Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 188 (2003) 543–572
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

An adaptive mesh redistribution method for nonlinear Hamilton–Jacobi equations in two- and three-dimensions

H.-Z. Tang [a], Tao Tang [b,c,*], Pingwen Zhang [a]

[a] School of Mathematical Sciences, Peking University, Beijing 100871, People's Republic of China
[b] Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong
[c] Institute of Computational Mathematics, The Chinese Academy of Sciences, Beijing 100080, People's Republic of China

Received 13 February 2002; received in revised form 1 March 2003; accepted 13 March 2003

Abstract

This paper presents an adaptive mesh redistribution (AMR) method for solving the nonlinear Hamilton–Jacobi equations and level-set equations in two- and three-dimensions. Our approach includes two key ingredients: a non-conservative second-order interpolation on the updated adaptive grids, and a class of monitor functions (or indicators) suitable for the Hamilton–Jacobi problems. The proposed adaptive mesh methods transform a uniform mesh in the logical domain to cluster grid points at the regions of the physical domain where the solution or its derivative is singular or nearly singular. Moreover, the formal second-order rate of convergence is preserved for the proposed AMR methods. Extensive numerical experiments are performed to demonstrate the efficiency and robustness of the proposed adaptive mesh algorithm.

Keywords: Moving adaptive grid method; Hamilton–Jacobi equations; Level set equations; Finite difference method

1. Introduction

Hamilton–Jacobi (H–J) equations are of practical importance with applications ranging from mathematical finance and differential games to front propagation and image enhancement. For this reason, there have been many theoretical and numerical studies for the H–J equations in the past two decades. Solutions of H–J equations are continuous and, in the generic case, form discontinuous derivatives in a finite time even with smooth initial data. This also introduces great difficulties in obtaining numerical solutions of the H–J equations. Traditional high-order methods are unsuitable, because spurious oscillations will generally

* Corresponding author.
E-mail addresses: hztang@math.pku.edu.cn (H.-Z. Tang), ttang@math.hkbu.edu.hk (T. Tang), pzhang@math.pku.edu.cn (P. Zhang).

0021-9991/03/$ - see front matter
doi:10.1016/S0021-9991(03)00192-X

occur in the presence of the discontinuous derivatives. Hence, some modern numerical techniques such as ENO schemes and central schemes have been proposed and studied [14,17,18,26,27].

Consider the Hamilton–Jacobi (H–J) equation

$$\phi_t + H(\mathbf{x}, t, \phi_{x_1}, \ldots, \phi_{x_d}) = 0, \tag{1.1}$$

where $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbf{R}^d$, $t > 0$. The definition of the viscosity solutions and the question of well-posedness were formulated and systematically studied by Crandall, Evans, Lions and many others, see e.g. [6,8,9]. In [9], Crandall and Lions studied the convergence of monotone finite difference schemes to the viscosity solutions of (1.1). Unfortunately, monotone schemes are at most first-order accurate, measured by local truncation errors in smooth regions of the solution. Some rigorous analysis on convergence rate for hyperbolic conservation laws and Hamilton–Jacobi equations can be found in [15,19,28,32,34]. Typically, there is a close relation between the H–J equations and hyperbolic conservation laws, and as a result concepts used for the conservation laws can be passed to the H–J equations. High resolution methods for solving the conservation laws and the H–J equations include high-order essentially nonoscillatory (ENO) schemes introduced by Osher and Sethian [26] and Osher and Shu [27]; weighted ENO schemes developed by Liu et al. [21], Jiang and Shu [14], and Jiang and Peng [13]; and the central high resolution schemes proposed by Kurganov and Tadmor [18] and Lin and Tadmor [24,25]. Jin and Xin [16] investigated the numerical passage of relaxation approximation for conservation laws to the H–J equations.

In the traditional way, numerical methods are applied with fixed, pre-assigned grids. In this case, higher-order (3rd, 4th or even higher) numerical schemes have to be developed in order to enhance the resolution of the numerical approximations. For unstructured meshes, high-order algorithms for H–J equations are relatively few. Abgrall [1] extended the monotone-type finite volume schemes to first-order H–J equations on triangular meshes. High-order approximation on triangular meshes is developed by Augoula and Abgrall [2]. In a recent work of Zhang and Shu [37], high-order WENO schemes for solving the nonlinear H–J equations on two-dimensional unstructured meshes are developed.

It is expected that lower-order scheme (which may be rather simple) can also produce high resolution with small number of grid points *if grid adaptation is employed.* To fulfill this purpose, it is desirable to develop efficient adaptive mesh redistribution (AMR) algorithms to solve the nonlinear H–J equation (1.1). It is a challenging problem to generate an effective moving mesh in two or more dimensions, especially when the underlying solution develops complicated structures and becomes singular or nearly singular. Several AMR techniques have been introduced based on solving elliptic PDEs first proposed by Winslow [36]. Winslow's formulation requires the solution of a nonlinear, Poisson-like equation to generate a mapping from a regular domain in a parameter space to an irregularly shaped domain in physical space. By connecting points in the physical space corresponding to discrete points in the parameter space, the physical domain can be covered with a mesh suitable for the solution of finite difference/element equations. 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 or its derivative has the largest gradients. Brackbill and Saltzman [5] generalize Winslow's method to produce satisfactory mesh concentration while maintaining relatively good smoothness and orthogonality. Their approach has become one of the most popular methods used for mesh generation and adaptation.

There are several difficulties in designing AMR schemes for solving the H–J equations. We list two major ones below. The first one is concerned with the monitor function, which in general is an indicator of the degree of singularity. In solving hyperbolic system of conservation laws with an AMR method, it is known that the gradient-based monitor can be employed, see e.g. [3,22,30]. Since there is a close relation between the hyperbolic system of conservation laws and the H–J equations, it is expected that the relation can be also incorporated into the monitor functions for the AMR methods to the H–J problems. The second issue is about interpolating numerical approximations on the updated adaptive grids. In solving the hyperbolic

system of conservation laws, a conservative-type interpolation seems crucial, due to the corresponding properties of the conservation laws. In the case of the H–J equations, this property seems only necessary for the derivative ϕ_x (or $\nabla_x\phi$), but not for the solution ϕ itself. The interpolation step and the PDE time-evolution should be well connected so that the (formal) higher-rate of convergence can be preserved.

Our first step is to solve the H–J equations in one- and two-space dimensions. After this is being done, the extension to 3D will be considered. The extension of an AMR method to 3D computations is always non-trivial. To our knowledge, successful 3D AMR algorithms are very few. In a recent work [23], we were able to carry out some 3D simulations for some simple nonlinear problems (one scalar and one system). However, much efforts have to be made in order to solve 3D hyperbolic problems or H–J equations with an adaptive mesh strategy.

The organization of this paper is as follows. The moving mesh algorithm is described in Section 2. Sections 3–5 contain numerical experiments in 1D, 2D and 3D, respectively. The numerical experiments will demonstrate the efficiency and robustness of the proposed AMR algorithms. Concluding remarks are given in Section 6.

2. Numerical scheme

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

Algorithm 1 (*Basic AMR procedure for H–J equations*).

1. Suppose a logically rectangular spatial grid is given on which the approximation to the H–J solutions on cell-vertex lives.
2. Update the grid by iterating an elliptic grid generator. Simultaneously update the approximate solution ϕ on the new grid by using a non-conservative second-order interpolation formula.
3. Update ϕ with a physical time Δt by solving the given PDE in the computational domain.

There are two main differences between the approach in this work and the one in [33] where adaptive solutions for the hyperbolic conservation laws are sought. The first difference is about Step 2 above. This step should be regarded as mesh redistribution together with a solution redistribution by interpolating the approximate solution on the evolving grid. In [33], the interpolation is a conservative one, but for the H–J equations we will solve a discrete H–J type equation to realize the solution redistribution. The second difference is about Step 3 above. In [33], this step is done on the new physical grid, but for the H–J equations we will do it in the fixed logical domain. The approach in [33] can clearly preserve the conservative properties of the hyperbolic conservation laws (and the Lax–Wendroff weak-solution result holds for the underlying moving mesh solution). However, there is no direct conservation properties involved for the H–J equations (1.1), and hence accuracy of the numerical solutions is the major issue under consideration. This is similar to 2D incompressible Navier–Stokes problems. If one employs the Navier–Stokes equations in terms of the primitive variables, then the solution accuracy and the satisfaction of the divergence-free restrictions are both important in designing a numerical scheme. However, if the stream-function is introduced, then the solution accuracy (and also efficiency) becomes the most important issue.

2.1. A simple PDE solver

In this subsection, we will describe Step 3 above, namely the numerical discretization for the H–J equation (1.1) in the logical domain. The mesh motion and solution interpolation parts will be provided in the next subsection.

2.1.1. 1D case

Consider the following 1D Hamilton–Jacobi equation

$$\phi_t + H(\phi_x) = 0, \quad a < x < b. \tag{2.1}$$

Assume that there exists a one-to-one coordinate transformation $x = x(\xi)$ from the logical domain $\Omega_c = [0,1]$ to the physical domain $\Omega_p = [a,b]$. We also assume that the logical domain Ω_c is covered by a uniform mesh $\xi_j = j\Delta\xi$, $0 \leqslant j \leqslant N+1$, where $\Delta\xi = 1/(N+1)$. With these assumptions, the 1D H–J equation (2.1) becomes

$$\phi_t + \widetilde{H}(\phi_\xi) = 0, \quad 0 < \xi < 1, \tag{2.2}$$

where $\widetilde{H}(\phi_\xi) \equiv H(\xi_x \phi_\xi)$. On the computational domain Ω_c, the above equation can be discretized by the following numerical scheme:

$$\phi_j^{n+1} = \phi_j^n - \Delta t_n \widetilde{H}\left(\frac{1}{2}(u_j^+ + u_j^-)\right) + \frac{\Delta t_n}{2}\mathscr{A}(u_j^+, u_j^-)\cdot(u_j^+ - u_j^-), \tag{2.3}$$

where

$$u_j^\pm := \Delta_\xi^\pm \phi_j^n = \pm(\phi_{j\pm1}^n - \phi_j^n)/\Delta\xi,$$
$$\mathscr{A}(u^+, u^-) = \max_{u \in I(u^-, u^+)} \{|\widetilde{H}_u|\}.$$

Here $I(u^-, u^+) = [\min\{u^-, u^+\}, \max\{u^-, u^+\}]$, and $\widetilde{H}_u$ stands for the derivative of $\widetilde{H}(u)$ with respect to u. The above scheme is also regarded as the local Lax–Friedrichs (LxF) scheme, which is formally of first-order accuracy in time and space. To improve its accuracy, the TVD-type high-order Runge–Kutta method of Shu and Osher [31] will be employed to replace the explicit Euler part in (2.3), and the initial value reconstruction technique of van Leer [35] will be used to approximate the initial function. For example, let $u_{j+\frac{1}{2}} = (\phi_{j+1} - \phi_j)/\Delta\xi$, which can be regarded as the cell average of ϕ_ξ. Using this notation, a piecewise linear function can be constructed

$$\overline{u}_{j+\frac{1}{2}}(\xi) = u_{j+\frac{1}{2}} + S_{j+\frac{1}{2}}(\xi - \xi_{j+\frac{1}{2}}), \quad \xi \in (\xi_j, \xi_{j+1}], \tag{2.4}$$

where $S_{j+\frac{1}{2}}$ is an approximate slope of u with respect to ξ, and is defined according to van Leer's limiter [35]. Similar order-enhancing strategy can be also used in multi-dimensional computations.

Remark. When $x_{j+1} - x_j \equiv$ const., the above 1D scheme is identical to the conventional local LxF scheme for the Hamilton–Jacobi equation [31].

Lemma 2.1. *The numerical scheme* (2.3) *is conservative in terms of* ϕ_x *and that the related numerical Hamiltonian is monotone.*

Proof. It follows easily from (2.3) that

$$\phi_j^{n+1} = \phi_j^n - \Delta t_n \bar{H}_j^n, \tag{2.5}$$

where

$$\bar{H}_j := \widetilde{H}\left(\frac{1}{2}(u_j^+ + u_j^-)\right) + \frac{1}{2}\mathscr{A}(u_j^+, u_j^-)\cdot(u_j^+ - u_j^-).$$

Let $v_{j+\frac{1}{2}}$ be the cell average of ϕ_x

$$v_{j+\frac{1}{2}} := \frac{1}{x_{j+1}-x_j}\int_{x_j}^{x_{j+1}} \phi_x \,\mathrm{d}x = \frac{\phi_{j+1}-\phi_j}{x_{j+1}-x_j}. \tag{2.6}$$

It follows from (2.5) that

$$v_{j+\frac{1}{2}}^{n+1} = v_{j+\frac{1}{2}}^{n} - \frac{\Delta t_n}{\Delta x_{j+\frac{1}{2}}^{n}}\left(\bar{H}_{j+1}^{n} - \bar{H}_{j}^{n}\right), \tag{2.7}$$

where $\Delta x_{j+\frac{1}{2}} = x_{j+1} - x_j$. The numerical flux above can be rewritten in the following form (in terms of $v_{j+\frac{1}{2}}$):

$$\bar{H}_j^n = H\left(\frac{\Delta x_{j+\frac{1}{2}}^n v_{j+\frac{1}{2}}^n}{\Delta x_{j+\frac{1}{2}}^n + \Delta x_{j-\frac{1}{2}}^n} + \frac{\Delta x_{j-\frac{1}{2}}^n v_{j-\frac{1}{2}}^n}{\Delta x_{j+\frac{1}{2}}^n + \Delta x_{j-\frac{1}{2}}^n}\right) + \frac{1}{2}\mathscr{A}\left(\Delta x_{j+\frac{1}{2}}^n v_{j+\frac{1}{2}}^n, \Delta x_{j-\frac{1}{2}}^n v_{j-\frac{1}{2}}^n\right)\cdot\left(\Delta x_{j+\frac{1}{2}}^n v_{j+\frac{1}{2}}^n - \Delta x_{j-\frac{1}{2}}^n v_{j-\frac{1}{2}}^n\right), \tag{2.8}$$

where H is the original Hamiltonian given in (2.1). It is seen that the difference equation (2.7) is a conservative approximation to the conservation laws $v_t + H(v)_x = 0$. Moreover, it follows from (2.8) that the numerical Hamiltonian $\bar{H}$ is monotone. □

