SIAM J. SCI. COMPUT.
Vol. 24, No. 1, pp. 118–142

A MOVING MESH METHOD BASED ON THE GEOMETRIC CONSERVATION LAW*

WEIMING CAO†, WEIZHANG HUANG‡, AND ROBERT D. RUSSELL§

Abstract. A new adaptive mesh movement strategy is presented, which, unlike many existing moving mesh methods, targets the mesh velocities rather than the mesh coordinates. The mesh velocities are determined in a least squares framework by using the geometric conservation law, specifying a form for the Jacobian determinant of the coordinate transformation defining the mesh, and employing a curl condition. By relating the Jacobian to a monitor function, one is able to directly control the mesh concentration. The geometric conservation law, an identity satisfied by any nonsingular coordinate transformation, is an important tool which has been used for many years in the engineering community to develop cell-volume-preserving finite-volume schemes. It is used here to transform the algebraic expression specifying the Jacobian into an equivalent differential relation which is the key formula for the new mesh movement strategy. It is shown that the resulting method bears a close relation with the Lagrangian method. Advantages of the new approach include the ease of controlling the cell volumes (and therefore mesh adaption) and a theoretical guarantee for existence and nonsingularity of the coordinate transformation. It is shown that the method may suffer from the mesh skewness, a consequence resulting from its close relation with the Lagrangian method. Numerical results are presented to demonstrate various features of the new method.

Key words. moving mesh method, geometric conservation law, mesh adaption, mesh movement

AMS subject classifications. 65M50, 65M60

PII. S1064827501384925

1. Introduction.

The critical importance of mesh adaption in the numerical solution of partial differential equations (PDEs) has been amply demonstrated in the past. One of the most promising directions in the field of adaptivity is the development of reliable moving mesh strategies. In other work [13, 14, 15], we have investigated a class of so-called moving mesh PDE (MMPDE) methods, whereby the mesh movement is driven by an MMPDE which is solved in conjunction with the direct physical PDE. Since the MMPDE is formulated as the gradient flow equation of a quadratic functional and is parabolic, the mesh locations are more or less globally distributed, and consequently, the mesh smoothness and skewness can be well controlled.

Another class of moving mesh methods which have generated considerable interest of late bear close relation to the Lagrangian method. For them, the PDE to solve for the mesh velocity arises either from minimizing a functional involving the mesh velocity or from physical considerations. While it is not always clear how a mesh which drifts away from the desired one will be corrected at a later time for such methods, numerical results clearly demonstrate that they can be quite successful. Examples of methods in this class include the moving finite element (MFE) method of Miller and Miller [21, 22], the deformation map method advocated by Liao and Anderson [18]

*Received by the editors February 12, 2001; accepted for publication (in revised form) June 20, 2001; published electronically May 20, 2002. This work was supported in part by the Faculty Research Award of the University of Texas at San Antonio under account 14-7510-01, NSF (USA) grant DMS-0074240, and NSERC (Canada) grant OGP-0008781.

http://www.siam.org/journals/sisc/24-1/38492.html

†Division of Mathematics and Statistics, University of Texas at San Antonio, San Antonio, TX 78249 (wcao@math.utsa.edu).

‡Department of Mathematics, University of Kansas, Lawrence, KS 66045 (huang@math.ukans.edu).

§Department of Mathematics, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (rdr@cs.sfu.ca).

and Semper and Liao [23], and the arbitrary Lagrangian–Eulerian (ALE) method first proposed by Hirt, Amsden, and Cook for the solution of fluid dynamic problems [11].

