Journal of Computational Physics **171,** 753–775 (2001)
doi:10.1006/jcph.2001.6809, available online at http://www.idealibrary.com on IDEAL®

Practical Aspects of Formulation and Solution of Moving Mesh Partial Differential Equations

Weizhang Huang

Department of Mathematics, University of Kansas, Lawrence, Kansas 66045
E-mail: huang@math.ukans.edu

Received December 7, 1999; revised April 6, 2001

Moving mesh partial differential equations (MMPDEs) are used in the MMPDE moving mesh method to generate adaptive moving meshes for the numerical solution of time dependent problems. How MMPDEs are formulated and solved is crucial to the efficiency and robustness of the method. In this paper, several practical aspects of formulating and solving MMPDEs are studied. They include spatial balance, scaling invariance, effective control of mesh concentration, bounds on time steps, multiple sub-steps, and two-level mesh movement. Numerical results are also given. © 2001 Academic Press

Key Words: mesh movement; adaptive mesh; mesh generation; PDE.

1. INTRODUCTION

In this paper we are concerned with moving mesh methods for the numerical solution of time dependent partial differential equations (PDEs). Within a moving mesh method, a mesh equation is employed to move the mesh points around in an orderly fashion while keeping them concentrated in regions of large solution variations. The key to the development of the method is to formulate and efficiently solve the mesh equation. In the past, a variety of one- and two-dimensional moving methods have been developed and successfully applied to many problems; e.g., see [12] and references therein. However, more robust and efficient methods have to be developed, especially in multi-dimensions.

We have recently developed the so-called moving mesh PDE (MMPDE) approach of moving mesh methods [13–16]. With this approach, an adaptive moving mesh is considered as the image of a reference mesh through a time dependent coordinate transformation. A continuous mesh equation, or an MMPDE, is then used to determine the coordinate transformation. The MMPDE is formulated as a modified gradient flow equation of a general adaption functional, which involves various properties of the mesh and physical solution. In one dimension, the approach has unified many existing methods and provided a framework to study their properties and develop robust and efficient new methods [13, 14]. The two-dimensional development of the method has also been very promising [15, 16]. It has been successfully used for generating both structured and unstructured meshes for a number of problems [4].

However, it should be pointed out that a number of issues related to the formulation and solution of the MMPDE are still unclear. For example, there are a variety of ways to define a modified gradient flow equation of a functional, and all the resultant MMPDEs work with a certain degree of success, although some of them are not as robust as others. Explicit control of mesh adaption is yet to be developed to avoid under or over concentration of mesh points. With the MMPDE approach, the physical PDE is replaced with an extended system consisting of the physical PDE and the mesh PDE. The system must be solved numerically for both the physical solution and the mesh. It is suggested in [15, 16] to use simultaneous solution methods for one-dimensional problems but alternating procedures in two dimensions for efficiency reasons. But, this type of alternating solution procedures may cause trouble in accuracy and stability since the physical and mesh PDEs can have significantly different time scales. Moreover, the time scale of the MMPDE depends on a user prescribed parameter τ, which makes it even more difficult to design a robust alternating solution procedure.

The objective of this paper is twofold. The first one is to attempt to offer answers to some of the questions raised by the aforementioned issues related to formulation and solution of the MMPDE. The other is to provide more implementation details than those provided in [15, 16] so that the interested reader would find an easier way to use the MMPDE moving mesh method for solving problems. To be more specific, we will show how to express the MMPDE in a simple form. Based on this form, we will then introduce the concepts of spatial balance and scaling invariance and study how to construct MMPDEs having these desired properties. Construction of the monitor function will be briefly discussed and a method will be presented for effective control of mesh concentration. Issues arising from the solution of MMPDEs, such as bounding time steps to get convergent meshes and employing multiple sub-steps for smoother mesh movement, will be investigated. A simple two-level mesh movement strategy will also be presented for further reducing the overhead of mesh generation for the numerical solution of PDEs.

An outline of the paper is as follows. In Section 2, the formulation of MMPDEs is briefly described. In Section 3, the concepts of spatial balancing and scaling invariance are introduced and discussed. Construction of the monitor function and effective control of mesh concentration are also addressed in Section 3. The discretization and solution procedure are given in Section 4 for two-dimensional MMPDEs. The choice of the time step bound and its effect are addressed also in this section. Numerical results are given in Section 5 for two problems having analytical solutions. A two-level mesh movement strategy is introduced for further improving the efficiency of the moving mesh method, and some numerical results using it are presented in Section 6. Finally, Section 7 contains the conclusions.

2. MOVING MESH PDEs

Let Ω be the domain where the physical problem is defined, and let Ω_c be the computational domain that is chosen artificially for the purpose of mesh generation. Denote by $\boldsymbol{x} = (x^1, x^2, x^3)^T$ and $\boldsymbol{\xi} = (\xi^1, \xi^2, \xi^3)^T$ the physical and computational coordinates in Ω

and Ω_c, respectively. Then, adaptive moving meshes for Ω can be generated as the images of a reference mesh in Ω_c through a one-to-one, time dependent, coordinate transformation or mapping $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$.

Mapping $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$ is defined in [15, 16] as the solution of the gradient flow equation of a quadratic functional that involves various properties of the mesh and physical solution. Specifically, denoting by $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{x}, t)$ the inverse mapping of $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$, we define the mesh adaption functional as

$$I[\boldsymbol{\xi}] = \frac{1}{2} \int_{\Omega} \sum_{i} (\nabla \xi^{i})^T G^{-1} \nabla \xi^{i} \, d\boldsymbol{x}, \tag{1}$$

where ∇ is the gradient operator with respect to $\boldsymbol{x}$ and G, the so-called monitor function, is a three-by-three symmetric positive definite matrix which interconnects the mesh and the physical solution. The MMPDE is then constructed as the (modified) gradient flow equation of $I[\boldsymbol{\xi}]$,

$$\frac{\partial \xi^{i}}{\partial t} = \frac{p}{\tau} \nabla \cdot (G^{-1} \nabla \xi^{i}), \quad i = 1, 2, 3, \tag{2}$$

where $\tau > 0$ is the user-defined parameter used for adjusting the time scale of mesh movement and p is a positive function to be chosen such that the MMPDE has desired properties. (1) is defined in terms of inverse mapping $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{x}, t)$. It is well known that the so defined functional is less likely to result in a singular coordinate transformation than a similar functional defined in terms of $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$; e.g., see [8]. In practice, however, it is more convenient to work directly with $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$ since it explicitly defines the locations of mesh points. Also, $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$ is easier to approximate numerically than $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{x}, t)$, which, by construction, has sharp layers in regions where mesh adaption is needed.

The MMPDE for $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$ can be obtained by interchanging the roles of dependent and independent variables in (2). To this end, it is convenient to introduce the covariant and contravariant base vectors

$$\boldsymbol{a}_i = \frac{\partial \boldsymbol{x}}{\partial \xi^{i}}, \quad \boldsymbol{a}^i = \nabla \xi^{i}, \quad i = 1, 2, 3, \tag{3}$$

which are related by

$$\boldsymbol{a}^i = \frac{1}{J} \boldsymbol{a}_j \times \boldsymbol{a}_k, \quad \boldsymbol{a}_i = J \boldsymbol{a}^j \times \boldsymbol{a}^k, \quad \boldsymbol{a}_i \cdot \boldsymbol{a}^l = \delta_i^l, \quad (i, j, k) \text{ cyclic}, \tag{4}$$

where δ_i^l is the Kronecker delta function, and J is the Jacobian

$$J = \boldsymbol{a}_1 \cdot (\boldsymbol{a}_2 \times \boldsymbol{a}_3). \tag{5}$$

With help of the following transformation relations

$$\nabla = \sum_i \boldsymbol{a}^i \frac{\partial}{\partial \xi^{i}} = \frac{1}{J} \sum_i \frac{\partial}{\partial \xi^{i}} J \boldsymbol{a}^i,$$
$$\frac{\partial J}{\partial \xi^{l}} = J \sum_i \boldsymbol{a}^i \cdot \frac{\partial \boldsymbol{a}_i}{\partial \xi^{l}},$$

$$\frac{\partial \boldsymbol{a}^i}{\partial \xi^l} = -\sum_s \left(\boldsymbol{a}^i \cdot \frac{\partial \boldsymbol{a}_s}{\partial \xi^l} \right) \boldsymbol{a}^s,$$

$$\frac{\partial \boldsymbol{x}}{\partial t} = -\sum_i \boldsymbol{a}_i \frac{\partial \xi^i}{\partial t}$$

that seem to first appear in [24], (2) can be transformed into

$$\tau \frac{\partial \boldsymbol{x}}{\partial t} = -\frac{p}{J} \sum_{i,j} \boldsymbol{a}_i \frac{\partial}{\partial \xi^j} (J \boldsymbol{a}^j \cdot G^{-1} \boldsymbol{a}^i), \tag{6}$$

or in a fully non-conservative form

$$\tau \frac{\partial \boldsymbol{x}}{\partial t} = p \left[\sum_{i,j} (\boldsymbol{a}^i \cdot G^{-1} \boldsymbol{a}^j) \frac{\partial^2 \boldsymbol{x}}{\partial \xi^i \partial \xi^j} - \sum_{i,j} \left(\boldsymbol{a}^i \cdot \frac{\partial G^{-1}}{\partial \xi^j} \boldsymbol{a}^j \right) \frac{\partial \boldsymbol{x}}{\partial \xi^i} \right]. \tag{7}$$

Functional (1) is very general, including several well-known steady-state methods as special cases. Indeed, we get the method based on harmonic maps [8] by taking $G = M/\sqrt{\det(M)}$ for some symmetric, positive definite matrix M, and Winslow's mesh adaption method [25] for $G = wI$ with some weight function w. The latter is generalized by Brackbill and Saltzman [3] to include terms for further mesh smoothness and orthogonality control. Theirs has become one of the most popular methods used for steady-state mesh adaption.

