Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 200 (2004) 347–367
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Moving mesh methods with locally varying time steps

Zhijun Tan [a], Zhengru Zhang [a], Yunqing Huang [b], Tao Tang [a,c,*]

[a] *Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong, China*
[b] *Department of Mathematics and Institute for Computational and Applied Mathematics, Xiangtan University, Xiangtan, Hunan 411105, China*
[c] *Institute of Computational Mathematics, The Chinese Academy of Sciences, Beijing 100080, China*

Received 20 May 2003; received in revised form 30 March 2004; accepted 3 April 2004
Available online 6 July 2004

Abstract

The time steps associated with moving mesh methods are proportional to the smallest mesh size in space and as a result they are very small at each time level. For some practical problems, the physical phenomena develop dynamically singular or nearly singular solutions in fairly localized regions, and therefore the smallest time step at each time level occurs only in these localized regions. In this work, we will develop a local time stepping algorithm for the moving mesh methods. The principal idea will be demonstrated by investigating the nonlinear hyperbolic conservation laws. Numerical experiments are carried out to demonstrate the efficiency and robustness of the proposed methods.

AMS: 35L65; 65M12; 65M30

Keywords: Moving mesh method; Local time stepping; Finite volume method; Hyperbolic conservation laws

1. Introduction

In this work, we shall discuss the class of adaptive grid methods often called moving mesh methods (or dynamic methods – in contrast to the static methods) for solving time dependent PDEs. These methods involve the solution of the underlying PDE for the physical problem in conjunction with a mesh movement for the mesh itself. The moving mesh methods have important applications in 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. In the past two decades, there have been many efforts in developing efficient moving mesh algorithms, see, e.g., [1,3,7,16,20,28].

* Corresponding author.
E-mail addresses: zjtan@math.hkbu.edu.hk (Z. Tan), zrzhang@math.hkbu.edu.hk (Z. Zhang), huangyq@xtu.edu.cn (Y. Huang), ttang@math.hkbu.edu.hk, ttang@lsec.cc.ac.cn (T. Tang).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2004.04.007

It is well known that the time steps associated with the moving mesh methods are proportional to the smallest mesh size in space. Since the moving mesh method is useful for problems whose solutions are singular in fairly localized regions, it reduces the allowable time step only in small part of the solution domain. It is then natural to use locally varying time steps to enhance the efficiency of the moving mesh methods. We will design an efficient local time stepping scheme for the nonlinear hyperbolic conservation laws and the convection-dominated problems.

Local time stepping for one-dimensional conservation laws was first proposed by Osher and Sanders [21]. Their schemes allow each element to take either an entire step or some fixed number of smaller steps. Another approach developed by Berger and Oliger [6] involves automatically taking smaller time steps where the mesh is refined. In their approach refined grids are laid over regions of the coarse mesh so that smaller time steps are taken on the refined mesh and larger time steps on the coarse mesh. Information is then passed between the grids by means of injection and interpolation. Flaherty et al. [11] have developed a parallel, adaptive discontinuous Galerkin method with a local forward Euler scheme which relies on interpolating values in time at interfaces between time steps of different sizes. Dawson and Kirby [10] developed upwind methods for solving conservation laws, which allow local time refinement to be coupled with local spatial refinement.

Note that most of the local time stepping techniques have been developed for fixed mesh structure. In this work, we will combine the local time refinement techniques with the moving mesh methods. At the regions where the underlying PDE has large solution gradients, it is expected that the spatial grid sizes generated by a moving mesh method will be very small and therefore local time steps should be used in these regions. Special attention will be taken so that the overall scheme is conservative. To demonstrate the principal idea of local stepping techniques, we will apply the moving mesh methods for the hyperbolic conservation laws. There has been some success in solving hyperbolic problems on adaptive spatial meshes, starting with Harten and Hyman [13] who extended a Godunov scheme to handle moving grids in one space dimension. Another static refinement strategy that has been proven very successful is the adaptive mesh refinement method of Berger and LeVeque [5]. In contrast with static refinement techniques is the moving mesh methods that move mesh points to where they are most needed while keeping the total number of grid points unchanged. In recent years, several moving mesh methods for hyperbolic problems have been proposed in the literature, including Azarenok et al. [1,2], Liu et al. [19], and Stockie et al. [23].

The paper is organized as follows. In Section 2, we describe the moving mesh methods and the local time stepping techniques for 1D problems. Numerical experiments are also provided to demonstrate the accuracy and efficiency of the local time stepping approaches. The extension of the 1D algorithms will be presented in Section 3, and some concluding remarks will be made in the final section.

2. 1D moving mesh methods with local time stepping

We consider the nonlinear conservation laws

$$u_t + f(u)_x = 0 \tag{2.1}$$

to demonstrate the principal idea of local stepping techniques; the strategies developed in this section can be easily extended to the convection-dominated equation:

$$u_t + f(u)_x = \epsilon(\sigma(u)u_x)_x, \tag{2.2}$$

where $0 < \epsilon \ll 1$ is a (small) viscosity coefficient, $\sigma > 0$. The moving mesh method to be used is based on two independent parts: a PDE evolution and a mesh-redistribution. At the time level $t = t_n$, we first advance the solution one time step based on an appropriate numerical scheme, which gives an approximation for u at $t = t_{n+1}$ on the mesh $\{x_j^n\}$. Then a grid restructuring procedure is carried out to obtain a new mesh $\{x_j^{n+1}\}$, which consists of solving the mesh redistributing equation (a generalized Laplacian equation) and inter-

polating the approximate solutions on the resulting new grid. A detailed solution flowchart was presented in [17,27].

The evolution for (2.1) will be based on the MUSCL-type finite volume methods. We partition the real line into intervals $I^n_{i+\frac{1}{2}} = [x^n_i, x^n_{i+1}]$. Denote $x^n_{i+\frac{1}{2}}$ and $u^n_{i+\frac{1}{2}}$ the center of $I^n_{i+\frac{1}{2}}$ and an approximation to the integral average of u over $I^n_{i+\frac{1}{2}}$ respectively, namely

$$x^n_{i+\frac{1}{2}} = \frac{1}{2}(x^n_i + x^n_{i+1}), \qquad u^n_{i+\frac{1}{2}} = \frac{1}{\left|I^n_{i+\frac{1}{2}}\right|}\int_{I^n_{i+\frac{1}{2}}} u(x,t_n)\mathrm{d}x.$$

Integrating (2.1) over the control volume $I^n_{i+\frac{1}{2}} \times [t_n, t_{n+1}]$ leads to the following finite volume method:

$$u^{n+1}_{i+\frac{1}{2}} = u^n_{i+\frac{1}{2}} - \frac{\Delta t_n}{\left|I^n_{i+\frac{1}{2}}\right|}(\tilde{f}^n_{i+1} - \tilde{f}^n_i), \tag{2.3}$$

where $\Delta t_n = t_{n+1} - t_n$, $\tilde{f}^n_i$ is some appropriate numerical flux. It is assumed that the numerical flux satisfies

$$\tilde{f}^n_i = \tilde{f}(u^{n,-}_i, u^{n,+}_i), \qquad \tilde{f}(u,u) = f(u),$$

where $u^{n,\pm}_i$ are the right and left states at an interface $x_{i+\frac{1}{2}}$, respectively. In our computations, we adopt the simple and inexpensive Lax–Friedrichs flux:

$$\tilde{f}(\alpha,\beta) = \frac{1}{2}\left[f(\alpha) + f(\beta) - \max_u |\, f_u \,| \cdot(\beta - \alpha)\right], \tag{2.4}$$

where the maximum is taken between α and β. The Godunov flux and Engquist–Osher flux can also be applied here. A linear reconstruction can be used to increase the solution accuracy:

$$u^{n,-}_i = u^n_{i-\frac{1}{2}} + s_{i-\frac{1}{2}}\left(x^n_i - x^n_{i-\frac{1}{2}}\right), \qquad u^{n,+}_i = u^n_{i+\frac{1}{2}} + s_{i+\frac{1}{2}}\left(x^n_i - x^n_{i+\frac{1}{2}}\right), \tag{2.5}$$

where $s_{i+\frac{1}{2}}$ is an approximation of the slope u_x at $x^n_{i+\frac{1}{2}}$

$$s^+_{i+\frac{1}{2}} = \frac{u^n_{i+\frac{3}{2}} - u^n_{i+\frac{1}{2}}}{x^n_{i+\frac{3}{2}} - x^n_{i+\frac{1}{2}}}, \quad s^-_{i+\frac{1}{2}} = \frac{u^n_{i+\frac{1}{2}} - u^n_{i-\frac{1}{2}}}{x^n_{i+\frac{1}{2}} - x^n_{i-\frac{1}{2}}}, \quad s_{i+\frac{1}{2}} = \left(\mathrm{sign}\left(s^-_{i+\frac{1}{2}}\right) + \mathrm{sign}\left(s^+_{i+\frac{1}{2}}\right)\right)\frac{\left|s^-_{i+\frac{1}{2}} \cdot s^+_{i+\frac{1}{2}}\right|}{\left|s^-_{i+\frac{1}{2}}\right| + \left|s^+_{i+\frac{1}{2}}\right|}.$$

A system of semi-discretized difference equations is obtained from the fully discretized numerical scheme (2.3)–(2.5), which is solved by a 3-stage Runge–Kutta method proposed by Shu and Osher [22].

