Interface preserving mesh optimization method for multi-material simulations

Chao Zhang[1], Nuo Lei[2], Juan Cheng[3] and Chi-Wang Shu[4]

Abstract

In this paper, we propose a mesh optimization method within the framework of the indirect arbitrary Lagrangian-Eulerian (ALE) approach that preserves material interfaces in multi-material problems. We design a simple and efficient objective function which ensures the rezoned mesh to retain good geometric quality and to stay as close as possible to the original Lagrangian mesh in order to keep high resolution in ALE simulations. For interface points, we use the geometric essentially non-oscillatory (GENO) interpolation method to generate an interpolated curve for the interface points, in which the interface points are allowed to move along this curve during optimization, ensuring that the interfaces are maintained. One advantage of the GENO interpolation method is its ability to provide a second-order smooth and non-oscillatory interface curve. Our method eliminates the need to reconstruct material interfaces or solve mixed equations of state in multi-material simulations. Furthermore, it can be applied to moving boundary problems by treating boundary points similarly to interface points. We validate the effectiveness of our method through a series of numerical examples.

Keywords: Rezone; Mesh optimization; Interface maintaining; Multi-material simulation; Arbitrary Lagrangian-Eulerian.

[1]Graduate School, China Academy of Engineering Physics, Beijing 100088, China. E-mail: zhangchao21@gscaep.ac.cn. Research is supported in part by NSFC grant 12031001.

[2]Hua Loo-Keng Center for Mathematical Sciences, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, 100190, China. E-mail: nuo_lei@lsec.cc.ac.cn. Research is supported in part by NSFC grant 12288201.

[3]Corresponding author. Academy for Multidisciplinary Studies, Capital Normal University, Beijing 100048, China. E-mail: chengjuan_bj@foxmail.com. Research is supported in part by National Key R&D Program of China No. 2023YFA1009003.

[4]Division of Applied Mathematics, Brown University, Providence, RI 02912. E-mail: chi-wang_shu@brown.edu. Research is supported in part by NSF grant DMS-2309249.

1 Introduction

In numerical simulations of computational fluid dynamics, the Eulerian and Lagrangian methods are the two most commonly used frameworks, which differ in how to describe the fluid motion. The Eulerian approach uses a fixed mesh to compute physical quantities when the fluid flows through it, while the Lagrangian method allows the mesh to move along with the fluid. Due to these different descriptions, both approaches have their own advantages and disadvantages in computation. The Eulerian method maintains good mesh quality due to the use of fixed mesh, which helps to preserve the accuracy of computational schemes. However, the use of a fixed mesh also makes it more difficult to handle moving boundary problems. In addition, the Eulerian method is less effective in treating material interfaces or contact discontinuities, and often requires other methods to reconstruct the interface, such as the volume-of-fluid (VOF) method or the level set method. For the Lagrangian method, because the mesh moves along with the fluid, it naturally has the ability to track material interfaces, making it more suitable for multi-material problems. The biggest disadvantage of Lagrangian method is that it is prone to large mesh deformations during multidimensional calculations, which can lead to poor simulation results and can even force the calculation to terminate.

Based on these two frameworks, Hirt et al. [14] proposed the arbitrary Lagrangian-Eulerian (ALE) method. The ALE method combines the advantages of both Eulerian and Lagrangian approaches. The mesh can retain sufficient fluid information as in the Lagrangian approach and can maintain good mesh quality as in the Eulerian approach by allowing flexible adjustment of the computational mesh, thus improving the accuracy of simulation and the stability of calculation.

The ALE method is divided into two types: direct ALE and indirect ALE. Direct ALE controls mesh movement through a dedicated moving mesh algorithm, while indirect ALE performs a rezoning step after one or several Lagrangian steps, followed by mapping the physical quantities from the Lagrangian mesh to the new rezoned mesh. Compared to direct

ALE, the indirect ALE is more flexible, making it more suitable to handle large deformation problems. In general, the indirect ALE method consists of three steps: the Lagrangian step, the rezoning step and the remapping step.

Our focus in this paper is on the rezoning step of the indirect ALE method. For indirect ALE, obtaining an appropriate rezoned mesh is crucial. A good rezoned mesh in fluid computations should satisfy the following conditions as much as possible:

1. The rezoned mesh should maintain good mesh geometric quality, and also maintain the smoothness, the orthogonality, and the quasi-uniformity of the mesh. Extremely skewed aspect ratios of any mesh elements will increase the computational error and computational time in ALE simulations;

2. For quadrilaterals and polygons with more edges, the mesh cells should remain convex;

3. The rezoned mesh should be as close as possible to the Lagrangian mesh to preserve sufficient fluid features, while also ensuring lower computational errors in the remapping steps, thereby increasing computational accuracy and resolution;

4. In multi-material fluid computations, the rezoned mesh should keep material interfaces, otherwise, mixed cells may be generated and their closure models could result in non-physical fluids.

It is very difficult to achieve all four conditions in a rezoning method, a good rezoning method usually has to strike a balance among these conditions.

In the earliest work on ALE by Hirt et al. [14], they computed a new mesh velocity based on the acceleration from the pressure gradient and used this new velocity to calculate the rezoned mesh. Obtaining this new mesh velocity usually requires implicit iterations. Furthermore, similar methods for solving the new mesh velocity for rezoning meshes were also adopted in [6, 21]. These methods for obtaining new mesh movement velocities often require solving a partial differential equation.

There is another class of methods which is quite simple; they only require some basic mathematical operations on the coordinates of the old mesh to obtain a new rezoned mesh,

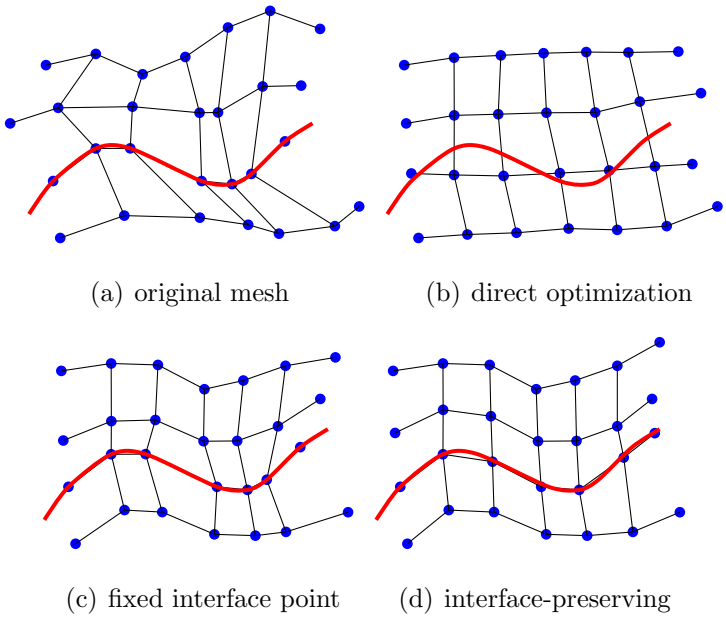
such as those in [1, 2, 9, 11]. Among them, the Laplacian smoothing method proposed by Field [9] is particularly representative. This method takes the arithmetic average of the coordinates of neighboring nodes around each mesh node in the old mesh as the new position for that grid node, effectively performing a smoothing operation on the entire mesh, making it a relatively straightforward rezoning method.

After that, many optimization-based methods were proposed. These methods typically construct an objective function based on mesh quality, treating the mesh rezoning process as equivalent to solving an optimization problem. The advantage of such optimization methods is that they can leverage a wide range of mature and diverse optimization techniques and can conveniently handle various constraint conditions. Among these, the most representative method is the reference Jacobian method proposed by Knupp et al. [18]. In addition to this, there are other optimization-based works [3, 5, 16]. The objective function of the reference Jacobian method cleverly uses the Jacobian matrix of the mesh in its construction. The rezoned mesh obtained through this optimization process not only possesses good mesh quality but also remains close to the Lagrangian mesh.

As mentioned above, handling the interfaces in multi-material problems has always been a critical problem in computational fluid dynamics (CFD). In most of the current studies, including the work discussed above, there are generally three common strategies for treating material interfaces.

One strategy mentioned in [14] is to maintain the Lagrangian movement of the interface points and optimize other mesh nodes during the rezoning process, in other words the interface points are always fixed during the rezoning step, as shown in Figure 1(c). In many CFD simulations, the shape of the material interface often becomes highly complex as the problem evolves. So there is often a situation that the mesh quality in other places is improved well, but the mesh quality near the interface is less improved. More importantly, this approach requires that the material interface in the fluid simulation does not undergo significant deformation, otherwise, the rezoned mesh may become distorted or tangled. Therefore, this

method has significant limitations and cannot accommodate many problems.



(a) original mesh (b) direct optimization

(c) fixed interface point (d) interface-preserving

Figure 1: Schematic diagrams of various optimization strategies

Another strategy is to optimize the entire mesh directly, the Lagrangian movement of the interface points is disregarded during the rezoning process, as illustrated in Figure 1(b). Then one would need to use interface reconstruction methods in the mixed cells. Common interface reconstruction methods include the gradient based method [25], the VOF method [10, 15, 17, 26] and the moment-of-fluid (MOF) method [7, 8]. These methods treat the interface as a region with thickness, where the mesh elements within this region consist of multiple materials. As a result, the mixed equation of state needs to be solved, which is usually inconsistent with the physical model.

The last strategy is to constrain the movement of interface and boundary points during the rezoning step, ensuring that they move on the interface and boundary lines, as illustrated in Figure 1(d), thereby preserving the interface and minimizing the generation of mixed cells as much as possible. In their work [27], Shashkov and Knupp constrained the interface and boundary points to move in the tangential direction.

In summary, the first strategy fixes all the interface points in the optimization process.

Although it maintains the material interface, it may generate low-quality or entangled mesh cells near the interface. The second strategy can obtain a high-quality mesh but does not maintain the interface, requiring the interface reconstruction methods to reconstruct interface information and then to calculate the mixed equations of state. To address the aforementioned issues, our idea follows the last strategy, allowing constrained movement of interface and boundary nodes during the mesh rezoning step. Unlike [27], we do not constrain the movement of interface and boundary points to the tangential direction. Instead, we aim to construct a high-order curve connecting these grid nodes, enabling the interface and boundary points to move along this high-order curve. This approach preserves the material interface, minimizes the generation of mixed cells, avoids calculating the mixed equations of state, improves mesh quality near the interface and minimizes the computational error. In addition, we can also apply the same treatment to boundary points, making our method applicable to moving boundary problems as well.

Our rezoning algorithm is based on optimization methods. We propose a simple and efficient objective function that ensures the rezoned mesh maintains good geometric quality while staying as close as possible to the Lagrangian mesh. Additionally, using optimization methods provides a convenient way to handle various constraints, facilitating the preservation of interfaces and boundaries.

The outline of our paper is as follows. In Section 2, we will introduce how we perform the geometric essentially non-oscillatory (GENO) interpolation method on interface and boundary points. In Section 3, we will discuss our optimization algorithm, including the design of the objective function, the optimization method, and how to handle interface and boundary points during the optimization process. In Section 4, we will describe our rezoning strategy, which determines when to perform a rezoning step during the ALE simulation. In Section 5, we will provide test examples of perturbed meshes and ALE simulations, including single-material and multi-material simulations. Finally, in the last section, we will draw our conclusions.

2 The interface interpolation algorithm

As mentioned above, if the mesh nodes on the interface are fixed while other nodes are optimized in multi-material simulations, this can lead to better results compared to a purely Lagrangian approach. However, due to the poor mesh quality near the interface, multi-material simulations often encounter failures. Therefore, we aim to move the interface nodes reasonably during mesh optimization, allowing them to move along the interface. To achieve this, we first need to obtain the interface curve.

Since the mesh nodes on the interface always follow the fluid motion and carry the interface information from the initial moment in the Lagrangian step, we consider the interface to be a smooth or at least piecewise smooth curve connecting these nodes. Therefore, we can obtain the corresponding interface curve by performing high-order interpolation on these mesh nodes.

In [27], in order to ensure the interface and boundary points move along the tangential direction, first-order linear interpolation is used to reconstruct the interface and boundary lines. As mentioned earlier, the interface should ideally be a smooth or piecewise smooth curve connecting all the interface nodes. Compared to the first-order linear interpolation, higher-order interpolation is more consistent with the physical properties of the interface. However, in the computation of two-dimensional fluid problems, the fluid structures sometimes can be very complex. For example, the discontinuities and multi-valued issues may arise in some local regions, in these cases, using higher-order interpolation methods may pose significant challenges.

