Adaptive mesh refinement for the compressible Euler equations using AMReX

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

Abstract

Adaptive Mesh Refinement (AMR) can be used to alter the domain's resolution dynamically in response to an evolving simulation. While applying AMR to the solution of the Euler equations, regions where sharp feature exists such as shock and contact discontinuity will be tagged and refined to higher resolution. This will result in a more accurate numerical approximation of the solution near areas where error are the greatest. This approach lowers the computational cost and memory overhead as compared to performing the simulation on regular mesh without any sacrifices on accuracy of the presentation.

1. Introduction

Adaptive Mesh Refinement (AMR) was developed based on the idea of having multiple component grid while solving the hyperbolic partial differential equations.[13] This technique is commonly applied to regions where the solution is difficult to approximate numerically such as steep gradients, shocks or discontinuities.[2] AMReX is a software developed as part of the U.S DOE Exascale Computing Project for massively parallel and block structured adaptive mesh refinement applications.

In this paper, AMRex will be used to perform adaptive mesh refinement on the compressible Euler equations in 1-D and 2-D. The HLLC solver proposed by Toro et al. is used to solve the Euler equations together with MUSCL-Hancock scheme to achieve second order accuracy in space. Dimensional splitting approach is used in this paper to extend the 1-D Euler system to 2-D.

This paper is structured as follow. Section 1 is the introduction. Section 2 provides an outline of the literature surrounding the HLLC solver and AMR. Section 3 details the governing equations for ideal Euler system. Section 4 describes the HLLC solver used in the paper to solve the system of equations. Section 5 includes a summary of the key aspects of AMReX. In section 6, results for several 1-D and 2-D tests are shown. Conclusions are presented in section 7.

2. Literature Review

Godunov's method has been widely used to solve hyperbolic partial differential equations with discontinuous solutions due to the ability to achieve high resolution at discontinuities and robustness of the scheme. Upwind scheme for prescribing Godunov flux use wave propagation information contained in the differential equations, which can be obtained by solving a local 1-D Riemann problem in the direction normal to cell interface.[10] Hence, the essential ingredients for adopting Godunov's method is the solution of a Riemann problem, whether it is an exact or approximated solution. Riemann solvers can also be classified as complete or incomplete depending whether all the characteristic fields present in the exact Riemann problem are being included in the wave model. For solving Euler equations, exact Riemann solvers exist but the requirement for an iterative procedure raises some concern in the computational cost. The problem becomes apparent when equations of state with complicated algebraic form or complex system of equations are needed to be solved.

Harten, Lax, and van Leer (HLL) has proposed an approximate Riemann solver that will yield positive solution if used with an appropriate choice of wave speed bounds. This scheme assumes the solution is a wave configuration that consists of two waves separating three constant states.

Toro et al. has presented a different approach by adopting a 3 wave model with 2 intermediate states

in the Riemann problem solution. The assumption of intermediate left and right states having the same pressure and velocity is valid for the contact wave, hence the solver is named HLLC ("C" stands for Contact).[10] Given that Euler equations have 3 distinct characteristic fields, the HLLC solver proposed is a complete Riemann solver for the Euler equations as the approximated wave structure encapsulates all the characteristic fields of the problem.

Adaptive method can be structured or unstructured depending on the way numerical solution is being presented. Unstructured triangulation offers superior geometrical flexibility but the coordinates for all the vertices need to be stored explicitly using graph or tree representations. This is advantageous for dynamical adaptation as tagged cells for refinement can be simply replaced by finer ones and the entire grid can be advanced simultaneously. However, for time-explicit finite volume scheme, the requirement for a global time step which satisfy the CFL condition for the smallest cell can lead to unnecessarily long simulation time. Additionally, the re-numbering and re-arrangement of cells' coordinates based on mesh topology can lead to high computational cost. On the other hand, the structured approach formulates the numerical scheme on a logically rectangular mesh, moderating some of the technical difficulties in unstructured approach. Furthermore, the use of a global integer coordinate system enable easy storage and evaluation of neighborhood relationship.[3]

Generally, there are three methods for doing grid refinement: r-refinement, p-refinement and h-refinement. With r-refinement, the cells are dynamically redistributed to increase resolution in specific parts of the grid while sacrificing the resolution somewhere else in the domain. The main advantage of this method is having a smooth transition between resolutions but requires complicated remapping of mesh. Secondly, the p-refinement works by holding the cell size fixed but changes the order of polynomial approximation within each cell to increase resolution locally. This is typical of polynomial spectral method and can usually lead to exponential decrease in numerical error.[6] Lastly, the AMR, another term for h-refinement overlay finer resolution cells over coarse grid to increase resolution locally.[4] While keeping the polynomial order constant, this method can usually lead to algebraically decay of numerical error.[6] In this paper, h-refinement or AMR will be discussed in details in Section 5 and implemented in all the test cases.

3. Mathematical Equations

3.1. Conservation Law for Ideal Euler

In this paper, time-dependent Euler equations will be solved within the framework of AMReX software. These equations are a system of non-linear hyperbolic conservation laws governing the dynamics of a compressible material, which are commonly used for modelling liquids or gasses at high pressures with negligible heat flux, viscous stresses and body forces.[10] Applying the fundamental laws of conservation of mass, Newton's Second Law and the conservation of energy, the Euler equations can be derived in the conserved variable form

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

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v} + \mathbf{I} p) = 0 \tag{1b}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot [(E + p)\mathbf{v}] = 0. \tag{1c}$$

Here ρ is the density, $\mathbf{v} = (v_x, v_y, v_z)^T$ the flow velocity, $E = \rho\varepsilon + \frac{1}{2}\rho \mathbf{v} \cdot \mathbf{v}$ the total energy, ε the specific internal energy and p the hydrodynamic pressure.

Given the Euler equations apply to a wide range of materials ranging from air to water, material specific behaviours need to be accounted for in the equations. The equation of state is a continuum description of the materials' thermodynamic properties.[7] In this paper, the ideal gas equation of state has been used in all the test cases. The equation can be written in the form which relates internal energy of a material to density and pressure,

$$p = (\gamma - 1)\rho\varepsilon \tag{2}$$

where $\gamma = \frac{C_p}{C_v}$. In this paper, $\gamma = 1.4$ has been used in all the test cases.

3.2. 1-D Ideal Euler

Given the basis of numerical method used to solve the Euler system only considers one dimensional equations, derivatives and Riemann problem, the evolution equations need to be written in the following form.[7]

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{u})}{\partial x} + \frac{\partial \mathbf{g}(\mathbf{u})}{\partial y} + \frac{\partial \mathbf{h}(\mathbf{u})}{\partial z} = 0 \tag{3}$$

Here $\mathbf{u}$ is the conserved variable, $\mathbf{f}(\mathbf{u})$, $\mathbf{g}(\mathbf{u})$, $\mathbf{h}(\mathbf{u})$ are the flux functions in x, y, z directions respectively.

The standard conservation form in Equations (1a) - (1c) can be written in 1-D as follow.

$$\frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho v_x \\ E \end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix} \rho v_x \\ \rho v_x^2 + p \\ (E+p)v_x \end{pmatrix} = 0 \tag{4}$$

The conservation form can also be written in primitive variable form which are variables comprising the conserved variables and easier to be physically measured.

$$\frac{\partial}{\partial t}\begin{pmatrix} \rho \\ v_x \\ p \end{pmatrix} + \begin{pmatrix} v_x & \rho & 0 \\ 0 & v_x & \frac{1}{\rho} \\ 0 & \rho c_s^2 & v_x \end{pmatrix} \frac{\partial}{\partial x}\begin{pmatrix} \rho \\ v_x \\ p \end{pmatrix} = 0 \tag{5}$$

3.3. 2-D Ideal Euler

When considering 2-D Euler system, evolution equations for both velocity components v_x and v_y are needed as velocity and momentum are vector quantities. As the pressure term only acts in the normal direction, it is only present in the x-component of ρv_x and y-component of ρv_y. The conservative form in Equations (1a) - (1c) can be written as follow

$$\frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ E \end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix} \rho v_x \\ \rho v_x^2 + p \\ \rho v_x v_y \\ (E+p)v_x \end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix} \rho v_y \\ \rho v_x v_y \\ \rho v_y^2 + p \\ (E+p)v_y \end{pmatrix} = 0. \tag{6}$$

Here the total energy, E is a combination of internal energy and kinetic energy. The kinetic energy which contains a contribution from both x- and y- velocity can be computed from

$$\text{KE} = \frac{1}{2}\rho \mathbf{v}\cdot\mathbf{v} = \frac{1}{2}\rho(v_x^2 + v_y^2). \tag{7}$$

3.4. Euler Wave Speed

The fact that solving the 1-D primitive variable form gave three independent eigenvalues means the Euler equations comprise three waves. Additionally, the eigenvalues also represents the wave speeds of the 1-D system.

$$\lambda_1 = v - c_s, \quad \lambda_2 = v, \quad \lambda_3 = v + c_s \tag{8}$$

While extending to 2-D, solving the primitive form gave four real but not distinct eigenvalues. This implies that the system is hyperbolic, but not strictly hyperbolic and the waves are not distinct to one another. The forth wave is labeled as the shear wave which is a wave with velocity moving perpendicular to the wave, a transverse velocity, jumps, while other variables do not.[7]

4. Numerical Solver

4.1. HLL Riemann Solver

The HLL solver is a two wave model which requires estimates for the fastest wave speeds emerging from contact discontinuity. As shown in Section 3.2, the Euler equations can be written as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla\cdot\mathbf{F}(\mathbf{u}) = 0. \tag{9}$$

The discretised form of (9) in x-direction using finite volume method can be written as

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{f}_{i+1/2}^n - \mathbf{f}_{i-1/2}^n\right). \tag{10}$$

Here $\mathbf{f}_{i\pm1/2}^n$ is the inter-cell flux which governing the amount of materials entering or leaving the cell.

Constructing a Riemann problem along the x-direction with piece-wise constant left and right initial states ($t = 0$) give

$$\mathbf{u}(x,0) = \begin{cases} \mathbf{u}_L, & \text{if } x < 0 \\ \mathbf{u}_R, & \text{if } x > 0 \end{cases} \tag{11}$$

Using the HLL appoach, the approximated Riemann solution can be obtained as

$$\mathbf{u}_{HLL} = \begin{cases} \mathbf{u}_L, & \text{if } S_L > 0 \\ \mathbf{u}^*, & \text{if } S_L \le 0 \le S_R \\ \mathbf{u}_R, & \text{if } S_R < 0 \end{cases} \tag{12}$$

Here the S_L and S_R are the fastest wave speeds for the left and right states. The intermediate subsonic state u^* can be calculated using

$$\mathbf{u}^* = \frac{S_R\mathbf{u}_R - S_L\mathbf{u}_L - (\mathbf{f}_R - \mathbf{f}_L)}{S_R - S_L} \tag{13}$$

where $f_L = f(u_L)$ and $f_R = f(u_R)$. Note that for the intermediate flux $f_{HLL} \neq f(u_{HLL})$ and must be computed using

$$\mathbf{f}_{HLL} = \frac{S_R\mathbf{f}_L - S_L\mathbf{f}_R + S_LS_R(\mathbf{u}_R - \mathbf{u}_L)}{S_R - S_L}. \tag{14}$$

4.2. HLLC Riemann Solver

The HLLC solver is a three wave model with the addition of a contact wave onto the existing two waves in HLL. This results in two intermediate states $\mathbf{u}_L^*$ and $\mathbf{u}_R^*$, separated by an interface travelling at speed S^*. The HLLC approximate Riemann solution is

$$\mathbf{u}_{HLLC}(x,t) = \begin{cases} \mathbf{u}_L, & \text{if} \quad \frac{x}{t} \le S_L \\ \mathbf{u}_L^*, & \text{if} \quad S_L \le \frac{x}{t} \le S^* \\ \mathbf{u}_R^*, & \text{if} \quad S^* \le \frac{x}{t} \le S_R \\ \mathbf{u}_R, & \text{if} \quad \frac{x}{t} \ge S_R \end{cases} \tag{15}$$

where the u_L^* and u_R^* can be calculated using the Rankine-Hugoniot conditions

$$\mathbf{F}_L^* = \mathbf{F}_L + S_L\left(\mathbf{u}_L^* - \mathbf{u}_L\right) \tag{16}$$

$$\mathbf{F}_R^* = \mathbf{F}_R + S_R\left(\mathbf{u}_R^* - \mathbf{u}_R\right). \tag{17}$$

4.3. HLLC Solver for Multidimensional Euler

In order to derive a HLLC solver specific to Euler, several conditions need to be imposed. In 1-D Euler,

$$\begin{aligned} p_L^* &= p_R^* = p^*, \\ v_{x,L}^* &= v_{x,R}^* = v^*. \end{aligned}$$

In multidimensional Euler, extra conditions are needed for the tangential velocities,

$$\begin{aligned} v_{y,L}^* &= v_{y,L} \quad , \quad v_{y,R}^* = v_{y,R} \\ v_{z,L}^* &= v_{z,L} \quad , \quad v_{z,R}^* = v_{z,R}. \end{aligned}$$

Additionally, we assume that the normal velocity component in the intermediate state can be computed from the intermediate wave speed

$$S^* = v^*.$$

By implementing the conditions on the Rankine-Hugoniot relationships defined in (16) and (17), the intermediate wave speed and HLLC solver for multidimensional Euler equation in x-direction can be derived.[12]

$$S^* = \frac{p_R - p_L + \rho_L v_{x,L}\left(S_L - v_{x,L}\right) - \rho_R v_{x,R}\left(S_R - v_{x,R}\right)}{\rho_L\left(S_L - v_{x,L}\right) - \rho_R\left(S_R - v_{x,R}\right)} \tag{18}$$

$$\mathbf{u}_K^* = \rho_K\left(\frac{S_K - v_{x,K}}{S_K - S_*}\right)\begin{bmatrix} 1 \\ S_* \\ v_{y,K} \\ v_{z,K} \\ \frac{E_K}{\rho_K} + \left(S_* - v_{x,K}\right)\left[S_* + \frac{p_K}{\rho_K\left(S_K - v_{x,K}\right)}\right] \end{bmatrix} \tag{19}$$

where $K \in \{L, R\}$.

