Hindawi
Scientific Programming
Volume 2017, Article ID 5896940, 17 pages
https://doi.org/10.1155/2017/5896940

Research Article

Parallel Pseudo Arc-Length Moving Mesh Schemes for Multidimensional Detonation

Jianguo Ning, Xinpeng Yuan, Tianbao Ma, and Jian Li

State Key Laboratory of Explosion Science and Technology, Beijing Institute of Technology, Beijing 100081, China

Correspondence should be addressed to Tianbao Ma; madabal@bit.edu.cn

Received 17 January 2017; Accepted 16 May 2017; Published 12 July 2017

Academic Editor: Piotr Luszczek

We have discussed the multidimensional parallel computation for pseudo arc-length moving mesh schemes, and the schemes can be used to capture the strong discontinuity for multidimensional detonations. Different from the traditional Euler numerical schemes, the problems of parallel schemes for pseudo arc-length moving mesh schemes include diagonal processor communications and mesh point communications, which are illustrated by the schematic diagram and key pseudocodes. Finally, the numerical examples are given to show that the pseudo arc-length moving mesh schemes are second-order convergent and can successfully capture the strong numerical strong discontinuity of the detonation wave. In addition, our parallel methods are proved effectively and the computational time is obviously decreased.

1. Introduction

Detonation is a type of combustion involving a supersonic exothermic front accelerating through a medium that eventually drives a shock front propagating directly in front of it. Detonations occur in both conventional solid and liquid explosives, [1] as well as in reactive gases [2]. Due to the strong discontinuity, one of the most important challenges in detonation simulations is to capture the precursor shock wave. To capture the strong discontinuity in detonation waves, mesh adaptation is an indispensable tool for use in the efficient numerical solution of this type of problem. The moving mesh method is one of the mesh adaptation methods, which relocate mesh point positions while maintaining the total number of mesh points and the mesh connectivity [3–5]. Particularly, Tang et al. proposed a moving mesh method that contained two parts: physical PDE time evolution and mesh redistribution [6–8]. The physical PDE time evolution and mesh redistribution were alternating, and conservative interpolation was used to transfer solutions from old mesh to new mesh. The method is shown to work well generally for hyperbolic conservation. After that, Ning et al. have improved this method and proposed the pseudo arc-length moving mesh schemes, which can deal with the multidimensional chemical reaction detonation problem [9].

In this paper, we will discuss the parallel computation of pseudo arc-length moving mesh schemes for multidimensional detonation. There are some works about the parallel schemes for Euler numerical scheme. Knepley and Karpeev developed a new programming framework, called Sieve, to support parallel numerical partial differential equation (PDE) algorithms operating over distributed meshes [10]. Morris et al. demonstrated how to build a parallel application that encapsulates the Message Passing Interface (MPI) without requiring the user to make direct calls to MPI except for startup and shutdown [11]. Mubarak et al. present us with a parallel algorithm of creating and deleting data copies, referred to as ghost copies, which localize neighborhood data for computation purposes while minimizing interprocess communication [12]. Wang et al. use the parallel scheme to reduce the cost for adaptive mesh refinement WENO scheme of multidimensional detonation [13]. However, there are no discussions about the parallel computation for moving mesh schemes, which will be of concern in this paper. Different from the traditional Euler numerical scheme, the data communications of moving mesh schemes between processors are more complex, which include physical values and mesh points. Besides, the processor communications for pseudo arc-length moving mesh schemes include adjacent processor and diagonal processor. Here, we adopt the software

architecture of MPI. The most important advantages of this model are twofold: achievable performance and portability. Performance is a direct result of available optimized MPI libraries and full user control in the program development cycle. Portability arises from the standard API and the existence of MPI libraries on a wide range of machines. In general, an MPI program runs on distributed memory machines. The processor communications for the parallel computation of pseudo arc-length moving mesh schemes are more complex than the traditional Euler scheme, and it includes adjacent processor and diagonal processor. Here, we will consider three kinds of processor partitions to show our parallel schemes. This article is organized as follows. Section 2 introduces the chemical and physical model. Section 3 presents the numerical scheme. Section 4 is devoted to the parallel computation. Section 5 conducts several numerical experiments to demonstrate our schemes. The paper ends with a conclusion and discussion in Section 6.

2. Governing Equations

Instead of using many real elementary reactions, a two-step model was utilized as the testing model. Two-step reaction model considers a complicated chemical reaction to be an induction and an exothermic reaction. For both induction reaction and exothermic reaction, the progress parameters α and β are unity at first, then α decreases to zero, and β decreases until an equilibrium state is reached. The rates ω_α and ω_β are given as follows [14].

$$\omega_\alpha = \frac{d\alpha}{dt} = -k_\alpha \rho \exp\left(-\frac{E_\alpha}{RT}\right),$$
$$\omega_\beta = \frac{d\beta}{dt} = \begin{cases} 0 & (\alpha > 0), \\ -k_\beta p^2 \left[\beta^2 \exp\left(-\frac{E_\beta}{RT}\right) - (1-\beta)^2 \exp\left(-\frac{E_\beta + Q}{RT}\right)\right] & (\alpha \le 0), \end{cases} \tag{1}$$

where ρ is the mass density, p the pressure, T the temperature, R the gas constant, Q the heat release parameter, k_α and k_β the constants of reaction rates, and E_α and E_β the activation energies.

In deriving fundamental equations, the gas is assumed to be perfect, nonviscous, and non-heat-conducting. In Cartesian coordinates, governing equations for gaseous detonation problem, including the above chemical reaction, are

$$\frac{\partial}{\partial t}\mathbf{w} + \nabla_{\mathbf{x}} \mathscr{F}(\mathbf{w}) = \mathbf{s}(\mathbf{w}), \quad \mathbf{t} \ge \mathbf{0}, \tag{2}$$

where $\mathbf{x}$ is the multidimensional vector, $\mathscr{F}(\mathbf{w})$ is the multidimensional matrix function, and $\mathbf{s}(\mathbf{w})$ is the chemical reaction source term. For the one-dimensional space,

$$\mathbf{w} = (\rho, \rho u, E, \rho\alpha, \rho\beta)^T,$$
$$\mathscr{F} = \left(\rho u, \rho u^2 + p, (E+p)u, \rho u\alpha, \rho u\beta\right)^T, \tag{3}$$
$$\mathbf{s} = (0, 0, 0, \omega_1, \omega_2)^T.$$

For the two-dimensional space,

$$\mathbf{w} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \\ \rho\alpha \\ \rho\beta \end{pmatrix},$$

$$\mathscr{F} = \begin{pmatrix} \rho u & \rho v \\ \rho u^2 + p & \rho uv \\ \rho uv & \rho v^2 + p \\ (E+p)u & (E+p)v \\ \rho u\alpha & \rho v\alpha \\ \rho u\beta & \rho v\beta \end{pmatrix},$$

$$\mathbf{s} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \omega_1 \\ \omega_2 \end{pmatrix}. \tag{4}$$

In the case of three-dimensional space,

$$\mathbf{w} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \\ \rho\alpha \\ \rho\beta \end{pmatrix},$$

$$\mathscr{F} = \begin{pmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + p & \rho uv & \rho uw \\ \rho vu & \rho v^2 + p & \rho vw \\ \rho wu & \rho wv & \rho w^2 + p \\ (E+p)\,u & (E+p)\,v & (E+p)\,w \\ \rho u\alpha & \rho v\alpha & \rho w\alpha \\ \rho u\beta & \rho v\beta & \rho w\beta \end{pmatrix},$$

$$\mathbf{s} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \omega_1 \\ \omega_2 \end{pmatrix}, \tag{5}$$

where u, v, and w are the Cartesian components of the fluid velocity in the x, y, and z directions, respectively. Total energy density E is defined as

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right) + \rho Q\beta. \tag{6}$$

Here γ is the specific heat ratio.

3. Numerical Method

Firstly, we present the framework of pseudo arc-length moving mesh schemes for the gaseous detonation problem (2). Our adaptive scheme is formed by two independents parts: the evolution of the governing equation and the iterative mesh redistribution.

3.1. Time Evolution of Governing Equations. The physical domain for computation is Ω_p. Given a partition of the mesh domain $\{K_{\bar{\mathbf{j}}} \mid K_{\bar{\mathbf{j}}} \in \Omega_p, K_{\bar{\mathbf{j}}} = [\mathbf{x}_{\mathbf{j}}, \mathbf{x}_{\mathbf{j}+1}], \bigcup_{\bar{\mathbf{j}}} K_{\bar{\mathbf{j}}} = \Omega_p\}$ and a partition of the time interval $[0, T]$. The average on the cell $K_{\bar{\mathbf{j}}}$ is

$$\overline{\mathbf{w}}_{\bar{\mathbf{j}}} = \frac{1}{\left|K_{\bar{\mathbf{j}}}\right|}\int_{K_{\bar{\mathbf{j}}}} \mathbf{w}_{\bar{\mathbf{j}}}\, d\sigma. \tag{7}$$

Here, the sign $|K_{\bar{\mathbf{j}}}|$ denotes the length for the region $K_{\bar{\mathbf{j}}}(K_{\bar{j}})$ in one dimension, the area for the region $K_{\bar{\mathbf{j}}}(K_{\bar{i},\bar{j}})$ in two dimensions, and the volumes for the region $K_{\bar{\mathbf{j}}}(K_{\bar{i},\bar{j},\bar{k}})$ in three dimensions. Integrating (2) over $K_{\bar{\mathbf{j}}}$, we have

$$\int_{K_{\bar{\mathbf{j}}}} \frac{\partial \mathbf{w}}{\partial t} d\sigma + \int_{K_{\bar{\mathbf{j}}}} \nabla_{\mathbf{x}} \mathscr{F}(\mathbf{w})\, d\sigma = \int_{K_{\bar{\mathbf{j}}}} \mathbf{s}(\mathbf{w})\, d\sigma. \tag{8}$$

Applying the Green-Gauss's theorem and rewriting, we have

$$\left|K_{\bar{\mathbf{j}}}\right| \frac{d\mathbf{w}_{\bar{\mathbf{j}}}}{dt} + \oint_{\partial K_{\bar{\mathbf{j}}}} \mathscr{F}(\mathbf{w}) \cdot \mathbf{n}_{\bar{\mathbf{j}}} d\sigma = \left|K_{\bar{\mathbf{j}}}\right| \mathbf{s}\left(\mathbf{w}_{\bar{\mathbf{j}}}\right), \tag{9}$$

where $\mathbf{n}_{\bar{\mathbf{j}}}$ is the outward unit normal vector of boundary external surface $\partial K_{\bar{\mathbf{j}}}$. The Lax-Friedrichs flux is defined by

$$\mathscr{H}(\mathbf{u}, \mathbf{v}, \mathbf{n}) = \frac{1}{2}\left[\mathscr{F}(\mathbf{u}) \cdot \mathbf{n} + \mathscr{F}(\mathbf{v}) \cdot \mathbf{n} - a(\mathbf{v} - \mathbf{u})\right], \tag{10}$$

where $a = \max_{\mathbf{u},\mathbf{v}} |\mathscr{F}'(\mathbf{u}) \cdot \mathbf{n}|$. It satisfies the conservation and consistency

$$\begin{aligned} \mathscr{H}(\mathbf{u}, \mathbf{v}, \mathbf{n}) &= -\mathscr{H}(\mathbf{u}, \mathbf{v}, -\mathbf{n}), \\ \mathscr{H}(\mathbf{u}, \mathbf{u}, \mathbf{n}) &= \mathscr{F}(\mathbf{u}) \cdot \mathbf{n}. \end{aligned} \tag{11}$$

Let $\{e^l_{\bar{\mathbf{j}}} \mid e^l_{\bar{\mathbf{j}}} \in \partial K_{\bar{\mathbf{j}}}, \bigcup_{l=1}^{N_l} e^l_{\bar{\mathbf{j}}} = \partial K_{\bar{\mathbf{j}}}\}$ be a partition of boundary external surface $\partial K_{\bar{\mathbf{j}}}$ and $\mathbf{n}_{\bar{\mathbf{j}}}$ be the outward unit normal vector of surface $e^l_{\bar{\mathbf{j}}}$. Then we have a semidiscrete scheme of (2)

$$\begin{aligned} \left|K_{\bar{\mathbf{j}}}\right| \frac{d\mathbf{w}_{\bar{\mathbf{j}}}}{dt} = &-\sum_{l=1}^{N_l} \int_{e^l_{\bar{\mathbf{j}}}} \mathscr{H}\left(\mathbf{w}^{\mathrm{int}(l)}_{\bar{\mathbf{j}}}, \mathbf{w}^{\mathrm{ext}(l)}_{\bar{\mathbf{j}}}, \mathbf{n}^l_{\bar{\mathbf{j}}}\right) d\sigma \\ &+ \left|K_{\bar{\mathbf{j}}}\right| \mathbf{s}\left(\mathbf{w}_{\bar{\mathbf{j}}}\right), \end{aligned} \tag{12}$$

where $\mathbf{w}^{\mathrm{int}(l)}_{\bar{\mathbf{j}}}$ or $\mathbf{w}^{\mathrm{ext}(l)}_{\bar{\mathbf{j}}}$, $l = 1, 2, \ldots, N_l$, is the internal or external approximate value at $e^l_{\bar{\mathbf{j}}}$.

