Fluid-Structure Interaction Using the Ghost Fluid Method

BCN: 8E0Q7

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

Abstract

This study investigates the use of Ghost Fluid Method (GFM) based methods for simulating fluid-structure interactions in two-dimensional compressible flows. The Euler equations are solved using high-resolution numerical schemes, with the level set method tracking the rigid body interface and a modified Riemann GFM enforcing dynamic boundary conditions. A series of numerical experiments are conducted to evaluate the performance of the proposed approach, covering shock wave interactions with rigid geometries, moving rigid bodies in initially static fluids, and time-varying velocity effects. The results demonstrate the method's effectiveness in accurately capturing key flow phenomena such as shock reflections, wave propagation, wake formation, and vortex dynamics.

1. Introduction

In this section, we introduce the two-dimensional compressible Euler equations [1] and high-resolution numerical methods for solving these equations.

1.1. Euler Equations

The compressible Euler equations in conservation form are given below:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) &= 0, \\ \frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v} + \mathbf{I}p) &= 0, \\ \frac{\partial E}{\partial t} + \nabla \cdot [(E + p)\mathbf{v}] &= 0, \end{aligned} \tag{1}$$

which comprise three conservation laws for density, ρ, momentum, $\rho\mathbf{v}$, and total energy, E. For the compressible Euler equations, the total energy can be expressed as the sum of internal energy and kinetic energy:

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

where ε is the specific internal energy, representing energy per unit mass. Due to the fourth variable, pressure, p, an equation of state is needed to close the system. For an ideal gas, the equation of state can be written as:

$$p = (\gamma - 1)\rho\varepsilon, \tag{3}$$

where γ, the adiabatic index, is a constant. For air and other diatomic molecular gases, $\gamma = 1.4$.

In two dimensions, when a Cartesian representation is considered, the Euler equations contain two velocities, in the x- and y-directions, respectively. Furthermore, both x- and y-direction evolutions need to be considered. The system of equations can be written as follows:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho v_x) + \frac{\partial}{\partial y}(\rho v_y) &= 0, \\ \frac{\partial}{\partial t}(\rho v_x) + \frac{\partial}{\partial x}(\rho v_x^2 + p) + \frac{\partial}{\partial y}(\rho v_x v_y) &= 0, \\ \frac{\partial}{\partial t}(\rho v_y) + \frac{\partial}{\partial x}(\rho v_x v_y) + \frac{\partial}{\partial y}(\rho v_y^2 + p) &= 0, \\ \frac{\partial E}{\partial t} + \frac{\partial}{\partial x}[(E + p)\, v_x] + \frac{\partial}{\partial y}[(E + p)\, v_y] &= 0, \end{aligned} \tag{4}$$

where v_x and v_y are velocities in x- and y-directions. Therefore, the vector form of the two-dimensional equations is given by:

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial}{\partial x}\mathbf{f}(\mathbf{u}) + \frac{\partial}{\partial y}\mathbf{g}(\mathbf{u}) = 0, \tag{5}$$

where $\mathbf{u}$ is the vector of conserved variables, $\mathbf{f}(\mathbf{u})$ and $\mathbf{g}(\mathbf{u})$ are the flux functions in x- and y-directions.

1.2. Numerical Methods

This part discusses the high-resolution numerical methods used to solve the Euler equations. The numerical methods are first described in one dimension, then the techniques required for their use in two-dimensional cases are introduced.

1.2.1. Finite Volume Method

Given that equation (5) is in continuous form, discretization should be performed to solve the Euler equations with numerical methods. By applying the finite volume method [2], we can discretize the simulation domain into regular cells with size $[\Delta x, \Delta y]$ and derive an update function for the conserved variables from the integral form of conservation law, as shown below:

$$\begin{aligned}\mathbf{u}_{i,j}^{n+1} = \mathbf{u}_{i,j}^{n} - \frac{\Delta t}{\Delta x}\left(\mathbf{f}_{i+1/2,j}^{n} - \mathbf{f}_{i-1/2,j}^{n}\right) \\ -\frac{\Delta t}{\Delta y}\left(\mathbf{g}_{i,j+1/2}^{n} - \mathbf{g}_{i,j-1/2}^{n}\right),\end{aligned} \tag{6}$$

where Δt is the time step, $\mathbf{u}_{i,j}^{n+1}$ and $\mathbf{u}_{i,j}^{n}$ are the variables at the cell $[i, j]$ at time $n+1$ and n, respectively. For the numerical fluxes, $\mathbf{f}_{i-1/2,j}^{n}$ and $\mathbf{f}_{i+1/2,j}^{n}$ are the x-direction fluxes at left and right cell boundaries at time n, while $\mathbf{g}_{i,j-1/2}^{n}$ and $\mathbf{g}_{i,j+1/2}^{n}$ are the y-direction fluxes at lower and upper cell boundaries. The calculation of numerical fluxes varies across different schemes.

1.2.2. 2D MUSCL-Hancock Scheme

The Monotone Upstream-centered Scheme for Conservation Laws (MUSCL) [3] is a high-resolution numerical method. The performance of this method follows three steps: slope-limiting data reconstruction, half-time step evolution, and a first-order update scheme. The details of the MUSCL-Hancock scheme are described in the appendix A.

To use MUSCL-Hancock in two-dimensional cases, data reconstruction and half-time step evolution just need to be performed twice, one in each direction. However, for the first-order update scheme, the constraint on the time step should be dealt with carefully. Recall the finite volume update function (6), which is used for the evolution of conserved variables in both centered and Riemann problem-based schemes, the computation of fluxes in either direction is treated as entirely one-dimensional. This means that Δx is used for the x-direction and Δy is used for the y-direction. Similarly, when solving the Riemann problem, when we compute a solution in the x-direction, all of the velocity terms are v_x, while in the y-direction, they are always v_y.

In this definition, the same time step is used for both directions. To keep the numerical updates stable,

$$\begin{aligned}\Delta t = C\frac{\min(\Delta x, \Delta y)}{a_{\max}}, \\ a_{\max} = \max_{i,j}\left(|\mathbf{v}_{i,j}| + c_{s,i,j}\right),\end{aligned} \tag{7}$$

where C is the CFL number. Since the total velocity could be diagonal in two-dimensional cases, we also need to halve the CFL number to keep numerical stability, which is not desirable. Therefore, the dimensional splitting approach [4] is proposed to solve this problem. This method updates the x-direction first, and then updates the y-direction using the results from the x-update, as shown below:

$$\begin{aligned}\bar{\mathbf{u}}_{i,j} = \mathbf{u}_{i,j}^{n} + \frac{\Delta t}{\Delta x}\left(\mathbf{f}_{i-1/2,j}^{n}(\mathbf{u}) - \mathbf{f}_{i+1/2,j}^{n}(\mathbf{u})\right), \\ \mathbf{u}_{i,j}^{n+1} = \bar{\mathbf{u}}_{i,j} + \frac{\Delta t}{\Delta y}\left(\mathbf{g}_{i,j-1/2}^{n}(\bar{\mathbf{u}}) - \mathbf{g}_{i,j+1/2}^{n}(\bar{\mathbf{u}})\right).\end{aligned} \tag{8}$$

The dimensional splitting method not only allows diagonal movement of information over a single time step but also keeps the CFL number the same as that for the one-dimensional fluxes. It can also be represented with operator notations:

$$\mathbf{u}_{i,j}^{n+1} = \mathcal{Y}^{(\Delta t)}\mathcal{X}^{(\Delta t)}(\mathbf{u}^{n}). \tag{9}$$

Equations (33) and (34) are first-order accurate in time. To obtain second-order temporal accuracy, the Strang splitting techniques [5] can be applied:

$$\begin{aligned}\mathbf{u}_{i,j}^{n+1} = \frac{1}{2}\left[\mathcal{Y}^{(\Delta t)}\mathcal{X}^{(\Delta t)} + \mathcal{X}^{(\Delta t)}\mathcal{Y}^{(\Delta t)}\right](\mathbf{u}^{n}), \\ \mathbf{u}_{i,j}^{n+1} = \mathcal{X}^{(\frac{1}{2}\Delta t)}\mathcal{Y}^{(\Delta t)}\mathcal{X}^{(\frac{1}{2}\Delta t)}(\mathbf{u}^{n}), \\ \mathbf{u}_{i,j}^{n+2} = \frac{1}{2}\left[\mathcal{Y}^{(\Delta t)}\mathcal{X}^{(\Delta t)}\mathcal{X}^{(\Delta t)}\mathcal{Y}^{(\Delta t)}\right](\mathbf{u}^{n}).\end{aligned} \tag{10}$$

Provided that the calculations of individual numerical updates are second-order accurate in time, these schemes are also second-order accurate in time.