2.1.2. 2D case

Consider the 2D H–J equation:

$$\phi_t + H(\phi_x, \phi_y) = 0, \quad (x,y) \in \Omega_{\mathrm{p}} \subseteq \mathbf{R}^2. \tag{2.9}$$

Again assume that there exists a one-to-one coordinate transformation $x = x(\xi,\eta), y = y(\xi,\eta)$ from the logical domain $\Omega_{\mathrm{c}} = [0,1]^2$ to the physical domain Ω_{p}. Moreover, the logical domain is covered by a fixed uniform mesh given by $\xi_i = i\Delta\xi, \eta_j = j\Delta\eta$, $0 \leqslant i \leqslant N_\xi + 1, 0 \leqslant j \leqslant N_\eta + 1$, and $\Delta\xi = 1/(N_\xi + 1), \Delta\eta = 1/(N_\eta + 1)$. After using the coordinate transformation, the 2D H–J equation becomes

$$\phi_t + \widetilde{H}(\phi_\xi, \phi_\eta) = 0, \quad 0 < \xi, \eta < 1, \tag{2.10}$$

where $\widetilde{H}(\phi_\xi, \phi_\eta) \equiv H(\xi_x\phi_\xi + \eta_x\phi_\eta, \xi_y\phi_\xi + \eta_y\phi_\eta)$. The two-dimensional version of the local Lax–Friedrichs (LxF) scheme for the above H–J equation on the logical domain Ω_{c} can be written as

$$\begin{aligned}\phi_{i,j}^{n+1} = \phi_{i,j}^n - \Delta t_n \widetilde{H}\left(\frac{1}{2}(u_{i,j}^+ + u_{i,j}^-), \frac{1}{2}(v_{i,j}^+ + v_{i,j}^-)\right) + \frac{\Delta t_n}{2}\mathscr{A}(u_{i,j}^\pm; v_{i,j}^\pm)\cdot(u_{i,j}^+ - u_{i,j}^-) + \frac{\Delta t_n}{2}\mathscr{B}(u_{i,j}^\pm; v_{i,j}^\pm) \\ \cdot (v_{i,j}^+ - v_{i,j}^-)\end{aligned} \tag{2.11}$$

where

$$\begin{aligned} &u_{i,j}^\pm := \Delta_\xi^\pm \phi_{i,j}^n, \quad v_{i,j}^\pm := \Delta_\eta^\pm \phi_{i,j}^n, \\ &\mathscr{A}(u^\pm; v^\pm) = \max\{|\widetilde{H}_1(u,v)|\,|\,u \in I(u^-,u^+), v \in [C,D]\}, \\ &\mathscr{B}(u^\pm; v^\pm) = \max\{|\widetilde{H}_2(u,v)|\,|\,v \in I(v^-,v^+), u \in [A,B]\}. \end{aligned}$$

In the above, $\widetilde{H}_1 = \xi_x H_1 + \xi_y H_2, \widetilde{H}_2 = \eta_x H_1 + \eta_y H_2$, where H_1 and H_2 are the partial derivatives of H with respect to ϕ_x and ϕ_y, respectively, or the Lipschitz constants of H with respect to ϕ_x and ϕ_y, if H is not differentiable. $[A,B]$ is the value range for $(\phi_\xi)_i$, and $[C,D]$ is the value range for $(\phi_\eta)_j$, over $0 \leqslant i \leqslant N_\xi$ and $0 \leqslant j \leqslant N_\eta$.

2.1.3. 3D case

Consider the 3D H–J equation

$$\phi_t + H(\phi_x, \phi_y, \phi_z) = 0, \quad (x, y, z) \in \Omega_{\mathrm{p}} \subseteq \mathbf{R}^3. \tag{2.12}$$

After using a coordinate transformation $x = x(\xi, \eta, \zeta), y = y(\xi, \eta, \zeta), z = z(\xi, \eta, \zeta)$ from the logical domain $\Omega_{\mathrm{c}} = [0, 1]^3$ to the physical domain Ω_{p}, we have

$$\phi_t + \widetilde{H}(\phi_\xi, \phi_\eta, \phi_\zeta) = 0, \quad (\xi, \eta, \zeta) \in \Omega_{\mathrm{c}}, \tag{2.13}$$

where $\widetilde{H}(\phi_\xi, \phi_\eta, \phi_\zeta) \equiv H(\xi_x\phi_\xi + \eta_x\phi_\eta + \zeta_x\phi_\zeta, \xi_y\phi_\xi + \eta_y\phi_\eta + \zeta_y\phi_\zeta, \xi_z\phi_\xi + \eta_z\phi_\eta + \zeta_z\phi_\zeta)$. The 3D version of the local LxF scheme for the H–J equation (2.13) is of the form

$$\begin{aligned}\phi^{n+1}_{i,j,k} = \phi^n_{i,j,k} - \Delta t_n \widetilde{H}\left(\frac{1}{2}(u^+_{i,j,k} + u^-_{i,j,k}), \frac{1}{2}(v^+_{i,j,k} + v^-_{i,j,k}), \frac{1}{2}(w^+_{i,j,k} + w^-_{i,j,k})\right) + \frac{\Delta t_n}{2}\mathscr{A}(u^\pm_{i,j,k}; v^\pm_{i,j,k}; w^\pm_{i,j,k}) \\ \cdot (u^+_{i,j,k} - u^-_{i,j,k}) + \frac{\Delta t_n}{2}\mathscr{B}(u^\pm_{i,j,k}; v^\pm_{i,j,k}; w^\pm_{i,j,k}) \cdot (v^+_{i,j,k} - v^-_{i,j,k}) + \frac{\Delta t_n}{2}\mathscr{C}(u^\pm_{i,j,k}; v^\pm_{i,j,k}; w^\pm_{i,j,k}) \cdot (w^+_{i,j,k} - w^-_{i,j,k}),\end{aligned} \tag{2.14}$$

where

$$\begin{aligned}&u^\pm_{i,j,k} := \Delta^\pm_\xi \phi^n_{i,j,k}, \quad v^\pm_{i,j,k} := \Delta^\pm_\eta \phi^n_{i,j,k}, \quad w^\pm_{i,j,k} := \Delta^\pm_\zeta \phi^n_{i,j,k}, \\ &\mathscr{A}(u^\pm; v^\pm; w^\pm) = \max\{|\widetilde{H}_1(u, v, w)|\, u \in I(u^-, u^+), v \in [C, D], w \in [E, F]\}, \\ &\mathscr{B}(u^\pm; v^\pm; w^\pm) = \max\{|\widetilde{H}_2(u, v, w)|\, v \in I(v^-, v^+), u \in [A, B], w \in [E, F]\}, \\ &\mathscr{C}(u^\pm; v^\pm; w^\pm) = \max\{|\widetilde{H}_3(u, v, w)|\, w \in I(w^-, w^+), u \in [A, B], v \in [C, D]\}.\end{aligned}$$

In the 3D case, $\widetilde{H}_1 = \nabla\xi \cdot \mathbf{H}, \widetilde{H}_2 = \nabla\eta \cdot \mathbf{H}$ and $\widetilde{H}_3 = \nabla\zeta \cdot \mathbf{H}$, where $\nabla = (\partial_x, \partial_y, \partial_z)^{\mathrm{T}}$, $\mathbf{H} = (H_1, H_2, H_3)^{\mathrm{T}}$. Here $H_1, H_2, [A, B]$ and $[C, D]$ are similar to that in 2D, and H_3 stands for the partial derivative for H with respect to ϕ_z, and $[E, F]$ is the value range for $(\phi_\zeta)_k$.

2.1.4. Time-discretization

By letting the time steps go to zero in (2.3), (2.11) or (2.14), we obtain a semi-discretized numerical scheme which leads to a system of ODEs. The ODE system can be solved by using the Runge–Kutta scheme of Shu and Osher [31]. More precisely, we solve

$$\Phi'(t) = L(\Phi), \tag{2.15}$$

with a third-order TVD (total variation non-increasing) Runge–Kutta scheme:

$$\Phi^{(1)} = \Phi^n + \Delta t L(\Phi^n),$$

$$\Phi^{(2)} = \frac{3}{4}\Phi^n + \frac{1}{4}\left(\Phi^{(1)} + \Delta t L(\Phi^{(1)})\right),$$

$$\Phi^{n+1} = \frac{1}{3}\Phi^n + \frac{2}{3}\left(\Phi^{(2)} + \Delta t L(\Phi^{(2)})\right).$$

2.2. Mesh motion

In this subsection, we describe Step 2 in Algorithm 1. In Tang and Tang [33], an adaptive mesh redistribution method based on the values of the cell average is proposed, resulting in a conservative interpolation scheme. Since there is no explicit conservation requirements for ϕ, we will design a new interpolation strategy which is in general non-conservative for ϕ, but conservative with respect to its gradient. Moreover, this interpolation formula will be of second-order accuracy and is governed by a discrete Hamilton–Jacobi equation which can be handled in the same way as described in the last subsection.

2.2.1. Mesh motion based on Gauss–Seidal iteration

For simplicity, we mainly illustrate the idea of the grid motion for 1D case. The extension to higher dimensions is straightforward, see [33]. Let x and ξ be 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

$$\begin{aligned} &x = x(\xi), \quad \xi \in [0,1], \\ &x(0) = a, \quad x(1) = b. \end{aligned} \tag{2.16}$$

We use the conventional 1D *equidistribution principle*:

$$\omega \frac{\partial}{\partial \xi} x(\xi) = \text{Const.} \tag{2.17}$$

or equivalently

$$\frac{\partial}{\partial \xi}\left\{ \omega \frac{\partial}{\partial \xi} x(\xi) \right\} = 0, \tag{2.18}$$

where ω is the monitor function, which in general depends on the underlying solution to be adapted. Solving (2.18) with the given boundary conditions leads to the desired mesh map $x = x(\xi)$.

For convenience, assume a (fixed) uniform mesh on the computational domain is given by

$$\xi_j = j/(N+1), \quad j = 0, 1, \ldots, N+1.$$

In practice, the monitor function ω is always associated with the underlying solution ϕ or/and its derivatives, but without loss of generality we assume that $\omega = \omega(\phi)$. For monitor functions involving derivatives, central differencing will be used to approximate these derivatives. Then the moving mesh equation (2.18) can be discretized by using central difference approximations:

$$\omega(\phi_{j+\frac{1}{2}})(x_{j+1} - x_j) - \omega(\phi_{j-\frac{1}{2}})(x_j - x_{j-1}) = 0, \quad 1 \leqslant j \leqslant N, \tag{2.19}$$

where $\phi_{j\pm\frac{1}{2}}$ are some averages based on the $\phi_{j\pm1}$ and ϕ_j. Solving (2.19) with the boundary conditions $x(0) = a$ and $x(1) = b$ gives a new grid partition, $\{\widetilde{x}_j\}$, in the physical domain. In our computations, we solve (2.19) by a Gauss–Seidal (GS) iteration

$$\omega(\phi^{[v]}_{j+\frac{1}{2}})\left(x^{[v]}_{j+1} - x^{[v+1]}_j\right) - \omega(\phi^{[v]}_{j-\frac{1}{2}})\left(x^{[v+1]}_j - x^{[v+1]}_{j-1}\right) = 0. \tag{2.20}$$

Once one loop of the above GS iteration is finished, the numerical approximation $\{\phi^{[v]}_j\}$ will be updated by a non-conservative formula to be provided below. When the updated approximation $\{\phi^{[v+1]}_j\}$ is obtained, the GS iteration (2.20) can be employed again to improve the quality of the mesh $\{x_j\}$. This leads to an iteration procedure on the grid motion and solution-interpolation. The iteration determines *progressively better values* of the new grid locations and the approximation values. The total iteration is continued

until there is no significant change in calculated new grids from one iteration to the next. Typically about 3 to 5 cycles of GS iteration are required, so in most computations in this work we use 5 GS iterations at each time level.

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

$$\omega_{j+\frac{1}{2}} \leftarrow \frac{1}{4}(\omega_{j+\frac{3}{2}} + 2\omega_{j+\frac{1}{2}} + \omega_{j-\frac{1}{2}}), \tag{2.21}$$

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

An 2D version of the above GS iteration formula is given by

$$\mathsf{p}_{i+\frac{1}{2},j}\left(\mathbf{r}^{[v]}_{i+1,j} - \mathbf{r}^{[v+1]}_{i,j}\right) - \mathsf{p}_{i-\frac{1}{2},j}\left(\mathbf{r}^{[v+1]}_{i,j} - \mathbf{r}^{[v+1]}_{i-1,j}\right) + \mathsf{q}_{i,j+\frac{1}{2}}\left(\mathbf{r}^{[v]}_{i,j+1} - \mathbf{r}^{[v+1]}_{i,j}\right) - \mathsf{q}_{i,j-\frac{1}{2}}\left(\mathbf{r}^{[v+1]}_{i,j} - \mathbf{r}^{[v+1]}_{i,j-1}\right) = 0, \tag{2.22}$$

for $1 \leqslant i \leqslant N_\xi$ and $1 \leqslant j \leqslant N_\eta$, where $\mathbf{r} := (x, y)$,

$$\mathsf{p}_{i\pm\frac{1}{2},j} = \omega\left(\phi^{[v]}_{i\pm\frac{1}{2},j}\right) = \omega\left(\frac{1}{2}(\phi^{[v]}_{i\pm1,j} + \phi^{[v]}_{i,j})\right),$$

$$\mathsf{q}_{i,j\pm\frac{1}{2}} = \omega\left(\phi^{[v]}_{i,j\pm\frac{1}{2}}\right) = \omega\left(\frac{1}{2}(\phi^{[v]}_{i,j\pm1} + \phi^{[v]}_{i,j})\right).$$

An 3D version can be similarly obtained and it will be omitted here to save space.

