ADAPTIVE MESH METHODS FOR ONE- AND TWO-DIMENSIONAL HYPERBOLIC CONSERVATION LAWS

HUAZHONG TANG * AND TAO TANG †

Abstract. We develop efficient moving mesh algorithms for one- and two-dimensional hyperbolic systems of conservation laws. The algorithms are formed by two independent parts: PDE evolution and mesh-redistribution. The first part can be any appropriate high-resolution schemes, and the second part is based on an iterative procedure. In each iteration, meshes are first redistributed by an equidistribution principle and then on the resulting new grids the underlying numerical solutions are updated by a conservative-interpolation formula proposed in this work. The iteration for the mesh-redistribution at a given time step is complete when the meshes governed by a nonlinear equation reach the equilibrium state. The main idea of the proposed method is to keep the mass-conservation of the underlying numerical solution at each re-distribution step. In 1D, we can show that the underlying numerical approximation obtained in the mesh-redistribution part satisfies the desired TVD property, which guarantees that the numerical solution at any time level is TVD provided that the PDE solver in the first part satisfies such a property. Several test problems in one- and two-dimensions are computed using the proposed moving mesh algorithm. The computations demonstrate that our methods are efficient for solving problems with shock discontinuities, obtaining the same resolution with much less number of grid points than the uniform mesh approach.

AMS subject classifications. 65M93, 35L64, 76N10.

Key words. Adaptive mesh method, hyperbolic conservation laws, finite volume method.

1. Introduction. Adaptive mesh methods have important applications for a variety of physical and engineering areas such as solid and fluid dynamics, combustion, heat transfer, material science etc. The physical phenomena in these areas develop dynamically singular or nearly singular solutions in fairly localized regions, such as shock waves, boundary layers, detonation waves etc. The numerical investigation of these physical problems may require extremely fine meshes over a small portion of the physical domain to resolve the large solution variations. In multi-dimensions, developing effective and robust adaptive grid methods for these problems becomes necessary. Successful implementation of the adaptive strategy can increase the accuracy of the numerical approximations and also decrease the computational cost. In the past two decades, there have seen many important progress in moving mesh methods for partial differential equations, including the variational approach of Winslow [34], Brackbill et al. [4, 5], and Ren and Wang [26]; finite element methods of Millers [23], and Davis and Flaherty [10]; moving mesh PDEs of Russell et al. [6, 31], Li and Petzold [20], and Ceniceros and Hou [7]; and moving mesh methods based on harmonic mapping of Dvinsky [11] and Li et al. [21].

The main objective of this paper is to develop an one- and two-dimensional moving mesh method for hyperbolic systems of conservation laws. Harten and Hyman [13] began the earliest study in this direction, by moving the grid at an adaptive speed in each time step to improve the resolution of shocks and contact discontinuities. After their work, many other moving mesh methods for hyperbolic problems have been proposed in the literature, including Azarenok [1], Azarenok and Ivanenko [2], Fazio and LeVeque [12], Liu et al. [22], Saleri and Steinberg [27], and Stockie et al. [31]. However, it is noticed that many existing moving mesh methods for hyperbolic problems are designed for one space dimension. In 1D, it is generally possible to compute on a very fine grid and so the need for moving mesh methods may not be clear. Multidimensional moving mesh methods are often difficult to use in fluid dynamics problems since the grid will typically suffer large distortions

* State Key Lab of Scientific and Engineering Computing, Institute of Computational Mathematics, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, P.O.Box 2719, Beijing 100080, P.R. China. Current address: School of Mathematical Sciences, Peking University, Beijing 100871, P. R. China. Email:hztang@math.pku.edu.cn

†Department of Mathematics, The Hong Kong Baptist University, Kowloon Tong, Hong Kong, P. R. China. Email:ttang@math.hkbu.edu.hk

and possible tangling. It is therefore useful to design a simple and robust moving mesh algorithm for hyperbolic problems in multi-dimensions.

Following Li et al. [21] we propose a moving mesh method containing two separate parts: PDE time-evolution and mesh-redistribution. The first part can be any suitable high-resolution methods such as the wave-propagation algorithm, central schemes, and ENO methods. Once numerical solutions are obtained at the given time level, the meshes will be redistributed based on an iteration procedure. At each iteration, the grids are moved based on a variational principle and the underlying numerical solutions at the new grids will be updated by using some simple methods (such as the conventional interpolation). It is noted that the direct use of the conventional interpolation is unsatisfactory for hyperbolic problems, since many physical properties such as mass-conservation and TVD (in 1D) may be destroyed. In order to preserve these physical properties, we propose to use conservative-interpolation in the solution-updating step. The idea of using the conservative-interpolation is new, and is shown to work successfully for hyperbolic problems. This approach also preserves the total mass of the numerical solutions, and by the well-known Lax-Wendroff theory the numerical solutions converge to the weak solution of the underlying hyperbolic system.

The paper is organized as follows. In Section 2, we briefly review some theory of variational approach for moving mesh methods, which is relevant to the mesh-redistribution part of our algorithm. In Section 3, we propose an 1D moving mesh algorithm for solving hyperbolic system of conservation laws. It will be extended to an 2D algorithm in Section 4. Numerical experiments are carried out in Sections 5 and 6, where several 1D and 2D examples are considered.

2. Mesh generation based on the variational approach. Let $\vec{x} = (x_1, x_2, \cdots, x_d)$ and $\vec{\xi} = (\xi_1, \xi_2, \cdots, \xi_d)$ denote the physical and computational coordinates, respectively. Here $d \geq 1$ denotes the number of spatial dimension. A one–to–one coordinate transformation from the computational (or logical) domain Ω_c to the physical domain Ω_p is denoted by

$$\vec{x} = \vec{x}(\vec{\xi}), \quad \vec{\xi} \in \Omega_c. \tag{2.1}$$

Its inversion is denoted by

$$\vec{\xi} = \vec{\xi}(\vec{x}), \quad \vec{x} \in \Omega_p. \tag{2.2}$$

In the variational approach, the mesh map is provided by the minimizer of a functional of the following form:

$$E(\vec{\xi}) = \frac{1}{2} \sum_k \int_{\Omega_p} \nabla \xi_k^T G_k^{-1} \nabla \xi_k d\vec{x} \ , \tag{2.3}$$

where $\nabla := (\partial_{x_1}, \partial_{x_2}, \cdots, \partial_{x_d})^T$, G_k are given symmetric positive definite matrices called *monitor functions*. In general, the monitor functions depend on the underlying solution to be adapted. More terms can be added to the functional (2.3) to control other aspects of the mesh such as orthogonality and mesh alignment with a given vector field [4, 5].

The variational mesh is determined by the Euler-Lagrange equation of the above functional:

$$\nabla \cdot \left(G_k^{-1} \nabla \xi_k \right) = 0, \qquad 1 \leq k \leq d. \tag{2.4}$$

One of the simplest choices of the monitor functions is $G_k = \omega I$, $1 \leq k \leq d$, where I is the identity matrix and ω is a positive weight function, e.g., $\omega = \sqrt{1 + |\nabla u|^2}$. Here u is the solution of the underlying PDE. In this case, we obtain Winslow's variable diffusion method [34]:

$$\nabla \cdot \left(\frac{1}{\omega} \nabla \xi_k \right) = 0, \qquad 1 \leq k \leq d. \tag{2.5}$$

By using the above equations, a map between the physical domain Ω_p and the logical domain Ω_c can be computed. Typically, the map transforms a uniform mesh in the logical domain to cluster grid points at the regions of the physical domain where the solution has the largest gradients.

2.1. 1D case. Although the main objective of this work is to provide an effective moving mesh algorithm for 2D conservation laws, it is easier to illustrate the basic moving mesh ideas by starting with some 1D discussions. Let x and ξ denote the physical and computational coordinates, respectively, which are (without loss of generality) assumed to be in $[a, b]$ and $[0, 1]$, respectively. A one–to–one coordinate transformation between these domains is denoted by

$$\begin{aligned} x &= x(\xi), & \xi &\in [0,1], \\ x(0) &= a, & x(1) &= b. \end{aligned} \tag{2.6}$$

The 1D Euler-Lagrange equations is of the form

$$(\omega^{-1}\xi_x)_x = 0. \tag{2.7}$$

Using the above equation we can obtain the conventional 1D *equidistribution principle*: ωx_ξ=constant or equivalently

$$(\omega x_\xi)_\xi = 0. \tag{2.8}$$

Both equations (2.7) and (2.8) are of the same form and therefore solving any of them will end up with desired mesh map $x = x(\xi)$. However, the situation is different in 2D case where we will choose to solve equations of the form (2.8), as to be described in the next subsection.

2.2. 2D case. We will consider the Winslow's variable diffusion method (2.5). The extension to the general Euler-Lagrange equation (2.4) is straightforward. Let us denote $(x, y) = (x(\xi, \eta), y(\xi, \eta))$ the mesh map in two dimensions. The equation (2.5) becomes

$$\begin{aligned} (\omega^{-1}\xi_x)_x + (\omega^{-1}\xi_y)_y &= 0, \\ (\omega^{-1}\eta_x)_x + (\omega^{-1}\eta_y)_y &= 0. \end{aligned} \tag{2.9}$$

In practice, the physical domain Ω_p may have very complex geometry and as a result directly solving elliptic system (2.9) on structured grids is unrealistic. Therefore we usually solve the corresponding mesh generation equations on the computational domain Ω_c by interchanging the dependent and independent variables in (2.9):

$$\begin{aligned} &\frac{x_\xi}{J}\left[\left(x_\eta\frac{1}{J\omega}x_\eta + y_\eta\frac{1}{J\omega}y_\eta\right)_\xi - \left(x_\xi\frac{1}{J\omega}x_\eta + y_\xi\frac{1}{J\omega}y_\eta\right)_\eta\right] \\ &\qquad + \frac{x_\eta}{J}\left[-\left(x_\eta\frac{1}{J\omega}x_\xi + y_\eta\frac{1}{J\omega}y_\xi\right)_\xi + \left(x_\xi\frac{1}{J\omega}x_\xi + y_\xi\frac{1}{J\omega}y_\xi\right)_\eta\right] = 0, \\ &\frac{y_\xi}{J}\left[\left(x_\eta\frac{1}{J\omega}x_\eta + y_\eta\frac{1}{J\omega}y_\eta\right)_\xi - \left(x_\xi\frac{1}{J\omega}x_\eta + y_\xi\frac{1}{J\omega}y_\eta\right)_\eta\right] \\ &\qquad + \frac{y_\eta}{J}\left[-\left(x_\eta\frac{1}{J\omega}x_\xi + y_\eta\frac{1}{J\omega}y_\xi\right)_\xi + \left(x_\xi\frac{1}{J\omega}x_\xi + y_\xi\frac{1}{J\omega}y_\xi\right)_\eta\right] = 0. \end{aligned} \tag{2.10}$$

Note that system (2.10) is more complicated than the Euler-Lagrange equation (2.9), which requires more computational efforts in obtaining numerical approximations. An alternative approach, as observed by Ceniceros and Hou [7], is to consider a functional defined in the computational domain:

$$\tilde{E}[x, y] = \frac{1}{2}\int_{\Omega_c}\left(\tilde{\nabla}^T x G_1 \tilde{\nabla} x + \tilde{\nabla}^T y G_2 \tilde{\nabla} y\right)\, d\xi d\eta, \tag{2.11}$$

to replace the conventional functional (2.3), where G_k are monitor functions, and $\widetilde{\nabla} = (\partial_\xi, \partial_\eta)^T$. The corresponding Euler-Lagrange equation is of the form

$$\begin{aligned} \partial_\xi(G_1\partial_\xi x) + \partial_\eta(G_1\partial_\eta x) = 0, \\ \partial_\xi(G_2\partial_\xi y) + \partial_\eta(G_2\partial_\eta y) = 0. \end{aligned} \tag{2.12}$$

In particular, with the choice $G = \omega I$ we have

$$\widetilde{\nabla} \cdot (\omega \widetilde{\nabla} x) = 0, \qquad \widetilde{\nabla} \cdot (\omega \widetilde{\nabla} y) = 0. \tag{2.13}$$

The monitor functions will be chosen based on the properties of the physical solutions. A typical choice used in [7] is $\omega = \sqrt{1 + \alpha_1|u|^2 + \alpha_2|\widetilde{\nabla} u|^2}$, or $\omega = \sqrt{1 + \alpha_1|u|^2 + \alpha_2|\nabla u|^2}$, where α_1, α_2 are some nonnegative constant.

3. 1D algorithm. For convenience, we assume that a fixed uniform mesh on the computational domain is given by $\xi_j = j/(J+1), 0 \le j \le J+1$. We denote cell average of solution $u(x)$ over the interval $[x_j, x_{j+1}]$ as

$$u_{j+\frac12} = \frac{1}{\Delta x_{j+\frac12}} \int_{x_j}^{x_{j+1}} u(x)\ dx,$$

where $\Delta x_{j+\frac12} = x_{j+1} - x_j$. In practice, the monitor function ω is always associated with the underlying solution u or/and its derivatives, but without loss of generality we assume that $\omega = \omega(u)$. For monitor functions involving first or second derivatives, central differencing will be used to approximate these derivatives.

3.1. Mesh-redistribution. In order to solve the mesh redistribution equation (2.8), we introduce an artificial time τ and solve

$$x_\tau = (\omega x_\xi)_\xi\,, \qquad 0 < \xi < 1 \tag{3.1}$$

subject to boundary conditions $x(0,\tau) = a$ and $x(1,\tau) = b$. We discretize (3.1) on the uniform mesh in Ω_c:

$$\tilde{x}_j = x_j + \frac{\Delta\tau}{\Delta\xi^2}\left[\omega(u_{j+\frac12})(x_{j+1} - x_j) - \omega(u_{j-\frac12})(x_j - x_{j-1})\right], \quad 1 \le j \le J \tag{3.2}$$

where $\Delta\xi = 1/(J+1)$ is the step size in Ω_c. Solving (3.2) with boundary conditions $x_0 = a$ and $x_{J+1} = b$ leads to a new grid in the physical domain Ω_p. Some advantages of using the approach (3.1) and (3.2) to solve the mesh redistribution equation (2.8) will be seen from Theorems 3.1 and 3.3.

3.2. Solution-updating on new grids. After obtaining the new grid $\{\widetilde{x}_j\}$, we need to update u at the grid point $\widetilde{x}_{j+\frac12} = (\widetilde{x}_j + \widetilde{x}_{j+1})/2$ based on the information of $\{x_{j+\frac12}, \widetilde{x}_{j+\frac12}, u_{j+\frac12}\}$. The traditional way to do it is using the conventional interpolation:

$$\widetilde{u}_{j+\frac12} = u_{k+\frac12} + \frac{u_{k+\frac12} - u_{k-\frac12}}{x_{k+\frac12} - x_{k-\frac12}}(\widetilde{x}_{j+\frac12} - x_{k+\frac12}), \quad \text{if } \widetilde{x}_{j+\frac12} \in [x_{k-\frac12}, x_{k+\frac12}]. \tag{3.3}$$

Since the monitor function ω is dependent on the underlying solution u, the grid redistribution equations (3.2)-(3.3) form a nonlinear system. It is therefore natural to make several iterations to solve (3.2)-(3.3) in order to gain better controls of the grid distribution near the regions where solution u has large gradient. In solving hyperbolic conservation laws with strong discontinuities (e.g. shock), the iteration techniques based on (3.2)-(3.3) have been employed and it is found that the results for the solution and the mesh are not satisfactory. The main problem is that the linear interpolation (3.3) can not preserve the conservation of the mass which, by the Lax-Wendroff theory, is an essential requirement for a good numerical scheme to hyperbolic conservation laws.