For the one-dimensional space, $N_l = 2$, and $e^1_{\bar{j}}$ ($e^2_{\bar{j}}$) is the left (right) point of line cell $K_{\bar{j}}$. Thus, (12) becomes

$$\begin{aligned} &\frac{d\mathbf{w}_{\bar{j}}}{dt} \\ &= \frac{1}{\left|K_{\bar{j}}\right|}\left[-\mathscr{H}\left(\mathbf{w}^{\mathrm{int}(1)}_{\bar{j}}, \mathbf{w}^{\mathrm{ext}(1)}_{\bar{j}}\right) + \mathscr{H}\left(\mathbf{w}^{\mathrm{int}(2)}_{\bar{j}}, \mathbf{w}^{\mathrm{ext}(2)}_{\bar{j}}\right)\right] \\ &\quad + \mathbf{s}\left(\mathbf{w}_{\bar{j}}\right). \end{aligned} \tag{13}$$

By the initial reconstruction technique [15] to reset $\mathbf{w}^{\mathrm{int}(l)}_{\bar{j}}$, $\mathbf{w}^{\mathrm{ext}(l)}_{\bar{j}}$, $l = 1, 2$, on the edge $e^l_{\bar{j}}$, we can obtain the second-order accurate spatial discretization:

$$\begin{aligned} \mathbf{w}^{\mathrm{int}(1)}_{\bar{j}} &= \mathbf{w}_{\bar{j}} + \frac{1}{2}\psi\left(\mathbf{w}_{\bar{j}} - \mathbf{w}_{\bar{j}+1}, \mathbf{w}_{\bar{j}-1} - \mathbf{w}_{\bar{j}}\right), \\ \mathbf{w}^{\mathrm{ext}(1)}_{\bar{j}} &= \mathbf{w}_{\bar{j}-1} + \frac{1}{2}\psi\left(\mathbf{w}_{\bar{j}} - \mathbf{w}_{\bar{j}-1}, \mathbf{w}_{\bar{j}-1} - \mathbf{w}_{\bar{j}-2}\right), \\ \mathbf{w}^{\mathrm{int}(2)}_{\bar{j}} &= \mathbf{w}^{\mathrm{ext}(1)}_{\bar{j}+1}, \\ \mathbf{w}^{\mathrm{ext}(2)}_{\bar{j}} &= \mathbf{w}^{\mathrm{int}(1)}_{\bar{j}+1}, \end{aligned} \tag{14}$$

where ψ is a nonlinear limiter function which is used to suppress the possible pseudo oscillation. In our computations, we use van Leer's limiter [15]

$$\psi(a, b) = \left(\mathrm{sign}(a) + \mathrm{sign}(b)\right)\frac{|ab|}{|a| + |b| + \varepsilon}. \tag{15}$$

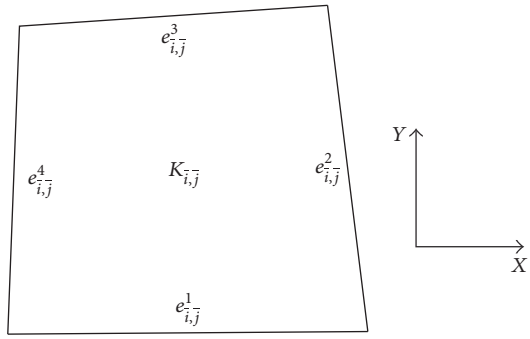

Figure 1: Schematic diagram of the quadrilateral element $K_{\bar{i},\bar{j}}$.

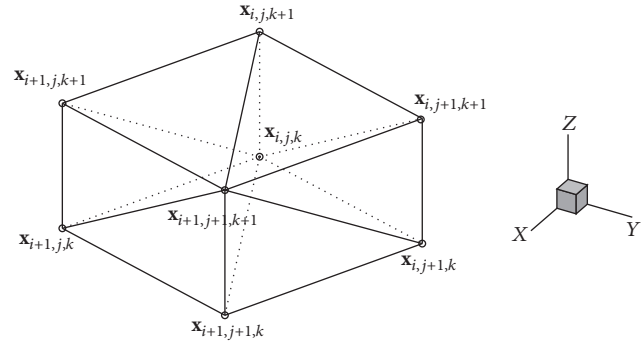

Figure 2: Schematic diagram of the quadrilateral element $K_{\bar{i},\bar{j},\bar{k}}$.

For the two-dimensional space, $N_l = 4$, and the diagram for edges $e^l_{\bar{i},\bar{j}}$, $l = 1, 2, 3, 4$, of the quadrilateral element $K_{\bar{i},\bar{j}}$ is shown in Figure 1.

The values $\mathbf{w}^{\mathrm{int}(l)}_{\bar{i},\bar{j}}$, $\mathbf{w}^{\mathrm{ext}(l)}_{\bar{i},\bar{j}}$ at edge $e^l_{\bar{i},\bar{j}}$, $l = 1, 2, 3, 4$, can be defined by

$$\begin{aligned}
\mathbf{w}^{\mathrm{int}(1)}_{\bar{i},\bar{j}} &= \mathbf{w}_{\bar{i},\bar{j}} + \frac{1}{2}\psi\left(\mathbf{w}_{\bar{i},\bar{j}} - \mathbf{w}_{\bar{i},\bar{j}+1}, \mathbf{w}_{\bar{i},\bar{j}-1} - \mathbf{w}_{\bar{i},\bar{j}}\right),\\
\mathbf{w}^{\mathrm{ext}(1)}_{\bar{i},\bar{j}} &= \mathbf{w}_{\bar{i},\bar{j}-1} + \frac{1}{2}\psi\left(\mathbf{w}_{\bar{i},\bar{j}} - \mathbf{w}_{\bar{i},\bar{j}-1}, \mathbf{w}_{\bar{i},\bar{j}-1} - \mathbf{w}_{\bar{i},\bar{j}-2}\right),\\
\mathbf{w}^{\mathrm{ext}(3)}_{\bar{i},\bar{j}} &= \mathbf{w}^{\mathrm{int}(1)}_{\bar{i},\bar{j}+1},\\
\mathbf{w}^{\mathrm{int}(3)}_{\bar{i},\bar{j}} &= \mathbf{w}^{\mathrm{ext}(1)}_{\bar{i},\bar{j}+1}\\
\mathbf{w}^{\mathrm{int}(2)}_{\bar{i},\bar{j}} &= \mathbf{w}_{\bar{i},\bar{j}} + \frac{1}{2}\psi\left(\mathbf{w}_{\bar{i},\bar{j}} - \mathbf{w}_{\bar{i}-1,\bar{j}}, \mathbf{w}_{\bar{i}+1,\bar{j}} - \mathbf{w}_{\bar{i},\bar{j}}\right),\\
\mathbf{w}^{\mathrm{ext}(2)}_{\bar{i},\bar{j}} &= \mathbf{w}_{\bar{i}+1,\bar{j}} + \frac{1}{2}\psi\left(\mathbf{w}_{\bar{i}+1,\bar{j}} - \mathbf{w}_{\bar{i}+2,\bar{j}}, \mathbf{w}_{\bar{i},\bar{j}} - \mathbf{w}_{\bar{i}+1,\bar{j}}\right),\\
\mathbf{w}^{\mathrm{ext}(4)}_{\bar{i},\bar{j}} &= \mathbf{w}^{\mathrm{int}(2)}_{\bar{i}-1,\bar{j}},\\
\mathbf{w}^{\mathrm{int}(4)}_{\bar{i},\bar{j}} &= \mathbf{w}^{\mathrm{ext}(2)}_{\bar{i}-1,\bar{j}}.
\end{aligned} \tag{16}$$

For the three-dimensional hexahedron element, the situation is complex. Due to the movement of the mesh points, the hexahedron element will change to polyhedron element. For calculating the element volume and the boundary external surface area, each surface of hexahedron element is divided into two surfaces and calculated. The control element $K_{\bar{i},\bar{j},\bar{k}}$ is illustrated in Figure 2. The boundary surfaces $e^l_{\bar{i},\bar{j},\bar{k}}$, $l = 1, 2, \ldots, 12$, are defined in Table 1.

The second-order values $\mathbf{w}^{\mathrm{int}(l)}_{\bar{i},\bar{j},\bar{k}}$, $\mathbf{w}^{\mathrm{ext}(l)}_{\bar{i},\bar{j},\bar{k}}$ at edge $e^l_{\bar{i},\bar{j},\bar{k}}$, $l = 1, 2, \ldots, 12$, can be obtained by

$$\begin{aligned}
\mathbf{w}^{\mathrm{int}(1)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{int}(2)}_{\bar{i},\bar{j},\bar{k}}\\
&= \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}}\\
&\quad + \frac{1}{2}\psi\left(\overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}} - \overline{\mathbf{w}}_{\bar{i}-1,\bar{j},\bar{k}}, \overline{\mathbf{w}}_{\bar{i}+1,\bar{j},\bar{k}} - \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}}\right),
\end{aligned}$$

Table 1: Definition of boundary surfaces $e^l_{\bar{i},\bar{j},\bar{k}}$, $l = 1, 2, \ldots, 12$.

Surfaces	Vertexes
$e^1_{\bar{i},\bar{j},\bar{k}}$	$\langle \mathbf{x}_{i+1,j,k+1}, \mathbf{x}_{i+1,j,k}, \mathbf{x}_{i+1,j+1,k+1} \rangle$
$e^2_{\bar{i},\bar{j},\bar{k}}$	$\langle \mathbf{x}_{i+1,j,k}, \mathbf{x}_{i+1,j+1,k}, \mathbf{x}_{i+1,j+1,k+1} \rangle$
$e^3_{\bar{i},\bar{j},\bar{k}}$	$\langle \mathbf{x}_{i,j,k}, \mathbf{x}_{i,j,k+1}, \mathbf{x}_{i,j+1,k+1} \rangle$
$e^4_{\bar{i},\bar{j},\bar{k}}$	$\langle \mathbf{x}_{i,j,k}, \mathbf{x}_{i,j+1,k+1}, \mathbf{x}_{i,j+1,k} \rangle$
$e^5_{\bar{i},\bar{j},\bar{k}}$	$\langle \mathbf{x}_{i,j+1,k}, \mathbf{x}_{i,j+1,k+1}, \mathbf{x}_{i+1,j+1,k+1} \rangle$
$e^6_{\bar{i},\bar{j},\bar{k}}$	$\langle \mathbf{x}_{i,j+1,k}, \mathbf{x}_{i+1,j+1,k+1}, \mathbf{x}_{i+1,j+1,k} \rangle$
$e^7_{\bar{i},\bar{j},\bar{k}}$	$\langle \mathbf{x}_{i,j,k+1}, \mathbf{x}_{i,j,k}, \mathbf{x}_{i+1,j,k+1} \rangle$
$e^8_{\bar{i},\bar{j},\bar{k}}$	$\langle \mathbf{x}_{i,j,k}, \mathbf{x}_{i+1,j,k}, \mathbf{x}_{i+1,j,k+1} \rangle$
$e^9_{\bar{i},\bar{j},\bar{k}}$	$\langle \mathbf{x}_{i,j,k+1}, \mathbf{x}_{i+1,j,k+1}, \mathbf{x}_{i+1,j+1,k+1} \rangle$
$e^{10}_{\bar{i},\bar{j},\bar{k}}$	$\langle \mathbf{x}_{i,j,k+1}, \mathbf{x}_{i+1,j+1,k+1}, \mathbf{x}_{i,j+1,k+1} \rangle$
$e^{11}_{\bar{i},\bar{j},\bar{k}}$	$\langle \mathbf{x}_{i+1,j,k}, \mathbf{x}_{i,j,k}, \mathbf{x}_{i+1,j+1,k} \rangle$
$e^{12}_{\bar{i},\bar{j},\bar{k}}$	$\langle \mathbf{x}_{i,j,k}, \mathbf{x}_{i,j+1,k}, \mathbf{x}_{i+1,j+1,k} \rangle$