The objective of this paper is to introduce a new moving mesh method which is in the spirit of this second class of methods. It is based upon the specification for the Jacobian of the coordinate transformation for mesh adaption and a use of the so-called geometric conservation law (GCL) and a curl condition. To be specific, consider the problem of adaptive mesh selection which arises in the context of solving a time-dependent PDE over a physical domain $\Omega \subset \Re^n,\ n = 1, 2$, or 3. To this end, a time-dependent coordinate transformation $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$ from the computational domain Ω_c to the physical domain Ω is required, and adaptive meshes are generated as images of a reference grid in Ω_c under the transformation. A simple and straightforward approach for determining the mesh transformation $\boldsymbol{x}(\boldsymbol{\xi}, t)$ is to specify its Jacobian J (the determinant of $(\partial \boldsymbol{x})/(\partial \boldsymbol{\xi})$). This idea for mesh adaption has been used by many researchers, e.g., see [3, 6, 16, 17, 18]. While specifying the Jacobian is appealing in a number of aspects, the differential equations obtained directly are not necessarily easy to solve, nor is mesh nonsingularity guaranteed [6].

The GCL is a tool which has been used for many years in the engineering community to develop cell-volume-preserving finite-volume schemes. For example, Trulio and Trigger [25] use the GCL, or "space conservation law," to eliminate numerical oscillations and preserve physical conservation laws for solutions on moving meshes. Thomas and Lombard [24] rediscover the GCL. Writing the physical PDEs in conservative form, they update the Jacobian at a new time based on the GCL, and accurate results are obtained for implicit finite-difference and finite-volume solution of unsteady Navier–Stokes equations and steady supersonic flow equations. Interestingly, instead of updating the cell volume at a new time level, Demirdžić and Perić [8, 9] modify the mesh speeds using the GCL in their finite-volume simulation of fluid flow problems with moving boundaries.

Here, we combine using the GCL, specifying the Jacobian of $\boldsymbol{x}(\boldsymbol{\xi}, t)$, and using a curl condition on $\boldsymbol{x}(\boldsymbol{\xi}, t)$ to produce an adaptive mesh movement strategy. More precisely, we first choose the Jacobian as a function (referred to as the monitor function) which involves the physical solution. We then specify the divergence of the mesh velocity using the GCL. Finally, the mesh velocity field is uniquely determined by further imposing a condition on its curl. Thus, our approach is to combine the ideas of adapting the mesh by specifying the Jacobian and controlling the mesh speed using the GCL and a curl condition. The theoretical underpinning of the approach is the Helmholtz theorem, which states that a smooth vector field can be decomposed into an orthogonal sum of divergence and curl terms.

The method which we develop is closely related to the deformation map method used by Liao and coworkers [2, 18, 23], and in fact one particular case is precisely that method. However, our approach has some distinct advantages. The motivation and derivation are very different, and the relation between it and several other moving mesh methods, including the Lagrangian method, is clearer and more easily understood. The resulting method is more general and allows for a greater diversity of implementations. In our limited experience a different implementation than that used by Liao is found to be more reliable. Fortuitously, the proof of local nonsingularity of the coordinate transformation follows straightforwardly from the GCL.

An outline of the paper is as follows. In section 2 we give a simple derivation of the GCL and a description of its general use for mesh adaption. A moving mesh strategy is then developed in section 3 based on the GCL, and its relation to and comparison

with some existing methods, including the Lagrangian method, the deformation map method, Miller's MFE, and the MMPDE method, are discussed in section 4. Section 5 is devoted to several implementations of the moving mesh strategy. In section 6 we present some numerical results obtained with these implementations for a selection of two-dimensional examples. Finally, section 7 contains conclusions and comments.

2. The geometric conservation law and mesh adaption. Let A_c be an arbitrary, fixed cell in the computational domain Ω_c enclosed by a smooth boundary ∂A_c, and let $A(t) = \{\boldsymbol{x} | \boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)\ \forall \boldsymbol{\xi} \in A_c\}$ be the corresponding cell in the physical domain Ω under the time-dependent coordinate transformation $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$. Then the change in volume of $A(t)$ equals the total flux through the surface $\partial A(t)$, i.e.,

$$\frac{d}{dt}\int_{A(t)} d\boldsymbol{x} = \int_{\partial A(t)} \boldsymbol{x}_t \cdot d\boldsymbol{S}, \tag{1}$$

