac{1}{2} \max \left(\left| \rho_{i+1,j} - \rho_{i-1,j} \right|, \left| \rho_{i,j+1} - \rho_{i,j-1} \right| \right) \geq \Delta\rho_{th} \quad (26)$$

where $\Delta\rho_{th}$ is a test-dependent minimum change in ρ across two neighboring cells.

In addition to tagging and generating higher resolution grids, AMR requires a series of additional steps in order to ensure that the results of the simulation are accurate and consistent with those obtained without mesh

refinement but at higher whole grid resolutions. These steps include the creation and handling of a set of ghost cells surrounding the finer resolution meshes in order to enforce internal boundary conditions, application of flux corrections at the boundaries between coarse and fine grids to maintain conservation of the numerical flux, and interpolation and averaging across levels in order to output results for specific numbers of cells. Furthermore, given that AMR is often used in a context in which the run time of the simulation is of paramount importance, an important aspect of any implementation is its ability to run simulations in parallel using multiple processors and also leverage the power of graphical processing units (GPUs).

Numerous software packages developed by a variety of large research groups have been developed, each tackling these issues in different ways with different application focuses in mind. Examples include BoxLib [19], Carpet [20], Chombo [21], Enzo [22], and FLASH [23], among many others; see [24] for a survey and a more detailed description of the approaches. Our work uses the AMReX [1] software package, a successor to BoxLib developed by the US Department of Energy. The library is written in C++ and Fortran and provides support for not just finite volume-based codes but also finite element data structures as well as an adaptive grid structure suitable for hyperbolic, parabolic, and elliptic problems. Additionally, the framework supports different parallelization paradigms including MPI, OpenMP, and hybrid MPI/OpenMP, in addition to support for GPU computing.

3.3. Convergence Analysis

Consider a simulation that seeks to find the value of a variable $u(x,t)$ as a function of space and time by numerically solving an underlying evolution equation or system of equations. If the simulation is implemented using a computational grid consisting of N cells separated by Δx, then the simulation has order of accuracy p if there is a number C independent of Δx such that [25]

$$|u_{\Delta x} - u_{ex}| \leq C(\Delta x)^p, \tag{27}$$

for sufficiently small Δx, where $u_{\Delta x}$ is the numerical approximation to u at an arbitrary cell and simulation time, u_{ex} is the exact value of u at that point, and where $(\Delta x)^p$ is known as the convergence rate of the numerical method. The error $|u_{\Delta x} - u_{ex}|$ is in many cases a smooth function of Δx, and there additionally exists an error coefficient D such that

$$|u_{\Delta x} - u_{ex}| = D(\Delta x)^p + O\left[(\Delta x)^{p+1}\right]. \tag{28}$$

If we know the exact solution u_{ex}, we can determine the empirical order of accuracy p by running the simulation at a range of Δx and fitting a linear function of $\log(\Delta x)$, since

$$\begin{aligned}\log|u_{\Delta x} - u_{ex}| &= \log|D(\Delta x)^p[1 + O(\Delta x)]| \\ &= \log|D| + p\log(\Delta x) + O(\Delta x).\end{aligned} \tag{29}$$

Another way to obtain p is to halve Δx and evaluate the ratio of the errors $|u_{\Delta x} - u_{ex}|$ and $\left|u_{\Delta x/2} - u_{ex}\right|$,

$$\begin{aligned}\frac{|u_{\Delta x} - u_{ex}|}{\left|u_{\Delta x/2} - u_{ex}\right|} &= \frac{D(\Delta x)^p + O\left[(\Delta x)^{p+1}\right]}{D\,(\Delta x/2) + O\left[(\Delta x/2)^{p+1}\right]} \\ &= 2^p + O(\Delta x).\end{aligned} \tag{30}$$

which means that

$$\log_2\left(\frac{|u_{\Delta x} - u_{ex}|}{\left|u_{\Delta x/2} - u_{ex}\right|}\right) = p + O(\Delta x). \tag{31}$$

The above analysis applies to a single pair of values of u_{num} and u_{ex}; extending this to multiple values evaluated at different points within a grid is commonly done by replacing $u_{\Delta x} - u_{ex}$ with a global error given by a suitable discrete norm. In this work we use the L_1-norm or L_1-error, defined as the sum of the absolute error over the grid multiplied by the grid spacing [26],

$$L_1\text{-error} = \sum_{i=0}^{N-1} \Delta x_i \left|u_{\Delta x,i} - u_{ex,i}\right|, \tag{32}$$

where $u_{\Delta x,i}$ is the numerical value for variable u at cell i, $u_{ex,i}$ is the exact value of variable $u$at cell i, and $\Delta x_i = x_{i+1} - x_i$ for a given simulation time t. For an evenly spaced grid with $x \in [0,1]$, $\Delta x_i = \Delta x$ is a constant, however, for simulations using AMR the grid points are in general not evenly spaced. Consistently with the previous section, we perform our convergence analysis using density, $u = \rho$, and we use an exact Riemann solver to obtain the values for ρ_{ex}.

4. Validation Tests

4.1. 1D Tests

4.1.1. Unrefined Tests

We validate our solver using the one dimensional Riemann tests with initial conditions shown in Table 1. Initially, we run all tests in one dimension at three different resolutions without mesh refinement to ensure that the software is running correctly, that the solution's accuracy increases as the resolution is increased, and to get

