Journal of Computational Physics 228 (2009) 8872–8891

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

High order conservative Lagrangian schemes with Lax–Wendroff type time discretization for the compressible Euler equations

Wei Liu [a], Juan Cheng [b,1], Chi-Wang Shu [c,*,2]

[a] Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
[b] Institute of Applied Physics and Computational Mathematics, Beijing 100088, China
[c] Division of Applied Mathematics, Brown University, Providence, RI 02912, USA

ARTICLE INFO

Article history:
Received 21 May 2009
Received in revised form 30 August 2009
Accepted 2 September 2009
Available online 6 September 2009

Keywords:
Lagrangian scheme
Lax–Wendroff type time discretization
High order accuracy
Conservative scheme
ENO reconstruction
ALE method

ABSTRACT

In this paper, we explore the Lax–Wendroff (LW) type time discretization as an alternative procedure to the high order Runge–Kutta time discretization adopted for the high order essentially non-oscillatory (ENO) Lagrangian schemes developed in [3,5]. The LW time discretization is based on a Taylor expansion in time, coupled with a local Cauchy–Kowalewski procedure to utilize the partial differential equation (PDE) repeatedly to convert all time derivatives to spatial derivatives, and then to discretize these spatial derivatives based on high order ENO reconstruction. Extensive numerical examples are presented, for both the second-order spatial discretization using quadrilateral meshes [3] and third-order spatial discretization using curvilinear meshes [5]. Comparing with the Runge–Kutta time discretization procedure, an advantage of the LW time discretization is the apparent saving in computational cost and memory requirement, at least for the two-dimensional Euler equations that we have used in the numerical tests.

1. Introduction

In this paper, we are interested in solving the compressible Euler gas dynamic equations written in the Lagrangian form. Lagrangian methods and Arbitrary Lagrangian–Eulerian methods (ALE, [12,20]) which contain a Lagrangian phase, are widely used in astrophysics and computational fluid dynamics. In the Lagrangian methods, a computational cell moves with the flow velocity. Comparing with the Eulerian method, Lagrangian type methods can reduce the numerical error due to the advection terms in the conservation equations and can capture contact discontinuities sharply [16]. Even though the Euler equations are much simpler in the Lagrangian framework, in the multi-dimensional case they are actually more difficult to solve since the mesh moves with the fluid and can easily get distorted. In the past years, many efforts have been made to develop Lagrangian methods. Some algorithms are developed from the nonconservative form of the Euler equations of which density, velocity and internal energy are solved directly, for example, those discussed in [1,2,18,29]. The other class of Lagrangian algorithms starts from the conservative form of the Euler equations. These schemes solve the conservative quantities such as mass, momentum and total energy directly, thus guaranteeing exact conservation. Examples include [6,14,17].

* Corresponding author. Tel.: +1 401 863 2549; fax: +1 401 863 1355.
E-mail addresses: liuwei@lsec.cc.ac.cn (W. Liu), cheng_juan@iapcm.ac.cn (J. Cheng), shu@dam.brown.edu (C.-W. Shu).

[1] Research is supported in part by NSFC Grants 10972043 and 10931004. Additional support is provided by the National Basic Research Program of China under Grant 2005CB321702.
[2] Research supported by AFOSR Grant FA9550-09-1-0126 and NSF Grant DMS-0809086.

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.09.001

Most existing Lagrangian type schemes for Euler equations are first or at most second-order accurate. In [3], Cheng and Shu developed a class of Lagrangian type schemes for solving the Euler equations which are based on the high order ENO reconstruction both in the Cartesian and cylindrical coordinates. These schemes are conservative for the density, momentum and total energy, and can maintain formal high order accuracy both in space and in time for the one-dimensional case and formal second-order accuracy for the two-dimensional case if the mesh consists of quadrilaterals with straight-line edges. In [5], Cheng and Shu developed a third-order Lagrangian type scheme on curved quadrilateral meshes in two space dimensions.

The time discretization used in [3,5] is based on high order TVD Runge–Kutta (RK) methods [26]. An alternative time discretization technique is the Lax–Wendroff (LW) type procedure. Lax–Wendroff time discretization is based on the idea of the classical Lax–Wendroff scheme [15], and it relies on converting all the time derivatives in a temporal Taylor expansion into spatial derivatives by repeatedly using the partial differential equation (PDE) and its differentiated versions. Lax–Wendroff type time discretization usually produces high order accuracy with a more compact spatial stencil. It also uses the original PDE more extensively. The original finite volume ENO scheme in [11] used this approach for the time discretization. Recently a Lax–Wendroff type time discretization procedure for high order finite difference WENO schemes was developed by Qiu and Shu [22]. This approach was also used by Titarev and Toro [28] and Schwartzkopff et al. [23], termed ADER (arbitrary high order schemes utilizing higher order derivatives), to construct a class of high order schemes for conservation laws in finite volume version. The Lax–Wendroff type time discretization was also used in the discontinuous Galerkin method [21]. Detailed descriptions of the Lax–Wendroff procedure for finite volume and discontinuous Galerkin schemes solving Euler equations in the Eulerian coordinates are given in, e.g. [10,9,8].

In this paper, we use the LW time discretization to develop a high order conservative Lagrangian type scheme for the compressible Euler equations. Comparing with the Runge–Kutta time discretization procedure, an advantage of the LW time discretization is the apparent saving in computational cost and memory requirement, at least for the two-dimensional Euler equations that we have used in the numerical tests. We provide a series of one and two-dimensional numerical examples both in quadrilateral meshes (second-order scheme) and in curvilinear meshes (third-order scheme) to demonstrate the performance of the method.

The organization of the paper is as follows. In Section 2, we describe the construction of the schemes with algorithm formulation and numerical examples in one space dimension. In Section 3, two-dimensional schemes with numerical examples are presented. Concluding remarks are given in Section 4.

2. One-dimensional case

2.1. The compressible Euler equations in the Lagrangian formulation

In fluid dynamics, the integral form of the conservation law of mass, momentum and energy can be expressed as:

$$\frac{d}{dt}\int_{\Omega_t}\mathbf{U}dV+\int_{\partial\Omega_t}\mathbf{F}dS=0, \tag{1}$$

where Ω_t is the moving control volume and $\partial\Omega_t$ is the boundary. The vector of the conserved variables $\mathbf{U}$ and the flux vector $\mathbf{F}$ are given by

$$\mathbf{U}=\begin{pmatrix}\rho\\ \mathbf{M}\\ E\end{pmatrix},\quad \mathbf{F}=\begin{pmatrix}f_D\\ f_M\\ f_E\end{pmatrix}=\begin{pmatrix}(\mathbf{u}-\dot{\mathbf{x}})\cdot\mathbf{n}\rho\\ (\mathbf{u}-\dot{\mathbf{x}})\cdot\mathbf{n}\mathbf{M}+p\cdot\mathbf{n}\\ (\mathbf{u}-\dot{\mathbf{x}})\cdot\mathbf{n}E+p\mathbf{u}\cdot\mathbf{n}\end{pmatrix}, \tag{2}$$

where ρ is the density, $\mathbf{u}$ is the velocity, $\mathbf{M}=\rho\mathbf{u}$ is the momentum, E is the total energy and p is the pressure. $\dot{\mathbf{x}}$ is the velocity of the control volume boundary $\partial\Omega_t$, and $\mathbf{n}$ denotes the unit outward normal to $\partial\Omega_t$. The system (1) is the governing equation for unsteady compressible flows in Cartesian coordinates.

In this paper, we consider the ideal gas. The following well known equation of state (EOS) is used to complete the set of equations

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

where $e=\frac{E}{\rho}-\frac{1}{2}|\mathbf{u}|^2$ is the specific internal energy, and γ is a constant representing the ratio of specific heat capacities of the fluid.

This paper focuses on solving the governing equations (1) and (2) in a Lagrangian framework, in which it is assumed that

$$\dot{\mathbf{x}}=\mathbf{u}, \tag{4}$$

hence the vectors $\mathbf{U}$ and $\mathbf{F}$ take the simpler form

$$\mathbf{U}=\begin{pmatrix}\rho\\ \mathbf{M}\\ E\end{pmatrix},\quad \mathbf{F}=\begin{pmatrix}0\\ p\cdot\mathbf{n}\\ p\mathbf{u}\cdot\mathbf{n}\end{pmatrix}. \tag{5}$$

2.2. The construction of the scheme in the 1D case

We develop a conservative Lagrangian finite volume scheme by solving the conserved variables density, momentum and total energy directly. All the conserved variables are in the form of cell averages.

The spatial domain Ω_t is discretized into N computational cells $I_i(t) = [x_{i-1/2}(t), x_{i+1/2}(t)]$ of sizes $\Delta x_i(t) = x_{i+1/2}(t) - x_{i-1/2}(t)$ with $i = 1, \ldots, N$. We denote $\Delta x = \max_i \Delta x_i$. For a given cell $I_i(t)$, the values of the cell averages of mass, momentum and total energy, denoted by $\bar{\rho}_i(t)$, $\overline{M}_i(t)$ and $\overline{E}_i(t)$, are defined as follows:

$$\bar{\rho}_i(t) = \frac{1}{\Delta x_i(t)} \int_{I_i(t)} \rho(x,t)dx, \quad \overline{M}_i(t) = \frac{1}{\Delta x_i(t)} \int_{I_i(t)} M(x,t)dx, \quad \overline{E}_i(t) = \frac{1}{\Delta x_i(t)} \int_{I_i(t)} E(x,t)dx.$$

Using the governing equation (1) on every single cell $I_i(t)$, we have

$$\frac{d}{dt}\begin{pmatrix} \bar{\rho}_i(t)\Delta x_i(t) \\ \overline{M}_i(t)\Delta x_i(t) \\ \overline{E}_i(t)\Delta x_i(t) \end{pmatrix} = -\begin{pmatrix} f_D(\mathbf{U}(x_{i+1/2}(t),t)) - f_D(\mathbf{U}(x_{i-1/2}(t),t)) \\ f_M(\mathbf{U}(x_{i+1/2}(t),t)) - f_M(\mathbf{U}(x_{i-1/2}(t),t)) \\ f_E(\mathbf{U}(x_{i+1/2}(t),t)) - f_E(\mathbf{U}(x_{i-1/2}(t),t)) \end{pmatrix}. \tag{6}$$

After integrating Eqs. (4) and (6) over $[t^n, t^{n+1}]$, we obtain:

$$x^{n+1}_{i\pm1/2} = x^n_{i\pm1/2} + \int_{t_n}^{t_{n+1}} u(x_{i\pm1/2}(t),t)dt \tag{7}$$

and

$$\begin{pmatrix} \bar{\rho}^{n+1}_i \Delta x^{n+1}_i \\ \overline{M}^{n+1}_i \Delta x^{n+1}_i \\ \overline{E}^{n+1}_i \Delta x^{n+1}_i \end{pmatrix} = \begin{pmatrix} \bar{\rho}^{n}_i \Delta x^{n}_i \\ \overline{M}^{n}_i \Delta x^{n}_i \\ \overline{E}^{n}_i \Delta x^{n}_i \end{pmatrix} - \int_{t_n}^{t_{n+1}} \begin{pmatrix} f_D(\mathbf{U}(x_{i+1/2}(t),t)) - f_D(\mathbf{U}(x_{i-1/2}(t),t)) \\ f_M(\mathbf{U}(x_{i+1/2}(t),t)) - f_M(\mathbf{U}(x_{i-1/2}(t),t)) \\ f_E(\mathbf{U}(x_{i+1/2}(t),t)) - f_E(\mathbf{U}(x_{i-1/2}(t),t)) \end{pmatrix} dt. \tag{8}$$

For the simplicity of description, in the following, we denote f as a component of the flux vector $\mathbf{F}$. Notice that f and $\mathbf{U}$ in the equations above represent the values of the flux and the conservative variables of the solution, not their cell averages. In this paper, we will always use $\mathbf{U}$ to denote the values of the solution, and $\mathbf{U}^{\pm}$ to denote the numerically reconstructed values at a given point along the cell boundary from the cell averages $\overline{\mathbf{U}}$. In order to obtain a fully discrete scheme from (7) and (8), we have to do two things. First, we would need to find an average velocity $\hat{u}$ (see Remark 4) to replace the physical velocity u in Eq. (7) and a numerical flux to replace the physical flux f in Eq. (8). Second, we would need to choose a numerical method to approximate the integrals. Suppose we want to get a rth order accurate scheme in both space and time, we should find a numerical flux $\hat{f}(\mathbf{U}^-, \mathbf{U}^+)$ which is at least a rth order approximation to the physical flux $f(\mathbf{U})$, an average velocity $\hat{u}$ which is at least a rth order approximation to the physical velocity u, and at least a $(r+1)$th order numerical integration to approximate the integral on the right hand side of (7) and (8). That is, we require

$$\begin{aligned} &\hat{f}\big(\mathbf{U}^-(x_{i+1/2}(t),t), \mathbf{U}^+(x_{i+1/2}(t),t)\big) = f\big(\mathbf{U}(x_{i+1/2}(t),t)\big) + \mathcal{O}(\Delta x^r), \\ &\hat{u}\big(u^-(x_{i+1/2}(t),t), u^+(x_{i+1/2}(t),t)\big) = u(x_{i+1/2}(t),t) + \mathcal{O}(\Delta x^r), \\ &\int_{t_n}^{t_{n+1}} \hat{f}\big(\mathbf{U}^-(x_{i+1/2}(t),t), \mathbf{U}^+(x_{i+1/2}(t),t)\big)dt = \Delta t \sum_{k=0}^{K} \alpha_k \hat{f}(t_n + \beta_k \Delta t) + \mathcal{O}(\Delta t^{r+1}), \\ &\int_{t_n}^{t_{n+1}} \hat{u}\big(u^-(x_{i+1/2}(t),t), u^+(x_{i+1/2}(t),t)\big)dt = \Delta t \sum_{k=0}^{K} \alpha_k \hat{u}(t_n + \beta_k \Delta t) + \mathcal{O}(\Delta t^{r+1}), \end{aligned} \tag{9}$$

where α_k are the coefficients of the Gaussian quadrature and $t_n + \beta_k \Delta t$ are the Gaussian quadrature points.