We conclude this section by remarking that formula (7) is much simpler and easier to implement than those given in [15, 16]. For Winslow's monitor function $G = wI$, (7) can further be simplified as

$$\tau \frac{\partial \boldsymbol{x}}{\partial t} = \frac{p}{w^2} \sum_{i,j} (\boldsymbol{a}^i \cdot \boldsymbol{a}^j) \frac{\partial}{\partial \xi^i} \left(\omega \frac{\partial \boldsymbol{x}}{\partial \xi^j} \right). \tag{8}$$

3. ASPECTS RELATED TO THE FORMULATION OF MMPDEs

We study in this section several aspects which are related to the formulation of the MMPDE and important to the efficiency and robustness of the moving mesh method. They include the choices of function p and the monitor function, scaling invariance, boundary correspondence, and the control of mesh concentration.

3.1. *Choice of Function p*

Ideally, p should be chosen so that all the mesh points move with a uniform time scale. An MMPDE, which has a uniform time scale, will be easier to integrate numerically and will work more reliably with a constant value of τ. Unfortunately, it is unclear mathematically how a PDE can be made to have a uniform time scale. We use here a heuristic, spatial balance criterion, i.e., choosing p such that the coefficients, especially those of the second-order derivatives, change evenly over the domain. In other words, we would like to choose p such that the MMPDE behaves more like a diffusion equation with an almost constant diffusion coefficient.

In [15, 16], p is taken as $1/\sqrt{g}$, where g is the determinant of G, motivated by the theory of harmonic maps. However, it is more appropriate to take $p = 1/\sqrt[d]{g}$, where d is the dimension of the spatial domain, from the view point of dimensional analysis. The latter leads to the MMPDE

$$\tau \frac{\partial \boldsymbol{x}}{\partial t} = \frac{1}{\sqrt[d]{g}} \left[\sum_{i,j} (\boldsymbol{a}^i \cdot G^{-1} \boldsymbol{a}^j) \frac{\partial^2 \boldsymbol{x}}{\partial \xi^i \partial \xi^j} - \sum_{i,j} \left(\boldsymbol{a}^i \cdot \frac{\partial G^{-1}}{\partial \xi^j} \boldsymbol{a}^j \right) \frac{\partial \boldsymbol{x}}{\partial \xi^i} \right]. \tag{9}$$

Noticing that $G^{-1} = O(1/\sqrt[d]{g})$, we estimate that the coefficients of (9) have the size $\sum_i \|\boldsymbol{a}^i\|^2/(\sqrt[d]{g})^2$. Thus, the MMPDE is spatially balanced if

$$\sum_i \|\boldsymbol{a}^i\|^2/(\sqrt[d]{g})^2 \approx \text{ constant.} \tag{10}$$

To see if (10) is true, we first point out that in one dimension, (10) reads as

$$\frac{1}{g} \frac{\partial \xi}{\partial x} \approx \text{ constant,} \tag{11}$$

which is the well-known equidistribution principle [14]. (10) can then be regarded as a generalization of the equidistribution principle. On the other hand, (11) can be derived from the Euler–Lagrange equation of functional

$$I[\xi] = \frac{1}{2} \int_\Omega \frac{1}{g} \left(\frac{\partial \xi}{\partial x} \right)^2 dx, \tag{12}$$

which is the one-dimensional version of functional (1). Thus, functional (1) can also be considered as a generalization of the equidistribution principle. Since both (1) and (10) are generalizations of the equidistribution principle, we can intuitively believe that the solution to (9), the gradient flow equation of (1), satisfies (10), at least when τ is small. In this sense, the coefficients of (9) will change evenly over the domain, and therefore, (9) is spatially balanced.

The simplest way to obtain well spatially balanced MMPDEs is to scale the terms on the right-hand side of (7) by some bound on the coefficients. For instance, we can take $p = 1/\sqrt{\sum_i (a_{ii}^2 + b_i^2)}$, where

$$\begin{aligned} a_{ij} &= \boldsymbol{a}^i \cdot G^{-1} \boldsymbol{a}^j, \\ b_i &= -\sum_j \boldsymbol{a}^i \cdot \frac{\partial G^{-1}}{\partial \xi^j} \boldsymbol{a}^j = -\boldsymbol{a}^i \cdot (\nabla \cdot G^{-1}). \end{aligned} \tag{13}$$

Note that the change of the monitor function has been taken into account in the scaling. The corresponding MMPDE is given by

$$\tau \frac{\partial \boldsymbol{x}}{\partial t} = \frac{1}{\sqrt{\sum_i (a_{ii}^2 + b_i^2)}} \left[\sum_{i,j} a_{ij} \frac{\partial^2 \boldsymbol{x}}{\partial \xi^i \partial \xi^j} + \sum_i b_i \frac{\partial \boldsymbol{x}}{\partial \xi^i} \right]. \tag{14}$$

By construction, the coefficients of the MMPDE have size $O(1)$.

Like row scaling used in Gaussian elimination for solving linear algebraic systems [10], it is difficult to understand theoretically how spatial balancing may affect the conditioning (or stiffness) of the MMPDE. Nevertheless, computational experience does show that the above two choices for p often work much better than no balancing (i.e., $p = 1$).

3.2. *Scaling Invariance*

Scaling invariance is important in formulating MMPDEs. It often indicates the robustness of MMPDEs to handle problems with physical domains and monitor functions having various scales. We are concerned here with the $\boldsymbol{x}$, $\boldsymbol{\xi}$, and G scaling invariance. The definition is as follows. An MMPDE is called $\boldsymbol{x}$ ($\boldsymbol{\xi}$, or G) scaling invariant if it is invariant under scaling transformation $\boldsymbol{x} \to \alpha \boldsymbol{x}$($\boldsymbol{\xi} \to \alpha \boldsymbol{\xi}$, or $G \to \alpha G$) for all $\alpha > 0$. The $\boldsymbol{x}$ scaling invariance is considered by suppressing the fact that the monitor function is a function of $\boldsymbol{x}$.

It is easy to verify that (9) is $\boldsymbol{\xi}$ scaling invariant while (14) is $\boldsymbol{x}$ and G scaling invariant. Thus, MMPDE (9) will change if we re-scale the physical domain and/or the monitor function. Obviously, the change can be compensated for by using a different value of τ. However, this is not good since a different value of τ must be used to obtain the same result after we simply re-scale Ω and/or G. On the other hand, a universal value for τ will work well with (14) for physical domains and monitor functions with various scales, provided that the computational domain is chosen to have a standard size.

3.3. *Boundary Correspondence*

To completely specify the coordinate transformation, the MMPDEs must be supplemented with suitable boundary conditions. Generally speaking, we can use three types of boundary conditions. The first is Dirichlet conditions with which the boundary points are held fixed. The second is orthogonal ones, for which one set of coordinate lines are required to be orthogonal to the physical boundary. For the other, presented in [15], the boundary point distribution is determined by a lower dimensional MMPDE. Since the third type of boundary conditions usually works better than the other two, we use it in our computation. For this reason and for completeness, we give it a more detailed description (in two dimensions) in the following.

Given a boundary segment Γ of $\partial\Omega$, let Γ_c be the corresponding boundary segment of $\partial\Omega_c$. Denoting by s the arc-length from a point on Γ to one of its end points and by ζ the arc-length from a point on Γ_c to one of its end points, we can identify Γ with $I = (0, \ell)$ and Γ_c with $I_c = (0, \ell_c)$. Then $s = s(\zeta, t)$ is determined by

$$\begin{aligned} \tau \frac{\partial s}{\partial t} &= \frac{1}{\sqrt{M^2 + (M_\zeta)^2}} \frac{\partial}{\partial \zeta}\left(M \frac{\partial s}{\partial \zeta} \right), \quad \zeta \in (0, \ell_c), \\ s(0) &= 0, \quad s(\ell_c) = \ell, \end{aligned} \tag{15}$$

where M, considered as function of s and t, is the one-dimensional monitor function. In our computation, we take M as the projection of the two-dimensional monitor function G along the boundary; i.e., if $\boldsymbol{t}$ is the unit tangent vector along the boundary then $M(s, t) = \boldsymbol{t}^T G \boldsymbol{t}$. Having obtained the arc-length coordinates for the boundary points, the corresponding physical coordinates are obtained through interpolation and the definition of the boundary.

3.4. *Monitor Functions*

The key to the success of the described MMPDE approach of mesh movement is to define a proper monitor function G. This issue is studied in [5] and a few guidelines are also given there. Consider the two-dimensional case. Generally, the monitor function can be defined through its eigen-decomposition, i.e.,

$$G = \lambda_1 \boldsymbol{v}(\boldsymbol{v})^T + \lambda_2 \boldsymbol{v}^\perp (\boldsymbol{v}^\perp)^T, \tag{16}$$

where $\boldsymbol{v}$ and $\boldsymbol{v}^\perp$ are normalized eigen-vectors that are perpendicular to each other. To perform mesh adaptation in the gradient direction of the physical solution $u = u(\boldsymbol{x}, t)$, a class of monitor functions can be constructed by choosing

$$\boldsymbol{v} = \frac{\nabla u}{\|\nabla u\|}, \quad \lambda_1 = \sqrt{1 + \|\nabla u\|^2}, \quad \lambda_2 \text{ is a function of } \lambda_1. \tag{17}$$

The choices $\lambda_2 = 1/\lambda_1$ and $\lambda_2 = \lambda_1$ lead to the monitor functions

$$G = \frac{1}{\sqrt{1 + |\nabla u|^2}} [I + (\nabla u)(\nabla u)^T] \quad \text{and} \quad G = \sqrt{1 + |\nabla u|^2} I, \tag{18}$$

which correspond to the method based on harmonic mappings [8] and Winslow's method [25], respectively. The generalization of the one-dimensional arc-length monitor function,

$$G = [I + (\nabla u)(\nabla u)^T]^{1/2}, \tag{19}$$

stays between the above two in the sense that the corresponding choice for λ_2 is 1.