In the following we will introduce a new method to update u, by noting that the mass-conservation is an essential requirement of hyperbolic conservation laws. To begin with, assume the difference between $\tilde{x}_{j+\frac{1}{2}}$ and $x_{j+\frac{1}{2}}$ is small. Let $\widetilde{u}_{j+\frac{1}{2}}$ and $u_{j+\frac{1}{2}}$ be cell averages of solution $u(x)$ over interval $[\widetilde{x}_j, \widetilde{x}_{j+1}]$ and $[x_j, x_{j+1}]$, respectively. We will derive a formula for $\widetilde{u}_{j+\frac{1}{2}}$ by using the perturbation method. If $\widetilde{x} = x - c(x)$ with a small speed $c(x)$, i.e. $|c(x)| \ll 1$, then we have

$$\begin{aligned}
\int_{\widetilde{x}_j}^{\widetilde{x}_{j+1}} u(\widetilde{x})\, d\widetilde{x} &= \int_{x_j}^{x_{j+1}} u(x - c(x))(1 - c'(x))\, dx \\
&\approx \int_{x_j}^{x_{j+1}} (u(x) - c(x)u_x(x))(1 - c'(x))\, dx \\
&\approx \int_{x_j}^{x_{j+1}} (u(x) - (cu)_x)\, dx \\
&= \int_{x_j}^{x_{j+1}} u(x)\, dx - ((cu)_{j+1} - (cu)_j)
\end{aligned} \tag{3.4}$$

where we have neglected higher order terms, and $(cu)_j$ denotes the value of cu at the j-th cell interface. The following conservative-interpolation follows from (3.4):

$$\Delta\widetilde{x}_{j+\frac{1}{2}}\widetilde{u}_{j+\frac{1}{2}} = \Delta x_{j+\frac{1}{2}} u_{j+\frac{1}{2}} - \big((cu)_{j+1} - (cu)_j\big). \tag{3.5}$$

where $\Delta\widetilde{x}_{j+\frac{1}{2}} = \widetilde{x}_{j+1} - \widetilde{x}_j$ and $c_j = x_j - \widetilde{x}_j$. Note that the above solution-updating method guarantees the conservation of mass in the following sense:

$$\sum_j \Delta\widetilde{x}_{j+\frac{1}{2}}\widetilde{u}_{j+\frac{1}{2}} = \sum_j \Delta x_{j+\frac{1}{2}} u_{j+\frac{1}{2}}. \tag{3.6}$$

The linear flux cu in (3.5) will be approximated by some upwinding numerical fluxes, see (3.11) below.

It can be justified that if the function u is suitably smooth then the size of the moving speed $c(x)$ is small. It is known that the first and the second derivatives of the parabolic type equation (3.1) are bounded, provided that the initial data and the monitor function satisfy some regularity requirements. By the definition of $c(x)$, we have

$$\begin{aligned}
c(x) &= x - \widetilde{x} = -(x_\tau)\Delta\tau = \mathcal{O}(\Delta\tau)\,, \\
c'(x) &= 1 - \tilde{x}_x = 1 - \frac{\widetilde{x}_\xi}{x_\xi} = -\frac{x_{\xi\tau}}{x_\xi}\Delta\tau = \mathcal{O}(\Delta\tau)\,,
\end{aligned}$$

which indicate that the moving speed at each cell is indeed very small.

3.3. Solution procedure. Our solution procedure is based on two independent parts: a mesh-redistribution algorithm and a solution algorithm. The first part will be based on an iteration procedure using (3.2) and (3.5). The second part will be independent of the first one, which can be any of the standard codes for solving the given PDEs, such as ENO schemes [29, 37], central schemes [16, 25], relaxation schemes [15, 24, 32], BGK schemes [36, 33], and several other types of high-resolutions methods, see, e.g., [19, 14, 17]. The solution procedure can be illustrated by the following flowchart:

Algorithm 0.

Step 1: Given a uniform (fixed) partition of the logical domain Ω_c, and use the equidistribution principle (2.8) to generate an initial partition $x_j^{[0]} := x_j$ of the physical domain Ω_p. Then compute the grid values $u_{j+\frac{1}{2}}^{[0]}$ based on the cell average for the initial data $u(x,0)$.

$\Downarrow$

Step 2: Move grid $\{x_j^{[\nu]}\}$ to $\{x_j^{[\nu+1]}\}$ based on scheme (3.2) and compute $\{u_{j+\frac{1}{2}}^{[\nu+1]}\}$ on the new grid based on scheme (3.5), for $\nu \geq 0$. Repeat the updating procedure for a fixed number of iterations or until $\|x^{[\nu+1]} - x^{[\nu]}\| \leq \epsilon$. The mesh-redistribution scheme (3.2) can be also replaced by an Gauss-Seidel iteration procedure (3.31), as discussed at the end of this section.

$\Downarrow$

Step 3: Evolute the underlying PDEs by using high-resolution finite volume methods on the mesh $\{x_j^{[\nu+1]}\}$ to obtain the numerical approximations $u_{j+\frac{1}{2}}^{n+1}$ at the time level t_{n+1}.

$\Downarrow$

Step 4: If $t_{n+1} \leq T$, then let $u_{j+\frac{1}{2}}^{[0]} := u_{j+\frac{1}{2}}^{n+1}$ and $x_j^{[0]} := x_j^{[\nu+1]}$ and go to **Step 2**.

3.3.1. Some discussions on step 2. A new mesh $x_j^{[\nu+1]}$ is obtained by using (3.2):

$$x_j^{[\nu+1]} = \alpha_{j+\frac{1}{2}} x_{j+1}^{[\nu]} + (1 - \alpha_{j+\frac{1}{2}} - \alpha_{j-\frac{1}{2}}) x_j^{[\nu]} + \alpha_{j-\frac{1}{2}} x_{j-1}^{[\nu]} \tag{3.7}$$

where

$$\alpha_{j+\frac{1}{2}} = \frac{\Delta\tau}{\Delta\xi^2}\omega(u_{j+\frac{1}{2}}^{[\nu]}).$$

The above equation is solved subject to the following stability condition:

$$\max_j \alpha_{j+\frac{1}{2}} \leq \frac{1}{2}. \tag{3.8}$$

Next, numerical solutions are updated on the new grids $\{x_j^{[\nu+1]}\}$ (note at the same time level) by using (3.5):

$$u_{j+\frac{1}{2}}^{[\nu+1]} = \beta_j^{[\nu]} u_{j+\frac{1}{2}}^{[\nu]} - \gamma_j^{[\nu]} \left((\widehat{cu})_{j+1}^{[\nu]} - (\widehat{cu})_j^{[\nu]} \right), \tag{3.9}$$

where

$$\gamma_j^{[\nu]} = \left(x_{j+1}^{[\nu+1]} - x_j^{[\nu+1]} \right)^{-1}, \qquad \beta_j^{[\nu]} = \gamma_j^{[\nu]} \cdot \left(x_{j+1}^{[\nu]} - x_j^{[\nu]} \right), \tag{3.10}$$

and the numerical flux $\widehat{cu}_j$ is defined by

$$(\widehat{cu})_j = \frac{c_j}{2}(u_{j+\frac{1}{2}} + u_{j-\frac{1}{2}}) - \frac{|c_j|}{2}(u_{j+\frac{1}{2}} - u_{j-\frac{1}{2}}). \tag{3.11}$$

The wave speed c_j above is defined by $c_j^{[\nu]} = x_j^{[\nu]} - x_j^{[\nu+1]}$.

Remark 3.1. In our numerical computation, the first-order numerical flux $(\widehat{cu})_j$ defined by (3.11) will be replaced by a second-order one as the following:

$$(\widehat{cu})_j = \frac{c_j}{2}(u_j^+ + u_j^-) - \frac{|c_j|}{2}(u_j^+ - u_j^-) \tag{3.12}$$

where u_j^+ and u_j^- will be defined by (3.18) below.

REMARK 3.2. In practice it is common to use some temporal or spatial *smoothing* on the monitor function to obtain smoother meshes. One of the reasons for using smoothing is to avoid very singular mesh and/or large approximation error around the stiff solution areas. In this work, we apply the following low pass filter to smooth the monitor:

$$\omega_{j+\frac{1}{2}} \leftarrow \frac{1}{4}(\omega_{j+\frac{3}{2}} + 2\omega_{j+\frac{1}{2}} + \omega_{j-\frac{1}{2}}), \tag{3.13}$$

where $\omega_{j+\frac{1}{2}} = \omega(u_{j+\frac{1}{2}})$.

3.3.2. Some discussions on Step 3. This step is independent of Step 2 and as a results it can be done by using any efficient modern numerical techniques for hyperbolic conservation laws. As an example, we consider a second-order finite-volume approach to solve the 1D scalar hyperbolic conservation laws

$$u_t + f(u)_x = 0\,. \tag{3.14}$$

Integrating (3.14) over the control volume $[t_n, t_{n+1}) \times [x_j^{[\nu+1]}, x_{j+1}^{[\nu+1]}]$ leads to the following (explicit) finite volume method

$$u_{j+\frac{1}{2}}^{n+1} = u_{j+\frac{1}{2}}^{n} - \frac{t_{n+1} - t_n}{x_{j+1}^{[\nu+1]} - x_j^{[\nu+1]}} \left(\widehat{f}_{j+1}^{n} - \widehat{f}_{j}^{n} \right), \tag{3.15}$$

where $\widehat{f}_j^n$ are some appropriate numerical flux satisfying

$$\widehat{f}_j^n = \widehat{f}(u_j^{n,-}, u_j^{n,+}), \qquad \widehat{f}(u,u) = f(u). \tag{3.16}$$

An example of the numerical flux is the Lax-Friedrichs flux:

$$\widehat{f}(a,b) = \frac{1}{2}\big[f(a) + f(b) - \max_u \{|f_u|\}\,(b-a)\big]. \tag{3.17}$$

In (3.16), $u_j^{n,\pm}$ are defined by

$$u_j^{n,\pm} = u_{j\pm\frac{1}{2}}^{n} + \frac{1}{2}(x_j^{[\nu+1]} - x_{j\pm 1}^{[\nu+1]})\tilde{S}_{j\pm\frac{1}{2}}, \tag{3.18}$$

where $\tilde{S}_{j+\frac{1}{2}}$ is an approximation of the slope u_x at $x_{j+\frac{1}{2}}^{[\nu+1]}$, defined by

$$\tilde{S}_{j+\frac{1}{2}} = \left(\mathrm{sign}(\tilde{S}_{j+\frac{1}{2}}^{+}) + \mathrm{sign}(\tilde{S}_{j+\frac{1}{2}}^{-})\right) \frac{|\tilde{S}_{j+\frac{1}{2}}^{+}\tilde{S}_{j+\frac{1}{2}}^{-}|}{|\tilde{S}_{j+\frac{1}{2}}^{+}| + |\tilde{S}_{j+\frac{1}{2}}^{-}|}, \tag{3.19}$$

with

$$\tilde{S}_{j+\frac{1}{2}}^{+} = \frac{u_{j+\frac{3}{2}}^{n} - u_{j+\frac{1}{2}}^{n}}{x_{j+\frac{3}{2}}^{[\nu+1]} - x_{j+\frac{1}{2}}^{[\nu+1]}}, \qquad \tilde{S}_{j+\frac{1}{2}}^{-} = \frac{u_{j+\frac{1}{2}}^{n} - u_{j-\frac{1}{2}}^{n}}{x_{j+\frac{1}{2}}^{[\nu+1]} - x_{j-\frac{1}{2}}^{[\nu+1]}}\,.$$

The MUSCL-type finite volume method (3.15)-(3.17), which is of second order accuracy in smooth regions, will be applied to the 1D numerical experiments.

3.4. Some theoretical results on the adaptive mesh solutions. In 1D, some good theoretical guarantees for the numerical grids can be obtained. In the following, we prove some theoretical results for the mesh-redistribution equation (3.7) and the solution-updating equation (3.9). We first demonstrate that the new mesh $x^{[\nu+1]}$ generated by (3.7) keeps the monotonic order of $x^{[\nu]}$.

THEOREM 3.1. *Assume* $x_{j+1}^{[\nu]} > x_j^{[\nu]}$ *for* $0 \le j \le J$. *If the new mesh* $x^{[\nu+1]}$ *is obtained by using* (3.7), *then* $x_{j-1}^{[\nu+1]} < x_j^{[\nu]} < x_{j+1}^{[\nu+1]}$ *for* $1 \le j \le J$ *and* $x_{j+1}^{[\nu+1]} > x_j^{[\nu+1]}$ *for* $0 \le j \le J$.

Proof. Using the stability condition (3.8) gives $1 - \alpha_{j+\frac{1}{2}} - \alpha_{j-\frac{1}{2}} \ge 0$. Moreover, $\alpha_{j\pm\frac{1}{2}}$ are all positive. Therefore, it follows from the assumption $x_{j+1}^{[\nu]} > x_j^{[\nu]}$ and (3.7) that $x_{j-1}^{[\nu+1]} < x_j^{[\nu]} < x_{j+1}^{[\nu+1]}$. We now re-write (3.7) into the following form:

$$x_j^{[\nu+1]} = \alpha_{j+\frac{1}{2}} \Delta x_{j+\frac{1}{2}}^{[\nu]} + x_j^{[\nu]} - \alpha_{j-\frac{1}{2}} \Delta x_{j-\frac{1}{2}}^{[\nu]}, \tag{3.20}$$

where $\Delta x_{j+\frac{1}{2}} = x_{j+1} - x_j$. It follows from the above equation that

$$\Delta x_{j-\frac{1}{2}}^{[\nu+1]} = \alpha_{j+\frac{1}{2}} \Delta x_{j+\frac{1}{2}}^{[\nu]} + (1 - 2\alpha_{j-\frac{1}{2}}) \Delta x_{j-\frac{1}{2}}^{[\nu]} + \alpha_{j-\frac{3}{2}} \Delta x_{j-\frac{3}{2}}^{[\nu]}.$$

Since the two coefficients of the right hand side are positive and one non-negative (due to the stability condition (3.8)), the assumption $x_{j+1}^{[\nu]} > x_j^{[\nu]}$ yields $\Delta x_{j-\frac{1}{2}}^{[\nu+1]} > 0$. This shows that $x_{j+1}^{[\nu+1]} > x_j^{[\nu+1]}$ for $0 \le j \le J$. ∎

REMARK 3.3. A consequence of Theorem *3.1* is that $x_j^{[\nu+1]} \in (x_{j-1}^{[\nu]}, x_{j+1}^{[\nu]})$, which implies that the speed of mesh-moving is finite. This is important in better controlling the grid distribution near the regions of large solution gradients.

Next, we provide a necessary condition under which the updated solution $u_{j+\frac{1}{2}}^{[\nu+1]}$ is to satisfy the TVD property.