We next describe the mesh redistribution which is a standard scheme used by many authors, see, e.g., [9,17]. The mesh equation, based on the standard equidistribution principle, is

$$(\omega x_\xi)_\xi = 0, \quad \xi \in [0,1], \tag{2.6}$$

where the function ω is called monitor function which in general depends on the underlying solution and is an indicator of the degree of singularity. In practice, the mesh-equation (2.6) is solved using the following Gauss–Seidel type iteration:

$$\omega_{j+\frac{1}{2}}(x_{j+1} - \tilde{x}_j) - \omega_{j-\frac{1}{2}}(\tilde{x}_j - \tilde{x}_{j-1}) = 0. \tag{2.7}$$

After obtaining the new mesh $\{\tilde{x}_j\}$, the numerical approximation on the new grid points $\tilde{x}_{j+\frac{1}{2}} = (\tilde{x}_j + \tilde{x}_{j+1})/2$ is updated based on the information of $\{x_{j+\frac{1}{2}}, \tilde{x}_{j+\frac{1}{2}}, u_{j+\frac{1}{2}}\}$. In [24], a second-order conservative interpolation formula was introduced. For scalar conservation laws, it is proved that the interpolation does not increase the total variation, and as a result the resulting adaptive mesh solutions satisfy

several fundamental properties for the hyperbolic conservation laws. In this work, the same conservative interpolation formula will be employed:

$$\Delta\tilde{x}_{j+\frac{1}{2}}\tilde{u}_{j+\frac{1}{2}} = \Delta x_{j+\frac{1}{2}}u_{j+\frac{1}{2}} - ((cu)_{j+1} - (cu)_j), \tag{2.8}$$

where $\Delta\tilde{x}_{j+\frac{1}{2}} = \tilde{x}_{j+1} - \tilde{x}_j$, $c_j = x_j - \tilde{x}_j$ denotes the local speed of the mesh motion. The linear flux cu is approximated by a second-order scheme as proposed in [24]:

$$(\widehat{cu})_j = \frac{c_j}{2}(u_j^+ + u_j^-) - \frac{|c_j|}{2}(u_j^+ - u_j^-), \tag{2.9}$$

where u_j^+ and u_j^- are defined by (2.5).

2.1. 1D local time stepping

For simplicity and also without loss of generality, we assume the fine meshes occur in one subinterval $(x^n_{\mathsf{iH}}, x^n_{\mathsf{iT}+1})$, see Fig. 1, where $ihead = \mathsf{iH}$, $itail = \mathsf{iT}$. If there are finitely many such refined mesh regions, the extension is straightforward. It is natural to use a larger time step $\Delta t_n = t_{n+1} - t_n$ in the regions outside $(x^n_{\mathsf{iH}}, x^n_{\mathsf{iT}+1})$, namely in

$$\mathscr{D}_1 = \{j \mid j \leqslant \mathsf{iH} - 1\}, \quad \text{and} \quad \mathscr{D}_3 = \{j \mid j \geqslant \mathsf{iT} + 1\},$$

and smaller time steps $\Delta t^{\eta_l}, 0 \leqslant l < M$, in

$$\mathscr{D}_2 = \{j \mid \mathsf{iH} \leqslant j \leqslant \mathsf{iT}\}.$$

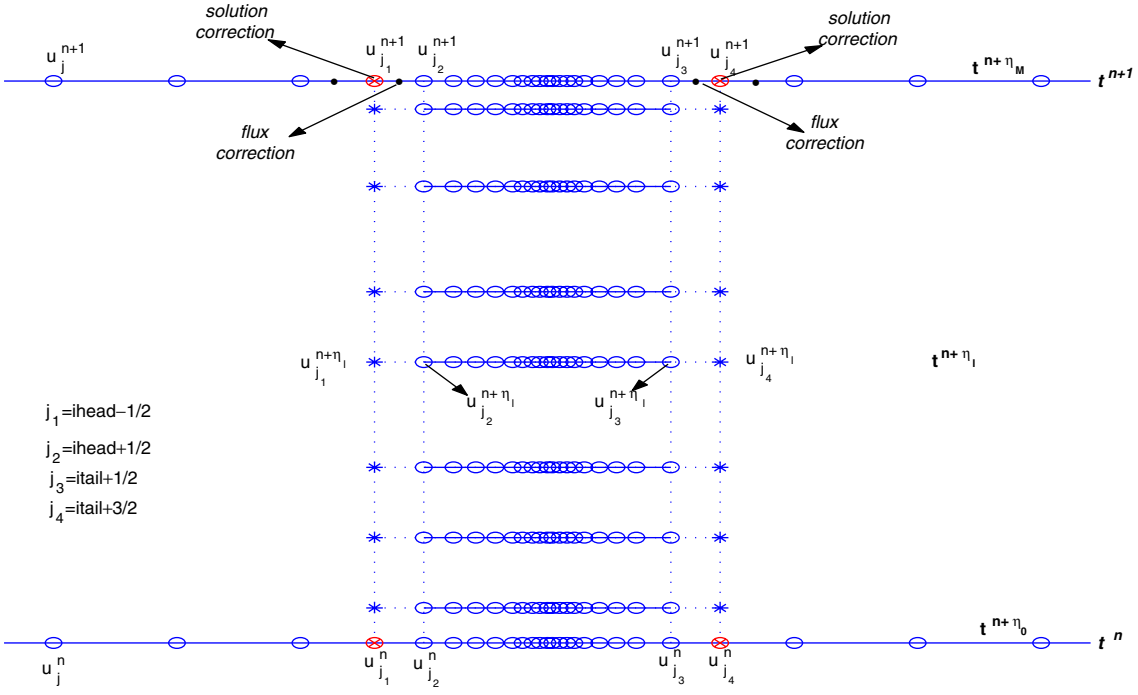

Fig. 1. Illustration of local time stepping from level n to level $n+1$.

We partition the time step $[t_n, t_{n+1})$ into $[t^{n+\eta_l}, t^{n+\eta_{l+1}}]$, $0 \leqslant l \leqslant M$ and let $\{\sigma_k\}_{k=1}^{M}$ be a sequence of positive numbers summing to unity. The numbers η_l are given as the sum of the σ_k, that is,

$$\eta_l = \sum_{k=1}^{l} \sigma_k, \quad 1 \leqslant l \leqslant M-1 \quad \text{and} \quad \eta_0 = 0, \quad \eta_M = 1.$$

Correspondingly, the sub-steps in this time interval are given by

$$t^{n+\eta_{l+1}} = t^{n+\eta_l} + \sigma_{l+1}\Delta t_n.$$

With these notations established, we will follow the scheme of Osher and Sanders [21] (see also [10]) to define a high resolution scheme in space. More precisely, on coarse grids:

$$u_{i+\frac{1}{2}}^{n+1} = u_{i+\frac{1}{2}}^{n} - \lambda_i^n(\tilde{f}_{i+1}^n - \tilde{f}_i^n) \qquad i \in \mathscr{D}_1 \quad \text{or} \quad i \in \mathscr{D}_3 \tag{2.10}$$

and on fine grids:

$$u_{i+\frac{1}{2}}^{n+\eta_l} = u_{i+\frac{1}{2}}^{n+\eta_{l-1}} - \lambda_i^n\sigma_l(\tilde{f}_{i+1}^{n+\eta_{l-1}} - \tilde{f}_i^{n+\eta_{l-1}}) \quad i \in \mathscr{D}_2 \tag{2.11}$$

for $1 \leqslant l \leqslant M$, where $\lambda_i^n = \Delta t_n / \left| I_{i+\frac{1}{2}}^n \right|$ is the ratio between the time step and the size of the local stencil. It follows from (2.11) that

$$u_{i+\frac{1}{2}}^{n+\eta_l} = u_{i+\frac{1}{2}}^{n} - \lambda_i^n \sum_{k=0}^{l-1} \sigma_{k+1}(\tilde{f}_{i+1}^{n+\eta_k} - \tilde{f}_i^{n+\eta_k}) \quad i \in \mathscr{D}_2, \tag{2.12}$$

which gives

$$u_{i+\frac{1}{2}}^{n+1} = u_{i+\frac{1}{2}}^{n} - \lambda_i^n \sum_{k=0}^{M-1} \sigma_{k+1}(\tilde{f}_{i+1}^{n+\eta_k} - \tilde{f}_i^{n+\eta_k}) \quad i \in \mathscr{D}_2. \tag{2.13}$$

On the interfaces, $u_{\mathsf{iH}-\frac{1}{2}}^{n+\eta_{l+1}}$ is obtained by linearly interpolating values at $\left(t_n, u_{\mathsf{iH}-\frac{1}{2}}^{n}\right)$ and $\left(t_{n+1}, u_{\mathsf{iH}-\frac{1}{2}}^{n+1}\right)$, and $u_{\mathsf{iT}+\frac{3}{2}}^{n+\eta_{l+1}}$ is obtained by interpolating values at $\left(t_n, u_{\mathsf{iT}+\frac{3}{2}}^{n}\right)$ and $\left(t_{n+1}, u_{\mathsf{iT}+\frac{3}{2}}^{n+1}\right)$. After completing the local time stepping in $\mathscr{D}_2$, we need to redefine the numerical fluxes at the coarse-fine grid interfaces in order to guarantee the mass conservation. The idea is to make the flux into (out of) the fine grid across a coarse cell boundary equal to the sum over the fine time steps of flux out of (into) the adjoining fine grid cell. This is done by requiring:

$$\Delta x_{i-\frac{1}{2}}^n u_{i-\frac{1}{2}}^{n+1} = \Delta x_{i-\frac{1}{2}}^n u_{i-\frac{1}{2}}^{n} - \Delta t_n \left(\sum_{l=0}^{M-1} \sigma_{l+1}\tilde{f}_i^{n+\eta_l} - \tilde{f}_{i-1}^n \right), \quad i = \mathsf{iH}, \tag{2.14}$$

$$\Delta x_{i+\frac{3}{2}}^n u_{i+\frac{3}{2}}^{n+1} = \Delta x_{i+\frac{3}{2}}^n u_{i+\frac{3}{2}}^{n} - \Delta t_n \left(\tilde{f}_{i+2}^n - \sum_{l=0}^{M-1} \sigma_{l+1}\tilde{f}_{i+1}^{n+\eta_l} \right), \quad i = \mathsf{iT}. \tag{2.15}$$

The above two equations can also be written as the following equivalent forms:

$$\Delta x_{i-\frac{1}{2}}^n u_{i-\frac{1}{2}}^{n+1} = \Delta x_{i-\frac{1}{2}}^n u_{i-\frac{1}{2}}^{n} - \Delta t_n(\tilde{f}_i^n - \tilde{f}_{i-1}^n) - \Delta t_n\,\delta\mathbf{F}_i^n, \quad i = \mathsf{iH}, \tag{2.16}$$

$$\Delta x_{i+\frac{3}{2}}^n u_{i+\frac{3}{2}}^{n+1} = \Delta x_{i+\frac{3}{2}}^n u_{i+\frac{3}{2}}^{n} - \Delta t_n(\tilde{f}_{i+2}^n - \tilde{f}_{i+1}^n) + \Delta t_n\,\delta\mathbf{F}_{i+1}^n, \quad i = \mathsf{iT}, \tag{2.17}$$

where

$$\delta \mathbf{F}_i^n = -\tilde{f}_i^n + \sum_{l=0}^{M-1} \sigma_{l+1} \tilde{f}_i^{n+\eta_l}. \tag{2.18}$$

We now summarize the moving mesh scheme with local time stepping in the following algorithm.