$$\begin{aligned}
\mathbf{w}^{\mathrm{ext}(1)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{ext}(2)}_{\bar{i},\bar{j},\bar{k}}\\
&= \overline{\mathbf{w}}_{\bar{i}+1,\bar{j},\bar{k}}\\
&\quad + \frac{1}{2}\psi\left(\overline{\mathbf{w}}_{\bar{i}+1,\bar{j},\bar{k}} - \overline{\mathbf{w}}_{\bar{i}+2,\bar{j},\bar{k}}, \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}} - \overline{\mathbf{w}}_{\bar{i}+1,\bar{j},\bar{k}}\right),\\
\mathbf{w}^{\mathrm{int}(5)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{int}(6)}_{\bar{i},\bar{j},\bar{k}}\\
&= \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}}\\
&\quad + \frac{1}{2}\psi\left(\overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}} - \overline{\mathbf{w}}_{\bar{i},\bar{j}-1,\bar{k}}, \overline{\mathbf{w}}_{\bar{i},\bar{j}+1,\bar{k}} - \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}}\right),\\
\mathbf{w}^{\mathrm{ext}(5)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{ext}(6)}_{\bar{i},\bar{j},\bar{k}}\\
&= \overline{\mathbf{w}}_{\bar{i},\bar{j}+1,\bar{k}}\\
&\quad + \frac{1}{2}\psi\left(\overline{\mathbf{w}}_{\bar{i},\bar{j}+1,\bar{k}} - \overline{\mathbf{w}}_{\bar{i},\bar{j}+2,\bar{k}}, \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}} - \overline{\mathbf{w}}_{\bar{i},\bar{j}+1,\bar{k}}\right),\\
\mathbf{w}^{\mathrm{int}(9)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{int}(10)}_{\bar{i},\bar{j},\bar{k}}\\
&= \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}}\\
&\quad + \frac{1}{2}\psi\left(\overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}} - \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}-1}, \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}+1} - \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}}\right),
\end{aligned}$$

$$
\begin{aligned}
\mathbf{w}^{\mathrm{ext}(9)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{ext}(10)}_{\bar{i},\bar{j},\bar{k}} \\
&= \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}+1} \\
&\quad + \frac{1}{2}\psi\left(\overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}+1} - \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}+2}, \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}} - \overline{\mathbf{w}}_{\bar{i},\bar{j},\bar{k}+1}\right), \\
\mathbf{w}^{\mathrm{int}(3)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{ext}(1)}_{\bar{i}-1,\bar{j},\bar{k}}, \\
\mathbf{w}^{\mathrm{int}(4)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{ext}(2)}_{\bar{i}-1,\bar{j},\bar{k}}, \\
\mathbf{w}^{\mathrm{ext}(3)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{int}(1)}_{\bar{i}-1,\bar{j},\bar{k}}, \\
\mathbf{w}^{\mathrm{ext}(4)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{int}(2)}_{\bar{i}-1,\bar{j},\bar{k}}, \\
\mathbf{w}^{\mathrm{int}(7)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{ext}(5)}_{\bar{i},\bar{j}-1,\bar{k}}, \\
\mathbf{w}^{\mathrm{int}(8)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{ext}(6)}_{\bar{i},\bar{j}-1,\bar{k}}, \\
\mathbf{w}^{\mathrm{ext}(7)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{int}(5)}_{\bar{i},\bar{j}-1,\bar{k}}, \\
\mathbf{w}^{\mathrm{ext}(8)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{int}(6)}_{\bar{i},\bar{j}-1,\bar{k}}, \\
\mathbf{w}^{\mathrm{int}(9)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{ext}(11)}_{\bar{i},\bar{j},\bar{k}-1}, \\
\mathbf{w}^{\mathrm{int}(9)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{ext}(11)}_{\bar{i},\bar{j},\bar{k}-1}, \\
\mathbf{w}^{\mathrm{ext}(10)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{int}(12)}_{\bar{i},\bar{j},\bar{k}-1}, \\
\mathbf{w}^{\mathrm{ext}(10)}_{\bar{i},\bar{j},\bar{k}} &= \mathbf{w}^{\mathrm{int}(12)}_{\bar{i},\bar{j},\bar{k}-1}.
\end{aligned}
\tag{17}
$$

Time discretization can be achieved by the strong stability preserving high order Runge-Kutta time discretization [16, 17]. The semidiscrete scheme (12) can be written as

$$
\widetilde{\mathbf{w}}_t = L\left(\widetilde{\mathbf{w}}\right). \tag{18}
$$

Here, the second-order TVD Runge-Kutta method for (18) in the time discretization is

$$
\begin{aligned}
\widetilde{\mathbf{w}}^{(1)} &= \widetilde{\mathbf{w}}^n + \Delta t L\left(\widetilde{\mathbf{w}}^n\right), \\
\widetilde{\mathbf{w}}^{n+1} &= \frac{1}{2}\widetilde{\mathbf{w}}^n + \frac{1}{2}\widetilde{\mathbf{w}}^{(1)} + \frac{1}{2}\Delta t L\left(\widetilde{\mathbf{w}}^{(1)}\right).
\end{aligned}
\tag{19}
$$

3.2. Pseudo Arc-Length Moving Mesh Scheme. Let $\mathbf{x} = \{x^1, x^2, \ldots, x^{N_d}\}$ and $\boldsymbol{\xi} = \{\xi^1, \xi^2, \ldots, \xi^{N_d}\}$ denote the physical and computational coordinates, respectively. Here, N_d denotes the number of spatial dimensions. A one-to-one coordinate transformation from the computational domain Ω_c to the physical domain Ω_p is denoted by

$$
\begin{aligned}
\mathbf{x} : \boldsymbol{\xi} &\longrightarrow \mathbf{x}, \\
\Omega_c &\longrightarrow \Omega_p.
\end{aligned}
\tag{20}
$$

In the variational approach, the mesh map between the domains Ω_c and Ω_p is usually provided by

$$
\Psi\left(\boldsymbol{\xi}\right) = \frac{1}{2}\sum_{r=1}^{N_d}\int_{\Omega_p}\left(\nabla_{\mathbf{x}}\xi^r\right)^T G_r^{-1}\left(\nabla_{\mathbf{x}}\xi^r\right) d\mathbf{x}, \tag{21}
$$

where $\nabla_{\mathbf{x}} := (\partial_{x^1}, \partial_{x^2}, \ldots, \partial_{x^{N_d}})^T$ and G_r, $r = 1, 2, \ldots, N_d$, are given symmetric positive definite matrices called monitor functions, depending on the underlying solution to be adaptive. The Euler-Lagrange equations of the functional (21) have the form

$$
\nabla_{\mathbf{x}} \cdot \left(G_r^{-1}\nabla_{\mathbf{x}}\xi^r\right) = 0, \quad 1 \le r \le N_d. \tag{22}
$$

In practice, Ω_p may have a very complex geometry, and as a result it is not realistic to solve (22) directly. An alternative is to consider a functional

$$
\widetilde{\Psi}\left(\mathbf{x}\right) = \frac{1}{2}\sum_{r=1}^{N_d}\int_{\Omega_c}\left(\nabla_{\boldsymbol{\xi}}x^r\right)^T G_r\left(\nabla_{\boldsymbol{\xi}}x^r\right) d\boldsymbol{\xi}, \tag{23}
$$

where $\nabla_{\boldsymbol{\xi}} := (\partial_{\xi^1}, \partial_{\xi^2}, \ldots, \partial_{\xi^{N_d}})^T$. In addition, one of the choices of monitor functions is $G = \omega I$, where I is the identity matrix and ω is a positive weight function. For the purpose to have more accuracy near the nonsmooth part of solutions, we introduce the monitor function of pseudo arc-length norm [18, 19]

$$
\omega = \sqrt{1 + \alpha_1 |W| + \alpha_2 |\nabla W|^2}, \tag{24}
$$

where α_1 and α_2 are some nonnegative constants. Thus, the corresponding Euler-Lagrange equations about (23) are

$$
\nabla_{\boldsymbol{\xi}} \cdot \left(\omega\nabla_{\boldsymbol{\xi}}x^r\right) = 0, \quad 1 \le r \le N_d. \tag{25}
$$

In our computations, we will use the Gauss-Seidel type iteration method to solve the mesh equation (25)

$$
\begin{aligned}
&\sum_{r=1}^{N_d}\Big[\vartheta^r_{j^1,j^2,\ldots,j^r+1/2,\ldots,j^{N_d}}\left(\mathbf{x}^{[\kappa]}_{j^1,j^2,\ldots,j^r+1,\ldots,j^{N_d}}\right. \\
&\qquad \left. - \mathbf{x}^{[\kappa+1]}_{j^1,j^2,\ldots,j^r,\ldots,j^{N_d}}\right) \\
&\qquad - \vartheta^r_{j^1,j^2,\ldots,j^r-1/2,\ldots,j^{N_d}}\left(\mathbf{x}^{[\kappa+1]}_{j^1,j^2,\ldots,j^r,\ldots,j^{N_d}}\right. \\
&\qquad \left. - \mathbf{x}^{[\kappa]}_{j^1,j^2,\ldots,j^r-1,\ldots,j^{N_d}}\right)\Big] = 0,
\end{aligned}
\tag{26}
$$

for $1 \le j^r \le N_{\xi^r}$, $1 \le r \le N_d$, and $\kappa = 0, 1, \ldots$, where

$$
\begin{aligned}
\vartheta^r_{j^1,j^2,\ldots,j^r-1/2,\ldots,j^{N_d}} &= \omega_{j^1,j^2,\ldots,j^r-1/2,\ldots,j^{N_d}} \\
&= \frac{1}{2\left(N_d - 1\right)}\sum_{s_r=j^r-1/2, s_k=j^k-1/2, j^k+1/2}\omega_{s_1,s_2,\ldots,s_r,\ldots,s_{N_d}}, \\
\vartheta^r_{j^1,j^2,\ldots,j^r+1/2,\ldots,j^{N_d}} &= \omega_{j^1,j^2,\ldots,j^r+1/2,\ldots,j^{N_d}} \\
&= \frac{1}{2\left(N_d - 1\right)}\sum_{s_r=j^r+1/2, s_k=j^k-1/2, j^k+1/2}\omega_{s_1,s_2,\ldots,s_r,\ldots,s_{N_d}},
\end{aligned}
\tag{27}
$$

where $\omega_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}} := \omega(\mathbf{w}^{[\kappa]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}})$. In addition, let us consider the solutions updating on new grids $\mathbf{x}^{[\kappa+1]}$ from the old grids $\mathbf{x}^{[\kappa]}$. Here, the solution updating should preserve conservative property for $\mathbf{w}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$ in the sense of that

$$\sum_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}} \left|K^{[\kappa]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}\right| \mathbf{w}^{[\kappa]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}} = \sum_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}} \left|K^{[\kappa+1]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}\right| \mathbf{w}^{[\kappa+1]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}. \tag{28}$$

Let D_l denote the region scanned by the boundary $e^l_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$ after one iterative step of (28). So we can have the following conservative interpolation scheme:

$$\left|K^{[\kappa+1]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}\right| \mathbf{w}^{[\kappa+1]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}} = \left|K^{[\kappa]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}\right| \mathbf{w}^{[\kappa]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}} + \sum_{l=1}^{N_l} \widehat{F}_l\left(\mathbf{w}^{\mathrm{int}(l)}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}, \mathbf{w}^{\mathrm{ext}(l)}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}\right), \tag{29}$$

where $\mathbf{w}^{\mathrm{int}(l)}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$ and $\mathbf{w}^{\mathrm{ext}(l)}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$ are the reconstructed left and right states on the boundary $e^l_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$ by $\mathbf{w}^{[\kappa]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$ (for more details see Section 3.2). $\widehat{F}_l(\cdot,\cdot)$, $l = 1,2,\ldots,N_l$, denotes the integration of $\mathbf{w}$ over the domain D_l. Here we have omitted the subscripts $\bar{j}^r$, $r = 1,2,\ldots,N_d$, of $\widehat{F}_l(\cdot,\cdot)$ and D_l for simplicity. Since only cell averages of $\mathbf{w}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$ over $K_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$ are known, we should give an approximation of $\widehat{F}_l(\cdot,\cdot)$ instead of its exact value. Then $\widehat{F}_l(\cdot,\cdot)$ can be approximately calculated as

$$\widehat{F}_l(\mathbf{u},\mathbf{v}) = \max\{D_l,0\}\cdot\mathbf{v} + \min\{D_l,0\}\cdot\mathbf{u}. \tag{30}$$

Next, we will show the examples to calculate D_l, $l = 1,2,\ldots,N_d$, according to different dimensions.