LEMMA 3.2. *If* $x_{j-1}^{[\nu+1]} \le x_j^{[\nu]} \le x_{j+1}^{[\nu+1]}$ *and* $x_{j+1}^{[\nu+1]} > x_j^{[\nu+1]}$, *then the solution-updating scheme* (3.9)-(3.11) *satisfies*

$$\mathsf{TV}\big(u^{[\nu+1]}\big) \le \mathsf{TV}\big(u^{[\nu]}\big), \qquad \mathsf{TV}(u) := \sum_j \big|u_{j+\frac{1}{2}} - u_{j-\frac{1}{2}}\big|.$$

Proof. For ease of notation we denote $\tilde{x} = x^{[\nu+1]}, x = x^{[\nu]}, \tilde{u} = u^{[\nu+1]}, u = u^{[\nu]}$. Note that $c_{j+1} - c_j = \Delta x_{j+\frac{1}{2}} - \Delta \tilde{x}_{j+\frac{1}{2}}$. This fact, together with the scheme (3.9) and the numerical flux (3.11), gives

$$\begin{aligned} \Delta \tilde{x}_{j+\frac{1}{2}} \tilde{u}_{j+\frac{1}{2}} &= \big(c_{j+1} - c_j + \Delta \tilde{x}_{j+\frac{1}{2}}\big) u_{j+\frac{1}{2}} + \frac{1}{2}\big(|c_{j+1}| - c_{j+1}\big) u_{j+\frac{3}{2}} \\ &\quad + \frac{1}{2}\big(c_j - |c_j| - c_{j+1} - |c_{j+1}|\big) u_{j+\frac{1}{2}} + \frac{1}{2}\big(|c_j| + c_j\big) u_{j-\frac{1}{2}} \\ &= \Delta \tilde{x}_{j+\frac{1}{2}} u_{j+\frac{1}{2}} + m_{j+1} u_{j+\frac{3}{2}} - m_{j+1} u_{j+\frac{1}{2}} - M_j u_{j+\frac{1}{2}} + M_j u_{j-\frac{1}{2}}, \end{aligned}$$

where $M_j = \max(0, c_j)$ and $m_j = -\min(0, c_j)$. Note both M_j and m_j are nonnegative. It follows from the above result that

$$\tilde{u}_{j+\frac{1}{2}} = u_{j+\frac{1}{2}} + \frac{m_{j+1}}{\Delta \tilde{x}_{j+\frac{1}{2}}} \delta u_{j+\frac{3}{2}} - \frac{M_j}{\Delta \tilde{x}_{j+\frac{1}{2}}} \delta u_{j+\frac{1}{2}}, \tag{3.21}$$

where $\delta u_{j+\frac{1}{2}} = u_{j+\frac{1}{2}} - u_{j-\frac{1}{2}}$. It follows from (3.21) that

$$\delta \tilde{u}_{j+\frac{1}{2}} = \delta u_{j+\frac{1}{2}} + \frac{m_{j+1}}{\Delta \tilde{x}_{j+\frac{1}{2}}} \delta u_{j+\frac{3}{2}} - \left(\frac{M_j}{\Delta \tilde{x}_{j+\frac{1}{2}}} + \frac{m_j}{\Delta \tilde{x}_{j-\frac{1}{2}}} \right) \delta u_{j+\frac{1}{2}} + \frac{M_{j-1}}{\Delta \tilde{x}_{j-\frac{1}{2}}} \delta u_{j-\frac{1}{2}},$$

which gives

$$\begin{aligned} \sum_j |\delta \tilde{u}_{j+\frac{1}{2}}| \le \sum_j \frac{m_j}{\Delta \tilde{x}_{j-\frac{1}{2}}} |\delta u_{j+\frac{1}{2}}| &+ \sum_j \left| 1 - \frac{M_j}{\Delta \tilde{x}_{j+\frac{1}{2}}} - \frac{m_j}{\Delta \tilde{x}_{j-\frac{1}{2}}} \right| |\delta u_{j+\frac{1}{2}}| \\ &+ \sum_j \frac{M_j}{\Delta \tilde{x}_{j+\frac{1}{2}}} |\delta u_{j+\frac{1}{2}}|. \end{aligned} \tag{3.22}$$

It can be verified by using the definition of c_j that the condition $\tilde{x}_{j-1} \le x_j \le \tilde{x}_{j+1}$ is equivalent to $-\Delta\tilde{x}_{j-\frac{1}{2}} \le c_j \le \Delta\tilde{x}_{j+\frac{1}{2}}$. This fact, together with the observation that $M_j = 0$ when $c_j \le 0$ and $m_j = 0$ when $c_j \ge 0$, yields

$$1 - \frac{M_j}{\Delta\tilde{x}_{j+\frac{1}{2}}} - \frac{m_j}{\Delta\tilde{x}_{j-\frac{1}{2}}} \ge 0. \tag{3.23}$$

It follows from (3.22) and (3.23) that $\mathsf{TV}(\tilde{u}) \le \mathsf{TV}(u)$. ▯

With the above two ingredients, the following TVD property for Step 2 of **Algorithm 0** is established.

THEOREM 3.3. *The iterated mesh and solution* $\{x^{[\nu+1]}, u^{[\nu+1]}\}$ *generated by* (3.7)-(3.11) *satisfies* $\mathsf{TV}(u^{[\nu+1]}) \le \mathsf{TV}(u^{[\nu]})$.

REMARK 3.4. If the *PDE* solver in Step *3* of **Algorithm 0** is *TVB* (i.e. TV-bounded) (or *TVD*), then the above theorem guarantees the *TVB* (or *TVD*) property of the moving-mesh solution at any time level. It can be also proved in similar manners that the l^∞- and the l^1-stabilities are also preserved.

THEOREM 3.4. *The moving-mesh solution generated by* Algorithm 0 *is a weak solution of the conservation law* (3.14).

Proof. Without lose of generality we assume that $u(x, 0)$ has a compact support and that only one iteration is used in Step 2 of Algorithm 0. Then given $\{x_j^n, u_{j+\frac{1}{2}}^n\}$, the solution $\{x_j^{n+1}, u_{j+\frac{1}{2}}^{n+1}\}$ is computed by the following operator-splitting type algorithm

$$U_{j+\frac{1}{2}}^n = u_{j+\frac{1}{2}}^n - \frac{\Delta t_n}{\Delta x_{j+\frac{1}{2}}^n}\left(f_{j+1}^n - f_j^n\right), \tag{3.24}$$

$$x_j^{n+1} = x_j^n + \frac{\Delta\tau}{\Delta\xi^2}\left[\omega(U_{j+\frac{1}{2}}^n)(x_{j+1}^n - x_j^n) - \omega(U_{j-\frac{1}{2}}^n)(x_j^n - x_{j-1}^n)\right], \tag{3.25}$$

$$\Delta x_{j+\frac{1}{2}}^{n+1} u_{j+\frac{1}{2}}^{n+1} = \Delta x_{j+\frac{1}{2}}^n U_{j+\frac{1}{2}}^n - \Big((cU^n)_{j+1} - (cU^n)_j\Big) \tag{3.26}$$

where $c_j = x_j^n - x_j^{n+1}$, and the numerical flux f_j^n satisfies the consistency requirement. Multiplying a test function $\phi \in C_0^\infty(\mathbf{R} \times (0, T])$ on the first equation above gives

$$\Delta x_{j+\frac{1}{2}}^n U_{j+\frac{1}{2}}^n \phi(x_j^n, t_n) = \Delta x_{j+\frac{1}{2}}^n u_{j+\frac{1}{2}}^n \phi(x_j^n, t_n) - \Delta t_n\left(f_{j+1}^n - f_j^n\right)\phi(x_j^n, t_n)$$

which, together with the interpolation step (3.26), gives

$$\begin{aligned} &\Delta x_{j+\frac{1}{2}}^{n+1} u_{j+\frac{1}{2}}^{n+1} \phi(x_j^n, t_n) + \Big((cU^n)_{j+1} - (cU^n)_j\Big)\phi(x_j^n, t_n) \\ &= \Delta x_{j+\frac{1}{2}}^n u_{j+\frac{1}{2}}^n \phi(x_j^n, t_n) - \Delta t_n\left(f_{j+1}^n - f_j^n\right)\phi(x_j^n, t_n). \end{aligned} \tag{3.27}$$

Making the standard summation tricks yields that

$$\begin{aligned} &\sum_j \sum_{n=0}^N \left[\Delta x_{j+\frac{1}{2}}^{n+1} u_{j+\frac{1}{2}}^{n+1} - \Delta x_{j+\frac{1}{2}}^n u_{j+\frac{1}{2}}^n\right]\phi(x_j^n, t_n) \\ &= -\sum_j \sum_{n=0}^N \Big((cU^n)_{j+1} - (cU^n)_j\Big)\phi(x_j^n, t_n) - \sum_{n=0}^N \sum_j \Delta t_n\left(f_{j+1}^n - f_j^n\right)\phi(x_j^n, t_n) \end{aligned}$$

and then

$$\begin{aligned} &-\sum_j \Delta x_{j+\frac{1}{2}}^0 u_{j+\frac{1}{2}}^0 \phi(x_j^0, 0) - \sum_j \sum_{n=1}^N \left[\phi(x_j^n, t_n) - \phi(x_j^{n-1}, t_{n-1})\right]\Delta x_{j+\frac{1}{2}}^n u_{j+\frac{1}{2}}^n \\ &= \sum_j \sum_{n=0}^N \left[\phi(x_j^n, t_n) - \phi(x_{j-1}^n, t_n)\right](x_j^n - x_j^{n+1})U_j^n + \sum_{n=0}^N \sum_j \Delta t_n\left[\phi(x_j^n, t_n) - \phi(x_{j-1}^n, t_n)\right] f_j^n \end{aligned} \tag{3.28}$$

where we have used the fact $\phi(x, t_N) = 0$ with $t_N = T$. We can show that $\Delta x^0_{j+\frac{1}{2}} \to 0$ as $L \to \infty$. Without lose of generality, assume that the monitor function is the one associated with the equi-distribution principle, i.e. $\omega(u) = \sqrt{1 + u_x^2}$. Then

$$L = \sqrt{1 + u_x^2(x_j^0, 0)}\Delta x^0_{j+\frac{1}{2}}, \qquad 0 \le j \le J-1 \tag{3.29}$$

is a constant independent of j. It follows from the definition of L that

$$L \le \left(1 + |u_x(x_j^0, 0)|\right) \Delta x^0_{j+\frac{1}{2}}, \qquad 0 \le j \le J-1$$

which gives

$$JL \le \text{the size of } u_0\text{'s support} + TV(u^0).$$

This, together with the definition (3.29), leads to

$$\Delta x^0_{j+\frac{1}{2}} \le Const.J^{-1} \to 0, \qquad \text{as} \quad J \to \infty. \tag{3.30}$$

Moreover, since $\{x_j^n\}$, with $n \ge 1$, are obtained by solving a parabolic equation, we have $\Delta x^n_{j+\frac{1}{2}} \sim \mathcal{O}(\Delta\xi) \to 0$ for $n \ge 1$. Taking limit on both sides of (3.28) leads to

$$-\int u(x,0)\phi(x,0)dx - \int\int(\phi_x x_t + \phi_t)udxdt = -\int\int \phi_x x_t udxdt + \int\int \phi_x f(u)dxdt$$

provided that the numerical solution is convergent to u almost everywhere. The above result leads to

$$\int\int(\phi_t u + \phi_x f(u))dxdt + \int u(x,0)\phi(x,0)dx = 0$$

which indicates that moving mesh solution is indeed a weak solution of the underlying conservation laws. □

3.5. Grid-motion by using the Gauss-Seidel iteration. In practice, we also use the following Gauss-Seidel type iteration to solve the mesh-moving equation (2.8):

$$\omega(u^{[\nu]}_{j+\frac{1}{2}})\left(x^{[\nu]}_{j+1} - x^{[\nu+1]}_j\right) - \omega(u^{[\nu]}_{j-\frac{1}{2}})\left(x^{[\nu+1]}_j - x^{[\nu+1]}_{j-1}\right) = 0. \tag{3.31}$$

It can also be demonstrated that the new mesh $x^{[\nu+1]}$ generated by (3.31) keeps the monotonic order of $x^{[\nu]}$.

THEOREM 3.5. *Assume $x^{[\nu]}_{j+1} > x^{[\nu]}_j$ for $0 \le j \le J$. If the new mesh $x^{[\nu+1]}$ is obtained by using the Gauss-Seidel iterative scheme* (3.31)*, then $x^{[\nu+1]}_j > x^{[\nu+1]}_{j-1}$ for $1 \le j \le J+1$. Moreover, $x^{[\nu]}_j > x^{[\nu+1]}_{j-1}$ for $1 \le j \le J+1$.*

Proof. We again denote $\tilde{x} = x^{[\nu+1]}, x = x^{[\nu]}$. It follows from (3.31) that

$$-\alpha_j x_{j+1} + \tilde{x}_j - \beta_j \tilde{x}_{j-1} = 0 \tag{3.32}$$

where $\alpha_j > 0, \beta_j > 0$ and $\alpha_j + \beta_j = 1$. It follows from the above equation that

$$\tilde{x}_j - x_{j+1} - \beta_j(\tilde{x}_{j-1} - x_j) = \beta_j(x_j - x_{j+1}) \le 0,$$

which gives that

$$\tilde{x}_j - x_{j+1} \le \left(\prod_{k=1}^{j}\beta_k\right)(\tilde{x}_0 - x_1) = \left(\prod_{k=1}^{j}\beta_k\right)(x_0 - x_1) < 0.$$

The above result yields $\tilde{x}_j < x_{j+1}$, which, together with (3.31), also leads to $\tilde{x}_j > \tilde{x}_{j-1}$. □

If we can further show that $x_j^{[\nu]} \le x_{j+1}^{[\nu+1]}$ for the Gauss-Seidel iteration (3.31), then based on Lemma 3.2 the solution-updating scheme (3.9) together with the mesh-redistribution scheme (3.31) will also satisfy the TVD property, i.e. $\mathsf{TV}(u^{[\nu+1]}) \le \mathsf{TV}(u^{[\nu]})$. However, it seems unlikely that $x_j^{[\nu]} \le x_{j+1}^{[\nu+1]}$ holds for (3.31) in general situations. On the other hand, the combination of (3.9) and (3.31) has been employed in our numerical experiments and the numerical results are quite satisfactory. Therefore, both (3.7) and (3.31) can be used in Step 2 of **Algorithm 0** to re-distribute grid points. In most test problems considered in this work, the grid updating procedure at each time step takes 5 full Gauss-Seidal iterations, although the difference between solutions with 3 and 5 iterations is very small.

4. 2D algorithm.

One of the advantages of the adaptive mesh methods described in the last section is that it can be naturally extended to 2D. In the following, we briefly discuss this extension, together with the boundary point re-distribution technique which is necessary for 2D mesh redistribution.

4.1. A conservative solution-updating method.

In 2D, the logical domain $\bar{\Omega}_c = \{(\xi,\eta) | 0 \le \xi \le 1, 0 \le \eta \le 1\}$ is covered by the square meshes:

$$\left\{(\xi_j,\eta_k)\,\Big|\, \xi_j = j/(J_x+1),\ \eta_k = k/(J_y+1);\ 0 \le j \le J_x+1, 0 \le k \le J_y+1\right\}.$$

Correspondingly, the numerical approximations to $x = x(\xi,\eta)$ and $y = y(\xi,\eta)$ are denoted by $x_{j,k} = x(\xi_j,\eta_k)$ and $y_{j,k} = y(\xi_j,\eta_k)$. As in 1D case, we will derive a *conservative* scheme to evaluate approximate values at new grid points. Let $A_{j+\frac{1}{2},k+\frac{1}{2}}$ be a control volume as shown in Figure 4.1. Let $\widetilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}$ denote the quadrangle of the finite control volume with four vertices $(\widetilde{x}_{j+p,k+q}, \widetilde{y}_{j+p,k+q})$, $0 \le p,q \le 1$, which is of setup similar to Figure 4.1.

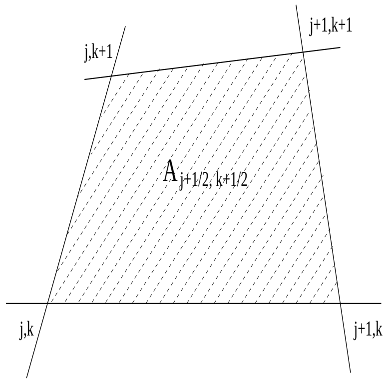

FIG. 4.1. A control volume.

