

An interface treating technique for compressible multi-medium flow with Runge-Kutta discontinuous Galerkin method

Chunwu Wang¹ and Chi-Wang Shu²

Abstract

The high-order accurate Runge-Kutta discontinuous Galerkin (RKDG) method is applied to the simulation of compressible multi-medium flow, generalizing the interface treating method given in [9]. In mixed cells, where the interface is located, Riemann problems are solved to define the states on both sides of the interface. The input states to the Riemann problem are obtained by extrapolation to the cell boundary from solution polynomials in the neighbors of the mixed cell. The level set equation is solved by using a high-order accurate RKDG method for Hamilton-Jacobi equations, resulting in a unified DG solver for the coupled problem. The method is conservative if we include the states in the mixed cells, which are however not used in the updating of the numerical solution in other cells. The states in the mixed cells are plotted to better evaluate the conservation errors, manifested by overshoots / undershoots when compared with states in neighboring cells. These overshoots / undershoots in mixed cells are problem dependent and change with time. Numerical examples show that the results of our scheme compare well with other methods for one and two-dimensional problems. In particular, the algorithm can capture well complex flow features of the one-dimensional shock entropy wave interaction problem and two-dimensional shock-bubble interaction problem.

AMS subject classification: 76M10, 76N10

Keywords: multi-medium flow, Runge-Kutta discontinuous Galerkin method, medium interface, interface treating method, compressible flow

¹College of Science, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu 210016, China. E-mail: wangcw@nuaa.edu.cn. Research is partially supported by the National Natural Science Foundation of China grant 10601023.

²Division of Applied Mathematics, Brown University, Providence, RI 02912, USA. E-mail: shu@dam.brown.edu. Research is partially supported by DOE grant DE-FG02-08ER25863 and NSF grant DMS-0809086.

1 Introduction

In this paper we develop a Runge-Kutta discontinuous Galerkin (RKDG) method for solving compressible multi-medium flow. The RKDG method was first introduced in [13, 12] for one-dimensional scalar conservation laws with the explicit TVD time discretizations developed in [32]. The extension of the RKDG schemes to one-dimensional systems was carried out in [11] and the multidimensional case for scalar conservation laws was given in [10]. The RKDG method for multi-dimensional systems [15] was completed by improving the work in [14] where the piecewise linear approximations were applied for the two-dimensional Euler equations of gas dynamics. The method gained more and more attention and has been widely used for compressible and incompressible flows, both steady and unsteady, as well as those under turbulent conditions. The method has also been used to simulate multi-medium flow [28, 29].

Front-tracking method developed by Glimm et al. [19, 20, 21] is an important class of numerical methods for multi-medium flow. In this method the interface moves with the mesh and the solution discontinuity is kept sharply. It is easy to implement for one-dimensional cases, while for multi-dimensional problems it is far less trivial as the reconstruction of the interface is needed at all times and could become very complicated for large interfacial deformation. Furthermore, mesh merging/splitting is needed to avoid the occurrence of small size cells.

It has been found that conservative Eulerian method will produce numerical oscillations near interfaces in multi-medium flow problems [1, 2]. The maintenance of the pressure equilibrium is difficult across the interface and pressure oscillations occur due to the different equations of state (EOS). In order to overcome this difficulty, various strategies have been pursued in many directions. The ghost fluid method (GFM) presents a fairly simple and flexible way to treat multi-medium flows and is easily extended to multi-dimensions. The GFM developed in [16, 17] works by defining the ghost nodes with the states at nodes close to the interface so that computation can be carried out as if in a single medium. The

numerical schemes for single medium flow can be employed without any changes and the methods are easily extended to multi-dimensions. The GFM has been successfully applied to various areas such as the treatment of discontinuities in compressible and incompressible flows [7, 22, 25], flame and detonation fronts [18, 26], and fluid-structure coupling [16]. In [23, 24] Liu et al. presented a new way to define the ghost nodes using Riemann solvers to make GFM behave better for problems with strong shocks impacting on the interface. In the GFM developed in [35, 36], not only the states at the ghost nodes are updated with the Riemann solver, but also the real nodes are redefined. In doing so, the real fluid states next to the interface instead of the ghost fluid states are predicted by solving a Riemann problem and the ghost fluid states are then obtained by solving an advection equation. The obtained method is more robust with less conservation errors. The modified GFM in [23, 24] and in [35, 36] work well for multi-medium flow, but they are still nonconservative schemes.

In addition to being available for arbitrarily high-order accuracy, the discontinuous Galerkin (DG) method works as a very compact numerical scheme. This is due to the fact that the solution representation in each cell is kept independent of the solutions in other cells, with communication occurring only with adjacent cells sharing a common edge. This makes the DG method extremely flexible in solving problems with complicated fluid boundary/interface. In this work, following the interface treating method given in [9], we extend the RKDG method to multi-medium flow problems. We track the medium interface by solving the level set equation. Riemann problems are constructed at the interface with the input states defined by using the solution representation in a single-medium cell closest to the interface in x and y directions, respectively. The Riemann solvers are duplicated as the states on both sides of the interface in the mixed cells. Therefore, the single-medium RKDG method can be employed to discretize the governing equations with small changes in the mixed cells. By doing so an interface tracking algorithm is developed and it is easy to evaluate the conservation errors in the mixed cells. This method avoids using mixed-cell information and is oscillation free with sharp medium interface by neglecting the mixed cells.

Comparing with the method developed in [28, 29], the main difference with our method is in the treatment of the interface. The level set equation is also solved by using the high-order accurate RKDG method for Hamilton-Jacobi equations [8], thereby producing a unified DG solver. Our method is conservative if we include the states in the mixed cells, which are however not used in the updating of the numerical solution in other cells. The states in the mixed cells are plotted to better evaluate the conservation errors, manifested by overshoots / undershoots when compared with states in neighboring cells. These overshoots / undershoots in mixed cells are problem dependent and change with time. Numerical examples show that the results of our scheme compare well with other methods for one and two-dimensional problems. In particular, the algorithm can capture well complex flow features of the one-dimensional shock entropy wave interaction problem and two-dimensional shock-bubble interaction problem.

The paper is organized as follows. In section 2, the governing equations, equations of states for gases and water, and the level set equation are provided. In section 2.3, a detailed description of the second- and third-order RKDG method [10, 15] for two-dimensional problems is presented. The implementation of the interface treating method and the coupling with the high accurate RKDG method is given section 3. In section 4 some comparisons are made between the simulations by second- and third-order RKDG methods and exact solutions for one-dimensional problems. The evolutions of two-dimensional shock-bubble interaction are also investigated. A brief conclusion is given in section 5.

2 Equations and numerical method

2.1 Governing equation

We shall limit our description to two-dimensional cases. The two-dimensional computational domain with the Cartesian coordinate system $\vec{x} = (x, y)$ is filled with two compressible fluids separated by a free moving interface. One fluid is air while the other is another gas or water.

The two-dimensional system for a compressible fluid can be written as follows:

$$\frac{\partial \vec{U}}{\partial t} + \text{div} \vec{F}(\vec{U}) = 0 \quad (2.1)$$

where

$$\begin{aligned} \vec{U} &= [\rho, \rho u, \rho v, E]^T, \\ \vec{F}(\vec{U}) &= [\vec{F}_1(\vec{U}), \vec{F}_2(\vec{U})], \\ \vec{F}_1(\vec{U}) &= [\rho u, \rho u^2 + p, \rho uv, (E + p)u]^T, \\ \vec{F}_2(\vec{U}) &= [\rho v, \rho uv, \rho v^2 + p, (E + p)v]^T. \end{aligned}$$

Here ρ is the density, u and v are the velocity components in the respective x and y directions, p is the pressure and E is the total energy per unit volume. The total energy is the sum of internal energy and kinetic energy,

$$E = \rho e + \frac{1}{2} \rho (u^2 + v^2) \quad (2.2)$$

where e is the internal energy per unit mass. For a closure of the system (2.1), the equation of state (EOS) is required. In the present work, our interest is centered on the compressible gas and water. The EOS for gas or water medium can be written uniformly as

$$p = (\gamma - 1) \rho e - \gamma B \quad (2.3)$$

where γ and B are treated as fluid constants, and will be specified in section 4.

2.2 Level set equation

To track the moving fluid interface, we employ the level set technique [27, 3]. The level set equation can be written as

$$\frac{\partial \phi}{\partial t} + \vec{u} \cdot \nabla \phi = 0. \quad (2.4)$$

In general $\phi(\vec{x}, t)$ starts off as a signed distance function. For complex interface evolution, it has been found that a higher-order accurate discretization can help to maintain the accurate

position of the interface [17]. In this work, the third-order accurate RKDG method for Hamilton-Jacobi equations [8] is used to track the interface.

When large gradient appears in the velocity field, the level set contours may be severely distorted or even the code may collapse. To overcome this difficulty, the extension velocity [4] is applied by solving a convection equation [17] to extrapolate the interface velocity to the whole computational domain. By solving equation (2.4) with the extension velocity, the uniformly distributed level set contour and accurate interface position can be obtained.

2.3 The implementation of the RKDG method

By defining the interface boundary condition (details can be found in next section), the high order accurate RKDG method can be extended to multi-medium flow. The one-dimensional RKDG method can be found in [11]. In this section we will focus on the description of the two-dimensional RKDG method [10, 15]. The coupling with the interface treating method will be presented in the next section.

The 2D weak formulation of (2.1) over cell K is given by

$$\frac{\partial}{\partial t} \int_K \vec{U}(\vec{x}, t) \phi(\vec{x}) d\vec{x} + \sum_{e \in \partial K} \int_e \vec{F}(\vec{U}(\vec{x}, t)) \cdot n_{e,K} \phi(\vec{x}) d\Gamma - \int_K \vec{F}(\vec{U}(\vec{x}, t)) \cdot \text{grad} \phi(\vec{x}) d\vec{x} = 0 \quad (2.5)$$

where $\vec{x} = (x, y)$, $\phi(\vec{x})$ is a test function, $n_{e,K}$ denotes the outward unit normal to edge e of cell K . The boundary integral in (2.5) is approximated by a Gaussian quadrature with sufficient accuracy as

$$\int_e \vec{F}(\vec{U}(\vec{x}, t)) \cdot n_{e,K} \phi(\vec{x}) d\Gamma \approx \sum_{l=1}^L \omega_l \vec{F}(\vec{U}(\vec{x}_{el}, t)) \cdot n_{e,K} \phi(\vec{x}_{el}) |e|. \quad (2.6)$$

The flux $\vec{F}(\vec{U}(\vec{x}_l, t)) \cdot n_{e,K}$ in (2.6) is replaced by a numerical flux. For example, the Lax-Friedrichs flux can be used

$$\mathcal{F}(\vec{U}(\vec{x}_{el}, t)) \cdot n_{e,K} = \frac{1}{2} [(\vec{F}(\vec{U}^+(\vec{x}_{el}, t)) + \vec{F}(\vec{U}^-(\vec{x}_{el}, t))) \cdot n_{e,K} - \alpha (\vec{U}^+(\vec{x}_{el}, t) - \vec{U}^-(\vec{x}_{el}, t))] \quad (2.7)$$

where \vec{x}_{el} is the Gaussian point at edge e , α is the biggest eigenvalue of the Jacobian $\frac{\partial}{\partial \vec{U}}(\vec{F}(\vec{U}) \cdot n_{e,K})$ for (\vec{x}, t) over the neighboring cells of edge e , and the superscripts \pm in $\vec{U}^\pm(\vec{x}_{el}, t)$ refer

to the values of \vec{U} in the neighboring cell and in the current cell K , respectively. We replace the volume integral in (2.5) by a numerical quadrature with sufficiently high order accuracy,

$$\int_K \vec{F}(\vec{U}(\vec{x}, t)) \cdot \text{grad}\phi(\vec{x})d\vec{x} \approx \sum_{m=1}^M \bar{\omega}_m \vec{F}(\vec{U}(\vec{x}_{K_m}, t)) \cdot \text{grad}\phi(\vec{x}_{K_m})|K|. \quad (2.8)$$

See [10, 15] for details of the numerical quadrature.