Once we have a consistent and Lipschitz continuous numerical flux, the following relationship holds:

$$\hat{f}(\mathbf{U}^-, \mathbf{U}^+) = f(\mathbf{U}) + \mathcal{O}(|\mathbf{U} - \mathbf{U}^-|, |\mathbf{U} - \mathbf{U}^+|). \tag{10}$$

Therefore, as long as we can get an approximation to the value $\mathbf{U}$ with at least rth order accuracy, the numerical flux will be a rth order approximation to the physical flux. Combined with the numerical integration, it is clear that we would only need to find an approximation at the Gaussian quadrature points with rth order accuracy:

$$\begin{aligned} &\mathbf{U}^{\pm}(x_{i+1/2}(t_n + \beta_k \Delta t), t_n + \beta_k \Delta t) = \mathbf{U}(x_{i+1/2}(t_n + \beta_k \Delta t), t_n + \beta_k \Delta t) + \mathcal{O}(\Delta x^r), \\ &u^{\pm}(x_{i+1/2}(t_n + \beta_k \Delta t), t_n + \beta_k \Delta t) = u(x_{i+1/2}(t_n + \beta_k \Delta t), t_n + \beta_k \Delta t) + \mathcal{O}(\Delta x^r). \end{aligned} \tag{11}$$

Since we are developing a finite volume scheme, we only have the information of the cell averages of the solution at the time t_n. We would therefore use a Taylor expansion in time to obtain

$$\mathbf{U}^{\pm}(x_{i+1/2}(t_n+\beta_k\Delta t),t_n+\beta_k\Delta t)=\mathbf{U}^{n\pm}_{i+1/2}+\sum_{l=1}^{r-1}\frac{(\beta_k\Delta t_n)^l}{l!}\frac{d^l\mathbf{U}^{n\pm}_{i+1/2}}{dt^l}+\mathcal{O}(\Delta t^r), \tag{12}$$

$$u^{\pm}(x_{i+1/2}(t_n+\beta_k\Delta t),t_n+\beta_k\Delta t)=u^{n\pm}_{i+1/2}+\sum_{l=1}^{r-1}\frac{(\beta_k\Delta t_n)^l}{l!}\frac{d^lu^{n\pm}_{i+1/2}}{dt^l}+\mathcal{O}(\Delta t^r),$$

where $\mathbf{U}^{n\pm}_{i+1/2}=\mathbf{U}^{\pm}(x_{i+1/2}(t_n),t_n)$ and $u^{n\pm}_{i+1/2}=u^{\pm}(x_{i+1/2}(t_n),t_n)$.

In order to guarantee (11) and (12), we would need to use reconstruction to obtain the values $\mathbf{U}^{n\pm}_{i+1/2}$, $u^{n\pm}_{i+1/2}$ and $\frac{d^l\mathbf{U}^{n\pm}_{i+1/2}}{dt^l}$, $\frac{d^lu^{n\pm}_{i+1/2}}{dt^l}$ from the cell average variables $\overline{\mathbf{U}}^n_i$ at time t^n such that the following holds:

$$\begin{aligned}&\mathbf{U}^{n\pm}_{i+1/2}=\mathbf{U}^{n}_{i+1/2}+\mathcal{O}(\Delta x^r),\\&u^{n\pm}_{i+1/2}=u^{n}_{i+1/2}+\mathcal{O}(\Delta x^r)\end{aligned} \tag{13}$$

and

$$\begin{aligned}&\frac{d^l\mathbf{U}^{n\pm}_{i+1/2}}{dt^l}=\frac{d^l\mathbf{U}^{n}_{i+1/2}}{dt^l}+\mathcal{O}(\Delta t^{r-l}),\\&\frac{d^lu^{n\pm}_{i+1/2}}{dt^l}=\frac{d^lu^{n}_{i+1/2}}{dt^l}+\mathcal{O}(\Delta t^{r-l}),\quad l=1,2,\ldots,r-1.\end{aligned} \tag{14}$$

The requirement (13) is relatively easy to satisfy, as many well known reconstructions can be applied. To satisfy (14), we would need to use a local Cauchy–Kowalewski procedure, which is a key step in the Lax–Wendroff procedure, to convert the time derivatives $\frac{d^l\mathbf{U}^{n\pm}_{i+1/2}}{dt^l}$ and $\frac{d^lu^{n\pm}_{i+1/2}}{dt^l}$ for $l=1,2,\ldots,r-1$, to spatial derivatives $\frac{\partial^l\mathbf{U}^{n\pm}_{i+1/2}}{\partial x^l}$ and $\frac{\partial^lu^{n\pm}_{i+1/2}}{\partial x^l}$. We take the scalar conservation law $u_t+f(u)_x=0$ as a example, the procedure is:

$$\begin{aligned}&u_t=-f'u_x,\\&u_{xt}=-[f''(u_x)^2+f'u_{xx}],\\&u_{tt}=-[f''u_tu_x+f'u_{xt}],\\&u_{xxt}=-[f'''(u_x)^3+3f''u_xu_{xx}+f'u_{xxx}],\\&u_{xtt}=-[f'''(u_x)^2u_t+f''(2u_xu_{xt}+u_tu_{xx})+f'u_{xxt}],\\&u_{ttt}=-[f'''(u_t)^2u_x+f''(2u_tu_{xt}+u_xu_{tt})+f'u_{xtt}].\end{aligned} \tag{15}$$

The procedure for the Euler equations is more complicated and will be introduced when we describe the actual algorithm. Through this procedure, we now only require the following to guarantee the validity of (13) and (14):

$$\begin{aligned}&\frac{\partial^l\mathbf{U}^{n\pm}_{i+1/2}}{\partial x^l}=\frac{\partial^l\mathbf{U}^{n}_{i+1/2}}{\partial x^l}+\mathcal{O}(\Delta x^{r-l}),\\&\frac{\partial^lu^{n\pm}_{i+1/2}}{\partial x^l}=\frac{\partial^lu^{n}_{i+1/2}}{\partial x^l}+\mathcal{O}(\Delta x^{r-l}),\quad l=0,1,\ldots,r-1.\end{aligned} \tag{16}$$

Fortunately, the above equation is satisfied by any polynomial reconstruction of degree $r-1$, for example the ENO reconstruction. Therefore, we can obtain an arbitrary rth order accurate scheme in space and time by using the standard ENO reconstruction and the Lax–Wendroff procedure outlined above.

Remark 1. In the analysis above, we have not distinguished between Δx and Δt when referring to the order of accuracy. This is because of the CFL condition:

$$\Delta t^n=\lambda\min_{i=1,\ldots,N}\left(\Delta x^n_i/c^n_i\right),$$

where c_i is the sound speed in the cell I_i. The Courant number λ is chosen as 0.6 in our one-dimensional computation.

Remark 2. We use the ENO idea to reconstruct polynomial functions on each I_i by using the information of the cell I_i and its neighbors, see [11,3]. The procedure allows us to obtain arbitrarily high order accurate approximation by a suitable reconstruction. In this paper, we use only second and third-order reconstructions in our numerical examples.

Remark 3. After the reconstruction, we obtain two approximation values at each cell boundary point $x_{i\pm1/2}$, one from the ENO reconstruction polynomial in the cell I_i and the other one from that in the cell I_{i+1}. A consistent Lipschitz continuous numerical flux is then used. In this paper, we will use the following four typical numerical fluxes: (1) the Godunov flux; (2) the Dukowicz flux; (3) the Lax–Friedrichs (L–F) flux; (4) the HLLC flux, although we will only show the results obtained with the Dukowicz flux and the HLLC flux to save space. We refer to [3] for implementation details of these fluxes in the context of Lagrangian schemes and will not repeat them here.

Remark 4. In the Lagrangian formulation, the grid moves with the fluid. In every time step, we should determine the new location of the grid points. This is determined by the velocity at the cell boundary point $x_{i\pm1/2}$. At this point, we have two approximate values of velocity, one from the left cell I_i and the other from the right cell I_{i+1}, which are obtained from the reconstructed values of density and momentum. For the Godunov flux or the Dukowicz flux, we solve (exactly or approximately) the Riemann problem at $x_{i\pm1/2}$, hence we obtain the value of velocity through this Riemann solver. For the L–F flux or the HLLC flux, the velocity is defined as the Roe average of the two reconstructed values from both sides

$$\hat{u}_{i+1/2} = \frac{\sqrt{\rho^-_{i+1/2}}u^-_{i+1/2} + \sqrt{\rho^+_{i+1/2}}u^+_{i+1/2}}{\sqrt{\rho^-_{i+1/2}} + \sqrt{\rho^+_{i+1/2}}}. \tag{17}$$

2.3. Algorithm description

We now describe our algorithm in detail. Starting from the mesh and the approximate cell averages of the solution at time level t^n, we will proceed as follows.

1. At each fixed $x_{i+1/2}$, use the rth order characteristic-wise ENO reconstruction to obtain the value $\mathbf{U}^{n\pm}_{i+1/2}$ and $\frac{\partial^l \mathbf{U}^{n\pm}_{i+1/2}}{\partial x^l}$, for $l = 1, 2, \ldots, r-1$, from the cell averages $\overline{\mathbf{U}}^n_i$ at t^n.
2. Calculate the derivatives in t of the variables $\frac{\partial^l \mathbf{U}^{n\pm}_{i+1/2}}{\partial t^l}$ and $\frac{\partial^l u^{n\pm}_{i+1/2}}{\partial t^l}$, for $l = 1, 2, \ldots, r-1$, at each $x_{i+1/2}$, using the information from the ENO reconstruction in the following way: First, compute u, p, u_x, p_x by using $\mathbf{U}^{n\pm}$ and $\frac{\partial \mathbf{U}^{n\pm}}{\partial x}$

$$\begin{aligned} &u = M/\rho, \\ &p = (\gamma-1)(E - 0.5\rho u^2), \\ &u_x = (M_x - u\rho_x)/\rho, \\ &p_x = (\gamma-1)(E_x - 0.5u_x M - 0.5uM_x). \end{aligned} \tag{18}$$

Then, compute $\rho_t, M_t, u_t, E_t, p_t$

$$\begin{aligned} &\rho_t = -M_x, \\ &M_t = -u_x M - uM_x - p_x, \\ &u_t = (M_t - u\rho_t)/\rho, \\ &E_t = -E_x u - Eu_x - p_x u - pu_x, \\ &p_t = -up_x - \gamma p u_x. \end{aligned} \tag{19}$$

In order to compute $\rho_{tt}, M_{tt}, E_{tt}$, first compute $u_{xx}, p_{xx}, \rho_{xt}, M_{xt}, E_{xt}, u_{xt}, p_{xt}$

$$\begin{aligned} &u_{xx} = (M_{xx} - u\rho_{xx} - 2u_x\rho_x)/\rho, \\ &p_{xx} = (\gamma-1)(E_{xx} - 0.5(Mu_{xx} + 2u_x M_x + uM_{xx})), \\ &\rho_{xt} = -M_{xx}, \\ &M_{xt} = -2u_x M_x - uM_{xx} - Mu_{xx} - p_{xx}, \\ &E_{xt} = -2u_x E_x - uE_{xx} - Eu_{xx} - 2p_x u_x - pu_{xx} - up_{xx}, \\ &u_{xt} = (M_{xt} - u\rho_{xt} - u_t\rho_x - u_x\rho_t)/\rho, \\ &p_{xt} = -up_{xx} - \gamma p u_{xx} - (1+\gamma)u_x p_x, \end{aligned} \tag{20}$$

then we can compute $\rho_{tt}, M_{tt}, E_{tt}, p_{tt}, u_{tt}$

$$\begin{aligned} &\rho_{tt} = -M_{xt}, \\ &M_{tt} = -uM_{xt} - Mu_{xt} - p_{xt} - u_x M_t - u_t M_x, \\ &E_{tt} = -uE_{xt} - Eu_{xt} - pu_{xt} - up_{xt} - u_x E_t - u_t E_x - p_x u_t - p_t u_x, \\ &p_{tt} = -up_{xt} - \gamma p u_{xt} - u_t p_x - \gamma p_t u_x, \\ &u_{tt} = (M_{tt} - 2\rho_t u_t - u\rho_{tt})/\rho. \end{aligned} \tag{21}$$

This procedure can be continued to higher t derivatives.

