Journal of Computational Physics **177,** 365–393 (2002)
doi:10.1006/jcph.2002.7002, available online at http://www.idealibrary.com on IDEAL®

A Moving Mesh Finite Element Algorithm for Singular Problems in Two and Three Space Dimensions

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

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

Received July 30, 2001; revised January 11, 2002

A framework for adaptive meshes based on the Hamilton–Schoen–Yau theory was proposed by Dvinsky. In a recent work (2001, *J. Comput. Phys.* **170**, 562–588), we extended Dvinsky's method to provide an efficient moving mesh algorithm which compared favorably with the previously proposed schemes in terms of simplicity and reliability. In this work, we will further extend the moving mesh methods based on harmonic maps to deal with mesh adaptation in three space dimensions. In obtaining the variational mesh, we will solve an *optimization problem* with some appropriate constraints, which is in contrast to the traditional method of solving the Euler–Lagrange equation directly. The key idea of this approach is to update the interior and boundary grids simultaneously, rather than considering them separately. Application of the proposed moving mesh scheme is illustrated with some two- and three-dimensional problems with large solution gradients. The numerical experiments show that our methods can accurately resolve detail features of singular problems in 3D. © 2002 Elsevier Science (USA)

Key Words: finite element method; moving mesh method; harmonic map; partial differential equations; optimization.

1. INTRODUCTION

Moving mesh methods have important applications in a variety of physical and engineering areas such as solid and fluid dynamics, combustion, heat transfer, material science, etc. The physical phenomena in these areas develop dynamically singular or nearly singular solutions in fairly localized regions, such as shock waves, boundary layers, detonation waves, etc. The numerical investigation of these physical problems may require extremely fine meshes over a small portion of the physical domain to resolve the large solution variations. In multidimensions, developing effective and robust adaptive grid methods for these problems

0021-9991/02 $35.00

becomes necessary. Successful implementation of the adaptive strategy can increase the accuracy of the numerical approximations and also decrease the computational cost.

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 [34]. 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. Brackbill and Saltzman [8] formulated the grid equations in a variational form to produce satisfactory mesh concentration while maintaining relatively good smoothness and orthogonality. Their approach has become one of the most popular methods used for mesh generation and adaptation. In [7], Brackbill incorporates an efficient directional control into the mesh adaptation, thereby improving both the accuracy and efficiency of the numerical schemes.

Dvinsky [14] suggests the possibility 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 finite element strategy based on harmonic mapping was proposed and studied by the authors in [21]. The key idea is to construct the harmonic map between the physical space and a parameter space by an *iteration* procedure. This procedure is simple, easy to program, and also enables us to keep the map harmonic even after long time of numerical integration. In practice, there are three types of adaptive methods using finite element approach, namely the *h-method*, *p-method*, and *r-method*. In the *h*-method, the overall method contains two parts, i.e., a solution algorithm and a mesh selection algorithm. These two parts are independent in the sense that the change of the PDEs will affect the first part only. However, in some of the existing *r*-methods (also known as *moving mesh methods*), 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. The algorithm proposed in [21] keeps 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 of each other). The simplicity and reliability were demonstrated by a number of numerical examples in two space dimensions.

Although the PDE time-evolution algorithm used in this work is based on finite element approach, it should be pointed out that there have been also extensive studies of moving mesh algorithms based on finite difference approaches; see, e.g., Azarenok [2], Azarenok and Ivanenko [3], Ceniceros and Hou [11], Dorfi and Drury [13], Liu *et al*. [24], and Tang [31]. Moving mesh methods based on finite difference (or finite volume) methods enjoy the simplicity in coding and are more appropriate for problems in which finite difference has shown advantages (such as nonlinear hyperbolic problems). Apart from the flexibility of solution domains, the other reasons that we use finite elements but not finite differences in this work are the following: first, with the finite element method we can avoid using interpolation which was used in many moving mesh finite difference methods; second, with the use of finite element approach, some error-estimator-based indicators available for finite

element methods can be chosen as monitor functions, which has been demonstrated in our recent work in solving elliptic optimal control problems with mesh adaptation [20]. In the past two decades, there have been extensive studies of moving mesh finite element methods, and some of them are based on interesting motivations and theories; see, e.g., Baines [4], Miller *et al*. [25, 10], and Tourigny *et al*. [32, 33].

To completely specify the coordinate transformation, the moving mesh methods must be supplemented with suitable boundary conditions. In theory, as pointed out in [7, 9], there are a number of ways to redistribute the mesh points on the boundary, such as using homogeneous Neumann boundary conditions, extrapolating the interior mesh points to the boundary, and relocating the mesh points by solving a lower-dimensional moving-mesh PDE. However, these methods seem quite inefficient if 3D problems are being considered. In this work, we present a moving mesh method, which is based on the minimization of the mesh energy, for solving problems in three space dimensions. In the mesh-restructuring step, we solve an *optimization problem* with some appropriate constraints, which is in contrast to the traditional method of solving the Euler–Lagrange equation directly. The key idea of this approach is to treat the interior and boundary grids as a whole, rather than considering them separately. Therefore, the new solution algorithm also provides an alternative boundary grid redistribution technique, which turns out to be useful in solving 3D problems. We point out that there are other mesh movement approaches that can deal with boundary and interior points in a unified manner. For example, to solve variational problems, the approach by Tourigny and Hülsemann [32] is dimension independent.

The main contributions of this work are two-fold: First, some theoretical and numerical issues in obtaining moving meshes by minimizing the mesh energy are investigated. Second, the algorithm is applied to two nonlinear 3D partial differential equations, one scalar and one system. To our knowledge, there have existed very few moving mesh results for three-dimensional problems. Using the moving mesh methods proposed in this work, we are able to carry out the 3D simulations successfully. An outline of the paper is as follows. In Section 2, we present a short overview over the harmonic mapping theory. Section 3 describes the numerical schemes to be used in this work. Section 4 provides some necessary details of the mesh-redistribution and solution-updating at a given time level. Numerical results for two- and three-dimensional problems are obtained in Section 5. Some discussions on the quality of the generated meshes are presented in Section 6. Finally, Section 7 contains some concluding remarks.

2. HARMONIC MAPPING

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

$$\vec{u}_t = L(\vec{u}) \quad \text{in } \Omega \times (0, T], \tag{2.1}$$

with boundary and initial conditions

$$B\vec{u}|_{\partial\Omega} = \vec{u}_b \quad \text{in } \partial\Omega \times [0, T] \tag{2.2}$$

$$\vec{u}|_{t=0} = \vec{u}_0 \quad \text{in } \Omega. \tag{2.3}$$

To solve the above time-dependent problem, we will introduce the moving mesh method based on harmonic mapping.

Let Ω and Ω_c be compact Riemannian manifolds of dimension n with metric tensors d_{ij} and $r_{\alpha\beta}$ in some local coordinates $\vec{x}$ and $\vec{\xi}$, respectively. Following Dvinsky [14] and Brackbill [7], we define the energy for a map $\vec{\xi} = \vec{\xi}(\vec{x})$ as

$$E(\vec{\xi}) = \frac{1}{2} \int \sqrt{d}\, d^{ij} r_{\alpha\beta} \frac{\partial \xi^\alpha}{\partial x^i} \frac{\partial \xi^\beta}{\partial x^j}\, d\vec{x}, \tag{2.4}$$

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

$$\frac{1}{\sqrt{d}} \frac{\partial}{\partial x^i} \sqrt{d}\, d^{ij} \frac{\partial \xi^k}{\partial x^j} + d^{ij} \Gamma^k_{\beta\gamma} \frac{\partial \xi^\beta}{\partial x^i} \frac{\partial \xi^\gamma}{\partial x^j} = 0, \tag{2.5}$$

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–Schoen–Yau [16, 29]). 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

$$\frac{\partial}{\partial x^i} \sqrt{d}\, d^{ij} \frac{\partial \xi^k}{\partial x^j} = 0. \tag{2.6}$$

We emphasize that $d = \det(d_{ij}) = 1/\det(d^{ij})$. 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

$$\frac{\partial}{\partial x^i} \left(G^{ij} \frac{\partial \xi^k}{\partial x^j} \right) = 0, \tag{2.7}$$

and the corresponding mesh energy is of the form

$$E(\vec{\xi}) = \sum_k \int_\Omega G^{ij} \frac{\partial \xi^k}{\partial x^i} \frac{\partial \xi^k}{\partial x^j}\, d\vec{x}. \tag{2.8}$$

Solutions to (2.7) are harmonic functions giving a continuous, one-to-one mapping with continuous inverse, which is differentiable and has a nonzero Jacobian. A detailed description for solving (2.7), as well as how to interchange its dependent and independent variables, can be found in Li *et al*. [19, 21].

In [14], Dvinsky suggests that harmonic function theory may provide a general framework for developing useful mesh generators. The idea is that one is free to specify G^{ij} as a function of physical coordinates when defining the energy (2.8), and that minimizing that energy will result in a harmonic mapping with the desired metric and robustness. 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 [16] and Schoen and Yau [29]. Such theoretical guarantees are rare in the field of adaptive mesh generation. To solve the Euler–Lagrange equation (2.7), one can usually use a deformation from a given homomorphism to the harmonic map by the heat equations. Namely, solving the heat equation

$$\frac{\partial \xi^k}{\partial \mu} = \frac{\partial}{\partial x^i}\left(G^{ij}\frac{\partial \xi^k}{\partial x^j}\right) \quad \text{to } \mu \to \infty \tag{2.9}$$

leads to the harmonic map defined by (2.7). The existence of the solution for the problem (2.9) is addressed by Eell and Sampson [15]. Moreover, the singularity of three-dimensional harmonic maps is discussed by Liao and Smale [22, 23] where harmonic maps having nontrivial singularities of dimension greater than zero are constructed. These theoretical results may have some impact on 3D adaptive schemes based on the harmonic mapping, although it has not been observed in the present 3D computations.

3. THE FRAMEWORK OF THE MOVING MESH SCHEME

For some physical problems, large solution gradients may exist initially or may 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. With the algorithms proposed in the past, special strategies are provided to redistribute the boundary grids. In 2D computations, one common practice is to redistribute the grids *inside* the physical domain by solving (2.7) and to move the boundary points by solving some appropriate 1D moving mesh equations; see, e.g., [9, 21, 31]. In other words, the grid redistributions for the interior domain and boundaries are carried out *separately*. It seems that the extension of this approach to three space dimensions is quite difficult, in particular when a finite element approach is used in evolving the underlying partial differential equations.