where $\boldsymbol{x}_t$ is the mesh velocity. This is the integral form of the GCL [24]. Using the change of variables defined by the coordinate transformation, the left-hand side of (1) can be rewritten as

$$\frac{d}{dt}\int_{A(t)} d\boldsymbol{x} = \frac{d}{dt}\int_{A_c} J(\boldsymbol{\xi}, t) d\boldsymbol{\xi} = \int_{A_c} \frac{D}{Dt} J(\boldsymbol{\xi}, t) d\boldsymbol{\xi},$$

where $(D)/(Dt)$ denotes the time derivative in the coordinate system $(\boldsymbol{\xi}, t)$. In the context of fluid dynamics, $(D)/(Dt)$ is also called the total or material time derivative, which is related to derivatives with respect to the physical variables $(\boldsymbol{x}, t)$ by

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \boldsymbol{x}_t \cdot \nabla,$$

where ∇ is the gradient operator with respect to $\boldsymbol{x}$ and $(\partial)/(\partial t)$ is the time derivative when $\boldsymbol{x}$ is fixed.

From the divergence theorem, the right-hand side of (1) can be written as

$$\int_{\partial A(t)} \boldsymbol{x}_t \cdot d\boldsymbol{S} = \int_{A(t)} \nabla \cdot \boldsymbol{x}_t \, d\boldsymbol{x} = \int_{A_c} (\nabla \cdot \boldsymbol{x}_t) J \, d\boldsymbol{\xi}.$$

Noting that A_c is arbitrary, we obtain the differential form of the GCL

$$\nabla \cdot \boldsymbol{x}_t = \frac{1}{J}\frac{DJ}{Dt}, \tag{2}$$

which will be instrumental in deriving our moving mesh methods.

A simple and straightforward way to use the Jacobian for mesh adaption is to set

$$J(\boldsymbol{\xi}, t) = \frac{c(\boldsymbol{\xi})}{\rho(\boldsymbol{x}(\boldsymbol{\xi}, t), t)}, \tag{3}$$

where $\rho = \rho(\boldsymbol{x}, t) > 0$ is a user-defined monitor function, the size of which reflects the local difficulty in approximating the solution of the underlying problem, and $c = c(\boldsymbol{\xi})$ is a time-independent function determined by the initial coordinate transformation. In one dimension (3) is precisely the well-known equidistribution principle [13], so it can be viewed as a generalization of the principle.

A direct use of (3) for mesh adaption appears to be impractical [6] because it leads to a highly nonlinear, difficult-to-solve system of differential equations. Instead, we use the GCL to transform the algebraic expression (3) for J into an equivalent differential relation. Specifically, substituting (3) into (2) gives

$$\nabla \cdot \boldsymbol{x}_t = -\frac{1}{\rho}\frac{D\rho}{Dt},$$

or

$$\rho \nabla \cdot \boldsymbol{x}_t = -\frac{\partial \rho}{\partial t} - \boldsymbol{x}_t \cdot \nabla \rho,$$

and finally,

$$\nabla \cdot (\rho \boldsymbol{x}_t) + \frac{\partial \rho}{\partial t} = 0. \tag{4}$$

We note that (4) is mathematically equivalent to (3) since (2) is an identity satisfied by any nonsingular coordinate transformation.

Their equivalence implies that using (4) we can define the coordinate transformation and realize mesh adaption using the mesh velocity field. (In this way, we will be able to formulate a well-posed PDE governing the mesh movement, as we see in the next section.) Moreover, since the monitor function is always chosen to be strictly positive, the equivalence guarantees that the coordinate transformation determined by (4) is locally nonsingular for all $t > 0$ if and only if it is initially for $t = 0$.

The computational coordinate does not appear explicitly in (4), so the equation can be considered as one defined in the physical domain, with $\boldsymbol{x}_t$ regarded as a vector field in Ω. Additional conditions are necessary to guarantee there is a unique solution.