It should be pointed out that monitor functions based on solution gradient is not always a best option and may fail in many cases. Instead, a better construction is based on some sort of error indicators such as an interpolation error estimate. For instance, if mesh adaption along gradient direction is desired, we can define $\boldsymbol{v}$ and λ_2 as in (17) but compute λ_1 using an error indicator. Since it needs lengthy discussion on error estimates, we will not discuss this topic further in the rest of the paper. Instead, we refer the interested reader to [6] for the details of using error indicators for mesh movement.

3.5. *Control of Mesh Concentration*

The monitor functions defined in the previous section lacks explicit control of mesh concentration. In fact, the monitor functions may over or under concentrate mesh points in regions of large solution gradient or errors, upon the distribution and magnitude of the solution gradient or the error indicator and thus upon problems and used numerical schemes. This will certainly make the underlying moving mesh method less robust.

A common remedy is to introduce a parameter (denoted by α) to control the intensity of mesh adaption. This issue has been discussed in one dimension by several researchers; e.g., see [1, 2, 22]. Particularly, Beckett and Mackenzie [1] define

$$g = 1 + \alpha |u_{\boldsymbol{xx}}|^{1/m}, \quad \alpha = \frac{1}{\langle |u_{\boldsymbol{xx}}|^{1/m} \rangle},$$

where m is an integer and $\langle \cdot \rangle$ denotes the average over the domain, based on the equidistribution principle (see (11)). They apply an adaptive scheme with this monitor function to the finite difference solution of a singularly perturbed, two-point boundary value problem and obtain a uniform convergence rate.

The application of the idea of Beckett and Mackenzie is not so straightforward in two dimensions. This is because the exact equidistribution principle such as (11) does not exist or at least is unknown so far in two dimensions. Moreover, G is now a matrix. Nevertheless, something can still be done with (16) and (17). Recall that mesh adaption along direction $\boldsymbol{v}(\boldsymbol{v}^{\perp})$ is primarily dominated by the change of $\lambda_1(\lambda_2)$ along $\boldsymbol{v}(\boldsymbol{v}^{\perp})$ [5]. With (17), λ_2 is defined as a function of λ_1, and mesh adaption is controlled by λ_1. Like in one dimension, we can expect that mesh points will be concentrated in regions where λ_1 is relatively large. Thus, we define

$$\lambda_1 = 1 + \alpha\phi, \quad \alpha = \frac{\beta}{\langle\phi\rangle(1-\beta)}, \tag{20}$$

where $\phi = \sqrt{1 + \|\nabla u\|^2} - 1$ and $\beta \in (0, 1)$ is a user-defined parameter. (ϕ can be defined as $\phi = \sqrt{1 + E^2} - 1$ when an error indicator E is available.) It is easy to see that when $\phi \approx \langle\phi\rangle, \lambda_1 \approx 1/(1-\beta)$, and an almost uniform mesh results. Note that we use $\phi = \sqrt{1 + \|\nabla u\|^2} - 1$ instead of $\phi = \|\nabla u\|$ to avoid the possible discontinuity at $\nabla u = 0$. It is easy to verify

$$\beta = \frac{\int_\Omega \lambda_1 d\boldsymbol{x} - \int_\Omega d\boldsymbol{x}}{\int_\Omega \lambda_1 \, d\boldsymbol{x}} = \frac{\int_\Omega (\alpha\phi)\, d\boldsymbol{x}}{\int_\Omega (1 + \alpha\phi)\, d\boldsymbol{x}}. \tag{21}$$

That is, β indicates the concentration in the region of large $\alpha\phi$ or λ_1. Thus, introduction of parameter β allows for effective control of mesh adaption.

3.6. *Smoothing of Monitor Functions*

It is common practice to smooth the monitor function in moving mesh methods. This is because the computed monitor function is often very non-smooth. At the same time, a smoother monitor function leads to a smoother mesh and also makes the MMPDE easier to integrate. In our computation, we use the arc-length monitor function (19) and the following smoothing algorithm. Let $\boldsymbol{x}_p$ be a mesh point in Ω and $\boldsymbol{\xi}_p$ the corresponding mesh point Ω_c. Then we define

$$G(\boldsymbol{x}_p) \leftarrow \frac{\int_{C(\boldsymbol{\xi}_p)} G(\boldsymbol{x}(\boldsymbol{\xi}))\, d\boldsymbol{\xi}}{\int_{C(\boldsymbol{\xi}_p)} d\boldsymbol{\xi}}, \tag{22}$$

where $C(\boldsymbol{\xi}_p) \subset \Omega_c$ is the union of neighboring grid cells having $\boldsymbol{\xi}_p$ as one of their vertices.

We also find that temporal smoothing of the monitor function is often useful. This is especially true for the case of generating the initial adaptive mesh using the MMPDE approach, where the temporal smoothing helps to obtain convergent meshes. We use

$$G^n \leftarrow 0.2G^{n-1} + 0.8G^n \tag{23}$$

in our computation although this has only a minor effect on time accurate integration.

4. DISCRETIZATION AND SOLUTION IN TWO DIMENSIONS

In this section we consider the discretization and solution of MMPDE (14) in two dimensions. For convenience, we rewrite the physical and computational coordinates as

$\boldsymbol{x} = (x, y)^T$ and $\boldsymbol{\xi} = (\xi, \eta)^T$. The Jacobian and contravariant base vectors are given by

$$J = x_\xi y_\eta - x_\eta y_\xi, \quad \boldsymbol{a}^1 = \frac{1}{J}\begin{bmatrix} y_\eta \\ -x_\eta \end{bmatrix}, \quad \boldsymbol{a}^2 = \frac{1}{J}\begin{bmatrix} -y_\xi \\ x_\xi \end{bmatrix}. \tag{24}$$

MMPDE (14) can be rewritten as

$$\tau \frac{\partial \boldsymbol{x}}{\partial t} = A\boldsymbol{x}, \tag{25}$$

where the differential operator A is defined as

$$A = \frac{1}{\sqrt{a_{11}^2 + a_{22}^2 + b_1^2 + b_2^2}}\left[a_{11}\frac{\partial^2}{\partial \xi^2} + 2a_{12}\frac{\partial^2}{\partial \xi \partial \eta} + a_{22}\frac{\partial^2}{\partial \eta^2} + b_1\frac{\partial}{\partial \xi} + b_2\frac{\partial}{\partial \eta}\right]; \tag{26}$$

and the coefficients are defined by (13), so

$$\begin{gathered} a_{11} = \boldsymbol{a}^1 \cdot G^{-1} \cdot \boldsymbol{a}^1, \quad a_{12} = \boldsymbol{a}^1 \cdot G^{-1} \cdot \boldsymbol{a}^2, \quad a_{22} = \boldsymbol{a}^2 \cdot G^{-1} \cdot \boldsymbol{a}^2, \\ b_1 = -\boldsymbol{a}^1 \cdot \left(\frac{\partial G^{-1}}{\partial \xi}\boldsymbol{a}^1 + \frac{\partial G^{-1}}{\partial \eta}\boldsymbol{a}^2\right), \\ b_2 = -\boldsymbol{a}^2 \cdot \left(\frac{\partial G^{-1}}{\partial \xi}\boldsymbol{a}^1 + \frac{\partial G^{-1}}{\partial \eta}\boldsymbol{a}^2\right). \end{gathered} \tag{27}$$

We use finite differences for the spatial discretization and the backward Euler scheme for the time integration of (25). For simplicity, we assume that the computational domain Ω_c is rectangular and a fixed, orthogonal mesh $\{(\xi_j, \eta_k), j = 0, \ldots, J, k = 0, \ldots, K\}$ is given on it. Denote the mesh at $t = t_n$ by $\boldsymbol{x}^n = \{\boldsymbol{x}_{j,k}^n = \boldsymbol{x}(\xi_j, \eta_k, t_n)\}$ and let $u^n = \{u_{j,k}^n\}$ be the physical solution on the mesh. Given $\boldsymbol{x}^n$ and u^n, the monitor function $G^n = G(\boldsymbol{x}^n, u^n)$ defined in (19) is computed through (16) by first transforming the first-order partial derivatives of u from the physical coordinates to the computational ones and then approximating them with central finite differences. The monitor function is then smoothed by applying (22) four times. For notation convenience, the smoothed monitor function will still be denoted by G.

Let

$$\begin{aligned} \Delta_\xi \boldsymbol{x}_{j,k} &= \frac{\boldsymbol{x}_{j+1,k} - \boldsymbol{x}_{j,k}}{\xi_{j+1} - \xi_{j,}}, \quad \nabla_\xi \boldsymbol{x}_{j,k} = \frac{\boldsymbol{x}_{j,k} - \boldsymbol{x}_{j-1,k}}{\xi_j - \xi_{j-1}}, \\ \delta_\xi^2 \boldsymbol{x}_{j,k} &= \frac{2}{\xi_{j+1} - \xi_{j-1}}\left(\frac{\boldsymbol{x}_{j+1,k} - \boldsymbol{x}_{j,k}}{\xi_{j+1} - \xi_j} - \frac{\boldsymbol{x}_{j,k} - \boldsymbol{x}_{j-1,k}}{\xi_j - \xi_{j-1}}\right), \\ \delta_{\xi\eta}^2 \boldsymbol{x}_{j,k} &= \frac{\boldsymbol{x}_{j+1,k+1} - \boldsymbol{x}_{j+1,k-1} - \boldsymbol{x}_{j-1,k+1} + \boldsymbol{x}_{j-1,k-1}}{(\xi_{j+1} - \xi_{j-1})(\eta_{j+1} - \eta_{j-1})}, \end{aligned}$$

and define difference operators in the η direction similarly. The numerical scheme for MMPDE (25) is then given by

$$\tau \frac{\boldsymbol{x}_{j,k}^{n+1} - \boldsymbol{x}_{j,k}^n}{\Delta t_n} = A_{j,k}\boldsymbol{x}_{j,k}^{n+1}, \tag{28}$$

where $A_{j,k}$ is a finite difference approximation of the differential operator A at $(\xi, \eta) = (\xi_j, \eta_k)$, viz.,

$$A_{j,k} = \frac{1}{\sqrt{a_{11,jk}^2 + a_{22,jk}^2 + b_{1,jk}^2 + b_{2,jk}^2}} \Bigg[a_{11,jk}\delta_\xi^2 + 2a_{12,jk}\delta_{\xi\eta}^2 + a_{22,jk}\delta_\eta^2 \\ + \frac{b_{1,jk} + |b_{1,jk}|}{2}\Delta_\xi + \frac{b_{1,jk} - |b_{1,jk}|}{2}\nabla_\xi + \frac{b_{2,jk} + |b_{2,jk}|}{2}\Delta_\eta + \frac{b_{2,jk} + |b_{2,jk}|}{2}\nabla_\eta \Bigg]. \tag{29}$$

Here, (25) is linearized by freezing the monitor function G and coefficients a_{11}, a_{12}, a_{22}, b_1, and b_2 at time $t = t_n$. An upwind treatment has been used in (28) for the first-order terms because the coefficients b_1 and b_2 can be large. But it is worth mentioning that the use of upwinding is by no means always critical, and a central discretization for the first-order terms often leads to very comparable results.