3. Compute $\frac{d^l \mathbf{U}^{n\pm}_{i+1/2}}{dt^l}$ and $\frac{d^l u^{n\pm}_{i+1/2}}{dt^l}$, for $l = 1, 2, \ldots, r-1$, via

$$\begin{aligned} \frac{d\mathbf{U}^{n\pm}_{i+1/2}}{dt} &= \frac{\partial \mathbf{U}^{n\pm}_{i+1/2}}{\partial t} + u^{n\pm}_{i+1/2}\frac{\partial \mathbf{U}^{n\pm}_{i+1/2}}{\partial x}, \\ \frac{du^{n\pm}_{i+1/2}}{dt} &= \frac{\partial u^{n\pm}_{i+1/2}}{\partial t} + u^{n\pm}_{i+1/2}\frac{\partial u^{n\pm}_{i+1/2}}{\partial x}, \end{aligned}$$

$$\frac{d^2\mathbf{U}^{n\pm}_{i+1/2}}{dt^2} = \frac{\partial}{\partial t}\left(\frac{d\mathbf{U}^{n\pm}_{i+1/2}}{dt}\right) + u^{n\pm}_{i+1/2}\frac{\partial}{\partial x}\left(\frac{d\mathbf{U}^{n\pm}_{i+1/2}}{dt}\right), \tag{22}$$
$$\frac{d^2u^{n\pm}_{i+1/2}}{dt^2} = \frac{\partial}{\partial t}\left(\frac{du^{n\pm}_{i+1/2}}{dt}\right) + u^{n\pm}_{i+1/2}\frac{\partial}{\partial x}\left(\frac{du^{n\pm}_{i+1/2}}{dt}\right).$$

4. Compute the value of the solution at the Gaussian quadrature points $\mathbf{U}^{\pm}_k = \mathbf{U}^{\pm}(x_{i+1/2}(t_n+\beta_k\Delta t), t_n+\beta_k\Delta t)$ and $u^{\pm}_k = u^{\pm}(x_{i+1/2}(t_n+\beta_k\Delta t), t_n+\beta_k\Delta t)$ for $k = 1,2,\ldots,K$, via

$$\mathbf{U}^{\pm}_k = \mathbf{U}^{n\pm}_{i+1/2} + \sum_{l=1}^{r-1}\frac{(\beta_k\Delta t_n)^l}{l!}\frac{d^l\mathbf{U}^{\pm}}{dt^l}, \tag{23}$$
$$u^{\pm}_k = u^{n\pm}_{i+1/2} + \sum_{l=1}^{r-1}\frac{(\beta_k\Delta t_n)^l}{l!}\frac{d^lu^{\pm}}{dt^l}.$$

5. Compute the numerical flux $\hat{f}(\mathbf{U}^-(t_n+\beta_k\Delta t), \mathbf{U}^+(t_n+\beta_k\Delta t))$ and the velocity $\hat{u}_k(\mathbf{U}^-_k, \mathbf{U}^+_k)$, for $k = 1,2,\ldots,K$, by one of the following four numerical fluxes: Godunov flux, Dukowicz flux, L–F flux, or HLLC flux.
6. Compute the time averaged flux $\bar{f}_{i+1/2} = \frac{1}{\Delta t_n}\int_{t_n}^{t_{n+1}}\hat{f}(\mathbf{U}^-(x_{i+1/2}(t),t), \mathbf{U}^+(x_{i+1/2}(t),t))dt$ and time averaged velocity $\bar{u}_{i+1/2} = \frac{1}{\Delta t_n}\int_{t_n}^{t_{n+1}}u(x_{i+1/2}(t),t)dt$ via

$$\bar{f}_{i+1/2} = \sum_{k=0}^{K}\alpha_k\hat{f}(\mathbf{U}^-_k, \mathbf{U}^+_k), \quad \bar{u}_{i+1/2} = \sum_{k=0}^{K}\alpha_k\hat{u}(\mathbf{U}^-_k, \mathbf{U}^+_k). \tag{24}$$

7. Compute the updated values $x^{n+1}_{i+1/2}$ and $\overline{\mathbf{U}}^{n+1}_i$ at t^{n+1} via

$$\begin{aligned}
x^{n+1}_{i+1/2} &= x^n_{i+1/2} + \Delta t_n\bar{u}_{i+1/2},\\
\bar{\rho}^{n+1}_i\Delta x^{n+1}_i &= \bar{\rho}^n_i\Delta x^n_i - \Delta t_n\left(\bar{f}^D_{i+1/2} - \bar{f}^D_{i-1/2}\right),\\
\overline{M}^{n+1}_i\Delta x^{n+1}_i &= \overline{M}^n_i\Delta x^n_i - \Delta t_n\left(\bar{f}^M_{i+1/2} - \bar{f}^M_{i-1/2}\right),\\
\overline{E}^{n+1}_i\Delta x^{n+1}_i &= \overline{E}^n_i\Delta x^n_i - \Delta t_n\left(\bar{f}^E_{i+1/2} - \bar{f}^E_{i-1/2}\right).
\end{aligned} \tag{25}$$

2.4. Numerical examples in the 1D case

2.4.1. Accuracy test

This is a problem with smooth solutions. We use this problem to test the accuracy of our schemes. The initial condition is

$$\rho(x,0) = 2+\sin(2\pi x), \quad u(x,0) = 1+0.1\sin(2\pi x), \quad p(x,0) = 1, \quad x\in[0,1]$$

with a periodic boundary condition. We take the numerical results by using the fifth-order Eulerian WENO scheme [13] with 8000 grids as the reference "exact" solution to compute the errors. To overcome the accuracy degeneracy phenomenon of the third-order ENO scheme, we have used the modified ENO procedure in [25], see [3] for more details. We summarize the errors and numerical rate of convergence of our second-order and third-order Lagrangian LW-ENO schemes with the Dukowicz flux at $t = 1$ in Tables 1 and 2. CPU time is also included in these tables. We always run our code several times and list the average CPU time when this time is less than 10s. For the purpose of comparison, the results for the Runge–Kutta time

Table 1
Second-order scheme. L_1 error. Runge–Kutta and Lax–Wendroff (boldface) time discretizations.

N_x	Time	Density	Order	Momentum	Order	Energy	Order	CPU time (s)
100	RK	1.64E−3	–	2.21E−3	–	4.27E−3	–	0.15
	LW	**9.93E−4**	–	**1.37E−3**	–	**2.66E−3**	–	**0.11**
200	RK	4.83E−4	1.76	6.09E−4	1.86	1.22E−3	1.80	0.36
	LW	**2.83E−4**	**1.81**	**3.65E−4**	**1.90**	**7.26E−4**	**1.87**	**0.22**
400	RK	1.27E−4	1.93	1.59E−4	1.94	3.18E−4	1.95	1.18
	LW	**7.49E−5**	**1.92**	**9.79E−5**	**1.90**	**1.92E−4**	**1.92**	**0.66**
800	RK	3.40E−5	1.90	4.24E−5	1.91	8.45E−5	1.91	4.38
	LW	**2.00E−5**	**1.91**	**2.54E−5**	**1.95**	**5.03E−5**	**1.93**	**2.36**
1600	RK	8.90E−6	1.94	1.10E−5	1.94	2.20E−5	1.94	17.3
	LW	**5.07E−6**	**1.98**	**6.40E−6**	**1.99**	**1.27E−5**	**1.98**	**9.1**
3200	RK	2.29E−6	1.96	2.80E−6	1.98	5.59E−6	1.98	69
	LW	**1.26E−6**	**2.01**	**1.59E−6**	**2.01**	**3.13E−6**	**2.02**	**36**

Table 2
Third-order scheme. L_1 error. Runge–Kutta and Lax–Wendroff (boldface) time discretizations.

N_x	Time	Density	Order	Momentum	Order	Energy	Order	CPU time (s)
100	RK	5.22E−5	–	6.95E−5	–	1.32E−4	–	0.48
	LW	**3.52E−5**	–	**4.99E−5**	–	**8.97E−5**	–	**0.22**
200	RK	6.59E−6	2.99	8.78E−6	2.98	1.66E−5	2.99	1.67
	LW	**4.45E−6**	**2.99**	**6.28E−6**	**2.99**	**1.13E−5**	**2.99**	**0.66**
400	RK	8.25E−7	3.00	1.10E−6	3.00	2.08E−6	3.00	6.40
	LW	**5.56E−7**	**3.00**	**7.84E−7**	**3.00**	**1.41E−6**	**3.00**	**2.51**
800	RK	1.03E−7	3.00	1.37E−7	3.00	2.61E−7	3.00	25.2
	LW	**6.95E−8**	**3.00**	**9.79E−8**	**3.00**	**1.77E−7**	**3.00**	**9.62**
1600	RK	1.29E−8	3.00	1.72E−8	3.00	3.26E−8	3.00	101
	LW	**8.69E−9**	**3.00**	**1.22E−8**	**3.00**	**2.21E−8**	**3.00**	**38.4**
3200	RK	1.61E−9	3.00	2.14E−9	3.01	4.06E−9	3.00	415
	LW	**1.08E−9**	**3.01**	**1.52E−9**	**3.01**	**2.75E−9**	**3.01**	**155**

discretization of the same Lagrangian ENO scheme [3] are also included. We can see that the Lax–Wendroff time discretization produces smaller errors and costs less CPU time.

2.4.2. Lax shock tube problem

This is a Riemann problem with the following initial condition

$$(\rho_L, u_L, p_L) = (0.445, 0.698, 3.528), \quad (\rho_R, u_R, p_R) = (0.5, 0, 0.571).$$

Fig. 1 shows the result using the Dukowicz flux with 100 initially uniform cells at $t = 1$. The left plot is the second-order result and the right one is the third-order result.

Comparing with the exact solution and the result by using Runge–Kutta time discretization, the Lax–Wendroff time discretization can produce the same satisfactory non-oscillatory results with high resolution.

2.4.3. Noh problem

In this problem [19], a shock is generated in a perfect ideal gas ($\gamma = 5/3$) by bringing the cold gas to rest at a rigid wall. The computational domain is $[0, 1]$ and the initial condition is $(\rho, u, e) = (1, -1, 0)$. We use an initially uniform mesh of 200 cells. Fig. 2 shows the result of the density at $t = 0.6$ using the Dukowicz flux. The left plot is the second-order result and the right one is the third-order result. The Lax–Wendroff time discretization produces similar results as the Runge–Kutta time discretization for this test case.

2.4.4. Two interacting blast waves

This is a test problem involving interactions of strong shock waves [29]. The initial condition consists of two discontinuities. The density and the velocity are unity everywhere and the pressure is large to the left and right and small in the middle:

$$\rho = 1, \quad u = 1, \quad p = \begin{cases} 10^3, & 0 < x < 0.1, \\ 10^{-2}, & 0.1 < x < 0.9, \\ 10^2, & 0.9 < x < 1.0. \end{cases}$$

A reflective boundary condition is applied at both $x = 0$ and $x = 1$. We use an initially uniform mesh with 400 cells to compute this problem. Fig. 3 is the density at time $t = 0.038$. The left plot is the second-order result and the right one is

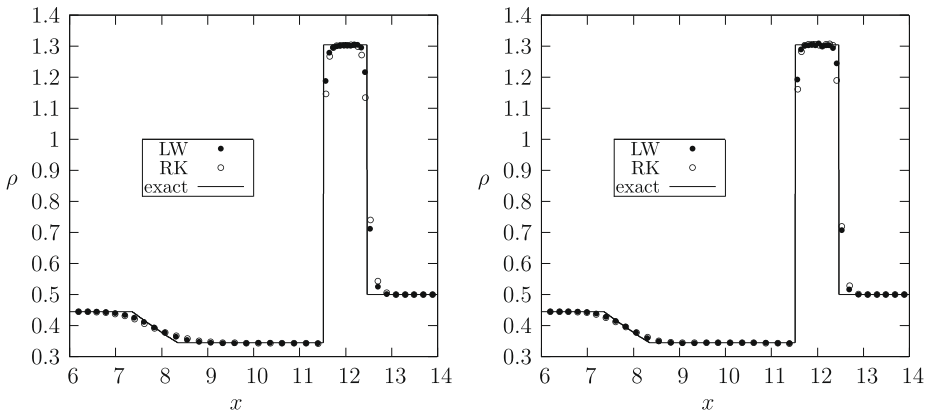


Fig. 1. The density of the Lax problem using the Dukowicz flux with 100 cells at $t = 1$. Left: second-order schemes; right: third-order schemes.

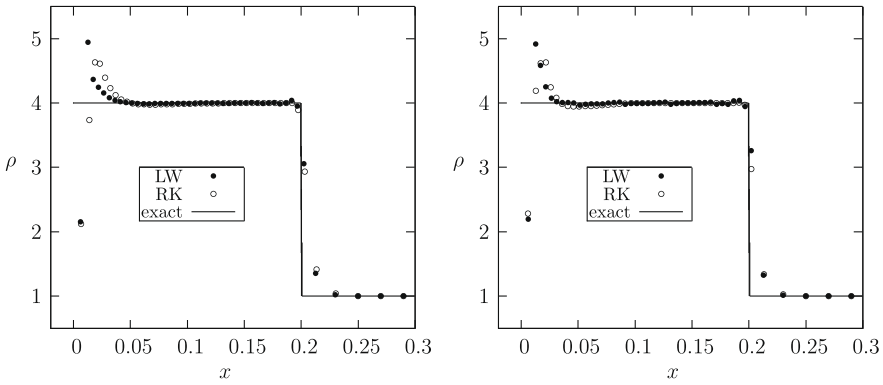


Fig. 2. The density of Noh problem using the Dukowicz flux with 200 cells at $t = 0.6$. Left: second-order schemes; right: third-order schemes.