To solve these problems, we consider the GENO method [28], which is originally applied to sub-pixel interpolated curve evolution. Unlike polynomial interpolation methods, GENO is a geometric interpolation method, meaning the interpolation curve is constructed from a series of geometric shapes rather than polynomial functions. Consequently, geometric interpolation methods naturally avoid the multi-valued issues.

The essentially non-oscillatory (ENO) method [13] selects the smoother stencil for in-

terpolation by comparing the divided differences of the left and right stencils, making it well-suited for handling discontinuous problems. GENO interpolation builds on the concept of ENO interpolation, the main difference is that GENO method uses a series of basic geometric curves for interpolation instead of polynomial functions.

In the second order GENO interpolation method, we use straight lines and circles as the basic geometric building blocks. In this case, we can no longer use divided differences to judge the smoothness of the stencils between both sides. Instead, we use curvature to evaluate smoothness. The curvature of a circle is equal to the reciprocal of its radius, and a straight line can be considered as a circle with an infinite radius, thus having zero curvature. When selecting the geometric interpolation stencil in GENO, we always choose the stencil with the smaller curvature. For example, in Figure 2, suppose points $P_{i-\frac{1}{2}}$ and $P_{i+\frac{1}{2}}$ are the two initial points, and points $P_{i-\frac{3}{2}}$ and $P_{i+\frac{3}{2}}$ are the candidate points on either side to add to the stencil. We calculate the radii of the circles passing through the points $\{P_{i-\frac{3}{2}}P_{i-\frac{1}{2}}P_{i+\frac{1}{2}}\}$ and $\{P_{i-\frac{1}{2}}P_{i+\frac{1}{2}}P_{i+\frac{3}{2}}\}$ respectively. By comparing the radii, we select the circle with the larger radius (smaller curvature) to serve as the geometric interpolation curve between $P_{i-\frac{1}{2}}$ and $P_{i+\frac{1}{2}}$. If a point on one side lies on the straight line with the initial points, as shown in the right subfigure of Figure 2, we choose a straight line as the geometric interpolation curve because the straight line has zero curvature.

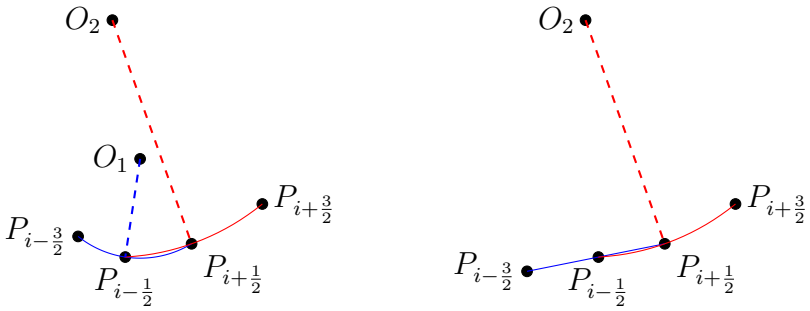


Figure 2: The stencil of the geometric ENO interpolation

Since GENO is a geometric interpolation method, the interpolated curve is composed of a series of geometric shapes. Therefore, after completing the GENO procedure, we record the geometric features of these shapes for each interval. For circles, we record their radius,

arc length, and center coordinates, while for straight lines, only the segment length needs to be recorded. With these geometric features, we only need to provide the arc length $\tilde{s}$ of a point, then we can get its corresponding coordinates.

We only use straight lines and circles for the geometric interpolation, which provides a second-order interpolation. One could choose Euler spirals and even higher-order spirals as further building blocks to get higher-order geometric interpolation curves, just like polynomial interpolation with higher degree polynomials. However this is unnecessary for our purpose, because the Lagrangian method for straight-line edged meshes has at most second-order accuracy [4], and we do not consider curved meshes in this paper.

After obtaining the corresponding geometric interpolation functions through the GENO geometric interpolation method, we can maintain the interface through keeping the mesh nodes on the interfaces to always move along the corresponding interpolated interface curves during the mesh rezoning process, similarly for the boundary nodes. The details of this procedure is described in Section 3.2.1.

The procedure for the GENO interpolation is described as follows in Algorithm 1.

Algorithm 1 The GENO interpolation procedure

Input: the coordinates $\boldsymbol{x}_i, i = 1, 2, \ldots, n+1$ of all mesh nodes on an interface or a boundary line in sequential order

1 define the interval based on the order of the input nodes, $I_i = \{\boldsymbol{x}_i, \boldsymbol{x}_{i+1}\}, i = 1, 2, \ldots, n$, for the first interval I_1, take $\{I_1, I_2\}$ as the stencil, while for the last interval I_n, we take $\{I_{n-1}, I_n\}$ as the stencil;

for $i = 2, 3, \ldots, n-1$ **do**

2 calculate the curvatures of the geometric interpolation curves for stencil $\{I_{i-1}, I_i\}$ and $\{I_i, I_{i+1}\}$, choose the one with smaller curvature;

3 for circle, compute and record the center (x_i, y_i), the radius r_i, and the arc length $\tilde{s}_i$; for straight line, simply record the length of the segment $\tilde{s}_i$.

end for

Output: Geometric information of the interpolation curve

3 Mesh optimization

In this section, we introduce the mesh optimization method, which applies to two-dimensional structured quadrilateral meshes. We will first present the objective function we have designed, followed by the optimization method and some additional details.

3.1 The objective function

In optimization problems, the objective function is essentially the mathematical expression of the functionality we want to achieve. In our mesh optimization process, we aim to achieve the following two goals:

1. Ensure the rezoned mesh is close to the original Lagrangian mesh as much as possible, in order to reduce the remapping errors and achieve better resolution;

2. Enhance the geometric quality of the rezoned mesh, allowing the ALE fluid computation to proceed for longer time.

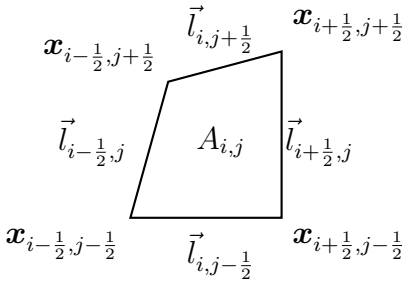


Figure 3: A quadrilateral cell with notations

Let us clarify some notations. N_x and N_y are the numbers of mesh cells in the x and y directions. One can see other notations in Figure 3, where $A_{i,j}$ represents the (i,j)-th cell, $\boldsymbol{x} = (x, y)$ represents the coordinate, $\vec{l}$ is the vector of the edge, the four edges and the coordinates of the four mesh nodes surrounding the cell $A_{i,j}$ are shown in Figure 3. The symbol $|A_{i,j}|$ represents the area of the cell $A_{i,j}$, and $|\tilde{A}_{i,j}|$ represents the area of the corresponding original Lagrangian cell. With the notations established, below we can

introduce the objective function (3.1) for our optimization method:

$$F = \sum_{i,j=1}^{N_x,N_y} \left[\omega \left(\frac{|\tilde{A}_{i,j}|}{|A_{i,j}|} + \frac{|A_{i,j}|}{|\tilde{A}_{i,j}|} \right) + \frac{1}{4} \sum_{k=1}^{4} \frac{|\vec{l}_{i+hx_k,j}|^2 + |\vec{l}_{i,j+hy_k}|^2}{\vec{l}_{i+hx_k,j} \times \vec{l}_{i,j+hy_k}} \right], \tag{3.1}$$

$$(hx_k, hy_k) = \begin{cases} (-\frac{1}{2}, -\frac{1}{2}) & \text{if } k = 1, \\ (\frac{1}{2}, -\frac{1}{2}) & \text{if } k = 2, \\ (\frac{1}{2}, \frac{1}{2}) & \text{if } k = 3, \\ (-\frac{1}{2}, \frac{1}{2}) & \text{if } k = 4. \end{cases} \tag{3.2}$$

The objective function (3.1) consists of two terms. The first term is the sum of the ratio of the Lagrangian mesh area to the rezoned mesh area and the ratio of the rezoned mesh area to the Lagrangian mesh area. This term reaches its minimum value when the areas of the Lagrangian mesh and the rezoned mesh are equal. Its primary purpose is to ensure that the rezoned mesh remains as close as possible to the Lagrangian mesh obtained from the Lagrangian step, thereby minimizing remapping errors and preserving the fluid characteristics reflected in the Lagrangian mesh.

The second term in the objective function (3.1) is the sum of the squares of the two adjacent edges divided by the sum of their cross product. This term is actually derived from the Laplace equations in Winslow's work [32] which was used to generate equilateral triangle mesh. The derivation process can be found in [18], where it is referred to as the global Winslow functional. Similar forms of this term are also used in [12, 31], and they are called the condition number mesh relaxation in these works.

Notice that the cross product can be expressed as

$$\vec{l}_{i+hx_k,j} \times \vec{l}_{i,j+hy_k} = |\vec{l}_{i+hx_k,j}| \cdot |\vec{l}_{i,j+hy_k}| \sin\theta, \tag{3.3}$$

where θ is the angle between the two edges. Substituting this expression (3.3) into the second term in the objective function (3.1) and simplifying, we can obtain

$$\frac{|\vec{l}_{i+hx_k,j}|^2 + |\vec{l}_{i,j+hy_k}|^2}{\vec{l}_{i+hx_k,j} \times \vec{l}_{i,j+hy_k}} = \Big(\frac{|\vec{l}_{i+hx_k,j}|}{|\vec{l}_{i,j+hy_k}|} + \frac{|\vec{l}_{i,j+hy_k}|}{|\vec{l}_{i+hx_k,j}|} \Big) \frac{1}{\sin\theta}. \tag{3.4}$$

When the two adjacent edges are equal and the angle $\theta = 90°$, the second term takes the minimum value. Therefore, the goal of this term is to make the quadrilateral elements as close to squares as possible, in order to achieve good geometric quality of the mesh. The coefficient $\frac{1}{4}$ in front of the second term is used to obtain an average from the summation.

Additionally, there is a weight factor ω in front of the first term in (3.1). If ω is small, the second term becomes more dominant, improving the geometric quality of the rezoned mesh. Conversely, if ω is large, the first term dominates, causing the rezoned mesh to more closely resemble the Lagrangian mesh.

3.2 The optimization method

During the optimization process, we aim to minimize the objective function (3.1) while ensuring that the interface and boundary points move along their corresponding interpolated curves. Conventional constrained optimization methods, such as the Lagrange multiplier method, penalty function method, or interior-point method, may result in significant computational cost. In contrast, our approach implicitly incorporates these constraints into the gradient calculation, avoiding additional computational cost and significantly saving computational costs.

We adopt the conjugate gradient method (see [23]) as our optimization approach. Unlike Newton and quasi-Newton methods, the conjugate gradient method does not require the computation of the exact or approximate Hessian matrix, making it significantly more resource-efficient, particularly for large-scale mesh optimization problems.

The iteration formula for the conjugate gradient method is shown below:

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k + \alpha_k \boldsymbol{d}_k, \tag{3.5}$$

$$\boldsymbol{d}_{k+1} = -(\boldsymbol{\nabla F})^{k+1} + \beta_k \boldsymbol{d}_k, \tag{3.6}$$

all italic bold symbols represent vectors, and k denotes the iteration step, $k = 0, 1, 2, \ldots$, $\boldsymbol{X}_k$ is the vector consisting of coordinates of all mesh nodes at the k-th iteration. $\boldsymbol{d}_k$ is

the search direction at the k-th iteration step, with $d_0 = -(\boldsymbol{\nabla F})^0$ at the beginning of the iteration. $(\boldsymbol{\nabla F})^k$ is the gradient of the objective function (3.1) with respect to the mesh nodes at the iteration k. α_k is the step size, determined by the line search method. β_k is the coefficient in the conjugate gradient method, where different values of β_k correspond to different choices of the conjugate gradient direction, for example:

1. Fletcher-Reeves:
$$\beta_{FR} = \frac{|(\boldsymbol{\nabla F})^{k+1}|^2}{|(\boldsymbol{\nabla F})^k|^2}, \tag{3.7}$$
2. Polak-Ribiere:
$$\beta_{PR} = \frac{(\boldsymbol{\nabla F})^{k+1}\big((\boldsymbol{\nabla F})^{k+1} - (\boldsymbol{\nabla F})^k\big)}{|(\boldsymbol{\nabla F})^k|^2}, \tag{3.8}$$
3. Hestenes-Stiefel:
$$\beta_{HS} = \frac{(\boldsymbol{\nabla F})^{k+1}\big((\boldsymbol{\nabla F})^k - (\boldsymbol{\nabla F})^{k-1}\big)}{\boldsymbol{d}_k\big((\boldsymbol{\nabla F})^k - (\boldsymbol{\nabla F})^{k-1}\big)}. \tag{3.9}$$

In our computation, the third conjugate gradient direction (3.9) is set as the default direction.