The scheme (28) has a nine-point stencil. The resulting algebraic system is solved with the preconditioned conjugate residual method until the mean-square-root residual is less than 10^{-8}. With the natural ordering of the unknown variables, the preconditioner is constructed as the modified (row sum equivalence), incomplete LU decomposition based on 13 points including the nine stencil points and four fill-in points with indices $(j, -2, k)$, $(j-2, k+1)$, $(j+2, k-1)$, and $(j+2, k)$. The construction of this ILU decomposition is standard, and the interested reader is referred to [21, 23] for the details.

The implementation of the Dirichlet type of boundary conditions is trivial. They can be solved simultaneously with (28). The implementations of the other two types of boundary conditions are not so straightforward. In principle, the orthogonal conditions can also be solved simultaneously with (28). However, because they are generally nonlinear and the existence of the solution to the problem with the these conditions is unclear, we adopt here an alternating procedure. More precisely, we first solve (28) with the fixed boundary points. The arc-length coordinates of the new boundary points are then obtained by requiring their distribution to be proportional to that of the points on the mesh line next to the boundary. The physical coordinates of the new points are finally obtained through interpolation from the arc-length coordinates and the definition of the boundary. Generally speaking, this extrapolation method only generates meshes nearly orthogonal to the boundary. For the same reason, the third type of boundary conditions is also implemented alternately. That is, the new point distributions $\{\boldsymbol{x}_{j,k}^{n+1}\}$ on all segments of the boundary are first obtained by solving the one-dimensional MMPDE (15) and using the boundary definition. The new locations $\{\boldsymbol{x}_{jk}^{n+1}\}$ of the interior grid points are then generated using (28) with the fixed new boundary points.

It is necessary to select the time step size Δt_n dynamically for efficient integration of (25). This is done by using a standard technique in the context of numerical ODEs. Specifically, we use

$$\Delta t_{n+1} = \Delta t_n \min\left(4, \max\left(0.1, 0.84\sqrt{\frac{\text{tol}}{\|\boldsymbol{e}\|_\infty}}\right)\right), \tag{30}$$

where tol is a prescribed error tolerance and $\boldsymbol{e}$ the error estimate. Let $\boldsymbol{x}^{n+1}$ be the solution of (28). We define the error estimate as $\boldsymbol{e} = \hat{\boldsymbol{x}}^{n+1} - \boldsymbol{x}^{n+1}$ with $\hat{\boldsymbol{x}}^{n+1}$ being the solution of

$$\begin{aligned} \tau \frac{\hat{\boldsymbol{x}}_{j,k}^{n+1} - \boldsymbol{x}_{j,k}^{n}}{\Delta t_n} &= A_{j,k} \hat{\boldsymbol{x}}_{j,k}^{n+1} + A_{j,k} \frac{\boldsymbol{x}_{j,k}^{n} - \boldsymbol{x}_{j,k}^{n+1}}{2}, \quad && \text{for } (\xi_j,\ \eta_k) \in \Omega_c \\ \hat{\boldsymbol{x}}_{j,k}^{n+1} &= \boldsymbol{x}_{j,k}^{n+1}, && \text{for } (\xi_j,\ \eta_k) \text{ on } \partial\Omega_c \end{aligned} \tag{31}$$

which is a second-order scheme for the linearized equation with the differential operator A approximated at time t_n. Schemes (28) and (31) have the same algebraic structure so that the coefficient matrix and the preconditioner built for (28) can also be used for solving (31).

One may notice that (28) is actually a semi-implicit scheme with special linearization such that the operator A is calculated at the previous time step. This is different from many other semi-implicit schemes where the exact Jacobian matrix of the underlying problem is used and treated implicitly. Moreover, the error estimate e is also based on such a specially linearized equation. As a consequence, an upper bound on Δt_n is often necessary. Since the time scale of the MMPDE is proportional to τ, we suggest to choose a bound depending on it; i.e.,

$$\Delta t_n \leq \tau \Delta \overline{t}_{max}. \tag{32}$$

Then, a universal value can be used for $\Delta \overline{t}_{max}$ (the bound of the scaled time step).

It is trivial to apply the above-described scheme to the generation of adaptive meshes for given steady-state functions. However, the application is not so straightforward for time dependent problems. In principle, the (time dependent) physical PDE and the MMPDE can be solved either simultaneously or alternately. But, simultaneous solution does not seem practical in two and three dimensions, since the coupling of the mesh and physical solution is often highly nonlinear and many structures (such as ellipticity and sparsity) in each of the mesh and physical PDEs may be lost in the coupled system. The following alternating procedure is used here for time dependent problems. To avoid possible confusion, we use hereafter Δt for the time step size associated with the MMPDE and Δt_{phy} for that related to the physical PDE.

Alternating procedure. Assume that the physical solution u^n, the mesh $\boldsymbol{x}^n$, and a time step size $\Delta t_{phy,n}$ are given at time $t = t_n$.

(i) Compute the monitor function $G^n(\boldsymbol{x}) = G(\boldsymbol{x}, t_n)$ using u^n and $\boldsymbol{x}^n$ and smooth it. G^n is understood as a continuous function in the sense of interpolation.

(ii) Integrate the MMPDE over the time period $[t_n, t_n + \Delta t_{phy,n}]$ using variable step size Δt_n and monitor function $G(\boldsymbol{x}) = G^n(\boldsymbol{x})$. More than one sub-step may be needed for the integration to reach $t = t_n + \Delta t_{phy,n}$. When this happens, the monitor function is updated from mesh to mesh via linear interpolation. The obtained mesh is denoted by $\boldsymbol{x}^{n+1}$.

(iii) Integrate the physical PDE with a fixed or variable step size. The mesh and mesh speed are calculated using linear interpolation:

$$\boldsymbol{x}(t) = \frac{t - t_n}{\Delta t_{phy,n}} \boldsymbol{x}^{n+1} + \frac{t_n + \Delta t_{phy,n} - t}{\Delta t_{phy,n}} \boldsymbol{x}^n. \tag{33}$$

(iv) When a variable step integrator is used in step (iii), the physical PDE may actually be integrated over a smaller step $\hat{\Delta} t_{phy,n} < \Delta t_{phy,n}$. In this case, the mesh at the actual new time level $t_{n+1} = t_n + \hat{\Delta} t_{phy,n}$ should be updated as $\boldsymbol{x}^{n+1} := \boldsymbol{x}(t_{n+1})$.

(v) Go to the next step with the step size predicted by the physical PDE solver.

In this procedure, the time step size used for integrating the MMPDE is implicitly bounded, i.e., $\Delta t_n \le \Delta t_{phy,n}$ or $\Delta \overline{t}_{max} \le \Delta t_{phy,n}/\tau$ (see (32)). There are several reasons why we need a more restrictive bound than this and therefore more than one sub-step used in step (ii). First, the time step $\Delta t_{phy,n}$ is determined only by the physical PDE and its solver. A different Δt_n should be used for solving the MMPDE since its time scale depends on τ and is often different from that of the physical PDE. Moreover, a smaller τ is desired in many cases for a prompter response of the mesh to the change of the physical solution. Thus, we should use a smaller Δt_n than $\Delta t_{phy,n}$. Furthermore, accuracy consideration also requires more than one sub-step and thus smaller Δt_n can be used in step (ii). Recall that MMPDE (25) is highly nonlinear, and the numerical scheme (28) is semi-implicit and based on a special linearization. A few more sub-steps are often necessary to obtain reasonable accuracy in the computed mesh. Finally, the alternating implementation of the boundary conditions and the MMPDE may also cause trouble, because one cycle of alternation is often insufficient for obtaining a reasonably convergent solution. A remedy is to use more than one sub-step for integrating the MMPDE.

Based on the above consideration, we use

$$\Delta t_n \le \min\left\{ \frac{\Delta t_{phy,n}}{m}, \tau \Delta \overline{t}_{max} \right\} \tag{34}$$

for some positive integer m. This condition guarantees that at least m sub-steps are taken in step (ii).

5. NUMERICAL EXPERIMENTS

In this section we present some numerical results obtained with the scheme described in the previous sections for two examples with analytical physical solutions. The third type of boundary point specification is used. Monitor function (19) is calculated by approximating the solution gradient with central finite differences based on solution values at grid points, and the tolerance tol for controlling mesh time steps in (30) is taken as tol $= 10^{-2}$.