In the following, we will introduce a new approach which will redistribute the interior and boundary grids *simultaneously*. To begin, we consider physical problems in two space dimensions. For simplicity, assume that Ω is a polyhedron. Let Γ_i and $\Gamma_{c,i}$ denote the corresponding edges of Ω and Ω_c, respectively. The geometrical constraint to the boundary grid movement is to keep the geometrical character of the physical domain unchanged. This implies that the vertices (edges) of the physical domain will be mapped to the corrersponding vertices (edges) of the computational domain. Therefore, it is reasonable to consider the following mapping set from $\partial\Omega$ to $\partial\Omega_c$:

$$K = \left\{\xi_b \in C^0(\partial\Omega) \mid \xi_b : \partial\Omega \to \partial\Omega_c;\ \xi_{b|\Gamma_i} \text{ is a linear segment and strictly increasing}\right\}. \tag{3.1}$$

One example of such a map is illustrated in Fig. 1. It follows from the theory of Eell and Sampson [15] that for every $\xi_b \in \mathbf{K}$, there exists a unique $\xi : \Omega \to \Omega_c$, such that $\xi|_{\partial\Omega} = \xi_b$ and ξ is the extreme of the functional (2.8). Let us denote the mapping as $\xi = P(\xi_b)$, and consider the optimization problem

$$\begin{aligned} &\min\ E(P(\xi_b)) \\ &s.t.\ \xi_b \in \mathbf{K}, \end{aligned} \tag{3.2}$$

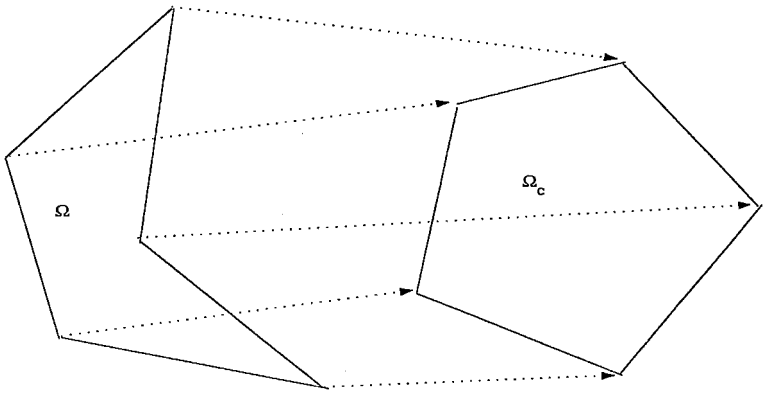


FIG. 1. A map between $\partial\Omega$ and $\partial\Omega_c$.

where the functional E is defined by (2.8). Since E is convex and P is linear, it is easy to see that

$$\frac{1}{2}E\left(P\left(\xi_b^{(1)}\right)\right)+\frac{1}{2}E\left(P\left(\xi_b^{(2)}\right)\right)\geq E\left(\frac{1}{2}P\left(\xi_b^{(1)}\right)+\frac{1}{2}P\left(\xi_b^{(2)}\right)\right)\geq E\left(P\left(\frac{1}{2}\xi_b^{(1)}+\frac{1}{2}\xi_b^{(2)}\right)\right).$$

Therefore, $E(P(\cdot))$ is a convex functional, and as a result, the optimization problem (3.2) has a unique solution in a closed subset of $\mathbf{K}$.

Based on the above discussion, we will solve the following constrained optimization problem:

$$\begin{aligned}&\min \sum_k \int_\Omega G^{ij}\frac{\partial \xi^k}{\partial x^i}\frac{\partial \xi^k}{\partial x^j}\,d\vec{x}\\ &s.t.\ \ \xi|_{\partial\Omega}=\xi_b\in\mathbf{K}.\end{aligned}\tag{3.3}$$

Note that the boundary values ξ_b are not fixed, instead they are unknowns in the same way as the interior points. This is one of the main differences between the present approach and the one proposed in our earlier work [21], where a Dirichlet problem is solved for ξ. By solving the above problem, the meshes on the logical domain will be updated at each iteration. The difference between the initial mesh and this updated mesh in the logical domain will yield the grid redistribution for both the interior *and* boundary grids [based on the formula (4.11) to be given in the next section]. One advantage of this approach is that the overall moving mesh scheme can be easily implemented for 3D problems.

3.1. Initial Grid Distribution

To solve probelms (2.1)–(2.3), we will separate the computation into two parts: mesh-moving and time-stepping. The mesh-moving computation 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 difference with the solution of the optimization problem (3.3) is used to move the mesh in the physical domain. Therefore, in the first step we choose a convex domain Ω_c as the logical domain on which an initial mesh will be constructed.

Traditionally (see, e.g., [34]), the initial mesh in the logical domain is obtained by solving the Poisson equation $\Delta\vec{\xi} = 0$ with some Dirichlet boundary condition. However, in order to match the new global approach (3.3), we generate the initial mesh by solving the following optimization problem:

$$\begin{aligned} &\min \sum_k \int_\Omega \sum_i \left(\frac{\partial \xi^k}{\partial x^i}\right)^2 d\vec{x} \\ &s.t.\ \ \xi|_{\partial\Omega} = \xi_b \in \mathbf{K}. \end{aligned} \tag{3.4}$$

In practice, if the physical domain is convex and is of regular shape (say a convex polygon), then we can simply choose the physical domain as the logical domain with uniform initial mesh.

3.2. Boundary and Interior Grid Redistribution

Once the initial mesh (in the logical domain) is given, it will be kept *unchanged* throughout the computation. This initial mesh in Ω_c, denoted by $\vec{\xi}^{(0)}$, is used as a reference grid only. After the solution u is computed at time level $t = t_n$, the inverse matrix of the monitor, G^{ij} (which in general depends on u), can be updated. By solving (3.3), 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 to obtain the updated values for u in the resulting new grid based on the following principles:

ALGORITHM 1 (MESH-REDISTRIBUTION ALGORITHM).

(a): solve the optimization problem (3.3) and compute the L^∞-difference between the solution of (3.3) and the fixed (initial) mesh in the logical domain. If the difference is smaller than a preassigned tolerance, then the mesh-redistribution at the given time level is complete. Otherwise, do

(b): obtain the direction and the magnitude of the movement for $\vec{x}$ by using the difference obtained in part (a) [see (4.10) in Section 5], and then move the mesh based on (4.11);

(c): update $\vec{u}$ on the new grid by solving a system of ODEs [see (4.17) in Section 5];

(d): update the monitor function by using $\vec{u}$ obtained in part (c), and go to part (a).

In part (a), a preassigned tolerance TOL is chosen so that the iteration is stopped when

$$\left\|\xi^* - \xi^{(0)}\right\|_{L^\infty} \leq \mathrm{TOL}. \tag{3.5}$$

The iteration above determines *progressively* better locations of the mesh grids in the physical domain. Typically about one or two iterations are required. Parts (b) and (c) above have been discussed in detail in Li *et al*. [21]. We refer the reader to that paper for the algorithm details, although some necessary details will be mentioned in the next section.

3.3. Time-Forwarding

After the interior and boundary grids are well redistributed based on the solution at $t = t_n$, we can use some appropriate numerical methods to solve the underlying PDEs at $t = t_{n+1}$ on the updated mesh in the physical space. This step is simple: it is irrelevant with the adaptive method and can be any appropriate finite element codes. The following is one of

the possible methods, which will be used in our numerical experiment sections. It follows from Eq. (2.1) that

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

where $V_{\mathcal{T}}(\Omega)$ is a finite element space. 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 Φ^j, we obtain

$$\int_\Omega \left\{ \frac{\partial U_i}{\partial t} \Phi^i \Phi^j - L(\vec{u})\Phi^j \right\} d\vec{x} = 0, \tag{3.6}$$

which is a system of ODEs for $U_i(t)$. It can be solved by any efficient ODE solver such as multistage Runge–Kutta schemes.

We point out that the method based on (3.6) only serves for the numerical experiments in this paper. In fact, this step is very flexible: any available methods/codes for the equation (2.1) can be employed in this step.

Remark. To extend the above theory to 3D, the only modification needed is to change the definition **K** in (3.1) slightly. In this case, the physical boundary is approximated by some piecewise planes, which will be mapped into the corresponding boundaries in the logical domain.

4. MESH-REDISTRIBUTION AND SOLUTION-UPDATING

This section gives some necessary details for Algorithm 1, the mesh-redistribution algorithm. Again, for simplicity we will demonstrate the main ideas for 2D geometry in this section. Let us discretize the optimization problem (3.3) in the linear finite element space. The triangulation of the physical domain is $\mathcal{T}$, with T_i as its elements, and X_i as its nodes. The corresponding triangulation on the computational domain is $\mathcal{T}_c$, with $T_{i,c}$ as its elements, and $\mathcal{A}_i$ as its nodes. The linear finite element space on the mesh is denoted as $H_h^1(\Omega)$. If the basis function on node X_i is denoted by ϕ^i, then ξ can be approximated by $\xi_i\phi^i$ (here the standard summation convention is assumed). The coordinates of X_i are $(X_i^1 \; X_i^2)^T$. Let the inner nodes be indexed from 1 to N_{inner} and the boundary nodes be indexed from $N_{\text{inner}} + 1$ to N. The coordinates of the nodes $\mathcal{A}_i$ in the computational domain are denoted as $(\mathcal{A}_i^1 \; \mathcal{A}_i^2)^T$. Denote $X = (X^1 \; X^2)^T$, $\mathcal{A} = (\mathcal{A}^1 \; \mathcal{A}^2)^T$, where $X^k = (X_1^k \; \cdots \; X_N^k)^T$, $\mathcal{A}^k = (\mathcal{A}_1^k \; \cdots \; \mathcal{A}_N^k)^T$, $k = 1, 2$. The objective function in (3.3) is approximated by

$$\sum_k \int_\Omega G^{ij} \frac{\partial \phi^\alpha}{\partial x^i} \frac{\partial \phi^\beta}{\partial x^j} \, d\vec{x} \xi_\alpha^k \xi_\beta^k. \tag{4.1}$$

As for the boundary points, we recall the assumption that ξ map a boundary (linear) segment L on $\partial\Omega$ to a linear segment L_c on $\partial\Omega_c$. This gives that

$$\langle \mathcal{A}_i, \mathbf{n}^i \rangle = b^i, \tag{4.2}$$

where $\langle\,\rangle$ denotes the standard inner product, $\mathbf{n}^i$ is the normal direction of a fixed segment of the boundary of Ω_c, and b^i is a given number. Since each $X_i \in \Gamma_i$ is mapped to a known segment of $\Gamma_{i,c}$, the relevant $\mathbf{n}^i$ and b^i are determined.

4.1. Mesh-Redistribution

We now discuss the part (b) of Algorithm 1. First, a linear system for $\mathcal{A}$ will be formed to determine the motion of the computational grids. Denote

$$H = \left(\int_\Omega G^{ij} \frac{\partial \phi^\alpha}{\partial x^i} \frac{\partial \phi^\beta}{\partial x^j} \, d\vec{x} \right)_{1 \le \alpha, \beta \le N.} \tag{4.3}$$

We further split the matrix H into the following form:

$$H = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix} \qquad \begin{matrix} \leftarrow 1 \text{ to } N_{\text{inner}} \text{ row} \\ \leftarrow N_{\text{inner}} + 1 \text{ to } N \text{ row} \end{matrix}$$
$$\begin{matrix} \uparrow & \uparrow \\ 1 \text{ to } N_{\text{inner}} \text{ column} & N_{\text{inner}} + 1 \text{ to } N \text{ column} \end{matrix}$$

Correspondingly, we use $\mathcal{A}_{\text{inner}}$ and $\mathcal{A}_{\text{bound}}$ to denote the interior and boundary part of the node coordinates, respectively. Then the objective function is given by

$$(\mathcal{A}^{1,T} \quad \mathcal{A}^{2,T}) \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \begin{pmatrix} \mathcal{A}^1 \\ \mathcal{A}^2 \end{pmatrix}. \tag{4.4}$$

Recall the earlier assumption that ξ maps a (linear) boundary segment L on $\partial\Omega$ to a linear segment L_c on $\partial\Omega_c$. This assumption leads to the following linear system

$$(0 \quad A_{12} \quad 0 \quad A_{22}) \begin{pmatrix} \mathcal{A}^1_{\text{inner}} \\ \mathcal{A}^1_{\text{bound}} \\ \mathcal{A}^2_{\text{inner}} \\ \mathcal{A}^2_{\text{bound}} \end{pmatrix} = \vec{b}, \tag{4.5}$$

where the matrices A_{12} and A_{22}, based on (4.2), are the entries of the unit normal of the boundary segments.

