Journal of Computational Physics **170,** 562–588 (2001)
doi:10.1006/jcph.2001.6749, available online at http://www.idealibrary.com on IDEAL®

Moving Mesh Methods in Multiple Dimensions Based on Harmonic Maps

Ruo Li,* Tao Tang,† and Pingwen Zhang*

**School of Mathematical Sciences, Peking University, Beijing 100871, China; and †Department of Mathematics, The Hong Kong Baptist University, Kowloon Tong, Hong Kong*
E-mail: rli@sxx0.math.pku.edu.cn; ttang@math.hkbu.edu.hk; pzhang@sxx0.math.pku.edu.cn

Received February 14, 2000; revised February 16, 2001

In practice, there are three types of adaptive methods using the finite element approach, namely the *h-method*, *p-method*, and *r-method*. In the *h*-method, the overall method contains two parts, a solution algorithm and a mesh selection algorithm. These two parts are independent of each other in the sense that the change of the PDEs will affect the first part only. However, in some of the existing versions of the *r*-method (also known as the *moving mesh method*), these two parts are strongly associated with each other and as a result any change of the PDEs will result in the rewriting of the whole code. In this work, we will propose a *moving mesh method* which also contains two parts, a solution algorithm and a mesh-redistribution algorithm. Our efforts are to keep the advantages of the *r*-method (e.g., keep the number of nodes unchanged) and of the *h*-method (e.g., the two parts in the code are independent). A framework for adaptive meshes based on the Hamilton–Schoen–Yau theory was proposed by Dvinsky. In this work, we will extend Dvinsky's method to provide an efficient solver for the mesh-redistribution algorithm. The key idea is to construct the harmonic map between the physical space and a parameter space by an *iteration* procedure. Each iteration step is to move the mesh *closer* to the harmonic map. This procedure is simple and easy to program and also enables us to keep the map harmonic even after long times of numerical integration.The numerical schemes are applied to a number of test problems in two dimensions. It is observed that the mesh-redistribution strategy based on the harmonic maps adapts the mesh extremely well to the solution without producing skew elements for multi-dimensional computations.

Key Words: finite element methods; adaptive grids; harmonic map; partial differential equations.

0021-9991/01 $35.00

1. INTRODUCTION

We consider the time-dependent partial differential equations (PDEs) in a domain $\Omega \subset \mathbb{R}^n$

$$\vec{u}_t = L(\vec{u}) \quad \text{in } \Omega \times [0, T] \tag{1.1}$$

with boundary and initial conditions

$$B\vec{u}|_{\partial\Omega} = \vec{u}_b \quad \text{in } \partial\Omega \times [0, T] \tag{1.2}$$

$$\vec{u}|_{t=0} = \vec{u}_0 \quad \text{in } \Omega. \tag{1.3}$$

The difficulty of the problem appears when there are very rapid variations or sharp layers in the solution. Adaptive methods are powerful in resolving these kinds of difficulties by increasing the solution accuracy and decreasing the cost of numerical computations. Applications of the adaptive methods have been extended to many areas of research such as computational fluid dynamics [5, 6, 17, 19, 32], computational finance [34] and grid generation [21, 27].

Mesh adaptation can be static or dynamic (moving). In the static case a new mesh is generated at a fixed time intervals. In a dynamic or a moving mesh adaptation [1, 13, 23, 29, 33], the mesh changes continuously in space and time to adapt to the dynamic changes of a time-evolving solutions. It is a challenging problem to generate an efficient moving mesh in two or more dimensions, especially when the underlying solution develops complicated structures and becomes singular or nearly singular. In practice, there are three types of adaptive method using finite element approach, namely the *h-method* (mesh refinement), the *p-method* (order enrichment), and the *r-method* (mesh motion). The *r-method*, which will be studied in this work, is also known as *moving mesh method*. The earliest work on adaptive methods, based on moving finite element approach (MFEM) was done by Miller [29, 30]. While the MFEM has been subject to some criticism because of its complexity and sensitivity with respect to certain user defined input parameters [18], proper choice of these parameters unquestionably leads to an efficient method. However, one disadvantage of some existing moving mesh methods is that they mix the mesh-redistribution algorithm and the solution algorithm together and as a result any change of the given PDE (1.1) will lead to the rewriting of the whole code. Moreover, even if the given PDEs are linear the resulting system of differential equations are strongly nonlinear; see, e.g., [11, 24, 35].

In this work, we will provide a moving finite element scheme which keeps the main advantages of the *h*- and *r*-methods: the efficiency of the moving mesh methods and the simplicity of the *h*-method. To this end, we will follow the *h*-method to form our scheme with two independent parts: a mesh-redistribution algorithm and a solution algorithm. The second part will be independent of the first one, which can be any of the standard codes for the given PDEs. Now the key question is how to make the first part efficient and robust? Several moving mesh techniques have been introduced in the past, in which the most advocated method is the one based on solving elliptic PDEs first proposed by Winslow [41]. Winslow's formulation requires the solution of a nonlinear, Poisson-like equation to generate a mapping from a regular domain in a parameter space Ω_c to an irregularly shaped domain in physical space Ω. By connecting points in the physical space corresponding to discrete points in the parameter space, the physical domain can be covered with a computation mesh suitable for the solution of finite difference/element equations. There have

been also many applications and extensions of Winslow's method; see, e.g., Brackbill and Saltzman [8, 9], Russell and co-workers [10, 22], Ren and Wang [36], and Thompson *et al.* [40]. In these extensions the mesh is generated by using the *variational approach.* Specifically, the mesh map is provided by the minimizer of a functional of the following form

$$E(\vec{\xi}) = \frac{1}{2}\sum_k \int_\Omega \frac{1}{\omega}\sum_i \left(\frac{\partial \xi^k}{\partial x^i}\right)^2 d\vec{x}, \tag{1.4}$$

where the positive function ω is a weighted function depending on the physical solution to be adapted. The variational mesh is determined by the Euler–Lagrange equation of the above functional:

$$\sum_i \frac{\partial}{\partial x^i}\left(\frac{1}{\omega}\frac{\partial \xi^k}{\partial x^i}\right) = 0. \tag{1.5}$$

The solution of the above system will be taken as the map between the physical domain and the logical domain. As an example, let us denote as $(x(\zeta, \eta), y(\zeta, \eta))$ the mesh map in two dimensions. Here (ζ, η) are the computational coordinates. In 2-D, the functional (1.4) becomes

$$E[\zeta, \eta] = \frac{1}{2}\int_\Omega \frac{1}{\omega}(|\nabla\zeta|^2 + |\nabla\eta|^2)\, dxdy, \tag{1.6}$$

where $\nabla := (\partial_x, \partial_y)$. In this case, the corresponding Euler–Lagrange equation is given by

$$\nabla\cdot\left(\frac{1}{\omega}\nabla\zeta\right) = 0, \quad \nabla\cdot\left(\frac{1}{\omega}\nabla\eta\right) = 0. \tag{1.7}$$

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

Brackbill and Saltzman [8] formulated the grid equations in the variational form (1.4) to produce satisfactory mesh concentration while maintaining relatively good smoothness and orthogonality. Their approach has become one of the most popular methods for mesh generation and adaptation. In [9], Brackbill incorporates an efficient directional control into the mesh adaptation, thereby improving both the accuracy and efficiency of the numerical schemes.

Dvinsky [15] suggests that harmonic function theory may provide a general framework for developing useful mesh generators. His method can be viewed as a generalization and extension of Winslow's method. However, unlike most other generalizations which add terms or functionals to the basic Winslow grid generator, his approach uses a single functional to accomplish the adaptive mapping. The critical points of this functional are *harmonic maps.* Meshes obtained by Dvinsky's method enjoy desirable properties of harmonic maps, particularly regularity, or smoothness.

Motivated by the work of Dvinsky, a moving mesh strategy will be proposed and studied in this work. Our approach is similar to Dvinsky's scheme, except that (a) a more efficient and robust mesh-redistribution algorithm will be designed and (b) a finite element approach

rather than a finite difference discretization will be used. In part (a), we make sure that the map between the logical and the physical domains are kept harmonic, even after long time of numerical integration. To this end, we construct the harmonic map between the physical mesh and the logical mesh by an *iteration* procedure. Each iteration step is to move the mesh *closer* to the harmonic map. The idea of iteration mapping was proposed by Tang and Trummer [39], Liu and Tang [28], and Ren and Wang [36]. Each iteration consists of three parts:

- (i) obtain the error of $\vec{\xi}$ between the Euler–Lagrange solution mesh and the fixed (initial) mesh,
- (ii) obtain the new location of $\vec{x}$ by using the error of $\vec{\xi}$, and
- (iii) update $\vec{u}$ on the new mesh.

It is well known that one of the advantages of finite element methods over other numerical methods for solving PDEs (such as finite difference methods and spectral approximations) is that it can handle irregular solution domains. However, this is not the only reason for using part (b) above. Apart from the flexibility of solution domains, finite element approach is found to be more reliable and efficient than finite difference method for mesh adaptation. In particular, with finite element method we can avoid the standard interpolation procedure used in the existing adaptive codes. After obtaining the numerical approximations in a given time level, we begin to move more grid points to the physical relevant regions, i.e., part (ii) above. In part (iii) above, the essential assumption for updating $\vec{u}$ on the new mesh is that the *surface of* $\vec{u}$ *on* Ω *will be kept unchanged.*

The paper is organized as follows. In Section 2, we first discuss harmonic function theory and its application to mesh generation. Some general discussion on our numerical scheme will be given in Section 3. The detail implementation of the numerical scheme is described in Section 4. Numerical experiments on mesh generation and on solving PDEs with large solution variations are reported in Sections 5 and 6, respectively. In Sections 7 and 8, we discuss the difference between the present moving mesh code and some existing moving mesh codes.

2. HARMONIC MAPS BETWEEN RIEMANNIAN MANIFOLDS

Let Ω and Ω_c be compact Riemannian manifolds of dimension n with metric tensors d_{ij} and $\tau_{\alpha\beta}$ in some local coordinates $\vec{x}$ and $\vec{\xi}$, respectively. Following Dvinsky [15] and Brackbill [9], we define the energy for a map $\vec{\xi} = \vec{\xi}(\vec{x})$ as

$$E(\vec{\xi}) = \frac{1}{2}\int \sqrt{d} d^{ij} r_{\alpha\beta} \frac{\partial \xi^\alpha}{\partial x^i}\frac{\partial \xi^\beta}{\partial x^j}\, d\vec{x}, \tag{2.1}$$

where $d = \det(d_{ij})$, $(d_{ij}) = (d^{ij})^{-1}$, and the standard summation convention is assumed. The Euler–Lagrange equations, whose solution minimizes the above energy, are given by

$$\frac{1}{\sqrt{d}}\frac{\partial}{\partial x^i}\sqrt{d} d^{ij}\frac{\partial \xi^k}{\partial x^j} + d^{ij}\Gamma^k_{\beta\gamma}\frac{\partial \xi^\beta}{\partial x^i}\frac{\partial \xi^\gamma}{\partial x^j} = 0, \tag{2.2}$$

where $\Gamma^k_{\beta\gamma}$ is the Christoffel symbol of the second kind, defined by

$$\Gamma^k_{\beta\gamma} = \frac{1}{2} r^{k\lambda}\left[\frac{\partial r_{\lambda\beta}}{\partial \xi^\gamma} + \frac{\partial r_{\lambda\gamma}}{\partial \xi^\beta} - \frac{\partial r_{\beta\gamma}}{\partial \xi^\lambda}\right].$$