Algorithm 1.

- **Step 1:** Given an initial partition $x_j^{[0]} := x_j$ of the physical domain Ω_p and a uniform (fixed) partition of the logical domain Ω_c, and compute grid values $u_{j+\frac{1}{2}}^{[0]}$ based on the cell average of initial data $u(x,0)$.
- **Step 2:** For $v \geqslant 0$, move grid $\{x_j^{[v]}\}$ to $\{x_j^{[v+1]}\}$ based on scheme (2.7) and interpolate $\{u_{j+\frac{1}{2}}^{[v+1]}\}$ on the new grid based on (2.8). Repeat the updating procedure until $\|x^{[v+1]} - x^{[v]}\|$ is sufficiently small.
- **Step 3:** Find the interface points (i.e. x_{iH}^n and $x_{\mathrm{iT}+1}^n$) at the time level t_n. If they exist, then do the following:
  (a) Determine the global Δt_n based on the CFL condition:

$$\Delta t_n = \lambda \min_{\substack{j\in\mathscr{D}_1 \\ j\in\mathscr{D}_3}} \left(\frac{x_{j+1}^n - x_j^n}{\left| f'\left(u_{j+\frac{1}{2}}^n\right)\right|} \right),$$

  where $0 < \lambda \leqslant 1$ is an CFL constant, and set $t_{n+1} = t_n + \Delta t_n$.
  (b) Evolve the values in the coarse grid region:

$$u_{j+\frac{1}{2}}^{n+1} = u_{j+\frac{1}{2}}^n - \frac{\Delta t_n}{\left|I_{j+\frac{1}{2}}^n\right|}(\tilde{f}_{j+1}^n - \tilde{f}_j^n), \quad j \in \mathscr{D}_1 \cup \mathscr{D}_3.$$

  (c) Set $l = 0, \eta_0 = 0$.
  (i) Determine the local Δt^{η_l} based on the CFL condition:

$$\Delta t^{\eta_l} = \lambda \min_{j\in\mathscr{D}_2} \left(\frac{x_{j+1}^n - x_j^n}{\left| f'\left(u_{j+\frac{1}{2}}^{n+\eta_l}\right)\right|} \right),$$

  and set $t^{n+\eta_{l+1}} = t^{n+\eta_l} + \Delta t^{\eta_l}$. If $\Delta t^{\eta_l} \geqslant t_{n+1} - t^{n+\eta_l}$, then set $\Delta t^{\eta_l} = t_{n+1} - t^{n+\eta_l}$.
  (ii) Evolve the solution with the time step Δt^{η_l}:

$$u_{j+\frac{1}{2}}^{n+\eta_{l+1}} = u_{j+\frac{1}{2}}^{n+\eta_l} - \frac{\Delta t^{\eta_l}}{\left|I_{j+\frac{1}{2}}^n\right|}\left(\tilde{f}_{j+1}^{n+\eta_l} - \tilde{f}_j^{n+\eta_l}\right), \quad j \in \mathscr{D}_2.$$

  (iii) Obtain the interface values using a linear interpolation.
  (iv) Set $u_{j+\frac{1}{2}}^{[0]} := u_{j+\frac{1}{2}}^{n+\eta_l}$ and $x_j^{[0]} := x_j^{[v+1]}$ for $j \in \mathscr{D}_2$. If $t^{n+\eta_{l+1}} \geqslant t_{n+1}$, then set $\eta_{l+1} = 1$, make the interface correction using (2.16) and (2.17), and goto **Step 4** below.
  (v) goto step (i) above.
  If there is no interface point (i.e. x_{iH}^n and $x_{\mathrm{iT}+1}^n$) at the time level t_n, then a standard global time step is used to evolve the PDE to $t = t_{n+1}$.
- **Step 4:** If $t_{n+1} \leqslant T$, then $u_{j+\frac{1}{2}}^{[0]} := u_{j+\frac{1}{2}}^{n+1}$, $x_j^{[0]} := x_j^{[v+1]}$, and goto **Step 2**.

We point out that an alternative way for solving the underlying PDE is to transfer the physical coordinates to a logical domain. For the conservation law (2.1), a mapping $x = x(\xi)$ will transfer (2.1) into the following equation:

$$u_t + \frac{1}{x_\xi} f(u)_\xi = 0, \quad 0 < \xi < 1. \tag{2.19}$$

This equation can be solved by a cell-center finite volume scheme similar to the one used for (2.1). In 1D, solving (2.19) with a moving mesh method does not have much advantage than solving (2.1). However, in multi-dimensions there are some advantages for solving the transformed equations, for example a regular solution domain in multi-dimensions allows the use of uniform mesh and as a consequence fast solution solvers (such as multi-grid methods) or domain decomposition methods may be employed easily, see, e.g., [8]. For these reasons, we will solve the transformed equations for two-dimensional problems in the next section.

We close this subsection by listing the following theoretical results which are essential for solving conservation laws.

Theorem 2.1. *Assume that the conservation laws (2.1) is defined in the real line and its initial data has compact support. Then the adaptive mesh solution obtained by using Algorithm 1 is conservative in the following sense:*

$$\sum_j \Delta x_j^{n+1} u_j^{n+1} = \sum_j \Delta x_j^n u_j^n. \tag{2.20}$$

Moreover, the numerical solution is a weak solution for the conservation law (2.1).

The first conclusion above is a direct consequence of the fact that the in and out fluxes across each cell interface cancel by (2.10) in the coarse and (2.13) in the fine mesh regions, and by Eqs. (2.14) and (2.15) for the boundary of the coarse and fine meshes. The convergence to the weak solution is based on the same observations together with the use of the conservative interpolation (2.8) in the mesh movement part. The detail proof of the above theorem can be found in [26].

2.2. Numerical experiment for 1D problems

Example 2.1. [Burgers' equation] Consider the inviscid Burgers' equation

$$u_t + \left(\frac{u^2}{2}\right)_x = 0, \quad 0 \leqslant x \leqslant 2\pi \tag{2.21}$$

subject to the 2π-periodic boundary condition and the initial data

$$u(x,0) = 0.5 + \sin(x), \quad x \in (0, 2\pi). \tag{2.22}$$

Adaptive solution is obtained for the above problem up to $t = 4$ using locally varying time steps. For comparison, the solutions using non-local time stepping are also computed. In the computation, both the global CFL constant and local CFL constant used are chosen as 0.6. The monitor function used is

$$\omega = \sqrt{1 + 0.2|u_\xi|^2}. \tag{2.23}$$

It is known that a good choice of the monitor function is very important in obtaining satisfactory adaptation effect. In many cases the monitor functions involve some user-defined parameters which have to be obtained by doing several experiments. This is the case for the choice of the constant 0.2 in (2.23) as well as for the monitor constants in other numerical examples of this paper. Obviously, some theoretical study on how to choose the monitor constants (or the general forms of the monitor functions) seems very useful. On

Table 1
Example 2.1: the l^1-error and CPU-time

	$t = 1$		$t = 2$	
	CPU time (s)	l^1-error	CPU time (s)	l^1-error
Non-local	0.01	3.86e−2	0.04	2.19e−2
Local	0.01	4.08e−2	0.02	1.32e−2
	$t = 3$		$t = 4$	
	CPU time (s)	l^1-error	CPU time (s)	l^1-error
Non-local	0.07	1.88e−2	0.1	1.59e−2
Local	0.03	1.29e−2	0.04	2.55e−2

this aspect, there have seen some efforts in this direction recently. For example, Huang and Sun [14] proposed two types of monitor functions based on the asymptotic estimates of interpolation errors. Their work seems useful in developing parameter-free monitor functions.

For evolving Eq. (2.21) a second-order MUSCL finite volume scheme (with the Lax–Friedrichs flux) and a 3rd order Runge–Kutta method are used. In Table 1, the l^1-error and CPU-time are listed. The total number of grid points is 50. As we expected, the computer time used for the computations with and without using the local time-stepping is almost the same before $t = 1$ (i.e., when the singularity is formed), and a difference of about ratio 2 is observed after $t = 1$. The corresponding l^1-errors are not increased until $t = 3$, and are increased at about $t = 4$. In Fig. 2, comparisons are made for the mesh trajectory ($0 \leqslant t \leqslant 3$) and the numerical solution ($t = 3$) obtained by using the local and non-local time-stepping techniques. These results are in good agreement.

Example 2.2. Consider the one-dimensional Buckley–Leverett equation:

$$u_t + f(u)_x = \epsilon(\sigma(u)u_x)_x, \quad 0 < x < 1, \tag{2.24}$$

where the flux function has an s-shaped form:

$$f(u) = \frac{u^2}{u^2 + (1-u)^2} \tag{2.25}$$

and the diffusion coefficient $\sigma(u)$ vanishes at $u = 0$ and 1:

$$\sigma(u) = 4u(1-u).$$

The initial function is

$$u(x,0) = \begin{cases} 1 - 3x & 0 \leqslant x \leqslant \frac{1}{3}, \\ 0 & \frac{1}{3} < x \geqslant 1 \end{cases} \tag{2.26}$$

and the boundary value of $u(0,t) = 1$ is kept fixed.

This is a prototype model for oil reservoir simulations (two-phase flow), where the flux is nonlinear. Typically, $\sigma(u)$ vanishes at some values of u, and (2.24) is a degenerate parabolic equation. In our computation, the diffusion coefficient ϵ is taken as 0.01.