2.2.2. A non-conservative interpolation

After obtaining the new grid $\{\widetilde{x}_j\}$, we will update ϕ at the grid point $\widetilde{x}_j$ based on the information of ϕ_j. The traditional way to do it is using the standard interpolation

$$\widetilde{\phi}_j = \phi_k + \frac{\phi_k - \phi_{k-1}}{x_k - x_{k-1}}(\widetilde{x}_j - x_k), \quad \text{if } \widetilde{x}_j \in [x_{k-1}, x_k]. \tag{2.23}$$

In solving the H–J equations with strong derivative discontinuities, the iteration techniques based on (2.19)–(2.23) produce poor numerical approximations, mainly due to the use of the linear interpolation (2.23) which introduces large dissipation at each iteration step. An efficient method to replace (2.23) will be outlined below. If the function is reasonably smooth, then using Taylor expansion gives

$$\phi(\widetilde{x}_j) \approx \phi(x_j) + \left(\frac{\partial\phi}{\partial x}\right)_{x=x_j}(\widetilde{x}_j - x_j) = \phi(x_j) - c_j\left(\frac{\partial\phi}{\partial\xi}\right)_j, \tag{2.24}$$

where

$$c_j := (x_j - \widetilde{x}_j)(\xi_x)_j. \tag{2.25}$$

Eq. (2.24) can be regarded as a numerical approximation of linear convective equation on the computational domain with wave speed c_j. Eq. (2.24) can be also regarded as the following Hamilton–Jacobi type equation

$$\widetilde{\phi} = \phi - c\phi_\xi, \tag{2.26}$$

where c is defined by (2.25). The above equation is of the same form of (2.2) with a linear Hamiltonian. The second-order method described in Section 2.1 will be adopted directly here to solve (2.26).

Remark.

- The interpolation formula given by (2.24) is

$$\tilde{\phi}_j = \phi_j - c_j(\phi_\xi)_j. \tag{2.27}$$

Let $v_{j+\frac{1}{2}}$ be the cell average of ϕ_x as defined by (2.6). Then it can be easily verified that the above interpolation formula preserves the mass for v in the following sense:

$$\sum_j \Delta\tilde{x}_{j+\frac{1}{2}}\tilde{v}_{j+\frac{1}{2}} = \sum \Delta x_{j+\frac{1}{2}} v_{j+\frac{1}{2}}.$$

- As demonstrated in Tang and Tang [33], the mesh moving speed c can be made of order $\mathcal{O}(\Delta\xi)$. As a result, it follows that the error of the interpolation formula (2.27) is of order $\mathcal{O}(\Delta\xi^2)$.

The interpolation formula in 2D is

$$\phi(\widetilde{x}_{i,j}, \widetilde{y}_{i,j}) \approx \phi(x_{i,j}, y_{i,j}) + (\phi_x)_{i,j}(\widetilde{x}_{i,j} - x_{i,j}) + (\phi_y)_{i,j}(\widetilde{y}_{i,j} - y_{i,j}) = \phi(x_{i,j}, y_{i,j}) - (c^\xi)_{i,j}(\phi_\xi)_{i,j} - (c^\eta)_{i,j}(\phi_\eta)_{i,j},$$

where $c^\xi = (\mathbf{r} - \tilde{\mathbf{r}}) \cdot \nabla\xi$ and $c^\eta = (\mathbf{r} - \tilde{\mathbf{r}}) \cdot \nabla\eta$. Here $\mathbf{r} = (x, y)$ and $\nabla = (\partial_x, \partial_y)^{\mathrm{T}}$. In 3D, the interpolation formula is

$$\phi(\widetilde{\mathbf{r}}_{i,j,k}) \approx \phi(\mathbf{r}_{i,j,k}) + \nabla\phi \cdot (\mathbf{r} - \tilde{\mathbf{r}}) = \phi_{i,j,k} - (c^\xi)_{i,j,k}(\phi_\xi)_{i,j,k} - (c^\eta)_{i,j,k}(\phi_\eta)_{i,j,k} - (c^\zeta)_{i,j,k}(\phi_\zeta)_{i,j,k},$$

where $\nabla = (\partial_x, \partial_y, \partial_z)^{\mathrm{T}}$, $\mathbf{r} = (x, y, z)$, $c^\xi = (\mathbf{r} - \tilde{\mathbf{r}}) \cdot \nabla\xi$, $c^\eta = (\mathbf{r} - \tilde{\mathbf{r}}) \cdot \nabla\eta$ and $c^\zeta = (\mathbf{r} - \tilde{\mathbf{r}}) \cdot \nabla\zeta$.

2.3. Summary of the AMR algorithm

An overview of the sequence of computations described in the last two subsections is given below.

Algorithm 2 (*Outline of the numerical algorithm*).

0. Determine the initial mesh based on the initial function.
1. Determine Δt based on CFL-type condition so that $t_n = t_{n-1} + \Delta t$.
2. Advance the solution one time step based on an appropriate numerical scheme for the given H–J equation.
3. Grid Restructuring
   (a) Solve the mesh redistributing equation (a generalized Laplace equation with given data for the monitor) by one cycle of Gauss–Seidal iteration, to get $\mathbf{x}^{[k],n}$, e.g. (2.20) for 1D.
   (b) Interpolating the approximate solutions on the new grid $\mathbf{x}^{[k],n}$, e.g. (2.26) for 1D.
   (c) A weighted average of the locally calculated monitor at each computational cell and the surrounding monitor values, see (2.21).
   (d) The iteration procedure (a)–(c) on grid motion and solution-interpolation is continued until there is no significant change in calculated new grids from one iteration to the next. In practice, a fixed number of iterations may be also used for the procedure (a)–(c).
4. Start new time step (go to 1 above).

The Algorithm 2 will be used to solve some test problems in 1D, 2D and 3D in the following sections.

3. Numerical experiments in 1D

In this section, we apply our adaptive mesh redistribution method to two 1D problems which can be found in [13,27]. The CFL number used in the 1D experiments is 0.48. The main purpose of the 1D experiments are twofolds: one is to discuss the choice of the monitor function and another is to discuss the convergence rate.

Example 3.1. We solve

$$\phi_t + H(\phi_x) = 0, \quad \phi(x,0) = -\cos(\pi(x - x_0)), \ -1 \leqslant x < 1, \tag{3.1}$$

with a convex H (Burgers' equation),

$$H(u) = \frac{1}{2}(u+1)^2. \tag{3.2}$$

The periodic boundary condition is used.

In this example, x_0 is chosen as 0.85. We first wish to verify the convergence rate for the moving mesh algorithm. It is easily seen that the numerical scheme described in Section 2 is of second-order rate of convergence in uniform mesh (when the underlying solution is smooth). It is natural to ask if the second-order rate of convergence can be preserved if an adaptive grid method is employed? To see this, we solve Example 3.1 up to $t = 0.5/\pi^2$ (when the solution is still smooth) with our moving mesh algorithm, where the number of Gauss–Seidal iteration for the grid motion at each time level is taken as 5. In our numerical experiments, the monitor function is taken as

$$\omega = \sqrt{1 + \alpha|\phi_x|^2 + \beta\left|\left(\frac{\phi_x}{\max\{|\phi_x|\}}\right)_x\right|^2}. \tag{3.3}$$

There is a connection between the above monitor and the gradient-based monitor for the hyperbolic conservation laws used in [33]. By the well-known relation that $u \sim \phi_x$, it is natural to choose (3.3) as the monitor function for the one-dimensional H–J equation. In this section, the derivatives appeared in the monitor function are all approximated numerically by central differencing such as

$$\phi_x \approx \frac{1}{2}\left(\frac{\Delta_x^+\phi_j}{\Delta_x^+ x_j} + \frac{\Delta_x^-\phi_j}{\Delta_x^- x_j}\right).$$

The L^1 and L^∞ errors are defined by

$$\sum_j |\phi^{\text{exa}}(x_{j+1/2}) - \phi^{\text{num}}_{j+1/2}|(x_{j+1} - x_j), \ \max_j\{|\phi^{\text{exa}}(x_j) - \phi^{\text{num}}_j|\},$$

respectively, where ϕ^{exa} denotes the exact solution and ϕ^{num} the correspondingly numerical solution. The L^1 and L^∞ errors and convergence orders obtained by using different constants in (3.3) are listed in Tables 1(a)–(d). A second-order rate of convergence is observed for the uniform mesh solution and the adaptive mesh solutions with some constants α and β. However, it is seen from Tables 1(c) and (d) that the rate of convergence for the adaptive mesh scheme are quite erratic. It is known that the convergence rate of the finite volume method depends on the ratio of area of neighboring control volume, i.e. $(x_{j+1} - x_j)/(x_j - x_{j-1})$, see e.g. [10]. Ideally, this ratio for our moving meshes is finite when the PDE solution is smooth. However, since the Gauss–Seidal iteration is used (i.e., we do not solve the resulting mesh

Table 1
Example 3.1: the errors and the rate of convergence of the moving mesh solution at $t = 0.5/\pi^2$, obtained by using the monitor function (3.3) with: (a) $(\alpha, \beta) = (0, 0)$, i.e., uniform mesh; (b) $(\alpha, \beta) = (0, 50^{-1})$; (c) $(\alpha, \beta) = (1, 16^{-1})$ and (d) $(\alpha, \beta) = (1, 8^{-1})$

N	L^1-error	L^1-order	L^∞-error	L^∞-order
(a)				
20	2.61e − 02	–	2.14e − 02	–
40	5.09e − 03	2.34	4.42e − 03	2.28
80	1.12e − 03	2.18	9.81e − 04	2.17
160	2.59e − 04	2.11	2.20e − 04	2.16
320	6.20e − 05	2.06	4.97e − 05	2.15
(b)				
20	2.40e − 02	–	2.05e − 02	–
40	4.57e − 03	2.39	4.14e − 03	2.31
80	9.88e − 04	2.21	8.85e − 04	2.23
160	2.20e − 04	2.17	1.94e − 04	2.19
320	5.43e − 05	2.02	4.50e − 05	2.11
(c)				
20	2.70e − 02	–	1.96e − 02	–
40	6.10e − 03	2.15	4.71e − 03	2.06
80	1.72e − 03	1.83	1.28e − 03	1.88
160	5.24e − 04	1.71	3.55e − 04	1.85
320	1.27e − 04	2.04	8.35e − 05	2.09
(d)				
20	2.53e − 02	–	1.89e − 02	–
40	5.44e − 03	2.22	4.27e − 03	2.15
80	1.42e − 03	1.94	1.07e − 03	2.00
160	3.77e − 04	1.91	2.81e − 04	1.93
320	8.45e − 05	2.16	6.27e − 05	2.16

Eq. (2.19) exactly), the 'regularity' of the ratio may be affected slightly, which may be the main reason responsible for the erratic rate observed.

It is seen from Tables 1(a)–(d) that in general the second-order rate of convergence is preserved when the solution is smooth. However, when solution singularity is developed the convergence rate should be reduced and the expected L^1-rate should be lower than 2. To see this, we display the solution errors and the corresponding convergence rates in Tables 2(a)–(d) at $t = 0.99/\pi^2$ when the derivative singularity is almost developed. It is seen from Tables 2(a)–(d) that the L^1-rate for the AMR solution is very erratic, since the large values of ϕ_{xx} in the monitor function yield (highly) non-quasi uniform meshes. Moreover, the (average) rate of convergence in both L^1 and L^∞ is about 2 for the AMR solution. The rate also depends on the choice of the parameters α and β in the monitor function.

In Fig. 1 the adaptive solution ϕ and its gradient ϕ_x obtained by using 41 grid points are presented. For comparison, the numerical solution on a uniform mesh (with the same number of grid points) is included in Fig. 1. To further demonstrate the improvement with the use of the adaptive mesh, another comparison between the uniform mesh solution and the adaptive mesh solution at $t = 7.2/\pi^2$ is given in Fig. 2. In both cases, higher resolution of the adaptive-mesh solutions over the uniform mesh ones is clearly observed.

Example 3.2. The second 1D example is the following Riemann problem with a non-convex flux:

$$\begin{cases} \phi_t + \frac{1}{4}(\phi_x^2 - 1)(\phi_x^2 - 4) = 0, \quad -1 < x < 1, \\ \phi(x, 0) = -2|x|. \end{cases} \tag{3.4}$$

Table 2
Example 3.1: Same as Tables 1(a)–(d), except $t = 0.99/\pi^2$

N	L^1-error	L^1-order	L^∞-error	L^∞-order
(a)				
20	4.92e − 02	–	5.17e − 02	–
40	1.57e − 02	1.65	2.01e − 02	1.36
80	4.83e − 03	1.70	8.46e − 03	1.25
160	1.60e − 03	1.59	3.72e − 03	1.19
320	7.57e − 04	1.08	1.50e − 03	1.31
(b)				
20	1.63e − 01	–	9.67e − 02	–
40	2.93e − 02	2.78	1.81e − 02	2.42
80	5.10e − 03	2.52	3.38e − 03	2.42
160	8.48e − 04	2.59	6.07e − 04	2.48
320	1.78e − 04	2.25	1.27e − 04	2.26
(c)				
20	1.67e − 01	–	9.42e − 02	–
40	2.72e − 02	2.62	1.68e − 02	2.49
80	1.41e − 03	4.27	4.22e − 03	1.99
160	3.53e − 04	2.00	6.21e − 04	2.76
320	7.23e − 05	2.29	1.93e − 04	1.69
(d)				
20	3.72e − 02	–	4.73e − 02	–
40	2.30e − 03	4.02	1.20e − 02	1.98
80	1.65e − 03	0.48	1.51e − 03	2.99
160	3.59e − 04	2.20	4.33e − 04	1.80
320	8.55e − 05	2.07	1.16e − 04	1.90

As pointed out by Zhang and Shu [37] that this is a demanding test case. Many schemes have poor resolutions or could not even converge to a non-viscosity solution for this case. Numerical results at $t = 1$ with 41 grid points are shown in Fig. 3, and are compared with the exact solution (solid lines). In this example, the monitor function is again taken as (3.3). It is found that the AMR algorithm with 41 grid points gives satisfactory numerical approximations, which seem more accurate than the third-order and fifth-order WENO solutions with 81 non-uniform grid points [37], and also more accurate than the high-resolution central scheme solution with 81 uniform grid points [17]. For comparison, the uniform mesh solution is also included in Fig. 3. The improvement of the AMR solution is then clearly demonstrated by comparing the adaptive mesh solutions and the uniform mesh solution.

4. Numerical experiments in 2D

We consider four test problems in this section. The first two are of smooth initial data, while the last two have discontinuous initial conditions. In fact, the last example in this section is a level-set equation which is concerned with a geometric motion. In all the 2D computations, we took five cycles of the Gauss–Seidal iterations in the mesh redistribution part. The CFL number is chosen as 0.24.