Assume that $\widetilde{u}_{j+\frac{1}{2},k+\frac{1}{2}}$ and $u_{j+\frac{1}{2},k+\frac{1}{2}}$ are cell average of $u(x,,y)$ over $\widetilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}$ and $A_{j+\frac{1}{2},k+\frac{1}{2}}$, respectively. Similar to the 1D case, we use the perturbation method to evaluate the numerical approximation on the resulting new grids $(\widetilde{x}_{j,k}, \widetilde{y}_{j,k})$. If $(\widetilde{x},\widetilde{y}) = (x - c^x(x,y),\, y - c^y(x,y))$ with small magnitudes of speed (c^x, c^y), then we have

$$\int_{\widetilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}} u(\widetilde{x},\widetilde{y})\, d\widetilde{x}d\widetilde{y}$$
$$= \int_{A_{j+\frac{1}{2},k+\frac{1}{2}}} u(x-c^x,\, y-c^y) \det\left(\frac{\partial(\tilde{x},\tilde{y})}{\partial(x,y)}\right) dxdy$$

$$
\begin{aligned}
&\approx \int_{A_{j+\frac12,k+\frac12}} (u(x,y) - c^x u_x - c^y u_y)(1 - c^x_x - c^y_y)\ dxdy \\
&\approx \int_{A_{j+\frac12,k+\frac12}} \Big[u(x,y) - c^x u_x - c^y u_y - c^x_x u - c^y_y u\Big] dxdy \\
&= \int_{A_{j+\frac12,k+\frac12}} \Big[u(x,y) - (c^x u)_x - (c^y u)_y\Big]\ dxdy \\
&= \int_{A_{j+\frac12,k+\frac12}} u(x,y)\ dxdy - \Big[(c_n u)_{j+1,k+\frac12} + (c_n u)_{j,k+\frac12}\Big] \\
&\qquad - \Big[(c_n u)_{j+\frac12,k+1} + (c_n u)_{j+\frac12,k}\Big],
\end{aligned}
\tag{4.1}
$$

where we have neglected higher order terms, $c_n := c^x n_x + c^y n_y$ with (n_x, n_y) the unit normal, and $(c_n u)_{j,k+\frac12}$ and $(c_n u)_{j+\frac12,k}$ denote the values of $c_n u$ at corresponding surface of the control volume $A_{j+\frac12,k+\frac12}$. From (4.1), we obtain a conservative-interpolation:

$$
\begin{aligned}
&|\widetilde{A}_{j+\frac12,k+\frac12}|\widetilde{u}_{j+\frac12,k+\frac12} = |A_{j+\frac12,k+\frac12}|u_{j+\frac12,k+\frac12} \\
&\quad - \Big[(c_n u)_{j+1,k+\frac12} + (c_n u)_{j,k+\frac12}\Big] - \Big[(c_n u)_{j+\frac12,k+1} + (c_n u)_{j+\frac12,k}\Big],
\end{aligned}
\tag{4.2}
$$

where $|\widetilde{A}|$ and $|A|$ denote the areas of control volume $\widetilde{A}$ and A, respectively. It can be verified that the above solution-updating scheme satisfies the mass-conservation:

$$
\sum_{j,k} |\widetilde{A}_{j+\frac12,k+\frac12}|\widetilde{u}_{j+\frac12,k+\frac12} = \sum_{j,k} |A_{j+\frac12,k+\frac12}|u_{j+\frac12,k+\frac12}.
\tag{4.3}
$$

4.2. Solution procedure. The solution procedure of our adaptive mesh strategy for two-dimensional hyperbolic problems is almost the same as **Algorithm 0** provided in Section 3.3. Some details of the steps used for our 2D algorithm are given below.

Step i Give an initial partition $\vec{z}^{[0]}_{j,k} = \big(x^{[0]}_{j,k}, y^{[0]}_{j,k}\big) := (x_{j,k}, y_{j,k})$ of the physical domain Ω_p and a uniform (fixed) partition of the logical domain Ω_c, and compute grid values $u^{[0]}_{j+\frac12,k+\frac12}$ by cell average over control volume $A_{j+\frac12,k+\frac12}$ based on the initial data $u(x,y,0)$.

Step ii For $\nu = 0, 1, 2, \ldots$, do the following:

(a). Move grid $\vec{z}^{[\nu]}_{j,k} = \{(x^{[\nu]}_{j,k}, y^{[\nu]}_{j,k})\}$ to $\vec{z}^{[\nu+1]}_{j,k} = \{(x^{[\nu+1]}_{j,k}, y^{[\nu+1]}_{j,k})\}$ by solving $\vec{z}_\tau = (\omega \vec{z}_\xi)_\xi + (\omega \vec{z}_\eta)_\eta$ with the conventional explicit scheme. This step can be also done by solving $(\omega \vec{z}_\xi)_\xi + (\omega \vec{z}_\eta)_\eta = 0$ with the following Gauss-Seidel iteration:

$$
\begin{aligned}
&\alpha_{j+\frac12,k}\big(\vec{z}^{[\nu]}_{j+1,k} - \vec{z}^{[\nu+1]}_{j,k}\big) - \alpha_{j-\frac12,k}\big(\vec{z}^{[\nu+1]}_{j,k} - \vec{z}^{[\nu+1]}_{j-1,k}\big) \\
&\quad + \beta_{j,k+\frac12}\big(\vec{z}^{[\nu]}_{j,k+1} - \vec{z}^{[\nu+1]}_{j,k}\big) - \beta_{j,k-\frac12}\big(\vec{z}^{[\nu+1]}_{j,k} - \vec{z}^{[\nu+1]}_{j,k-1}\big) = 0,
\end{aligned}
\tag{4.4}
$$

for $1 \le j \le J_x$ and $1 \le k \le J_y$, where

$$
\begin{aligned}
\alpha_{j\pm\frac12,k} &= \omega\big(u^{[\nu]}_{j\pm\frac12,k}\big) = \omega\big(\tfrac12(u^{[\nu]}_{j\pm\frac12,k+\frac12} + u^{[\nu]}_{j\pm\frac12,k-\frac12})\big), \\
\beta_{j,k\pm\frac12} &= \omega\big(u^{[\nu]}_{j,k\pm\frac12}\big) = \omega\big(\tfrac12(u^{[\nu]}_{j+\frac12,k\pm\frac12} + u^{[\nu]}_{j-\frac12,k\pm\frac12})\big).
\end{aligned}
$$

(b). Compute $\{u^{[\nu+1]}_{j+\frac12,k+\frac12}\}$ on the new grids based on the conservative-interpolation (4.2). The approximations for c^x, c^y etc are direct extensions of those defined for 1D.

(c). Repeat the updating procedure (a) and (b) for a fixed number of iterations (say, 3 or 5) or until $\|\vec{z}^{[\nu+1]} - \vec{z}^{[\nu]}\| \le \epsilon$.

Step iii Evolute the underlying PDEs by using 2D high-resolution finite volume methods on the mesh $\{(x_{j,k}^{[\nu+1]}, y_{j,k}^{[\nu+1]})\}$ to obtain the numerical approximations $u_{j+\frac{1}{2},k+\frac{1}{2}}^{n+1}$ at the time level t_{n+1}.

Step iv If $t_{n+1} \le T$, then let $u_{j+\frac{1}{2},k+\frac{1}{2}}^{[0]} := u_{j+\frac{1}{2},k+\frac{1}{2}}^{n+1}$ and $(x_{j,k}^{[0]}, y_{j,k}^{[0]}) := (x_{j,k}^{[\nu+1]}, y_{j,k}^{[\nu+1]})$, and go to Step ii.

4.3. Boundary redistribution. In many flow situations, discontinuities may initially exist in boundaries or move to boundaries in a later time. As a consequence, boundary points redistribution should be made in order to improve the quality of the solution near boundary. A simple redistribution strategy is proposed as follows. For convenience, our attention is restricted to the case that the physical domain Ω_p is rectangular. Assume a new set of grid points $\{\tilde{x}_{j,k}, \tilde{y}_{j,k}\}$ is obtained in Ω_p by solving the moving mesh equation (4.4). Then the speeds of internal grid point $(x_{j,k}, y_{j,k})$ are given by:

$$(c^1, c^2)_{j,k} := (\widetilde{x} - x, \widetilde{y} - y)_{j,k}, \quad \text{for} \quad 1 \le j \le J_x,\ 1 \le j \le J_y.$$

We assume that the points of the left and bottom boundaries are moving with the same speed as the tangential component of the speed for the internal points adjacent to those boundary points, namely

$$\begin{aligned} (c^1, c^2)_{0,k} &= (0, c^2_{1,k}), \qquad 1 \le j \le J_y, \\ (c^1, c^2)_{j,0} &= (c^1_{j,1}, 0), \qquad 1 \le j \le J_x. \end{aligned}$$

Thus new boundary points $(\widetilde{x}_{0,k}, \widetilde{y}_{0,k})$ and $(\widetilde{x}_{j,0}, \widetilde{y}_{j,0})$ are defined by

$$\begin{aligned} (\widetilde{x}, \widetilde{y})_{0,k} &= (x, y)_{0,k} + (c^1, c^2)_{0,k}, \qquad 1 \le k \le J_y, \\ (\widetilde{x}, \widetilde{y})_{j,0} &= (x, y)_{j,0} + (c^1, c^2)_{j,0}, \qquad 1 \le j \le J_x. \end{aligned}$$

The redistribution for other boundaries can be carried out in a similar way. Numerical experiments show that the above procedure to move the boundary points is useful in improving the solution resolution.

5. Numerical experiment for 1D problems. In this section, we first implement our adaptive mesh methods presented in last section for several one-dimensional model problems. One of the main advantages of **Algorithm 0** is that the solution algorithm (i.e. PDE solver) and the mesh re-distribution algorithm are independent of each other. Several solution schemes, such as MUSCL-type finite volume method (3.15)-(3.17), the second-order MUSCL type gas-kinetic approach [36] and second-order central scheme [25], have been employed to evolute the underlying PDEs in Step 3 of **Algorithm 0**. The results obtained by the three methods are shown in good agreement.

5.1. 1D example. Three examples will be considered in this subsection. All of them have been used by several authors to test various numerical schemes.

Example 5.1. **Burgers' equation.** This example is the inviscid Burgers' equation

$$u_t + \left(\frac{u^2}{2}\right)_x = 0, \qquad 0 \le x \le 2\pi, \tag{5.1}$$

subject to the 2π-periodic initial data

$$u(x, 0) = 0.5 + \sin(x), \quad x \in [0, 2\pi).$$

The solution propagates to the right, steepening until the critical time $t_c = 1$, at which a shock forms. Figure 5.1 shows the solutions at $t = 2$, when the shock is well developed. Also shown in

Figure 5.1 is the trajectory of the grid points up to $t = 2$, obtained with $J = 30$ and 50. The ability of the adaptive mesh method to capture and follow the moving shock is clearly demonstrated in this figure. Some details in this example are the following: the monitor function used in the computation is $\omega = \sqrt{1 + 0.2|u_\xi|^2}$; the number of the Gauss-Seidel iteration used is 5; the scheme for evoluting the Burgers' equation is a (formally) second-order MUSCL finite volume scheme (with the Lax-Friedrichs flux) together with an 2nd order Runge-Kutta discretization; the CFL number used is 0.3.

TABLE 5.1
Example 5.1: The L^1-error and convergence order at $t = 0.9$ and $t = 0.999$.

J	40	80	160	320
$t = 0.9$	4.73e-2 (–)	1.48e-2 (1.68)	3.76e-3 (1.98)	7.90e-4 (2.25)
t=0.999	5.84e-2 (–)	1.85e-2 (1.67)	5.23e-3 (1.82)	1.33e-3 (1.98)

In Table 5.1, the L^1-error and the convergence rate are listed for $t = 0.9$ and $t = 0.999$. It is observed that a second-order rate of convergence can be obtained for the adaptive mesh method.

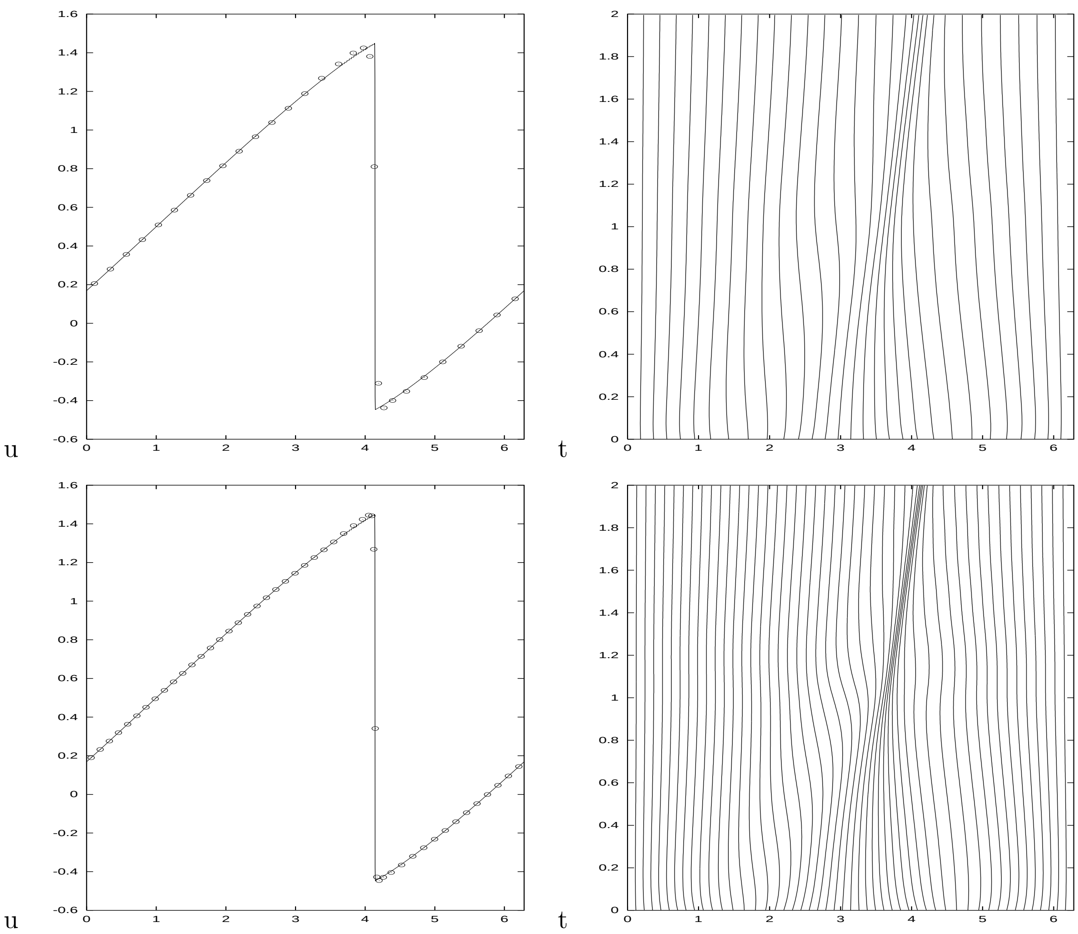

FIG. 5.1. Example 5.1: left are the numerical solution ("o") and the exact solution (solid line) at $t = 2$, and the right are the trajectory of the mesh for $0 \leq t \leq 2$. Top: $J = 30$ and bottom: $J = 50$.

