STUDIA UNIV. "BABEŞ–BOLYAI", MATHEMATICA, Volume **XLIX**, Number 2, June 2004

A MOVING FINITE DIFFERENCE METHOD FOR PARTIAL DIFFERENTIAL EQUATIONS

GUOJUN LIAO, JIANZHONG SU, ZHONG LEI, GARY C. DE LA PENA, AND DALE ANDERSON

Abstract. A moving grid method which has its origin from differential geometry is studied. The method deforms an intial grid according to a vector field calculated by a Poisson equation. The forcing term of the Poisson equation is determined by the time derivative of a positive monitor function. It adapts the grid at each time step by keeping the volume of each cell proportional to the (normalized) time-dependent monitor function. A moving finite difference method is formulated which transforms a time dependent partial differential equation by the grid mapping and then simulates the transformed equation on a fixed orthogonal grid in the computational domain. The method is demonstrated by solving model problems and an incompressible flow problem.

1. Introduction

In numerical simulation of partial differential equations (PDEs), a well-constructed grid is required to yield satisfactory results. For general grid generation methods we refer to [2],[4],[5], and [6].

Fixed uniform or non-uniform grids have been widely used. In this approach the grid points are distributed in the physical domain prior to a simulation of the PDEs. The same grid point locations are then used throughout the computation. A drawback in using this technique happens when the solution to the PDE exhibits

Received by the editors: 26.05.2004.

2000 *Mathematics Subject Classification.* 65M(N)05, 65M(N)50.

Key words and phrases. adaptive grids, moving finite difference, deformation method.

Supported by NSF under Grant DMS-9732742.

large variation due to, for example, shock waves and boundary layers. Because of its static feature the grid is unable to effectively capture such variations.

An approach that can lead to improved accuracy and efficiency is the use of solution-adaptive grids. The idea is to generate the grids according to the salient features of the solution that is being calculated. The objective is to distribute more grid points in regions where the solution possess fine scale structures in order to improve accuracy, and fewer grid points in regions where small changes in the solution occur to improve efficiency.

There are two basic strategies for grid adaptation: local refinement and relocation of nodes. This paper develops a deformation method which moves the nodes according to the numerical solution as it is being computed. A key element of successful moving grid methods is the construction of a transformation $\phi : \Omega \times [0, T] \to \Omega$ which moves the nodes of an initial grid in accordance with the computed solution through a monitor function or an error estimator. To qualify as a transformation, ϕ must be one to one (injective) and onto (surjective). Variational methods [1], [3] and elliptic PDE methods [2] have been used to define this transformation.Various aspects of the grid such as orthogonality ("skewness"), smoothness, and cell size are adjusted through a linear combination of individual functionals. The resulting system of PDEs for grid generation are often nonlinear and require intensive computation. Results have also been reported in controlling the cell size through the Jacobian determinants of the transformation [7]. A moving finite element method has been developed by Miller [8] (see [10] for a survey). Recently, methods based on Moving Mesh Partial Differential Equations [11],[12] were developed with remarkable capability to track rapid spatial and temporal transitions for multiple dimensional problems. The special issue of Comput. Methods Appl. Mech. Engerg. edited by Kallinderis [28] contains a collection of papers on adaptive grid methods for compressible CFD. It is an excellent source in the topic covered by this study.

A common weakness of current moving grid methods is that they do not provide mathematical assurance that the "grid transformation ϕ" is indeed a transformation in three dimensional domains. The method formulated in this paper is

based on a deformation method by J. Moser and B. Dacorogna [13],[14] in the study of volume elements. This method was originally developed in the context of Riemannian geometry, and has recently been applied to numerical methodology [15]. It has the advantage of providing direct control over the cell size of the adaptive grid, and the transformation can be easily computed. The method inherently defines a transformation which is necessarily injective, thus ensuring in theory that the grid lines do not cross even in three dimensions. The approach is remarkably robust, in that it can be applied to time dependent differential equations in combination with any commonly used numerical methods (finite element, finite difference, finite volume, etc).

The paper is organized as follows. In Section 2, the mathematical formulation of the deformation is presented. A 3D numerical grid example is shown in Figure 1. In Section 3, a moving finite difference algorithm is formulated. The algorithm is used to solve several model problems. In Section 4, the incompressible Navier-Stokes equation is solved using this method. Finally in Section 5, we give conclusions and indicate further research directions.

2. **Adaptive Grid Generation by the Deformation Method**

2.1. **Mathematical Formulation.** The deformation method generates a time-dependent nodal mapping from a domain D_1 to another domain D_2. It assures direct volume control through the Jacobian determinant. The way the mapping is constructed assures the existence and uniqueness of the mesh. The vector field in which the nodes move is calculated by a scalar Poisson equation. Thus the mesh lines are quite smooth. Due to the fact that the vector field is irrotational, the cells maintain acceptable shapes during the calculation. The partial differential equations are transformed by the moving mesh mapping and solved on a fixed uniform mesh on a computational domain. For a general description of the method, consider the PDE for $u(\vec{x}, t)$,

$$u_t(\vec{x}, t) = L(u) \tag{1}$$

where u is a scalar or vector variable, L is a differential operator in $\vec{x}$ only, on a physical domain Ω_p in R^d, $d = 1, 2, 3$, for $t > 0$. Suppose that the solution to (1) has

been computed at time step $t = t_{n-1}$, and a preliminary computation has been done at time level $t = t_n$. Assume that we are provided with some positive error estimator (or gradient approximation) $\delta(\vec{x}, t)$. Define the monitor function

$$f(\vec{x}, t) = \frac{C}{\delta(\vec{x}, t)} \tag{2}$$

where C is a positive scaling parameter so that at each time step we have

$$\int_\Omega \left(\frac{1}{f(\vec{x}, t_n)} - 1 \right) dA = 0. \tag{3}$$

Note that f is small in regions of large error and becomes larger in regions where the error is small. Let Ω_c denote the computation domain and Ω_p denote the physical domain. The deformation method constructs a transformation $\phi : \Omega_c \to \Omega_p$ such that

$$\det \nabla\phi(\vec{\xi}, t) = f(\phi(\vec{\xi}, t), t), \qquad t_{n-1} \le t \le t_n, \tag{4}$$

$$\phi(\vec{\xi}, t_{n-1}) = \phi_{n-1}(\vec{\xi}), \qquad \vec{\xi} \text{ on } \Omega_c,$$

where $\vec{\xi}$ is a node of an initial grid, $\phi_{n-1}(\vec{\xi})$ represents the coordinates of the node at $t = t_{n-1}$. We require that $\phi(\vec{\xi}) \in \partial\Omega$, for all $\vec{\xi} \in \partial\Omega_p$. Note that (4) ensures the size of the transformed cells will conform to the function f, i.e. the grid will be appropriately "refined" in regions of large error and "coarsened" in regions of small error.