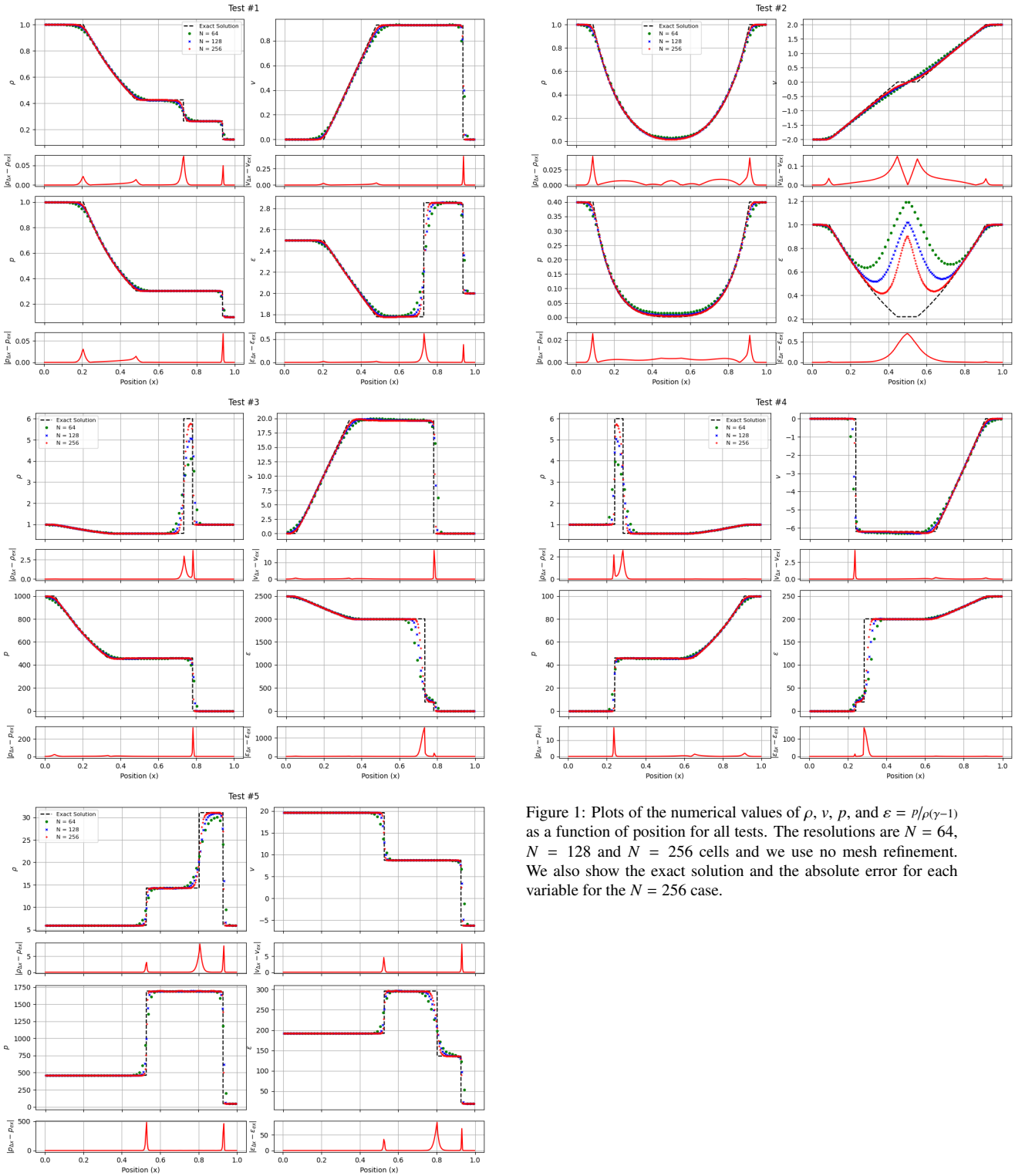


Figure 1: Plots of the numerical values of ρ, v, p, and $\varepsilon = p/\rho(\gamma-1)$ as a function of position for all tests. The resolutions are $N = 64$, $N = 128$ and $N = 256$ cells and we use no mesh refinement. We also show the exact solution and the absolute error for each variable for the $N = 256$ case.

Test	ρ_L	v_L	p_L	ρ_R	v_R	p_R	t_f
1	1.0	0.0	1.0	0.125	0.0	0.1	0.25
2	1.0	-2.0	0.4	1.0	2.0	0.4	0.15
3	1.0	0.0	1000.0	1.0	0.0	0.01	0.012
4	1.0	0.0	0.01	1.0	0.0	100.0	0.035
5	5.99924	19.5975	460.894	5.99242	-6.19633	46.0950	0.035

Table 1: Initial data for one-dimensional Riemann validation tests, with t_f being the final simulation time. These tests are standard validation tests for the ideal gas compressible Euler equations in one dimension, and were compiled by Toro [9]. All tests use $\gamma = 1.4$ and transmissive boundary conditions.

an idea of where the numerical errors are concentrated. Figure 1 shows plots for ρ, v, p, and internal energy $\varepsilon = {}^{p}/_{\rho(\gamma-1)}$ for $N = 64$, $N = 128$, and $N = 256$ cells, on top of the exact solution and together with the absolute error for each variable for the $N = 256$ case, all as a function of position. From the figure, it is apparent the largest errors occur at discontinuities, which justifies our choiceof equation 26 as our tagging criterion, although we do note that Test 2 also shows a considerable amount of error away from sharp discontinuities, due to the presence of an extended low density region.

4.1.2. Refined vs Unrefined Tests

We now demonstrate the effects of mesh refinement by comparing the outputs of refined and unrefined simulations with a base grid resolution of $N_{base} = 64$. We use a refinement ratio $r = 2$, which means that the refined grids have double the resolution of the coarse grid, or half the Δx. We define the percentage coverage ratio α as the percentage of the coarse grid that is covered by the finer grids produced by the AMR procedure, ie. $\alpha = {}^{N_{ref}}/_{N_{base}} - 1$, where N_{ref} is the total number of cells in the simulation with AMR and N_{base} is the total number of cells in the coarse grid, and fine tune our refinement threshold such that at the last simulation timestep, $\alpha = 18.75\%$, which is a reasonable figure [17]. Given the low base resolution, we use a blocking factor of 4 and a maximum grid size of 8, and employ a single refinement level, which is enough for illustrative purposes. We use refinement threshold values (equation 26) of $\Delta\rho_{th,1} = 0.038$, $\Delta\rho_{th,2} = 0.075$, $\Delta\rho_{th,3} = 0.25$, $\Delta\rho_{th,4} = 0.08$, and $\Delta\rho_{th,5} = 3.50$, where the subscript indicates the test number.

Figure 2 shows the results of the simulations. It can be readily seen that the addition of a small number of cells by AMR can dramatically increase the accuracy of the solution. Because of the nature of our refinement criterion, AMR adds grid cells at regions close to discontinuities, where the error is more significant; it is apparent that the refined grid is denser at these points, allowing the numerical solution to be substantially closer to the exact solution.