To optimize the objective function (3.1) using the conjugate gradient method, we need to determine the step size α_k and the gradient of the objective function (3.1) at each iteration. Therefore, we will separately explain the gradient computation and the line search procedure in the following subsections.

3.2.1 Gradient calculation

The most straightforward way to compute the gradient of the objective function (3.1) is to derive its analytical formula through precise differentiation. However, this approach faces several issues.

First, deriving the exact gradient expression of (3.1) can be quite cumbersome, especially when the problem involves multiple interfaces and boundaries, which often requires deriving several gradient expressions. Second, since the interface curves are obtained through a GENO interpolation, their functional gradient expressions are usually complex. Additionally,

different problems may have different interfaces, and manually deriving gradient expressions for each problem would reduce the flexibility of the algorithm.

Considering all these factors, we choose to use the numerical approximation

$$\frac{\partial F}{\partial x} = \frac{F(\ldots x+\epsilon, y, \ldots) - F(\ldots x, y, \ldots)}{\epsilon}, \tag{3.10}$$

$$\frac{\partial F}{\partial y} = \frac{F(\ldots x, y+\epsilon, \ldots) - F(\ldots x, y, \ldots)}{\epsilon}, \tag{3.11}$$

to obtain the gradient for the normal mesh nodes of the objective function (3.1), where ϵ is set to 10^{-7}. In [18, 27], this numerical approximation method to calculate the gradients was also used.

The advantage of numerical approximation over analytical formulas lies in its reduced computational cost. Additionally, since the optimal search direction is unknown, minor errors in the numerical approximation of the gradient will have little impact on the overall speed of the optimization iteration [18].

For points on the interface and boundary, the situation is slightly more complex. For these points, they cannot move freely in all directions but only move along a curve, meaning they have only one degree of freedom in their direction of movement. Therefore, we cannot use formulas (3.10) and (3.11) to calculate the gradient $(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y})$. Instead, we should select other variables to reduce the degrees of freedom.

In Section 2, we have obtained the geometric interpolation function for each interface via the GENO interpolation method. We can determine the coordinates of a point by simply providing its arc length $\tilde{s}$, so we can naturally compute the numerical partial derivative of the objective function (3.1) with respect to the arc length $\tilde{s}$ at these points,

$$\frac{\partial F}{\partial \tilde{s}} = \frac{F(\ldots \tilde{s}+\epsilon, \ldots) - F(\ldots \tilde{s}, \ldots)}{\epsilon}, \tag{3.12}$$

thus limiting the direction to only one degree of freedom, ensuring that interface and boundary grid points always move along the interface or boundary curves.

Additionally, it is important to note that for the corner points that determine the physical shape of the computation area, as well as the triple points where interfaces intersect, these

points have zero degrees of freedom and are considered fixed points that cannot be moved during the optimization process. For these points, we only need to consistently set their gradients to zero during the gradient computation procedure.

3.2.2 Line search algorithm

Our line search algorithm aims to achieve two goals:

1. The step size α_k should decrease the value of the objective function (3.1) as much as possible, ideally reaching its minimum.

2. Any quadrilateral element in the updated mesh $\boldsymbol{X}_k + \alpha_k \boldsymbol{d}_k$ based on the step size α_k must have a positive oriented area and cannot be concave.

For quadrilateral grid elements, we check the oriented areas of the two pairs of triangles formed by their diagonals. Assume we have a triangle with vertices (x_0, y_0), (x_1, y_1), (x_2, y_2) ordered counterclockwise, the oriented area s_{oa} is given by the following formula,

$$S_{oa} = \frac{1}{2} \begin{vmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{vmatrix}. \tag{3.13}$$

If all triangles in a quadrilateral element have positive oriented areas, the quadrilateral element is convex with a positive oriented area, as shown by the quadrilateral $\square_{ABCD}$ in the left picture in Figure 4. If any triangle has a negative oriented area (like the red triangle $\triangle_{BCD}$ in the middle picture in Figure 4), then the quadrilateral element becomes concave. If two triangles have negative oriented areas (like the two red triangles $\triangle_{ABC}$ and $\triangle_{BCD}$ in the right picture in Figure 4), the quadrilateral element is tangled. We must ensure that all triangles in the quadrilateral element have positive oriented areas in our procedure.

At the beginning of the line search, we select the length of the shortest edge in the mesh as the initial step size α_k and calculate the objective function value F_0 for the initial mesh. Then, we compute the oriented areas of the triangular elements formed by the diagonals of all quadrilateral elements in the updated mesh $\boldsymbol{X}_k + \alpha_k \boldsymbol{d}_k$ using equation (3.13). If any of these oriented areas are negative, we reduce the step size α_k by half, setting $\alpha_k = \frac{1}{2}\alpha_k$, and repeat this process until no element has a negative oriented area triangle. At this point, we

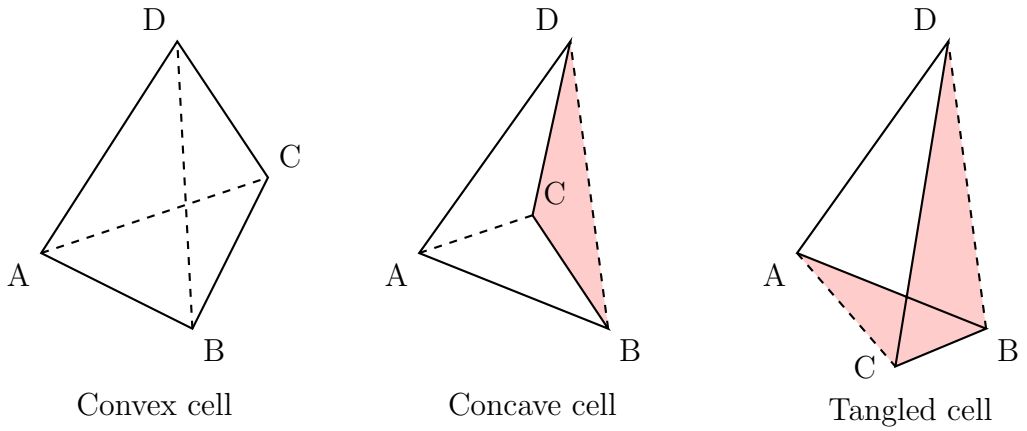


Figure 4: Different oriented area cases of quadrilateral

have obtained an updated mesh within the feasible domain:

$$\Omega = \{\ S_{oa}^{i,j,k} > 0 \mid \forall i = 1, 2, \ldots, N_x, \forall j = 1, 2, \ldots, N_y, \forall k = 1, 2, 3, 4\}. \tag{3.14}$$

Next, we compute the value of the objective function (3.1) for the updated mesh within the feasible region Ω, denoted as $F_1 = F(\boldsymbol{X}_k + \alpha_k \boldsymbol{d}_k)$. If $F_1 \geq F_0$, we halve the step size α_k and repeat this process until $F_1 < F_0$. At this point, we set $F_0 = F_1$ and reduce the step size to $\frac{1}{2}\alpha_k$, then calculate the new objective function value $F_1 = F(\boldsymbol{X}_k + \frac{1}{2}\alpha_k \boldsymbol{d}_k)$. If F_1 is still smaller than F_0, we repeat this operation until $F_1 \geq F_0$. The step size α_k that corresponds to the objective function value F_0 is the one we need.

Else, if $F_1 < F_0$, we set $F_0 = F_1$, then double the step size $\alpha_k = 2\alpha_k$ to compute the new objective function value $F_1 = F(\boldsymbol{X}_k + \alpha_k \boldsymbol{d}_k)$. We continue to increase the step size until $F_1 > F_0$ or the updated mesh is outside the feasible domain. At this point, the step size α_k that corresponds to the objective function value F_0 is the step size we need.

The algorithm for the line search method is shown in Algorithm 2.

Algorithm 2 The line search program

Input: the iteration mesh, the conjugate gradient direction

1 calculate the objective function F_0 of the input mesh;

2 set the length of the shortest edge as the initial value of the step size α_k, if it is outside the feasible domain, continuously reduce this step size until it is in the feasible domain;

3 calculate the objective function F_1 for this step size;

if $F_1 \geq F_0$ **then**

  while $F_1 > F_0$ **do**

    4 halve the step size α_k, calculate the new objective function value F_1;

  end while

  while $F_1 < F_0$ **do**

    5 set $F_0 = F_1$, continue halving the step size, and calculate the new objective function value F_1;

  end while

else

  while $F_1 < F_0$ & mesh $\in \Omega$ **do**

    6 set $F_0 = F_1$, double the step size to compute the new objective function value F_1;

  end while

end if

Output: the step size α_k

3.2.3 Optimization algorithm flowchart

To summarize, the overall algorithm for optimizing the mesh while keeping the interface and boundary mesh points moving only along the interface or boundary is shown in Algorithm 3. The input variables include the Lagrangian step mesh and the indices of interface (boundary) points to help the program identify whether the mesh points are located on the interface or boundary. The output variable is the rezoned mesh. The iteration stopping criteria are set to when any of the following three conditions are met:

1. The total number of the iterations exceeds 500.

2. The maximum absolute value of the gradient components is less than 10^{-2}.

3. The difference between two consecutive objective function values is less than 10^{-4}.

Algorithm 3 The overall optimization program

Input: Lagrangian mesh, the indices of interface and boundary nodes

1 perform GENO interpolation for the interface and boundary points;

2 check each mesh node to see if it is an interface (boundary) node or not. If it is, transform the corresponding coordinates (x, y) into arc length $\tilde{s}$;

3 calculate the initial value of the objective function F_0 and the initial numerical gradient $\boldsymbol{\nabla F}_0$, set the initial search direction to be the negative gradient direction $\boldsymbol{d}_0 = -\boldsymbol{\nabla F}_0$;

for $N_{iter} \leq 500$ or $\max|\boldsymbol{\nabla F}_k| > 10^{-2}$ or $|\boldsymbol{F}_{k+1} - \boldsymbol{F}_k| > 10^{-4}$ **do**

4 call the line search algorithm to obtain the step size α_k in the search direction, then compute the next updated mesh using the formula (3.5);

5 compute the new mesh's gradient $\boldsymbol{\nabla F}_{k+1}$ and obtain the new conjugate gradient direction $\boldsymbol{d}_{k+1}$ using the formula (3.6);

end for

6 Transform the arc length $\tilde{s}$ into corresponding coordinates (x, y) if the node is an interface or boundary node;

Output: Rezoned mesh

4 Arbitrary Lagrangian-Eulerian simulation

As mentioned above, the indirect ALE simulation consists of three steps: the Lagrangian step, the rezone step and the remap step. Here, we use the high-order positivity-preserving Lagrangian method [4] and the intersection-based remapping method [20] in the indirect ALE simulation. As for the rezone step, we employ our interface preserving mesh optimization method.

In this part, we solve the Euler equation,

$$\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x + \mathbf{g}(\mathbf{u})_y = \mathbf{0}, \tag{4.1}$$

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{bmatrix} \tag{4.2}$$

within the indirect ALE framework. Here ρ is the density, u is the velocity in the x direction, v is the velocity in the y direction, and p is the pressure, according to the state of equation in ideal gas,

$$p = (\gamma - 1)\rho e, \tag{4.3}$$

where γ is the specific heat ratio and e is the internal energy. E is the total energy, and we have $E = \rho e + \frac{1}{2}\rho(u^2 + v^2)$.

In this paper, we adopt the finite volume scheme framework on structured quadrilateral meshes. Specifically, we consider partitioning the computational domain Ω into $N_x \times N_y$ cells $\{A_{i,j}\}_{i,j}^{N_x,N_y}$. Here, we employ a cell-centered finite volume scheme, where the cell averages of the conservative physical variables $\{\bar{\mathbf{u}}_{i,j}\}_{i,j=1}^{N_x,N_y}$ are defined at the cell centroids, and the fluid velocities are defined at the nodes $\{\boldsymbol{x}_{i+\frac{1}{2},j+\frac{1}{2}}\}_{i,j=0}^{N_x,N_y}$.