In contrast with the Burgers' equation, the flux function for this problem is non-convex, which introduces some extra numerical difficulties and so serves as a good test problem [6]. Both the global and local CFL constants are taken as 0.4. The monitor function used is

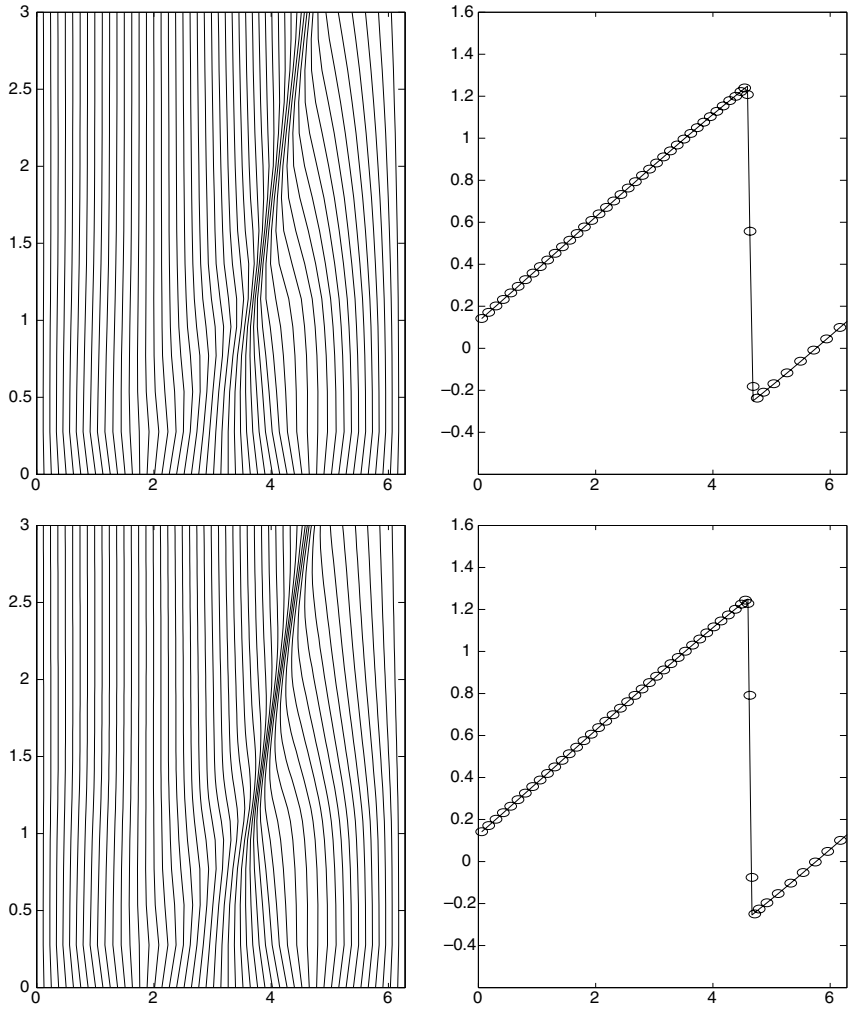

Fig. 2. Example 2.1: mesh trajectory for $0 \leqslant t \leqslant 3$ and numerical solution at $t = 3$. Top is obtained by using the locally varying time steps, and the bottom is with non-local time stepping.

$$\omega = \sqrt{1 + 50u_x^2}. \tag{2.27}$$

In the solution domain $[0, 1]$, 30 grid points are used. In Table 2, the l^1-error and CPU-time at different time levels are presented. A speed-up of about 1.65–2.1 times with the local time stepping is observed. It is also noted that the l^1-errors are smaller when the local time stepping is used. This accuracy improvement may be due to the use of smaller time steps in the steep layer regions, which uses more detailed information from the level t_n so that the approximation at the level t_{n+1} is more accurate. In Fig. 3, the mesh trajectory up to $t = 0.5$ and the numerical solution at $t = 0.5$ are plotted. The agreement between the moving mesh solutions with and without the local stepping techniques is good.

We close this section by discussing the choice of the monitor functions. It is seen that the forms of the monitors (2.23) and (2.27) are different, with ξ-derivative variable for Example 2.1, and x-derivative variable for Example 2.2. In our experience, for solutions with discontinuity or extremely large gradients then

Table 2
Example 2.2: the l^1-error and CPU-time

	$t = 0.2$		$t = 0.3$	
	CPU time (s)	l^1-error	CPU time (s)	l^1-error
Non-local	0.33	8.21e−3	0.67	7.98e−3
Local	0.20	1.12e−3	0.36	1.44e−3
	$t = 0.4$		$t = 0.5$	
	CPU time (s)	l^1-error	CPU time (s)	l^1-error
Non-local	1.04	9.96e−3	1.43	9.56e−3
Local	0.52	1.58e−3	0.68	1.18e−3

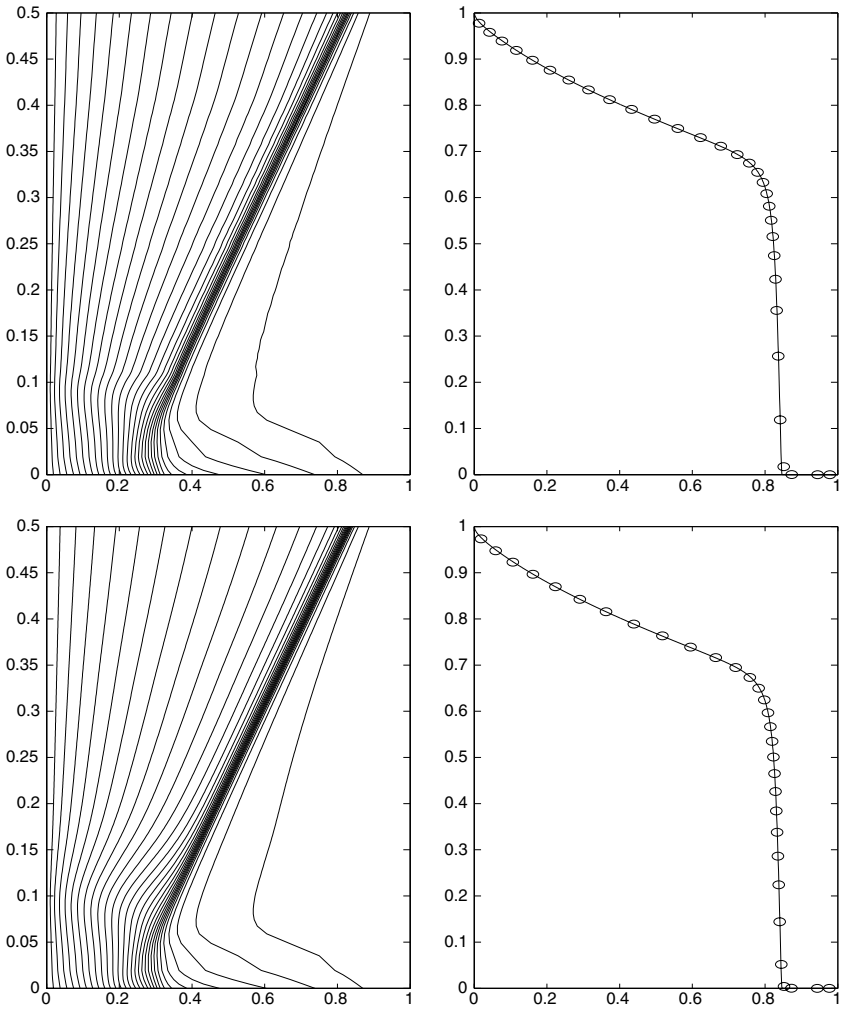

Fig. 3. Example 2.2: left is the mesh trajectory for $0 \leqslant t \leqslant 0.5$ and the right is the numerical solution at $t = 0.5$. Top is obtained by using the locally varying time steps, and the bottom is with non-local time stepping.

an ξ-derivative is preferred. This will avoid clustering too many points in the neighborhood of the discontinuities. In other words, this choice can better balance the number of points inside and outside a steep internal layer. In the case that the solution is smooth and that the solution gradient is not extremely large then an x-derivative can be used. In general (and in particularly in multi-dimensions), a monitor with ξ-derivative seems more appropriate (see, e.g., [8,24,25]). Moreover, some regularization factors (0.2 in (2.23) and 50 in (2.27)) may be used in the monitor functions, which allow us to reduce (or increase) the magnitude of the monitor function in situations where the derivatives are very large (or not very large). This is to avoid over-resolution of steep layers. There have existed several works to discuss the choice of the monitor functions, see, e.g., [1,4,23].