Existence and uniqueness of the harmonic map are guaranteed when the Riemannian curvature of Ω_c is nonpositive and its boundary is convex (see Hamilton [20] and Schoen and Yau [37]). Since Ω_c is obtained by construction, both requirements can usually be satisfied. With a Euclidean metric, $\Gamma^k_{\beta\gamma} = 0$, the Euler–Lagrange equations become

$$\frac{\partial}{\partial x^i}\sqrt{d}d^{ij}\frac{\partial \xi^k}{\partial x^j} = 0. \tag{2.3}$$

we emphasize that $d = \det(d_{ij}) = 1/\det(d^{ij})$. In 2-D, if we use the same notations as used in the last section, then the energy functional is given by

$$E[\zeta, \eta] = \frac{1}{2}\int_\Omega \sqrt{\det(D)}\left(\nabla\zeta^T D^{-1}\nabla\zeta + \nabla\eta^T D^{-1}\nabla_\eta\right) dxdy,$$

where $D^{-1} = (d^{ij})$ is a symmetric positive definite matrix depending on (x, y). In this case, the corresponding Euler–Lagrange equation is given by

$$\nabla\cdot(\sqrt{\det(D)}D^{-1}\nabla\zeta) = 0, \quad \nabla\cdot(\sqrt{\det(D)}D^{-1}\nabla\eta) = 0.$$

For ease of notation, we let $G^{ij} = \sqrt{d}d^{ij}$. The inverse of (G^{ij}) is called *monitor functions*. Therefore, the Euler–Lagrange equations, with Euclidean metric for the logical domain Ω_c, are given by

$$\frac{\partial}{\partial x^i}\left(G^{ij}\frac{\partial \xi^k}{\partial x^j}\right) = 0. \tag{2.4}$$

The simplest harmonic mapping is obtained by assuming Euclidean metric in both domains, which gives

$$G^{ij} = \delta^{ij}, \tag{2.5}$$

with δ^{ij} the Kronecker delta. In [9], it is shown that the generator suggested by Winslow (1.4) forms a harmonic mapping and hence the Hamilton–Schoen–Yau existence and uniqueness theories ([20, 37]) can be applied. The equations (1.5) and (2.4) are identical when the metric in (2.1) and ω in (1.4) are related by

$$G^{ij} = \delta^{ij}/\omega. \tag{2.6}$$

Another monitor function proposed by Dvinsky [15] is

$$(d^{ij}) = I + \frac{f(F)}{\|F\|^2}\nabla F\cdot\nabla F^T,$$

where $f(F)$ is a function of the distance from a given point to the given curve $F(\vec{x}) = 0$. More general forms of monitor functions are proposed and studied in Brackbill [9] and Huang and Russell [22].

To solve the Euler–Lagrange equations numerically, one usually interchanges dependent and independent variables. The solution of (2.4) requires evaluating derivatives of $\vec{\xi}$ with respect to the physical coordinates $\vec{x}$. In moving mesh computation, however, one usually specifies the logical arrangement of grid points and computes the physical coordinates of

the grid points. In other words, we solve for $\vec{x}(\vec{\xi})$, the inverse mapping of $\vec{\xi}$, because it directly defines the mesh on Ω. For the variational approach (1.4)–(1.5), the components of the physical coordinates are governed by

$$g^{ij}\frac{\partial}{\partial \xi^i}\omega\frac{\partial x^k}{\partial \xi^j} = 0, \tag{2.7}$$

where

$$g^{ij} = \frac{\partial \xi^i}{\partial x^\alpha}\frac{\partial \xi^j}{\partial x^\alpha}.$$

The detailed derivation for the equation (2.7) can be found in [9].

In developing the theory of harmonic maps, a deformation from a given homomorphism to the harmonic map by the heat equations has been investigated by several researchers; see, e.g., [16]. Solving the heat equation

$$\frac{\partial \xi^k}{\partial \mu} = \frac{\partial}{\partial x^i}\left(G^{ij}\frac{\partial \xi^k}{\partial x^j}\right) \quad \text{to } \mu \to \infty \tag{2.8}$$

will lead to the harmonic map defined by (2.4). The reason for not solving the elliptic system (2.4) directly but instead solving the heat equation is again to provide a useful way to obtain a map from $\vec{\xi}$ to $\vec{x}$, as to be demonstrated below. Using the identity

$$\frac{\partial^2 \xi^k}{\partial x^i \partial x^j} = -\frac{\partial \xi^k}{\partial x^\beta}\frac{\partial^2 x^\beta}{\partial \xi^\gamma \partial \xi^\delta}\frac{\partial \xi^\gamma}{\partial x^i}\frac{\partial \xi^\delta}{\partial x^j}$$

and noting that for an arbitrary function $f(\vec{x}(\vec{\xi}, \mu), \mu)$

$$\left.\frac{\partial f}{\partial \mu}\right|_{\text{fixed } \vec{x}} = \left.\frac{\partial f}{\partial \mu}\right|_{\text{fixed } \vec{\xi}} - \frac{\partial f}{\partial x^k}\frac{\partial x_k}{\partial \mu},$$

we can obtain from (2.8) that

$$\begin{aligned}
-\frac{\partial x^l}{\partial \mu} &= \frac{\partial \xi^k}{\partial \mu}\frac{\partial x^l}{\partial \xi^k} \\
&= \frac{\partial}{\partial x^i}G^{ij}\frac{\partial \xi^k}{\partial x^j}\frac{\partial x^l}{\partial \xi^k} + G^{ij}\frac{\partial^2 \xi^k}{\partial x^i \partial x^j}\frac{\partial x^l}{\partial \xi^k} \\
&= \frac{\partial}{\partial x^i}G^{ij}\delta^{lj} - G^{ij}\frac{\partial \xi^k}{\partial x^\beta}\frac{\partial^2 x^\beta}{\partial \xi^\gamma \partial \xi^\delta}\frac{\partial \xi^\gamma}{\partial x^i}\frac{\partial \xi^\delta}{\partial x^j}\frac{\partial x^l}{\partial \xi^k} \\
&= \frac{\partial}{\partial x^i}G^{il} - G^{ij}\frac{\partial^2 x^\beta}{\partial \xi^\gamma \partial \xi^\delta}\frac{\partial \xi^\gamma}{\partial x^i}\frac{\partial \xi^\delta}{\partial x^j}\delta^{l\beta} \\
&= \frac{\partial}{\partial x^i}G^{il} - G^{ij}\frac{\partial^2 x^l}{\partial \xi^\gamma \partial \xi^\delta}\frac{\partial \xi^\gamma}{\partial x^i}\frac{\partial \xi^\delta}{\partial x^j},
\end{aligned}$$

where $\delta^{l\beta}$ is the Kronecker delta. By letting $\tilde{G}^{\gamma\delta} = G^{ij}\frac{\partial \xi^\gamma}{\partial x^i}\frac{\partial \xi^\delta}{\partial x^j}$, we obtain the equation

$$\frac{\partial x^l}{\partial \mu} = \tilde{G}^{\gamma\delta}\frac{\partial^2 x^l}{\partial \xi^\gamma \partial \xi^\delta} - \frac{\partial}{\partial x^i}G^{il}, \tag{2.9}$$

or equivalently

$$\frac{\partial x^l}{\partial \mu} = \tilde{G}^{\gamma\delta} \frac{\partial^2 x^l}{\partial \xi^\gamma \partial \xi^\delta} - \frac{\partial \xi^k}{\partial x^i} \frac{\partial G^{il}}{\partial \xi^k}. \tag{2.10}$$

The equation (2.10) gives the desired map from $\vec{\xi}$ to $\vec{x}$. In our computation, the equation (2.9) will lead to a finite element formula for computing such a map; see (4.13).

Since the fundamental work of Eell and Sampson [16], harmonic maps have attracted considerable attention from both mathematicians and physicists. Dvinsky is the first to note the practical importance of using harmonic map theory for mesh adaptation [15]. A good feature of the adaptive methods based on harmonic mapping is that existence, uniqueness, and nonsingularity for the continuous map can be guaranteed from the theory of harmonic maps. The existence and uniqueness of harmonic maps are established by Hamilton [20] and Schoen and Yau [37]; the existence of the solution for the problem (2.8) is addressed by Eell and Sampson [16]; The singularity of three-dimensional harmonic maps is discussed by Liao and Smale [25, 26]. Such theoretical guarantees are rare in the field of adaptive mesh generation. In a recent work of Bertalmio *et al.* [7], harmonic maps, together with level-set techniques, have been used successfully to solve variational problems and PDEs on implicit surfaces, with particular application to image processing and computer graphics.

3. THE FRAME OF OUR NUMERICAL SCHEME

In this work, we will employ finite element methods together with moving mesh strategy to solve problem (1.1)–(1.3). The non-uniform mesh of finite element methods is more flexible than that of finite difference methods that enables us to handle more complicated physical domains. Moreover, as to be demonstrated in next section, with finite element methods we can avoid the standard interpolation procedure used in most existing adaptive codes.

To solve problem (1.1)–(1.3), we will separate the computation into two parts: mesh-moving and time-stepping. The mesh-moving is a procedure of *iteration* to construct the harmonic map between the physical mesh and the logical mesh. Each iteration step is to move the mesh *closer* to the harmonic map. In the process of the numerical computation, we always keep the initial mesh in the logical domain fixed. This mesh is not used to solve any PDEs, but its error with the solution of the Poisson equation (2.4) is used to move the mesh in the physical domain. More precisely, in the first step we choose a convex domain Ω_c as the logical domain on which an initial mesh will be constructed. By solving the Poisson equation $\Delta\vec{\xi} = 0$ with some Dirichlet boundary condition, we obtain a mesh in the logical domain. Once this initial mesh is obtained, it will be kept unchanged throughout the computation. The role of this initial mesh in Ω_c, denoted by $\vec{\xi}^{(0)}$, is used as a reference grid only. Once the solution u is computed at time step $t = t_n$, the inverse matrix of the monitor, G^{ij} (which in general depends on u), can be updated. By solving the Euler–Lagrange equation (2.4), we will obtain a mesh in the logical domain, denoted by $\vec{\xi}^*$. If the difference between this $\vec{\xi}^*$ and the initial mesh $\vec{\xi}^{(0)}$ is not small, we move the mesh in the physical space and obtain the updated values for u in the resulting new grid based on the following principles:

- (a) obtain the error between the solution of (2.4) and the fixed (initial) mesh in the logical domain,

- (b) obtain the direction and the magnitude of the movement for $\vec{x}$ by using the error of $\vec{\xi}$, and
- (c) update $\vec{u}$ on the new grid by solving a system of ODEs, see (4.12).

This procedure is *repeated* until the difference between $\vec{\xi}^*$ and the initial mesh $\vec{\xi}^{(0)}$ is sufficiently small. Then we can use some appropriate numerical methods, with the updated mesh in the physical space, to solve the given problems to obtain solution for $t = t_{n+1}$. This procedure is also illustrated by the following flowchart:

Solve the given PDE problems for $t = t_n$

$\Downarrow$

Update the mesh and solution based on an iteration procedure

$\Downarrow$

Solve the given PDE problems for $t = t_{n+1}$

The first step above involves a solution algorithm, which is essentially irrelevant with the second part. The solution algorithm can be any standard finite element codes or a semi-discretized finite element method in conjunction with the method of line. In other words, we can clearly separate mesh-moving and time-forwarding so that the code is easy to program: In the time-forwarding part, the numerical methods used have no difference with those without mesh redistribution; and for different PDE problems the only possible change in the mesh-redistribution part of the codes is to change the monitor function. It is relevant to point out that some ideas of the so-called *moving space–time finite element method* [4] may be implemented in our moving mesh scheme.

In second step above, the iteration procedure is given by the following algorithm:

ALGORITHM 1.

(i) Solve the Euler–Lagrange equation (2.4) to obtain $\vec{\xi}^*$.

(ii) Judge if L_2-norm of $\vec{\xi}^* - \vec{\xi}^{(0)}$ is small. if yes, the iteration is over. otherwise, do (iii)–(vi).