With the above preparation, the optimization problem (3.3) is equivalent to solving the linear system

$$\begin{pmatrix} H_{11} & H_{12} & 0 & 0 & 0 \\ H_{21} & H_{22} & 0 & 0 & A'_{12} \\ 0 & 0 & H_{11} & H_{12} & 0 \\ 0 & 0 & H_{21} & H_{22} & A'_{22} \\ 0 & A_{12} & 0 & A_{22} & 0 \end{pmatrix} \begin{pmatrix} \mathcal{A}^1_{\text{inner}} \\ \mathcal{A}^1_{\text{bound}} \\ \mathcal{A}^2_{\text{inner}} \\ \mathcal{A}^2_{\text{bound}} \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vec{b} \end{pmatrix}, \tag{4.6}$$

where λ is the Lagrange multiplier. As the number of points in the computational domain increases, solving the above system efficiently becomes crucial. In our computations, two methods were tested: the first one uses BiCG with an LU preconditioner to solve the system (4.6), and the second is somehow more efficient but not an exact method. Here we briefly outline the idea of the second method. It is to decouple the above system to the following

forms:

$$\begin{pmatrix} H_{22} & 0 & A'_{12} \\ 0 & H_{22} & A'_{22} \\ A_{12} & A_{22} & 0 \end{pmatrix} \begin{pmatrix} \mathcal{A}^1_{\text{bound}} \\ \mathcal{A}^2_{\text{bound}} \\ \lambda \end{pmatrix} = \begin{pmatrix} -H_{21}\mathcal{A}^1_{\text{inner}} \\ -H_{21}\mathcal{A}^2_{\text{inner}} \\ \vec{b} \end{pmatrix} \tag{4.7}$$

and

$$\begin{pmatrix} H_{11} & 0 \\ 0 & H_{11} \end{pmatrix} \begin{pmatrix} \mathcal{A}^1_{\text{inner}} \\ \mathcal{A}^2_{\text{inner}} \end{pmatrix} = -\begin{pmatrix} -H_{12}\mathcal{A}^1_{\text{bound}} \\ -H_{12}\mathcal{A}^2_{\text{bound}} \end{pmatrix}. \tag{4.8}$$

The system (4.7) is small and is solved efficiently by using BiCG and GMRES. The system (4.8) is symmetric and positive definite and is solved by using a multigrid solver. Since the accuracy of the system is not very crucial [the solution of the system (4.6) is for the location of mesh but not for the physical solution], a nonexact but efficient algorithm is more useful. Therefore, we prefer to use the second method in obtaining the approximate solutions of the system (4.6). Since a very good initial guess is available at each time step, we first solve (4.7), with the previous information for the right-hand side $\mathcal{A}_{\text{inner}}$, to update the boundary nodes. Then these new $\mathcal{A}_{\text{bound}}$ are used to update the interior nodes by solving (4.8). In general, no iteration is needed: solving each of (4.7) and (4.8) once is sufficient.

After obtaining the solution of (4.6), we can obtain a new logical mesh $\mathcal{T}_c^*$ with nodes $\mathcal{T}^*$. For a given element E in $\mathcal{T}$, with $X_{E_k}, 0 \le k \le 2$ as its vertices, the piecewise linear map from $V_{\mathcal{T}_c^*}(\Omega_c)$ to $V_{\mathcal{T}}(\Omega)$ has constant gradient on E and satisfies the following linear system:

$$\begin{pmatrix} \mathcal{A}^{*,1}_{E_1} - \mathcal{A}^{*,1}_{E_0} & \mathcal{A}^{*,1}_{E_2} - \mathcal{A}^{*,1}_{E_0} \\ \mathcal{A}^{*,2}_{E_1} - \mathcal{A}^{*,2}_{E_0} & \mathcal{A}^{*,2}_{E_2} - \mathcal{A}^{*,2}_{E_0} \end{pmatrix} \begin{pmatrix} \frac{\partial x^1}{\partial \xi^1} & \frac{\partial x^1}{\partial \xi^2} \\ \frac{\partial x^2}{\partial \xi^1} & \frac{\partial x^2}{\partial \xi^2} \end{pmatrix} = \begin{pmatrix} X^1_{E_1} - X^1_{E_0} & X^1_{E_2} - X^1_{E_0} \\ X^2_{E_1} - X^2_{E_0} & X^2_{E_2} - X^2_{E_0} \end{pmatrix}. \tag{4.9}$$

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 i-th node is defined by

$$\delta X_i = \frac{\sum_{E \in T_i} |E| \left|\frac{\partial \vec{x}}{\partial \xi}\right|_{inE} \delta \mathcal{A}_i}{\sum_{E \in T_i} |E|}, \tag{4.10}$$

where $|E|$ is the volume of the element E, and $\delta \mathcal{A} = \mathcal{A}^{(0)} - \mathcal{A}^*$ is the difference between the fixed mesh $\mathcal{T}_c$ (with nodes $\mathcal{A}^{(0)}$) and the logical mesh $\mathcal{T}_c^*$ (with nodes $\mathcal{A}^*$). 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 $\mathcal{T}^*$ on the physical domain is taken as

$$X^* = X + \tau \delta X, \tag{4.11}$$

where τ is a parameter in [0, 1] and is used to prevent mesh tangling. In our numerical experiments, it is found that the selection of τ is quite insensitive. In 2D, one choice of τ is the following:

$$\tau = \min(0.5/\|\delta \mathcal{A}\|_{2,} 0.618).$$

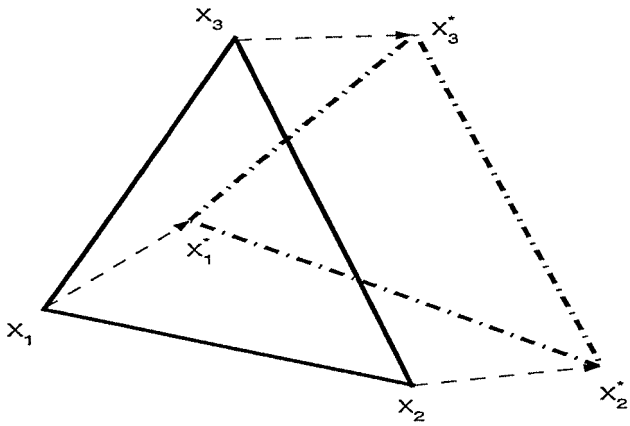


FIG. 2. A demonstration of the element motion.

In 3D, one of the choices for τ is given by (5.2). The motion of one element based on (4.11) is illustrated in Fig. 2.

4.2. Solution-Updating

After the mesh-redistribution in the physical domain Ω, we need to update the solution $\vec{u}$ on the new mesh. This part has been described in [21], and for the sake of clarity we will outline it again. Each element of $\mathcal{T}$ with X as its nodes corresponds uniquely to an element of $\mathcal{T}^*(\tau)$ 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 each element of $\mathcal{T}^*(\tau)$ to $\mathcal{T}$, 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 may be moved to new locations. Then $\vec{u}$, as the function of $\vec{x}$ at time t_n, is independent of the parameter τ. That is

$$\frac{\partial \vec{u}}{\partial \tau} = 0. \tag{4.12}$$

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

$$\vec{u} = \vec{u}(\vec{x}) = \vec{u}(\vec{x}, \tau).$$

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

$$\vec{u} = U_i(\tau)\Phi^i(\vec{x}, \tau), \tag{4.13}$$

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

$$\frac{\partial \Phi^i(\vec{x}, \tau)}{\partial \tau} = -\frac{\partial \Phi^i(\vec{x}, \tau)}{\partial x^j}(\delta \vec{x})_j, \tag{4.14}$$

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

$$0 = \frac{\partial \vec{u}}{\partial \tau} = \frac{\partial U_i}{\partial \tau}\Phi^i(\vec{x},\tau) + U_i(\tau)\frac{\partial \Phi^i}{\partial \tau} = \frac{\partial U_i}{\partial \tau}\Phi^i(\vec{x},\tau) - U_i(\tau)\frac{\partial \Phi^i}{\partial x^j}(\delta\vec{x})_j. \quad (4.15)$$

Using the expression for $\vec{u}$ in the finite element space, i.e., (4.13), we obtain from the above result that

$$\frac{\partial U_i}{\partial \tau}\Phi^i(\vec{x},\tau) - \nabla_{\vec{x}}\vec{u}\delta\vec{x} = 0. \quad (4.16)$$

Then a semidiscrete system for updating $\vec{u}$ follows from the above result:

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

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

$$\int_\Omega \Phi^i \Phi^j\, d\vec{x}\frac{\partial U_i}{\partial \tau} = \int_\Omega \frac{\partial \Phi^i}{\partial x^k}(\delta\vec{x})_k \Phi^j\, d\vec{x} U_i(\tau), \quad 1 \le j \le N. \quad (4.17)$$

The above ODE system can be solved by a 3-stage Runge–Kutta scheme. This procedure, based on the fact that the surface of $\vec{u}$ in Ω is unchanged, provides an interpolation-free solution-updating on the new mesh.

Remark. We close this section with two remarks, one concerns with the computational complexity and another with the boundary projection.

1. *Computational complexity:* Let us make some comparison on operation numbers with the method used in [21] and the one proposed in this work. In the former approach, the major work in obtaining the new grids are to find $\mathcal{A}_{\text{inner}}$ by solving (4.8), where its right-hand side is given. So the extra effort in the present approach is to solve (4.7). Since the sizes of H_{21}, H_{12}, H_{22}, A_{12} and A_{22} are substantially smaller than that of H_{11}, the workload for the method used in [21] is similar, at least asymptotically, to the one in this work.

2. *Boundary projection:* In practical computation, we also need to project the moving vector of the boundary nodes onto the boundary. Otherwise, the residual error accumulation will cause the boundary nodes away from the physical boundary. Although the direction of the motion on the logical boundary is always following the boundary, the mesh-redistribution using the formula (4.11) may push some boundary points away from the physical boundary. One of the remedies is to project them back to the physical domain, as illustrated in Fig. 3.

5. NUMERICAL EXPERIMENTS

In this section, we will implement the moving mesh method described in the last two sections to some test problems. To make some useful comparisons, we begin by revisiting some 2D examples whose moving mesh results have been obtained by several authors; see, e.g., [9, 26, 21].

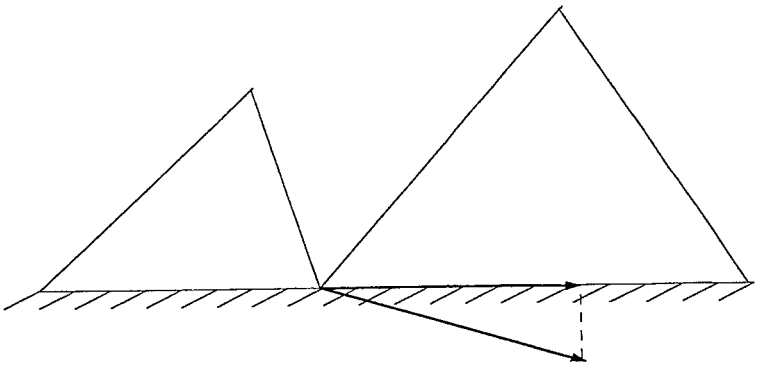


FIG. 3. A demonstration of boundary projection.

5.1. 2D Examples

EXAMPLE 5.1. Our first example in 2D is to compute a moving oblique shock whose governing equation is the Burgers' equation

$$\frac{\partial U}{\partial t} + UU_x + UU_y = a\Delta U, \tag{5.1}$$

defined in the unit square $\Omega = (0, 1)^2$. The initial condition and Dirichlet boundary condition are chosen such that the exact solution to the underlying problem is

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

It should be pointed out that (5.1) is just a special case of Burgers' equation. The standard Burgers' equations in 2D are a system of two PDEs for the velocity components derived from the Navier–Stokes equations. In our computation the viscosity coefficient is chosen as $a = 0.005$, and the monitor function is chosen as $\sqrt{1 + |\nabla U|^2}I$. 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, which makes the use of the moving mesh methods meaningful. Since the exact solution of the underlying problem is given, it is convenient to compare the errors obtained by various methods.