Our 2D monitor function is a direct extension of (3.3). It is chosen as ωI, with I the identity matrix and

$$\omega = \sqrt{1 + \alpha|\nabla_p \phi|^2 + \beta\left|\nabla_p \cdot \left(\frac{\nabla_p \phi}{\max\{|\nabla_p \phi|\}}\right)\right|^2}, \tag{4.1}$$

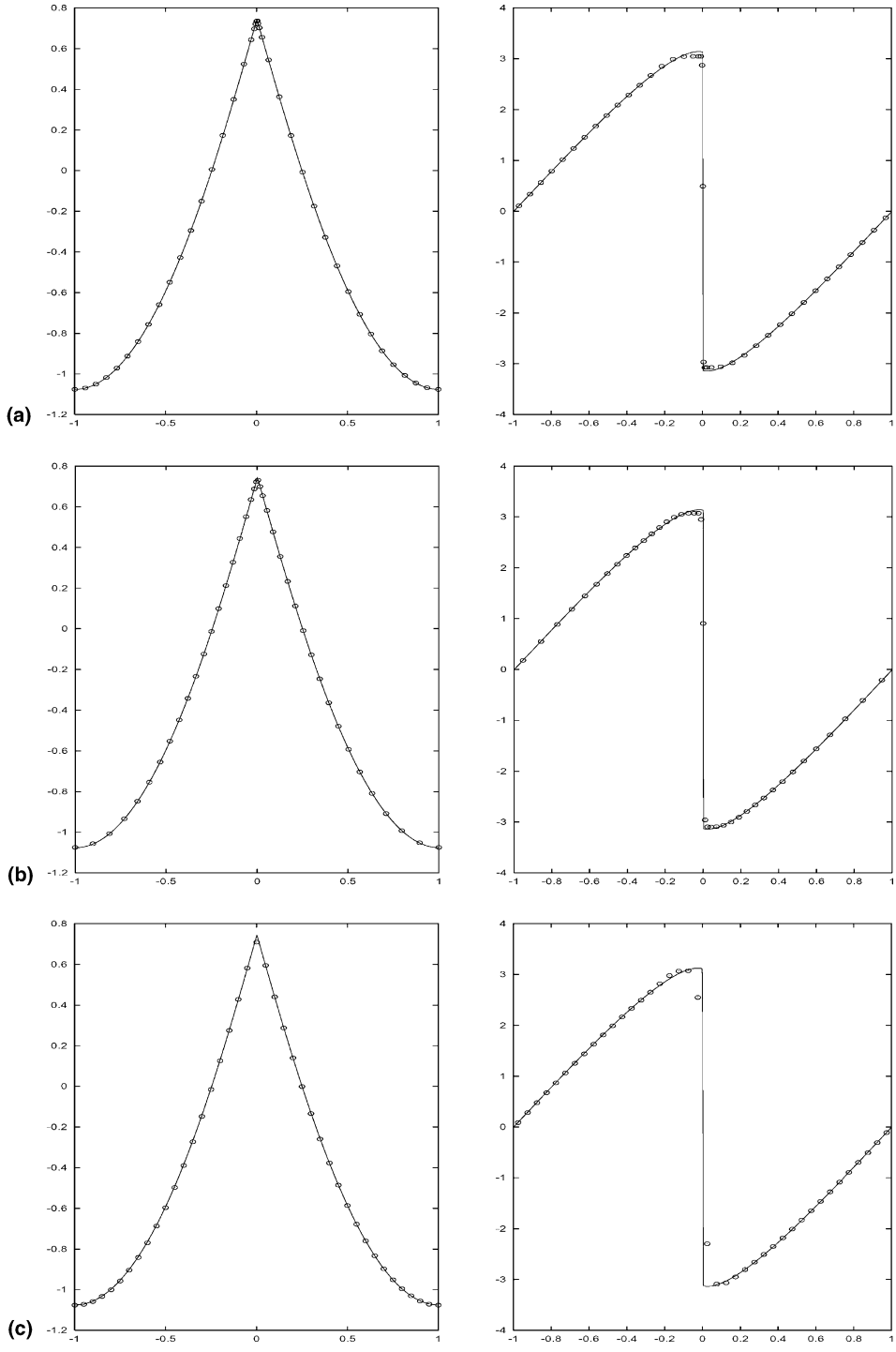

Fig. 1. Example 3.1: the adaptive mesh solutions (○) and the exact solution (solid line) for ϕ (left) and ϕ_x (right) with 40 grid points. $T = 1.5/\pi^2$. Monitor function (3.3) is used with: (a) $(\alpha, \beta) = (1, 16^{-1})$, (b) $(\alpha, \beta) = (0, 50^{-1})$, and (c) $(\alpha, \beta) = (0, 0)$, i.e., uniform mesh.

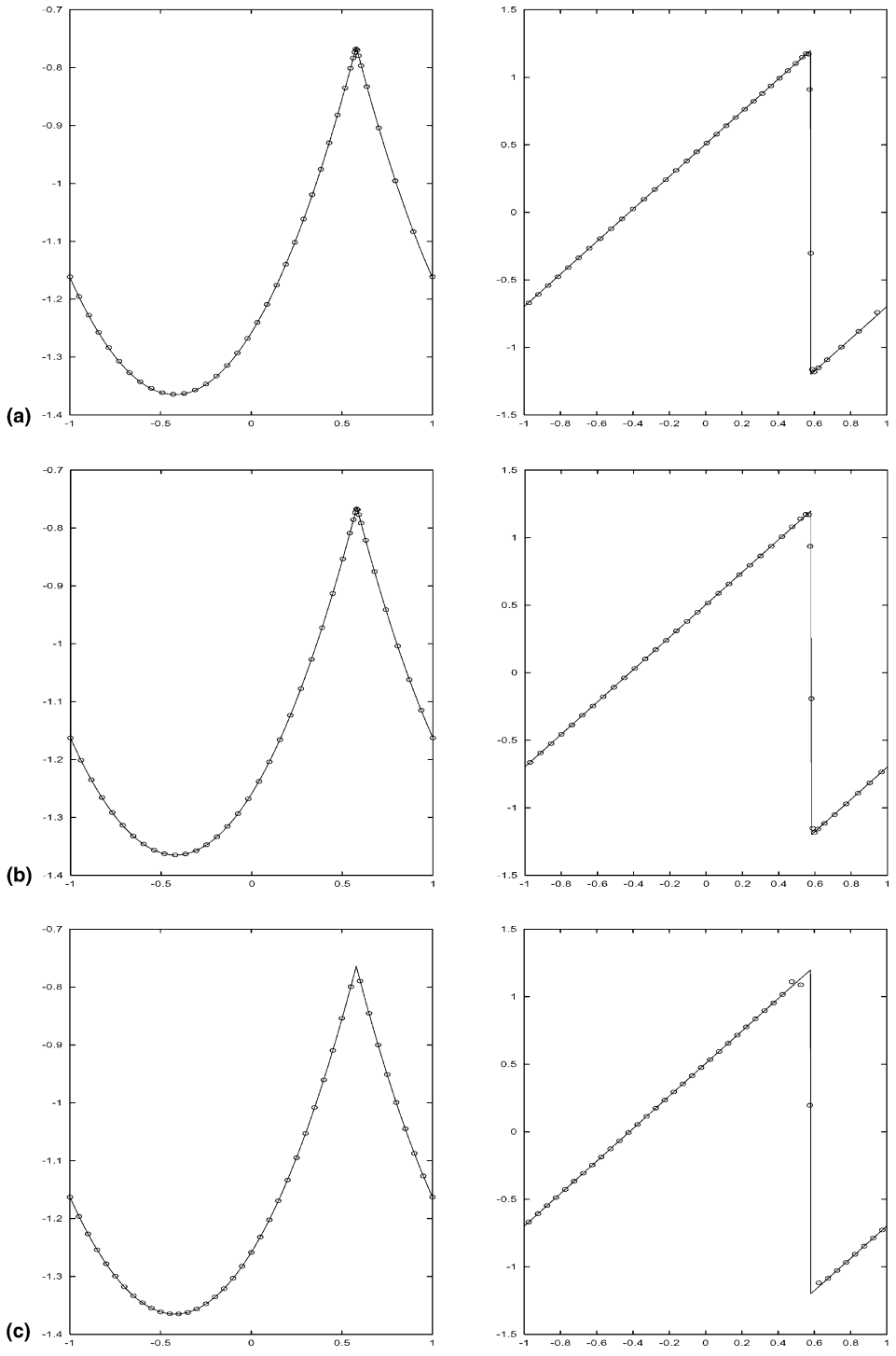

Fig. 2. Example 3.1: the adaptive mesh solutions (○) and the exact solution (solid line) for ϕ (left) and ϕ_x (right) with 40 grid points. $T = 7.2/\pi^2$. Monitor function (3.3) is used with: (a) $(\alpha, \beta) = (1, 16^{-1})$, (b) $(\alpha, \beta) = (0, 50^{-1})$, and (c) $(\alpha, \beta) = (0, 0)$, i.e., uniform mesh.

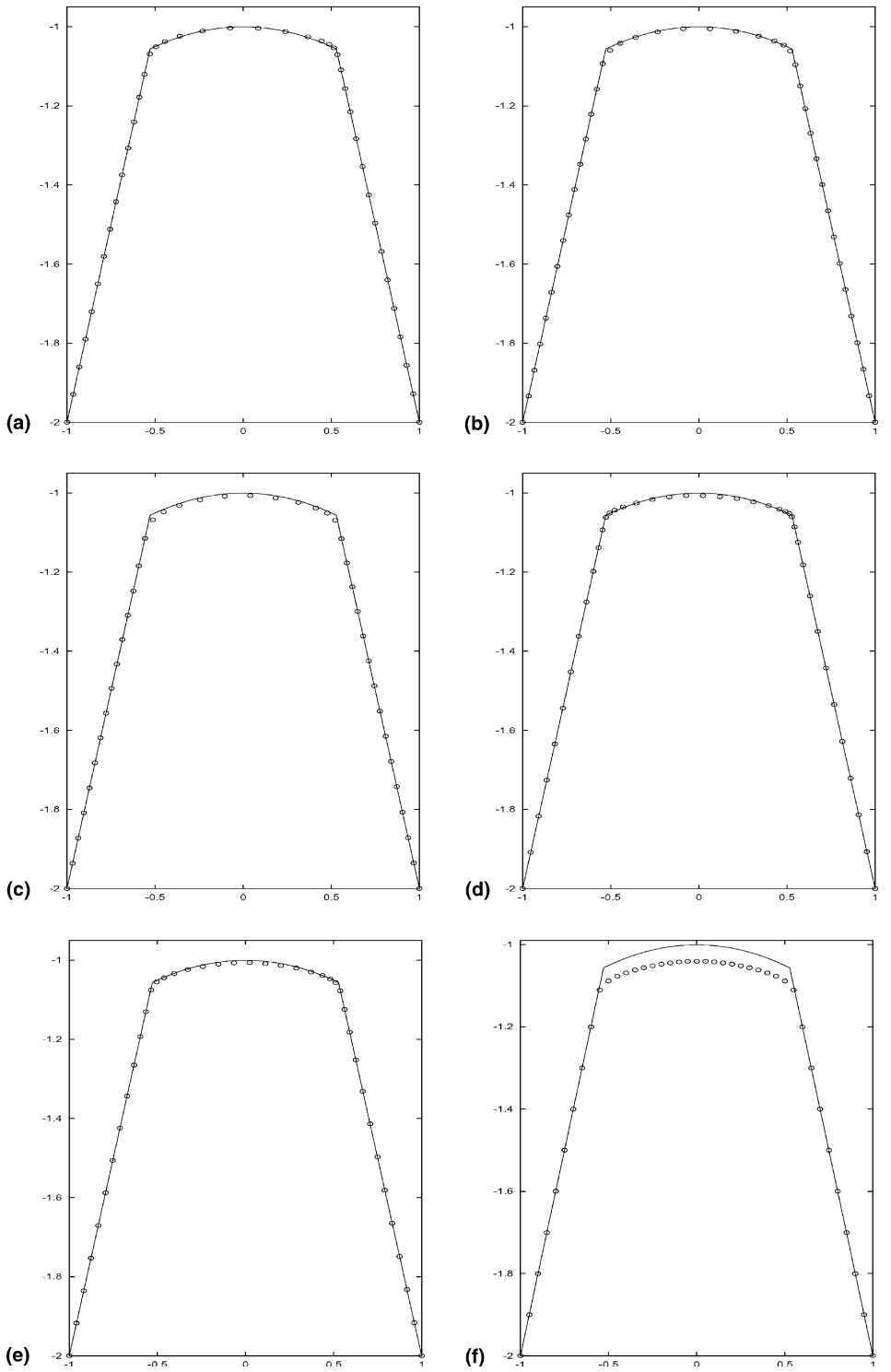

Fig. 3. Example 3.2: numerical solutions (○) and the exact solution (solid line) at $t = 1$ with monitor (3.3): (a) $(\alpha, \beta) = (6, 0.5)$, (b) $(\alpha, \beta) = (6, 0.25)$, (c) $(\alpha, \beta) = (6, 0.125)$, (d) $(\alpha, \beta) = (1, 0.25)$, (e) $(\alpha, \beta) = (1, 0.125)$, and (f) $(\alpha, \beta) = (0, 0)$, i.e., uniform mesh. 41 grid points are used.

where $\nabla_p = (\partial_x, \partial_y)^{\mathrm{T}}$, $\max\{|\nabla_p \phi|\}$ (a function of time t) is a scaling factor. With the use of this scaling factor, the coefficient of the high-order derivative term, β, can be chosen as an $\mathcal{O}(1)$ constant.