Now we give the details of the RKDG method. Consider a rectangular cell $K = I_{i,j} = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}]$ with the cell center $\vec{x}_{i,j} = (x_i, y_j)$. For simplicity, we consider a uniform cell size in x and y directions, denoted by Δx and Δy , respectively, and $h = \max(\Delta x, \Delta y)$. The RKDG method can of course be implemented on any non-uniform or unstructured meshes. Replacing the exact solution \vec{U} by the approximate solution \vec{U}^h , the test function $\phi \in V_h^k(K)$, and $K = I_{i,j}$, where $V_h^k(K)$ is the numerical solution as well as the test function space given as $V_h^k(K) = \{p : p|_K \in P^k(K)\}$, $P^k(K)$ being the space of polynomials with degree $\leq k$, we obtain the semi-discrete scheme as finding $\vec{U}^h \in V_h^k(K)$, such that

$$\begin{aligned} \frac{\partial}{\partial t} \int_{I_{i,j}} \vec{U}^h(\vec{x}, t) \phi(\vec{x})d\vec{x} = & - \sum_{e \in \partial I_{i,j}} \sum_{l=1}^L \omega_l \vec{F}(\vec{U}^h(\vec{x}_{el}, t)) \cdot n_{e, I_{i,j}} \phi(\vec{x}_{el})|e| \\ & + \sum_{m=1}^M \bar{\omega}_m \vec{F}(\vec{U}^h(\vec{x}_{I_{i,j}m}, t)) \cdot \text{grad}\phi(\vec{x}_{I_{i,j}m})|I_{i,j}| \end{aligned} \quad (2.9)$$

holds for all test function $\phi \in V_h^k(K)$. The local orthogonal basis functions are adopted as

$$\phi_0 = 1, \quad \phi_1 = \frac{x - x_i}{\Delta x/2}, \quad \phi_2 = \frac{y - y_j}{\Delta y/2}, \quad \phi_3 = \phi_1 \phi_2, \quad \phi_4 = \phi_1^2 - \frac{1}{3}, \quad \phi_5 = \phi_2^2 - \frac{1}{3}, \quad \dots,$$

over I_{ij} for two-dimensional problems. The numerical solution $\vec{U}^h(\vec{x}, t)$ of (2.9) over I_{ij} can then be written in the test function space $V_h^k(K)$ as

$$\vec{U}_{i,j}^h(\vec{x}, t) = \sum_{\ell=0}^N \vec{U}_{i,j}^{(\ell)}(t) \phi_\ell(\vec{x}), \quad \text{for } \vec{x} \in I_{i,j}. \quad (2.10)$$

Multiplying equation (2.10) by $\phi_k(\vec{x})$ on both sides and integrating over $I_{i,j}$, we have

$$\vec{U}_{i,j}^{(k)}(t) = \frac{1}{a_k} \int_{I_{i,j}} \vec{U}_{i,j}^h(\vec{x}, t) \phi_k(\vec{x})d\vec{x}$$

as the basis functions are orthogonal, where $a_k = \int_{I_{i,j}} (\phi_k(\vec{x}))^2 d\vec{x}$ are the normalization constants. We replace $\phi(\vec{x})$ by $\phi_k(\vec{x})$ in equation (2.9) to rewrite the semi-discrete scheme for $\vec{U}^{(k)}(t)$ as

$$\frac{\partial \vec{U}_{i,j}^{(k)}(t)}{\partial t} = L(\vec{U}_{i,j}^{(k)}(t)), \quad k = 0, 1, 2, \dots, \quad (2.11)$$

$$L(\vec{U}_{i,j}^{(k)}(t)) = \frac{1}{a_k} \left(\sum_{e \in \partial I_{i,j}} \sum_{l=1}^L \omega_l \vec{F}(\vec{U}^h(\vec{x}_{el}, t)) \cdot n_{e, I_{i,j}} \phi_k(\vec{x}_{el}) |e| + \sum_{m=1}^M \bar{\omega}_m \vec{F}(\vec{U}^h(\vec{x}_{I_{i,j}m}, t)) \cdot \text{grad} \phi_k(\vec{x}_{I_{i,j}m}) |I_{i,j}| \right). \quad (2.12)$$

We solve (2.11) by using the third-order TVD Runge-Kutta discretizations in [32].

If there are strong discontinuities in the approximate solution, oscillations can occur or even the method can break down. To enhance the stability of the method and eliminate possible spurious oscillations in the approximate solution, a local slope limiting is introduced in [11, 15] as follows. For a scalar equation, we denote

$$\begin{aligned} \vec{U}_{i+\frac{1}{2},j}^- &= \vec{U}_{i,j}^{(0)} + \tilde{U}_{i,j}^x, & \vec{U}_{i-\frac{1}{2},j}^+ &= \vec{U}_{i,j}^{(0)} - \tilde{U}_{i,j}^x, \\ \vec{U}_{i,j+\frac{1}{2}}^- &= \vec{U}_{i,j}^{(0)} + \tilde{U}_{i,j}^y, & \vec{U}_{i,j-\frac{1}{2}}^+ &= \vec{U}_{i,j}^{(0)} - \tilde{U}_{i,j}^y, \end{aligned}$$

where

$$\tilde{U}_{i,j}^x = \sum_{k=1}^N \vec{U}_{i,j}^{(k)}(t_n) \phi_k(\vec{x}_{i+\frac{1}{2},j}), \quad \tilde{U}_{i,j}^x = - \sum_{k=1}^N \vec{U}_{i,j}^{(k)}(t_n) \phi_k(\vec{x}_{i-\frac{1}{2},j}). \quad (2.13a)$$

$$\tilde{U}_{i,j}^y = \sum_{k=1}^N \vec{U}_{i,j}^{(k)}(t_n) \phi_k(\vec{x}_{i,j+\frac{1}{2}}), \quad \tilde{U}_{i,j}^y = - \sum_{k=1}^N \vec{U}_{i,j}^{(k)}(t_n) \phi_k(\vec{x}_{i,j-\frac{1}{2}}). \quad (2.13b)$$

We modify $\tilde{U}_{i,j}^x$, $\tilde{U}_{i,j}^x$, $\tilde{U}_{i,j}^y$ and $\tilde{U}_{i,j}^y$ in (2.13) by

$$\tilde{U}_{i,j}^{x(mod)} = \bar{m}(\tilde{U}_{i,j}^x, \vec{U}_{i+1,j}^{(0)} - \vec{U}_{i,j}^{(0)}, \vec{U}_{i,j}^{(0)} - \vec{U}_{i-1,j}^{(0)}), \quad (2.14a)$$

$$\tilde{U}_{i,j}^{x(mod)} = \bar{m}(\tilde{U}_{i,j}^x, \vec{U}_{i+1,j}^{(0)} - \vec{U}_{i,j}^{(0)}, \vec{U}_{i,j}^{(0)} - \vec{U}_{i-1,j}^{(0)}), \quad (2.14b)$$

$$\tilde{U}_{i,j}^{y(mod)} = \bar{m}(\tilde{U}_{i,j}^y, \vec{U}_{i,j+1}^{(0)} - \vec{U}_{i,j}^{(0)}, \vec{U}_{i,j}^{(0)} - \vec{U}_{i,j-1}^{(0)}), \quad (2.14c)$$

$$\tilde{U}_{i,j}^{y(mod)} = \bar{m}(\tilde{U}_{i,j}^y, \vec{U}_{i,j+1}^{(0)} - \vec{U}_{i,j}^{(0)}, \vec{U}_{i,j}^{(0)} - \vec{U}_{i,j-1}^{(0)}). \quad (2.14d)$$

where \bar{m} is the modified *minmod* function [31, 12] defined by

$$\bar{m}(a_1, \dots, a_m) = \begin{cases} a_1, & \text{if } |a_1| \leq M\Delta x^2, \\ m(a_1, \dots, a_m), & \text{otherwise,} \end{cases} \quad (2.15)$$

for (2.14a) and (2.14b), and with a change of Δx to Δy in (2.15) for (2.14c) and (2.14d).

The *minmod* function m is defined by

$$m(a_1, \dots, a_m) = \begin{cases} s \min_i |a_i|, & \text{if } s = \text{sign}(a_1) = \dots = \text{sign}(a_m), \\ 0, & \text{otherwise,} \end{cases}$$

and M in (2.15) is the TVB limiter constant which should be larger than the second derivative of the solution near smooth extrema and can be chosen suitably for different problems. We replace $\tilde{U}_{i,j}^x$, $\tilde{\tilde{U}}_{i,j}^x$, $\tilde{U}_{i,j}^y$, $\tilde{\tilde{U}}_{i,j}^y$ in (2.13) by $\tilde{U}_{i,j}^{x(mod)}$, $\tilde{\tilde{U}}_{i,j}^{x(mod)}$, $\tilde{U}_{i,j}^{y(mod)}$ and $\tilde{\tilde{U}}_{i,j}^{y(mod)}$ in (2.14). This allows a unique set of solution for the modified $\vec{U}_{i,j}^{(k)}(t_n)$ with $k \geq 1$ for the P^1 case. For the P^2 case, if any of the modified values in (2.14) is different from the original values in (2.13), we set $\vec{U}_{i,j}^{(3)}(t_n) = 0$ and then we can uniquely solve the remaining modified $\vec{U}_{i,j}^{(k)}(t_n)$ with $k \geq 1$.

For systems, the limiting is carried out in the local characteristic directions in order to improve the effectiveness of the limiter in controlling oscillations. We give the details of limiting the vector $\tilde{U}_{i,j}^x$ in (2.13) in the following:

- Transform the three vectors $\tilde{U}_{i,j}^x$, $\vec{U}_{i+1,j}^{(0)} - \vec{U}_{i,j}^{(0)}$ and $\vec{U}_{i,j}^{(0)} - \vec{U}_{i-1,j}^{(0)}$ to the characteristic field by left-multiplying with the matrix \vec{R}^{-1} where \vec{R} diagonalizes the Jacobian matrix as

$$\vec{R}^{-1} \frac{\partial \vec{F}_1(\vec{U})}{\partial \vec{U}} \vec{R} = \Lambda.$$

- Apply the scalar limiter to each component of the transformed vectors $\vec{R}^{-1} \tilde{U}_{i,j}^x$, $\vec{R}^{-1}(\vec{U}_{i+1,j}^{(0)} - \vec{U}_{i,j}^{(0)})$ and $\vec{R}^{-1}(\vec{U}_{i,j}^{(0)} - \vec{U}_{i-1,j}^{(0)})$.
- Transform the limited version of $\vec{R}^{-1} \tilde{U}_{i,j}^x$ back to the original space by left-multiplying with the matrix \vec{R} to obtain the limited version of $\tilde{U}_{i,j}^x$.

3 The interface treating method

As we can see, the neighbors' information is needed, in the form of the numerical flux, when we evolve the solution (2.11) over a cell with high-order accurate RKDG method to the next time step. For multi-medium flow these neighbors may be mixed and the information in the mixed cells, which is not physical, should not be used in the scheme. In this section, we will propose an interface treating method by constructing and solving a Riemann problem at the interface to impose the states in the mixed cell. Moreover, those approximate Riemann solvers are also used to define the states over the cells changing from mixed to single-medium when the interface moves from one cell to another. The procedure is an extension of the technique in [9].

3.1 One-dimensional problem

For the sake of clarity, we change the subscripts ij to i in equations (2.9)-(2.14) which are for two-dimensional problems, and denote the approximate solution at time t_n as \vec{U}_i^h . If one cell and its neighbors are both single-medium, then the RKDG scheme can be directly used here. Our attention will therefore be focused on the mixed cells whose neighbors are occupied by single medium for one dimensional problems. Assume $I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ is a mixed cell in which the interface lies, and the cells I_{i-1} and I_{i+1} are single-medium cells as shown in Figure 3.1. We construct a Riemann problem by taking the input data as $\vec{U}_L = \vec{U}_{i-\frac{1}{2}}^h$, $\vec{U}_R = \vec{U}_{i+\frac{1}{2}}^h$ with $\vec{U}_{i-\frac{1}{2}}^h = \vec{U}_{i-1}^h(x_{i-\frac{1}{2}}, t_n)$ and $\vec{U}_{i+\frac{1}{2}}^h = \vec{U}_{i+1}^h(x_{i+\frac{1}{2}}, t_n)$, defined by (2.10). The two-shock Riemann solver [34] provides the intermediate states $\vec{U}_L^* = (\rho_L^*, u^*, p^*)$ and $\vec{U}_R^* = (\rho_R^*, u^*, p^*)$ at the interface. These states are then used to replace $(\vec{U}^h)_i^+(x_{i-\frac{1}{2}}, t_n)$ and $(\vec{U}^h)_i^-(x_{i+\frac{1}{2}}, t_n)$ in (2.7) to compute the numerical flux $\mathcal{F}(\vec{U}(\vec{x}_{i-\frac{1}{2}}, t_n))$ and $\mathcal{F}(\vec{U}(\vec{x}_{i+\frac{1}{2}}, t_n))$. We can solve equation (2.11) to obtain the degrees of freedom at next sub-step of the third order RK method.