In [21], a simple redistribution strategy is proposed as follows. The basic idea is to move the boundary points by solving 1D moving mesh equations (refer to as the 1D boundary grid redistribution method). Without loss of generality, we consider a simple boundary $[a, b]$ in the x direction. Solving the two-point boundary value problem for $(\omega x_\xi)_\xi = 0$ 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 $\omega|_{[x_j, x_{j+1}]}$ equal this constant, which establishes a connection between the boundary and interior grid redistributions.

In Fig. 4, the L^1-errors obtained by using the 1D boundary grid redistribution method of [21] and the optimization-based approach proposed in this work are compared, with no significant difference observed. In Fig. 5, we plot the moving meshes obtained by using the moving mesh approach proposed in this work with 20×20 nodes (about 930 triangular

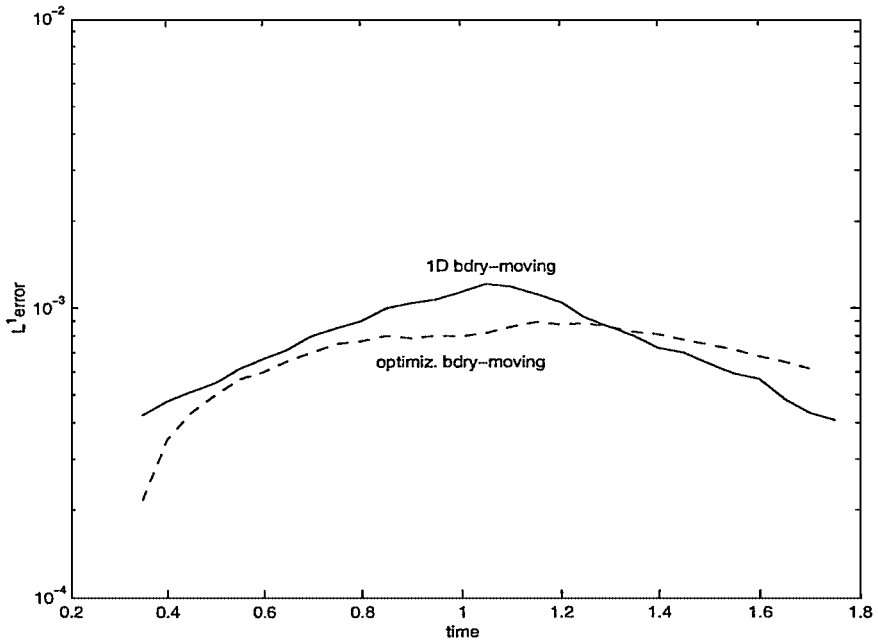


FIG. 4. Example 5.1: The L^1-error in time obtained by solving the Euler-Lagrange equation [21] (solid line) and by using the optimization-based method proposed in this work (dashed line), with 20×20 nodes.

elements). It is observed that they are very similar to those obtained in [21] where the meshes are moved by solving the Euler–Lagrange equations.

Our next numerical example is a coupled nonlinear reaction–diffusion system modeling a combustion process [1, 17]. The main purpose of this example is to demonstrate that for some problems there is indeed some difference between the moving mesh methods described in [21] and in this work.

EXAMPLE 5.2. The mathematical model is a system of coupled nonlinear reaction–diffusion equations

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

for $\vec{x} = (x_1, x_2)^T \in \Omega = (-1, 1)^2$, $t > 0$, and where u and T represent, respectively, the dimensionless species concentration and temperature of a chemical which is undertaking a one-step reaction. The initial and boundary conditions are

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

The physical parameters are set to be $Le = 0.9$, $\alpha = 1$, $\delta = 20$, and $R = 5$. In this problem, a sharp wave front is moving toward the boundary $\partial\Omega$. A detailed numerical procedure for time discretization of this problem was provided in [21]. In our computation, we used 30×30 nodes and the monitor function is chosen as $\sqrt{1 + |\nabla T|^2} I$. Figure 6 depicts the adaptive meshes at $t = 0.283$ obtained by solving the Euler–Lagrange equation

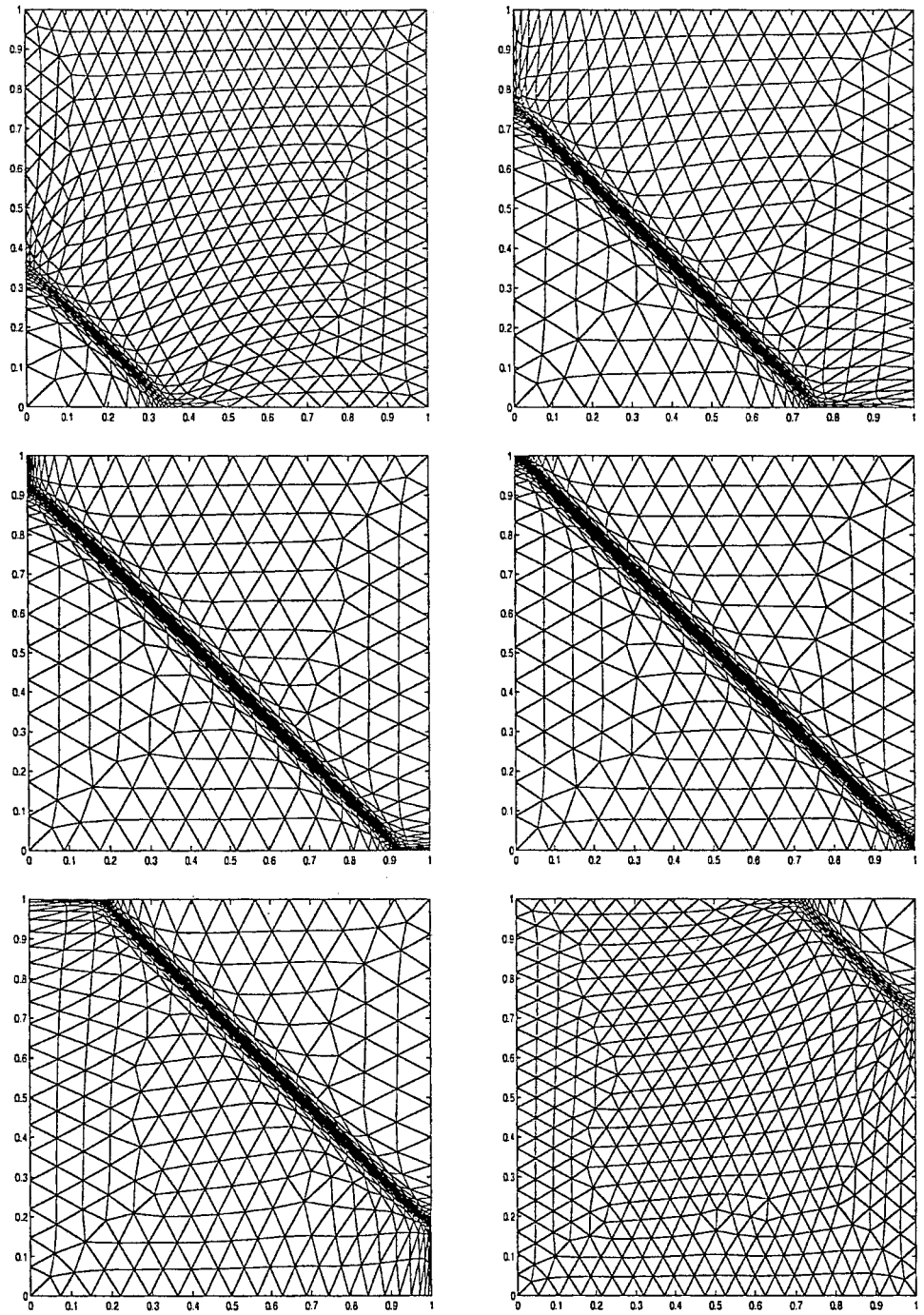


FIG. 5. Example 5.1: The adaptive meshes using 20^2 nodes from $t = 0.15$ to $t = 1.85$, with the moving mesh approach proposed in this work.

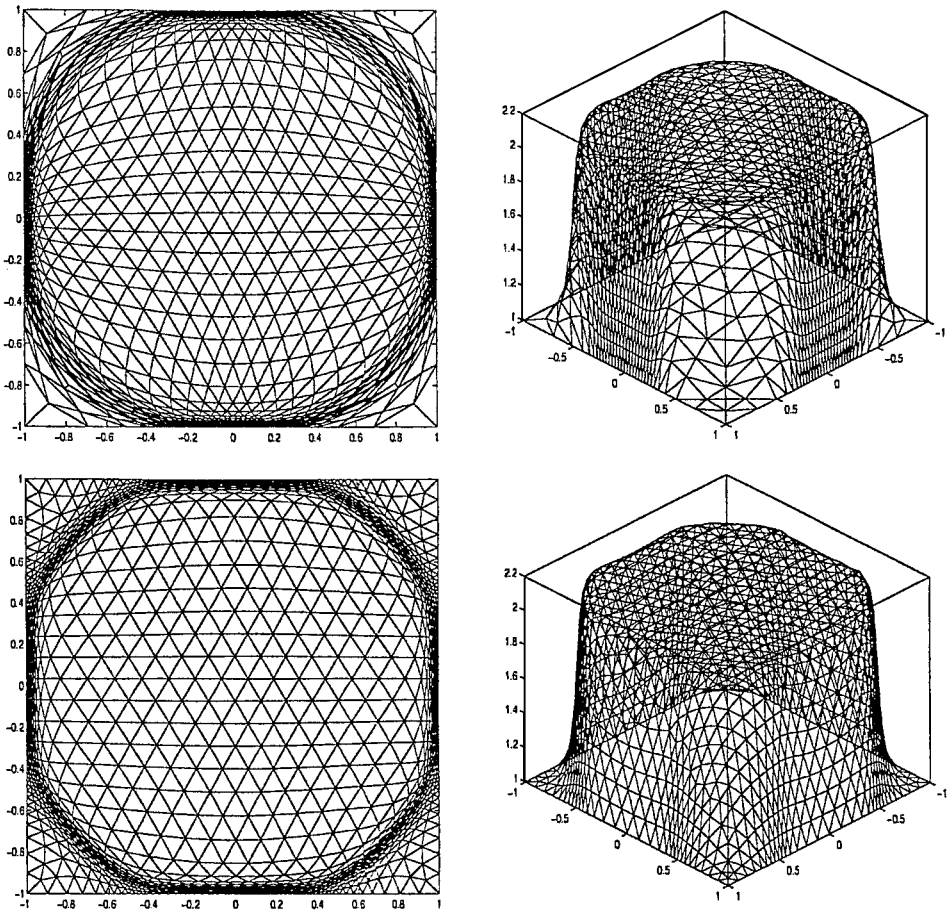


FIG. 6. Example 5.2: The adaptive mesh and temperature T obtained by using the method of [21] (top) and the one proposed in this work (bottom), at $t = 0.283$.

together with the 1D boundary grid-redistribution technique and by the method proposed in this work. It is observed that both methods can efficiently solve this nonlinear system. However, it is quite obvious that the meshes obtained by using the optimization method appear to be more reasonable, which adapts more grid points into the reaction interface. The solution errors, obtained by using a benchmark solution with very fine mesh, are compared in Fig. 7. The results indicate that the L^1- and L^∞-errors are reduced by using the optimization approach, although the reduction is not significant.