(iii) Using the difference $\vec{\xi}^* - \vec{\xi}^{(0)}$ to compute the mesh-moving vector $\delta\vec{x}$.

(iv) Move the mesh to a new location based on the result in (iii).

(v) Update the numerical approximations at new grids obtained in (iv).

(vi) Go to (i).

In part (i), two methods will be used in our computation. Method I is to solve Eq. (2.9). This method works in space $\mathcal{H}$ formed by all homomorphism from Ω to Ω_c. The energy functional is defined on $\mathcal{H}$. Equation (2.9) goes down along the direction of the negative gradient of $E(\vec{\xi})$: see, e.g., [16]. Method II is to use the harmonic map itself. After solving Eq. (2.4), we can obtain the harmonic map $\vec{\xi}$ from Ω to Ω_c. Then we interpolate the nodes of the new mesh as the images of the nodes for the logical mesh of $\vec{\xi}^{-1}$. The detail implementation for both methods will be given in next section. It is found that both Methods I and II work well for the test problems in Sections 6 and 7. The difference is that Method II is easier than Method I in coding.

In part (ii), in order to guarantee the quality of the harmonic map, we repeat the mesh-moving process until the L_2-norm for the distance between the solution of Eq. (2.4) and the initial logical mesh $\vec{\xi}^{(0)}$ is smaller than a preassigned tolerance `TOL`.

In part (V), the node values at new grids are updated based on the assumption that the *surface* of $\vec{u}$ on Ω will be kept unchanged at each fixed time step. Based on this principle,

the approximate solutions are re-distributed nicely on new grids based on a linear system of ordinary differential equations (ODEs), see Eq. (4.12).

4. THE NUMERICAL SCHEME

In this section, we will discuss the detail numerical procedures for our moving mesh scheme. The following steps provide the key ingredients for our numerical scheme.

4.1. *Prepare the Initial Mesh on the Logical Domain*

Let the physical domain Ω be triangularized into some simple finite elements, denoted by T. The finite element space is chosen as a *linear finite element* space. Let $X = (X_i)$ be the nodes, and $\Phi = (\Phi^j)$ be basis functions such that $\Phi^j(X_i) = \delta^{ij}$, where δ^{ij} is the Kronecker delta. We also choose a convex domain Ω_c as the logical domain. By solving the Poisson equation

$$\Delta\vec{\xi} = 0, \quad \vec{x} \in \Omega \tag{4.1}$$

with Dirichlet boundary condition

$$\vec{\xi}|_{\partial\Omega} = \vec{\xi}_b, \tag{4.2}$$

we obtain a mesh T_c in the logical domain, with nodes $\mathcal{A} = (\mathcal{A}_i)$.

4.2. *Prepare the Initial Mesh on the Physical Domain*

Once an initial mesh $\vec{\xi}^{(0)} = T_c$ is obtained as described in the above section, the initial adaptive mesh on the physical domain Ω can be computed as the numerical solution of the inverse map of (2.4). With the choice of Winslow's monitor function (2.6), the equation for the inverse map is given by (2.7). For more general choices of G_{ij}, the equations governing the inverse map can also be formulated, by following the similar derivations in [22]. We can also avoid deriving and directly using the inverse map equations. This can be done by using steps (P_1) and (P_2), described in the following section.

In the case where the initial function $\vec{u}_0$ is stiff (or near singular), we use a method based on continuation on the initial function. Without lose of generality, let $G = (G^{ij})$ be the function of $\vec{u}$ only, i.e. $G = G(\vec{u})$. Then we decompose (2.4) at the initial step into the following iteration procedure:

$$\nabla \cdot (G(\tau_{n-1}\vec{u}_0(\vec{x}_{n-1}))\nabla\vec{\xi}) = 0. \quad n \geq 1 \tag{4.3}$$

To start with, we let $\tau_0 = 0$ and $\vec{x}_0$ be the uniform mesh. The inverse map of (4.3) is defined as $\vec{x}_n$ which can be computed by some standard methods mentioned above. By increasing $\{\tau_n\}$ from 0 to 1, we will end up with a desired initial mesh on Ω.

4.3. *Mesh-Moving*

Suppose now we have obtained the value of $U_i = \vec{u}(X_i)$ on the current nodes X_i at a time step $t = t_n$. We now need to obtain the new location of the nodes $X^* = (X_i^*)$ and the

new value of $U^* = (U_i^*)$ on the new nodes. The operation can be divided into the following three parts if we use Method II.

(P_1) *Obtain the error of* $\vec{\xi}$. We first solve the following generalized Poisson equation

$$\frac{\partial}{\partial x^i}\left(G^{ij}\frac{\partial \xi^k}{\partial x^j}\right) = 0 \tag{4.4}$$

with the boundary condition (4.2). By doing so, we can obtain a new logical mesh T_c^* with its nodes $\mathcal{A}^*$. We are interested in the error of $\vec{\xi}$:

$$\delta\mathcal{A} = \mathcal{A} - \mathcal{A}^*.$$

The above error function will be used in the next step to predict the movement of the numerical grid in the physical space Ω, see (4.6) below.

(P_2) *Obtain the movement of* $\vec{x}$. For a given element E in T, with X_{E_k}, $0 \le k \le n$ as its vertexes, the piecewise linear map from $V_{T_c^*}(\Omega_c)$ to $V_T(\Omega)$ such that $\mathcal{A}_i^* \mapsto X_i$ has constant gradient on E and satisfies the linear system

$$\begin{pmatrix} \mathcal{A}_{E_1}^{*,1} - \mathcal{A}_{E_0}^{*,1} & \mathcal{A}_{E_2}^{*,1} - \mathcal{A}_{E_0}^{*,1} & \cdots & \mathcal{A}_{E_n}^{*,n} - \mathcal{A}_{E_0}^{*,n} \\ \mathcal{A}_{E_1}^{*,2} - \mathcal{A}_{E_0}^{*,2} & \mathcal{A}_{E_2}^{*,2} - \mathcal{A}_{E_0}^{*,2} & \cdots & \mathcal{A}_{E_n}^{*,2} - \mathcal{A}_{E_0}^{*,2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{A}_{E_1}^{*,n} - \mathcal{A}_{E_0}^{*,n} & \mathcal{A}_{E_2}^{*,n} - \mathcal{A}_{E_0}^{*,n} & \cdots & \mathcal{A}_{E_n}^{*,n} - \mathcal{A}_{E_0}^{*,n} \end{pmatrix} \begin{pmatrix} \frac{\partial x^1}{\partial \xi^1} & \frac{\partial x^1}{\partial \xi^2} & \cdots & \frac{\partial x^1}{\partial \xi^n} \\ \frac{\partial x^2}{\partial \xi^1} & \frac{\partial x^2}{\partial \xi^2} & \cdots & \frac{\partial x^2}{\partial \xi^n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x^n}{\partial \xi^1} & \frac{\partial x^n}{\partial \xi^2} & \cdots & \frac{\partial x^n}{\partial \xi^n} \end{pmatrix}$$

$$= \begin{pmatrix} X_{E_1}^1 - X_{E_0}^1 & X_{E_2}^1 - X_{E_0}^1 & \cdots & X_{E_n}^1 - X_{E_0}^1 \\ X_{E_1}^2 - X_{E_0}^2 & X_{E_2}^2 - X_{E_0}^2 & \cdots & X_{E_n}^2 - X_{E_0}^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_{E_1}^n - X_{E_0}^n & X_{E_2}^n - X_{E_0}^n & \cdots & X_{E_n}^n - X_{E_0}^n \end{pmatrix}. \tag{4.5}$$

Solving the above linear system gives $\partial\vec{x}/\partial\xi$ in E. If we take the volume of the element as the weight, the weighted average error of X at the ith node is defined by

$$\delta X_i = \frac{\sum_{E\in T_i} |E| \left.\frac{\partial \vec{x}}{\partial \xi}\right|_{in\ E} \delta\mathcal{A}_i}{\sum_{E\in T_i} |E|} \tag{4.6}$$

in which $|E|$ is the volume of element E. It can be shown that the above volume-weighted average converges to a smooth solution in measure when the size of mesh goes to 0. The location of the nodes in the new mesh T^* on the physical domain is taken as

$$X^* = X + \tau\delta X,$$

in which τ is a parameter in $[0, 1]$. In our numerical experiments, it is found that the selection of τ is quite insensitive. One choice of τ is the following:

$$\tau = \min(0.5/\|\delta\mathcal{A}\|_2, 0.618).$$

In general, the 2-norm of δA decreases with a rate about $2/3$ in each iteration step after $||\delta\mathcal{A}||_2$ becomes small. However, when the solution $\vec{u}$ is very singular (e.g., very large gradients exist for the solution), the decrease of $||\delta\mathcal{A}||_2$ will become slow.

(P_3) *Update $\vec{u}$ on the new mesh.* Each element of T with X as its nodes corresponds uniquely to an element of $T^*(\tau)$ with $X + \tau\delta X$ as its nodes. There is also an affine map between the two elements. By combining all those affine maps from every element of $T^*(\tau)$ to T, we obtain a map from Ω_c to Ω piecewise affine. The surface of $\vec{u}$ on Ω will not move, though the nodes of the mesh will be moved to new locations. Then $\vec{u}$, as the function of $\vec{x}$ at time t_n, is independent on the parameter τ. That is,

$$\frac{\partial \vec{u}}{\partial \tau} = 0. \tag{4.7}$$

During the moving of the mesh, $\vec{u}$ is expressed as

$$\begin{aligned} \vec{u} &= \vec{u}(\vec{x}) \\ &= \vec{u}(\vec{x}, \tau). \end{aligned}$$

In the finite element space, $\vec{u}$ is expressed as

$$\vec{u} = U_i(\tau)\Phi^i(\vec{x}, \tau), \tag{4.8}$$

where $\Phi^i(\vec{x}, \tau)$ is the basis function of the finite element space at its node $X_i + \tau\delta X_i$. Direct computation gives

$$\frac{\partial \Phi^i(\vec{x}, \tau)}{\partial \tau} = -\frac{\partial \Phi^i(\vec{x}, \tau)}{\partial x^j}(\delta\vec{x})_j, \tag{4.9}$$

where $\delta\vec{x} := \delta X_i \Phi^i$. Differentiating $\vec{u}$ with respect to τ gives

$$\begin{aligned} 0 = \frac{\partial \vec{u}}{\partial \tau} &= \frac{\partial U_i}{\partial \tau}\Phi^i(\vec{x}, \tau) + U_i(\tau)\frac{\partial \Phi^i}{\partial \tau} \\ &= \frac{\partial U_i}{\partial \tau}\Phi^i(\vec{x}, \tau) - U_i(\tau)\frac{\partial \Phi^i}{\partial x^j}(\delta\vec{x})_j. \end{aligned} \tag{4.10}$$

Using the expression for $\vec{u}$ in the finite element space, i.e., (4.8), we obtain from the above result that

$$\frac{\partial U_i}{\partial \tau}\Phi^i(\vec{x}, \tau) - \nabla_{\vec{x}}\vec{u}\delta\vec{x} = 0. \tag{4.11}$$

Then the semi-discrete system for updating $\vec{u}$ follows from the above result:

$$\int_\Omega \left\{ \frac{\partial U_i}{\partial \tau}\Phi^i(\vec{x}, \tau) - \nabla_{\vec{x}}\vec{u}\delta\vec{x} \right\} v d\vec{x} = 0 \quad \forall v \in V_T(\Omega).$$

By letting v be the basis function of $V_T(\Omega)$, i.e., $v = \Phi^j(\vec{x}, \tau)$, we obtain a system of (linear) ODEs for U_i:

$$\int_\Omega \Phi^i \Phi^j d\vec{x} \frac{\partial U_i}{\partial \tau} = \int_\Omega \frac{\partial \Phi^i}{\partial x^k}(\delta\vec{x})_k \Phi^j d\vec{x} U_i(\tau). \tag{4.12}$$

We will solve the above ODE system with a three-stage Runge–Kutta scheme.

