Journal of Computational Physics **149,** 221–244 (1999)
Article ID jcph.1998.6151, available online at http://www.idealibrary.com on IDEAL®

# An *r*-Adaptive Finite Element Method Based upon Moving Mesh PDEs

Weiming Cao,* Weizhang Huang,† and Robert D. Russell*

**Department of Mathematics and Statistics, Simon Fraser University, Burnaby, British Columbia V5A 1S6, Canada; †Department of Mathematics, the University of Kansas, Lawrence, Kansas 66045*
E-mail: wcao@cs.sfu.ca, huang@math.ukans.edu, rdr@cs.sfu.ca

Received July 30, 1998; revised November 5, 1998

We present an *r*-adaptive finite element method for solving time dependent partial differential equations. A moving mesh partial differential equation, or MMPDE, is used to move the (unstructured) mesh in time. A key to the application of the MMPDE to unstructured mesh movement is to define a computational domain and then compute the corresponding computational mesh as the image of an initial mesh on the given physical domain. The finite element discretization of physical PDEs on moving meshes is addressed. Numerical results are presented to demonstrate the capability of the mesh movement strategy and the *r*-adaptive finite element method. A fully developed *r*-adaptive finite element method can be expected to be ideally suited to complement the currently popular *h*-*p* finite element methods and to provide increased reliability and efficiency for mesh adaptation. © 1999 Academic Press

*Key Words:* moving mesh method; adaptive finite element method; unstructured mesh adaptation.

## 1. INTRODUCTION

As is well known, adaptive techniques can greatly improve the accuracy and efficiency of finite element methods (FEMs) for solving partial differential equations (PDEs) by concentrating the elements in regions where the solution changes rapidly. Adaptivity is particularly advantageous when the region requiring high resolution consists of only a small fraction of the entire domain. There are three main types of adaptive techniques for the finite element method: (i) the *h*-method, which refines and coarsens the mesh locally according to certain error indicators, (ii) the *p*-method, which selects the polynomial degree used in the finite element approximation in each element according to the smoothness of the solutions, and (iii) the *r*-method, or *moving mesh method*, which relocates the element vertices (mesh points) to concentrate them in the desired regions.

There have been extensive studies of the *h* and *p* methods, as well as their combination, the *h*-*p* method, and they have proven to be successful for a wide variety of problems in



0021-9991/99 $30.00
Copyright © 1999 by Academic Press
All rights of reproduction in any form reserved.

computational mechanics and fluid dynamics. The $r$-method has been less popular for the finite element community. The main difficulty seems to be the lack of a reliable, efficient, and general procedure to determine the mesh movement. Nevertheless, the $r$-method offers features distinct from those of the $h$ and $p$ methods, viz., the mesh changes continuously, making it easier to incorporate a time integrator, and the data structure is simpler, making it easier to implement. These attractive features have of course been utilized in previous studies. Miller [22] first introduced moving FEMs, and for his approach both the moving mesh and the numerical approximation of the physical solution are determined by minimizing the residual of the PDEs over an enlarged trial subspace composed of the usual basis functions and their derivatives. For steady state problems or problems leading to an equilibrium state, this method can result in optimal solutions in certain norms for all possible choices of the meshes with the connectivity fixed [21]. For some nonlinear convection diffusion problems, this moving FEM has produced quite accurate solutions with very small numbers of elements [8, 22]. However, since the minimization procedure can become degenerate, its successful application requires the proper modulation of some penalty terms. Care must also be taken to prevent mesh tangling [8]. In recent work, these deficiencies are remedied to some extent by the local refinement technique. Similar ideas for determining the mesh movement by minimizing the residuals are utilized by Baines [2].

There are several theoretical studies of the FEM for nonfixed meshes. For instance, Dupont [10] obtained error estimates for the FEM using meshes changing with time (continuously or discontinuously). Bank and Santos [3] used the framework of space-time finite element approximations with changing meshes and obtained a symmetric error estimate. But in these analyses, no particular technique for moving the meshes is advocated.

Moving mesh methods have attracted considerable attention in the mesh generation community, especially for the solution of aerodynamics problems using the finite difference method and structured grids. Several techniques for creating and relocating the meshes have been advocated, perhaps most notably the method based upon solving elliptics PDEs [4, 5, 26]. In recent work of Huang and Russell [17, 18], a moving mesh technique is developed which determines the mesh movement by solving a system of parabolic equations called the moving mesh partial differential equations (MMPDEs). The basic idea of the approach is to formulate the MMPDE as the gradient flow equation of a functional which measures the approximation difficulty of the physical solution. Mesh adaptation to the underlying physical solution, mesh alignment to some known vector fields, and mesh orthogonality control are all taken into account in the definition of the mesh generation functional. For structured grids, the finite difference discretization of the MMPDE produces quite satisfactory meshes which move smoothly in time and are concentrated in regions where the solution changes rapidly.

In this paper, we describe an $r$-adaptive finite element method based upon the MMPDEs developed in [17, 18]. The key to this application to unstructured mesh movement is to define and compute the computational domain and computational mesh (see Section 4). Having obtained the computational mesh, the adaptive finite element method involves solving the MMPDE (based upon the numerical solution at the current time step) to obtain a mesh at the next time level and then solving the physical PDE to produce the solution at the new time level. To minimize the overhead involved in moving the mesh and to avoid the possible introduction of singularity into the mesh by the numerical scheme, we use a finite element method with linear basis functions for solving the MMPDE. The use of relatively crude approximations to determine the mesh points is somewhat justified by a theoretical

analysis like that of Babuška and Rheinboldt [1]. Further, the finite element discretization of the MMPDE enables us to use unstructured meshes, which is common in finite element computations. With the mesh points computed from the MMPDE at the new time level, the physical PDE can be discretized by the method of lines (MOL), i.e., by first discretizing the physical PDE in the spatial direction with FEM to obtain an ODE system and then integrating the system to obtain the numerical solution at the new time level.

An outline of the paper is as follows: In Section 2 we present a brief description of the formulation of the MMPDEs. A number of practical issues for the unstructured mesh generation with MMPDEs, e.g., the choice of the computational domain and corresponding computational meshes, are addressed in Section 3. Some numerical results are also presented in this section to demonstrate the ability of the mesh movement strategy. In Section 4 we describe how to discretize a physical PDE by the finite element method on (unstructured) moving meshes. In Section 5 we present several numerical examples to illustrate the performance of the moving mesh technique and the $r$-adaptive finite element method. Finally, Section 6 contains the conclusions.

## 2. FORMULATION OF MOVING MESH PDEs

In this section we briefly describe the moving mesh PDE approach introduced in [17] for structured mesh movement. This MMPDE for a coordinate transformation between the physical and computational domains is based upon the gradient flow equation of a functional which measures the difficulty of spatial approximation of the physical problem.

We begin by describing a general functional form for steady state mesh generation and adaptation. Let $\Omega \subset R^2$ be an open domain where the physical problem is defined, and let $\Omega_c \subset R^2$ be an artificially chosen auxiliary domain which is used to compute adaptive meshes in $\Omega$. Hereafter $\Omega$ and $\Omega_c$ are referred to as the physical and computational domains, respectively, and the corresponding spatial coordinates are denoted by $\boldsymbol{x} = (x, y)$ and $\boldsymbol{\xi} = (\xi, \eta)$. To construct a mesh on $\Omega$, it suffices to define a one-to-one mapping $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi})$ from $\Omega_c$ onto $\Omega$, or equivalently its inverse mapping $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{x})$ from $\Omega$ onto $\Omega_c$, and then to define the mesh on $\Omega$ as the image of the mesh on $\Omega_c$ under the mapping $\boldsymbol{x}(\boldsymbol{\xi})$.

A widely used approach is to define $\boldsymbol{\xi}(\boldsymbol{x})$ as the minimizer of a certain quadratic functional $I[\boldsymbol{\xi}]$ which addresses the desired mesh adaptation properties [4, 26, 28]. Generally, $I[\boldsymbol{\xi}]$ is of the form

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

where $\nabla = (\partial/\partial x, \partial/\partial y)^{\mathrm{T}}$ and $G$ is a $2 \times 2$ symmetric positive definite matrix, usually referred to as the *monitor function*. There are a number of practical considerations when defining $G$, e.g., the mesh should concentrate at certain corners and edges of $\Omega$ or at certain regions, where high resolution of the numerical solution is required, and the mesh can be required to align to some directions. To define $G$ satisfying these requirements, it is convenient to use its eigen-decomposition form

$$G = \lambda_1 \boldsymbol{v}_1 \boldsymbol{v}_1^{\mathrm{T}} + \lambda_2 \boldsymbol{v}_2 \boldsymbol{v}_2^{\mathrm{T}}, \tag{2}$$

where $\boldsymbol{v}_1, \boldsymbol{v}_2$ are normalized eigenvectors and $\lambda_1, \lambda_2$ the corresponding eigenvalues. It is generally difficult to predict the precise mesh behavior from a given monitor function. Our

previous study [6] reveals that if the mesh on $\Omega_c$ is uniformly distributed, then the adaptive mesh generated by minimizing $I[\boldsymbol{\xi}]$ tends to be concentrated in regions where $\lambda_1$ and $\lambda_2$ change significantly. A general guideline is also given in [6] for choosing the monitor function. For instance, a typical choice is

$$\begin{aligned} &\boldsymbol{v}_1 = \nabla u/|\nabla u|, \quad \boldsymbol{v}_2 = \boldsymbol{v}_1^{\perp}, \\ &\lambda_1 = \sqrt{1+|\nabla u|^2} \end{aligned} \tag{3}$$

when it is desired to have the mesh dense in regions where the physical solution $u(\boldsymbol{x})$ changes rapidly. The eigenvalue $\lambda_2$ can generally be chosen as a function of $\lambda_1$. In particular, the choice of $\lambda_2 = \lambda_1$, which leads to $G = \lambda_1 I$, corresponds to Winslow's well-known functional [29]; the choice $\lambda_2 = 1/\lambda_1$, which gives $G = M/\sqrt{\det(M)}$, where $M = I + (\nabla u)(\nabla u)^{\mathrm{T}}$, corresponds to the method based upon harmonic mapping [11], while the case $\lambda_2 = 1$ gives $G = (1 + (\nabla u)(\nabla u)^T)^{1/2}$, which is a generalization of the arc-length monitor function used in one dimension [19]. The effects of these different types of monitor functions are studied in [6].

Given $G$, the mapping $\boldsymbol{\xi}(\boldsymbol{x})$ is determined from the Euler–Lagrange equation

$$\nabla(G^{-1}\nabla\boldsymbol{\xi}) = 0. \tag{4}$$

In actual computation, we solve for $\boldsymbol{x}(\boldsymbol{\xi})$, the inverse mapping of $\boldsymbol{\xi}(\boldsymbol{x})$, because it directly defines the mesh on $\Omega$. Using the coordinate transformation relations

$$\nabla\xi = \frac{1}{J}\begin{pmatrix} y_\eta \\ -x_\eta \end{pmatrix}, \quad \nabla\eta = \frac{1}{J}\begin{pmatrix} -y_\xi \\ x_\xi \end{pmatrix}, \tag{5}$$

where $J = x_\xi y_\eta - y_\xi x_\eta$ is the Jacobian of $\boldsymbol{x}(\boldsymbol{\xi})$, (4) can be expressed as

$$\begin{aligned} &\frac{\partial}{\partial\xi}\left(\frac{\boldsymbol{x}_\eta^{\mathrm{T}} G \boldsymbol{x}_\eta}{Jg}\right) - \frac{\partial}{\partial\eta}\left(\frac{\boldsymbol{x}_\xi^{\mathrm{T}} G \boldsymbol{x}_\eta}{Jg}\right) = 0, \\ &-\frac{\partial}{\partial\xi}\left(\frac{\boldsymbol{x}_\eta^{\mathrm{T}} G \boldsymbol{x}_\xi}{Jg}\right) + \frac{\partial}{\partial\eta}\left(\frac{\boldsymbol{x}_\xi^{\mathrm{T}} G \boldsymbol{x}_\xi}{Jg}\right) = 0, \end{aligned} \tag{6}$$

where $g = \det(G)$.