2. Methodology

This section discusses how to simulate fluid interaction with rigid bodies. The fluid is modeled with the Euler equations and the numerical methods introduced in Section 1 can still be used. However, since the rigid body is a different material from the fluid, the level set method is applied to locate the interface, and ghost fluid method-like techniques are implemented to determine the dynamic boundary conditions.

2.1. Level Set Method

The level set methods [6, 7] are a common technique to track an implicit interface. The level set refers to a set of points which all take the same value. In mathematics, this can be represented with a level set function, ϕ. The interface is often conveniently set as the zero-contour of the level set function, $\phi(\mathbf{x}) = 0$. As far as only fluid-rigid

body interaction is considered, the interface motion can be simply realized by recalculating the level set functions in each iteration, rather than updating the level set. In addition, since the level set functions are not evolved, reinitialization is also unnecessary.

The normal vector and curvature of the level set functions are easy to compute:

$$\hat{\mathbf{n}} = \frac{\nabla\phi}{|\nabla\phi|}, \quad \kappa = \nabla\cdot\hat{\mathbf{n}} = \nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right). \tag{11}$$

Although both of them are computed at cell centers, given the level set function is smooth, they are accurate when calculated enough close to the interface. Another important feature of the level set function is that it is a signed distance function (SDF). As an SDF, the magnitude of the level set function is the shortest distance to the interface in the normal direction. Furthermore, an SDF has $|\nabla\phi| = 1$ everywhere.

In this study, the level set function is used to locate the interface between the rigid body and fluid. This means that the zero-contour of the level set function is the boundary of the rigid body. Given the current center position and shape of the rigid body, the level set function can be calculated in each iteration. By using the level set function, we can distinguish between the real material region and ghost fluid region for the fluid:

$$\left.\begin{array}{ll} \phi(x,y,t) < 0, & \text{ghost fluid region,} \\ \phi(x,y,t) = 0, & \text{on the interface,} \\ \phi(x,y,t) > 0, & \text{real material region,} \end{array}\right\} \tag{12}$$

where the level set function inside the rigid body is defined as negative, whilst that outside is positive.

2.2. Ghost Fluid Method

Ghost Fluid Methods (GFM) are a class of numerical techniques that provide dynamic boundary conditions at the interfaces between different materials. Although level set functions tell us which cells are directly adjacent to the interface, they do not influence the evolution at all. Therefore, we need GFM to determine the boundary conditions at these interface cells and extrapolate the interface states into the ghost fluid region.

The original GFM [8] was the first in this class to be able to work for more than a few specific materials and test cases. Assuming constant entropy across the interface, this method establishes a thermodynamically consistent boundary condition by copying velocity and pressure from the real material to the ghost fluid. However, one of the key weaknesses of the original GFM is that it cannot deal with strong shock waves, which break the constant entropy assumption.

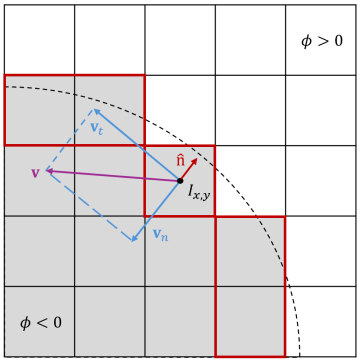

Figure 1: An illustration of interface cells and velocity decomposition

Within about 10 years after the publication of the original GFM, plenty of new methods were proposed on its basis, dealing with the shortages and improving the performance. The method used in this study is a Riemann problem-based GFM, which uses the star states of the mixed-material Riemann problem as the dynamic boundary conditions [9].

Before solving Riemann problems along the interface, we should first identify the cells adjacent to the interface by comparing the signs of the level set functions in neighboring cells. As shown in Figure 1, assuming that we are calculating the ghost fluid region for the white material with the positive level set, which is also referred to as the right material in the Riemann problem, the interface cells in the ghost fluid region are marked with red frames.

As mentioned in Section 1, when solving a Riemann problem in two-dimensional space, only the normal velocity is needed for the Riemann problem solver, whilst the tangential velocity only jumps across the contact discontinuity (shear wave). Therefore, after identifying an interface cell, the next step is to determine the normal vector and normal velocity. Given the level set function, normal vectors can be directly obtained through equation (11) with a centered difference scheme, and the normal velocity can be calculated as follows:

$$\mathbf{v}_n = (\hat{\mathbf{n}}\cdot\mathbf{v})\hat{\mathbf{n}}. \tag{13}$$

The tangential velocity is now straightforward:

$$\mathbf{v}_t = \mathbf{v} - v_n\hat{\mathbf{n}}. \tag{14}$$

In Figure 1, for an interface cell $I_{x,y}$, the velocity and its normal and tangential components are illustrated.

The method to select suitable initial left and right states for the Riemann problem varies between different

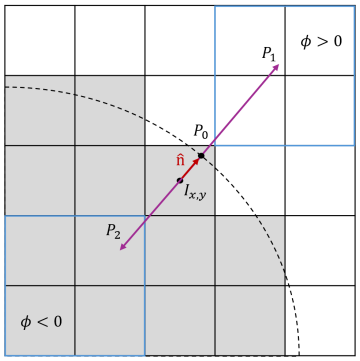

Figure 2: An illustration of interpolating Riemann problem initial states in Riemann GFM

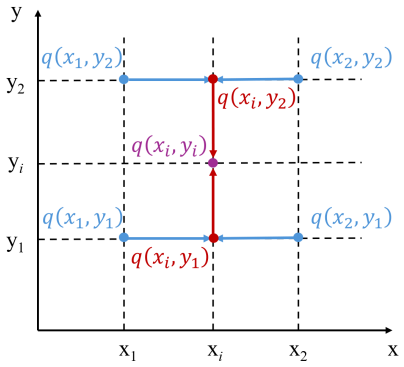

Figure 3: An illustration of bilinear interpolation of a function $q(x, y)$ in a Cartesian frame

Riemann problem-based GFMs. The method used in this study is the Riemann GFM, which allows information to come from any cell suitably close to the interface. In specific, the initial states of the Riemann problem are interpolated based on the information around the interface. The procedure is shown in Figure 2. Firstly, for a given interface cell $I_{x,y}$, we need to find a point on the interface which is closest to $I_{x,y}$. By taking advantage of the SDF feature of the level set function, we can directly realize this by using the normal vector:

$$\mathbf{x}_{P_0} = \mathbf{x}_{I_{x,y}} - \phi\hat{\mathbf{n}}. \tag{15}$$

The identified interface point is then used to interpolate two positions equidistant from the interface along the same normal vector:

$$\mathbf{x}_{P_1,P_2} = \mathbf{x}_{P_0} \pm \delta\hat{\mathbf{n}}, \tag{16}$$

where δ is the interpolation distance, for which $1.5\Delta x$ is usually used as a starting point. To obtain the initial states, for each one of the points P_1 and P_2, four surrounding cells are identified, as shown in Figure 2 with blue frames. The variables in these cells are used to perform bilinear interpolation:

$$\begin{aligned}
q(x_i, y_1) &\approx \frac{x_2 - x_i}{x_2 - x_1} q(x_1, y_1) + \frac{x_i - x_1}{x_2 - x_1} q(x_2, y_1), \\
q(x_i, y_2) &\approx \frac{x_2 - x_i}{x_2 - x_1} q(x_1, y_2) + \frac{x_i - x_1}{x_2 - x_1} q(x_2, y_2), \\
q(x_i, y_i) &\approx \frac{y_2 - y_i}{y_2 - y_1} q(x_i, y_1) + \frac{y_i - y_1}{y_2 - y_1} q(x_i, y_2).
\end{aligned} \tag{17}$$

The procedure of bilinear interpolation is visualized in Figure 3, where the interpolated function $q(x, y)$ could be any variable in the four surrounding cells, either in conservation or primitive form. Notice that the four surrounding cells should be in the same material, if this is not the case, the interpolation distance δ could be adjusted to fix the problem. If this does not work, real GFM could be used instead, which picks a single, most suitable cell from all adjacent cells. In other words, the cell that aligns most closely with the normal vector is chosen as the initial state.