A steady version of this method has been applied to two-dimensional steady flow problems [19]. The moving grid method has been applied to one-dimensional time-dependent problems [17]. The computation of ϕ consists of two steps:

The first step is to find a vector field $\vec{u}(\vec{\xi}, t)$ satisfying

$$\operatorname{div} \vec{u}(\vec{\xi}, t) = -\frac{\partial}{\partial t} \left(\frac{1}{f(\vec{\xi}, t)} \right), \qquad \vec{\xi} \in \Omega_c,\ t_{n-1} \le t \le t_n \tag{5}$$

$$\frac{\partial \vec{v}}{\partial \mathbf{n}} = 0, \quad \vec{\xi} \in \partial\Omega,\ \mathbf{n} = \text{outward normal to } \partial\Omega_c.$$

The vector field $\vec{v}$ can be found by solving for w from the scalar Poisson equation (for a fixed t)

$$\triangle w(\vec{\xi}, t) = -\frac{\partial}{\partial t} \left(\frac{1}{f(\vec{\xi}, t)} \right), \qquad \vec{\xi} \in \Omega \tag{6}$$

$$\frac{\partial w}{\partial \mathbf{n}} = 0, \qquad \vec{\xi} \in \partial\Omega_c,$$

then setting $\vec{u} = \nabla w$.

The second step is to solve for the new location $\phi(\vec{\xi}, t)$ at time t of any node $\vec{\xi} \in \overline{\Omega}$ of the grid at $t = 0$ from the ODE system

$$\frac{\partial}{\partial t}\phi(\vec{\xi}, t) = \vec{(v)}(\phi(\vec{\xi}, t), t), \qquad t_{n-1} \le t \le t_n,\ \vec{\xi} \in \Omega, \tag{7}$$

$$\phi(\vec{\xi}, t_{n-1}) = \phi_{n-1}(\vec{\xi}),$$

where the node velocity $\vec{(v)}(\vec{\xi}, t) = f(\vec{\xi}, t)\vec{u}(\vec{\xi}, t)$. The mathematical foundation of the method is provided by the following

Theorem. [17] $\det \nabla\phi(\vec{\xi}, t) = f(\phi(\vec{\xi}, t), t)$ for each $\vec{\xi}$ in D each $t > 0$.

The Jacobian determinant of a mapping $\phi(\vec{\xi}, t)$ from D_1 to D_2 in R^n, $n =$ 1, 2, 3, is

$$J(\phi) = \det \nabla\phi = \frac{|dA'|}{|dA|}, \tag{8}$$

where dA' is the image of a volume (area, in 2D) element dA. In our case, $J(\phi)$ $= f(\phi,t) > 0$ since the monitor function f is chosen to be positive. Thus, the theorem assures precise control over the cell size relative to that of the fixed initial grid in both 2D and 3D.

The theorem is proved by showing $\frac{d}{dt}(J(\phi)g(\phi, t)) = 0$ and therefore $Jg = 1$ since $Jg\Big|_{t=0} = 1$. For the convenience of the reader, we include an outline of the proof given in [17].

Proof. Let $g(\vec{\xi}, t) = \frac{1}{f(\vec{\xi},t)}$ ($\Rightarrow g(\phi, t) = \frac{1}{f(\phi,t)}$) and $\vec{u}$ be a vector field satisfying

$$\operatorname{div} \vec{u}(\vec{\xi}, t) = -\frac{\partial}{\partial t} g(\vec{\xi}, t) (\Rightarrow \operatorname{div}_\phi \vec{u}(\phi, t) = -\frac{\partial}{\partial t} g(\phi, t).) \tag{9}$$

Let $\eta = \vec{u}f$ (i.e. $\vec{(v)}(\vec{\xi}, t) = \vec{u}(\vec{\xi}, t)f(\vec{\xi}, t) \Rightarrow \vec{(v)}(\phi, t) = \vec{u}(\phi, t)f(\phi, t)$). Let $J = J(\phi(x, t)) = \det \nabla\phi(\vec{\xi}, t)$, be the Jacobian determinant. First,

$$\frac{d}{dt}\nabla_{\vec{\xi}}\phi(\vec{\xi}, t) = \nabla_{\vec{\xi}}(\frac{d}{dt}\phi(\vec{\xi}, t)) = \nabla_\phi(\vec{(v)}(\phi, t))(\nabla_{\vec{\xi}}\phi). \tag{10}$$

Using the formula: $\frac{d}{dt}M(t) = A(t)M(t) \Rightarrow \frac{d}{dt}(\det\ M(t)) = (\mathrm{Tr}\ A(t))(\det\ M(t))$, we get

$$\frac{dJ}{dt} = J(\mathrm{div}_\phi\, \vec{(v)}(\phi,t)) = J(f\mathrm{div}_\phi\, \vec{u}(\phi,t) + \langle \vec{u}, \nabla_\phi f \rangle).$$

by (9),

$$J^{-1}\frac{dJ}{dt} = -f\frac{\partial g}{\partial t} + \langle \vec{u}, \nabla_\phi f \rangle. \tag{11}$$

Second,

$$\begin{aligned}
\frac{d}{dt}(J(\phi)g(\phi,t)) &= g\frac{dJ}{dt} + J\frac{d}{dt}g(\phi,t) \\
&= g\frac{dJ}{dt} + J(\Big\langle \nabla_\phi g, \frac{d\phi}{dt} \Big\rangle + \frac{\partial}{\partial t}g(\phi,t)) \\
&= g\frac{dJ}{dt} + J(\Big\langle \nabla_\phi g, f(\phi,t)\vec{u}(\phi,t) \Big\rangle + \frac{\partial}{\partial t}g(\phi,t)).
\end{aligned}$$

