Journal of Computational Physics 396 (2019) 544–578

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A quasi-Lagrangian moving mesh discontinuous Galerkin method for hyperbolic conservation laws

Dongmi Luo [a], Weizhang Huang [b], Jianxian Qiu [a,*]

[a] School of Mathematical Sciences and Fujian Provincial Key Laboratory of Mathematical Modeling and High-Performance Scientific Computing, Xiamen University, Xiamen, Fujian 361005, China
[b] Department of Mathematics, University of Kansas, Lawrence, KS 66045, USA

ARTICLE INFO

Article history:
Received 6 December 2018
Received in revised form 2 May 2019
Accepted 27 June 2019
Available online 15 July 2019

Keywords:
Discontinuous Galerkin
High-order method
Moving mesh
Conservation law

ABSTRACT

A moving mesh discontinuous Galerkin method is presented for the numerical solution of hyperbolic conservation laws. The method is a combination of the discontinuous Galerkin method and the mesh movement strategy which is based on the moving mesh partial differential equation approach and moves the mesh continuously in time and orderly in space. It discretizes hyperbolic conservation laws on moving meshes in the quasi-Lagrangian fashion with which the mesh movement is treated continuously and no interpolation is needed for physical variables from the old mesh to the new one. Two convection terms are induced by the mesh movement and their discretization is incorporated naturally in the DG formulation. Numerical results for a selection of one- and two-dimensional scalar and system conservation laws are presented. It is shown that the moving mesh DG method achieves the second and third order of convergence for P^1 and P^2 elements, respectively, for problems with smooth solutions and is able to capture shocks and concentrate mesh points in non-smooth regions. Its advantage over uniform meshes and its insensitiveness to mesh smoothness are also demonstrated.

1. Introduction

We consider the numerical solution of hyperbolic conservation laws in the form

$$\begin{cases} u_t + \nabla \cdot F = 0, \\ u(\mathbf{x}, 0) = u_0(\mathbf{x}), \end{cases} \tag{1.1}$$

where $\mathbf{x} \in \Omega \subset \mathbb{R}^d$ $(d = 1, 2)$, Ω is a bounded domain, $F = (f_1(u), \cdots, f_d(u))$, and u, $f_1(u)$, $\cdots$, $f_d(u)$ are either scalars or vectors. The major difficulty in solving nonlinear hyperbolic conservation laws in (1.1) is that the solution can develop discontinuities even if the initial condition is smooth. The discontinuous Galerkin (DG) method is an increasingly popular approach to solve the equations. The DG method was first introduced by Reed and Hill [35] for solving linear hyperbolic problems associated with neutron transfer. Then a major development of the DG method for time-dependent nonlinear hyperbolic conservation laws was carried out by Cockburn et al. in a series of papers [6–9]. The DG method can capture the weak discontinuity without any modification. But the nonlinear limiters must be applied to control the spurious oscillations

* Corresponding author.
E-mail addresses: dongmiluo@stu.xmu.edu.cn (D. Luo), whuang@ku.edu (W. Huang), jxqiu@xmu.edu.cn (J. Qiu).

https://doi.org/10.1016/j.jcp.2019.06.061
0021-9991/© 2019 Elsevier Inc. All rights reserved.

in the numerical solution for strong shocks. One type of these limiters is based on the slope methodology such as the minmod type limiters [6–9]. These limiters are effective in controlling oscillations. However, the accuracy of the DG method may decrease if they are mistakenly used in smooth regions. Another type of limiter is based on the weighted essentially non-oscillatory (WENO) methodology [15,24], which can achieve both high-order accuracy and non-oscillatory properties. The WENO limiters in [34,50] and the Hermite WENO (HWENO) limiters in [29,32,33,49] belong to this type limiter and require a wide stencil especially for the high-order accuracy. Recently, a simple WENO limiter [48,51] and compact HWENO limiter [52,53] for Runge-Kutta DG (RKDG) were developed. The key idea of these limiters is to reconstruct the whole polynomial in the target element instead of point values or moments.

The physical phenomena in a variety of fields may develop dynamically singular solutions, such as shock waves and detonation waves. If we use the globally uniform mesh method, the computation can become prohibitively expensive when dealing with two or higher dimensional systems. Then adaptive mesh methods that can increase the accuracy of the numerical approximations and decrease the computational cost are in critical need. In general, there are three types of mesh adaptive methods. The first one is h-methods which are widely used and generate a new mesh by adding or removing points to an old mesh. The second one is p-methods with which the order of the polynomial is increased or decreased from place to place according to the solution error. The last one is r-methods which are also called moving mesh methods and relocate the mesh points while keeping the total number of mesh points and the mesh connectivity unchanged. A number of moving mesh methods have been developed in the past; e.g., see the books or review articles [1,2,5,22,39] and references therein and some recent applications [42,45,46].

One of the advantages of DG method is that its numerical solution is discontinuous at edges of mesh elements and flexible for mesh adaptation strategies. There exist a few research works in this aspect. A combination of the hp-method and DG method was developed in [4] for the hyperbolic conservation laws. Li and Tang [28] solved two-dimensional conservation laws using rezoning moving mesh methods where the physical variables are interpolated from the old mesh to the new one using conservative interpolation schemes. The methods are shown to perform well although it is unclear that the method can be high order. Machenzie and Nicola [30] solved the Hamilton-Jacobi equations by the DG method using a moving mesh method based on the moving mesh partial differential equation (MMPDE) strategy. Uzunca, Karasözen, and Küçükseyhan [41] employed the moving mesh symmetric interior penalty Galerkin (SIPG) method to solve PDEs with traveling waves.

In this paper a moving mesh DG method is proposed for the numerical solution of hyperbolic conservation laws. The method is different from those [28,40] in mesh movement strategy and discretization of physical equations. We use here the MMPDE moving mesh strategy [20–22] which employs a meshing functional based on the equidistribution and alignment conditions and a matrix-valued function that provides the information needed to control the size, shape, and orientation of mesh elements from place to place. Moreover, the newly developed discretization of the MMPDE [18] is used here, which makes the implementation more easier and much more reliable since there is a theoretical guarantee for mesh nonsingularity [19]. In contrast to the moving mesh finite difference WENO method [44] where the smoothness of the mesh is extremely important for accuracy, we find that the moving mesh DG method presented in this work is not sensitive to the smoothness of the mesh. Furthermore, the method discretizes hyperbolic conservation laws on moving meshes in the quasi-Lagrangian fashion with which the mesh movement is treated continuously and no interpolation is needed for physical variables from the old mesh to the new one. Numerical results for a selection of one- and two-dimensional scalar and system conservation laws will be presented to demonstrate that the moving mesh DG method achieves the second and third order of convergence for P^1 and P^2 elements, respectively, for problems with smooth solutions and is able to capture shocks and concentrate mesh points in non-smooth regions. The sensitivity of the accuracy of the method to mesh smooth will also be discussed.

It is noted that adaptive moving mesh methods like the one proposed in this work are closely related to so-called arbitrary Lagrangian-Eulerian (ALE) methods [14,31] widely used in computational fluid dynamics and both groups have the quasi-Lagrange version (that treats mesh movement continuously in time) and the rezoning version (that treats mesh movement in a discontinuous fashion and interpolates or remaps the physical variables from the old mesh to the new one); e.g. see [22, Sections 1.5 and 7.1] for more detailed discussion. Interestingly, it is not difficult to show that the quasi-Lagrange DG discretization on moving meshes considered in the current work is the same as the ALE-DG method of [13] when the mesh velocity field is calculated through linear interpolation from the nodal mesh velocities. We should emphasize that the current work focuses on the application of the MMPDE moving mesh approach while [13] gives an analysis of the ALE-DG method for a general mesh velocity field assumed to be bounded and to result in a regular and nonsingular mesh. Several adaptive moving mesh finite element methods have been analyzed under similar assumptions on mesh movement by, e.g., Bank and Santos [3], Dupont [11], and Dupont and Liu [12].

The organization of the paper is as follows. In Section 2, the DG method, identification of troubled cells, and limiters on moving meshes are described in detail. The MMPDE moving mesh strategy is described in Section 3. The algorithm of the moving mesh DG method is presented in Section 4. In Section 5, one- and two-dimensional numerical examples are presented to demonstrate the accuracy and the mesh adaptation capability of the scheme. Conclusions are drawn in Section 6.

2. DG method on moving meshes

In this section we describe a Runge-Kutta DG (RKDG) method for the numerical solution of conservation laws in the form of (1.1) on a moving triangular mesh and pay special attention to limiters and the identification of troubled cells. We will discuss the generation of adaptive moving meshes in the next section.

2.1. The RKDG discretization on moving meshes

For the moment we assume that a moving mesh $\mathscr{T}_h(t)$ for the domain Ω is given at time instants

$$t_0 = 0 < t_1 < \cdots t_n < t_{n+1} < \cdots \le T.$$

The appearances are denoted by $\mathscr{T}_h^n$, $n = 0, 1, \ldots$. Since they belong to the same mesh, they have the same number of elements (N) and vertices (N_v) and the same connectivity, and differ only in the location of the vertices. Denote the coordinates of the vertex of $\mathscr{T}_h^n$ by $\mathbf{x}_j^n$, $j = 1, 2, \cdots, N_v$. Between any time interval t_n and t_{n+1}, the coordinates and velocities of the vertices of the mesh are defined as

$$\begin{aligned} \mathbf{x}_j(t) &= \frac{t-t_n}{\Delta t_n}\mathbf{x}_j^{n+1} + \frac{t_{n+1}-t}{\Delta t_n}\mathbf{x}_j^n, \quad j = 1, 2, \cdots, N_v, \\ \dot{\mathbf{x}}_j(t) &= \frac{\mathbf{x}_j^{n+1} - \mathbf{x}_j^n}{\Delta t_n}, \quad j = 1, 2, \cdots, N_v, \end{aligned} \tag{2.1}$$

where $\Delta t_n = t_{n+1} - t_n$. The DG finite element space is defined as

$$V_h^k = \{p(\mathbf{x}, t) : p|_K \in P^k(K), \forall K \in \mathscr{T}_h(t)\},$$

where $P^k(K)$ is the space of polynomials of degree $\le k$ defined on K. Notice that $P^k(K)$ can be expressed as

$$P^k(K) = \text{span}\{\phi_1(\mathbf{x}, t), \cdots, \phi_L(\mathbf{x}, t)\},$$

where $L = k+1$ for one dimensional case, $L = \frac{(k+1)(k+2)}{2}$ for two dimensional case, and $\{\phi_1(\mathbf{x}, t), \cdots, \phi_L(\mathbf{x}, t)\}$ is a basis of $P^k(K)$. Notice that the time dependence of the basis functions comes from the time dependence of the location of the vertices.

The semi-discrete DG approximation for (1.1) is to find $u_h(\cdot, t) \in V_h^k$, $t \in (0, T]$ such that

$$\int_K \frac{\partial u_h}{\partial t} v dxdy + \int_{\partial K} F \cdot \vec{n} v ds - \int_K F \cdot \nabla v dxdy = 0, \qquad \forall v \in P^k(K), \quad \forall K \in \mathscr{T}_h(t) \tag{2.2}$$

where $\vec{n} = (n_x, n_y)$ is the outward unit normal vector of the triangular boundary ∂K. Expressing u_h as

$$u_h(\mathbf{x}, t) = \sum_{j=1}^{L} u_j^K(t)\phi_j(\mathbf{x}, t), \quad \forall \mathbf{x} \in K$$

we can find its time derivative as

$$\frac{\partial u_h}{\partial t} = \sum_{j=1}^{L} \frac{du_j^K(t)}{dt}\phi_j(\mathbf{x}, t) + \sum_{j=1}^{L} u_j^K(t)\frac{\partial \phi_j(\mathbf{x}, t)}{\partial t}.$$

It is not difficult to show [25] that

$$\frac{\partial \phi_j(\mathbf{x}, t)}{\partial t} = -\nabla \phi_j(\mathbf{x}, t) \cdot \dot{X}(\mathbf{x}, t),$$

where $\dot{X}(\mathbf{x}, t)$ is the linear interpolation of the nodal mesh speed in K, viz.,

$$\dot{X}(\mathbf{x}, t) = \sum_{l=1}^{3} \dot{\mathbf{x}}_l \Phi_l(\mathbf{x}, t)$$

where $\dot{\mathbf{x}}_l$, $l = 1, 2, 3$ are the nodal mesh speed and $\Phi_l(\mathbf{x}, t)$, $l = 1, 2, 3$ are the linear basis functions. Then, we get

$$\frac{\partial u_h}{\partial t} = \sum_{j=1}^{L} \frac{du_j^K(t)}{dt}\phi_j(\mathbf{x}, t) - \nabla u_h(\mathbf{x}, t) \cdot \dot{X}(\mathbf{x}, t).$$

Inserting this to (2.2) yields

$$\int_K \left(\sum_{j=1}^{L} \frac{du_j^K(t)}{dt} \phi_j(\mathbf{x},t) - \nabla u_h(\mathbf{x},t)\cdot \dot{X}(\mathbf{x},t) \right) v dxdy + \int_{\partial K} F\cdot \vec{n} v ds - \int_K F\cdot \nabla v dxdy = 0.$$

Integrating the second term by parts, we obtain

$$\int_K \sum_{j=1}^{L} \frac{du_j^K(t)}{dt} \phi_j v dxdy + \int_{\partial K} (F - u_h \dot{X})\cdot \vec{n} v ds - \int_K \left(F\cdot \nabla v - u_h \nabla\cdot(\dot{X} v)\right) dxdy = 0. \tag{2.3}$$

From the above equation we can see that two advection terms are induced by the mesh movement. Denote

$$H(u_h) \equiv (F(u_h) - u_h \dot{X}(\mathbf{x},t))\cdot \vec{n}, \qquad H_1(u_h) \equiv F\cdot \nabla v - u_h \nabla\cdot(\dot{X} v).$$

Applying the Gauss quadrature rule to the second and third terms in (2.3), we get

$$\int_{\partial K} H(u_h) v ds \approx \sum_e \sum_{G_e} H(u_h(\mathbf{x}_{G_e})) v(\mathbf{x}_{G_e}) w_{G_e} |e|,$$

$$\int_K H_1(u_h) dxdy \approx \sum_G H_1(u_h(\mathbf{x}_G)) w_G |K|,$$

where e represents the edges of the element K, $|K|$ is the volume of the element K, and $\mathbf{x}_G$ and $\mathbf{x}_{G_e}$ represent the Gaussian points on K and e, respectively. The summations $\sum_e$, $\sum_G$, and $\sum_{G_e}$ are taken over the edges of ∂K, Gauss points on K, and Gauss points on e, respectively. Replacing the flux H by the numerical flux $\hat{H}$, we obtain

$$\begin{cases} \int_K \sum_{j=1}^{L} \frac{du_j^K(t)}{dt} \phi_j(\mathbf{x},t) v dxdy + \sum_e \sum_{G_e} \hat{H} v(\mathbf{x}_{G_e}^{int}) w_{G_e} |e| \\ \qquad\qquad - \sum_G H_1(u_h(\mathbf{x}_G)) w_G |K| = 0, \quad \forall K \in \mathscr{T}_h, \quad v \in V_h^k \\ \int_K (u_h(\mathbf{x},0) - u_0(\mathbf{x})) v dxdy = 0, \quad \forall K \in \mathscr{T}_h, \quad v \in V_h^k \end{cases} \tag{2.4}$$

where the numerical flux has the form $\hat{H} = \hat{H}(u(\mathbf{x}_{G_e}^{int}), u(\mathbf{x}_{G_e}^{ext}))$ and $u(\mathbf{x}_{G_e}^{int})$ and $u(\mathbf{x}_{G_e}^{ext})$ are defined as the values from the interior and exterior of K, respectively, i.e.,

$$u(\mathbf{x}_{G_e}^{int}, t) = \lim_{\mathbf{x}\to\mathbf{x}_{G_e}, \mathbf{x}\in K} u_h(\mathbf{x},t), \quad u(\mathbf{x}_{G_e}^{ext}, t) = \lim_{\mathbf{x}\to\mathbf{x}_{G_e}, \mathbf{x}\notin K} u_h(\mathbf{x},t).$$

The numerical flux $\hat{H}(a,b)$ is required to satisfy the following conditions:

(i) $\hat{H}(a,b)$ is Lipschitz continuous in both arguments a and b;
(ii) $\hat{H}$ is consistent with $H(u)$, namely, $\hat{H}(u,u) = H(u)$.

In this work, we use the local Lax-Friedrichs flux,

$$\hat{H}(a,b) = \frac{1}{2}\left(H(a) + H(b) - \alpha_{e,K}(b-a)\right),$$

where $\alpha_{e,K}$ is the numerical viscosity constant taken as the largest eigenvalues in magnitude of

$$\frac{\partial}{\partial u}(F(\bar{u}_K) - \bar{u}_K \dot{X}(\mathbf{x},t))\cdot \vec{n}, \qquad \frac{\partial}{\partial u}(F(\bar{u}_{K'}) - \bar{u}_{K'} \dot{X}(\mathbf{x},t))\cdot \vec{n},$$

where K and K' are the elements sharing the common edge e and

$$\bar{u}_K = \frac{1}{|K|}\int_K u_h dxdy, \quad \bar{u}_{K'} = \frac{1}{|K'|}\int_{K'} u_h dxdy. \tag{2.5}$$

Finally the semi-discrete scheme (2.4) is discretized in time. Here, we use an explicit, the third order TVD Runge-Kutta scheme [36]. Casting (2.4) in the form

$$\frac{\partial u_h}{\partial t} = L_h(u_h, t),$$

the scheme reads as

$$\begin{aligned}
u_h^* &= u_h^n + \Delta t_n L_h(u_h^n, t_n),\\
u_h^{**} &= \frac{3}{4}u_h^n + \frac{1}{4}(u_h^* + \Delta t_n L_h(u_h^*, t_n + \Delta t_n)),\\
u_h^{n+1} &= \frac{1}{3}u_h^n + \frac{2}{3}(u_h^{**} + \Delta t_n L_h(u_h^{**}, t_n + \frac{1}{2}\Delta t_n)).
\end{aligned} \tag{2.6}$$

The time step Δt_n is chosen to ensure the stability of the method. For a fixed mesh, the time step is commonly taken as

$$\Delta t' = \text{CFL} \cdot \frac{\min\limits_j R_j^n}{\max\limits_u |F'(u_h^n)\cdot \vec{n}|}, \tag{2.7}$$

where R_j^n is the radius of the inscribed circle of the jth element at t_n. For a moving mesh, we need to consider the effects of mesh movement and thus take the time step as

$$\Delta t'' = \text{CFL} \cdot \frac{\min\limits_j R_j^{n+1}}{\max\limits_u |(F'(u_h^n) - \dot{X}^n)\cdot \vec{n}|}. \tag{2.8}$$

Finally, the time step is taken as $\Delta t_n = \min\{\Delta t', \Delta t''\}$.

2.2. Identification of troubled cells and the limiting procedure

A DG method can capture weak discontinuities without any modification. However, nonlinear limiters must be applied to control spurious oscillations in the numerical solution for strong shocks. We follow the general procedure of Qiu and Shu [34], i.e., first identify 'troubled' cells that may need the limiting procedure and then add the nonlinear limiters on those cells.

There exist a number of indicators, such as KXRCF [27] and the minmod-type TVB limiters in [32,34,48]. We use the minmod-type TVB limiter to detect the discontinuity in the current work. To save space, we omit here the detail of the limiter and refer the reader to [6–9].

Having detected the troubled cells, we apply the nonlinear limiters those cells. Many limiters have been studied in the literature for RKDG such as the minmod-type TVB limiter [6–9], the limiter based on the classic WENO method [34,50] and HWENO method [29,32,33,49], the simple WENO limiter [48,51], and the simple and compact HWENO limiter [52,53]. For the minmod-type TVB limiter, the accuracy may decrease if it is mistakenly used in smooth regions. For the limiters in [29,32–34,49,50], they can achieve high order, but this type of limiter require a wide stencil for high-order accuracy. The simple WENO limiters in [48,51] and simple HWENO limiters in [52,53] have the advantages including the compactness of its stencil and freedom in choosing linear weights. However, from [48] we can observe the simple WENO limiters may not be robust enough for problems containing the strong shocks or low pressure problems, especially for higher order polynomials. Thus, in this paper, we adopt the simple and compact HWENO limiters in [52,53], which are more robust than the simple WENO limiters in [48,51]. The key idea of the procedure is to reconstruct the entire polynomial on a troubled cell as a convex combination of the DG solution polynomial on this cell and the "modified" DG solution polynomials on its immediately neighboring cells. The modification procedure is in a least square manner [10]. Since the limiting procedure is the same as for a fixed mesh, we omit the detail here. The interested reader is referred to [52,53] for the detail.

3. The MMPDE moving mesh strategy

In this section we discuss the generation of $\mathcal{T}_h^{n+1}$ using the MMPDE moving mesh method. To this end, we assume that the mesh $\mathcal{T}_h^n$ at $t = t_n$ and the numerical approximation u_h^n of a physical variable u are given. We also assume that a reference computational mesh $\hat{\mathcal{T}}_c = \{\hat{\boldsymbol{\xi}}_j\}_{j=1}^{N_v}$, a deformation of the physical mesh, has been chosen. This mesh is fixed for the whole computation and can be taken as the initial physical mesh. For the purpose of mesh generation, we need to use another mesh, called the computational mesh $\mathcal{T}_c = \{\boldsymbol{\xi}_j\}_{j=1}^{N_v}$, which is also a deformation of the physical mesh and will be used as an intermediate variable.

The MMPDE method views any nonuniform mesh as a uniform one in some metric specified by a metric tensor $\mathbb{M} = \mathbb{M}(\mathbf{x})$. The metric tensor $\mathbb{M}(\mathbf{x})$ is a symmetric and positive definite matrix for each $\mathbf{x}$ and uniformly positive definite on Ω. It provides the information needed to control the size, shape and orientation of the mesh elements throughout the domain. In our computation we use

$$\mathbb{M} = \det(\mathbb{I} + |H(u_h^n)|)^{-\frac{1}{d+4}}(\mathbb{I} + |H(u_h^n)|), \tag{3.1}$$

where d is the dimension of the domain ($d=1$ for one dimension and $d=2$ for two dimensions), $\mathbb{I}$ is the $d \times d$ identity matrix, $H(u_h^n)$ is a recovered Hessian from the numerical solution u_h^n, $|H(u_h^n)| = Q\,\mathrm{diag}(|\lambda_1|, \cdots, |\lambda_d|)Q^T$ with $Q\,\mathrm{diag}(\lambda_1, \cdots, \lambda_d)Q^T$ being the eigen-decomposition of $H(u_h^n)$, and $\det(\cdot)$ is the determinant of a matrix. The metric tensor (3.1) is known to be optimal for the L^2 norm of linear interpolation error [23]. In our computation the Hessian is recovered using the quadratic least square fitting [47]. More specifically, for a given node, at least five neighboring nodes from its element patch are selected. If the patch does not contain a sufficient number of nodes, nodes from the neighboring patches are then added to the selection. Then quadratic polynomials are formed by least squares fitting the function values at the selected nodes. Finally, the Hessian is obtained by differentiating the obtained polynomial twice. It has been shown in [26] that such a recovered Hessian works well in mesh adaptation although it may not be necessarily convergent on a general mesh.