We consider now mesh movement for time dependent problems. To formulate the mesh equation which moves the mesh smoothly while adapting to the physical solution $u = u(\boldsymbol{x}, t)$ at all time levels, Huang and Russell [17, 18] use the gradient flow equation of functional $I[\boldsymbol{\xi}]$. More specifically, the MMPDE is defined for the mapping $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{x}, t)$ as

$$\frac{\partial\boldsymbol{\xi}}{\partial t} = \frac{1}{\tau\sqrt{g}}\nabla(G^{-1}\nabla\boldsymbol{\xi}), \tag{7}$$

where $\tau > 0$ is a time smoothing parameter. For small $\tau$, the mesh adapts more quickly to the monitor function at each time level, and for large $\tau$ the mesh moves more smoothly with time.

Once again it is more convenient to work with $\boldsymbol{x}(\boldsymbol{\xi}, t)$. Noting that

$$\left.\frac{\partial f}{\partial t}\right|_{\text{fixed } \boldsymbol{x}} = \left.\frac{\partial f}{\partial t}\right|_{\text{fixed } \boldsymbol{\xi}} - \nabla f \cdot \frac{\partial \boldsymbol{x}}{\partial t}$$

for an arbitrary function $f$, we can rewrite (7) as

$$\frac{\partial \boldsymbol{x}}{\partial t} = -\frac{\boldsymbol{x}_\xi}{\tau J\sqrt{g}}\left\{\frac{\partial}{\partial \xi}\left(\frac{\boldsymbol{x}_\eta^{\mathrm{T}} G \boldsymbol{x}_\eta}{Jg}\right) - \frac{\partial}{\partial \eta}\left(\frac{\boldsymbol{x}_\xi^{\mathrm{T}} G \boldsymbol{x}_\eta}{Jg}\right)\right\} - \frac{\boldsymbol{x}_\eta}{\tau J\sqrt{g}}\left\{-\frac{\partial}{\partial \xi}\left(\frac{\boldsymbol{x}_\eta^{\mathrm{T}} G \boldsymbol{x}_\xi}{Jg}\right) + \frac{\partial}{\partial \eta}\left(\frac{\boldsymbol{x}_\xi^{\mathrm{T}} G \boldsymbol{x}_\xi}{Jg}\right)\right\}. \tag{8}$$

This is our basic moving mesh PDE. By solving it numerically, a mesh can be obtained which adapts to the monitor function from one time level to the next. Equation (8) has a more convenient explicit parabolic form,

$$\frac{\partial \boldsymbol{x}}{\partial t} = A\boldsymbol{x}_{\xi\xi} + B\boldsymbol{x}_{\xi\eta} + C\boldsymbol{x}_{\eta\eta} + D\boldsymbol{x}_\xi + E\boldsymbol{x}_\eta, \tag{9}$$

where the coefficients are given as

$$\begin{aligned}
A &= \frac{1}{\tau J^3 g^{3/2}}\left\{-\left(\boldsymbol{x}_\eta^{\mathrm{T}} G \boldsymbol{x}_\eta\right)\boldsymbol{x}_\xi \boldsymbol{x}_\eta^{\mathrm{T}} S + J\boldsymbol{x}_\eta \boldsymbol{x}_\eta^{\mathrm{T}} G + \left(\boldsymbol{x}_\eta^{\mathrm{T}} G \boldsymbol{x}_\xi\right)\boldsymbol{x}_\eta \boldsymbol{x}_\eta^{\mathrm{T}} S\right\},\\
B &= \frac{1}{\tau J^3 g^{3/2}}\left\{\left(\boldsymbol{x}_\eta^{\mathrm{T}} G \boldsymbol{x}_\eta\right)\boldsymbol{x}_\xi \boldsymbol{x}_\xi^{\mathrm{T}} S - J\boldsymbol{x}_\xi \boldsymbol{x}_\eta^{\mathrm{T}} G + \left(\boldsymbol{x}_\xi^{\mathrm{T}} G \boldsymbol{x}_\eta\right)\boldsymbol{x}_\xi \boldsymbol{x}_\eta^{\mathrm{T}} S\right\}\\
&\quad + \frac{1}{\tau J^3 g^{3/2}}\left\{-\left(\boldsymbol{x}_\xi^{\mathrm{T}} G \boldsymbol{x}_\xi\right)\boldsymbol{x}_\eta \boldsymbol{x}_\eta^{\mathrm{T}} S - J\boldsymbol{x}_\eta \boldsymbol{x}_\xi^{\mathrm{T}} G - \left(\boldsymbol{x}_\eta^{\mathrm{T}} G \boldsymbol{x}_\xi\right)\boldsymbol{x}_\eta \boldsymbol{x}_\xi^{\mathrm{T}} S\right\},\\
C &= \frac{1}{\tau J^3 g^{3/2}}\left\{\left(\boldsymbol{x}_\xi^{\mathrm{T}} G \boldsymbol{x}_\xi\right)\boldsymbol{x}_\eta \boldsymbol{x}_\xi^{\mathrm{T}} S + J\boldsymbol{x}_\xi \boldsymbol{x}_\xi^{\mathrm{T}} G - \left(\boldsymbol{x}_\xi^{\mathrm{T}} G \boldsymbol{x}_\eta\right)\boldsymbol{x}_\xi \boldsymbol{x}_\xi^{\mathrm{T}} S\right\},\\
D &= \frac{1}{\tau J^2 g^{1/2}}\left\{-\boldsymbol{x}_\eta^{\mathrm{T}} \frac{\partial}{\partial \xi}\left(\frac{G}{g}\right)\boldsymbol{x}_\eta + \boldsymbol{x}_\xi^{\mathrm{T}} \frac{\partial}{\partial \eta}\left(\frac{G}{g}\right)\boldsymbol{x}_\eta\right\},\\
E &= \frac{1}{\tau J^2 g^{1/2}}\left\{\boldsymbol{x}_\eta^{\mathrm{T}} \frac{\partial}{\partial \xi}\left(\frac{G}{g}\right)\boldsymbol{x}_\xi - \boldsymbol{x}_\xi^{\mathrm{T}} \frac{\partial}{\partial \eta}\left(\frac{G}{g}\right)\boldsymbol{x}_\xi\right\},
\end{aligned} \tag{10}$$

and $S = \left[\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right]$.

To completely specify the coordinate transformation, the MMPDE must be supplemented with suitable boundary conditions. This is trivial in the case where the grid points are held fixed on the boundary. To let the grids move on the boundary, a natural choice is to impose homogeneous Neumann boundary conditions; however, our numerical experience has shown that this choice is not very robust and often produces a nonsmooth grid. We instead take Dirichlet boundary conditions determined from the solution of a one-dimensional MMPDE. More precisely, given a boundary segment $\Gamma$ of $\partial\Omega$, let $\Gamma_c$ be the corresponding boundary segment of $\partial\Omega_c$. Denoting by $s$ the arc-length from a point on $\Gamma$ to one of its end points and by $\zeta$ the arc-length from a point on $\Gamma_c$ to one of its end points, we can identify $\Gamma$ with $I = (0, \ell)$ and $\Gamma_c$ with $I_c = (0, \ell_c)$. Then the mesh on $\Gamma$ is determined by the boundary definition and the solution $s(\zeta, t)$ of the one-dimensional MMPDE,

$$\begin{aligned}
&\frac{\partial s}{\partial t} + \frac{1}{\tau_b}\frac{\partial}{\partial \zeta}\left(\frac{1}{M(s,t)(\partial s/\partial \zeta)}\right) = 0, \quad \zeta \in (0, \ell_c),\\
&s(0) = 0, \quad s(\ell_c) = \ell,
\end{aligned}$$

where $\tau_b$ is a parameter determining the strength of the temporal smoothing for the moving grids on boundaries. In general it may take the same value as the $\tau$ in (8). $M$ is a one-dimensional arc-length monitor function chosen as the projection of the two-dimensional monitor function $G$ along the boundary; i.e., if $\mathbf{s}$ is the unit tangent vector along the boundary then $M(\zeta, t) = \mathbf{s}^{\mathrm{T}} G \mathbf{s}$. This is similar to MMPDE5 in [19].

## 3. IMPLEMENTATION OF UNSTRUCTURED MESH MOVEMENT

We have seen in the previous section that the MMPDE is formulated in terms of the coordinate transformation $\boldsymbol{x}(\boldsymbol{\xi})$. As is the case for other coordinate transformation approaches for mesh generation and adaptation, the definition of a computational domain $\Omega_c$ and a corresponding computational mesh $\Omega_c^h$ must be done initially. However, there is an essential difference in how this is done in the cases of structured and unstructured grids.

For the case of structured grids, it is common practice to choose $\Omega_c$ as a simple geometry, typically a square, and to choose $\Omega_c^h$ to be a uniform mesh on $\Omega_c$. The adaptive mesh on the physical domain $\Omega$ is then determined by solving the mesh equation (6) or (9). Due to the simple structure of $\Omega_c$ and $\Omega_c^h$, this approach is too restrictive when the shape of $\Omega$ is relatively complicated. This limitation can sometimes be dealt with by a multiblock approach, where $\Omega$ is broken up into a number of simply shaped subregions, and $\Omega_c$ is mapped onto each of these subregions individually, e.g., as in [26].

In contrast, for the unstructured grid case, it is typical to generate first an initial mesh $\bar{\Omega}^h(0)$ over the physical domain $\Omega$, by one of various mesh generators such as a Delaunay triangulation in the finite element context. (The shape of $\Omega$ itself poses little difficulty in generating $\bar{\Omega}^h(0)$ compatible with the domain geometry. Indeed, one of the major reasons for the wide-spread use of FEM is their ability to deal with complicated solution domains in such a way.) The computational domain $\Omega_c$ and computational mesh $\Omega_c^h$ should then be defined and computed in accordance with $\Omega$ and $\bar{\Omega}^h(0)$.

The choice of $\Omega_c$ can be quite arbitrary. The basic guideline is to choose $\Omega_c$ such that (6) defines a one-to-one mapping between $\Omega$ and $\Omega_c$. Unfortunately, for a general monitor function $G$, there are no obvious conditions on $\Omega_c$ which guarantee that a unique solution to (6) exists and defines a one-to-one mapping between $\Omega$ and $\Omega_c$. An exceptional case is for a harmonic mapping. Then if $\Omega_c$ is convex and the mapping from $\partial\Omega$ to $\partial\Omega_c$ is smooth, it follows from the theory of harmonic mappings that the solution to (6) is unique and defines a one-to-one mapping between $\Omega$ and $\Omega_c$ [11, 16]. Lack of convexity of $\Omega_c$ may result in a degenerate mapping and mesh crossing. Thus, we recommend that $\Omega_c$ be taken as a convex domain. If $\Omega$ is convex itself, one may simply take $\Omega_c = \Omega$.