EXAMPLE 4.1. The first example is to generate an adaptive mesh for the model problem of interaction of an oblique shock and a boundary layer. The physical solution is represented by a single scalar function $u(x, y) = \tanh(Ry) - \tanh(R(0.5x - y - 1))$ with $R = 50$ on the rectangular domain $[0, 4] \times [0, 2]$. This example has been used by several researchers to demonstrate their mesh adaptive methods; e.g., see [11]. We solve this problem using the moving mesh method with $\tau = 1$. Since $u = u(x, y)$ is time independent, adaptive meshes can also be obtained by directly solving the steady-state mesh equation, i.e., (25) without the mesh speed term. But, since the mesh equation is highly nonlinear, nonlinear iteration methods such as Newton's often fail to converge. The interested reader is referred to [18, 19] for discussion on numerical solution of steady-state mesh equations for the mesh generation case (i.e., $u \equiv 1$).

Figure 1 shows a typical result obtained on a 31×31 mesh with $\beta = 0.5$, $\Delta \overline{t}_{max} = 0.5$, and temporal smoothing for the monitor function. The computation is terminated when the L_2 norm of the mesh speed is smaller than 10^{-6} (after about 69 time steps). It is clear that the grid points of the convergent adaptive mesh are concentrated around the regions

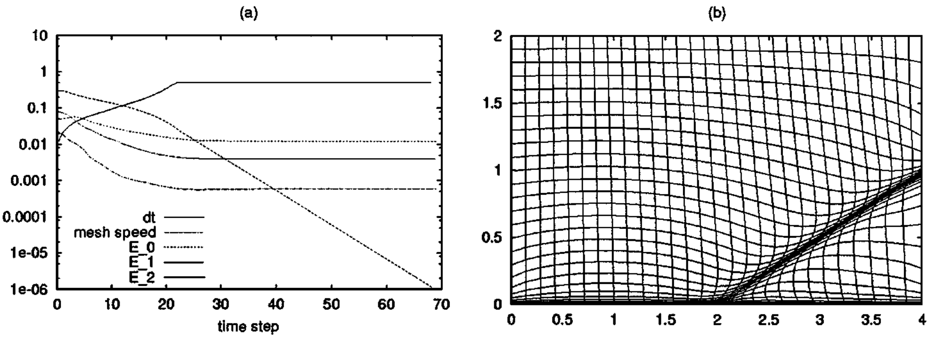

FIG. 1. A typical result for Example 1 obtained with $\beta = 0.5$ (concentration control parameter), $\Delta \bar{t}_{max} = 0.5$, and the temporal smoothing of the monitor function. (a) Δt, the L_2 norm of the mesh speed, E_0, E_1, and E_2 as functions of time steps. (b) Convergent mesh of size 31×31.

of the boundary layer and oblique shock. The mesh speed decreases monotonically. It is also interesting to see that after about 23 steps, the time, step size Δt takes its allowed maximum (0.5 in this case) and maintains the value for the rest of computation. Also shown in Fig. 1 are the L_2 norm of the constant-, linear-, and quadratic-polynomial interpolation errors E_0, E_1, and E_2. All of them drop quickly in the first 20 steps and stay nearly constant afterward. The interpolation errors are much smaller on the convergent adaptive mesh than those on the uniform one. In fact, E_0 drops by a factor of about 4, E_1 by 21, and E_2 by 39. This result indicates that a higher order method may gain more from mesh adaption than a lower order one.

We next solve the problem with a bigger $\Delta \bar{t}_{max} = 2.0$. The results are shown in Fig. 2. Like the case $\Delta \bar{t}_{max} = 0.5$, Δt reaches its allowed maximum quickly. But in this case, the mesh speed does not decrease monotonically any more. Instead, it decreases monotonically to about 0.001, then goes up to 0.01 (constant Δt is used during this period), and oscillates for the rest of computation. The oscillations also occur in the mesh, Δt, and interpolation errors. A convergent mesh cannot be obtained for this case. We stop the computation after 200 time steps. The mesh at this time is shown in Fig. 2b. Interestingly, we can see that the mesh, as well as the interpolation errors, are not much different from those shown in Fig. 1. Thus, the oscillations will not do much harm when the method is applied to steady-state

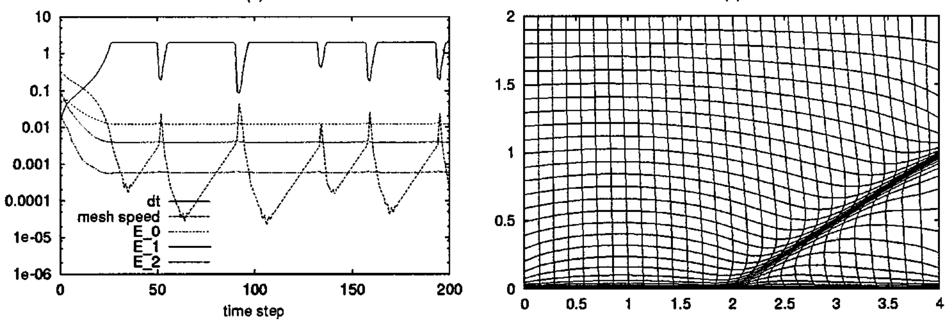

FIG. 2. Results for Example 1 obtained with $\beta = 0.5$ and $\Delta \bar{t}_{max} = 2.0$. (a) Δt, the L_2 norm of the mesh speed, E_0, E_1, and E_2 as functions of time steps. (b) The mesh of size 31×31 at the 200th time step.

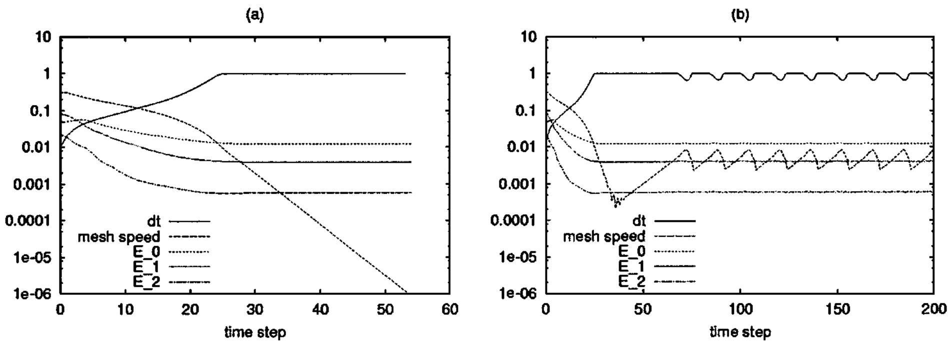

FIG. 3. Results for Example 1 obtained with $\Delta\bar{t}_{max} = 1$ and (a) with or (b) without temporal smoothing for the monitor function.

problems, since we can always stop the computation after a certain number of steps and obtain a mesh sufficiently close to the convergent one. However, the situation is different for time dependent problems. Oscillatory mesh movement may make the physical PDE difficult to integrate and even cause numerical instability.

We carry out a number of computations with tighter tolerance tol. We have found that the oscillations associated with larger Δt cannot be totally eliminated by reducing tolerance tol although their average and amplitude do decrease with tol. Comparatively, temporal smoothing of the monitor function provides a more effective tool in suppressing the oscillations. We show in Fig. 3 the results obtained with and without temporal smoothing. It can be seen that without temporal smoothing, the mesh speed oscillates and no convergent mesh can be obtained for $\Delta\bar{t}_{max} = 1.0$, whereas this happens only at a larger $\Delta\bar{t}_{max}$ (about 2 in this case) when temporal smoothing is applied.

Hence, it is essential to limit Δt in order to obtain a convergent solution to the MMPDE. Unfortunately, the choice of the bound is generally problem dependent. In principle, a suitable bound has to be found by trial and error. Nevertheless, it is often not difficult to take a working value for $\Delta\bar{t}_{max}$. In fact, we have found that $\Delta\bar{t}_{max} = 0.5$ works for all of the tested problems. Moreover, the choice of $\Delta\bar{t}_{max}$ seems to be independent of mesh size. For the current case, we have tried various meshes, including the finest one 121×121, and $\Delta\bar{t}_{max} = 0.5$ works fine.

Next we demonstrate how mesh adaption (concentration) can be controlled with parameter β. To this end, we carry out computations with $\beta = 0.2, 0.5$, and 0.8 and calculate the percent of the mesh points at which λ_1 is greater than its average. The convergent meshes and the percentages are shown in Fig. 4. The concentration control with β is clearly effective, namely, a smaller value of β leads to lower mesh adaption while a larger value results in a higher mesh concentration. One may also notice from Fig. 4d that the computed percentage does not match up with the used value of β on the convergent mesh. This is mainly due to the fact that β, defined in (21), is based on the distribution of λ_1, where λ_1 is not proportionally related to the mesh density or mesh concentration.