3. 2D moving mesh methods with local time stepping

To demonstrate the principal idea of local stepping techniques we again consider the 2D nonlinear conservation law:

$$u_t + f(u)_x + g(u)_y = 0, \quad (x,y) \in \Omega_{\mathrm{p}}, \tag{3.1}$$

where Ω_{p} is the physical domain. As mentioned at the end of Section 2.1, there are some advantages in solving (3.1) in a logical domain. In our computations, we regard Ω_{p} as the image of a computational (logical) domain Ω_{c} under some suitable mapping

$$x = x(\xi,\eta), \quad y = y(\xi,\eta) \quad \text{and} \quad \xi = \xi(x,y), \quad \eta = \eta(x,y), \tag{3.2}$$

where (x,y) and (ξ,η) are the physical and computational coordinates, respectively. The corresponding transformed equation for (3.1) is of the form

$$u_t + \frac{1}{J}F(u)_\xi + \frac{1}{J}G(u)_\eta = 0, \quad (\xi,\eta) \in \Omega_{\mathrm{c}}, \tag{3.3}$$

where $J = x_\xi y_\eta - x_\eta y_\xi$ is the Jacobian of the coordinate transformation, and

$$F(u) = y_\eta f(u) - x_\eta g(u), \quad G(u) = x_\xi g(u) - y_\xi f(u).$$

It is noted that the transformation (3.2) is time-independent and therefore the time derivative in (3.1) is not transformed to a moving frame. However, the meshes in the physical domain are indeed time-dependent, which is realized by using the solution-dependent monitor functions. Using a cell-center finite volume method to discretize (3.3) gives

$$\bar{u}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}} = \bar{u}^{n}_{j+\frac{1}{2},k+\frac{1}{2}} - \lambda^n_{j,k}\left(\tilde{F}^n_{j+1,k+\frac{1}{2}} - \tilde{F}^n_{j,k+\frac{1}{2}}\right) - \mu^n_{j,k}\left(\tilde{G}^n_{j+\frac{1}{2},k+1} - \tilde{G}^n_{j+\frac{1}{2},k}\right), \tag{3.4}$$

where

$$\lambda^n_{j,k} = \frac{\Delta t_n}{\Delta\xi J_{j+\frac{1}{2},k+\frac{1}{2}}}, \qquad \mu^n_{j,k} = \frac{\Delta t_n}{\Delta\eta J_{j+\frac{1}{2},k+\frac{1}{2}}},$$

$$\bar{u}^n_{j+\frac{1}{2},k+\frac{1}{2}} = \frac{1}{\Delta\xi\Delta\eta}\int_{A_{j+\frac{1}{2},k+\frac{1}{2}}} u(\xi,\eta,t_n)\,\mathrm{d}\xi\,\mathrm{d}\eta$$

and $A_{j+\frac{1}{2},k+\frac{1}{2}}$ is the control cell $[\xi_j,\xi_{j+1}]\times[\eta_k,\eta_{k+1}]$.

The local time stepping for an 2D moving mesh method uses a similar procedure as described in Algorithm 1. At time level $t = t_n$, numerical solution on coarse mesh is obtained using (3.4) with a single larger

time step; and solution on finer mesh is obtained using smaller time step. More precisely, in the finer mesh region, for each $l = 1, \ldots, M$,

$$\bar{u}^{n+\eta_l}_{j+\frac{1}{2},k+\frac{1}{2}} = \bar{u}^{n+\eta_{l-1}}_{j+\frac{1}{2},k+\frac{1}{2}} - \sigma_l \left[\lambda^n_{j,k} \left(\tilde{F}^{n+\eta_{l-1}}_{j+1,k+\frac{1}{2}} - \tilde{F}^{n+\eta_{l-1}}_{j,k+\frac{1}{2}} \right) + \mu^n_{j,k} \left(\tilde{G}^{n+\eta_{l-1}}_{j+\frac{1}{2},k+1} - \tilde{G}^{n+\eta_{l-1}}_{j+\frac{1}{2},k} \right) \right], \tag{3.5}$$

which gives

$$\bar{u}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}} = \bar{u}^{n}_{j+\frac{1}{2},k+\frac{1}{2}} - \sum_{l=0}^{M-1} \sigma_{l+1} \left[\lambda^n_{j,k} \left(\tilde{F}^{n+\eta_l}_{j+1,k+\frac{1}{2}} - \tilde{F}^{n+\eta_l}_{j,k+\frac{1}{2}} \right) + \mu^n_{j,k} \left(\tilde{G}^{n+\eta_l}_{j+\frac{1}{2},k+1} - \tilde{G}^{n+\eta_l}_{j+\frac{1}{2},k} \right) \right]. \tag{3.6}$$

At the interface where large time-step region and small time-step region are adjacent, unknown solutions on the boundary are again obtained using linear interpolation. After M fine-time steps have been computed, we redefine the numerical fluxes at the coarse/fine grid interfaces using the method similar to that in the 1D case. For example, the numerical approximation for the cell east to the interface of $\left(j + \frac{1}{2}, k + \frac{1}{2}\right)$ (see Fig. 4) is given by

$$\bar{u}^{n+1}_{j+\frac{3}{2},k+\frac{1}{2}} := \bar{u}^{n+1}_{j+\frac{3}{2},k+\frac{1}{2}} + \lambda^n_{j+1,k}\, \delta \mathbf{F}^n_{j+1,k+\frac{1}{2}}, \tag{3.7}$$

where

$$\delta \mathbf{F}^n_{j+1,k+\frac{1}{2}} = -\tilde{F}^n_{j+1,k+\frac{1}{2}} + \sum_{l=0}^{M-1} \sigma_{l+1} \tilde{F}^{n+\eta_l}_{j+1,k+\frac{1}{2}}.$$

If the cell to north of $\left(j + \frac{1}{2}, k + \frac{1}{2}\right)$ is not refined, then it is redefined as

$$\bar{u}^{n+1}_{j+\frac{1}{2},k+\frac{3}{2}} := \bar{u}^{n+1}_{j+\frac{1}{2},k+\frac{3}{2}} + \mu^n_{j,k+1}\, \delta \mathbf{G}^n_{j+\frac{1}{2},k+1}, \tag{3.8}$$

where

$$\delta \mathbf{G}^n_{j+\frac{1}{2},k+1} = -\tilde{G}^n_{j+\frac{1}{2},k+1} + \sum_{l=0}^{M-1} \sigma_{l+1} \tilde{G}^{n+\eta_l}_{j+\frac{1}{2},k+1}.$$

We now describe the mesh redistribution in two space dimension. It is assumed that a fixed (square uniform) mesh is given on the computational domain. The mesh generation equation is of the form

$$\partial_\xi (G_1 \partial_\xi x) + \partial_\eta (G_1 \partial_\eta x) = 0,$$
$$\partial_\xi (G_2 \partial_\xi y) + \partial_\eta (G_2 \partial_\eta y) = 0,$$

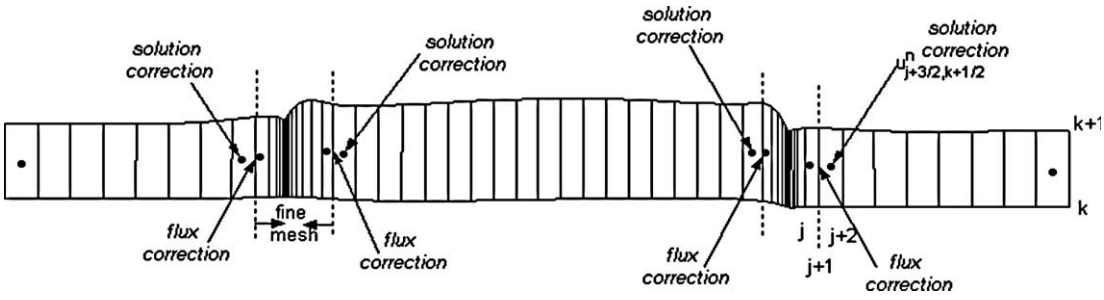

Fig. 4. Solution is corrected at a coarse grid/fine grid interface.

where G_1, G_2 are monitor functions. In our computations, a very simple monitor is chosen so that the mesh generation equation is of the form

$$\tilde{\nabla} \cdot (\omega \tilde{\nabla} x) = 0, \quad \tilde{\nabla} \cdot (\omega \tilde{\nabla} y) = 0, \tag{3.9}$$

where $\tilde{\nabla} = (\partial_\xi, \partial_\eta)^{\mathrm{T}}$. Moreover, the conservative interpolation proposed by Tang and Tang [24] is used to update the approximate solutions on the new grid:

$$\left|\tilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}\right| \tilde{u}_{j+\frac{1}{2},k+\frac{1}{2}} = \left|A_{j+\frac{1}{2},k+\frac{1}{2}}\right| u_{j+\frac{1}{2},k+\frac{1}{2}} - \left[(c^x u)_{j+1,k+\frac{1}{2}} - (c^x u)_{j,k+\frac{1}{2}}\right] - \left[(c^y u)_{j+\frac{1}{2},k+1} - (c^y u)_{j+\frac{1}{2},k}\right], \tag{3.10}$$

where $c^x_{j,k} = x_{j,k} - \tilde{x}_{j,k}$, $c^y_{j,k} = y_{j,k} - \tilde{y}_{j,k}$.

Below we will provide some numerical examples to demonstrate the performance of the local time stepping method in 2D.

Example 3.1. [2D inviscid Burgers' problem]. Our first 2D problem is concerned with the two-dimensional inviscid Burgers' equation

$$u_t + \left(\frac{u^2}{2}\right)_x + \left(\frac{u^2}{2}\right)_y = 0, \quad 0 \leqslant x, y \leqslant 1. \tag{3.11}$$

We take as an initial condition a function consisting of two cone shapes, one with height 1 and the other with height -1. The initial condition is displayed in Fig. 5.

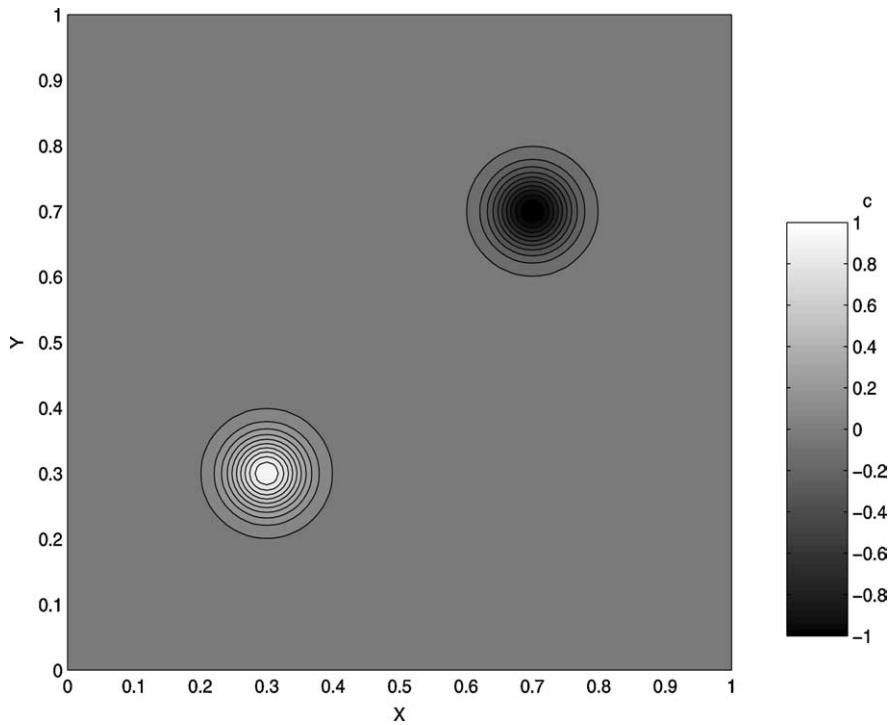

Fig. 5. Initial condition for Example 3.1.