Once $\Omega_c$ is selected, one way to define an $\Omega_c^h$ to have the same mesh topology as $\Omega^h(0)$ is to first specify a correspondence between the boundaries $\partial\Omega$ and $\partial\Omega_c$ by a mapping $\phi(\boldsymbol{x})$ and then let $\Omega_c^h$ be the image of $\bar{\Omega}^h(0)$ under the mapping $\bar{\boldsymbol{\xi}}(\boldsymbol{x})$ satisfying

$$\begin{aligned} &\nabla^2 \bar{\boldsymbol{\xi}} = 0, && \text{in } \Omega, \\ &\bar{\boldsymbol{\xi}}(\boldsymbol{x}) = \phi(\boldsymbol{x}), && \text{on } \partial\Omega. \end{aligned} \tag{11}$$

The above Laplace equations can be solved numerically by, e.g., the finite element method. Once $\Omega_c^h$ is obtained, the initial adaptive mesh $\Omega^h(0)$ can be computed as the numerical solution of (6) on $\Omega_c^h$.

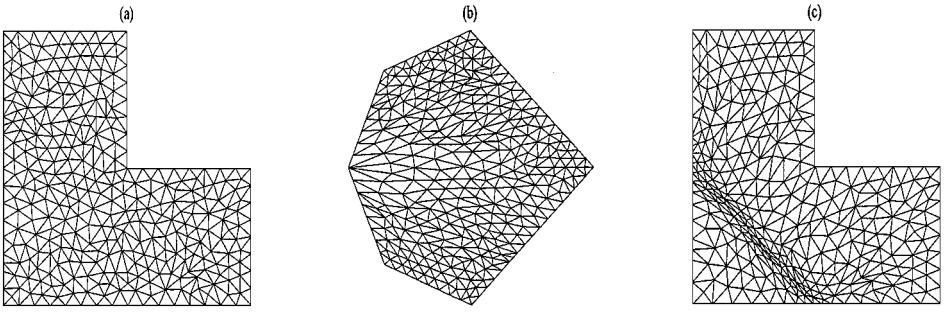


**FIG. 1.** Meshes are plotted for Example 3.1: (a) initial mesh $\bar{\Omega}^h(0)$ on physical domain; (b) computational mesh $\Omega_c^h$ obtained by solving (11); (c) initial adaptive mesh $\Omega^h(0)$ obtained by solving (6).

EXAMPLE 3.1. The above mesh generation procedure is applied on the L-shape domain $(0,1)\times(0,\frac{1}{2})\cup(0,\frac{1}{2})\times(\frac{1}{2},1)$. The initial mesh $\bar{\Omega}^h(0)$ is as shown in Fig. 1a. $\Omega_c$ is chosen to be a hexagon and the boundary correspondence is defined using piecewise linear functions. The obtained computational mesh $\Omega_c^h$ is shown in Fig. 1b. Note that the mesh topologies in $\Omega$ and $\Omega_c$ are identical. Finally, we solve (6) with the artificially chosen monitor function $G=(1+10\,\mathrm{sech}(50(x+y-\frac{1}{2}))I$ to get an initial adaptive mesh $\Omega^h(0)$ as shown in Fig. 1c.

Numerical solution of the MMPDE involves discretization in both the spatial and temporal directions. In the spatial direction, we use a standard linear finite element method; viz., we seek the approximate solution in the space of continuous piecewise linear polynomials $S^h(\Omega_c)$. For discretization in the temporal direction, we use the backward Euler formula at the time levels $0=t_0<t_1<\cdots<t_n<\cdots$. To avoid solving the nonlinear algebraic equations, the MMPDE (9) is linearized with the coefficients $A$, $B$, $C$, $D$, and $E$ calculated at time $t_n$. The full discretization of the MMPDE is then given as

Find $\boldsymbol{x}(t_{n+1})\in S^h(\Omega_c)$ such that

$$(\boldsymbol{x}_t(t_n),\boldsymbol{v})+\left(\boldsymbol{x}_\xi(t_{n+1}),\left(A(t_n)^{\mathrm{T}}\boldsymbol{v}\right)_\xi\right)+\left(\boldsymbol{x}_\eta(t_{n+1}),\left(C(t_n)^{\mathrm{T}}\boldsymbol{v}\right)_\eta\right)+\frac{1}{2}\Big[\left(\boldsymbol{x}_\xi(t_{n+1}),\left(B(t_n)^{\mathrm{T}}\boldsymbol{v}\right)_\eta\right)$$
$$+\left(\boldsymbol{x}_\eta(t_{n+1}),\left(B(t_n)^{\mathrm{T}}\boldsymbol{v}\right)_\xi\right)\Big]-\left(D(t_n)\boldsymbol{x}_\xi(t_{n+1})+E(t_n)\boldsymbol{x}_\eta(t_{n+1}),\boldsymbol{v}\right)=0\quad\forall\boldsymbol{v}\in S_0^h(\Omega_c),\tag{12}$$

where $(\cdot,\cdot)$ stands for the inner product in $L^2(\Omega_c)$, $S_0^h(\Omega_c)$ is the subspace of $S^h(\Omega_c)$ consisting of functions which vanish on $\partial\Omega_c$,

$$\boldsymbol{x}_t(t_n)=\frac{\boldsymbol{x}(t_{n+1})-\boldsymbol{x}(t_n)}{t_{n+1}-t_n},$$

and $A(t_n)$, $B(t_n)$, $C(t_n)$, $D(t_n)$, and $E(t_n)$ are calculated using $\boldsymbol{x}(t_n)$. The resulting linear system for $\boldsymbol{x}(t_{n+1})$ is solved using the iterative method BiCGStab2 [14, 27] with $\boldsymbol{x}(t_n)$ as the initial approximation. For all numerical results which follow, the iteration is continued until the mean square root of the residual is less than $10^{-8}$.

Unlike in the structured grid case, both the initial mesh $\bar{\Omega}^h(0)$ and the computational mesh $\Omega_c^h$ are, in general, nonuniform. The effects of this initial nonuniformity on subsequent meshes generated by (6) or (9) are very complicated, and it is not our intention to study them here. However, it is interesting to analyze the one-dimensional case, for which the

effects of nonuniformity can be seen more clearly, partly because the boundary effect does not complicate matters.

For this, suppose that we are given the physical and computational domains $\Omega = \Omega_c = [0, 1]$, a monitor function $M$, and an initial nonuniform mesh $\bar{\Omega}^h(0)$. Formally speaking, this mesh can be regarded as the image of a uniform computational mesh under a mapping $\bar{x} = \bar{x}(\xi)$. Following the strategy discussed above, a nonuniform computational mesh $\Omega_c^h$ (its coordinate denoted by $\bar{\xi}$) is defined by solving a one-dimensional version of (11), viz.,

$$\frac{d^2\bar{\xi}}{d\bar{x}^2} = 0, \quad \bar{\xi}(0) = 0,\ \bar{\xi}(1) = 1. \tag{13}$$

(Since $\Omega = \Omega_c$, the solution of (13) is the identity map $\bar{\xi} = \bar{x}$.) The initial adaptive mesh $\Omega^h(0)$ is determined by the equidistribution principle (i.e., the one-dimensional version of (6) [19]),

$$\frac{d}{d\xi}\left(\frac{1}{M(dx/d\bar{\xi})}\right) = 0, \quad \bar{\xi}(0) = 0,\ \bar{\xi}(1) = 1, \tag{14}$$

or equivalently,

$$M\frac{dx}{d\xi} = C \tag{15}$$

for suitable constant $C$ [19]. Since $\bar{x} = \bar{\xi}$, (15) becomes

$$M\frac{dx}{d\bar{x}} = C \tag{16}$$

or

$$\frac{M}{(d\bar{x}/d\xi)}\frac{dx}{d\xi} = C. \tag{17}$$

We conclude that the initial adaptive mesh can be interpreted as arising from using a uniform computational grid and equidistributing the modified monitor function $\tilde{M} = M/(d\bar{x}/d\xi)$. From the form of $\tilde{M}$, we see that there is the propensity for the initial adaptive mesh to be concentrated in the same regions as the initial mesh $\bar{\Omega}^h(0)$.

To illustrate the effects of the choices of the computational domain and the initial mesh, we present in the following some numerical examples for which the adaptation functions are given.

EXAMPLE 3.2. In this example, the L-shape domain from Example 3.1 is used again to study the effects that different choices of the computational domain $\Omega_c$ have on the initial adaptive mesh. $\bar{\Omega}^h(0)$ is taken to be a uniform triangulation, as shown in Fig. 2a. We choose the different computational domains as the convex polygons having three, four, five, and six sides and the unit circle. The mesh for each $\Omega_c$ obtained by solving (11) is plotted in Figs. 2b1–b5. Note that the mapping between $\Omega$ and $\Omega_c$ is not regular at certain corner points, and in cases (b1), (b2), and (b5) the meshes in the computational domain are highly irregular (the smallest angles in some elements are almost zero). Nevertheless, we experience no difficulty in generating the adaptive meshes for a monitor function of Winslow's type and adaptation function $u(\boldsymbol{x}) = 1/(1 + \exp(100(x + y - 1/2)))$; moreover, the initial adaptive

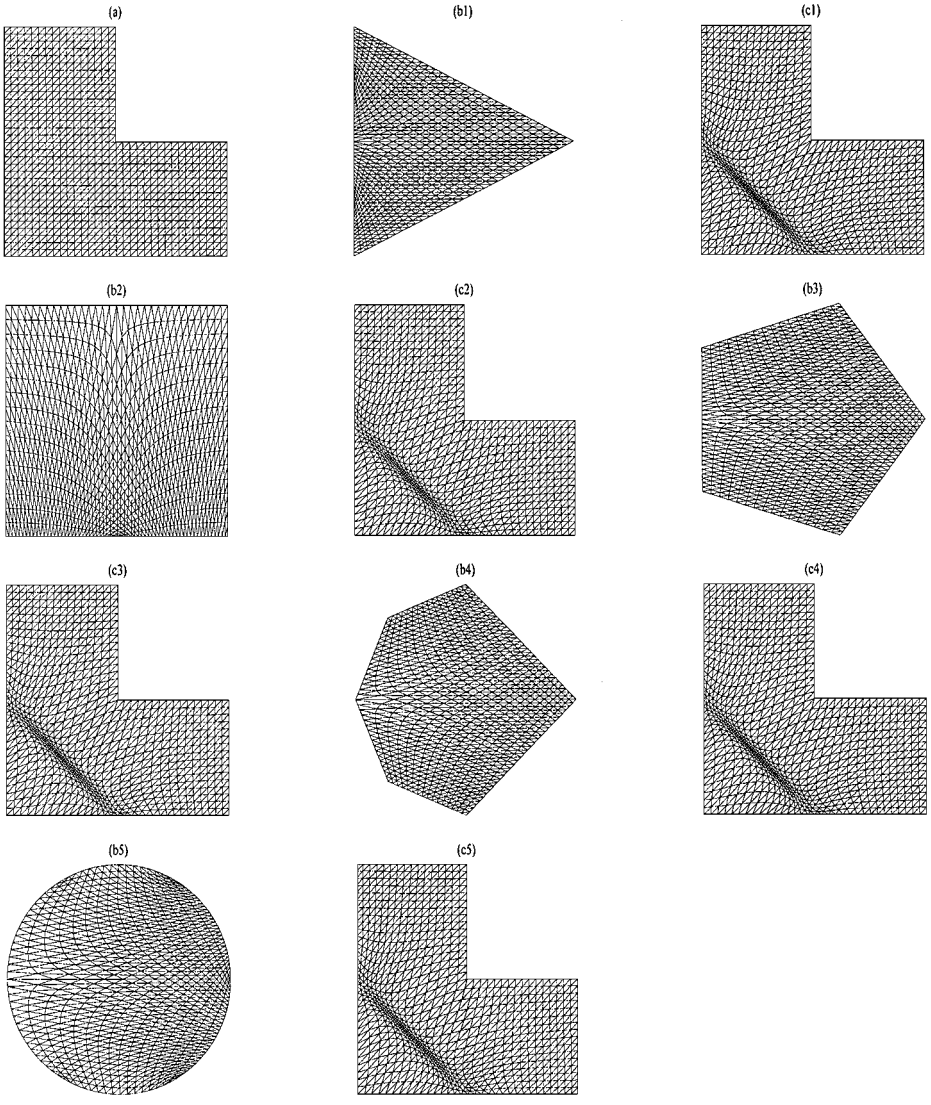


**FIG. 2.** Meshes are shown for Example 3.2: (a) the original mesh; (b) computational domains and their meshes; (c) adaptive meshes.

meshes $\Omega^h(0)$ are almost the same for these very different choices of $\Omega_c$ (see Figs. 2c1–c5). This example suggests that the adaptive mesh is relatively insensitive to the choice of $\Omega_c$. However, the adaptive mesh generation and movement techniques are clearly not applicable if the mapping between $\Omega$ and $\Omega_c$ actually results in degenerate elements in $\Omega_c$.