It is common practice in moving mesh computation to smooth the metric tensor/monitor function for smoother meshes. To this end, we apply a low-pass filter [22,44] to the smoothing of the metric tensor several sweeps every time it is computed.

For scalar equations, we use the solution to the equation as u in computing $\mathbb{M}$. For the Euler system, motivated by the choice in [38], we take u to be the quantity $S = 0.5\sqrt{1+\beta(\frac{\rho}{\max(\rho)})^2} + 0.5\sqrt{1+\beta(\frac{E}{\max(E)})^2}$. Its nodal value at $\mathbf{x}_j$ ($j = 1, 2, \cdots, N_v$) is computed as

$$S_j = 0.5\sqrt{1+\beta(\frac{\rho_j}{\max\limits_{1\leqslant m\leqslant N_v}(\rho_m)})^2} + 0.5\sqrt{1+\beta(\frac{E_j}{\max\limits_{1\leqslant m\leqslant N_v}(E_m)})^2}, \tag{3.2}$$

$$\rho_j = \frac{\sum\limits_{K\in\omega_j}|K|\rho_K}{\sum\limits_{K\in\omega_j}|K|}, \qquad E_j = \frac{\sum\limits_{K\in\omega_j}|K|E_K}{\sum\limits_{K\in\omega_j}|K|},$$

where ω_j is the element patch associated with $\mathbf{x}_j$ and β is a positive parameter. The choice of β is given in Section 5.

We now describe the MMPDE moving mesh strategy. A mesh $\mathcal{T}_h$ is uniform in the metric $\mathbb{M}$ with reference to a computational mesh $\mathcal{T}_c$ will be referred to as an $\mathbb{M}$-uniform mesh with respect to $\mathcal{T}_c$. It is known [16] that such a mesh satisfies the equidistribution and alignment conditions

$$|K|\sqrt{\det(\mathbb{M}_K)} = \frac{\sigma_h|K_c|}{|\Omega_c|}, \quad \forall K \in \mathcal{T}_h \tag{3.3}$$

$$\frac{1}{d}\mathrm{tr}((F_K')^{-1}\mathbb{M}_K^{-1}(F_K')^{-T}) = \det((F_K')^{-1}\mathbb{M}_K^{-1}(F_K')^{-T})^{\frac{1}{d}}, \quad \forall K \in \mathcal{T}_h \tag{3.4}$$

where K_c is the element in $\mathcal{T}_c$ corresponding to K, F_K is the affine mapping from K_c to K and F_K' is its Jacobian matrix, $\mathbb{M}_K$ is the average of $\mathbb{M}$ over K, $\mathrm{tr}(\cdot)$ is the trace of a matrix, and

$$|\Omega_c| = \sum_{K_c\in\mathcal{T}_c}|K_c|, \quad \sigma_h = \sum_{K\in\mathcal{T}_h}|K|\det(\mathbb{M}_K)^{\frac{1}{2}}.$$

The equidistribution condition (3.3) determines the size of the element K through the metric tensor $\mathbb{M}$. The volume $|K|$ is smaller in regions where $\det(\mathbb{M}_K)^{\frac{1}{2}}$ is larger. On the other hand, the alignment condition (3.4) determines the shape and orientation of K through $\mathbb{M}_K$ and K_c.

The objective of the MMPDE moving mesh method is to generate a mesh satisfying the above two conditions as closely as possible. This is done by minimizing the energy function

$$I_h(\mathcal{T}_h, \mathcal{T}_c) = \sum_{K\in\mathcal{T}_h}|K|\sqrt{\det(\mathbb{M}_K)}(\mathrm{tr}((F_K')^{-1}\mathbb{M}_K^{-1}(F_K')^{-T}))^{\frac{3d}{4}}$$
$$+ d^{\frac{3d}{4}}\sum_{K\in\mathcal{T}_h}|K|\sqrt{\det(\mathbb{M}_K)}\left(\frac{|K_c|}{|K|\sqrt{\det(\mathbb{M}_K)}}\right)^{\frac{3}{2}}, \tag{3.5}$$

which is a Riemann sum of a continuous functional developed in [17] based on equidistribution and alignment for variational mesh adaptation. Notice that $I_h(\mathcal{T}_h, \mathcal{T}_c)$ is a function of the vertices $\{\boldsymbol{\xi}_j\}$ of the computational mesh $\mathcal{T}_c$ and the vertices $\{\mathbf{x}_j\}$ of the physical mesh $\mathcal{T}_h$. Here we use the ξ-formulation where we take $\mathcal{T}_h = \mathcal{T}_h^n$ and minimize $I_h(\mathcal{T}_h^n, \mathcal{T}_c)$ by solving its gradient system with respect to $\{\boldsymbol{\xi}_j\}$. Thus, the mesh equation reads as

$$\frac{d\boldsymbol{\xi}_j}{dt} = -\frac{P_j}{\tau}\left(\frac{\partial I_h}{\partial \boldsymbol{\xi}_j}\right)^T, \quad j = 1, 2, \cdots, N_v \tag{3.6}$$

where $\frac{\partial I_h}{\partial \boldsymbol{\xi}_j}$ is considered as a row vector, $\tau > 0$ is a parameter used to adjust the time scale of the mesh movement to respond the changes in $\mathbb{M}$, and P_j is a positive function used to make the MMPDE to have desired invariant properties. In this paper, we take $P_j = \det(\mathbb{M}(\mathbf{x}_j))^{\frac{1}{4}}$ so that (3.6) is invariant under scaling transformations of $\mathbb{M}$. Using the notion of scalar-by-matrix differentiation, we can find the analytical formulations of the derivatives in (3.6) [18] and rewrite the mesh equation as

$$\frac{d\boldsymbol{\xi}_j}{dt} = \frac{P_j}{\tau} \sum_{K\in\omega_j} |K| \boldsymbol{v}^K_{j_K}, \quad j = 1, 2, \cdots, N_v \tag{3.7}$$

where j_K is the local index of $\mathbf{x}_j$ in K and $\boldsymbol{v}^K_{j_K}$ is the local velocity for $\mathbf{x}_j$ contributed by K. The local velocities contributed by K to its vertices are given by

$$\begin{bmatrix} (\boldsymbol{v}_1^K)^T \\ \vdots \\ (\boldsymbol{v}_d^K)^T \end{bmatrix} = -E_K^{-1} \frac{\partial G}{\partial \mathbb{J}} - \frac{\partial G}{\partial \det(\mathbb{J})} \frac{\partial \det(\hat{E}_K)}{\partial \det(E_K)} \hat{E}_K^{-1}, \quad \boldsymbol{v}_0^K = -\sum_{i=1}^{d} \boldsymbol{v}_i^K, \tag{3.8}$$

where the vertices of K and K_c are denoted by $\mathbf{x}_j^K$, $j = 0, 1, ..., d$ and $\boldsymbol{\xi}_j^K$, $j = 0, 1, ..., d$, respectively, $E_K = [\mathbf{x}_1^K - \mathbf{x}_0^K, \cdots, \mathbf{x}_d^K - \mathbf{x}_0^K]$ and $\hat{E}_K = [\boldsymbol{\xi}_1^K - \boldsymbol{\xi}_0^K, \cdots, \boldsymbol{\xi}_d^K - \boldsymbol{\xi}_0^K]$ are the edge matrices of K and K_c, the function $G = G(\mathbb{J}, \det(\mathbb{J}), \mathbb{M}_K)$ with $\mathbb{J} = (F_K')^{-1} = \hat{E}_K E_K^{-1}$ is associated with the energy function (3.5), and its definition and derivatives are given by

$$\begin{aligned} &G(\mathbb{J}, \det(\mathbb{J}), \mathbb{M}) = \sqrt{\det(\mathbb{M})}(\mathrm{tr}(\mathbb{J}\mathbb{M}^{-1}\mathbb{J}^T))^{\frac{3d}{4}} + d^{\frac{3d}{4}} \sqrt{\det(\mathbb{M})} \left(\frac{\det(\mathbb{J})}{\sqrt{\det(\mathbb{M})}} \right)^{\frac{3}{2}}, \\ &\frac{\partial G}{\partial \mathbb{J}} = \frac{3d}{2} \sqrt{\det(\mathbb{M})}(\mathrm{tr}(\mathbb{J}\mathbb{M}^{-1}\mathbb{J}^T))^{\frac{3d}{4}-1} \mathbb{M}^{-1}\mathbb{J}^T, \\ &\frac{\partial G}{\partial \det(\mathbb{J})} = \frac{3}{2} d^{\frac{3d}{4}} \det(\mathbb{M})^{-\frac{1}{4}} \det(\mathbb{J})^{\frac{1}{2}}. \end{aligned}$$

In practical computation, we first compute the edge matrices and the local velocities for all elements and then use (3.7) to obtain the nodal mesh velocities. The mesh equation is modified for the boundary mesh points. For fixed points, the mesh velocity can be set to be zero. For those on a boundary edge, the mesh velocities should be modified to ensure they stay on the boundary.

The mesh equation (3.7) (with the proper modifications for boundary vertices) can be integrated from t_n to t_{n+1}, starting with the reference computational mesh $\hat{\mathscr{T}}_c$ as an initial mesh. Then the new computational mesh $\mathscr{T}_c^{n+1}$ is obtained and forms a correspondence ψ_h with the physical mesh $\mathscr{T}_h^n$ with the property $\mathbf{x}_j^n = \psi_h(\boldsymbol{\xi}_j^{n+1})$, $j = 1, 2, \cdots, N_v$. Finally, the new physical mesh $\mathscr{T}_h^{n+1}$ is defined as $\mathbf{x}_j^{n+1} = \psi_h(\hat{\boldsymbol{\xi}}_j)$, $j = 1, 2, \cdots, N_v$, which can readily be computed using linear interpolation.

The equation (3.7) is called the ξ-formulation of the MMPDE moving mesh method. Alternatively, we can use the x-formulation where we take $\mathscr{T}_c = \hat{\mathscr{T}}_c$ and minimize I_h by integrating its gradient system with respect to $\{\mathbf{x}_j\}$; see [16]. Although its implementation is more complex and costly than the ξ-formulation, the x-formulation has the advantage that it can be shown analytically [19] that the moving mesh governed by the x-formulation will stay free of tangling and cross-over for both convex or concave domains. Such a theoretical result is not available for the ξ-formulation although the numerical experiment shows that it also produces nonsingular moving meshes.

4. The moving mesh DG method

We now summarize the procedure of the method. Assume the physical solution u_h^n and the mesh $\mathbf{x}^n$ (or denoted by $\mathscr{T}_h^n$) are given at $t = t_n$.

- **Step 1: Moving mesh.**
  - Compute the time step using (2.7);
  - Use u_h^n and $\mathbf{x}^n$ to compute the monitor function $\mathbb{M}$ and then smooth it several times;
  - Integrate the MMPDE (3.7) with an initial mesh $\hat{\mathscr{T}}_c$ to get the new computational mesh $\mathscr{T}_c^{n+1}$. Recall that the correspondence between $\mathscr{T}_c^{n+1}$ and $\mathscr{T}_h^n$ is denoted by ψ_h, i.e., $\mathscr{T}_h^n = \psi_h(\mathscr{T}_c^{n+1})$;
  - Define the new physical mesh $\tilde{\mathscr{T}}_h^{n+1} = \psi_h(\hat{\mathscr{T}}_c)$ and then compute the velocity of the nodes by (2.1):

$$\dot{\mathbf{x}}_j(t) = \frac{\tilde{\mathbf{x}}_j^n - \mathbf{x}_j^n}{\Delta t'}, \quad j = 1, 2, \cdots, N_v, \quad t_n \leqslant t \leqslant t_n + \Delta t'.$$

Table 5.1
Example 5.1: Solution error with periodic boundary conditions and $T=\frac{0.5}{\pi}$.

k	N	160	320	640	1280	2560	5120
1	L^1	1.561e-5	4.107e-6	1.061e-6	2.705e-7	6.856e-8	1.733e-8
	Order		1.93	1.95	1.97	1.98	1.98
	L^2	4.214e-5	1.133e-5	2.956e-6	7.560e-7	1.922e-7	4.880e-8
	Order		1.89	1.94	1.97	1.98	1.98
	L_∞	5.199e-5	1.367e-5	3.544e-6	8.970e-7	2.257e-7	5.701e-8
	Order		1.93	1.95	1.98	1.99	1.98
2	L^1	1.267e-7	1.735e-8	2.216e-9	2.619e-10	2.927e-11	3.223e-12
	Order		2.87	2.97	3.08	3.16	3.18
	L^2	8.116e-7	1.444e-7	2.250e-8	2.984e-9	3.336e-10	3.197e-11
	Order		2.49	2.68	2.91	3.16	3.38
	L_∞	2.924e-6	5.921e-7	1.053e-7	1.612e-8	2.120e-9	2.427e-10
	Order		2.30	2.49	2.71	2.93	3.13

- Compute the time step $\Delta t''$ by (2.8) and let $\Delta t_n = \min(\Delta t', \Delta t'')$. Finally, the physical mesh $\mathcal{T}_h^{n+1}$ is obtained as

$$\mathbf{x}_j^{n+1} = \mathbf{x}_j^n + \Delta t_n \cdot \dot{\mathbf{x}}_j, \quad j = 1, 2, \cdots, N_v.$$

- **Step 2: Solve the physical equations.** The conservation law (1.1) is integrated from t_n to $t_{n+1} = t_n + \Delta t_n$ using the DG method described in Section 2. The numerical solution u_h^{n+1} at $t = t_{n+1}$ is obtained.
- **Step 3:** If $t_{n+1} < T$, we compute the time step $\Delta t'$ using the CFL stability condition and go to Step 1.

5. Numerical examples

In this section we present numerical results obtained with the moving mesh DG method described in the previous sections for a selection of one- and two-dimensional examples. Recall that the method has been described in two dimensions. Its implementation in one dimension is similar. The CFL number in time step selection is set to be 0.3 for P^1 elements and 0.15 for P^2 elements. The parameter τ in (3.6) is taken as 0.01 for accuracy test problems and 10^{-3} and 10^{-4} for one- and two-dimensional systems with discontinuities, respectively. The parameter β in (3.2) is taken as 10 for one-dimensional examples and 1 for two-dimensional problems, unless otherwise stated. Moving and uniform meshes will be denoted by "MM" and "UM", respectively. Unless otherwise stated, three sweeps of a low-pass filter [22,44] are applied to the smoothing of the metric tensor every time it is computed. In addition, for examples having an exact solution, the error of the computed solution is measured in the global norm, i.e.,

$$\|e_h\|_{L^q} = \left(\int_0^T \int_\Omega |e_h(\mathbf{x}, t)|^q d\mathbf{x} dt \right)^{\frac{1}{q}}, \quad q = 1, 2, \infty.$$

5.1. One-dimensional examples

Example 5.1. We first consider Burgers' equation

$$u_t + \left(\frac{u^2}{2}\right)_x = 0, \quad x \in (0, 2)$$

subject to the initial condition $u(x, 0) = 0.5 + \sin(\pi x)$ and a periodic boundary condition. We compute the solution up to $T = \frac{0.5}{\pi}$ when the solution is still smooth and the exact solution can be computed using Newton's iteration. The error is listed in Table 5.1, which shows the convergence of the second order for P^1 elements and the third order for P^2 elements for the moving mesh DG method.

To test the convergence of the method for discontinuous solutions, we compute Burgers' equation up to $T = \frac{1.5}{\pi}$ when the shock appears. The error of L^1 is listed in Table 5.3. The results indicate that the computed solution is convergent although the convergence order decreases to one for both P^1 and P^2 elements.

To see how the smoothness of the mesh affects the accuracy of the method, we list the L^1 error in Tables 5.2 and 5.3 for the solutions computed with different numbers of sweeps of the low-pass filter applied to the metric tensor. One can see that the results are almost the same. In Table 5.3 where the solution is discontinuous, the error is slightly worse for the case with more sweeps. This is because more sweeps lead to smoother meshes with less concentration near the shock. Overall, the results show that the accuracy of the method is not sensitive to the smoothness of the mesh, which is in contrast with the situation for the moving mesh finite difference WENO method [44].

Table 5.2
Example 5.1: Solution error in L^1 norm with periodic boundary conditions and $T = \frac{0.5}{\pi}$. Various numbers of sweeps of a low-pass filter have been applied to the smoothing of the metric tensor.

k	Sweeps\\N	160	320	640	1280	2560	5120
1	3	1.561e-5	4.107e-6	1.061e-6	2.705e-7	6.856e-8	1.733e-8
	Order		1.93	1.95	1.97	1.98	1.98
	30	1.569e-5	4.115e-6	1.062e-6	2.705e-7	6.856e-8	1.733e-8
	Order		1.93	1.95	1.97	1.98	1.98
	100	1.585e-5	4.133e-6	1.064e-6	2.707e-7	6.859e-8	1.733e-8
	Order		1.94	1.96	1.97	1.98	1.98
2	3	1.267e-7	1.735e-8	2.216e-9	2.619e-10	2.927e-11	3.223e-12
	Order		2.87	2.97	3.08	3.16	3.18
	30	1.033e-7	1.457e-8	1.941e-9	2.403e-10	2.800e-11	3.168e-12
	Order		2.83	2.91	3.01	3.10	3.14
	100	9.208e-8	1.309e-8	1.779e-9	2.259e-10	2.699e-11	3.115e-12
	Order		2.81	2.88	2.98	3.06	3.12

Table 5.3
Example 5.1: Solution error in L^1 norm with periodic boundary conditions and $T = \frac{1.5}{\pi}$. Various numbers of sweeps of a low-pass filter have been applied to the smoothing of the metric tensor.

k	Sweeps\\N	20	40	80	160	320	640	1280
1	3	4.187e-3	9.542e-4	2.662e-4	9.171e-5	3.843e-5	1.876e-5	1.015e-5
	Order		2.13	1.84	1.54	1.25	1.03	0.89
	30	1.133e-2	1.803e-3	3.658e-4	1.195e-4	5.016e-5	2.450e-5	1.311e-5
	Order		2.65	2.30	1.61	1.25	1.03	0.90
	100	1.620e-2	3.615e-3	6.444e-4	1.753e-4	7.009e-5	3.347e-5	1.762e-5
	Order		2.16	2.49	1.88	1.32	1.07	0.93
2	3	1.336e-3	2.308e-4	8.626e-5	4.070e-5	2.198e-5	1.304e-5	7.809e-6
	Order		2.53	1.42	1.08	0.89	0.75	0.74
	30	5.134e-3	7.975e-4	1.586e-4	6.101e-5	3.074e-5	1.725e-5	9.943e-6
	Order		2.69	2.33	1.38	0.99	0.83	0.79
	100	6.611e-3	2.133e-3	3.631e-4	1.042e-4	4.599e-5	2.398e-5	1.335e-5
	Order		1.63	2.55	1.80	1.18	0.94	0.84

Table 5.4
Example 5.1: Solution error at $T = 1$.

k	N	20	40	80	160	320	640
1	L^1	4.595e-3	1.094e-3	4.654e-4	2.014e-4	8.903e-5	4.409e-5
	Order		2.07	1.23	1.21	1.18	1.01
2	L^1	4.037e-3	9.605e-4	4.041e-4	1.750e-4	7.880e-5	3.812e-5
	Order		2.07	1.25	1.21	1.15	1.05

We also compute Burgers' equation with a discontinuous initial condition

$$u(x,0) = \begin{cases} 1, \text{ for } & -1 \leqslant x \leqslant 0 \\ 0, \text{ for } & 0 < x \leqslant 1. \end{cases}$$

The error for both P^1 and P^2 elements at $T = 1$ is listed in Table 5.4. Once again, the results show that the computed solution is convergent at a rate of the first order for both P^1 and P^2 elements.