Remark. If Method I is used in this step, the only difference is how to obtain the error of $\vec{x}$. In this case, we express $\vec{x}$ with the basis function in the finite element space of the logical domain as

$$\frac{\partial x^l}{\partial \mu} = \frac{\partial X_i^l}{\partial \mu}\Phi_c^i,$$
$$x^l = X_i^l \Phi_c^i.$$

The above equations are substituted into the parabolic equation (2.9). We further assume that the functions G and $\tilde{G}$ are piecewise constant on the logical domain. We can then obtain the system for $\partial X/\partial \mu$ as

$$\begin{aligned}\int_{\Omega_c} \frac{\partial X_i^l}{\partial \mu}\Phi_c^i v d\vec{\xi} &= \int_{\Omega_c}\left\{\tilde{G}^{\gamma\delta}\frac{\partial^2 x^l}{\partial \xi^\gamma \partial \xi^\delta} - \frac{\partial}{\partial x^i}G^{il}\right\} v d\vec{\xi} \\ &= -\int_{\Omega_c}\left\{\tilde{G}^{\gamma\delta}\frac{\partial x^l}{\partial \xi^\gamma}\frac{\partial v}{\partial \xi^\delta} - G^{il}\frac{\partial \xi^k}{\partial x^i}\frac{\partial v}{\partial \xi^k}\right\} d\vec{\xi} \\ &= -\int_{\Omega_c}\left\{\tilde{G}^{\gamma\delta}X_k^l\frac{\partial \Phi_c^k}{\partial \xi^\gamma}\frac{\partial v}{\partial \xi^\delta} - G^{il}\frac{\partial \xi^k}{\partial x^i}\frac{\partial v}{\partial \xi^k}\right\} d\vec{\xi} \qquad (4.13)\end{aligned}$$

for $v \in V_{T_c,0}(\Omega_c)$. By letting v be the basis function of $V_{T_c,0}(\Omega_c)$, we obtain from the above result a linear system for $\partial X/\partial \mu$,

$$\int_{\Omega_c}\frac{\partial X_i^l}{\partial \mu}\Phi_c^i\Phi_c^j d\vec{\xi} = -\int_{\Omega_c}\left\{\tilde{G}^{\gamma\delta}X_k^l\frac{\partial \Phi_c^k}{\partial \xi^\gamma}\frac{\partial \Phi_c^l}{\partial \xi^\delta} - G^{il}\frac{\partial \xi^k}{\partial x^i}\frac{\partial \Phi_c^j}{\partial \xi^k}\right\} d\vec{\xi}. \qquad (4.14)$$

We take $\partial X/\partial \mu$ as the direction of the error for $\vec{x}$. The step length of mesh-moving is taken as the length of $\partial X/\partial \mu$.

4.4. *Time-Forwarding*

This step is trivial: it is irrelevant with the adaptive method and can be any appropriate finite element code. The following is one of the possible methods, which will be used in our numerical experiment sections. It follows from Eq. (1.1) that

$$\int_\Omega \{\vec{u}_t - L(\vec{u})\}\, v d\vec{x} = 0 \quad \forall v \in V_T(\Omega).$$

Using the expression for $\vec{u}$ in the finite element space, i.e., $\vec{u} = U_i(t)\Phi^i(\vec{x})$, and letting v be the basis function, we obtain

$$\int_\Omega \left\{\frac{\partial U_i}{\partial t}\Phi^i\Phi^j - L(\vec{u})\Phi^j\right\} d\vec{x} = 0, \qquad (4.15)$$

which is a systems of ODEs for $U_i(t)$. It can be solved by any efficient ODE solvers such as multi-stage Runge–Kutta schemes.

We point out again that the method based on (4.15) only serves for the numerical experiments in this paper. In fact, this step is very flexible: any available methods/codes for Eq. (1.1) can be employed in this step.

5. NUMERICAL EXPERIMENTS: MESH GENERATION

In Section 4.3 an adaptive grid generation procedure is proposed. In this step, one of the important issues requiring some attention is the monitor function G^{ij} used in Eq. (4.4). There are several excellent papers in this direction, including Brackbill and Saltzman, [8] and Cao *et al.* [10]. Due to the limitation of space, we will concentrate on the implementation of our numerical scheme proposed in last section and therefore will not discuss the issues of the monitor function. In this work we will just use the simple monitor function (2.6).

Several test problems have been computed and the numerical results indicate that our mesh generation procedure is indeed efficient and robust. Here we only report results for one example which was also tested by several other authors; see, e.g., [11].

EXAMPLE 5.1. As an example, the performance of the moving mesh technique is examined for the case of a solution domain with very rough boundaries. The adaptation function is chosen as $u(\vec{x}, t) = \tanh(50(x_1 - t))$, where $\vec{x} = (x_1, x_2)^T$.

An unstructured grid in the logical domain Ω_c is initially generated by using the scheme in Section 4.1. We choose the logical domain as a convex polygon having the same number of boundary segments as Ω and use a monitor function $G^{-1} = \sqrt{1 + 0.1\|\nabla u\|_2^2} I$. Using Section 4.2, we obtain the initial mesh in the physical domain Ω. The initial unstructured grids in Ω and Ω_c are displayed in Fig. 1. The moving grid at various time levels is also shown in Fig. 2. One can see that the generated mesh is satisfactory in the sense that it conforms very well to the adaptation function.

6. NUMERICAL EXPERIMENTS: SOLVING PDES

In this section we present some numerical examples to demonstrate the performance of our moving mesh finite-element methods for solving time dependent PDEs. We noticed that a recent paper of Cao *et al.* [11] provides a number of test problems and we will basically follow their examples, except the one with application to the Navier-Stokes equations. A

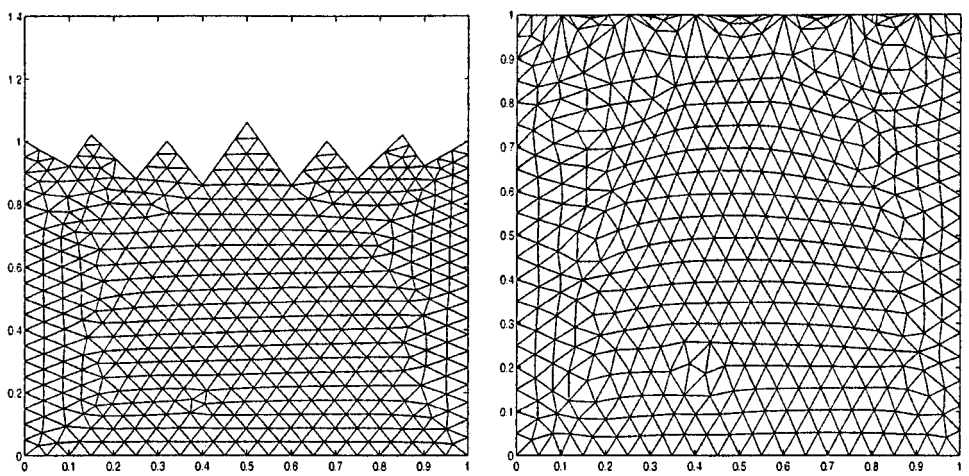

FIG. 1. The initial mesh in Ω (left) and Ω_c (right) for Example 5.1.

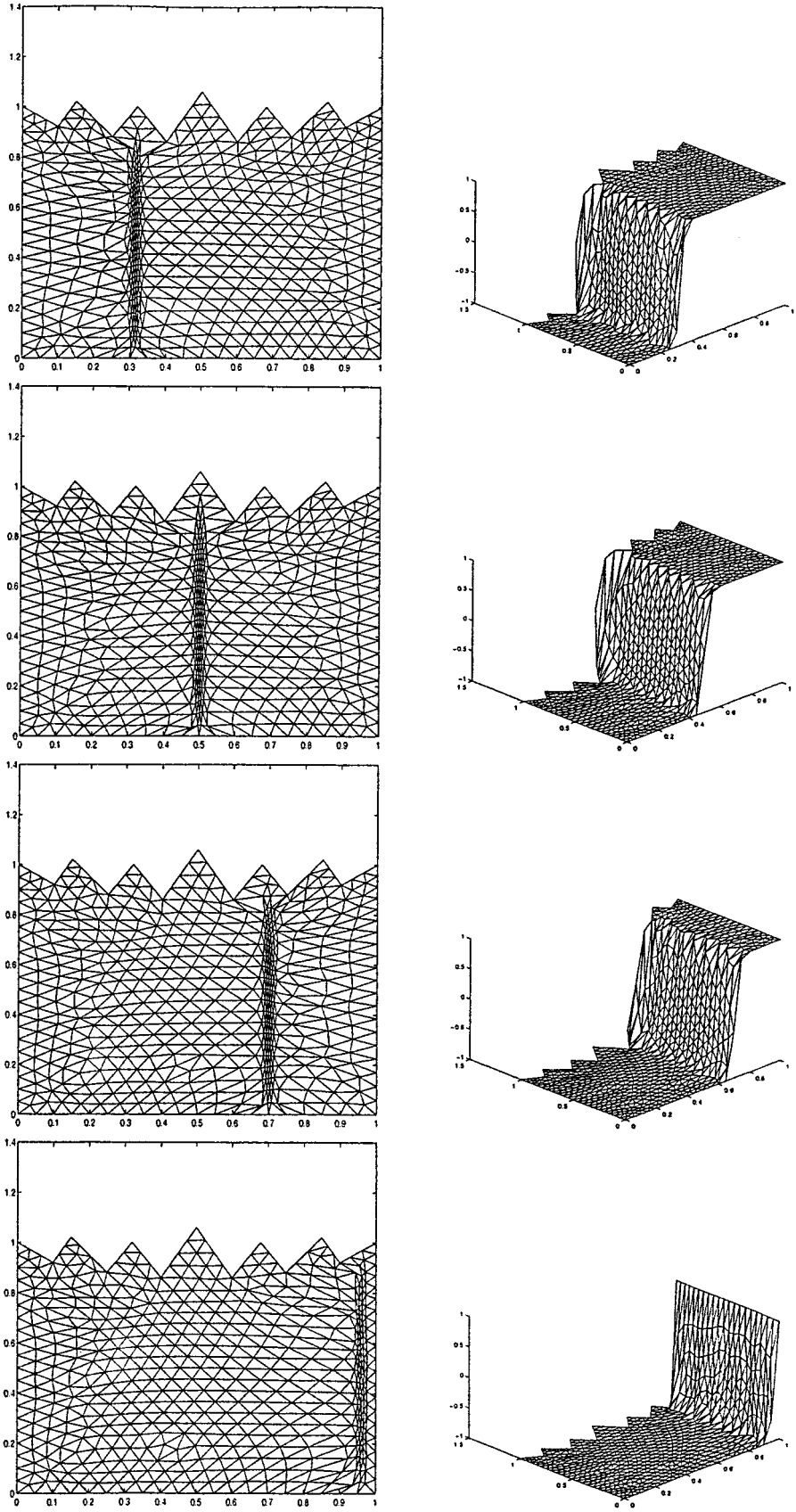

FIG. 2. The adaptive meshes and the numerical solutions at $t = 0.3$, 0.5, 0.7, and 0.97 for Example 5.1.

detailed study of the application of our moving mesh method to the Navier-Stokes equations will be provided in a future work.