4.4. Wave Speed Estimate

In order to use the approximate Riemann solvers, bound estimates for the two fastest wave speeds emerging from the solution of Riemann problem needs to be provided.[10] For solving Euler system, maximum and minimum wave speeds S_L and S_R can be computed from the sum of velocity and wave speed as follow

$$S^+ = \max\left(\left|v_{x,L}\right| + c_{s,L}, \left|v_{x,R}\right| + c_{s,R}\right) \tag{20}$$

$$S_L = -S^+, \quad S_R = S^+. \tag{21}$$

where $c_{s,K}$ is the speed of sound. For an ideal gas,

$$c_{s,K} = \sqrt{\frac{\gamma p_K}{\rho_K}} \tag{22}$$

4.5. Dimensional Splitting

Dimensional splitting approach has been adopted in this paper to solve the multidimensional Euler system numerically. In this approach, 1-D method is being applied to each coordinate direction, starting with the x-direction update followed by y-direction update.[10]

$$\begin{aligned} \overline{\mathbf{u}}_{i,j} &= \mathbf{u}_{i,j}^n + \frac{\Delta t}{\Delta x}\left(\mathbf{f}_{i-1/2,j}^n(\mathbf{u}) - \mathbf{f}_{i+1/2,j}^n(\mathbf{u})\right) \\ \mathbf{u}_{i,j}^{n+1} &= \overline{\mathbf{u}}_{i,j} + \frac{\Delta t}{\Delta y}\left(\mathbf{g}_{i,j-1/2}^n(\overline{\mathbf{u}}) - \mathbf{g}_{i,j+1/2}^n(\overline{\mathbf{u}})\right) \end{aligned} \tag{23}$$

This method allows diagonal movement of information and the standard CFL number can be used without any reduction. In addition, the solution can be extended to second order accurate in space and time using Strang splitting method.[7]

$$\mathbf{u}_{i,j}^{n+1} = \frac{1}{2}\left[\mathcal{Y}^{(\Delta t)}\mathcal{X}^{(\Delta t)} + \mathcal{X}^{(\Delta t)}\mathcal{Y}^{(\Delta t)}\right](\mathbf{u}^n) \tag{24}$$

Note that in this paper Strang splitting was not implemented.

4.6. Second Order Extension

The Monotone Upstream-centred Scheme for Conservation Laws (MUSCL) approach allows the construction of higher order schemes by modifying the piece-wise constant data. In order to maintain the monotonicity of the scheme, the reconstruction is constrained so spurious oscillation can be avoided. [10]

4.6.1. Data Reconstruction

The first step for constructing a second order extension flux $\mathbf{f}_{i+0.5}$ is to replace the cell averaged values by reconstructed boundary values using derivative approximation.

$$\mathbf{u}_i^L = \mathbf{u}_i^n - \frac{1}{2}\Delta_i; \quad \mathbf{u}_i^R = \mathbf{u}_i^n + \frac{1}{2}\Delta_i \tag{25}$$

Given that derivatives do not exist at discontinuities, spurious oscillations can be generated in regions of strong gradients. This aligns with the Godunov's Theorem which states that the desirable properties of accuracy and monotonicity are contradictory requirements.[10] In order to construct a second order Total Variation Diminishing (TVD) scheme, the slopes Δ_i in the reconstruction steps can be replaced by limited slopes $\bar{\Delta}_i = \xi_i \Delta_i$, according to some TVD constraints. The reconstruction will be fully limited to piece-wise constant in cases of high change in slope while no limiting will be applied in cases of smooth change in slope. In this paper, VAN LEER-type slope limiter has been used to limit the slopes. A VAN LEER-type slope limiter is

$$\xi_{VL}(r) = \begin{cases} 0, & \text{if } r \leq 0 \\ 2r, & \text{if } 0 \leq r \leq \frac{1}{2} \\ 1, & \text{if } \frac{1}{2} \leq r \leq 1 \\ \min\{r, \xi_R(r), 2\} & \text{if } r \geq 1 \end{cases} \tag{26}$$

4.6.2. MUSCL Hancock Scheme

Within each cell, the boundary extrapolated values are evolved locally at half time step as

$$\begin{aligned} \overline{\mathbf{u}}_i^L &= \mathbf{u}_i^L + \frac{1}{2}\frac{\Delta t}{\Delta x}\left[\mathbf{f}\left(\mathbf{u}_i^L\right) - \mathbf{f}\left(\mathbf{u}_i^R\right)\right] \\ \overline{\mathbf{u}}_i^R &= \mathbf{u}_i^R + \frac{1}{2}\frac{\Delta t}{\Delta x}\left[\mathbf{f}\left(\mathbf{u}_i^L\right) - \mathbf{f}\left(\mathbf{u}_i^R\right)\right] \end{aligned} \tag{27}$$

The evolved boundary values $\overline{\mathbf{u}}_i^L$ and $\overline{\mathbf{u}}_{i+1}^R$ are then being treated as the initial values for HLL and HLLC Riemann Solver to produce second order accurate numerical simulation. Once the intercell fluxes have been computed, Godunov's method (10) can be used as a means to update the data.

5. AMReX

5.1. Adaptive Mesh Refinement

Fundamentally, the algorithm present a hierarchical solution at various level of resolution. Grid situated at the bottom of the hierarchy is known as the coarse or base grid and the grid size should be fine enough to capture most of the features of the solution. At specific regions of interest, resolution can be increased locally by introducing finer child grid patches nested within the boundaries of the corresponding coarser grid. Note that the base grid will not change over revolution. Refinement is applied on both space and time hence grids with differing size will be advanced on different time step. [1] One of the upsides of using the hierarchical AMR is the preservation of the Cartesian grid structure hence the same numerical method can be applied to both coarse and fine grids. Additionally, this technique breaks down the base grid into sub-domains and allows the use of different methods on different regions, e.g. lower order method for shocks and higher order method on regions where a smooth solution is expected.[1]

5.2. Grid Generation

The error estimation and adaptation steps are to ensure the fine meshes move in line with the flow features, maintaining high grid resolution at regions of interest. Error estimation is done by considering the local cell to cell difference of the tracer value[5],

$$\xi_{ijk} = \max\left(\left|\varphi_{i+1jk} - \varphi_{i-1jk}\right|, \left|\varphi_{ij+1j} - \varphi_{ij-1k}\right| \right. \\ \left. \left|\varphi_{ijk+1} - \varphi_{ijk-1}\right|\right) \tag{28}$$

where ξ is the error estimate and φ is the tracer value. The cells will be flagged for refinement when the error estimate exceeds a pre-defined value. Additionally, any cells that are about to be overlaid by a new finer mesh will be tagged to ensure the adapted meshes remain nested.

Flagged cells are then clustered into rectangular patches. The clustering method described by Bell et al. (1994) requires summing the number of flagged cells in each plane and partition when a zero has been found. New clusters are then formed by taking the smallest enclosing rectangular patch covering the flagged cells in each partition, subjected to the efficiency condition. Efficiency is simply the ratio of tagged cells over total number of cells in the patch. Typically, efficiency between 0.6 to 0.9 is selected as a balance between few large meshes and many small meshes.[5]

5.3. Boundary Condition

Adaptive mesh is not recommended to be updated frequently due to the associated computational cost.[2] In order to ensure features can propagate more than one cell before regridding happens, each mesh is surrounded by ghost/buffer cells which defines the boundary conditions for the mesh. The values in these cells are copied over from neighboring meshes at the same grid level if possible or filled by interpolated values from the underlying coarse mesh if the earlier approach is not possible. Additionally, buffer cells for refined meshes which overlaps the domain boundary are set using the physical boundary conditions.[2] Note

that in this paper, transmissive boundary condition has been applied to all the test cases.

5.4. Flux Correction

At the boundaries between coarse and fine meshes, two fluxes present; one from the coarse level while another one as a sum of fluxes from each of the refined cells over sub-cycled time step. In order to maintain the conservation across this boundary, flux correction is performed by taking the difference between these two fluxes and applying to the corresponding coarse grid cell. Within AMReX, there is already an inbuilt command for flux correction algorithm in the *reflux* function.

5.5. Grid Operation

This section demonstrates the sequence of events happening while AMR is being used. Assuming a 2 level grid hierarchy G_0 and G_1 with a refinement ratio of 2, the following grid operation will be carried out. First the coarse grid G_0 is integrated with Δt followed by integration of the fine grid G_1 twice with half this Δt to advance the solution to the same time level as grid G_1. Note that the solution from these integration will be stored regardless of the grid hierarchy. The next step is to project the solution from G_1 onto G_0 for a more accurate representation of results at G_0. Adaptation will be carried out on the fine grid before the second complete integration of G_0. This cycle is repeated until the end time of the simulation has been reached.[5]

5.6. Performance

Generally, parallelisation can be done in two ways; one with all the memory being accessible by all the processors while the other with the memory distributed around the processors hence explicit communication over some form of network is required. The machines carrying out the parallelisation are known as shared memory processor (SMP) for the earlier method and massively parallel processor (MPP) for the later one. For both SMP and MPP, some form of memory controller needs to be in place for parallelisation to occur smoothly. Open MP library is widely used for SMP programming while MPI dominates the space for MPP.[8]

Parallelisation of AMR code can be achieved by decomposing the coarsest level across multiple processors. In order to facilitate interpolation and projection, refined patches are typically being placed on the same processor as their parents. Load balancing algorithm is applied to each AMR level independently. Based on the user-defined `max_grid_size` parameter, large patches are broken down into smaller ones for distribution to multiple processors. This parameter ensures no grid is longer in any direction than the `max_grid_size` cells. In addition to this, the parameter `blocking_factor` enforce the dimensions of every new grids being created is divisible by itself to improve multi-grid efficiency for single-level solvers. The `BoxArray` of new grids at every level are then broadcasted to every MPI rank. `DistributionMapping` describes which process owns the data living on the domains specified by the boxes in a `BoxArray`. During this process, equal distribution of work to the nodes is important to ensure good scaling. The default load balancing algorithm in AMREX is the Morton-ordering space-filling curve (SFC) algorithm which takes in a cost function proportional to the number of cells per grid and implement an initial data distribution across the MPI ranks.[13]

OpenMP is incorporated at the `MFIter` level which is an iterator for looping over the `FArrayBoxes` in `MultiFab`. Tiling is a loop transformation technique into tiling loops which iterate over tiles and element loops which iterate over data elements within a tile. When OpenMP is added above the `MFIter` loop, the `MFIter` includes tiling and the `ParallelFor` translates to a serial loop over the cells in a tile. Note that tiling at `MFIter` will be switched off when compiling with GPU supports and the `ParallelFor` translates to a GPU kernel launch.[9]

6. Test Results

6.1. 1-D Toro's Test

The Euler code implemented in AMReX framework was first tested using the test cases in Table 2. Test 1 (Sod's test) has solution consisting of a left rarefaction, a contact and a right shock. Test 2 is known as the 123 problem and its solution consists of two strong rarefaction and a contact discontinuity in the middle. Test 3 and 4 are the respective left half and right half of Woodward and Colella's blast wave problem. The solution for test 3 contains a left rarefaction, a contact and a right shock; while the solution for test 4 contains a left shock, a contact discontinuity and a right rarefaction. The initial condition for test 5 is made up of the left and right shocks emerging from the solution to test 3 and 4. The expected solution for test 5 consists of a left facing shock which travels to the right,

Table 1: *Convergence analysis for Toro's test.*

Test	**Mesh**	L_1	**Refinement ratio**	$L_1(AMRx2)$	**Refinement ratio**	$L_1(AMRx2x2)$
	128	0.01592	-	-	-	-
1	256	0.01043	0.02	0.01175	-	-
	512	0.00694	-	-	0.02; 0.02	0.01128
	128	0.01789	-	-	-	-
2	256	0.01193	0.002	0.01593	-	-
	512	0.00807	-	-	0.002; 0.001	0.01557
	128	0.202593	-	-	-	-
3	256	0.144503	0.03	0.1463	-	-
	512	0.0996481	-	-	0.03; 0.03	0.1031
	128	0.204278	-	-	-	-
4	256	0.13465	0.03	0.1352	-	-
	512	0.0893195	-	-	0.03; 0.03	0.09722
	128	0.873503	-	-	-	-
5	256	0.563338	0.6	0.6125	-	-
	512	0.403056	-	-	0.6; 0.6	0.4035

a contact discontinuity moving to the right and right travelling shock wave.[10]

These test cases have been simulated at several grid resolutions of N=128, 256, 512 across the simulation domain of $x \in \{0, 1\}$. The simulations have been initiated using the data in Table 2 with the initial discontinuity at $x = 0.5$. Transmissive boundary condition and Cfl number of 0.8 have been applied to all the test cases. The plots for density, velocity, pressure and specific internal energy have been produced for all the test cases. The exact solution for all the test cases have been computed using the exact Riemann Solver and included in the plots. Results for test case 1 are presented in Figure 2a to 2d while the rest of the test cases' results are included in Appendix A. As observed in the plots, the numerical solution matches the expected characteristics for all the test cases with increasing accuracy as the mesh resolution increases.