Example 5.2. To see the accuracy of the method for system problems, we compute the Euler equations,

$$\begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 + P \\ u(E+P) \end{pmatrix}_x = 0, \tag{5.1}$$

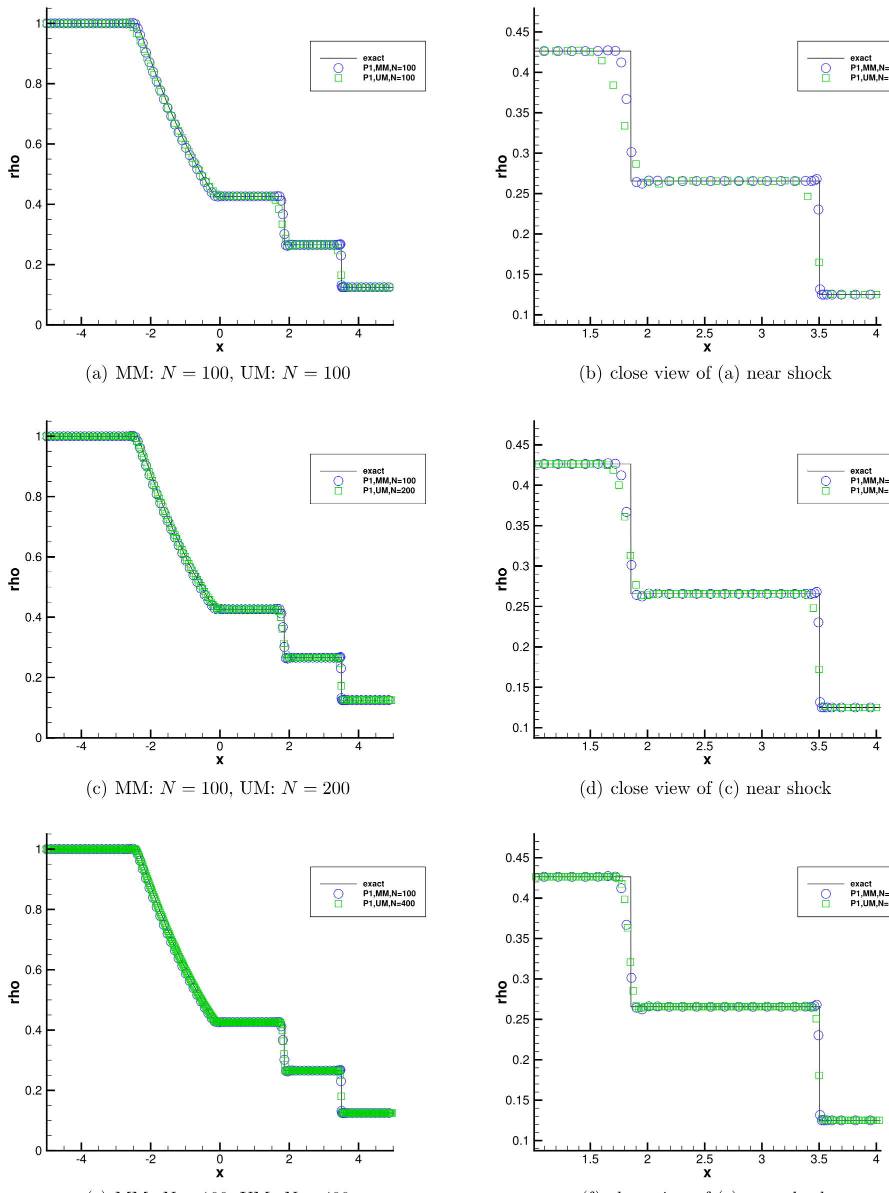
where ρ is the density, u is the velocity, E is the energy density, and P is the pressure. The equation of state is $E = \frac{P}{\gamma - 1} + \frac{1}{2}\rho u^2$ with $\gamma = 1.4$. The initial condition is

$$\rho(x,0) = 1 + 0.2\sin(\pi x), \quad u(x,0) = 1, \quad P(x,0) = 1,$$

and a periodic boundary condition is used. The exact solution for this problem is

Table 5.5
Example 5.2: Error in computed density for periodic boundary conditions and $T = 1.0$, $\beta = 100$.

k	N	10	20	40	80	160	320	640
1	L^1	5.629e-3	1.370e-3	3.334e-4	8.171e-5	1.998e-5	4.909e-6	1.230e-6
	Order		2.04	2.04	2.03	2.03	2.02	2.00
	L^2	5.241e-3	1.266e-3	3.070e-4	7.570e-5	1.877e-5	4.721e-6	1.220e-6
	Order		2.05	2.04	2.02	2.01	1.99	1.95
	L_∞	1.312e-2	3.110e-3	7.213e-4	1.764e-4	4.625e-5	1.235e-5	3.324e-6
	Order		2.08	2.11	2.03	1.93	1.90	1.89
2	L^1	3.420e-4	4.887e-5	6.837e-6	9.274e-7	1.191e-7	1.406e-8	1.530e-9
	Order		2.81	2.84	2.88	2.96	3.08	3.20
	L^2	3.821e-4	6.070e-5	9.058e-6	1.316e-6	1.791e-7	2.148e-8	2.211e-9
	Order		2.65	2.74	2.78	2.88	3.06	3.28
	L_∞	1.247e-3	2.363e-4	3.994e-5	6.651e-6	1.067e-6	1.504e-7	1.700e-8
	Order		2.40	2.56	2.59	2.64	2.83	3.15

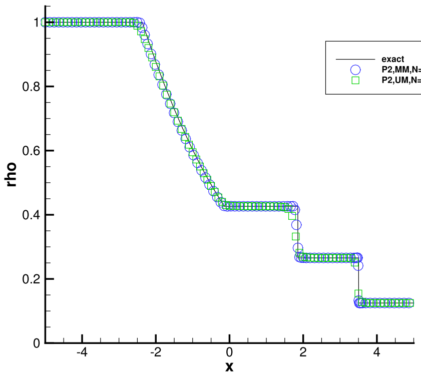


(a) MM: $N = 100$, UM: $N = 100$ (b) close view of (a) near shock

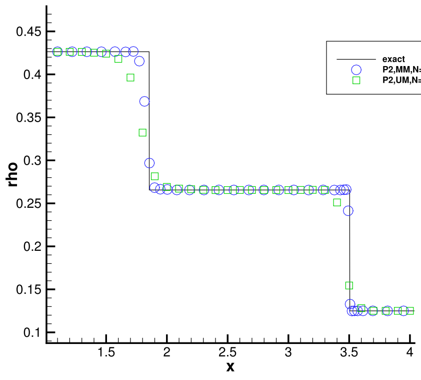
(c) MM: $N = 100$, UM: $N = 200$ (d) close view of (c) near shock

(e) MM: $N = 100$, UM: $N = 400$ (f) close view of (e) near shock

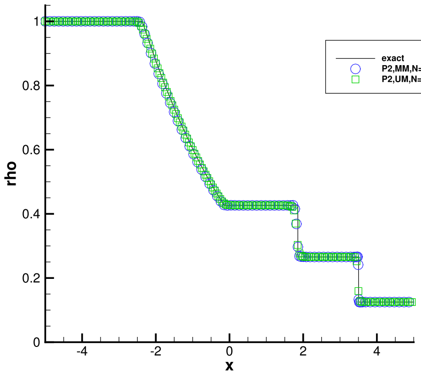
Fig. 5.1. Example 5.3 (Sod Problem). The moving mesh solution (density) with $N = 100$ is compared with the uniform mesh solutions with $N = 100$, 200, and 400. P^1 elements are used.



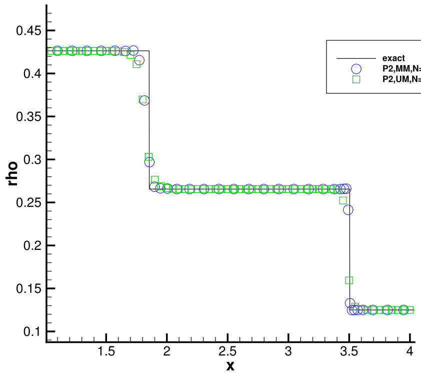
(a) MM: $N = 100$, UM: $N = 100$



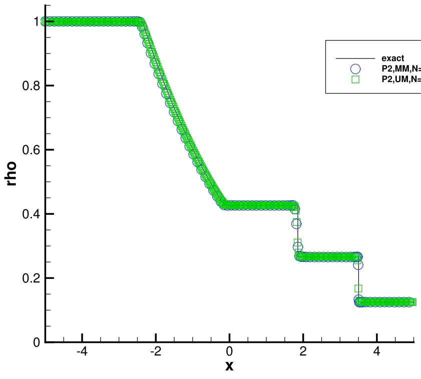
(b) close view of (a) near shock



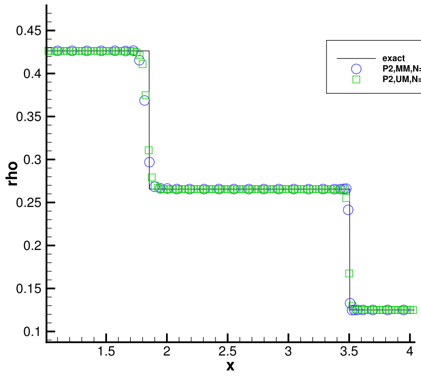
(c) MM: $N = 100$, UM: $N = 200$



(d) close view of (c) near shock



(e) MM: $N = 100$, UM: $N = 400$



(f) close view of (e) near shock

Fig. 5.2. Example 5.3 (Sod Problem). The moving mesh solution (density) with $N = 100$ is compared with the uniform mesh solutions with $N = 100$, 200, and 400. P^2 elements are used.

$$\rho(x,t) = 1 + 0.2\sin(\pi(x-t)), \quad u(x,t) = 1, \quad P(x,t) = 1.$$

The final time is $T = 1.0$. The parameter β in (3.2) is set to be 100. The error in computed density is listed in Table 5.5. From the table one can see that the $(k+1)^{\text{th}}$ order of accuracy of the scheme is achieved for this nonlinear system.

Example 5.3. This example is the Sod problem of the Euler equation (5.1) subject to the inflow/outflow boundary condition and a Riemann initial condition

$$(\rho, u, p) = \begin{cases} (1, 0, 1), & \text{for} \quad x < 0 \\ (0.125, 0, 0.1), & \text{for} \quad x > 0. \end{cases}$$

The computational domain is $(-5, 5)$ and the final time is $T = 2.0$.

The moving mesh DG solution (density) obtained with $N = 100$ is compared with the uniform mesh solutions obtained with $N = 100$, 200, and 400 in Figs. 5.1 and 5.2. One can see that the moving mesh solutions are more accurate than

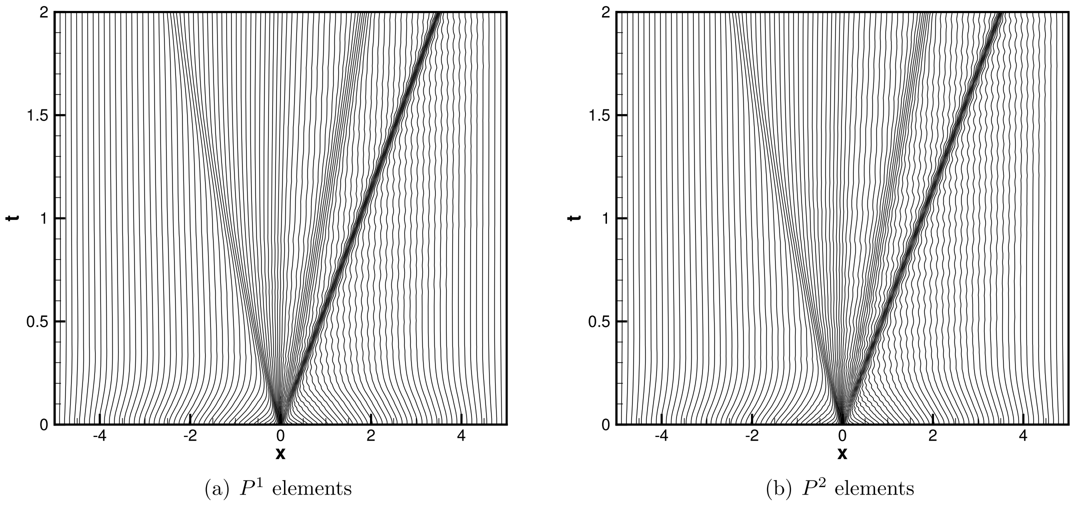


Fig. 5.3. Example 5.3 (Sod Problem). The trajectories of a moving mesh with $N = 100$ are plotted.

the uniform mesh solutions for the same number of points and comparable with those with $N = 400$ for both P^1 and P^2 elements.

The trajectories of a moving mesh are plotted in Fig. 5.3. From the figure, one can observe that the points are concentrated at $x = 0$ initially where the initial condition is discontinuous. As time evolves, the moving mesh can capture not only the shock but also the contact discontinuity well. In addition, the points are also clustered at the front and the tail of the rarefaction since the Hessian is used in the computation of the metric tensor $\mathbb{M}$.

Small oscillations near the shock are visible in Fig. 5.1. Interestingly, when the entropy $S = \log(\frac{\rho^\gamma}{P})$ is used for computing the metric tensor $\mathbb{M}$, more mesh points are concentrated in the shock region (see the mesh trajectories in Fig. 5.6) and the oscillations disappear (Figs. 5.4 and 5.5). However, less points are now available to resolve the rarefaction wave.

Example 5.4. In this example we consider the Lax problem of the Euler equations (5.1) with the initial condition

$$(\rho, u, p) = \begin{cases} (0.445, 0.698, 3.528), & \text{for} \quad x < 0 \\ (0.5, 0, 0.571), & \text{for} \quad x > 0 \end{cases}$$

and the inflow/outflow boundary condition. The computational domain is $(-5, 5)$ and the integration is stopped at $T = 1.3$. The computed density is plotted in Figs. 5.7 and 5.8 and the trajectories of the moving mesh are shown in Fig. 5.9.

From the figures, one can see that the moving mesh solution (density) obtained with $N = 100$ is more accurate than that obtained with a uniform mesh with $N = 400$ for both P^1 and P^2 elements in this example. From the trajectories, one can see that the points are concentrated at the shock, contact discontinuity and the rarefaction the same as for the Sod problem.

The moving mesh DG (MMDG) solution (density) obtained with $N = 100$ is compared with the fifth-order moving mesh WENO (MMWENO5 of [44]) solution obtained with $N = 200$ in Fig. 5.10. One can see that MMDG-P^1 and -P^2 solutions with $N = 100$ are more accurate overall than the MMWENO5 solution with $N = 200$. Moreover, by close examination one may see that oscillations are visible at the top-right corner of the computed shock by MMDG but at the top-left corner of the computed shock by MMWENO5.

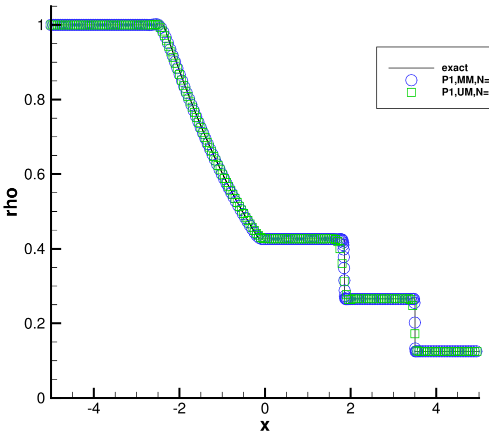
The oscillations can be eliminated if the entropy is used to compute the monitor function. The results of the density for P^1 and P^2 are shown in Figs. 5.11 and 5.12, respectively. The trajectories of the moving mesh are shown in Fig. 5.13, where one can observe that the points are not concentrated at the rarefaction.

Example 5.5. The Shu-Osher problem [37] is considered in this example, which contains both shocks and complex smooth region structures. We solve the Euler equations (5.1) with a moving shock (Mach = 3) interacting with a sine wave in density. The initial condition is

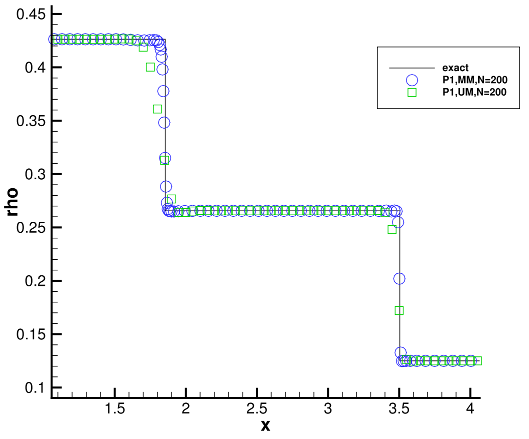
$$(\rho, u, p) = \begin{cases} (3.857143, 2.629369, 10.333333), & \text{for} \quad x < -4 \\ (1 + 0.2\sin(5x), 0, 1), & \text{for} \quad x > -4. \end{cases}$$

The physical domain is taken as $(-5, 5)$ in the computation. The computed density is shown at $T = 1.8$ against an “exact solution” obtained by a fifth-order finite volume WENO scheme with 10,000 uniform points.

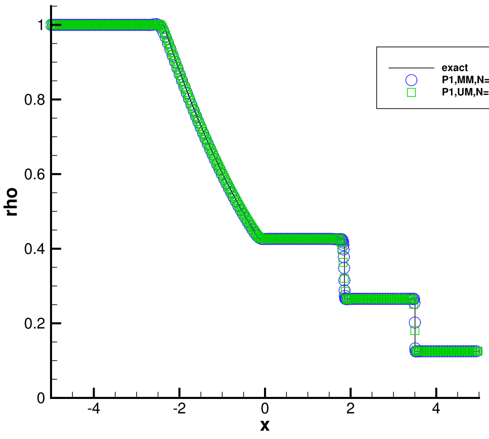
The trajectories are plotted in Fig. 5.16. The moving mesh solution (density) with $N = 150$ is compared with the uniform mesh solutions with $N = 150$, 400, and 600 in Figs. 5.14 and 5.15. From the figures, one can observe that for the same



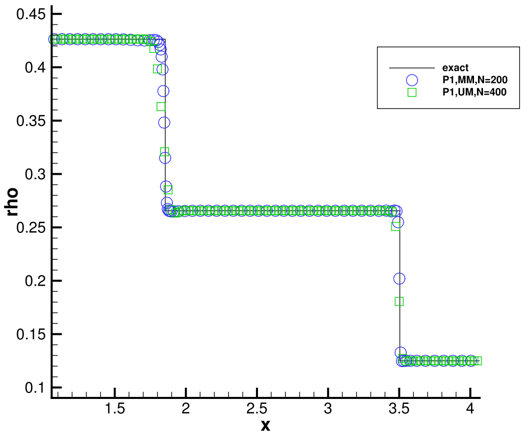
(a) MM: $N = 200$, UM: $N = 200$



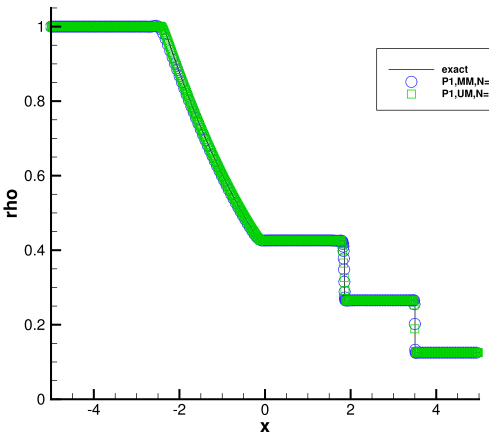
(b) close view of (a) near shock



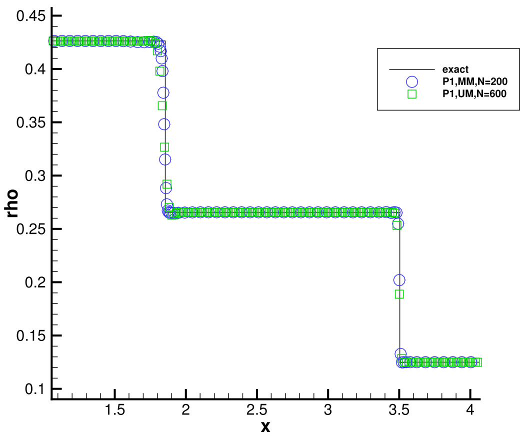
(c) MM: $N = 200$, UM: $N = 400$



(d) close view of (c) near shock

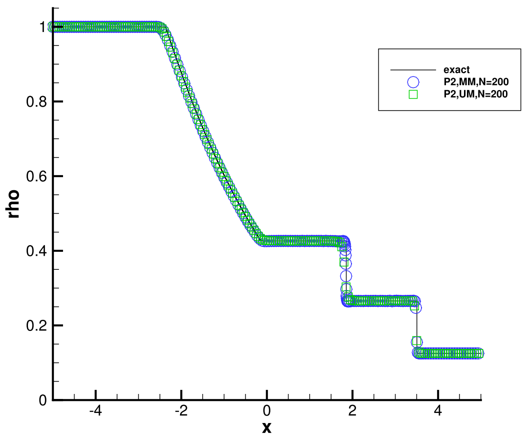


(e) MM: $N = 200$, UM: $N = 600$

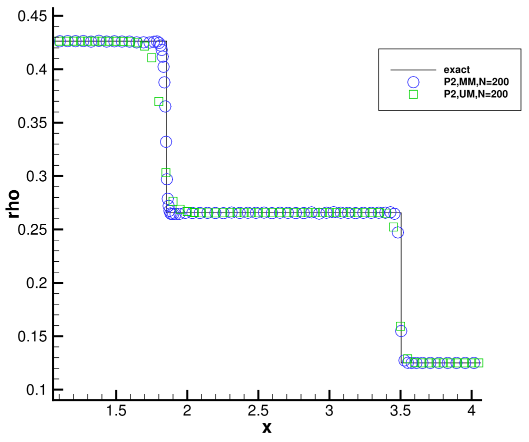


(f) close view of (e) near shock

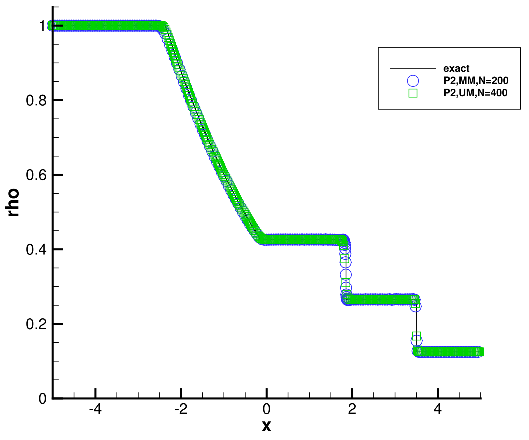
Fig. 5.4. Example 5.3 (Sod Problem). The moving mesh solution (density) with $N = 200$ is compared with the uniform mesh solutions with $N = 200$, 400, and 600. P^1 elements are used.



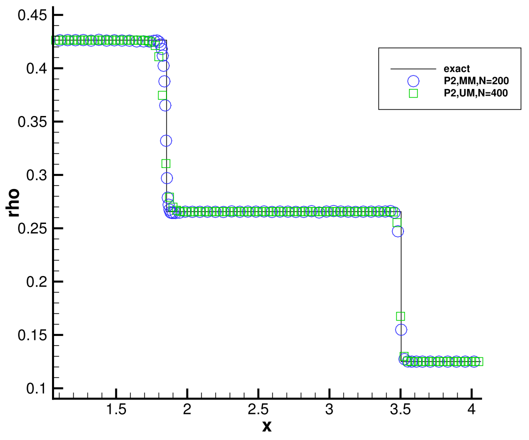
(a) MM: $N = 200$, UM: $N = 200$



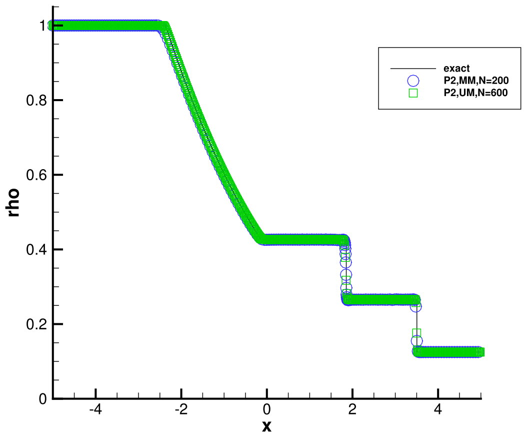
(b) close view of (a) near shock



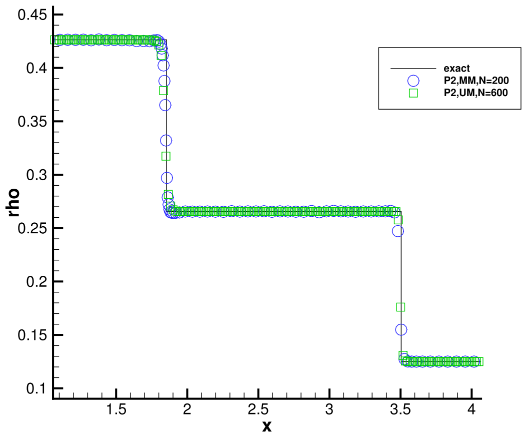
(c) MM: $N = 200$, UM: $N = 400$



(d) close view of (c) near shock



(e) MM: $N = 200$, UM: $N = 600$



(f) close view of (e) near shock

Fig. 5.5. Example 5.3 (Sod Problem). The moving mesh solution (density) with $N = 200$ is compared with the uniform mesh solutions with $N = 200$, 400, and 600. P^2 elements are used.