In one-dimensional space, D_1, D_2 can be obtained from the following:

$$D_1 = x^{[\kappa]}_j - x^{[\kappa+1]}_j, \qquad D_2 = x^{[\kappa+1]}_{j+1} - x^{[\kappa]}_{j+1}. \tag{31}$$

In two-dimensional space, the changed area $D_l, l = 1,2,\ldots,4$, are illustrated in Figure 3. As an example, we compute D_1. Noticing $\mathbf{x}_{i,j} = (x_{i,j}, y_{i,j})$, we have

$$D_1 := \frac{1}{2}\left[\left(x^{[\kappa+1]}_{i+1,j} - x^{[\kappa]}_{i,j}\right)\left(y^{[\kappa]}_{i+1,j} - y^{[\kappa+1]}_{i,j}\right) - \left(y^{[\kappa+1]}_{i+1,j} - y^{[\kappa]}_{i,j}\right)\left(x^{[\kappa]}_{i+1,j} - x^{[\kappa+1]}_{i,j}\right)\right]. \tag{32}$$

Obviously, D_1 is the area, which is positive if $\mathbf{x}^{[\mathbf{z}+1]}_{\mathbf{i-1,j-1}}$, $\mathbf{x}^{[\mathbf{z}+1]}_{\mathbf{i,j-1}}$, $\mathbf{x}^{[\mathbf{z}]}_{\mathbf{i,j-1}}$, and $\mathbf{x}^{[\mathbf{z}]}_{\mathbf{i-1,j-1}}$ are located in a counter-clockwise order; otherwise it is negative if those four points are located in a clockwise order.

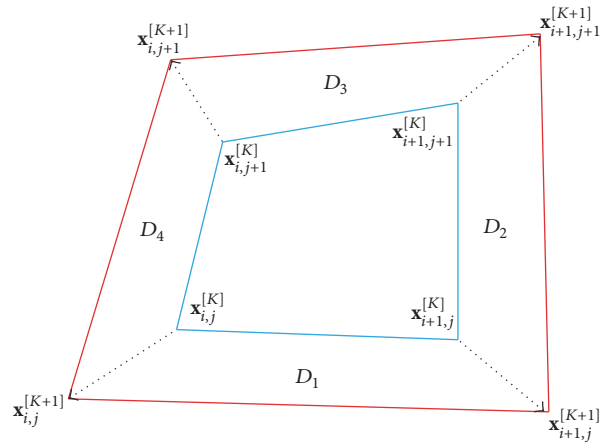

FIGURE 3: Schematic diagram of the two-dimensional changed areas D_l, $l = 1,2,3,4$.

In the three-dimensional space, the control element $K_{i+1/2,j+1/2,k+1/2}$ is illustrated in Figure 4. Similarly, we compute D_1 as an example. The changed area D_1 is illustrated in Figure 5. D_1 is composed by three tetrahedrons, so we have

$$D_1 = V_{\mathbf{x}^{[\kappa]}_{i+1,j,k+1}\mathbf{x}^{[\kappa+1]}_{i+1,j,k+1}\mathbf{x}^{[\kappa+1]}_{i+1,j+1,k+1}\mathbf{x}^{[\kappa+1]}_{i,j,k+1}} + V_{\mathbf{x}^{[\kappa]}_{i+1,j,k+1}\mathbf{x}^{[\kappa+1]}_{i+1,j+1,k+1}\mathbf{x}^{[\kappa]}_{i,j,k+1}\mathbf{x}^{[\kappa+1]}_{i,j,k+1}} + V_{\mathbf{x}^{[\kappa]}_{i+1,j,k+1}\mathbf{x}^{[\kappa+1]}_{i+1,j+1,k+1}\mathbf{x}^{[\kappa]}_{i+1,j+1,k+1}\mathbf{x}^{[\kappa]}_{i,j,k+1}}. \tag{33}$$

Each tetrahedron volume can be calculated by the volume formula

$$V_{OABC} = \frac{1}{6}\begin{vmatrix} x_{OA} & y_{OA} & z_{OA} \\ x_{OB} & y_{OB} & z_{OB} \\ x_{OC} & y_{OC} & z_{OC} \end{vmatrix}. \tag{34}$$

Obviously, V_{OABC} is the volume, which is positive if A, B, and C are located in a counter-clockwise order; otherwise it is negative if those three points are located in a clockwise order.

The solution procedure can be illustrated by the following flowchart.

Algorithm 1.

Step 1. Give a uniform mesh $\{\boldsymbol{\xi}^0_{j^1,j^2,\ldots,j^{N_d}}\}$ on the computational domain Ω_c. Initialize the physical mesh $\{\mathbf{x}^0_{j^1,j^2,\ldots,j^{N_d}}\}$ on the physical domain Ω_p, and compute the grid initial values $\mathbf{w}^0_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$. Then, let $\mathbf{x}^{[0]}_{j^1,j^2,\ldots,j^{N_d}} := \mathbf{x}^0_{j^1,j^2,\ldots,j^{N_d}}$ and $\mathbf{w}^{[0]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}} := \mathbf{w}^0_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$.

Step 2. With scheme (26), move the grid from $\mathbf{x}^{[\kappa]}_{j^1,j^2,\ldots,j^{N_d}}$ to $\mathbf{x}^{[\kappa+1]}_{j^1,j^2,\ldots,j^{N_d}}$ and the corresponding grid values from $\mathbf{w}^{[\kappa]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$ to $\mathbf{w}^{[\kappa+1]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$ based on scheme (29) for $\kappa = 0,1,2,\ldots$. Repeat

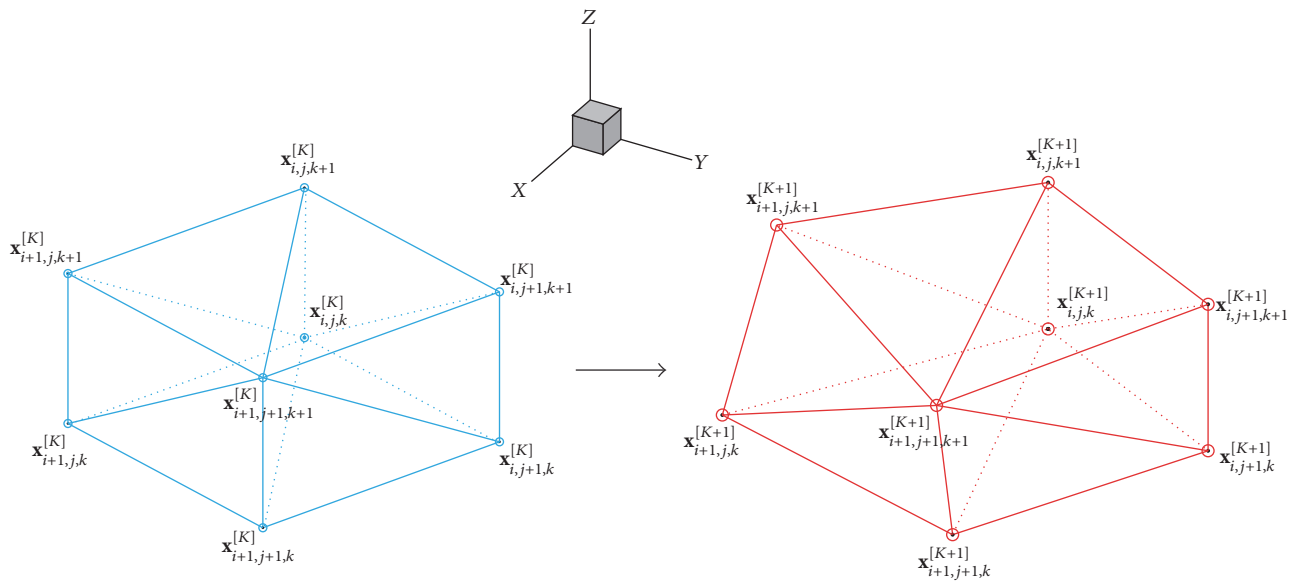

Figure 4: Schematic diagram of the control element $K_{i+1/2,j+1/2,k+1/2}$.

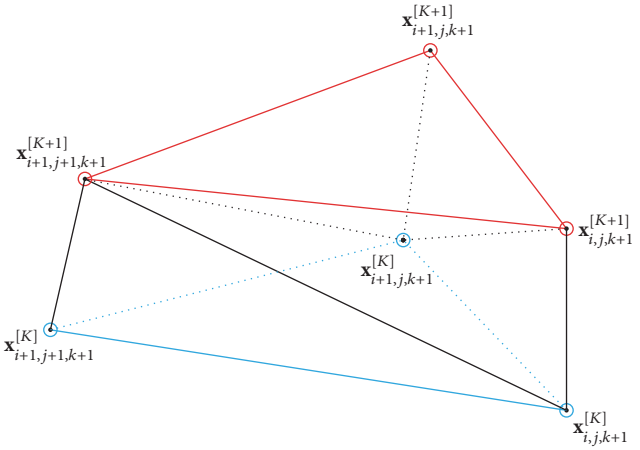

Figure 5: Three-dimensional changed area D_1.

the updating procedure for a fixed number of iterations or until $\|\mathbf{x}^{[\kappa+1]}_{j^1,j^2,\ldots,j^{N_d}} - \mathbf{x}^{[\kappa]}_{j^1,j^2,\ldots,j^{N_d}}\| \leq \varepsilon$. Then, let $\widetilde{\mathbf{x}}_{j^1,j^2,\ldots,j^{N_d}} = \mathbf{x}^{[\kappa+1]}_{j^1,j^2,\ldots,j^{N_d}}$ and $\widetilde{\mathbf{w}}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}} = \mathbf{w}^{[\kappa+1]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$ at time level t_n.

Step 3. Evolve the underlying PDEs using finite volume scheme (12) and time discretization scheme (19) on the new mesh $\{\widetilde{\mathbf{x}}_{j^1,j^2,\ldots,j^{N_d}}\}$ to obtain the numerical approximations $\mathbf{w}^{n+1}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$ at the time level t_{n+1}.

Step 4. If $t_{n+1} \leqslant T$, then let $\mathbf{x}^{[0]}_{j^1,j^2,\ldots,j^{N_d}} = \widetilde{\mathbf{x}}_{j^1,j^2,\ldots,j^{N_d}}$ and $\mathbf{w}^{[0]}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}} = \mathbf{w}^{n+1}_{\bar{j}^1,\bar{j}^2,\ldots,\bar{j}^{N_d}}$, and go to Step 2.

Remark 2. In practice, it is common to use some temporal or spatial smoothing on the weight function ω to obtain smoother meshes in order to avoid very singular mesh and/or large approximation errors near those regions where the solution has a large gradient. A low pass filter should be used to smooth the weight function 2~3 times for each κ: in one-dimensional space:

$$\omega_j = \frac{1}{4}\left(\omega_{j-1} + 2\omega_j + \omega_{j+1}\right); \tag{35}$$

in two-dimensional space:

$$\begin{aligned}\omega_{i,j} \longleftarrow\ & \frac{1}{4}\omega_{i,j} + \frac{1}{8}\left(\omega_{i+1,j} + \omega_{i,j+1} + \omega_{i-1,j} + \omega_{i,j-1}\right) \\ & + \frac{1}{16}\left(\omega_{i+1,j+1} + \omega_{i+1,j-1} + \omega_{i-1,j+1} + \omega_{i-1,j-1}\right);\end{aligned} \tag{36}$$

in three-dimensional space:

$$\begin{aligned}\omega_{i,j,k} \longleftarrow\ & \frac{1}{8}\omega_{i,j,k} + \frac{1}{16}\left(\omega_{i-1,j,k} + \omega_{i+1,j,k} + \omega_{i,j-1,k}\right. \\ & \left. + \omega_{i,j+1,k} + \omega_{i,j,k-1} + \omega_{i,j,k+1}\right) + \frac{1}{32}\left(\omega_{i-1,j-1,k}\right. \\ & \left. + \omega_{i-1,j+1,k} + \omega_{i+1,j-1,k} + \omega_{i+1,j+1,k}\right) + \frac{1}{32}\left(\omega_{i-1,j,k-1}\right. \\ & \left. + \omega_{i-1,j,k+1} + \omega_{i+1,j,k-1} + \omega_{i+1,j,k+1}\right) + \frac{1}{32}\left(\omega_{i,j-1,k-1}\right. \\ & \left. + \omega_{i,j-1,k+1} + \omega_{i,j+1,k-1} + \omega_{i,j+1,k+1}\right) \\ & + \frac{1}{64}\left(\omega_{i-1,j-1,k-1} + \omega_{i+1,j-1,k-1} + \omega_{i-1,j+1,k-1}\right. \\ & \left. + \omega_{i-1,j-1,k+1} + \omega_{i-1,j+1,k+1} + \omega_{i+1,j-1,k+1}\right. \\ & \left. + \omega_{i+1,j+1,k-1} + \omega_{i+1,j+1,k+1}\right).\end{aligned} \tag{37}$$