Once both initial states are determined, we can solve the Riemann problem with left and right initial states shown below in the primitive form:

$$\mathbf{w}_L = \begin{pmatrix} \rho_L \\ v_{x,L} \\ v_{y,L} \\ p_L \end{pmatrix}, \quad \mathbf{w}_R = \begin{pmatrix} \rho_R \\ v_{x,R} \\ v_{y,R} \\ p_R \end{pmatrix}. \tag{18}$$

Before applying the Riemann solver defined in Section 1, the velocity vector should be firstly transferred into the normal-tangential frame:

$$(v_x, v_y)_L, (v_x, v_y)_R \rightarrow (v_n, v_t)_L, (v_n, v_t)_R. \tag{19}$$

The star states in the Riemann problem solution would then be in the form:

$$\mathbf{w}_L^* = \begin{pmatrix} \rho_L^* \\ v_n^* \\ v_{t,L} \\ p^* \end{pmatrix}, \quad \mathbf{w}_R^* = \begin{pmatrix} \rho_R^* \\ v_n^* \\ v_{t,R} \\ p^* \end{pmatrix}. \tag{20}$$

As we are assuming that the ghost fluid region of the white material is being calculated, which is also the right material in the Riemann problem, thus only $\mathbf{w}_R^*$ is used as the ghost fluid state at the interface cell $I_{x,y}$ after being transferred back to Cartesian coordinates. The same operations should be performed for all the other interface cells to determine the dynamic boundary condition at the interface.

Given the ghost fluid states at the interface, we still need to extrapolate them into the ghost fluid region to complete the dynamic boundary conditions. In order to populate the ghost cells, constant extrapolation can be applied, which is the solution to a boundary-value problem:

$$\hat{\mathbf{n}} \cdot \nabla Q = 0, \tag{21}$$

where Q is any variable to be extrapolated, and the normal vector can still be calculated from the level set function. The techniques used to solve this problem include the iterative method [10], fast-sweeping [11], and fast-marching [12]. Considering efficiency and implementation, the fast-sweeping method is utilized in this study.

When implementing the fast-sweeping method, the first step is to clear all the values in ghost fluid region cells that are not adjacent to the interface by setting them to a sufficiently large value. Therefore, if any updated variable quantity is smaller than the current one in magnitude, it would be accepted as a better approximation. The sweeps in this method are taken along different axes in different directions. The four possible sequences of a single sweep are shown below:

$$\left.\begin{array}{l} i = 1 : I,\ j = 1 : J, \\ i = I : 1,\ j = 1 : J, \\ i = I : 1,\ j = J : 1, \\ i = 1 : I,\ j = J : 1, \end{array}\right\} \tag{22}$$

where I and J are the total numbers of cells in x- and y-direction, respectively. In each sweep, for a single cell to be updated, the x- and y-neighbors closer to the interface are selected by picking a smaller level set magnitude. The information in neighbors is used in the update function:

$$|n_x|\left(\frac{Q_{i,j} - Q_x}{\Delta x}\right) + |n_y|\left(\frac{Q_{i,j} - Q_y}{\Delta y}\right) = 0, \tag{23}$$

where n_x, n_y are the x- and y-components of the normal vector, Q_x and Q_y are the variables in x- and y-neighboring cells, $Q_{i,j}$ is the updated value. If the magnitude of $Q_{i,j}$ is smaller than that of the current value, it is accepted as the new value.

In a single sweep, only the cells in the ghost fluid region and are not adjacent to the interface are updated, whilst those in the real material region and on the interface are kept unchanged. Once the whole ghost fluid region is swept, the current sweep is completed and the next one can be started. For two-dimensional cases, all four possible sweepings need to be performed at least once, which would be enough to populate the ghost fluid region.

2.3. Adaptions for Rigid Bodies

The Riemann GFM introduced above is designed for multi-physics scenarios. However, the fluid-rigid body interaction in this study is not exactly such a case. Although there are two distinct materials in the system, the state within the rigid body is not cared for. Therefore, only the fluid state is in evolution by numerical methods, whilst the fluid-structure interactions are modeled with adaptions on the Riemann initial states.

When a static rigid body is considered, given its shape and center position, the level set function only needs to be calculated once and then kept unchanged. The key difference occurs in Riemann GFM. As shown in Figure 2, now the white material is the fluid, and the gray material is the rigid body. An interface cell $I_{x,y}$ and a point P_0 on the interface are still identified, however, now the bilinear interpolation only needs to be performed at P_1 inside the real material. The left and right initial states of the Riemann problem is shown below:

$$\mathbf{w}_L = \begin{pmatrix} \rho_R \\ -v_{n,R} \\ v_{t,R} \\ p_R \end{pmatrix}, \quad \mathbf{w}_R = \begin{pmatrix} \rho_R \\ v_{n,R} \\ v_{t,R} \\ p_R \end{pmatrix}, \tag{24}$$

where fluid is the right material and rigid body is the left material. Compared with equation (18), now the left initial state is generated by copying the fluid density, pressure, and tangential velocity while reversing the normal velocity. The Riemann problem is solved in the same way as before and w_R^* is still used as the interface state in the ghost fluid region.

As for a moving rigid body, the level set function calculation needs to be reimplemented at the beginning of each iteration based on the shape, velocity, and initial center position of the rigid body. For the Riemann problem initial states, a reference frame transformation can be performed to use the same method as that for static rigid bodies:

$$\mathbf{v}_{relative} = \mathbf{v}_{fluid} - \mathbf{v}_{rigid}. \tag{25}$$

The above equation transfers the fluid velocity from the world reference frame into the rigid body reference frame, in which the rigid body is static. Therefore, the same operations as those for a static rigid body can be performed, only with $\mathbf{v}_{relative}$ replacing $\mathbf{v}_{fluid}$. After obtaining the right star state w_R^* as shown in equation (20), the interface state in Cartesian coordinates is:

$$\mathbf{u}_{\text{interface}} = \begin{pmatrix} \rho_R^* \\ v_n^* \hat{\mathbf{n}} + \mathbf{v}_t + \mathbf{v}_{\text{rigid}} \\ p_R^* \end{pmatrix}. \tag{26}$$

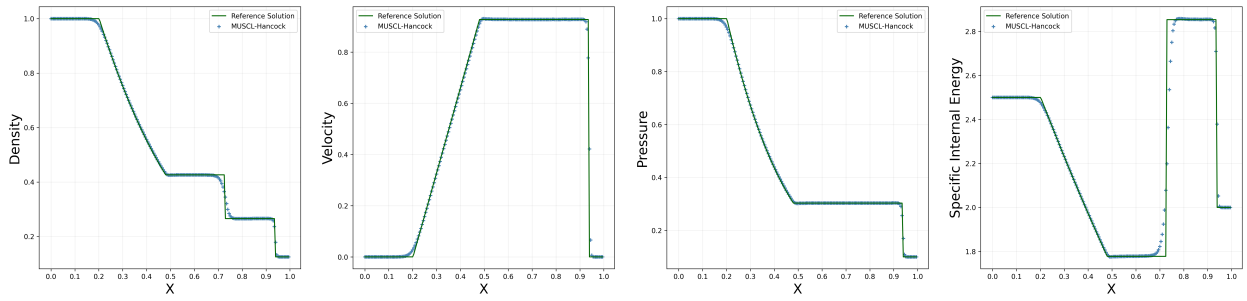

Figure 4: Sod Test results in x-direction

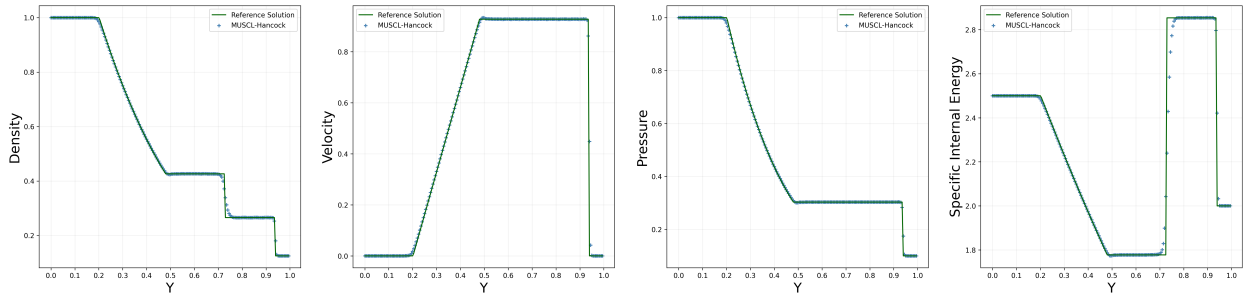

Figure 5: Sod Test results in y-direction

3. Numerical Results

In this section, the methodology is tested against several fluid-rigid body interaction cases, as well as a single material Euler equation test for validation of the implementation of numerical methods. The numerical simulation results are shown for static rigid bodies, a moving rigid body, and a rigid body with time-varying velocity. In all experiments, the CFL number is 0.8 and the resolution is 500×500.

3.1. Single Material Euler Equations

Before solving fluid-structure interaction problems, we need to confirm the effectiveness of the realized numerical methods for two-dimensional Euler equations. Therefore, in this sub-section, the Sod Test and Cylindrical Explosion Test from Toro's book [13] are applied, and a second-order accurate MUSCL-Hancock scheme is utilized to solve these tests. For both tests, the material is an ideal gas with adiabatic index $\gamma = 1.4$.

