Journal of Computational Physics **159,** 103–122 (2000)
doi:10.1006/jcph.2000.6432, available online at http://www.idealibrary.com on IDEAL®

Level-Set-Based Deformation Methods for Adaptive Grids

Guojun Liao,*[,1] Feng Liu,† Gary C. de la Pena,*[,2] Danping Peng,‡ and Stanley Osher§[,3]

**Department of Mathematics, University of Texas, Arlington, Texas 76019-0408; †Department of Mechanical and Aerospace Engineering, University of California Irvine, Irvine, California 92027; ‡ Barra Inc., Berkeley, California 94704; and § Department of Mathematics, University of California at Los Angeles, Los Angeles, California 90095-1555*

E-mail: liao@uta.edu, fliu@uci.eng.edu, delapena@math.uta.edu, danpingpeng@barra.com, sjo@math.ucla.edu

Received December 7, 1998; revised December 23, 1999

A new method for generating adaptive moving grids is formulated based on physical quantities. Level set functions are used to construct the adaptive grids, which are solutions of the standard level set evolution equation with the Cartesian coordinates as initial values. The intersection points of the level sets of the evolving functions form a new grid at each time. The velocity vector in the evolution equation is chosen according to a monitor function and is equal to the node velocity. A uniform grid is then deformed to a moving grid with desired cell volume distribution at each time. The method achieves precise control over the Jacobian determinant of the grid mapping as the traditional deformation method does. The new method is consistent with the level set approach to dynamic moving interface problems. © 2000 Academic Press

Key Words: adaptive grids; deformation methods; level set functions.

1. INTRODUCTION

Key problems in numerical simulation of time-dependent partial differential equations are grid generation and grid adaptation. General grid generation methods are discussed by Thompson *et al.* [2], Zegeling [3], Knupp and Steinberg [4], Carey [5], and Liseikin [6]. The problem this paper deals with is how to generate adaptive moving grids.

The tasks of simulating transient problems on three-dimensional domains become enormously difficult when tens of millions of nodes are needed. This is especially so in transient problems with moving fronts, shock waves, etc. For instance, to correctly simulate the

[1] Supported by NSF under Grant DMS-9732742.

[2] Supported by NSF under Grant DMS-9732742.

[3] Supported by NSF and DARPA under Grants DMS-9706827 and DMS-9615854.

dendritic growth of a crystal modeled by a Stefan problem, one must use fine grids near the interface between the solid phase and the liquid phase. Figure 8 of [26] shows the importance of grid sizes. Using a state-of-the-art level set method, the 100×100, 200×200, and 300×300 fixed uniform grids on the unit square give rise to unsatisfactory results at the interface. It takes the 400×400 fixed grid to produce a sharp result. A 3D simulation would need 64 million nodes. This would be too costly.

We hope to improve the accuracy and efficiency of the simulation by using adaptive grids. The idea is to generate the grids according to the salient features of the solution, so that the nodes will be concentrated in regions where the solution changes rapidly in order to improve accuracy, and fewer grid points are used in regions where small changes in the solution occur.

We now describe a general idea of moving grids. Suppose that we want to simulate a scalar or vector field $\mathbf{u}(\mathbf{x}, t)$ satisfying

$$\mathbf{u}_t(\mathbf{x}, t) = L(\mathbf{u}); \tag{1}$$

here L is a differential operator defined on a physical domain $\Omega = D_2$ in R^n, $n = 1, 2, 3$. A common idea is to construct a transformation ϕ: $D_1 \times [0, T] \to D_2$ which moves a fixed number of grid points on Ω to adapt to the numerical solution as it is being computed on the computational domain D_1. To be qualified as a transformation, ϕ must be one to one and onto. Variational methods (cf. [1, 7]) and elliptic PDE methods (cf. [2]) define this transformation as the solution of a system of PDEs which is created to control various aspects of the grid such as orthogonality ("skewness"), smoothness, and cell size. The resulting system of PDEs for grid generation is often nonlinear and its solution requires intensive computation. Significant contributions were made to dynamically adapt the grid by controlling the cell size through the Jacobian determinant of the transformation in [1, 2, 7, 8]. The moving finite element method was developed in [9] and is useful for certain unsteady problems. Recently, moving mesh methods based on moving mesh partial differential equations [12] were developed that have remarkable capability to track rapid spatial and temporal transitions for some model problems. Hybrid techniques that use both grid motion and local refinement showed their effectiveness for 2D problems [10].

The deformation method originated from differential geometry (see [13, 14]). It determines the node velocity by a monitor function and thus the time-dependent differential equations can be transformed by nodal mapping into the computational domain. The transformed equation can then be simulated on a fixed orthogonal grid. The static version of the deformation method was used with a finite volume solver in flow calculation problems [19]. A one-dimensional version of the method was used with a discontinuous Galerkin finite element method in numerical simulation of a convection–diffusion problem [17]. For finite difference algorithms that are based on orthogonal grids, we transform Eq. (1) by $\mathbf{x} = \phi(\boldsymbol{\xi}, t)$ and solve the transformed equation on a fixed orthogonal grid on the $\boldsymbol{\xi}$-domain (the commutational domain).

In this paper, we formulate a new deformation method which is based on the level set approach and the transport formula from fluid dynamics. The level set deformation method moves the nodes with a proper velocity so that the nodal mapping has the desired Jacobian determinant. Thus it precisely controls the cell size distribution according to a positive monitor function. As in the deformation method, the velocity vector field is constructed by solving a Poisson equation determined by the monitor function. The main difference between the two methods is that the deformation method uses a system of ODEs to move the nodes while the level set deformation method uses a system of level set evolution PDEs to generate the moving grid. The ODEs are, of course, the characteristic equations for the PDEs.

Earlier work using the level set method for grid generation was done by Sethian in [24]. Our technique is quite different. We control the Jacobian of the grid mapping, while in [24] the main idea is to create a body-fitted coordinate using the level set function. Then the other coordinates are obtained by solving ODEs. We note that our method can also be extended to obtain body-fitted coordinates while still controlling the Jacobian.

Recall that the Jacobian determinant of a mapping $\phi(\mathbf{x}, t)$ from D_1 to D_2 in R^n, $n = 1, 2, 3$, is $J(\phi) = \det \nabla\phi = |dA'|/|dA|$, where dA' is the image of a volume (area, in 2D) element dA. Our goal is to construct ϕ such that $J(\phi) = f(\phi, t)$, since this will give precise control over the cell size in any dimensions.

Suppose that the solution to (1) has been computed at time step $t = t_{k-1}$, and a preliminary computation has been done at time level $t = t_k$. Assume that we are provided with some positive error estimator $\delta(\mathbf{x}, t)$ at the time step t_k. Define a monitor function,

$$f(\mathbf{x}, t) = C_1/\delta(\mathbf{x}, t), \tag{2}$$

where $C_1 = C_1(t)$ is a positive scaling parameter such that at each time step we have

$$\int_{D_2} \left(\frac{1}{f(\mathbf{x}, t_k)} - 1 \right) dA = 0. \tag{3}$$

We then seek a transformation ϕ: $D_1 \to D_2 = D_2$ such that

$$\begin{aligned} \det \nabla\phi(\mathbf{x}, t) &= f(\phi(\mathbf{x}, t), t) \quad && t_{k-1} \le t \le t_k, \\ \phi(\mathbf{x}, t_{k-1}) &= \phi_{k-1}(\mathbf{x}) && \mathbf{x} \text{ on } D_1, \end{aligned} \tag{4}$$

where $\mathbf{x}$ is a grid node of an initial grid and $\phi_{k-1}(\mathbf{x})$ represents the coordinates of the node at $t = t_{k-1}$ ((3) is necessary for (4) to be true). We specify that $\phi(\mathbf{x}) \in \partial D_2$ for all $\mathbf{x} \in \partial D_1$. Note that (4) ensures the size of the transformed cells will be proportional to f; i.e., the grid will be appropriately condensed in regions of high error and stretched in regions of small error. It is well known that if the Jacobian determinant of such a transformation ϕ is positive in D_1, then ϕ is one-to-one in all of D_1, ensuring that the grid will not fold onto itself. Various equidistribution principles can be used to construct the monitor function. A posteriori error estimates (if available), residuals, truncation errors, etc. are redistributed evenly over the whole domain. In most cases, we want to put refined grids in the regions where u changes rapidly. For instance, if the flow patterns exhibit shock waves, we can take (for Euler flows)

$$f = C_1/\big(1 + C_2|\nabla p|^2\big), \tag{5}$$

where p is the pressure, C_2 is a constant for adaptation itensity, and C_1 is a normalization parameter. In addition to the gradient of p, terms involving the value of p and the second derivatives of p (or the curvature of its level sets) can also be included. For instance,

$$f = C_1/\big(1 + \alpha|p|^2 + \beta|\nabla p|^2 + \gamma|\nabla^2 p|^2\big). \tag{6}$$

For interface resolution, we can, for instance, construct f by using the signed distance function d from the interface as follows: Let f be piecewise linear such that

$$f = \begin{cases} 1 & \text{if } |d| > 0.1 \\ 0.2 & \text{if } d = 0. \end{cases} \tag{7}$$

Normalize f so that $\int_D (1/f - 1) = 0$, which is required if (4) is to be satisfied. The constants 0.1 and 0.2 in (7) can be changed according to the desired intensity of adaptation at the interface. The signed distance function can easily be computed as was done in [25].