In practice it is common to use some temporal or spatial *smoothing* on the monitor function or directly on the mesh map $\vec{x}$ to obtain smoother meshes, this was found to be extremely useful in the work of [11, 13, 38]. One of the reasons for using smoothing is to avoid very singular meshes and large approximation error around the stiff solution areas. Several smoothing techniques have been proposed to enhance the quality of the meshes. In this work, we propose a rather different smoothing procedure: First we interpolate the monitor function $M := G^{-1}$ from $L^2(\Omega)$ into $H_{1,h}(\Omega)$, namely from piecewise constant to piecewise linear, by the formula:

$$(\pi_h M)|_{\text{at } P} = \frac{\sum_{\tau: P \text{ is vertex of } \tau} M|_{\text{on } \tau} |\tau|}{\sum_{\tau: P \text{ is vertex of } \tau} |\tau|}. \tag{6.1}$$

Second we project it back into $L^2(\Omega)$ by the formula:

$$M|_{\text{on } \tau} = \frac{1}{n+1} \sum_{P \text{ is vertex of } \tau} (\pi_h M)_{\text{at } P}, \tag{6.2}$$

where n is the dimension of Ω. Our numerical experiments have shown that this smoothing for the monitor function (2.6) not only enhances the quality of the meshes but also increases the accuracy of the numerical approximations.

EXAMPLE 6.1. Consider the wave equation

$$\frac{\partial U}{\partial t} - y\frac{\partial U}{\partial x} + x\frac{\partial U}{\partial y} = 0$$

on the unit circle with initial value

$$U(\vec{x}, 0) = \begin{cases} e^{-32((x_1-1/2)^2+x_2^2)} & \text{if } (x_1 - 1/2)^2 + x_2^2 < 1/4, \\ e^{-32((x_1+1/2)^2+x_2^2)} & \text{if } (x_1 + 1/2)^2 + x_2^2 < 1/4, \\ 0 & \text{elsewhere} \end{cases}$$

and zero boundary condition.

The solution of Example 6.1 possesses a twin peak (of fixed shape) rotating counter-clockwise around the origin. A linear finite element discretization based upon the moving mesh scheme as described in Section 4 is applied. The initial mesh is obtained from a quasi-uniform triangulation with 1700 elements, as shown in Fig. 3. A fixed time step size $\delta t = 0.05$ is used for the integration of the ODE system (see Section 4.4). In the mesh redistribution part, the monitor function used is $\sqrt{1 + 6u^2 + \|\nabla u\|_2^2} I$.

This test problem has been considered by several authors; see, e.g., Baines [3], Davis and Flaherty [14], and Cao *et al.* [11], to test the quality of the meshes generated by adaptive schemes. As noted in [11] that some existing moving mesh techniques produce meshes with points sticking to the rotating peaks, causing the mesh to become increasingly skew until the computation eventually breaks down. From Fig. 3, it is clear that our moving mesh scheme has no such difficulty, and the mesh adapts extremely well to the solution without producing skew elements.

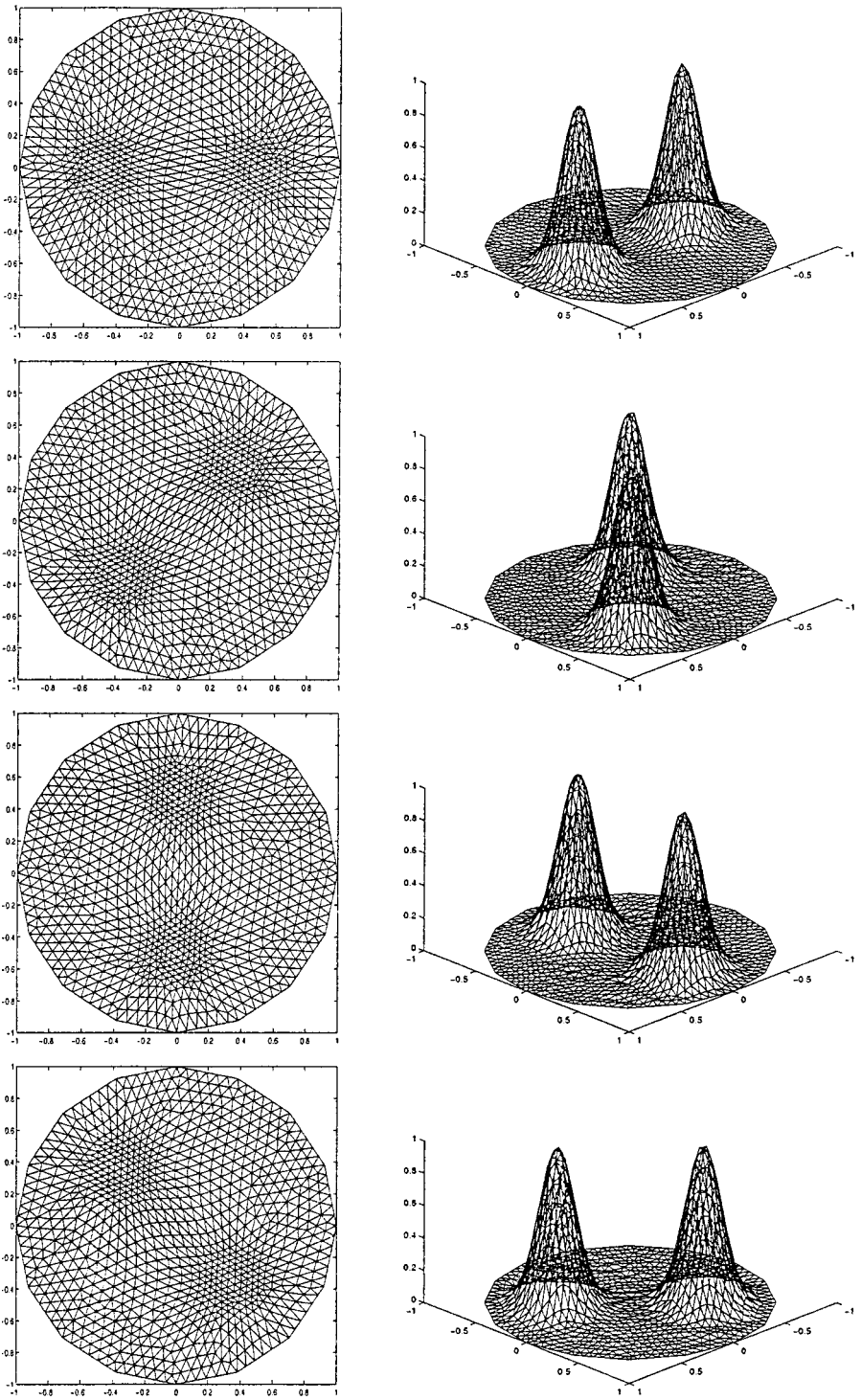

FIG. 3. The adaptive meshes and the numerical solutions at $t = 0, \pi/8, \pi/4$, and $3\pi/8$ for Example 6.1.

EXAMPLE 6.2. Our second example is to compute a moving oblique shock. The governing equation for it is the Burgers equation

$$\frac{\partial U}{\partial t} + UU_x + UU_y = a\Delta U$$

defined in the unit square $\Omega = (0, 1) \times (0, 1)$. The initial condition and Dirichlet boundary condition are chosen such that the exact solution to the underlying problem is

$$U(x, y; t) = (1 + \exp((x + y - t)/(2a))^{-1}.$$

In our computation we choose $a = 0.005$. It is noted that the smaller a is, the more convection dominates, and the higher the concentration of mesh points required around the wave front. Figure 4 shows the movement of a moving mesh solution with 20×20 nodes. The monitor function used is $\sqrt{1 + \|\nabla u\|_2^2} I$.

In this problem, large gradient solutions 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. A simple redistribution strategy is proposed as follows. The basic idea is to move the boundary points by solving 1-D moving mesh equations. Without lose of generality, we consider a simple boundary $[a, b]$ in the x-direction. Solving the two-point boundary value problem for $(wx_\xi)_\xi = 0$ with uniform mesh in ξ will lead to a new boundary redistribution. Assume $[x_j, x_{j+1}] \subset [a, b]$. Then there exists exactly one element T_j whose one edge is $[x_j, x_{j+1}]$. Note that the gradient monitor in T_j is a constant (due to the use of the linear element). We let the monitor function $w|_{[x_j, x_{j+1}]}$ equals to this constant, which establishes a connection between the boundary and interior grid redistribution. This redistribution strategy is applied to both Examples 6.2 and 6.3.

In Fig. 5, we plot the L^1-errors obtained by using the fixed and moving meshes with 20×20 nodes (930 triangular elements). As expected, the numerical solution with a moving mesh is much more accurate than the one obtained by using a fixed mesh. The L^1-error of the finite element solution with a moving mesh is about 9% of that with a fixed mesh. For comparison, we also plot in Fig. 5 the L^1-errors with 16×16 nodes obtained by using the present moving mesh algorithm (second line from the top) and the moving mesh PDE approach (third curve from the top) obtained in [11]. It is observed that with the same number of nodes the numerical error of our moving mesh algorithm is smaller than that of the moving mesh PDE approach.

EXAMPLE 6.3. Our third example is concerned with the buoyancy-driven horizontal spreading of heat and chemical species through a fluid saturated porous medium. The physical problem is discussed and formulated in [11]. The domain Ω in the physical domain is shown in Fig. 7, and the governing equation is

$$\begin{aligned}
-\Delta\psi &= Ra\left(\frac{\partial T}{\partial x} + N\frac{\partial C}{\partial x}\right),\\
\frac{\partial T}{\partial t} &+ \frac{\partial(T, \psi)}{\partial(x, y)} = \Delta T,\\
\frac{\phi}{\sigma}\frac{\partial C}{\partial t} &+ \frac{\partial(C, \psi)}{\partial(x, y)} = \frac{1}{Le}\Delta C,
\end{aligned}$$

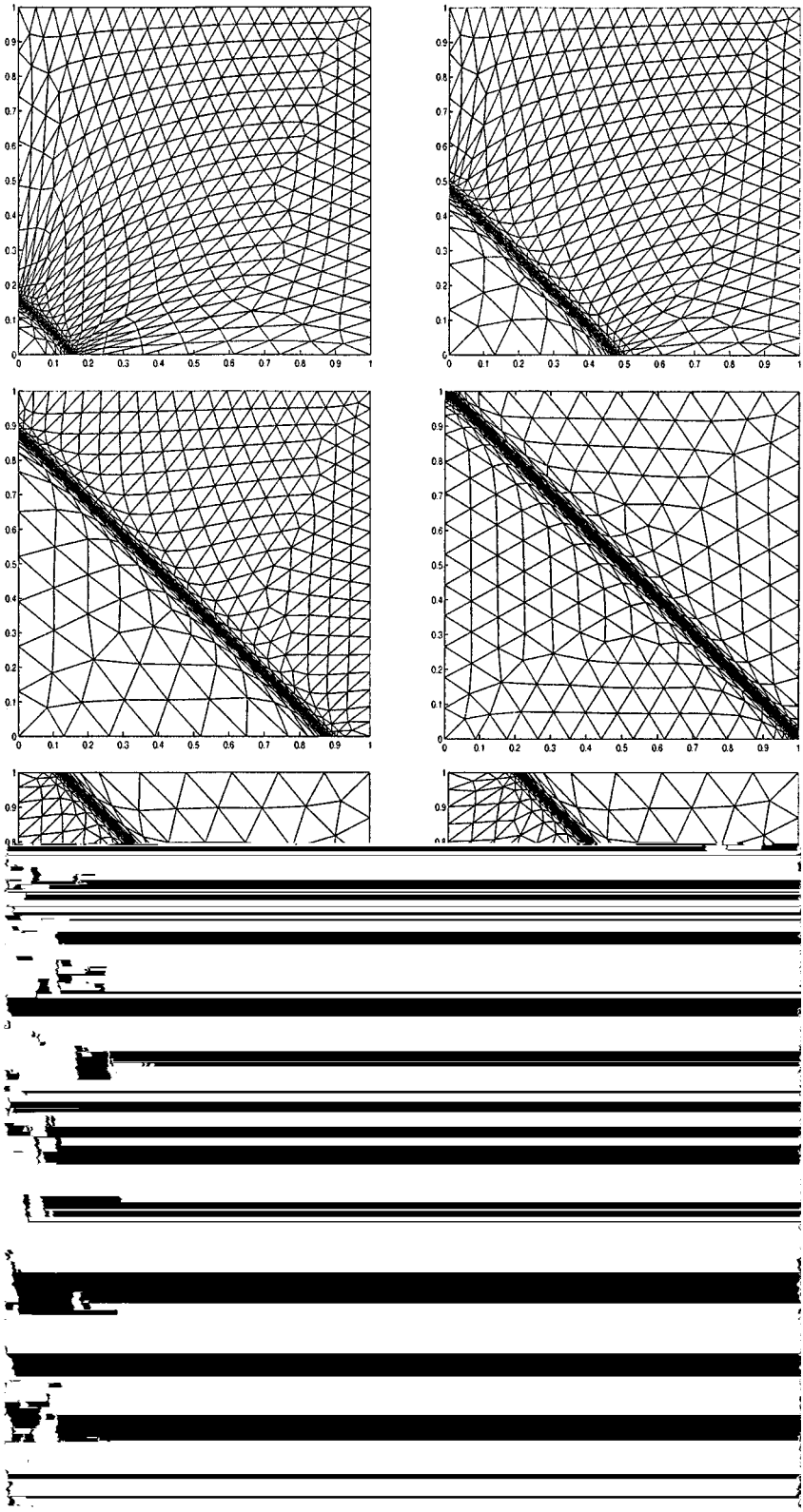

FIG. 4. Example 6.2: The adaptive meshes with 20×20 nodes from $t = 0.15$ to $t = 1.8$.