The test cases are then repeated at the lowest resolution of N=128 but with adaptive mesh refinement. In order to limit the refinement to only regions of sharp features including shock and contact discontinuity, a threshold value φ_{grad} is defined for every individual test cases. The refinement will only happen at regions satisfying the condition of

$$\xi_{ijk} \geq \varphi_{grad} \tag{29}$$

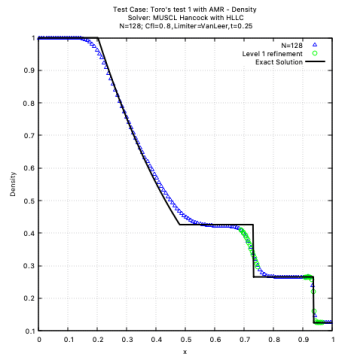
where ξ_{ijk} is the error estimate defined in (28) with φ being the value of density for all the test cases. The values of φ_{grad} used for both level 1 and 2 refinements are presented in Table 1. The results for test 1 with refinement are presented in Figure 1a and 1b while the rest of the plots are included in the Appendix B.

Convergence analysis has been performed on both the numerical simulations with and without mesh refinement. The $L_1 - norms$ are calculated from the sum of all the local errors as follow.[11]

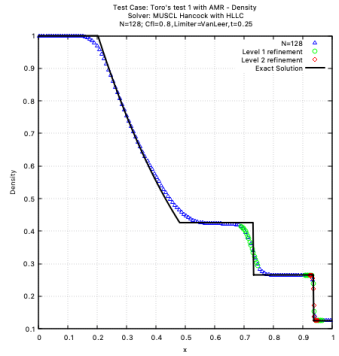
$$L_1 = \left(\Delta x \sum_{i=-\infty}^{i=\infty} \left| \varphi_i^N - \varphi\left(x_i, T_N\right) \right| \right)$$

For calculating the $L_1 - norms$ of simulations with AMR, local errors are computed based on the smallest numerical stencil present.

As shown in Table 1, for numerical results without AMR, $L_1 - norms$ decreases as the resolution increases validate the theory of increasing accuracy as the number of meshes increase. Furthermore, for simulations running with AMR, the $L_1 - norms$ show a decreasing trend as the number of refinement level increases. It can be observed that the $L_1 - norms$ for test runs using AMR are greater than the equivalent simulations with resolution matches their highest refined level. This can be explained by the fact that in AMR simulations, not all the cells will be refined hence error in these unrefined cells are not improved.



(a) Density plot with AMR Level 1



(b) Density plot with AMR Level 2

Figure 1: Toro's test 1 with AMR

Table 2: *Initial data for Toro's test.*

Test	u_L	**Value**	u_R	**Value**	Time
	ρ	1.0	ρ	0.125	
1	v	0.0	v	0.0	0.25
	p	1.0	p	0.1	
	ρ	1.0	ρ	1.0	
2	v	-2.0	v	2.0	0.15
	p	0.4	p	0.4	
	ρ	1.0	ρ	1.0	
3	v	0.0	v	0.0	0.012
	p	1000.0	p	0.01	
	ρ	1.0	ρ	1.0	
4	v	0.0	v	0.0	0.035
	p	0.01	p	100.0	
	ρ	5.99924	ρ	5.99242	
5	v	19.5975	v	-6.19633	0.035
	p	460.894	p	46.0950	

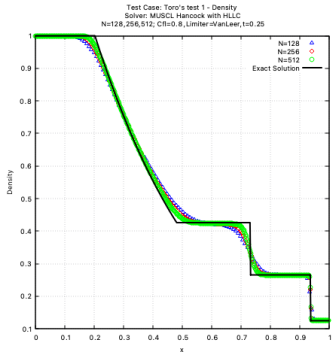
6.2. 2-D Toro's test

The 2-D Euler code is first tested using the cases in Table 2 at a resolution of N=128*128 across a square domain of $x \in \{0, 1\}, y \in \{0, 1\}$ without AMR. Two versions for each test cases have been carried out; first one where the initial contact discontinuity exists at $x = 0.5$, and second one where the initial contact discontinuity exists at $y = 0.5$. Colour plots for the simulation of test 1 presented in Figure 3a to 4d shows that the solutions obtained are identical to the 1-D tests.

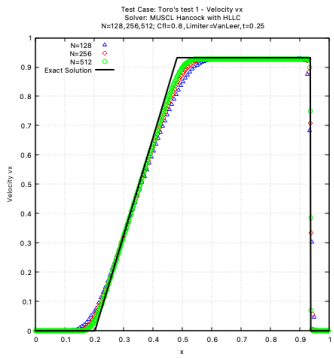
Using the same test cases, the simulations are repeated with AMR using the same refinement condition as defined in Table 1. The test 1 results presented in Figure D.22a to D.22d shows that the refinement is being applied to the same regions as 1-D tests. Finally, the test cases are again repeated but with the initial discontinuity being defined at the line $y = 1 - x$. The test 1 plots presented in Figure 6a to 6d reassured the correct implementation of 2-D fluxes.

Note that the colour plots for tests 2-5 are presented in the Appendix C, D, E and F.

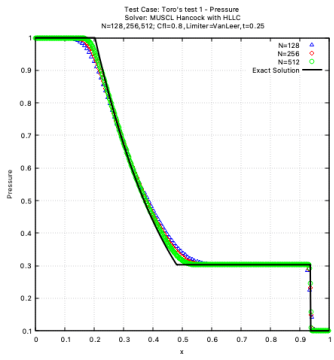
1-D and 2-D Sod test results



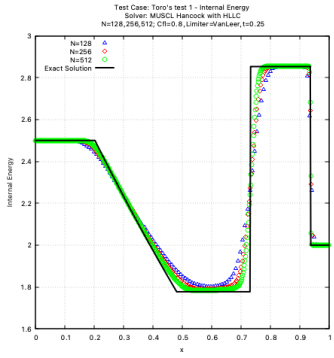
(a) Density plot



(b) Velocity vx plot

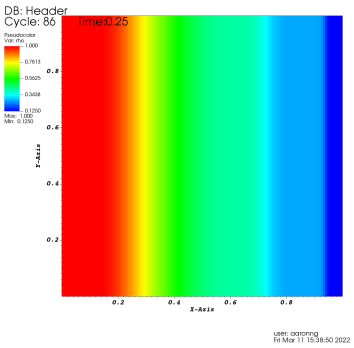


(c) Pressure plot

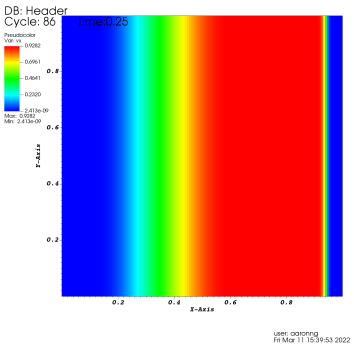


(d) Specific Internal Energy plot

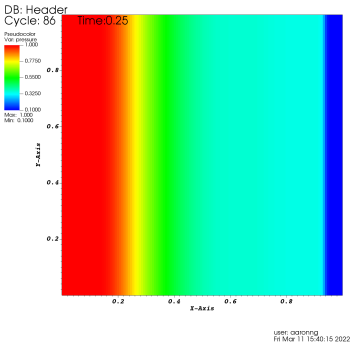
Figure 2: Toro's test 1



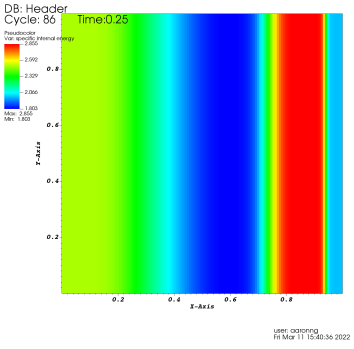
(a) Density plot



(b) Velocity vx plot

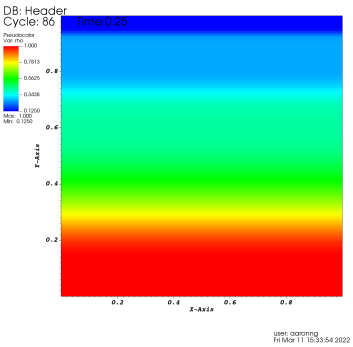


(c) Pressure plot

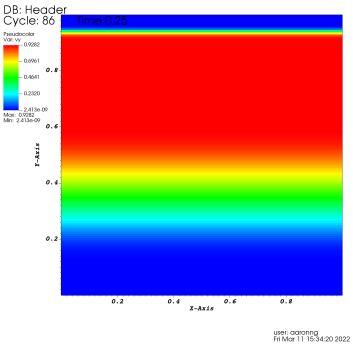


(d) Specific Internal Energy plot

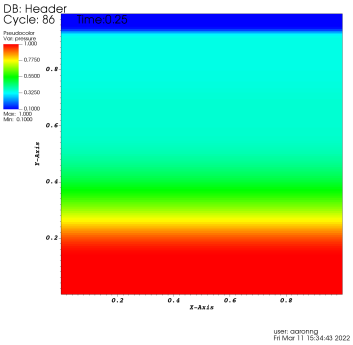
Figure 3: 2-D Toro's test 1 with x-split Initial Condition (IC)



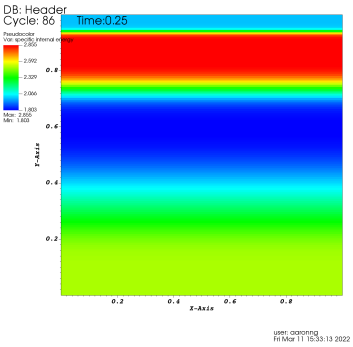
(a) Density plot



(b) Velocity vy plot

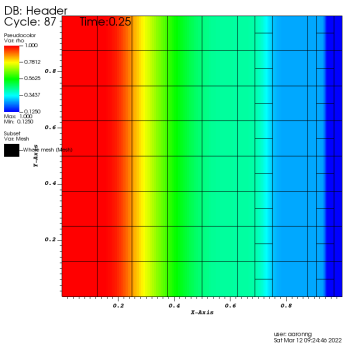


(c) Pressure plot

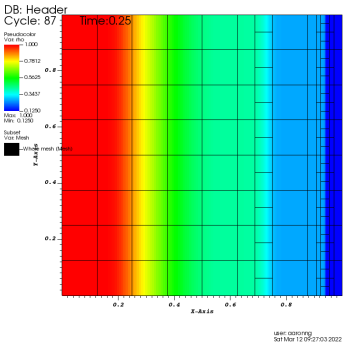


(d) Specific Internal Energy plot

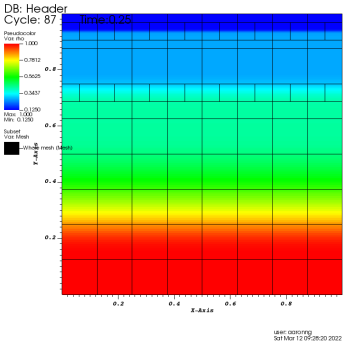
Figure 4: 2-D Toro's test 1 with y-split Initial Condition (IC)