The Sod Test is essentially a one-dimensional test. In a domain of $x \in [0, 1], y \in [0, 1]$, the initial state is defined as below:

$$\left.\begin{array}{ll} \rho_L = 1.0, & \rho_R = 0.125, \\ u_L = 0.0, & u_R = 0.0, \\ v_L = 0.0, & v_R = 0.0, \\ p_L = 1.0, & p_R = 0.1, \end{array}\right\} \tag{27}$$

where u is the velocity in the x-direction and v is the velocity in y-direction. In the Sod Test in the x-direction, the left state corresponds to $x \leq 0.5$ while the right to $x > 0.5$. In the y-direction Sod Test, the left corresponds to $y \leq 0.5$ and the right to $y > 0.5$. In this study, both of these two versions are run to a final time $t = 0.25$s, with one-dimensional plots taken along the corresponding axes shown in Figures 4 and 5. In the figures shown above, identical behaviors can be observed in the x- and y-direction. The green lines represent the reference solution from an exact Riemann solver.

The cylindrical explosion test is performed in a larger domain of $x \in [0, 2], y \in [0, 2]$, with the initial condition

$$\left.\begin{array}{ll} \rho_{ins} = 1.0, & \rho_{out} = 0.125, \\ u_{ins} = 0.0, & u_{out} = 0.0, \\ v_{ins} = 0.0, & v_{out} = 0.0, \\ p_{ins} = 1.0, & p_{out} = 0.1. \end{array}\right\} \tag{28}$$

The initial condition regions are distinguished by a circle of radius $R = 0.4$ centered at point $(1, 1)$. Subscripts *ins* and *out* denote values inside and outside the circle respectively. Given the cylindrical symmetry property, the simulation results can be shown in one-dimensional plots, as in Figure 6. To further ensure the correctness of the results, surface plots of density and pressure are also illustrated in Figures 7 and 8. All the results above are in good agreement with those in Toro's book [13].

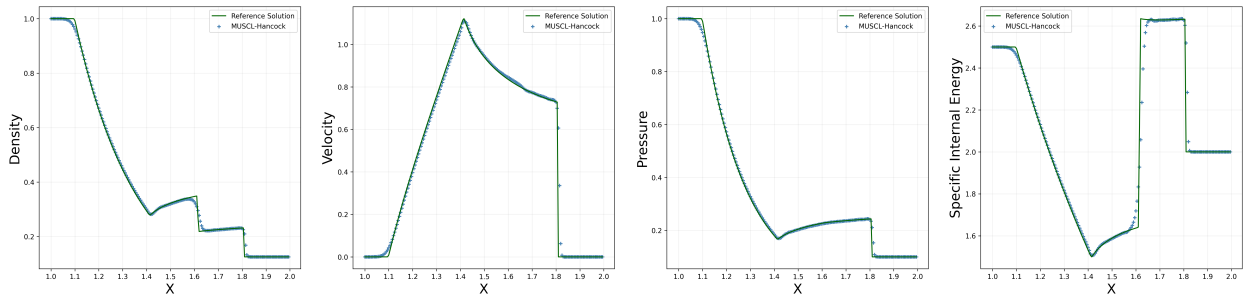

Figure 6: Cylindrical Explosion Test results in a half section

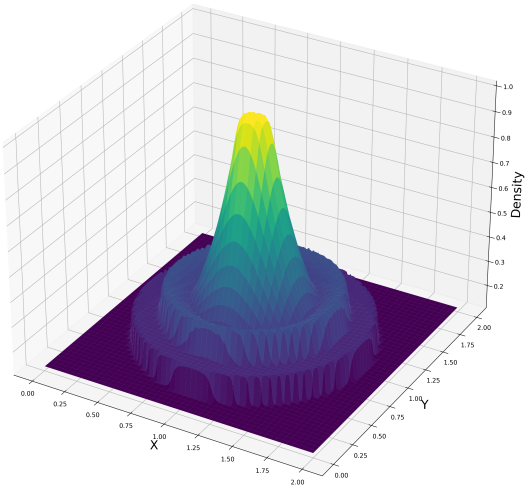

Figure 7: Surface plot of the density in Cylindrical Explosion Test

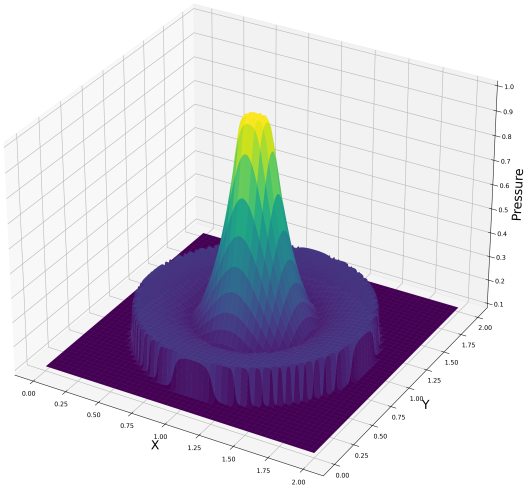

Figure 8: Surface plot of the pressure in Cylindrical Explosion Test

3.2. *Interactions with Rigid Geometries*

After confirming the effectiveness of the numerical methods, we start to simulate fluid-rigid body interactions. To begin with, this study considers the interaction of a shock wave with a static rigid geometry, where the level set method and Riemann GFM discussed in Section 2 are utilized. This class of tests is carried out on a simulation domain of $x \in [0, 1], y \in [0, 1]$, with the final time $t = 0.4$s. The initial condition of the fluid is the same for each case, which is:

$$\left.\begin{array}{ll} \rho_L = 1.3764, & \rho_R = 1.0, \\ u_L = 0.394, & u_R = 0.0, \\ v_L = 0.0, & v_R = 0.0, \\ p_L = 1.5698, & p_R = 1.0, \end{array}\right\} \tag{29}$$

where the left state corresponds to $x \leq 0.2$. Four tests are performed with distinct rigid geometries. For each case, the density, velocity magnitude, and pressure evolution at time $t = 0.2$s and $t = 0.4$s are shown. To identify the waves in the solutions clearly, the mock-schlieren plots of the density gradient are added, for which the variable is defined as:

$$\exp\left(\frac{-20|\nabla\rho|}{1000\rho}\right), \tag{30}$$

where $\nabla\rho$ is the density gradient.

To begin with, Figure 9 illustrates the first test case, where the rigid body is a circle of radius $R = 0.2$ centered at $(0.6, 0.5)$. The density and pressure fields show that the shock first hits the left side of the rigid body. This creates a high-density region that moves around it. A reflected wave appears near the front. The shock moves around the obstacle, causing the flow to become uneven. The velocity field shows that the flow slows down where the shock hits and speeds up in the wake.