Once the degrees of freedom are available, the local slope limiter (2.14) is used to eliminate the spurious oscillation that may occur at discontinuity. One should note that we correct

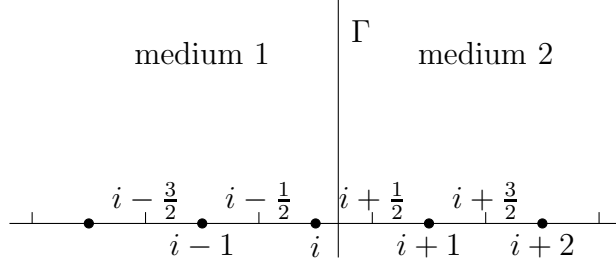


Figure 3.1: One-dimensional Riemann problem

\tilde{U}_{i-1}^x , $\tilde{\tilde{U}}_{i-1}^x$, \tilde{U}_{i+1}^x and $\tilde{\tilde{U}}_{i+1}^x$ with the obtained approximate Riemann solver as

$$\begin{aligned} \overline{m}(\tilde{U}_{i-1}^x, \vec{U}_L^* - \vec{U}_{i-1}^{(0)}, \vec{U}_{i-1}^{(0)} - \vec{U}_{i-2}^{(0)}), & \quad \overline{m}(\tilde{\tilde{U}}_{i-1}^x, \vec{U}_L^* - \vec{U}_{i-1}^{(0)}, \vec{U}_{i-1}^{(0)} - \vec{U}_{i-2}^{(0)}), \\ \overline{m}(\tilde{U}_{i+1}^x, \vec{U}_{i+2}^{(0)} - \vec{U}_{i+1}^{(0)}, \vec{U}_{i+1}^{(0)} - \vec{U}_R^*), & \quad \overline{m}(\tilde{\tilde{U}}_{i+1}^x, \vec{U}_{i+2}^{(0)} - \vec{U}_{i+1}^{(0)}, \vec{U}_{i+1}^{(0)} - \vec{U}_R^*), \end{aligned} \quad (3.16)$$

if I_i is a mixed cell.

By doing so at each sub-step of the third-order RK method we refrain from using the information in mixed cells when evolving the approximate solution from time t_n to t_{n+1} .

The obtained approximate Riemann solver is also used to track the interface by solving the following equation

$$\frac{dx_I}{dt} = u^*.$$

From time t_n to t_{n+1} , two cases are possible: the interface x_I remains in cell I_i or the interface moves from I_i to one of its neighbors. If the interface is still in I_i , then the evolution is complete and we proceed to the next time step. Else, if the interface moves left to the cell I_{i-1} , i.e. $x_I \in I_{i-1}$, the approximate Riemann solver is applied to correct the states over the cells I_i

$$(\vec{U}_i^{(0)})^{\text{new}}(t_{n+1}) = \vec{U}_R^*, \quad (\vec{U}_i^{(k)})^{\text{new}}(t_{n+1}) = 0, \quad k = 1, 2, \dots,$$

and the mixture in cell I_i flows into the cell I_{i-1} , by correcting $\vec{U}_{i-1}^{(k)}(t_{n+1})$ with

$$(\vec{U}_{i-1}^{(0)})^{\text{new}}(t_{n+1}) = \vec{U}_i^{(0)}(t_{n+1}) + \vec{U}_{i-1}^{(0)}(t_{n+1}) - \vec{U}_R^*, \quad (\vec{U}_{i-1}^{(k)})^{\text{new}}(t_{n+1}) = 0, \quad k = 1, 2, \dots$$

The global conservation is kept, i.e.,

$$(\vec{U}_i^{(0)})^{\text{new}}(t_{n+1}) + (\vec{U}_{i-1}^{(0)})^{\text{new}}(t_{n+1}) = \vec{U}_i^{(0)}(t_{n+1}) + \vec{U}_{i-1}^{(0)}(t_{n+1}).$$

Similarly, if the interface moves right to the cell I_{i+1} , i.e. $x_I \in I_{i+1}$, we will correct the states over the cells I_i and I_{i+1} as

$$\begin{aligned} (\vec{U}_i^{(0)})^{\text{new}}(t_{n+1}) &= \vec{U}_L^*, & (\vec{U}_i^{(k)})^{\text{new}}(t_{n+1}) &= 0, & k &= 1, 2, \dots, \\ (\vec{U}_{i+1}^{(0)})^{\text{new}}(t_{n+1}) &= \vec{U}_{i+1}^{(0)}(t_{n+1}) + \vec{U}_i^{(0)}(t_{n+1}) - \vec{U}_L^*, & (\vec{U}_{i+1}^{(k)})^{\text{new}}(t_{n+1}) &= 0, & k &= 1, 2, \dots \end{aligned}$$

We also have conservation

$$(\vec{U}_{i+1}^{(0)})^{\text{new}}(t_{n+1}) + (\vec{U}_i^{(0)})^{\text{new}}(t_{n+1}) = \vec{U}_{i+1}^{(0)}(t_{n+1}) + \vec{U}_i^{(0)}(t_{n+1}).$$

From the above description we can see that the mixed cells make no contribution to the evolution of the scheme, and the scheme is conservative if values in the mixed cells are accounted for. However, overshoots or undershoots may occur in the mixed cells, which indicate an effective conservation error of the scheme when values in those mixed cells are ignored. In the next section we plot over- and under-shoots for one and two dimensional problems to assess effective conservative errors of the scheme. Notice that in [9] the method is referred to as conservative and values in the mixed cells are not plotted in the figures for the numerical solution.

3.2 Two-dimensional problem

As given in [9], the one-dimensional algorithm will be extended to two-dimension in a dimension-by-dimension manner.

We track the interface by solving the level set equation (2.4). A cell is a single-medium cell if $\text{sign}(\phi)$ at its four vertexes is the same, and this cell is marked by $\text{sign}(\phi)$. Otherwise, the cell is mixed and marked by 0. Furthermore, a mixed cell is marked by 0^+ when $\phi^c > 0$, and 0^- when $\phi^c < 0$, where ϕ^c over I_{ij} is defined by

$$\phi_{ij}^c = \frac{\phi_{i-\frac{1}{2},j-\frac{1}{2}} + \phi_{i+\frac{1}{2},j-\frac{1}{2}} + \phi_{i+\frac{1}{2},j+\frac{1}{2}} + \phi_{i-\frac{1}{2},j+\frac{1}{2}}}{4}.$$

We describe the treatment of mixed cells $I_{i,j}$ below. The Riemann problem will be constructed and solved in x and y directions respectively for the computation of the corresponding numerical fluxes in both directions. In the following, details of the algorithm in

the x direction will be described and that in the y direction can be implemented in a similar manner.

The algorithm is simple for one-dimensional problems, but the extension to two-dimensional problems is not straightforward. In one-dimensional problems we redefine the states in mixed cells I_i by the approximate Riemann solver between cells I_{i-1} and I_{i+1} , which are occupied by single-medium. However, the two-dimensional situation is more complicated as one or both of $I_{i,j}$'s neighbors may also be mixed if the interface is close to horizontal. In order to refrain from using the information of mixed cells, we redefine the states of the mixed neighbor(s) by the weighted average of its (their) neighboring cells. We consider the first situation in Figure 3.2(a) where the mixed cell $I_{i,j}$'s neighbor (namely $I_{i-1,j}$) is also mixed. The construction of the Riemann problem in the x direction between $I_{i-1,j}$ and $I_{i+1,j}$ is given in the following. $I_{i-1,j}$ is marked by $-\text{sign}(\phi_{i+1,j}^c)$, to ensure the Riemann problem is constructed between two different fluids, and the states $\vec{U}_{i-1,j}^h$ are replaced by the weighted average of the states of all its neighbors marked by $-\text{sign}(\phi_{i+1,j}^c)$, that is

$$\vec{U}_{i-1,j}^{(k)} = \frac{D_x \vec{U}_{i-2,j}^{(k)} + D_{xy} \vec{U}_{i-2,j-1}^{(k)} + D_y \vec{U}_{i-1,j-1}^{(k)}}{D_x + D_y + D_{xy}}, \quad k = 0, 1, 2, \dots, \quad (3.17)$$

where the weights, D_x , D_y and D_{xy} are inversely proportional to the distance between the cell $I_{i,j}$ and all its neighbors marked by $-\text{sign}(\phi_{i+1,j}^c)$, written as

$$D_x = \frac{1}{\Delta x}, \quad D_y = \frac{1}{\Delta y}, \quad D_{xy} = \frac{1}{\sqrt{(\Delta x)^2 + (\Delta y)^2}}. \quad (3.18)$$

Note that the replacement of these values is only for the construction of the Riemann problem. We take the input states of the Riemann problem on the left side of the interface as $\vec{U}_L = \vec{U}_{i-1,j}^h(\vec{x}_{i-\frac{1}{2},j}, t_n)$, where $\vec{U}_{i-1,j}^h(\vec{x}_{i-\frac{1}{2},j}, t_n)$ is defined in (2.10). On the right side $I_{i+1,j}$ is a single-medium cell, we can take $\vec{U}_R = \vec{U}_{i+1,j}^h(\vec{x}_{i+\frac{1}{2},j}, t_n)$. In the second situation, $I_{i-1,j}$ and $I_{i+1,j}$ are marked by $\text{sign}(\phi_{i-1,j}^c)$ and $-\text{sign}(\phi_{i-1,j}^c)$ if $|\phi_{i-1,j}^c| > |\phi_{i+1,j}^c|$, otherwise they are marked by $-\text{sign}(\phi_{i+1,j}^c)$ and $\text{sign}(\phi_{i+1,j}^c)$. In Figure 3.2(b), $I_{i-1,j}$ and $I_{i+1,j}$ are marked by $\text{sign}(\phi_{i-1,j}^c)$ and $-\text{sign}(\phi_{i-1,j}^c)$, $\vec{U}_{i-1,j}^h$ and $\vec{U}_{i+1,j}^h$ are replaced by

$$\vec{U}_{i-1,j}^{(k)} = \frac{D_x \vec{U}_{i-2,j}^{(k)} + D_{xy} \vec{U}_{i-2,j-1}^{(k)} + D_y \vec{U}_{i-1,j-1}^{(k)} + D_{xy} \vec{U}_{i,j-1}^{(k)}}{D_x + D_y + 2D_{xy}}, \quad k = 0, 1, 2, \dots,$$

$$\vec{U}_L = \vec{U}_{i-1,j}^h(\vec{x}_{i-\frac{1}{2},j}, t_n), \quad (3.19)$$

and

$$\begin{aligned} \vec{U}_{i+1,j}^{(k)} &= \frac{D_x \vec{U}_{i+2,j}^{(k)} + D_{xy} \vec{U}_{i+2,j+1}^{(k)} + D_y \vec{U}_{i+1,j+1}^{(k)} + D_{xy} \vec{U}_{i,j+1}^{(k)}}{D_x + D_y + 2D_{xy}}, \quad k = 0, 1, 2, \dots, \\ \vec{U}_R &= \vec{U}_{i+1,j}^h(\vec{x}_{i+\frac{1}{2},j}, t_n), \end{aligned} \quad (3.20)$$

where D_x , D_y and D_{xy} are defined in (3.18).

Another situation is shown in Figure 3.2(c), $I_{i-1,j}$ is marked by -1, but we can not find a cell marked by -1 from $I_{i-1,j}$'s neighbors. This could happen when the interface is highly curved. In this case, we simply look for a nearest cell marked by -1 from $I_{i-1,j}$'s surrounding cells.