4.1 The Lagrangian method

We can rewrite the Euler equation (4.1) in the reference frame of a moving control volume,

$$\frac{d}{dt}\int_{\Omega(t)} \mathbf{u} d\Omega + \int_{\Gamma(t)} \mathbf{F}(\mathbf{u}) d\Gamma = 0 \tag{4.4}$$

where

$$\mathbf{F}(\mathbf{u}) = \begin{pmatrix} 0 \\ pn_x \\ pn_y \\ p(un_x + vn_y) \end{pmatrix}.$$

Then we have the following semi-discrete cell-centered finite volume scheme,

$$\frac{d}{dt}\left(\bar{\mathbf{u}}_{i,j}\left|A_{i,j}\right|\right) = -\int_{\partial A_{i,j}} \widehat{\mathbf{F}}\left(\mathbf{u}^{\text{in}}, \mathbf{u}^{\text{ex}}, \boldsymbol{n}\right) dl \tag{4.5}$$

where $\mathbf{u}^{\text{in}}, \mathbf{u}^{\text{ex}}$ are the two physical variables on the interior and exterior side of the cell boundary $\partial A_{i,j}$, respectively. The advection numerical flux $\widehat{\mathbf{F}}\left(\mathbf{u}^{\text{in}}, \mathbf{u}^{\text{ex}}, \boldsymbol{n}\right)$ is consistent with the physical fluxes and we consider HLLC flux in our simulation.

1. Employing the multi-resolution WENO idea [35], reconstruct quadratic polynomials at time level $t = t^n$,

   $$\mathbf{u}_{i,j}^n(x,y) = (\rho(x,y), (\rho u)(x,y), (\rho v)(x,y), E(x,y))_{i,j}^{n,\top}$$

   where the reconstruction stencils are centered at cell $A_{i,j}$. Then utilize the positivity-preserving limiter proposed in [34] to modify the reconstructed polynomials $\mathbf{u}_{i,j}^n(x,y)$, and get the modified polynomials $\tilde{\mathbf{u}}_{i,j}^n(x,y)$.

2. Calculate the values at the quadrature points with $\tilde{\mathbf{u}}_{i,j}^n(x,y)$ on the cell boundaries for the advection numerical flux $\int_{\partial A_{i,j}^n} \widehat{\mathbf{F}}\left(\mathbf{u}^{\text{in}}, \mathbf{u}^{\text{ex}}, \boldsymbol{n}\right) dl$.

3. Calculate the fluid velocity $u_{i+\frac{1}{2},j+\frac{1}{2}}^n, v_{i+\frac{1}{2},j+\frac{1}{2}}^n$ at the node $(x_{i+\frac{1}{2},j+\frac{1}{2}}^n, y_{i+\frac{1}{2},j+\frac{1}{2}}^n)$ and update the computational mesh with time step τ,

   $$x_{i+\frac{1}{2},j+\frac{1}{2}}^{n+1} = x_{i+\frac{1}{2},j+\frac{1}{2}}^n + \tau u_{i+\frac{1}{2},j+\frac{1}{2}}^n, \quad y_{i+\frac{1}{2},j+\frac{1}{2}}^{n+1} = y_{i+\frac{1}{2},j+\frac{1}{2}}^n + \tau v_{i+\frac{1}{2},j+\frac{1}{2}}^n.$$

4. Calculate the new cell averages at the next time level $t = t^{n+1}$ with the first-order Euler forward time discretization,

   $$\overline{\mathbf{u}}_{i,j}^{n+1} = \frac{|A_{i,j}^n|}{|A_{i,j}^{n+1}|}\overline{\mathbf{u}}_{i,j}^n - \frac{\tau^n}{|A_{i,j}^{n+1}|}\int_{\partial A_{i,j}^n} \widehat{\mathbf{F}}\left(\mathbf{u}^{\text{in}}, \mathbf{u}^{\text{ex}}, \boldsymbol{n}\right) dl.$$

Notice that, here we only demonstrate the first-order Euler forward time discretization, however we use the third-order SSP Runge-Kutta time discretization by repeating the above flow chart for several times, so we can achieve high-order accuracy both in space and time. In the meantime, it have been verified that this Lagrangian method can preserve positivity without losing high-order accuracy.

4.2 The rezone strategy

In this subsection, we will show when to perform rezone and remap procedures in the ALE simulations.

In our calculations for quadrilateral meshes, we find two main factors affecting Lagrangian calculations: the length of the shortest edge in the Lagrangian mesh and the angles of the Lagrangian mesh. The former primarily influences the time step in the Lagrangian calculation; a shorter edge results in a smaller time step, which increases the overall computational time. The latter primarily affects the quality of the mesh. When an interior angle of the quadrilateral approaches 180°, the quadrilateral degenerates into a triangle or even becomes a concave quadrilateral, leading to computation failure. Therefore, a simple and straightforward idea is to perform the rezoning procedure whenever there are obvious short edges or very large angles in the mesh during the Lagrangian step.

For meshes with different scales, angles are dimensionless, while the lengths of the edges are not. To eliminate the influence of dimensionality, we do not directly use the length of the shortest edge for judgment, we use the aspect ratio instead. The aspect ratio AR is defined as the ratio of the shortest edge to the longest edge of a cell:

$$AR_{i,j} = \frac{\min\{|\vec{l}_{i,j-\frac{1}{2}}|, |\vec{l}_{i,j+\frac{1}{2}}|, |\vec{l}_{i-\frac{1}{2},j}|, |\vec{l}_{i+\frac{1}{2},j}|\}}{\max\{|\vec{l}_{i,j-\frac{1}{2}}|, |\vec{l}_{i,j+\frac{1}{2}}|, |\vec{l}_{i-\frac{1}{2},j}|, |\vec{l}_{i+\frac{1}{2},j}|\}}. \tag{4.6}$$

If the aspect ratio $AR_{i,j}$ is too small, it indicates that the quadrilateral cell $A_{i,j}$ is excessively narrow and has a relatively short edge.

The flowchart of the ALE simulation is shown below in Figure 5. First, a Lagrangian

calculation is performed on the initial mesh to obtain the corresponding Lagrangian mesh. Then, we compute the minimum aspect ratio $AR_{\min}$ and the maximum angle $Ang_{\max}$ of the Lagrangian mesh. If the rezoning conditions

$$AR_{\min} < a \text{ or } Ang_{\max} > b, \tag{4.7}$$

are not satisfied, where a and b are two adjustable parameters, the Lagrangian step is performed continuously. Otherwise, the rezoning step is executed. This is followed by the remapping step, where the physical quantities from the old Lagrangian mesh are mapped onto the new rezoned mesh. After that, we return to the Lagrangian step and continue the cycle until the final time is reached.

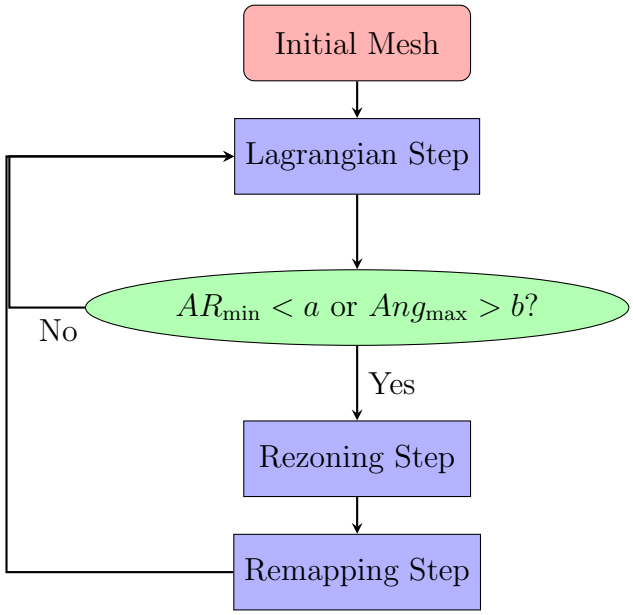


Figure 5: Flowchart of the ALE simulation

4.3 The remapping method

After the rezone step, we need to transfer cell averages $\{\bar{\mathbf{u}}_{i,j}^{n+1}\}$ on the old mesh after the Lagrangian step $\{A_{i,j}^{n+1}\}$ to the new rezoned mesh $\{\tilde{A}_{k,l}^{n+1}\}$ after the rezone step, and calculate the new cell averages $\{\bar{\tilde{\mathbf{u}}}_{k,l}^{n+1}\}$. Then we can play the Lagrangian scheme on $\{\tilde{A}_{k,l}^{n+1}\}$ with $\bar{\tilde{\mathbf{u}}}_{k,l}^{n+1}$ and finish the loop at time level $t = t^{n+1}$.

Here, we adopt the high-order positivity-preserving conservative remapping method developed in [20] to remap the conservative physical variables. This method is based on the precise calculation of each intersection between the old and new meshes, and compared to the flux-based methods, it allows for topology changes, making it suitable for multi-material hydrodynamics problems and large-scale rezoning problems. The remapping method will be divided into the following four steps.

- **Clipping** Use the Sutherland-Hodgman polygon clipping algorithm [29] to clip a target cell $\tilde{A}_{k,l}^{n+1}$ within a source cell $A_{i,j}^{n+1}$ and get the intersection $\tilde{A}_{k,l}^{n+1} \bigcap A_{i,j}^{n+1}$. This algorithm processes each edge sequentially, intersecting the polygon with each edge of the clipping region to produce the final clipped intersection.

- **Reconstruction** Reconstruct the quadratic polynomials $\{\mathbf{u}_{i,j}^{n+1}(x,y)\}_{i,j=0}^{N_x,N_y}$ via cell averages $\{\bar{\mathbf{u}}_{i,j}^{n+1}\}_{i,j=0}^{N_x,N_y}$ with the multi-resolution WENO idea which has been mentioned above. Then modify the reconstructed polynomials with the positivity-preserving limiter and obtain the modified polynomials $\{\hat{\mathbf{u}}_{i,j}^{n+1}(x,y)\}_{i,j=0}^{N_x,N_y}$.

- **Conservative remapping** Compute the cell averages for the new mesh,

$$\bar{\tilde{\mathbf{u}}}_{k,l}^{n+1} = \frac{1}{|\tilde{A}_{k,l}^{n+1}|} \sum_{i,j=1}^{N_x,N_y} \int_{A_{i,j}^{n+1} \bigcap \tilde{A}_{k,l}^{n+1}} \hat{\mathbf{u}}_{i,j}^{n+1}(x,y)dxdy.$$

  thereby completing the remapping process.

Overall, the remapping method, as outlined above, ensures the preservation of positivity for the corresponding physical variables as long as the cell averages input to the remapping method are positive. By combining the high order positivity-preserving Lagrangian method, remapping algorithm, and rezoning method, we have developed a high order positivity-preserving indirect ALE method.

5 Numerical Test

In this section, we will first test several individual perturbed meshes and then apply our method to ALE simulations. Unless otherwise specified, the weight w in the objective function (3.1) is set to 1000, and the parameters a and b in the rezoning strategy (4.7) are set to 0.1 and 145°, respectively.

5.1 The perturbed mesh tests

In this section, we construct several perturbed mesh examples to test the effectiveness of our algorithm. We first generate high-quality meshes and then randomly perturb the mesh points on the interfaces and boundaries to create the perturbed meshes. Next, we apply our optimization program to these meshes to observe whether it can improve the mesh quality, particularly near the boundaries and interfaces.

We create four perturbed mesh examples. Since these examples are artificial test meshes and there is no need to maintain the rezoned mesh close to the Lagrangian mesh as in ALE simulations, we focus solely on the geometric quality of the mesh. Therefore, we set ω to a smaller value of 15 for these cases.

The first test case is a smooth mesh obtained through trigonometric mapping. We select one edge in the middle as the interface and then randomly perturb this interface and four boundaries. The perturbed mesh restores the smooth appearance after using our method as shown in Figure 6.

The second example features relatively complex boundaries, where the uneven distribution of nodes along the boundary results in poor mesh geometric quality, especially near the boundary corners. After applying our optimization method, the mesh quality in these regions is significantly improved as shown in Figure 7.