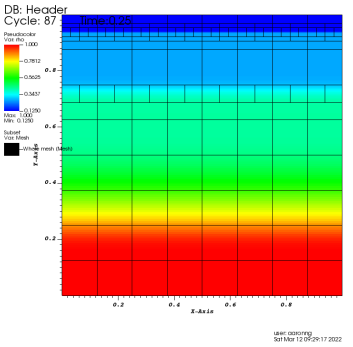
(a) Density plot from x-split IC with AMR x2 refinement



(b) Density plot from x-split IC with AMR x2x2 refinement

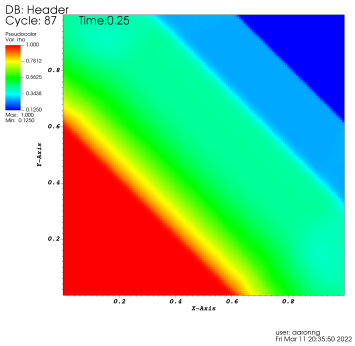


(c) Density plot from y-split IC with AMR x2 refinement

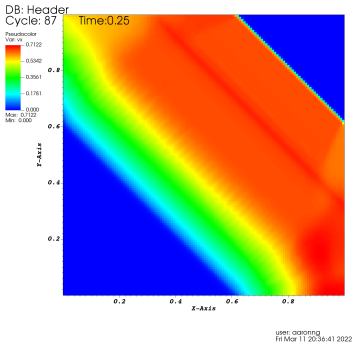


(d) Density plot from y-split IC with AMR x2x2 refinement

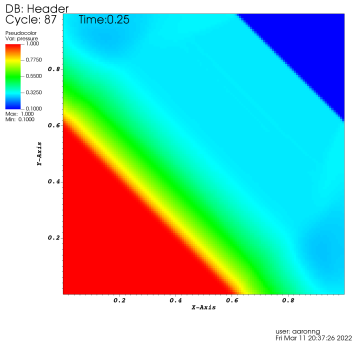
Figure 5: 2-D Toro's test 1 with AMR



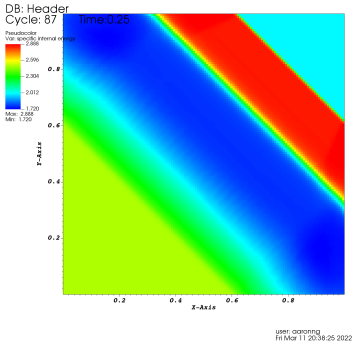
(a) Density plot



(b) Velocity vx plot



(c) Pressure plot



(d) Specific Internal Energy plot

Figure 6: 2-D Toro's test 1 with diagonally aligned initial condition

6.3. Cylindrical Explosion test

For this test case, the 2-D Euler equations are being solved on a square domain of $x \in \{0, 2\}, y \in \{0, 2\}$. The initial condition is made up of a region inside a circle of radius R=0.4 centred at (1,1) and the region outside the circle. Transmissive boundary condition has been applied and the limiter function used at the slope reconstruction step is VAN-LEER. The values of the initial condition are as follow

$$\begin{aligned} \rho_{ins} &= 1.0, \quad \rho_{\text{out}} = 0.125, \\ u_{ins} &= 0.0, \quad u_{\text{out}} = 0.0, \\ v_{ins} &= 0.0, \quad v_{\text{out}} = 0.0, \\ p_{ins} &= 1.0, \quad p_{\text{out}} = 0.1. \end{aligned} \tag{30}$$

where subscripts *ins* and *out* represent inside and outside the circle respectively. The expected solution for this test case is a circular shock travelling outwards from centre with the contact surface and a circular rarefaction travelling towards the centre.

The test case has been simulated at different resolutions (N=128*128, N=256*256, N=512*512) without AMR. The density plots are presented in Figure 7a to 7c in an increasing resolution order. As observed in these plots, the increasing resolution improves the display of the sharp features hence accuracy of the solution. Additionally, the test cases are simulated with the lowest resolution of N=128*128 using AMR. The AMR simulations are repeated at different refinement condition ($\varphi_{grad} = 0.005, 0.015, 0.025$) to demonstrate the effect of φ_{grad}. Figure 8a to 8c show the results for one level of refinement while Figure 9a to 9c show the results for two level of refinement. As observed in the plots, refinement happens mainly at regions with sharp features such as shock, contact and rarefaction. In addition, increasing the φ_{grad} leads to lesser area being refined as the refinement condition becomes more stringent.

6.4. 2-D Timing Exercise

In this exercise, the cylindrical explosion test described in Section 6.3 is used to investigate the simulation time. For consistency, all the simulation runs for timing tests have been performed using the same machine within the Laboratory of Scientific Computing, Hydra 04. The specifications for this machine have been detailed in Table 3. In addition, all the tests have been repeated three times and the presented values in Table 5, 6 and 7 are average values of the repeated runs.

Table 3: *Hardware specifications.*

Property	**Value**
Machine	Hydra04
Chip	Intel(R)Xeon(R)CPU E5-2650 0
OS	Linux (Ubuntu)
RAM Storage(GB)	64
Core	16
CPU (GHz)	2.00

The test is first simulated without AMR at 3 resolutions of N=256*256, N=384*384, N=512*512 and the simulation time are recorded in Table 5. For each resolution, the test is repeated using different number of cores (2,4,8,16) and threads (2,4,8,16). The speed up in performance is taken as the ratio of simulation time with no parallelism to the respective simulations with parallelism at the same resolution. As observed in Table 5, the simulation speed increases as the number of cores being used increases from 2 to 16. The uplift in performance becomes significant at high resolution, with simulation for N=512*512 using 16 cores running at 10 times faster than running on a single core. This phenomena is also known as strong scaling and can be explained by the increasing work load on each processors which outweighs the communication cost between them.[2] For the test cases using Open MP to parallelise over threads, the speed up in simulation time lags behind MPI. This is showing weak scaling as the communication cost between higher number of threads have now outweighed the computational cost. In order to fully benefit from the parallelism being offered by OpenMP, the resolution can be increased to N=1024*1024 to add more work load to each processors.

AMR is then used to run the test at N=256*256 with one level of refinement to increase the mesh resolution at regions with sharp features. After several trial and errors, the value of φ_{grad} has been set to 0.015 as this value ensures the shock and contact are being captured for refinement. As recorded in Table 6, the simulation run using AMR provided an uplift in speed of 2.3 as compared to the simulation at N=512*512. The simulation using AMR is then repeated with different number of cores (2,4,8,16) and threads (2,4,8,16). In summary, the fastest speed up achieved using parallel computing is 16.5, while performing the simulations using AMR over 16 cores. Comparing this to the simulation of N=512*512 using 16 cores which has a speed up of 9.8, the usage of AMR is justifiable.

Up to this point, the default parameters for grid creation as listed in Table 4 have been used for all the 2-D simulations. The `max_grid_size` sets the upper bound for number of cells per grid that can exist in every direction while the `blocking_factor` essentially sets the lower bound. The `n_error_buf` sets the number of buffer cells around each tagged cell. Another parameter worth mentioning is the `grid_eff` which ensure the grids do not contain too large fraction of untagged cells.[9]

Table 4: *Grid creation specifications.*

Specification	**Value**
`max_grid_size`	128
`blocking_factor`	8
`n_error_buf`	1
`grid_eff`	0.7

The simulation using AMR is repeated using different φ_{grad}, `blocking_factor` and `n_error_buf` to study the effect of these parameters on the applications of AMR. As recorded in Table 7, the simulation time becomes shorter as the φ_{grad} increases. This can be explained by the decreasing tagged cells for refinement as the φ_{grad} increases hence less computational work is required. From the table, it can also be observed that by increasing the `blocking_factor` increases the simulation time. This is due to the increasing number of cells for refinement as the `blocking_factor` increases. The simulation time also increases with `n_error_buf` due to the same reason previously regarding the number of cells for refinement.

Table 5: *Timing result for 2-D cylindrical test using multiple cores (MPI) or threads (OpenMP) without AMR.*

Without AMR					
Mesh	**AMR**	**MPI/Core**	**OpenMP/Thread**	**Time**	**Speed Up**
256	-	-	-	89.340	1.0
384	-	-	-	289.759	1.0
512	-	-	-	700.550	1.0
MPI test					
Mesh	**AMR**	**MPI/Core**	**OpenMP/Thread**	**Time**	**Speed Up**
256	-	2	-	47.652	1.9
256	-	4	-	26.815	3.3
256	-	8	-	18.376	4.9
256	-	16	-	15.669	5.8
Mesh	**AMR**	**MPI/Core**	**OpenMP/Thread**	**Time**	**Speed Up**
384	-	2	-	163.415	1.8
384	-	4	-	101.431	2.9
384	-	8	-	78.202	3.7
384	-	16	-	50.893	5.7
Mesh	**AMR**	**MPI/Core**	**OpenMP/Thread**	**Time**	**Speed Up**
512	-	2	-	372.663	1.9
512	-	4	-	185.535	3.8
512	-	8	-	109.803	6.4
512	-	16	-	71.633	9.8
OpenMP test					
Mesh	**AMR**	**MPI/Core**	**OpenMP/Thread**	**Time**	**Speed Up**
256	-	-	2	54.187	1.6
256	-	-	4	32.984	2.7
256	-	-	8	33.583	2.7
256	-	-	16	38.907	2.3
Mesh	**AMR**	**MPI/Core**	**OpenMP/Thread**	**Time**	**Speed Up**
384	-	-	2	184.125	1.6
384	-	-	4	123.569	2.3
384	-	-	8	93.649	3.1
384	-	-	16	67.136	4.3
Mesh	**AMR**	**MPI/Core**	**OpenMP/Thread**	**Time**	**Speed Up**
512	-	-	2	404.113	1.7
512	-	-	4	236.495	3.0
512	-	-	8	159.187	4.4
512	-	-	16	122.687	5.7

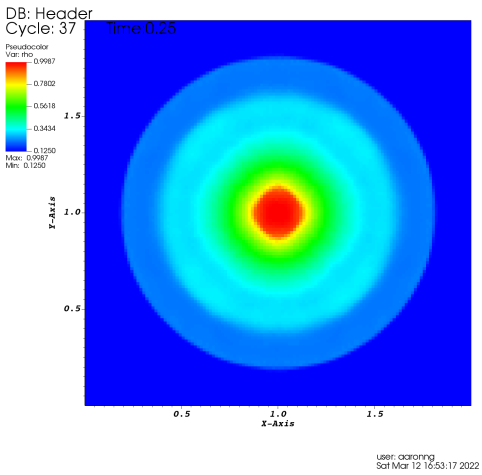
Table 6: Timing result for 2-D cylindrical test using multiple cores (MPI) or threads (OpenMP) with AMR.

With AMR					
Mesh	**AMR**	**MPI/Core**	**OpenMP/Thread**	**Time**	**Speed Up**
512	-	-	-	700.550	1.0
256	x2	-	-	303.255	2.3
256	x2	2	-	144.784	4.8
256	x2	-	2	165.393	4.2
256	x2	4	-	81.801	8.6
256	x2	-	4	108.808	6.4
256	x2	8	-	54.507	12.9
256	x2	-	8	93.956	7.5
256	x2	16	-	42.344	16.5
256	x2	-	16	101.153	6.9

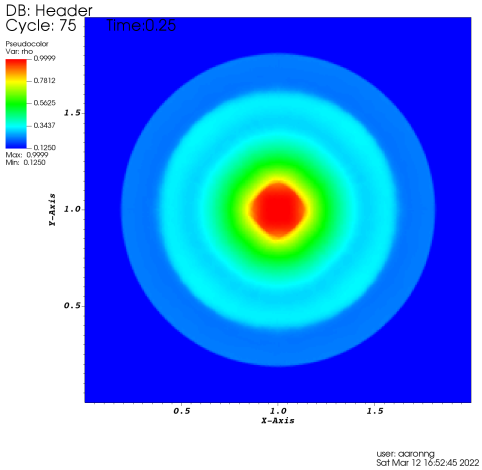
Table 7: Timing result for 2-D cylindrical test with varying AMR specifications

Varying φ_{grad}						
Mesh	**AMR**	**Phigrad**	**MPI/Core**	**OpenMP/Thread**	**Time**	**Speed Up**
256	x2	0.015	-	-	303.255	1.0
256	x2	0.005	-	-	337.741	0.9
256	x2	0.010	-	-	324.182	0.95
256	x2	0.020	-	-	288.371	1.05
256	x2	0.025	-	-	269.880	1.13
Varying blocking factor						
Mesh	**AMR**	**Blocking Factor**	**MPI/Core**	**OpenMP/Thread**	**Time**	**Speed Up**
256	x2	8	-	-	303.255	1.0
256	x2	2	-	-	264.663	1.2
256	x2	4	-	-	281.541	1.1
256	x2	16	-	-	359.257	0.8
Varying number of buffer cells						
Mesh	**AMR**	**Buffer Cell**	**MPI/Core**	**OpenMP/Thread**	**Time**	**Speed Up**
256	x2	1	-	-	303.255	1.0
256	x2	2	-	-	329.633	0.9
256	x2	4	-	-	391.893	0.7
256	x2	8	-	-	464.627	0.6
256	x2	16	-	-	554.859	0.5