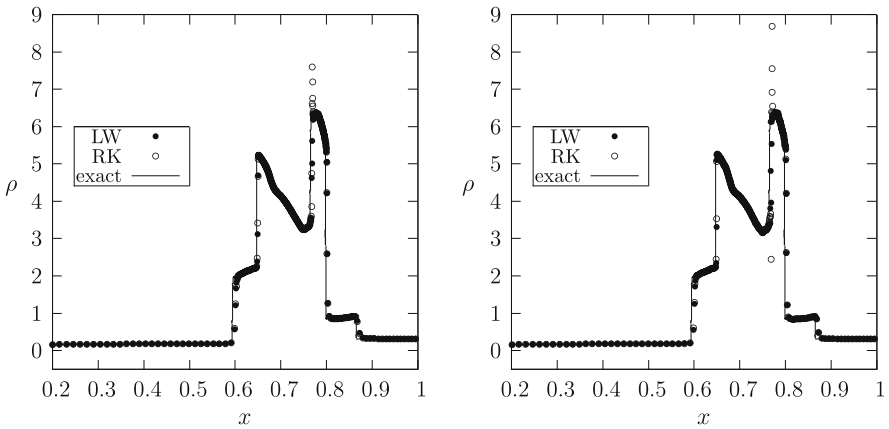


Fig. 3. The density of the blast waves problem using the HLLC flux with 400 cells at $t = 0.038$. Left: second-order schemes; right: third-order schemes.

the third-order result. For this problem, it seems that the Lax–Wendroff time discretization does not produce any spurious overshoots near the contact discontinuity, which is better than the performance of the Runge–Kutta time discretization.

2.4.5. Leblanc shock tube problem

This is a difficult test case with a very strong shock. The initial condition is as follows:

$$(\rho, u, e) = (1, 0, 0.1), \quad 0 \leqslant x < 3,$$
$$(\rho, u, e) = (0.001, 0, 10^{-7}), \quad 3 < x \leqslant 9.$$

The computational domain is $[0, 9]$ filled with an ideal gas with $\gamma = 5/3$. We use 1000 initially uniform cells to compute this problem to $t = 6$. Fig. 4 is the second-order result and Fig. 5 is the third-order result. By comparing with the exact solution, we can see that the positions of the contact discontinuity and the shock can be maintained better when high order schemes are used. Notice that in this and a few other numerical examples, there are noticeable spurious oscillations in the numerical results which are not typical for high order non-oscillatory schemes in Eulerian coordinates. This is one of the disadvantages of Lagrangian methods, which already appears for first-order schemes. In fact, higher order non-oscillatory Lagrangian methods often have smaller spurious oscillations than the first-order methods, see for example [3] for more details. The Lax–Wendroff time discretization performs similarly (slightly better) as the Runge–Kutta time discretization for this case.

2.4.6. Shock entropy wave interactions

This problem [27] contains both shocks and complex smooth region structures. The computational domain is $[-10, 5]$ and the initial condition is

$$(\rho, u, e) = (3.85714, 2.629369, 10.33333), \quad x < -4,$$
$$(\rho, u, e) = (1 + \epsilon \sin(kx), 0, 1), \quad x \geqslant -4,$$

where ϵ and k are the amplitude and wave number of the entropy wave. We take $\epsilon = 0.2$ and $k = 5$ in our test and the final time is $t = 1.8$. Figs. 6 and 7 are the second and third-order results using 200 cells with the Dukowicz flux and the HLLC flux

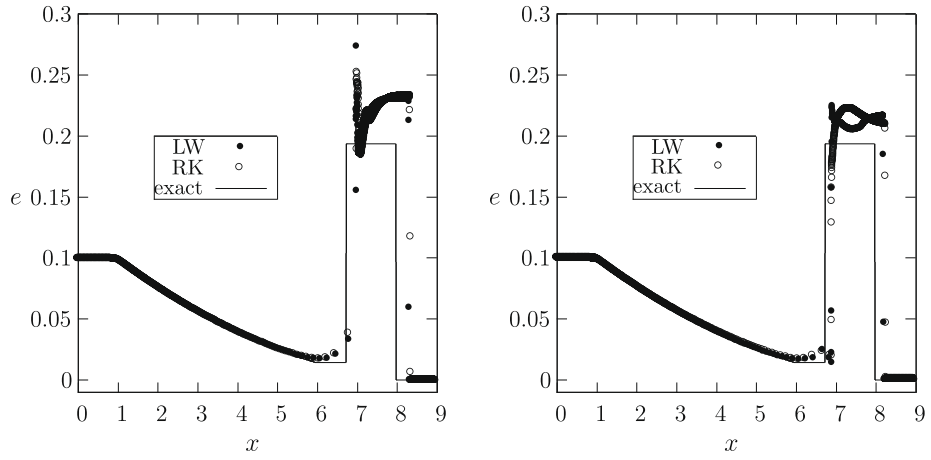


Fig. 4. Second-order results: internal energy of the Leblanc problem using 1000 cells at $t = 6$. Left: with the Dukowicz flux; right: with the HLLC flux.

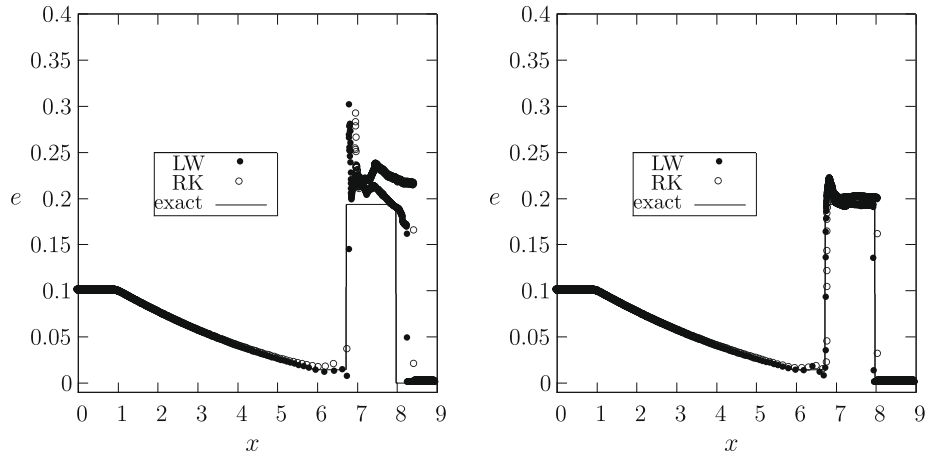


Fig. 5. Third-order results: internal energy of the Leblanc problem using 1000 cells at $t = 6$. Left: with the Dukowicz flux; right: with the HLLC flux.

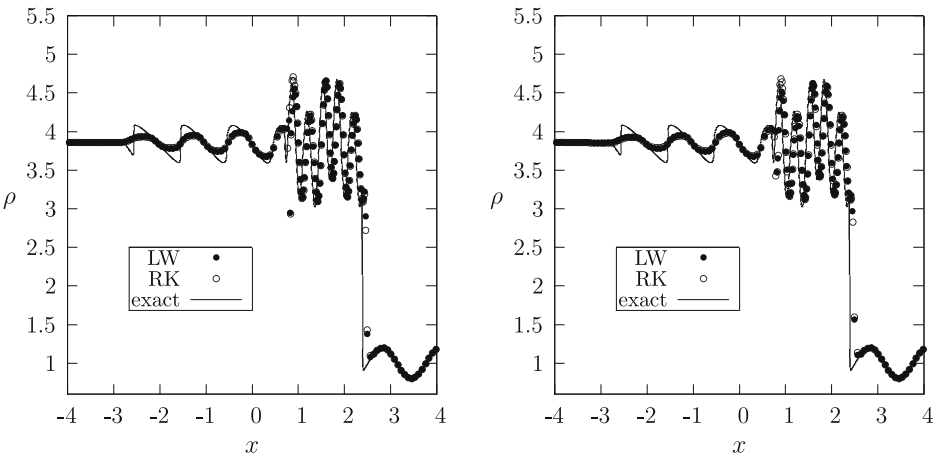


Fig. 6. Second-order results: density of the shock entropy wave interactions problem using 200 cells at $t = 1.8$. Left: with the Dukowicz flux; right: with the HLLC flux.

respectively. We can see clearly that for this example the third-order scheme has much better resolution than the second-order one on the same mesh and the Lax–Wendroff time discretization produces less spurious overshoots than the Runge–Kutta time discretization.

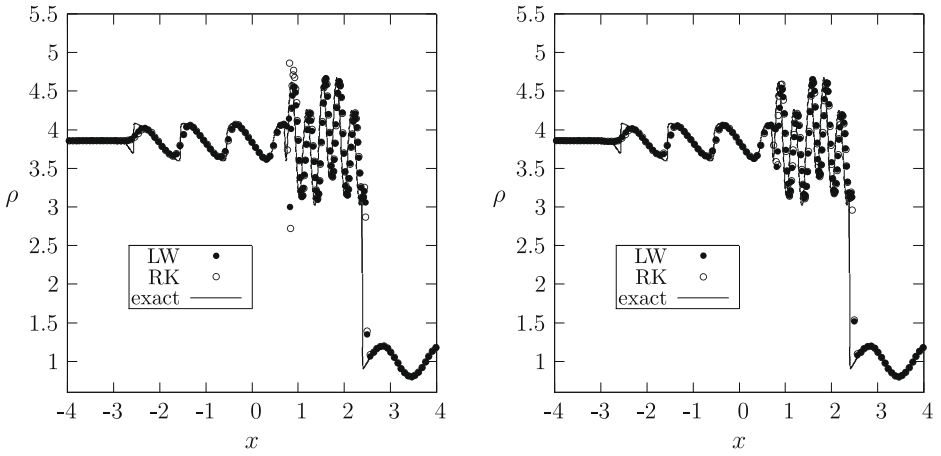


Fig. 7. Third-order results: density of the shock entropy wave interactions problem using 200 cells at $t = 1.8$. Left: with the Dukowicz flux; right: with the HLLC flux.

3. Two-dimensional case

3.1. Construction of the scheme in the 2D case

In the 2D case, we develop our scheme on both quadrilateral and curvilinear (curved quadrilateral) meshes. In [5], it is demonstrated that, because of the error from the quadrilateral mesh with straight-line edges, Lagrangian methods on quadrilateral meshes are at most second-order accurate. In order to obtain a third-order accurate method, we must consider curvilinear meshes. The 2D spatial domain Ω_t is discretized into $N_x \times N_y$ computational cells. $I_{ij}(t)$ is the cell identified by its four vertices $(x_{i-1/2,j-1/2}(t), y_{i-1/2,j-1/2}(t))$, $(x_{i+1/2,j-1/2}(t), y_{i+1/2,j-1/2}(t))$, $(x_{i-1/2,j+1/2}(t), y_{i-1/2,j+1/2}(t))$ and $(x_{i+1/2,j+1/2}(t), y_{i+1/2,j+1/2}(t))$. For the curved quadrilateral meshes with quadratic curved boundaries, the information of the four middle points along each side of the curved quadrilateral is also needed to identify the cell. We use h_{ij} to denote the mesh size (the diameter of the smallest circle containing the cell I_{ij}) and $h = \max_{i,j} h_{ij}$. $s_{ij}(t)$ denotes the area of the cell $I_{ij}(t)$. The fluid velocity $(u_{i\pm1/2,j\pm1/2}, v_{i\pm1/2,j\pm1/2})$ is defined at the vertex of the mesh. For the curved quadrilateral meshes, the fluid velocity is also needed at the center along each side of the curved quadrilateral cell. Since we are using a non-staggered mesh, the conservative variables are stored in the form of cell averages. The values of the cell averages of density $\bar{\rho}_{ij}(t)$, x-momentum $\overline{M}_{ij}(t)$, y-momentum $\overline{N}_{ij}(t)$ and total energy $\overline{E}_{ij}(t)$ are defined as follows:

$$\bar{\rho}_{ij}(t) = \frac{1}{s_{ij}(t)} \int\int_{I_{ij}(t)} \rho(x(t), y(t), t) dxdy,$$

$$\overline{M}_{ij}(t) = \frac{1}{s_{ij}(t)} \int\int_{I_{ij}(t)} M(x(t), y(t), t) dxdy,$$

$$\overline{N}_{ij}(t) = \frac{1}{s_{ij}(t)} \int\int_{I_{ij}(t)} N(x(t), y(t), t) dxdy,$$

$$\overline{E}_{ij}(t) = \frac{1}{s_{ij}(t)} \int\int_{I_{ij}(t)} E(x(t), y(t), t) dxdy.$$

The construction of the scheme in 2D is similar to the 1D case. The main difference is that, in the 2D case, we have to approximate the multiple integral both in time and along the edge of the cell to obtain the numerical flux.