4.1.3. Convergence

The goal of AMR is to be able to obtain numerical results that are comparable to those of high resolution simulations at a fraction of the computational cost. Ideally, a properly fine-tuned AMR simulation with an effective resolution $N_{eff} = rN_{base}$ will show the same error and convergence properties of a simulation without mesh refinement for which N_{base} is equal to the N_{eff} of the simulation with mesh refinement. In other words, the convergence plots for the refined and unrefined simulations should overlap.

As before, we set $\Delta\rho_{th}$ such that $\alpha = 18.75\%$ for the lowest resolution. This value decreases as N_{base} increases, because the base Δx decreases and as a result $\Delta\rho$ also decreases, which means that fewer cells are tagged for refinement. We use a blocking factor of 4 and a maximum grid size of 8 in the $N_{base} = 64$ simulation, and a blocking factor of 8 and a maximum grid size of 16 in all others.

The results of the convergence analysis are presented in 3. It can be seen that for Tests 3 and 4, the convergence plots for the refined and unrefined cases are practically identical, which means that our mesh refinement setup is very effective. For Test 2, the convergence rates for both curves are very similar, but the refined simulations have slightly more error than the corresponding unrefined ones. This test is different from the others in that there is a significant amount of error distributed over a wide region in which the density gradient is not large; this kind of test is not particularly well-suited for AMR. In the case of Tests 1 and 5, we find that the threshold $\Delta\rho_{th}$ that leads to $\alpha = 18.75\%$ in the lowest resolution case is not low enough to produce the same kind of convergence as the unrefined case, because not all the discontinuities are captured. However, if we set $\Delta\rho_{th,1} = 0.02$ and $\Delta\rho_{th,5} = 2.0$, we obtain much better convergence and errors comparable to those of the unrefined case, at the expense of α increasing to 50.00% and 28.13%, respectively for each test at the lowest resolution.

We note that even though the MUSCL-Hancock scheme is formally a second order accurate scheme, the convergence analysis for these tests gives a range of L_1 orders between 0.60 and 0.83 in the unrefined case, which is much lower than 2. The reason is that orders of convergence for schemes are in general calculated using smooth solutions [27], not solutions containing discontinuities like the tests analysed in the present work. In

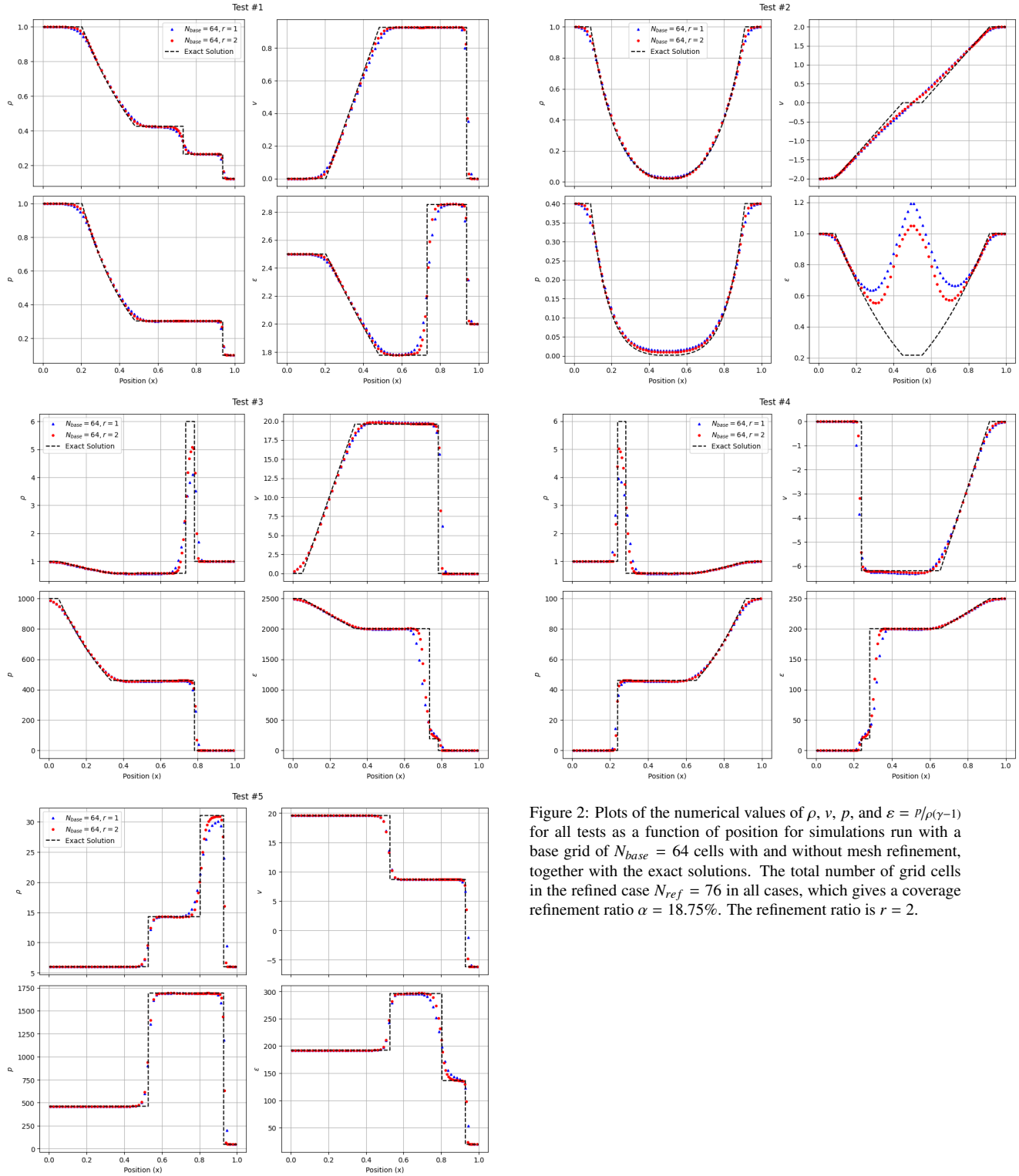
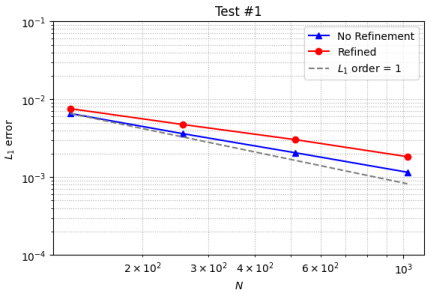


Figure 2: Plots of the numerical values of ρ, v, p, and $\varepsilon = p/\rho(\gamma-1)$ for all tests as a function of position for simulations run with a base grid of $N_{base} = 64$ cells with and without mesh refinement, together with the exact solutions. The total number of grid cells in the refined case $N_{ref} = 76$ in all cases, which gives a coverage refinement ratio $\alpha = 18.75\%$. The refinement ratio is $r = 2$.