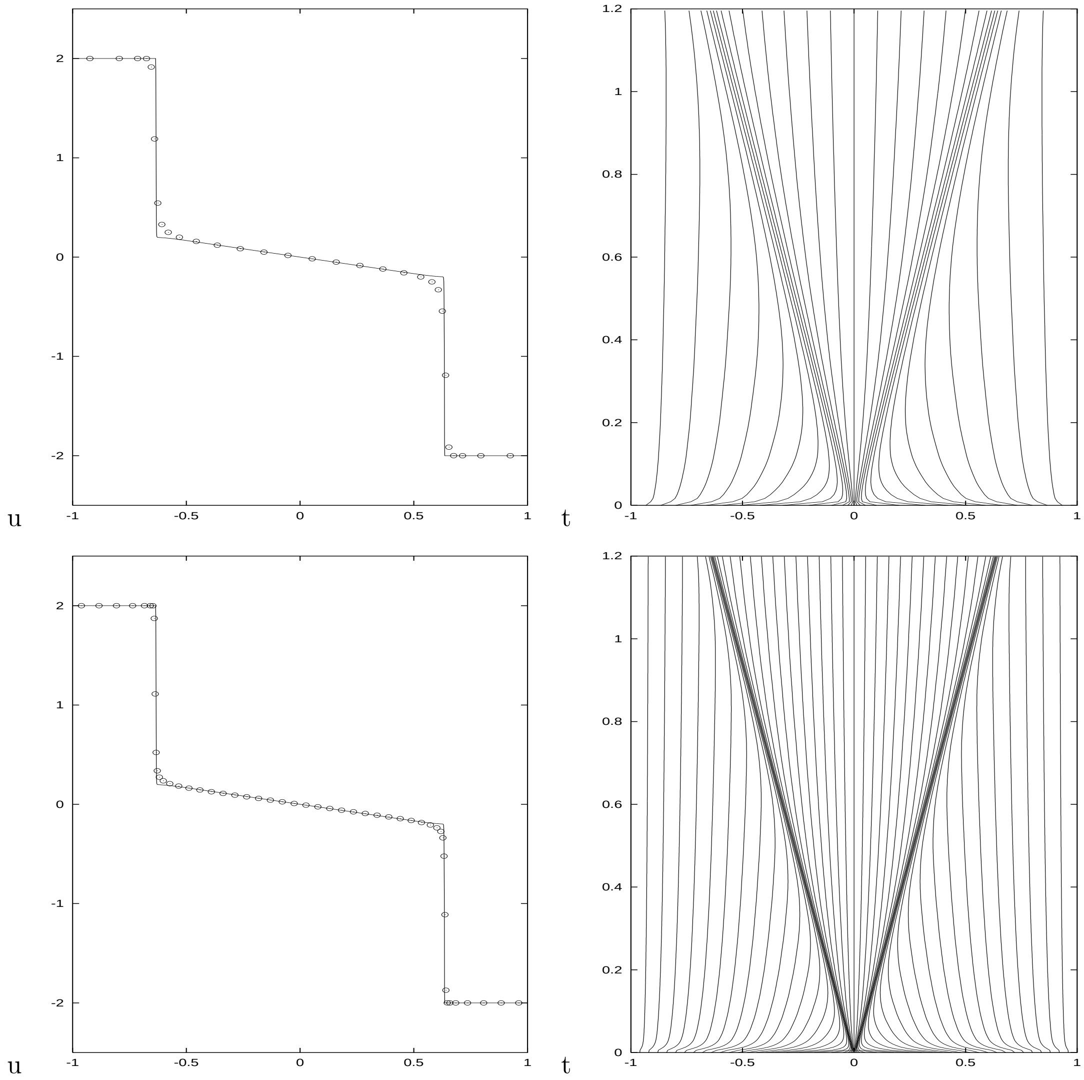

FIG. 5.2. Example 5.2: left are the numerical solution ("o") and the exact solution (solid line) at $t = 1.2$, and the right are the trajectory of the mesh for $0 \leq t \leq 1.2$. Top: $J = 30$ and bottom: $J = 50$.

EXAMPLE 5.2. **Non-convex conservation laws.** Here we apply the adaptive mesh algorithm to the Riemann problem of scalar hyperbolic conservation laws with nonlinear non-convex flux:

$$u_t + f(u)_x = 0; \qquad f(u) = \frac{1}{4}(u^2 - 1)(u^2 - 4). \tag{5.2}$$

The initial data is $u(x, 0) = -2\text{sign}(x)$. The problem was also considered in [16]. In contrast with Burgers' equation, the flux function for this problem is non-convex, which leads to difficulties with some numerical schemes and so serves as a good test problem. The numerical solution at $t = 1.2$ is shown for an adaptive mesh in Figure 5.2, with $J = 30$ and 50 respectively. Some details in this example are the following: the monitor function used in the computation is $\omega = \sqrt{1 + |u_\xi|^2}$; the number of the Gauss-Seidel iteration used is 5; the scheme for evoluting the equation (5.2) is a (formally) second-order MUSCL finite volume scheme (with the Lax-Friedrichs flux) together with an 2nd order Runge-Kutta discretization. It is seen that the numerical solution gives sharp shock profile.

$\alpha_1 = 20, \alpha_2 = 100$

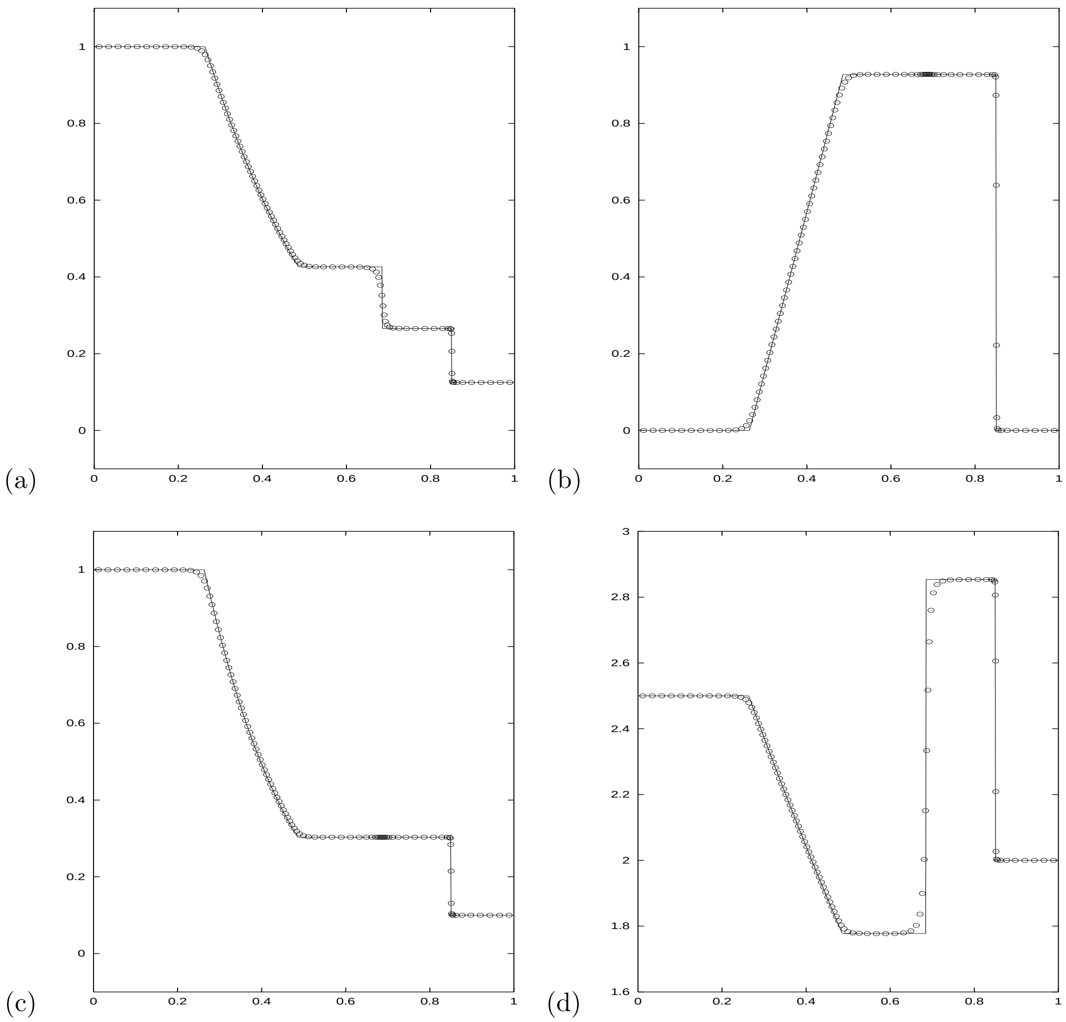

FIG. 5.3. Adaptive mesh solutions at $t = 0.2$ for Sod's shock tube problem. (a): density; (b): velocity; (c): pressure and (d): internal energy. "o" and solid lines denote numerical and exact solutions, respectively.

EXAMPLE 5.3. **Euler equations of gas dynamics.** In this example, we test our adaptive mesh algorithm to the one–dimensional Euler equations of gas dynamics

$$\left[\begin{array}{c} \rho \\ \rho u \\ E \end{array}\right]_t + \left[\begin{array}{c} \rho u \\ \rho u^2 + p \\ u(E+p) \end{array}\right]_x = 0, \tag{5.3}$$

where ρ, u, p, and E are density, velocity, pressure, and the total energy, respectively. The above system is closed by the equation of state, $p = (\gamma - 1)(E - \rho u^2/2)$. The initial data is chosen as

$$(\rho, \rho u, E) = \begin{cases} (1,\ 0,\ 2.5), & \text{if } x < 0.5, \\ (0.125,\ 0,\ 0.25), & \text{if } x > 0.5. \end{cases}$$

This is a well-known test problem proposed by Sod [30]. The monitor function employed for this

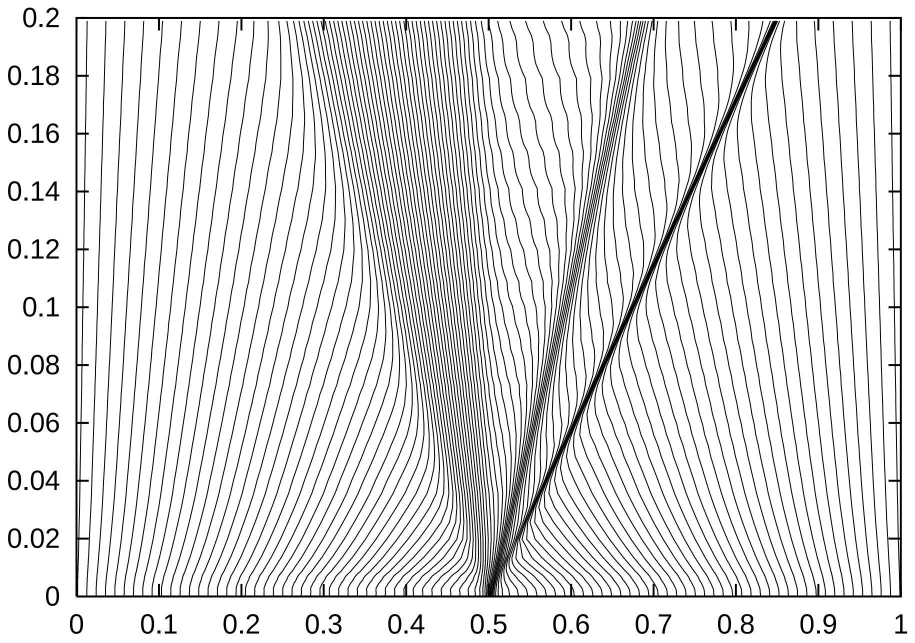

FIG. 5.4. Trajectory of the grid points for Sod's shock tube problem with $\alpha_1 = 20, \alpha_2 = 100$.

computation is $G = \omega I$ with

$$\omega = \sqrt{1 + \alpha_1 \left(\frac{u_\xi}{\max_\xi |u_\xi|}\right)^2 + \alpha_2 \left(\frac{s_\xi}{\max_\xi |s_\xi|}\right)^2} \tag{5.4}$$

where $s = p/\rho^\gamma$, and the parameters α_i $(i = 1, 2)$ are some nonnegative constants. The above monitor function was suggested by Stockie et al. [31] who also discussed several other choices for the monitor function. The numerical results are obtained with $J = 100$, $\alpha_1 = 20, \alpha_2 = 100$ and are presented in Figure 5.3. It is found that the contact and shock discontinuities are well resolved, although quite a number of grid points are also moved to the rarefaction wave region. This can also be observed from the mesh contour plotted in Figure 5.4.

6. Numerical experiment for 2D problems. In this section, we will test our adaptive mesh algorithm presented in Section 3.3 for some two-dimensional problems, including 2D Riemann problems, a double Mach reflection problem, and flow past a circular cylinder.

6.1. 2D grid generation. We begin by testing Step 2 of **Algorithm 0**, i.e. to test mesh distribution part with given functions.

EXAMPLE 6.1. **2D grid generation.** We consider grid generation in the physical domain $[-1, 1] \times [-1, 1]$ for the following functions:

$$u(x, y) = \exp(-8(4x^2 + 9y^2 - 1)^2), \tag{6.1}$$

$$u(x, y) = \exp(-100(y - x^2 + 0.5)^2), \tag{6.2}$$

$$u(x, y) = 50\exp(-2500(x^2 + y^2)), \tag{6.3}$$

$$u(x, y) = \begin{cases} 1, & \text{if } |x| \le |y|, \\ 0, & \text{otherwise} \end{cases} \tag{6.4}$$

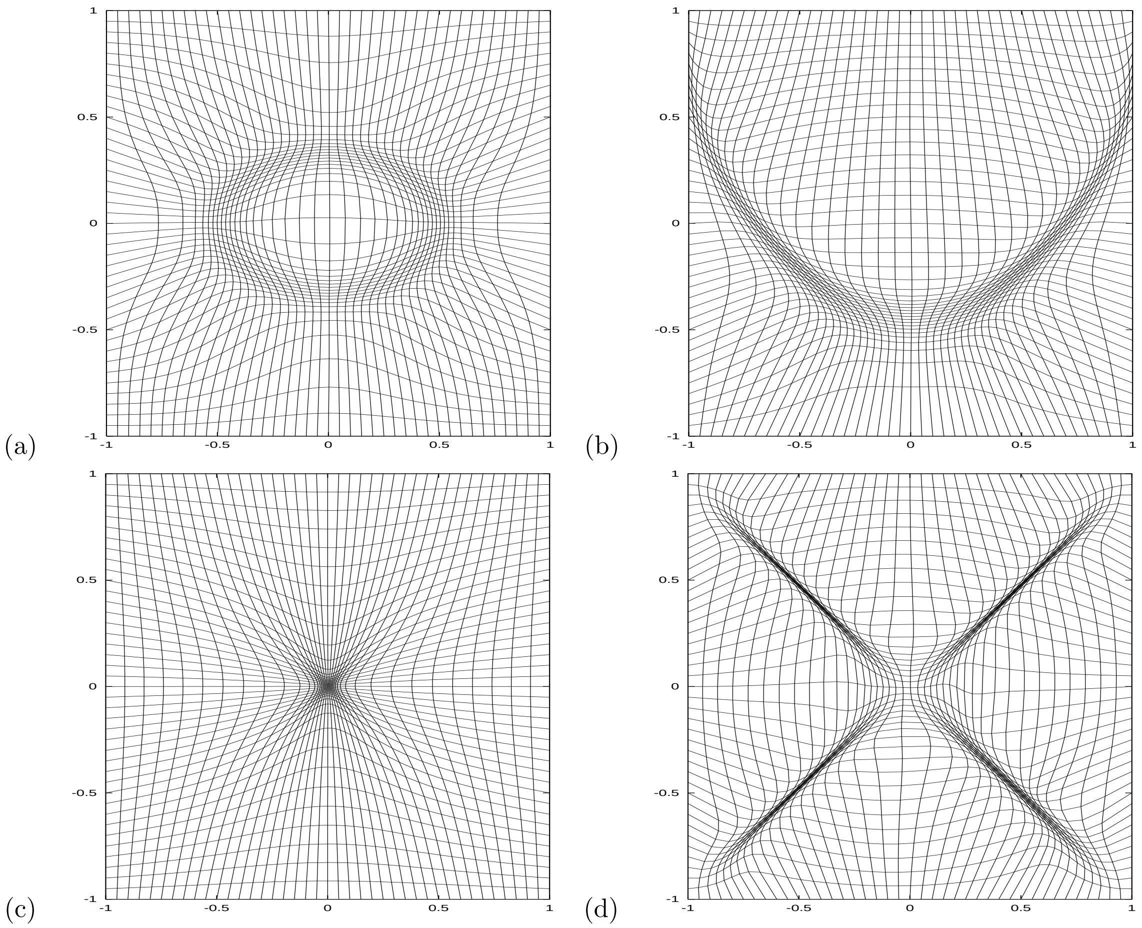

Fig. 6.1. Example 6.1: The adaptive mesh for (a) function (6.1), (b) function (6.2), (c) function (6.3), (d) function (6.4).

The monitor function is taken as $G = \omega I$ with $\omega = \sqrt{1 + \alpha u^2}$, with $\alpha = 100$. Grid generations based on above functions have been investigated by many authors, see e.g. [6, 21, 26]. Our results plotted in Figure 6.1 can be favorably compared with the published results, which indicates that Step 2 of **Algorithm 0** for 2D performs well for functions with large solution gradients or singularities.

6.2. 2D examples for Euler equations of gas dynamics. In this subsection we consider some well known test examples in 2D, including three Riemann problems and a double Mach reflection problem.