As in the 1D case, after using the equation on $I_{ij}(t)$ and integrating over $[t_n, t_{n+1}]$, we obtain

$$x^{n+1}_{i\pm1/2,j\pm1/2} = x^n_{i\pm1/2,j\pm1/2} + \int_{t_n}^{t_{n+1}} u(x_{i\pm1/2}(t), y_{j\pm1/2}(t), t) dt, \tag{26}$$

$$y^{n+1}_{i\pm1/2,j\pm1/2} = y^n_{i\pm1/2,j\pm1/2} + \int_{t_n}^{t_{n+1}} v(x_{i\pm1/2}(t), y_{j\pm1/2}(t), t) dt, \tag{27}$$

$$\begin{pmatrix} \bar{\rho}^{n+1}_{ij} s^{n+1}_{ij} \\ \overline{M}^{n+1}_{ij} s^{n+1}_{ij} \\ \overline{N}^{n+1}_{ij} s^{n+1}_{ij} \\ \overline{E}^{n+1}_{ij} s^{n+1}_{ij} \end{pmatrix} = \begin{pmatrix} \bar{\rho}^{n}_{ij} s^{n}_{ij} \\ \overline{M}^{n}_{ij} s^{n}_{ij} \\ \overline{N}^{n}_{ij} s^{n}_{ij} \\ \overline{E}^{n}_{ij} s^{n}_{ij} \end{pmatrix} - \int_{t_n}^{t_{n+1}} \int_{\partial I_{ij}} \begin{pmatrix} f_D(\mathbf{U}(x(t), y(t), t)) \\ f_M(\mathbf{U}(x(t), y(t), t)) \\ f_N(\mathbf{U}(x(t), y(t), t)) \\ f_E(\mathbf{U}(x(t), y(t), t)) \end{pmatrix} dl dt. \tag{28}$$

We would like to obtain a fully discrete scheme from the above equations. First we have to use a numerical flux to replace the physical flux on every edge of the mesh and use a numerical velocity to replace the physical velocity at each vertex of the mesh and the middle point along each side of the curved quadrilateral. Second, an approximation to the integral should be chosen. If we want to get a rth order accurate scheme, then the numerical flux should approximate the physical flux to at least rth order accuracy; the numerical integration to $\int_{\partial I_{ij}}$ should be at least rth order accurate; and the numerical integration to $\int_{t_n}^{t_{n+1}}$ should be at least $(r+1)$th order accurate. We use S to denote the edges of the cell $I_{i,j}$, l_s to denote the length of the sth edge, Q to denote the Gaussian quadrature points on every edge, ω_q to denote the corresponding quadrature coefficients, K to denote the Gaussian quadrature points in time interval $[t^n, t^{n+1}]$ and α_k to denote the corresponding quadrature coefficients. We would require

$$\begin{aligned}
&\hat{f}(\mathbf{U}^-(x(t),y(t),t),\mathbf{U}^+(x(t),y(t),t)) = f(\mathbf{U}(x(t),y(t),t)) + \mathcal{O}(h^r),\\
&\hat{u}(u^-(x(t),y(t),t),u^+(x(t),y(t),t)) = u(x(t),y(t),t) + \mathcal{O}(h^r),\\
&\hat{v}(v^-(x(t),y(t),t),v^+(x(t),y(t),t)) = v(x(t),y(t),t) + \mathcal{O}(h^r),
\end{aligned} \tag{29}$$

$$\int_{\partial I_{ij}} \hat{f}(\mathbf{U}^-(x(t),y(t),t),\mathbf{U}^+(x(t),y(t),t))dl = \sum_{s=1}^{S}\sum_{q=1}^{Q} \omega_q \hat{f}(\mathbf{U}_q^-(t),\mathbf{U}_q^+(t))l_s + \mathcal{O}(h^r), \tag{30}$$

$$\begin{aligned}
&\int_{t_n}^{t_{n+1}} \hat{f}(\mathbf{U}_q^-(t),\mathbf{U}_q^+(t))dt = \Delta t\sum_{k=0}^{K}\alpha_k \hat{f}(t_n+\beta_k\Delta t) + \mathcal{O}(\Delta t^{r+1}),\\
&\int_{t_n}^{t_{n+1}} \hat{u}_q(t)dt = \Delta t\sum_{k=0}^{K}\alpha_k \hat{u}_q(t_n+\beta_k\Delta t) + \mathcal{O}(\Delta t^{r+1}),\\
&\int_{t_n}^{t_{n+1}} \hat{v}_q(t)dt = \Delta t\sum_{k=0}^{K}\alpha_k \hat{v}_q(t_n+\beta_k\Delta t) + \mathcal{O}(\Delta t^{r+1}).
\end{aligned} \tag{31}$$

Similar to the 1D case, if we use a consistent and Lipschitz continuous numerical flux, combined with accurate numerical integration, we only need to find a reconstruction of $\mathbf{U}$ which is at least rth order accurate on the Gaussian quadrature points for a rth order accurate scheme

$$\begin{aligned}
&\mathbf{U}_q^{\pm}(t_n+\beta_k\Delta t) = \mathbf{U}_q(t_n+\beta_k\Delta t) + \mathcal{O}(h^r),\\
&u_q^{\pm}(t_n+\beta_k\Delta t) = u_q(t_n+\beta_k\Delta t) + \mathcal{O}(h^r),\\
&v_q^{\pm}(t_n+\beta_k\Delta t) = v_q(t_n+\beta_k\Delta t) + \mathcal{O}(h^r).
\end{aligned} \tag{32}$$

Since we only have cell average variables at t^n, we would need to use a Taylor expansion to get the value at the Gaussian quadrature points

$$\begin{aligned}
&\mathbf{U}_q^{\pm}(t_n+\beta_k\Delta t) = \mathbf{U}_q^{n\pm} + \sum_{l=1}^{r-1}\frac{(\beta_k\Delta t_n)^l}{l!}\frac{d^l\mathbf{U}_q^{\pm}}{dt^l} + \mathcal{O}(\Delta t^r),\\
&u_q^{\pm}(t_n+\beta_k\Delta t) = u_q^{n\pm} + \sum_{l=1}^{r-1}\frac{(\beta_k\Delta t_n)^l}{l!}\frac{d^l u_q^{\pm}}{dt^l} + \mathcal{O}(\Delta t^r),\\
&v_q^{\pm}(t_n+\beta_k\Delta t) = v_q^{n\pm} + \sum_{l=1}^{r-1}\frac{(\beta_k\Delta t_n)^l}{l!}\frac{d^l v_q^{\pm}}{dt^l} + \mathcal{O}(\Delta t^r).
\end{aligned} \tag{33}$$

The Taylor expansion tells us that in order to get a rth order accurate scheme, we should find a reconstruction to satisfy the following equations

$$\begin{aligned}
&\mathbf{U}_q^{n\pm} = \mathbf{U}_q^n + \mathcal{O}(h^r),\\
&u_q^{n\pm} = u_q^n + \mathcal{O}(h^r),\\
&v_q^{n\pm} = v_q^n + \mathcal{O}(h^r),
\end{aligned} \tag{34a}$$

$$\begin{aligned}
&\frac{d^l\mathbf{U}_q^{n\pm}}{dt^l} = \frac{d^l\mathbf{U}_q^n}{dt^l} + \mathcal{O}(\Delta t^{r-l}),\\
&\frac{d^l u_q^{n\pm}}{dt^l} = \frac{d^l u_q^n}{dt^l} + \mathcal{O}(\Delta t^{r-l}),\\
&\frac{d^l v_q^{n\pm}}{dt^l} = \frac{d^l v_q^n}{dt^l} + \mathcal{O}(\Delta t^{r-l}), \quad l = 1,2,\ldots,r-1.
\end{aligned} \tag{34b}$$

As before, using the Lax–Wendroff procedure, (34a) and (34b) can be guaranteed if we have

$$
\begin{aligned}
&\frac{\partial^l \mathbf{U}_q^{n\pm}}{\partial x^l} = \frac{\partial^l \mathbf{U}_q^{n}}{\partial x^l} + \mathcal{O}(h^{r-l}),\\
&\frac{\partial^l u_q^{n\pm}}{\partial x^l} = \frac{\partial^l u_q^{n}}{\partial x^l} + \mathcal{O}(h^{r-l}),\\
&\frac{\partial^l v_q^{n\pm}}{\partial x^l} = \frac{\partial^l v_q^{n}}{\partial x^l} + \mathcal{O}(h^{r-l}), \quad l = 0,1,2,\ldots,r-1
\end{aligned}
\tag{35a}
$$

$$
\begin{aligned}
&\frac{\partial^l \mathbf{U}_q^{n\pm}}{\partial y^l} = \frac{\partial^l \mathbf{U}_q^{n}}{\partial y^l} + \mathcal{O}(h^{r-l}),\\
&\frac{\partial^l u_q^{n\pm}}{\partial y^l} = \frac{\partial^l u_q^{n}}{\partial y^l} + \mathcal{O}(h^{r-l}),\\
&\frac{\partial^l v_q^{n\pm}}{\partial y^l} = \frac{\partial^l v_q^{n}}{\partial y^l} + \mathcal{O}(h^{r-l}), \quad l = 0,1,2,\ldots,r-1.
\end{aligned}
\tag{35b}
$$

Again, this is automatically achieved by a rth order ENO reconstruction. Therefore, we can use an ENO reconstruction to get our second-order scheme on quadrilateral meshes and third-order scheme on curvilinear meshes.

Remark 1. The time step Δt^n is chosen as follows

$$\Delta t^n = \lambda \min_{i=1,\ldots,N_x\, j=1,\ldots,N_y} \left(\Delta l_{i,j}^n / c_{i,j}^n\right),$$

where $\Delta l_{i,j}^n$ is the length of the shortest edge of the cell $I_{i,j}$ and $c_{i,j}^n$ is the sound speed within the cell. The Courant number λ is set to be 0.5 in the following numerical tests unless otherwise stated.

Remark 2. In the third-order case, we also use the WENO reconstruction in some test cases because it is more robust than the ENO procedure. The details of the choice for the weights can be found in [3,5].

Remark 3. After the reconstruction, we obtain two approximation values at each Gaussian quadrature points on the edge of the mesh. Similar to the 1D case, A consistent Lipschitz continuous numerical flux is then used. Again, we will use the following four typical numerical fluxes: (1) the Godunov flux; (2) the Dukowicz flux; (3) the Lax–Friedrichs (L–F) flux; (4) the HLLC flux, although we will only show the results obtained with the Dukowicz flux and the HLLC flux to save space. When approximating the integral on each cell edge, we use a trapezoid rule for the second-order case and a four-point Gauss–Lobatto quadrature in the third-order case.

Remark 4. Vertex velocity (and, for the third-order case, also cell edge center velocity) is determined in a similar fashion as in the 1D case. The tangential velocity of the vertex (or edge center) along the edge is defined as a simple average of that in both sides. The normal velocity is defined as the Roe average for the L–F flux and the HLLC flux, and for the Godunov flux and the Dukowicz flux, it is obtained by the value of velocity through the Riemann solver solved along the normal direction of the edge.

3.2. Algorithm description

We now describe our algorithm in detail. Starting from the mesh and the approximate cell averages of the solution at time level t^n, we will proceed as follows.

1. At each Gaussian quadrature point on the edge of the cell, use a rth characteristic-wise ENO (or WENO) reconstruction to get the value of $\mathbf{U}_q^{n\pm}$ and $\frac{\partial^l \mathbf{U}_q^{n\pm}}{\partial x^l}$ for $l = 1,2,\ldots,r-1$, from the cell averages $\overline{\mathbf{U}}_{ij}^n$.
2. Calculate the derivatives at the spatial Gaussian quadrature points by using the information from the reconstruction polynomial. First, compute $u, v, p, u_x, u_y, v_x, v_y, p_x, p_y$ by using $\mathbf{U}_q^{n\pm}$, $\frac{\partial \mathbf{U}_q^{n\pm}}{\partial x}$ and $\frac{\partial \mathbf{U}_q^{n\pm}}{\partial y}$

$$
\begin{aligned}
&u = M/\rho,\\
&v = N/\rho,\\
&p = (\gamma-1)(E - 0.5\rho u^2 - 0.5\rho v^2),\\
&u_x = (M_x - u\rho_x)/\rho,\\
&u_y = (M_y - u\rho_y)/\rho,\\
&v_x = (N_x - v\rho_x)/\rho,\\
&v_y = (N_y - v\rho_y)/\rho,\\
&p_x = (\gamma-1)(E_x - 0.5u_xM - 0.5uM_x - 0.5v_xN - 0.5vN_x),\\
&p_y = (\gamma-1)(E_y - 0.5u_yM - 0.5uM_y - 0.5v_yN - 0.5vN_y).
\end{aligned}
\tag{36}
$$

Then, compute $\rho_t, M_t, N_t, u_t, v_t, E_t, p_t$

$$\begin{aligned}
&\rho_t = -M_x - N_y,\\
&M_t = -u_x M - uM_x - v_y M - vM_y - p_x,\\
&N_t = -u_x N - uN_x - v_y N - vN_y - p_y,\\
&u_t = (M_t - u\rho_t)/\rho,\\
&v_t = (N_t - v\rho_t)/\rho,\\
&E_t = -E_x u - Eu_x - p_x u - pu_x - E_y v - Ev_y - p_y v - pv_y,\\
&p_t = -up_x - vp_y - \gamma p(u_x + v_y).
\end{aligned} \tag{37}$$