Once \vec{U}_L and \vec{U}_R , the input states of the Riemann problem, are provided, the two-shock Riemann solver in [34] can be applied to give the intermediate states $(\vec{U}_L^*)^x$ and $(\vec{U}_R^*)^x$ on both sides of the interface. These intermediate states are the solutions of the Riemann problem constructed in the x direction. With the same procedure, we can obtain the intermediate states of the Riemann solvers in the y direction denoted as $(\vec{U}_L^*)^y$ and $(\vec{U}_R^*)^y$. As we have done for the one-dimensional problem, these intermediate states are used to replace the states in mixed cell $I_{i,j}$ to compute the numerical flux in (2.7) and the slope limiter in (2.14).

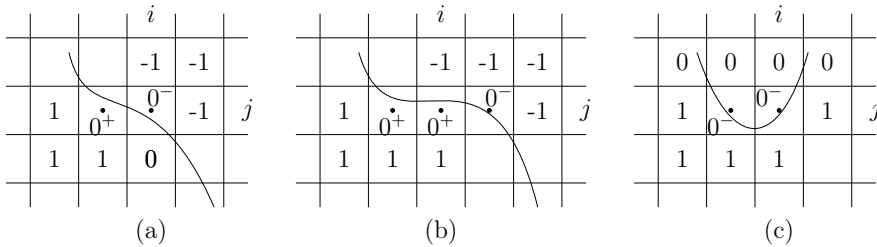


Figure 3.2: Cell configurations in the construction of the Riemann problem in the x direction for a two-dimensional problem.

With the evolution of the level set function from time t_n to t_{n+1} , a cell may change from mixed to single-medium. Due to the CFL number, the interface does not shift more than one cell in one time step, so only three cases are considered to illustrate the movement of the

interface in Figure 3.3. In Figure 3.3(a) and (b), we claim that the interface has moved from cell $I_{i,j}$ to the right, in this situation only one cell (e.g. $I_{i+1,j}$) of $I_{i,j}$'s four neighbors ($I_{i+1,j}$, $I_{i,j+1}$, $I_{i-1,j}$ and $I_{i,j-1}$) changes mark from 1 to 0. The states in the cells $I_{i,j}$ and $I_{i+1,j}$ are updated with a conservative algorithm as

$$(\vec{U}_{i,j}^{(0)})^{\text{new}}(t_{n+1}) = (\vec{U}_L^*)^x, \quad (\vec{U}_{i,j}^{(k)})^{\text{new}}(t_{n+1}) = 0, \quad k = 1, 2, \dots,$$

$$(\vec{U}_{i+1,j}^{(0)})^{\text{new}}(t_{n+1}) = \vec{U}_{i+1,j}^{(0)}(t_{n+1}) + \vec{U}_{i,j}^{(0)}(t_{n+1}) - (\vec{U}_L^*)^x, \quad (\vec{U}_{i+1,j}^{(k)})^{\text{new}}(t_{n+1}) = 0, \quad k = 1, 2, \dots$$

The local conservation property is preserved. In Figure 3.3(c), we claim that the interface has moved from cell $I_{i,j}$ in the lower-right direction, in this situation two cells (e.g. $I_{i+1,j}$ and $I_{i,j-1}$) change mark from 1 to 0. The states in the cell $I_{i,j}$ and these two cells with changed mark are updated with a conservative algorithm in a different way as

$$(\vec{U}_{i,j}^{(0)})^{\text{new}}(t_{n+1}) = \frac{(\vec{U}_L^*)^x + (\vec{U}_R^*)^y}{2}, \quad (\vec{U}_{i,j}^{(k)})^{\text{new}}(t_{n+1}) = 0, \quad k = 1, 2, \dots,$$

$$\begin{aligned} (\vec{U}_{i+1,j}^{(0)})^{\text{new}}(t_{n+1}) &= \vec{U}_{i+1,j}^{(0)}(t_{n+1}) + \frac{\vec{U}_{i,j}^{(0)}(t_{n+1})}{2} - \frac{(\vec{U}_{i,j}^{(0)})^{\text{new}}(t_{n+1})}{2}, \\ (\vec{U}_{i+1,j}^{(k)})^{\text{new}}(t_{n+1}) &= 0, \quad k = 1, 2, \dots \end{aligned}$$

$$\begin{aligned} (\vec{U}_{i,j-1}^{(0)})^{\text{new}}(t_{n+1}) &= \vec{U}_{i,j-1}^{(0)}(t_{n+1}) + \frac{\vec{U}_{i,j}^{(0)}(t_{n+1})}{2} - \frac{(\vec{U}_{i,j}^{(0)})^{\text{new}}(t_{n+1})}{2}, \\ (\vec{U}_{i,j-1}^{(k)})^{\text{new}}(t_{n+1}) &= 0, \quad k = 1, 2, \dots \end{aligned}$$

where $(\vec{U}_L^*)^x$ is the left state of the Riemann problem solver in the x direction, and $(\vec{U}_R^*)^y$ is the right state of the Riemann problem solver in the y direction. By doing so, we also have conservation

$$(\vec{U}_{i,j}^{(0)})^{\text{new}}(t_{n+1}) + (\vec{U}_{i+1,j}^{(0)})^{\text{new}}(t_{n+1}) + (\vec{U}_{i,j-1}^{(0)})^{\text{new}}(t_{n+1}) = \vec{U}_{i+1,j}^{(0)}(t_{n+1}) + \vec{U}_{i,j}^{(0)}(t_{n+1}) + \vec{U}_{i,j-1}^{(0)}(t_{n+1}).$$

4 Numerical examples

In this section we present several numerical examples with the interface treating method and P^1 and P^2 (second-order and third-order accurate) RKDG methods for one- and two-dimensional problems. For one-dimensional problems, we also plot the density and velocity in

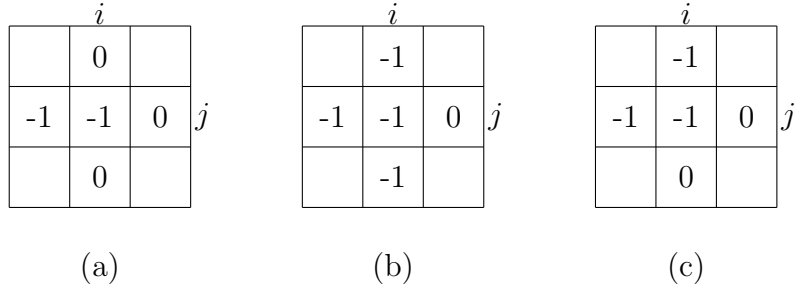


Figure 3.3: Cell configurations with cell $I_{i,j}$ changing from mixed to single-medium.

the mixed cell. These values do not participate in the updating of the solution in neighboring cells and are usually not plotted, e.g. in [9]. However they can help us to evaluate the conservation errors, especially when overshoot or undershoot exists, since the scheme is conservative when these values are accounted for. We take $CFL = 0.3$ for P^1 and $CFL = 0.15$ for P^2 .

4.1 One-dimensional examples

Example 4.1 The first example is to demonstrate the advantage of higher order methods. This example was used in [33, 11] in the context of a single-medium fluid. It contains both shocks and fine structures in smooth regions, serving as a simple model for shock-turbulence interactions. The initial discontinuity is located at $x_I = -4.0$. The states on the left and right sides of the initial discontinuity are defined respectively as

$$(\rho, u, p, \gamma, B) = \begin{cases} (3.857143, 2.629369, 10.333333, 1.4, 0), & x < x_I, \\ (1 + 0.2 \sin(5x), 0, 1, 1.666666, 0), & x > x_I. \end{cases}$$

We compute the solution of this problem to $t = 1.8$ with 101 grid points and take the TVB limiter constant $M = 150$. The computational domain is $[-5,5]$. Figure 4.4 plots the density obtained by the P^1 method (left) and by the P^2 method (right). The values in the mixed cell are also plotted. The solid line is the solution obtained with 2000 cells using the P^2 method and can be considered as a converged solution. It is found that both methods can capture the contact discontinuity sharply and accurately, and the results by the P^2 method are in better agreement with the converged solution than that by the P^1 method, especially

for the regions with fine structures. We can see that there are smaller undershoots with the P^2 method than that with the P^1 method, indicating that the higher-order accurate P^2 method can also reduce the conservation error for this example.

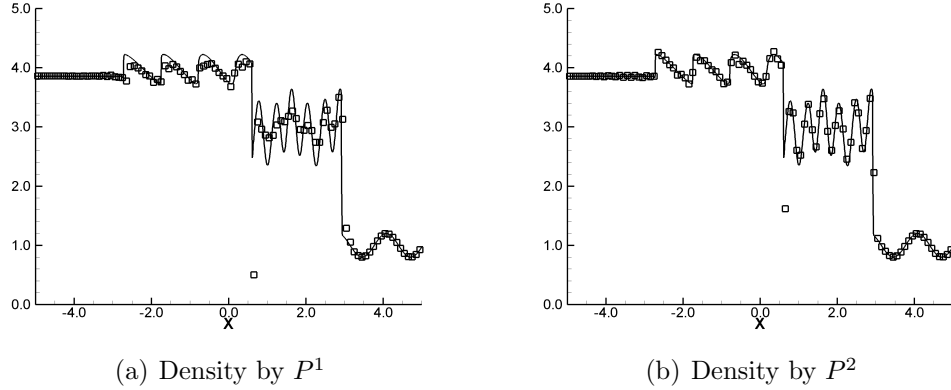


Figure 4.4: Example 4.1: The numerical results (“□”) calculated by the RKDG method. The solid line denotes the converged reference solution.

Example 4.2 This example is the same as the standard Lax shock tube problem except that we change γ from 1.4 to 1.6 in the right domain of the initial condition. The interface is located at $x_I = 5.0$. The states on the left and right sides of the interface are defined respectively as

$$(\rho, u, p, \gamma, B) = \begin{cases} (0.445, 0.699, 3.527, 1.4, 0), & x < x_I, \\ (0.5, 0, 0.8565, 1.6, 0), & x > x_I. \end{cases}$$

We compute the solution of this problem to $t = 1.3$ with 201 grid points and take the TVB limiter constant $M = 250$. The computational domain is $[0, 10]$. Figure 4.5 plots the density, velocity, and pressure obtained by the P^1 method (left) and by the P^2 method (right) without plotting values in the mixed cell. It is found that both methods can capture the interface sharply and accurately, and the results by the P^2 method are in better agreement with the theoretical solution than those by the P^1 method, which has the density profile slightly smeared at the interface. Figure 4.6 plots density and velocity obtained by the P^2 method showing also the values in the mixed cell. We can see that these values manifesting themselves as undershoots, which give a visual “measurement” of the conservation error when the mixed cell values are not considered.