5.2. 3D Examples

For simplicity, we assume that the solution domain is a unit cube $[0, 1]^3$. Initially, the physical domain is divided into some small cubes (with same size), and then every cube is cut into six tetrahedron. In this case, the logical domain and its initial mesh are chosen as the same as the corresponding counterpart in the physical domain.

In order to avoid possible mesh tangling, the following strategy is proposed. In the mesh-motion step, consider a tetrahedron with vertices $\vec{x}_i$, whose moving directions are $\delta\vec{x}_i$, $0 \le i \le 3$. It can be shown that the mesh may be tangled if the step length τ used in

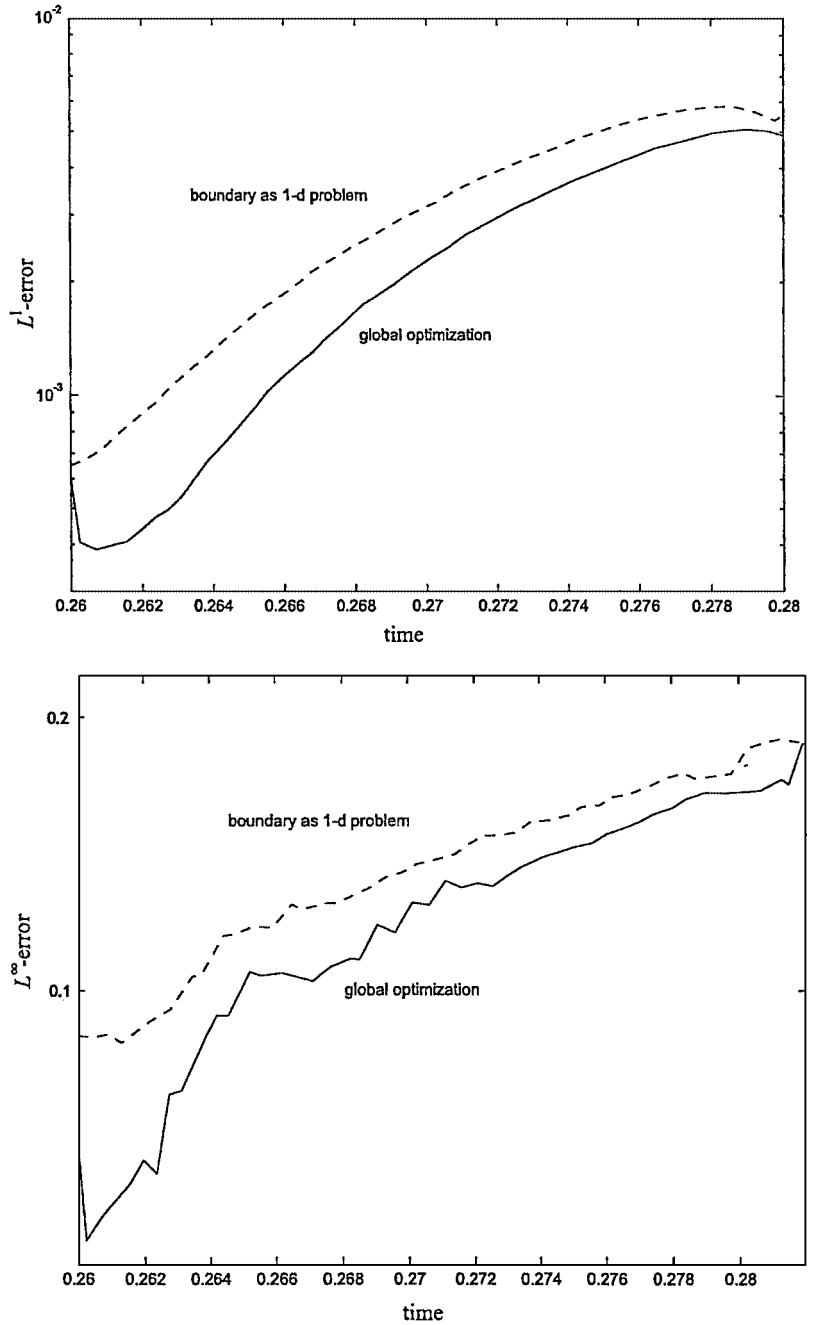


FIG. 7. Numerical errors for Example 5.2 obtained by using the 1D boundary-grid re-distribution (dashed line) and by using the optimization approach proposed in this work (solid line): upper figure is for L^1-error and lower one is for L^∞-error.

the mesh motion formula (4.11) is larger than a critical number τ^*, which is defined as the least positive root for the equation

$$\det\begin{pmatrix} 1 & 1 & 1 & 1 \\ \vec{x}_0 - \tau\delta\vec{x}_0 & \vec{x}_1 - \tau\delta\vec{x}_1 & \vec{x}_2 - \tau\delta\vec{x}_2 & \vec{x}_3 - \tau\delta\vec{x}_3 \end{pmatrix} = 0. \tag{5.2}$$

The parameter τ used in (4.11) will be set as half of the minimal τ^* over all the tetrahedron. In our numerical computations, it is found that the selection of $\tau = 0.5$ for (4.11) causes no numerical difficulties (i.e. no mesh tangling) in almost all situations.

EXAMPLE 5.3. Our first example in 3D is concerned with a special Burgers' equation

$$\frac{3}{2}\frac{\partial U}{\partial t} + UU_x + UU_y + UU_z = a\Delta U, \tag{5.3}$$

defined in $\Omega = (0, 1)^3$. The initial condition and Dirichlet boundary condition are chosen such that the exact solution to the underlying problem is

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

The problem (5.3) is now a very special case of Burgers' equation. The standard Burgers' equations in 3D are a system of three PDEs for the velocity components derived from the Navier–Stokes equations. In our computation, the diffusion coefficient is again chosen as $a = 0.005$, and the monitor function is chosen as $\sqrt{1 + |\nabla U|^2}I$. For this small coefficient a, the underlying physical problem corresponds to a viscous shock traveling from the lower left corner to the upper right corner. Note that the exact solution has very large gradients along the 2D plane $x + y + z = t$ for $0 < t < 3$.

In Fig. 8, the L^1- and L^∞-errors obtained by using the uniform mesh and the moving mesh are displayed. In both cases, the solution domain is divided into $17 \times 17 \times 17$ small cubes, and then every cube is cut into six tetrahedron. It is observed that the L^1-error, as a function of time, is almost oscillation free, and in average is about 10 times smaller than that for the uniform mesh. Moreover, Fig. 8(b) indicates that the moving mesh solution yields about four times error reduction. In Fig. 9, the grid distribution at various times is demonstrated. In the region before the viscous shock front, the numerical solution is approximately 1, and after the viscous shock it is about 0. Figure 9 clearly indicates that our moving mesh algorithms cluster quite a number of grid points in the region where the solution has very large gradients. It is also seen that the *locations* of the viscous shock at various times are clearly presented, which provides one of the most useful information for the viscous shock problems.

EXAMPLE 5.4. The last example is the same as Example 5.2, except that the solution domain becomes $\Omega = (-1, 1)^3$.

In Example 5.3, the special viscous Burgers' problem is a scalar equation whose largest solution gradients occur in a simple 2D plane. In Example 5.4, the problem is a nonlinear *system* whose largest solution gradients occur in a quite complicated surface, which are more or less like a sphere $x^2 + y^2 + z^2 = r^2$, where the radius r is a function of time. We solve the underlying PDEs by simply extending the 2D finite element codes developed in [21]: we first consider a finite element spatial discretization, leaving the time variable continuous (method of lines); and then a RK3 is used for the time discretization. The mesh-motion strategy proposed in Algorithm 1 is employed. The monitor function used is $\sqrt{1 + |\nabla T|^2}I$.

Figure 10 depicts the moving meshes at various times on the plane $y + z = 0$ obtained by using the same number of nodes as in the last example. The numerical solution for the temperature is 1 outside the circular region and a constant greater than 1 inside the circle. It is observed that although the temperature T has a very thin layer of large variation, our

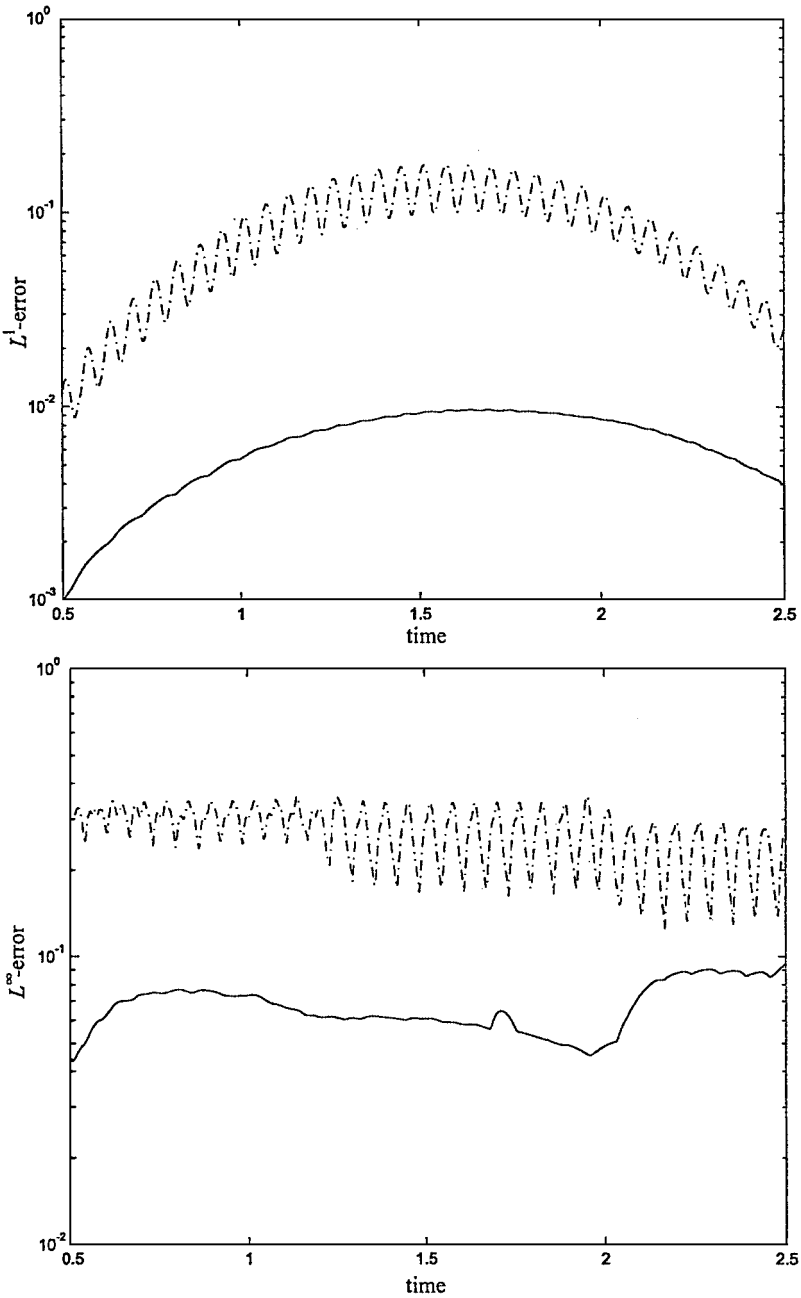
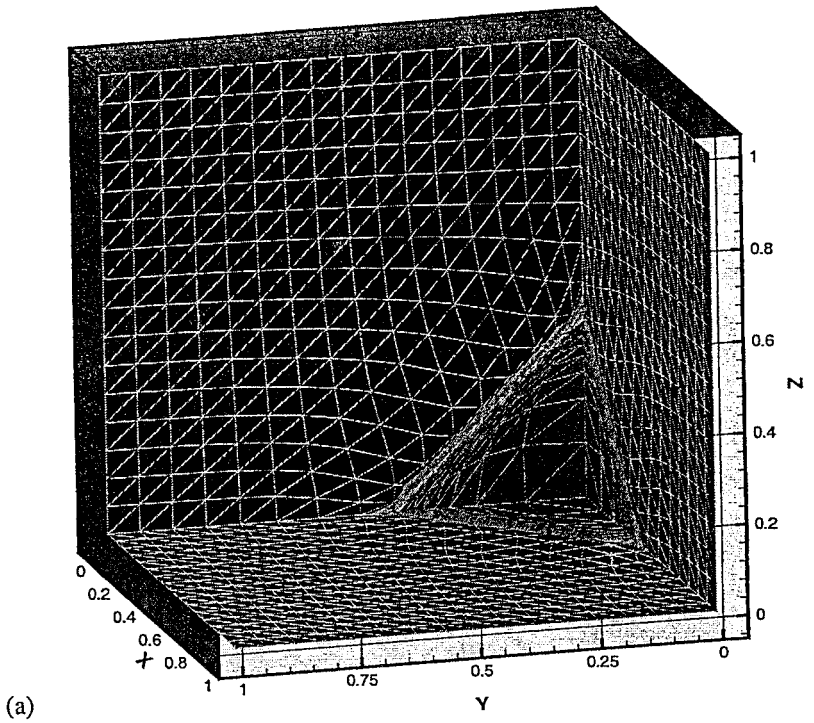
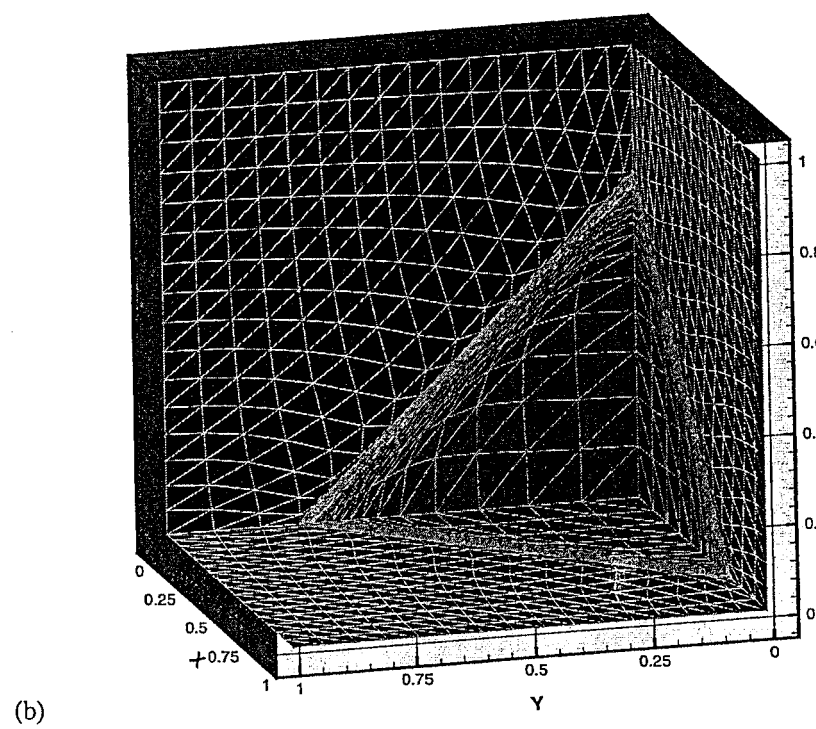