EXAMPLE 3.3. 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 $u(\boldsymbol{x}, t) = \tanh(50(x - t))$. An unstructured grid as shown in Fig. 3a is initially generated with the Delaunay mesh triangulator *Triangle* developed in [25]. We choose the computational domain as a convex polygon having the same number of boundary segments as $\Omega$ and use a monitor function of Winslow's type in (9). The resulting moving mesh is shown in Fig. 3 for various times. One can see that the generated mesh is satisfactory in the sense

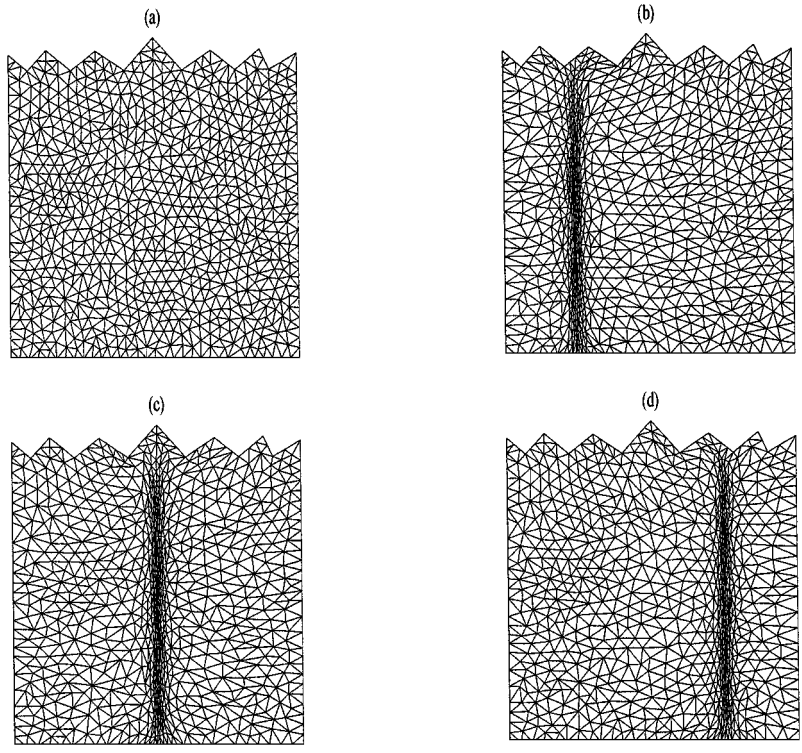


**FIG. 3.** An adaptive mesh is plotted at $t = 0, 0.25, 0.5, 0.75$ for Example 3.3.

that it conforms very well to the adaptation function. However, one can also see from the picture that elements near the rough boundary can sometimes become very skew. Although a straightforward application of the moving mesh technique does work here, for general nonconvex domains there is no theoretical guarantee for the existence and invertibility of the continuous and/or discrete mapping between $\Omega_c$ and $\Omega$ generated with the mesh movement strategy. When $\Omega$ is a nonconvex region with corners, it can be important to use smaller time steps when the wave front reaches corners, to use finer meshes around corners to reduce the numerical errors, or to use local refinement to change the topology of the mesh to eliminate the singular elements. Application of the mesh movement strategy for general nonconvex domains is certainly an area demanding further investigation.

EXAMPLE 3.4. In this example, we test the moving mesh method for the case where $\Omega$ is multiconnected. The solution domain, shown in Fig. 4a, is a typical one for an airfoil analysis problem. The initial mesh $\overline{\Omega}^h(0)$ is again obtained by the Delaunay triangulator with weighted area constraints. We choose $\Omega = \Omega_c$ and $\overline{\Omega}^h(0) = \Omega_c^h$ for this example. The adaptation function $u(\boldsymbol{x}, t) = \tanh(50(3x - |y| - t))$ is used to simulate a moving shock wave. The moving meshes at three different times are plotted in Figs. 4b, c, and d. Note that the moving mesh method produces satisfactory adaptive meshes even though for this case the physical and computational domains are multiconnected. Also, observe that the initial mesh concentration around the solid surface of the airfoil is maintained by the adaptive meshes, a feature anticipated in our analysis.

In summary, the unstructured mesh movement strategy developed in this section has been seen to work satisfactorily in a variety of circumstances. The numerical results indicate that

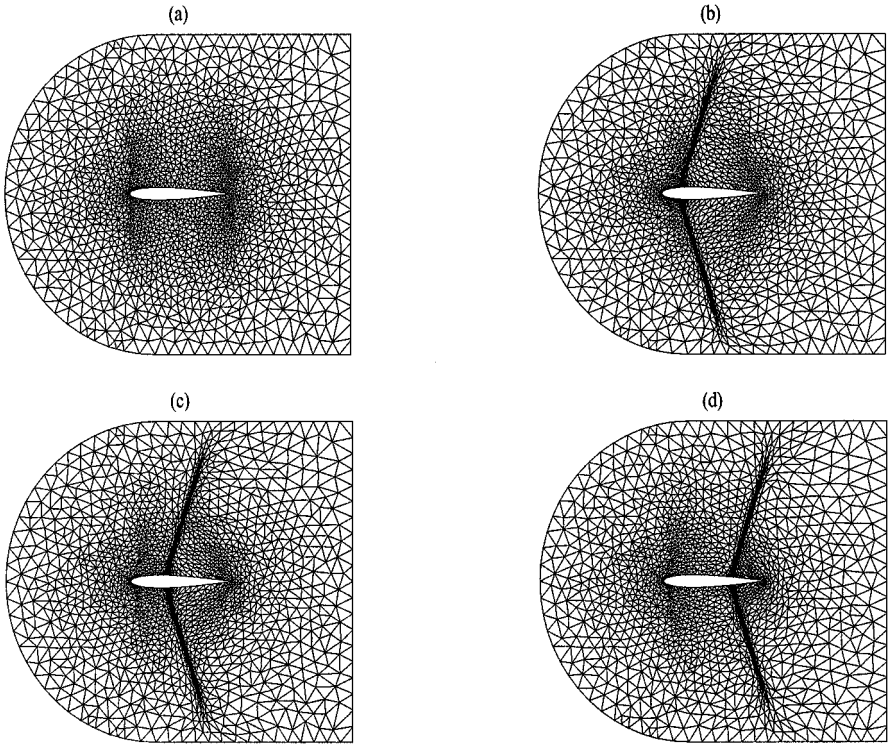


**FIG. 4.** An adaptive mesh is depicted at $t = 0, 0.5, 1.0, 2.0$ for Example 3.4.

the adaptive mesh $\Omega^h(0)$ is relatively insensitive to the choice of the computational domain. Nevertheless, we recommend that $\Omega_c$ should generally be chosen to be convex and to have the same number of boundary segments as $\Omega$ in order to avoid introducing degenerate elements in the computational mesh.

## 4. $r$-ADAPTIVE FEM FOR PHYSICAL PDEs

We consider now the finite element method for numerically solving time dependent PDEs on moving meshes. Specifically, consider the time dependent PDE

$$\frac{\partial U}{\partial t}(t) + \mathcal{L}U(t) = 0 \quad \text{in } \Omega \times (0, T], \tag{18}$$

where $\Omega$ is an open bounded domain in $R^2$, the solution $U(t)$ lies in a function space $H(\Omega)$, and $\mathcal{L}$ is a spatial differentiation operator. The system, supplemented with suitable initial and boundary conditions, is assumed to be well-posed.

Assume that $[0, T]$ is partitioned into time levels $0 = t_0 < t_1 < \cdots < t_N = T$ and that at time level $t_n$ a mesh $\Omega^h(t_n)$ on $\Omega$ and the numerical solution $u(t_n)$ (approximating $U(t_n)$) are given. In order to compute the numerical solution $u(t_{n+1})$ at the next time level, we first use the moving mesh method described in the previous sections to determine a mesh $\Omega^h(t_{n+1})$ on $\Omega$. Recall that for the moving mesh method, the connectivity of $\Omega^h(t_{n+1})$ is the same as that of $\Omega^h(t_n)$. Thus, each element $K(t_{n+1})$ of $\Omega^h(t_{n+1})$ corresponds uniquely to an element $K(t_n)$ of $\Omega(t_n)$. Let $\mathcal{F}_K(t_n)$ be the affine map from $K(t_n)$ to $K(t_{n+1})$. For any

point $\tilde{\boldsymbol{x}} \in K(t_n)$, define

$$\boldsymbol{x}(\tilde{\boldsymbol{x}}, t) = \frac{t - t_n}{t_{n+1} - t_n} \mathcal{F}_K(t_n)(\tilde{\boldsymbol{x}}) + \frac{t_{n+1} - t}{t_{n+1} - t_n} \tilde{\boldsymbol{x}}$$

and for $t \in (t_n, t_{n+1})$,

$$K(t) = \{\boldsymbol{x}(\tilde{\boldsymbol{x}}, t) \mid \forall \tilde{\boldsymbol{x}} \in K(t_n)\}.$$