Remark 3. Because discontinuities may initially exist on boundaries or move to boundaries at a later time, redistribution of boundary points should be made in order to improve the quality of the solution near boundary. For convenience, our attention is restricted to the case in which the physical domain Ω_p is straight boundary. Assume that a new set of grid points $\{\mathbf{x}^{[\kappa+1]}_{j^1,j^2,\ldots,j^{N_d}}\}$ is obtained in Ω_p by solving the moving mesh equation (26). Then the speeds of the internal grid points $\{\mathbf{x}^{[\kappa]}_{j^1,j^2,\ldots,j^{N_d}}\}$ are given by

$$\begin{aligned}&\left(c^{j^1}_{j^1,j^2,\ldots,j^{N_d}}, c^{j^2}_{j^1,j^2,\ldots,j^{N_d}}, \ldots, c^{j^{N_d}}_{j^1,j^2,\ldots,j^{N_d}}\right)\\ &\quad = \mathbf{x}^{[\kappa+1]}_{j^1,j^2,\ldots,j^{N_d}} - \mathbf{x}^{[\kappa]}_{j^1,j^2,\ldots,j^{N_d}},\end{aligned} \tag{38}$$

where $1 \le j^r \le N_{x^r}$, $1 \le r \le N_d$. We assume that the points of boundaries are moving with the same speed as the tangential component of the speed for the internal points adjacent to those boundary point. For an example, when the boundary surface is the boundary surface of $\mathbf{x}_{0,j^2,\ldots,j^{N_d}}$, we have

$$\begin{aligned}&\left(c^{j^1}_{0,j^2,\ldots,j^{N_d}}, c^{j^2}_{0,j^2,\ldots,j^{N_d}}, \ldots, c^{j^{N_d}}_{0,j^2,\ldots,j^{N_d}}\right)\\ &\quad = \left(0, c^{j^2}_{j^1,j^2,\ldots,j^{N_d}}, \ldots, c^{j^{N_d}}_{j^1,j^2,\ldots,j^{N_d}}\right),\end{aligned} \tag{39}$$

where $1 \le j^r \le N_{x^r}$, $1 \le r \le N_d$. Thus new boundary points $\{\mathbf{x}^{[\kappa+1]}_{0,j^2,\ldots,j^{N_d}}\}$ are defined by

$$\begin{aligned}\mathbf{x}^{[\kappa+1]}_{0,j^2,\ldots,j^{N_d}} &= \mathbf{x}^{[\kappa]}_{0,j^2,\ldots,j^{N_d}}\\ &\quad + \left(c^{j^1}_{0,j^2,\ldots,j^{N_d}}, c^{j^2}_{0,j^2,\ldots,j^{N_d}}, \ldots, c^{j^{N_d}}_{0,j^2,\ldots,j^{N_d}}\right),\end{aligned} \tag{40}$$

where $1 \le j^r \le N_{x^r}$, $1 \le r \le N_d$. The redistribution for other boundaries can be carried out in a similar way. Numerical experiments show that the above procedure for moving the boundary points is useful in improving the solution resolution.

4. Parallelization Strategy

The perfect parallel strategy can reach the conflicting goals of portability and high performance. Here, we adopt the software architecture of MPI, which is a de facto standard for parallel programming on distributed memory systems [20]. The primary importance with the MPI model is its discrete memory view in programming, which makes it hard to write and often involves a very long development cycle [21, 22]. Carefully thought-out strategies for partitioning the data and managing the resultant communication are essential for a good MPI program. Secondly, due to the distributed nature of the model, global data may be duplicated for performance reasons, resulting in an increase in the overall memory requirement. Besides, because of machine instabilities and the lack of fault tolerance in the model, it is very challenging to get very large MPI jobs running on large parallel systems (although this is true in general for other programming models). Considering our numerical schemes, the schematic diagram for our parallel computation is given in Figure 6. Here, N corresponds to the number of processors. Next, the illustrations aiming at the critical problems for the schematic diagram will be discussed.

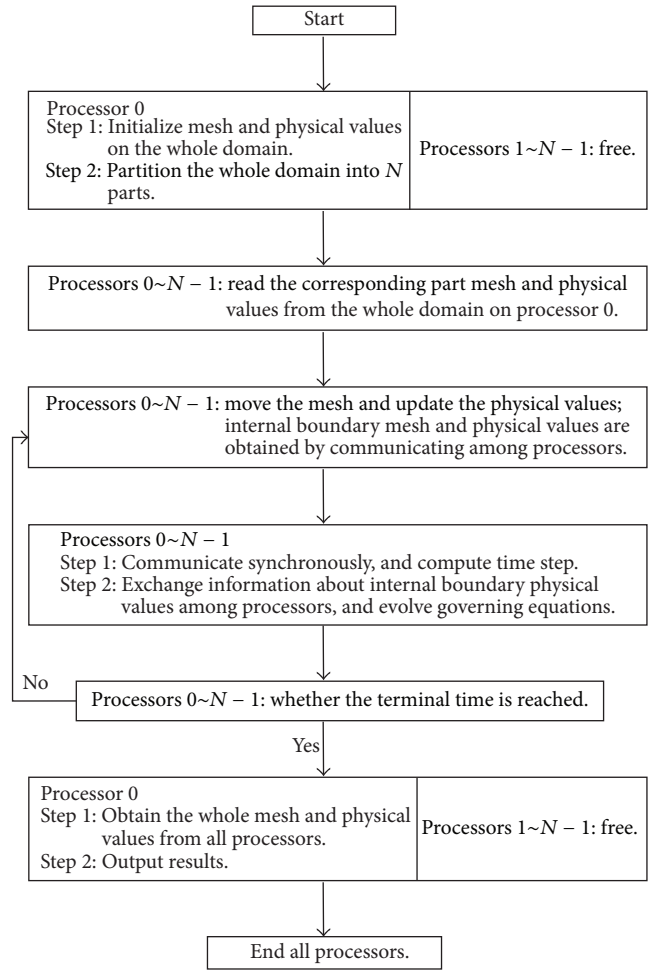

Figure 6: Schematic diagram of the parallel computation.

4.1. Partition the Spatial Mesh Domain. In the parallel computation, the whole domain is divided into some blocks, and each block is distributed to different processors. The distribution is such that each processor gets allocated with one block. Because the computational complexity is related to the number of mesh points, the whole spatial domain is divided according to mesh points rather than spatial coordinates. In addition, the number of mesh points for each processor computation is the same as far as possible. The common partitions for whole spatial domain are given in Figure 7. Figure 7(a) is line partition which fits the stripe domain. The surface partition is shown in Figure 7(b) which can be used for the surface domain and body domain. The body partition in Figure 7(c) is used for the three-dimensional spatial domain. Next, we will discuss relationship between the whole domain and processors, and pseudocodes will be given. It is assumed that *CellNum* is the number of cells in the whole spatial domain. The array *Cell_to_P*(*CellNum*)

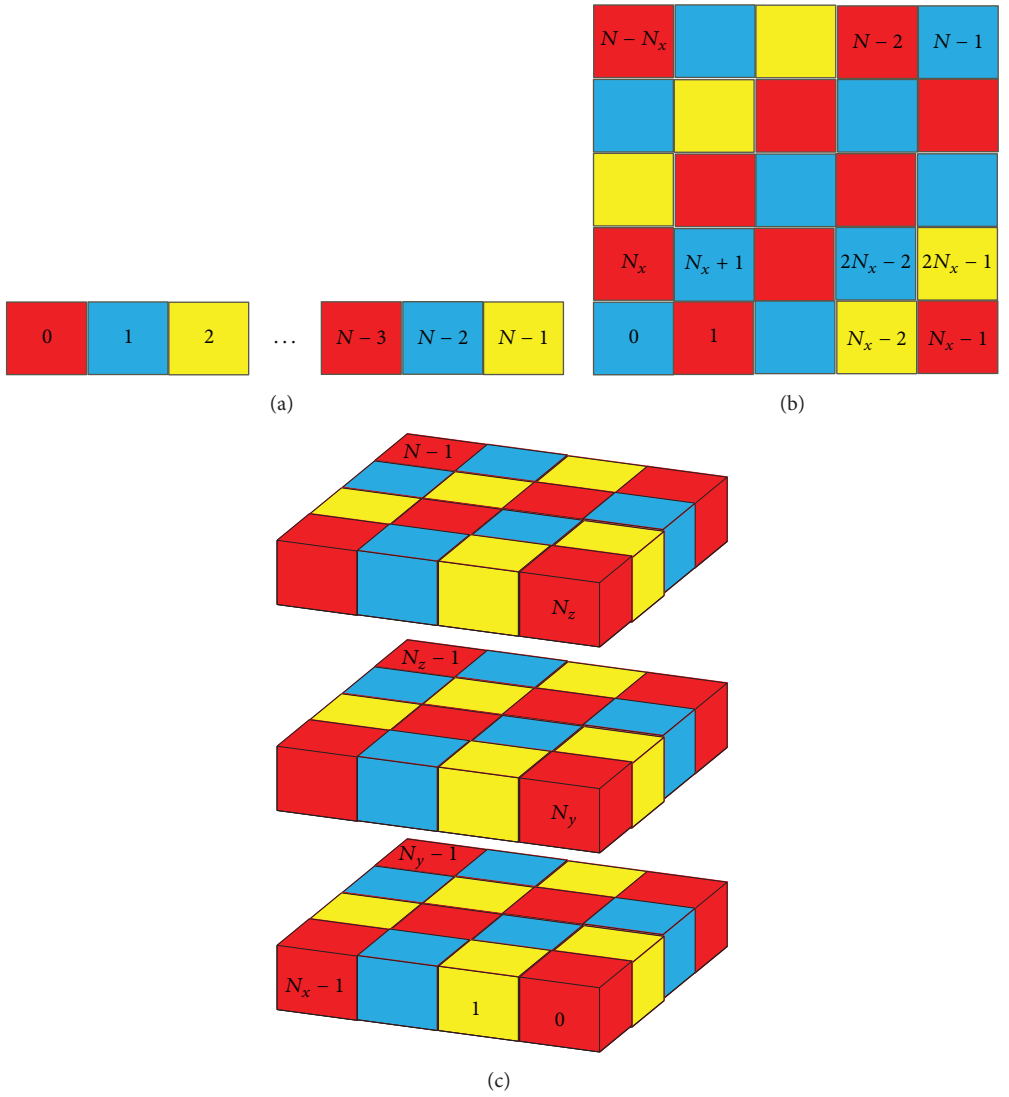

Figure 7: Partitions for spatial domain. (a) Line partition; (b) surface partition; (c) body partition.

is created to store which processor is assigned to each cell, and array $P\_CellNum(N)$ is created to store the number of cells for each processor. The new number $Cell\_in\_P(i)$ in processor for ith cell can be obtained by the following pseudocode:

$P\_CellNum(1{\sim}N) = 0$

$do \quad i = 1, CellNum$

$processor = Cell\_to\_P(i)$

$P\_CellNum(processor) = P\_CellNum(processor) + 1$

$Cell\_in\_P(i) = P\_CellNum(processor)$

$enddo$

In addition, the array $Sub\_to\_entire(max(P\_CellNum(N)), N)$ is created to store the relationship between the whole domain and processors. The pseudocode is

$do \quad i = 1, CellNum$

$processor = Cell\_to\_P(i)$

$sub\_i = Cell\_in\_P(i)$

$Sub\_to\_entire(sub\_i, processor) = i$

$enddo$

4.2. Reduction and Synchronization. The size of spatial mesh is different for each processor, which leads to the time step for each processor being different. Thus time step is needed to synchronize in the computation. *reduction* and *broadcast* are used to achieve this. Let h_t' be the time step of each processor computation with CFL condition. h_t is the time step after synchronization. The codes for MPI Fortran are

$CALL\ MPI\_Reduce(h_t', h_t, 1, MPI\_DOUBLE\_PRECISION, \&\ MPI\_MIN, 0, COMM, IERR)$

$CALL\ MPI\_Bcast(h_t, 1, MPI\_DOUBLE\_PRECISION, 0, COMM, IERR)$

4.3. Processor Communication. For data communication, it is necessary to know neighbor processors for each processor. For line partition, there are two neighbors for current processor to communicate, and if the current processor is on the boundary or its neighbor does not exist, the neighbor processor will be signed *null*. For the adjacent two processors P_A and P_B (P_A is left; P_B is right), there are Ne_j adjacent cells for processors P_A and P_B. Thus, processor P_A needs Ne_j cells to communicate with processor P_B. In the same way, processor P_B needs N_aj cells to communicate with processor P_A. The following pseudocode is given to show the communication between P_A and P_B.

```
C_A/C_B = 1
do  i = 1, P_CellNum(A)
  if  (righcell belong to P_B)  then
    P_A_send(C_A) = i
    P_B_receive(C_A) = i
    C_A = C_A + 1
  endif
enddo
do  i = 1, P_CellNum(B)
  if  (leftcell belong to P_A)  then
    P_B_send(C_B) = i
    P_A_receive(C_B) = i
    C_B = C_B + 1
  endif
enddo
```