FIG. 8. Numerical errors for 3D Burgers' equation: upper figure is for L^1-error and lower is for L^∞ error. In each figure, the dashed curve is with the uniform mesh and the solid curve is with the moving mesh.

moving mesh scheme adapts the mesh extremely well to the regions with large solution gradients. Unlike the regular finite element meshes, an arbitrary cut-plane may not have enough grid points for the moving meshes. With the graphics software, the nearby points are projected to the chosen plane. As observed in Fig. 10, the projection may yield a few *spurious* elements, since there are only (in average) 17×17 nodes on each given plane.

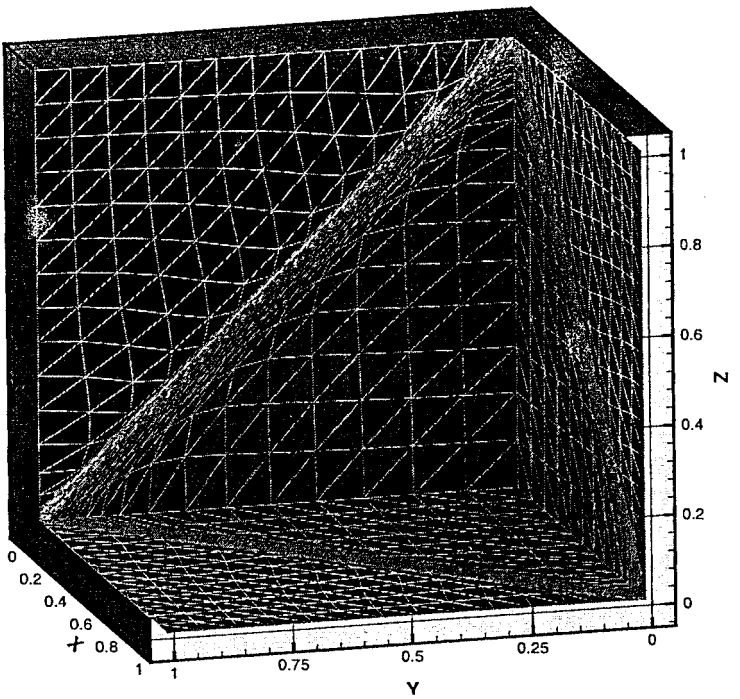


(a)

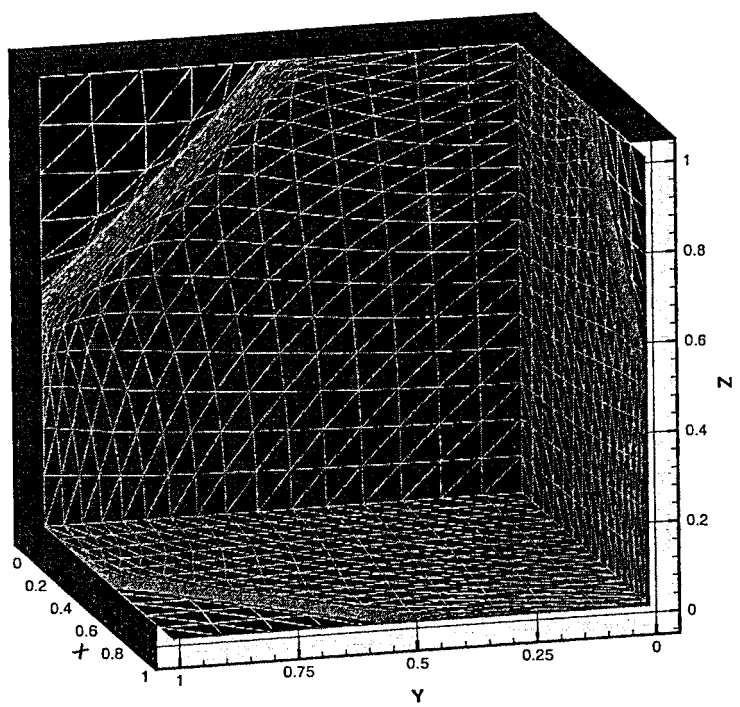


(b)

FIG. 9. The grid distribution for Example 5.3 at (a): $t = 0.4$, (b): $t = 0.7$, (c): $t = 1$ and (d): $t = 1.5$.



(c)



(d)

FIG. 9—*Continued*

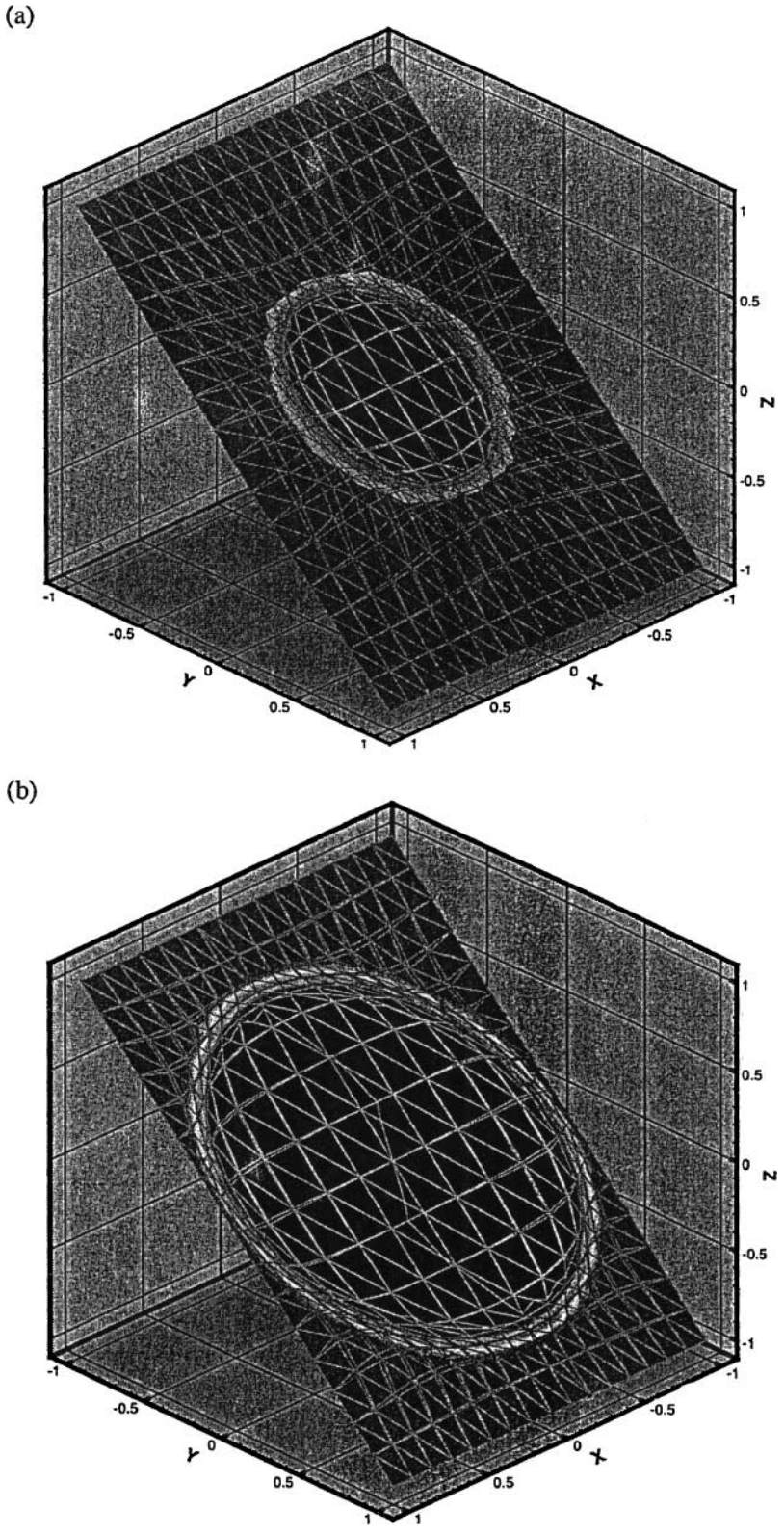
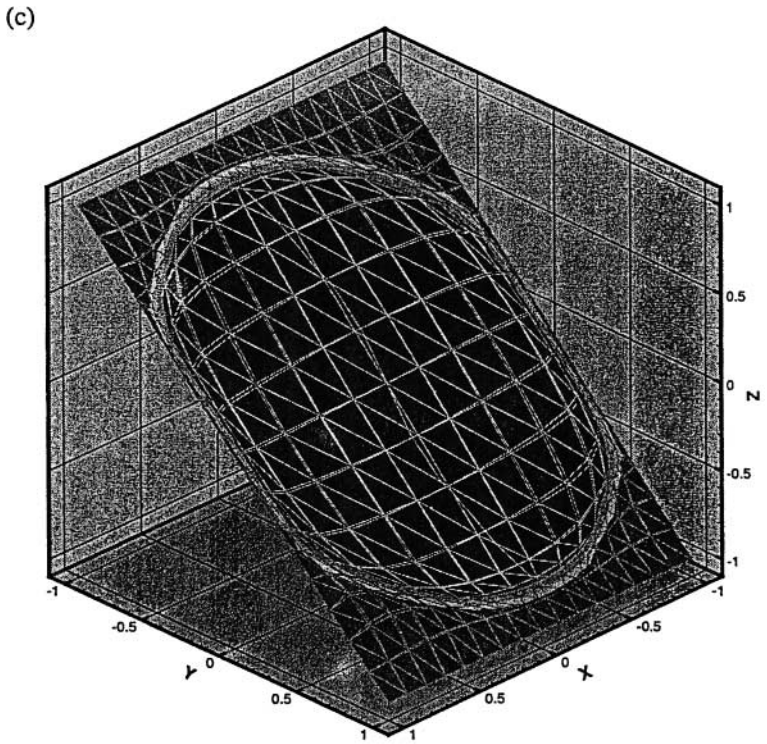