2. A LEVEL SET DEFORMATION METHOD

In this section, a new moving grid method is formulated. The usual evolution equation for level set functions (as in [27]) with the Cartesian coordinates as initial values is solved. The velocity vector in the evolution equation is chosen according to a monitor function. The intersection points of the level sets of the evolving solutions will form a new grid at each time. Numerical examples will be provided in which a uniform grid is deformed to moving grids with prescribed cell size distribution at each time.

We first set up the principle of redistribution in the one-dimensional case before describing the method in multiple dimensions. Suppose that a positive monitor function $f(x, t)$ is given. We want to construct a grid of $N + 1$ nodes,

$$x_0(t) = 0 < x_1(t) < x_2(t) < \cdots < x_i(t) < x_{i+1}(t) < \cdots < x_N(t) = 1,$$

on [0, 1] at each time t with the length $x_{i+1} - x_i = f(x_i', t)/N$ where x_i' is the midpoint of the subinterval $[x_i, x_{i+1}]$. We seek a level set function ϕ from [0, 1] to [0, 1], which sends x_i to $k_i = \phi(x_i)$, where points $k_i = i/N$, $i = 0, 1, 2, \ldots, N$, form a uniform grid on the interval [0, 1] of the y-axis. The condition $x_{i+1} - x_i = f(x_i', t)/N$ is equivalent to

$$(1/N)/(x_{i+1} - x_i) = 1/f(x_i', t),$$

whose left hand side tends to the Jacobian determinant $\partial\phi/\partial x$ as $N \to \infty$. At the limit, we get the condition for ϕ that

$$\partial\phi/\partial x = 1/f(x, t) = g(x, t) \quad (\text{denoting } g = 1/f). \tag{8}$$

In 1D, this equation can be solved by direct integration,

$$\phi(x) = \int_0^x (1/f(s, t))\, ds,$$

where f is normalized to satisfy the condition $\phi(1) = \int_0^1 (1/f(x, t))\, dx = 1$ for each t. The preimages of the evenly placed points $k_i = i/N$ under the transformation $x \to \phi(x, t)$ are the level sets of ϕ and they form the new nodes. This is illustrated by Fig. 1 (see also Figs. 2a–2i of Example 1). As one can see, evenly placed horizontal lines intersect the graph of a monotonic function ϕ. The projections of the intersection points onto the x-axis are the nodes x_i, $i = 1, 2, 3, \ldots,$ of the new grid. By properly evolving the function ϕ, we can control the spacing of the moving grid on $0 \le x \le 1$ precisely.

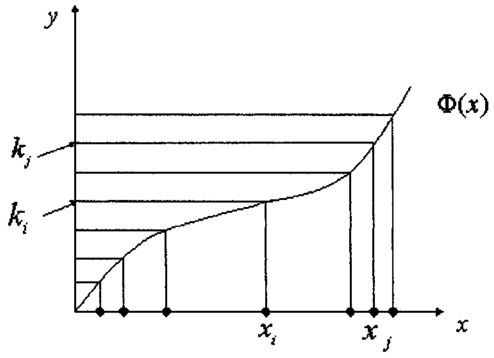


FIG. 1. Level curve.

This idea extends to multiple dimensions as follows. The goal is to generate a nodal mapping Φ from $D_1 \to D_2$ such that $J(\Phi) = 1/f$ for a positive monitor function f. To begin, let us recall the basic concept of modeling a moving front by level sets. Suppose that there is a moving front in a fluid flow with a velocity field $\mathbf{v} = (x_t, y_t, z_t)$, where $\mathbf{x} = (x, y, z)$ is the position of a (fluid) particle at time t. We introduce a smooth function $\phi(x, y, z, t)$ with the property that the front is given by the zero level set of ϕ, i.e., $\phi(x, y, z, t) = 0$. Differentiate the identity with respect to t; we get $\phi_t + \phi_x x_t + \phi_y y_t + \phi_z z_t = 0$, which can be written as

$$\phi_t(x, y, z, t) + \langle \nabla\phi, \mathbf{v}\rangle = 0. \tag{9}$$

This is the evolution equation for the level set function ϕ. In the calculations of fluid flows, $\mathbf{v}$ is the velocity of the fluid particle occupying the point $(x(t), y(t), z(t))$ at time t. Other important geometric parameters such as the curvature of the front and the normal vector to the front can easily be determined from ϕ. See [27] for an overview. In the level set moving grid method, $\mathbf{v}$ is the node velocity.

Our purpose is to generate an adaptive grid according to a positive monitor function f. In two dimensions, we construct two functions ϕ and ψ by (9) with a suitable vector field $\mathbf{v}$ ($\mathbf{v}$ is determined in (18) below). Then the intersections of their level set curves will be the new nodes. Thus, let $\phi(x, y, t)$ and $\psi(x, y, t)$ be solutions to the evolution PDE

$$\begin{cases} \phi_t(x, y, t) + \langle \nabla\phi, \mathbf{v}\rangle = 0 \\ \psi_t(x, y, t) + \langle \nabla\psi, \mathbf{v}\rangle = 0. \end{cases} \tag{10}$$

The initial conditions are $\phi(x, y, 0) = x$, $\psi(x, y, 0) = y$, respectively. The boundary conditions are

$$\phi(0, y, t) = 0, \quad \phi(1, y, t) = 1, \quad \phi(x, 0, t) = \phi(x, 1, 0) = x;$$
$$\psi(x, 0, t) = 0, \quad \psi(x, 1, t) = 1, \quad \psi(0, y, t) = \psi(1, y, t) = y.$$

Let $\Phi = (\phi, \psi)$. The vector field $\mathbf{v}$ is chosen so that the Jacobian determinant Φ is equal to the reciprocal of f, namely

$$J(\Phi) = \partial(\phi, \psi)/\partial(x, y) = 1/f(x, y, t). \tag{11}$$

Note that this condition is the natural extension of the 1D condition (8).

In three dimensions, we solve for three functions ϕ_1, ϕ_2, ϕ_3 from the equations

$$(\phi_i)_t + \langle \nabla\phi_i, \mathbf{v}\rangle = 0, \quad i = 1, 2, 3, \tag{12}$$

with the same type of initial and boundary conditions as in 2D.

Let $\Phi = (\phi_1, \phi_2, \phi_3)$. A suitable vector field $\mathbf{v}$ can be determined so that the Jacobian determinant of the mapping Φ is equal to the reciprocal of a monitor function f, namely

$$J(\phi) = D(\phi_1, \phi_2, \phi_3)/D(x, y, z) = 1/f(x, y, z, t). \tag{13}$$

The intersections of their level sets form the nodes of the moving grid.

The key to the success of the proposed method is to determine the velocity vector field $\mathbf{v}$ so that $J = 1/f = g$ at every t. Thus, the grid cell size can be precisely controlled, resulting in a moving grid that is adaptive according to the monitor function f. A proper velocity vector $\mathbf{v}$ can be determined by the condition

$$g_t + \operatorname{div}(g\mathbf{v}) = 0. \tag{14}$$

This choice of $\mathbf{v}$ is based on the transport formula in fluid dynamics, which can be found in any standard textbook on fluid dynamics [28, 29].

Let Ω_t be the image of an initial region Ω_0 under the flow of the velocity field $\mathbf{v} = (x_t, y_t, z_t)$, where $\mathbf{x} = (x, y, z)$ is the position of a (fluid) particle at time t.

THEOREM (Transport Formula). *For any function $h(\mathbf{x}, t)$, we have*

$$\frac{d}{dt}\int_{\Omega_t} h\,dV = \int_{\Omega_t}\left(\frac{\partial h}{\partial t} + \operatorname{div}(h\mathbf{v})\right)dV. \tag{15}$$

Let $J = D(\phi_1, \phi_2, \phi_3)/D(x, y, z)$ be the Jacobian determinant of the transformation $(x, y, z) \to (\phi_1, \phi_2, \phi_3)$. Taking $h = g(\mathbf{x}, t) = 1/f$ in (15), we get, by a change of variables, that

$$\frac{d}{dt}\int_{\Omega_t} g\,dV = \frac{d}{dt}\int_{\Omega_0} gj^{-1}\,dV = \int_{\Omega_t}\left(\frac{\partial g}{\partial t} + \operatorname{div}(g\mathbf{v})\right)dV = 0, \tag{16}$$

where (14) is used to get the last equation. In the change of variables, $D(x, y, z)/D(\phi_1, \phi_2, \phi_3) = J^{-1}$ is used. (16) implies that $gJ^{-1} =$ constant since Ω_0 is arbitrary. Choose $(gJ^{-1})|_{t=0} = 1$. Then we get $gJ^{-1} = 1$ for any $t > 0$ as desired.