The third test case resembles a mountain shape with the upper boundary featuring waves of varying amplitudes. The mesh quality near the upper and lower boundaries deteriorates significantly after perturbation, with many elements exhibiting large angles and extreme

aspect ratios. After optimization, the mesh quality visibly improved, as shown in Figure 8.

The fourth test case is similar to the third one, the difference is that the mesh is not Lipschitz continuous at the peak position, making it a wave-like shape. After perturbation, the mesh becomes denser near the peak and relatively sparser in other regions, resulting in poor mesh quality. After optimization, the mesh becomes smoother, and its quality improves, as shown in Figure 9.

For the examples above, the zoomed-in images clearly show that the mesh nodes located on the interface and boundary are able to move along their corresponding interface or boundary lines without altering the physical boundaries of the mesh. This not only improves the quality of the mesh in the interior but also significantly enhances the mesh quality near the boundaries and interfaces.

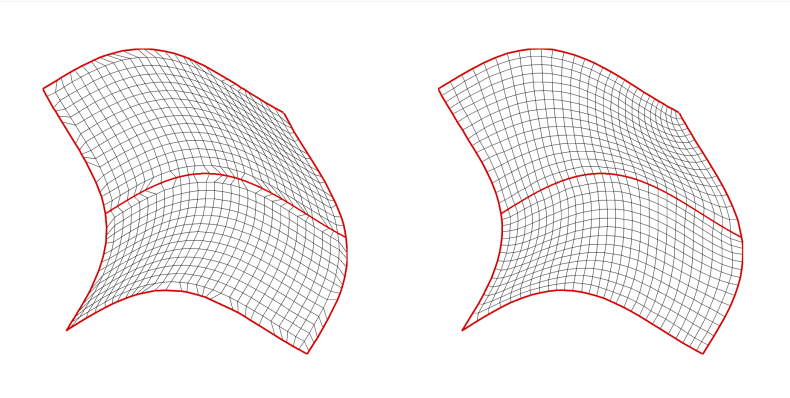


Figure 6: Perturbed mesh test 1. Left: the mesh before optimization; right: the mesh after optimization. The interface and four boundaries are marked in red.

5.2 The ALE simulation tests

5.2.1 The single-material simulation tests

The Sedov problem. The Sedov problem is a classic test case in computational fluid dynamics (CFD), which is used to simulate the propagation of strong shock waves caused

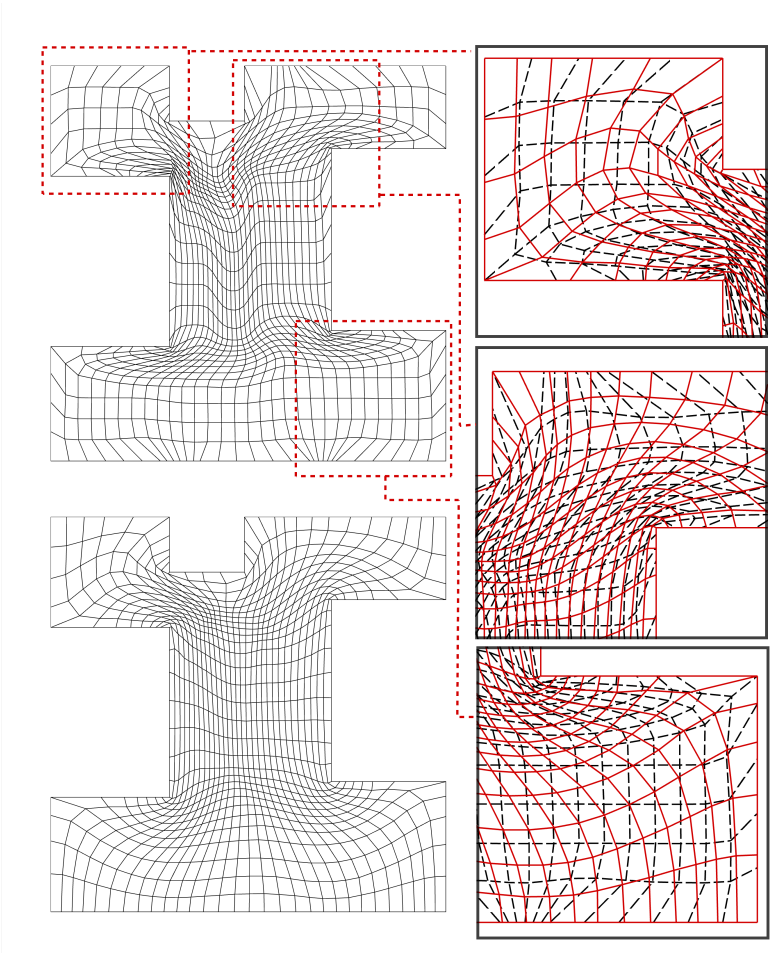


Figure 7: Perturbed mesh test 2. Top left: the mesh before optimization; bottom-left: the mesh after optimization; right: zoomed-in local regions, where the black dashed lines represent the mesh before optimization, and the red solid lines represent the mesh after optimization.

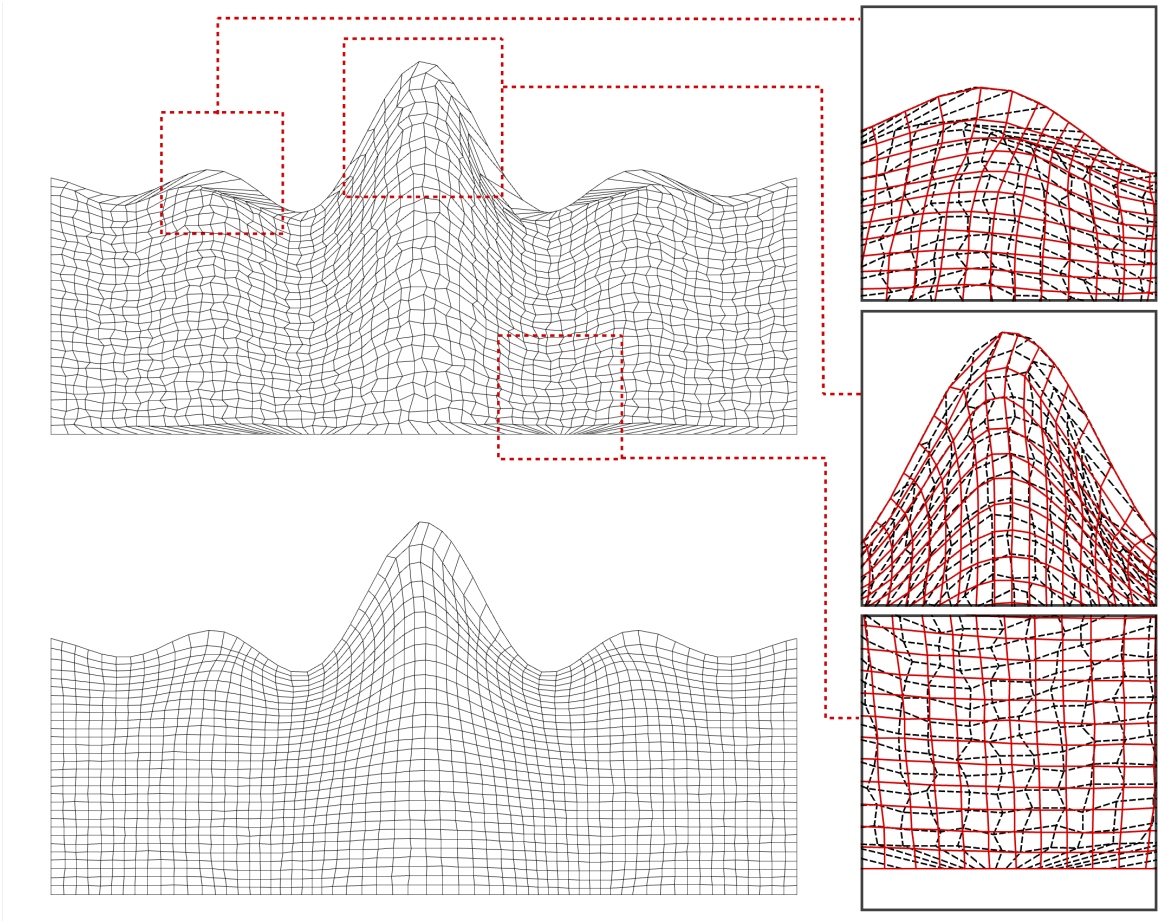


Figure 8: Perturbed mesh test 3. Top left: the mesh before optimization; bottom-left: the mesh after optimization; right: zoomed-in local regions, where the black dashed lines represent the mesh before optimization, and the red solid lines represent the mesh after optimization.

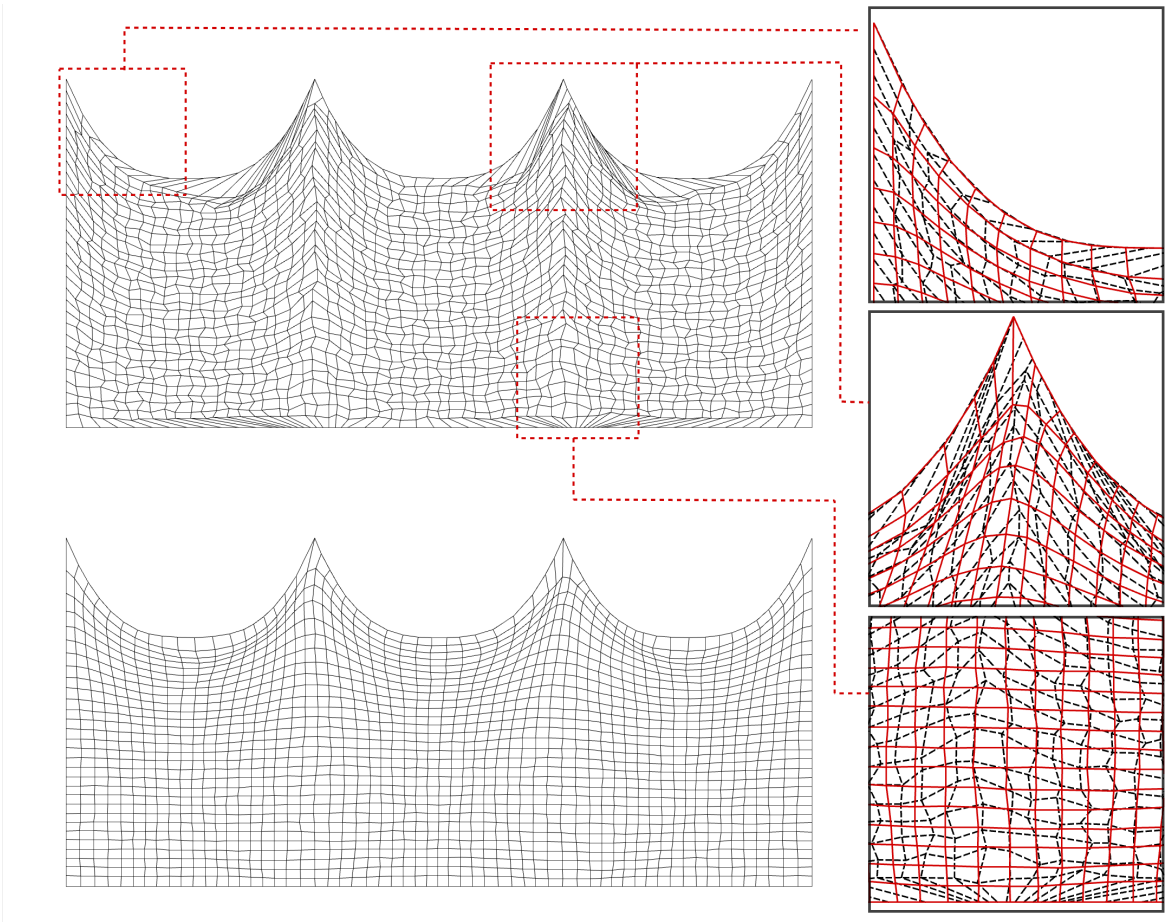


Figure 9: Perturbed mesh test 4. Top left: the mesh before optimization; bottom-left: the mesh after optimization; right: zoomed-in local regions, where the black dashed lines represent the mesh before optimization, and the red solid lines represent the mesh after optimization.

by point source explosions. The initial condition is

$$\begin{cases} \rho = 1, \\ u = 0, \\ v = 0. \end{cases} \tag{5.1}$$

The internal energy e equals 182.09 at the cell near the origin, while in other cells, the internal energy is set to 10^{-14}. The calculation domain is $[0, 1.1] \times [0, 1.1]$ and we choose a mesh size of 30×30. Reflective boundary condition is applied to all four boundaries. We use both our interface preserving method and pure Lagrangian method to calculate this problem at the final time $t = 1$ and show the density and pressure results in Figure 10. Because the rezoned mesh closely resembles the Lagrangian mesh near the shock wave, our method can preserve high resolution at the shock wave location while maintaining high geometric quality.