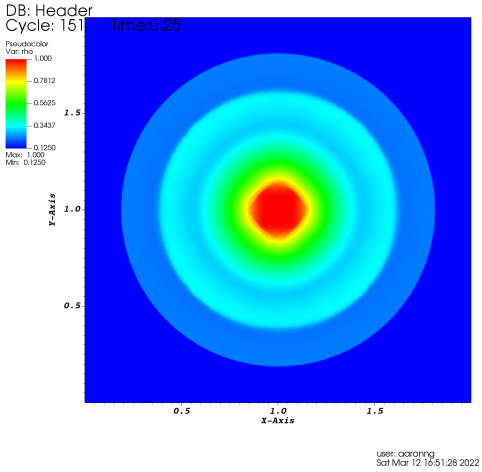
2-D Cylindrical test results



(a) Density plot from 2-D cylindrical test at N=128*128

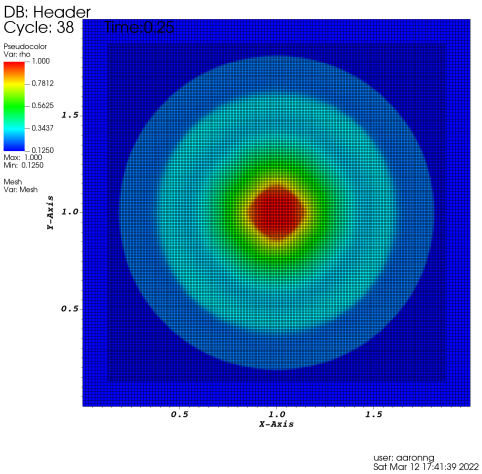


(b) Density plot from 2-D cylindrical test at N=256*256

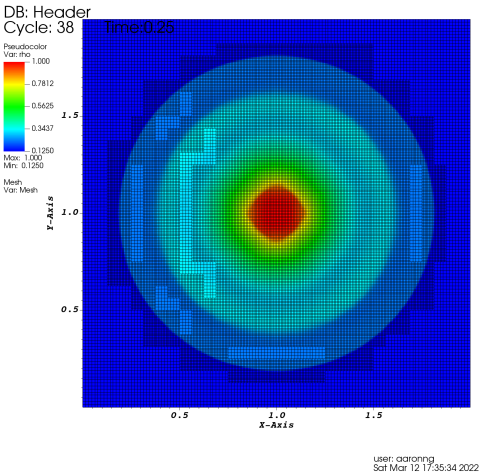


(c) Density plot from 2-D cylindrical test at N=512*512

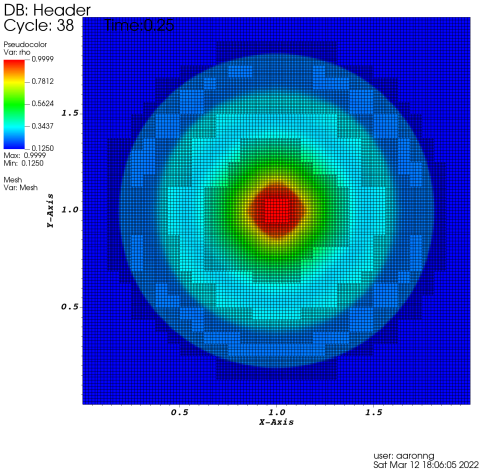
Figure 7: 2-D cylindrical explosion test



(a) Density plot using phigrad = 0.005

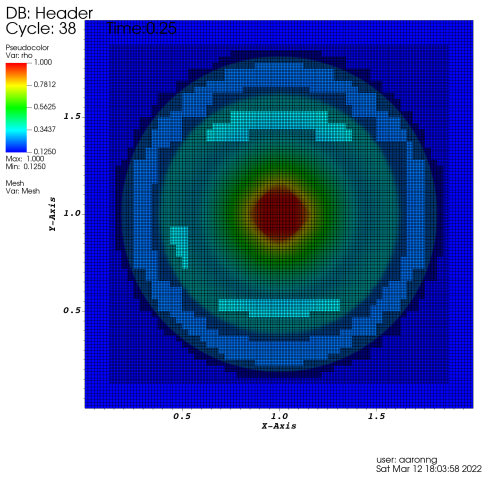


(b) Density plot using phigrad = 0.015

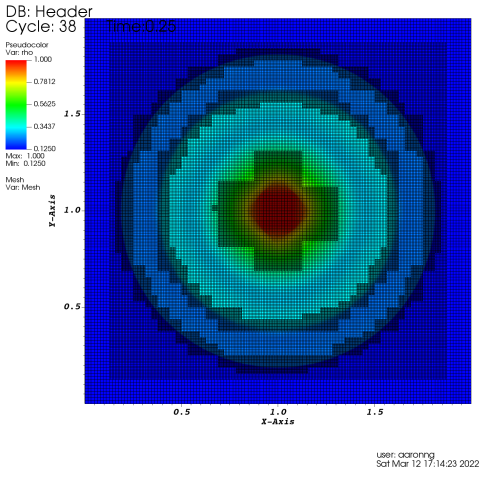


(c) Density plot using phigrad = 0.025

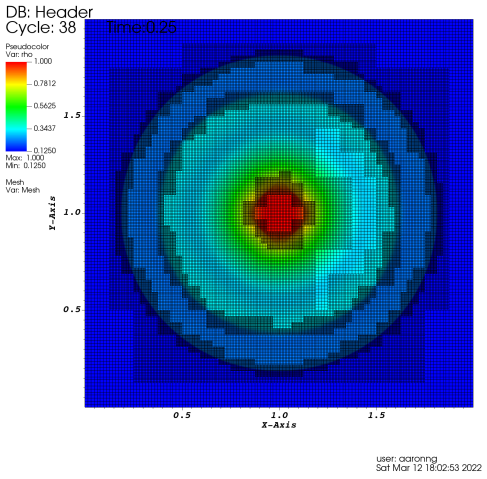
Figure 8: 2-D cylindrical explosion test with AMR x2 refinement



(a) Density plot using phigrad = 0.005



(b) Density plot using phigrad = 0.015



(c) Density plot using phigrad = 0.025

Figure 9: 2-D cylindrical explosion test with AMR x2x2 refinement

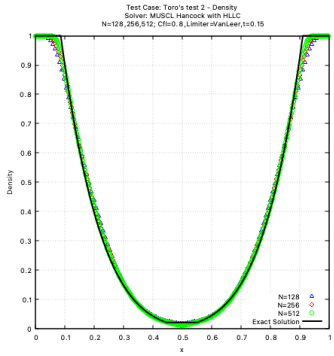
7. Conclusions

In this paper, h-type refinement or AMR has been used to refine the solution of the Euler equations in 1-D and 2-D. The convergence analysis in Table 1 shows that by constraining the refinement to regions containing sharp features, the $L_1 - norms$ is lower as compared to simulations running at the same resolution without any AMR. Although the $L_1 - norms$ for simulations with AMR are still greater than running the same test at high resolutions, the savings in computational cost (2.3 times for the test case of running 2-D cylindrical explosion at N=256*256 with AMR) justify the advantages of using AMR. In addition, parallel computing has been used to speed up the simulations by splitting the work load onto multiple machines. Simulations ran on multiple machines have shown significant uplift in speed especially in the case of using AMR.

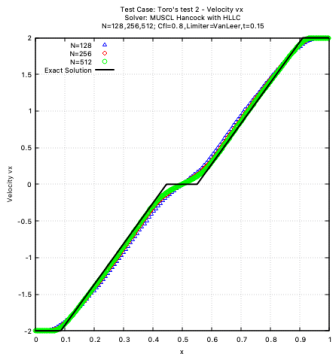
One of the identified future improvement to the performance of AMReX is to run the simulation using a GPU. AMReX uses the native programming language for GPUs: CUDA for NVIDIA, HIP for AMD and DPC++ for Intel. The recommended strategy for AMReX is to use CUDA/HIP/DPC++ together with MPI and Open MP/Open ACC to control the offloading of subroutines to the GPU. [9] Furthermore, it is worth noting that for most of the test cases, same value has been used for the threshold of tagging cells for refinement at all the refinement levels. The threshold value can be more specific to the regions of interest and further savings on computational cost can be achieved as refinement will only be focused on required area. Finally, there are other AmrCore parameters which can be tailored to specific applications such as the `grid_eff`, `max_grid_size` and refinement ratio.

In summary, AMR is recommended to focus computational resources in area of interests such as shock wave, material interaction and chemical reactions. Apart from the improvement suggested above, AMR should be explored with other numerical methods such as the centred schemes. Simulations of other highly complex physical problems such as Magnetohydrodynamics (MHD) using AMR is also an area of interest given the growing need of more sophisticated plasma simulation.