To solve for $\mathbf{v}$ from (14), we first observe that, in one dimension, condition (14) becomes

$$\frac{\partial^2 w}{\partial x^2} = \partial(gv)/\partial x = -g_t, \quad \text{where } w = gv,$$

and we can solve for v by direct integration and get

$$v(x, t) = -\left(\int_0^x g_t\right)\Big/ g = \frac{2w}{2x}\Big/ g.$$

This suggests a simple method for determining the velocity vector field $\mathbf{v}$ in multiple dimensions. Let $f(x, y, z, t) > 0$ be the desired grid cell size distribution, which is constructed according to the physical variables being simulated and is normalized as in (3). Let

$g = 1/f$. We first solve for a real-valued function (potential) w from the Poisson equation on the xyz-domain,

$$\triangle w = -g_t, \tag{17}$$

with the Neumann boundary condition. Then we set

$$\mathbf{v} = \nabla w / g. \tag{18}$$

It follows that $\operatorname{div}(g\mathbf{v}) = \operatorname{div}(\nabla w) = \triangle w = -g_t$ as desired. The method is based on solving a scalar Poisson equation, and thus it works for general three-dimensional domains.

2.1. *Numerical Examples*

EXAMPLE 1 (Fig. 2). Moving grids on [0, 1]. Let $f = 1/g$ where $g = 1 + 10t(x^2 - x + 1/6)$. By direct integration, we can verify that

$$\int_0^1 g\,dx = [x + 10t(x^3/3 - x^2/2 + x/6)]_{x=0}^{x=1} = 1, \text{ for every } t.$$

Thus the normalization condition (3) is analytically satisfied. We want to generate a moving grid which is a uniform grid on [0, 1] at $t = 0$. In 1D the velocity field is a real-valued function. Solving for v from condition (14),

$$\operatorname{div}(gv) = (gv)_x = -g_t,$$

we get, by direct integration,

$$v = 10(-x^3/3 + x^2 - x/6)/g.$$

Next, solve for ϕ: $[0, 1] \times [0, T] \rightarrow [0, 1]$ from the evolution equation

$$\phi_t(x, t) + \langle \phi_x, v \rangle = 0$$

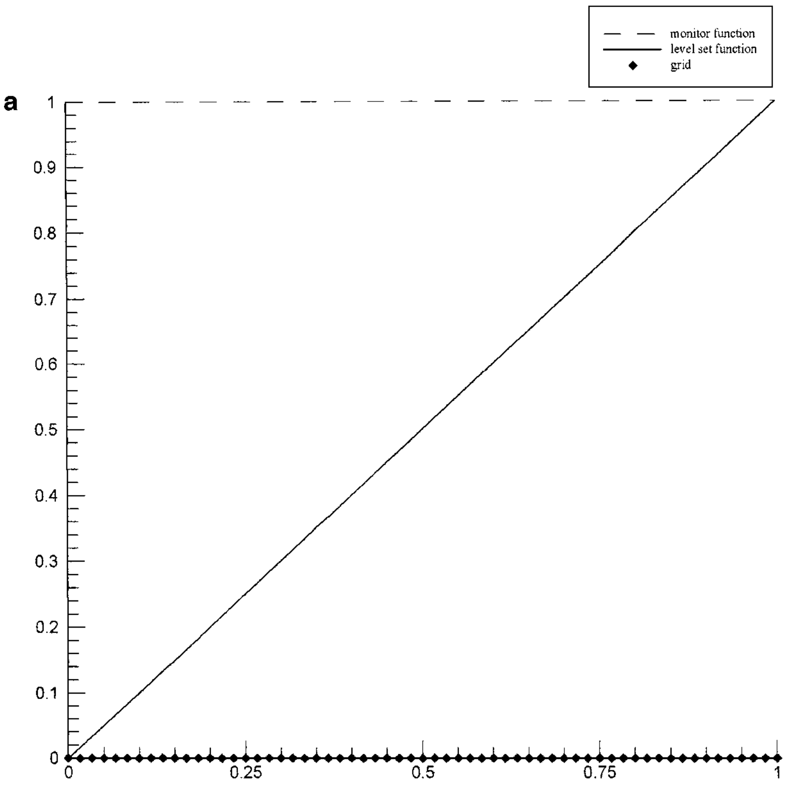
with the initial and boundary conditions $\phi(x, 0) = x$, $\phi(0, t) = 0$, $\phi(1, t) = 1$.

Let $1/N$ be the spacing of a uniform grid on [0, 1]. The preimages of the nodes of the uniform grid on [0, 1] form the moving grid at selected time t. In Fig. 2, f and ϕ are plotted along with the nodes of the moving grid with $N = 60$. Note that the grid spacing near $x = 0$ and $x = 1$ is getting smaller and it is getting larger near $x = 0.5$. In fact, this is the intended distribution since $d\phi/dx = g = 1/f$, which means $\Delta x_i = f/N$.

EXAMPLE 2 (Fig. 3). A 60×60 uniform grid is deformed into a grid concentrated around a pair of circles and the grid moves appropriately as the circles merge into each other. Here the function d is the product of level set functions for two moving circles,

$$d = \left((x - a_1)^2 + (y - b_1)^2 - r^2\right)\left((x - a_2)^2 + (y - b_2)^2 - \rho^2\right),$$

where (a_1, b_1) is the (moving) center of the first circle and (a_2, b_2) is the (moving) center of the second circle. The zero set of d consists of the two circles and r and ρ are their varying radii. Initially $(a_1, b_1) = (0.4, 0.5)$, $(a_2, b_2) = (0.6, 0.5)$, and $r = \rho = 0.18$. The level set



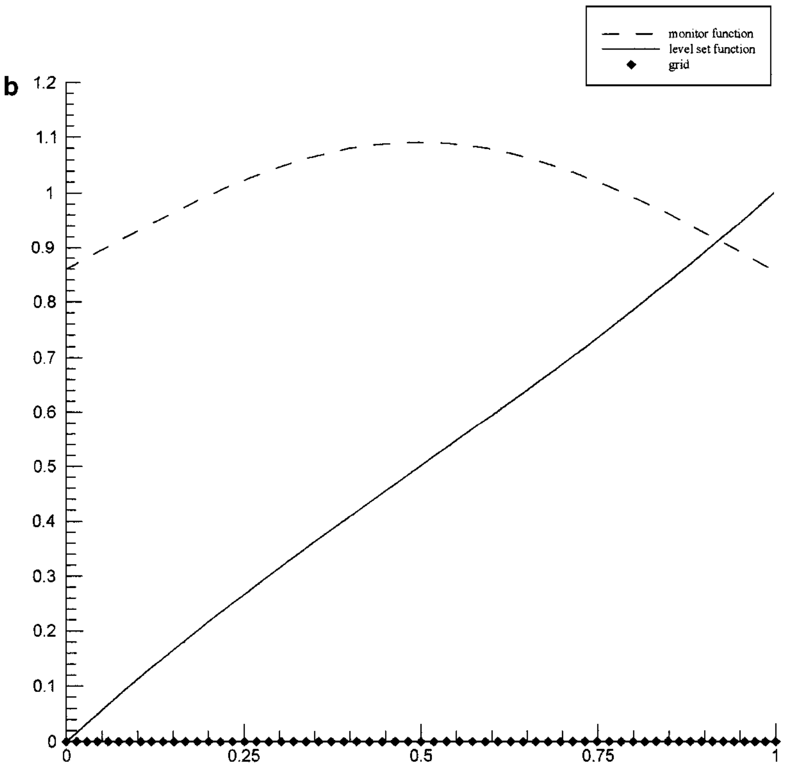
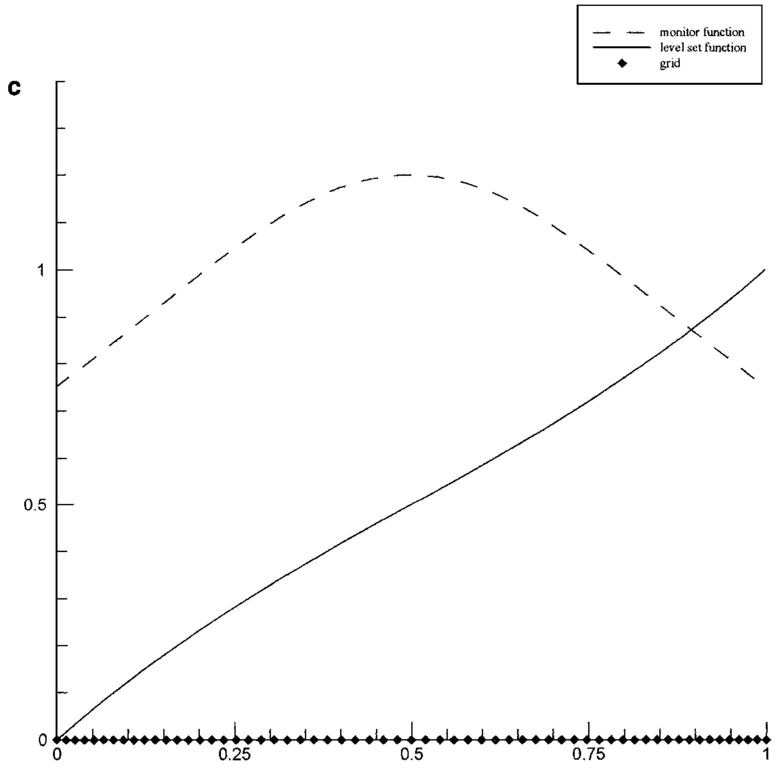


FIG. 2. Monitor functions, level set functions, and grid plots for Example 1.