Assuming that the boundary points are not permitted to move out of the domain, the boundary condition

$$\boldsymbol{x}_t \cdot \boldsymbol{n} = 0 \quad \text{ on } \partial\Omega$$

holds. Assuming further that the physical domain Ω does not change with time, applying the divergence theorem to (4) we get the compatibility condition

$$\frac{d}{dt}\int_{\Omega} \rho(\boldsymbol{x}, t) d\boldsymbol{x} = 0. \tag{5}$$

Equivalently, (5) follows from (3) since

$$\int_{\Omega} \rho \, d\boldsymbol{x} = \int_{\Omega_c} \rho J \, d\boldsymbol{\xi} = \int_{\Omega_c} c(\boldsymbol{\xi}) d\boldsymbol{\xi}$$

is constant. One can view (5) as a conservation of mass condition, with ρ being a density function.

3. A moving mesh method based on the GCL. We are now in a position to derive the moving mesh strategy based on (4). First, for a given $\rho(\boldsymbol{x}, t)$, solving (4) is insufficient to uniquely determine the mesh velocity because it specifies only the divergence of the vector field. To see what is lacking, we recall the classical decomposition theorem of Helmholtz: A continuous and differentiable vector field can be resolved into the orthogonal sum of gradient of a scalar field ("solenoidal field")

and the curl of a vector field ("vortex field"). From this theorem, we conclude that in addition to determining the divergence of the mesh velocity field through (4), its curl may also be specified. Therefore, we let

$$\nabla \times w(\boldsymbol{v} - \boldsymbol{u}) = 0 \quad \text{in } \Omega, \tag{6}$$

where w and $\boldsymbol{u}$ are, respectively, a weight function and a background velocity field to be specified. The choice of these functions will be addressed in section 5. Hereafter, to avoid confusion we use $\boldsymbol{v}$ to denote the mesh velocity field and $\boldsymbol{x}_t$ to denote the time derivative of the coordinate transformation $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$.

Equation (6) implies that there exists a potential function ϕ such that

$$\boldsymbol{v} = \boldsymbol{u} + \frac{1}{w}\nabla\phi. \tag{7}$$

Substituting (7) into (4) and the boundary condition

$$\boldsymbol{v} \cdot \boldsymbol{n} = 0 \qquad \text{on } \partial\Omega \tag{8}$$

give rise to an elliptic system for ϕ,

$$\begin{cases} \nabla \cdot \left(\frac{\rho}{w}\nabla\phi\right) = -\frac{\partial \rho}{\partial t} - \nabla \cdot (\rho \boldsymbol{u}) & \text{in } \Omega, \\ \frac{\partial \phi}{\partial n} = -w\boldsymbol{u} \cdot \boldsymbol{n} & \text{on } \partial\Omega. \end{cases} \tag{9}$$

Solving for $\phi(\boldsymbol{x}, t)$, $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$ can then be determined from $\boldsymbol{x}(\boldsymbol{\xi}, 0)$ by integrating

$$\boldsymbol{x}_t = \boldsymbol{u}(\boldsymbol{x}, t) + \frac{1}{w(\boldsymbol{x}, t)}\nabla\phi(\boldsymbol{x}, t). \tag{10}$$

The mesh velocity field $\boldsymbol{v}(\boldsymbol{x}, t)$ can also be determined using a least squares framework. Defining the least squares functional

$$I[\boldsymbol{v}] = \frac{1}{2}\int_\Omega \left\{ \left| \nabla \cdot (\rho \boldsymbol{v}) + \frac{\partial \rho}{\partial t} \right|^2 + \left(\frac{\rho}{w}\right)^2 |\nabla \times w(\boldsymbol{v} - \boldsymbol{u})|^2 \right\} d\boldsymbol{x}, \tag{11}$$

a minimization is done over the space of functions satisfying the boundary condition (8). Once the mesh velocity has been determined, the mesh or coordinate transformation is defined as a family of trajectories of the vector field

$$\boldsymbol{x}_t(\boldsymbol{\xi}, t) = \boldsymbol{v}(\boldsymbol{x}(\boldsymbol{\xi}, t), t) \qquad \boldsymbol{\xi} \in \Omega_c. \tag{12}$$

The following theorem guarantees that the desired solution can be obtained through this minimization procedure. It can be proven by utilizing the orthogonality of the gradient and curl operators in the L^2 norm. For completeness and to better understand the role of (5), we give a detailed proof.