The arrays $P\_A\_send(N\_aj)$ and $P\_A\_receive(N\_aj)$ are the cells which P_A send and receive, while The arrays $P\_B\_send(N\_aj)$ and $P\_B\_receive(N\_aj)$ are the cells which P_B send and receive. For the surface partition, there are eight neighbor processors for the current processor, which is shown in Figure 8(a) (here, *Cen* is the current processor). Because neighbors Ne_i, $i = 1, 2, 3, 4$, have the common communicating edge with the current processor *Cen*, we can use the similar way as the line partition to communication. The position of processor Ne_i, $i = 5, 6, 7, 8$, can be obtained with Ne_i, $i = 1, 2, 3, 4$. For example, the following pseudocode is to show the communication between Ne_5 and *Cen*.

```
C_Cen/C_Ne5 = 1
do  i = 1, P_CellNum(Ne5)
  if (righcell belong to Ne1 and abovecell belong
  to Ne4)  then
    Ne5_send(C_Cen) = i
    Cen_receive(C_Cen) = i
    C_Cen = C_Cen + 1
  endif
enddo
do  i = 1, P_CellNum(Cen)
  if  (leftcell belong to Ne4 and belowcell
  belong to Ne1)  then
    Cen_send(C_Ne5) = i
    Ne5_receive(C_Ne5) = i
    C_Ne5 = C_Ne5 + 1
  endif
enddo
```

When the current processor is on the boundary, which is shown in Figure 8(b), the ghost cells Cb_c can be obtained with the current processor *Cen*, other ghost cells Cd_a and Cd_b for *Cen* can be obtained by communicating with Ne_a and Ne_b. The following pseudocode is for *Cen* obtaining Cd_a

```
C_Cen = 1
do  i = 1, P_CellNum(Nea)
  if  (righcell belong to null and belowcell
  belong to Cen)  then
    Nea_send(C_Cen) = i
    Cen_receive(C_Cen) = i
    C_Cen = C_Cen + 1
  endif
enddo
```

For body partition, there are 26 neighbor processors around the current processor *Cen*, which is shown in Figure 8(a). We have divided the neighbors into three kinds Nm_i, Nl_j, and Nv_k, $i = 1, 2, \ldots, 6$, $j = 1, 2, \ldots, 12$, and $k = 1, 2, \ldots, 8$, where they are corresponding to three kinds of communication which is shown in Figure 9(b). Nm_1, Nl_1, and Nv_1 are the examples, of which we will give the pseudocode to show the communication of body partition.

For Nm_1,

```
C_Cen = 1
do i = 1, P_CellNum(A)
  if (behindcell belong to Cen) then
    Nm1_send(C_Cen) = i
    Cen_receive(C_Cen) = i
    C_Cen = C_Cen + 1
  endif
enddo
```

For Nl_1,

```
C_Cen = 1
do i = 1, P_CellNum(A)
  if (behindcell belong to Nm3 and rightcell
  belong to Nm1) then
    Nl1_send(C_Cen) = i
    Cen_receive(C_Cen) = i
    C_Cen = C_Cen + 1
```

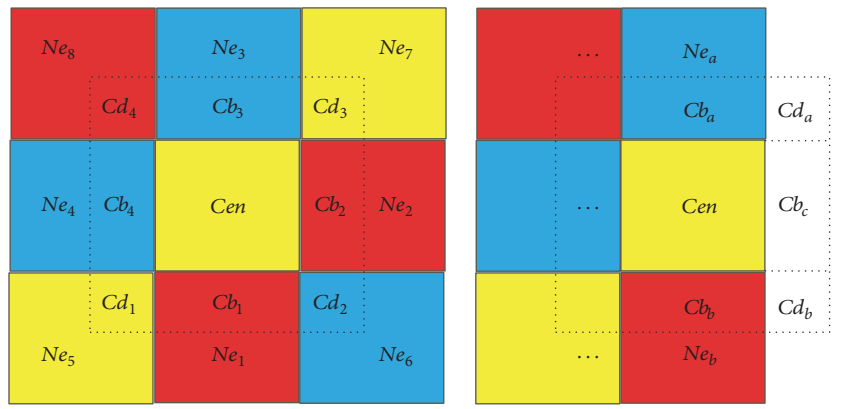

FIGURE 8: Processor communication for surface partition; (a) internal communication; (b) boundary communication.

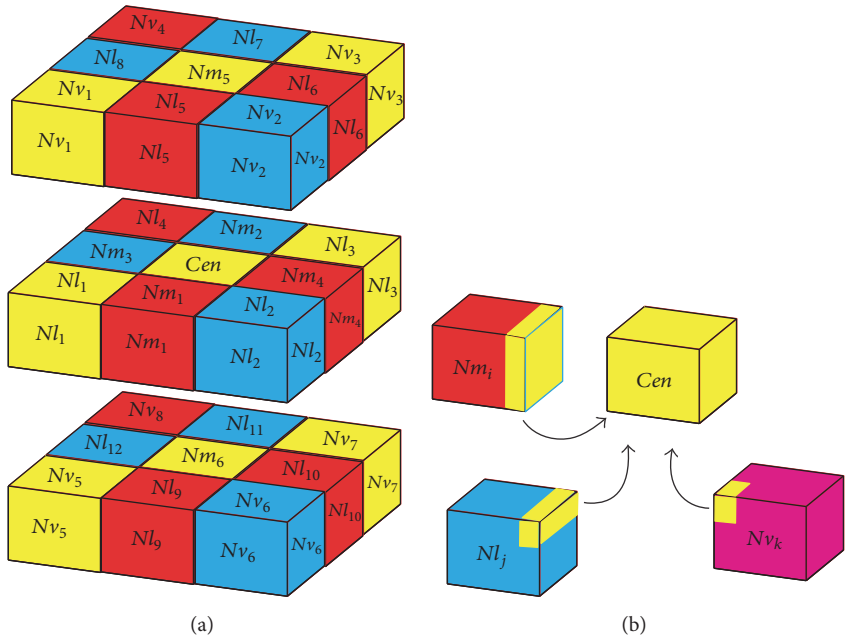

FIGURE 9: Processor communication for body partition; (a) neighbor processors; (b) communication blocks.

endif

enddo

For Nv_1,

$C\_Cen = 1$

do $i = 1, P\_CellNum(A)$

if (belowcell belong to Nl_1 and behindcell belong to Nl_8 & and rightcell belong to Nl_5) *then*

$Nv_1\_send(C\_Cen) = i$

$Cen\_receive(C\_Cen) = i$

$C\_Cen = C\_Cen + 1$

endif

enddo

Figure 10 shows the communications when the current processor is the boundary processor. Figure 10(a) is the surface boundary while Figure 10(b) is the edge boundary. For Figure 10(a), the communication is the same as the situations in Figure 8, while Figure 10(b) is the similar situation as the line partition. Thus, we leave it out.

5. Numerical Examples

In this section, the numerical convergence and the efficiency for our parallel schemes will be shown. In addition, the practical problem will be considered. In our simulation, the parameters are taken as $\gamma = 1.2$, $E_\alpha = 1.7$, $E_\beta = 0.35$, $Q = 52.6$, $k_\alpha = 287.0$, and $k_\beta = 0.0359$. The simulations are taken on the computer group of four nodes. There are two Intel E5620CPUs on each node, and each CPU is quad-core.

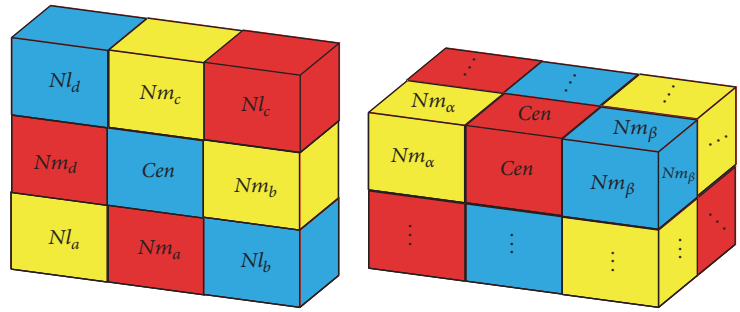

Figure 10: Boundary processor communication; (a) boundary surface processors; (b) boundary edge processors.

5.1. Example 1. Firstly, the numerical accuracy is tested for the one-dimensional Euler equations. A periodic boundary condition is used and the initial conditions are set to be $\rho(x,0) = 1.0 + 0.2\sin(x)$, $u(x,0) = 0.5$, $p(x,0) = 1.0$, $\alpha(x,0) = 0.5 + 0.2\sin(x)$, and $\beta(x,0) = 0.5 + 0.2\sin(x)$. The exact solution for this problem is

$$\begin{aligned} \rho(x,t) &= 1.0 + 0.2\sin(x - 0.5t), \\ u(x,t) &= 0.5, \\ p(x,t) &= 1.0, \\ \alpha(x,t) &= 0.5 + 0.2\sin(x - 0.5t), \\ \beta(x,t) &= 0.5 + 0.2\sin(x - 0.5t). \end{aligned} \tag{41}$$

The final time for this computation is $T = 2.0$. The monitor function for this computation is

$$M = \sqrt{1 + \alpha_1 s_\xi}, \tag{42}$$

where $s = \rho^\gamma / p$ and $\alpha_1 = 0.1$. The parameter γ is 1.2. The computational domain is taken as $[-\pi, \pi]$. The errors and convergence orders in the density are obtained and compared in Table 2. From Table 2, we can see that our scheme is the second-order convergent scheme. In addition, Sod's classical shock tube problem is studied to show that our schemes can capture the singularities and avoid the nonphysical phenomena. The Riemann initial values are

$$(\rho, u, p) = \begin{cases} (1.0, 0.0, 1.0) & \text{if } 0 \le x < 0.5, \\ (0.125, 0.0, 0.1) & \text{if } 0.5 \le x < 1.0 \end{cases} \tag{43}$$

and reflecting boundary conditions at $x = 0$ and $x = 1$. The terminal time is $T = 0.2$. The solutions about density and the trajectory of mesh are shown in Figure 11. There are 100 cells used in the simulation. The monitor function employed for this computation is

$$M = \sqrt{1 + \alpha_1 \left(\frac{\rho_\xi}{\max_\xi (\rho_\xi)} \right) + \alpha_2 \left(\frac{s_\xi}{\max_\xi (s_\xi)} \right)}. \tag{44}$$

It is found that the mesh is adaptive with the solutions and our schemes can capture the discontinuous jump in numerical simulations. The computational complexity and errors analysis are compared between the fixed mesh and moving mesh schemes in Table 3. Comparing (F1) and (M1), we can use 50 mesh points to reach the effect of 100 fixed mesh points while costing the same CPU time. In addition, the different arc-length parameters with 100 mesh points are simulated to compare 200 and 250 fixed mesh points schemes. Our schemes can reach better results with less mesh points. Particularly, L^2, L^∞ errors are obviously reduced. Thus we can conclude that our schemes can capture the singularities and avoid the nonphenomena.

Table 2: Errors and convergence orders in the density.

Cells	Fixed mesh		Moving mesh	
	Error	Order	Error	Order
20	$3.4932E-02$	—	$3.5100E-02$	—
40	$1.0219E-02$	1.7733	$1.0244E-02$	1.7767
80	$2.6508E-03$	1.9468	$2.6380E-03$	1.9572
160	$6.3429E-04$	2.0632	$6.2496E-04$	2.0776
320	$1.4987E-04$	2.0815	$1.4569E-04$	2.1009

5.2. Example 2. This is the two-dimensional example. Firstly, the numerical accuracy is tested with the periodic boundary condition and initial conditions $\rho(x,0) = 1.0 + 0.2\sin(x + y)$, $u(x,0) = v(x,0) = 0.5$, $p(x,0) = 1.0$, $\alpha(x,0) = 0.5 + 0.2\sin(x + y)$, and $\beta(x,0) = 0.5 + 0.2\sin(x + y)$. The source terms are adjusted so that the governing equations (2) can fit the exact solution

$$\begin{aligned} \rho(x,t) &= 1.0 + 0.2\sin(x + y - t), \\ u(x,t) &= v(x,t) = 0.5, \\ p(x,t) &= 1.0, \\ \alpha(x,t) &= 0.5 + 0.2\sin(x + y - t), \\ \beta(x,t) &= 0.5 + 0.2\sin(x + y - t). \end{aligned} \tag{45}$$

Here the monitor function is about entropy (42). The final time for the computation is $T = 2.0$. The errors and convergence orders about density are shown in Table 4, which implies that our schemes are the second-order convergence.