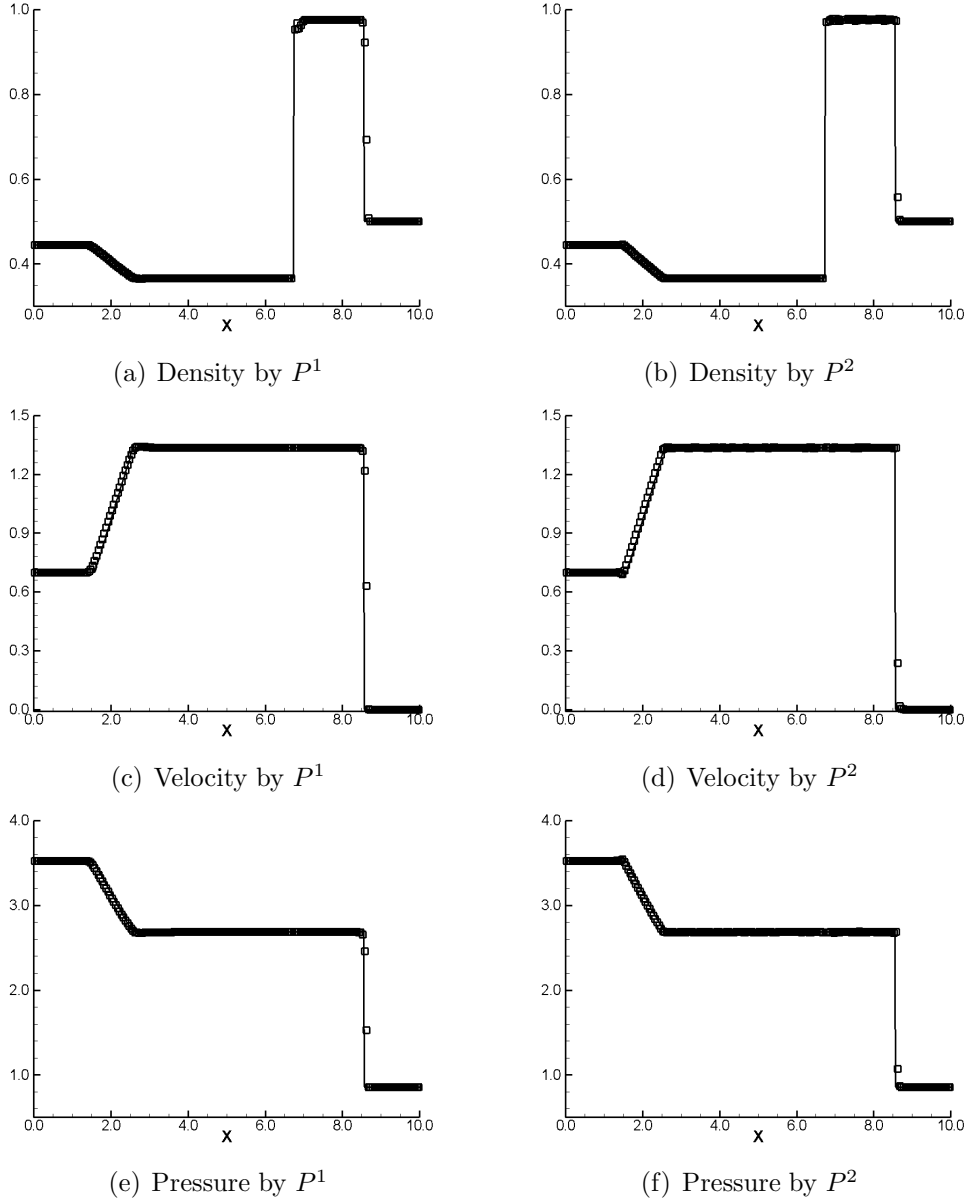


Figure 4.5: Example 4.2: The numerical results (“□”) calculated by the RKDG method. The solid line denotes the exact solution.

Example 4.3 This example is the same as the regular Sod shock tube problem except that we change γ from 1.4 to 1.6 in the right domain of the initial condition. The interface is located at $x_I = 5.0$. The states on the left and right sides of the interface are defined respectively as

$$(\rho, u, p, \gamma, B) = \begin{cases} (1, 0, 1.0, 1.4, 0), & x < x_I, \\ (1, 0, 0.1, 1.6, 0), & x > x_I. \end{cases}$$

We compute the solution of this problem to $t = 2.0$ with 201 grid points and take the TVB

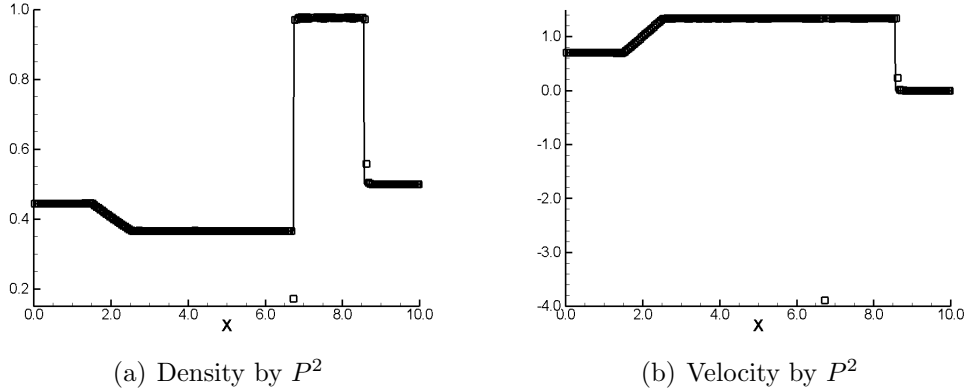


Figure 4.6: Example 4.2: The numerical results (“□”) with the mixed cell calculated by the RKDG method. The solid line denotes the exact solution.

limiter constant $M = 160$. The computational domain is $[0,10]$. Figure 4.7 plots the density, velocity, and pressure obtained by the P^1 method (left) and by the P^2 method (right). As before, slight improvement can be observed in the velocity by the P^2 method. The results obtained with both methods compare reasonably well with the theoretical results, the position of the interface and the shock front are predicted accurately. Figure 4.8 shows the density and velocity obtained by the P^2 method with the mixed cell. This time, there are much smaller overshoots or undershoots, indicating that effective conservation error is very small for this example.

Example 4.4 This is also a gaseous shock-tube problem with stiffer initial condition. The interface is located at $x_I = 50.0$ with the computational domain $[0,100]$. The states on the left and right sides of the interface are defined respectively as

$$(\rho, u, p, \gamma, B) = \begin{cases} (1, 0, 500, 1.4, 0), & x < x_I, \\ (1, 0, 0.2, 1.6, 0), & x > x_I. \end{cases}$$

We compute the solution of this problem to $t = 1.5$ with 401 grid points and take the TVB limiter constant $M = 40$. Figure 4.9 shows the density, velocity, and pressure obtained by the P^1 method (left) and by the P^2 method (right). The results obtained with both methods compare reasonably well with the theoretical results. It is clear that the P^2 method works better with less smearing at the interface and has better resolution at the rarefaction and shock front. Figure 4.10 shows the density and velocity obtained by the P^2 method with the

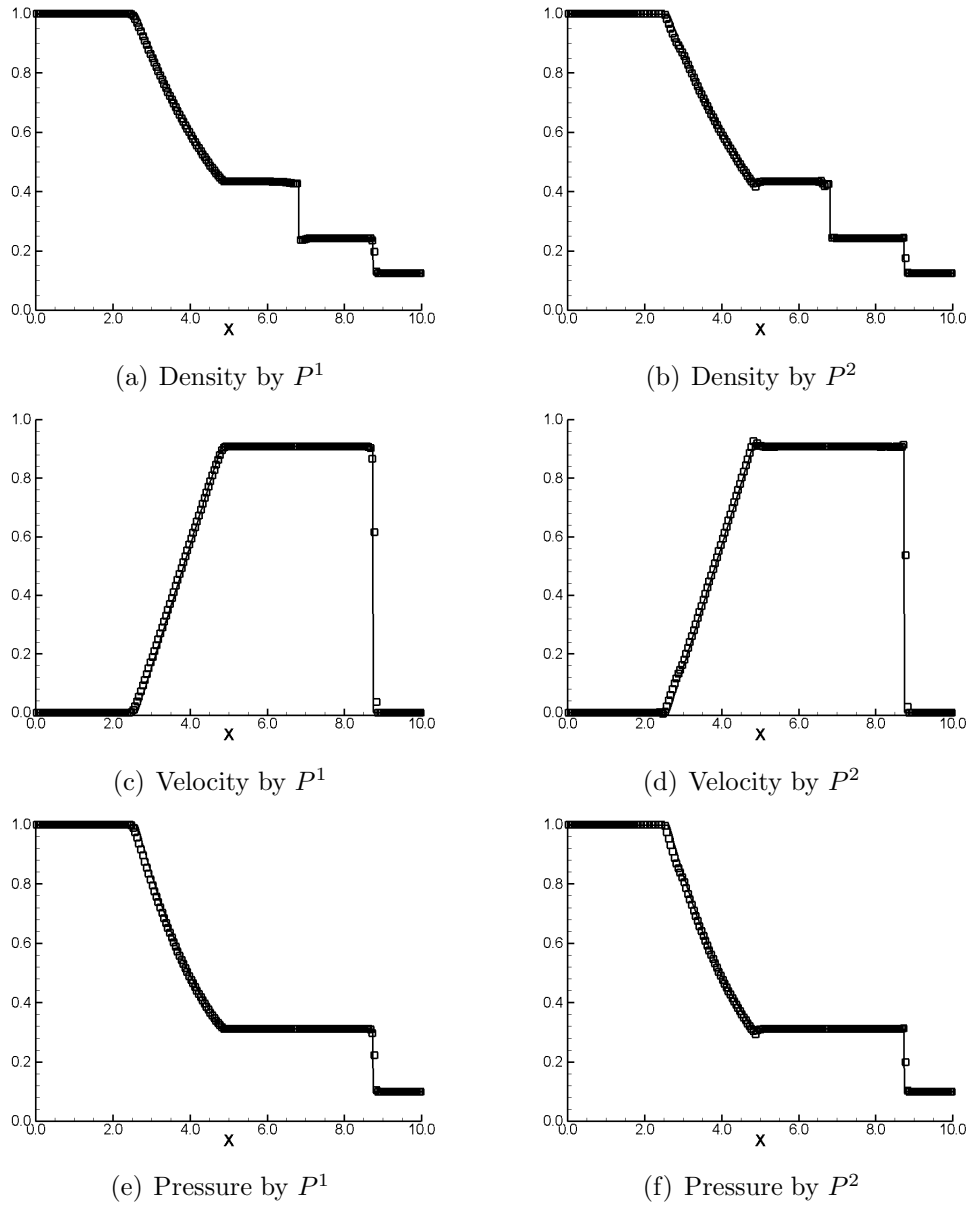


Figure 4.7: Example 4.3: The numerical results (“□”) calculated by the RKDG method. The solid line denotes the exact solution.

mixed cell. There is no overshoot or undershoot, indicating that effective conservation error is very small for this example.

Example 4.5 The last one-dimensional example is a gas-water shock-tube problem. The interface is located at $x_I = 0.5$ with the computational domain $[0,1]$. The states on the left

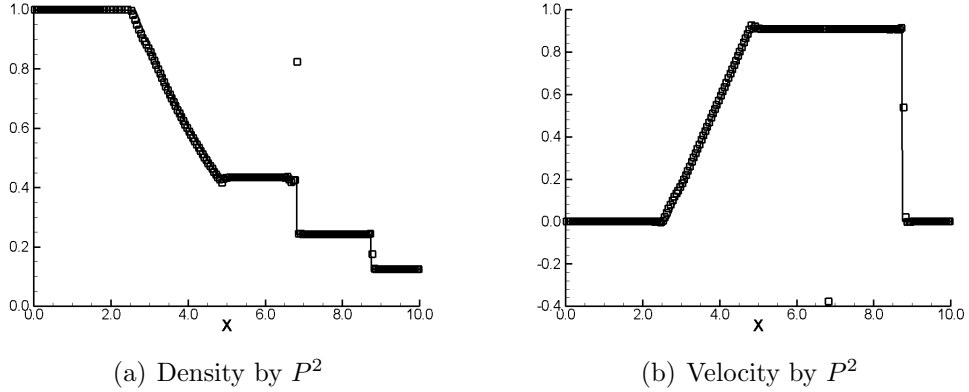


Figure 4.8: Example 4.3: The numerical results (“□”) with the mixed cell calculated by the RKDG method. The solid line denotes the exact solution.

and right sides of the interface are defined respectively as

$$(\rho, u, p, \gamma, B) = \begin{cases} (1, 0, 1.0, 1.4, 0), & x < x_I, \\ (1, 0, 0.1, 1.6, 0), & x > x_I. \end{cases}$$

We compute the solution of this problem to $t = 0.0015$ with 201 grid points and take the TVB limiter constant $M = 220$. Figure 4.11 shows the density, velocity, and pressure obtained by the P^1 method (left) and by the P^2 method (right). The results obtained with both methods compare reasonably well with the theoretical results. It is found there is no evident improvement with the P^2 method for this example, presumably because water is very sensitive to the TVB limiter constant and one has to be conservative in choosing the constant in order to avoid the appearance of negative pressure. Figure 4.12 shows the density and velocity obtained by the P^2 method with the mixed cell. This time there is only a very small overshoot, indicating very small effective conservation error for this example.