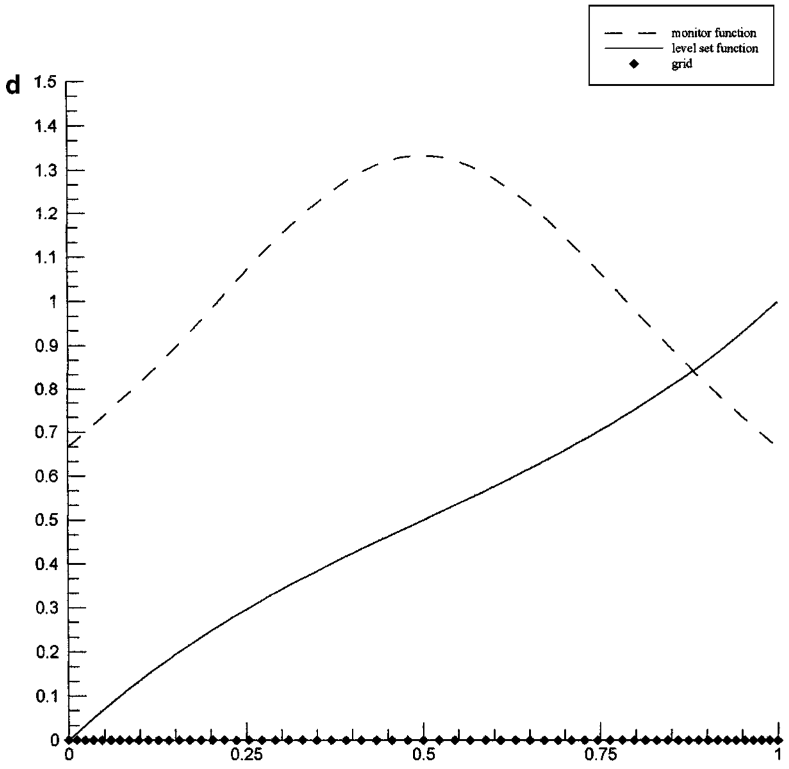
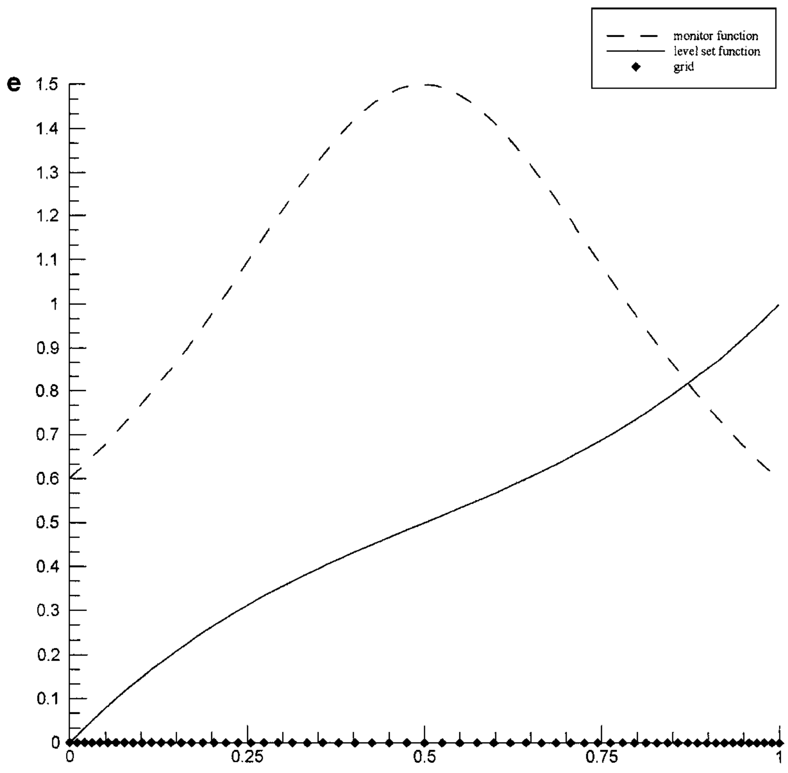


FIG. 2—*Continued*



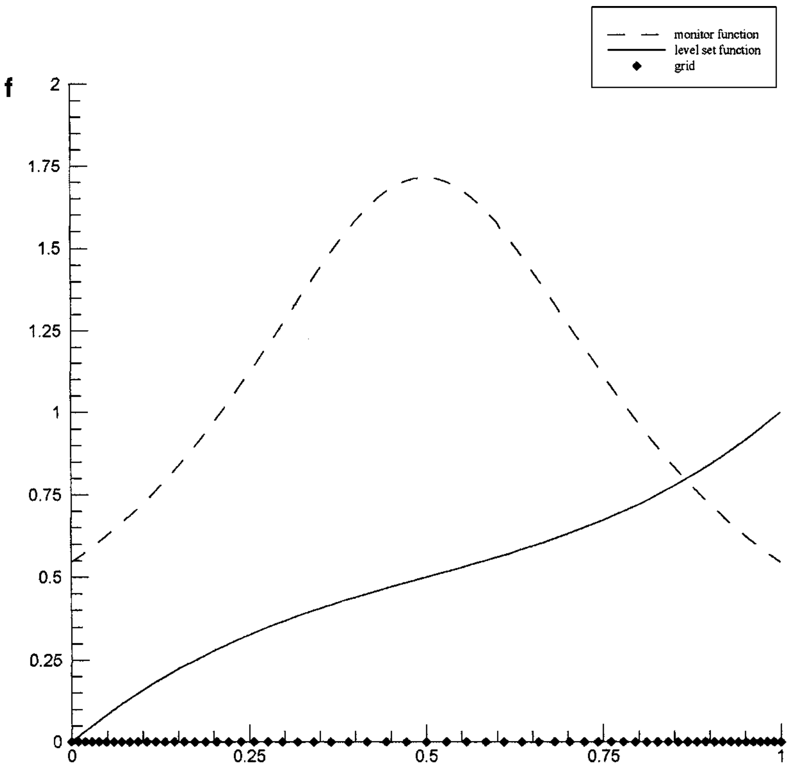
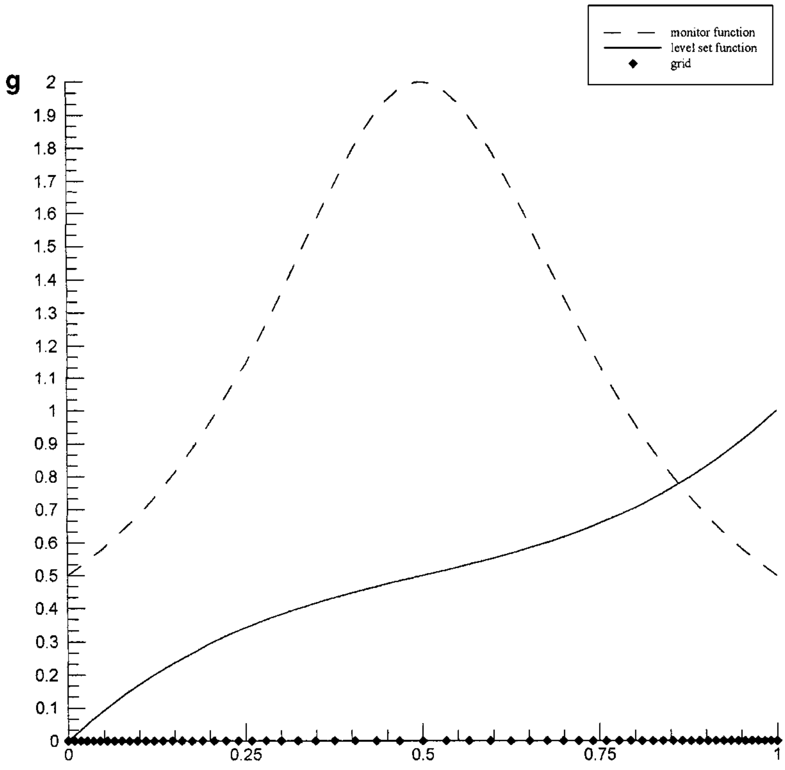