In the Burgers' problem, larger values of the solution give rise to larger velocities. Consequently, the centers of the cones are advected along at a higher rate than the edges. Moreover, the positive cone has positive velocity and the negative cone has negative velocity, and as a result the cones approach each other and collide in the center of the domain.

The numerical solutions are obtained using the moving mesh method with local time stepping techniques as described earlier. In our computations, 80×80 grid points are used and both the local and global CFL constants used are 0.4. The monitor function in (3.9) is of the form

$$\omega = \sqrt{1 + \alpha_1 \mid u \mid^2 + \alpha_2 \mid \tilde{\nabla} u \mid^2}. \tag{3.12}$$

The constants α_1 and α_2 are chosen as 6 and 1, respectively. It was pointed out in [25] that the inclusion of the $\mid u \mid$ term in the monitor function may increase the adaptation effect. Our computational results also suggest that the inclusion of the $\mid u \mid$ term enhances the efficiency of the moving mesh algorithm although the improvement is not significant. Therefore, we only use the $\mid u \mid$ term in this example.

Table 3 shows the CPU-time at different time levels. A speed-up of about 2.12–2.47 for the local time stepping scheme is observed. In Fig. 6, a comparison on the meshes and numerical solutions obtained using and without using the local time stepping is made. The agreement between the local time stepping results and the global time stepping results is very good.

Example 3.2. [2D viscous Burgers' problem]. Our second 2D problem is concerned with the two-dimensional viscous Burgers' equation

$$u_t + \left(\frac{u^2}{2}\right)_x + \left(\frac{u^2}{2}\right)_y = \epsilon \nabla^2 u, \quad 0 \leqslant x, y \leqslant 1. \tag{3.13}$$

The initial condition and Dirichlet boundary condition are chosen so that the exact solution to the underlying problem is given by

$$u(x, y; t) = \left(1 + \mathrm{e}^{(x+y-t)/2\epsilon}\right)^{-1}.$$

This solution describes a straight-line wave (u is constant along the line $x + y = c$) moving in the direction $\theta = 45°$. In our computation, we consider the case with a moderately small diffusion coefficient $\epsilon = 0.005$. It is noted that the smaller ϵ is, the more convection dominates, and the higher the concentration of mesh points required around the wave front. The monitor function used is (3.12) with $(\alpha_1, \alpha_2) = (0, 1)$. In this problem, large solution gradients will be developed to the boundaries in a later time. As a consequence, boundary point redistribution should be made in order to improve the quality of the adaptive mesh. This is done by solving some 1D moving mesh equations on boundaries, see, e.g., [17]. In our computations, both the global and local CFL constants are 0.4, and an 80×80 grid is used. Fig. 7 shows the adaptive meshes and the corresponding point-wise errors at $t = 1$, obtained by using the moving mesh methods with and without local time stepping. As expected, a quite large portion of the grid points is moved to the steep layer region. It is observed that the meshes obtained by using the two time-stepping approaches are almost the

Table 3
Example 3.1: CPU time comparison

	$t = 0.4$ (s)	$t = 0.6$ (s)	$t = 0.8$ (s)	$t = 1$ (s)
Non-local	17.7	22.2	25.1	27.2
Local	7.2	9.9	11.8	12.8

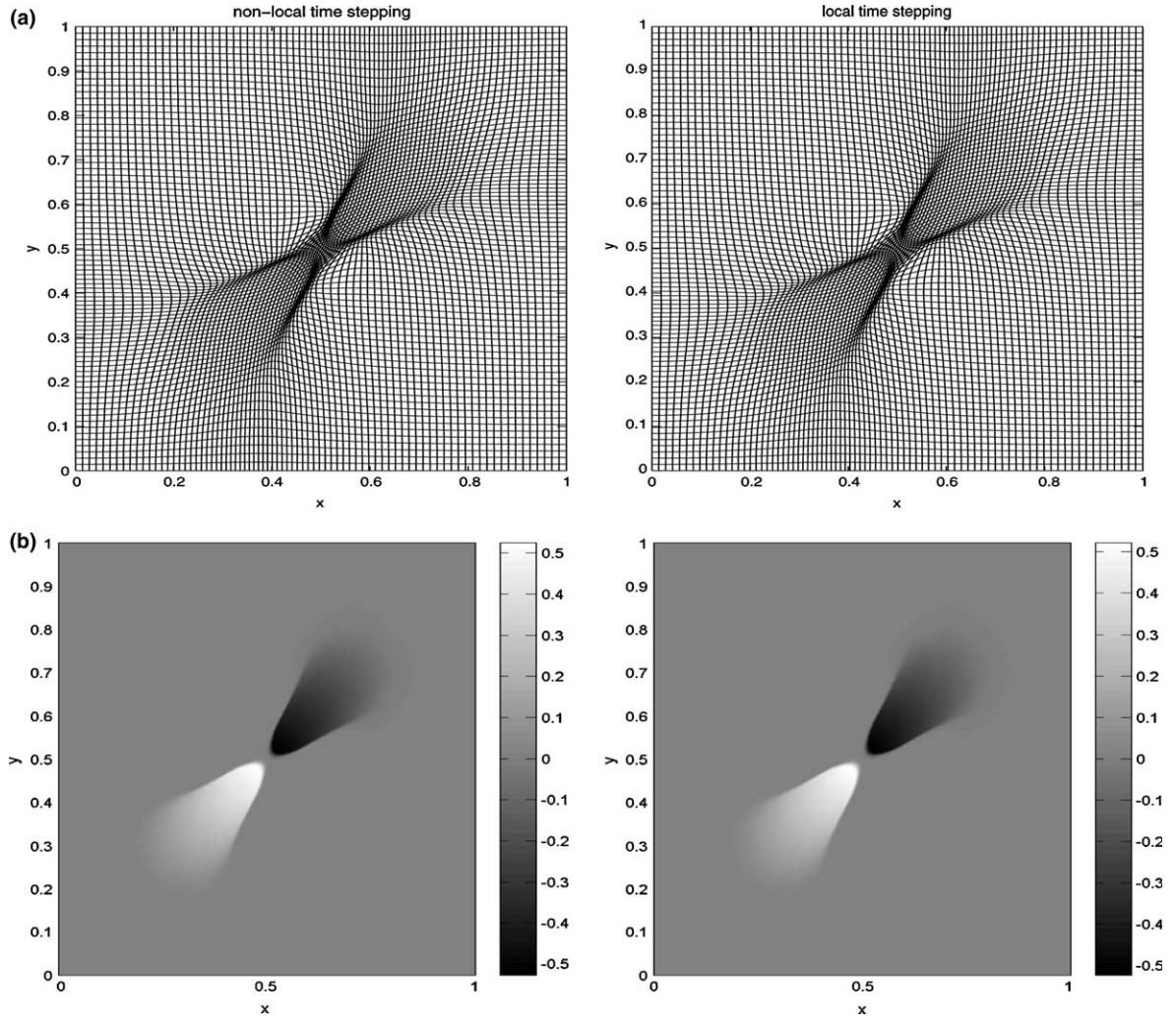

Fig. 6. Meshes and solutions for Example 3.1: left and right are for the non-local and local time stepping results, respectively. (a) mesh at $t = 0.4$, (b) solution at $t = 0.4$, (c) mesh at $t = 1$, and (d) solution at $t = 1$.

same, but the point-wise errors are quite different. It is also observed from Table 4 that with the local time stepping approach the resulting l^1-errors become smaller and there are about 2–4 times saving in computational cost.

It should be pointed out that if an explicit method is applied to a problem without steep layers then even a singularly perturbed convection-diffusion problem can be solved as the time-step restriction would be that of the CFL condition. However, if there is a thin internal layer, as the case in this example, then the mesh needs to be highly adapted to resolve such a layer, which will affect the choice of time-step restriction of an explicit method. Roughly speaking, the time-step restriction in this region should satisfy $\Delta t \sim \min\{\Delta t_{\mathrm{CFL}}, \Delta t_{\mathrm{vis}}\}$, where Δt_{CFL} is the standard CFL condition and Δt_{vis} is the viscous time step in the layer regions. The viscous time step is defined by $\Delta t_{\mathrm{vis}} \sim \Delta x^2/\epsilon$, where Δx is the mesh diameter in the layer region. For the viscous Burgers' equation the layer has width $\mathcal{O}(\epsilon)$, which implies that