Third, by (11) and since $fg = 1$ implies $g\nabla f + f\nabla g = 0$

$$\frac{1}{J}\frac{d}{dt}(Jg) = \Big[-gf\frac{\partial g}{\partial t} + \Big\langle \vec{u}, g\nabla_\phi f \Big\rangle\Big] + \Big[\Big\langle f\nabla_\phi g, \vec{u} \Big\rangle + \frac{\partial}{\partial t}g\Big] = 0, \tag{12}$$

implies that

$$\Rightarrow \frac{d}{dt}(Jg) = 0 \Rightarrow Jg = 1 \text{ for each } t > 0, \text{since } Jg\Big|_{t=0} = 1.$$

Thus

$$J(\phi) = f(\phi,t), \quad t > 0.\S$$

Consequently, ϕ is indeed injective (non-folding). Also by choosing $f > 0$ to be continuous (or smooth), the Jacobian determinant can be made to change continuously (or smoothly), which is important in obtaining high accuracy in the computation. The deformation method does not have direct control of the orthogonality of the grid lines on the physical domain. The fact that the vector field $\vec{u}$ is irrotational, i.e. $\vec{u} = \nabla w$, and thus curl $\vec{u} = 0$, helps to prevent excessive skewness in the grid. In this paper we transform equation (1) through the grid mapping $\vec{x} = \vec{x}(\vec{\xi}, t)$ and solving the transformed equation on an orthogonal grid on the $\vec{\xi}$-domain. This approach enables us to have the benefits of the adaptive grids as well as the advantages of using a fixed orthogonal grid.

The monitor function f is constructed during the solution process. This establishes the dynamic coupling of the mesh movement with the PDE solver through the transformed equation, the Poisson equation, and the deformation ODEs.

Construction of a proper monitor function is a challenging task (see [29]). A common way to construct the monitor function is the equidistribution principle. A posteriori error estimates (if available), residues, and truncation errors, etc. are to be redistributed evenly over the whole domain. In many cases, truncation errors are difficult to compute. Thus, in engineering calculation, gradient of the solution is often used to detect the regions where refined grids are needed. For instance, in the calculation of Euler flows with shock waves [19], we used

$$f = \frac{C_1}{1 + C_2|\nabla p|} \tag{13}$$

where p is the pressure, C_2 is a constant for adaptation intensity, C_1 is a normalization parameter. For viscous flows one can use the Mach number M in place of the pressure. In general, in addition to the gradient of the unknown variable z, terms involving the value of z and the second derivatives of (or curvatures) can also be included. For instance,

$$f = \frac{k}{1 + \alpha|z| + \beta|\nabla z| + \gamma\,|\nabla^2 z|} \tag{14}$$

where $\alpha, \beta,$ and γ are parameters controlling the intensity of adaptation.

For interface resolution, f can be constructed using a signed distance function d as follows: Let f be piecewise linear such that

$$f = \begin{cases} 1 & \text{if } |d| > \varepsilon \\ 0.2 & \text{if } d = 0 \end{cases} \tag{15}$$

f is then normalized so that $\int_\Omega (\frac{1}{f} - 1) = 0$, which is required for (4) to be satisfied. We will discuss the issue in more details below.

In [26] a different version of the deformation method is formulated. It is based on the level set evolution equation and the transport formula in fluid dynamics (see, i.e., [30]). It assures the same direct control over the Jacobian determinant.

2.2. **Numerical Examples.** We present numerical examples of the deformation implemented using a finite-difference method. In [18], this method was implemented using a least-squares finite-element method. Two main tasks are required to numerically implement this method:

1. Solve a Poisson equation on D

$$w_{\xi\xi} + w_{\eta\eta} = -\frac{\partial}{\partial t}\Big(\frac{1}{f(\xi,\eta,t)}\Big), \qquad (\xi,\eta) \in \Omega_c \tag{16}$$

$$\frac{\partial w}{\partial \vec{n}} = 0 \quad (\xi,\eta) \in \Gamma, \quad \text{where } \vec{n} \text{ is the outward normal,}$$

2. Solve a system of ordinary differential equations (the deformation ODEs)

$$\frac{\partial}{\partial t}\phi(\xi,\eta,t) = f(\phi(\xi,\eta,t),t)\vec{u}(\phi(\xi,\eta,t),t)\,,\ t_{k-1} \le t \le t_k,\ (\xi,\eta) \in \Omega_c \tag{17}$$

$$\phi(\xi,\eta,t_{k-1}) = \phi_{k-1}(\xi,\eta).$$

Let $w_{i,j} = w(i\Delta\xi, j\Delta\eta)$, for $i = 1,\ldots,m$, $j = 1\ldots,n,$ where $\Delta\xi = \frac{1}{m}$, $\Delta\eta = \frac{1}{n}$. For simplicity we assume $\Delta\xi = \Delta\eta = h$ and $m = n$.

A monitor, function f, is formed at time $t = t_k$, $k \ge 1$. The monitor function f is normalized to satisfy (3). To accomplish this, let $\overline{f}$ denote the non-normalized monitor function and f denote the normalized monitor function at $t = t_k$. Then

$$f = C \cdot \overline{f} \text{where} \quad C = \int_D \frac{1}{\overline{f}(\xi,\eta,t)}\, dA. \tag{18}$$

Now form $-\frac{\partial}{\partial t}(\frac{1}{f}),$ the right-hand side of (16), using the normalized monitor function. The time derivative is approximated on each node in a uniform grid by

$$g_t(\xi,\eta) = -\left(\frac{\frac{1}{f(\xi,\eta,t_k)} - \frac{1}{f(\xi,\eta,t_{k-1})}}{\tau_k}\right), \quad \tau_k = t_k - t_{k-1} \tag{19}$$

The node velocity can then be found by solving (17). In this study (16) is approximated using central difference approximations for both derivatives to obtain

$$\frac{w_{i-1,j} - 2w_{i,j} + w_{i+1,j}}{h^2} + \frac{w_{i,j-1} - 2w_{i,j} + w_{i,j+1}}{h^2} = (\frac{\partial}{\partial t}\frac{1}{f})_{i,j} \tag{20}$$