4.2 Two-dimensional problems

In this section, we calculate the interaction between a shock and a bubble by the RKDG method for multi-medium flow described in sections 2.3 and 3. Figures 4.14 and 4.17 show the Schlieren image of the numerical results displaying the magnitude of the density gradient $|\nabla\rho|$ for different times. In these figures we have used the following shading function

$$\exp\left(-\frac{C|\nabla\rho|}{\max(|\nabla\rho|)}\right), \quad C = 200$$

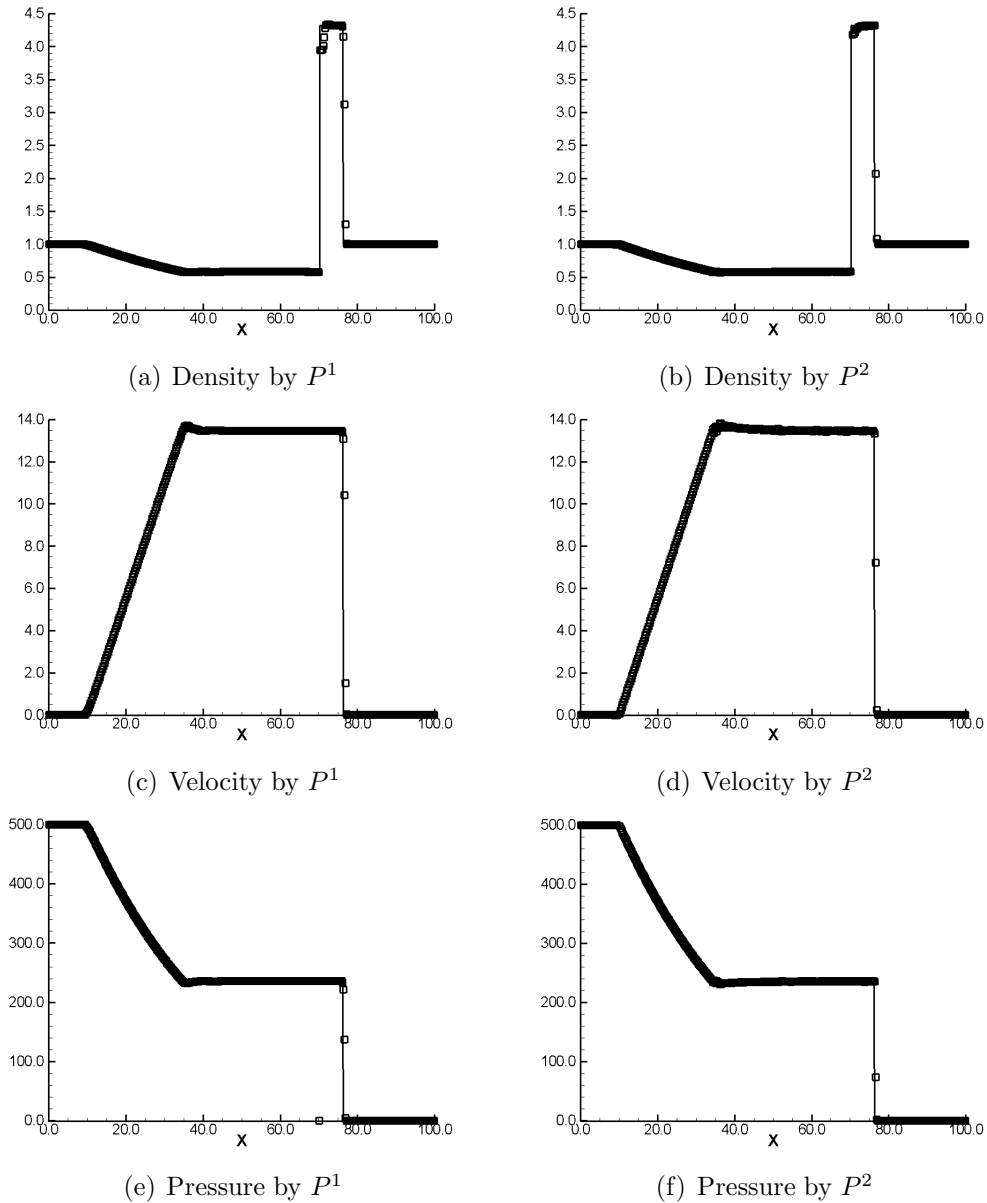


Figure 4.9: Example 4.4: The numerical results (“□”) calculated by the RKDG method. The solid line denotes the exact solution.

where the numerical density derivatives are computed using central difference.

Example 4.6 This example is about a weak shock impacting on a cylindrical helium bubble. Numerical and experimental results can be found in [9, 30]. The physical domain is taken as $[0, 20] \times [0, 5]$, and divided into 600×150 mesh cells. The lower boundary is the symmetry plane, and non-reflecting conditions are specified on the upper and left boundaries, while the right boundary conditions are taken as the post-shock values (the non-reflecting) before

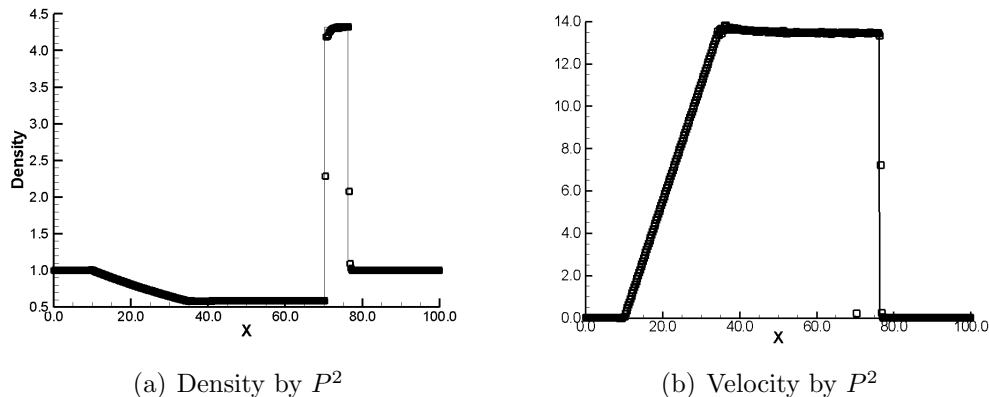


Figure 4.10: Example 4.4: The numerical results (“□”) with the mixed cell calculated by the RKDG method. The solid line denotes the exact solution.

(after) the rarefaction wave reaches the right boundary.

Initially, a cylindrical helium bubble with a diameter of 5 is centered at (10,0), see the schematic diagram shown in Figure 4.13. Inside the cylindrical helium bubble, the physical variables $(\rho, u, v, p, \gamma, B)$ are taken as $(\frac{4}{29}, 0, 0, 1, \frac{5}{3}, 0)$. A weak shock wave lies at 17.5 with pressure strength of 1.5 and the pre- and post-shock states of the incident water shock wave are

$$(\rho, u, v, p, \gamma, B) = \begin{cases} (\frac{4}{3}, -0.3535, 0, 1.5, 1.4, 0), & x > 17.5, \\ (1, 0, 0, 1, 1.4, 0), & x < 17.5, \end{cases}$$

where the post-shock values are obtained by using the Rankine-Hugoniot condition.

We use the P^2 RKDG method to compute this problem with the TVB limiter constant $M = 30$ in (2.15). Figure 4.14 shows a sequence of schlieren images from the simulation of the shock and helium bubble interaction. Figure 4.14(a) shows a view of the helium bubble at $t = 4.5$, when there is a curved refracted shock which lies inside the bubble, and the refracted shock wave moves left and is well ahead of the incident shock due to the higher sound speed in the helium bubble than in the surrounding air. Outside the bubble, a curved reflected wave is a weak rarefaction wave which moves to the right. At about $t = 5.0$, the refracted wave emerges from the left-hand side of the bubble to become the transmitted shock wave

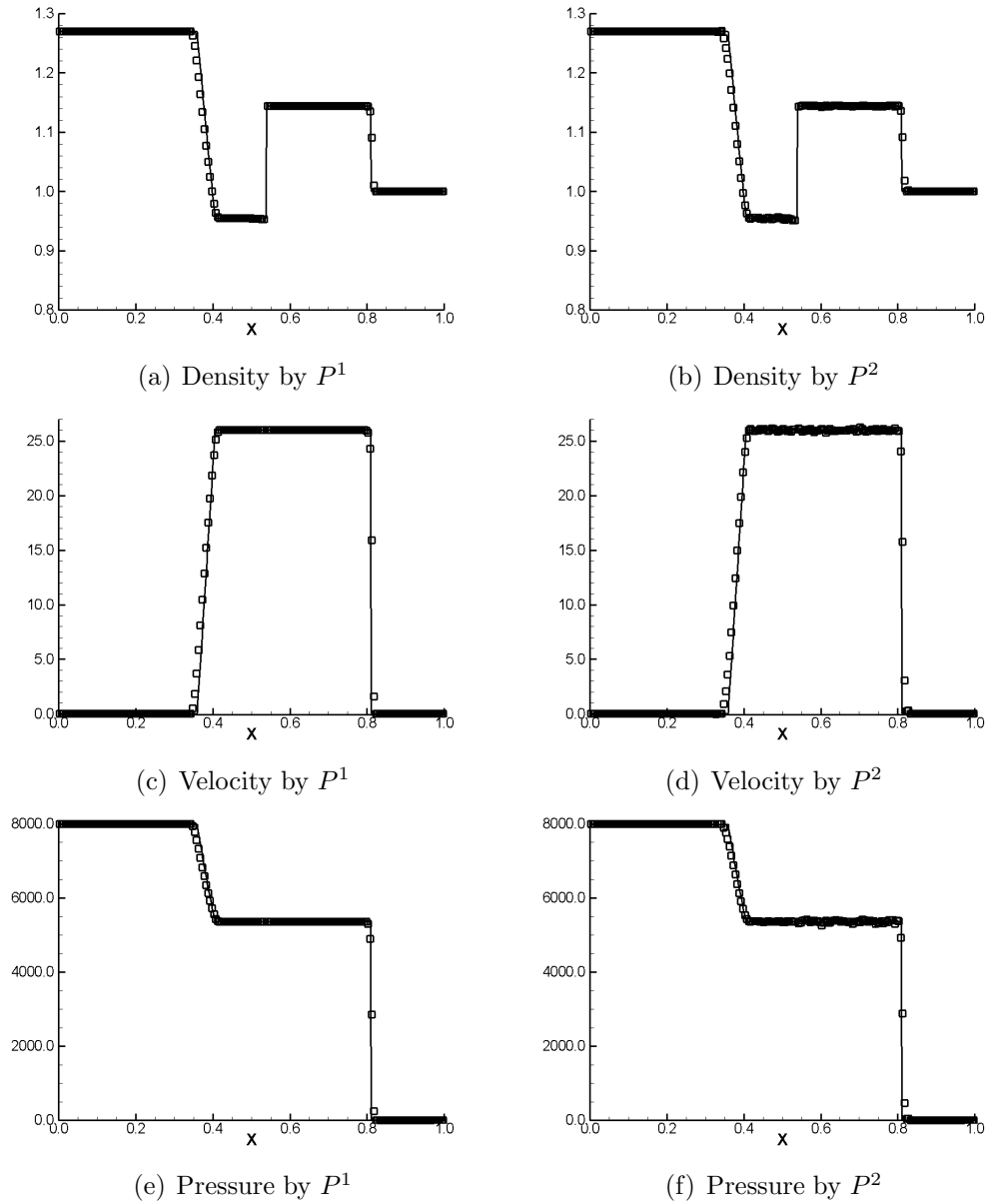


Figure 4.11: Example 4.5: The numerical results (“□”) calculated by the RKDG method. The solid line denotes the exact solution.

and the reflected wave shows a complicated structure. At $t = 7.0$, both the original reflected wave and the transmitted shock have been reflected from the top and bottom boundary, and as can be seen from Figure 4.14(c) these reflections are very strong and can lead to large increases in the local pressure. As time moves on, in Figure 4.14(d) a jet is formed and runs to the left along the symmetry plane. The bubble evolves slowly into a kidney shape as a result of the generation of vorticity near the bubble interface. The interface starts to roll up

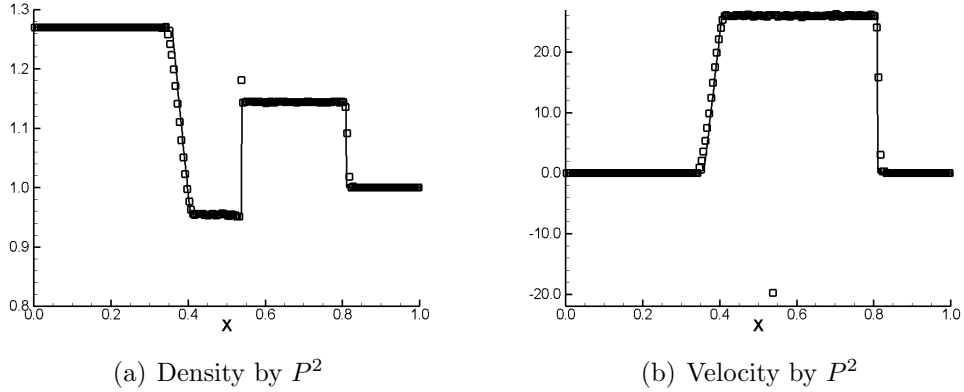


Figure 4.12: Example 4.5: The numerical results (“□”) with the mixed cell calculated by the RKDG method. The solid line denotes the exact solution.