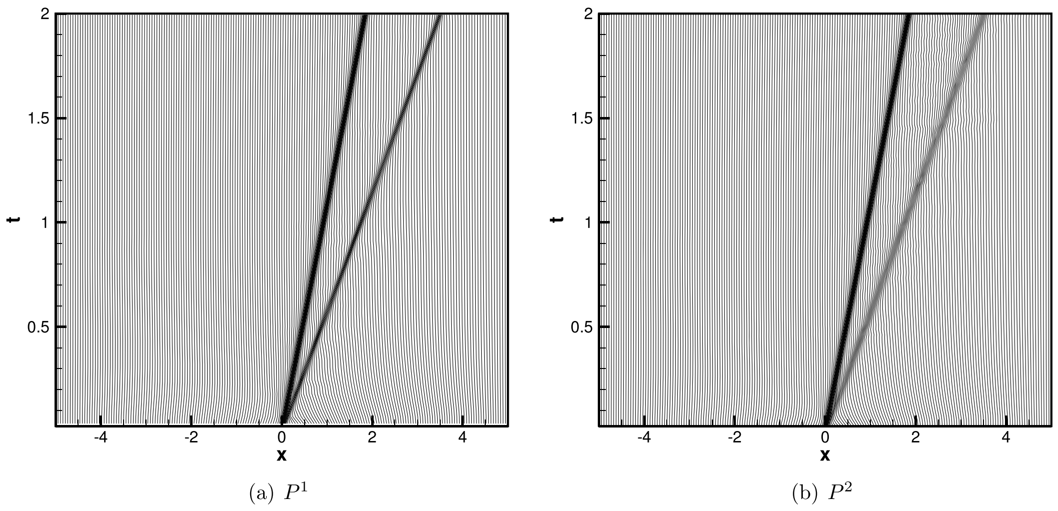


Fig. 5.6. Example 5.3 (Sod Problem). The trajectories of a moving mesh with $N = 200$ are plotted.

number of mesh points, the moving mesh results are clearly better than the uniform mesh ones. Moreover, the moving mesh solution with $N = 150$ is comparable with the uniform mesh solution with $N = 400$ when P^2 elements are used in Fig. 5.15 and is better than the uniform mesh solution with $N = 600$ for P^1 elements in Fig. 5.14. These results demonstrate the advantage of the moving mesh method in improving the computational accuracy.

The P^1 and P^2 MMDG solutions (density) are compared with the MMWENO5 solution obtained with $N = 200$ in Fig. 5.17. From the figure, we can see that the P^1 MMDG solution with $N = 150$ and the P^2 MMDG solution with $N = 100$ are more accurate than the MMWENO5 solution with $N = 200$, which indicates the proposed method can use less points than MMWENO5 to achieve the same level of error.

Example 5.6. We consider the interaction of blast waves of the Euler equations (5.1), which was first used by Woodward and Colella [43] as a test problem for various numerical schemes. The initial condition is given by

$$(\rho, u, p) = \begin{cases} (1.0, 0, 1000), & \text{for} \quad 0 \le x < 0.1 \\ (1.0, 0, 0.01), & \text{for} \quad 0.1 \le x < 0.9 \\ (1.0, 0, 100), & \text{for} \quad 0.9 \le x \le 1. \end{cases}$$

The physical domain is taken as $(0, 1)$ and a reflective boundary condition is applied to both ends. The results at time $T = 0.038$ are plotted against an "exact solution" computed by a fifth-order finite difference WENO scheme [24] with 81,920 uniform mesh points.

In this example, the parameter β in (3.2) is taken as 1. The trajectories of a moving mesh method are plotted in Fig. 5.20 which show that the two blast waves propagate in the right direction and finally collide. From Figs. 5.18 and 5.19, we can see that for the same number of mesh points, the solutions with a moving mesh are much better than those with a uniform mesh. In addition, the moving mesh solution obtained with $N = 150$ is comparable with the uniform mesh solution obtained with $N = 600$.

The MMDG solutions (density) obtained with $N = 300$ are compared with the MMWENO5 solution obtained with $N = 400$ in Fig. 5.21. From the figure, we can observe that the MMDG solutions with $N = 300$ are more accurate than the MMWENO5 solution with $N = 400$.

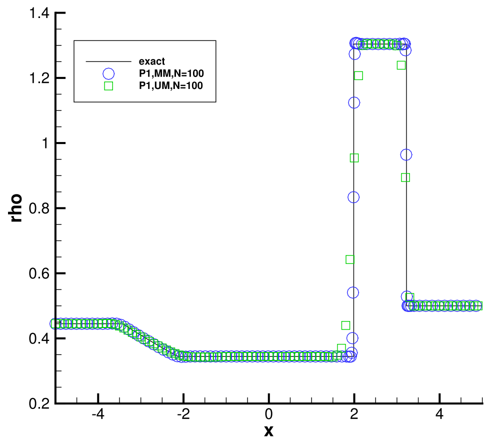
5.2. Two-dimensional examples

For two-dimensional examples, an initial triangular mesh is obtained by dividing any rectangular element into four triangular elements; see Fig. 5.22. A moving mesh associated with the initial mesh in Fig. 5.22 is denoted by $N = 10 \times 10 \times 4$. Other meshes will be denoted similarly.

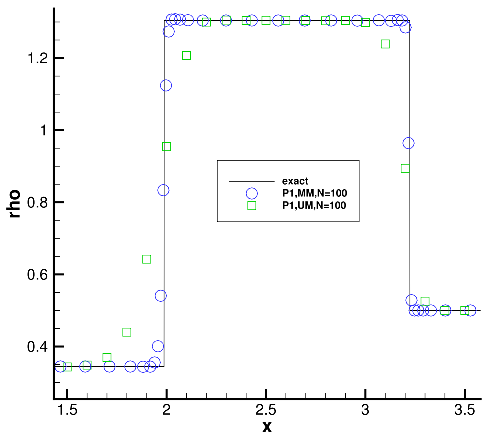
Example 5.7. We solve Burgers' equation in two dimensions,

$$u_t + \left(\frac{u^2}{2}\right)_x + \left(\frac{u^2}{2}\right)_y = 0, \quad (x, y) \in (0, 4) \times (0, 4)$$

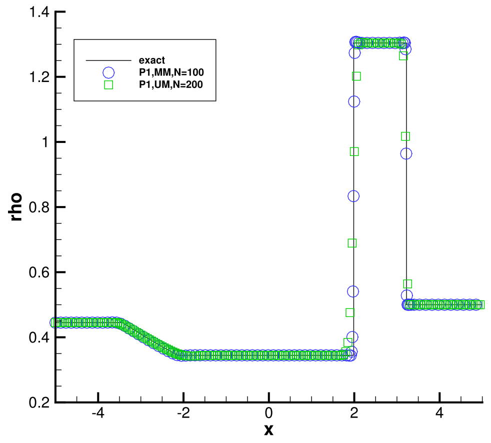
subject to the initial condition $u(x, y, 0) = 0.5 + \sin(\frac{\pi(x+y)}{2})$ and a periodic boundary condition in both directions.



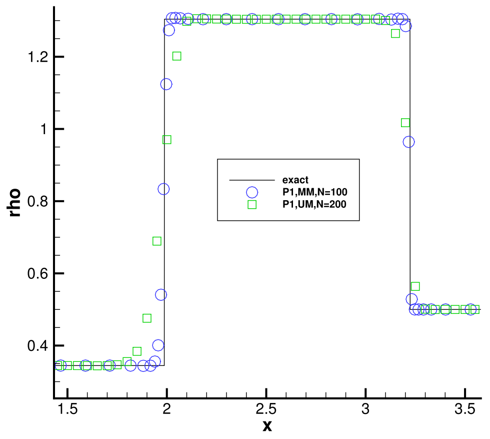
(a) MM: $N = 100$, UM: $N = 100$



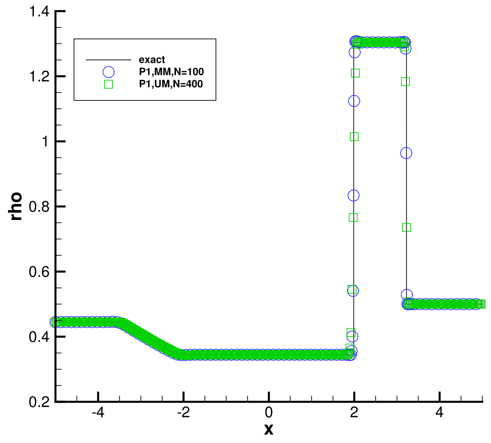
(b) close view of (a) near shock



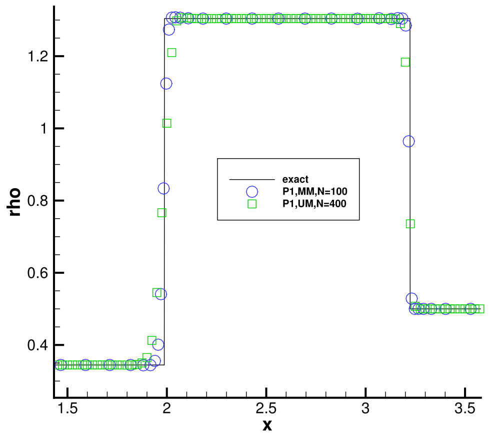
(c) MM: $N = 100$, UM: $N = 200$



(d) close view of (c) near shock

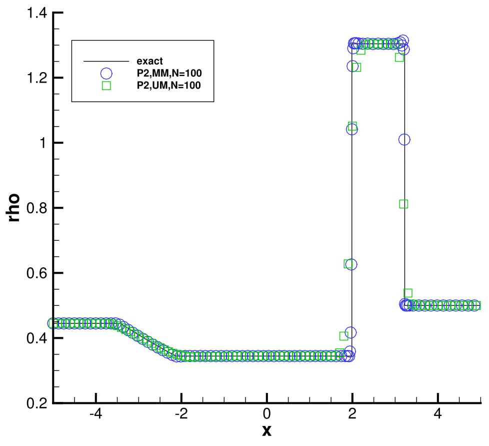


(e) MM: $N = 100$, UM: $N = 400$

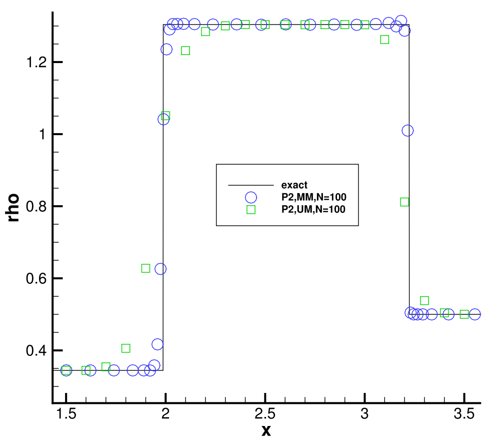


(f) close view of (e) near shock

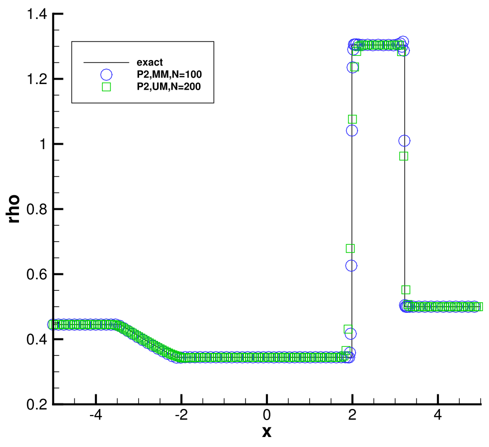
Fig. 5.7. Example 5.4 (Lax Problem). The moving mesh solution (density) with $N = 100$ is compared with the uniform mesh solutions with $N = 100$, 200, and 400. P^1 elements are used.



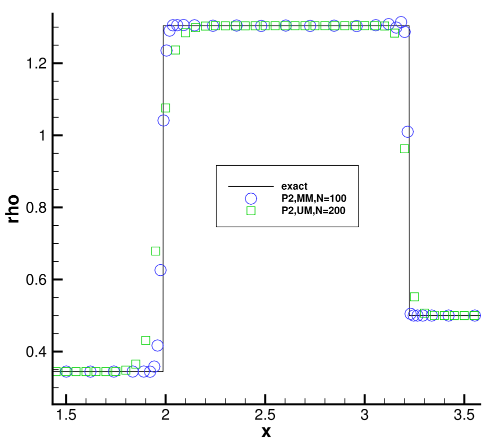
(a) MM: $N = 100$, UM: $N = 100$



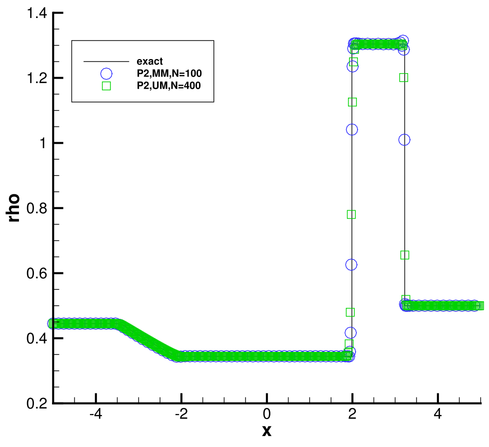
(b) close view of (a) near shock



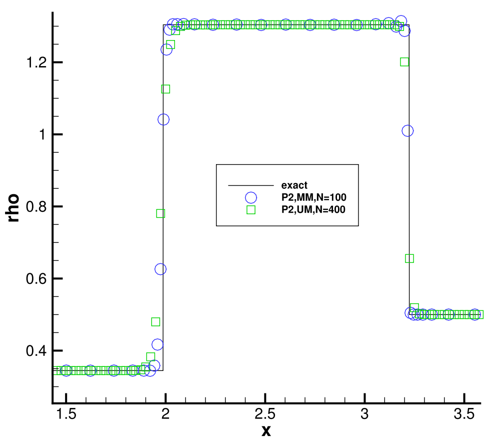
(c) MM: $N = 100$, UM: $N = 200$



(d) close view of (c) near shock



(e) MM: $N = 100$, UM: $N = 400$



(f) close view of (e) near shock

Fig. 5.8. Example 5.4 (Lax Problem). The moving mesh solution (density) with $N = 100$ is compared with the uniform mesh solutions with $N = 100$, 200, and 400. P^2 elements are used.

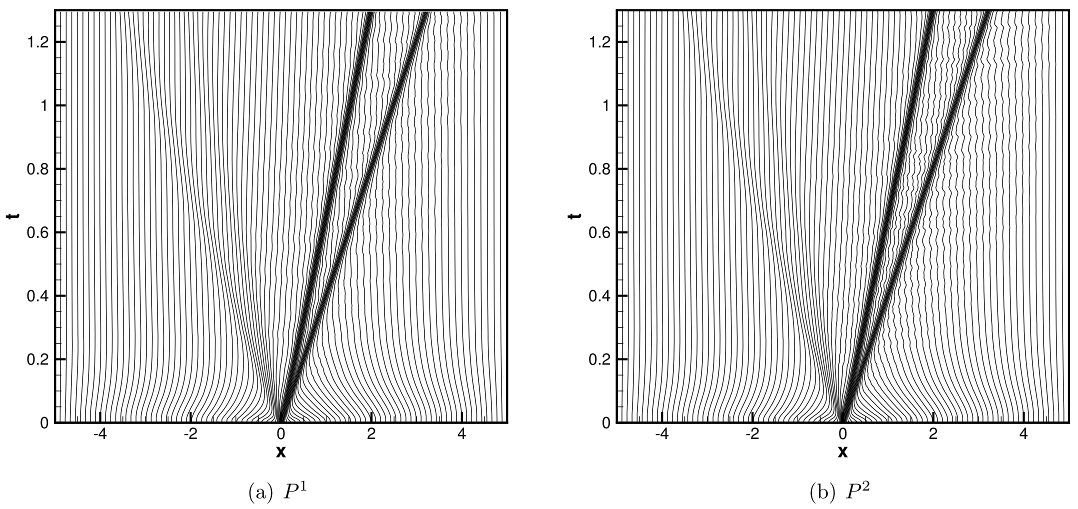


Fig. 5.9. Example 5.4 (Lax Problem). The trajectories of a moving mesh with $N = 100$ are plotted.

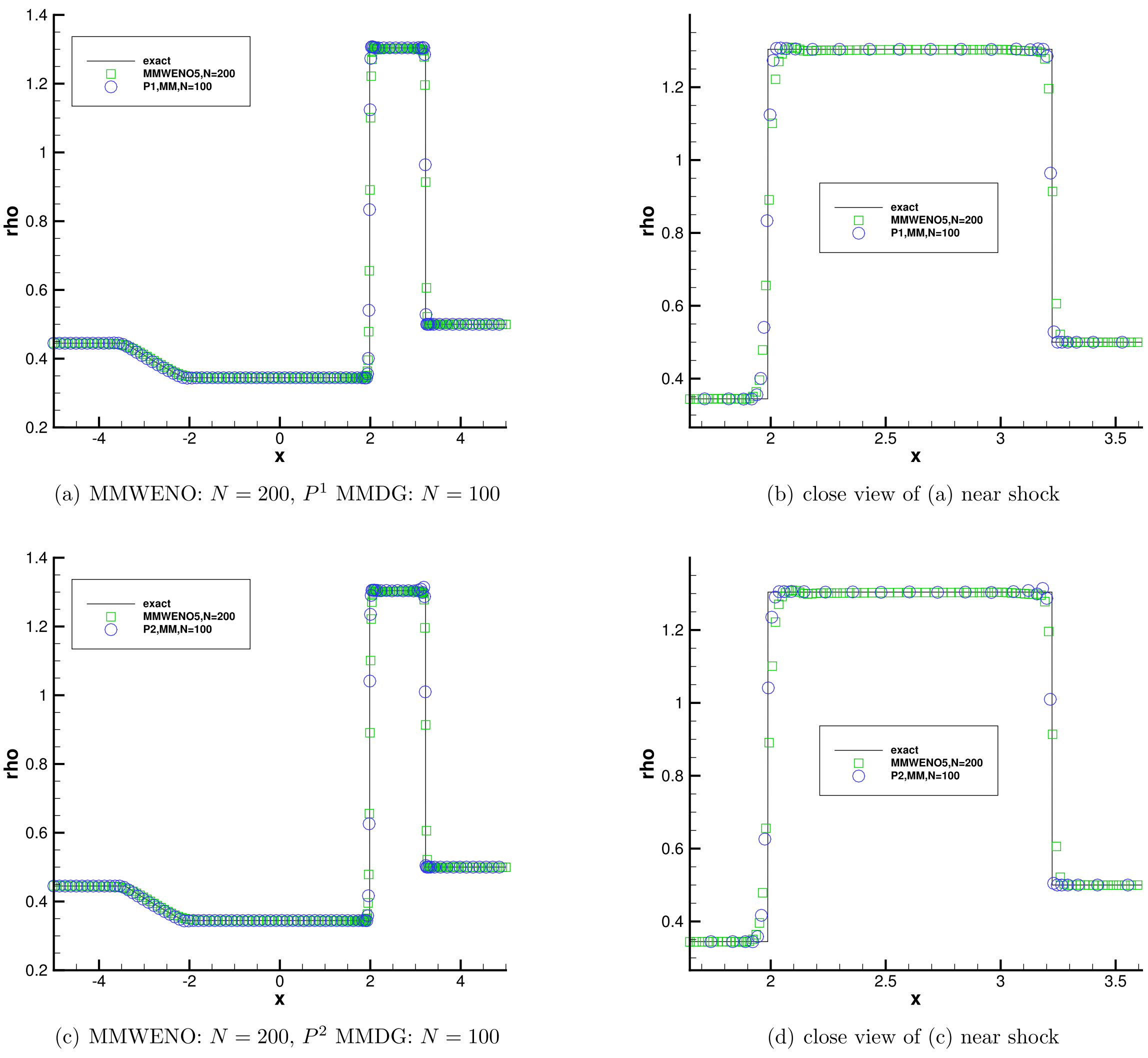
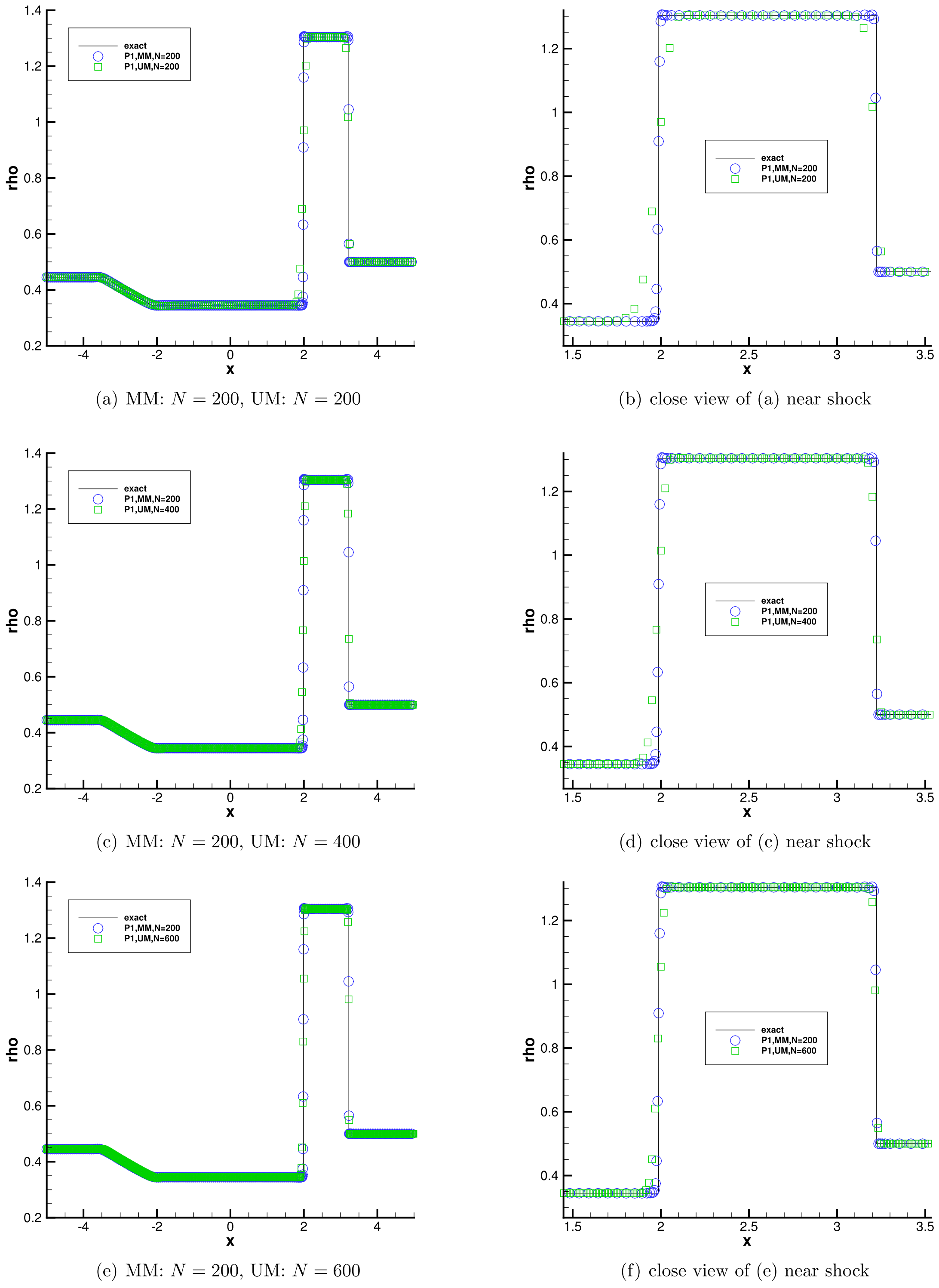


Fig. 5.10. Example 5.4 (Lax Problem). The moving mesh DG solution (density) with $N = 100$ is compared with the moving mesh WENO solution with $N = 200$.