No Refinement

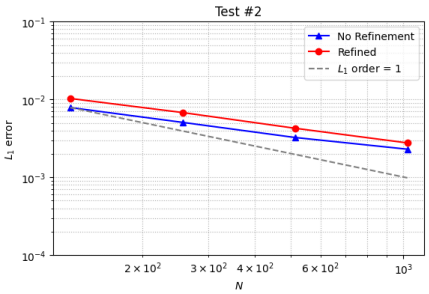
N_{base}	L_1 error	L_1 order
128	0.00657	
256	0.00362	0.86057
512	0.00206	0.81345
1024	0.00115	0.83362

Best fit: 0.834

Refined

N_{base}	N_{eff}	N_{act}	L_1 error	L_1 order
64	128	104	0.00757	-
128	256	168	0.00474	0.67513
256	512	272	0.00303	0.64525
512	1024	532	0.00183	0.72844

Best fit: 0.679



No Refinement

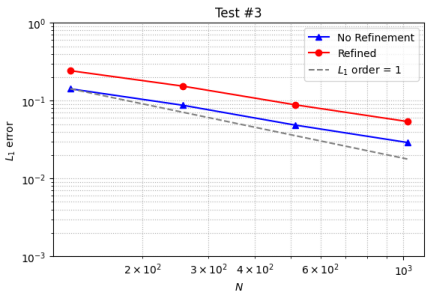
N_{base}	L_1 error	L_1 order
128	0.00787	-
256	0.00508	0.63270
512	0.00325	0.64546
1024	0.00230	0.49899

Best fit: 0.598

Refined

N_{base}	N_{eff}	N_{act}	L_1 error	L_1 order
64	128	124	0.01030	-
128	256	240	0.00679	0.60206
256	512	344	0.00427	0.66970
512	1024	580	0.00276	0.62628

Best fit: 0.636



No Refinement

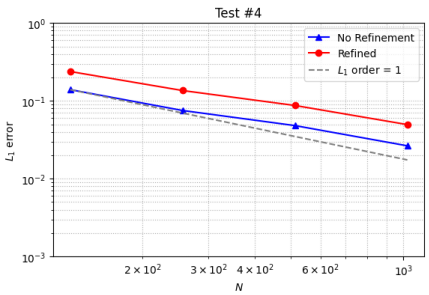
N_{base}	L_1 error	L_1 order
128	0.14221	-
256	0.08740	0.70228
512	0.04862	0.84617
1024	0.02902	0.74466

Best fit: 0.773

Refined

N_{base}	N_{eff}	N_{act}	L_1 error	L_1 order
64	128	74	0.24290	-
128	256	144	0.15396	0.65785
256	512	276	0.08860	0.79722
512	1024	532	0.05398	0.71488

Best fit: 0.731



No Refinement

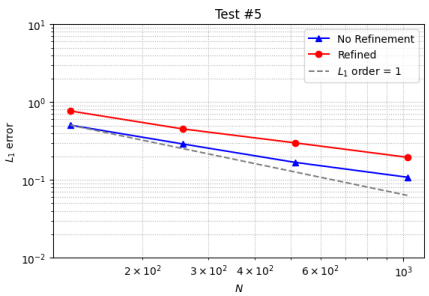
N_{base}	L_1 error	L_1 order
128	0.13942	-
256	0.07523	0.89009
512	0.04811	0.64488
1024	0.02645	0.86314

Best fit: 0.784

Refined

N_{base}	N_{eff}	N_{act}	L_1 error	L_1 order
64	128	74	0.23840	-
128	256	140	0.13576	0.81234
256	512	280	0.08726	0.63762
512	1024	532	0.04954	0.81667

Best fit: 0.744



No Refinement

N_{base}	L_1 error	L_1 order
128	0.50566	
256	0.28964	0.80390
512	0.16800	0.78579
1024	0.10820	0.63480

Best fit: 0.746

Refined

N_{base}	N_{eff}	N_{act}	L_1 error	L_1 order
64	128	84	0.76897	-
128	256	164	0.45201	0.76658
256	512	284	0.29927	0.59488
512	1024	540	0.19521	0.61641

Best fit: 0.653

Figure 3: Convergence analysis for all tests with and without mesh refinement. We plot the L_1-error (equation 32) as a function of the effective resolution $N_{eff} = rN_{base}$, where r is the refinement ratio; $r = 1$ for the simulation with no mesh refinement and $r = 2$ for the refined case. The L_1 order at each resolution is calculated using equation 31, with the best fit being the slope p of the best fit line defined by equation 29.