EXAMPLE 4.2. The next example is the well-known Burgers' equation

$$\frac{\partial u}{\partial t} = a\Delta u - u\frac{\partial u}{\partial x} - u\frac{\partial u}{\partial y}, \quad t \in (0.25, 1.5] \tag{35}$$

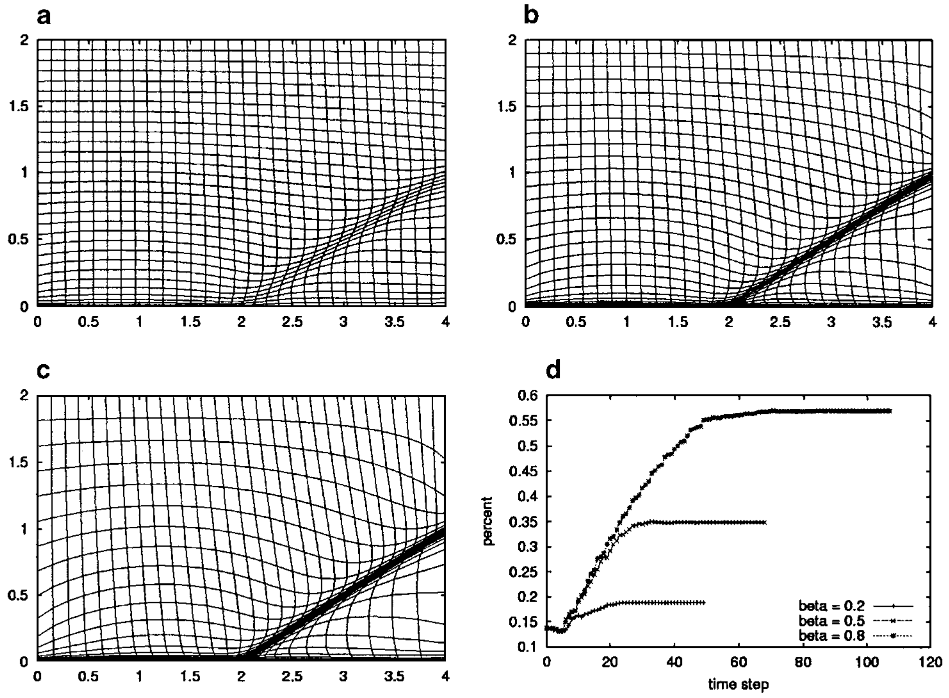

FIG. 4. Results for Example 1 obtained with $\Delta\bar{t}_{max} = 0.1$ and various values of β (concentration control parameter). Convergent meshes are shown in (a) $\beta = 0.2$, (b) $\beta = 0.5$, and (c) $\beta = 0.8$, and (d) shows the percentage (as a function of time steps) of mesh points at which λ_1 is greater than its average.

defined in the unit square. The initial and Dirichlet boundary conditions are chosen so that the problem has exact solution $u(x, y, t) = 1/\{1 + \exp[(x + y - t)/(2a)]\}$. This solution describes a straight-line wave (u is constant along line $x + y = c$) moving in the direction $\theta = 45^\circ$. We take $a = 0.005$.

We use finite differences for the spatial discretization of (35) and the three-stage third-order singly diagonally implicit Runge–Kutta (SDIRK) method [7] for the time integration. The alternating procedure described in Section 3 is used for solving the extended system consisting of the physical and mesh PDEs. A fixed step $\Delta t_{phy} = 0.01$ is used. The adaptive initial mesh is obtained by solving the MMPDE until the L_2 mesh speed is less than 10^{-6}. In this initial mesh generation process, we take $\tau = 1$ and compute the arc-length monitor function with the initial solution. In the results presented in the following, the (global) error

$$e(t) = \int_0^t \|u - u^{comput}\|_2 \, dt \tag{36}$$

is used. We take $\Delta\bar{t}_{max} = 0.5$ in (34) for this example.

Figure 5 shows a typical moving mesh at various time instants. The mesh is obtained with $\tau = 0.01$, $\beta = 0.5$, and $m = 3$ (so three sub-steps in each alternation between solving the mesh and physical PDEs are used for integrating the MMPDE). The corresponding mesh speed and error are plotted in Fig. 6a as functions of time. It can be seen from these results that the mesh points are concentrated around the moving wave front.

Figure 6b shows the mesh speed and error obtained without restricting Δt. Recall that the alternating solution procedure implies that $\Delta\bar{t}_{max} \leq \Delta t_{phy}/\tau$. Thus, we actually have

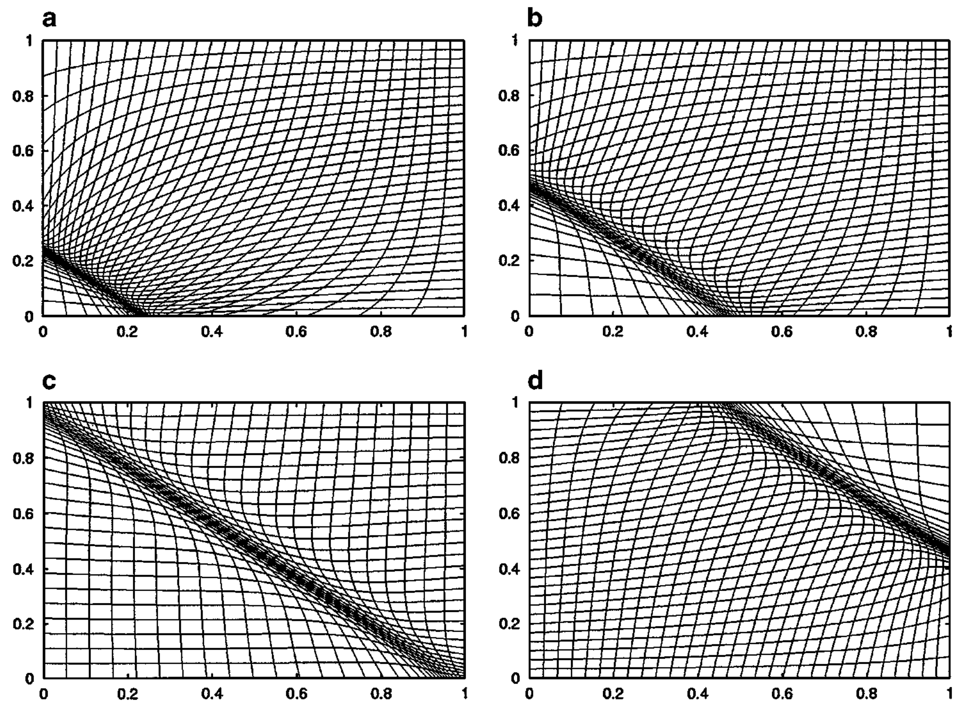

FIG. 5. A typical 31 × 31 mesh at various time instants is obtained for Example 2 with $\tau = 0.01$, $\beta = 0.5$, and $m = 3$. (a) $t = 0.26$, (b) $t = 0.5$, (c) $t = 0.98$, and (d) $t = 1.46$.

$\Delta\bar{t}_{max} = 1$ for this case since $\Delta t_{phy} = \tau = 10^{-2}$. Comparing the results in Fig. 6, it is clear that without restriction of Δt_n (or more precisely, with a too large Δt_n), the mesh speed and thus mesh movement becomes oscillatory, whereas a more smoothly moving mesh results with a suitable limit of Δt_n. It is interesting to observe that the error function $e(t)$ behaves similarly for these two cases, although the smoother one does lead to a better result.

Figure 7, together with Fig. 6a, shows the effect of τ on the solution and mesh movement. Note that the result for $\tau = 0.01$ is nearly identical to that with $\tau = 0.001$.

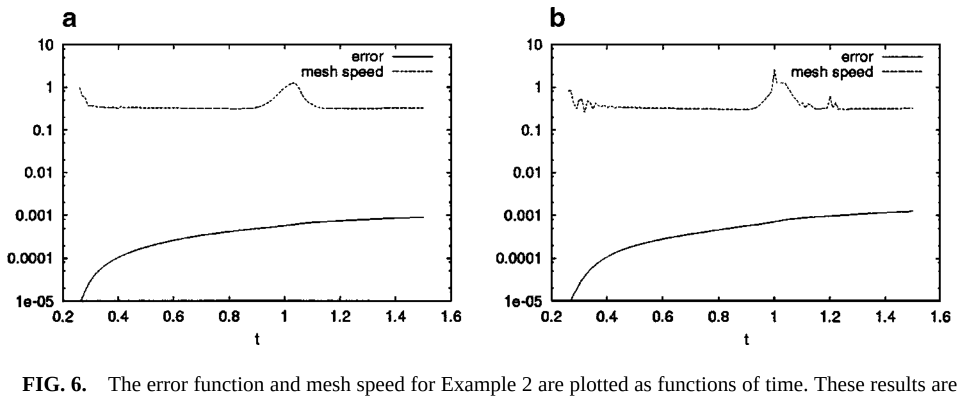

FIG. 6. The error function and mesh speed for Example 2 are plotted as functions of time. These results are obtained (a) with ($m = 3$) or (b) without bounding the time step size for mesh movement.

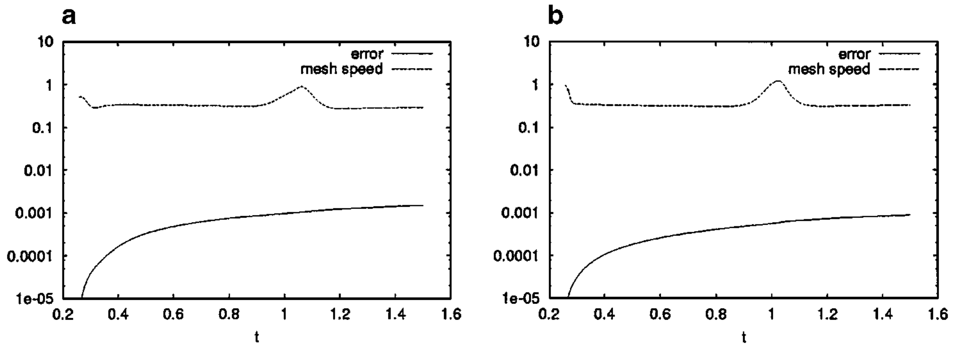

FIG. 7. The error function and mesh speed for Example 2 are obtained with various values of τ; (a) $\tau = 0.1$ and (b) $\tau = 0.001$. See also Figs. 5 and 6a for the results with $\tau = 0.01$.