(a) MM: $N = 200$, UM: $N = 200$

(b) close view of (a) near shock

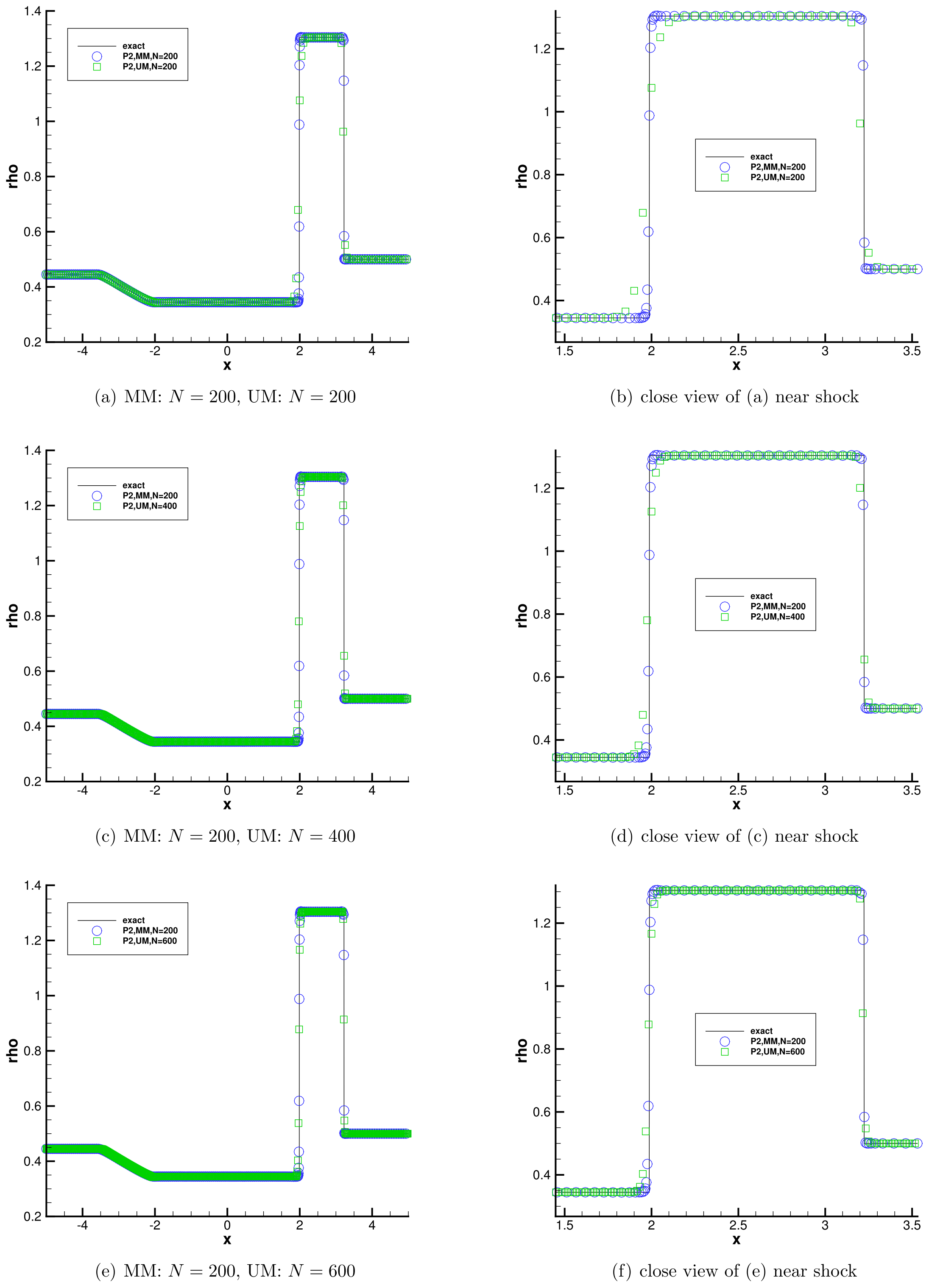
(c) MM: $N = 200$, UM: $N = 400$

(d) close view of (c) near shock

(e) MM: $N = 200$, UM: $N = 600$

(f) close view of (e) near shock

Fig. 5.11. Example 5.4 (Lax Problem). The moving mesh solution (density) with $N = 200$ is compared with the uniform mesh solutions with $N = 200$, 400, and 600. P^1 elements are used.



(a) MM: $N = 200$, UM: $N = 200$

(b) close view of (a) near shock

(c) MM: $N = 200$, UM: $N = 400$

(d) close view of (c) near shock

(e) MM: $N = 200$, UM: $N = 600$

(f) close view of (e) near shock

Fig. 5.12. Example 5.4 (Lax Problem). The moving mesh solution (density) with $N = 200$ is compared with the uniform mesh solutions with $N = 200$, 400, and 600. P^2 elements are used.

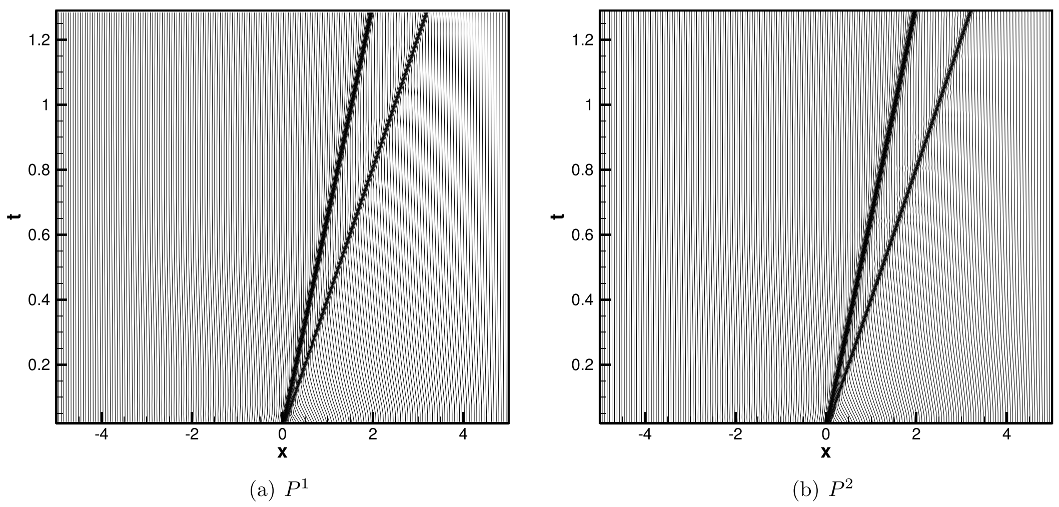


Fig. 5.13. Example 5.4 (Lax Problem). The trajectories of a moving mesh with $N = 200$ are plotted.

We compute the solution up to $T = \frac{0.5}{\pi}$ when the solution is still smooth. The error is listed in Table 5.6. The results show the anticipated $(k+1)$th order convergence of the moving mesh DG method when P^k elements ($k = 1$ and 2) are used.

To see how the smoothness of the mesh affects the accuracy of the method, we list the L^1 error in Tables 5.7 for the discontinuous solutions computed with $T = \frac{1.5}{\pi}$ and different numbers of sweeps of the low-pass filter applied to the metric tensor. As for Example 5.1, one can see that the error and convergence order are comparable although the error is slightly worse for the case with more sweeps.

To show the efficiency of the moving mesh method, we plot the L^1 error as a function of the CPU time for both uniform and moving meshes in Fig. 5.23 when $T = \frac{1.5}{\pi}$. The results are obtained on a single Intel Core i5 CPU with 2.0 GHz and 8.00 GB of RAM using Matlab R2018b. From the figure, one can see that the moving mesh method is more efficient than the uniform mesh method for small L^1 error whereas the latter is more efficient for large L^1 error.

Example 5.8. We solve the Euler equations

$$u_t + f_1(u)_x + f_2(u)_y \equiv \frac{\partial}{\partial t}\begin{pmatrix}\rho\\ \rho\mu\\ \rho\nu\\ E\end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix}\rho\mu\\ \rho\mu^2+P\\ \rho\mu\nu\\ \mu(E+P)\end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix}\rho\nu\\ \rho\mu\nu\\ \rho\nu^2+P\\ \nu(E+P)\end{pmatrix} = 0, \tag{5.2}$$

where ρ is the density, μ and ν are the velocity components in the x- and y-direction, respectively, E is the energy density, and P is the pressure. The equation of state is $E = \frac{P}{\gamma-1} + \frac{1}{2}\rho(\mu^2+\nu^2)$ with $\gamma = 1.4$. The initial condition is $\rho(x,y,0) = 1 + 0.2\sin(\frac{\pi(x+y)}{2})$, $\mu(x,y,0) = 0.7$, $\nu(x,y,0) = 0.3$, $p(x,y,0) = 1$ and a periodic boundary condition is applied in both directions.

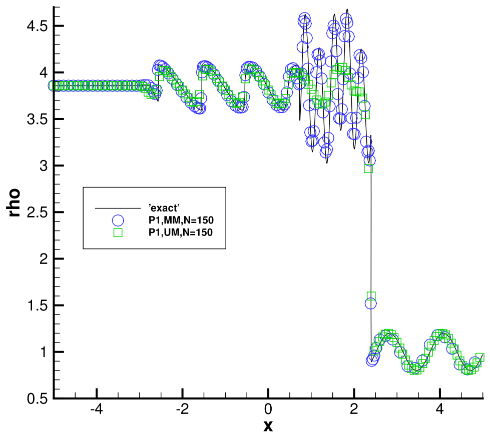
The computational domain is $(0,2) \times (0,2)$ and the final time is $T = 1$. We take the parameter β in (3.2) as 100. The results in Table 5.8 show the convergence order of the second order for $k = 1$ and the third order for $k = 2$ for the moving mesh DG method for the Euler system in two dimensions.

Example 5.9. This is the double Mach reflection problem [43]. We solve the Euler equations (5.2) in a computational domain of $(0,4) \times (0,1)$. The initial condition is given by

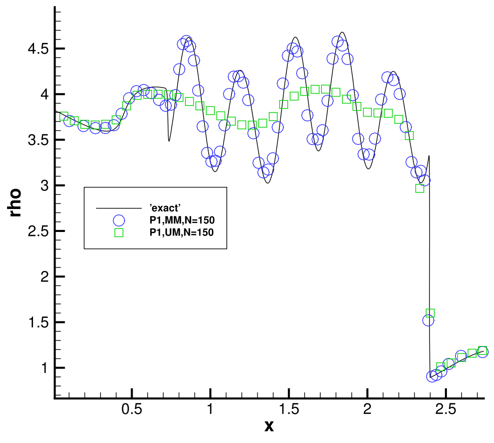
$$u = \begin{cases} (8, 57.1597, -33.0012, 563.544)^T, & \text{for} \quad y \geq h(x,0) \\ (1.4, 0, 0, 2.5)^T, & \text{otherwise} \end{cases}$$

where $h(x,t) = \sqrt{3}(x - \frac{1}{6}) - 20t$. The exact post shock condition is imposed from 0 to $\frac{1}{6}$ at the bottom while the reflection boundary condition for the rest of the bottom boundary. At the top, the boundary condition is the values that describe the exact motion of the Mach 10 shock. On the left and right boundaries, the inflow and outflow boundary conditions are used, respectively. The final time is $T = 0.2$.

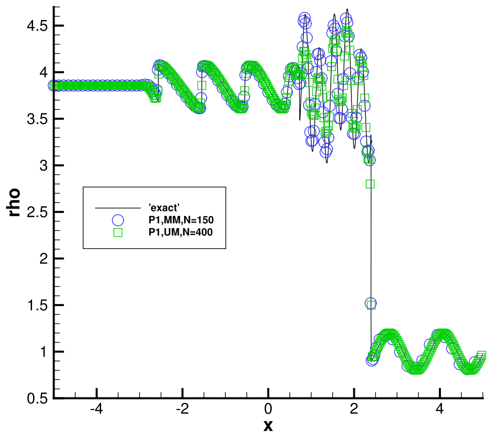
The density contours are shown in Figs. 5.24–5.31 on $(0,3) \times (0,1)$. From these figures, one can see that more elements are concentrated in the regions where the shock and the complex structures are located. The result of the moving mesh method with $N = 120 \times 30 \times 4$ is comparable with that obtained with 1.5 times more uniform mesh points. The same is observed with a moving mesh of $N = 240 \times 60 \times 4$.



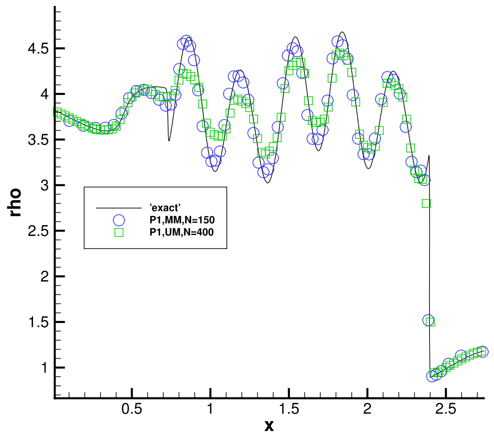
(a) MM: $N = 150$, UM: $N = 150$



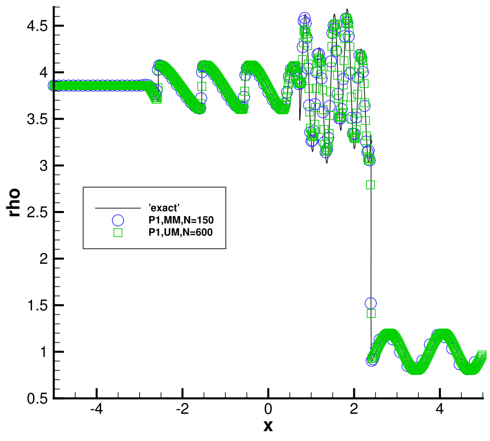
(b) close view of (a)



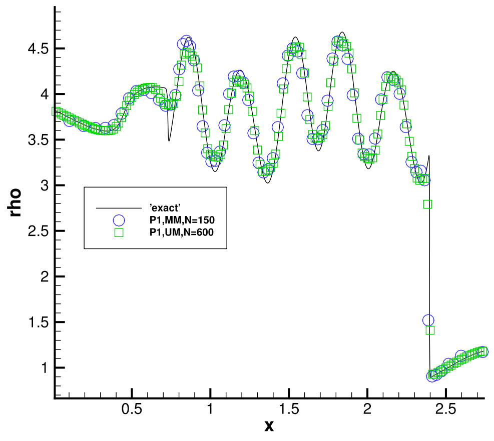
(c) MM: $N = 150$, UM: $N = 400$



(d) close view of (c)

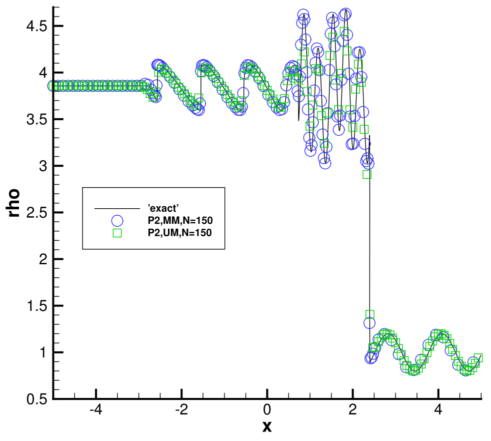


(e) MM: $N = 150$, UM: $N = 600$

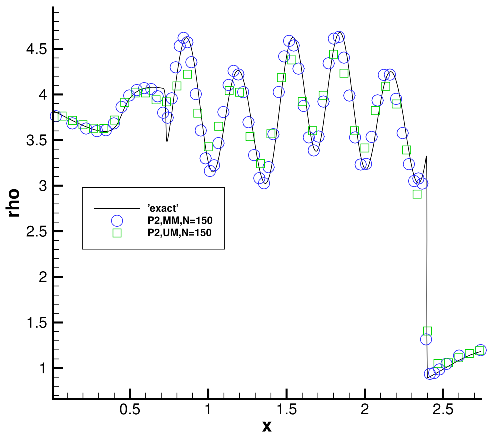


(f) close view of (e)

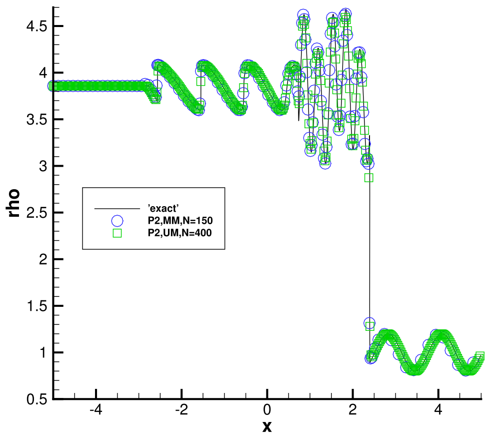
Fig. 5.14. Example 5.5 (Shu-Osher Problem). The moving mesh solution (density) with $N = 150$ is compared with the uniform mesh solutions with $N = 150$, 400, and 600. P^1 elements are used.