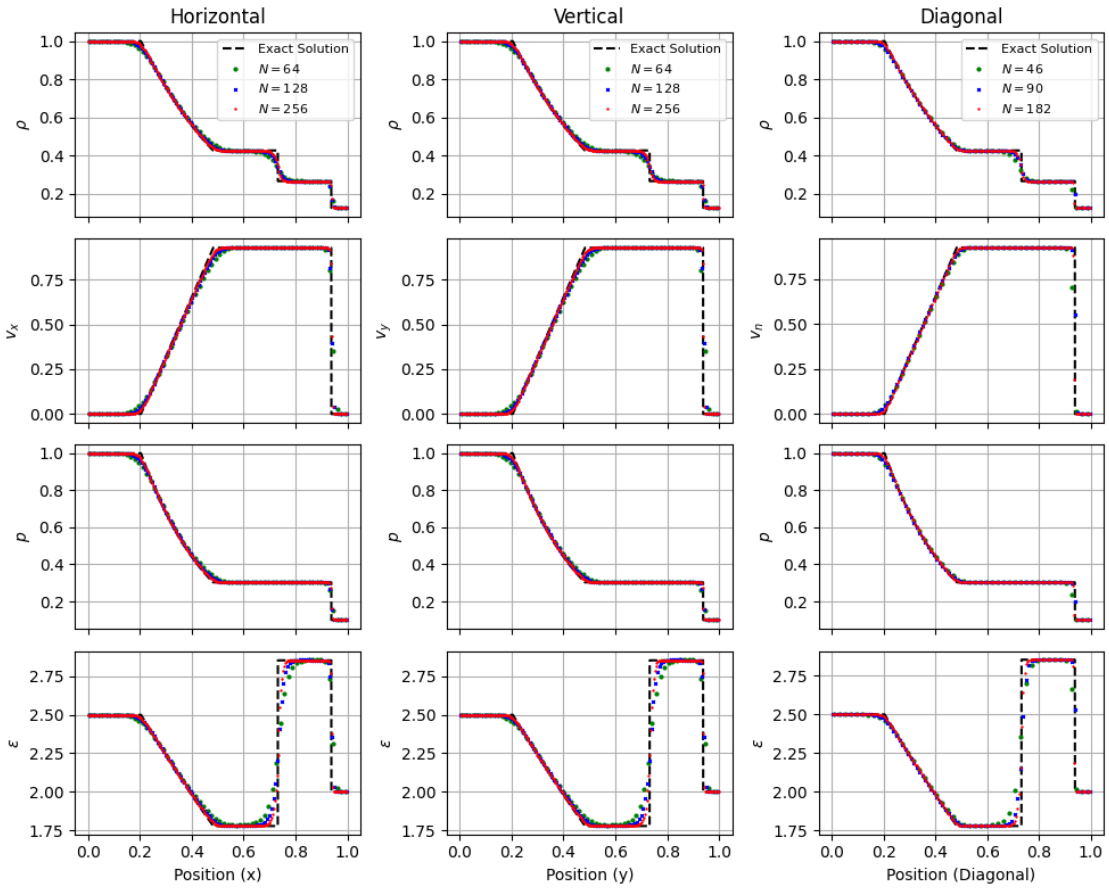


Figure 4: Results for the two dimensional version of Test 1 for ρ, p, normal velocity and $\varepsilon = {}^{p}/_{\rho(\gamma-1)}$ using three different resolutions for horizontal, vertical, and diagonal orientations, together with the exact solutions. The plots correspond to slices along the lines in the $x - y$ plane that are perpendicular to the line separating the initial states. Although the grid sizes for the simulation is the same for all orientations, the effective resolution in the diagonal case is lower.

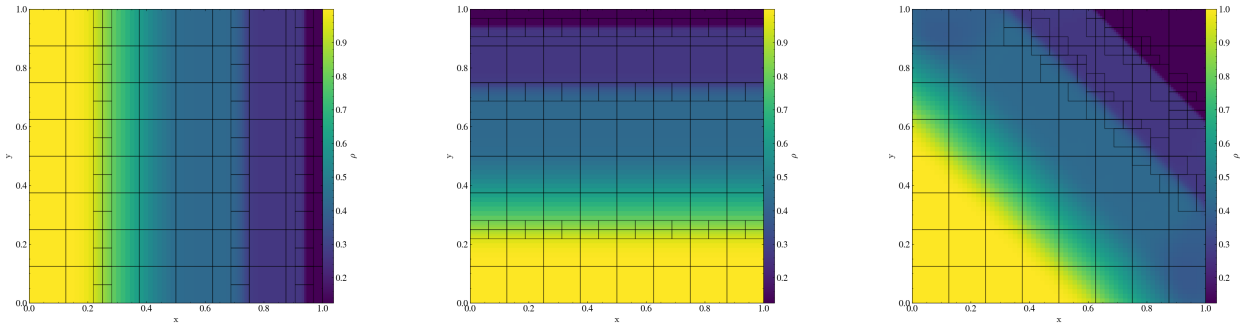


Figure 5: Colormaps of density ρ for the horizontal, vertical, and diagonal orientations of the two dimensional extension of Test 1, run with mesh refinement. The coarse grid resolution is 64×64 cells, and we use a refinement ratio of 2, a blocking factor of 4, and a maximum grid size of 8. The colormaps are overlaid with the subgrid boundaries.

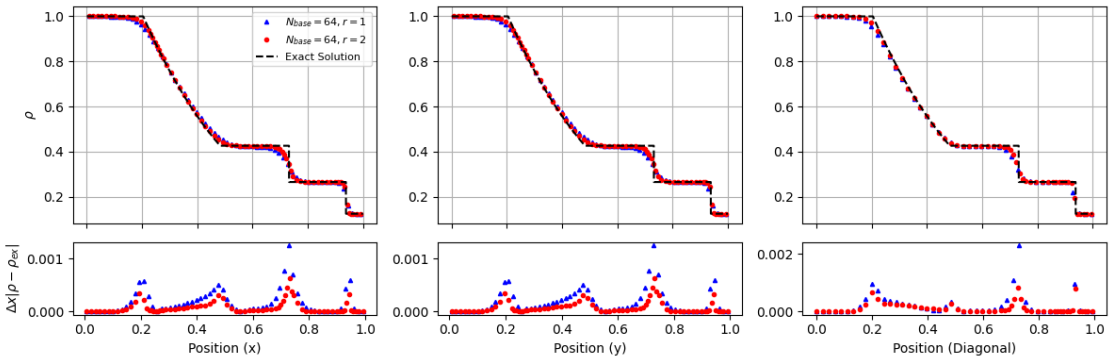


Figure 6: Plots of the numerical values of ρ and their scaled absolute error $\Delta x\,|\rho - \rho_{ex}|$ for Test 1 as a function of position for simulations run with a base grid of $N_{base} = 64 \times 64$ cells with ($r = 2$) and without ($r = 1$) mesh refinement, together with the exact solution, for horizontal, vertical, and diagonal orientations. The plots correspond to slices along the lines in the $x - y$ plane that are perpendicular to the line separating the initial states. The total number of grid cells in the direction of the slice for the refined case is $N_{ref} = 76$ in the horizontal and vertical cases and $N_{ref} = 56$ in the diagonal case, which gives a coverage refinement ratios of 18.75% and 21.74%, respectively.