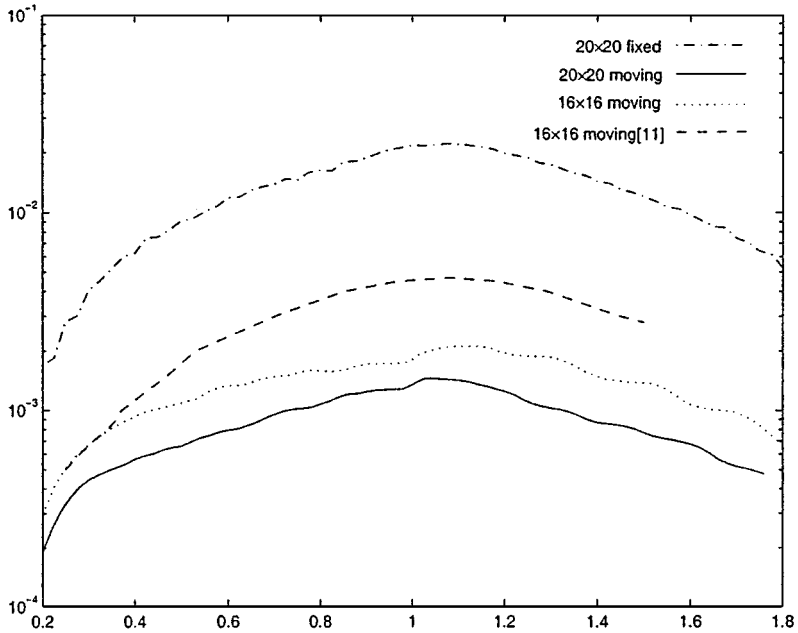

FIG. 5. Example 6.2: L^1-error in time for fixed mesh with 20×20 nodes (– · –), moving mesh with 16×16 nodes (· · ·), and moving mesh with 20×20 nodes (–). The dashed line (– –) is the moving mesh PDE result with 16×16 nodes obtained in [11].

where ψ is the stream function of the flow, T the temperature, C the concentration of the constituent, Ra the Darcy-modified Rayleigh number, N the buoyancy ratio, Le the Lewis number, ϕ the porosity ratio, σ the heat capacity ratio, and

$$\frac{\partial(f,g)}{\partial(x,y)} := \frac{\partial f}{\partial x}\frac{\partial g}{\partial y} - \frac{\partial f}{\partial y}\frac{\partial g}{\partial x}.$$

The initial conditions are given by

$$\psi|_{t=0} = 0, \quad T|_{t=0} = C|_{t=0} = \begin{cases} 1, & \text{for } x \le 0.5, \\ 0, & \text{for } x > 0.5. \end{cases} \tag{6.3}$$

The boundary conditions are given by

$$\psi\big|_{\partial\Omega} = 0, \quad \frac{\partial T}{\partial n}\Big|_{\partial\Omega} = \frac{\partial C}{\partial n}\Big|_{\partial\Omega} = 0. \tag{6.4}$$

In this problem, the fluid is initially of different degrees of temperature and concentration of a certain constituent. At the beginning, the warm fluid on the left side of the domain has a less pronounced vertical gradient of hydrostatic pressure than the cold fluid on the right side. This horizontal difference of pressure will start to push the cold fluid to the left side at the bottom and warm fluid to the right side at the top. This keeps the fluid convecting until the cold fluid rests under the warm one. Meanwhile, the diffusion effect will gradually smooth out the temperature and concentration differences between the initially cold and warm fluids. We will stimulate this phenomenon for the case of a large Rayleigh number, $Ra = 1000$. Other parameters in the governing equations are $N = 0$, $Le = 1$ and $\phi/\sigma = 1$. In the logical

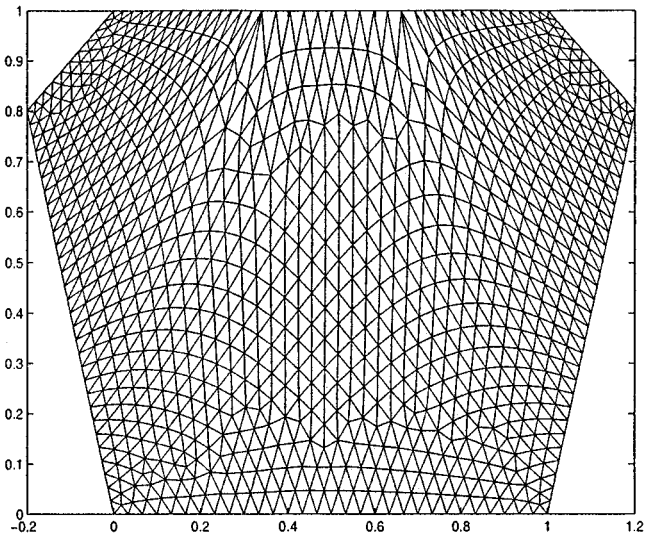

FIG. 6. Meshes in the logical domain for Example 6.3.

domain, a quasi-uniform triangulation with 1784 elements is shown in Fig. 6. Physically, if the Rayleigh number is large enough a thin layer of large variation of temperature and concentration will keep existing until the warm fluid settles completely on top of the cold one and eventually the temperature and concentration become uniform in the whole fluid. These phenomena are clearly observed in Fig. 7. It is seen that the mesh adapts well to the temperature and follows successfully the motion of the thin layer of large temperature and concentration variation.

7. NUMERICAL EXPERIMENTS: REACTION–DIFFUSION EQUATIONS

One of the key ideas of our moving mesh scheme is to keep the time scales of the given equation (1.1) and the moving mesh equation (2.9) different. This approach is different from the one used in moving mesh PDEs (see e.g., [10, 13, 24]) in which μ in (2.9) is replaced by the physical time t. Integrating (1.1) and (2.9) with different time scale can avoid the difficulty that the time step δt has to be restricted by *both* Eqs. (1.1) and (2.9). In the method of moving mesh PDEs, this difficulty has to be partially overcome by introducing a non-physical parameter to the right hand side of (2.9).

Our next numerical example is a combustion problem which was investigated numerically in [11, 31]. The main purpose of this example is to demonstrate that Section 4.4 can be independent of other steps in Section 4. In particular, the mesh-moving step and the time-forwarding step are *independent* of each other.

EXAMPLE 7.1. The mathematical model is a system of coupled nonlinear reaction–diffusion equations,

$$\frac{\partial u}{\partial t} - \nabla^2 u = -\frac{R}{\alpha\delta} u e^{\delta(1-1/T)},$$

$$\frac{\partial T}{\partial t} - \frac{1}{Le}\nabla^2 T = \frac{R}{\delta Le} u e^{\delta(1-1/T)},$$

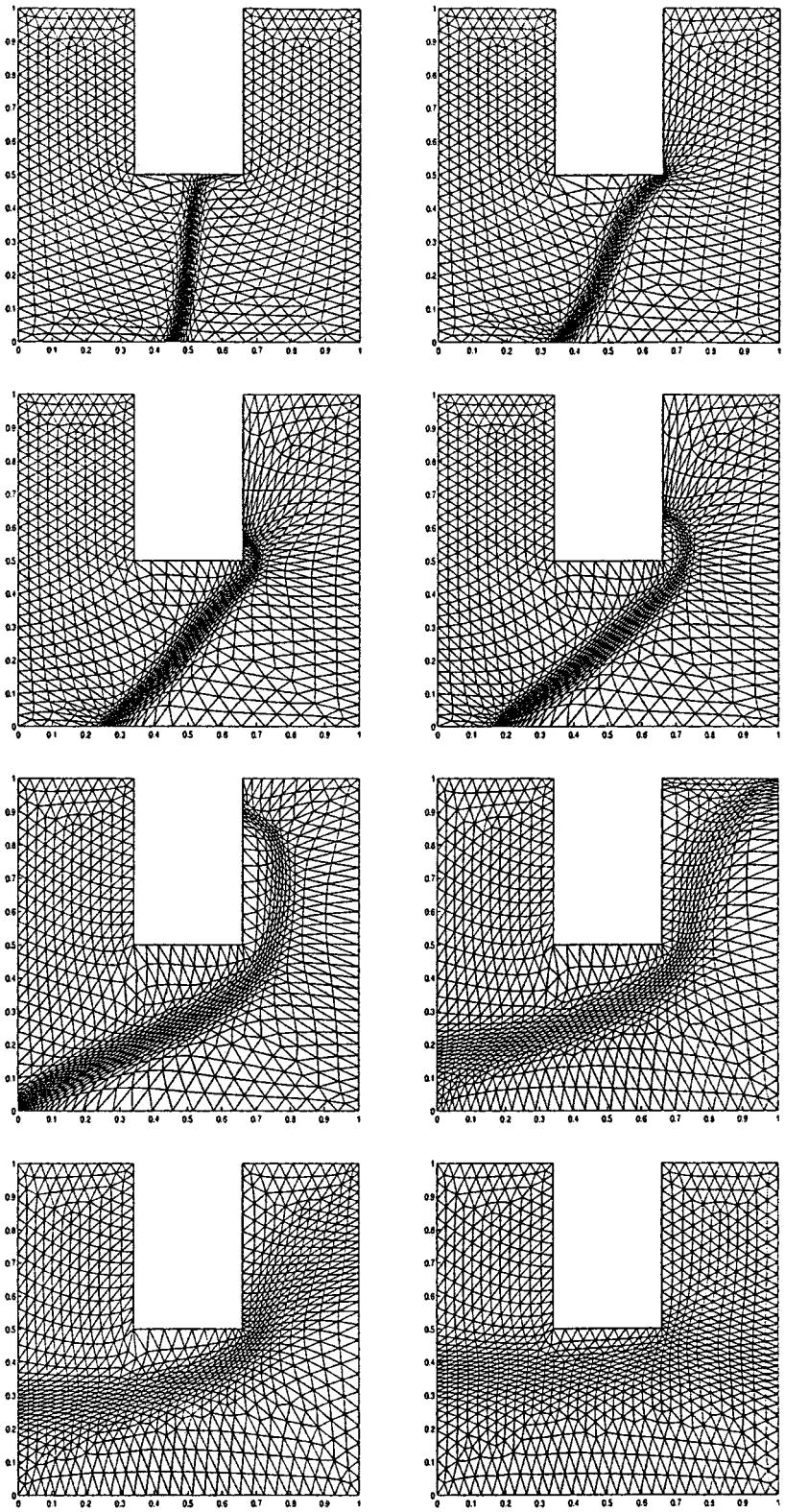

FIG. 7. The adaptive meshes at various times for Example 6.3.

for $\vec{x} = (x_1, x_2)^T \in \Omega, t > 0$, and where u and T represent the dimensionless concentration and temperature of a chemical which is undertaking a one-step reaction. As in [31], we consider a simple square domain $\Omega = [-1, 1] \times [-1, 1]$, with the initial and boundary conditions

$$u|_{t=0} = T|_{t=0} = 1, \quad \text{in } \Omega,$$
$$u|_{\partial\Omega} = T|_{\partial\Omega} = 1, \quad \text{for } t > 0.$$

The physical parameters are set to be $Le = 0.9, \alpha = 1, \delta = 20$, and $R = 5$.