In order to compute $\rho_{tt}, M_{tt}, E_{tt}$, first compute $u_{xx}, u_{yy}, u_{xy}, v_{xx}, v_{yy}, v_{xy}, p_{xx}, p_{yy}, p_{xy}, \rho_{xt}, \rho_{yt}, M_{xt}, M_{yt}, N_{xt}, N_{yt}, E_{xt}, E_{yt}, u_{xt}, u_{yt}, v_{xt}, v_{yt}, p_{xt}, p_{yt}$

$$\begin{aligned}
&u_{xx} = (M_{xx} - u\rho_{xx} - 2u_x\rho_x)/\rho,\\
&u_{yy} = (M_{yy} - u\rho_{yy} - 2u_y\rho_y)/\rho,\\
&u_{xy} = (M_{xy} - u\rho_{xy} - u_y\rho_x - u_x\rho_y)/\rho,\\
&v_{xx} = (N_{xx} - v\rho_{xx} - 2v_x\rho_x)/\rho,\\
&v_{yy} = (N_{yy} - v\rho_{yy} - 2v_y\rho_y)/\rho,\\
&v_{xy} = (N_{xy} - v\rho_{xy} - v_y\rho_x - v_x\rho_y)/\rho,\\
&p_{xx} = (\gamma-1)(E_{xx} - 0.5(Mu_{xx} + 2u_xM_x + uM_{xx}) - 0.5(Nv_{xx} + 2v_xN_x + vN_{xx})),\\
&p_{yy} = (\gamma-1)\big(E_{yy} - 0.5(Mu_{yy} + 2u_yM_y + uM_{yy}) - 0.5(Nv_{yy} + 2v_yN_y + vN_{yy})\big),\\
&p_{xy} = (\gamma-1)(E_{xy} - 0.5(Mu_{xy} + u_xM_y + u_yM_x + uM_{xy}) - 0.5(Nv_{xy} + v_xN_y + v_yN_x + vN_{xy})),\\
&\rho_{xt} = -M_{xx} - N_{xy},\\
&\rho_{yt} = -M_{xy} - N_{yy},\\
&M_{xt} = -2u_xM_x - uM_{xx} - Mu_{xx} - v_xM_y - vM_{xy} - v_yM_x - v_{xy}M - p_{xx},\\
&M_{yt} = -2v_yM_y - vM_{yy} - Mv_{yy} - u_xM_y - uM_{xy} - u_yM_x - u_{xy}M - p_{xy},\\
&N_{xt} = -2u_xN_x - uN_{xx} - Nu_{xx} - v_xN_y - vN_{xy} - v_yN_x - v_{xy}N - p_{xy},\\
&N_{yt} = -2v_yN_y - vN_{yy} - Nv_{yy} - u_xN_y - uN_{xy} - u_yN_x - u_{xy}N - p_{yy},\\
&E_{xt} = -2u_xE_x - uE_{xx} - Eu_{xx} - 2p_xu_x - pu_{xx} - up_{xx} - v_xE_y - vE_{xy} - v_yE_x - v_{xy}E - p_{xy}v - p_yv_x - p_xv_y - pv_{xy},\\
&E_{yt} = -2v_yE_y - vE_{yy} - Ev_{yy} - 2p_yv_y - pv_{yy} - vp_{yy} - u_yE_x - uE_{xy} - u_xE_y - u_{xy}E - p_{xy}u - p_xu_y - p_yu_x - pu_{xy},\\
&u_{xt} = (M_{xt} - u\rho_{xt} - u_t\rho_x - u_x\rho_t)/\rho,\\
&u_{yt} = (M_{yt} - u\rho_{yt} - u_t\rho_y - u_y\rho_t)/\rho,\\
&v_{xt} = (N_{xt} - v\rho_{xt} - v_t\rho_x - v_x\rho_t)/\rho,\\
&v_{yt} = (N_{yt} - v\rho_{yt} - v_t\rho_y - v_y\rho_t)/\rho,\\
&p_{xt} = -up_{xx} - \gamma pu_{xx} - (1+\gamma)u_xp_x - v_xp_y - vp_{xy} - \gamma p_xv_y - \gamma pv_{xy},\\
&p_{yt} = -vp_{yy} - \gamma pv_{yy} - (1+\gamma)v_yp_y - u_yp_x - up_{xy} - \gamma p_yu_x - \gamma pu_{xy}.
\end{aligned} \tag{38}$$

Then we can compute $\rho_{tt}, M_{tt}, E_{tt}, p_{tt}, u_{tt}$

$$\begin{aligned}
&\rho_{tt} = -M_{xt} - N_{yt},\\
&M_{tt} = -uM_{xt} - Mu_{xt} - p_{xt} - u_xM_t - u_tM_x - Mv_{yt} - vM_{yt} - v_yM_t - v_tM_y,\\
&N_{tt} = -uN_{xt} - Nu_{xt} - p_{yt} - u_xN_t - u_tN_x - Nv_{yt} - vN_{yt} - v_yN_t - v_tN_y,\\
&E_{tt} = -uE_{xt} - Eu_{xt} - pu_{xt} - up_{xt} - u_xE_t - u_tE_x - p_xu_t - p_tu_x - vE_{yt} - Ev_{yt} - pv_{yt} - vp_{yt} - v_yE_t - v_tE_y - p_yv_t - p_tv_y,\\
&p_{tt} = -up_{xt} - \gamma pu_{xt} - u_tp_x - \gamma p_tu_x - vp_{yt} - \gamma pv_{yt} - v_tp_y - \gamma p_tv_y,\\
&u_{tt} = (M_{tt} - 2\rho_tu_t - u\rho_{tt})/\rho,\\
&v_{tt} = (N_{tt} - 2\rho_tv_t - v\rho_{tt})/\rho.
\end{aligned} \tag{39}$$

3. Compute $\frac{d^l\mathbf{U}_q^{n\pm}}{dt^l}$, $\frac{d^lu_q^{n\pm}}{dt^l}$ and $\frac{d^lv_q^{n\pm}}{dt^l}$ for $l = 1, 2, \ldots, r-1$

$$
\begin{aligned}
&\frac{d\mathbf{U}_q^{n\pm}}{dt} = \frac{\partial \mathbf{U}_q^{n\pm}}{\partial t} + u_q^{n\pm}\frac{\partial \mathbf{U}_q^{n\pm}}{\partial x} + v_q^{n\pm}\frac{\partial \mathbf{U}_q^{n\pm}}{\partial y},\\
&\frac{du_q^{n\pm}}{dt} = \frac{\partial u_q^{n\pm}}{\partial t} + u_q^{n\pm}\frac{\partial u_q^{n\pm}}{\partial x} + v_q^{n\pm}\frac{\partial u_q^{n\pm}}{\partial y},\\
&\frac{dv_q^{n\pm}}{dt} = \frac{\partial v_q^{n\pm}}{\partial t} + u_q^{n\pm}\frac{\partial v_q^{n\pm}}{\partial x} + v_q^{n\pm}\frac{\partial v_q^{n\pm}}{\partial y},\\
&\frac{d^2\mathbf{U}_q^{n\pm}}{dt^2} = \frac{\partial}{\partial t}\left(\frac{d\mathbf{U}_q^{n\pm}}{dt}\right) + u_q^{n\pm}\frac{\partial}{\partial x}\left(\frac{d\mathbf{U}_q^{n\pm}}{dt}\right) + v_q^{n\pm}\frac{\partial}{\partial y}\left(\frac{d\mathbf{U}_q^{n\pm}}{dt}\right),\\
&\frac{d^2u_q^{n\pm}}{dt^2} = \frac{\partial}{\partial t}\left(\frac{du_q^{n\pm}}{dt}\right) + u_q^{n\pm}\frac{\partial}{\partial x}\left(\frac{du_q^{n\pm}}{dt}\right) + v_q^{n\pm}\frac{\partial}{\partial y}\left(\frac{du_q^{n\pm}}{dt}\right),\\
&\frac{d^2v_q^{n\pm}}{dt^2} = \frac{\partial}{\partial t}\left(\frac{dv_q^{n\pm}}{dt}\right) + u_q^{n\pm}\frac{\partial}{\partial x}\left(\frac{dv_q^{n\pm}}{dt}\right) + v_q^{n\pm}\frac{\partial}{\partial y}\left(\frac{dv_q^{n\pm}}{dt}\right).
\end{aligned}
\tag{40}
$$

4. Compute the values of the solution at the Gaussian quadrature points $\mathbf{U}_{qk}^{\pm} = \mathbf{U}_q^{\pm}(t_n + \beta_k\Delta t)$ and $u_{qk}^{\pm} = u_q^{\pm}(t_n + \beta_k\Delta t)$ for $k = 1, 2, \ldots, K$

$$
\begin{aligned}
&\mathbf{U}_{qk}^{\pm} = \mathbf{U}_q^{n\pm} + \sum_{l=1}^{r-1}\frac{(\beta_k\Delta t_n)^l}{l!}\frac{d^l\mathbf{U}_q^{\pm}}{dt^l},\\
&u_{qk}^{\pm} = u_q^{n\pm} + \sum_{l=1}^{r-1}\frac{(\beta_k\Delta t_n)^l}{l!}\frac{d^lu_q^{\pm}}{dt^l},\\
&v_{qk}^{\pm} = v_q^{n\pm} + \sum_{l=1}^{r-1}\frac{(\beta_k\Delta t_n)^l}{l!}\frac{d^lv_q^{\pm}}{dt^l}.
\end{aligned}
\tag{41}
$$

5. We use four different numerical fluxes in our code: Godunov flux, Dukowicz flux, L–F flux, HLLC flux. Compute the numerical flux $\hat{f}_{qk}\left(\mathbf{U}_{qk}^{-}(t_n + \beta_k\Delta t), \mathbf{U}_{qk}^{+}(t_n + \beta_k\Delta t)\right)$ and the velocity $\hat{u}_{qk}\left(\mathbf{U}_{qk}^{-}(t_n + \beta_k\Delta t), \mathbf{U}_{qk}^{+}(t_n + \beta_k\Delta t)\right)$, for $k = 1, 2, \ldots, K$.
6. Compute the time integral of the flux $\bar{f}_{\partial l_{ij}} = \frac{1}{\Delta t_n}\int_{t_n}^{t_{n+1}}\int_{\partial l_{ij}}\hat{f}(\mathbf{U}^{-}(x(t), y(t), t), \mathbf{U}^{+}(x(t), y(t), t))dldt$ and of the velocity $\overline{u}_q = \frac{1}{\Delta t_n}\int_{t_n}^{t_{n+1}}\hat{u}_q dt$ by

$$
\bar{f}_{\partial l_{ij}} = \sum_{k=0}^{K}\alpha_k\left(\sum_{s=1}^{S}\left(\sum_{q=1}^{Q}\omega_q\hat{f}\left(\mathbf{U}_{qk}^{-}, \mathbf{U}_{qk}^{+}\right)\right)l_s\right),
\tag{42}
$$

$$
\overline{u}_q = \sum_{k=0}^{K}\alpha_k\hat{u}_q\left(\mathbf{U}_{qk}^{-}, \mathbf{U}_{qk}^{+}\right).
\tag{43}
$$

7. Compute $x_{i\pm1/2,j\pm1/2}^{n+1}, y_{i\pm1/2,j\pm1/2}^{n+1}$ and $\overline{\mathbf{U}}_{ij}^{n+1}$ at t_{n+1} (for the third-order Lagrangian method on curvilinear meshes, also compute the coordinates of the four middle points $(x_{i,j\pm1/2}^{n+1},\ y_{i,j\pm1/2}^{n+1}), (x_{i\pm1/2,j}^{n+1},\ y_{i\pm1/2,j}^{n+1})$ along each side of the cell)

$$
\begin{aligned}
&x_{i\pm1/2,j\pm1/2}^{n+1} = x_{i\pm1/2,j\pm1/2}^{n} + \Delta t_n\overline{u}_{i\pm1/2,j\pm1/2},\\
&y_{i\pm1/2,j\pm1/2}^{n+1} = y_{i\pm1/2,j\pm1/2}^{n} + \Delta t_n\overline{v}_{i\pm1/2,j\pm1/2},\\
&\bar{\rho}_{ij}^{n+1}s_{ij}^{n+1} = \bar{\rho}_{ij}^{n}s_{ij}^{n} - \Delta t_n\bar{f}_{\partial l_{ij}}^{D},\\
&\overline{M}_{ij}^{n+1}s_{ij}^{n+1} = \overline{M}_{ij}^{n}s_{ij}^{n} - \Delta t_n\bar{f}_{\partial l_{ij}}^{M},\\
&\overline{N}_{ij}^{n+1}s_{ij}^{n+1} = \overline{N}_{ij}^{n}s_{ij}^{n} - \Delta t_n\bar{f}_{\partial l_{ij}}^{N},\\
&\overline{E}_{ij}^{n+1}s_{ij}^{n+1} = \overline{E}_{ij}^{n}s_{ij}^{n} - \Delta t_n\bar{f}_{\partial l_{ij}}^{E}\\
&\Big(x_{i,j\pm1/2}^{n+1} = x_{i,j\pm1/2}^{n} + \Delta t_n\overline{u}_{i,j\pm1/2},\\
&y_{i,j\pm1/2}^{n+1} = y_{i,j\pm1/2}^{n} + \Delta t_n\overline{v}_{i,j\pm1/2},\\
&x_{i\pm1/2,j}^{n+1} = x_{i\pm1/2,j}^{n} + \Delta t_n\overline{u}_{i\pm1/2,j},\\
&y_{i\pm1/2,j}^{n+1} = y_{i\pm1/2,j}^{n} + \Delta t_n\overline{v}_{i\pm1/2,j}\Big).
\end{aligned}
\tag{44}
$$

3.3. Numerical examples in 2D

3.3.1. Accuracy test

This is a two-dimensional vortex evolution problem with a diagonal flow. The mean flow is $\rho = p = u = v = 1$, and we add to this mean flow an isentropic vortex perturbations in (u, v) and the temperature $T = p/\rho$, no perturbation in the entropy $S = p/\rho^{\gamma}$

$$(\delta u, \delta v) = \frac{\epsilon}{2\pi} e^{0.5(1-r^2)}(-\bar{y}, \bar{x}), \quad \delta T = -\frac{(\gamma-1)\epsilon^2}{8\gamma\pi^2} e^{(1-r^2)}, \quad \delta S = 0,$$

where $(-\bar{y}, \bar{x}) = (y-5, x-5)$, $r^2 = \bar{x}^2 + \bar{y}^2$, and the vortex strength is $\epsilon = 5$. The computational domain is $[0,10] \times [0,10]$ and the boundary condition is periodic.