Example 4.1. Two-dimensional Burgers equation

$$\begin{cases} \phi_t + H(\phi_x, \phi_y) = 0, \quad (x, y) \in (-2, 2)^2, \\ \phi(x, y, 0) = -\cos(\pi(x + y)/2), \end{cases} \tag{4.2}$$

with a strictly convex Hamiltonian H:

$$H(u, v) = \frac{1}{2}(u + v + 1)^2, \tag{4.3}$$

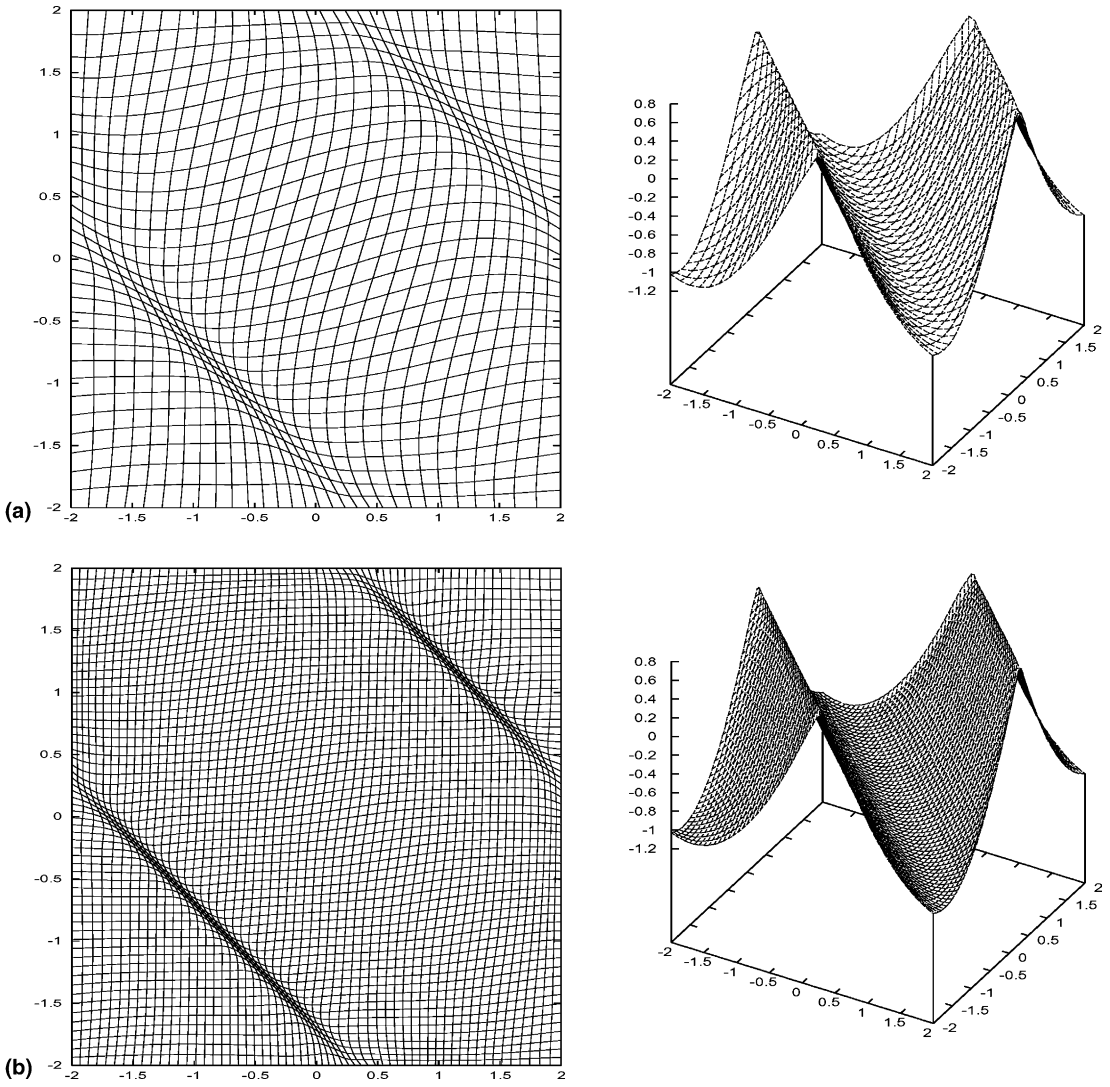

Fig. 4. The convex Hamiltonian (4.3) for Example 4.1: the adaptive mesh (left) and numerical approximation (right) at $t = 1.5/\pi^2$, using the monitor (4.1) and $(\alpha, \beta) = (1, 5)$: (a) 30^2 grid points; (b) 60^2 grid points.

and a non-convex Hamiltonian H:

$$H(u,v) = -\cos(u+v+1). \tag{4.4}$$

The periodic boundary condition is used.

This problem was proposed in [27] and is now a standard test problem. In Fig. 4, the adaptive mesh and numerical solutions at $t = 1.5/\pi^2$ for the convex Hamiltonian (4.3) are presented, and those for the non-convex Hamiltonian (4.4) are plotted in Fig. 5. In both cases, the monitor function ω is of form (4.1) with $(\alpha,\beta) = (1,5)$. It is seen from Figs. 4 and 5 that satisfactory effects of the mesh adaptation are obtained.

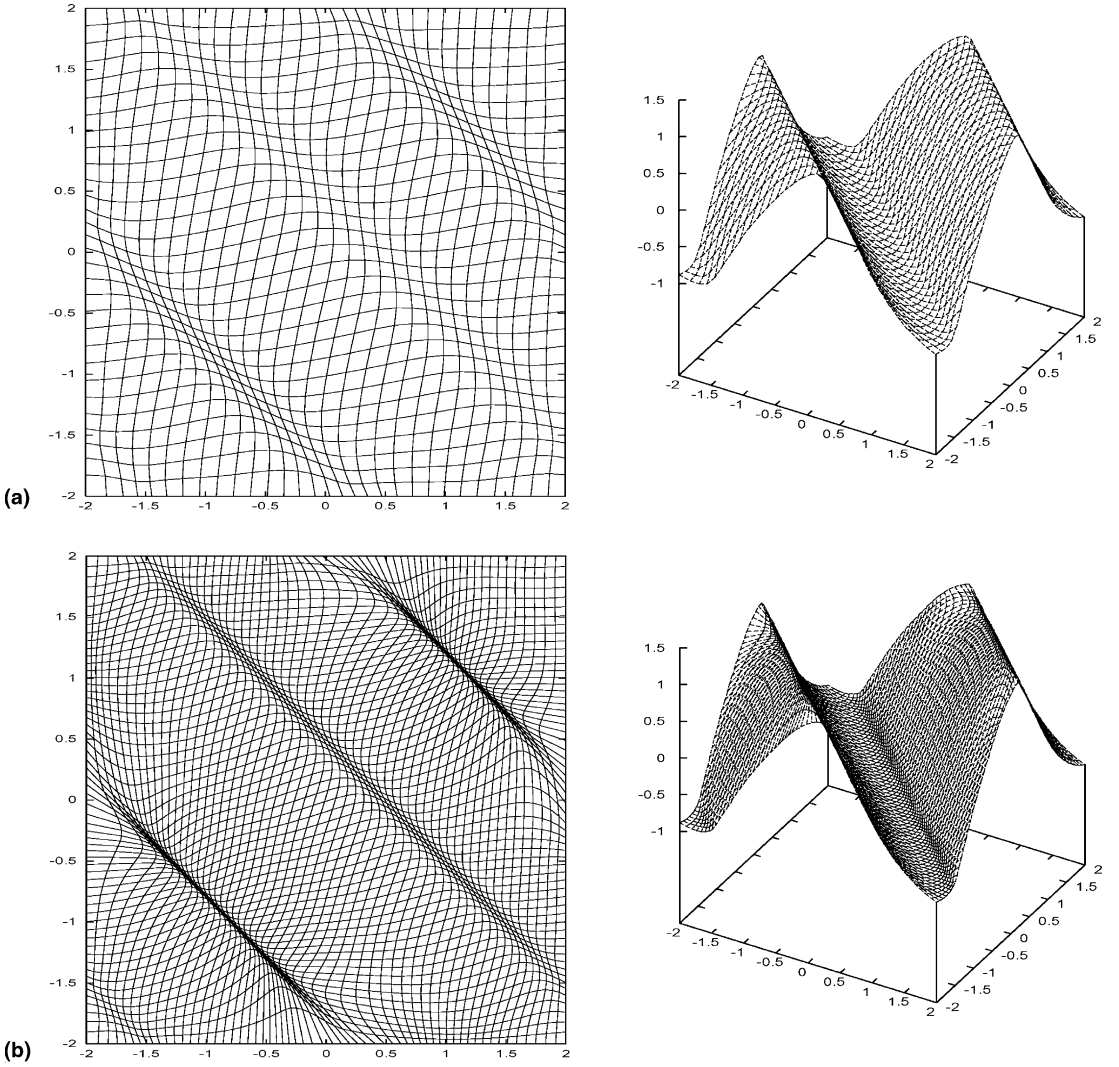

Fig. 5. The non-convex Hamiltonian (4.4) for Example 4.1: the adaptive mesh (left) and numerical approximation (right) at $t = 1.5/\pi^2$, by using the monitor (4.1) with $(\alpha,\beta) = (1,5)$: (a) 30^2 grid points, (b) 60^2 grid points.

It is natural to ask what will be happened if the constant α in (4.1) is set to zero. In Fig. 6, the adaptive mesh and numerical solutions at $t = 1.5/\pi^2$ for the non-convex Hamiltonian (4.4) are plotted, where $(\alpha, \beta) = (0, 5)$ is used. It is observed that similar results to those presented in Fig. 5 are obtained. However, a monitor with non-zero α is useful in enhancing stability, as shown in Table 3 for the next example.

Example 4.2. The second 2D problem is a prototype model in geometrical optics [16,24,26], which is a Cauchy problem for an 2D H–J equation with a non-convex Hamiltonian and a periodic boundary condition

$$\begin{cases} \phi_t + \sqrt{\phi_x^2 + \phi_y^2 + 1} = 0, \quad (x, y) \in (0, 1)^2, \\ \phi(x, y, 0) = 0.25(\cos(2\pi x) - 1)(\cos(2\pi y) - 1) - 1. \end{cases} \tag{4.5}$$

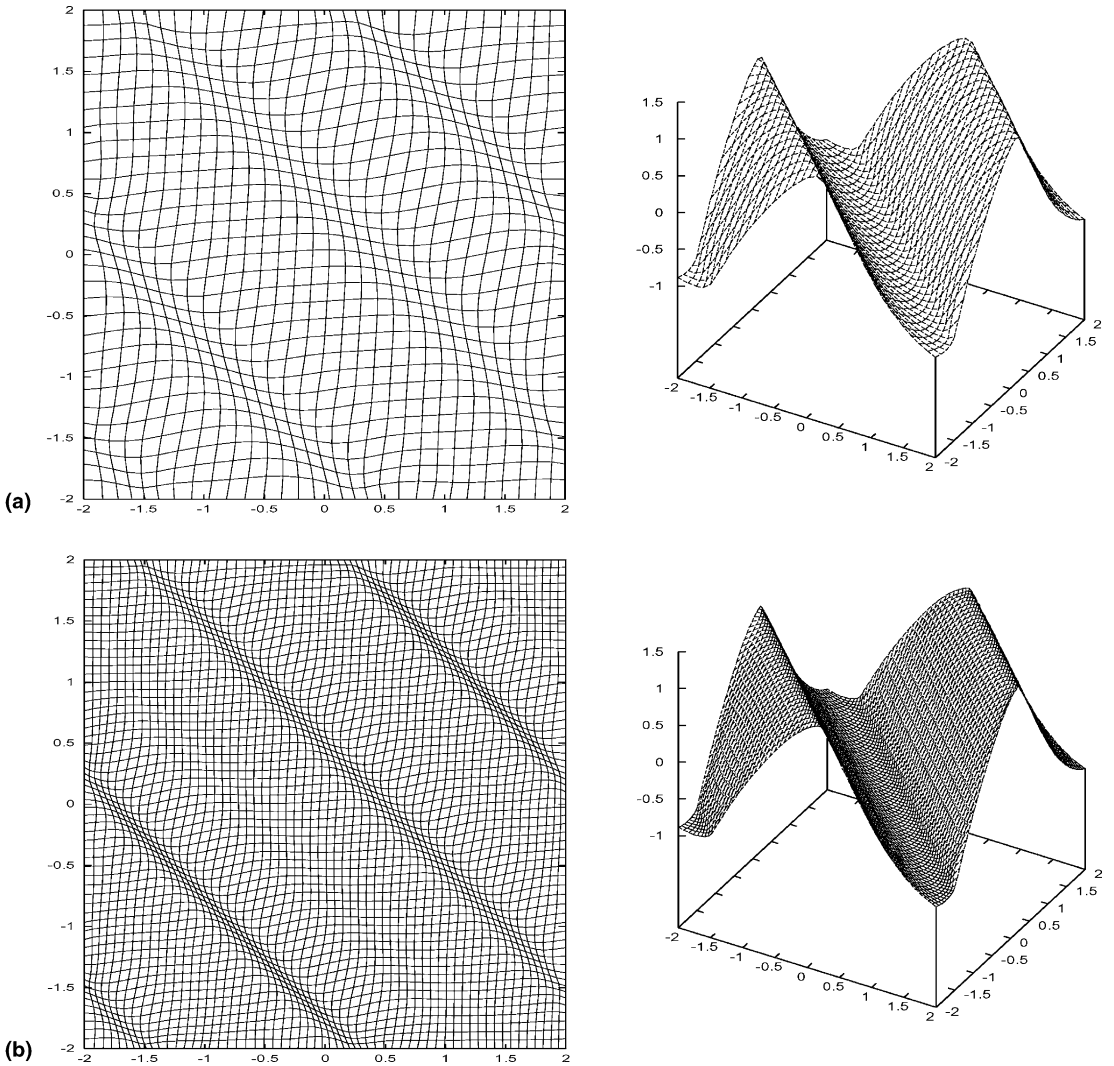

Fig. 6. Same as Fig. 5, except with $(\alpha, \beta) = (0, 5)$.