To begin with, choosing $v \in H_0^1(\Omega)$ to be a test function gives

$$\begin{aligned} 0 &= \int_\Omega \left\{ \frac{\partial u}{\partial t} - \nabla^2 u + \frac{R}{\alpha\delta} u e^{\delta(1-1/T)} \right\} v dx \\ &= \int_\Omega \left\{ \frac{\partial u}{\partial t} v + \nabla u \cdot \nabla v + \frac{R}{\alpha\delta} u e^{\delta(1-1/T)} v \right\} dx, \\ 0 &= \int_\Omega \left\{ \frac{\partial T}{\partial t} - \frac{1}{Le} \nabla^2 T - \frac{R}{\delta Le} u e^{\delta(1-1/T)} \right\} v dx \\ &= \int_\Omega \left\{ \frac{\partial T}{\partial t} v + \frac{1}{Le} \nabla u \cdot \nabla v - \frac{R}{\delta Le} u e^{\delta(1-1/T)} v \right\} dx. \end{aligned}$$

For convenience, let $\phi^i, 1 \le i \le N$ be basis functions at inner nodes, and let $\psi^i, 1 \le i \le M$ be basis functions at boundary nodes. We then express u and T as

$$u = u_i^{(i)} \phi^i + u_l^{(b)} \psi^l, \quad T = T_i^{(i)} \phi^i + T_l^{(b)} \psi^l.$$

Using the test functions $\phi^j \in V_{h,0}(\Omega)$ in the above equations leads to

$$\begin{aligned} 0 = \int_\Omega \Bigg\{ & \frac{\partial u_i^{(i)}}{\partial t} \phi^i \phi^j + \left(u_i^{(i)} \nabla \phi^i + u_l^{(b)} \nabla \psi^l \right) \cdot \nabla \phi^j, \\ & + \frac{R}{\alpha\delta} \left(u_i^{(i)} \phi^i + u_l^{(b)} \right) e^{\delta(1-1/T)} \phi^j \Bigg\} dx, \\ 0 = \int_\Omega \Bigg\{ & \frac{\partial T_i}{\partial t} \phi^i \phi^j + \frac{1}{Le} \left(T_i^{(i)} \nabla \phi^i + T_l^{(b)} \nabla \psi^l \right) \cdot \nabla \phi^j \\ & - \frac{R}{\delta Le} \left(u_i^{(i)} \phi^i + u_l^{(b)} \psi^l \right) e^{\delta(1-1/T)} \phi^j \Bigg\} dx. \end{aligned}$$

By using the linear approximation for the term $e^{\delta(1-1/T)}$

$$e^{\delta(1-1/T)} \approx e^{\delta(1-1/T_k^{(i)})} \phi^k + e^{\delta(1-1/T_m^{(b)})} \psi^m,$$

we obtain the discretized system to be used for the computation,

$$0 = \int_\Omega \left\{ \frac{\partial u_i^{(i)}}{\partial t}\phi^i\phi^j + \left(u_i^{(i)}\nabla\phi^i + u_l^{(b)}\nabla\psi^l\right)\cdot\nabla\phi^j \right.$$
$$\left. + \frac{R}{\alpha\delta}\left(u_i^{(i)}\phi^i + u_l^{(b)}\psi^l\right)\left(e^{\delta(1-1/T_k^{(i)})}\phi^k + e^{\delta(1-1/T_m^{(b)})}\psi^m\right)\phi^j \right\} dx,$$
$$0 = \int_\Omega \left\{ \frac{\partial T_i^{(i)}}{\partial t}\phi^i\phi^j + \frac{1}{Le}\left(T_i^{(i)}\nabla\phi^i + T_l^{(b)}\nabla\psi^l\right)\cdot\nabla\phi^j \right.$$
$$\left. - \frac{R}{\delta Le}\left(u_i^{(i)}\phi^i + u_l^{(b)}\psi^l\right)\left(e^{\delta(1-1/T_k^{(i)})}\phi^k + e^{\delta(1-1/T_m^{(b)})}\psi^m\right)\phi^j \right\} dx.$$

We can write the above system in matrix forms by letting

$$A_{ij} = -\int_\Omega \phi^i\phi^j dx$$
$$\left(B_u^{(i)}\right)_{ij} = \int_\Omega \left\{ \nabla\phi^i\cdot\nabla\phi^j + \frac{R}{\alpha\delta}\left(e^{\delta(1-1/T_k^{(i)})}\phi^k + e^{\delta(1-1/T_m^{(b)})}\psi^m\right)\phi^i\phi^j \right\} dx$$
$$\left(B_u^{(b)}\right)_{lj} = \int_\Omega \left\{ \nabla\psi^l\cdot\nabla\phi^j + \frac{R}{\alpha\delta}\left(e^{\delta(1-1/T_k^{(i)})}\phi^k + e^{\delta(1-1/T_m^{(b)})}\psi^m\right)\psi^l\phi^j \right\} dx$$
$$\left(B_{T,T}^{(i)}\right)_{ij} = \int_\Omega \frac{1}{Le}\nabla\phi^i\cdot\nabla\phi^j dx$$
$$\left(B_{T,u}^{(i)}\right)_{ij} = \int_\Omega -\frac{R}{\delta Le}\left(e^{\delta(1-1/T_k^{(i)})}\phi^k + e^{\delta(1-1/T_m^{(b)})}\psi^m\right)\phi^i\phi^j dx$$
$$\left(B_{T,T}^{(b)}\right)_{lj} = \int_\Omega \frac{1}{Le}\nabla\psi^l\cdot\nabla\phi^j dx$$
$$\left(B_{T,u}^{(b)}\right)_{lj} = \int_\Omega -\frac{R}{\delta Le}\left(e^{\delta(1-1/T_k^{(i)})}\phi^k + e^{\delta(1-1/T_m^{(b)})}\psi^m\right)\psi^l\phi^j dx$$
$$\mathbf{u}^{(i)} = \left(u_1^{(i)},\ldots,u_N^{(i)}\right)^T \quad \mathbf{T}^{(i)} = \left(T_1^{(i)},\ldots,T_N^{(i)}\right)^T$$
$$\mathbf{u}^{(b)} = \left(u_1^{(b)},\ldots,u_M^{(b)}\right)^T \quad \mathbf{T}^{(b)} = \left(T_1^{(b)},\ldots,T_M^{(b)}\right)^T.$$

Then the semi-discrete system for Example 7.1 is given by

$$A\frac{\partial\mathbf{u}^{(i)}}{\partial t} = B_u^{(i)}\mathbf{u}^{(i)} + B_u^{(b)}\mathbf{u}^{(b)}$$
$$A\frac{\partial\mathbf{T}^{(i)}}{\partial t} = \left(B_{T,T}^{(i)}\ B_{T,T}^{(b)}\right)\begin{pmatrix}\mathbf{T}^{(i)}\\ \mathbf{T}^{(b)}\end{pmatrix} + \left(B_{T,u}^{(i)}\ B_{T,u}^{(b)}\right)\begin{pmatrix}\mathbf{u}^{(i)}\\ \mathbf{u}^{(b)}\end{pmatrix}.$$

A three-stage Runge–Kutta method is used to integrate the above ODE system. At each time step, when the numerical approximation is obtained we use the mesh-moving techniques as described in Section 4.3 to re-distribute the mesh in the physical domain. We emphasize again that Section 4.3 only requires the outputs of Section 4.4 but does not care how these outputs are obtained. In Fig. 8, the moving meshes obtained with 30×30 nodes are plotted, together in this figure are the numerical solutions for the temperature T at various time levels. Although the temperature T has a very thin layer of large variation, our moving mesh scheme adapts the mesh extremely well to the regions with large solution gradients.

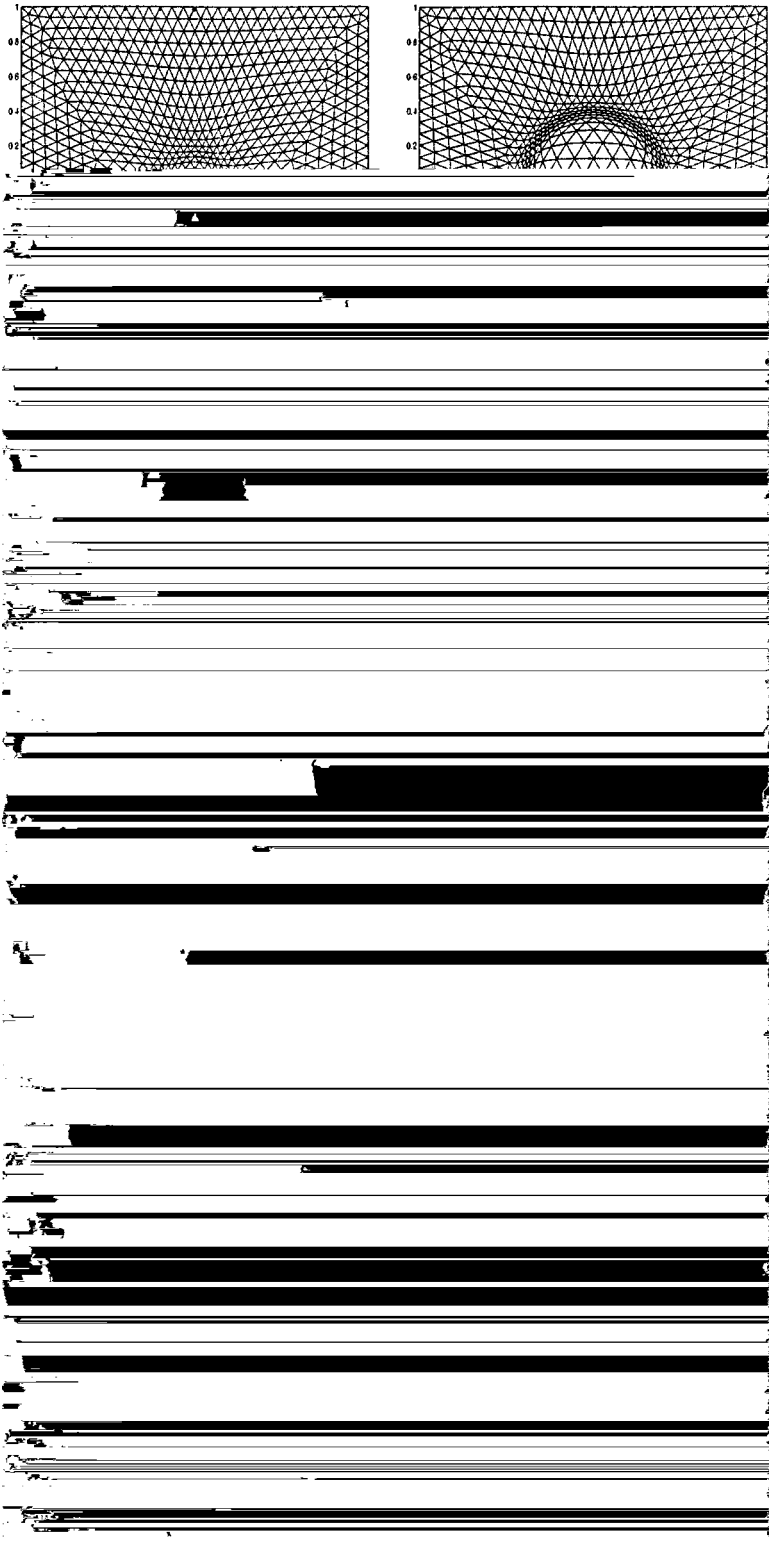

FIG. 8. Example 7.1: Moving mesh and temperature T at $t = 0.259$, 0.263, 0.271, and 0.288.