Clearly, $\{K(t)\}$ defines a mesh $\Omega^h(t)$ on $\Omega$. For each element $K(t)$, let $\hat{K}$ be the standard element, i.e., $\hat{K}$ is the unit square if $K(t)$ is quadrilateral, or the unit right triangle if $K(t)$ is triangular. Denote the affine map from $\hat{K}$ to $K(t)$ by $F_{K(t)}$ and the corresponding approximation space at time $t$ as

$$\mathcal{S}^h(t) = \left\{ v \in H(\Omega) \mid v|_{K(t)} \circ F_{K(t)} \in P(\hat{K}) \quad \forall K(t) \in \Omega^h(t) \right\},$$

where $P(\hat{K})$ is a given set of polynomials on $\hat{K}$—in our case the set of linear functions. If $\{\hat{\phi}_i(\hat{\boldsymbol{x}})\}$ is a standard basis for $P(\hat{K})$, then $\{\phi_j(\boldsymbol{x}, t)\}$, where for each $j$, $\phi_j(\boldsymbol{x}, t)|_{K(t)} = \hat{\phi}_i(F^{-1}_{K(t)}(\boldsymbol{x}))$ for some $i$, is a basis of the space $\mathcal{S}^h(t)$ on $K(t)$. We have

$$\frac{\partial \phi_j}{\partial t} = \nabla_{\hat{\boldsymbol{x}}} \hat{\phi}_i \cdot \frac{\partial \hat{\boldsymbol{x}}}{\partial t} = \left( \left[ \frac{\partial \boldsymbol{x}}{\partial \hat{\boldsymbol{x}}} \right]^{\mathrm{T}} \nabla_{\boldsymbol{x}} \phi_j \right) \cdot \frac{\partial \hat{\boldsymbol{x}}}{\partial t} = \nabla_{\boldsymbol{x}} \phi_j \cdot \left( \frac{\partial \boldsymbol{x}}{\partial \hat{\boldsymbol{x}}} \frac{\partial \hat{\boldsymbol{x}}}{\partial t} \right)$$

and

$$\mathbf{0} = \frac{\partial \boldsymbol{x}}{\partial t} + \frac{\partial \boldsymbol{x}}{\partial \hat{\boldsymbol{x}}} \frac{\partial \hat{\boldsymbol{x}}}{\partial t},$$

where $\nabla_{\hat{\boldsymbol{x}}}$ is the gradient operator with respect to the coordinate $\hat{\boldsymbol{x}}$ in the standard element and $\partial \hat{\boldsymbol{x}} / \partial \boldsymbol{x}$ is the inverse of the Jacobian of the affine mapping $F_{K(t)}$. Hence,

$$\frac{\partial \phi_j}{\partial t} = -\frac{\partial \boldsymbol{x}}{\partial t} \cdot \nabla_{\boldsymbol{x}} \phi_j. \tag{19}$$

For any $v(t) \in \mathcal{S}^h(t)$ having the representation

$$v(\boldsymbol{x}, t) = \sum_j v_j(t) \phi_j(\boldsymbol{x}, t),$$

it follows from (19) that

$$\frac{\partial v(\boldsymbol{x}, t)}{\partial t} = \sum_j \frac{dv_j}{dt}(t) \phi_j(\boldsymbol{x}, t) - \left[ \frac{\partial \boldsymbol{x}}{\partial t} \cdot \nabla_{\boldsymbol{x}} \right] v(\boldsymbol{x}, t).$$

The semi-discrete finite element approximation for (18) thus involves finding $u(\boldsymbol{x}, t) \in \mathcal{S}^h(t)$ for $t \in (t_n, t_{n+1}]$ such that

$$\int_\Omega \left\{ \sum_i \frac{du_j}{dt}(t) \phi_j(\boldsymbol{x}, t) - \left[ \frac{\partial \boldsymbol{x}}{\partial t} \cdot \nabla_{\boldsymbol{x}} \right] u(\boldsymbol{x}, t) + \mathcal{L} u(\boldsymbol{x}, t) \right\} v \, d\boldsymbol{x} = 0 \quad \forall v \in \mathcal{S}_0^h(t), \tag{20}$$

where $\mathcal{S}_0^h(t)$ is the subspace of $\mathcal{S}^h(t)$ consisting of functions vanishing on the part of $\partial\Omega$ where essential boundary conditions are imposed. If $\mathcal{J}$ is the set of indices corresponding to the mesh points and $\mathcal{J}_0$ is the subset of $\mathcal{J}$ excluding those corresponding to Dirichlet boundary conditions, then $\mathcal{S}_0^h(t)$ can be expressed as $\mathcal{S}_0^h(t) = \operatorname{span}\{\phi_j(\boldsymbol{x}, t) \mid j \in \mathcal{J}_0\}$, and (20) can be written as a system of ODEs

$$M(t)\frac{d\mathbf{u}}{dt} = \mathbf{F}(t, \mathbf{u}), \tag{21}$$

where $\mathbf{u} = (u_i)_{i\in\mathcal{J}_0}$ is the unknown vector, $M(t) = (m_{ij}(t))_{i,j\in\mathcal{J}_0}$ is the mass matrix, $\mathbf{F} = (F_i)_{i\in\mathcal{J}_0}$ is the load vector, and

$$m_{ij}(t) = \int_\Omega \phi_i(\boldsymbol{x}, t)\phi_j(\boldsymbol{x}, t)\, d\boldsymbol{x},$$

$$F_i = F_i(t, \mathbf{u}) = \int_\Omega \left( -\mathcal{L}u(\boldsymbol{x}, t) + \left[\frac{\partial \boldsymbol{x}}{\partial t} \cdot \nabla_{\boldsymbol{x}}\right] u(\boldsymbol{x}, t) - \sum_{j\in\mathcal{J}\setminus\mathcal{J}_0} \frac{du_j}{dt}(t)\phi_j(\boldsymbol{x}, t) \right) \phi_i\, d\boldsymbol{x}.$$

Except for the mesh movement portion, this procedure is simply the method of lines approach on moving ("quasi-Lagrangian") coordinates. With a suitably chosen time integrator, the approximation to the solution at time $t_{n+1}$ can be obtained straightforwardly by integrating (21). While in principle any type of time integrator can be used to solve (21), for our numerical examples in the next section we choose a two-stage (second-order) singly diagonal implicit Runge–Kutta method (SDIRK) for its relative efficiency and favourable stability properties. The time step size $\delta t$ is either fixed or adaptively selected with the help of an embedding scheme (see [15] for details). The resulting nonlinear algebraic system is iteratively solved using BiCGStab2 [14, 27] until the mean square root of the residual is less than $10^{-8}$.

## 5. NUMERICAL RESULTS

In this section we present some numerical examples to demonstrate the performance of the $r$-adaptive FEM for solving time-dependent PDEs.

EXAMPLE 5.1. 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

$$U(\boldsymbol{x}, 0) = \begin{cases} 1 - 16\left(\left(x - \frac{1}{2}\right)^2 + \frac{3}{2}y^2\right), & \text{if } \left(x - \frac{1}{2}\right)^2 + \frac{3}{2}y^2 < \frac{1}{16}, \\ 1 - 16\left(\left(x + \frac{1}{2}\right)^2 + \frac{3}{2}y^2\right), & \text{if } \left(x + \frac{1}{2}\right)^2 + \frac{3}{2}y^2 < \frac{1}{16}, \\ 0, & \text{elsewhere.} \end{cases}$$

Note that there are no boundary conditions needed for this problem since the boundary is in a characteristic direction of the PDE. The solution possesses a twin peak (of fixed shape) rotating counterclockwise around the origin. A linear finite element discretization

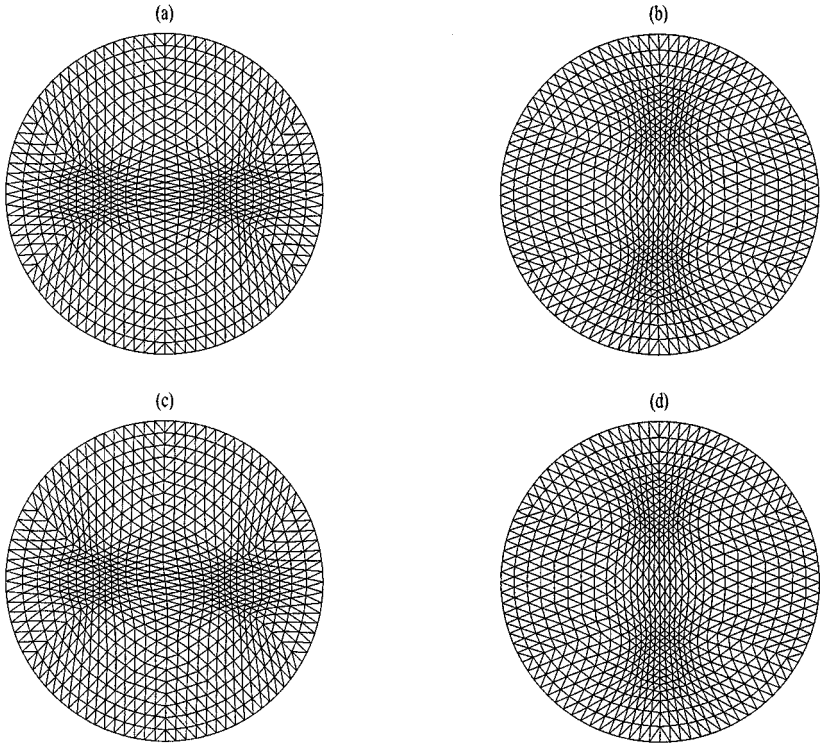


**FIG. 5.** A moving mesh is shown at $t = 0, \pi/2, \pi, 3\pi/2$ for the wave equation.

based upon moving meshes as described in Section 4 is applied. The initial mesh is obtained from a quasi-uniform triangulation with 1536 elements. A fixed time step size $\delta t = 0.005$ is used for the integration of the physical PDE. For mesh movement, the monitor function is calculated from (3) with $\lambda_2 = \lambda_1$ (Winslow's type) using the numerical solution, and the time smoothing parameter $\tau$ is taken to be $10^{-1}$. The grid points on the boundary are kept fixed. The resulting mesh is showed in Fig. 5 for various times. For comparison purposes, we solve this problem using both fixed and moving meshes. The error with the moving mesh is found to be about 2/3 of that with the fixed mesh, so the improvement using moving meshes is not particularly significant for this example. This is because the solution is relatively smooth and high accuracy can be obtained on a relatively coarse uniform mesh. Nevertheless, the problem is a serious test for mesh movement strategies. Indeed, many 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 [2, 9, 30]. From Fig. 5, it is clear that our moving mesh method has no such difficulty, and the mesh adapts extremely well to the solution without producing skew elements.

EXAMPLE 5.2. Our second example is Burgers' equation

$$\frac{\partial U}{\partial t} = a\nabla^2 U + UU_x + UU_y \tag{22}$$

defined on the domain $\Omega = (0, 1) \times (0, 1)$. The Dirichlet boundary and initial conditions

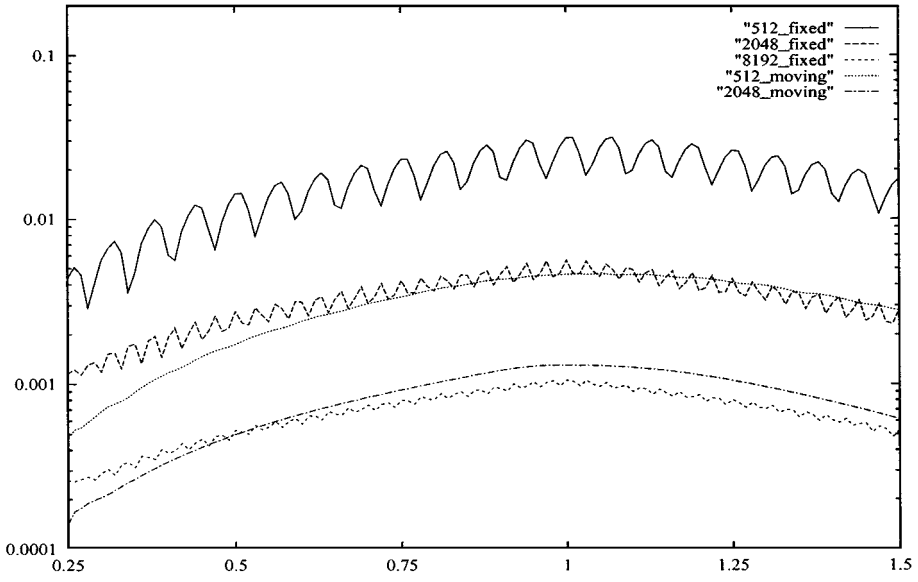


**FIG. 6.** The $L^1$ errors in time of the linear finite element solutions for Example 5.2 are obtained with fixed and moving meshes of various numbers of elements.

are chosen such that the exact solution to the underlying problem is $U(x, y, t) = 1/(1 + \exp((x + y - t)/(2a))$. We test our moving finite element method for the case $a = 0.005$. The smaller $a$ is, the more convection dominates and the higher the concentration of mesh points required around the wave front.