The resulting system of linear algebraic equations is then solved with the successive over-relaxation (SOR) method implemented in the following two steps

$$\overline{w}_{i,j} = \frac{1}{4}(w^{\text{new}}_{i-1,j} + w^{\text{new}}_{i,j-1} + w^{\text{old}}_{i,j+1} + w^{\text{old}}_{i+1,j} - h^2 \frac{\partial}{\partial t}\frac{1}{f_{i,j}}) - w^{\text{old}}_{i,j} \tag{21}$$

$$w^{\text{new}}_{i,j} = w^{\text{old}}_{i,j} + \lambda\, \overline{w}_{i,j} \tag{22}$$

On the boundary, Neumann conditions are implemented with a consistent second-order central-difference scheme. This is done by introducing a "ghost point" outside of the region and approximating the boundary condition with

$$\frac{1}{2h}(w_{i,1} - w_{i,-1}) = 0 \text{ and } \frac{1}{2h}(w_{i,n-1} - w_{i,n+1}) = 0 \tag{23}$$

$i = 1, \ldots, n-1,$ for the lower and upper boundaries respectively, and

$$\frac{1}{2h}(w_{1,j} - w_{-1,j}) = 0 \text{ and } \frac{1}{2h}(w_{n-1,j} - w_{n+1,j}) = 0 \tag{24}$$

$j = 1, \ldots, n-1,$ for the left and right boundaries, respectively.

To initialize the SOR iterations at each time step, the approximations at the previous time step can be used. Finally, the vector field $\vec{u}$ and hence the nodal velocities were computed by setting $\vec{u} = \nabla w$ using second-order centered differences. The nodes are then moved by solving the deformation ODEs (17) by a Runge-Kutta method.

Example 1. A three dimensional uniform grid in a unit cube of 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 definition of the monitor function is based on the following level set function d:

$$d = ((x-a_1)^2 + (y-b_1)^2 + (z-c_1)^2 - r^2)((x-a_2)^2 + (y-b_2)^2 + (z-c_2)^2 - \rho^2)$$

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 with varying radii r and ρ. In this example, $r = \rho = constant$. The spheres initially intersect each other and gradually merge to one sphere. The deformation method deforms an

initial uniform grid to a grid adapted to a pair of intersecting circles at $t = 0$ with the monitor function

$$\begin{cases} 1 & d \le y \le d - 0.1 \\ 0.3 - 7t(y-d) + (1-t) & d - 0.1 < y \le d \\ 0.3 + 7t(y-d) + (1-t) & d < y \le d + 0.1 \\ 1 & d + 0.1 < y \le 2.0 \end{cases}. \tag{25}$$

The deformation method then continues to deform the adaptive grid for $1 < t \le 2$ according to the following monitor function

$$\begin{cases} 1 & 0 \le y \le d - 0.1 \\ 0.3 - 7t(y-d) & d - 0.1 < y \le d \\ 0.3 + 7t(y-d) & d < y \le d + 0.1 \\ 1 & d + 0.1 < y \le 2.0 \end{cases}. \tag{26}$$

In this example, the time step is $\Delta t = 0.025$. The grids are shown in Figure 1.

3. **A Moving Grid Finite Difference Method**

A concern over various moving grid methods is the lack of orthogonality of the grid generated. Indeed, while finite element and finite volume methods can be implemented on non-orthogonal grids, finite difference methods usually are implemented on an orthogonal grid. In this section we describe a common moving grid strategy ([2]) that will be used to implement the deformation method on an orthogonal computational grid.

Let us consider the time dependent equation for a variable z with proper boundary and initial conditions

$$z_t\,(\vec{x}, t) = L(z), \tag{27}$$

where$L$ is a differential operator in $\vec{x}$, only, on a domain Ω_p in R^d, $d = 1, 2, 3$, for $t > 0$. Suppose a positive monitor function $f(\vec{x}, t)$ has been constructed according to the solution that is being calculated. By the deformation method we construct a transformation $\phi : x^l = \phi^l(\vec{\xi}, t)$, $l = 1, 2, 3$, from a cubic computational domain Ω_c into Ω_p at each t such that $J(\phi)(\vec{\xi}, t) = f(\phi(\vec{\xi}, t), t)$. Let $n = 3$ and let $\{\xi_{ijk} \,|\, i, j, k =$

$0, 1, 2, ..., N\}$ be the nodes of a uniform grid on Ω_c. Then the image $x_{ijk}(t) = \phi(\xi_{ijk}, t)$ of a node ξ_{ijk} is the new node of the moving grid at time t. We now transform (27) to the computational domain Ω_c with the coordinates $\vec{\xi} = (\xi^1, \xi^2, \xi^3)$.

Let $Z(\vec{\xi}, t) = z(\phi(\vec{\xi}, t), t))$. By the Chain Rule, we have,

$$\frac{\partial Z}{\partial t} = \frac{\partial z}{\partial x^i}\frac{\partial \phi^i}{\partial t} + \frac{\partial z}{\partial t}. \tag{28}$$

Thus (27), becomes

$$\frac{\partial Z}{\partial t} = \frac{\partial z}{\partial x^i}\frac{\partial \phi^i}{\partial t} + L(z). \tag{29}$$

Note: The components of the node velocity $\partial\phi^i/\partial t$, $i = 1, 2, 3$ are directly determined by equation (7). In fact one of the advantages of the deformation method is the determination of the node velocity $\partial\phi/\partial t$ by the desired monitor function f through an explicit formula.

To transform the terms in $L(z)$, we use the formulas in [2]. By the deformation method (see Theorem, 2.1) that

$$J(\phi) = f(\phi, t) > 0.$$

Thus, the transformation formulas are valid.

To demonstrate the method, let us consider the hyperbolic PDE for $d = 2$:

$$\frac{\partial z}{\partial t} + a(x, y, t)\frac{\partial z}{\partial x} + b(x, y, t)\frac{\partial z}{\partial y} = 0 \tag{30}$$

on the unit square $\Omega = [0, 1] \times [0, 1]$. We begin with a uniform grid on the computational domain Ω_c, with coordinates $(\xi^1, \xi^2) = (\xi, \eta)$. We are seeking a transformation $\phi : \Omega_c \to \Omega_p$ in the form of

$$x = x(\xi, \eta, t),\ y = y(\xi, \eta, t)$$

such that the cell size of the moving grid on Ω is evenly distributed according to a positive monitor function $f(\xi, \eta, t)$. Let $Z(\xi, \eta, t) = z(x(\xi, \eta, t), y(\xi, \eta, t), t)$. By (28), (30) becomes

$$\frac{\partial Z}{\partial t} - \frac{\partial z}{\partial x}\frac{\partial x}{\partial t} - \frac{\partial z}{\partial y}\frac{\partial y}{\partial t} + a(x, y, t)\frac{\partial z}{\partial x} + b(x, y, t)\frac{\partial z}{\partial y} = 0. \tag{31}$$