Table 3: Comparison of various fixed and moving mesh.

Description	CPU time	Time steps	L^1 error	L^2 error	L^∞ error
Fixed mesh (cells):					
(F1) 100	$1.56E-02$	48	$7.9754E-03$	$1.5122E-02$	$6.4432E-02$
(F2) 200	$6.24E-02$	97	$4.2999E-03$	$1.0988E-02$	$7.1633E-02$
(F3) 250	$9.36E-02$	121	$3.2221E-03$	$8.5794E-03$	$6.7484E-02$
Moving mesh $(cells, \alpha_1, \alpha_2)$*:*					
(M1) 50, 20, 50	$1.56E-02$	69	$7.8921E-03$	$1.4041E-02$	$6.4284E-02$
(M2) 100, 20, 100	$4.68E-02$	129	$4.2209E-03$	$1.0347E-02$	$7.1121E-02$
(M3) 100, 10, 10	$7.80E-02$	236	$3.1650E-03$	$7.9560E-03$	$6.5914E-02$

Table 4: Errors and convergence orders in the density.

Cells	Fixed mesh		Moving mesh	
	Error	Order	Error	Order
20	$2.4050E-01$	—	$2.4980E-01$	—
40	$6.6037E-02$	1.8647	$6.6661E-02$	1.9058
80	$1.6605E-02$	1.9916	$1.6590E-02$	2.0056
160	$3.9552E-03$	2.0698	$3.9543E-03$	2.0688
320	$9.2949E-04$	2.0892	$9.2871E-04$	2.0901

Next, we apply our schemes to a two-dimensional explosion problem. The computational domain is illustrated in Figure 12. The boundary condition is reflective, and the initial conditions in the unreacted region are $(\rho, u, v, w, \alpha, \beta) = (1, 0, 0, 0, 1, 1)$. The initial conditions in the reacted region are obtained by the *ZND* model [23]. There are 240×240 cells in our computation. The monitor function

$$M = \sqrt{1 + \alpha_1 \rho^2 + \alpha_2 \left|\nabla \rho\right|^2} \tag{46}$$

is used in the computation. The terminal time is 0.5. The parallel cost with different processors is compared in Tables 5 and 6. Table 5 is the comparison of parallel efficiency with different processors for line partition, while Table 6 is for the surface partition. The *Speedup* and *Efficiency* for parallel computations are defined by

$$\begin{aligned} &\text{Speedup with } N \text{ processors} \\ &\quad = \frac{\text{Cost time with one processor}}{\text{Cost time with } N \text{ processors}}, \\ &\text{Efficiency with } N \text{ processors} \\ &\quad = \frac{\text{Speedup with } N \text{ processors}}{N}. \end{aligned} \tag{47}$$

Both tables illustrate that our parallel schemes are effective and the computational cost time is reduced. Figure 13 is the computational results at time 0.15, and Figure 14 is at time 0.3. The solutions figures show that our schemes can capture the shock waves and the mesh trajectory is adaptive.

5.3. Example 3. The last example is the three-dimensional problem. The numerical exact test is taken firstly. The periodic boundary condition is used and the initial conditions are set

Table 5: Parallel efficiency with different processors for line partition.

Number of processors	CPU time	Speedup	Efficiency
1 (1×1)	15778	1.0	1.0
2 (2×1)	7894	1.9987	0.9994
3 (3×1)	5281	2.9877	0.9959
4 (4×1)	3976	3.9683	0.9921
5 (5×1)	3354	4.7042	0.9408
6 (6×1)	2891	5.4576	0.9096

Table 6: Parallel efficiency with different processors for surface partition.

Number of processors	CPU time	Speedup	Efficiency
1 (1×1)	15778	1.0	1.0
4 (2×2)	3965	3.9793	0.9948
9 (3×3)	2053	7.6853	0.8539
16 (4×4)	1620	9.7395	0.6087

to be $\rho(x, 0) = 1.0 + 0.2 \sin(x + y + z)$, $u(x, 0) = v(x, 0) = w(x, 0) = 0.5$, $p(x, 0) = 1.0$, $\alpha(x, 0) = 0.5 + 0.2 \sin(x + y + z)$, and $\beta(x, 0) = 0.5 + 0.2 \sin(x + y + z)$. The terminal time is $T = 2.0$ and the monitor function is about entropy (42). The exact solution is

$$\begin{aligned} &\rho(x, t) = 1.0 + 0.2 \sin(x + y + z - 1.5t), \\ &u(x, t) = v(x, t) = w(x, t) = 0.5, \\ &p(x, t) = 1.0, \\ &\alpha(x, t) = 0.5 + 0.2 \sin(x + y + z - 1.5t), \\ &\beta(x, t) = 0.5 + 0.2 \sin(x + y + z - 1.5t). \end{aligned} \tag{48}$$

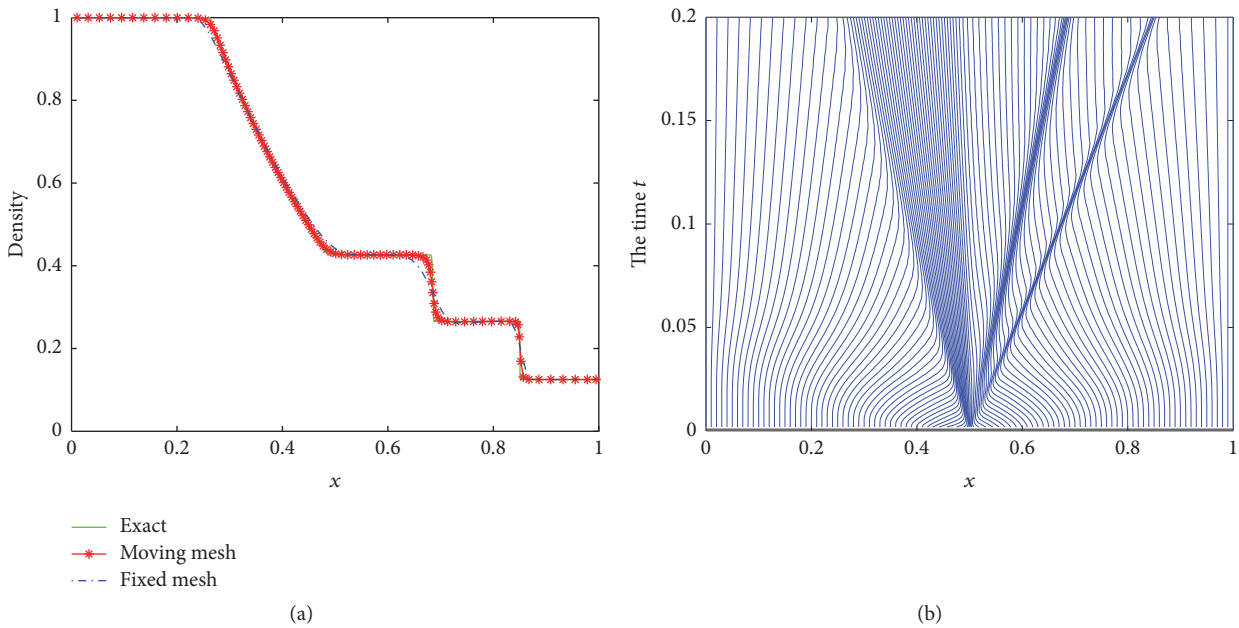

Figure 11: The simulation results; (a) solution about density; (b) trajectory of mesh.

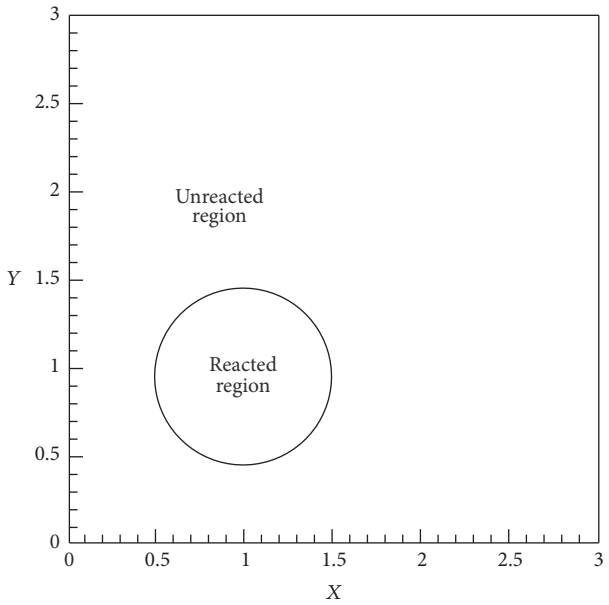

Figure 12: The computational domain.

The errors and convergence orders about density are shown in Table 7, which implies that our schemes are the second-order convergence. The computational domain and boundary conditions are shown in Figure 15. The initial values for reacted region and unreacted region are the same as Example two. Table 8 is used to show the parallel efficiency for body partition, which illustrates that the computational time is reduced and our parallel scheme can deal with the three-dimensional problems. There are $120 \times 120 \times 120$ cells in our simulations. The monitor function

$$M = \sqrt{1 + \alpha_1 \rho^2 + \alpha_2 \left|\nabla \rho\right|^2} \tag{49}$$

Table 7: Errors and convergence orders in the density.

Cells	Fixed mesh		Moving mesh	
	Error	Order	Error	Order
30	1.0109	—	1.0138	—
40	$5.8547E-01$	1.8987	$5.8571E-01$	1.9071
50	$3.8052E-01$	1.9310	$3.8027E-01$	1.9358
60	$2.6448E-01$	1.9952	$2.6421E-01$	1.9970
70	$1.9246E-01$	2.0620	$1.9223E-01$	2.0633

Table 8: Parallel efficiency with different processors for body partition.

Number of processors	CPU time	Speedup	Efficiency
1 ($1 \times 1 \times 1$)	24298	1.0	1.0
8 ($2 \times 2 \times 2$)	3370	7.2101	0.9013
27 ($3 \times 3 \times 3$)	1145	21.2209	0.7859
32 ($4 \times 4 \times 2$)	1053	23.0750	0.7211

is used in the computation. The terminal time is 0.5. Figures 16 and 17 show the adaptive solutions. Figure 16 shows the three-dimensional sliced effect of the adaptive mesh with the solutions at different time points. Figure 17(a) is the simulated result of the block in the center of the domain about density at time 0.15, while Figure 17(b) is a x-direction of the sliced simulation about density at time 0.3. The spatial cells are distorted to adapt to the shock wave front. They show us that our schemes can capture shock waves.

6. Conclusion and Discussion

In this paper, we have discussed the parallel computation for pseudo arc-length moving mesh schemes. Different from the traditional Euler numerical scheme, the communications

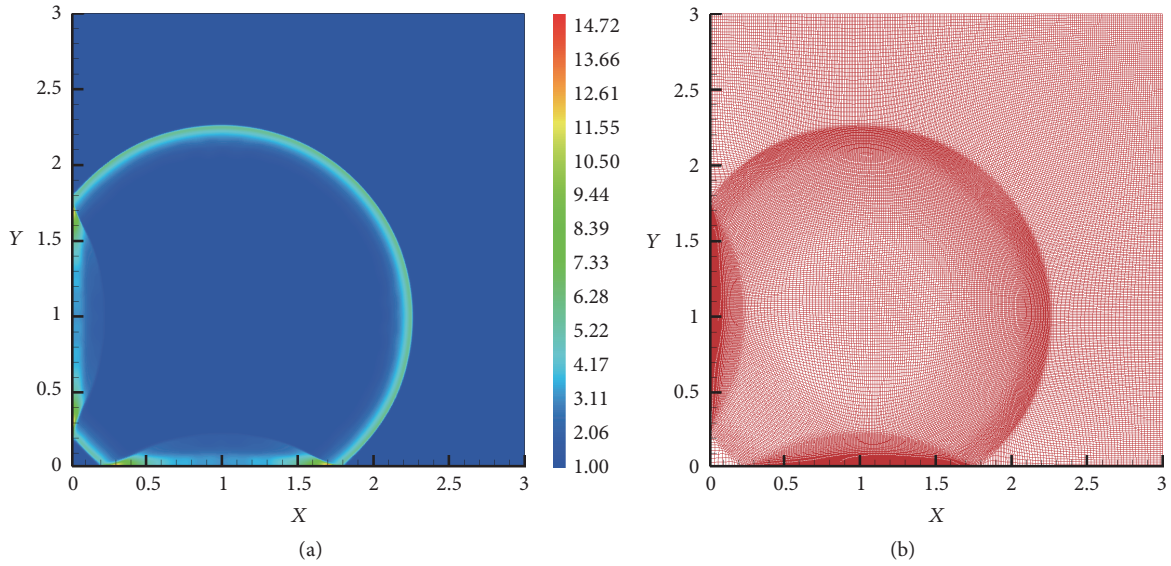

Figure 13: The computational results at time 0.15; (a) solution about density; (b) trajectory of mesh.