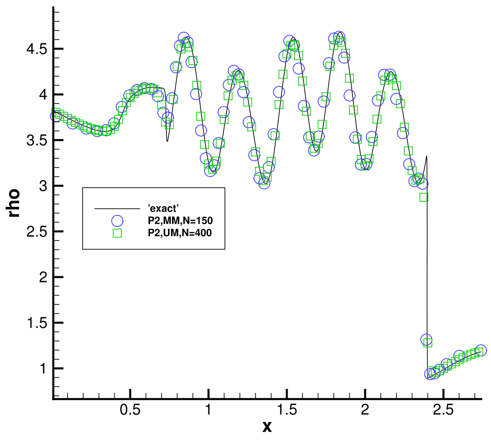
(a) MM: $N = 150$, UM: $N = 150$



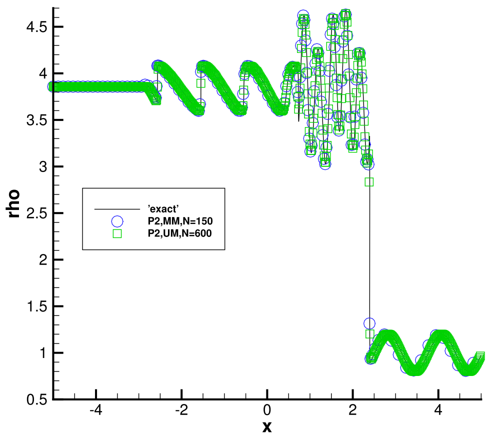
(b) close view of (a)



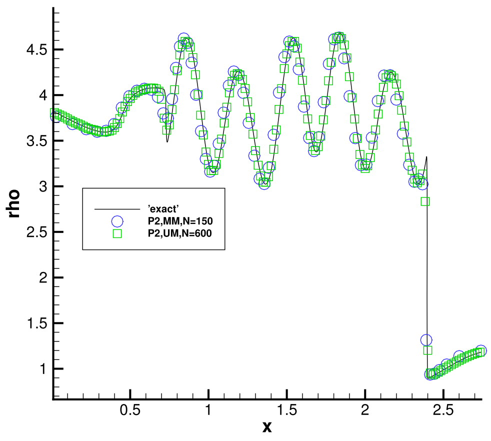
(c) MM: $N = 150$, UM: $N = 400$



(d) close view of (c)

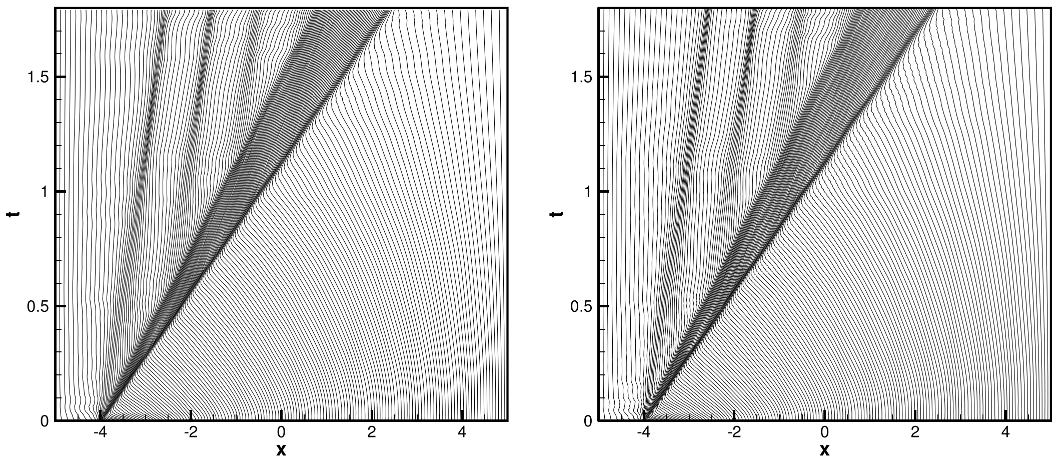


(e) MM: $N = 150$, UM: $N = 600$



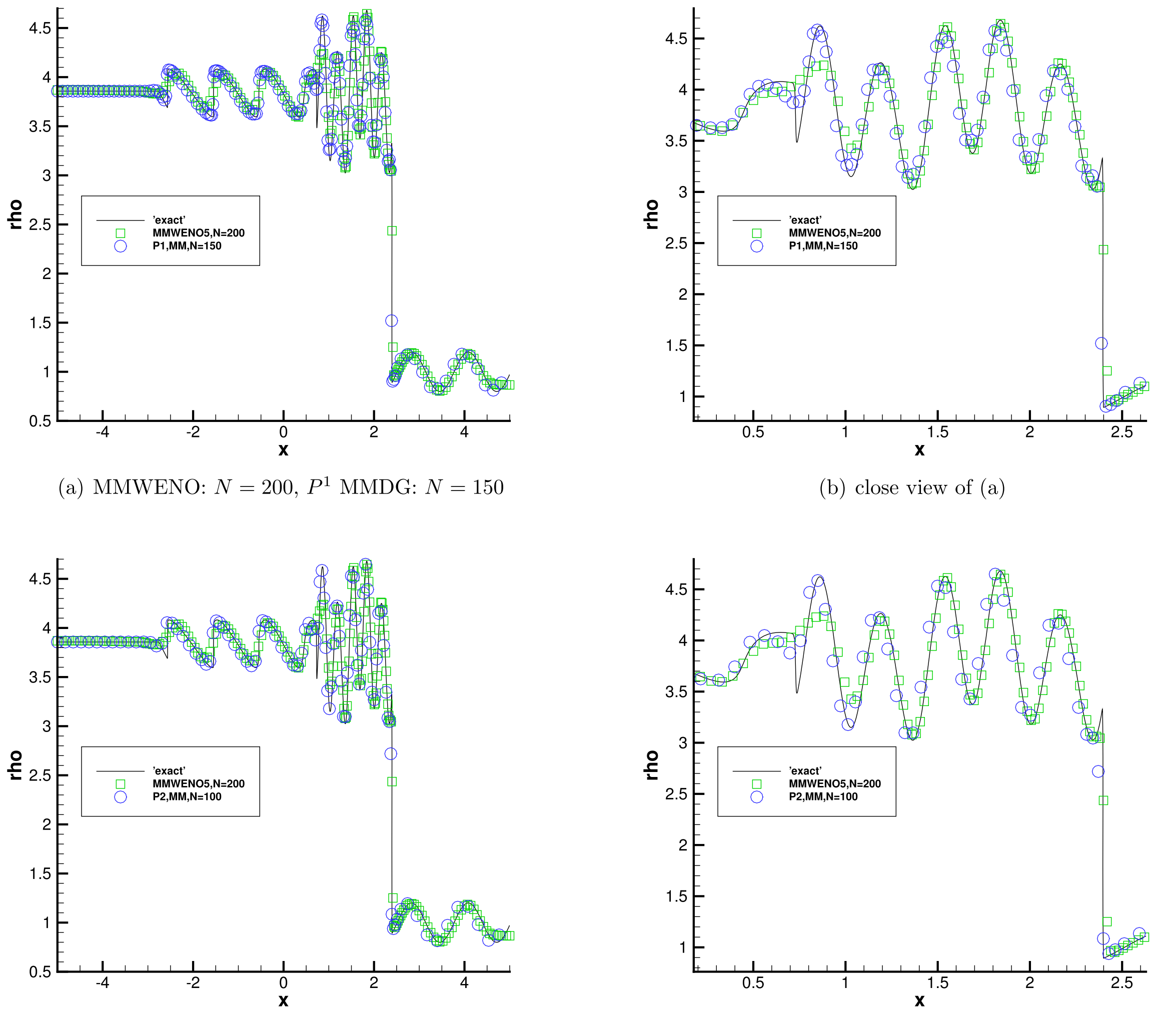
(f) close view of (e)

Fig. 5.15. Example 5.5 (Shu-Osher Problem). The moving mesh solution (density) with $N = 150$ is compared with the uniform mesh solutions with $N = 150$, 400, and 600. P^2 elements are used.



(a) P^1 elements (b) P^2 elements

Fig. 5.16. Example 5.5 (Shu-Osher Problem). The trajectories of a moving mesh with $N = 150$ are plotted.



(a) MMWENO: $N = 200$, P^1 MMDG: $N = 150$

(b) close view of (a)

(c) MMWENO: $N = 200$, P^2 MMDG: $N = 100$

(d) close view of (c)

Fig. 5.17. Example 5.5 (Shu-Osher Problem). The moving mesh DG solution (density) is compared with the moving mesh WENO solutions with $N = 200$.

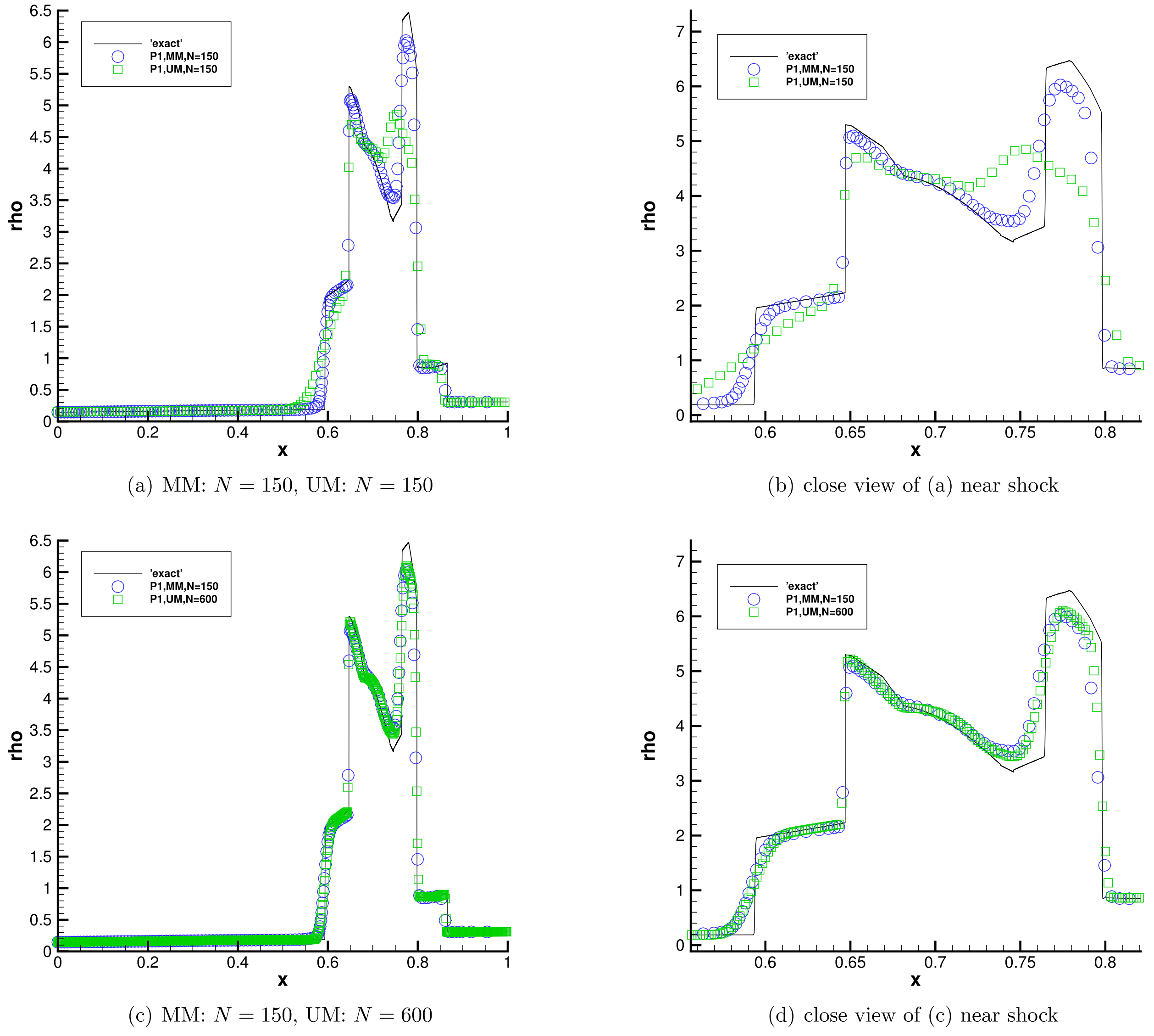


Fig. 5.18. Example 5.6 (Blast Wave Problem). The moving mesh solution (density) with $N = 150$ is compared with the uniform mesh solutions with $N = 150$ and 600. P^1 elements are used.

Example 5.10. The last example is the forward step problem [43]. We solve the Euler equations (5.2) in a computational domain of $(0, 3) \times (0, 1)$. The problem is set up as follows: the wind tunnel is 1 unit wide and 3 units long. The step is 0.2 units high and is located 0.6 units from the left-hand end of the tunnel. The problem is initialized by a right-going Mach 3 flow, namely,

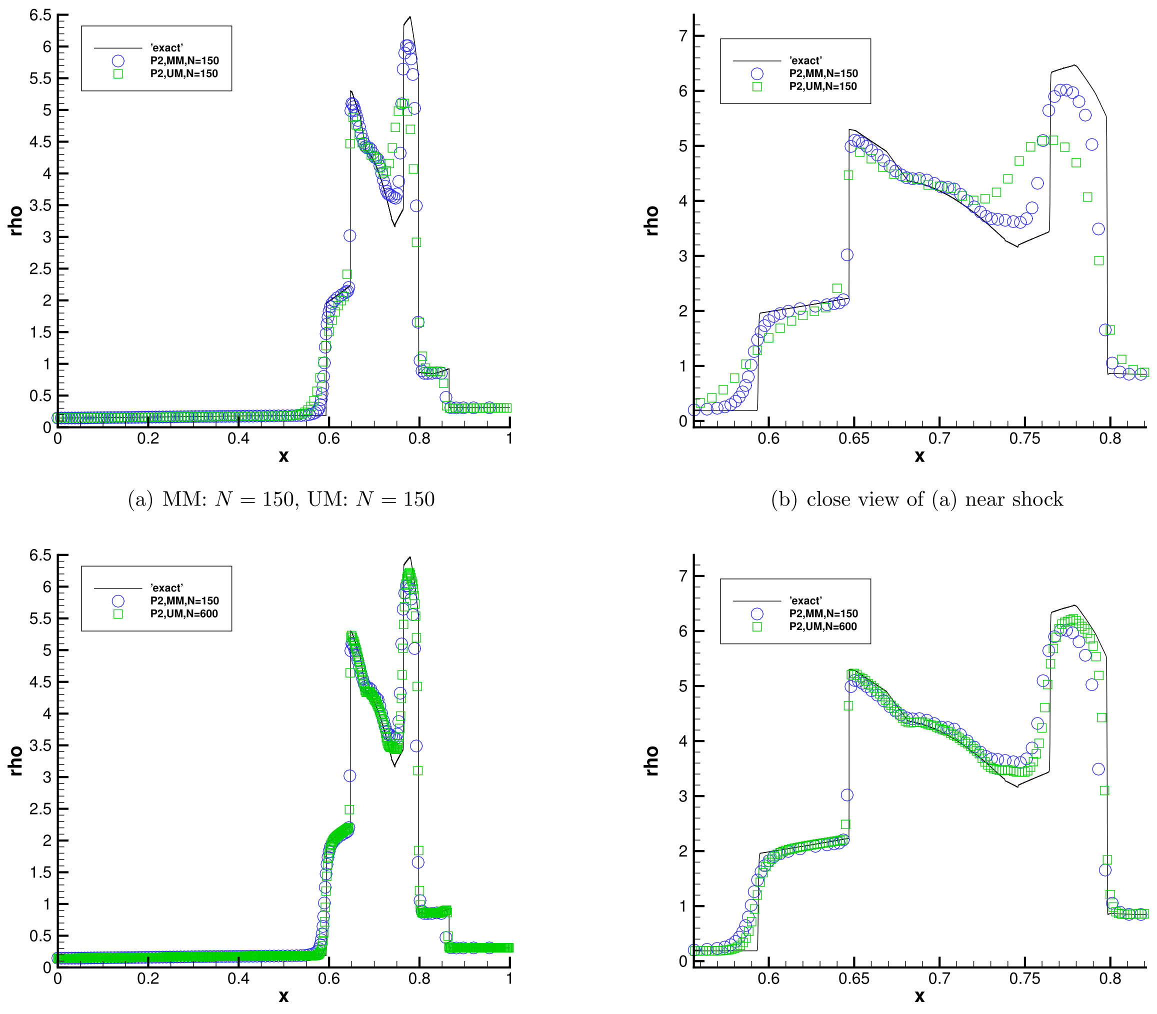
$$(\rho, \mu, \nu, p) = (1.4, 3, 0, 1).$$

Reflective boundary conditions are employed along the wall of the tunnel and inflow and outflow boundary conditions are applied at the entrance and exit, respectively. The final time is $T = 4$.

The density contours from 0.32 to 6.15 are plotted in Figs. 5.32 and 5.33. One can clearly see that the numerical solutions with the moving mesh method have better resolution than those with a uniform mesh of the same number of points.

6. Conclusions

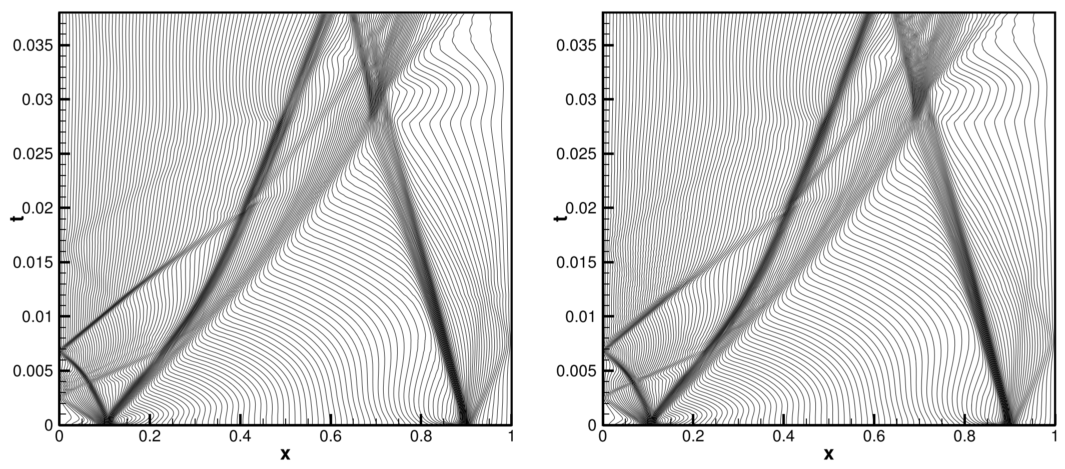
In the previous sections we have presented a moving mesh DG method for the numerical solution of hyperbolic conservation laws. The mesh is moved using the MMPDE moving mesh strategy where the nodal mesh velocities are defined as



(a) MM: $N = 150$, UM: $N = 150$ (b) close view of (a) near shock

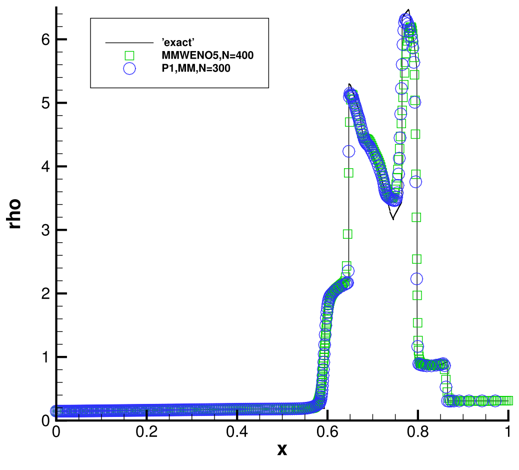
(c) MM: $N = 150$, UM: $N = 600$ (d) close view of (c) near shock

Fig. 5.19. Example 5.6 (Blast Wave Problem). The moving mesh solution (density) with $N = 150$ is compared with the uniform mesh solutions with $N = 150$ and 600. P^2 elements are used.

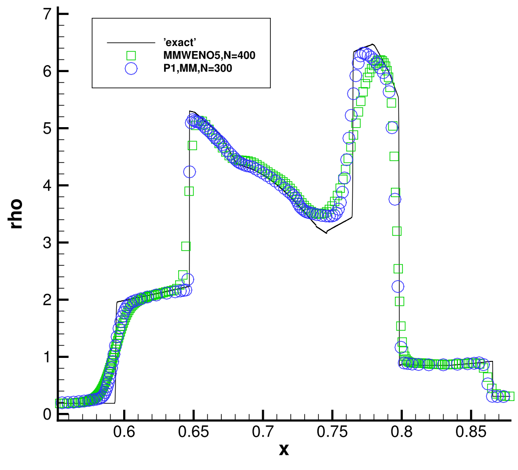


(a) P^1 elements (b) P^2 elements

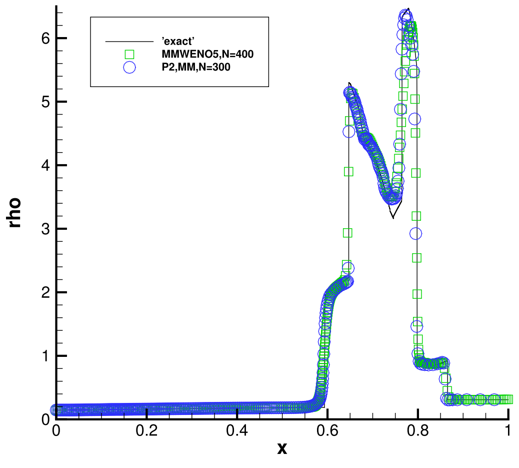
Fig. 5.20. Example 5.6 (Blast Wave Problem). The trajectories of a moving mesh with $N = 150$ are plotted.



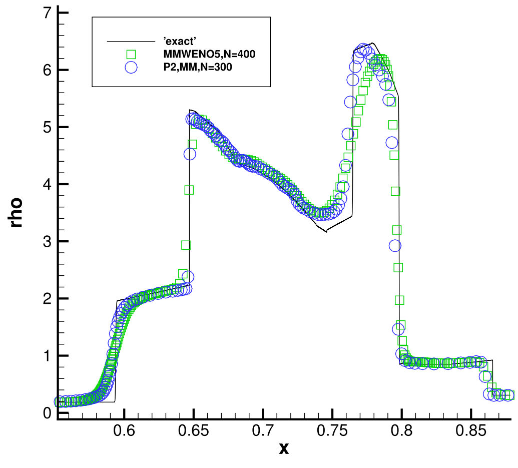
(a) MMWENO: $N = 400$, P^1 MMDG: $N = 300$



(b) close view of (a) near shock



(c) MMWENO: $N = 400$, P^2 MMDG: $N = 300$



(d) close view of (c) near shock

Fig. 5.21. Example 5.5 (Blast Wave Problem). The moving mesh DG solutions (density) with $N = 300$ are compared with the moving mesh WENO solution with $N = 400$.

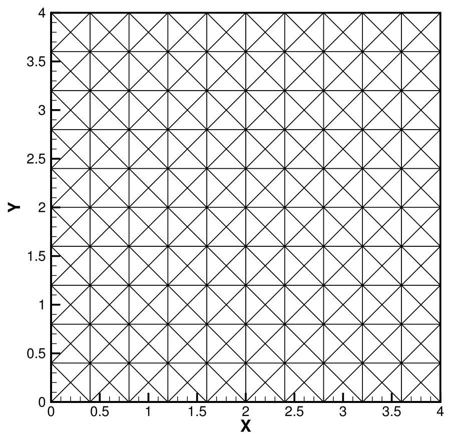


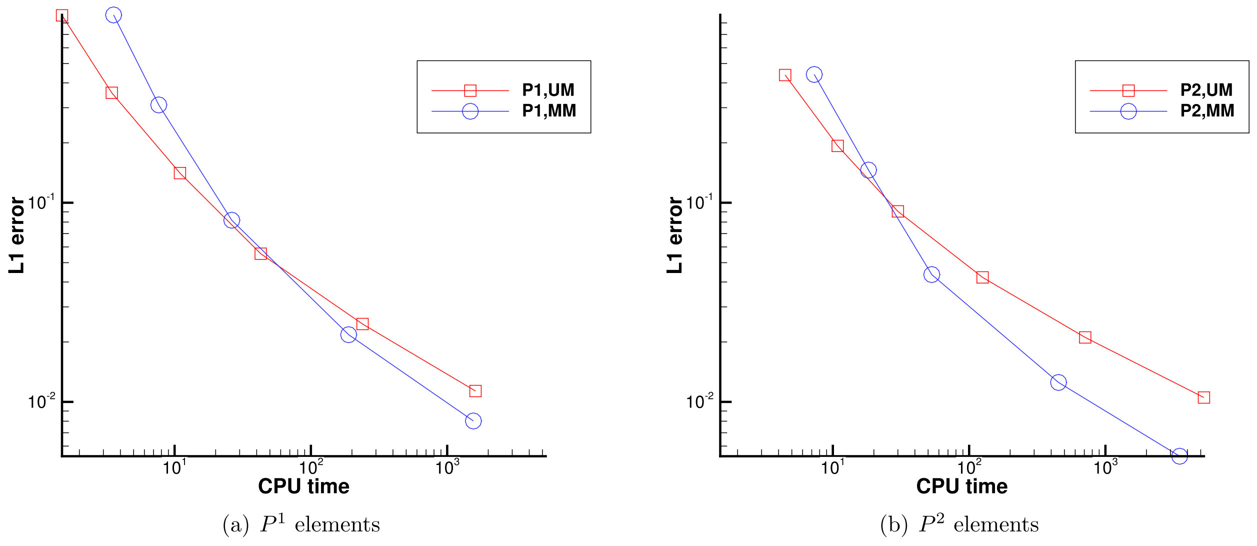
Fig. 5.22. An initial triangular mesh used in two dimensional computation. A moving mesh associated with this initial mesh is denoted by $N = 10 \times 10 \times 4$.