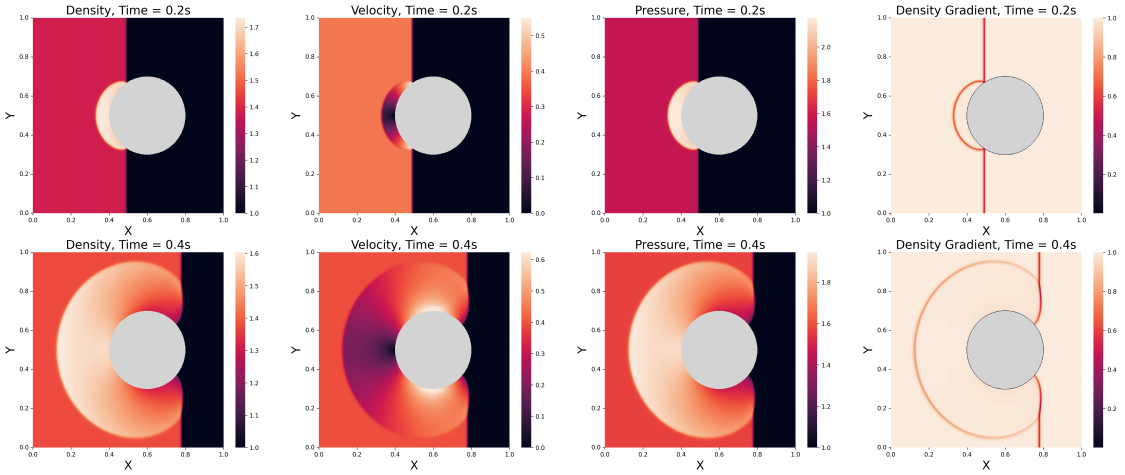

Figure 9: Simulation results of shock wave interacting with a circle rigid body

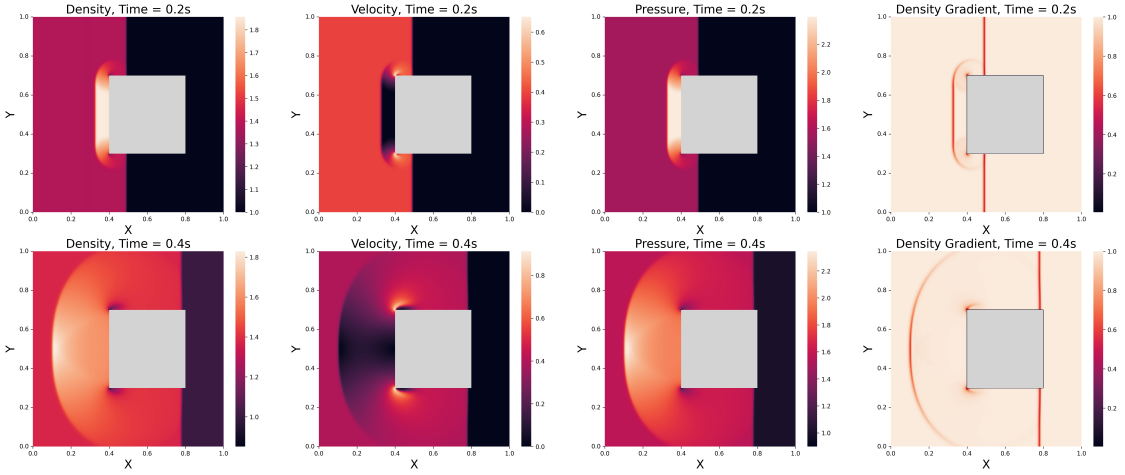

Figure 10: Simulation results of shock wave interacting with a square rigid body

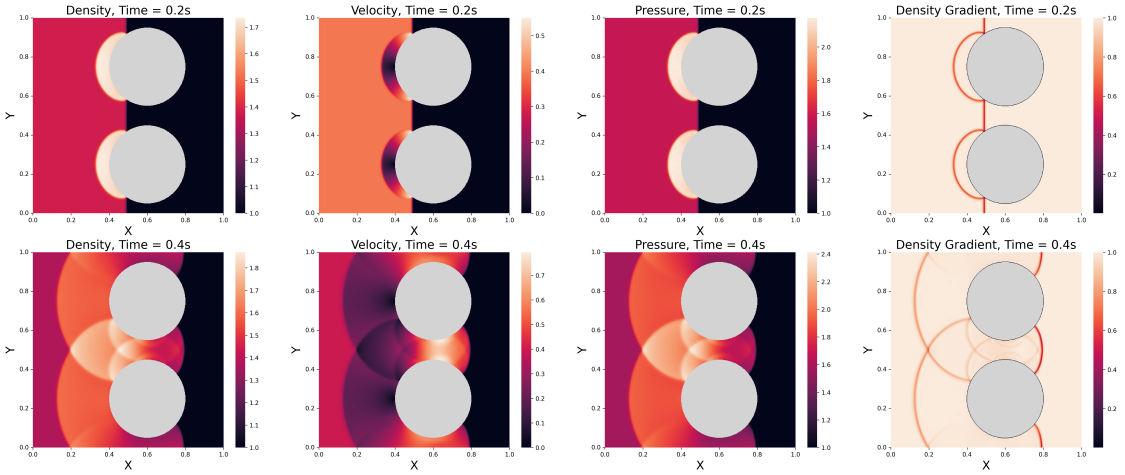

Figure 11: Simulation results of shock wave interacting with two circle rigid bodies

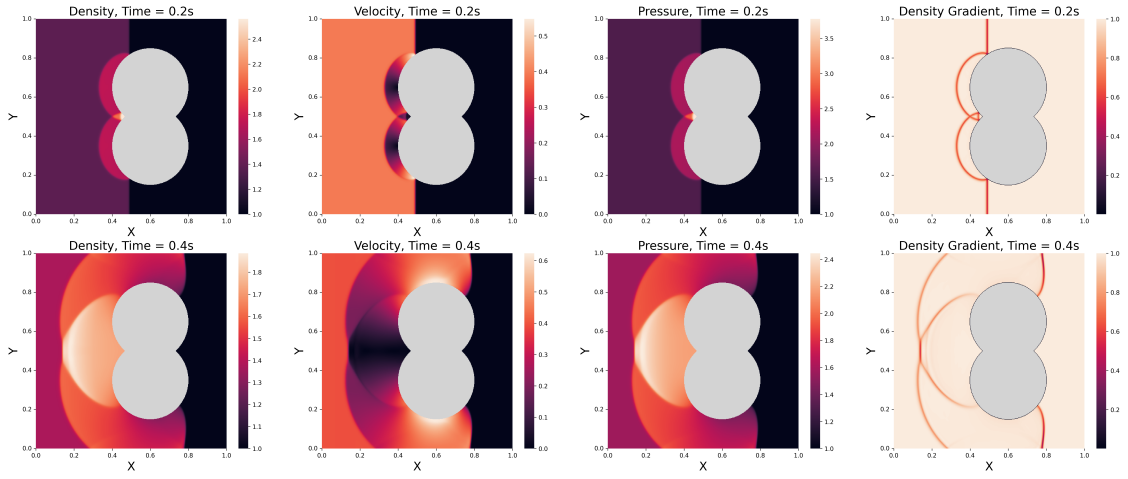

Figure 12: Simulation results of shock wave interacting with two overlapping circle rigid bodies

The flow spreads out after passing the body. The density gradient visualization shows the shock front, its reflection, and other waves made by the shock hitting the surface.

In contrast, Figure 10 presents the results for a square rigid body with edge length $l = 0.4$ centered at the same position. Compared to the circular obstacle, the density and pressure fields show stronger and more focused compression at the front face. The shock bends sharply around the edges instead of wrapping around smoothly. The velocity field shows a big slowdown in front of the square. High-speed areas appear near the corners because of diffraction. The density gradient visualization shows strong shock reflections at the flat front and clear secondary waves from the edges.

For the third case, involving two circular rigid bodies of radius $R = 0.2$ centered at $(0.6, 0.25)$ and $(0.6, 0.75)$, Figure 11 illustrates the interaction between the shock wave and multiple obstacles. The density and pressure fields show that the shock hits both circles at the same time. This creates high-density areas around their front surfaces. As the shock moves, waves interact between the two bodies. Many reflected waves and interference patterns appear. The velocity field shows the flow slows down in front of the obstacles and speeds up in the narrow space between them. The density gradient visualization shows detailed wave patterns caused by fluid interactions between the two bodies.

Finally, Figure 12 presents results for two overlapping circles of radius $R = 0.2$, centered at $(0.6, 0.35)$ and $(0.6, 0.65)$. The initial shock hits both bodies at the same time. Because they are close together, the wave interactions are different from the separated case. The density and pressure fields show a single compressed area at the front. The velocity field shows strong slowing down in this area and faster flow around the outer edges. The density gradient visualization shows strong shock reflections and secondary waves caused by both bodies blocking the flow.

3.3. Moving Rigid Body

This sub-section considers a circular rigid body moving in an otherwise initially static flow field, in comparison with a case in which the flow is moving while the rigid body is static. In each case, the rigid body is a circle with $R = 0.2$ and the final time is $t = 1.0$s. For the test with a static rigid body, the center of the circle is at $(0.5, 0.5)$ and the initial state of the fluid is

$$(\rho_R, u_R, v_R, p_R) = (1, 1, 0, 1). \tag{31}$$

For the test with a moving rigid body, the circle center is initially set at $(1.5, 0.5)$ and moves with a constant velocity $v_{\text{rigid}} = (-1, 0)$. Thus the final position of the circle center is also $(0.5, 0.5)$. The initial data of fluid is

$$(\rho_R, u_R, v_R, p_R) = (1, 0, 0, 1). \tag{32}$$

For comparison convenience, a simulation domain of $x \in [0, 2], y \in [0, 1]$ is used for both tests. Figures 13 and 14 illustrate the simulation results of each case at time $t = 0.5$s and $t = 1.0$s respectively.