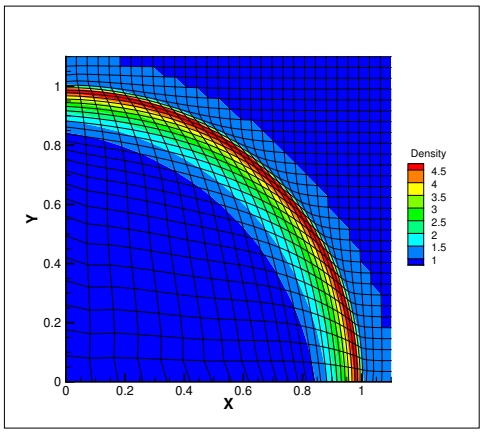
The Noh problem. The second single-material test case is the Noh problem [24], which was first proposed by Noh in 1978. The Noh problem is commonly used to assess the performance of numerical methods in solving shock wave problems and strong impact wave problems, especially in the numerical calculation of radial shock wave propagation problems. When using Cartesian meshes, the Lagrangian method often leads to severe mesh distortions in the later stage of the computation.

The initial computational domain is $[0, 1] \times [0, 1]$, and we choose the computational mesh size to be 30×30. The initial condition is

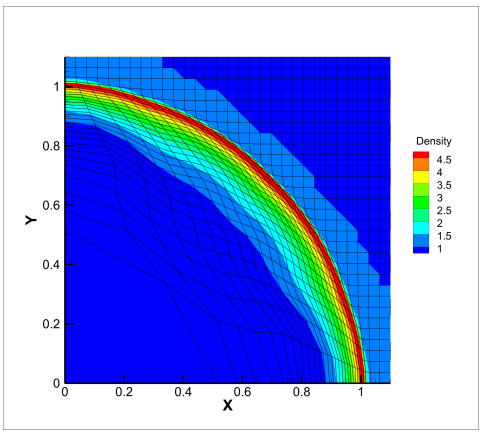
$$\begin{cases} \rho = 1, \\ u_r = -1, \\ e = 10^{-4}. \end{cases} \tag{5.2}$$

u_r is the radial velocity, specific heat ratio $\gamma = \frac{5}{3}$. The boundary conditions on the left and bottom sides are set as reflective boundary condition, while on the top and right sides are set as free boundary condition. These initial conditions will result in a spherically propagating shock wave with a constant velocity.

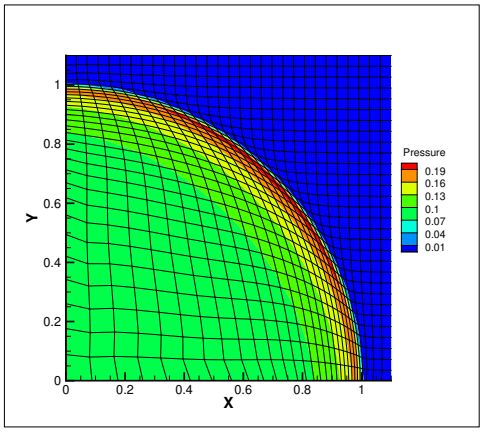
We use our method to compute up to $t = 0.6$. Since the pure Lagrangian method fails around $t = 0.54$, we only present its results up to the failure time, with the corresponding



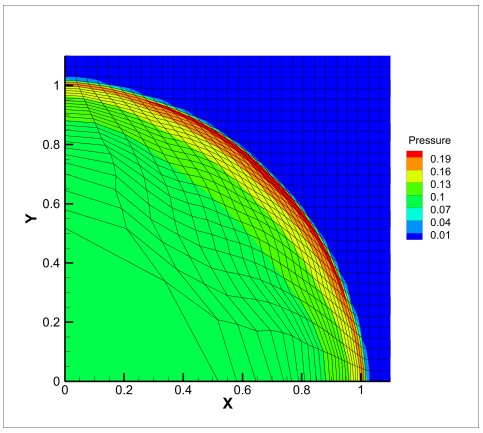
(a) density, interface preserving ALE method



(b) density, Lagrangian method



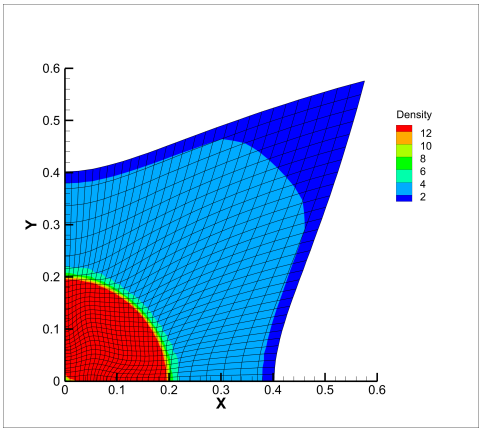
(c) pressure, interface preserving ALE method



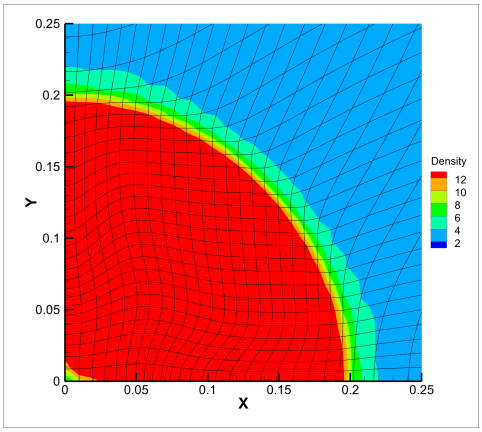
(d) pressure, Lagrangian method

Figure 10: Results of the Sedov problem, $t = 1$

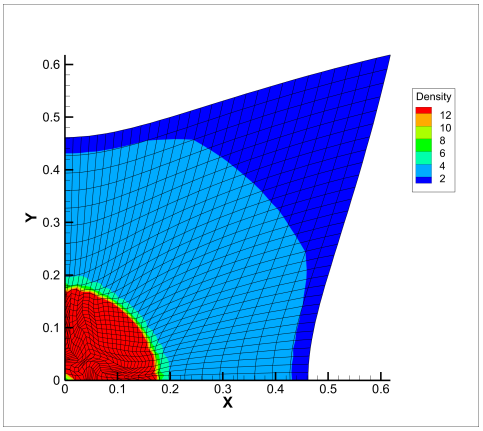
density results shown in Figure 11 below. Compared to the results of the Lagrangian method, our approach significantly improves the mesh quality inside the shock wave while maintaining good resolution near the shock. At this point, the Lagrangian method has already failed due to severe mesh deformation.



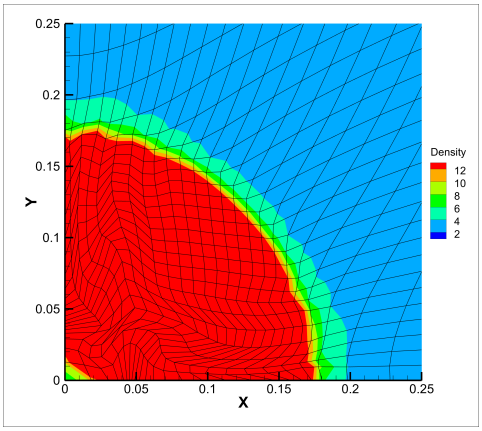
(a) density, interface preserving ALE method, $t = 0.6$



(b) zoom image



(c) density, Lagrangian method, $t = 0.54$



(d) zoom image

Figure 11: Results of the Noh problem

5.2.2 The multi-material simulation tests

We solve three multi-material problems: the spherical implosion problem, the triple point problem and the Rayleigh-Taylor instability problem. These problems are more difficult to

solve than single-material problems because of the existence of the distorted interface.

The spherical implosion problem. The first example is a spherical implosion problem, which was first calculated by Young [33]. This spherical implosion problem has some similarities with the inertial confined fusion (ICF) simulation. The light material inside the sphere is surrounded by the heavy material on the outer layer. The schematic diagram of the initial condition is shown in Figure 12. In [33], the specific heat ratio of the heavy material was also set to $\frac{5}{3}$. We set the value of the specific heat ratio γ_h to be 1.4 in order to make the problem a multi-material problem.

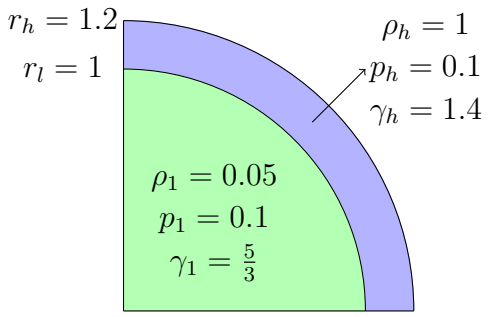


Figure 12: The initial condition of the spherical implosion problem

At the initial moment, we apply the following perturbations to the material interface,

$$r^{pert} = r * (1 + a_0 \cos 8\theta), \tag{5.3}$$

where a_0 is the perturbation coefficient and then we apply a time-varying pressure on the outer side of the sphere to start this implosion:

$$p(t) = \begin{cases} 13, & \text{if } t \in [0, 0.04] \\ 13 - 2.5\frac{t-0.04}{0.125-0.04}, & \text{if } t \in [0.04, 0.125] \\ 0.5. & \text{if } t \in [0.125, 0.3] \end{cases} \tag{5.4}$$

For this problem, the shock will continuously rebound at the center of the circle, causing the outer heavy material to slow down and enter a relatively stagnant stage, at which time the radius of the interface reaches its minimum value. Afterwards, because the interface

is perturbed at the initial moment, this perturbation will exponentially increase over time due to the existence of the Rayleigh-Taylor instability phenomenon. This Rayleigh-Taylor instability phenomenon is caused by the light material pushing the heavy material during the fluid deceleration stage.

We choose the perturbation coefficient $a_0 = 1.0 \times 10^{-3}$ and use a polar mesh with the size of 45×40. Since the Lagrangian mesh near the origin in this problem tends to tangle, therefore, we set $\omega = 80$ and $b = 120^\circ$ to achieve better mesh quality. The initial mesh division is shown in the Figure 13. In order to make the mass near the interface basically at the same level, the mesh division in the heavy material region is slightly denser compared to that in the light material. We present the density results at $t = 0.3$ and 0.35 in Figure 14, the clear interface instability phenomenon can be observed in this image. If we use the pure Lagrangian method, this problem will fail at $t = 0.24$ because of the entanglement of the computational mesh [10].

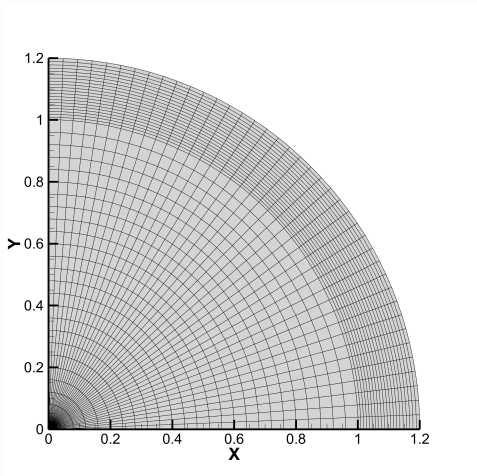
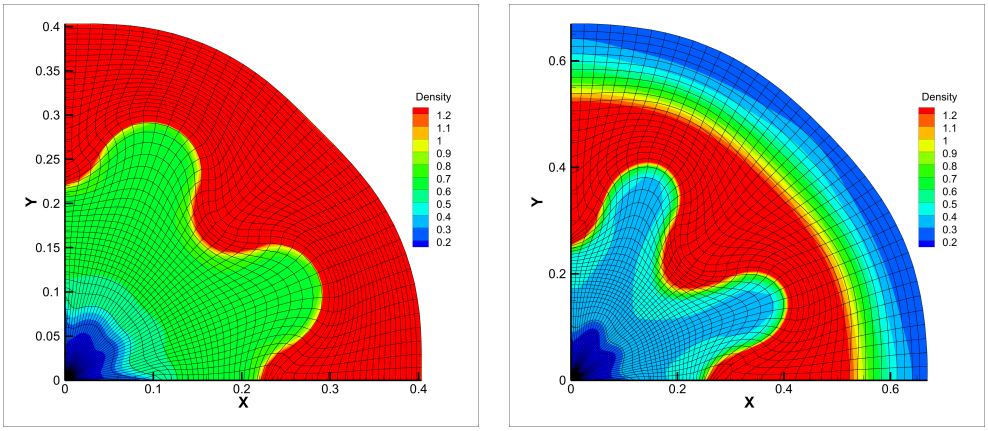


Figure 13: The initial mesh for the spherical implosion problem

The triple point problem. We first calculate the well-known triple point problem in CFD [10, 22]. This problem is a two-dimensional Riemann problem with two materials and three states and is commonly used to test the ability of numerical algorithms to handle the material interfaces of multi-material problems. The calculation area is a rectangular region



(a) $t = 0.3$ (b) $t = 0.35$

Figure 14: The spherical implosion problem, density image

of $[0, 7] \times [0, 3]$ and is divided into three sub areas, the initial condition is shown below in Figure 15.