Table 5.6
Example 5.7 (Burgers' equation): Solution error with periodic boundary conditions and $T = \frac{0.5}{\pi}$.

k	N	$4 \times 4 \times 4$	$8 \times 8 \times 4$	$16 \times 16 \times 4$	$32 \times 32 \times 4$	$64 \times 64 \times 4$	$128 \times 128 \times 4$	$256 \times 256 \times 4$
1	L^1	1.747e-1	4.126e-2	9.444e-3	2.441e-3	6.334e-4	1.611e-4	4.079e-5
	Order		2.08	2.13	1.95	1.95	1.98	1.98
	L^2	1.714e-1	4.156e-2	9.642e-3	2.531e-3	6.750e-4	1.738e-4	4.451e-5
	Order		2.04	2.11	1.93	1.91	1.96	1.97
	L_∞	1.024e-1	2.935e-2	8.746e-3	2.303e-3	6.527e-4	1.643e-4	4.055e-5
	Order		1.80	1.75	1.93	1.82	1.99	2.02
2	L^1	2.301e-2	4.140e-3	5.379e-4	7.488e-5	9.759e-6	1.155e-6	1.320e-7
	Order		2.47	2.94	2.84	2.94	3.08	3.13
	L^2	2.226e-2	5.360e-3	7.435e-4	1.264e-4	2.082e-5	2.650e-6	2.719e-7
	Order		2.05	2.85	2.56	2.60	2.97	3.28
	L_∞	1.236e-2	5.641e-3	1.171e-3	2.343e-4	4.322e-5	6.119e-6	6.826e-7
	Order		1.13	2.27	2.32	2.44	2.82	3.16

Table 5.7
Example 5.7: Solution error in L^1 norm with periodic boundary conditions and $T = \frac{1.5}{\pi}$. Various numbers of sweeps of a low-pass filter have been applied to the smoothing of the metric tensor.

k	Sweeps\\N	$2 \times 2 \times 4$	$4 \times 4 \times 4$	$8 \times 8 \times 4$	$16 \times 16 \times 4$	$32 \times 32 \times 4$	$64 \times 64 \times 4$	$128 \times 128 \times 4$
1	3	2.083e0	8.769e-1	3.099e-1	8.168e-2	2.175e-2	8.022e-3	3.924e-3
	Order		1.25	1.50	1.92	1.91	1.44	1.03
	30	2.084e0	8.693e-1	3.439e-1	1.125e-1	3.492e-2	1.379e-2	6.502e-3
	Order		1.26	1.34	1.61	1.69	1.34	1.08
	100	2.084e0	8.724e-1	3.527e-1	1.331e-1	4.521e-2	1.862e-2	8.712e-3
	Order		1.26	1.31	1.41	1.56	1.28	1.10
2	3	1.073e0	4.406e-1	1.459e-1	4.358e-2	1.252e-2	5.336e-3	3.244e-3
	Order		1.28	1.59	1.74	1.80	1.23	0.72
	30	1.077e0	4.368e-1	1.786e-1	6.670e-2	2.515e-2	1.117e-2	5.858e-3
	Order		1.30	1.29	1.42	1.41	1.17	0.93
	100	1.077e0	4.378e-1	1.892e-1	8.369e-2	3.326e-2	1.571e-2	8.025e-3
	Order		1.30	1.21	1.18	1.33	1.08	0.97



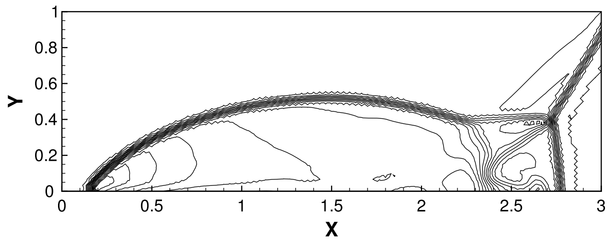
(a) P^1 elements (b) P^2 elements

Fig. 5.23. Example 5.7. L^1 error is plotted as a function of CPU time (in seconds).

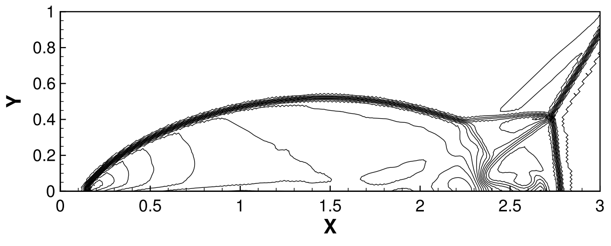
the gradient system of an energy function associated with mesh equidistribution and alignment. Moreover, hyperbolic conservation laws are discretized on a moving mesh in the quasi-Lagrangian fashion with which the mesh movement is treated continuously and thus no interpolation is needed for physical variables from the old mesh to the new one. Furthermore, the mesh movement introduces two extra convection terms in the finite element formulation of conservation laws and their discretization can be incorporated into the DG discretization naturally.

Table 5.8
Example 5.8: Solution error (in density) for periodic boundary conditions and $T = 1$, $\beta = 100$.

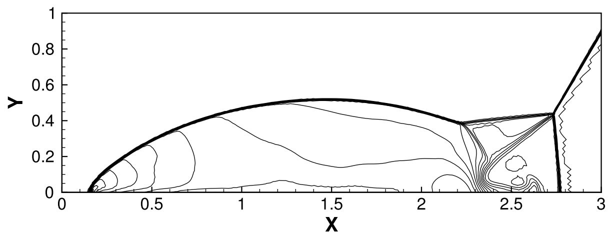
k	N	$2 \times 2 \times 4$	$4 \times 4 \times 4$	$8 \times 8 \times 4$	$16 \times 16 \times 4$	$32 \times 32 \times 4$	$64 \times 64 \times 4$	$128 \times 128 \times 4$
1	L^1	2.590e-1	5.771e-2	1.263e-2	2.991e-3	7.470e-4	1.873e-4	4.635e-5
	Order		2.17	2.19	2.08	2.00	2.00	2.02
	L^2	1.510e-1	3.754e-2	8.453e-3	2.018e-3	5.031e-4	1.293e-4	3.256e-5
	Order		2.01	2.15	2.07	2.00	1.96	1.99
	L_∞	1.543e-1	8.735e-2	2.249e-2	5.674e-3	1.663e-3	4.273e-4	9.618e-5
	Order		0.82	1.96	1.99	1.77	1.96	2.15
2	L^1	5.116e-2	1.059e-2	1.727e-3	2.590e-4	3.557e-5	4.255e-6	4.595e-7
	Order		2.27	2.62	2.74	2.86	3.06	3.21
	L^2	3.474e-2	7.500e-3	1.285e-3	2.100e-4	3.157e-5	3.913e-6	3.936e-7
	Order		2.21	2.55	2.61	2.73	3.01	3.31
	L_∞	5.626e-2	2.082e-2	4.512e-3	8.601e-4	1.608e-4	2.278e-5	2.329e-6
	Order		1.43	2.21	2.39	2.42	2.82	3.29



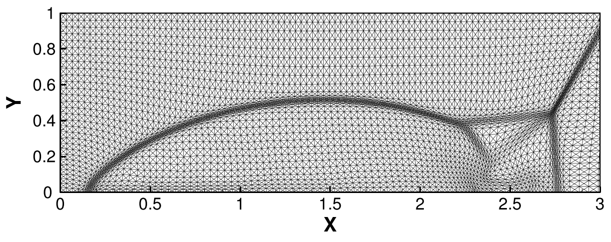
(a) Fixed Mesh, $N = 120 \times 30 \times 4$



(b) Fixed Mesh, $N = 180 \times 45 \times 4$



(c) Moving Mesh, $N = 120 \times 30 \times 4$



(d) Moving mesh

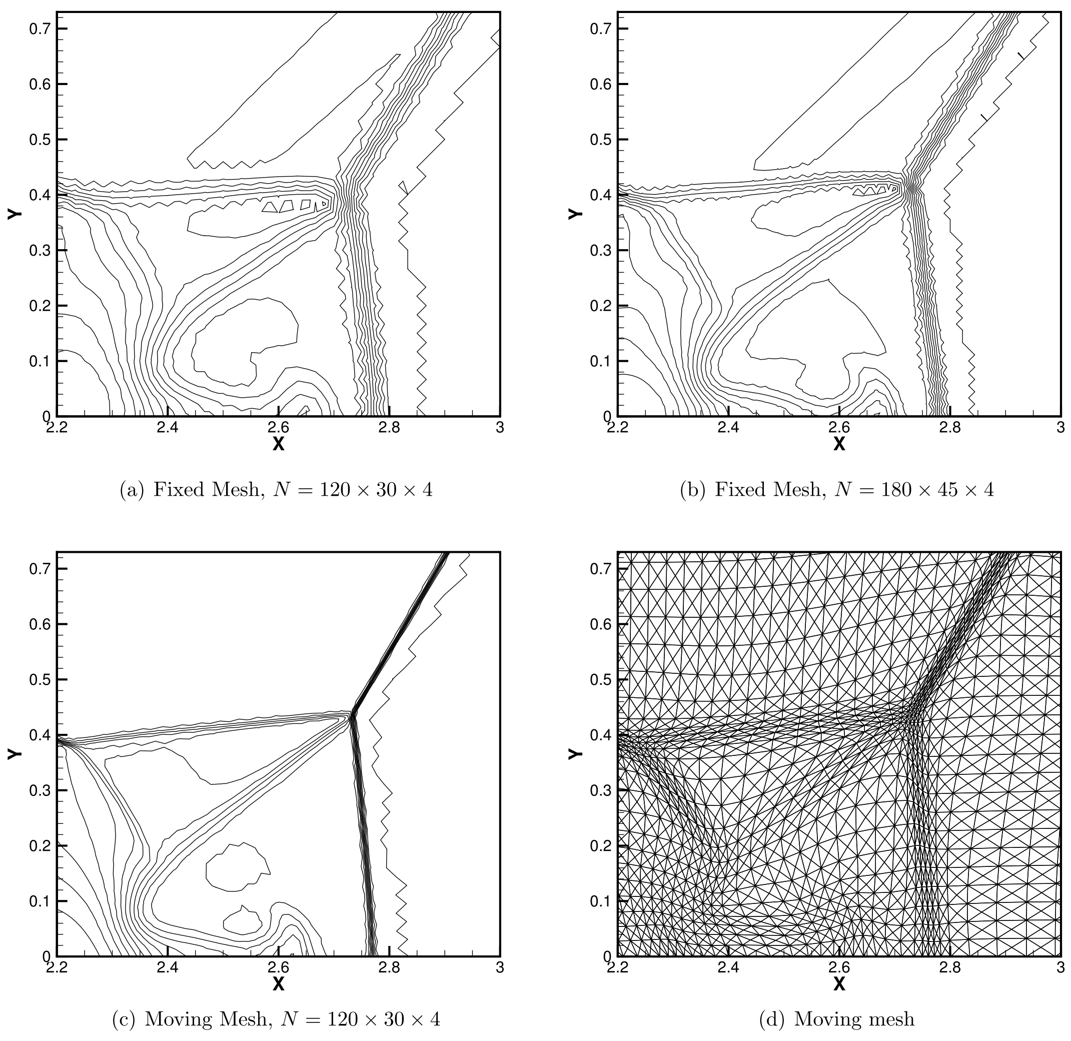
Fig. 5.24. P^1 elements are used. 30 equally spaced density contours from 1.4 to 22.1183 are used in the contour plots.

The numerical results for a selection of one- and two-dimensional examples have been presented. They show that the moving mesh DG method achieves the second and third order of convergence for P^1 and P^2 elements, respectively, for problems with smooth solutions and is able to capture shocks and concentrate mesh points in non-smooth regions. Moreover, it is shown that the numerical solution with a moving mesh is generally more accurate than that with a uniform mesh of the same number of points and often comparable with the solution obtained with a much finer uniform mesh. Furthermore, numerical results for problems with smooth and discontinuous solutions in one and two dimensions have shown that the accuracy of the method is not sensitive to the smoothness of the mesh, which is in contrast with the situation for the moving mesh finite difference WENO method [44].

We recall that the Hessian of a physical variable has been used in this work to guide the mesh adaptation (cf. (3.1)). This is based on linear interpolation error [23] and has been known to work for many problems. Nevertheless, it would be advantageous to define the metric tensor based on some a posteriori error estimate. Investigations of using residual-based metric tensors for the moving mesh DG method presented in this work have been underway.

Acknowledgements

The work was supported in part by China NSAF grant U1630247, Science Challenge Project, No. TZ2016002 and NSFC grant 11571290. D.M. Luo gratefully acknowledges the financial support from China Scholarship Council under the grant 201506310112 to visit the Department of Mathematics, University of Kansas from September 2015 to September 2017. He also thanks Professor Jun Zhu at Nanjing University of Aeronautics and Astronautics for his help with the HWENO limiters.



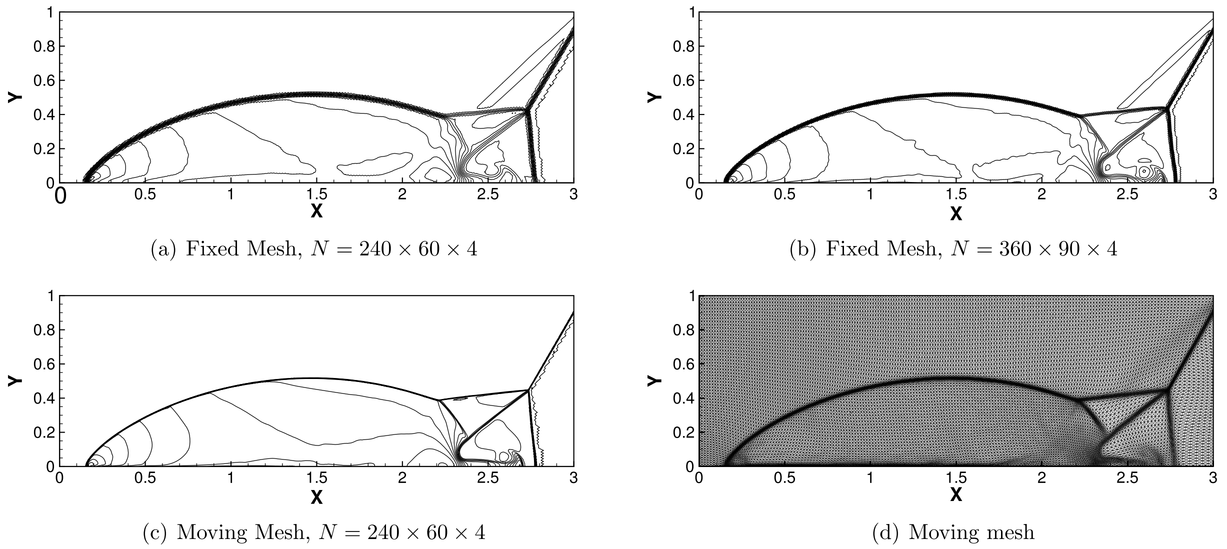
(a) Fixed Mesh, $N = 120 \times 30 \times 4$

(b) Fixed Mesh, $N = 180 \times 45 \times 4$

(c) Moving Mesh, $N = 120 \times 30 \times 4$

(d) Moving mesh

Fig. 5.25. P^1 elements are used. Close view of the complex zone in Fig. 5.24.



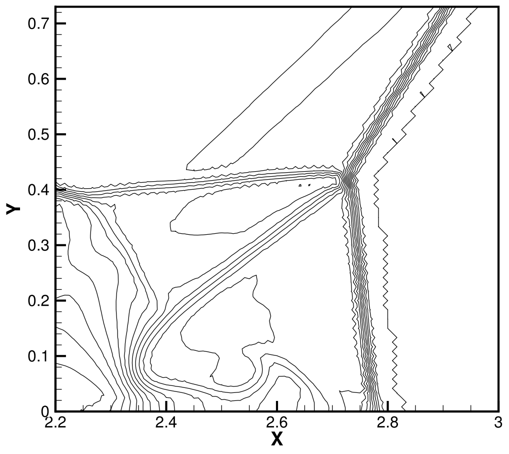
(a) Fixed Mesh, $N = 240 \times 60 \times 4$

(b) Fixed Mesh, $N = 360 \times 90 \times 4$

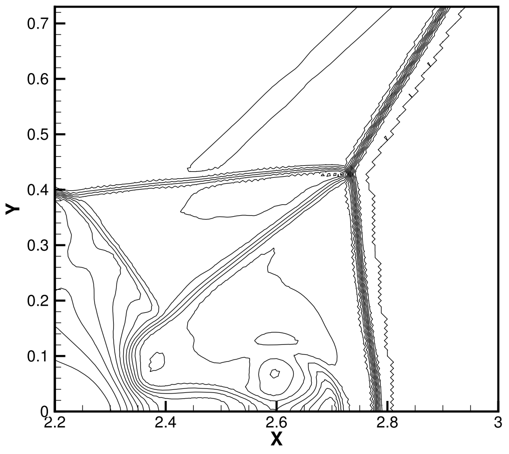
(c) Moving Mesh, $N = 240 \times 60 \times 4$

(d) Moving mesh

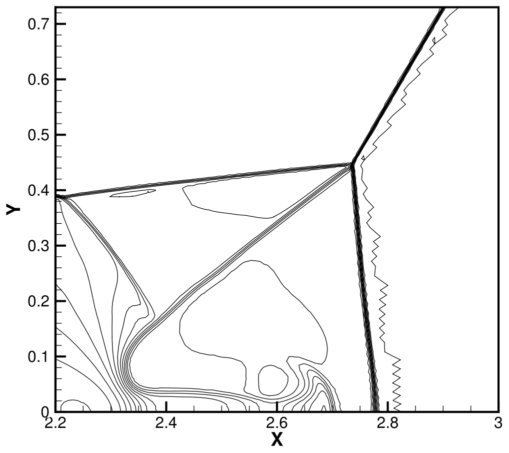
Fig. 5.26. P^1 elements are used. 30 equally spaced density contours from 1.4 to 22.1183 are used in the contour plots.



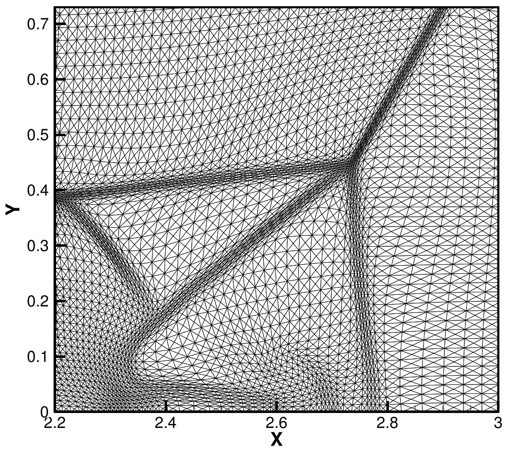
(a) Fixed Mesh, $N = 240 \times 60 \times 4$



(b) Fixed Mesh, $N = 360 \times 90 \times 4$

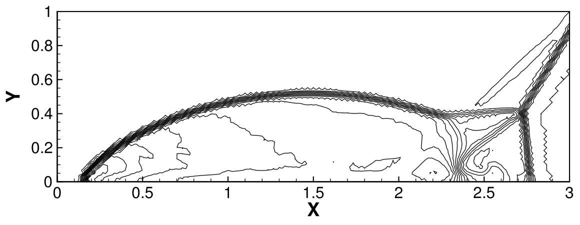


(c) Moving Mesh, $N = 240 \times 60 \times 4$

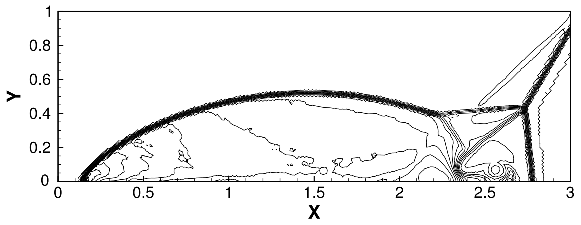


(d) Moving mesh

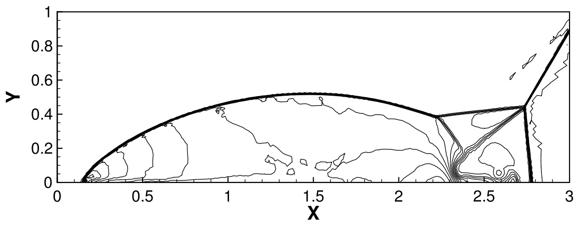
Fig. 5.27. P^1 elements are used. Close view of the complex zone in Fig. 5.26.