Tables 3–5 contain the results of our Lax–Wendroff time discretization scheme comparing with the Runge–Kutta time discretization scheme by using the Dukowicz flux. Table 3 is for the L_1 errors of the second-order schemes; Table 4 is for the third-order schemes on quadrilateral meshes with straight-line edges, and Table 5 is for the third-order schemes on curvilinear meshes. We can see that, as in [3,5], quadrilateral meshes with straight-line edges can only yield second-order accuracy, while curvilinear meshes consisting of quadrilateral cells with quadratic curved edges can yield third-order accurate results. We can see that our Lax–Wendroff time discretization only costs about half of the time compared with the Runge–Kutta time discretization in the second-order case. In third-order case, the cost of CPU time is about 1/3 of that for the Runge–Kutta time discretization. The Lax–Wendroff time discretization also uses less computer memory than the Runge–Kutta time discretization.

3.3.2. The Saltzman problem

This problem concerns a strong one-dimensional shock wave driven by a piston, propagating in an initially specified, non-uniform two-dimensional plane mesh. A box of width 1.0 and height 0.1 is filled with a perfect gas with $\gamma = 5/3$. A piston moves from left to right with a velocity of 1.0. Most Lagrangian codes experience severe difficulty with this problem. The initial mesh is composed of 100×10 cells

$$x(i,j) = (i-1)\Delta x + (11-j)\sin(0.01\pi(i-1))\Delta y, \quad y(i,j) = (j-1)\Delta y,$$

with $\Delta x = \Delta y = 0.01$, the initial mesh is displayed in Fig. 8. The initial mesh is deliberately distorted to set it as a more demanding test case. The initial conditions involve a stationary gas with a unity density and an internal energy of 10^{-4}.

Table 3
Second-order schemes. L_1 error. Runge–Kutta and Lax–Wendroff (boldface).

$N_x * N_y$	Time	Density	Order	Momentum	Order	Energy	Order	CPU time (s)
20 * 20	RK	6.46E−3	–	1.48E−2	–	2.79E−2	–	0.1
	LW	**4.92E−3**	–	**1.19E−2**	–	**1.79E−2**	–	**0.04**
40 * 40	RK	1.69E−3	1.94	4.01E−3	1.88	7.28E−3	1.94	0.6
	LW	**1.08E−3**	**2.19**	**3.08E−3**	**1.95**	**4.74E−3**	**1.92**	**0.2**
80 * 80	RK	5.03E−4	1.74	1.16E−3	1.79	2.10E−3	1.79	3.5
	LW	**3.18E−4**	**1.77**	**8.38E−4**	**1.88**	**1.33E−3**	**1.84**	**1.4**
160 * 160	RK	1.57E−4	1.68	3.31E−4	1.81	5.96E−4	1.82	24.0
	LW	**8.58E−5**	**1.89**	**2.17E−4**	**1.95**	**3.60E−4**	**1.88**	**10.1**

Table 4
Third-order schemes on quadrilateral meshes with straight-line edges. L_1 error. Runge–Kutta and Lax–Wendroff (boldface).

$N_x * N_y$	Time	Density	Order	Momentum	Order	Energy	Order	CPU time (s)
20 * 20	RK	3.09E−3	–	5.66E−3	–	1.14E−2	–	0.83
	LW	**2.86E−3**	–	**6.96E−3**	–	**1.16E−2**	–	**0.28**
40 * 40	RK	7.69E−4	2.01	1.38E−3	2.03	2.78E−3	2.04	5.7
	LW	**7.72E−4**	**1.89**	**1.71E−3**	**2.02**	**3.40E−3**	**1.77**	**1.7**
80 * 80	RK	1.87E−4	2.04	3.28E−4	2.08	6.65E−4	2.06	41.5
	LW	**1.99E−4**	**1.95**	**4.13E−4**	**2.05**	**8.56E−4**	**1.99**	**12.7**
160 * 160	RK	4.69E−5	1.99	8.13E−5	2.01	1.64E−4	2.02	279.5
	LW	**5.52E−5**	**1.85**	**1.09E−4**	**1.93**	**2.24E−4**	**1.93**	**98.3**

Table 5
Third-order schemes on curved quadrilateral meshes with quadratic curved edges. L_1 error. Runge–Kutta and Lax–Wendroff (boldface).

$N_x * N_y$	Time	Density	Order	Momentum	Order	Energy	Order	CPU time (s)
20 * 20	RK	1.39E−3	–	3.15E−3	–	6.27E−3	–	9.1
	LW	**1.21E−3**	–	**2.78E−3**	–	**5.10E−3**	–	**3.1**
40 * 40	RK	2.00E−4	2.81	4.75E−4	2.73	9.01E−4	2.81	64.4
	LW	**1.70E−4**	**2.83**	**3.74E−4**	**2.89**	**7.12E−4**	**2.84**	**21.7**
80 * 80	RK	2.61E−5	2.93	6.01E−5	2.98	1.14E−4	2.98	483.4
	LW	**2.25E−5**	**2.91**	**5.04E−5**	**2.89**	**9.23E−5**	**2.95**	**163.8**
160 * 160	RK	3.49E−6	2.91	7.99E−6	2.91	1.48E−5	2.95	3746.7
	LW	**3.26E−6**	**2.79**	**8.04E−6**	**2.65**	**1.30E−5**	**2.82**	**1264.3**

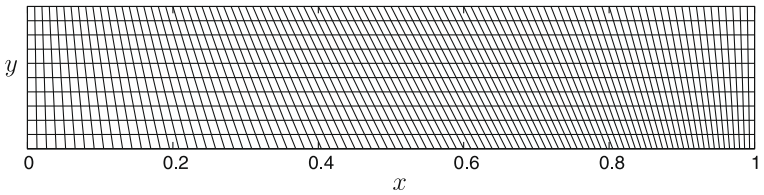


Fig. 8. The initial mesh of the Saltzman problem.

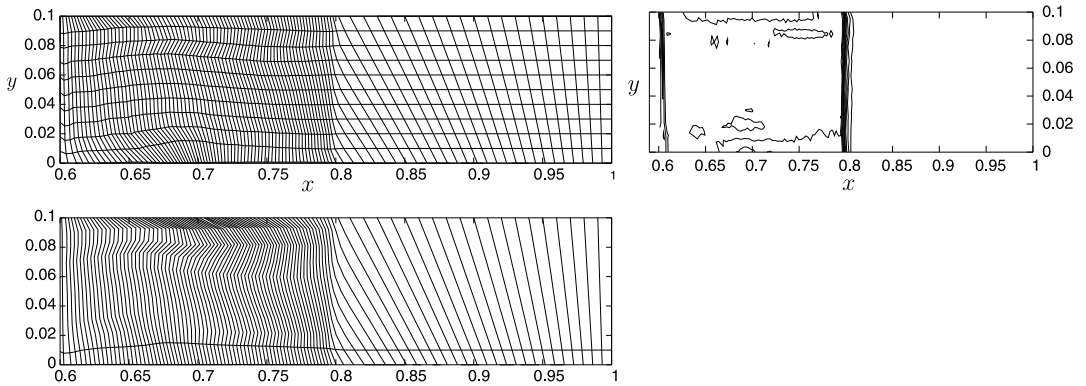
Reflective boundary conditions are used on the right, upper and lower boundaries. The Courant number λ is set to be 0.01 initially and returns to be 0.5 after $t = 0.01$. The analytic post shock density is 4.0 and the shock speed is 1.333. The numerical results are shown at time $t = 0.6$. Fig. 9 shows the results of our second-order scheme using the Dukowicz flux and HLLC flux. Fig. 10 shows the results of our third-order scheme with the Dukowicz flux, using a quadrilateral mesh with straight-line edges and a curvilinear mesh respectively. At this time, the shock is expected to be located at $x = 0.8$. Comparing with those in the literature, the computational results are satisfactory except for the region near the up and bottom wall boundaries where the results are affected by the boundary conditions.

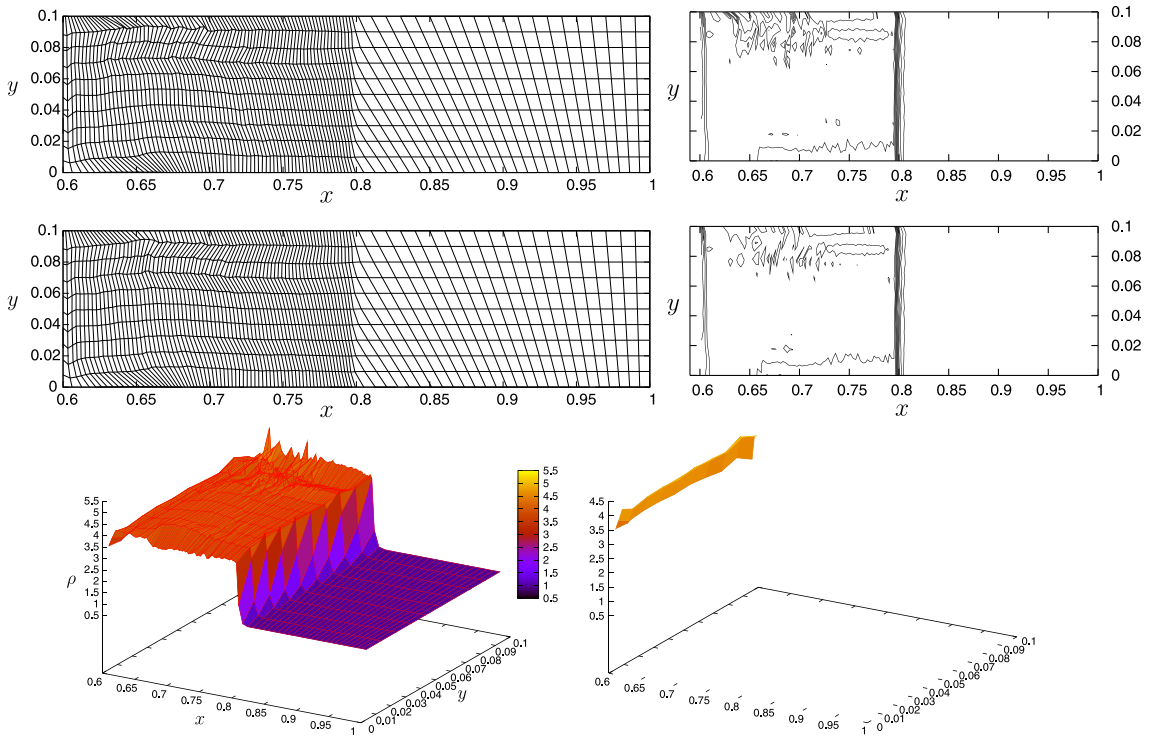
3.3.3. The Sedov blast wave problem in a Cartesian coordinate system

The Sedov problem [24] models the expanding wave by an intense explosion in a perfect gas. The initial grid is uniform mesh with 30×30 rectangular cells with a total edge length of 1.1 in both directions. The initial density is unity and the initial velocity is zero. The specific internal energy is zero except in the first zone where it has a value of 182.09. The analytical solution gives, for $\gamma = 1.4$, a shock at radius unity at time unity with a peak density of 6. Fig. 11 shows the results by the purely Lagrangian calculations with the Dukowicz flux at the time $t = 1$. The left plots are the results of the second-order scheme and the right plots are the results of the third-order scheme on quadrilateral meshes. The Courant number for the third-order scheme needs to be slightly smaller at 0.45. The computational results are comparable in quality with those in the literature.

3.3.4. The Dukowicz problem

The Dukowicz problem is a two-dimensional shock refraction problem on an uneven mesh designed by Dukowicz and Meltz [7].





The computational domain consists of two adjacent regions with different densities but equal pressure. The left region is a 36×30 mesh with a vertical left boundary and a right boundary aligned at 30° to the horizontal direction. The right region is a 40×30 mesh uniformly slanted at 30° to the horizontal direction. The initial conditions of the two regions are $\rho_L = 1, u_L = 0, p_L = 1$ and $\rho_R = 1, u_R = 0, p_R = 1.5$, respectively. In both regions $\gamma = 1.4$. The upper and lower boundaries are reflective and the left boundary is a piston, which moves from the left with velocity 1.48. The problem is run to a time of 1.3, just before the shock would leave the right region.

The results show the interface along with the incident and the transmitted shocks clearly. The reflective shock does not show up clearly due to the small difference in density across it. Fig. 12 is the initial mesh. Fig. 13 shows the second-order and third-order results using the HLLC flux. We remark that for this difficult test case, a rezoning is often necessary when the mesh is seriously distorted, as is the case for the ENO Lagrangian scheme based on Runge–Kutta time discretization [3]. For our third-order scheme, when the time step is less than 2×10^{-3}, we rezone the meshes by keeping the vertices at the left and right boundaries unchanged and redistributing the points, using the conservative ENO remapping algorithm in [4].