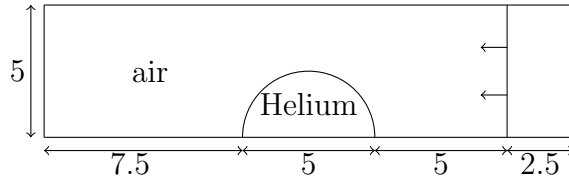


Figure 4.13: The geometry of Example 4.6.

in Figure 4.14(e) due to the shear instability and further evolution is shown in Figure 4.14(f). To show the states in the mixed cells to gauge the conservation error, Figures 4.15(a)-(d) plot the density distribution in a zoomed-in domain. We can find the high-frequency overshoots / undershoots along the interface which are more complicated than those in one-dimensional problems. The magnitude of these overshoots / undershoots represents the level of effective conservation error.

Our simulation reproduces all the known features of the interaction process. We can also capture the weak wave and shear instability on the relative coarse mesh. These results demonstrate that the combination of the P^2 RKDG method and the interface treating method works well.

Example 4.7 In this example a strong water shock impacts on a cylindrical air bubble in water. This problem has been numerically and experimentally investigated by several authors, see e.g. [5, 6]. Here, we consider a similar setup of the initial and boundary

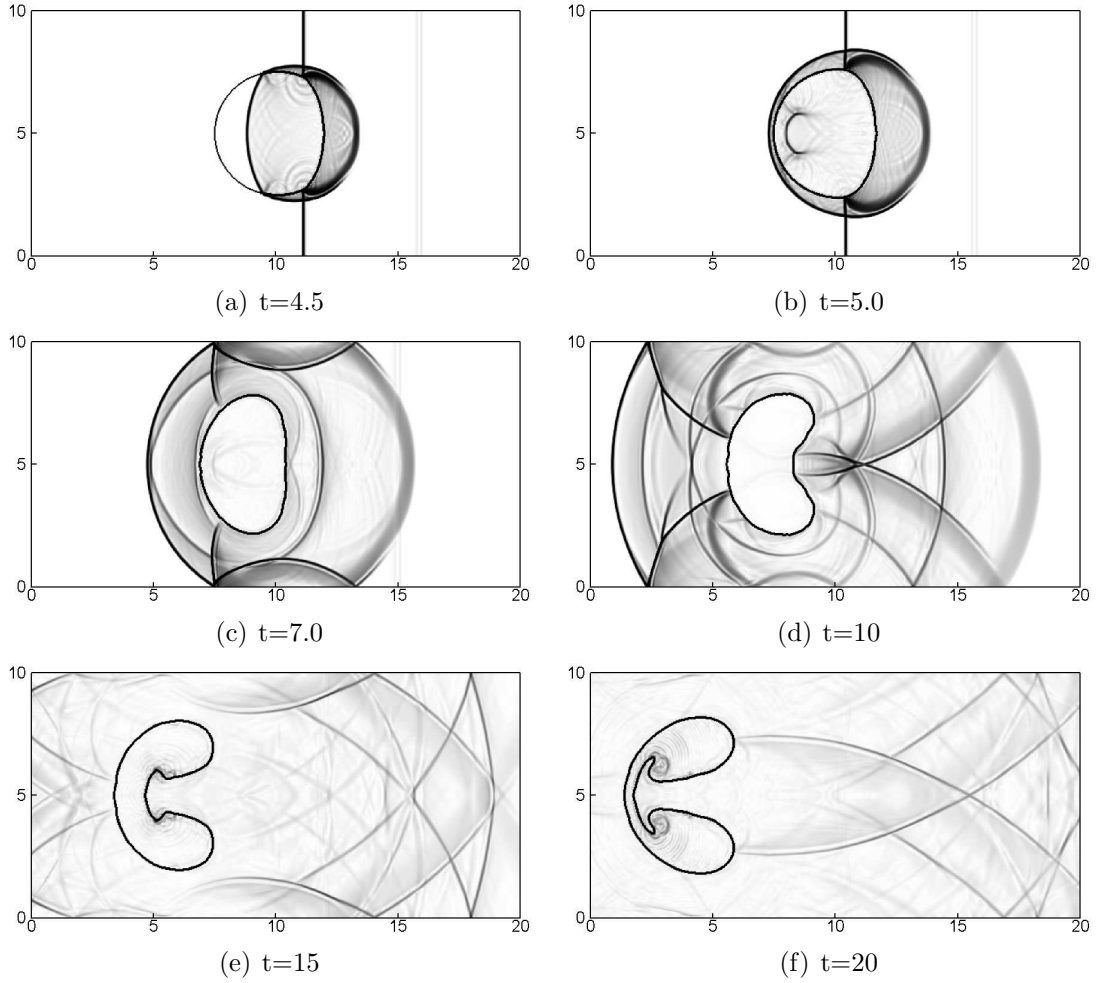


Figure 4.14: Example 4.6: Evolution of the shock-bubble interaction.

conditions as in [5], but they are in a non-dimensional form and only the upper half of the problem is simulated.

The computational domain is taken as $[0, 15] \times [0, 6]$ with 300×120 mesh cells. The lower boundary is the symmetry plane, the non-reflecting conditions are specified on the upper and left boundaries, while the right boundary conditions are taken as the post-shock values (the non-reflecting) before (after) the rarefaction wave reaches the right boundary.

Initially, a cylindrical air bubble with a diameter of 6 is immersed in water, and is centered at $(9,0)$, see the schematic diagram shown in Figure 4.16. Inside the cylindrical air bubble, the physical variables $(\rho, u, v, p, \gamma, B)$ are taken as $(0.001, 0, 0, 1, 1.4, 0)$. The pre-

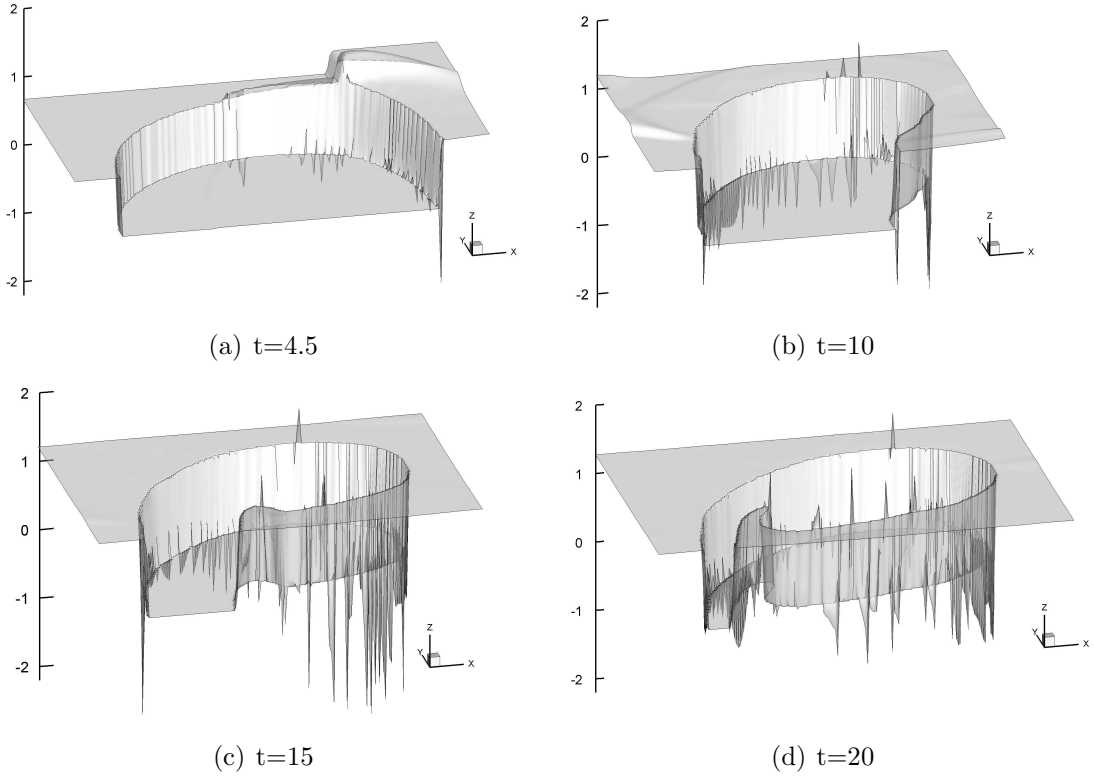


Figure 4.15: Example 4.6: Density distribution in the zoomed-in domain.

and post-shock states of the incident water shock wave are

$$(\rho, u, v, p, \gamma, B) = \begin{cases} (1.313345, 67.3267, 0, 19000, 7, 3309), & x > 12, \\ (1, 0, 0, 1, 7, 3309), & x < 12. \end{cases}$$

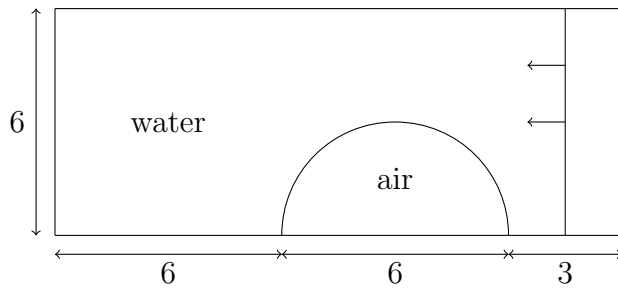


Figure 4.16: The geometry of Example 4.7.

We use the P^2 RKDG method to compute this problem with the TVB limiter constant $M = 0.15$ in (2.15) and show the results in Figure 4.17. All elapsed times are measured from the first hit on the air bubble by the shock wave. At time $t = 0.012$, a relatively weak curved shock wave is transmitted into the air and moves leftward, and a strong rarefaction wave is

produced and moves to the right. At $t = 0.025$ a water jet is formed and runs to the left along the symmetry plane $y = 6$. At $t = 0.030$, the water jet reaches the left wall of the air bubble, resulting in bubble collapse. On impact, an intense blast wave in the surrounding water is produced at $t = 0.034$. At $t = 0.037$, a clear compression wave propagates outwards from the air bubble (also see this phenomena in Figure 16 of [5] which was obtained on an adaptive mesh). This appears as a result of the transmission of the air shock into the water. At the same time, the air bubble begins to be drawn into the vortex core. Figures 4.18(a)-(d) show the density distribution in a zoomed-in domain. We again observe the high-frequency overshoots / undershoots which represent the level of effective conservation error.

Once more, the results reproduce all the known features of the interaction process and are in good agreement with the experimental results in [6]. This reinforces the conclusion that the combination of the P^2 RKDG method and the interface treating method works effectively.