Differentiating, we have

$$Z_\xi = z_x \frac{\partial x}{\partial \xi} + z_y \frac{\partial y}{\partial \xi}, \tag{32}$$

$$Z_\eta = z_x \frac{\partial x}{\partial \eta} + z_y \frac{\partial y}{\partial \eta}. \tag{33}$$

Since $J = \det \nabla\phi = f > 0$. By Cramer's Rule, we can find z_x and z_y from (32) and (33) and get

$$z_x = \frac{1}{J}\left(Z_\xi \frac{\partial y}{\partial \eta} - Z_\eta \frac{\partial y}{\partial \xi}\right) \tag{34}$$

$$z_y = \frac{1}{J}\left(Z_\eta \frac{\partial x}{\partial \xi} - Z_\xi \frac{\partial x}{\partial \eta}\right) \tag{35}$$

where $J = x_\xi\, y_\eta - y_\xi\, x_\eta$. Let $A(\xi,\eta,t) = a(x((\xi,\eta,t), y(\xi,\eta,t), t)$ and $B(\xi,\eta,t) = b(x(\xi,\eta,t), y(\xi,\eta,t), t)$. Substituting (34) and (35) into (31) and rearranging terms, (31) becomes

$$\frac{\partial Z}{\partial t} + \mathcal{A} Z_\xi + \mathcal{B} Z_\eta = 0 \tag{36}$$

where

$$\mathcal{A} = \frac{1}{J}\left[\left(A - \frac{\partial x}{\partial t}\right)\frac{\partial y}{\partial \eta} - \left(B - \frac{\partial y}{\partial t}\right)\frac{\partial x}{\partial \eta}\right] \tag{37}$$

$$\mathcal{B} = \frac{1}{J}\left[\left(B - \frac{\partial y}{\partial t}\right)\frac{\partial x}{\partial \xi} - \left(A - \frac{\partial x}{\partial t}\right)\frac{\partial y}{\partial \xi}\right] \tag{38}$$

In general, since the Jacobian determinant $J(\phi)$ is positive, the equation is invariant. That is, the transformed equation remains elliptic, parabolic, or hyperbolic depending on the type of the original equation 27.

An algorithm implementing the moving finite-difference method will now be formulated. The algorithm consists of an initialization procedure and a time integration loop. The initialization procedure is an iteration procedure for determining the vector field $\vec{v}$ at $t = 0$. The procedure is needed since the node velocity $\vec{v} = (x_t, y_t)$ at $t = 0$ in the transformed equation can not be determined from the initial value z_0 alone. For the subsequent time steps, either an explicit or implicit scheme can be used.

The numerical algorithm in 2D is as follows (extension to 3D is straightforward):

I. Initialization ($t = 0$)

1. Construct a uniform grid in the logical domain (ξ, η) in Ω_c.
2. Use initial values $z_0 = z(x, y, 0)$ to construct the monitor function f at $t = 0$.
3. Use the static version of the deformation method to generate an initial adaptive grid on (x, y) in Ω_p determined by the initial monitor function
$$f(\xi, \eta, 0) = \frac{C_1}{1 + C_2|\nabla z_0|}.$$
Set $g(s) = \frac{C_1}{1+ s\ C_2|\nabla z_0|}$ and deform the uniform grid on Ω_p according to $g(s)$ in artificial time s from $s = 0$ to $s = 1$.
4. Let $\vec{v}^{(\mathbf{0})}|_{t=0}$ be the vector field $\vec{v}|_{s=1}$ of the static deformation method. Derive the transformed equation at $t = 0$, setting $(x_t,\ y_t) = f|_{t=0}\, \vec{v}^{(\mathbf{0})}|_{t=0}$.
5. Solve the transformed PDE to obtain $U|_{t=\frac{1}{2}\Delta t}$.
6. Use $Z|_{t=0.5\Delta t}$ to construct the monitor function $f|_{t=0.5\Delta t}$.
7. Use values of the monitor function at $t = 0$ and $t = 0.5\Delta t$ to construct the source term of the Poisson equation at $t = 0$:
$$w_{\xi\xi} + w_{\eta\eta} = -\left(\frac{\frac{1}{f|_{t=0.5\Delta t}} - \frac{1}{f|_{t=0}}}{0.5\Delta t}\right)$$
8. Set $\vec{u}^{(\mathbf{1})}|_{t=0} = \nabla w$ at $t = 0$.
9. Compute the transformed equation again at $t = 0$, setting $\vec{v}^{(\mathbf{1})}|_{t=0} = f|_{t=0}\, \vec{u}^{(\mathbf{1})}|_{t=0}$.
If
$$\left|\vec{v}^{(\mathbf{1})}|_{t=0} - \vec{v}^{(\mathbf{0})}_{t=0}\right| < \epsilon,$$
where ϵ is a preset tolerance, stop; Otherwise, repeat the above procedures until
$$\left|\vec{v}^{(\mathbf{k+1})}|_{t=0} - \vec{v}^{(\mathbf{k})}|_{t=0}\right| < \epsilon.$$

10. Define the transformed equation at $t = 0$ by setting $\vec{v} = f|_{t=0}\, \vec{u}^{(\mathbf{k+1})}|_{t=0}$.

II. Time Integration ($t = \Delta t, 2\Delta t,\ 3\Delta t,\ 4\Delta t, ...$)

1. Solve the above PDE and get $Z(\xi, \eta, \Delta t)$.
2. Use $Z(\xi,\ \eta, \Delta t)$ to form the monitor function $f|_{t=\Delta t}$.
3. Use $f|_{t=\Delta t}$ and $f|_{t=0}$ to form the right hand side of the Poisson equation at $t = \Delta t$:
$$w_{\xi\xi} + w_{\eta\eta} = -\left(\frac{\frac{1}{f|_{t=dt}} - \frac{1}{f|_{t=0}}}{\Delta t}\right)$$
4. Set $\vec{u}|_{t=\Delta t} = \nabla w$;
5. Compute the transformed equation at $t = \Delta t$, setting $(x_t, y_t) = f|_{t=\Delta t}\, \vec{u}|_{t=\Delta t}$.
6. Repeat the above procedures for $t = 2\Delta t, 3\Delta t, 4\Delta t, ...$

The moving grid finite-difference scheme is used to solve the following two-dimensional model equations.