Table 3
Example 4.2: efficiency vs. effectiveness with different monitor functions

(α, β)	No. of time evolution (30^2 grids)	No. of time evolution (60^2 grids)	Quality of mesh adaptation
$(6, 0.1)$	252	650	Very good
$(6, 0)$	206	344	Satisfactory
$(1, 0)$	152	282	Poor
$(0, 0.1)$	219	Very large	–

For this example, the monitor function is of the form (4.1) with $(\alpha, \beta) = (6, 0.1)$. Using our moving mesh algorithm, we record data at $t = 0.6$ (after singularity) with 30^2 and 60^2 grids. In Fig. 7, the adaptive mesh and the approximation solutions are presented. It is seen that the problem is well resolved with 30^2 grid

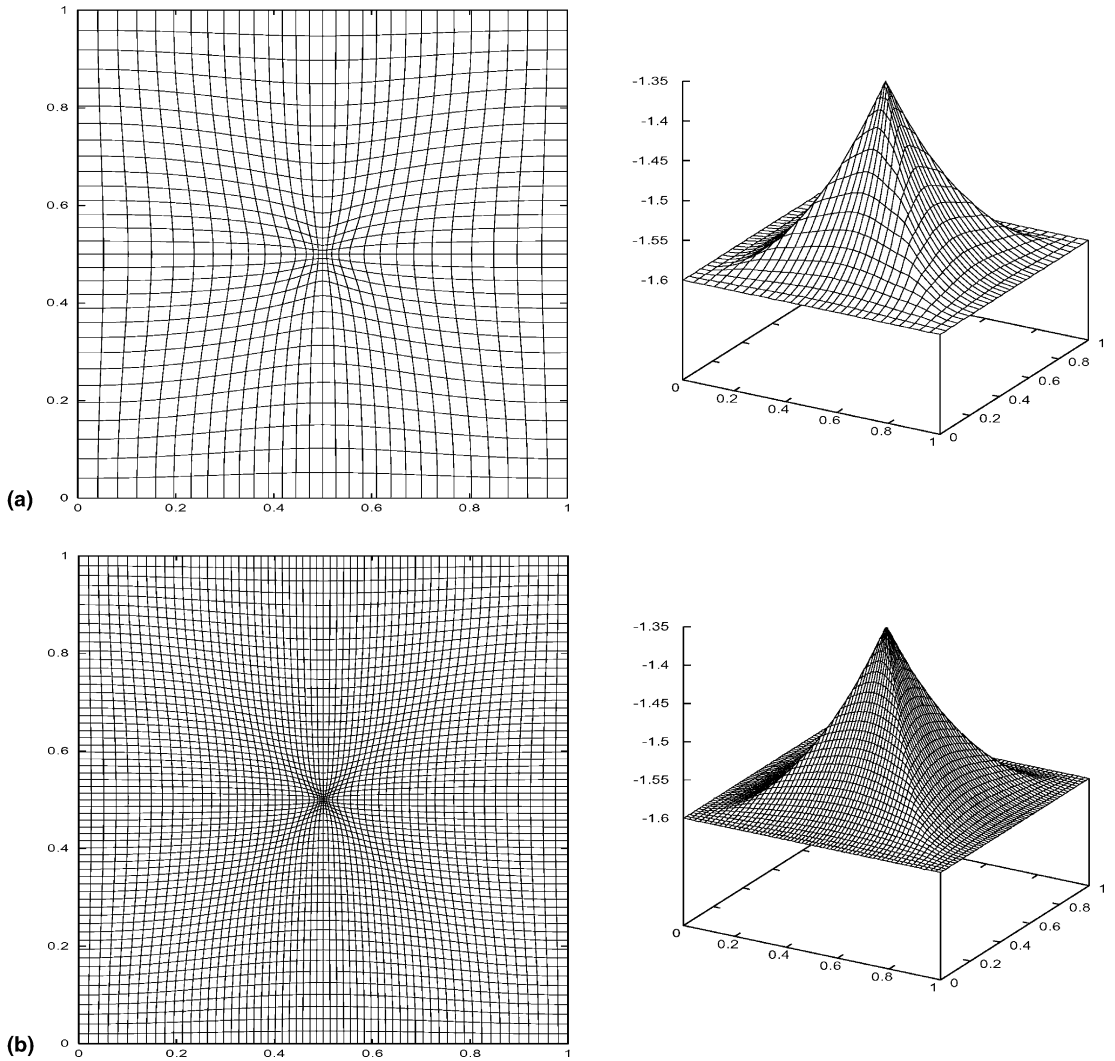

Fig. 7. Example 4.2: moving mesh (left) and its solution (right) with: (a) 30^2 and (b) 60^2 (bottom) grid points. Monitor function used is (4.1), with $(\alpha, \beta) = (6, 0.1)$. $T = 0.6$.

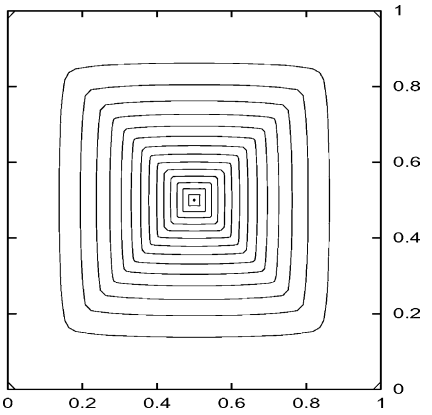

Fig. 8. Example 4.2: contours of moving mesh solution with 60^2 grid points at $t = 0.6$ (15 contour lines are used.)

points, and the desired mesh adaptation effect is clearly observed. The contour lines at $t = 0.6$ obtained by using 60^2 grid points is presented in Fig. 8.

It will be interesting to test the gradient-based monitor for this problem, namely dropping the curvature-like term in (4.1). Numerical results on the moving mesh and the approximate solutions are plotted in Fig. 9 by using $(\alpha, \beta) = (6, 0)$. By comparing Figs. 7 and 9, we found that more grid points moved to the peak solution regions when the curvature-based monitor (4.1) is used, indicating the useful impact of the curvature-like term in the monitor.

A number of observations from our computation to this example are listed below:

- The coefficient of the higher-order derivative term in the monitor (4.1) can not be too large, i.e., the coefficient β in (4.1) should be quite small. One of the reasons is that if β is large then the time step in the time evolution stage may become very small. In Example 4.2, $\beta \geqslant 0.5$ may lead to serious problems: in this case the mesh is too fine around the singularity and as a result an extremely small time step has to be used.
- If a gradient-monitor is chosen, i.e., $\beta = 0$ in (4.1), then the coefficient of the gradient term can not be too small. Otherwise the effect of mesh adaptation will not be obtained (i.e., the mesh is almost not moved). In Example 4.2, if we use $(\alpha, \beta) = (1, 0)$ then the resulting adaptive mesh almost becomes uniform.
- The choices of the coefficients α and β in (4.1) have also quite big impact on the total CPU time used. In Table 3, we list the number of time evolutions used in solving this test problem with different choice of the monitor functions: $(\alpha, \beta) = (6, 0.1), (6, 0), (1, 0)$ and $(0, 0.1)$.

The next example is concerned with a linear scalar advection problem. One of the differences between this one and the last two examples is that the initial condition for ϕ is no longer continuous, which has some impact on the choice of the monitor functions.

Example 4.3. The 2D problem to be considered is a scalar advection equation

$$\phi_t + u\phi_x + v\phi_y = 0, \quad (x, y) \in (0, 1)^2, \tag{4.6}$$

where the velocity field corresponds to a swirling deformation flow of the form

$$u(x, y, t) = \sin^2(\pi x)\sin(2\pi y)g(t),$$

$$v(x, y, t) = -\sin(2\pi x)\sin^2(\pi y)g(t).$$

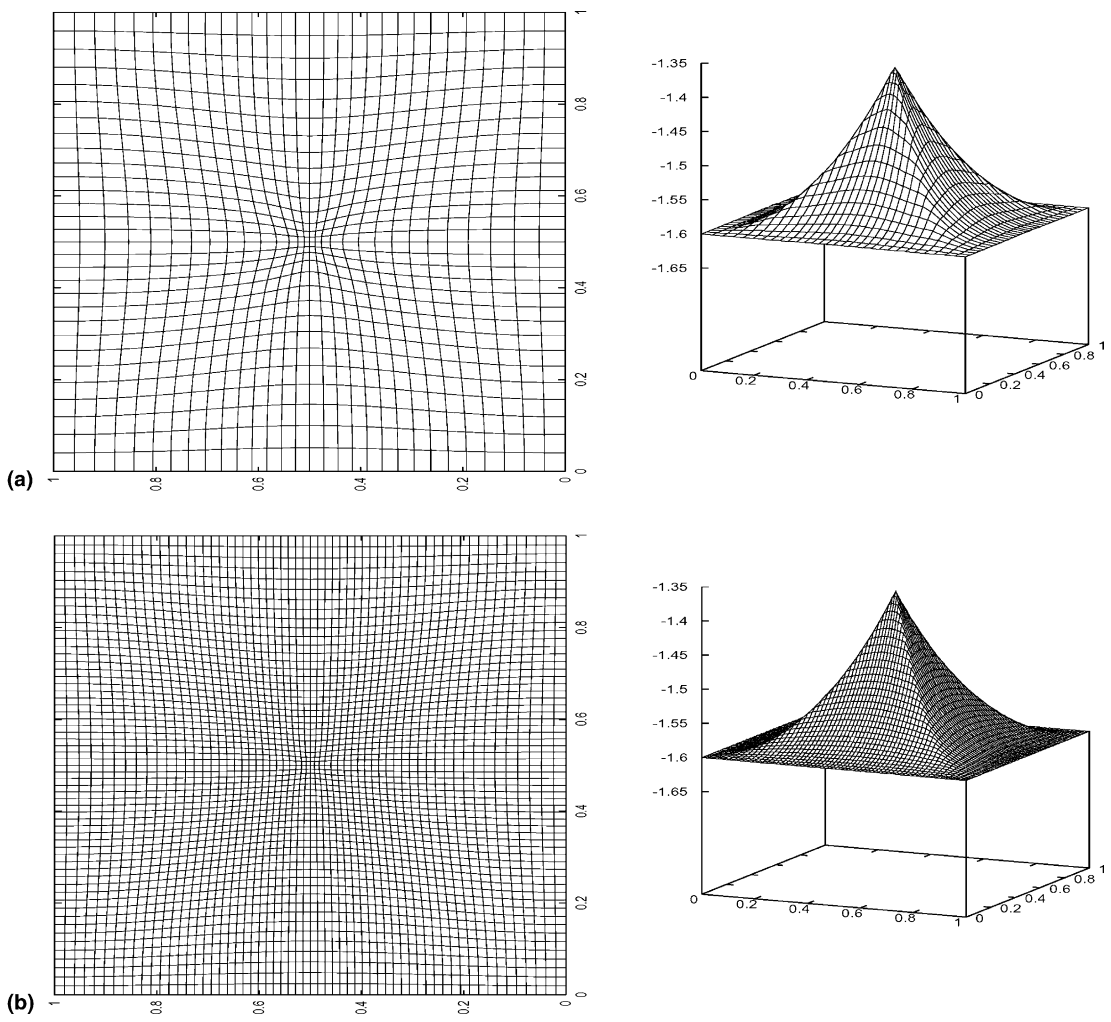

Fig. 9. Same as Fig. 7, except that $(\alpha, \beta) = (6, 0)$.

The time dependency $g(t)$ is given by $g(t) = \cos(\pi t/T_0)$. Initially a discontinuity is placed at $x = 0.5$:

$$\phi(x, y, 0) = \begin{cases} 1, & x \leqslant 0.5, \\ 0, & x > 0.5. \end{cases} \tag{4.7}$$

Due to the velocity dependency, the flow reverses at $t = (1/2)T_0$, and the initial discontinuity should be recovered at $t = T_0$. In the present computation, T_0 is chosen as 0.8. Since the initial condition (4.7) is discontinuous, it is reasonable to believe that the gradient-based monitor function should be employed. The idea is similar to that for the shock problems, see e.g. [3,33]. The monitor function is taken as ωI with

$$\omega = \sqrt{1 + |\nabla_p \phi|^2}. \tag{4.8}$$

The numerical results with 40^2 grid points are shown in Fig. 10. This problem was proposed by LeVeque [20]. The solution of this problem reverses direction in such a way that the initial data should be recovered

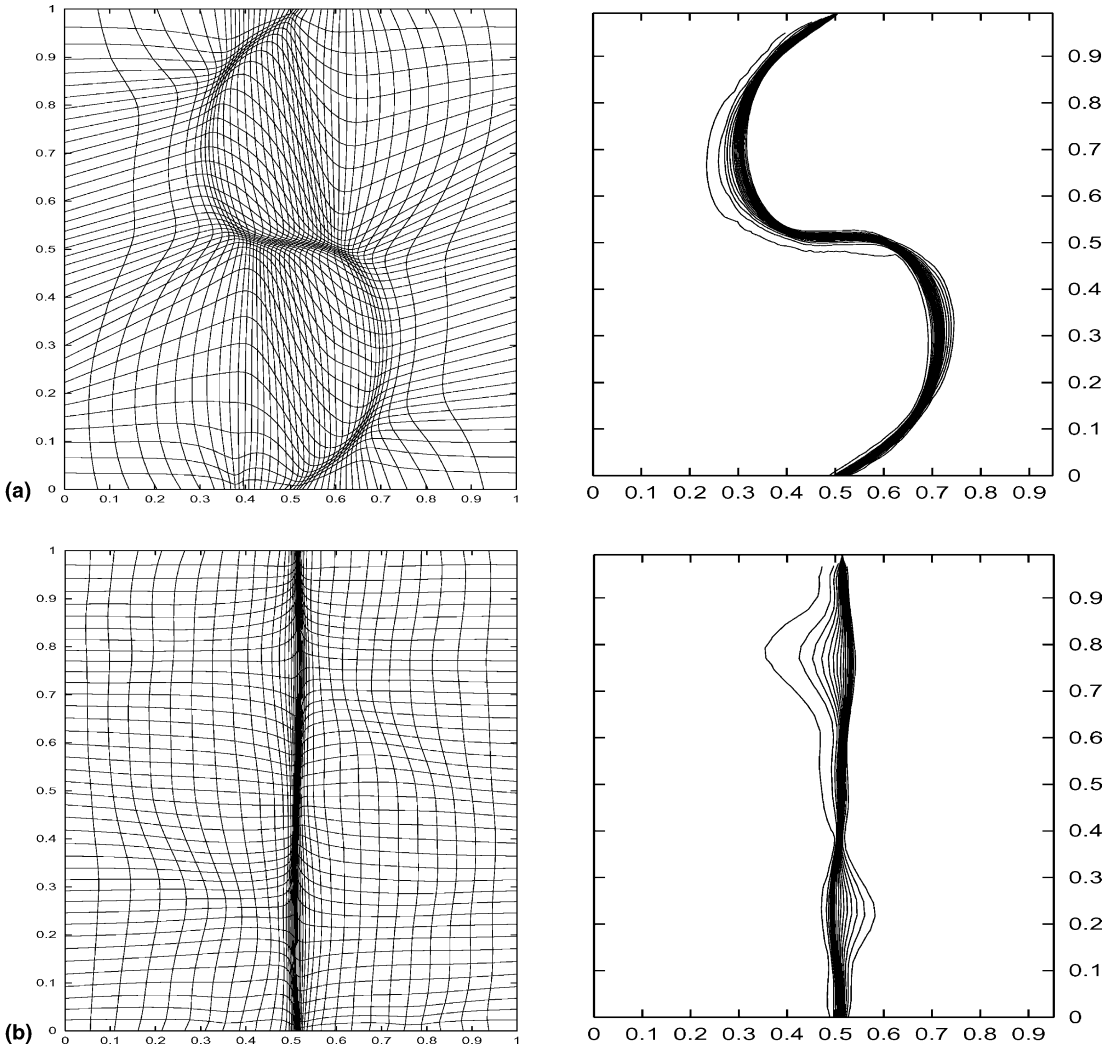

Fig. 10. Example 4.3: moving mesh (left) and its solution (right) with 40^2 grid points at: (a) $t = T_0/2$ and (b) $t = T_0$.

at time T_0: $\phi(x, y, T_0) = \phi(x, y, 0)$. This gives a very useful test problem since the true solution at time T_0 is known even thought the flow field has a quite complicated structure. Our numerical results show that the initial shapes have been recovered fairly well at $t = T_0$, and that the AMR scheme adapts the mesh very well to the regions with large solution gradients. Of course, the smearing introduced during the deformation will not be eliminated as the flow inverses, and so the resolution seen here seems quite good.