3.3.5. Double Mach reflection

We will test the performance of our scheme in the ALE calculations in this problem. At each time step, we first use the Lagrangian scheme to update the solution and the mesh, then we rezone the Lagrangian mesh back to the old rectangular mesh and remap the numerical solution to the rezoned mesh by using the conservative remapping method based on the ENO methodology in [4].

The computational domain for this problem is chosen to be $[0, 4] \times [0, 1]$. The reflecting wall lies at the bottom of the computational domain starting $x = 1/6$. Initially a right-moving Mach 10 shock is positioned at $x = 1/6,\ y = 0$ and makes a 60° angle with the x-axis. For the bottom boundary, the exact post-shock condition is imposed for the part from $x = 0$ to $x = 1/6$ and a reflective boundary condition is used for the rest. At the top boundary of our computational domain, the flow values are set to describe the exact motion of the Mach 10 shock. Here for the third scheme the initial Courant number needs to be smaller at 0.4 and after time $t = 0.01$ the Courant number returns to 0.5. The density results at the time $t = 0.2$ on a 240×60 uniform grid are shown in Fig. 14, which demonstrate our schemes also perform well in the ALE framework. Comparing with the fifth-order finite difference WENO simulation [13] for the same problem using the same number of cells on an Eulerian coordinate (see Fig. 12(b) in [13]), we can observe that the current second-order and especially third-order ALE scheme gives slightly better resolution for the contact discontinuities (the roll-ups under the Mach stem), even though it has lower formal

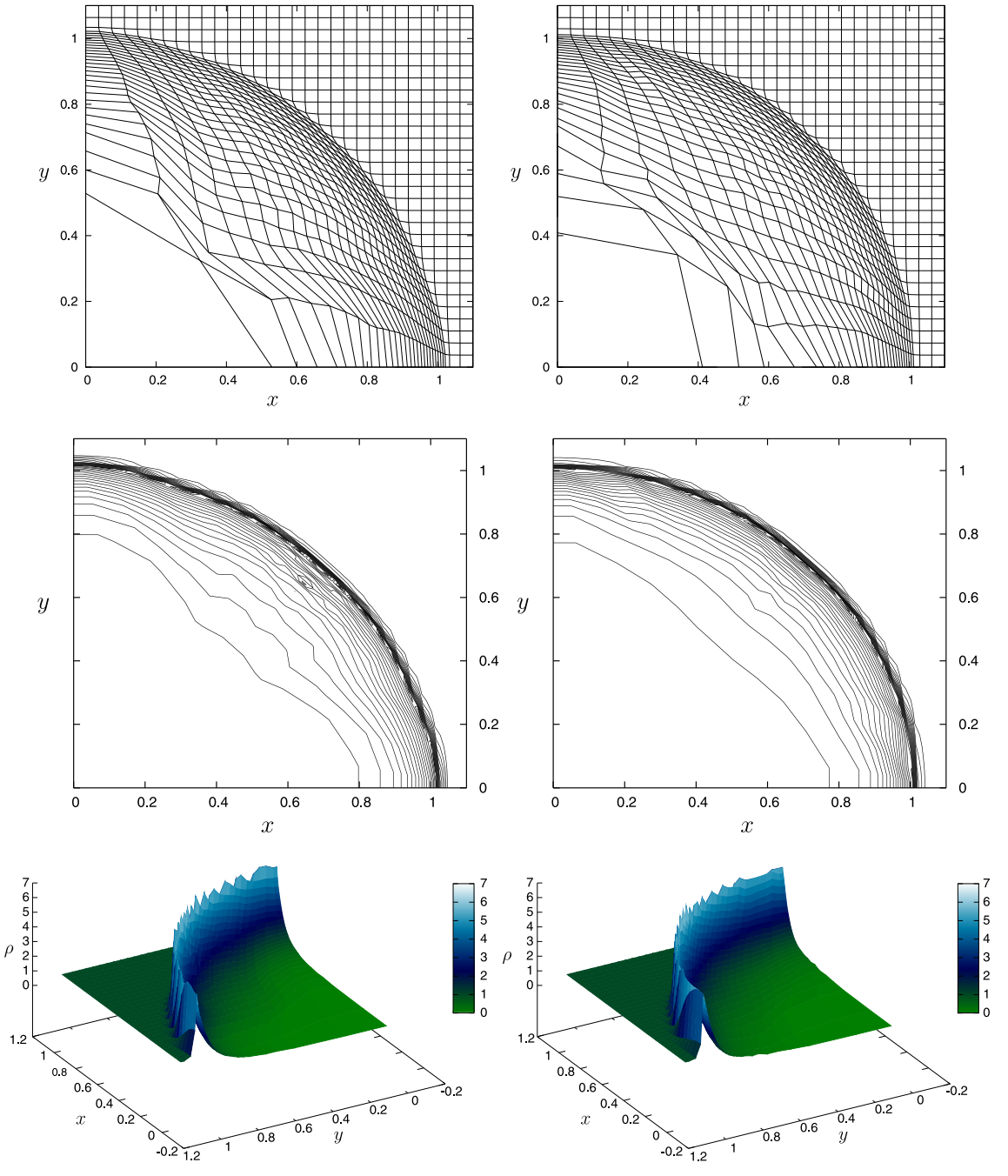
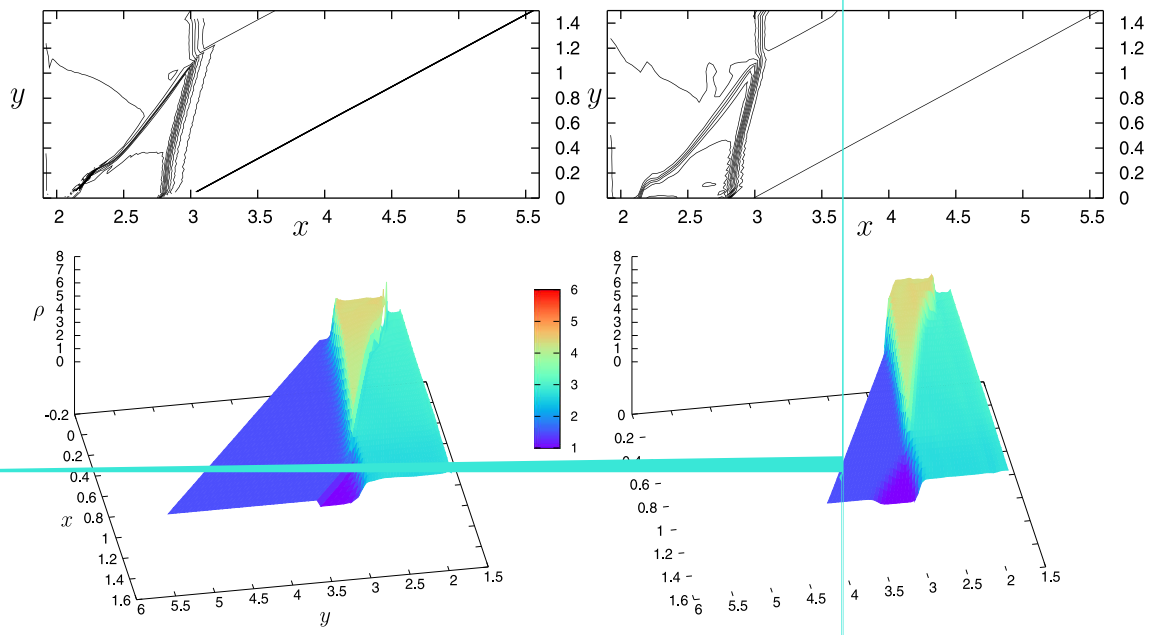


Fig. 11. LW-WENO results for the Sedov problem with the Dukowicz flux. Top row: meshes; middle row: contours of the density; bottom row: surfaces of the density. Left: second-order scheme; right: third-order scheme.



order of accuracy than the Eulerian fifth-order WENO scheme. This might be an indication of the better resolution for contact discontinuities of Lagrangian and ALE schemes.

4. Concluding remarks

In this paper, we have developed a Lax–Wendroff type time discretization method for the high order Lagrangian scheme for the compressible Euler equations based on high order essentially non-oscillatory (ENO) reconstruction. The algorithm is described in detail in one and two spatial dimensions, and extensive numerical tests are provided. Comparing with the Runge–Kutta time discretization, a major advantage of the Lax–Wendroff type time discretization is the saving in computational

cost and storage, at least for the one and two-dimensional cases that we have tested. The computational experiments for both smooth and discontinuous test cases indicate that the Lax–Wendroff time discretization produces results as good as or better than those obtained with the Runge–Kutta time discretization. The relative cost comparison between the Lax–Wendroff type and Runge–Kutta type time discretizations may change in three dimensions. The design and implementation of the three-dimensional Lagrangian method, especially for higher than second-order accuracy where curved meshes are needed, are however highly nontrivial, for both types of time discretization, and are therefore left for future work.

References

[1] D.J. Benson, Momentum advection on a staggered mesh, Journal of Computational Physics 100 (1992) 143–162.
[2] E.J. Caramana, D.E. Burton, M.J. Shashkov, P.P. Whalen, The construction of compatible hydrodynamics algorithms utilizing conservation of total energy, Journal of Computational Physics 146 (1998) 227–262.
[3] J. Cheng, C.-W. Shu, A high order ENO conservative Lagrangian type scheme for the compressible Euler equations, Journal of Computational Physics 227 (2007) 1567–1596.
[4] J. Cheng, C.-W. Shu, A high order accurate conservative remapping method on staggered meshes, Applied Numerical Mathematics 58 (2008) 1042–1060.
[5] J. Cheng, C.-W. Shu, A third order conservative Lagrangian type scheme on curvilinear meshes for the compressible Euler equation, Communications in Computational Physics 4 (2008) 1008–1024.
[6] J.K. Dukowicz, M.C. Cline, F.L. Addessio, A general topology Godunov method, Journal of Computational Physics 82 (1989) 29–63.
[7] J.K. Dukowicz, B.J.A. Meltz, Vorticity errors in multidimensional Lagrangian codes, Journal of Computational Physics 99 (1992) 115–134.
[8] M. Dumbser, M. Kaser, V.A. Titarev, E.F. Toro, Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems, Journal of Computational Physics 226 (2007) 204–243.
[9] M. Dumbser, C.D. Munz, Building blocks for arbitrary high order discontinuous Galerkin Schemes, Journal of Scientific Computing 27 (2006) 215–230.
[10] R.W. Dyson, Technique for very high order nonlinear simulation and validation, NASA Technical Report TM-2001-210985, 2001.
[11] A. Harten, B. Engquist, S. Osher, S.R. Chakravarthy, Uniformly high order accurate essentially non-oscillatory schemes, III, Journal of Computational Physics 71 (1987) 231–303.
[12] C. Hirt, A. Amsden, J. Cook, An arbitrary Lagrangian–Eulerian computing method for all flow speeds, Journal of Computational Physics 14 (1974) 227–253.
[13] G. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, Journal of Computational Physics 126 (1996) 202–228.
[14] B. Koobus, C. Farhat, Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes, Computer Methods in Applied Mechanics and Engineering 170 (1999) 103–129.
[15] P.D. Lax, B. Wendroff, Systems of conservation laws, Communications on Pure and Applied Mathematics 13 (1960) 217–237.
[16] H. Luo, J.D. Baum, R. Löhner, On the computation of multi-material flows using ALE formulation, Journal of Computational Physics 194 (2004) 304–328.
[17] P.H. Maire, R. Abgrall, J. Breil, J. Ovadia, A Lagrangian scheme for multidimensional compressible flow problems, SIAM Journal on Scientific Computing 29 (2007) 1781–1824.
[18] J. von Neumann, R.D. Richtmyer, A method for the calculation of hydrodynamics shocks, Journal of Applied Physics 21 (1950) 232–237.
[19] W.F. Noh, Errors for calculations of strong shocks using an artificial viscosity and an artificial heat flux, Journal of Computational Physics 72 (1987) 78–120.
[20] J.S. Peery, D.E. Carroll, Multi-material ALE methods in unstructured grids, Computer Methods in Applied Mechanics and Engineering 187 (2000) 591–619.
[21] J. Qiu, M. Dumbser, C.-W. Shu, The discontinuous Galerkin method with Lax–Wendroff type time discretizations, Computer Methods in Applied Mechanics and Engineering 194 (2005) 4528–4543.
[22] J. Qiu, C.-W. Shu, Finite difference WENO schemes with Lax–Wendroff-type time discretizations, SIAM Journal on Scientific Computing 24 (2003) 2185–2198.
[23] T. Schwartzkopff, C.D. Munz, E.F. Toro, ADER: a high-order approach for linear hyperbolic systems in 2D, Journal of Scientific Computing 17 (2002) 231–240.
[24] L.I. Sedov, Similarity and Dimensional Methods in Mechanics, Academic Press, New York, 1959.
[25] C.-W. Shu, Numerical experiments on the accuracy of ENO and Modified ENO schemes, Journal of Scientific Computing 5 (1990) 127–149.
[26] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, Journal of Computational Physics 77 (1988) 439–471.
[27] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock capturing schemes II, Journal of Computational Physics 83 (1989) 32–78.
[28] V.A. Titarev, E.F. Toro, ADER schemes for three-dimensional nonlinear hyperbolic systems, Journal of Computational Physics 204 (2005) 715–736.
[29] P.R. Woodward, P. Colella, The numerical simulation of two-dimensional fluid flow with strong shocks, Journal of Computational Physics 54 (1984) 115–173.