Example 6.2. **2D Riemann problem I: shock waves.** Two–dimensional Euler equations of gas dynamics can be written as

$$\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}_t + \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ u(E+p) \end{bmatrix}_x + \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}_y = 0, \tag{6.5}$$

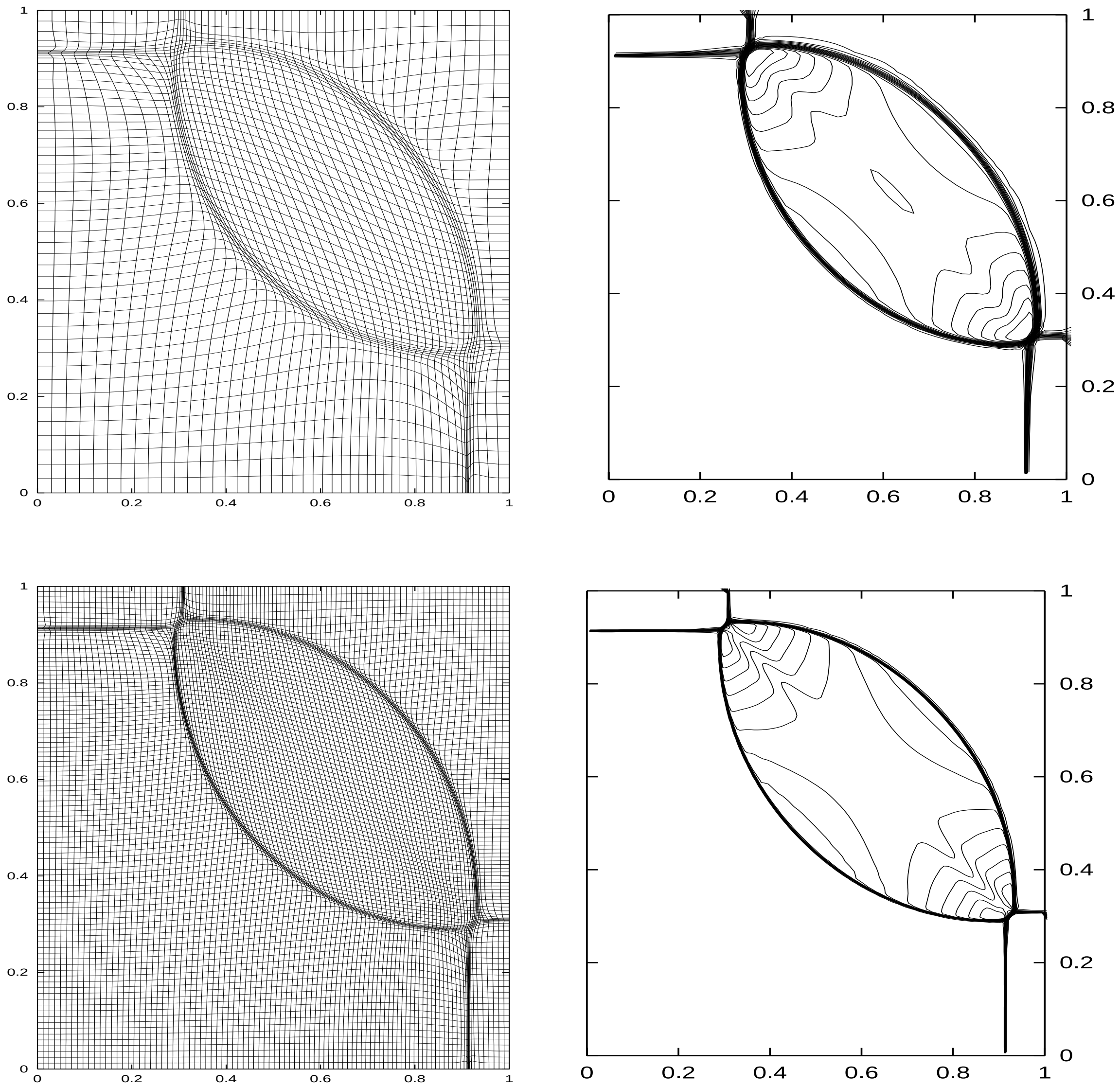

FIG. 6.2. Example 6.2: The contours of mesh (left) and the density (right). Top: $J_x = J_y = 50$ and bottom: $J_x = J_y = 100$. 30 equally spaced contour lines are used for the density.

where ρ, (u, v), p, and E are density, velocity, pressure, and total energy, respectively. For an ideal gas, the equation of state, $p = (\gamma - 1)(E - \rho(u^2 + v^2)/2)$, is provided. The initial data is chosen as

$$(\rho, u, v, p) = \begin{cases} (1.1,\ 0.0,\ 0.0,\ 1.1) & \text{if } x > 0.5, \quad y > 0.5, \\ (0.5065,\ 0.8939,\ 0.0,\ 0.35) & \text{if } x < 0.5, \quad y > 0.5, \\ (1.1,\ 0.8939,\ 0.8939,\ 1.1) & \text{if } x < 0.5, \quad y < 0.5, \\ (0.5065,\ 0.0,\ 0.8939,\ 0.35) & \text{if } x > 0.5, \quad y < 0.5, \end{cases}$$

which corresponds to the case of left forward shock, right backward shock, upper backward shock, and lower forward shock. We refer the readers to [18, 28] for details.

In [18], Lax and Liu computed 2D Riemann problems with various initial data using the positive schemes. The problem considered here corresponds to the Configuration 4 discussed in their paper.

We use our adaptive mesh algorithm with $(J_x, J_y) = (50, 50)$ and $(100, 100)$ to compute this Riemann problem and display the meshes and density at $t = 0.25$ in Figure 6.2. It is found that our results with $J_x = J_y = 100$ give sharper shock resolution than that of the positive schemes with $J_x = J_y = 400$ (see [18], p. 333). The monitor function used in this computation is $G = \omega I$ with $\omega = \sqrt{1 + 2(\rho_\xi^2 + \rho_\eta^2)}$.

EXAMPLE 6.3. **2D Riemann problem II: contact discontinuities.** We re-consider the Configurations 6 and 7 in Lax-Liu's paper [18], whose solutions contain contact discontinuities. The first configuration has the initial data

$$(\rho, u, v, p) = \begin{cases} (1,\ 0.75,\ -0.5,\ 1) & \text{if } x > 0.5, \quad y > 0.5, \\ (2,\ 0.75,\ 0.5,\ 1) & \text{if } x < 0.5, \quad y > 0.5, \\ (1,\ -0.75,\ 0.5,\ 1) & \text{if } x < 0.5, \quad y < 0.5, \\ (3,\ -0.75,\ -0.5,\ 1) & \text{if } x > 0.5, \quad y < 0.5, \end{cases}$$

and the second configuration has the initial data

$$(\rho, u, v, p) = \begin{cases} (1,\ 0.1,\ 0.1,\ 1) & \text{if } x > 0.5, \quad y > 0.5, \\ (0.5197,\ -0.6259,\ 0.1,\ 0.4) & \text{if } x < 0.5, \quad y > 0.5, \\ (0.8,\ 0.1,\ 0.1,\ 0.4) & \text{if } x < 0.5, \quad y < 0.5, \\ (0.5197,\ 0.1,\ -0.6259,\ 0.4) & \text{if } x > 0.5, \quad y < 0.5, \end{cases}$$

The adaptive mesh results for the Configuration 6 at $t = 0.3$ and for the Configuration 7 at $t = 0.25$ are displayed in Fig. 6.3. The number of grid points are $J_x = J_y = 100$. It can be observed that the adaptive mesh results with $J_x = J_y = 100$ for both cases are comparable with those obtained by using the positive schemes with $J_x = J_y = 400$ (see [18], p. 334). For this computation, the monitor function can be chosen as $G = \omega I$ with $\omega = \sqrt{1 + \alpha(\rho_\xi^2 + \rho_\eta^2)}$. It is found that in both cases if the parameter α is chosen in the range $[0.1, 1]$ then the efficiency and effectiveness of the adaptive mesh approach seem satisfactory. The results in Fig. 6.3 are obtained by using $\alpha = 0.1$ (for Configuration 6) and $\alpha = 0.9$ (for Configuration 7).

EXAMPLE 6.4. **The double-Mach reflection problem** This problem was studied extensively in Woodward and Colella [35] and later by many others. We use exactly the same setup as in [35], i.e. the same initial and boundary conditions and same solution domain $\Omega_p = [0, 4] \times [0, 1]$. Initially a right-moving Mach 10 shock is positioned at $x = \frac{1}{6}$, $y = 0$ and makes a 60° angle with the x-axis. More precisely, the initial data are

$$U = \begin{cases} U_L, & \text{for} \quad y \geq h(x, 0), \\ U_R, & \text{otherwise}, \end{cases}$$

where the state on the left, the state on the right, and the shock height are

$$\begin{aligned} &U_L = (8,\ 57.1597,\ -33.0012,\ 563.544)^T, \\ &U_R = (1.4,\ 0.0\ 0.0,\ 2.5)^T, \\ &h(x, t) = \sqrt{3}(x - 1/6) - 20t. \end{aligned}$$

As in [35], only the results in $[0, 3] \times [0, 1]$ are displayed. In Figure 6.4, the adaptive meshes with $(J_x, J_y) = (80, 20)$, $(160, 40)$ and $(320, 80)$ are displayed, while the corresponding contours of density are displayed in Figure 6.5. By comparing the density plots, it is found the adaptive computation results with $(J_x, J_y) = (320, 80)$ have similar resolution to the results obtained by the second-order discontinuous Galerkin methods with $(J_x, J_y) = (960, 240)$ (p. 214, [9]) and by the second-order central scheme with $(J_x, J_y) = (960, 240)$ (p. 67, [9]). Moreover, the adaptive results

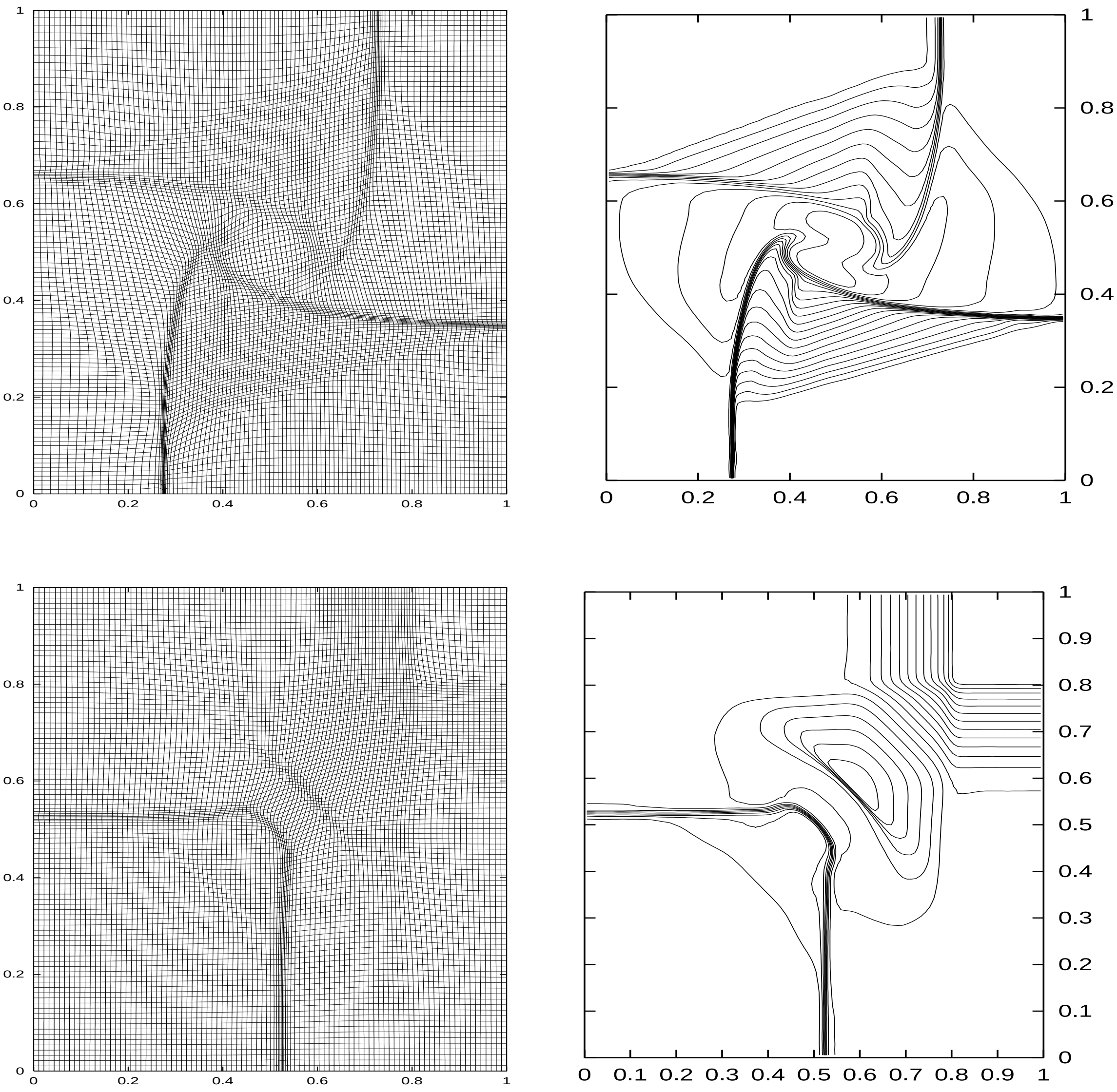

Fig. 6.3. Example 6.3: The adaptive mesh results for the Configuration 6 (top) and the Configuration 7 (bottom), as proposed in [18]. Left is the adaptive mesh and the right the contours of density, obtained with 100×100 grid points. 19 equally spaced contour lines are used for the density.

with $(J_x, J_y) = (160, 40)$ have slightly better resolution than the results of fifth order weighted ENO and the fourth order ENO with 480×119 grids (p. 406, [9]). Of course, this is not too surprising since these published results are computed by use of uniform meshes.

Figure 6.6 shows the adaptive mesh in the *whole* physical domain $[0, 4] \times [0, 1]$, with 120×30 grid points. It is observed that less grid points are distributed for $x > 3$, which is a desired feature for this test problem. Similar observations are observed for the computations with 80×20, 160×40 and 320×80 grid points. The monitor function used for this example is taken as $G = \omega I$ with $\omega = \sqrt{1 + 0.125(\rho_\xi^2 + \rho_\eta^2)}$.