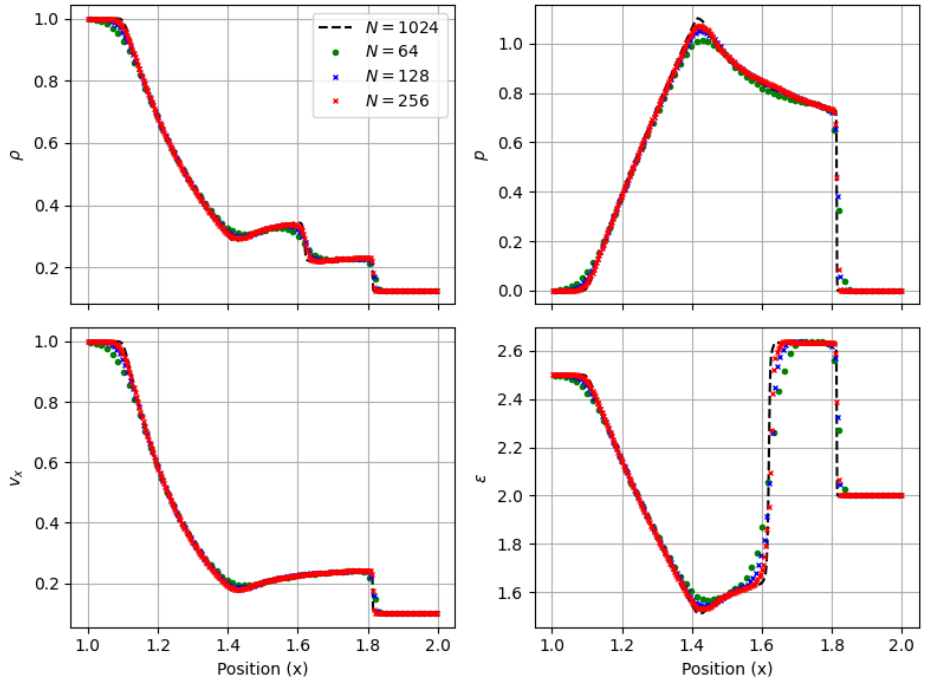


Figure 7: Plots of ρ, p, v_x, and ε for a slice along the line $y = 1$, with $x \in [1, 2]$, at four different resolutions for the cylinder explosion test. The number of cells N in these plots is half the number of cells along each axis in the full simulation since we only plot half of the domain given the symmetric nature of the problem.

the case of solutions with discontinuities, the errors near these discontinuities dominate the total error, and this error does not decrease at the same rate as the resolution is increased as does the error in the smooth parts of the solution.

4.2. 2D Extension

To test our two dimensional implementation of the solver, we initially ran the same tests as in the previous section, except we now use a two dimensional grid and vary their orientations. We illustrate this using Test 1. First, we ensure that we obtain the same results for horizontal, vertical, and diagonal orientations of the test by plotting slices along a line perpendicular to the line separating the states, and that these results are the same as those in 1. We can see in Figure 4 that this is in fact the case for Test 1. We note that in the diagonal case, we only plot the solution for the grid coordinates in which $(x - x_0)^2 + (y - y_0)^2 < r^2$, with $x_0 = y_0 = r = 0.5$; this allows us to compare the shape of the slice directly to that of the horizontal and diagonal cases as well as the exact solution, but it also implies that the resolution of the diagonal test is lower than in the other two cases, even though the grid size is the same.

Next, we plot colormaps of density for all three orientations using mesh refinement and a coarse grid resolution of 64×64 cells, as seen in Figure 5. We overlay the colormaps with the simulation subgrid boundaries; we can see in the figure the places in which AMR refines, which aligns with the discontinuities. We note that in the diagonal case, AMR does not refine the first discontinuity properly, this has to do with the fact that our tagging criterion calculates $\Delta\rho$ along the $x-$ and $y-$ directions, and the discontinuities in this case are not oriented normal to either axis. This could be solved by calculating $\Delta\rho$ along the diagonal direction as well, but in practice it is easier to simply rotate the coordinate system.

Finally, we plot comparisons between the refined and unrefined simulations for all three orientations (Figure 6), similarly to the 1D case shown in Figure 2. Our results match those of the 1D case except for the diagonal orientation because of the reduced effective resolution and the way our tagging criterion works, as explained above.

4.3. Timing Exercise

As a final test, we study a genuine two dimensional problem and illustrate how parallelization and AMR can lead to significant savings in CPU time for a given effective resolution. The test in question is the cylinder

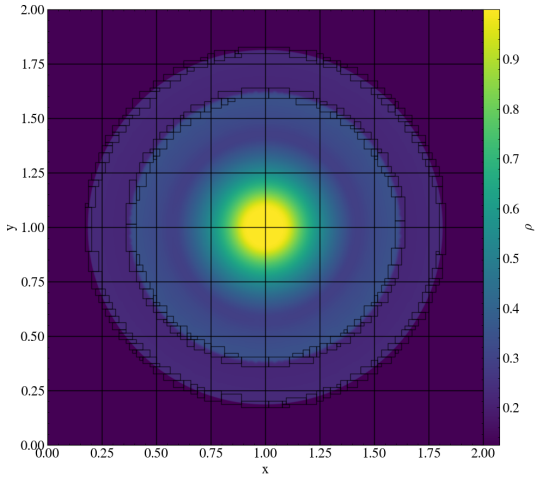


Figure 8: Colormap of ρ for the cylinder explosion test, run with mesh refinement. The coarse grid resolution is 512×512 cells, and we use a refinement ratio of 2, a blocking factor of 8, and a maximum subgrid size of 64×64. The colormap is overlaid with the subgrid boundaries.