The monitor function and the time smoothing parameter are defined in the same way as in Example 5.1. A fixed time step size 0.001 is used for the time integration of the ODE system (21). To examine the accuracy of the finite element solver, we first solve (22) using fixed, uniform meshes with 512, 2048, and 8192 triangular elements (i.e., we successively reduce the element diameter by a factor of $\frac{1}{2}$). The $L^1$ errors are plotted in Fig. 6. As expected for the linear FEM, the error drops quadratically as the element diameter decreases. We then use the finite element solver (for the same parameter settings) on moving meshes of 512 and 2048 triangular elements. For the 2048 element case, the obtained moving mesh is plotted in Fig. 7 for four time levels. For the same number of elements, the $L^1$-error of the finite element solution with a moving mesh is about $1/5$ of that with a fixed mesh (see Fig. 6). Furthermore, without resorting to any upwinding treatment, the oscillations in front of and behind the steep solution front, which are typical for the finite element discretization of convection dominated problems, are essentially eliminated through the use of the moving meshes.

EXAMPLE 5.3. The moving finite element method is now applied to a few more practical problems, for which the analytical solutions are not available. The first of these problems describes the buoyancy-driven horizontal spreading of heat and chemical species through a fluid-saturated porous medium. Consider the two-dimensional porous medium region sketched in Fig. 8. The boundary of the region is assumed to be adiabatic and impermeable. The fluid which saturates the porous medium 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

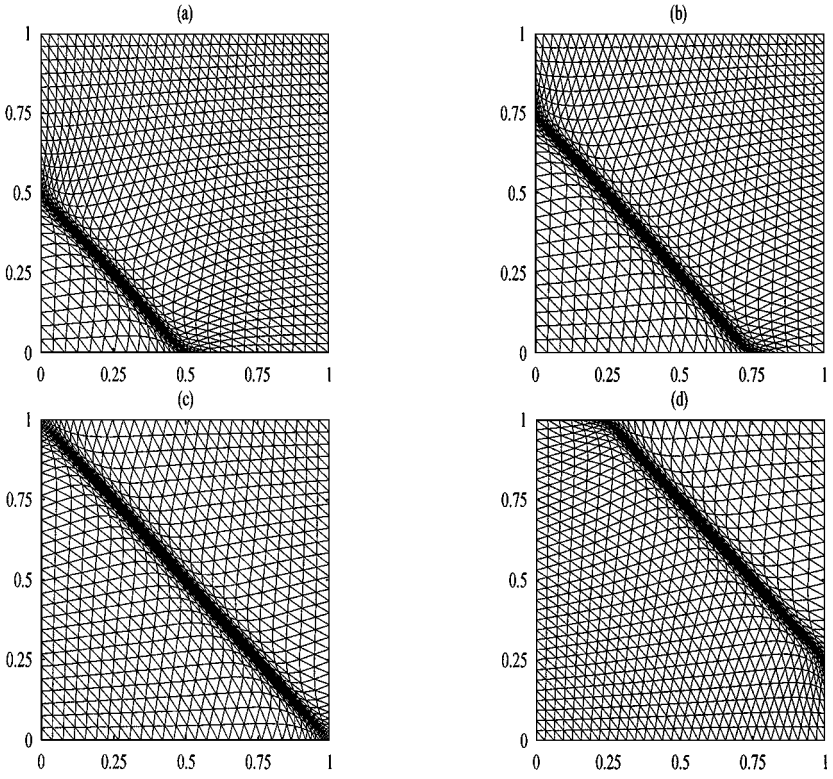


**FIG. 7.** A typical moving mesh of 2048 triangular elements for Example 5.2 is shown at $t = 0.5, 0.75, 1.0, 1.25$.

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 the 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. 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.

Using Darcy's law and the homogeneous porous medium model, the conservations for mass, momentum, energy, and the constituent give rise to the system [24]:

$$\begin{aligned} -\nabla^2\psi &= \mathrm{Ra}\left(\frac{\partial T}{\partial x} + N\frac{\partial C}{\partial x}\right), \\ \frac{\partial T}{\partial t} + \frac{\partial(T,\psi)}{\partial(x,y)} &= \nabla^2 T, \\ \frac{\phi}{\sigma}\frac{\partial C}{\partial t} + \frac{\partial(C,\psi)}{\partial(x,y)} &= \frac{1}{\mathrm{Le}}\nabla^2 C, \end{aligned} \tag{23}$$

where $\psi$ is the stream function of the flow, $T$ is the temperature, $C$ is the concentration of the constituent, Ra is the Darcy-modified Rayleigh number, $N$ is the buoyancy ratio, Le is

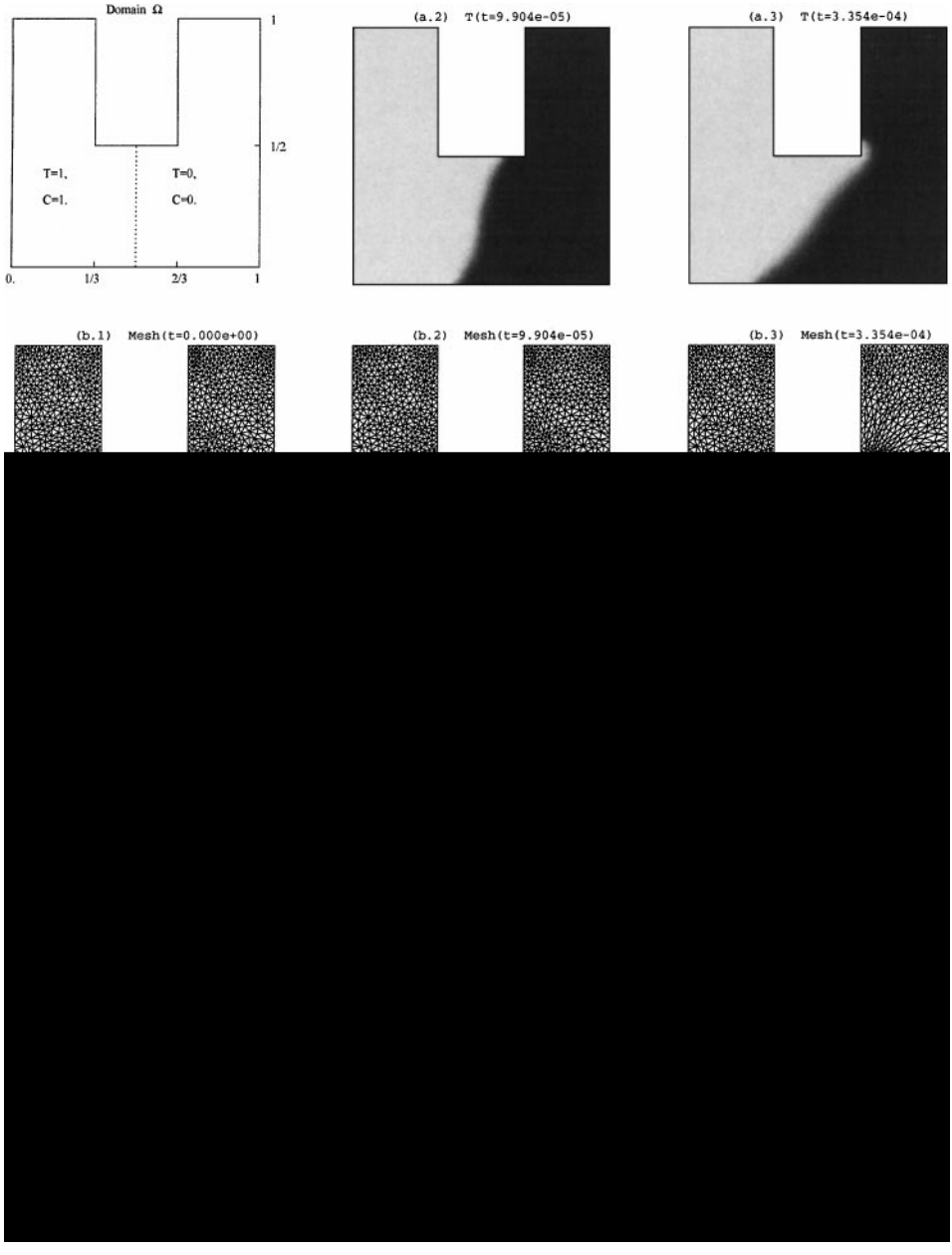


**FIG. 8.** The contour plot of the computed temperature $T$ (where the white represents 1 and black represents 0) and the corresponding moving mesh are depicted at various times for Example 5.3.

the Lewis number, $\phi$ is the porosity ratio, $\sigma$ is the heat capacity ratio, and $\partial(f, g)/\partial(x, y) = (\partial f/\partial x)(\partial g/\partial y) - (\partial f/\partial y)(\partial g/\partial x)$ for two arbitrary functions $f$ and $g$. The initial and boundary conditions are

$$\psi|_{t=0} = 0,$$
$$T|_{t=0} = C|_{t=0} = \begin{cases} 1, & \text{for } x \le \frac{1}{2}, \\ 0, & \text{for } x > \frac{1}{2}, \end{cases} \tag{24}$$

and

$$\begin{aligned} \psi|_{\partial\Omega} &= 0, && \text{for } t > 0, \\ \left.\frac{\partial T}{\partial \mathbf{n}}\right|_{\partial\Omega} &= \left.\frac{\partial C}{\partial \mathbf{n}}\right|_{\partial\Omega} = 0, && \text{for } t > 0, \end{aligned} \tag{25}$$

where $\mathbf{n}$ denotes the unit outward normal to the boundary $\partial\Omega$.

We simulate this phenomenon for the case of a large Rayleigh number, $\mathrm{Ra} = 1000$. Other parameters in (23) are $N = 0$, $\mathrm{Le} = 1$, and $\phi/\sigma = 1$. The solution domain is initially partitioned using the Delaunay triangulator *Triangle* into 3833 elements. For mesh movement we use the monitor function $G = \sqrt{1 + |\nabla T|^2} I$ and the time smoothing parameter $\tau = 10^{-2}$. The ODE system resulting from the linear finite element discretization of the PDEs is integrated using a dynamical time step size selection procedure with error tolerance $10^{-5}$. The results are shown in Fig. 8. It is clear that the mesh adapts well to the temperature and follows successfully the motion of the thin layer of large temperature and concentration variation. While the variation in temperature and concentration is gradually smoothed out by diffusion, the mesh is also becoming more uniform.