The simulation results of the two cases show very similar flow patterns. Both cases have high-density and high-pressure regions at the front of the circle because the flow stops there. A low-pressure wake forms behind the body because the flow separates. The velocity field

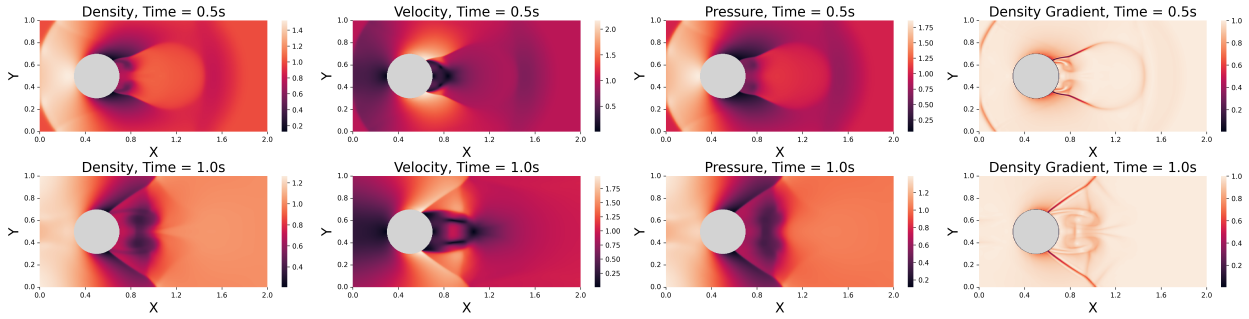

Figure 13: Simulation results of moving fluid around a stationary rigid body

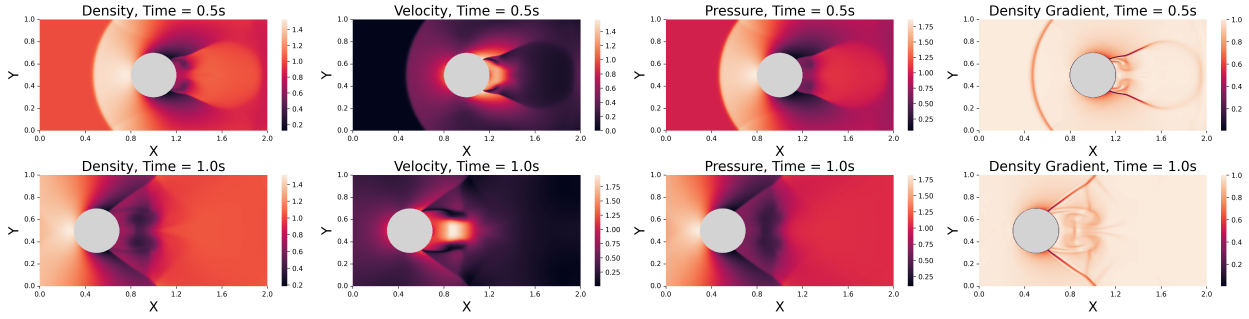

Figure 14: Simulation results of a moving rigid body interacting with an initially static flow field

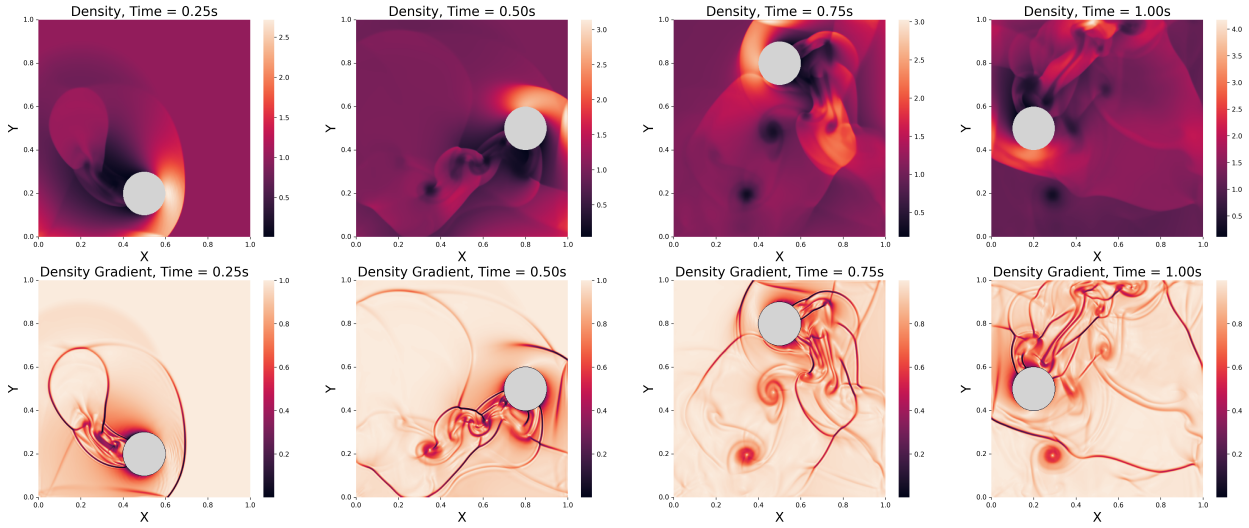

Figure 15: Simulation results of a moving rigid body with time-varying velocity interacting with an initially static flow field

shows slowing down at the front and faster flow along the sides. This creates two symmetric shear layers in the wake. The density gradient visualization shows the flow separating at the back of the body, with clear shear layers and vortex structures forming over time.

A close comparison of the two cases shows small differences in the density and pressure fields. These differences may come from how the simulation handles body motion and background flow. Also, since the rigid body shape is modeled with a level set function, it changes slightly in the moving body case. This may also cause some of the differences seen in the results. The overall similarity between the two cases proves that a still obstacle in moving fluid is the same as a moving obstacle in still fluid. This shows that the flow pattern depends on the movement between the fluid and the body, rather than the fixed position of either one.

3.4. Time-varying Velocity

This test deals with the interaction between a moving rigid body with time-varying velocity and an initially static flow field. In this case a domain of $x \in [0, 1], y \in [0, 1]$ is used and the final time is $t = 1.0$s. In contrast to the previous tests, reflective domain boundary conditions are applied at all walls, instead of the transmissive boundary condition used before. Here the rigid body is a circle of radius $R = 0.1$ initially centered at $(0.2, 0.5)$. A time-varying velocity field is attached to the circle, driving it to make a circular motion of radius $r = 0.3$ around the central point of the domain $(0.5, 0.5)$. Assuming constant angular velocity, the linear velocity field of the rigid body is defined as:

$$\begin{gathered}\mathbf{v}_{\text{rigid}} = \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} -wrsin\alpha \\ wrcos\alpha \end{pmatrix}, \\ \alpha = \pi + wt, \ t \in [0, 1],\end{gathered} \tag{33}$$

where α is the current angle of the rigid body in the circular motion, the initial angle is defined as π corresponding to the initial position, and t is the current time. To make sure the rigid body center passes the points $(0.5, 0.2)$, $(0.8, 0.5)$, $(0.5, 0.8)$, and finally back to $(0.2, 0.5)$ at $t = 1$, the angular velocity $w = 2\pi$. Given the velocity field, the center position of the rigid body can be calculated as:

$$\mathbf{P}_{\text{rigid}} = \begin{pmatrix} 0.5 + rcos\alpha \\ 0.5 + rsin\alpha \end{pmatrix}, \tag{34}$$

which is used to obtain the current level set function.

The simulation results of this test are demonstrated in Figure 15, which includes two-dimensional plots of density and density gradient fields at the four positions required in this case. As the body moves in a circular path, it creates high-density and high-pressure areas at the front. Flow separation in the wake forms low-density vortex structures. The velocity field shows changing shear layers and vortex shedding as the body changes direction. The density gradient visualization shows strong waves coming from the moving body. These waves hit the domain boundaries, causing many reflections and complex shock patterns. Compared to earlier cases with open boundaries, this test shows how wave reflections increase flow complexity. The moving body reflected waves, and forming vortices interact, creating an unsteady and detailed flow field.

4. Conclusions

This study focuses on using GFM-like methods to simulate two-dimensional fluid-structure interactions. The underlying PDE is the compressible Euler equations, and high-resolution numerical methods MUSCL-Hancock scheme is employed to solve these equations. The numerical implementation is validated through representative tests from Toro's book [13]. To model the interactions, multi-physics techniques such as the level set method and GFM are utilized. The level set function tracks the rigid body's location, while the Riemann GFM enforces dynamic boundary conditions in the ghost fluid region. Additionally, modifications to the Riemann GFM are introduced to better accommodate fluid-rigid body interactions. Various test cases, including shock-wave interactions with static rigid geometries, moving rigid bodies, and time-varying velocity effects, confirm the effectiveness of the proposed methods in capturing key physical phenomena.

The results show that the numerical schemes and interface modeling techniques handle complex fluid-structure interactions. They capture shock reflections, wave movement, wake formation, and vortex dynamics. The level set method provides a robust and flexible framework for tracking rigid body motion, while the modified Riemann GFM ensures accurate enforcement of boundary conditions. Despite these strengths, further improvements are needed to enhance numerical stability in extreme flow conditions and to minimize diffusion near interfaces. Future work could extend these methods to three-dimensional problems, incorporate turbulence modeling, or optimize computational efficiency for large-scale simulations. Additionally, comparisons with experimental data would help further validate the approach and refine its predictive capabilities.