6.3. Example of a non-convex physical domain. So far the numerical examples in 2D are restricted to rectangular domain. In the final example we consider a test problem whose domain is

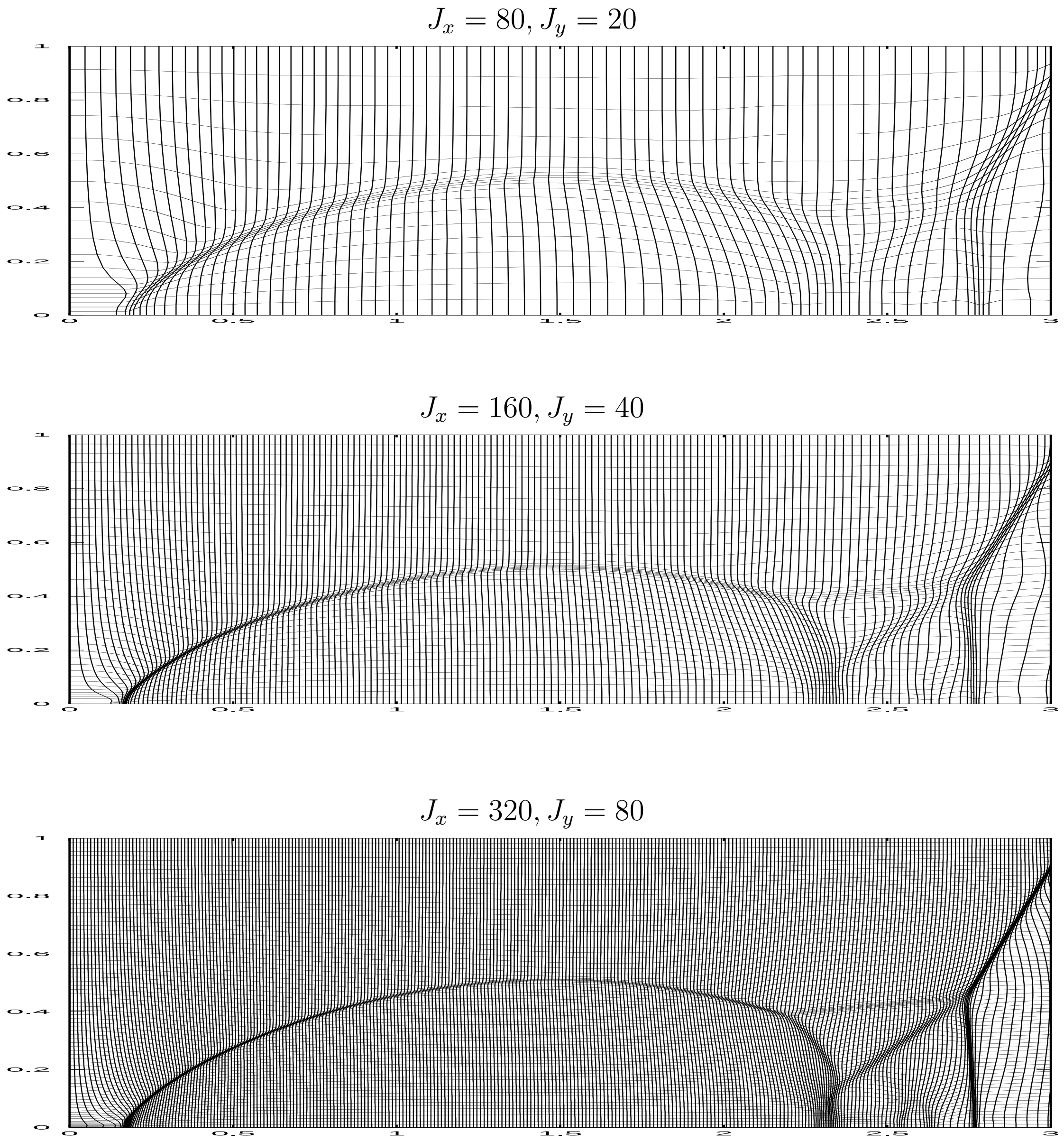

FIG. 6.4. 2D double Mach reflection at $t = 0.2$: the contours of meshes. From top to bottom: $(J_x, J_y) = (80, 20), (160, 40)$ and $(320, 80)$. i

not even convex. In this case, as long as the domain can be smoothly transformed to a rectangle, the adaptive mesh algorithm can be handily applied.

EXAMPLE 6.5. **Flow past a cylinder.** This example is concerned with the supersonic flow past a cylinder with unit radius, which is positioned at the origin on a x–y plane. The problem is initialized by a Mach 3 *free-stream* moving toward the cylinder from left. Since the physical domain Ω_p is non-convex, we first transform Ω_p to a square domain $\widehat{\Omega}_c = [0, 1] \times [0, 1]$ by using the following mapping:

$$
\begin{aligned}
x &= -\big(R_x - (R_x - 1)\widehat{x}\big) \cos\big(\theta(2\widehat{y} - 1)\big) \\
y &= \big(R_y - (R_y - 1)\widehat{x}\big) \sin\big(\theta(2\widehat{y} - 1)\big)
\end{aligned}
\tag{6.6}
$$

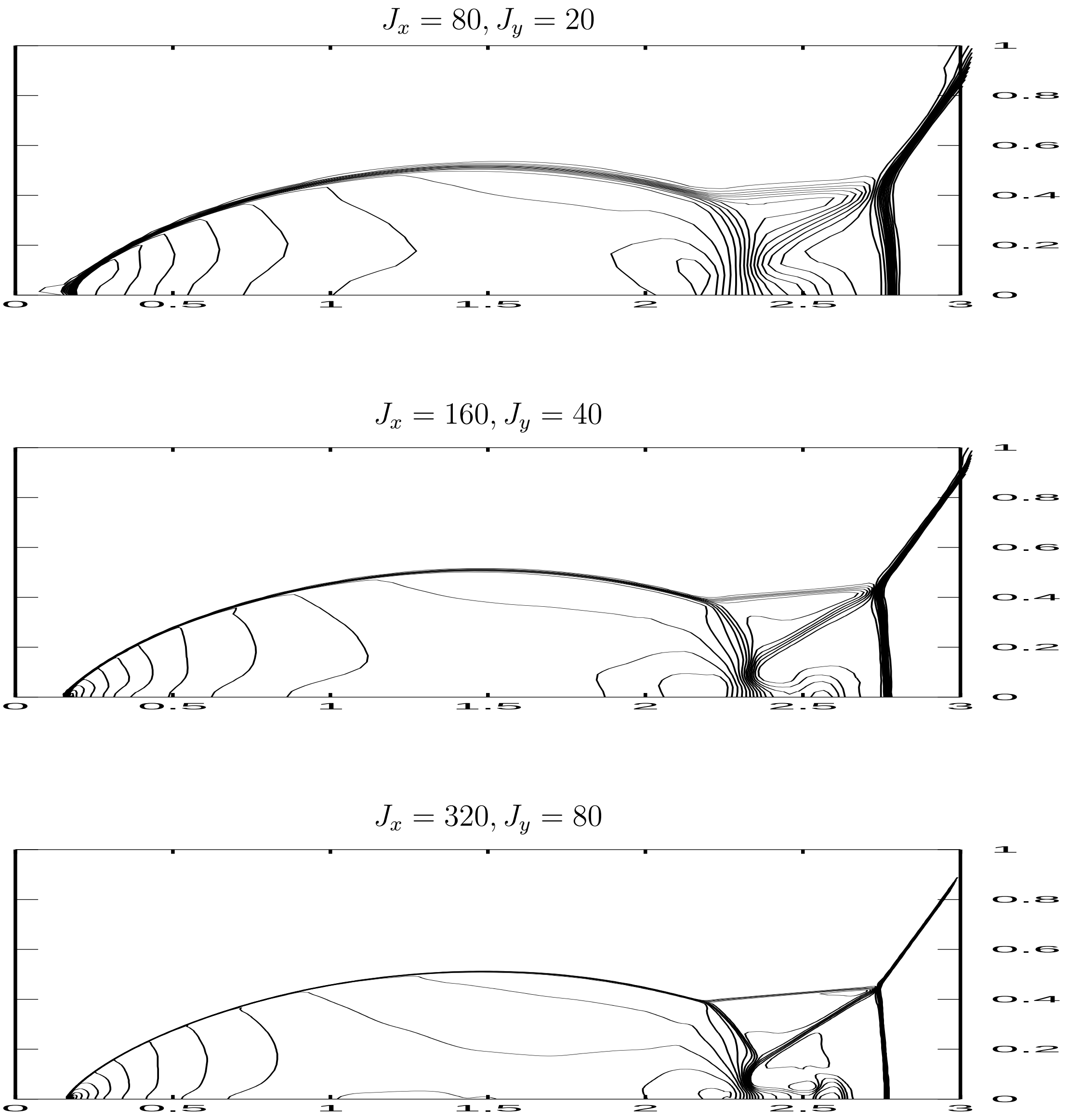

FIG. 6.5. 2D double Mach reflection at $t = 0.2$: the contours of density. From top to bottom: $(J_x, J_y) = (80, 20), (160, 40)$ and $(320, 80)$. 30 equally spaced contour lines are used.

with $R_x = 3, R_y = 5$ and $\theta = 5\pi/12$. Reflective boundary condition is imposed at the surface of the cylinder, i.e. $\hat{x} = 1$, inflow boundary condition is applied at $\hat{x} = 0$ and outflow boundary conditions are applied at $\hat{y} = 0$ and 1. We then solve the problem in $\widehat{\Omega}_c$ by using the adaptive mesh algorithm, with a logical domain Ω_c as used before. This procedure will lead to numerical solution in $\widehat{\Omega}_c$, and the mapping (6.6) finally gives the numerical approximation in the physical domain Ω_p.

We present an illustration of the mesh in the physical space and the pressure contour in Figure 6.7, by using 30×40, 60×80 and 120×160 grid points. The monitor function used for this example is taken as $G = \omega I$ with $\omega = \sqrt{1 + 0.125(\rho_\xi^2 + \rho_\eta^2)}$. As can be seen from these figures, the advantages of the adaptive mesh methods are quite obvious. The shock location computed by our adaptive mesh algorithm is approximately 0.703 (the distance between the shock curve and the surface of the

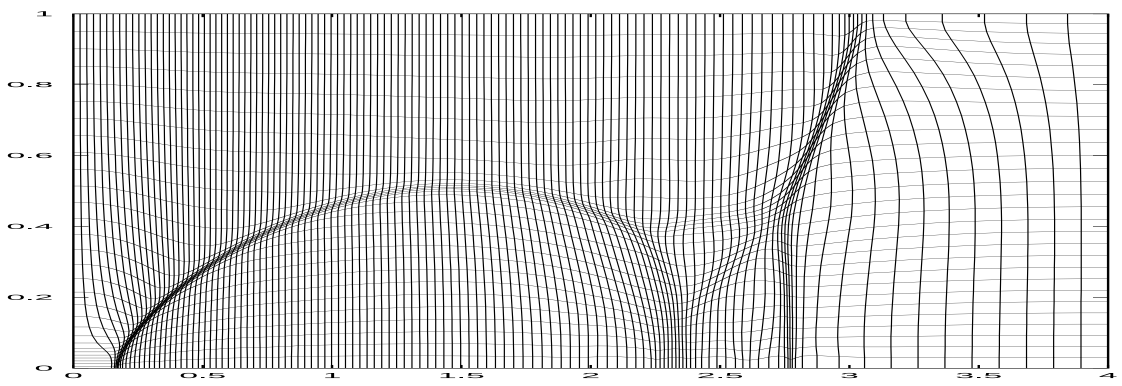

FIG. 6.6. The adaptive mesh in the whole physical domain for the double Mach reflection problem with 120×30 grid points.

cylinder), which is in good agreement with the experimental results reported in [3].

6.4. Fine details of the double Mach reflection problem. To better appreciate the effectiveness of the adaptive mesh algorithm, we show a *blow up* portion around the double Mach region in Figure 6.8. The fine details of the complicated structure in this region was previously obtained by Cockburn and Shu [8] who used high-order discontinuous Galerkin methods with 960×240 and 1920×480 grid points. In our computations, we used 640×160 and 960×240 grid points. We can see that our result with 960×240 has qualitatively the same resolution as the finer mesh results of the high-order discontinuous Galerkin computations. The corresponding mesh contours in the blow up region are shown in Figure 6.9. The smallest Δx and Δy in these runs are listed in Tables 6.1 and 6.2. It is seen that ratios between the largest and smallest mesh sizes in the adaptive grids are quite large (≥ 20), which is a desired feature of the adaptive grid methods.

TABLE 6.1
The smallest mesh sizes for the double Mach reflection problem with 640×160 *grid points.*

	$\min\{\Delta x\}$	$\max\{\Delta x\}$	$\max\{\Delta x\}/\min\{\Delta x\}$
Δx	6.5E-04	2.0E-02	30.8
Δy	4.8E-04	1.0E-02	20.8
$\sqrt{\Delta x^2 + \Delta y^2}$	8.2E-04	2.0E-02	24.4

TABLE 6.2
The smallest mesh sizes for the double Mach reflection problem with 960×240 *grid points.*

	$\min\{\Delta x\}$	$\max\{\Delta x\}$	$\max\{\Delta x\}/\min\{\Delta x\}$
Δx	4.3E-04	1.1E-02	25.6
Δy	3.1E-04	5.9E-03	19.0
$\sqrt{\Delta x^2 + \Delta y^2}$	5.3E-04	1.2E-02	22.6

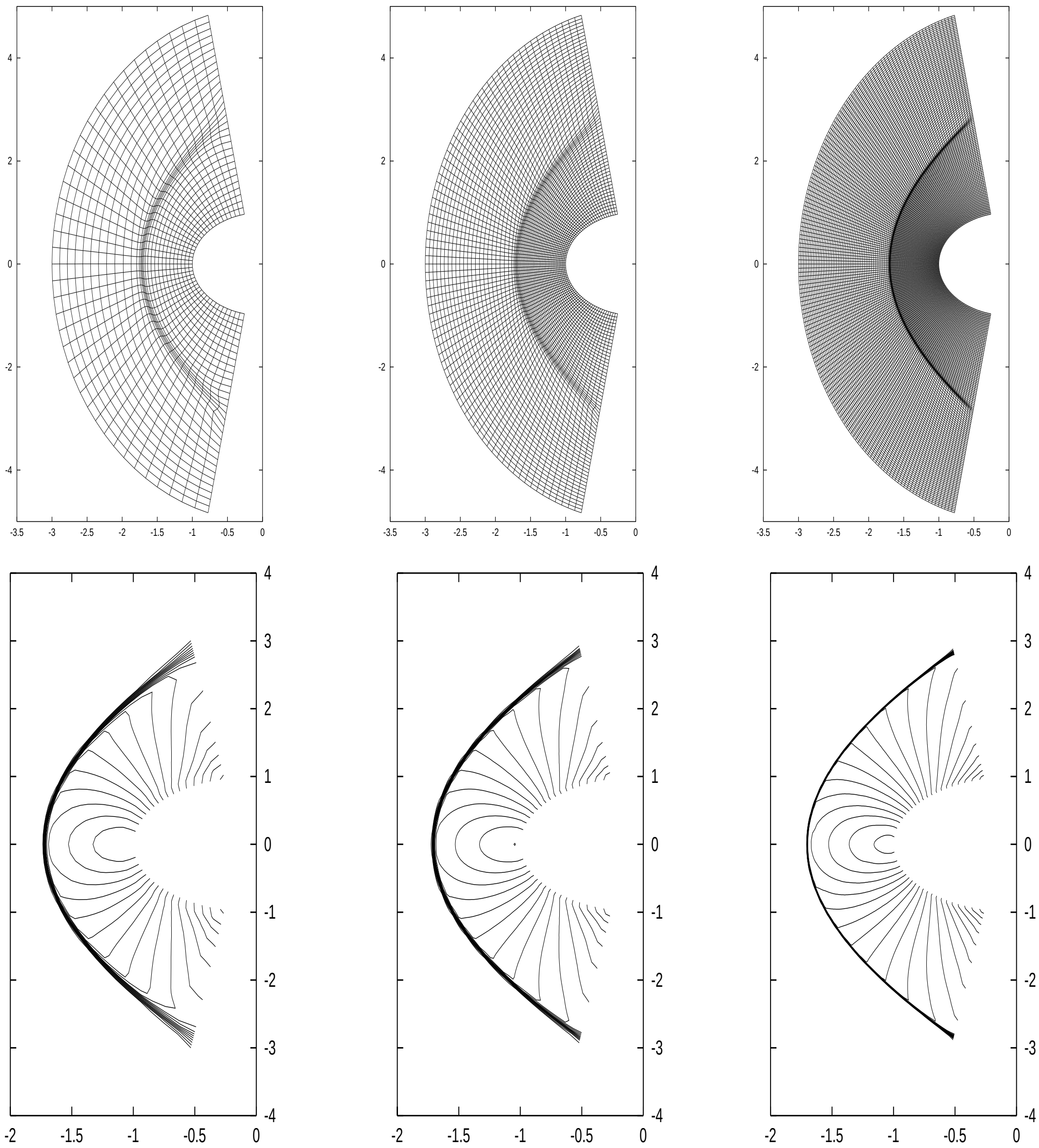

FIG. 6.7. Example 6.5: the adaptive meshes (top) and the pressure contours (bottom). From left to right: results obtained by using 30×40, 60×80 and 120×160 grids respectively. 20 equally spaced contour lines are used for the pressure.