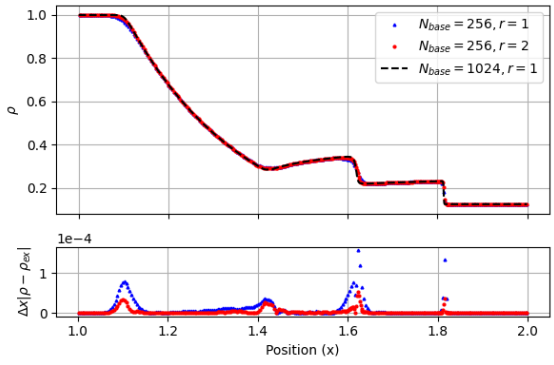


Figure 9: Plot of the numerical values of ρ and its scaled absolute error $\Delta x\,|\rho - \rho_{ex}|$ as a function of position along the line $y = 1$ for the right half of the domain. The total number of grid cells in the direction of the slice for the refined case is $N_{ref} = 276$, which gives a coverage refinement ratio of 7.81%.

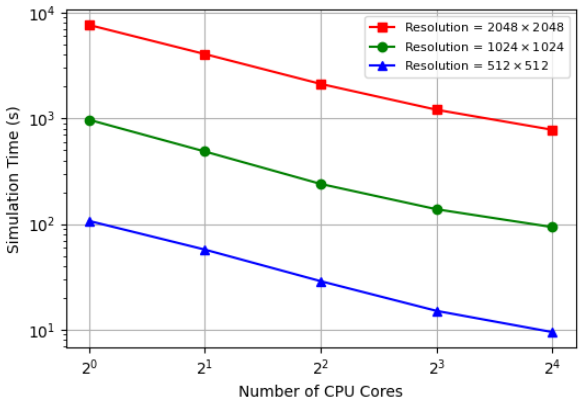


Figure 10: Time to simulation completion for 512×512 cells, 1024×1024 cells, and 2048×2048 cells resolutions as a function of CPU cores used.

Resolution	CPU Cores	Time (s)	Speedup
512×512	1	107.0	
512×512	2	57.3	1.87x
512×512	4	28.8	1.99x
512×512	8	15.1	1.91x
512×512	16	9.5	1.59x
1024×1024	1	968.0	
1024×1024	2	485.0	2.00x
1024×1024	4	239.3	2.03x
1024×1024	8	137.9	1.74x
1024×1024	16	93.5	1.47x
2048×2048	1	7,629.6	
2048×2048	2	4,044.8	1.89x
2048×2048	4	2,111.8	1.92x
2048×2048	8	1,202.9	1.76x
2048×2048	16	777.8	1.55x

Table 2: Time to simulation completion for 512×512 cells, 1024×1024 cells, and 2048×2048 cells as a function of CPU cores used, and the associated speedup.

Blocking Factor	Time (s)	Speedup	α
8	219.5		9.63%
16	252.9	0.87x	14.80%
32	322.6	0.78x	28.40%
64	422.6	0.76x	53.12%

Table 3: Time to simulation completion, associated speedup, and refinement coverage ratio α as a function of the blocking factor for simulations with base resolution of 512×512 cells and $r = 2$.

explosion test given in Section 17.1 of [9]; it is a two dimensional version of Test 1 from above with the initial states separated by a circle in the $x - y$ plane. In two dimensions, $\mathbf{w} = (\rho, v_x, v_y, p)^T$; the state inside the circle at $t = 0$ is $\mathbf{w}_{in} = (1, 0, 0, 1)^T$, and the one outside is $\mathbf{w}_{out} = (0.125, 0, 0, 0.1)^T$. The grid is square, with $x \in [0, 2]$, $y \in [0, 2]$, and the circle is centered at $x_0 = y_0 = 1$ and has radius $r = 0.4$. Boundary conditions are transmissive along both axes. The solution of this test exhibits a complex wave pattern, with a circular shock wave travelling outwards from the center, a circular contact surface moving in the same direction, and a rarefaction travelling inwards towards the origin.

We begin by verifying that our solution is correct and that increasing the resolution increases the accuracy of the solution. We do not have an exact solution for this particular problem, but we can verify that our results match those in [9] and that the discontinuities become sharper as the resolution is increased. Figure 7 shows plots of ρ, p, v_x, and ε for a slice along the line $y = 1$, with $x \in [1, 2]$, at four different resolutions. We observe that the results are consistent with those in [9], and that it is clear that the solution becomes more accurate as the resolution is increased. Next, we find an appropriate $\Delta\rho_{th}$ that is able to properly capture the shock and the contact discontinuitie in the test; we find that $\Delta\rho_{th} = 0.015$ works well for our base resolution of 512×512 cells, as shown in Figure 8. We also ensure that the error is reduced by mesh refinement, which is made apparent by Figure 9.

We now record time to completion for three resolutions starting with 512×512 cells, without refinement, as a function of the number of CPU cores used, as shown in Figure 10 and Table 2. We find that using twice as many cores reduces the simulation time by roughly a factor of 2, as expected, although the effect is more pronounced when less cores are used because of the reduced parallelization overhead incurred. Next, we activate AMR with $r = 2$ and run the simulation on a single CPU core with our lowest coarse grid resolution 512×512 cells, which means that the effective resolution is 1024×1024. This simulation takes 232.7 seconds to run, a 4.16x speedup with respect to the 1024×1024 resolution single core simulation without AMR. If we now set the refinement ratio to $r = 4$, achieving an effective resolution of 2048×2048, we achieve a speedup of 9.21x with respect to the 2048×2048 simulation with no mesh refinement. Finally, if in addition we use all 16 CPU cores, our simulation takes just 68 seconds, which is a 112x speedup with respect to the 2048×2048 simulation without mesh refinement on a single core.

As a final exercise, we consider the effects of increasing the blocking factor. Using a base resolution of 512×512 cells, a refinement ratio $r = 2$, and a single CPU core, we increase the maximum size of the sub-

grids to 128×128 and then increase the blocking factor progressively from 8 to 64, doubling it each time. The results of this experiment are shown on Table 3. We find that increasing the blocking factor in this scenario leads to an increase in the simulation time, for the simple reason that it forces the algorithm to increase the size of the refined meshes, which increases the refinement coverage ratio α and hence the computational load.