Appendix A. MUSCL-Hancock Scheme

As described in the introduction, there are three steps in the MUSCL-Hancock scheme. The first step, data reconstruction, takes the finite volume quantity $\mathbf{u}_i^n$ and fits a function through it to get $\mathbf{u}_i(x)$ by using the information of its neighbors. For example, a linear reconstruction is given by:

$$\begin{aligned} \mathbf{u}_i(x) &= \mathbf{u}_i^n + (x - x_i)\frac{\Delta_i}{\Delta x}, \\ \Delta_i &= \frac{1}{2}(1+\omega)\Delta_{i-1/2} + \frac{1}{2}(1-\omega)\Delta_{i+1/2}, \\ \Delta_{i-1/2} &= \mathbf{u}_i^n - \mathbf{u}_{i-1}^n, \quad \Delta_{i+1/2} = \mathbf{u}_{i+1}^n - \mathbf{u}_i^n, \end{aligned} \tag{A.1}$$

where $\omega \in [-1, 1]$ is a parameter that weighs in either positive or negative directions. For the Euler equations, $\omega = 0$ is a safe choice. We can now define the cell boundary values:

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

During reconstruction, slope limiters can be applied to the slope quantities:

$$\begin{aligned} \bar{\mathbf{u}}_i^L &= \mathbf{u}_i^n - \frac{1}{2}\bar{\Delta}_i = \mathbf{u}_i^n - \frac{1}{2}\xi(r)\Delta_i, \\ \bar{\mathbf{u}}_i^R &= \mathbf{u}_i^n + \frac{1}{2}\bar{\Delta}_i = \mathbf{u}_i^n + \frac{1}{2}\xi(r)\Delta_i, \end{aligned} \tag{A.3}$$

where $\xi(r)$ is a slope limiter function. The slope limiter applied in this study is called Superbee, which is defined in Toro *et al.* [14] as:

$$\xi(r) = \begin{cases} 0, & r \le 0, \\ r, & 0 < r \le \frac{1}{2}, \\ 1, & \frac{1}{2} < r \le 1, \\ \min(r, \xi_R(r), 2), & r > 1, \end{cases} \tag{A.4}$$

in this definition, $\xi_R(r)$ is calculated as:

$$\xi_R(r) = \frac{2}{1+r}, \tag{A.5}$$

and the slope ratio r is defined as:

$$r = \frac{\Delta_{i-1/2}}{\Delta_{i+1/2}} = \frac{q_i - q_{i-1}}{q_{i+1} - q_i}, \tag{A.6}$$

where q is the quantity to be reconstructed.

The second step, half-time step evolution, is based on the two boundary values, $\bar{\mathbf{u}}_i^L$ and $\bar{\mathbf{u}}_i^R$, obtained from reconstruction equation (A.3). By using the flux functions defined in equation (5), these boundary values are updated according to:

$$\begin{aligned} \bar{\mathbf{u}}_i^{L,n+1/2} &= \bar{\mathbf{u}}_i^L - \frac{1}{2}\frac{\Delta t}{\Delta x}\left[\mathbf{f}\left(\bar{\mathbf{u}}_i^R\right) - \mathbf{f}\left(\bar{\mathbf{u}}_i^L\right)\right], \\ \bar{\mathbf{u}}_i^{R,n+1/2} &= \bar{\mathbf{u}}_i^R - \frac{1}{2}\frac{\Delta t}{\Delta x}\left[\mathbf{f}\left(\bar{\mathbf{u}}_i^R\right) - \mathbf{f}\left(\bar{\mathbf{u}}_i^L\right)\right]. \end{aligned} \tag{A.7}$$

The first-order update scheme applied for MUSCL-Hancock is the Godunov's method [15], which uses Riemann problems to solve partial differential equations (PDE). The Riemann problem [16] is an initial value problem, comprising a conservation equation and a piecewise-constant initial state with a single discontinuity located at $x = x_0$, as shown below:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{u})}{\partial x} &= 0, \\ \mathbf{u}(x, 0) &= \begin{cases} \mathbf{u}_L, & x < x_0, \\ \mathbf{u}_R, & x > x_0, \end{cases} \end{aligned} \tag{A.8}$$

which is possible to be solved exactly for the Euler equations with an ideal gas.

In the MUSCL-Hancock scheme, Godunov's method uses the half-time step updated boundary values obtained from equation (A.7) as the two initial states for the Riemann problem. Then the Riemann problem is solved, either by exact or approximate methods, and the solution provides information to determine the numerical flux at the cell boundary. Godunov's method in one dimension is shown below:

$$\begin{aligned} \mathbf{u}_i^{n+1} &= \mathbf{u}_i^n - \frac{\Delta t}{\Delta x}\left[\mathbf{f}\left(\mathbf{u}_{i+1/2}\right) - \mathbf{f}\left(\mathbf{u}_{i-1/2}\right)\right], \\ \mathbf{u}_{i+1/2} &= \mathbf{u}_{i+1/2}\left(\bar{\mathbf{u}}_i^{R,n+1/2}, \bar{\mathbf{u}}_{i+1}^{L,n+1/2}\right), \end{aligned} \tag{A.9}$$

where the numerical fluxes are computed directly with boundary values solved in the Riemann problem.

Before solving the Riemann problem, it is necessary to introduce the general form of the Riemann problem solution for the Euler equations, which is shown in the x-t diagram in Figure A.16. To obtain this diagram, the eigenvalues and characteristics of Euler equations need to be calculated. In equation (4) we defined the conservation form of the Euler equations in two dimensions, now a slice in the x-direction can be taken without loss of generality:

$$\frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ E \end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix} \rho v_x \\ \rho v_x^2 + p \\ \rho v_x v_y \\ (E+p)v_x \end{pmatrix} = 0. \tag{A.10}$$

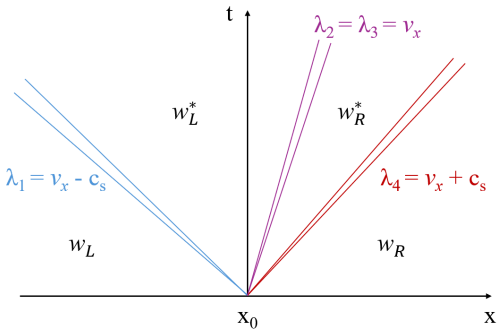

Figure A.16: A general Riemann problem solution in x-direction for two-dimensional Euler equations, with double-lines for generic wave

To compute the eigenvalues, it is better to use the primitive variable form:

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ v_x\\ v_y\\ p\end{pmatrix}+\begin{pmatrix}v_x & \rho & 0 & 0\\ 0 & v_x & \frac{1}{\rho} & 0\\ 0 & 0 & v_x & 0\\ 0 & \rho c_s^2 & 0 & v_x\end{pmatrix}\frac{\partial}{\partial x}\begin{pmatrix}\rho\\ v_x\\ v_y\\ p\end{pmatrix}=0, \quad \text{(A.11)}$$

where c_s is the speed of sound. For an ideal gas, c_s is defined as:

$$c_s = \sqrt{\frac{\gamma p}{\rho}}. \quad \text{(A.12)}$$

The eigenvalues of the primitive variable matrix are straightforward:

$$\lambda_1 = v_x - c_s, \quad \lambda_2 = \lambda_3 = v_x, \quad \lambda_4 = v_x + c_s. \quad \text{(A.13)}$$

There are four real eigenvalues, but only three of them are distinct. As a result, there are only three distinct waves in the Riemann problem solution, as in the one-dimensional cases. In specific, λ_1 represents the slope of the left wave in Figure A.16, λ_4 corresponds to the right wave, and λ_2 is for the contact discontinuity in the middle. The fourth wave, corresponding to λ_3, is degenerate, overlapping with the contact discontinuity. Since the characteristic variable for the degenerate wave is the transverse velocity (v_y in this example), this wave is also called a shear wave, in which the velocity moves perpendicular to the wave. Furthermore, across the shear wave, only transverse velocity jumps, and the shear wave is the only place transverse velocity jumps. Given that only density jumps across the contact discontinuity, the star states in the Riemann problem solution can be written in primitive variables as:

$$\mathbf{w}_L^* = \begin{pmatrix}\rho_L^*\\ v_x^*\\ v_{y,L}\\ p^*\end{pmatrix}, \quad \mathbf{w}_R^* = \begin{pmatrix}\rho_R^*\\ v_x^*\\ v_{y,R}\\ p^*\end{pmatrix}, \quad \text{(A.14)}$$

where the transverse velocities are simply those in the initial states, and the other three variables can be calculated using Riemann solvers. Note that the Riemann problem above is constructed in x-direction. For y-direction, the star states are similar, only with v_x as the transverse velocity.