THEOREM 3.1. *If the compatibility condition* (5) *is satisfied, then the minimizer of* $I[\boldsymbol{v}]$ *in* (11) *satisfies* (4) *and* (6).

Proof. For notational simplicity, denote

$$\boldsymbol{a} = \left(\frac{\rho}{w}\right)^2 [\nabla \times w(\boldsymbol{v} - \boldsymbol{u})], \qquad g = \nabla \cdot (\rho \boldsymbol{v}) + \frac{\partial \rho}{\partial t}. \tag{13}$$

Taking variations, for an arbitrary function $\delta\boldsymbol{v}$ satisfying the boundary condition (8), i.e., $\delta\boldsymbol{v} \cdot \boldsymbol{n} = 0$ on $\partial\Omega$, we have

$$\delta I = \int_\Omega [g\nabla \cdot (\rho\delta\boldsymbol{v}) + \boldsymbol{a} \cdot (\nabla \times w\delta\boldsymbol{v})]\, d\boldsymbol{x}.$$

Using the identities

$$c\nabla\cdot\boldsymbol{d} = \nabla\cdot(c\boldsymbol{d}) - \nabla c\cdot\boldsymbol{d} \quad \forall \boldsymbol{d}\in\Re^3, \tag{14}$$

$$\boldsymbol{b}\cdot(\nabla\times\boldsymbol{d}) = \nabla\cdot(\boldsymbol{d}\times\boldsymbol{b}) + \boldsymbol{d}\cdot(\nabla\times\boldsymbol{b}) \quad \forall \boldsymbol{b},\ \boldsymbol{d}\in\Re^3 \tag{15}$$

and the divergence theorem, we have

$$\begin{aligned}\delta I &= \int_\Omega \{[\nabla\cdot(g\rho\delta\boldsymbol{v}) - (\nabla g)\cdot\rho\delta\boldsymbol{v}] + [\nabla\cdot(w\delta\boldsymbol{v}\times\boldsymbol{a}) + w\delta\boldsymbol{v}\cdot(\nabla\times\boldsymbol{a})]\}\,d\boldsymbol{x}\\ &= \int_\Omega [-\rho(\nabla g)\cdot\delta\boldsymbol{v} + w(\nabla\times\boldsymbol{a})\cdot\delta\boldsymbol{v}]\,d\boldsymbol{x} + \int_{\partial\Omega}[g\rho\delta\boldsymbol{v}\cdot\boldsymbol{n} + w(\boldsymbol{a}\times\boldsymbol{n})\cdot\delta\boldsymbol{v}]\,d\boldsymbol{S}.\end{aligned}$$

Thus, setting $\delta I = 0$ gives the Euler–Lagrange equation

$$-\rho\nabla\left[\nabla\cdot(\rho\boldsymbol{v}) + \frac{\partial\rho}{\partial t}\right] + w\nabla\times\left(\frac{\rho}{w}\right)^2[\nabla\times w(\boldsymbol{v}-\boldsymbol{u})] = 0 \quad \text{in } \Omega \tag{16}$$

and the natural boundary condition

$$\{[\nabla\times w(\boldsymbol{v}-\boldsymbol{u})]\times\boldsymbol{n}\}\cdot\boldsymbol{d} = 0 \qquad \text{on } \partial\Omega$$

for all functions $\boldsymbol{d}$ satisfying $\boldsymbol{d}\cdot\boldsymbol{n} = 0$ on $\partial\Omega$. Since $[\nabla\times w(\boldsymbol{v}-\boldsymbol{u})]\times\boldsymbol{n}$ is orthogonal to $\boldsymbol{n}$, this boundary condition is simply

$$[\nabla\times w(\boldsymbol{v}-\boldsymbol{u})]\times\boldsymbol{n} = 0 \qquad \text{on } \partial\Omega. \tag{17}$$