EXAMPLE 5.4. Our next example is a combustion problem considered in [17, 23]. The mathematical model is a system of coupled nonlinear reaction–diffusion equations

$$\begin{aligned} \frac{\partial u}{\partial t} - \nabla^2 u &= -\frac{R}{\alpha\delta} u\, e^{\delta(1-1/T)}, \\ \frac{\partial T}{\partial t} - \frac{1}{\mathrm{Le}}\nabla^2 T &= \frac{R}{\delta\,\mathrm{Le}} u\, e^{\delta(1-1/T)}, \end{aligned} \tag{26}$$

where $u$ and $T$ represent the dimensionless concentration and temperature of a chemical which is undertaking a one-step reaction. We consider the $J$-shape solution domain shown in Fig. 9. The initial and boundary conditions are

$$\begin{aligned} u|_{t=0} = T|_{t=0} &= 1, \quad \text{in } \Omega, \\ u|_{\partial\Omega} = T|_{\partial\Omega} &= 1, \quad \text{for } t > 0, \end{aligned} \tag{27}$$

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

The solution of this problem features a temperature that initially increases slowly from unity, forming a hot spot at the center of the right rectangular region, and then quickly jumping to approximately $1 + \alpha$ there. A sharp flame front is developed and propagates towards the boundary of the right rectangular region. Shortly after the flame front passes through the bridge in the middle of $\Omega$ and enters into the left rectangular region, another hot spot is formed in the left region. Then the left flame front expands and joins the right one to move to the boundary of $\Omega$, where it eventually settles due to the boundary conditions.

We start with an initial mesh consisting of 3293 quasi-uniform triangular elements. The monitor function $G$ is defined as $\sqrt{1 + \frac{1}{2}|\nabla T|^2} I$, to give higher mesh concentration in the flame front regions. The scaling factor $\frac{1}{2}$ is used to avoid excessive mesh concentration in regions of large $|\nabla T|$. This is equivalent to the use of Winslow's monitor function in (3) with a scaled adaptation function $(1/\sqrt{2})T$. The time-smoothing parameter $\tau$ and the time integrator for the physical ODE system are chosen as in the previous example. Figures 9 and 10 show the contour plot of the computed temperature and the moving mesh at different time levels. Once again, the moving mesh captures the solution features very well.

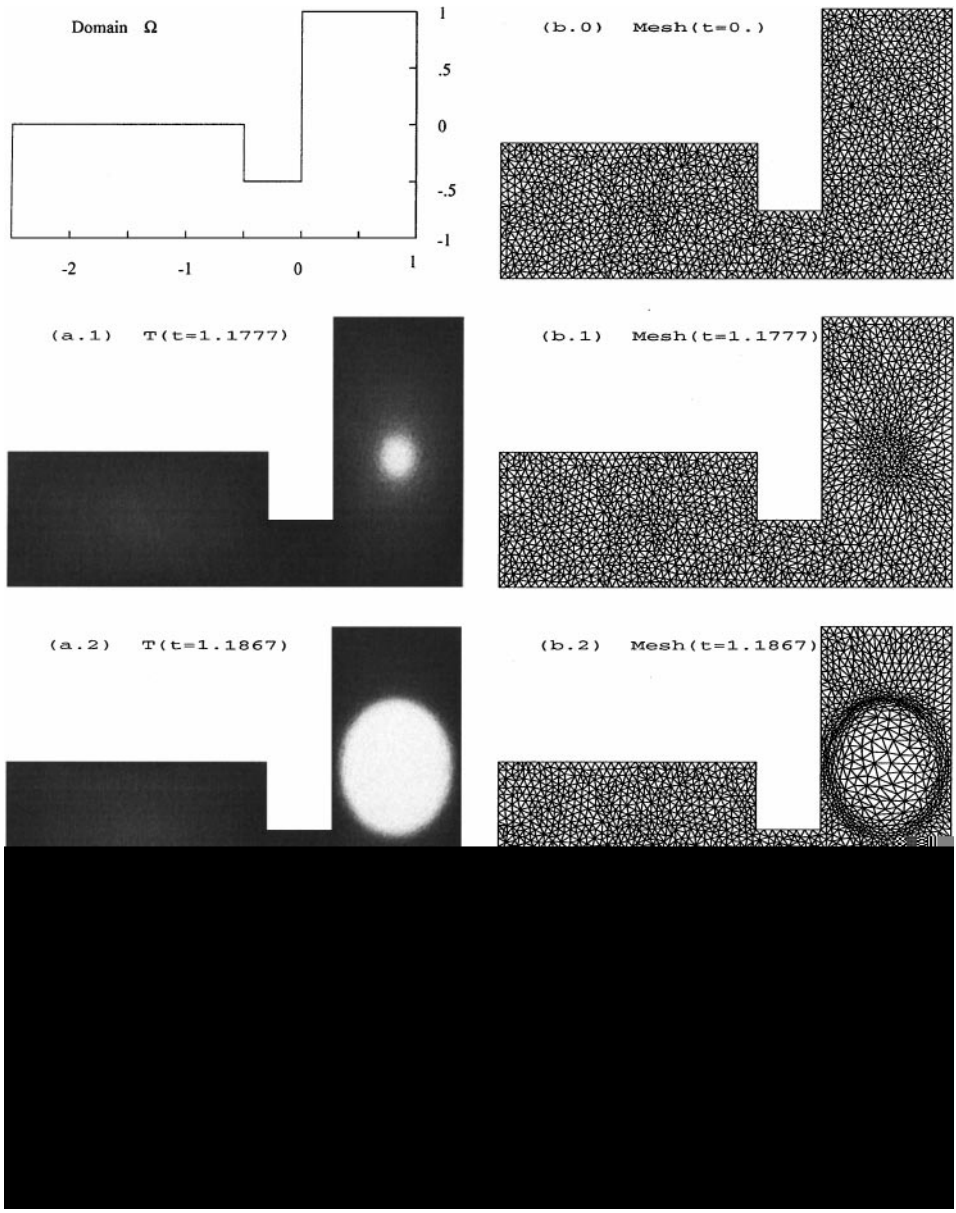


**FIG. 9.** The contour plot of the temperature $T$ (where white represents 2.2 and black represents 1) and the moving mesh for Example 5.4 are shown at various times.

EXAMPLE 5.5. Finally, we apply the moving finite element method to the problem of fluid flow past a cylinder.

Let $\psi$ and $\omega$ be the stream function and vorticity, respectively. The motion of the incompressible fluid is governed by the PDEs

$$\begin{aligned} \frac{\partial \omega}{\partial t} + \frac{\partial(\omega, \psi)}{\partial(x, y)} - \frac{2}{\mathrm{Re}} \nabla^2 \omega = 0, \\ -\nabla^2 \psi = \omega, \end{aligned} \tag{28}$$

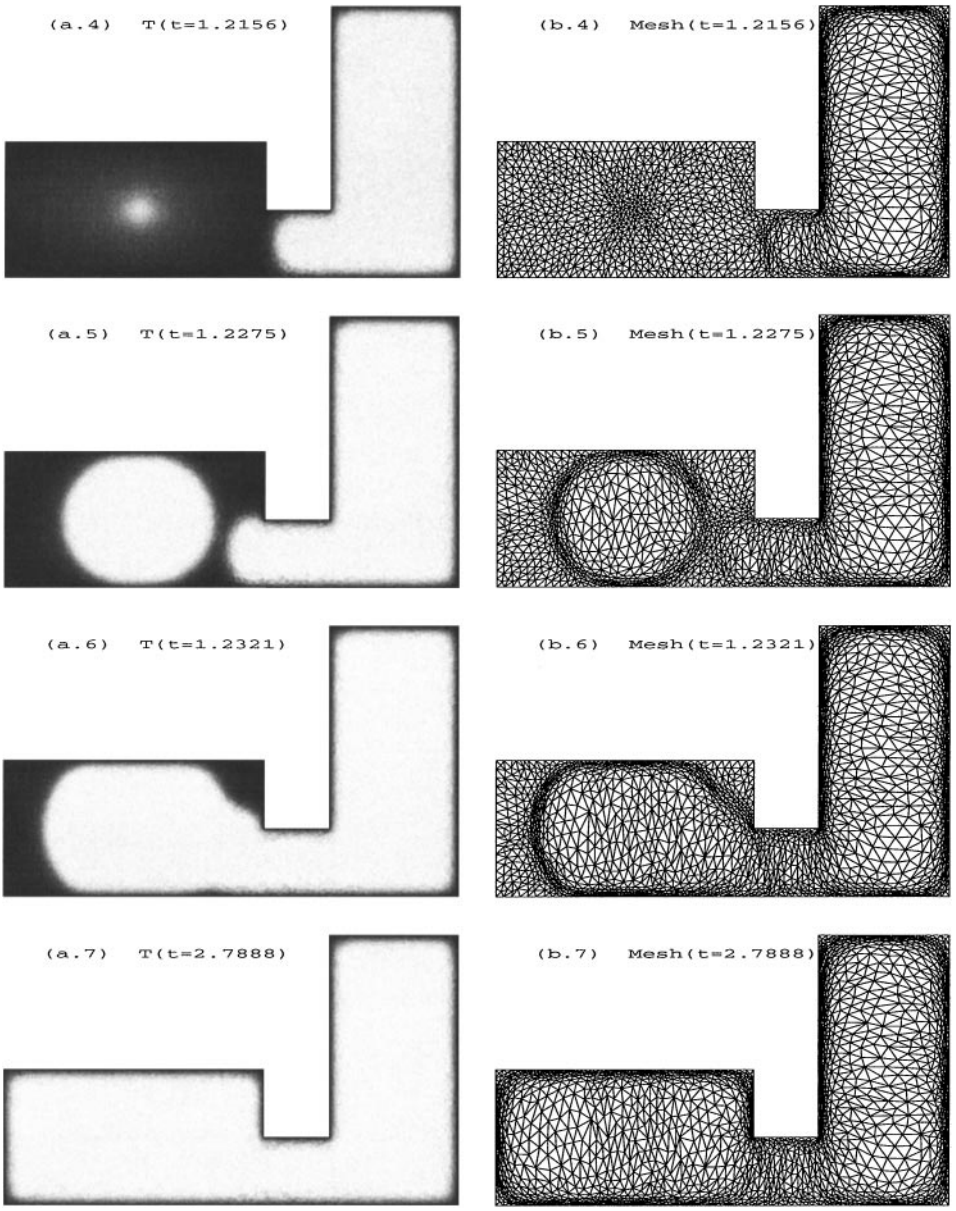


**FIG. 10.** Continued from Fig. 9.

with the Reynolds number being defined as $\mathrm{Re} = 2U_\infty a/\nu$, where $U_\infty$ is the speed of free flow in the far field, $a$ is the radius of the cylinder, and $\nu$ is the viscosity.

For numerical computation, we take $a = 1$ and $U_\infty = 1$ and truncate the infinite domain outside the cylinder at a circle with radius $r_\infty = 20$. On the surface of the cylinder, the nonslip boundary condition is applied, i.e.,

$$\psi = 0, \quad \frac{\partial \psi}{\partial \mathbf{n}} = 0.$$

At the outer boundary $r = r_\infty$, we choose the Dirichlet conditions

$$\psi = y, \quad \omega = 0.$$

The weak formulation of (28) is used to form the finite element discretization. More precisely, a weak solution $(\omega, \psi) \in S_\omega \times S_\psi$ is sought to satisfy

$$\left(\frac{\partial \omega}{\partial t}, \phi\right) - \left(\omega, \frac{\partial(\phi, \psi)}{\partial(x, y)}\right) + \frac{2}{\mathrm{Re}}(\nabla\omega, \nabla\phi) = 0 \quad \forall \phi \in W_\psi,$$
$$(\nabla\psi, \nabla\phi) - (\omega, \phi) = 0 \quad \forall \phi \in W_\omega, \tag{29}$$

where $W_\psi$, $W_\omega$, $S_\psi$, $S_\omega$ are suitable functional spaces (see [13] for details). Since our main concern in the present study is the steady state solution, a simple implicit Euler method is used for the time discretization of (29), with the nonlinear term approximated semi-implicitly by $(\omega(t_{n+1}), \partial(\phi, \psi(t_n))/\partial(x, y))$. The resulting linear system of equations at each time level is then solved by the BiCGStab2 method.