Model Problem I (Weiss Model). Consider the hyperbolic initial-boundary-value problem

$$z_t \ = \ -\sin 2\pi x\, \cos 2\pi y\, u_x \ + \sin 2\pi y\, \cos 2\pi x\, u_y \quad x, y \in [0,1] \times [0,1], t > 0 \tag{39}$$

$$z(x, y, 0) = \ 1 - x \quad 0 \le x, y \le 1, \tag{40}$$

$$\begin{cases} z(0, y, t) = 1,\ z(1, y, t) = 0, & y \in [0, 1],\ t > 0 \\ z(x, 1, t) = z(x, 0, t) = 1 - x, & x \in [0, 1],\ t > 0. \end{cases} \tag{41}$$

Since the equation does not have exact solution, we take the numerical solution on a fine uniform grid with $200X200$ nodes as a satisfactory approximation.

The contour plot of the approximate solution is shown in Figure 2. The contour plot exhibits boundary layers at $x = 0, x = 1, y = 0$, and $y = 1$, as well as interior layers at $x = 0.5$. Our task is to generate a moving grid with significantly less nodes which can resolve these layers.

In Figure 3, we showed the moving grid with $100X100$ nodes and in Figure 4 the solution contour plot. The initial grid is uniform. As can be seen by comparison

to Figure 2, the layers are resolved well except at the two small regions near $(0.5, 0)$ and $(0.5, 1)$, where boundary and interior layers meet. Clearly, more resolution there is needed.

To provide the needed resolution, an initial grid that is refined near $y = 0$ and $y = 1$ is used. The results are shown in Figure 5 and 6. The contour plot appears to be comparable to that in Figure 2.

This example shows that

1. Grid resolution is a key factor for calculation involving fine structures, as we all agree;

2. Construction of a proper monitor function is a challenging task. It is an active research area (see, i.e., [29]) and some interesting new ideas appear to be very promising. We will explore some of the ideas in subsequent study.

3. Assuming a proper monitor function is formed by the user, the deformation method does generate real-time moving grid with specified cell size distribution as the theory predicts. The term "real-time" means that the grid is updated in one time step as the PDE is being solved. The time t in the grid generation equations is the same as that in the PDE. In particular, $\vec{v}$ is the actual node velocity.

Model Problem II (Whirlpool Problem). Consider the hyperbolic problem

$$z_t = -\frac{v_r}{v_{r_{max}}}\frac{y}{r}u_x + \frac{v_r}{v_{r_{max}}}\frac{x}{r}u_y \quad x, y \in [0,1]\times[0,1], t > 0 \tag{42}$$

where

$$r = \sqrt{x^2 + y^2 + \varepsilon} \quad v_r = \frac{\tanh(r)}{\cosh^2(r)} \text{ and } v_{r_{max}} = 0.385. \tag{43}$$

$$\tag{44}$$

$$\begin{cases} z(x, y, 0) = -\tanh(\frac{y}{2}) \quad 0 \le x, y \le 1, \\ \frac{\partial z}{\partial n} = 0 \end{cases} \tag{45}$$

The moving grid and contour plot of the solution are shown in Figure 7, 8. The solution exhibits strong rotation. In contrast to the Lagrange method, our grid nodes do not rotate with the "flow". Instead, they move properly to form cells that are small in the regions where the rotation is strong. The cell shapes remain acceptable

due to the fact that the vector $\vec{u}$ is irrotational. In the cavity flow example to be studied below, these characteristics will be seen also.

In the above model problems, the monitor functions are based on the gradient of the variable z. Prior to solving the equation for $t > 0$, an initial grid is generated using the static mode of the deformation method. The initial grid is adapted to the initial condition by using the following monitor function:

$$f_{\text{initial}} = \frac{C_1}{1 + s\, C_2 |\nabla z_0|} \tag{46}$$

where s is an artificial time that goes from 0 to 1. Ten time steps in the steady mode are used to generate the initial grid. The initial could be taken as an uniform grid as well. If prior information about the solution is known, an adaptive grid could be generated and used as the initial grid.

To solve the transformed equation, the time discretization is accomplished by using a second-order Runge-Kutta scheme given by

$$\tilde{Z}_{i,j}^{n+1} = Z_{i,j}^{n} + \Delta t\, \mathcal{L}(Z_{i,j}^{n}) \tag{47}$$

$$Z_{i,j}^{n+1} = \frac{1}{2} Z_{i,j}^{n} + \frac{1}{2} \tilde{Z}_{i,j}^{n+1} - \frac{1}{2} \Delta t\, \mathcal{L}(\tilde{Z}_{i,j}^{n+1}) \tag{48}$$

Here, $\mathcal{L}$ is the discrete approximation to the differential operator L. A second-order essentially non-oscillatory (ENO) method is used to approximate the spatial derivatives of Z as described in [24]. The transformation parameters x_ξ, x_η, y_ξ, y_η are discretized using central differencing and the time derivatives x_t, y_t are interpolated from the computational grid. To approximate the boundary conditions the first-order linear extrapolation scheme given by

$$\begin{cases} Z_{i,0} = 2Z_{i,1} - Z_{i,2} \\ Z_{0,j} = 2Z_{1,j} - Z_{2,j}. \end{cases} \tag{49}$$

is used. The grids are generated using the monitor function f given by

$$f(\xi_i, \eta_j, t) = \frac{C_1}{1 + C_2 |\nabla z|}, \tag{50}$$

where ∇z is transformed and calculated at the uniform nodes (ξ_i, η_j) at each time t. The constant C_2 is the same as in the initial grid stage. The forcing term of the

Poisson equation is approximated using the difference scheme

$$\frac{\partial}{\partial t}\Big(\frac{1}{f}\Big)^n_{i,j} = \frac{3(\frac{1}{f})^n_{i,j} - 4(\frac{1}{f})^{n-1}_{i,j} + (\frac{1}{f})^{n-2}_{i,j}}{2\Delta t}. \tag{51}$$