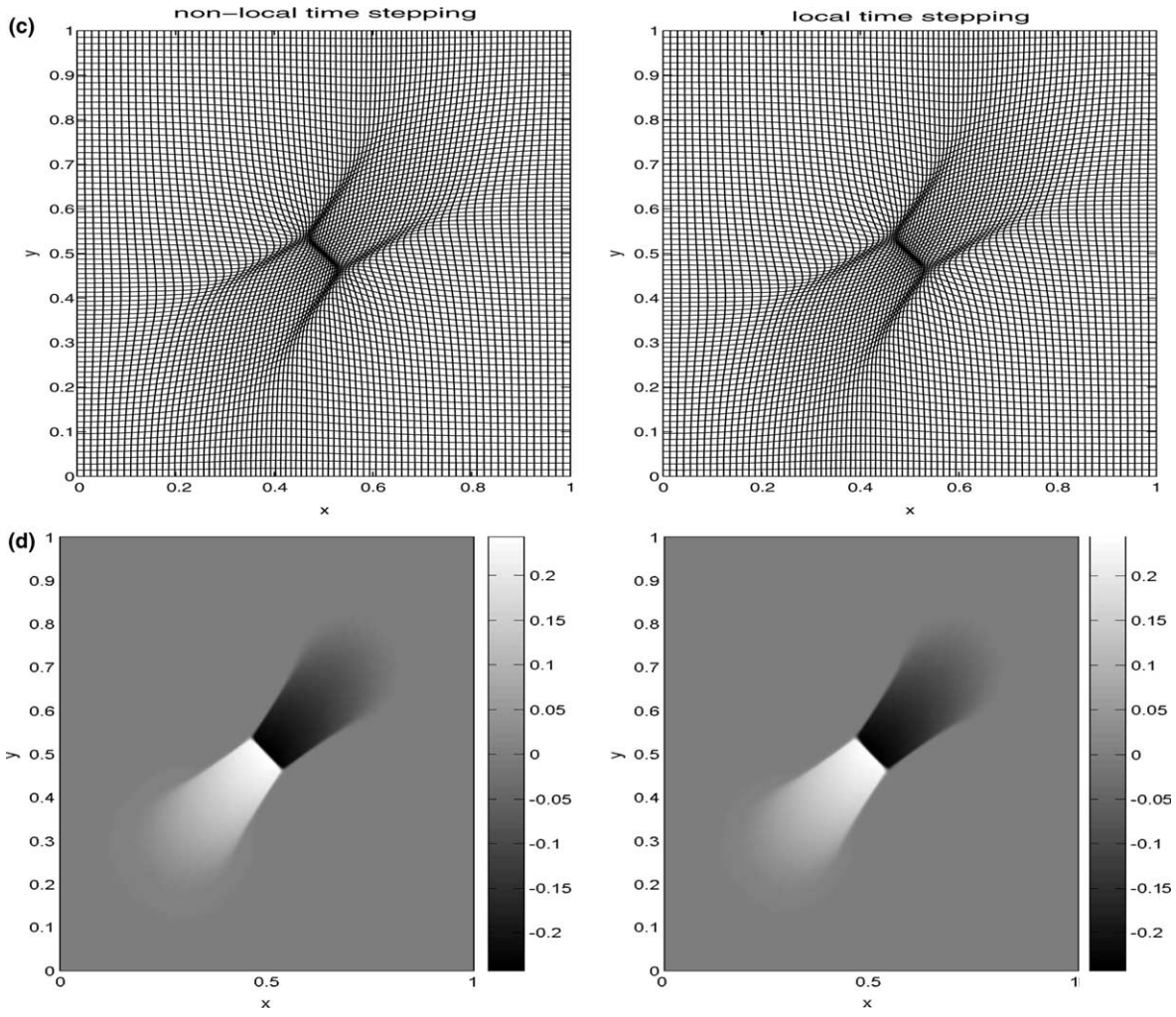

Fig. 6 (*continued*)

Table 4
Example 3.2: the l^1-error and CPU-time

	$t = 0.2$		$t = 0.4$	
	CPU time (s)	l^1-error	CPU time (s)	l^1-error
Non-local	48.25	1.87e−4	98.14	3.00e−4
Local	11.23	1.78e−4	32.72	1.76e−4
	$t = 0.6$		$t = 0.8$	
	CPU time (s)	l^1-error	CPU time (s)	l^1-error
Non-local	156.50	4.24e−4	252.30	6.55e−4
Local	65.59	1.86e−4	133.58	2.45e−4

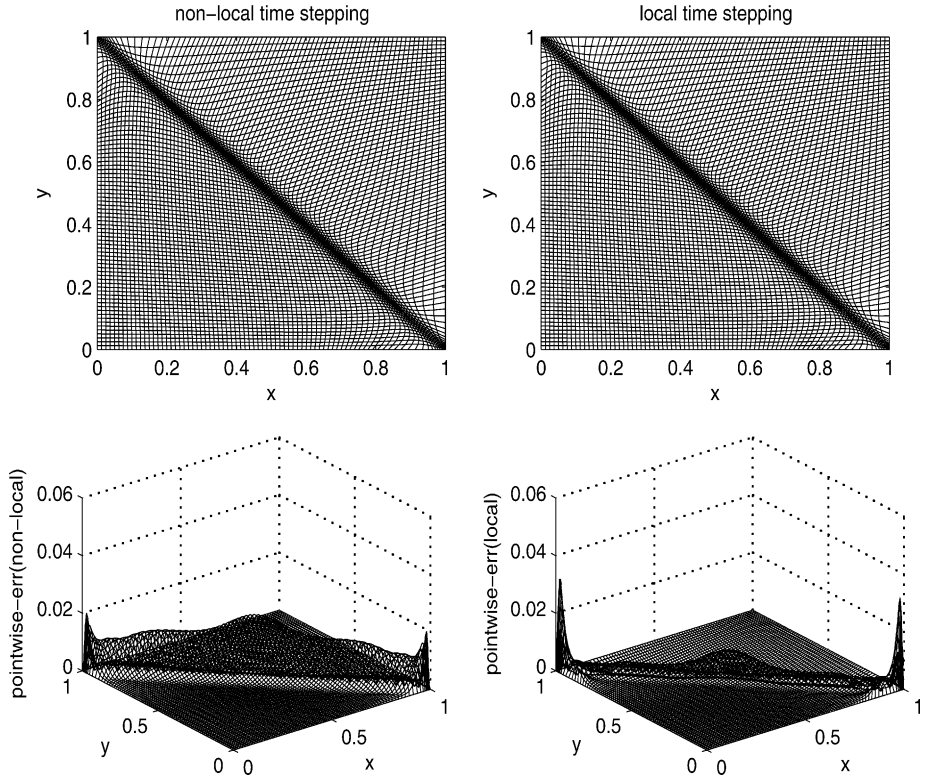

Fig. 7. Meshes and point-wise errors for Example 3.2 at $t = 1$: left is obtained by using the non-local time stepping and the right using the local time-stepping.

$$\Delta x \sim c_1 \epsilon. \tag{3.14}$$

It follows that if the proportional constant c_1 is sufficiently small then the time-step restriction of an explicit method will be determined by the viscous time step rather than the CFL condition. In our computations, since the proportional constant c_1 in (3.14) is not too small the value of Δt_{vis} is similar to that of Δt_{CFL}. However, in case that c_1 is very small (e.g., this will occur if the monitor constant α_2 in (3.12) is large enough), then with an explicit scheme the (very small) viscous time step has to be used. In this case, one possibility to handle the extremely small time steps (only for a few points near the viscous shock curves) is to use appropriate implicit methods.

Example 3.3. [2D Buckley–Leverett equation]. Consider the two-dimensional convection-diffusion equation

$$u_t + f(u)_x + g(u)_y = \epsilon \nabla^2 u, \tag{3.15}$$

with $\epsilon = 0.01$. The flux function is of the form

$$f(u) = \frac{u^2}{u^2 + (1-u^2)}, \quad g(u) = f(u)(1 - 5(1-u^2)) \tag{3.16}$$

and the initial data is

Table 5
Example 3.3: CPU time comparison

	$t = 0.1$ (s)	$t = 0.2$ (s)	$t = 0.3$ (s)	$t = 0.4$ (s)	$t = 0.5$ (s)
Non-local	32.8	64.1	94.3	123.0	150.7
Local	12.5	30.2	47.4	64.1	78.9

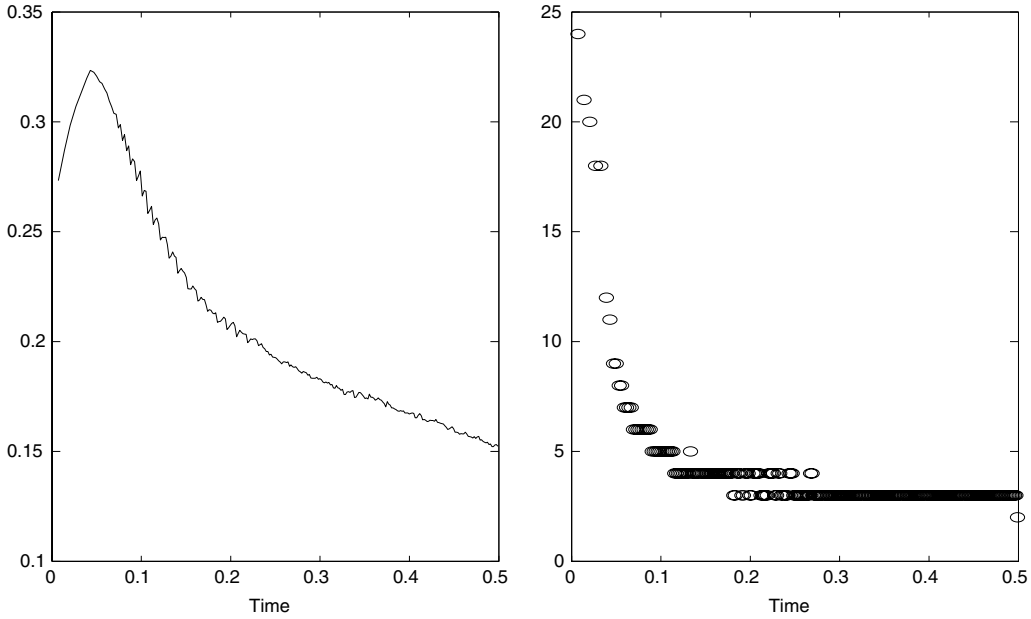

Fig. 8. Example 3.3: left is the percentage of elements taking the local time step; and the right is the ratio between the largest and smallest time steps.