5 Conclusions

The high-order accurate Runge-Kutta discontinuous Galerkin (RKDG) method is applied to the simulation of the compressible multi-medium flow. Riemann problems are constructed and solved to define the states on both sides of the interface. The input states of the Riemann problem are obtained by interpolating at the interface with the polynomials in neighbors of the mixed cell. The level set equation is solved by using a high-order accurate RKDG method for Hamilton-Jacobi equations, thus producing a unified DG solver for the combined problem. The method is conservative when the states in the mixed cells are accounted for, these states in the mixed cells are plotted to better assess effective conservation errors, as overshoots / undershoots in the mixed cells indicate the level of effective conservation errors. Numerical examples show that the results compare well with exact solutions for one-dimensional problems and the algorithm can capture all the known features of the two-dimensional shock-bubble interaction problems. In particular, the advantage of higher order

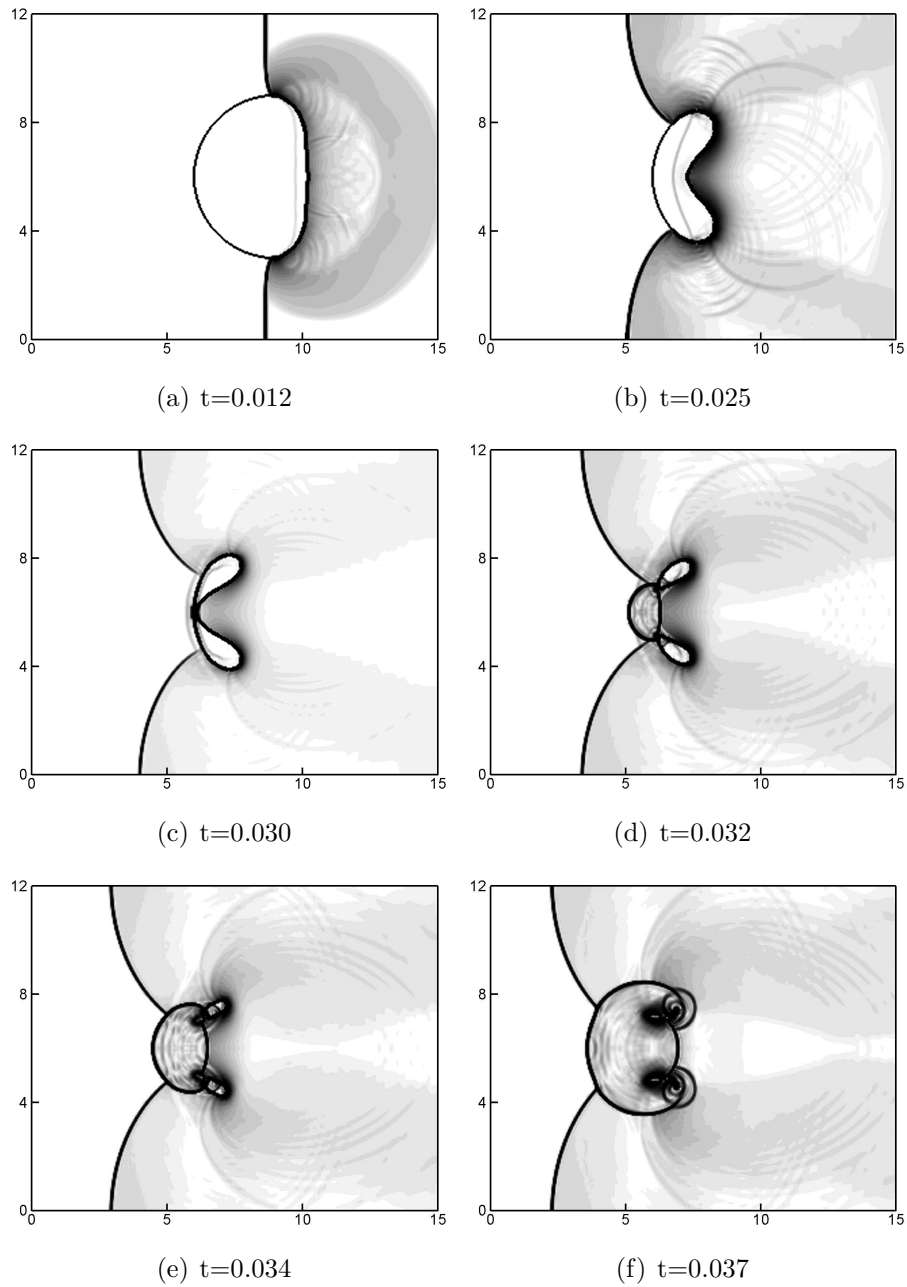


Figure 4.17: Example 4.7: Evolution of the shock-bubble interaction.

method is demonstrated through a simple model of shock interacting with density waves.

References

- [1] R. Abgrall, How to prevent oscillations in multicomponent flow calculations: A quasi conservative approach, *J. Comput. Phys.* 125 (1996) 150-160.

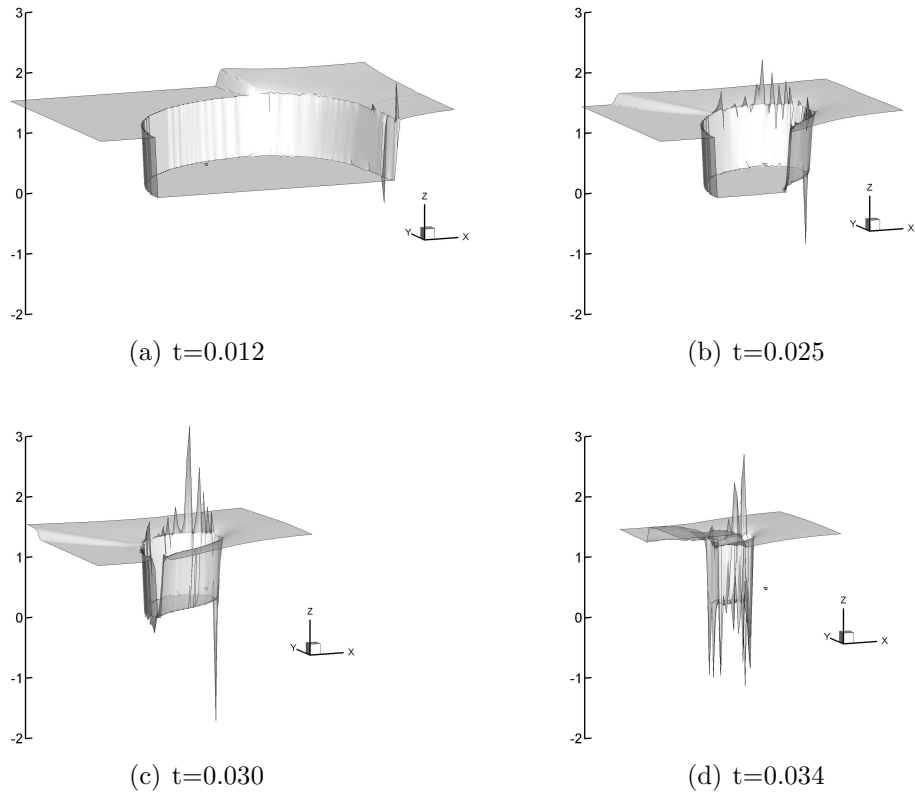


Figure 4.18: Example 4.7: Density distribution in the zoomed-in domain.

- [2] R. Abgrall and S. Karni, Computations of compressible multifluids, *J. Comput. Phys.* 169 (2001) 594-623.
- [3] D. Adalsteinsson and J.A. Sethian, A fast level set method for propagating interfaces, *J. Comput. Phys.*, 118 (1995) 269-277.
- [4] D. Adalsteinsson and J.A. Sethian, The fast construction of extension velocities in level set methods, *J. Comput. Phys.*, 148 (1999) 2-22.
- [5] G.J. Ball, B.P. Howell, T.G. Leighton and M.J. Schofield, Shock-induced collapse of a cylindrical air bubble in water: a free-Lagrange simulation, *Shock Waves*, 10 (2000) 265-276.
- [6] N.K. Bourne and J.E. Field, Shock-induced collapse of single cavities in liquids, *J. Fluid Mech.* 244 (1992) 225-240.

- [7] R. Caiden, R.P. Fedkiw and C. Anderson, A numerical method for two-phase flow consisting of separate compressible and incompressible regions, *J. Comput. Phys.* 166 (2001) 1-27.
- [8] Y.D. Cheng and C.-W. Shu, A discontinuous Galerkin finite element method for directly solving the Hamilton-Jacobi equations, *J. Comput. Phys.*, 223 (2007) 398-415.
- [9] A. Chertock, S. Karni and A. Kurganov, Interface tracking method for compressible multifluids, *Math. Model. Numer. Anal.* 42 (2008) 991-1019.
- [10] B. Cockburn, S. Hou and C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case, *Math. Comp.* 54 (1990) 545-581.
- [11] B. Cockburn, S.-Y. Lin and C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One dimensional systems, *J. Comput. Phys.* 84 (1989) 90-113.
- [12] B. Cockburn and C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws II: general framework, *Math. Comp.* 52 (1989) 411-435.
- [13] B. Cockburn and C.-W. Shu, The Runge-Kutta local projection P1-discontinuous Galerkin method for scalar conservation laws, *Math. Model. Numer. Anal.* 25 (1991) 337-361.
- [14] B. Cockburn and C.-W. Shu, The P1-RKDG Method for Two-dimensional Euler Equations of Gas Dynamics, ICASE Report 91-32, NASA Langley Research Center, 1991.
- [15] B. Cockburn and C.-W. Shu, The Runge-Kutta discontinuous Galerkin method for conservation laws V: Multidimensional systems, *J. Comput. Phys.* 141 (1998) 199-224.

- [16] R.P. Fedkiw, Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method, *J. Comput. Phys.* 175 (2002) 200-224.
- [17] R.P. Fedkiw, T. Aslam, B. Merriman and S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), *J. Comput. Phys.* 152 (1999) 457-492.
- [18] R.P. Fedkiw, T. Aslam and S. Xu, The ghost fluid method for deflagration and detonation discontinuities, *J. Comput. Phys.* 154 (1999) 393-427.
- [19] J. Glimm, J.W. Grove, X.L. Li, K.-M. Shyue, Y. Zeng and Q. Zhang, Three-dimensional front tracking, *SIAM J. Sci. Comput.* 19 (1998) 703-727.
- [20] J. Glimm, X.L. Li, Y. Liu and N. Zhao, Conservative front tracking and level set algorithms, *Proc. Natl. Acad. Sci. USA* 98 (2001) 14198-14201.
- [21] J. Glimm, Y. Liu, Z. Xu and N. Zhao, Conservative front tracking with improved accuracy, *SIAM J. Numer. Anal.* 41 (2003) 1926-1947.
- [22] M. Kang, R.P. Fedkiw and X.D. Liu, A boundary condition capturing method for multiphase incompressible flow, *J. Sci. Comput.* 15 (2000) 323-360.
- [23] T.G. Liu, B.C. Khoo and C.W. Wang, The ghost fluid method for gas-water simulation, *J. Comput. Phys.* 204 (2005) 193-221.
- [24] T.G. Liu, B.C. Khoo and W.F. Xie, The modified ghost fluid method as applied to extreme fluid-structure interaction in the presence of cavitation, *Commun. Comput. Phys.* 1 (2006) 898-919.
- [25] X.D. Liu, R.P. Fedkiw and M. Kang, A boundary condition capturing method for Poisson's equation on irregular domain, *J. Comput. Phys.* 160 (2000) 151-178.
- [26] D.Q. Nguyen, R.P. Fedkiw and M. Kang, A boundary condition capturing method for incompressible flame discontinuities, *J. Comput. Phys.* 172 (2001) 71-98.

- [27] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations, *J. Comput. Phys.* 79 (1988) 12-49.
- [28] J.X. Qiu, T.G. Liu and B.C. Khoo, Runge-Kutta discontinuous Galerkin methods for compressible two-medium flow simulations: one dimensional case, *J. Comput. Phys.* 222 (2007) 353-373.
- [29] J.X. Qiu, T.G. Liu and B.C. Khoo, Simulations of compressible two-medium flow by Runge-Kutta discontinuous Galerkin methods with the ghost fluid method, *Commun. Comput. Phys.* 3 (2008) 479-504.
- [30] J.J. Quirk and S. Karni, On the dynamics of a shock-bubble interaction. *J. Fluid Mech.* 318 (1996) 129-163.
- [31] C.-W. Shu, TVB uniformly high-order schemes for conservation laws, *Math. Comp.* 49 (1987) 105-121.
- [32] C.-W. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, *J. Comput. Phys.* 77 (1988) 439-471.
- [33] C.-W. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, II, *J. Comput. Phys.* 83 (1989) 32-78.
- [34] E.F. Toro, Riemann solvers and numerical method for fluid dynamics: a practical introduction, Berlin: Springer-Verlag, 1997.
- [35] C.W. Wang, T.G. Liu and B.C. Khoo, A real-ghost fluid method for the simulation of multi-medium compressible flow, *SIAM J. Sci. Comput.* 28 (2006) 278-302.
- [36] C.W. Wang, H.Z. Tang and T.G. Liu, An adaptive ghost fluid finite volume method for compressible gas-water simulations, *J. Comput. Phys.* 227 (2008) 6385-6409