We now show that the Euler–Lagrange equation (16) and boundary condition (17) imply (4) if the compatibility condition is satisfied for the user-specified function ρ. Rewriting (16) as

$$\sqrt{\frac{\rho}{w}}\nabla g = \sqrt{\frac{w}{\rho}}\nabla\times\boldsymbol{a}, \tag{18}$$

multiplying by $\sqrt{w/\rho}$, and applying $(\nabla\times)$ leads to

$$\nabla\times\left(\frac{w}{\rho}\nabla\times\boldsymbol{a}\right) = 0. \tag{19}$$

Integrating the norm squared of both sides of (18) gives

$$\begin{aligned}\int_\Omega \frac{\rho}{w}\mid\nabla g\mid^2 d\boldsymbol{x} &= \int_\Omega \frac{w}{\rho}\mid\nabla\times\boldsymbol{a}\mid^2 d\boldsymbol{x}\\ &= \int_\Omega \frac{w}{\rho}(\nabla\times\boldsymbol{a})\cdot(\nabla\times\boldsymbol{a})d\boldsymbol{x}\\ &= \int_\Omega \nabla\cdot\left[\frac{w}{\rho}\boldsymbol{a}\times(\nabla\times\boldsymbol{a})\right] + \boldsymbol{a}\cdot\left[\nabla\times\left(\frac{w}{\rho}\nabla\times\boldsymbol{a}\right)\right]d\boldsymbol{x} \quad \text{(using (15))}\\ &= \int_{\partial\Omega}\frac{w}{\rho}\boldsymbol{a}\cdot[(\nabla\times\boldsymbol{a})\times\boldsymbol{n}]\,d\boldsymbol{S} \quad \text{(using (19) and div. thm.)}\\ &= -\int_{\partial\Omega}\frac{w}{\rho}\cdot(\boldsymbol{a}\times\boldsymbol{n})\cdot(\nabla\times\boldsymbol{a})d\boldsymbol{S}\\ &= 0,\end{aligned}$$

where we have used (19) and boundary condition (17). Hence, $\nabla g = 0$, so $g = c$, or

$$\nabla \cdot (\rho \boldsymbol{v}) + \frac{\partial \rho}{\partial t} = c. \tag{20}$$

Integrating over Ω and applying Gauss's theorem, the compatibility condition (5) implies $c = 0$, so (4) follows.

We now show that (6) also follows from the compatibility condition. From (4) and (16),

$$\nabla \times \left(\frac{\rho}{w}\right)^2 [\nabla \times w(\boldsymbol{v} - \boldsymbol{u})] = 0.$$

Multiplying by $w(\boldsymbol{v} - \boldsymbol{u})$, integrating over Ω, integrating by parts, and using the boundary condition (17), we obtain

$$\int \left(\frac{\rho}{w}\right)^2 \mid \nabla \times w(\boldsymbol{v} - \boldsymbol{u}) \mid^2 d\boldsymbol{x} = 0, \tag{21}$$

which implies (6). This completes the proof. □

Since $I[\boldsymbol{v}]$ is quadratic and bounded below, it has a unique minimizer whether the compatibility condition (5) is satisfied or not. However, when (5) is violated, the mesh velocity no longer satisfies (4) and equivalently the Jacobian is not specified by (3). Instead, (20) can be seen to hold with

$$c = \frac{\partial \bar{\rho}}{\partial t},$$

where $\bar{\rho}$ is the average of the function ρ over Ω. Comparing with the GCL condition (2) shows that

$$-\frac{1}{\rho}\frac{D(\rho - \bar{\rho})}{Dt} = \frac{1}{J}\frac{DJ}{Dt}.$$

So while (3) does not hold when (5) is violated, if $(D\bar{\rho})/(Dt)$ is small, then it is approximately satisfied.