We show in Fig. 8 (also see Fig. 5c) the mesh at $t = 0.98$, the error function, and the percentage of mesh points at which λ_1 is greater than its average for various values of β. The effectiveness for using β to control mesh concentration is clear. From Fig. 8c, one can see that for the case $\beta = 0.2$, the mesh points are less concentrated and this causes the error to grow quickly with time. Figure 8d shows that the percentage of the mesh points with large λ_1 does not stay constant in time. It takes its maximum as the solution wave crosses the line $y = -x$ at $t = 1$. That is, more mesh points are concentrated at $t = 1$ near the

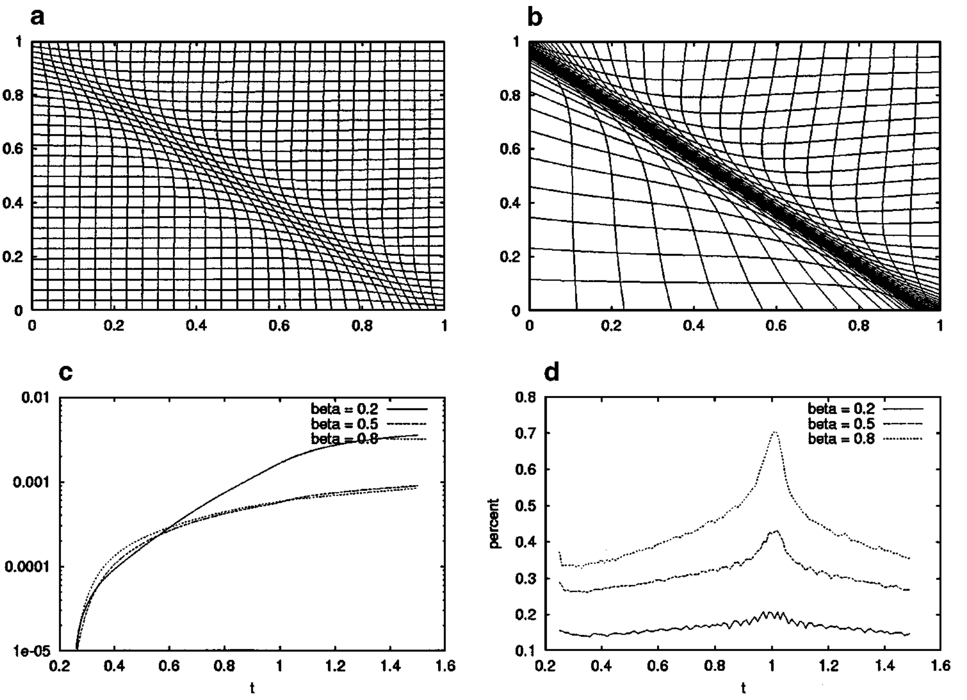

FIG. 8. Results for Example 2 are obtained with various values of β. (a) $\beta = 0.2$ and $t = 0.98$, (b) $\beta = 0.8$ and $t = 0.98$, and (c) and (d) $\beta = 0.2, 0.5$, and 0.8.

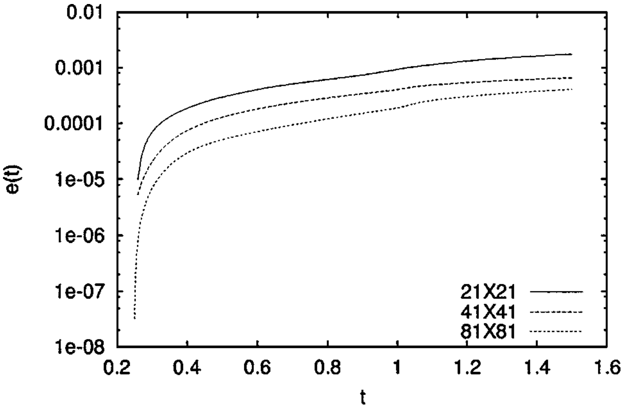

FIG. 9. The errors are obtained for Example 2 with various mesh sizes.

solution wave. This can also be seen clearly from the mesh in Fig. 5 and the mesh speed in Fig. 6a.

Finally, we investigate the convergence of the moving mesh method. Results obtained with various numbers of mesh points are shown in Fig. 9 and Table I. Note that in the 81×81 case, the mesh is highly concentrated around the wave front (see Figs. 11a and 11b), and a small size of time step has to be used for solving the physical PDE. This can be clearly seen in Table I, where it is shown that the case *a* with an 81×81 mesh and $\Delta t_{phy} = 0.01$ yields a result no better than that obtained with a 41×41 mesh. For this reason, we use $\Delta t_{phy} = 10^{-3}$ for the 81×81 case and plot the result in Fig. 9, along with the results obtained with $\Delta t_{phy} = 0.01$ for the other cases. The error decreases as the number of mesh points is increased. However, the convergence is only about the first-order, instead of the expected second-order. This result seems to be consistent with the observation made for one-dimensional problems [17]: with the moving mesh method, the error decreases quickly for small numbers of mesh points, then slowly at about the first-order rate, and finally at the

TABLE I
CPU Time and Errors Obtained with Fixed, One-Level, and Two-Level Moving Meshes for Example 4.2

Physical mesh		$JM = KM$	Total CPU	% CPU for mesh	$e(t = 1.5)$ (ratio)
Moving	21×21	1	36	67%	1.72e-3
	41×41	1	364	67%	6.52e-4 (2.64)
	81×81^a	1	2052	67%	7.77e-4 (0.84)
	81×81^b	1	5827	67%	4.06e-4 (1.61)
Moving	21×21	1	36	67%	1.72e-3
	41×41	2	82	29%	4.21e-4 (4.09)
	81×81	4	289	8%	1.13e-4 (3.73)
Fixed	21×21		10		6.34e-2
	41×41		55		2.26e-2 (2.81)
	81×81		306		6.29e-3 (3.59)

Note. $\Delta t_{phy} = 10^{-2}$ and 10^{-3} are used for cases *a* and *b*, respectively, on an 81×81 mesh. The other results are obtained with $\Delta t_{phy} = 10^{-2}$.

rate of second-order for large numbers of mesh points. A similar observation is also made in [2].

6. TWO-LEVEL MESH MOVEMENT

The efficiency of the moving mesh method described in the previous sections can further be improved when combined with a two-level mesh movement strategy. With this strategy, the mesh movement is performed on a relatively coarse mesh and a fine mesh used for solving the physical PDE is obtained via interpolation. This is reasonable since, unlike the physical solution, the mesh points do not have to be calculated to high accuracy. A similar idea has been used by Fiedler and Trapp [9] for the dynamic generation of adaptive meshes using an elliptic differential equation system and by Mulholland *et al.* [20], where an adaptive finite difference mesh is used for the pseudo-spectral solution of near-singular problems.

For the two-level mesh movement, the relation between the coarse and fine meshes must be defined first. Denote the coarse mesh by $\{(x^c_{jk}, y^c_{jk}),\ j = 0, \ldots, J^c, k = 0, \ldots, K^c\}$ and the fine mesh by $\{(x_{jk}, y_{jk}),\ j = 0, \ldots, J, k = 0, \ldots, K\}$. They can be related through the projection

$$x^c_{j,k} = x_{JC(j),KC(k)}, \quad y^c_{j,k} = y_{JC(j),KC(k)}, \tag{37}$$

where the arrays *JC* and *KC* satisfy

$$\begin{aligned} JC(0) &= 0 < JC(1) < \cdots < JC(J^c) = J, \\ KC(0) &= 0 < KC(1) < \cdots < KC(K^c) = K. \end{aligned} \tag{38}$$

In our computation, we choose

$$\begin{aligned} JC(j) &= j \cdot JM, \quad j = 0, 1, \ldots, J^c, \\ KC(j) &= k \cdot KM, \quad k = 0, 1, \ldots, K^c, \end{aligned} \tag{39}$$

where *JM* and *KM* are two prescribed positive integers. (Note that J, K and J^c, K^c must satisfy $J = J^c \cdot JM$ and $K = K^c \cdot KM$.) To capture the fine structures of the physical solution, the monitor function is first computed on the fine mesh and then projected via area averaging to the coarse mesh. Having obtained the new coarse mesh by solving the MMPDE, we compute the fine mesh at the new time step via linear interpolation.

Note that the coarse mesh cannot be chosen too coarse to catch the fine structures of the physical solution. On the other hand, it cannot be chosen too fine to reduce the necessary overhead of mesh movement. Computational experience shows that choices *JM*, *KM*=2,3,4 often lead to satisfactory results.

Table I and Fig. 10 show the results obtained with the moving mesh method combined with the two-level mesh movement strategy for Burgers' equation (Example 4.2). We use here a coarse mesh of size 21×21 and two fine meshes of sizes 41×41 and 81×81 which correspond to *JM*=*KM*=2 and 4, respectively. For comparison, we also list the results obtained with fixed, uniform meshes of three corresponding sizes. The CPU time listed in Table I is in seconds on a Dell workstation with single Pentium III 500 MHz processor. It is interesting to point out that the CPU time used for mesh generation is about 67% of the

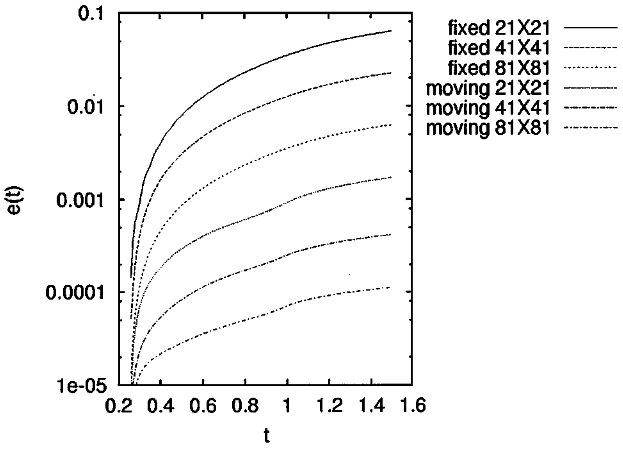

FIG. 10. Example 2: The error is obtained for Example 2 with $\tau = 0.01$, $\beta = 0.5$, and $m = 3$ and with fixed and two-level moving meshes.