4. **Navier-Stokes Equation**

For an incompressible flow, the governing equations can be written in conservative form as

$$\frac{\partial u_i}{\partial t} + \frac{1}{J}\frac{\partial}{\partial \xi_j}\left[J\left(U_j - V_j\right)u_i\right] = -\frac{1}{J}\frac{\partial}{\partial \xi_j}\left(J\frac{\partial \xi_j}{\partial x_i}p\right) + \frac{1}{J}\frac{1}{Re}\frac{\partial}{\partial \xi_j}\left(Jq_{jk}\frac{\partial u_i}{\partial \xi_k}\right), \tag{52}$$

$$\frac{\partial \xi_k}{\partial x_i}\frac{\partial}{\partial \xi_k}\left[\frac{1}{J}\frac{\partial}{\partial \xi_j}\left(J\frac{\partial \xi_j}{\partial x_i}p\right)\right] = -\frac{1}{J}\frac{\partial D}{\partial t} + \frac{1}{J}\frac{\partial}{\partial \xi_k}\frac{\partial \xi_k}{\partial x_i}\frac{\partial}{\partial \xi_j}\left[J\left(U_j - V_j\right)u_i\right]. \tag{53}$$

where in the pressure Poisson equation (PPE) (53), the viscous terms have been removed by using the continuity equation. U_i is the contravariant velocity and V_i is determined by the node velocity $\vec{x}_t$ of the moving grid:

$$U_i = \frac{\partial \xi_i}{\partial x_j}u_j,\ V_i = \frac{\partial \xi_i}{\partial t} = \frac{\partial \xi_i}{\partial t} = \frac{\partial x_j}{\partial t}\frac{\partial \xi_i}{\partial x_j}. \tag{54}$$

The Jacobian and metrics are defined as

$$J = \frac{\partial(x_1, x_2, x_3)}{\partial(\xi_1, \xi_2, \xi_3)}, q_{jk} = \frac{\partial \xi_j}{\partial x_i}\frac{\partial \xi_k}{\partial x_i}. \tag{55}$$

The deformation method uses a scalar Poisson to get the node velocity vector field and then move the nodes by ODEs. This is in contrast to the elliptic grid generators, which determines node position directly. The flow in a square cavity whose top wall is driven by a lid with uniform velocity has served as a model problem for testing and evaluating numerical techniques. This is a typical complex flow in a simple geometry with a strong vortex near the center and two secondary vortexes. Our task is to generate a moving grid which resolves the vortexes as the computation proceeds. The construction of a suitable monitor function is an important aspect of adaptive algorithms.In this paper, instead of searching for the best possible monitor function, we use a simple and effective monitor function based on the stream function ψ to demonstrate the method. More precisely, in order to resolve the main as well as the secondary vortexes, we use the product of ψ and $\psi - \psi_{\min}$. Also, instead of using the best flow solver, we used a reliable flow solver written by the second author. The

present study compares results on moving grids to the benchmark solutions found in [25].

The marker-and-cell (MAC) method is used with collocate grid arrangement in this study. The convective terms are discretized using the QUICK scheme to remove numerical wiggles, while the viscous terms are discretized by central difference scheme. The second-order Adams-Bashforth method is used for time integration to solve the momentum equation (52). Both the pressure Poisson equation (PPE) (53) and the deformation Poisson equation (6) are discretized by central differences and then solved with the ADI method. An explicit 2nd-order Runge-Kutta method is used in solving equation (7). The usual technique: geometric conservation law (GCL) for correcting the errors caused by moving grids is not used in this study. Our experience is that the correction is unnecessary for the viscous flow with low Reynolds number as is the case of the cavity flow. It was necessary for inviscid flows or viscous flows with high Reynolds number. For instance, it was used in [27] for the cylindrical implosion problem.

The monitor function in this study is based on $d = \psi(\psi_{\min} - \psi)$. An initial adapted grid is generated at the artificial time $s = 1$ according to the monitor function $f_{\text{init}} = (1 - s) + sf$ where f is given by

$$f = \begin{cases} 1 & \text{if } |d| > 0.004 \\ 0.344 - 164d & \text{if -0.004} \le d \le 0 \\ 0.344 + 164d & \text{if } 0 \le d \le 0.004 \end{cases} . \tag{56}$$

After the initial time, a moving grid is formed using the same f. The moving grid and the stream function contour at the steady state are shown in Figure 9 and 10. The comparisons on stream function and vorticity are given in Table 1. The values presented are nodal values.

TABLE 1. Comparison to Benchmark Solution, Re = 1000

	Uniform Grid (129 × 129)	Moving Grid (51 × 51)
Primary Vortex	Benchmark [25]	Moving Grid MAC
$\psi_{\min}$	-0.117929	-0.117967
$\omega_{\text{v.c}}$	2.04968	2.19837
Location (x, y)	$(0.5313, 0.5625)$	$(0.5228, 0.5589)$

5. **Conclusion**

We have formulated a moving grid finite difference approach for time dependent PDEs. We have also established rigorously that the method does not lead to tangled grids in three dimensions. Computational experiments indicate that the moving grid method is robust and efficient, and that it can put more nodes in the regions where the need for higher resolution exists. The method is not as fast as on fixed grid (with the same amount of nodes) due to the fact that it requires solving a scalar Poisson equation on a uniform grid at each time step. The method is more efficient when compared to other PDE based grid generators that solve non-linear PDE systems.

A moving finite difference algorithm is presented, which transforms the host partial differential equations via the grid mapping. The transformed equations then are simulated on an orthogonal computational grid. The method is demonstrated by model problems and the Navier-Stokes problem. The calculations showed that the method is capable of significantly enhance resolution where and when it is needed. On the other hand, an additional Poisson equation is solved at each time step, and smaller timestep may be necessary on fine grids. Thus, the method is expected to be used only for solving large, complex PDE systems for which the extra efforts for solving the additional Poisson equation is insignificant and some local resolution enhancement is necessary.

Works are underway to develop methods for adaptive multiple block grids and unstructured meshes for unsteady partial differential equations in 2D and 3D general domains.

References

[1] Brackbill, J. U. and Saltzman, J. S. *Adaptive Zoning for Singular Problems in Two Dimensions*, J. Comput. Phys., 46(1982).

[2] Thompson, J. F., Warsi, Z. U. A. and Mastin, C. W., *Numerical Grid Generation*, North-Holland, Amsterdam, 1985.

[3] Castillo, J., Steinberg, S. and Roache, P. J., *Mathematical Aspects of Variational Grid Generation*, J. Comput. Appl. Math., 20(1987).