For mesh adaptation, there are a number of possible choices for the monitor function, depending upon where the mesh concentration is desired. We have used two different choices. One is based upon vorticity, i.e., $G = \sqrt{1 + |\omega|^4}\, I$. This choice gives higher mesh concentration in regions where the vorticity is large (see [6]). Another choice is based upon the stream function, viz., $G = \sqrt{1 + 1/(\epsilon + |\psi|)(1 + e^{1-x})}\, I$ with $\epsilon > 0$. This choice gives larger eigenvalues of $G$ around the stream line $\psi = 0$ behind the cylinder, and thus a high mesh concentration there for small $\epsilon$. We choose $\epsilon = 0.01$, although we have not had sufficient experience to comment on what range of values of $\epsilon$ would be suitable in general. The motivation for using the second type of monitor function is that for the current flow problem the regions enclosed by certain streamlines contain the most interesting fine structures of the flow which require higher resolution. This choice has advantages if these fine structures are of primary concern.

The adaptive meshes and the contour plots of the vorticity and stream functions obtained with the corresponding choices of the monitor function are shown in Fig. 11 for the case $\mathrm{Re} = 20$. Since the adaptive meshes are fairly fine around the cylinder in both cases, the solution profiles obtained are nearly indistinguishable. However, the meshes resulting from the two monitor functions do achieve our respective goals of having higher concentration around the regions of large vorticity or around $\psi = 0$. If a primary concern were the fine flow structures past the cylinder, then the better resolution from the second type of monitor function would be desirable. The standard benchmarking quantities for this problem are the lift and drag coefficients. Our results for these coefficients agree well with those reported in [12, 20] for Reynolds numbers in the range $\mathrm{Re} < 45$, for which the solution approaches a steady state and the comparison is thus meaningful.

## 6. CONCLUSIONS

In the previous sections we have presented an $r$-adaptive finite element method for solving time-dependent PDEs. The distinguishing features of the method are that the underlying mesh is unstructured and that the mesh movement for the FEM is done using MMPDEs developed in [17]. The first makes the method very general—it can be applied to many practical problems with complicated domains. The second feature facilitates smooth mesh evolution in time, which is needed to preserve stability of the method.

A key to the use of MMPDEs for unstructured mesh movement is to initially define the computational domain $\Omega_c$ and to compute the computational mesh $\Omega_c^h$. Unlike in the case of structured mesh movement, $\Omega_c$ and $\Omega_c^h$ cannot now be chosen simply to be a square and a uniform rectangular mesh, respectively. Instead, they must be defined and computed in

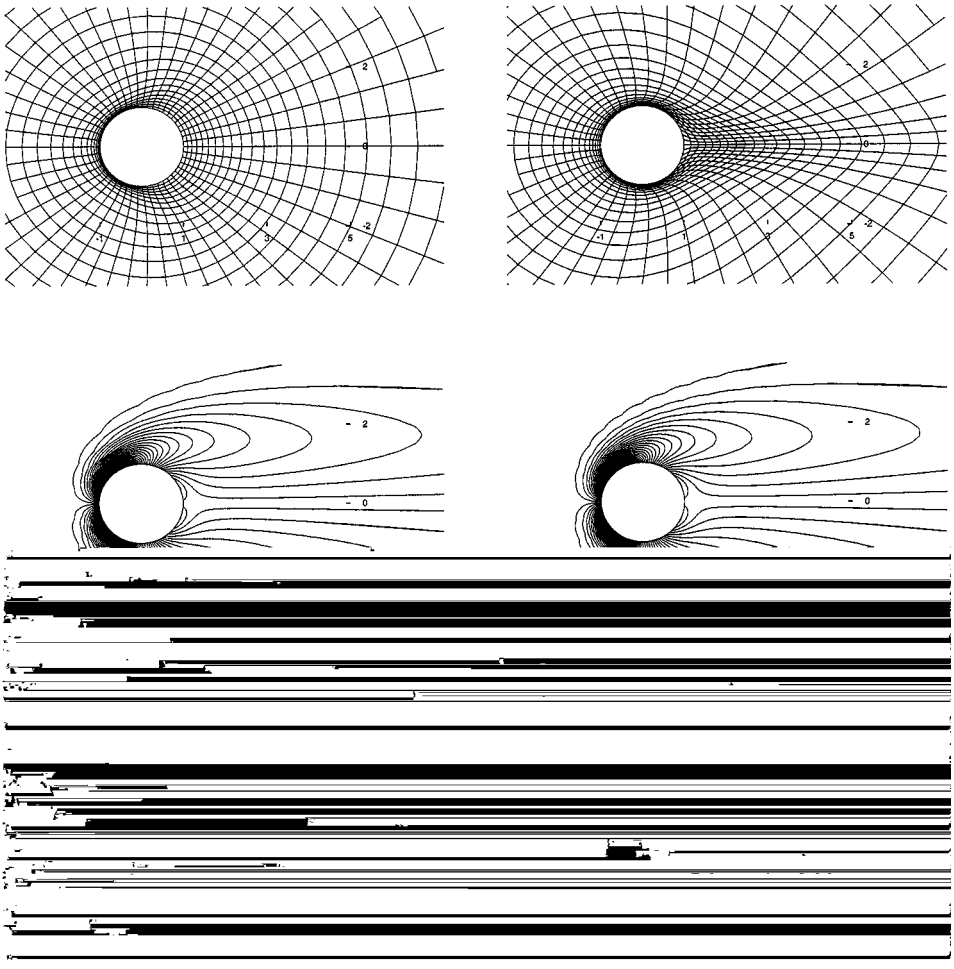

**FIG. 11.** The adaptive mesh, vorticity, and stream lines for Example 5.5 with $\mathrm{Re} = 20$. The pictures on the left are results obtained with $G = \sqrt{1 + |\omega|^4 I}$ and those on the right side obtained with $G = \sqrt{1 + 1/(0.01 + |\psi|)(1 + e^{1-x})I}$.

conjunction with $\Omega$ and the given mesh $\bar{\Omega}^h(0)$. A detailed discussion of this issue is given in Section 3, as well as some illustrative numerical results. It is shown that the resulting initial adaptive mesh $\Omega^h(0)$ is insensitive to the choice of the computational domain. Nevertheless, we recommend that $\Omega_c$ should generally be chosen to be convex and to have the same number of boundary segments as $\Omega$ in order to avoid introducing degenerate elements in the computational mesh. Furthermore, analysis and numerical results show that $\Omega^h(0)$ has the propensity to concentrate the points in the same regions as the initial mesh $\bar{\Omega}^h(0)$.

The FEM discretization of physical PDEs has some special features caused by mesh movement on an unstructured grid, and these are discussed in Section 4. While the numerical examples which follow show the versatility of our moving FEM, there are several limitations, perhaps the most significant being that there is currently not the capability to do error estimation nor to add or remove grid points (viz., $h$-refinement). As a result, if a solution has several distinct qualitative features of interest (as in the last example, where mesh concentration can be desired around the front of the cylinder and where vortex

shedding occurs in the back), then selecting a monitor function which properly balances these two concerns is very difficult. Nevertheless, this $r$-method should ideally complement the currently popular $h$-$p$ FEM approach, and one of our next goals is to improve the efficiency of this latter approach by providing it with the moving mesh capability.

## ACKNOWLEDGMENTS

This work was supported in part by NSERC (Canada) Grant OGP-0008781 and NSF (USA) Grant DMS-9626107.

## REFERENCES

1. I. Babuška and W. C. Rheinboldt, Adaptive approaches and reliability estimations in finite element analysis, *Comput. Methods Appl. Mech. Engrg.* **18**, 323 (1979).
2. M. J. Baines, *Moving Finite Elements* (Oxford Univ. Press, Oxford, 1994).
3. R. E. Bank and R. F. Santos, An analysis of some moving space-time finite element methods, *SIAM J. Numer. Anal.* **30**, 1 (1993).
4. J. U. Brackbill, An adaptive grid with direction control, *J. Comput. Phys.* **108**, 38 (1993).
5. J. U. Brackbill and J. S. Saltzman, Adaptive zoning for singular problems in two dimensions, *J. Comput. Phys.* **46**, 342 (1982).
6. W. Cao, W. Huang, and R. D. Russell, A study of monitor functions for two dimensional adaptive mesh generation, *SIAM J. Sci. Comput.*, to appear.
7. W. Cao, W. Huang, and R. D. Russell, A moving mesh method in multi-block regions with application to a combustion problem, in preparation.
8. N. Carlson and K. Miller, Design and application of a gradient-weighted moving finite element code. Part II. In two dimensions, *SIAM J. Sci. Comput.* **19**, 766 (1998).
9. S. F. Davis and J. E. Flaherty, An adaptive finite element method for initial boundary values problems for partial differential equations, *SIAM J. Sci. Stat. Comput.* **3**, 6 (1982).
10. T. D. Dupont, Mesh modification for evolution equations, *Math. Comput.* **39**, 85 (1982).
11. A. S. Dvinsky, Adaptive grid generation from harmonic maps on Riemannian manifolds, *J. Comput. Phys.* **95**, 450 (1991).
12. B. Fornberg, Steady viscous flow past a circular cylinder up to Reynolds 600, *J. Comput. Phys.* **61**, 297 (1985).
13. M. D. Gunzburger and J. S. Peterson, Finite element method for the stream function vorticity equations: Boundary condition treatments and multiply connected domains, *SIAM J. Sci. Comput.* **9**, 650 (1988).
14. M. H. Gutknecht, Variants of BICGSTAB for matrices with complex spectrum, *SIAM J. Sci. Comput.* **14**, 1020 (1993).
15. E. Hairer, S. P. Norsett, and G. Wanner, *Solving Ordinary Differential Equations*, Vols. I, II (Springer-Verlag, Berlin/New York, 1987).
16. R. Hamilton, *Harmonic Maps of Manifolds with Boundary*, Lecture Notes in Mathematics, Vol. 471 (Springer-Verlag, New York, 1975).
17. W. Huang and R. D. Russell, Moving mesh strategy based upon a gradient flow equation for two dimensional problems, *SIAM J. Sci. Comput.*, to appear.
18. W. Huang and R. D. Russell, A high dimensional moving mesh strategy, *Appl. Numer. Math.* **26**, 63 (1997).
19. W. Huang, Y. Ren, and R. D. Russell, Moving mesh partial differential equations (MMPDEs) based upon the equidistribution principle, *SIAM J. Numer. Anal.* **31**, 709 (1994).
24. D. A. Neid and A. Bejan, *Convection in Porous Media* (Springer-Verlag, New York, 1992).
25. J. R. Shewchuk, Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator, Technical report, School of Computer Science, Carnegie Mellon University, 1996.
26. J. F. Thompson, Z. A. Warsi, and C. W. Mastin, *Numerical Grid Generation* (North-Holland, Amsterdam, 1985).

27. H. A. van der Vorst, BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.* **13**, 631 (1992).
28. Z. U. A. Warsi and J. F. Thompson, Application of variational methods in the fixed and adaptive grid generation, *Comput. Math. Appl.* **19**, 31 (1990).
29. A. Winslow, Numerical solution of the quasi-linear Poisson equation in a nonuniform triangle mesh, *J. Comput. Phys.* **1**, 149 (1967).
30. P. A. Zegeling, *Moving-grid methods for time-dependent partial differential equations* (CWI, Amsterdam, 1993), Tract 94.