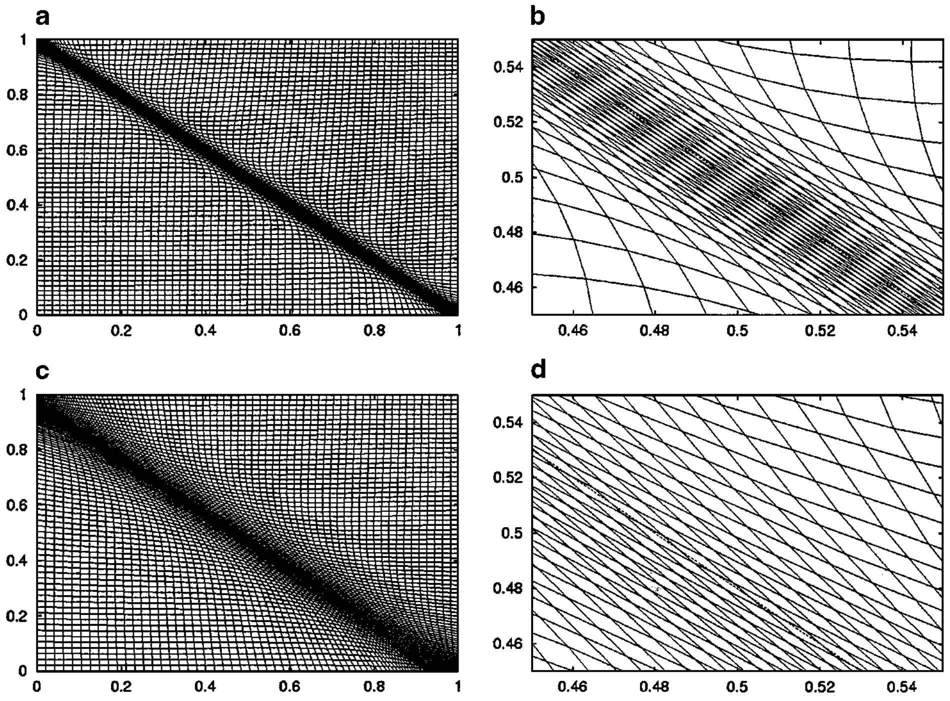

FIG. 11. 81×81 meshes and their closer views near point (0.5, 0.5). The dashed line indicates the position of the wave front. (a) Mesh at $t = 0.998$ obtained by solving the MMPDE with $dt_{phy} = 10^{-3}$. (b) A closer view of the mesh in (a). (c) Mesh at $t = 0.98$ obtained by linearly interpolating a 21×21 moving mesh (the two-level mesh movement). (d) A closer view of the mesh in (c).

total CPU time when only a one-level mesh is used (for both mesh movement and physical PDE), and it decreases to 29% and 8% for the two-level mesh movement with $JM = KM = 2$ and 4, respectively. The efficiency gain is significant. The second-order convergence of the method is also clear. Furthermore, unlike the case with a one-level moving 81×81 mesh, a larger time step $\Delta t_{phy} = 0.01$ works well for all of the three cases with the two-level mesh movement. This is due to the fact that a fine mesh obtained by interpolating a coarse, moving mesh is less concentrated and skewed than one obtained by directly solving the MMPDE; see Fig. 11.

7. CONCLUSIONS

Several practical aspects of formulating and numerically solving the moving mesh PDE have been studied in the previous sections. It was shown how the MMPDE can be expressed in the form (7). The simplicity of (7) allows for the use of concepts, spatial balance and invariance, in formulating more robust MMPDEs. For instance, the function p in (7) can be chosen so that it is easier to choose τ and integrate the resultant MMPDE. With the particular choice of the function, $p = 1/\sqrt{\sum_i (a_{ii}^2 + b_i^2)}$, we obtained MMPDE (14), which is well spatially balanced and is invariant under the G and $\boldsymbol{x}$ scaling transformations. These properties are desired for the robustness of the MMPDE.

Defining a proper monitor function is always the key to the success of the moving mesh method. The monitor function can be constructed based on solution gradient or error estimates. However, the so-defined monitor function often under or over concentrates mesh points in regions of large gradient or errors. A remedy was proposed in Section 3 to allow for an explicit control of mesh concentration. The basic idea behind the method is to scale λ_1, the leading eigen-value of the monitor function, by its average.

A numerical scheme for solving the two-dimensional MMPDE was presented. It is based on finite differences for the spatial discretization and the backward Euler method for the time integration. The MMPDE is linearized in a special manner such that the differential operator (26) is calculated at the current time step $t = t_n$. The resulting linear algebraic system is solved with the preconditioned conjugate residual method with the preconditioner being constructed as the modified incomplete LU decomposition based on 13 stencil points. A dynamical selection procedure is used for time steps. Note that the mesh equations for x and y can be solved separately and one coefficient matrix and preconditioner calculated at a time step can be then used in solving both equations and estimating the errors, which is needed for the time step selection.

Numerical results were presented for two test problems having analytical solutions. The effectiveness of the proposed method for controlling mesh concentration was demonstrated. It was also found that a suitable bound on the size of time steps is essential for integrating the MMPDE in order to obtain convergent meshes for steady-state problems and smoothly moving meshes for time dependent problems. For time dependent problems, we used an alternating solution procedure for the extended system consisting of the physical and mesh PDEs. Use of more sub-steps in integrating the MMPDE often produces meshes with smoother movement.

Finally, a two-level mesh movement strategy was discussed. The numerical results showed that the overhead of mesh generation in the moving mesh method can be significantly reduced when combined with this strategy. The two-level mesh movement also leads to a better convergence rate and allows for use of larger time steps in integrating the physical PDE than the one-level mesh movement.

ACKNOWLEDGMENTS

The author thanks Weiming Cao for his useful comments on this work. This work was supported in part by the NSF under Grants DMS-9626107 and DMS-0074240.

REFERENCES

1. G. Beckett and J. A. Mackenzie, Convergence analysis of finite-difference approximations on equidistributed grids to a singularly perturbed boundary value problems, *Appl. Numer. Math.* **35**, 109 (2000).
2. G. Beckett, J. A. Mackenzie, A. Ramage, and D. M. Sloan, On the numerical solution of one-dimensional PDEs using adaptive methods based on equidistribution, *J. Comput. Phys.*, in press.
3. J. U. Brackbill and J. S. Saltzman, Adaptive zoning for singular problems in two dimensions, *J. Comput. Phys.* **46**, 342 (1982).
4. W. Cao, W. Huang, and R. D. Russell, An r-adaptive finite element method based upon moving mesh PDEs, *J. Comput. Phys.* **149**, 221 (1999).
5. W. Cao, W. Huang, and R. D. Russell, A study of monitor functions for two dimensional adaptive mesh generation, *SIAM J. Sci. Comput.* **20**, 1978 (1999).
6. W. Cao, W. Huang, and R. D. Russell, Comparison of two-dimensional r-adaptive finite element methods using various error indicators, *Math. Comput. Simulation*, in press.
7. J. R. Cash, Diagonally implicit Runge–Kutta formulae with error estimates, *J. Inst. Math. Appl.* **24**, 293 (1979).
8. A. S. Dvinsky, Adaptive grid generation from harmonic maps on riemannian manifolds, *J. Comput. Phys.* **95**, 450 (1991).
9. B. H. Fiedler and R. J. Trapp, A fast dynamic grid adaption scheme for meteorological flows, *Mon. Weather Rev.* **121**, 2879 (1993).
10. G. H. Golub and C. F. van Loan, *Matrix Computation* (Johns Hopkins Press, Baltimore, 1983).
11. R. Hagmeeijer, Grid adaption based on modified anisotropic diffusion equations formulated in the parametric domain, *J. Comput. Phys.* **115**, 169 (1994).
12. D. F. Hawken, J. J. Gottlieb, and J. S. Hansen, Review of some adaptive node-movement techniques in finite element and finite difference solutions of PDEs, *J. Comput. Phys.* **95**, 254 (1991).
13. W. Huang, Y. Ren, and R. D. Russell, Moving mesh methods based on moving mesh partial differential equations, *J. Comput. Phys.* **113**, 279 (1994).
14. W. Huang, Y. Ren, and R. D. Russell, Moving mesh partial differential equations (MMPDEs) based upon the equidistribution principle, *SIAM J. Numer. Anal.* **31**, 709 (1994).
15. W. Huang and R. D. Russell, A high dimensional moving mesh strategy, *Appl. Numer. Math.* **26**, 63 (1997).
16. W. Huang and R. D. Russell, Moving mesh strategy based upon a gradient flow equation for two dimensional problems, *SIAM J. Sci. Comput.* **20**, 998 (1999).
17. W. Huang, L. Zheng, and X. Zhan, Adaptive moving mesh methods for simulating groundwater problems with sharp moving fronts (submitted for publication).
18. S. A. Jordan and M. L. Spaulding, A fast algorithm for grid generation, *J. Comput. Phys.* **104**, 118 (1993).
19. J. S. Mathur and S. K. Chakrabartty, An approximate factorization scheme for elliptic grid generation with control functions, *Numer. Meth. PDEs* **10**, 703 (1994).
20. L. S. Mulholland, W. Huang, and D. M. Sloan, Pseudospectral solution of near-singular problems using numerical coordinate transformations based on adaptivity, *SIAM J. Sci. Comput.* **19**, 1261 (1998).
21. G. E. Schneider and M. Zedan, A modified strongly implicit procedure for the numerical solution of field problems, *Numer. Heat Transfer* **4**, 1 (1981).

22. J. M. Stockie, J. A. Mackenzie, and R. D. Russell, A moving mesh method for one dimensional hyperbolic conservation law, *SIAM J. Sci. Comput.* **22**, 1791 (2000).

23. H. L. Stone, Iterative solution of implicit approximation of multidimensional partial differential equations, *SIAM J. Numer. Anal.* **5**, 530 (1968).

24. Z. U. A. Warsi, Conservation form of the Navier–Stokes equations in general non-steady coordinates, *AIAA J.* **19**, 240 (1981).

25. A. Winslow, Numerical solution of the quasi-linear poisson equation in a nonuniform triangle mesh, *J. Comput. Phys.* **1**, 149 (1967).