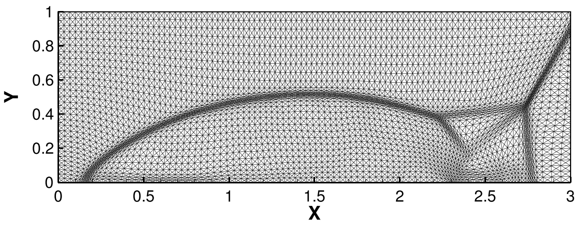
(a) Fixed Mesh, $N = 120 \times 30 \times 4$



(b) Fixed Mesh, $N = 180 \times 45 \times 4$

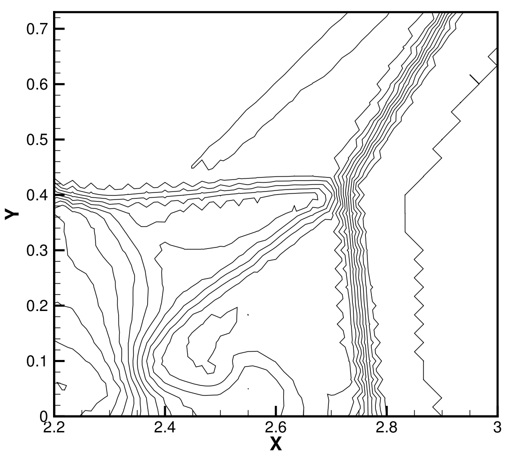


(c) Moving Mesh, $N = 120 \times 30 \times 4$

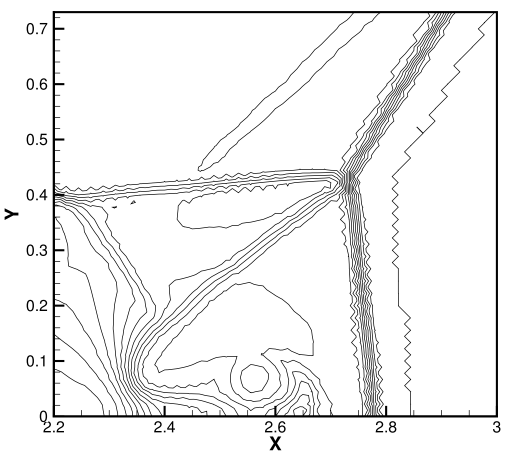


(d) Moving mesh

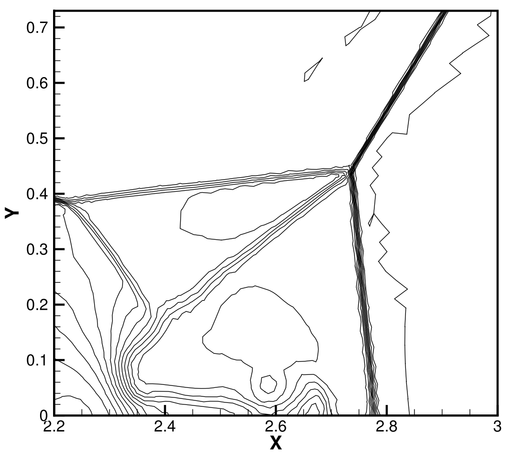
Fig. 5.28. P^2 elements are used. 30 equally spaced density contours from 1.4 to 22.1183 are used in the contour plots.



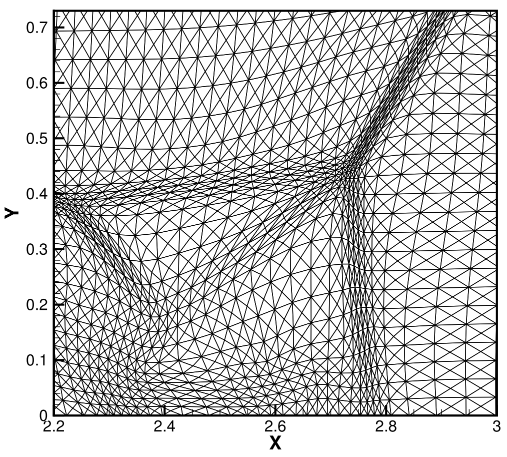
(a) Fixed Mesh, $N = 120 \times 30 \times 4$



(b) Fixed Mesh, $N = 180 \times 45 \times 4$

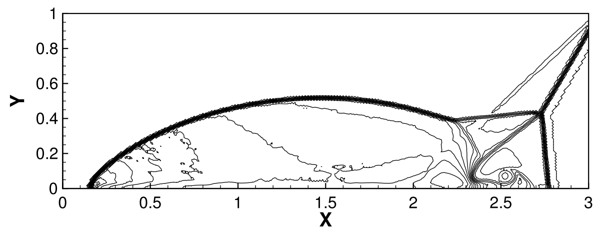


(c) Moving Mesh, $N = 120 \times 30 \times 4$

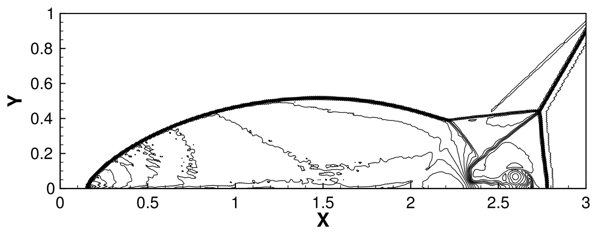


(d) Moving mesh

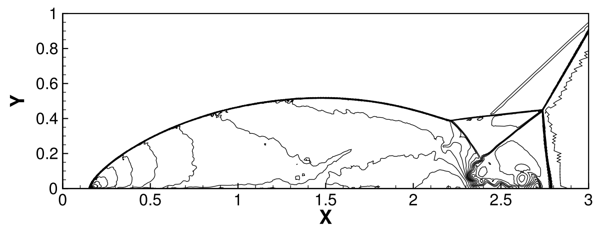
Fig. 5.29. P^2 elements are used. Close view of the complex zone in Fig. 5.28.



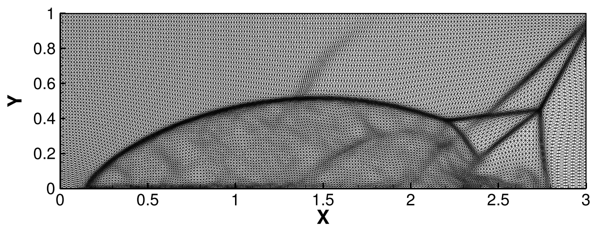
(a) Fixed Mesh, $N = 240 \times 60 \times 4$



(b) Fixed Mesh, $N = 360 \times 90 \times 4$

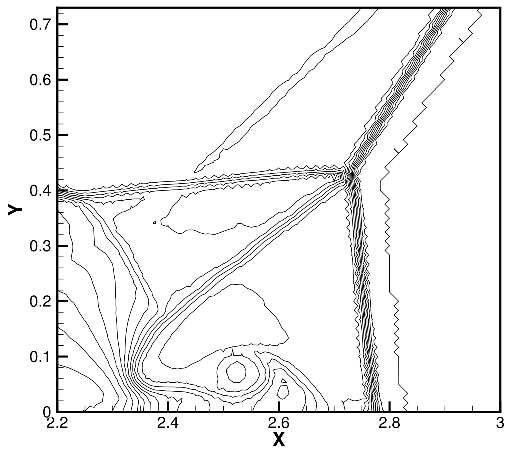


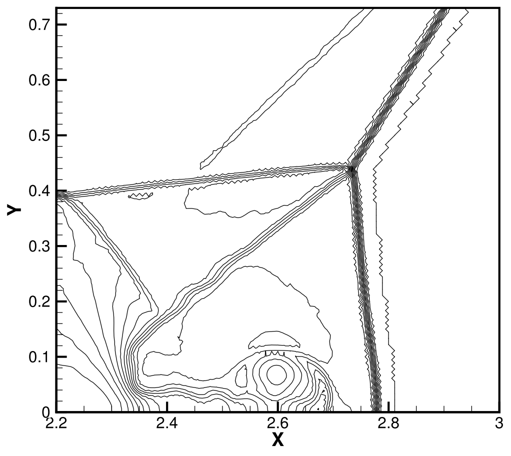
(c) Moving Mesh, $N = 240 \times 60 \times 4$

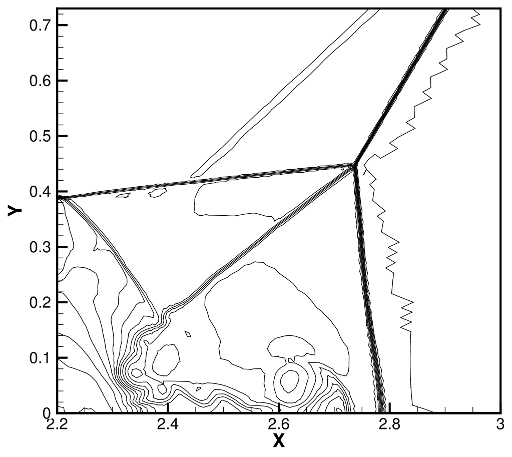


(d) Moving mesh

Fig. 5.30. P^2 elements are used. 30 equally spaced density contours from 1.4 to 22.1183 are used in the contour plots.


(a) Fixed Mesh, $N = 240 \times 60 \times 4$


(b) Fixed Mesh, $N = 360 \times 90 \times 4$


(c) Moving Mesh, $N = 240 \times 60 \times 4$

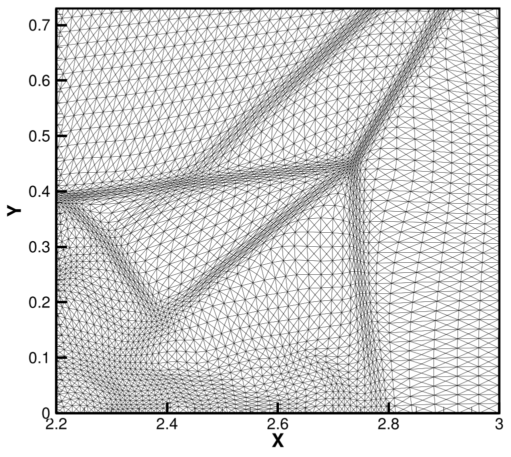
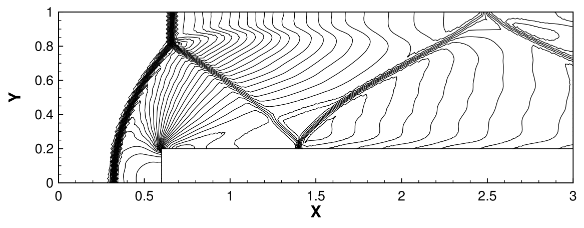
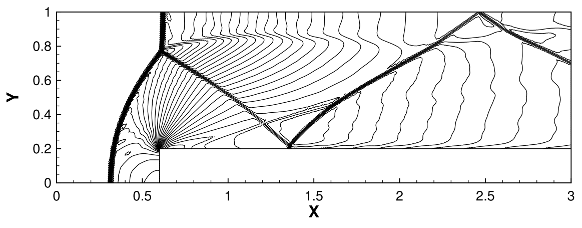
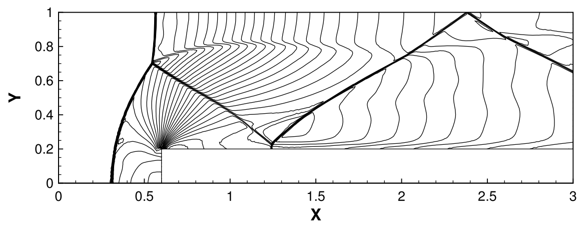

(d) Moving mesh

Fig. 5.31. P^2 elements are used. Close view of the complex zone in Fig. 5.30.


(a) Fixed Mesh, $N = 120 \times 40 \times 4$


(b) Fixed Mesh, $N = 240 \times 80 \times 4$


(c) Moving Mesh, $N = 120 \times 40 \times 4$

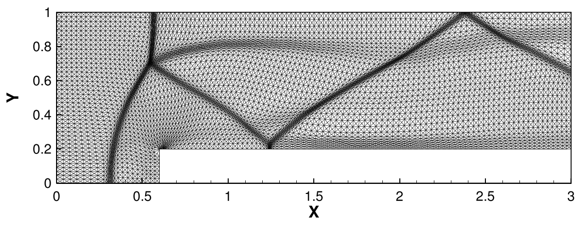

(d) Moving mesh

Fig. 5.32. P^1 elements are used. 30 equally spaced density contours from 0.32 to 6.15 are used in the contour plots.

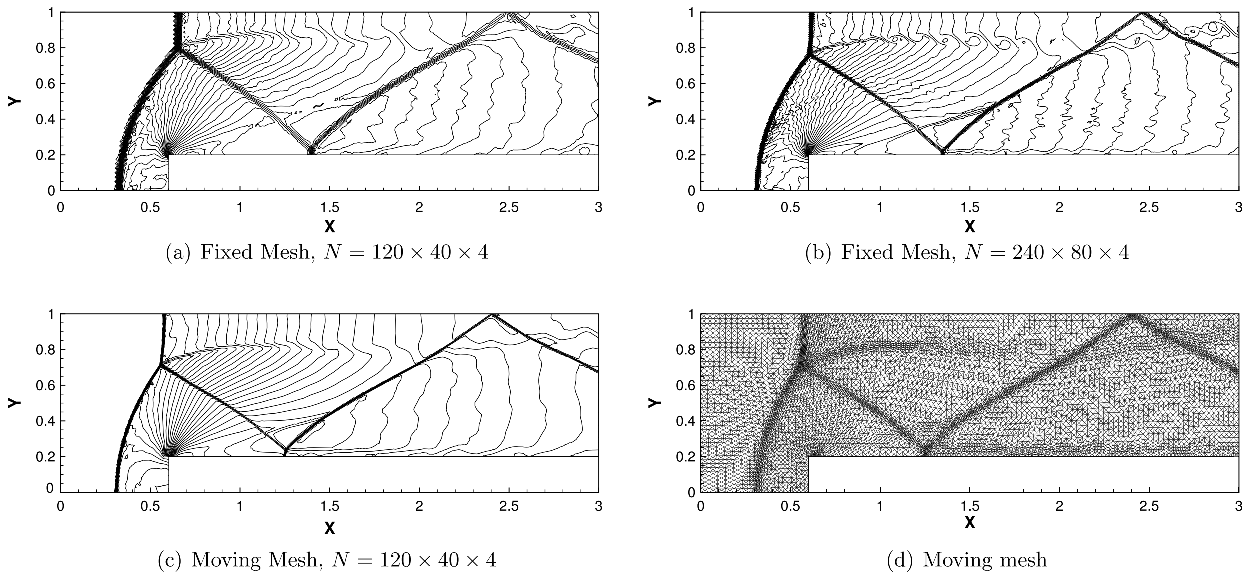

(a) Fixed Mesh, $N = 120 \times 40 \times 4$
(b) Fixed Mesh, $N = 240 \times 80 \times 4$
(c) Moving Mesh, $N = 120 \times 40 \times 4$
(d) Moving mesh

Fig. 5.33. P^2 elements are used. 30 equally spaced density contours from 0.32 to 6.15 are used in the contour plots.

References

[1] M.J. Baines, Moving Finite Elements, Oxford University Press, Oxford, 1994.
[2] M.J. Baines, M.E. Hubbard, P.K. Jimack, Velocity-based moving mesh methods for nonlinear partial differential equations, Commun. Comput. Phys. 10 (2011) 509–576.
[3] R.E. Bank, R.F. Santos, Analysis of some moving space-time finite element methods, SIAM J. Numer. Anal. 30 (1993) 1–18.
[4] K. Bey, J. Oden, *hp*-version discontinuous Galerkin methods for hyperbolic conservation laws, Comput. Methods Appl. Mech. Eng. 133 (1996) 259–286.
[5] C.J. Budd, W. Huang, R.D. Russell, Adaptivity with moving grids, Acta Numer. 18 (2009) 111–241.
[6] B. Cockburn, S. Hou, C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws IV: the multidimensional case, Math. Comput. 54 (1990) 545–581.
[7] B. Cockburn, S.-Y. Lin, C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws III: one dimensional systems, J. Comput. Phys. 84 (1989) 90–113.
[8] B. Cockburn, C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws II: general framework, Math. Comput. 52 (1989) 411–435.
[9] B. Cockburn, C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws V: multidimensional systems, J. Comput. Phys. 141 (1998) 199–224.
[10] M. Dumbser, D.S. Balsara, E.F. Toro, C.D. Munz, A unified framework for the construction of one-step finite-volume and discontinuous Galerkin schemes on unstructured meshes, J. Comput. Phys. 227 (2008) 8209–8253.
[11] T. Dupont, Mesh modification for evolution equations, Math. Comput. 39 (1982) 85–107.
[12] T. Dupont, Y. Liu, Symmetric error estimates for moving mesh Galerkin methods for advection-diffusion equations, SIAM J. Numer. Anal. 40 (2002) 914–927.
[13] P. Fu, G. Schnücke, Y. Xia, Arbitrary Lagrangian-Eulerian discontinuous Galerkin method for conservation laws on moving simplex meshes, Math. Comput. (2019) 1–35.
[14] C.W. Hirt, A.A. Amsden, J.L. Cook, An arbitrary Lagrangian-Eulerian computing method for all flow speeds, J. Comput. Phys. 14 (1974) 227–253.
[15] C. Hu, C.-W. Shu, Weighted essentially non-oscillatory schemes on triangular meshes, J. Comput. Phys. 150 (1999) 97–127.
[16] W. Huang, Mathematical principles of anisotropic mesh adaptation, Commun. Comput. Phys. 1 (2006) 276–310.
[17] W. Huang, Variational mesh adaptation: isotropy and equidistribution, J. Comput. Phys. 174 (2001) 903–924.
[18] W. Huang, L. Kamenski, A geometric discretization and a simple implementation for variational mesh generation and adaptation, J. Comput. Phys. 301 (2015) 322–337.
[19] W. Huang, L. Kamenski, On the mesh nonsingularity of the moving mesh PDE method, Math. Comput. 87 (2018) 1887–1911.
[20] W. Huang, Y. Ren, R.D. Russell, Moving mesh methods based on moving mesh partial differential equations, J. Comput. Phys. 113 (1994) 279–290.
[21] W. Huang, Y. Ren, R.D. Russell, Moving mesh partial differential equations (MMPDEs) based upon the equidistribution principle, SIAM J. Numer. Anal. 31 (1994) 709–730.
[22] W. Huang, R.D. Russell, Adaptive Moving Mesh Methods, Springer-Verlag, New York, 2011.
[23] W. Huang, W. Sun, Variational mesh adaptation II: error estimates and monitor functions, J. Comput. Phys. 184 (2003) 619–648.
[24] G. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1996) 202–228.
[25] P. Jimack, A. Wathen, Temporal derivatives in the finite-element method on continuously deforming grids, SIAM J. Numer. Anal. 28 (1991) 990–1003.
[26] L. Kamenski, W. Huang, How a nonconvergent recovered Hessian works in mesh adaptation, SIAM J. Numer. Anal. 52 (2014) 1692–1708.
[27] L. Krivodonova, J. Xin, J.-F. Remacle, N. Chevaugeon, J.E. Flaherty, Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws, Appl. Numer. Math. 48 (2004) 323–338.
[28] R. Li, T. Tang, Moving mesh discontinuous Galerkin method for hyperbolic conservation laws, J. Sci. Comput. 27 (2006) 347–363.
[29] H. Luo, J.D. Baum, R. Lohner, A Hermite WENO-based limiter for discontinuous Galerkin method on unstructured grids, J. Comput. Phys. 225 (2007) 686–713.
[30] J. Machenzie, A. Nicola, A discontinuous Galerkin moving mesh method for Hamilton-Jacobi equations, SIAM J. Sci. Comput. 29 (2007) 2258–2282.
[31] L.G. Margolin, Introduction to "an arbitrary Lagrangian-Eulerian computing method for all flow speeds", J. Comput. Phys. 135 (1997) 198–202.

[32] J. Qiu, C.-W. Shu, Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method: one dimensional case, J. Comput. Phys. 193 (2003) 115–135.
[33] J. Qiu, C.-W. Shu, Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method II: two dimensional case, Comput. Fluids 34 (2005) 642–663.
[34] J. Qiu, C.-W. Shu, Runge-Kutta discontinuous Galerkin method using WENO limiters, SIAM J. Sci. Comput. 26 (2005) 907–929.
[35] W.H. Reed, T.R. Hill, Triangular Mesh Methods for Neutron Transport Equation, Los Alamos Scientific Laboratory Report LA-UR-73-479, 1973.
[36] C.-W. Shu, Total-variation-diminishing time discretizations, SIAM J. Sci. Stat. Comput. 9 (1988) 1073–1084.
[37] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock capturing schemes II, J. Comput. Phys. 83 (1989) 32–78.
[38] J.M. Stockie, J.A. Mackenzie, R.D. Russell, A moving mesh method for one-dimensional hyperbolic conservation laws, SIAM J. Sci. Comput. 22 (2001) 1791–1813.
[39] T. Tang, Moving mesh methods for computational fluid dynamics flow and transport, in: Recent Advances in Adaptive Computation, Hangzhou, 2004, in: AMS Contemporary Mathematics, vol. 383, Amer. Math. Soc., Providence, RI, 2005, pp. 141–173.
[40] H. Tang, T. Tang, Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws, SIAM J. Numer. Anal. 41 (2003) 487–515.
[41] M. Uzunca, B. Karasözen, T. Küçükseyhan, Moving mesh discontinuous Galerkin methods for PDEs with traveling wave, Appl. Math. Comput. 292 (2017) 9–18.
[42] E.S. Wise, B.T. Cox, B.E. Treeby, Mesh density functions based on local bandwidth applied to moving mesh methods, Commun. Comput. Phys. 22 (2017) 1286–1308.
[43] P.R. Woodward, P. Colella, The numerical simulation of two-dimensional fluid flow with strong shocks, J. Comput. Phys. 54 (1984) 115–173.
[44] X. Yang, W. Huang, J. Qiu, A moving mesh WENO method for one-dimensional conservation laws, SIAM J. Sci. Comput. 34 (2012) A2317–A2343.
[45] F. Zhang, W. Huang, X. Li, S. Zhang, Moving mesh finite element simulation for phase-field modeling of brittle fracture and convergence of Newton's iteration, J. Comput. Phys. 356 (2018) 127–149.
[46] H. Zhang, P.A. Zegeling, A moving mesh finite difference method for non-monotone solutions of non-equilibrium equations in porous media, Commun. Comput. Phys. 224 (2017) 935–964.
[47] Z. Zhang, A. Naga, A new finite element gradient recovery method: superconvergence property, SIAM J. Sci. Comput. 26 (2005) 1192–1213.
[48] X. Zhong, C.-W. Shu, A simple weighted essentially nonoscillatory limiter for Runge-Kutta discontinuous Galerkin methods, J. Comput. Phys. 232 (2013) 397–415.
[49] J. Zhu, J. Qiu, Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method III: unstructured meshes, J. Sci. Comput. 39 (2009) 293–321.
[50] J. Zhu, J. Qiu, C.-W. Shu, M. Dumbser, Runge-Kutta discontinuous Galerkin method using WENO limiters II: unstructured meshes, J. Comput. Phys. 227 (2008) 4330–4353.
[51] J. Zhu, X. Zhong, C.-W. Shu, J. Qiu, Runge-Kutta discontinuous Galerkin method using a new type of WENO limiters on unstructured meshed, J. Comput. Phys. 248 (2013) 200–220.
[52] J. Zhu, X. Zhong, C.-W. Shu, J. Qiu, Runge-Kutta discontinuous Galerkin method with a simple and compact Hermite WENO limiter, Commun. Comput. Phys. 19 (2016) 944–969.
[53] J. Zhu, X. Zhong, C.-W. Shu, J. Qiu, Runge-Kutta discontinuous Galerkin method with a simple and compact Hermite WENO limiter on unstructured meshes, Commun. Comput. Phys. 21 (2017) 623–649.