FIG. 10. Example 5.4: The grid distribution on the cut-plane $y + z = 0$ at (a): $t = 0.312$, (b): $t = 0.324$, (c): $t = 0.331$ and (d): $t = 0.335$.



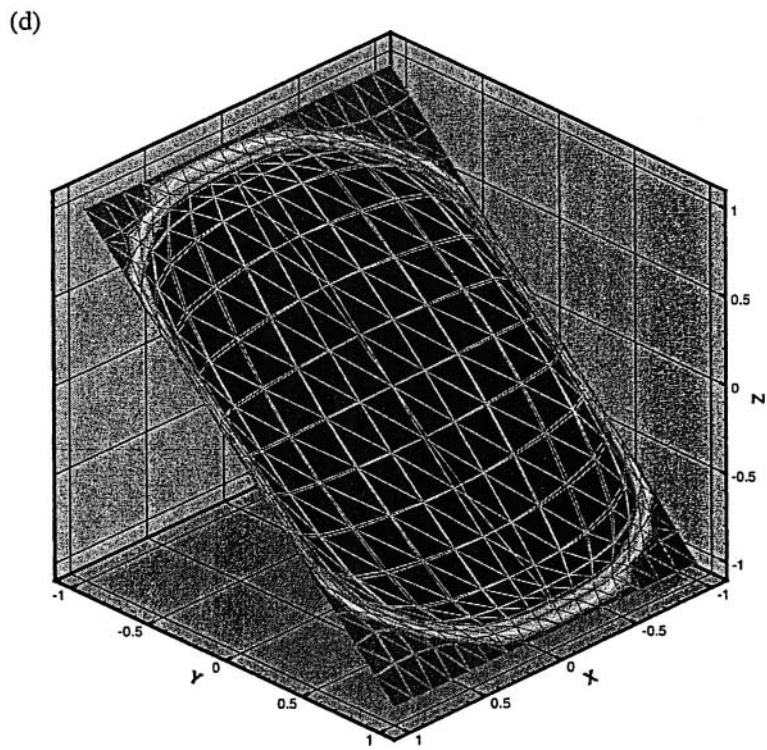


FIG. 10—*Continued*

5.3. Some Computational Details

We close this section with some discussions on some computational details. The first issue concerns the monitor function $(G^{ij})^{-1}$. There are several papers investigating the selection of the monitor functions, including Beckett *et al*. [5, 6] and Brackbill [7]. One observation in our numerical experiments is that the numerical algorithm developed in this work is robust for the selection of the monitor functions. For both the viscous shock and the combustion problems, a simple monitor function $\sqrt{1 + c|\nabla u|^2}I$ works very well, where u is the underlying physical solution and $c > 0$ is some constant. In our numerical experiences, some moving mesh methods, such as moving finite-volume method (see, e.g., [31]), and finite-difference method (see, e.g., [30]), depend heavily on the choice of feasible values of c, in particular for three-dimensional computations. However, in the numerical experiments of this work, we can always choose $c = 1$, and the numerical solutions are found quite insensitive to the choice of the parameter c. Possible reasons leading to this good property are the use of the harmonic mapping and/or the interpolation-free on the updated moving grids.

The second issue is about the number of iterations needed in the construction of the harmonic map, which is crucial in order to access the computational cost of the proposed method. The answer to this question is related to the choice of the tolerance TOL in (3.5). In our computations, it is found that setting

$$\mathsf{TOL} = \text{minimum element diameter in } \Omega_c \times 10\%$$

is sufficient in giving satisfactory mesh-redistributions. For all of the examples considered in this section,

- TOL used is of the order 10^{-2};
- the initial mesh constructions require 10 to 20 iterations;
- at any time level $t_n > 0$, one or two iterations are sufficient to realize Algorithm 1 in Sect. 3.2.

There is no difference for the number of iterations between 2D and 3D. However, when the geometry becomes quite complicated, the number of iterations may increase. For example, with the U-shape domain of Example 6.3 in [21] 1 to 5 iterations are needed at various time steps, but in average the number of iterations is not more than two.

The third issue concerns filtering or smoothing of the monitor functions. 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. 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 Li *et al*. [21], a smoothing procedure was proposed: The monitor function $M := G^{-1}$ was firstly interpolated from $L^2(\Omega)$ into $H_{1,h}(\Omega)$, namely from piecewise constant to piecewise linear, by the following formula:

$$(\pi_h M)|_{\text{at } P} = \frac{\sum_{\tau: P \text{ is vertex of } \tau} M|_{\text{on}\,\tau}|\tau|}{\sum_{\tau: P \text{ is vertex of } \tau}|\tau|}. \tag{5.4}$$

Then it is projected back into $L^2(\Omega)$ by the following formula:

$$M|_{\text{on}\,\tau} = \frac{1}{n+1}\sum_{P \text{ is vertex of } \tau}(\pi_h M)_{\text{at } P}, \tag{5.5}$$

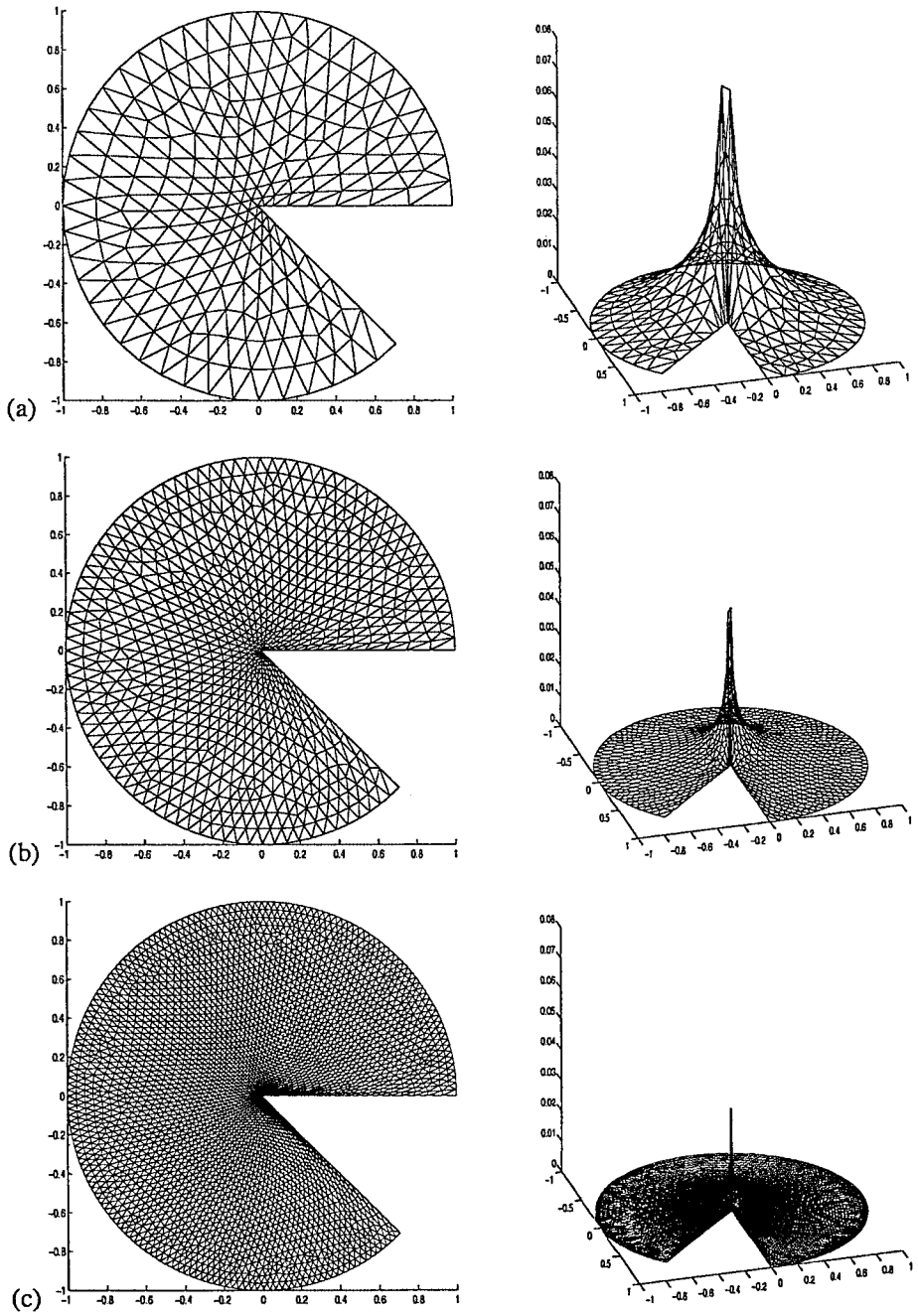


FIG. 11. The adaptive meshes (left) and pointwise errors (right) for Example 6.1: (a) $N = 416$, 13 iterations; (b) $N = 1558$, 18 iterations; (c) $N = 6068$, 26 iterations.

where n is the dimension of Ω. The same smoothing technique is applied in this work, for both 2D and 3D computations. Our numerical experiments have shown that this smoothing procedure not only enhances the quality of the meshes but also increases the accuracy of the numerical approximations.

6. QUALITY OF THE GENERATED MESHES

It would be interesting to know how close the generated meshes obtained by our moving mesh algorithm are to an optimal mesh. Mesh optimality in this context means that L^2- or H^1-error of the finite element approximation converges with the optimal order as the mesh size tends to zero. To this end, we will consider a 2D time-independent problem with a corner singularity. The convergence rate of the finite element approximation on uniform grids is determined by the strength of the singularity. On optimally graded meshes, the optimal convergence can be recovered.

EXAMPLE 6.1. The governing equation is a Laplace's equation with a Dirichlet boundary condition,

$$\begin{aligned} \Delta u &= 0, && (x, y) \in \Omega, \\ u &= r^{2/7} \sin(2\theta/7), && (x, y) \in \partial\Omega, \end{aligned}$$

where $\Omega = \{r < 1,\ 0 < \theta < 7\pi/4\}$, and r and θ are the usual polar coordinates.