FIG. 2—*Continued*



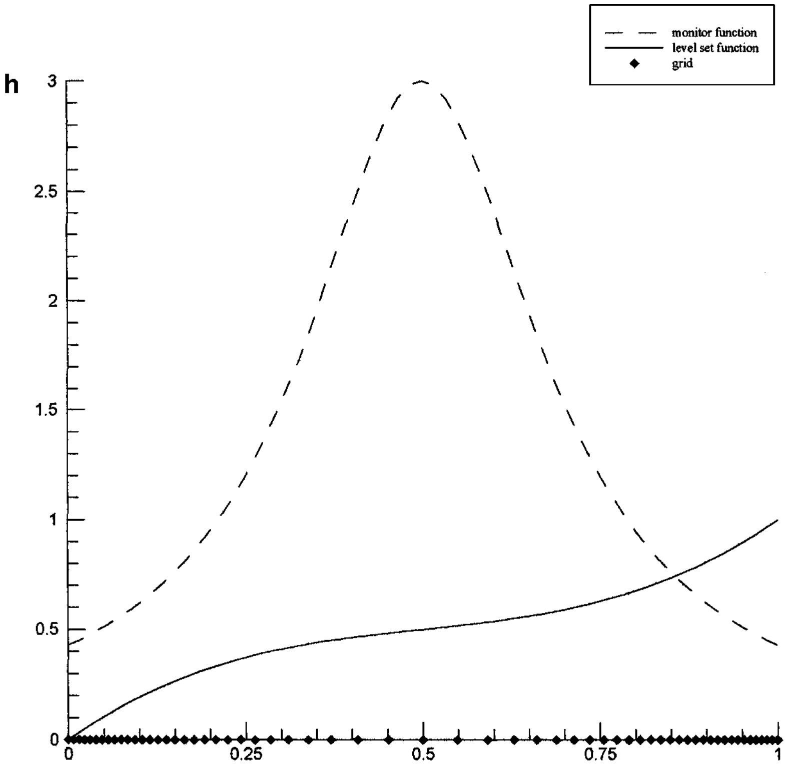


FIG. 2—*Continued*

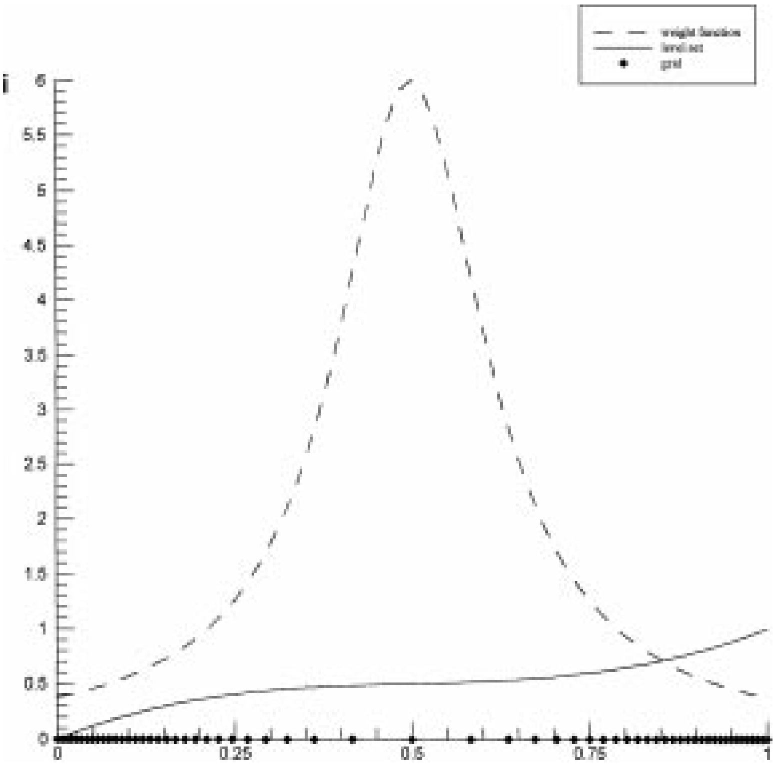


FIG. 2—*Continued*

deformation method deforms the initial uniform grid from $t = 0$ to $t = 0.5$ using the monitor function

$$f = \begin{cases} 1 - 2t + 2t(0.2 - 8d) & \text{if } -0.1 < d < 0 \\ 1 - 2t + 2t(0.2 + 8d) & \text{if } 0 < d < 0.1 \\ 1 & \text{if } |d| > 0.1. \end{cases} \tag{19}$$

Then the adaptive grid $t = 0.5$ continues to deform according to the monitor function

$$f = \begin{cases} 0.2 - 8d & \text{if } -0.1 < d < 0 \\ 0.2 + 8d & \text{if } 0 < d < 0.1 \\ 1 & \text{if } |d| > 0.1. \end{cases} \tag{20}$$

The time discretization of the evolution equations used a second-order TVD Runge–Kutta scheme and the spatial derivatives were approximated by a second-order ENO scheme (as in [31]). The second-order ENO scheme is also a TVD scheme. However, higher order ENO schemes do not have the property. The main reason we used an ENO scheme is that the functions ϕ and ψ must remain monotone. We note that the extra cost incurred by using this method rather than Lax–Wendroff, for example, is tiny compared to the overall cost of the algorithm. The elliptic solver used to compute the velocity field is by far the most expensive part of the algorithm. The Poisson equation was approximated using central differencing for both derivatives. The resulting system of linear algebraic equations was then solved with

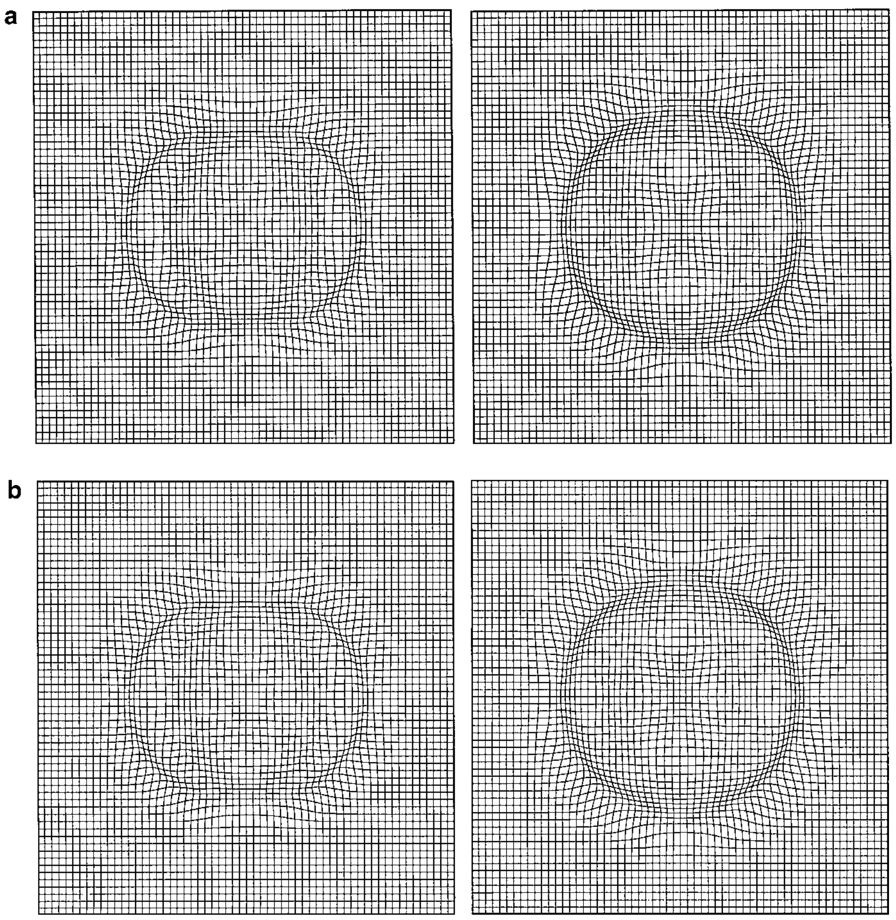


FIG. 3. Grid plots for Example 2.

the successive overelaxation (SOR) method, where the value of the relaxation constant was chosen as 1.3. The Neumann boundary conditions were implemented using ghost points. The new position of the nodes were obtained by the following scheme (see Fig. 4): Let $\Phi\colon (x, y) \to (\phi, \psi)$. Then by (11), we know that (t is dropped for simplicity of presentation)

$$J(\Phi^{-1}) = f(x(k_i, k_j), y(k_i, k_j)),$$

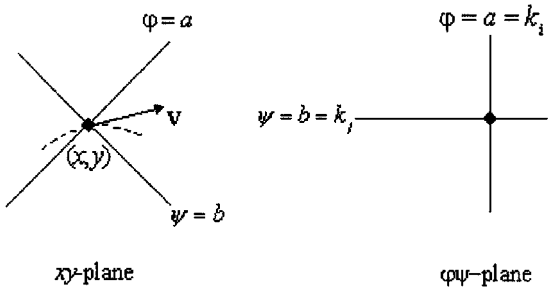


FIG. 4. Grid lines in the xy-plane.