8. CONCLUDING REMARKS

Solving the physical PDE using a moving mesh PDE approach would involve changing the FEM solver to deal with the mesh movement terms, but it may expect that this inconvenience would be outweighed by allowing larger time steps as the solution would to appear to change much more slowly with respect to the moving reference frame. In other words, although the present method is attractive since a standard piece of software can be used, the price to pay here *may be* that smaller time steps have to be used as the steep solution would appear to change rapidly in time with respect to a fixed reference frame. To verify this, we plot in Fig. 9 the step sizes employed in a very recent work of Cao *et al.* [12] (which are generated by a rather complicated selection formula) and those used in the last section for Example 7.1, with the same number of finite elements. This problem is very difficult to solve numerically and so serves as a good test for the comparison purpose. Since large solution gradients only occur after $t \approx 0.24$ (i.e., the effect of mesh moving becomes important after this time), we only plot the time step sizes for $t \geq 0.24$. It is seen from Fig. 9 that the time steps used with the present moving mesh method are comparable with the ones selected with the moving mesh PDE approach. Although our (average) time step is about three times smaller than the one with the moving mesh PDE approach, the present method is overall more efficient, by noting that in our approach simpler equations are solved and *explicit* ODE solvers are used, but in [12] an *implicit* ODE solver is employed.

In conclusion, we developed in this paper a moving mesh scheme based on harmonic mapping for solving partial differential equations. The proposed scheme favorably compares with previously proposed methods in terms of simplicity and reliability. Our moving mesh scheme has been seen to work satisfactorily in a variety of circumstances. Although the examples shown in this work are in two space dimensions, in principle our method can be extended to higher dimensions. In fact, some preliminary results for three-dimensional

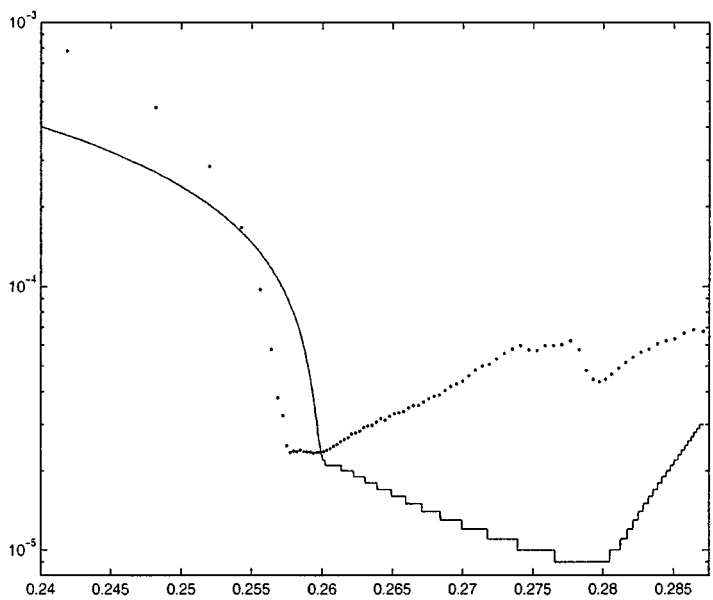

FIG. 9. Example 7.1: the variable time step sizes δt selected with an implicit ODE solver in [12] ($\cdots$) and with an explicit RK3 method used in Section 7 (–).

problems have been obtained based on the present framework. The detailed results for 3D will be reported in a future work.

ACKNOWLEDGMENTS

This work was supported in part by NSERC (Canada) Grant OGP-0105545, Hong Kong Baptist University, and Hong Kong Research Grants Council RGC/99-00/33. The research of PWZ is partially supported by Special Funds for Major State Basic Research Projects of China. We thank Jerry Brackbill, Weimin Cao, Tom Hou, Stan Osher, Bob Russell, and Xiaoping Wang for suggestions and discussions.

REFERENCES

1. S. Adjerid and J. E. Flaherty. A moving finite element method with error estimation and refinement for one-dimensional time dependent partial differential equations. *SIAM J. Numer. Anal.* **23**, 778–796 (1986).
2. D. A. Anderson, Equidistribution schemes, Poisson generators, and adaptive grids, *Appl. Math. Comput.* **24**, 211–227 (1987).
3. M. J. Baines, *Moving Finite Elements* (Oxford Univ. Press, Oxford, 1994).
4. R. E. Bank and R. F. Santos, Analysis of some moving space-time finite element methods, *SIAM J. Numer. Anal.* **30**, 1–18 (1993).
5. J. Bell, M. Berger, J. S. Saltzman, and M. Welcome, Three dimensional adaptive mesh refinement for hyperbolic conservation laws, *SIAM J. Sci. Comput.* **15**, 127–138 (1994).
6. M. Berger and R. LeVeque, Adaptive mesh Refinement for two-dimensional hyperbolic systems and the AMRCLAW software, *SIAM J. Numer. Anal.* **35**, 2298–2316 (1998).
7. M. Bertalmio, L.-T. Cheng, S. Osher, and G. Sapiro, *Variational Problems and Partial Differential Equations on Implicit Surfaces: The Framework and Examples in Image Processing and Pattern Formation*, UCLA CAM Report 00–23 (2000).
8. J. U. Brackbill and J. S. Saltzman, Adaptive zoning for singular problems in two dimensions, *J. Comput. Phys.* **46**, 342–368 (1982).
9. J. U. Brackbill, An adaptive grid with direction control, *J. Comput. Phys.* **108**, 38–50 (1993).
10. W. M. Cao, W. Z. Huang, and R. D. Russell, A study of monitor functions for two dimensional adaptive mesh generation, *SIAM J. Sci. Comput.* **20**, 1978–1994 (1999).
11. W. M. Cao, W. Z. Huang, and R. D. Russell, An *r*-adaptive finite element method based upon moving mesh PDEs, *J. Comput. Phys.* **149**, 221–244 (1999).
12. W. M. Cao, W. Z. Huang, and R. D. Russell, An error indicator monitor function for an *r*-adaptive finite element method. To appear in JCP, 2001.
13. H. D. Ceniceros and T. Y. Hou, An efficient dynamically adaptive mesh for potentially singular solutions, *J. Comput. Phys.* to appear (2001).
14. S. F. Davis and J. E. Flaherty, An adaptive finite element method for initial–boundary value problems for partial differential equations, *SIAM J. Sci. Stat. Comput.* **3**, 6–27 (1982).
15. A. S. Dvinsky, Adaptive grid generation from harmonic maps on Riemannian manifolds, *J. Comput. Phys.* **95**, 450–476 (1991).
16. J. Eell and J. H. Sampson, Harmonic mappings of Riemannian manifolds, *Amer. J. Math.* **86**, 109–160 (1964).
17. M. Fortin, Anisotropic mesh adaptation through hierarchical error estimators, Private communication. To appear in *Advances in computation: theory and practice*, Nova Science, P. Minev, Y. Lin and Y. Wong (eds), 2001.
18. R. M. Furzeland, J. G. Verwer and P. A. Zegeling. A numerical study of three moving gird methods for I-D partial differential equations which are based on the method of lines, *J. Comput. Phys.* **89**, 349–399 (1990).
19. T. Gebner and D. Kroner, Dynamic mesh adaptation for supersonic reactive flow, personal communication; to appear in *Proc. for 8th Int. Conf. Hyperbolic Problems*, edited by G. Warnecke and H. Freistuehler (2001).

20. R. Hamilton, *Harmonic Maps of Manifolds with Boundary*, Lecture Notes in Mathematics (Springer-Verlag, New York, 1975), Vol. 471.
21. Ph. Hoch and M. Rascle, Hamilton–Jacobi equations on a manifold and applications to grid generation or refinement, personal communication.
22. W. Z. Huang and R. D. Russell, Moving mesh strategy based on a gradient flow equation for two-dimensional problems. *SIAM J. Sci. Comput.* **20**, 3, 998–1015 (1999).
23. W. Z. Huang and D. M. Sloan, A simple adaptive grid method in two dimensions, *SIAM J. Sci. Comput.* **15**, 776–797 (1994).
24. S. Li, L. Petzold, and Y. Ren, Stability of moving mesh systems of PDEs, *SIAM J. Sci. Comput.* **20**, 719–738 (1998).
25. G. Liao, A study of regularity problem of harmonic maps, *Pacific J. Math.* **131**, 291–302 (1988).
26. G. Liao and N. Smale, Harmonic maps with nontrivial higher-dimensional singularities, in Lecture Notes in Pure and Applied Mathematics (Dekker, New York, 1993), Vol. 144, pp. 79–89.
27. G. Liao, F. Liu, C. de la Pena, D. Peng, and S. Osher, Level-set-based deformation methods for adaptive grids, *J. Comput. Phys.* **159**, 103–122 (2000).
28. W.-B. Liu and T. Tang, Spectral methods for singular perturbation problems, in *Proceedings of Symposia in Applied Mathematics*, edited by W. Gautschi (Amer. Math. Soc., Providence, 1994), Vol. 48, pp. 323–326.
29. K. Miller and R. N. Miller, Moving finite element methods, I, *SIAM J. Numer. Anal.* **18**, 1019–1032 (1981).
30. K. Miller, Moving finite element methods, II, *SIAM J. Numer. Anal.* **18**, 1033–1057 (1981).
31. P. K. Moore and J. E. Flaherty, Adaptive local overlapping grid methods for parabolic systems in two space dimensions, *J. Comput. Phys.* **98**, 54–63 (1992).
32. R. Pember, J. Bell, P. Collela, W. Crutchfield, and M. Welcome, An adaptive cartesian grid method for unsteady compressible flow in irregular regions, *J. Comput. Phys.* **120**, 278–304 (1995).
33. L. Petzold, Observations on an adaptive moving grid method for one-dimensional systems of partial differential equations, *Appl. Numer. Math.* **3**, 347–360 (1987).
34. O. Pironneau and F. Hecht, Mesh adaptation for the Black and Scholes equations, *East–West J. Numer. Math.* **8**, 25–35 (2000).
35. Y. Qiu, D. M. Sloan, and T. Tang, Numerical solution of a singularly perturbed two-point boundary value problem using equidistribution: Analysis of convergence, *J. Comput. Appl. Math.* **116**, 121–143 (2000).
36. W. Ren and X.-P. Wang, An iterative grid redistribution method for singular problems in multiple dimensions, *J. Comput. Phys.* **159**, 246–273 (2000).
37. R. Schoen and S.-T. Yau, On univalent harmonic maps between surfaces, *Invent. Math.* **44**, 265–278 (1978).
38. J. H. Smith and A. M. Stuart, *Analysis of Continuous Moving Mesh Equations*, Technical Report, SCCM Program (Stanford University, Stanford, CA, 1996).
39. T. Tang and M. R. Trummer, Boundary layer resolving pseudospectral methods for singular perturbation problems, *SIAM J. Sci. Comput.* **17**, 430–438 (1996).
40. J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *Numerical Grid Generation* (North-Holland, New York, 1985).
41. A. Winslow, Numerical solution of the quasi-linear Poisson equation in a nonuniform triangle mesh, *J. Comput. Phys.* **1**, 149–172 (1967).