The above example has been used by many authors; see, e.g., Tourigny and Hülsemann [32] where a moving finite element algorithm was proposed for variational problems and a detailed analysis for convergence rate in $H_0^1(\Omega)$ is presented. We point out that this is not the most appropriate example for our purpose to test the mesh quality, since quite weak singularity appears at an isolate point. Our main objective for the moving mesh methods is to resolve large gradients on a curve (in 2D) or on a plane (in 3D), such as the problems in Section 5. Moreover, this is not a time-dependent problem. However, since this problem is well studied and the exact solution is known, we will use it to obtain some ideas on the mesh quality. For the time-independent problem, the Algorithm 1 will be modified as below:

ALGORITHM 2 (MOVING MESH ALGORITHM FOR ELLIPTIC EQUATIONS)

(a): solve the optimization problem (3.3) and compute the L^∞-difference between the solution of (3.3) and the fixed (initial) mesh in the logical domain. If the difference is smaller than a preassigned tolerance TOL, then the mesh-redistribution is complete. Otherwise, do

(b): obtain the direction and the magnitude of the movement for $\vec{x}$ by using the difference obtained in part (a), see (4.10) in Section 5, and then move the mesh based on (4.11);

(c): solve the Laplace's problem with finite element method on the new mesh obtained in part (b);

(d): update the monitor function by using $\vec{u}$ obtained in part (c), and go to part (a).

TABLE I
Example 6.1: The Rate of Convergence of Error in L^2-Norm

Moving mesh			Uniform mesh		
N	$\|I_h u - u_h\|_{L^2(\Omega)}$	Order	N	$\|I_h u - u_h\|_{L^2(\Omega)}$	Order
416	1.62422e-2	—	416	3.21009e-2	—
1558	6.04666e-3	1.50	1558	1.97644e-2	0.73
6068	2.04652e-3	1.59	6068	1.15090e-2	0.80
24,142	6.24831e-4	1.72	24,142	6.58991e-3	0.81

TABLE II
Example 6.1: The Rate of Convergence of Error in H_0^1-Norm

Moving mesh			Uniform mesh		
N	$\|I_h u - u_h\|_{H_0^1(\Omega)}$	Order	N	$\|I_h u - u_h\|_{H_0^1(\Omega)}$	Order
416	1.53211e-1	—	416	2.07425e-1	—
1558	1.03804e-1	0.59	1558	1.73109e-1	0.27
6068	6.69192e-2	0.65	6068	1.46320e-1	0.25
24,142	4.06005e-2	0.72	24,142	1.23653e-1	0.24

It should be pointed out that due to weak singularity, large adaptation constant c in the monitor $\sqrt{1 + c|\nabla u|^2} I$ has to be used for this example, which is in contrast to the examples in Section 5 where $c \equiv 1$ is used. For this problem, c should be chosen in the range $10^6 \sim 10^8$; otherwise almost no adaptation effects can be obtained. In Fig. 11, the meshes, with 238, 838, and 3150 vertices and 416, 1558, and 6038 triangles, are shown respectively. The parameter values $\mathsf{TOL} = 2 * 10^{-3}$ for the tolerance in the above algorithm and $c = 10^8$ for the monitor constant are used. The tolerance used is demonstrated sufficiently small by comparing the solution errors. The numbers of iteration used for are 13, 18 and 26, respectively.

Following Tourigny and Hülsemann [32], we will denote by e_N the error for a mesh with N triangles, and

$$e_N \sim N^{-\gamma/2}.$$

Therefore, a local decay rate γ_{MN} is given by

$$\gamma_{MN} = \frac{\log(e_M/e_N)}{\log\sqrt{M/N}}, \quad M > N.$$

Moreover, $\|I_h u - u_h\|$ will be used to represent the errors of $\|u - u_h\|$, where $I_h u$ is the interpolant of the exact solution u. The reates of convergence in L^2 and H_0^1 are shown in Tables I and II. It is observed that the approximation on the adaptive meshes appears to converge at higher convergence orders than that on uniform meshes. However, the optimal rates of convergence in neither L^2 nor H_0^1 are recovered, although the improvement with adaptive mesh is quite substantial, in particular in L^2 space.

7. CONCLUDING REMARKS

In this work, we have extended an earlier work of moving mesh methods based on harmonic maps [21] to deal with mesh adaptation in three space dimensions. The main difference between the numerical scheme proposed in this work and the one proposed in [21] is the following: In obtaining the variational mesh, we solve an *optimization problem* with some boundary-point constraints. But in [21], the moving meshes are obtained by solving the Euler–Lagrange equation, together with the boundary-point adjustments. In the latter case, the grid redistributions for the interior domain and boundaries are carried out *separately*. However, this approach seems difficult when being extended to 3D problems, in particular if a finite element approach is employed for spatial discritization. In this work, by

solving the constrained optimization problem we are able to implement the moving mesh mehods to three-dimensional problems successfully. Several goods features of our moving mesh algorithm have been observed: e.g., the commonly used solution interpolation can be avoided; the algorithm is robust to the adaptation constant in front of the derivatives in the monitor functions; very few iterations are needed in the mesh-redistribution process.

The next step of the research is to apply our moving mesh algorithm to solve some practical problems. Some useful applications based on various moving mesh techniques can be found in a number of papers, including [3, 11, 28, 31] for computational fluid dynamics, [12, 18] for computational geometry and [27] for simulations of the reactive flows. In solving real-life problems with moving mesh strategy, some additional difficulties may occur. For example, when using the moving mesh methods to solve hyperbolic system of conservation laws, the additional requirements, such as mass conservation and entropy consistency, may introduce extra numerical difficulties in the mesh moving process. As a result, there are only very few successful moving mesh results for multidimensional hyperbolic system, in particular for 3D. In our future study, we will apply the moving mesh algorithms developed in this work to some practical problems in areas such as fluid mechanics, chemical engineering, and image processing. Our particular attention will be given to solving the practical problems in higher space dimensions.

ACKNOWLEDGMENTS

This work was supported by the Special Funds for Major State Basic Research Projects of China (Grant G1999032804) and the Hong Kong Research Grants Council (Grant 2033/99P and 2044/00P). We thank the referees for very careful reading of the manuscript and for many useful suggestions.

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 (1986).
2. B. N. Azarenok, Variational barrier method of adaptive grid generation in hyperbolic problems of gas dynamics, *SIAM J. Numer. Anal.*, to appear. Available at `http://www.math.ntnu.no/conservation/2001/042.html`.
3. B. N. Azarenok and S. A. Ivanenko, Application of adaptive grids in numerical analysis of time-dependent problems in gas dynamics, *Comput. Maths. Math. Phys.* **40**, 1330 (2000).
4. M. J. Baines, *Moving Finite Elements* (Oxford Univ. Press, London 1994).
5. 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.* **167**, 372 (2001).
6. G. Beckett, J. A. Mackenzie, and M. L. Robertson, A moving mesh finite element method for the solution of two-dimensional Stephan problems, *J. Comput. Phys.* **168**, 500 (2001).
7. J. U. Brackbill, An adaptive grid with directional control, *J. Comput. Phys.* **108**, 38 (1993).
8. J. U. Brackbill and J. S. Saltzman, Adaptive zoning for singular problems in two dimensions, *J. Comput. Phys.* **46**, 342 (1982).
9. 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 (1999).
10. N. Carlson and K. Miller, Design and application of a gradient-weighted moving finite code, Part II, 2-D, *SIAM J. Sci. Comput.* **19**, 766 (1998).
11. H. D. Ceniceros and T. Y. Hou, An efficient dynamically adaptive mesh for potentially singular solutions, *J. Comput. Phys.* **172**, 609 (2001).

12. V. Critini, J. Blawzdziewicz, and M. Loewenberg, An adaptive mesh algorithm for evolving surfaces: simulations of drop breakup and coalescence, *J. Comput. Phys.* **168**, 445 (2001).
13. E. A. Dorfi and L. O'c. Drury, Simple adaptive grids for 1-D initial value problems, *J. Comput. Phys.* **69**, 175 (1987).
14. A. S. Dvinsky, Adaptive grid generation from harmonic maps on Riemannian manifolds, *J. Comput. Phys.* **95**, 450 (1991).
15. J. Eell and J. H. Sampson, Harmonic mappings of Riemannian manifolds, *Amer. J. Math.* **86**, 109 (1964).
16. R. Hamilton, *Harmonic Maps of Manifolds with Boundary,* Lecture Notes in Mathematics (Springer-Verlag, New York, 1975), Vol. 471.
17. A. K. Kapila, *Asymptotic Treatment of Chemically Reacting Systems* (Pitman, Boston, 1983).
18. S. Kawk and C. Pozrikidis, Adaptive triangulation of evolving, closed, or open surfaces by the advancing-front method, *J. Comput. Phys.* **145**, 61 (1998).
19. R. Li, *Moving Mesh Method and its Application,* Ph.D. thesis (in Chinese) (School of Mathematical Sciences, Peking University, May 2000).
20. R. Li, W.-B. Liu, H.-P. Ma, and T. Tang, Adaptive finite element approximation for distributed elliptic optimal control problems, submitted for publication (under revision). Also available at `http://www.math.hkbu.edu.hk/~ttang.`
21. R. Li, T. Tang, and P. W. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, *J. Comput. Phys.* **170**, 562 (2001).
22. G. Liao, A study of regularity problem of harmonic maps, *Pacific J. Math.* **131**, 291 (1988).
23. G. Liao and N. Smale, *Harmonic Maps with Nontrivial Higher-Dimensional Singularities,* Lecture Notes in Pure Applied Mathematics (Dekker, New York, 1993), Vol. 144, pp. 79–89.
24. F. Liu, S. Ji, and G. Liao, An adaptive grid method and its application to steady Euler flow calculations, *SIAM J. Sci. Comput.* **20**, 811 (1998).
25. K. Miller and R. N. Miller, Moving finite element methods I, *SIAM J. Numer. Anal.* **18**, 1019 (1981).
26. P. K. Moore and J. E. Flaherty, Adaptive local overlapping grid methods for parabolic systems in two space dimensions, *J. Comput. Phys.* **98**, 54 (1992).
27. E. S. Oran and J. P. Boris, *Numerical Simulations of Reactive Flow,* 2nd ed. (Cambridge Univ. Press, Cambridge, UK, 2000).
28. W. Ren and X.-P. Wang, An iterative grid redistribution method for singular problems in multiple dimensions, *J. Comput. Phys.* **159**, 246 (2000).
29. R. Schoen and S.-T. Yau, On univalent harmonic maps between surfaces, *Invent. Math.* **44**, 265 (1978).
30. J. M. Stockie, J. A. Mackenzie, and R. D. Russell, A moving mesh method for one-dimensional hyperbolic conservation laws, *SIAM J. Sci. Comput.* **22**, 1791 (2001).
31. H.-Z. Tang and T. Tang, Moving mesh methods for one- and two-dimensional hyperbolic conservation laws. Preprint, 2001. Available at `http://www.math.ntnu.no/conservation/2001/014.html.`
32. Y. Tourigny and F. Hülsemann, A new moving mesh algorithm for the finite element solution of variational problems, *SIAM J. Numer. Anal.* **35**, 1416 (1998).
33. Y. Tourigny and M. J. Baines, Analysis of an algorithm for generating locally optimal meshes for L_2 approximation by discontinuous piecewise polynomials, *Math. Comp.* **66**, 623 (1997).
34. A. Winslow, Numerical solution of the quasi–linear Poisson equation, *J. Comput. Phys.* **1**, 149 (1967).