5. Conclusions

We have presented a numerical solver for the two-dimensional compressible Euler equations that makes use of adaptive mesh refinement as implemented by the AMReX software package. We have tested this solver using a number of standard one- and two-dimensional tests containing different types of discontinuities and shown that a properly tuned AMR simulation can achieve errors and convergence rates comparable to those of a simulation performed using an evenly spaced grid with a much larger number of cells. We have also investigated the effects of AMR on simulation time and determined that it can be used to significantly speed up simulation completion. Used in conjunction with parallelization, it is clear that adaptive mesh refinement can be a very powerful tool.

The solver presented can be enhanced in a number of ways. First, it could be extended to three dimensions, allowing it to perform more realistic simulations. Second, it could be adapted to other coordinate systems, such as spherical or cylindrical; this would be very useful in tests similar to the cylindrical explosion test of subsection 4.3. Another helpful modification would be to enhance the cell tagging procedure: we have seen that for some tests such as Test 2 and the diagonally oriented Test 1 the cell tagging procedure that we have implemented was not able to refine the grid in an ideal way. Finally, our scheme could be extended to other systems of hyperbolic PDEs, such as the equations of ideal magnetohydrodynamics, and higher order convergence could be achieved by going beyond the MUSCL-Hancock reconstruction procedure and instead using the PPM scheme [28] or the ENO and WENO schemes [29].

References

[1] W. Zhang, A. Myers, K. Gott, A. Almgren, J. Bell, Amrex: Block-structured adaptive mesh refinement for multiphysics applications, The International Journal of High Performance Computing Applications 35 (6) (2021) 508–526.

[2] K. S. Thorne, R. D. Blandford, Modern classical physics: optics, fluids, plasmas, elasticity, relativity, and statistical physics, Princeton University Press, 2017.

[3] M. J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, Journal of computational Physics 53 (3) (1984) 484–512.

[4] M. J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, Journal of computational Physics 82 (1) (1989) 64–84.

[5] S. K. Godunov, I. Bohachevsky, Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics, Matematičeskij sbornik 47 (3) (1959) 271–306.

[6] E. F. Toro, M. Spruce, W. Speares, Restoration of the contact surface in the hll-riemann solver, Shock waves 4 (1994) 25–34.

[7] B. Van Leer, On the relation between the upwind-differencing schemes of godunov, engquist–osher and roe, SIAM Journal on Scientific and statistical Computing 5 (1) (1984) 1–20.

[8] C. B. Laney, Computational gasdynamics, Cambridge university press, 1998.

[9] E. F. Toro, Riemann solvers and numerical methods for fluid dynamics: a practical introduction, Springer, 2009.

[10] S. M. Millmore, N. Nikiforakis, L. Michael, Computational continuum modelling lecture notes (2022).

[11] S. Chen, G. D. Doolen, Lattice boltzmann method for fluid flows, Annual review of fluid mechanics 30 (1) (1998) 329–364.

[12] S. M. Millmore, Advanced continuum modelling lecture notes (2022).

[13] B. Van Leer, Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method, Journal of computational Physics 32 (1) (1979) 101–136.

[14] A. Harten, P. D. Lax, B. v. Leer, On upstream differencing and godunov-type schemes for hyperbolic conservation laws, SIAM review 25 (1) (1983) 35–61.

[15] E. F. Toro, L. O. Müller, A. Siviglia, Bounds for wave speeds in the riemann problem: Direct theoretical estimates, Computers & Fluids 209 (2020) 104640.

[16] S. Davis, Simplified second-order godunov-type methods, SIAM Journal on Scientific and Statistical Computing 9 (3) (1988) 445–473.

[17] P. Blakely, Continuum modelling - amr lecture notes (2023).

[18] J. Bell, M. Berger, J. Saltzman, M. Welcome, Three-dimensional adaptive mesh refinement for hyperbolic conservation laws, SIAM Journal on Scientific Computing 15 (1) (1994) 127–138.

[19] J. Bell, A. Almgren, V. Beckner, M. Day, M. Lijewski, A. Nonaka, W. Zhang, Boxlib user guide, github. com/BoxLib-Codes/BoxLib (2012).

[20] E. Schnetter, S. H. Hawley, I. Hawke, Evolutions in 3d numerical relativity using fixed mesh refinement, Classical and quantum gravity 21 (6) (2004) 1465.

[21] P. Colella, D. T. Graves, T. Ligocki, D. Martin, D. Modiano, D. Serafini, B. Van Straalen, Chombo software package for amr applications design document, Available at the Chombo website: http://seesar. lbl. gov/ANAG/chombo/(September 2008) (2009).

[22] G. L. Bryan, M. L. Norman, B. W. O'Shea, T. Abel, J. H. Wise, M. J. Turk, D. R. Reynolds, D. C. Collins, P. Wang, S. W. Skillman, et al., Enzo: An adaptive mesh refinement code for astrophysics, The Astrophysical Journal Supplement Series 211 (2) (2014) 19.

[23] B. Fryxell, K. Olson, P. Ricker, F. Timmes, M. Zingale, D. Lamb, P. MacNeice, R. Rosner, J. Truran, H. Tufo, Flash: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes, The Astrophysical Journal Supplement

Series 131 (1) (2000) 273.
[24] A. Dubey, A. Almgren, J. Bell, M. Berzins, S. Brandt, G. Bryan, P. Colella, D. Graves, M. Lijewski, F. Löffler, et al., A survey of high level frameworks in block-structured adaptive mesh refinement packages, Journal of Parallel and Distributed Computing 74 (12) (2014) 3217–3227.
[25] O. Runborg, Sf2520 applied numerical methods lecture notes (2020).
[26] E. F. Toro, Advanced computational algorithms for pdes - lecture 5 (2021).
[27] V. Titarev, Derivative riemann problem and ader schemes, Ph.D. thesis (2005).
[28] P. Colella, P. R. Woodward, The piecewise parabolic method (ppm) for gas-dynamical simulations, Journal of computational physics 54 (1) (1984) 174–201.
[29] X.-D. Liu, S. Osher, T. Chan, Weighted essentially non-oscillatory schemes, Journal of computational physics 115 (1) (1994) 200–212.