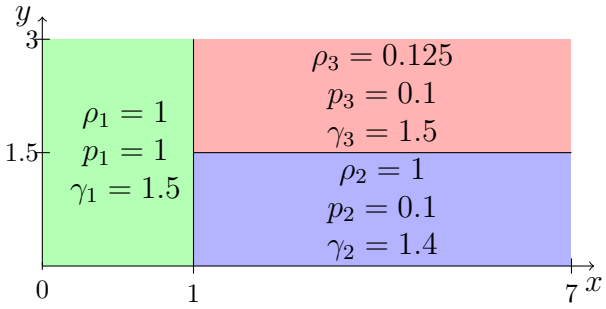
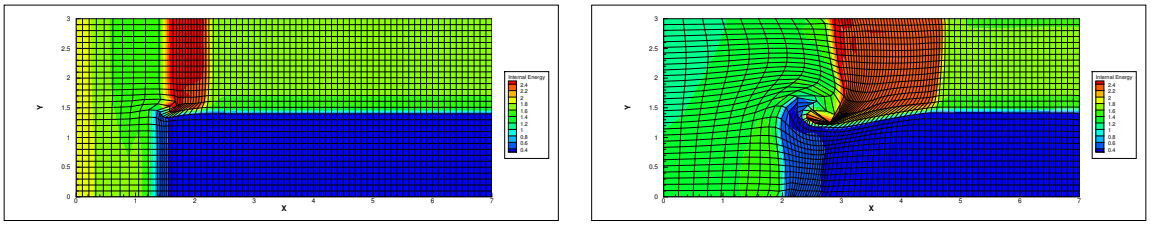


Figure 15: The initial condition of the triple point problem

The initial velocities of the three gases are all 0, and we use reflection boundary conditions on all four boundaries. Due to the different initial states of gases in the three regions, a vortex will be generated at the triple point junction of the three regions over time. And as time goes on, the shape of the vortex will become clearer, the deformation of the computational mesh will become more severe, which brings great difficulties for Lagrangian calculations.

We first attempt to use the pure Lagrangian method and the ALE method with fixed material interface points to solve this problem. It was found that for the pure Lagrangian method the simulation failed at $t = 0.72$, due to the decreased mesh quality and small time

step. The ALE method with fixed interface points would also fail at $t = 2.1$ with the same reason. The corresponding failed results are shown in Figure 16.



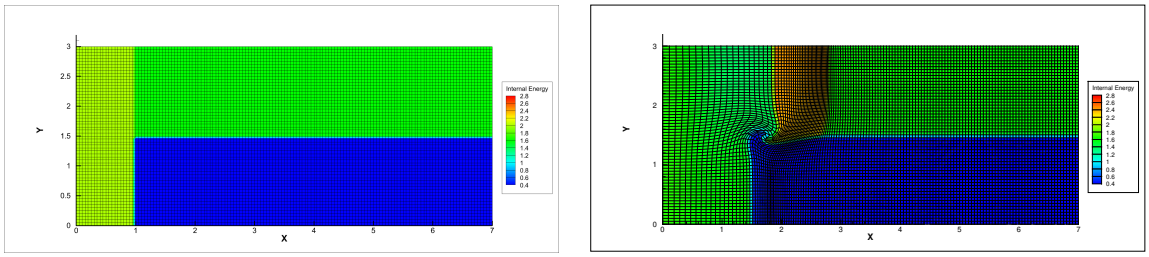
(a) pure Lagrangian method, $t = 0.72$ (b) ALE method with fixed points, $t = 2.1$

Figure 16: The triple point problem, failed results, internal energy

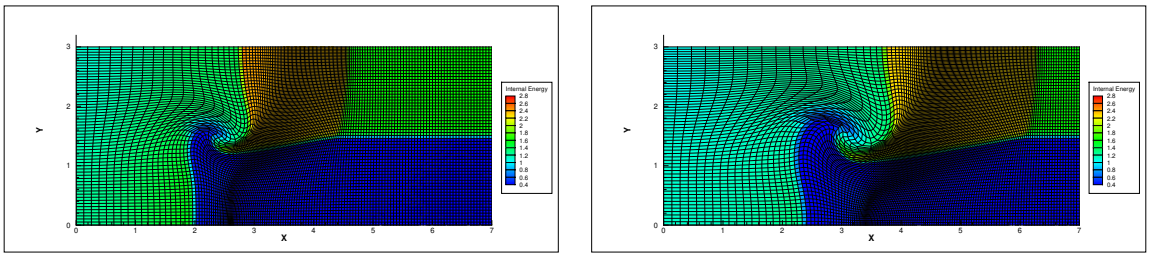
Now, using our method, we choose the mesh size of 140×60, and the results at time $t = 0, 1, 2, 3, 4, 5$ are shown in Figure 17. It can be seen that our method can successfully calculate at time $t = 5$ and display a clear vortex structure. This is because our method preserves the material interfaces, allowing interface points to move along their corresponding interface curves, which avoids the generation of mixed cells while producing high-quality meshes.

The Rayleigh-Taylor instability problem. We now calculate the Rayleigh-Taylor instability problem [19,30], which describes an unstable phenomenon in fluid mechanics. When two materials of different densities are simultaneously in a gravitational field, if the heavier material is located above the lighter material, the interface between the two materials becomes unstable. If a small perturbation is given to the interface at the initial time, the small perturbation in the interface will rapidly amplify and evolve into a very complex fluid structure over time. This instability phenomenon is widely present in various fields, including astrophysics, nuclear physics, materials mechanics, and fluid mechanics.

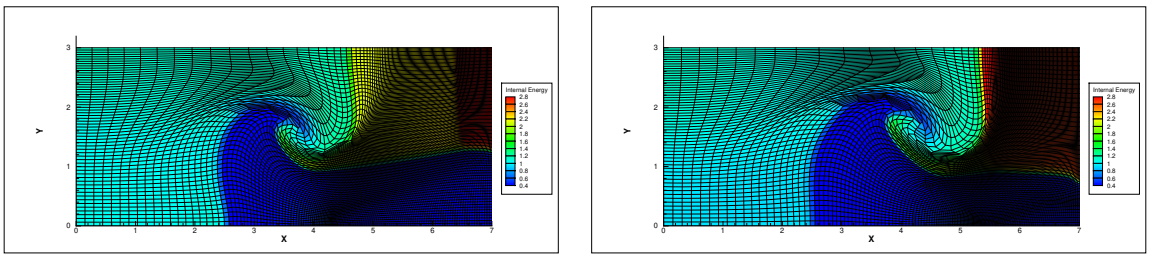
In this test case, the calculation area is $[0, \frac{1}{3}] \times [0, 1]$. The initial condition is shown in Figure 18, above $y = \frac{1}{2}$ is a relatively heavy material with a density of 2, and below is a relatively light material with a density of 1. The specific heat ratio γ of both materials is



(a) $t = 0$ (b) $t = 1$



(c) $t = 2$ (d) $t = 3$



(e) $t = 4$ (f) $t = 5$

Figure 17: The triple point Problem, internal energy, interface preserving ALE method

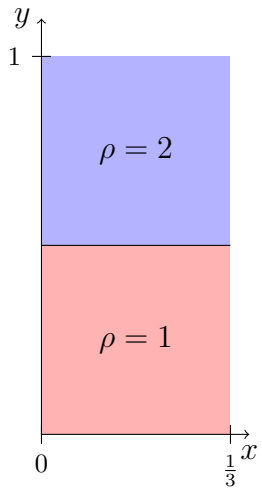


Figure 18: The initial condition of the Rayleigh–Taylor instability problem

1.4, and initially the two materials are in a stationary state, with a gravity field $g = 0.1$ which needs to be considered when calculating the total energy and momentum in the y-axis direction. p_l is the pressure of the light material and p_h is the pressure of the heavy material, they are described by

$$p_l = 1 + \frac{1}{2}\rho_h g + \rho_l g(\frac{1}{2} - y), \tag{5.5}$$

$$p_h = 1 + \rho_h g(1 - y). \tag{5.6}$$

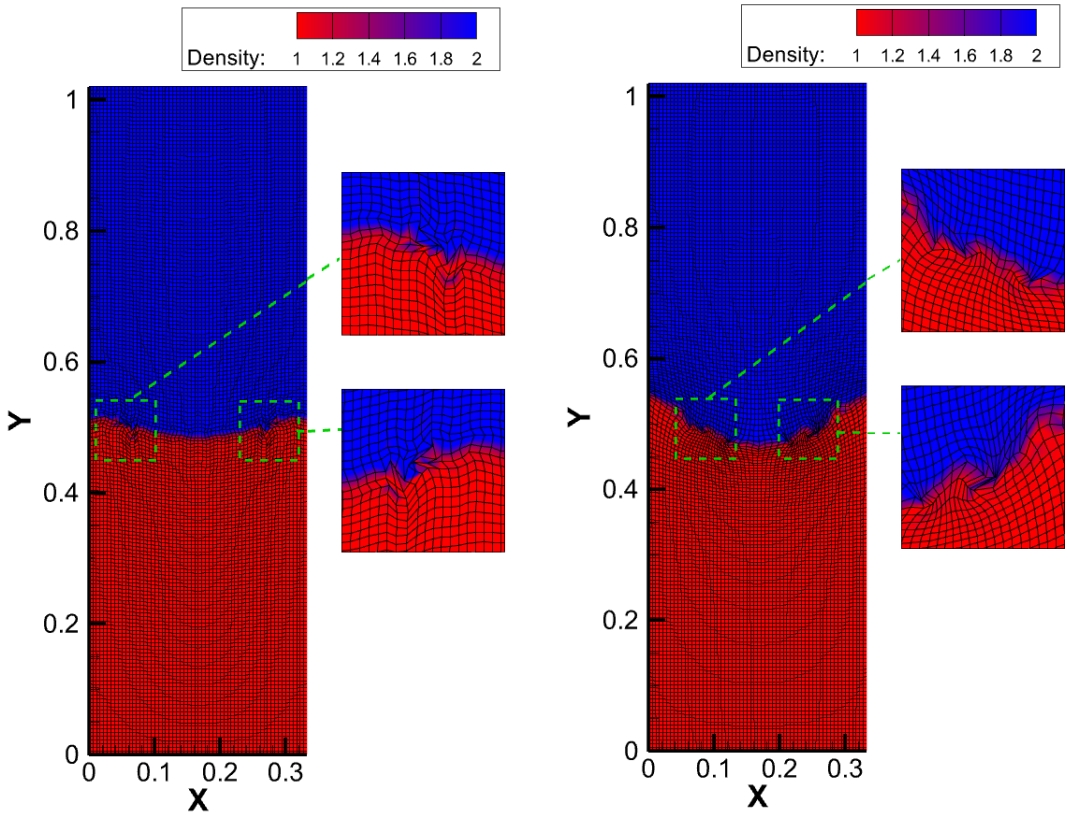
To capture the instability phenomenon of the interface, we make a small perturbation on the interface points at the initial moment, and the perturbation formula is:

$$y = \frac{1}{2} + 0.01\cos 6\pi x. \tag{5.7}$$

If the pure Lagrangian method is used for the calculation, the simulation will terminate at $t = 1.4$ due to the distortion of the mesh nearby the interface, as shown in Figure 19(a). If we use the ALE method with the interface points fixed, the simulation still terminates at $t = 2.9$ with the same reason, the result is shown in Figure 19(b).