7. Concluding remarks. In this work we proposed an adaptive mesh algorithm for solving hyperbolic system of conservation laws. Firstly, a simple mesh redistribution scheme based on the variational principle is employed. Secondly, we proposed a *conservative-interpolation* to update the approximate solutions on new meshes. This step can guarantee the overall mass-conservation which is extremely important for solving hyperbolic conservation laws. In 1D, some theoretical results on the quality of the mesh-moving and solution-updating are obtained.

The basic idea of our adaptive mesh algorithm can be summarized as the following:

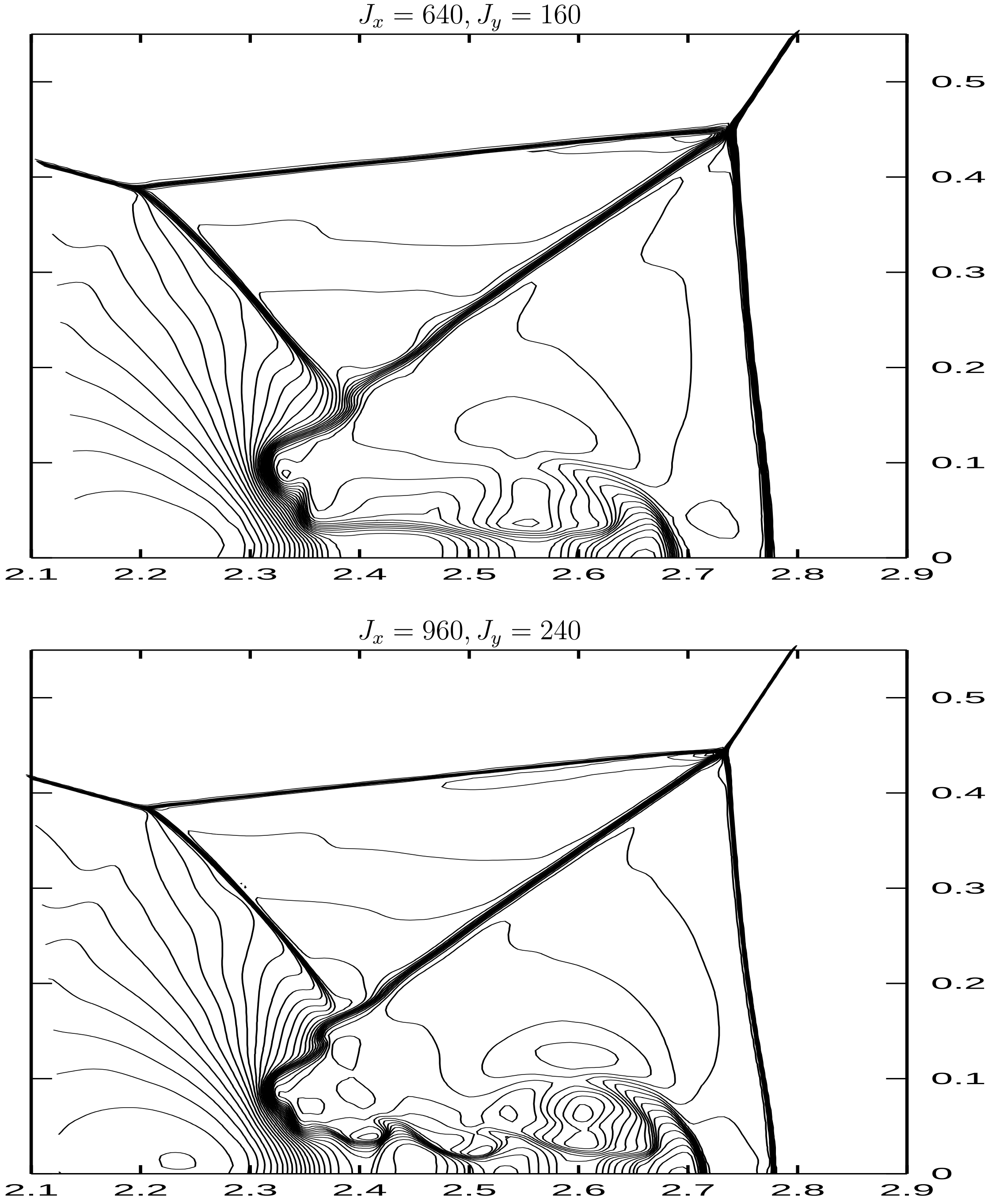

FIG. 6.8. The density contour for the double Mach reflection problem with 640×160 grid points (top) and 960×240 grid points (bottom). 45 equally spaced contour lines are used.

1. Suppose a logically rectangular spatial grid is given on which the cell centered approximation to the PDE's solution lives.
2. Update the grid by iterating an artificial time (τ) elliptic (or parabolic) grid generator. Simultaneously update the cell centered approximate solution u by iterating $\partial_\tau u = 0$ on the moving (with respect to τ) grid.
3. Update u by a physical time Δt according to a fixed-grid conservative difference scheme

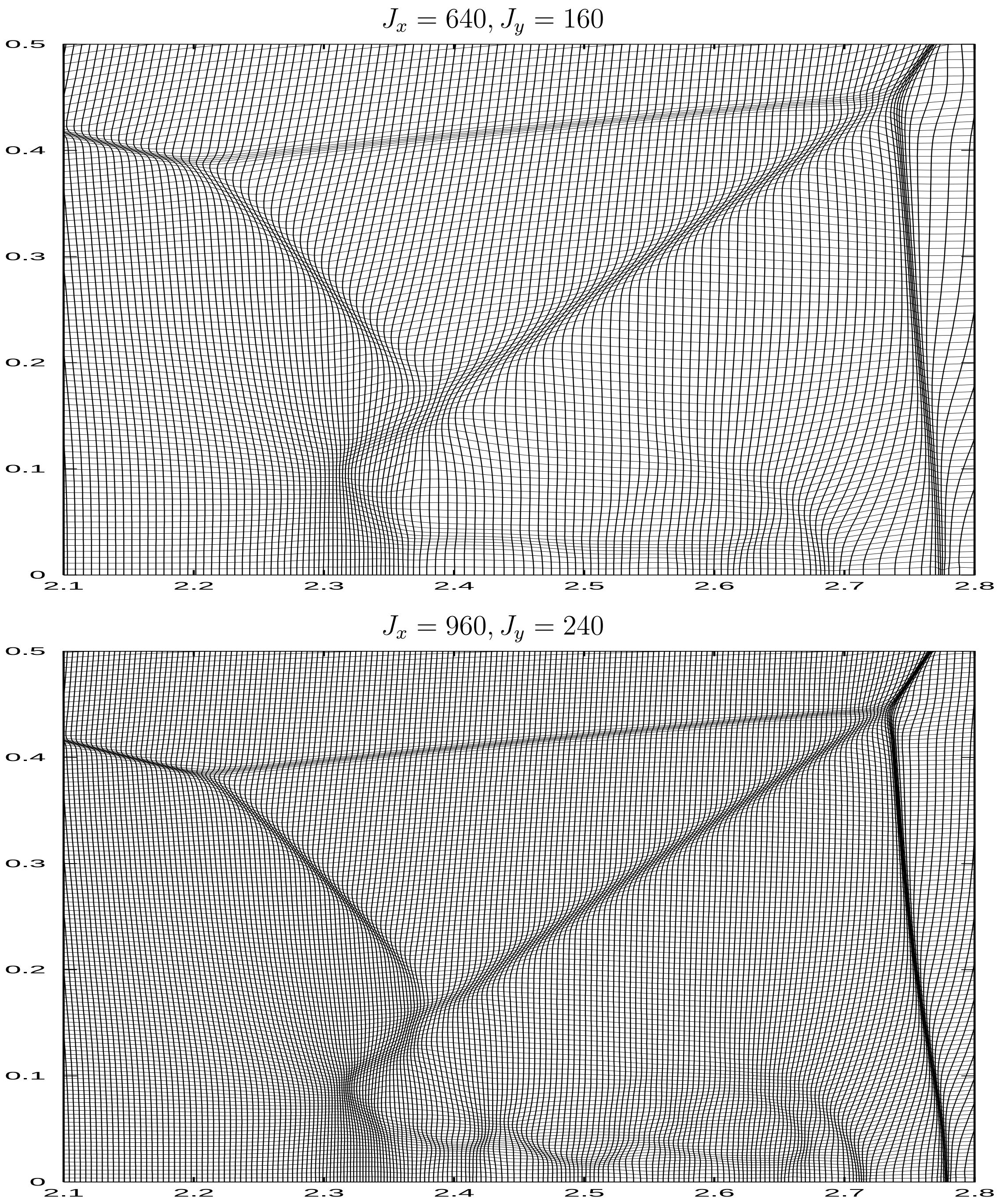

FIG. 6.9. The adaptive mesh for the double Mach reflection problem with 640×160 grid points (top) and 960×240 grid points (bottom).

which is consistent with the original conservation law.

Step 2 above should be regarded as mesh redistribution together with a conservative method to interpolate the approximate solution on the evolving grid. Therefore, the proposed algorithm is in fact a *splitting scheme*: First update the solution to $\partial_t u = 0$ on a moving grid. Second update this consistent to $\partial_t u + \nabla \cdot f(u) = 0$ on the new fixed grid.

Acknowledgment. The authors thank Professors Chi-Wang Shu and Eitan Tadmor for numerous discussions during the preparation of this work. We also thank the referees for many helpful suggestions. This research was supported by Hong Kong Baptist University and Hong Kong Research Grants Council. The research of HZT is partially supported by the National Natural Science Foundation of China (No. 19901031) and the Special Funds for Major State Basic Research Projects of China.

REFERENCES

[1] B.N. Azarenok, Variational barrier method of adaptive grid generation in hyperbolic problems of gas dynamics, Preprint, 2001.

[2] B.N. Azarenok and S. A. Ivanenko, Application of adaptive grids in numerical analysis of time-dependent problems in gas dynamics, *Comput. Maths. Math. Phys.*, 40 (2000), pp. 1330-1349.

[3] O.M. Belotserkovskii, Computation of flows around the circular cylinder with detached shock waves, *Comput. Math.* (in Russian), 3(1958), pp. 149-185.

[4] J.U. Brackbill, An adaptive grid with directional control, *J. Comput. Phys.* 108 (1993), pp. 38-50.

[5] J.U. Brackbill and J.S. Saltzman, Adaptive zoning for singular problems in two dimensions, *J. Comput. Phys.* 46 (1982), pp. 342–368.

[6] W.M. Cao, W.Z. Huang and R.D. Russell, An r-adaptive finite element method based upon moving mesh PDEs, *J. Comput. Phys.* 149 (1999), pp. 221–244.

[7] H. D. Ceniceros and T. Y. Hou, An efficient dynamically adaptive mesh for potentially singular solutions. To appear in JCP, 2001.

[8] B. Cockburn and C.-W. Shu, The Runge-Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems, *J. Comput. Phys.* 141 (1998), pp. 199-224.

[9] B. Cockburn, C. Johnson, C.-W. Shu and E. Tadmor, *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*. Lecture Notes in Math. 1697 (A. Quarteroni ed.). Springer, 1997.

[10] S. F. Davis and J. E. Flaherty, An adaptive finite element method for initial-boundary value problems for partial differential equations, *SIAM J. Sci. Stat. Comput.* 3 (1982), 6-27.

[11] A.S. Dvinsky, Adaptive grid generation from harmonic maps on Riemannian manifolds, *J. Comput. Phys.* 95 (1991), pp. 450–476.

[12] R. Fazio and R. LeVeque, Moving–mesh methods for one–dimensional hyperbolic problems using CLAWPACK. To appear in Comp. Math. Appl., 2001.

[13] A. Harten and J.M. Hyman, Self–adjusting grid methods for one–dimensional hyperbolic conservation laws, *J. Comput. Phys.* 50(1983), pp. 235–269.

[14] K. H. Karlsen, K.-A. Lie and N. H. Risebro, A front tracking method for conservation laws with boundary conditions. In *Hyperboilic problems: theory, numerics, applications* (eds. M. Fey and R. Jeltsch), Int. Series of Numer. Math., Birkhauser Verlag, 129 (1999), pp. 493-502.

[15] S. Jin and Z.P. Xin, The relaxation schemes for systems of conservation laws in arbitrary space dimensions, *Commun. Pure Appl. Math.* 48(1995), pp.235–276.

[16] A. Kurganov and E. Tadmor, New high-resolution central schemes for nonlinear conservation laws and convective–diffusion equations, *J. Comput. Phys.* 160 (2000), pp.214–282.

[17] P.D. Lax and X.D. Liu, Positive schemes for solving multi-dimensional hyperbolic systems of conservation laws. *J. Comput. Fluid Dynam.* 5 (1996), pp. 133-156.

[18] P.D. Lax and X.D. Liu, Solutions of two–dimensional Riemann problems of gas dynamics by positive schemes, *SIAM J. Sci. Comput.* 19(1998), pp.319–340.

[19] R. J. LeVeque, High-resolution finite volume methods on arbitrary grids via wave propagation, *J. Comput. Phys.* 78 (1988), pp. 36-63.

[20] S. Li and L. Petzold, Moving mesh methods with upwinding schemes for time-dependent PDEs, *J. Comput. Phys.* 131 (1997), pp. 368-377.

[21] R. Li, T. Tang, and P.W. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, *J. Comput. Phys.* 170 (2001), pp. 562-588.

[22] F. Liu, S. Ji and G. Liao, An adaptive grid method and its application to steady Euler flow calculations, *SIAM J. Sci. Comput.* 20 (1998), pp. 811-825.

[23] K. Miller and R.N. Miller, Moving finite element. I, *SIAM J. Numer Anal.* 18(1981), pp. 1019–1032.

[24] R. Natanlini, Convergence to equilibrium for the relaxation approximations of conservation laws. *Comm. Pure Allp. Math.* 49 (1996), pp. 795-823.

[25] H. Nessyahu and E. Tadmor, Non-oscillatory central differencing for hyperbolic conservation laws, *J. Comput. Phys.* 87 (1990), pp. 408-463.

[26] W.Q. Ren and X.P. Wang, An iterative grid redistribution method for singular problems in multiple dimensions,

J. Comput. Phys. 159(2000), pp.246–273.

[27] K. Saleri and S. Steinberg, Flux-corrected transport in a moving grid. *J. Comput. Phys.* 111 (1994), pp. 24-32.

[28] C.W. Schulz–Rinne, J.P. Collins, and H.M. Glaz, Numerical solution of the Riemann problem for two–dimensional gas dynamics, *SIAM J. Sci. Comput.* 14(1993), pp.1394–1414.

[29] C.-W. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes II, *J. Comput. Phys.* 83 (1989), 32-78.

[30] G. A. Sod, A survey of finite difference methods for systems of nonlinear hyperbolic conservation laws. *J. Comput. Phys.* 27 (1978), 1-31.

[31] J.M. Stockie, J.A. Mackenzie, and R.D. Russell, A moving mesh method for one–dimensional hyperbolic conservation laws, *SIAM J. Sci. Comput.*, 22 (2001), pp. 1791-1813.

[32] E. Tadmor and T. Tang, Pointwise error estimates for relaxation approximations to conservation laws. *SIAM J. Math. Anal.*, 32 (2001), pp. 870-886.

[33] H. Z. Tang and H.-M. Wu, Kinetic flux vector splitting for radiation hydrodynamical equations. *Computers & Fluids*, 29 (2000), 917-933.

[34] A. Winslow, Numerical solution of the quasi–linear Poisson equation, *J. Comput. Phys.* 1 (1967), pp.149–172.

[35] P. Woodward and P. Colella, The numerical simulation of two dimensional fluid flow with strong shocks, *J. Comput. Phys.* 54 (1984), pp. 115-173.

[36] K. Xu, *Gas-Kinetic Schemes for Unsteady Compressible Flow Simulations*, von Karman Institute for Fluid Dynamics Lecture Series 1998-03 (1998).

[37] H. Yang, A local extrapolation method for hyperbolic conservation laws I: The ENO underlying schemes, *J. Sci. Comput.* 15 (2000), pp. 231-264.