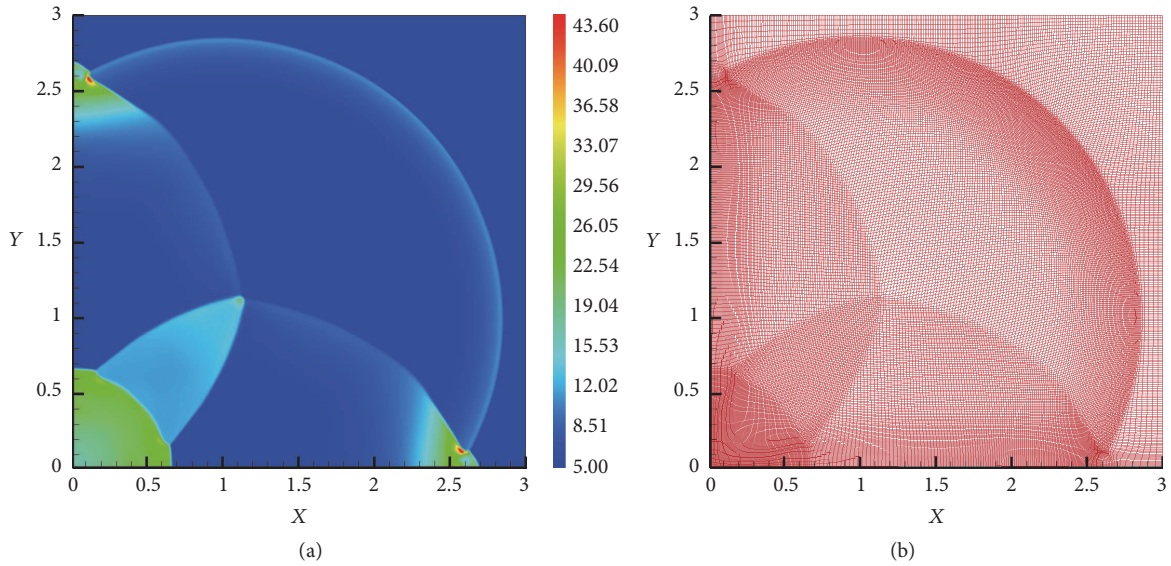

Figure 14: The computational results at time 0.3; (a) solution about pressure; (b) trajectory of mesh.

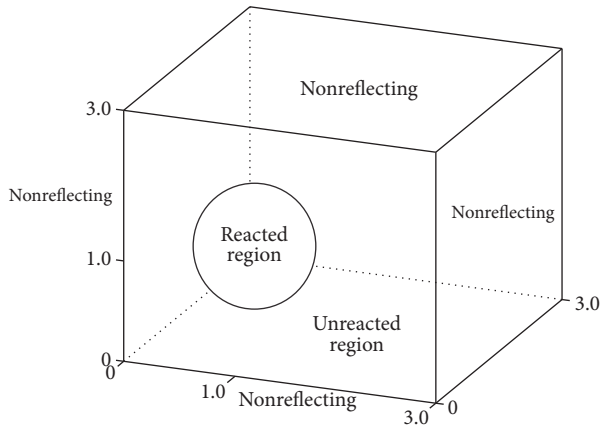

Figure 15: The computational domain.

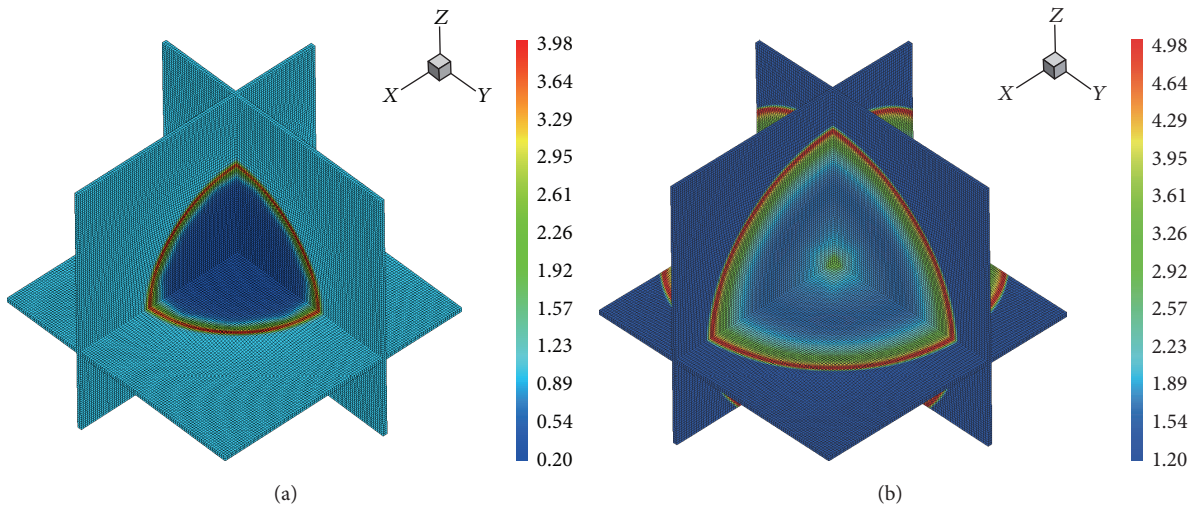

Figure 16: The simulation solutions; (a) solution about density at time 0.15; (b) solution about pressure at time 0.3.

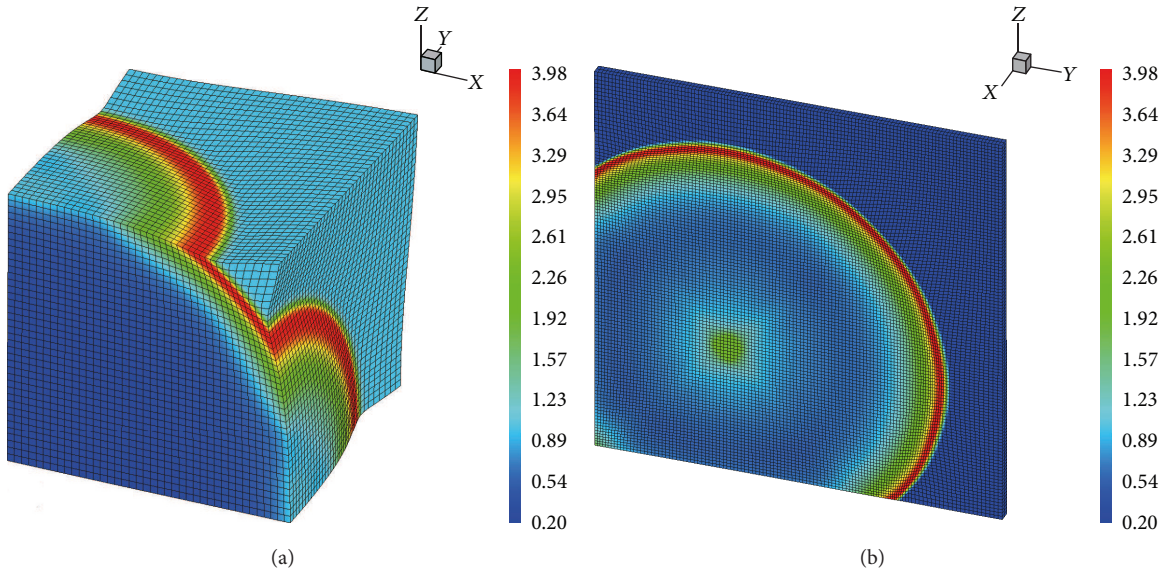

Figure 17: The simulation solutions; (a) a block solutions about density at time 0.15; (b) a piece of solutions about density at time 0.3.

between processors are more complex, which include adjacent processor and diagonal processor. The parallel schemes including line, surface, and body partition are all considered in this work and the pseudocodes and schematic diagram are given. Finally, the numerical examples are shown to illustrate that our parallel schemes are effective and the pseudo arc-length moving mesh schemes are convergent and can be used to capture the shock wave.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 11390363 and 11532012).

References

[1] F. Davis, Detonation, Univ. California Press, 1979.

[2] G. Wang, D. Zhang, K. Liu, and J. Wang, "An improved CE/SE scheme for numerical simulation of gaseous and two-phase detonations," *Computers and Fluids. An International Journal*, vol. 39, no. 1, pp. 168–177, 2010.

[3] A. Harten and J. M. Hyman, "Self-adjusting grid methods for one-dimensional hyperbolic conservation laws," *Journal of Computational Physics*, vol. 50, no. 2, pp. 235–269, 1983.

[4] J. M. Stockie, J. A. Mackenzie, and R. . Russell, "A moving mesh method for one-dimensional hyperbolic conservation laws," *SIAM Journal on Scientific Computing*, vol. 22, no. 5, pp. 1791–1813, 2000.

[5] X. Yang, W. Huang, and J. Qiu, "A moving mesh WENO method for one-dimensional conservation laws," *SIAM Journal on Scientific Computing*, vol. 34, no. 4, pp. A2317–A2343, 2012.

[6] H. Tang and T. Tang, "Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws," *SIAM Journal on Numerical Analysis*, vol. 41, no. 2, pp. 487–515, 2003.

[7] J. Han and H. Tang, "An adaptive moving mesh method for two-dimensional ideal magnetohydrodynamics," *Journal of Computational Physics*, vol. 220, no. 2, pp. 791–812, 2007.

[8] P. He and H. Tang, "An adaptive moving mesh method for two-dimensional relativistic magnetohydrodynamics," *Computers and Fluids. An International Journal*, vol. 60, pp. 1–20, 2012.

[9] J. Ning, X. Yuan, T. Ma, and C. Wang, "Positivity-preserving moving mesh scheme for two-step reaction model in two dimensions," *Computers and Fluids. An International Journal*, vol. 123, pp. 72–86, 2015.

[10] M. G. Knepley and D. A. Karpeev, "Mesh algorithms for PDE with sieve I: mesh distribution," *Scientific Programming*, vol. 17, no. 3, pp. 215–230, 2009.

[11] K. Morris, D. W. I. Rouson, M. N. Lemaster, and S. Filippone, "Exploring capabilities within ForTrilinos by solving the 3D Burgers equation," *Scientific Programming*, vol. 20, no. 3, pp. 275–292, 2012.

[12] M. Mubarak, S. Seol, Q. Lu, and M. S. Shephard, "A parallel ghosting algorithm for the flexible distributed mesh database," *Scientific Programming*, vol. 21, no. 1-2, pp. 17–42, 2013.

[13] C. Wang, X. Dong, and C.-W. Shu, "Parallel adaptive mesh refinement method based on WENO finite difference scheme for the simulation of multi-dimensional detonation," *Journal of Computational Physics*, vol. 298, pp. 161–175, 2015.

[14] S. Taki and T. Fujiwara, "Numerical simulation on the establishment of gaseous detonation," *Prog Astronaut Aeronaut*, vol. 9, pp. 186–200, 1984.

[15] B. van Leer, "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method," *Journal of Computational Physics*, vol. 32, no. 1, pp. 101–136, 1979.

[16] C.-W. Shu and S. Osher, "Efficient implementation of essentially nonoscillatory shock-capturing schemes," *Journal of Computational Physics*, vol. 77, no. 2, pp. 439–471, 1988.

[17] S. Gottlieb and C.-W. Shu, "Total variation diminishing Runge-Kutta schemes," *Mathematics of Computation*, vol. 67, no. 221, pp. 73–85, 1998.

[18] K. Chen, "Error equidistribution and mesh adaptation," *SIAM Journal on Scientific Computing*, vol. 15, no. 4, pp. 798–818, 1994.

[19] X. Wang, T.-B. Ma, and J.-G. Ning, "A pseudo arc-length method for numerical simulation of shock waves," *Chinese Physics Letters*, vol. 31, no. 3, Article ID 030201, 2014.

[20] H. Jin, D. Jespersen, P. Mehrotra, R. Biswas, L. Huang, and B. Chapman, "High performance computing using MPI and OpenMP on multi-core parallel systems," *Parallel Computing*, vol. 37, no. 9, pp. 562–575, 2011.

[21] L. Hochstein and V. R. Basili, "The ASC-alliance projects: a case study of large-scale parallel scientific code development," *Computer*, vol. 41, no. 3, pp. 50–58, 2008.

[22] L. Hochstein, F. Shull, and L. B. Reid, "The role of MPI in development time: A case study," in *Proceedings of the 2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2008*, usa, November 2008.

[23] W. Fickett and W. W. Wood, "Flow calculations for pulsating one-dimensional detonations," *Physics of Fluids*, vol. 9, no. 5, pp. 903–916, 1966.