Now we use our method with a mesh size of 60×180 for the calculation. The classic Rayleigh-Taylor instability problem is a single-material problem, the specific heat ratio is 1.4 for both upper and lower materials. We modify the value of specific heat ratio γ to make it a real multi-material problem, by taking the specific heat ratio $\gamma = 1.5$ for the light material, while keeping the specific heat ratio $\gamma = 1.4$ for the heavy material. The results at time $t = 4, 5, 6, 7$ are shown in the Figure 20.

For longer time development, the interface in this problem gradually develops into a mushroom shape and rolls up into a vortex, making it difficult for ALE simulations to continue. Using our method to preserve the interface can significantly slow the emergence of mixed cells. For this problem, we compute up to the final time $t = 7$, further simulations would experience a significant decline in accuracy due to severe mesh deformation. However, it can still be seen that our method has made significant progress in computing the Rayleigh-Taylor instability problem compared to pure Lagrangian method and the ALE method with



(a) Pure Lagrangian method, $t = 1.4$ (b) ALE method with fixed points, $t = 2.9$

Figure 19: The Rayleigh–Taylor instability problem, density image

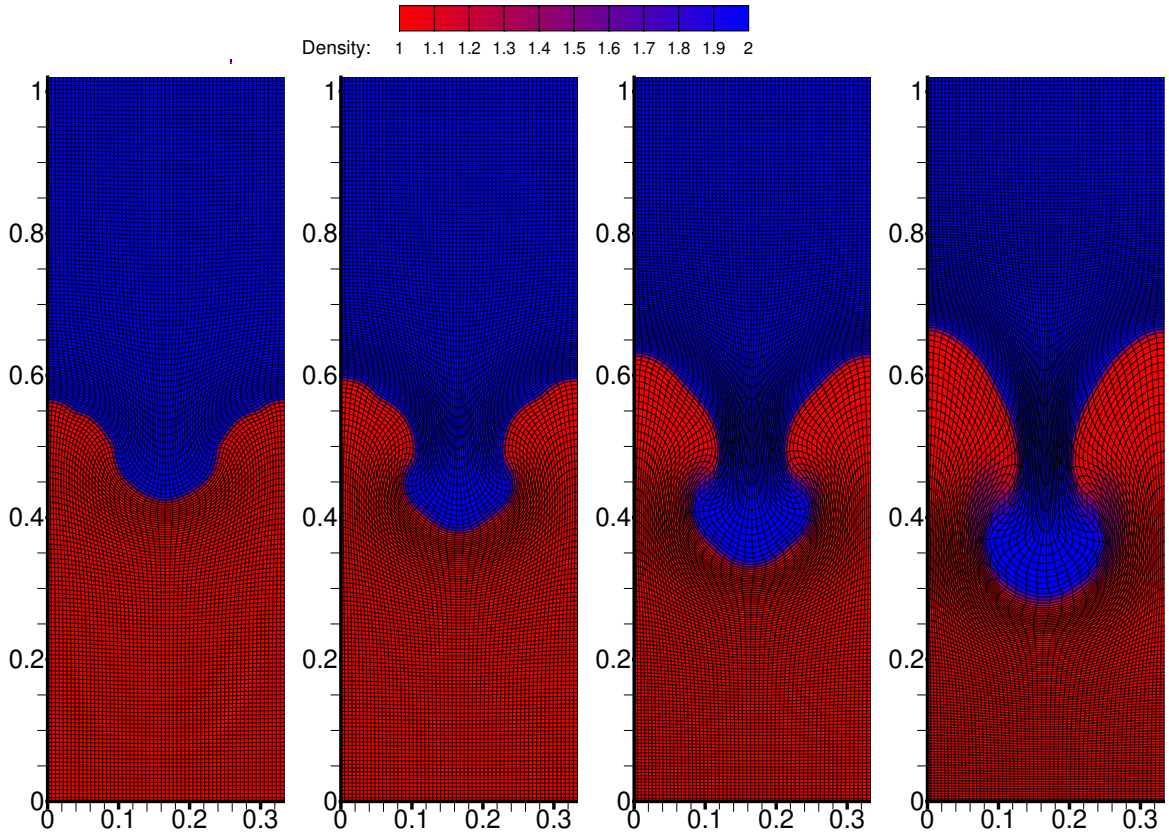


Figure 20: The Rayleigh-Taylor instability problem, $\gamma_{heavy} = 1.4$, $\gamma_{light} = 1.5$, density figure at $t = 4, 5, 6, 7$

the interface points fixed. By preserving the material interface, we have obtained meshes with good geometric quality, making the ALE simulation perform well.

6 Conclusion

For multi-material simulations in computational fluid dynamics, we have developed a mesh optimization method based on the indirect ALE approach to obtain high geometric quality meshes while maintaining the material interface. We use the GENO interpolation method to construct second-order geometric interpolation curves for interface and boundary points, ensuring that these points move along the respective interpolated interface and boundary lines during optimization. Additionally, we present a simple and efficient objective function which enables the rezoned mesh to maintain a high geometric quality while staying close to the Lagrangian mesh as much as possible. Our mesh optimization employs the classical conjugate gradient method, integrating the interface and boundary point movement algorithm into the conjugate gradient calculations, enabling the interface-preserving capability without additional computational cost. Using our approach, there is no need for VOF-based interface reconstruction in multi-material simulations, thereby also avoiding the need to handle the mixed cells. A series of numerical tests validate the effectiveness of our algorithm, particularly for multi-material cases with large interface deformations.

Currently, our work focuses on two-dimensional structured quadrilateral meshes. For ALE simulations, both the geometric quality and the topology of the mesh significantly influence the simulation's performance. In future research, we aim to extend our method to unstructured polygonal meshes to mitigate the impact of topology on computations.

References

[1] J. L. F. Aymone, E. Bittencourt and G. J. Creus, *Simulation of 3D metal-forming using an arbitrary Lagrangian–Eulerian finite element method*, Journal of Materials Processing

Technology, 110, 2001, 218-232.

[2] D. J. Benson, *An efficient, accurate, simple ALE method for nonlinear finite element programs*, Computer Methods in Applied Mechanics and Engineering, 72, 1989, 305-350.

[3] Y. Chen and S. Jiang, *An optimization-based rezoning for ALE methods*, Communications in Computational Physics, 4, 2008, 1216-1244.

[4] J. Cheng and C.-W. Shu, *Positivity-preserving Lagrangian scheme for multi-material compressible flow*, Journal of Computational Physics, 257, 2014, 143-168.

[5] M. D'Elia, D. Ridzal, K. J. Peterson, P. Bochev and M. Shashkov, *Optimization-based mesh correction with volume and convexity constraints*, Journal of Computational Physics, 313, 2016, 455-477.

[6] J. K. Dukowicz, M. C. Cline and F. L. Addessio, *A general topology Godunov method*, Journal of Computational Physics, 81, 1989, 29-63.

[7] V. Dyadechko and M. Shashkov, *Moment-of-fluid interface reconstruction*, Los Alamos Report LA-UR-05-7571, 49, 2005.

[8] V. Dyadechko and M. Shashkov, *Multi-material interface reconstruction from the moment data*, Journal of Computational Physics, 2006.

[9] D. A. Field, *Laplacian smoothing and Delaunay triangulations*, Communications in Applied Numerical Methods, 4, 1988, 709-712.

[10] S. Galera, P. H. Maire and J. Breil, *A two-dimensional unstructured cell-centered multi-material ALE scheme using VOF interface reconstruction*, Journal of Computational Physics, 229, 2010, 5755-5787.

[11] S. Giuliani, *An algorithm for continuous rezoning of the hydrodynamic grid in arbitrary Lagrangian-Eulerian computer codes*, Nuclear Engineering and Design, 72, 1982, 205-212.

[12] P. T. Greene, S. P. Schofield and R. Nourgaliev, *Dynamic mesh adaptation for front evolution using discontinuous Galerkin based weighted condition number relaxation*, Journal of Computational Physics, 335, 2017, 664-687.

[13] A. Harten, B. Engquist, S. Osher and S. R. Chakravarthy, *Uniformly high order accurate essentially non-oscillatory schemes, III*, Journal of Computational Physics, 131, 1997, 3-47.

[14] C. W. Hirt, A. A. Amsden and J. L. Cook, *An arbitrary Lagrangian-Eulerian computing method for all flow speeds*, Journal of Computational Physics, 14, 1974, 227-253.

[15] C. W. Hirt and B. D. Nichols, *Volume of fluid (VOF) method for the dynamics of free boundaries*, Journal of Computational Physics, 39, 1981, 201-225.

[16] J. Kim, T. Panitanarak and S. M. Shontz, *A multiobjective mesh optimization framework for mesh quality improvement and mesh untangling*, International Journal for Numerical Methods in Engineering, 94, 2013, 20-42.

[17] M. S. Kim and W. I. Lee, *A new VOF-based numerical scheme for the simulation of fluid flow with free surface. Part I: New free surface-tracking algorithm and its verification*, International Journal for Numerical Methods in Fluids, 42, 2003, 765-790.

[18] P. Knupp, L. G. Margolin and M. Shashkov, *Reference Jacobian optimization-based rezone strategies for arbitrary Lagrangian Eulerian methods*, Journal of Computational Physics, 176, 2002, 93-128.

[19] H. J. Kull, *Theory of the Rayleigh-Taylor instability*, Physics Reports, 206, 1991, 197-325.

[20] N. Lei, J. Cheng and C.-W. Shu, *A high order positivity-preserving conservative WENO remapping method on 2D quadrilateral meshes*, Computer Methods in Applied Mechanics and Engineering, 373, 2021, 113-497.

[21] R. Löhner and C. Yang, *Improved ALE mesh velocities for moving bodies*, Communications in Numerical Methods in Engineering, 12, 1996, 599-608.

[22] R. Loubere, P. H. Maire, M. Shashkov, J. Breil and S. Galera, *ReALE: a reconnection-based arbitrary-Lagrangian–Eulerian method*, Journal of Computational Physics, 229, 2010, 4724-4761.

[23] J. Nocedal and S. J. Wright, *Numerical optimization*, Springer New York, 1999.

[24] W. F. Noh, *Errors for calculations of strong shocks using an artificial viscosity and an artificial heat flux*, Journal of Computational Physics, 72, 1987, 78-120.

[25] B. J. Parker and D. L. Youngs, *Two and three dimensional Eulerian simulation of fluid flow with material interfaces*, Atomic Weapons Establishment, 1992.

[26] Jr. J. E. Pilliod and E. G. Puckett, *Second-order accurate volume-of-fluid algorithms for tracking material interfaces*, Journal of Computational Physics, 199, 2004, 465-502.

[27] M. Shashkov and P. Knupp, *Optimization-based reference-matrix rezone strategies for arbitrary Lagrangian-Eulerian methods on unstructured meshes*, Selçuk Journal of Applied Mathematics, 3, 2002, 81-99.

[28] K. Siddiqi, B. B. Kimia and C.-W. Shu, *Geometric shock-capturing ENO schemes for subpixel interpolation, computation and curve evolution*, Graphical Models and Image Processing, 59, 1997, 278-301.

[29] I. E. Sutherland and G. W. Hodgman, *Reentrant polygon clipping*, Communications of the ACM, 17, 1974, 32–42.

[30] G. I. Taylor, *The instability of liquid surfaces when accelerated in a direction perpendicular to their planes, I*, Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 201, 1950, 192-196.

[31] P. Vachal and P. H. Maire, *Discretizations for weighted condition number smoothing on general unstructured meshes*, Computers & Fluids, 46, 2011, 479-485.

[32] A. M. Winslow, *Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh*, Journal of Computational Physics, 1, 1966, 149-172.

[33] D. L. Youngs, *Multi-mode implosion in cylindrical 3D geometry*, 11th International Workshop on the Physics of Compressible Turbulent Mixing (IWPCTM11), Santa Fe, 2008.

[34] X. Zhang and C.-W. Shu, *On positivity preserving high order discontinuous Galerkin schemes for compressible Euler equations on rectangular meshes*, Journal of Computational Physics, 229, 2010, 8918-8934.

[35] J. Zhu and C.-W. Shu, *A new type of multi-resolution WENO schemes with increasingly higher order of accuracy*, Journal of Computational Physics, 375, 2018, 659-683.