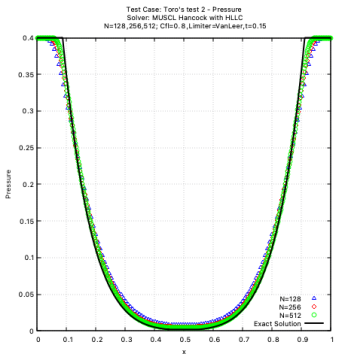
Appendix A. 1-D Toro's test 2-5



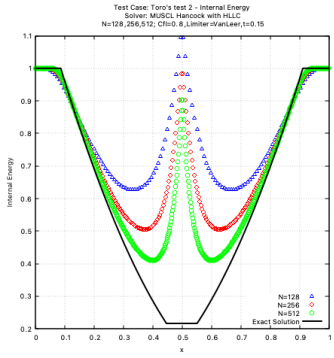
(a) Density plot



(b) Velocity vx plot

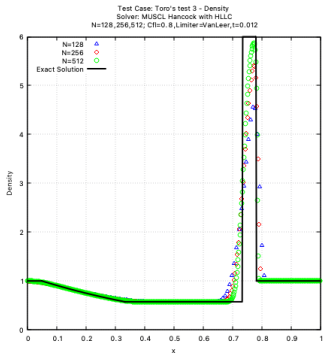


(c) Pressure plot

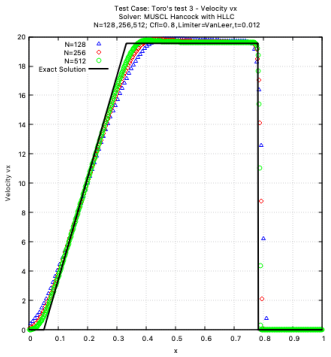


(d) Specific Internal Energy plot

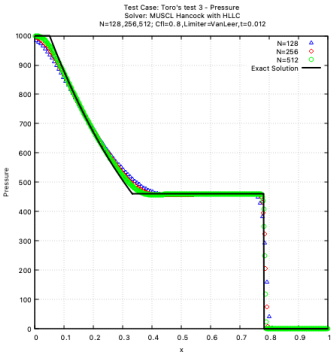
Figure A.10: Toro's test 2



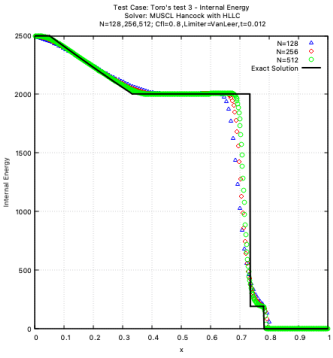
(a) Density plot



(b) Velocity vx

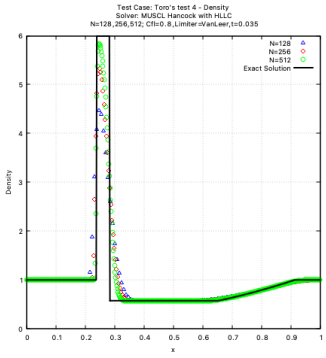


(c) Pressure plot

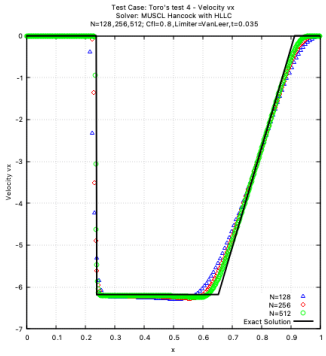


(d) Specific Internal Energy plot

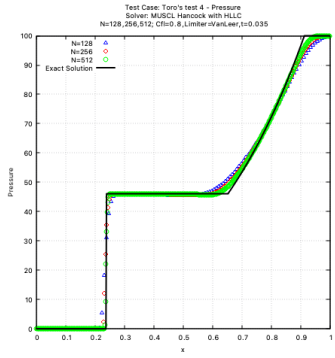
Figure A.11: Toro's test 3



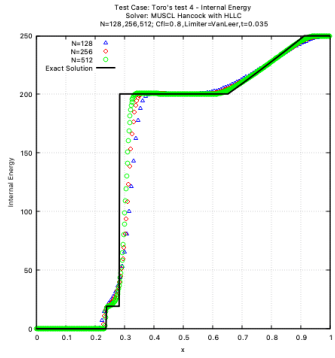
(a) Density plot



(b) Velocity vx plot

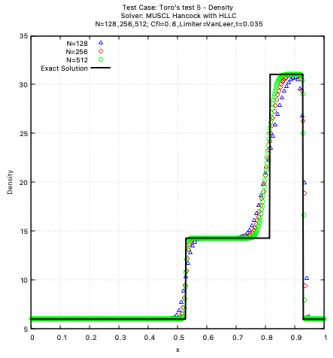


(c) Pressure plot

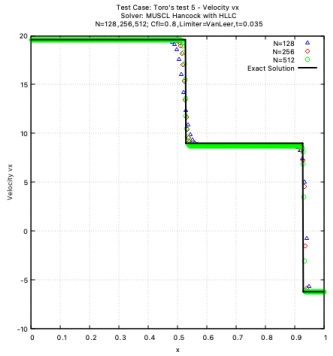


(d) Specific Internal Energy plot from Toro's test 4

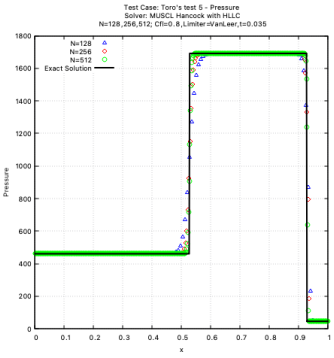
Figure A.12: Toro's test 4



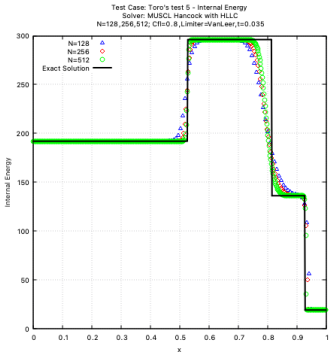
(a) Density plot



(b) Velocity vx plot



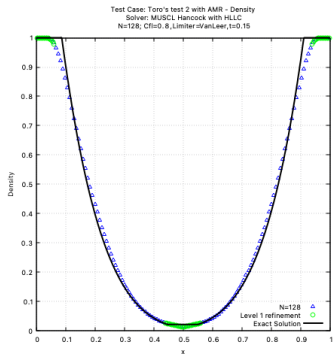
(c) Pressure plot



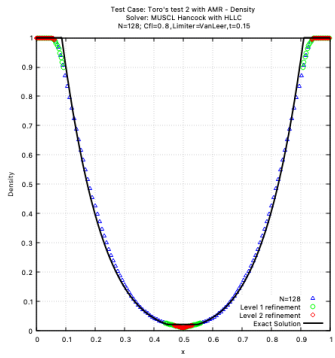
(d) Specific Internal Energy plot

Figure A.13: Toro's test 5

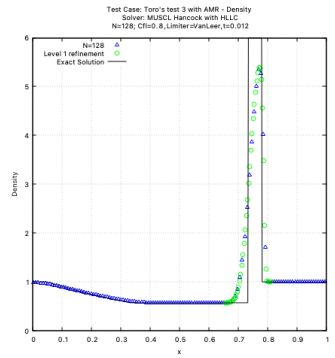
Appendix B. 1-D Toro's test 2-5 with AMR



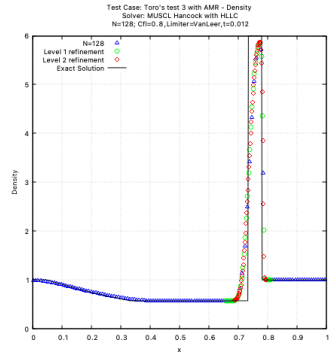
(a) Density plot from Toro's test 2 with AMR Level 1



(b) Density plot from Toro's test 2 with AMR Level 2

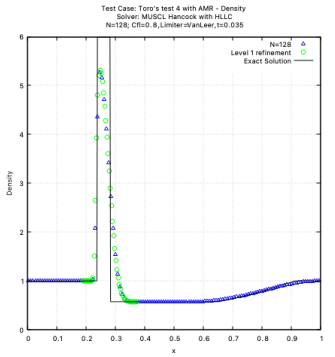


(c) Density plot from Toro's test 3 with AMR Level 1

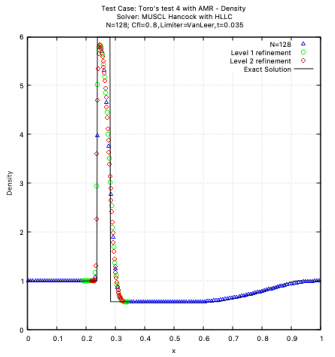


(d) Density plot from Toro's test 3 with AMR Level 2

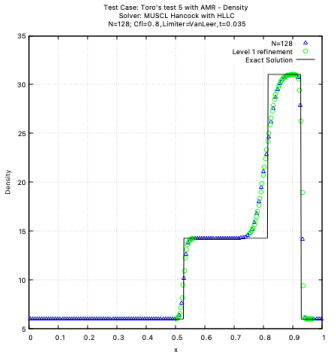
Figure B.14: Toro's test 2 and 3 with AMR



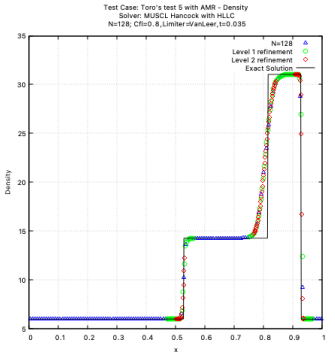
(a) Density plot from Toro's test 4 with AMR Level 1



(b) Density plot from Toro's test 4 with AMR Level 2



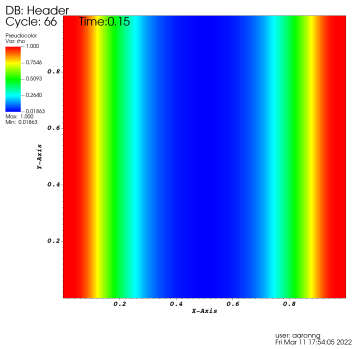
(c) Density plot from Toro's test 5 with AMR Level 1



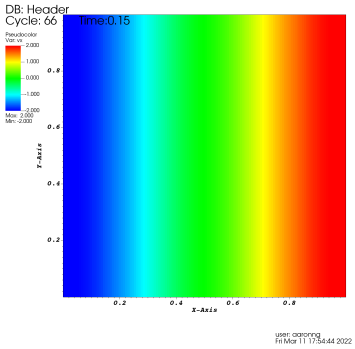
(d) Density plot from Toro's test 5 with AMR Level 2

Figure B.15: Toro's test 4 and 5 with AMR

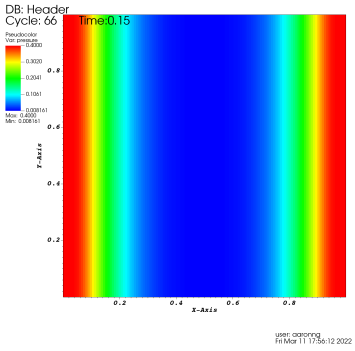
Appendix C. 2-D Toro's test 2



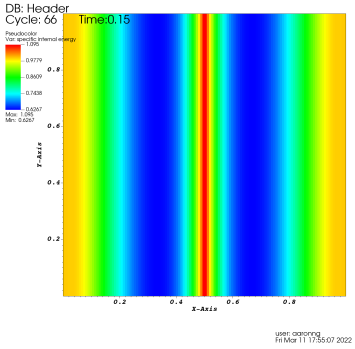
(a) Density plot



(b) Velocity vx plot

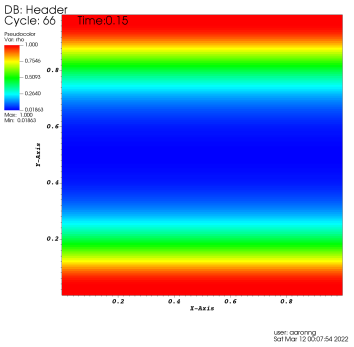


(c) Pressure plot

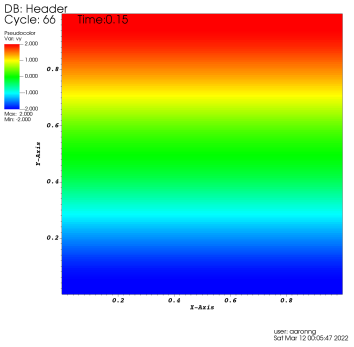


(d) Specific Internal Energy plot

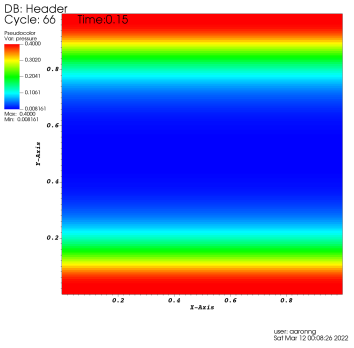
Figure C.16: 2-D Toro's test 2 with x-split Initial Condition (IC)



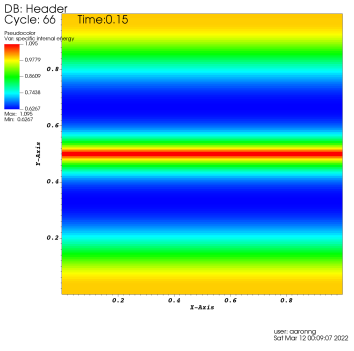
(a) Density plot



(b) Velocity vy plot

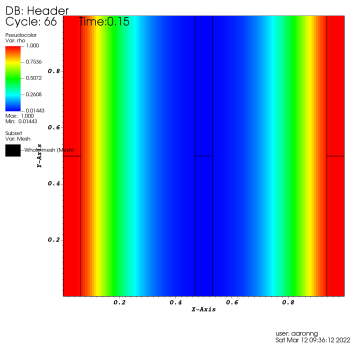


(c) Pressure plot

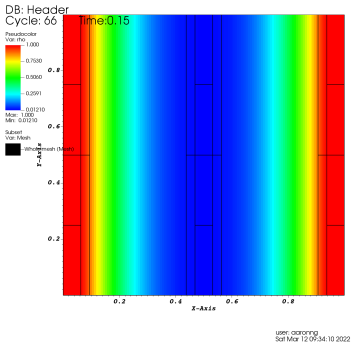


(d) Specific Internal Energy plot

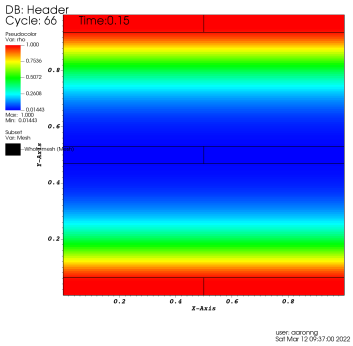
Figure C.17: 2-D Toro's test 2 with y-split Initial Condition (IC)



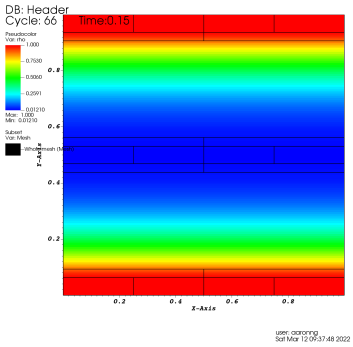
(a) Density plot from x-split IC with AMR x2 refinement