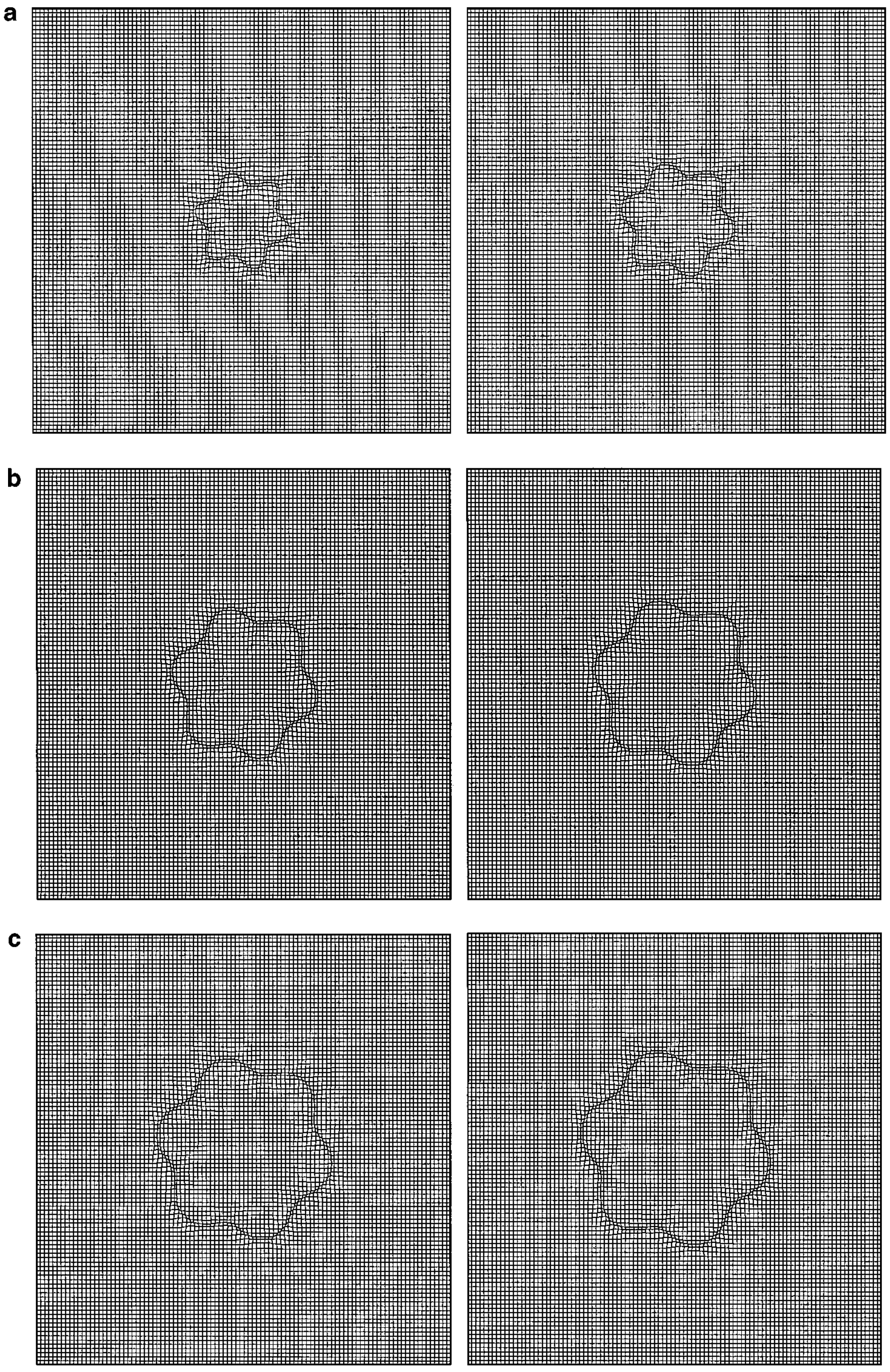
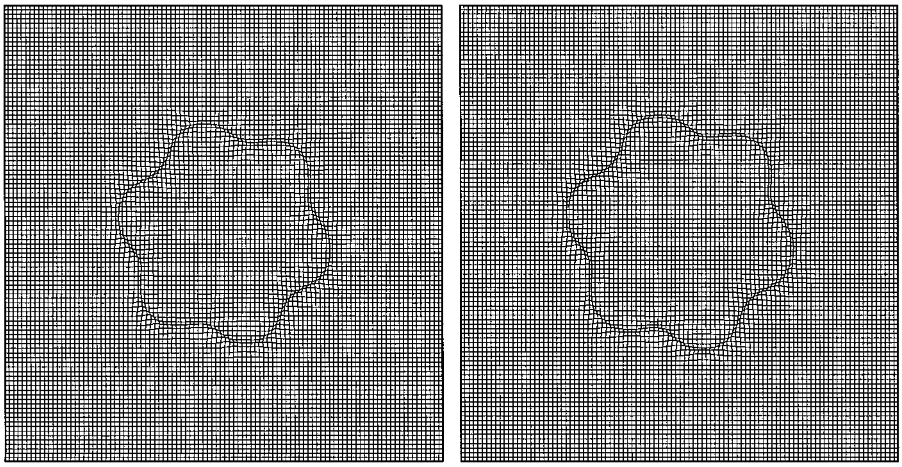


FIG. 5. Grid plots for Example 3.

d



e

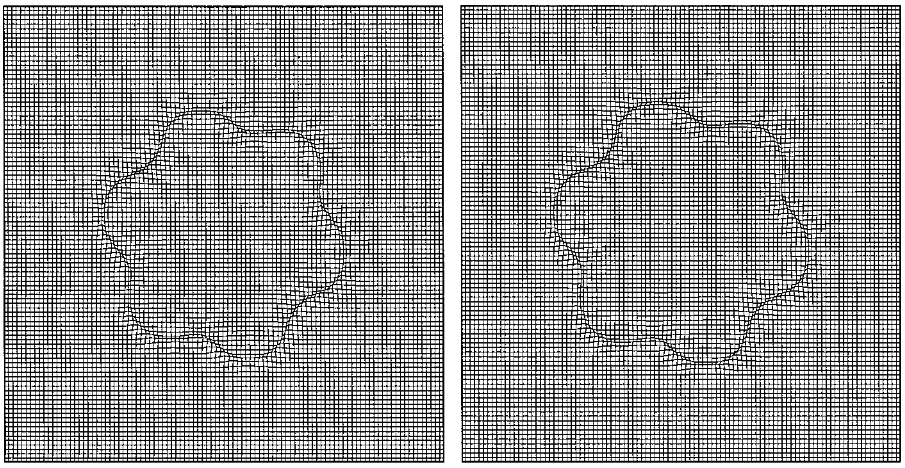


FIG. 5—*Continued*

where $(x(k_i, k_j), y(k_i, k_j))$ are the new node coordinates. Setting $\phi = k_i$ and $\psi = k_j$, then $(x(k_i, k_j), y(k_i, k_j)) = \Phi^{-1}(k_i, k_j)$. Thus the new nodal positions can be obtained by interpolation.

EXAMPLE 3 (Fig. 5). A 100×100 initial uniform grid deforms from $t = 0$ to $t = 0.5$ to a grid clustered around the interface of the solidification phenomenon modeled by the Stefan equation. The monitor function f is defined by

$$f = \begin{cases} 1 - 2t + 2t(0.2 - 4d) & \text{if } -0.2 < d < 0 \\ 1 - 2t + 2t(0.2 + 4d) & \text{if } 0 < d < 0.2 \\ 1 & \text{if } |d| > 0.2, \end{cases} \tag{21}$$

where d is proportional to the level set function ϕ, which is calculated by a level set method (see [26]), that is, $d = c\phi(x, y, t)$, where c is the adaptation constant (in the example $c = 10$). The vector field $\mathbf{v}$, the solutions to the level set evolution equation, and the new node positions were obtained as in example 2.