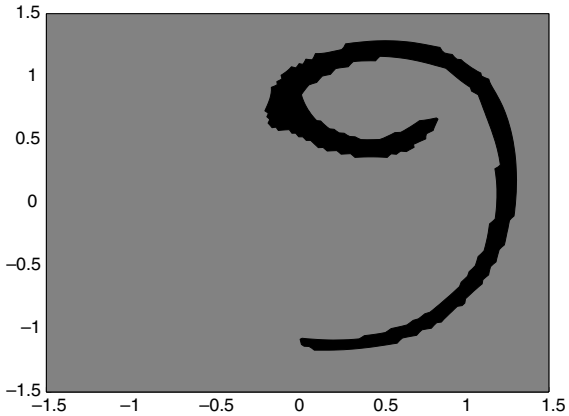

Fig. 9. Example 3.3: distribution of regions using local time stepping at $t = 0.5$. Darker (lighter) region uses smaller (larger) time steps.

$$u(x,y,0) = \begin{cases} 1 & x^2 + y^2 < 0.5, \\ 0 & \text{otherwise.} \end{cases} \tag{3.17}$$

Note that the above model includes gravitational effects in the y-direction. This example is taken from [15]. For this problem, 80×80 grid points are used and both the global and local CFL constants are chosen as 0.4. The monitor function used is (3.12) with $(\alpha_1, \alpha_2) = (0, 1)$. Table 5 shows the CPU-time used at different time level. It is observed that CPU time has a speed-up of about 1.9–2.6 times for local time stepping scheme. In Fig. 8, the percentage of elements taking local time step and the ratio between the largest and smallest time steps are presented. It is seen that the percentage is decreasing for $t \geqslant 0.1$, and the ratio is also decreasing. This indicates that the CPU time saving is getting less when t is larger. In Fig. 9, the region using locally varying time steps at $t = 0.5$ is presented. Numerical solutions at $t = 0.5$ obtained by using the moving mesh methods are shown in Fig. 10. The results obtained using and without using the locally varying time steps are in good agreement.

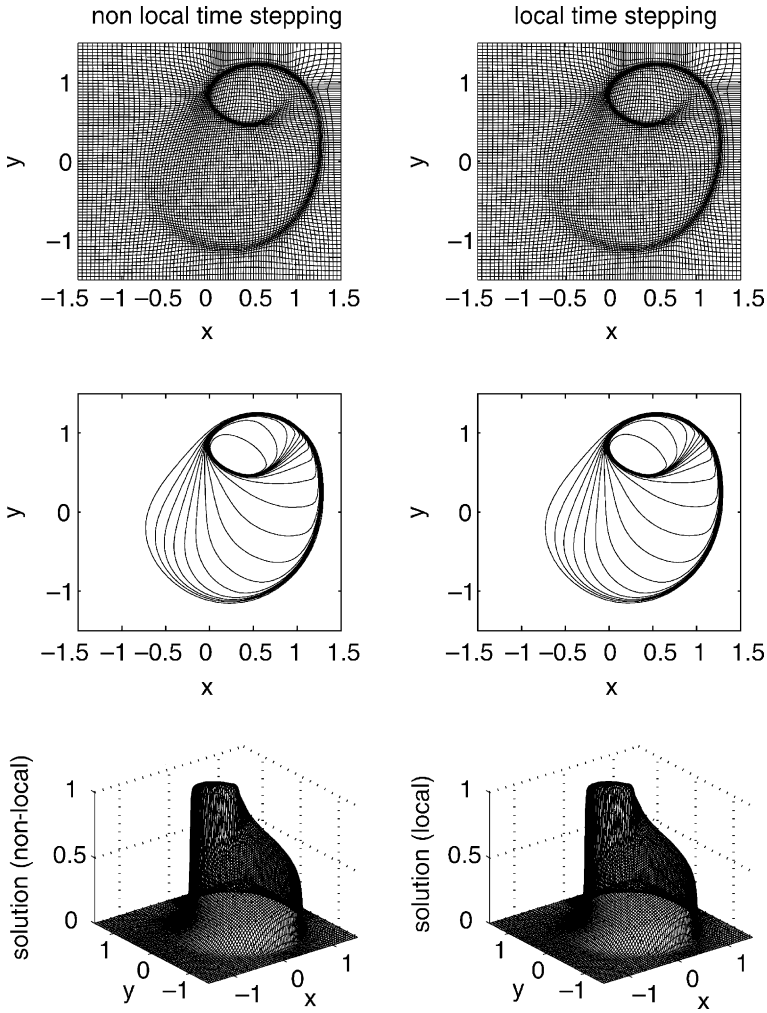

Fig. 10. Meshes and solutions for Example 3.3 at $t = 0.5$: left are obtained using the locally varying time steps, and right with the non-local time stepping.

4. Concluding remarks

In this work, a local time stepping technique for moving mesh methods has been developed. The algorithm proposed is relatively simple and has some good properties such as maintaining global conservation and convergence to a weak solution of scalar hyperbolic conservation laws. Numerical experiments in both 1D and 2D indicate that the local time stepping methods exhibit similar accuracy and stability to the global time stepping schemes, at a fraction of the computational cost.

It should be pointed out that the moving mesh method used in this paper is not representative of all moving mesh methods. For example, in the moving mesh PDE (MMPDE) approach of Russell and others [8,12,18], the physical PDE is solved on a moving mesh, whereas in this work the physical PDE is always solved on a fixed mesh and the effect of the mesh movement is achieved through the grid restructuring. The latter approach has been developed and used by several authors recently, see, e.g., [1,2,17,19,28]. One of the advantages of using this approach is that it delinks the PDE evolution part and the mesh-redistribution part so that the existing PDE solvers can be employed directly. This approach can also easily keep some desired properties of the numerical solutions such as mass conservation (in particular in multi-dimensions). However, the stability restriction on a fixed mesh will be governed by the smallest mesh size and therefore a local time stepping technique seems necessary for this type of moving mesh methods.

Acknowledgements

We would like to thank the referees for many helpful suggestions. The research of Y. Huang was supported in part by the special funds for Major State Basic Research Projects of China. The research of T. Tang was supported in part by CERG grants of Hong Kong Research Grants Council.

References

[1] B.N. Azarenok, Variational barrier method of adaptive grid generation in hyperbolic problems of gas dynamics, SIAM J. Numer. Anal. 40 (2002) 651–682.
[2] B.N. Azarenok, S.A. Ivanenko, T. Tang, Adaptive mesh redistribution method based on Godunov's scheme, Commun. Math. Sci. 1 (2003) 152–179.
[3] M.J. Baines, Moving Finite Elements, Oxford University Press, Oxford, 1994.
[4] G. Beckett, J.A. Mackenzie, On a uniformly accurate finite difference approximation of a singularly perturbed reaction-diffusion problem using grid equidistribution, J. Comput. Appl. Math. 131 (2001) 381–405.
[5] M.J. Berger, R.J. LeVeque, Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems, SIAM J. Numer. Anal. 35 (1998) 2298–2316.
[6] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484–512.
[7] J.U. Brackbill, J.S. Saltzman, Adaptive zoning for singular problems in two dimensions, J. Comput. Phys. 46 (1982) 342–368.
[8] H.D. Ceniceros, T.Y. Hou, An efficient dynamically adaptive mesh for potentially singular solutions, J. Comput. Phys. 172 (2001) 609–639.
[9] W.M. Cao, W.Z. Huang, R.D. Russell, A study of monitor functions for two dimensional adaptive mesh generation, SIAM J. Sci. Comput. 20 (1999) 1978–1994.
[10] C.N. Dawson, R. Kirby, High resolution schemes for conservation laws with locally varying time steps, SIAM J. Sci. Comput. 22 (2001) 2256–2281.
[11] J.E. Flaherty, R.M. Loy, M.S. Shephard, B.K. Szymanski, J.D. Teresco, L.H. Ziantz, Adaptive local refinement with octree load-balancing for the parallel solution of three-dimensional conservation laws, J. Parallel Distributed Comput. 47 (1997) 139–152.
[12] W. Huang, Y. Ren, R.D. Russell, Moving mesh methods based on moving mesh partial differential equations, J. Comput. Phys. 113 (1994) 279–290.
[13] A. Harten, J.M. Hyman, Self-adjusting grid methods for one–dimensional hyperbolic conservation laws, J. Comput. Phys. 50 (1983) 235–269.
[14] W. Huang, W. Sun, Variational mesh adaptation II: error estimates and monitor functions, J. Comput. Phys. 184 (2003) 619–648.

[15] A. Kurganov, E. Tadmor, New high-resolution central schemes for nonlinear conservation laws and convection–diffusion equations, J. Comput. Phys. 160 (2000) 241–282.
[16] R. Li, T. Tang, P. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, J. Comput. Phys. 170 (2001) 562–588.
[17] R. Li, T. Tang, P. Zhang, A moving mesh finite element algorithm for singular problems in two and three space dimensions, J. Comput. Phys. 177 (2002) 365–393.
[18] S. Li, L. Petzold, Moving mesh methods with upwinding schemes for time-dependent PDEs, J. Comput. Phys. 131 (1997) 368–377.
[19] F. Liu, S. Ji, G. Liao, An adaptive grid method and its application to steady Euler flow calculations, SIAM J. Sci. Comput. 20 (1998) 811–825.
[20] K. Miller, R.N. Miller, Moving finite element methods I, SIAM J. Numer. Anal. 18 (1981) 1019–1032.
[21] S. Osher, R. Sanders, Numerical approximations to nonlinear conservation laws with locally varying time and space grids, Math. Comp. 41 (1983) 321–336.
[22] C.-W. Shu, S. Osher, Efficient implement of essentially non-oscillatory shock-wave schemes, II, J. Comput. Phys. 83 (1989) 32–78.
[23] 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.
[24] H.Z. Tang, T. Tang, Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws, SIAM J. Numer. Anal. 41 (2003) 487–515.
[25] H.Z. Tang, T. Tang, P.-W. Zhang, An adaptive mesh redistribution method for nonlinear Hamilton–Jacobi equations in two- and three dimensions, J. Comput. Phys. 188 (2003) 543–572.
[26] Z.-R. Zhang, Moving Mesh Methods for Convection-Dominated Equations and Nonlinear Conservation Laws, PhD Thesis, Hong Kong Baptist University, 2003.
[27] Z.-R. Zhang, T. Tang, An adaptive mesh redistribution algorithm for convection-dominated problems, Commun. Pure Appl. Anal. 1 (2002) 341–357.
[28] P.A. Zegeling, H.P. Kok, Adaptive moving mesh computations for reaction-diffusion systems, J. Comput. Appl. Math. 168 (2004) 519–528.