(b) Density plot from x-split IC with AMR x2x2 refinement

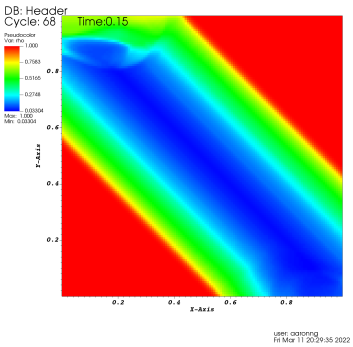


(c) Density plot from y-split IC with AMR x2 refinement

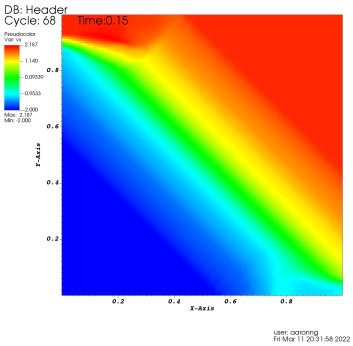


(d) Density plot from y-split IC with AMR x2x2 refinement

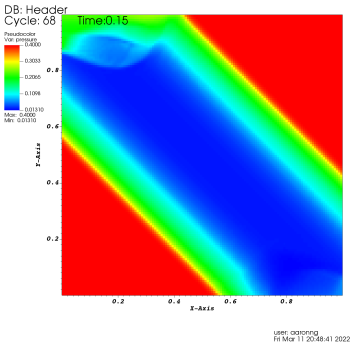
Figure C.18: 2-D Toro's test 2 with AMR



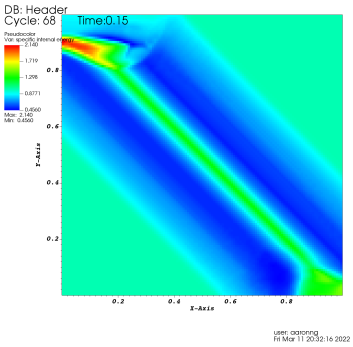
(a) Density plot



(b) Velocity vx plot



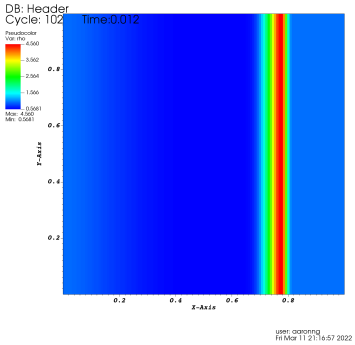
(c) Pressure plot



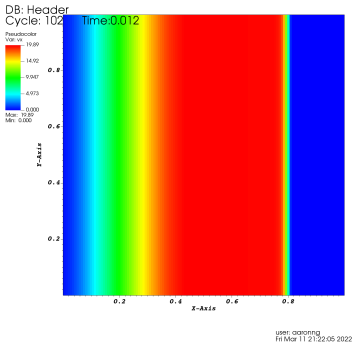
(d) Specific Internal Energy plot

Figure C.19: 2-D Toro's test 2 with diagonally aligned Initial Condition (IC)

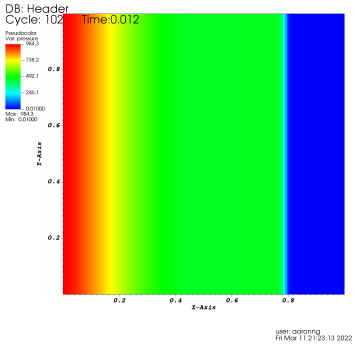
Appendix D. 2-D Toro's test 3



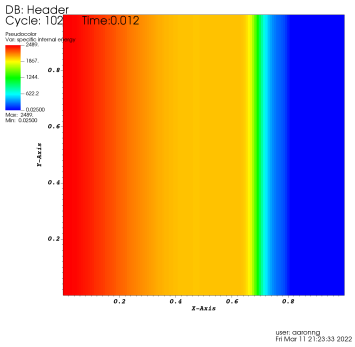
(a) Density plot



(b) Velocity vx plot

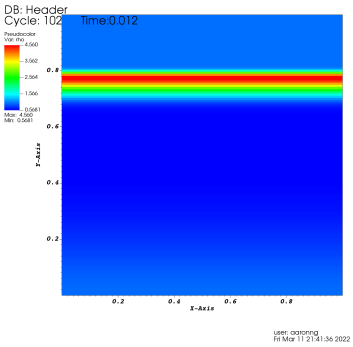


(c) Pressure plot

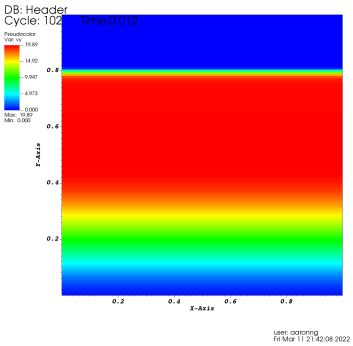


(d) Specific Internal Energy plot

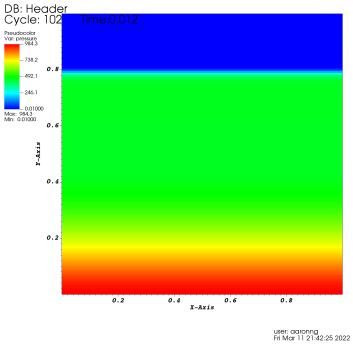
Figure D.20: 2-D Toro's test 3 with x-split Initial Condition (IC)



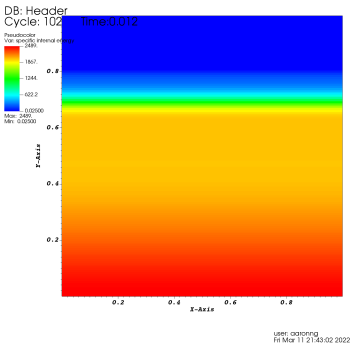
(a) Density plot



(b) Velocity vy plot

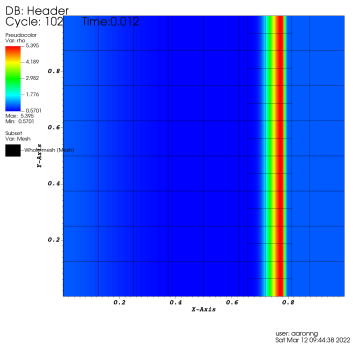


(c) Pressure plot

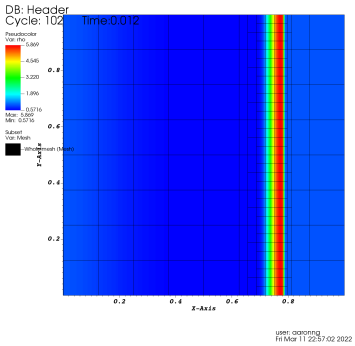


(d) Specific Internal Energy plot

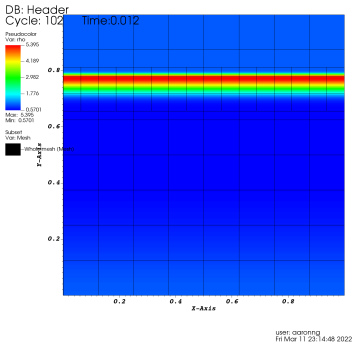
Figure D.21: 2-D Toro's test 3 with y-split Initial Condition (IC)



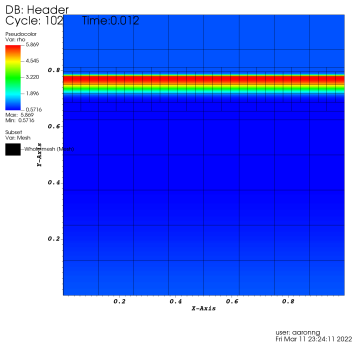
(a) Density plot from x-split with AMR x2 refinement



(b) Density plot from x-split with AMR x2x2 refinement

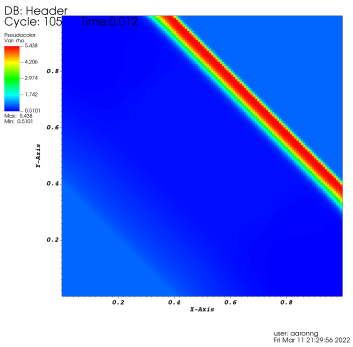


(c) Density plot from y-split with AMR x2 refinement

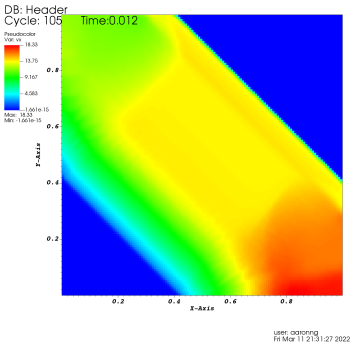


(d) Density plot from y-split with AMR x2x2 refinement

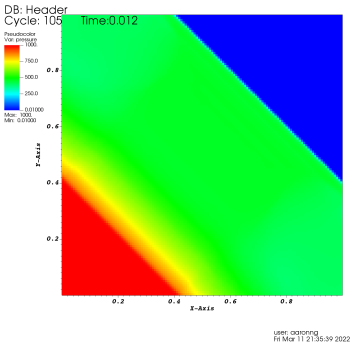
Figure D.22: 2-D Toro's test 3 with AMR



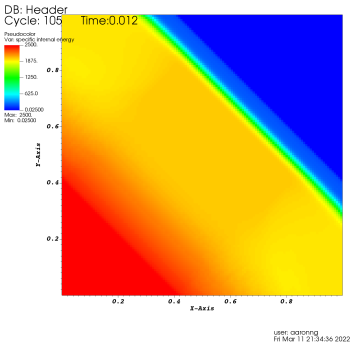
(a) Density plot



(b) Velocity vx plot



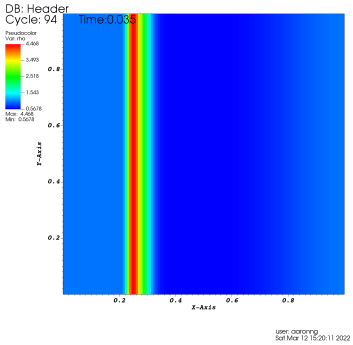
(c) Pressure plot



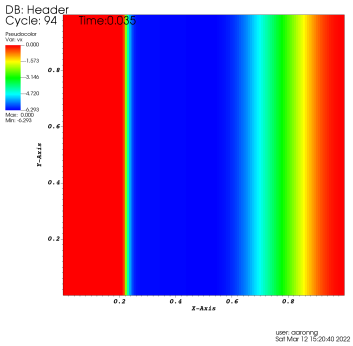
(d) Specific Internal Energy plot

Figure D.23: 2-D Toro's test 3 with diagonally aligned Initial Condition (IC)

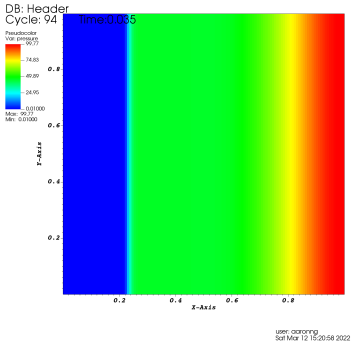
Appendix E. 2-D Toro's test 4



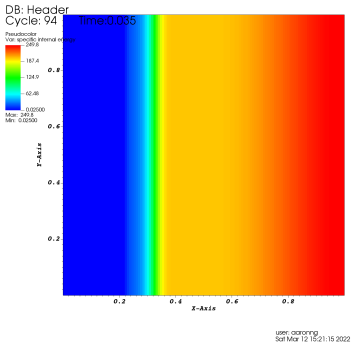
(a) Density plot



(b) Velocity vx plot

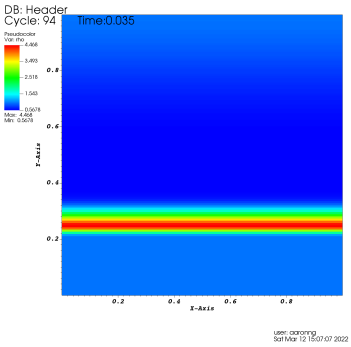


(c) Pressure plot

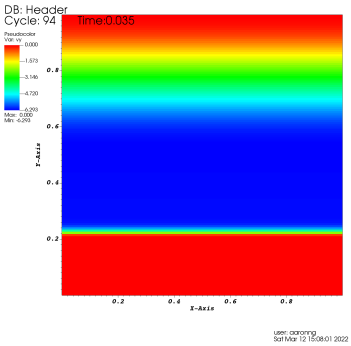


(d) Specific Internal Energy plot

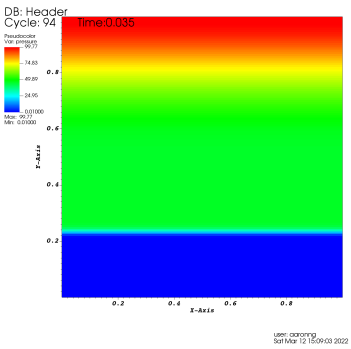
Figure E.24: 2-D Toro's test 4 with x-split Initial Condition (IC)



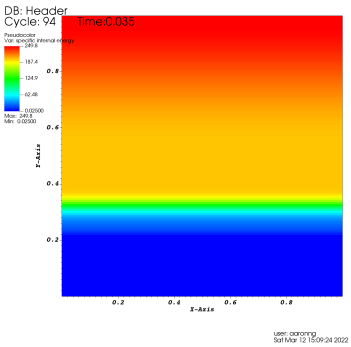
(a) Density plot



(b) Velocity vy plot

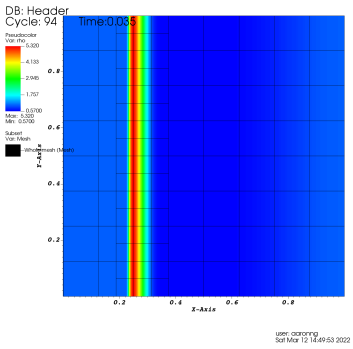


(c) Pressure plot

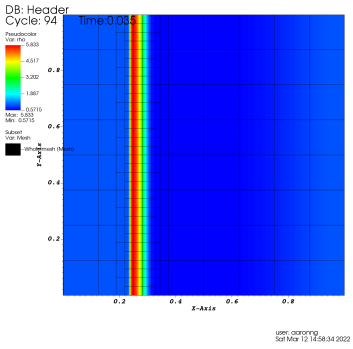


(d) Specific Internal Energy plot

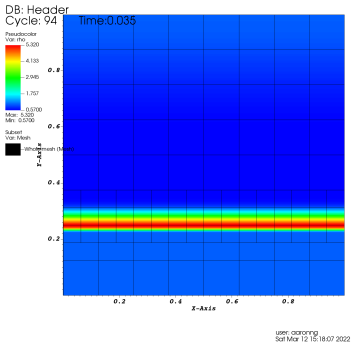
Figure E.25: 2-D Toro's test 4 with y-split Initial Condition (IC)