For the Riemann problem solver, an exact method from Toro's book [13] is used in this study, which is defined as:

$$\begin{aligned} f(p^*, \mathbf{u}_L, \mathbf{u}_R) &= g(p^*, \mathbf{u}_R) - g(p^*, \mathbf{u}_L) \\ &= f_R(p^*, \mathbf{u}_R) + f_L(p^*, \mathbf{u}_L) + \Delta v = 0, \end{aligned} \quad \text{(A.15)}$$

where p^* is pressure in the star states of the Riemann problem solution, $\mathbf{u}_L$ and $\mathbf{u}_R$ are the left and right initial states, corresponding to $\bar{\mathbf{u}}_i^{R,n+1/2}$ and $\bar{\mathbf{u}}_{i+1}^{L,n+1/2}$ in equation (A.9), respectively. In the exact solver, the star state pressure p^* is firstly solved iteratively using the implicit equation (A.15), with the Newton-Raphson method:

$$p^*_{(m)} = p^*_{(m-1)} - \frac{f(p^*_{(m-1)}, \mathbf{u}_L, \mathbf{u}_R)}{f'(p^*_{(m-1)}, \mathbf{u}_L, \mathbf{u}_R)}, \quad \text{(A.16)}$$

where m is the iteration step, f' is the derivative of f with respect to p^*. The functions f_L and f_R in equation (A.15) are defined as:

$$\begin{aligned} f_K(p^*, \mathbf{u}_K) &= \begin{cases} (p - p_K)\sqrt{\frac{A_K}{p^* + B_K}}, & p^* > p_K, \\ \frac{2c_{s,K}}{\gamma - 1}\left[\left(\frac{p^*}{p_K}\right)^{\frac{\gamma-1}{2\gamma}} - 1\right], & p^* < p_K, \end{cases} \\ A_K &= \frac{2}{(\gamma+1)\rho_K}, \quad B_K = \frac{\gamma - 1}{\gamma + 1} p_K. \end{aligned} \quad \text{(A.17)}$$

After p^* is converged, the densities and velocity in the star states can be computed according to the wave types. In the Riemann problem solution for Euler equations, both the left and right waves can be shock or rarefaction waves, depending on whether p^* is larger than the initial left and right pressure. An x-t diagram of a Riemann problem solution with a left-moving rarefaction and a right-moving shock wave is shown in Figure A.17. If $p^* < p_K$, a rarefaction wave is generated. Across a rarefaction, the star state density and the wave speeds are calculated as:

$$\begin{aligned} \rho_K^* &= \rho_K \left(\frac{p^*}{p_K}\right)^{1/\gamma}, \\ R_K^{\text{head}} &= v_K \mp c_{s,K}, \quad R_K^{\text{tail}} = v^* \mp c^*_{s,K}, \end{aligned} \quad \text{(A.18)}$$

where the $\mp$ signs correspond to left and right waves, respectively, and R_K^{head} is for the boundary between

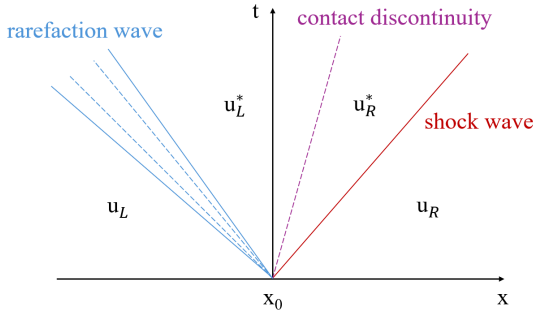

Figure A.17: An exact Riemann problem solution for the Euler equations

rarefaction fan and initial states, while R_K^{tail} is for the boundary between rarefaction fan and star states.

If when $p^* > p_K$, a shock wave is generated. The star state density and shock wave speed can be calculated as:

$$\rho_K^* = \rho_K \frac{\frac{p^*}{p_K} + \frac{\gamma-1}{\gamma+1}}{\frac{\gamma-1}{\gamma+1}\frac{p^*}{p_K} + 1},$$
$$S_K = v_K \pm c_{s,K}\sqrt{\left(\frac{\gamma+1}{2\gamma}\right)\frac{p^*}{p_K} + \frac{\gamma-1}{2\gamma}}, \tag{A.19}$$

where the ± signs correspond to left and right waves, respectively. For both wave types, the star state velocity can be obtained through the formula:

$$v^* = \frac{1}{2}(v_L + v_R) + \frac{1}{2}\left[f_R(p^*, \mathbf{u}_R) - f_L(p^*, \mathbf{u}_L)\right]. \tag{A.20}$$

Until now, all the star state variables have been determined. To complete the Riemann problem solution, one more state to be defined is that inside the rarefaction fan:

$$\rho_K^{\text{raref.}} = \rho_K\left[\frac{2}{\gamma+1} \pm \frac{\gamma-1}{\gamma+1}c_{s,K}\left(v_K - \frac{x-x_0}{t-t_0}\right)\right]^{\frac{2}{\gamma-1}},$$
$$v_K^{\text{raref.}} = \frac{2}{\gamma+1}\left[\pm c_{s,K} + \frac{\gamma-1}{2}v_K + \frac{x-x_0}{t-t_0}\right],$$
$$p_K^{\text{raref.}} = p_K\left[\frac{2}{\gamma+1} \pm \frac{\gamma-1}{\gamma+1}c_{s,K}\left(v_K - \frac{x-x_0}{t-t_0}\right)\right]^{\frac{2\gamma}{\gamma-1}}, \tag{A.21}$$

where the ± signs correspond to left and right waves, respectively. The above are all the definitions of the exact Riemann problem solver. Given the whole solution, the boundary values used in Godunov's method defined in equation (A.9) can be determined. For example, in the case shown in Figure A.17,

$$u_{i+1/2} = u(0,t) = \begin{cases} u_L, & \text{if } R_L^{\text{head}} \geq 0, \\ u_L^{\text{raref.}}, & \text{if } R_L^{\text{head}} < 0 \leq R_L^{\text{tail}}, \\ u_L^*, & \text{if } R_L^{\text{tail}} < 0 \leq v^*, \\ u_R^*, & \text{if } v^* < 0 \leq S_R, \\ u_R, & \text{if } S_R < 0, \end{cases} \tag{A.22}$$

where $u_{i+1/2}$ is the boundary value in conservation form. By substituting it into the equation (A.9), the evolution in x-direction is completed. Then the whole MUSCL-Hancock scheme, including data reconstruction, half-time step evolution, and Godunov's method, is performed once again in the y-direction to complete the overall evolution.

References

[1] L. Euler, Principes généraux du mouvement des fluides, Mémoires de l'académie des sciences de Berlin (1757) 274–315.

[2] R. Eymard, T. Gallouët, R. Herbin, Finite volume methods, Handbook of numerical analysis 7 (2000) 713–1018.

[3] P. Woodward, P. Colella, The numerical simulation of two-dimensional fluid flow with strong shocks, Journal of computational physics 54 (1) (1984) 115–173.

[4] E. Hansen, A. Ostermann, Dimension splitting for evolution equations, Numerische Mathematik 108 (4) (2008) 557–570.

[5] G. Strang, On the construction and comparison of difference schemes, SIAM journal on numerical analysis 5 (3) (1968) 506–517.

[6] J. A. Sethian, et al., Level set methods and fast marching methods, Vol. 98, Cambridge Cambridge UP, 1999.

[7] S. Osher, R. P. Fedkiw, Level set methods: an overview and some recent results, Journal of Computational physics 169 (2) (2001) 463–502.

[8] R. P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method), Journal of computational physics 152 (2) (1999) 457–492.

[9] S. K. Sambasivan, H. UdayKumar, Ghost fluid method for strong shock interactions part 1: Fluid-fluid interfaces, Aiaa Journal 47 (12) (2009) 2907–2922.

[10] W.-K. Jeong, R. T. Whitaker, A fast iterative method for eikonal equations, SIAM Journal on Scientific Computing 30 (5) (2008) 2512–2534.

[11] H. Zhao, A fast sweeping method for eikonal equations, Mathematics of computation 74 (250) (2005) 603–627.

[12] J. A. Sethian, Fast marching methods, SIAM review 41 (2) (1999) 199–235.

[13] E. F. Toro, Riemann solvers and numerical methods for fluid dynamics: a practical introduction, Springer Science & Business Media, 2013.

[14] E. F. Toro, S. Billett, Centred tvd schemes for hyperbolic conservation laws, IMA Journal of Numerical Analysis 20 (1) (2000) 47–79.

[15] 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.
[16] B. Riemann, Über die Fortpflanzung ebener Luftwellen von endlicher Schwingungsweite, Vol. 8, Verlag der Dieterichschen Buchhandlung, 1860.