To demonstrate the ability of our moving mesh strategy, we will consider a geometric-motion problem investigated by Barth and Sethian [4]. The problem is about the motion of a simple closed curve which is the boundary of an "**H**" shape, as plotted in the top right of Fig. 11.

Example 4.4. The governing equation is a level-set equation

$$\phi_t - \kappa|\nabla\phi| = 0, \quad (x, y) \in (0, 1)^2, \tag{4.9}$$

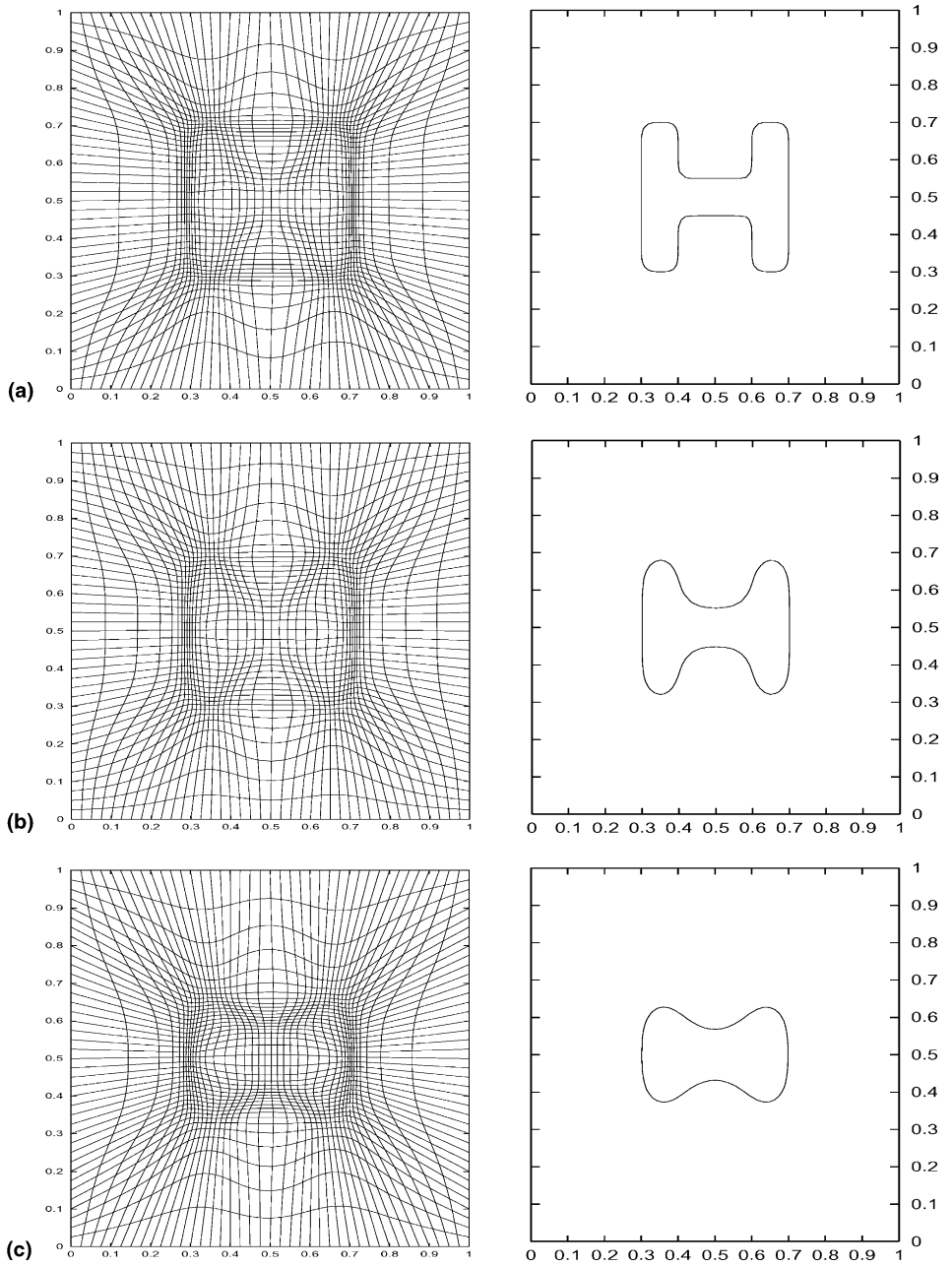

Fig. 11. Example 4.4: adaptive mesh and numerical solution with 80^2 grid points, at: (a) $t = 0$, (b) 10^{-3}, (c) 3×10^{-3}, (d) 5×10^{-3}, (e) 10^{-2} and (f) 1.4×10^{-2}.

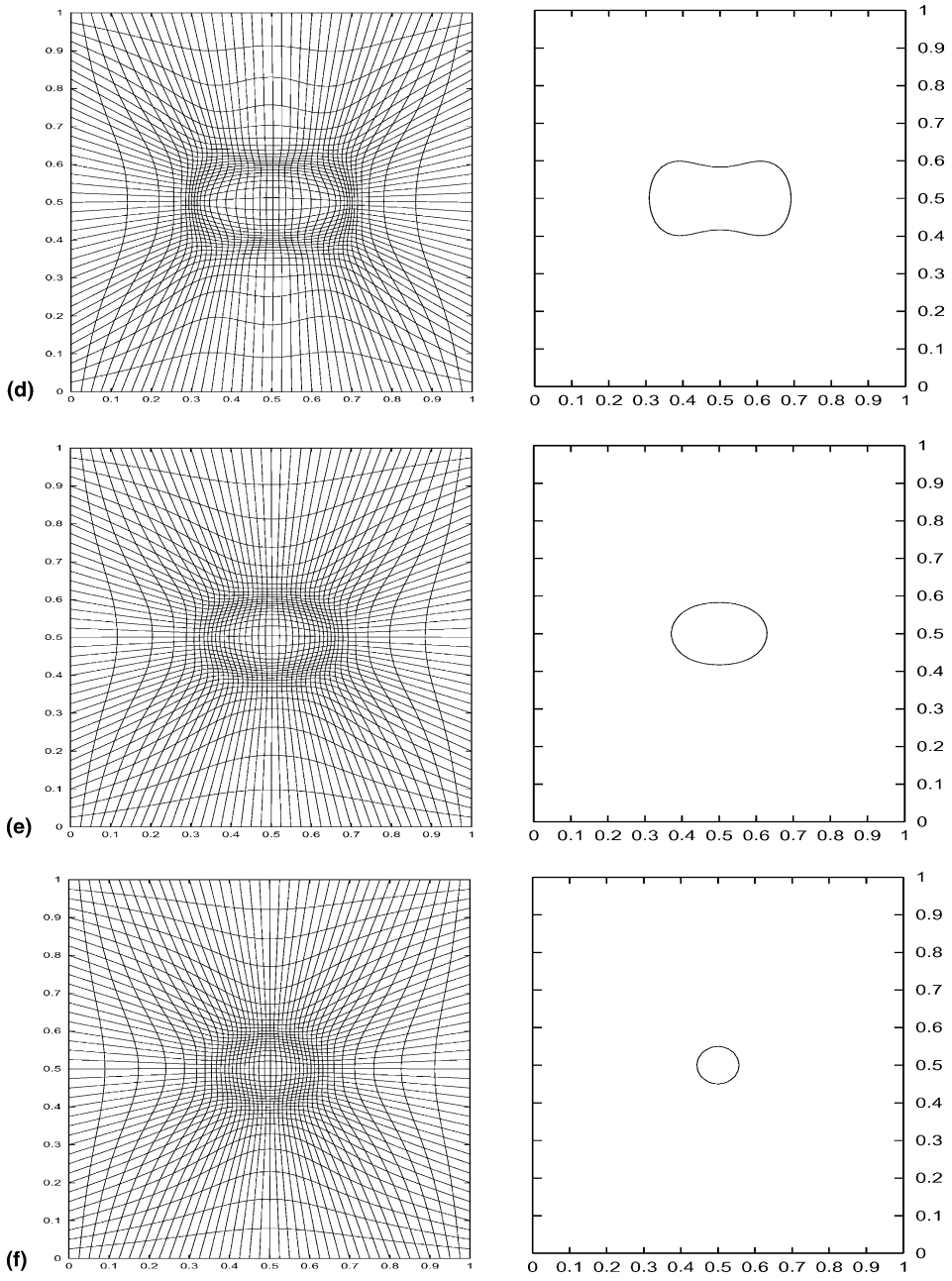

Fig. 11. (*continued*)

where κ is the curvature of ϕ, defined by $\kappa = \nabla_p \cdot \nabla_p\phi/|\nabla_p\phi|$, and the initial data are given by

$$\phi(x,y,0) = \begin{cases} 1, & (x,y) \in \mathbf{H}, \\ -1, & (x,y) \in \Omega_{\mathrm{p}}/\overline{\mathbf{H}}, \\ 0, & \text{otherwise.} \end{cases}$$

This is an example of curvature driven movement of an interface. The zero level set of ϕ defines the interface. We wish to numerically simulate the geometric motion by solving the above level-set equation collapsing with speed proportional to local curvature. Grayson [11] proved that all simple closed curves moving with curvature will eventually collapse to a round point. This example serves as a challenging test problem to verify Grayson's theory.

We briefly discuss how to discretize the curvature κ which involves second derivatives of ϕ. In our algorithms, central-differencing is used to discretize the derivatives involved in the curvature. Namely, they are all approximated by second-order central difference approximations in the computational domain. For example,

$$\phi_x = \frac{1}{J}(\phi_\xi y_\eta - \phi_\eta y_\xi)$$

is approximated by

$$(\phi_x)_{j,k} \approx \frac{1}{J_{j,k}}\left(\frac{\phi_{j+1,k}-\phi_{j-1,k}}{2\Delta\xi}\frac{y_{j,k+1}-y_{j,k-1}}{2\Delta\eta} - \frac{\phi_{j,k+1}-\phi_{j,k-1}}{2\Delta\eta}\frac{y_{j+1,k}-y_{j-1,k}}{2\Delta\xi}\right),$$

where $J = x_\xi * y_\eta - x_\eta * y_x$, whose derivatives are approximated again by central differencing.

It is suggested in Sethian's book [29] that for curvature driven flow the discretization should be of central difference type as the equation is parabolic. It can be shown that if the scheme (2.11) is used, together with a central-differencing approach to the curvature which involves higher-order derivatives, then the overall scheme is indeed of central difference type.

Since the initial function is discontinuous, we again choose the gradient-based monitor function (4.8) in our AMR algorithm. Numerical results obtained by using 40^2 grid points are shown in Fig. 11, which indicate that more grid points are moved into the regions of physical significance. As desired, adaptive mesh is also part of the numerical solution: the areas with dense points also give rough shape of the moving curve. For this simple example, we demonstrated numerically that Grayson's theory on the motion of simple closed curves holds.

5. Numerical experiments in 3D

A challenging task for a useful adaptive mesh redistribution method is its feasibility for 3D computations, since 3D mesh adaptation is much more complicated than that of 1D or 2D. In 2D, it is demonstrated in the last section that our AMR algorithm adapts the mesh extremely well to the solution without producing skew volumes. In this section, we wish to demonstrate that this is also true in 3D. The last example in this work is concerned with an 3D advection problem. The CFL number used is 0.12.

Example 5.1. The last example is the same as Example 4.3, except that the solution domain becomes $\Omega = (-1,1)^3$. More precisely, this problem is concerned with a scalar advection equation

$$\phi_t + u\phi_x + v\phi_y + w\phi_z = 0, \quad (x,y,z) \in (-1,1)^3, \tag{5.1}$$

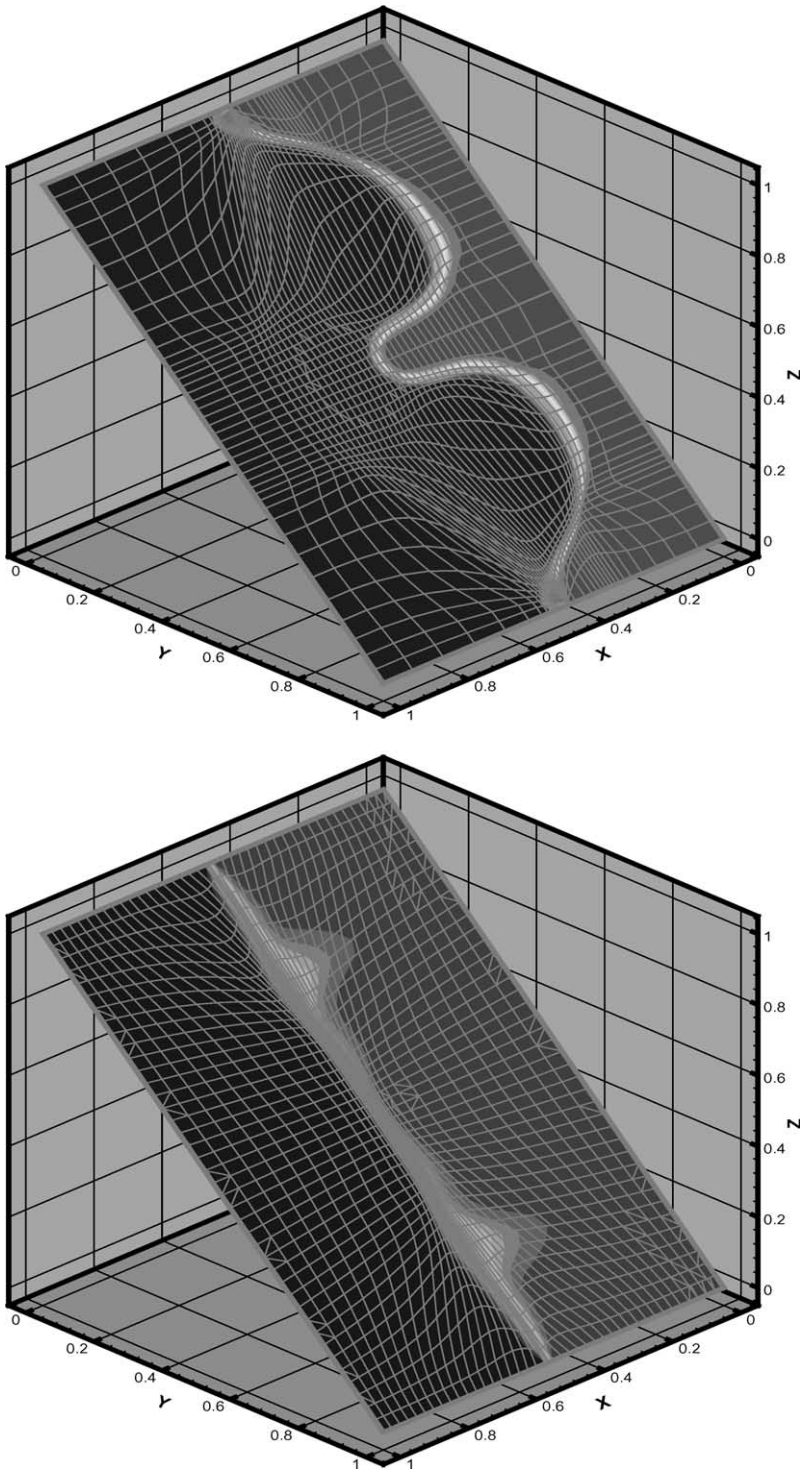

Fig. 12. Numerical solution and the corresponding mesh at $t = T_0/2$ (top) and $t = T_0$ (bottom), on the cut-plane $y + z = 1$.