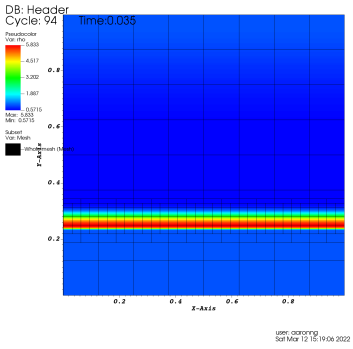
(a) Density plot from x-split IC with AMR x2 refinement



(b) Density plot from x-split IC with AMR x2x2 refinement

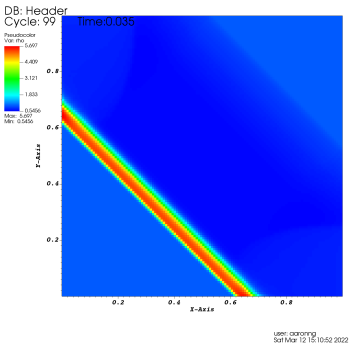


(c) Density plot from y-split IC with AMR x2 refinement

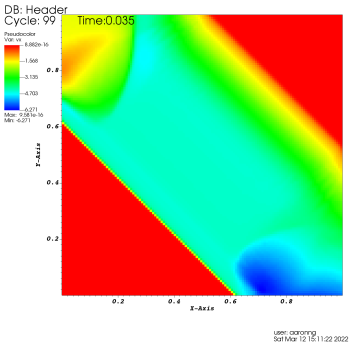


(d) Density plot from y-split IC with AMR x2x2 refinement

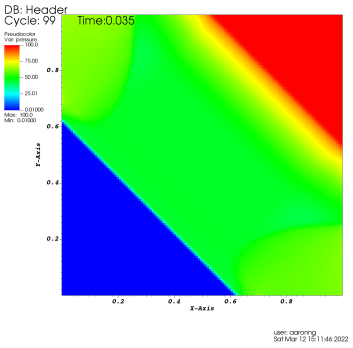
Figure E.26: 2-D Toro's test 4 with AMR



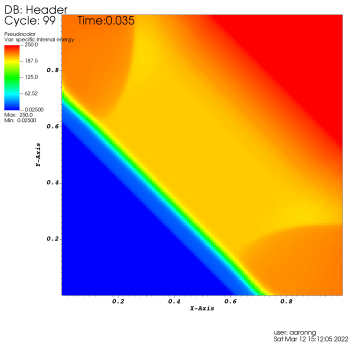
(a) Density plot



(b) Velocity vx plot



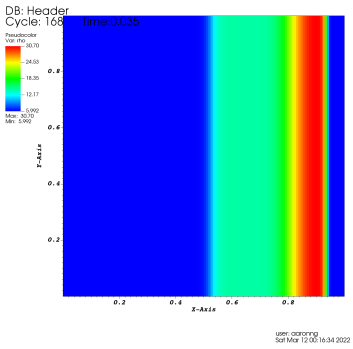
(c) Pressure plot



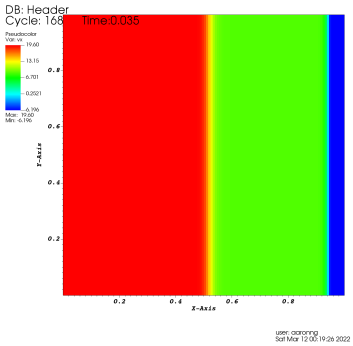
(d) Specific Internal Energy plot

Figure E.27: 2-D Toro's test 4 with diagonally aligned Initial Condition (IC)

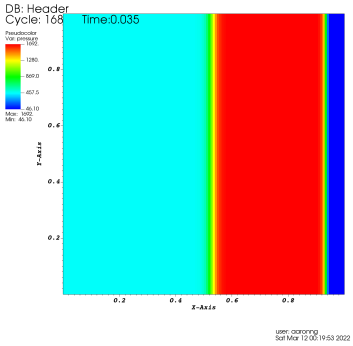
Appendix F. 2-D Toro's test 5



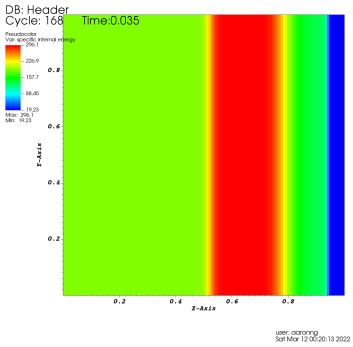
(a) Density plot



(b) Velocity vx plot

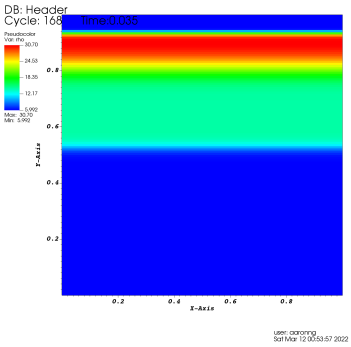


(c) Pressure plot

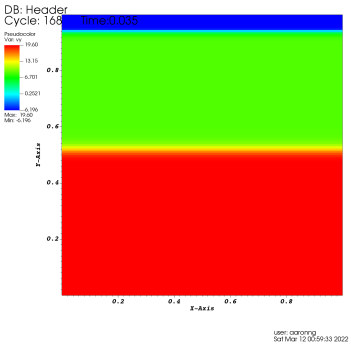


(d) Specific Internal Energy plot

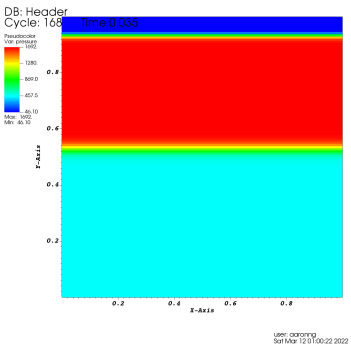
Figure F.28: 2-D Toro's test 5 with x-split Initial Condition (IC)



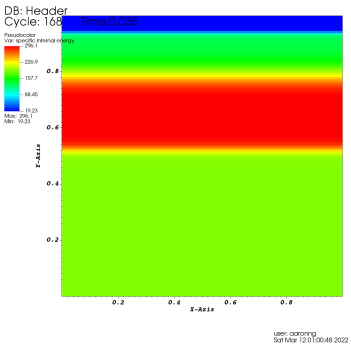
(a) Density plot



(b) Velocity vy plot

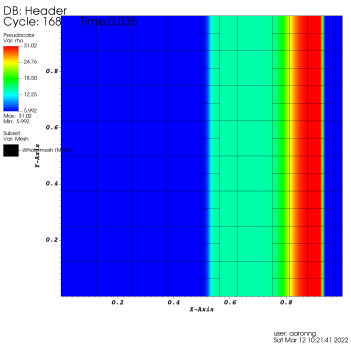


(c) Pressure plot

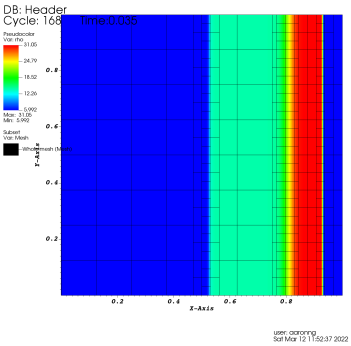


(d) Specific Internal Energy plot

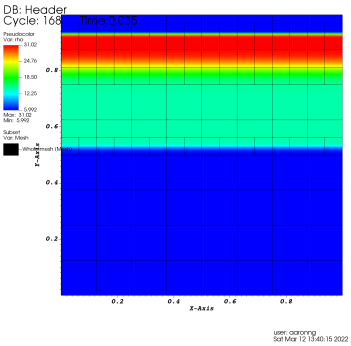
Figure F.29: 2-D Toro's test 5 with y-split Initial Condition (IC)



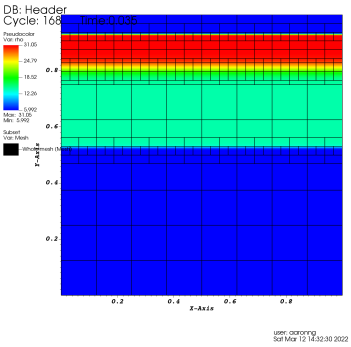
(a) Density plot from x-split IC with AMR x2 refinement



(b) Density plot from x-split IC with AMR x2x2 refinement

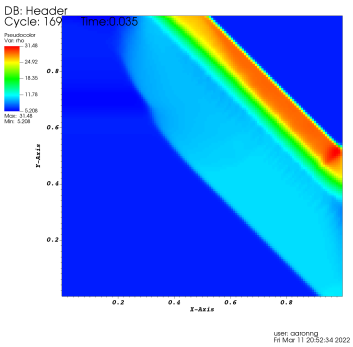


(c) Density plot from y-split IC with AMR x2 refinement

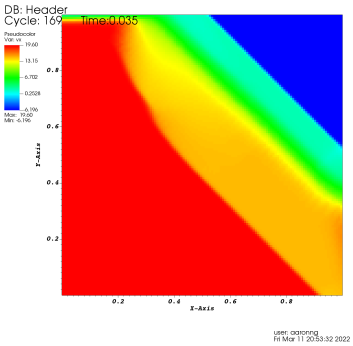


(d) Density plot from y-split IC with AMR x2x2 refinement

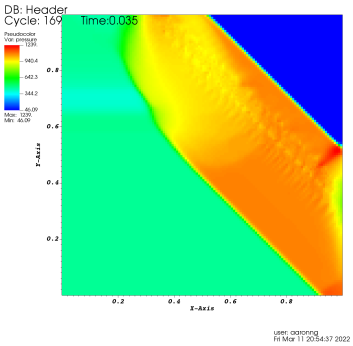
Figure F.30: 2-D Toro's test 5 with AMR



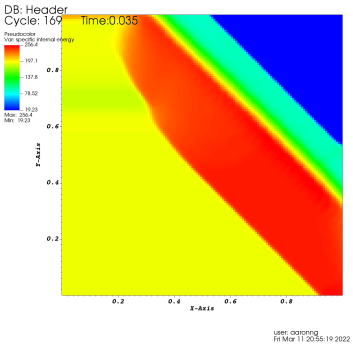
(a) Density plot



(b) Velocity vx plot



(c) Pressure plot



(d) Specific Internal Energy plot

Figure F.31: 2-D Toro's test 5 with diagonally aligned Initial Condition (IC)

References

[1] Marsha J Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512, 1984.

[2] Philip Blakely. Lecture notes in continuum modelling-amr, January 2022.

[3] Ralf Deiterding. Block-structured adaptive mesh refinement - theory, implementation and application. *ESAIM: Proc.*, 34:97–150, 2011.

[4] Jared Ferguson, Christiane Jablonowski, Hans Johansen, Peter McCorquodale, Phillip Colella, and Paul Ullrich. Analyzing the adaptive mesh refinement (amr) characteristics of a high-order 2d cubed-sphere shallow-water model. *Monthly Weather Review*, 144, 09 2016.

[5] Matthew Hubbard and Nikolaos Nikiforakis. A three-dimensional, adaptive, godunov-type model for global atmospheric flows. *Monthly Weather Review - MON WEATHER REV*, 131, 08 2003.

[6] George Karniadakis and Spencer Sherwin. *Spectral/HP Element Methods for Computational Fluid Dynamics*. 06 2005.

[7] Stephen Millmore, Louisa Michael, and Nikos Nikiforakis. Lecture notes in computational continuum modelling, October 2021.

[8] MJ Rutter. Lecture notes in mpi course, November 2021.

[9] AMReX Team. amrex documentation, February 2022.

[10] Eleuterio Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. 01 2009.

[11] Eleuterio Toro. Lecture notes in advanced computational algorithm for pdes - ader, November 2021.

[12] Eleuterio Toro. Lecture notes in advanced computational algorithm for pdes - hllc, November 2021.

[13] Weiqun Zhang, Andrew Myers, Kevin Gott, Ann Almgren, and John Bell. Amrex: Block-structured adaptive mesh refinement for multiphysics applications. 09 2020.