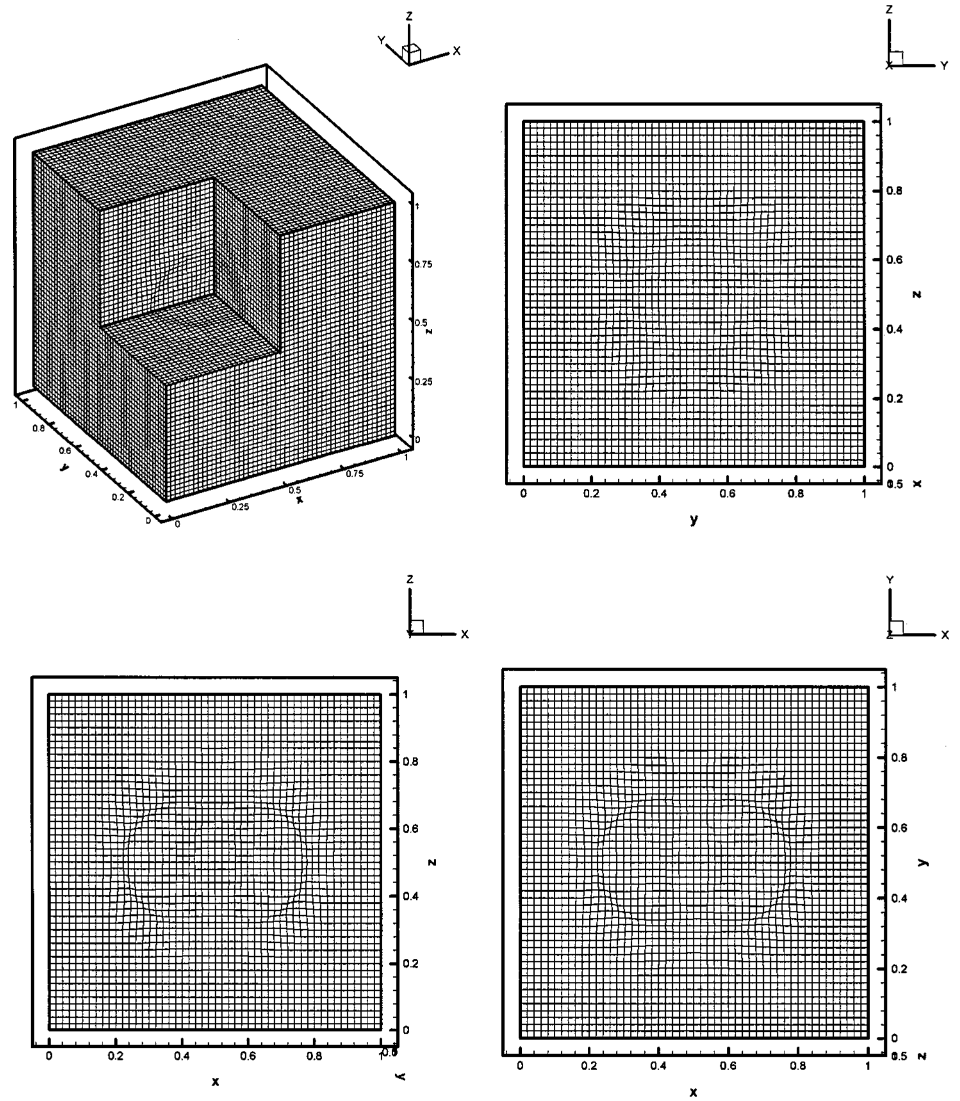


FIG. 6. Grid plots for Example 4.

Figure 5 shows the moving grid follows the interface closely.

EXAMPLE 4 (Fig. 6). A $50 \times 50 \times 50$ uniform grid on the unit cube in R^3 is deformed into a grid concentrated around a pair of spheres and the grid moves appropriately as the spheres merge into each other. The monitor function f is also defined by (21) with the function d

$$d = \left((x - a_1)^2 + (y - b_1)^2 - r^2\right)\left((x - a_2)^2 + (y - b_2)^2 - \rho^2\right)$$

where (a_1, b_1, c_1) is the (moving) center of the first sphere and (a_2, b_2, c_2) is the (moving) center of the second sphere. The zero set of d consists of the two spheres and r and ρ are their varying radii. Initially $(a_1, b_1, c_1) = (0.4, 0.5, 0.5)$, $(a_2, b_2, c_2) = (0.6, 0.5, 0.5)$, and $r = \rho = 0.18$.

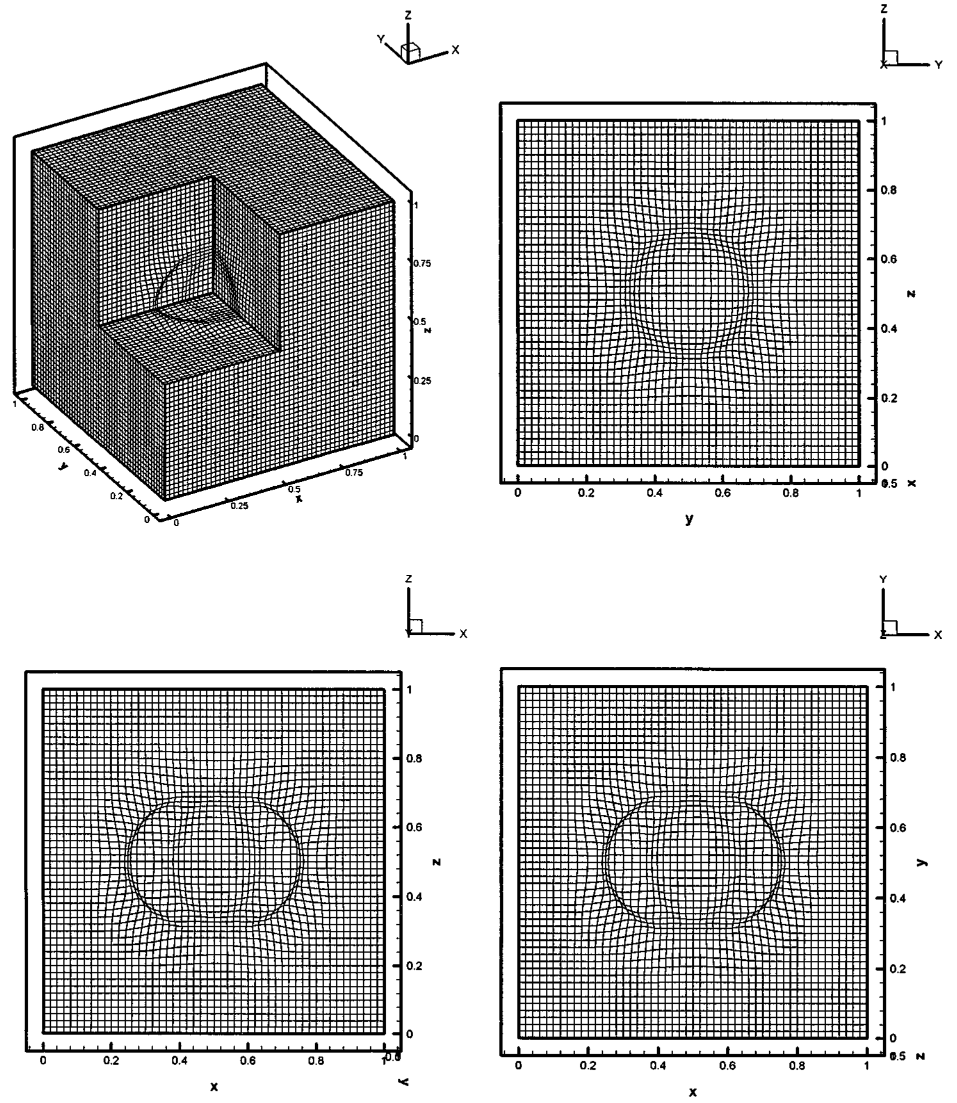


FIG. 6—*Continued*

2.2. *Application to Time-Dependent PDEs*

We describe the procedures of using our method for solving time-dependent PDE (1), which works for any dimensions. For simplicity, let us consider the dimension $n = 2$. A monitor function f is determined by the solution being calculated. Then we determine $\mathbf{v} = f\nabla w$, where w satisfies

$$\triangle w = -\left(\frac{1}{f}\right)_t,$$

with the Neumann boundary condition. Next, solve for ϕ and ψ from (10),

$$\begin{cases} \phi_t(x, y, t) + \langle \nabla\phi, \mathbf{v}\rangle = 0 \\ \psi_t(x, y, t) + \langle \nabla\psi, \mathbf{v}\rangle = 0, \end{cases} \tag{22}$$

with the same boundary and initial conditions.