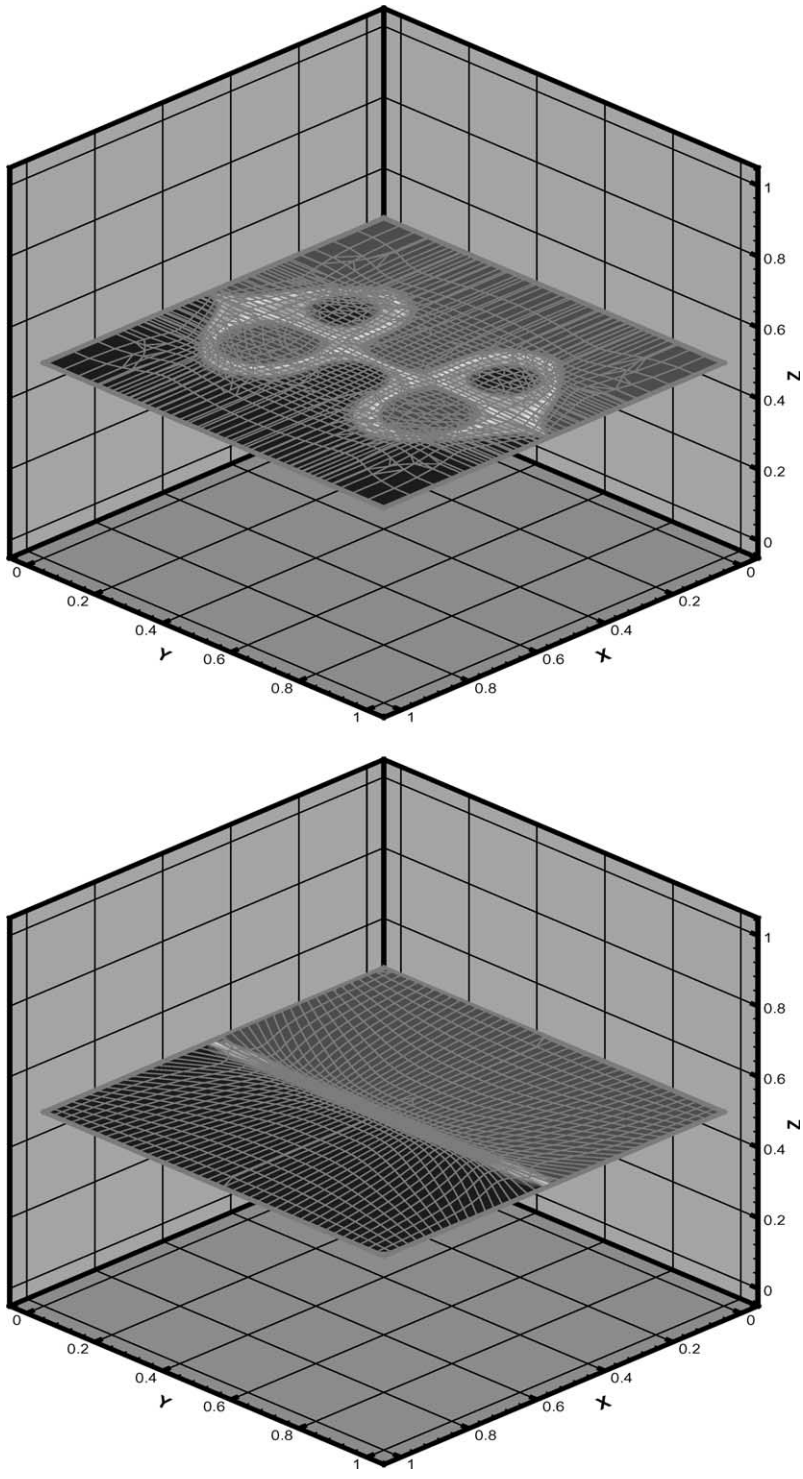

Fig. 13. Same as Fig. 12, except on the cut-plane $z = 0.5$.

where the velocity field corresponds to a swirling deformation flow of the form

$$u(x,y,z,t) = 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)g(t),$$

$$v(x,y,z,t) = -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)g(t),$$

$$w(x,y,z,t) = -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)g(t).$$

The time dependency $g(t)$ is the same as that defined in Example 4.3. The initial condition is

$$\phi(x,y,z,0) = \begin{cases} 1, & x \leqslant 0.5, \\ 0, & x > 0.5. \end{cases} \tag{5.2}$$

As in 2D, the flow reverses at $t = (1/2)T_0$, and the initial discontinuity is recovered at $t = T_0$. This problem was proposed and solved by LeVeque [20]. Again, due to the discontinuity for the solution ϕ, the gradient-based function $\omega = \sqrt{1 + 2|\nabla_p \phi|^2}$ is used. In the present computation, we take $T_0 = 0.8$, and 40^3 grid points are used. Figs. 12 and 13 present the adaptive mesh solutions at two different time levels, $t = 0.4 = T_0/2$ and $t = 0.8 = T_0$. It is observed that our AMR scheme adapts the mesh extremely well to the regions with large solution gradients. Unlike on the structured mesh, an arbitrary cut-plane may not have enough grid points from the adaptive meshes. To handle this problem, the nearby points are projected into the chosen plane. As observed in Figs. 12 and 13, the projection may yield a few *spurious* volumes, due to the complicated structure of the 3D grids.

6. Concluding remarks

One of efficient ways to increase the resolution for the numerical approximations of the Hamilton–Jacobi equations is to use higher-order numerical discretization, such as the high-order ENO scheme of Osher and Shu [27], high-resolution central schemes of Kurganov and Tadmor [18], and high-order discontinuous Galerkin finite element method of Cockburn and Shu [7] and Hu and Shu [12]. In this work, we have demonstrated that adaptive mesh redistribution method can also increase the solution resolution, so that accurate numerical solutions can be obtained by using relatively small number of grid points.

The main contributions of this work are twofolds. First, we describe a simple numerical scheme which consists of a second-order finite volume scheme and a novel AMR strategy. Second, selection of the monitor functions is investigated. It is concluded that if the numerical solution is continuous but the derivatives are discontinuous (in the generic case, H–J solutions form discontinuous derivatives in a finite time even with smooth initial conditions) then a monitor function involving a curvature-like term is desired. However, if the solution has initial discontinuity, then a gradient-based monitor is sufficient.

A number of numerical experiments have been carried out, including 2D and 3D convection problems and a level-set problem. The numerical computations indicate that the adaptive mesh algorithm described in this work can cluster grid points into the regions of the physical domain where the solution has singularity behaviors, and as a result high resolution of the numerical approximation can be achieved with relatively small number of grid points.

Acknowledgements

The research of HZT and PWZ was supported by the Special Funds for Major State Basic Research Projects of China and the National Natural Science Foundation of China. HZT was also supported by

Alexander von Humboldt Foundation. TT was supported in part by the Hong Kong Research Grants Council. The authors wish to thank the referees for useful suggestions.

References

[1] R. Abgrall, Numerical discretization of first order Hamilton–Jacobi equations on triangular meshes, Commun. Pure Appl. Math. 49 (1996) 1339–1373.
[2] S. Augoula, R. Abgrall, High order numerical discretization for Hamilton–Jacobi equations on triangular meshes, J. Sci. Comp. 15 (2000) 197–229.
[3] B.N. Azarenok, S.A. Ivanenko, T. Tang, Adaptive mesh redistribution method based on Godunov's scheme, Commun. Math. Sci. 1 (2003) 152–179.
[4] T.J. Barth, J.A. Sethian, Numerical schemes for the Hamilton–Jacobi and Level set equations on triangulated domains, J. Comput. Phys. 145 (1998) 1–40.
[5] J.U. Brackbill, J.S. Saltzman, Adaptive zoning for singular problems in two dimensions, J. Comput. Phys. 46 (1982) 342–368.
[6] M.G. Crandall, P.L. Lions, Viscosity solutions of Hamilton–Jacobi equations, Trans. Am. Math. Soc. 277 (1983) 1–42.
[7] B. Cockburn, C.-W. Shu, Runge–Kutta discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comp. 16 (2001) 173–261.
[8] M.G. Crandall, H. Ishi, P.L. Lions, User's guide to viscosity solutions of second order partial differential equations, Bull. Am. Math. Soc. 27 (1992) 1–67.
[9] M.G. Crandall, P.L. Lions, Two approximations of solutions of Hamilton–Jacobi equations, Math. Comput. 43 (1984) 1–19.
[10] C.A.J. Fletcher, Computational Techniques for Fluid Dynamics, second ed., Springer, Berlin, 1991.
[11] M. Grayson, The heat equation shrinks embedded plane curves to round points, J. Diff. Geom. 26 (1987) 285–314.
[12] C.Q. Hu, C.W. Shu, A discontinuous Galerkin finite element method for Hamilton–Jacobi equations, SIAM J. Sci. Comput. 21 (2000) 666–690.
[13] G.S. Jiang, D.P. Peng, Weighted ENO schemes for Hamilton–Jacobi equations, SIAM J. Sci. Comput. 21 (2000) 2126–2143.
[14] G. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1996) 202–228.
[15] E.R. Jakobsen, K.H. Karlsen, N.H. Risebro, On the convergence rate of operator splitting for Hamilton–Jacobi equations with source terms, SIAM J. Numer. Anal. 39 (2001) 499–518.
[16] S. Jin, Z.P. Xin, Numerical passage from systems of conservation laws to Hamilton–Jacobi equations, and relaxing schemes, SIAM J. Numer. Anal. 35 (1998) 2385–2404.
[17] A. Kurganov, E. Tadmor, New high-resolution central schemes for nonlinear conservation laws and convective-diffusion equations, J. Comput. Phys. 160 (2000) 214–282.
[18] A. Kurganov, E. Tadmor, New high-resolution semi-discrete central schemes for Hamilton–Jacobi equations, J. Comput. Phys. 160 (2000) 720–742.
[19] N.N. Kuznetsov, Accuracy of some approximate methods for computing the weak solutions of a first-order quasi-linear equation, USSR Comput. Math. Math. Phys. 16 (1976) 105–119.
[20] R.J. LeVeque, High-resolution conservative algorithms for advection in incompressible flow, SIAM J. Numer. Anal. 33 (1996) 627–665.
[21] X.-D. Liu, S. Osher, T. Chan, Weighted essentially nonoscillatory schemes, J. Comput. Phys. 115 (1994) 200–212.
[22] R. Li, T. Tang, P.W. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, J. Comput. Phys. 170 (2001) 562–588.
[23] R. Li, T. Tang, P.-W. Zhang, A moving mesh finite element algorithm for singular problems in two and three space dimensions, J. Comput. Phys. 177 (2002) 365–393.
[24] C.T. Lin, E. Tadmor, High-resolution non-oscillatory central schemes for Hamilton–Jacobi equations, SIAM J. Sci. Comput. 21 (2000) 2163–2186.
[25] C.T. Lin, E. Tadmor, L^1-stability and error estimates for approximate Hamilton–Jacobi solutions, Numer. Math. 87 (2001) 701–735.
[26] S. Osher, J. Sethian, Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[27] S. Osher, C.W. Shu, High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations, SIAM J. Numer. Anal. 28 (1991) 907–922.
[28] B. Perthame, Uniqueness and error estimates in first order quasilinear conservation laws via the kinetic entropy defect measure, J. Math. Pure Appl. 77 (1998) 1055–1064.
[29] J. Sethian, Level Set Methods, Cambridge University Press, New York, 1996.
[30] 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.

[31] C.-W. Shu, S. Osher, Efficient implement of essentially non-oscillatory shock-wave schemes, II, J. Comput. Phys. 83 (1989) 32–78.
[32] E. Tadmor, T. Tang, Pointwise error estimates for scalar conservation laws with piecewise smooth solutions, SIAM J. Numer. Anal. 36 (1999) 1739–1758.
[33] H.Z. Tang, T. Tang, Moving mesh methods for one- and two-dimensional hyperbolic conservation laws, SIAM J. Numer. Anal. 41 (2003) 487–515.
[34] T. Tang, Z.H. Teng, The sharpness of Kuznetsov's $\mathcal{O}(\sqrt{\Delta x})$ L^1-error estimate for monotone difference schemes, Math. Comp. (1995) 581–589.
[35] B. van Leer, Towards the ultimate conservative difference scheme V: a second-order sequel to Godunov's method, J. Comput. Phys. 32 (1979) 101–136.
[36] A. Winslow, Numerical solution of the quasi-linear Poisson equation, J. Comput. Phys. 1 (1967) 149–172.
[37] Y.-T. Zhang, C.-W. Shu, High order WENO schemes for Hamilton–Jacobi equations on triangular meshes, SIAM J. Sci. Comp. 24 (2003) 1005–1030.