[4] Zegeling, Paul A., *Moving Grid Methods*, Dissertation, The Utrecht University Press, 1992.

[5] Knupp, P. and Steinberg, S., *The Fundamentals of Grid Generation,* CRC Press, Boca Raton, 1993.

[6] Carey, G., *Computational Grid Generation, Adaptation and Solution Strategies*, Taylor and Francis, Washington D.C., 1997.

[7] Anderson, D. A., *Grid Cell Volume Control with an Adaptive Grid Generator*, Appl. Math. and Comput., 35(1990).

[8] Miller, K., *Recent Results on Finite Element Methods with Moving Nodes*, in Accuracy, Estimates and Adaptive Methods in Finite Element Computations, Babuska, Zienkiewicz, Gago, and Oliveira,eds., John Wiley & Sons, 1986.

[9] Arney, D. and Flaherty, J., *An Adaptive Mesh-moving and Local Refinement Method for Time-dependent Partial Differential Equations*, ACM Transactions on Math software, Vol. 16, No. 1, 1900, 48-71.

[10] Hawken, A. et.al., *Review of Some Adaptive Node Movement Techniques in Finite Element and Finite difference Solutions of Partial Difference Equations*, J. Comput. Phys., 95(1991).

[11] Huang, W., Ren, Y. and R. Russell, *Moving Mesh Methods Based on Moving Mesh Partial, Differential Equations*, J. of Comput. Phys., 113(1994).

[12] Huang, W., Ren, Y. and Russell, R., *Moving Mesh Partial Differential Equations (MMPDES) Based on the Equidistribution Principle*, SIAM J. Numer. Anal., 31(1994).

[13] Moser, J., *Volume Elements of a Riemann Manifold*, Trans AMS, 120(1965).

[14] Dacorogna, B. and Moser, J., *On a PDE Involving the Jacobian Determinant*, Ann. Inst. H. Poincare, 7(1990).

[15] Liao, G. and Anderson, D., *A New Approach to Grid Generation*, Appl. Anal., 44(1992).

[16] Liao, G. and Su, J., *A Moving Grid Method for (1+1) Dimension*, Appl. Math. Lett., 8(1995).

[17] Semper, B. and Liao, G., *A Moving Grid Finite Element Method using Grid Deformation*, Numer. Meth. PDEs, 11(1995).

[18] Bochev, P., Liao, G. and dela Pena, G., *Analysis and Computation of Adaptive Moving Grids by Deformation*, Numer. Meth. PDEs, 12(1996).

[19] Liu, F., Ji, S. and Liao, G., *An Adaptive Grid Method with Cell-Volume Control and its Applications to Euler Flow Calculations*, SIAM J. Sci. Comput., 20(1998).

[20] Liao, G. and Su, J., *A Direct Method in Dacorogna-Moser's Approach of Grid Generation Problems*, Appl. Anal., 49(1993).

[21] Liao, G., Pan, T. and Su, J., *A Numerical Grid Generator Based on Moser's Deformation Method*, Numer. Meth. PDEs, 10(1994).

[22] Strikwerda, J. C., *Finite Difference Schemes and PDEs*, Wadswasth & Brooks/Cole, 1989.

[23] Chen, S., Merriman, B., Osher, S. and Smereka, P., *A Simple Level Set Methods for Solving Stefan Problems*, J. of Comput. Phys., 135(1997).

[24] Shu, C. W. and Osher, S., *Efficient Implementation of Essentially Non-oscillatory Shock Capturing Schemes*, J. of Comput. Phys., 77(1988).

[25] Ghia, U., Ghia, K. N. and Shin, C. T., *High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method*, J. Comput. Phys., 48(1982).

[26] Liao, G., F. Liu, G. dela Pena, D. Peng, and S. Osher, *Level-Set-Based Deformation Methods for Adaptive Grids*, J. Comput. Phys., 159(2000), 103-122.

[27] Liao, G., Lei, Z. and dela Pena, G., *Adaptive grids for resolution enhancement*, Shock Waves, An International Journal on Shock Waves, Detonations and Explosions by Springer, 12(2002), 153-156.

[28] Kallinderis, K. (ed.), *Special Issue: Adaptive Methods for Compressible CFD*, Comput. Methods Appl. Mech. Engerg., 189(2000).

[29] Soni, B., Koomullil, R., Thompson, D. and Thornburg, H., *Solution adaptive strategies based on point redistribution*, Comput. Methods Appl. Mech. Engerg., 189(2000), 1183-1204.

[30] Chorin, A. and Marsden, J., *A Mathematical Introduction to Fluid Mechanics*, Springer-Verlag, 1993.

Figure 1.1. (Clockwise from top-left) Grid plots for Example 1 for $t = 1.0$. Cutaway plot, grid slice at $x = 0.5$, grid slice at $z = 0.5$, grid slice at $y = 0.5$.

Figure 1.2. (Clockwise from top-left) Grid plots for example 1 for $t = 5.5$. Cutaway plot, grid slice at $x = 0.5$, grid slice at $y = 0.5$, grid slice at $z = 0.5$.

Figure 2. Weiss Model: Contour Plot. Fixed uniform grid. $200X200$.

Figure 3. Weiss Model: Moving grid with uniform initial grid $100X100$.

Figure 4. Weiss Model: Contour Plot. Moving grid with uniform initial grid. $100X100$.

Figure 5. Weiss Model: Moving grid with adapted initial grid. $100X100$.

Figure 6. Weiss Model: Contour Plot. Moving grid with adapted initial grid. $100X100$.

Figure 7. Wrpool Model: Moving grid with uniform initial grid. $100X100$.

Figure 8. Wrpool Model: Contour Plot. Moving grid with uniform initial grid. $100X100$.

Figure 9. Steady Cavity Flow: Moving grid at steady state. $50X50$.

Figure 10. Steady Cavity Flow: Contour Plot. Moving grid. $50X50$.

Department of Mathematics, University of Texas,
Arlington, Texas 76019

Department of Mathematics, University of Texas,
Arlington, Texas 76019

VINAS Co., Ltd., Osaka, Japan 550-0002

Department of Mathematics, University of Rhode Island,
Kingston, RI 02881

Department of Mechanical and Aerospace Engineering,
University of Texas, Arlington, Texas 76019