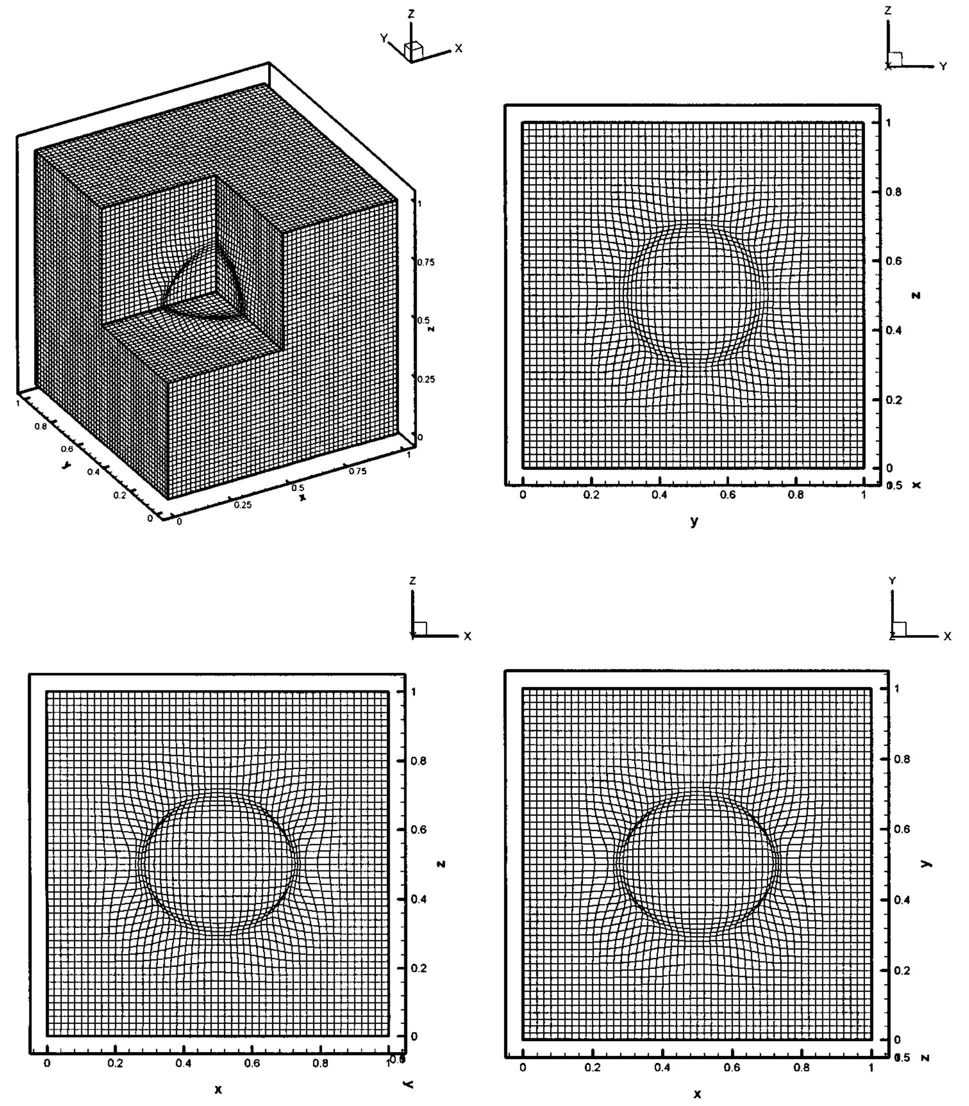


FIG. 6—*Continued*

Let $\Phi = (\phi, \psi)$. Put a uniform grid on the unit square $[0, 1] \times [0, 1]$ of the $\phi\psi$-plane. The node $\phi = a$, $\psi = b$ in the $\phi\psi$-plane has a preimage $(x(a, b, t), y(a, b, t))$ under Φ at each time t, which is the node of the moving grid in the xy-plane corresponding to (a, b) on the $\phi\psi$-plane (see Fig. 4). Consider

$$\begin{cases} \phi(x, y, t) = a \\ \psi(x, y, t) = b. \end{cases} \tag{23}$$

Differentiating (23) with respect to t, we get

$$\begin{cases} \phi_t(x, y, t) + \phi_x \dot{x} + \phi_y \dot{y} = 0 \\ \psi_t(x, y, t) + \psi_x \dot{x} + \psi_y \dot{y} = 0 \end{cases} \tag{24}$$

Comparing (24) with (22), we get

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \mathbf{v}.$$

Namely, the node velocity is equal to **v**.

For simplicity of presentation, let us consider the case where $u(x, y, t)$ is a scalar function. Let $U(\phi, \psi, t) = u(x(\phi, \psi, t), y(\phi, \psi, t), t)$. Then

$$U_t = u_x \dot{x} + u_y \dot{y} + u_t,$$

where $u_t = L(u)$ by (1), $(\dot{x}, \dot{y}) = \mathbf{v}$. The derivatives that are in $L(u)$, such as u_x, u_y, u_{xx}, u_{xy}, and u_{yy}, are transformed also. For instance, from

$$\begin{aligned} U_\phi &= u_x x_\phi + u_y y_\phi \\ U_\psi &= u_x x_\psi + u_y y_\psi, \end{aligned}$$

we can solve for u_x and u_y uniquely, since

$$\frac{1}{f} = \begin{vmatrix} x_\phi & y_\phi \\ x_\psi & y_\psi \end{vmatrix} > 0.$$

The higher derivatives can be obtained similarly. The transformed equation for $U(\phi, \psi, t)$ takes the form of

$$U_t = \tilde{L}(U), \tag{25}$$

where $\tilde{L}$ is a differential operator in ϕ and ψ.

Finally, (25) will be solved on a fixed uniform grid in the $\phi\psi$-plane.

3. CONCLUSION

A new moving grid method is formulated that is based on the standard evolution equation for level set functions. A suitable velocity vector field can be constructed from a positive monitor function by solving a scalar Poisson equation. The resulting moving grid has the desired cell size distribution specified by the monitor function at each time. Numerical examples in 1D, 2D, and 3D are given to demonstrate the method. Further numerical results for solving time-dependent PDEs based on Section 2.2 will be published in subsequent papers.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grants DMS-9732742, DMS-9706827, and DMS-9615854. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. The first author is grateful to the Department of Mathematics of UCLA for the hospitality he received during his visit in the fall of 1997.

REFERENCES

1. J. U. Brackbill and J. S. Saltzman, Adaptive zoning for singular problems in two dimensions, *J. Comput. Phys.* **46**, 342 (1982).
2. J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *Numerical Grid Generation* (1985).
3. Paul A. Zegeling, *Moving Grid Methods* (Utrecht Univ. Press, 1992).
4. P. Knupp and S. Steinberg, *The Fundamentals of Grid Generation* (CRC Press, Boca Raton, FL, 1994).
5. G. Carey, *Computational Grid Generation* (Taylor & Francis, London, 1997).
6. V. Liseikin, *Grid Generation Methods* (Springer-Verlag, Berlin/New York 1999).
7. J. Castillo, S. Steinberg, and P. J. Roache, Mathematical aspects of variational grid generation, *J. Comput. Appl. Math.* **20**, 127 (1987).
8. D. A. Anderson, Grid cell volume control with an adaptive grid generator, *Appl. Math. Comput.* **35**, 35 (1990).
9. K. Miller, Recent results on finite element methods with moving nodes, in *Accuracy Estimates and Adaptive Methods in Finite Element Computations*, edited by Babuska, Zienkiewicz, Gago, and Oliveira (Wiley, New York, 1986), p. 325.
10. D. Arney and J. Flaherty, *ACM Trans. Math. Software* **16**(1), 48 (1990).
11. D. F. Hawken, J. J. Gottlieb, and J. S. Hansen, Review of some adaptive node-movement techniques in finite-element and finite-Difference solutions of partial difference equations, *J. Comput. Phys.* **95**, 254 (1991).
12. W. Huang, Y. Ren, and R. Russell, Moving mesh methods based on moving mesh partial differential equations, *J. Comput. Phys.* **113**, 279 (1994).
13. J. Moser, Volume elements of a Riemann manifold, *Trans. AMS* **120**, 286 (1965).
14. B. Dacorogna and J. Moser, On a PDE involving the Jacobian determinant, *Ann. Inst. H. Poincaré* **7** (1990).
15. G. Liao and D. Anderson, A new approach to grid generation, *Applicable Anal.* **44**, 285 (1992).
16. G. Liao and J. Su, A moving grid method for $(1+1)$ dimension, *Appl. Math. Lett.* **8**, 47 (1995).
17. B. Semper and G. Liao, A moving grid finite element method using grid deformation, *Numer. Methods PDEs* **11**, 603 (1995).
18. P. Bochev, G. Liao, and G. dela Pena, Analysis and computation of adaptive moving grids by deformation, *Numer. Methods PDEs* **12**, 489 (1996).
19. F. Liu, S. Ji, and G. Liao, An adaptive grid method with cell-volume control and its applications to Euler flow calculations, *SIAM J. Sci. Comput.* **20**(3), 811 (1998).
20. F. Liu and A. Jameson, Multi-grid Navier–Stokes calculation for three-dimensional cascades, *AIAA J.* **31**(10), 1785 (1993).
21. F. Liu and Zheng, A strongly coupled time-marching method for solving the Navier–Stokes and turbulance model equations with multigrid, *J. Comput. Phys.* **128**, 289 (1996).
22. G. Liao, in *Proceedings of the 15th IAMCS World Congress*, Vol. 2, p. 155, Berlin, August 1997.
23. G. Liao and G. dela Pena, A moving grid finite difference algorithm for PDEs, preprint.
24. J. A. Sethian, Curvature flow and entropy conditions to grid generation, *J. Comput. Phys.* **115**, 440 (1994).
25. D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, A PDE based fast level set method, UCLA CAM Report 98-25 (1998).
26. S. Chen, B. Merriman, S. Osher, and P. Smereka, A simple level set method for solving Stefan problems, *J. Comput. Phys.* **134**, 236 (1997).
27. S. Osher and J. Sethian, Fronts propagating with curvature dependent speed: Algorithms based on Hamilton–Jacobi formulations, *J. Comput. Phys.* **79**, 12 (1988).
28. Chorin and Marsden, *A Mathematical Introduction to Fluid Dynamics,* 3rd ed. (Springer-Verlag, Berlin/New York, 1993).
29. R. Meyer, *Introduction to Mathematical Fluid Dynamics* (Wiley, New York 1971).
30. S. Osher and C. W. Shu, High order essentially non-oscillatory schemes for Hamilton–Jacobi equations, *SIAM J. Numer. Anal.* **28**, 907 (1991).
31. M. Sussman, P. Smereka, and S. Osher, A level set method for computing solutions to incompressible two-phase flow, *J. Comput. Phys.* **114**, 146 (1994).
32. B. Merriman, R. Caflisch and S. Osher, Level set methods with an application to modelling the growth of thin films, *Proc of 1997 Congress on Free Boundary Problems, Heraklion, Crete, Greece (1997)*, to appear.