Summary of formulations. Thus far we have seen that given the condition (5) there are several mathematically equivalent systems for determining $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$. For convenience, they are summarized below. Some implementation aspects will be discussed in section 5.

For the first, a system written explicitly in terms of $\boldsymbol{x}(\boldsymbol{\xi}, t)$, or more precisely, the time derivative $\boldsymbol{x}_t$, is solved. Specifically, combining (12) with (16), (8), and (17), we solve

$$\begin{cases} -\rho\nabla\left[\nabla \cdot (\rho \boldsymbol{x}_t) + \frac{\partial \rho}{\partial t}\right] + w\nabla \times \left(\frac{\rho}{w}\right)^2 [\nabla \times w(\boldsymbol{x}_t - \boldsymbol{u})] = 0 & \text{in } \Omega, \\ \boldsymbol{x}_t \cdot \boldsymbol{n} = 0 & \text{on } \partial\Omega, \\ [\nabla \times w(\boldsymbol{x}_t - \boldsymbol{u})] \times \boldsymbol{n} = 0 & \text{on } \partial\Omega. \end{cases} \tag{22}$$

This system is highly nonlinear because ρ, w, and $\boldsymbol{u}$ are functions of $\boldsymbol{x}$, and ∇ is defined in terms of the physical variables. The PDE is also nonstandard (neither parabolic nor hyperbolic) since the time derivative $\boldsymbol{x}_t$ appears in the highest spatial derivative terms.

For the second, $\boldsymbol{x}_t$ is obtained by solving for the potential function ϕ from (9) and (10), i.e.,

$$\text{(23)}\qquad \begin{cases} \boldsymbol{x}_t = \boldsymbol{u}(\boldsymbol{x},t) + \frac{1}{w(\boldsymbol{x},t)}\nabla\phi(\boldsymbol{x},t) & \text{in } \Omega, \\ \text{where } \phi \text{ satisfies} & \\ \nabla\cdot\left(\frac{\rho}{w}\nabla\phi\right) = -\frac{\partial\rho}{\partial t} - \nabla\cdot(\rho\boldsymbol{u}) & \text{in } \Omega, \\ \frac{\partial\phi}{\partial n} = -w\boldsymbol{u}\cdot\boldsymbol{n} & \text{on } \partial\Omega. \end{cases}$$

For the third, the coordinate transformation is obtained by computing the mesh velocity field $\boldsymbol{v}(\boldsymbol{x},t) = \boldsymbol{x}_t$. One approach for obtaining $\boldsymbol{v}$ is to directly minimize the functional $I[\boldsymbol{v}]$ subject to boundary condition (8), i.e.,

$$\text{(24)}\qquad \begin{cases} \boldsymbol{x}_t = \boldsymbol{v}(\boldsymbol{x},t) \quad \text{in } \Omega, \\ \text{where } \boldsymbol{v} \text{ directly minimizes } I[\boldsymbol{v}] \text{ in (11) subject to} \\ \boldsymbol{v}\cdot\boldsymbol{n} = 0 \quad \text{on } \partial\Omega. \end{cases}$$

A possible alternative approach is to find $\boldsymbol{v}$ by solving the div-curl system (4), (6), and (8), or

$$\text{(25)}\qquad \begin{cases} \boldsymbol{x}_t = \boldsymbol{v}(\boldsymbol{x},t) & \text{in } \Omega, \\ \text{where } \boldsymbol{v} \text{ satisfies} & \\ \nabla\cdot(\rho\boldsymbol{v}) + \frac{\partial\rho}{\partial t} = 0 & \text{in } \Omega, \\ \nabla\times w(\boldsymbol{v}-\boldsymbol{u}) = 0 & \text{in } \Omega, \\ \boldsymbol{v}\cdot\boldsymbol{n} = 0 & \text{on } \partial\Omega. \end{cases}$$

We conclude this section by making some observations about the existence and smoothness of the coordinate transformation. Since from (12) $\boldsymbol{x}(\boldsymbol{\xi},t)$ is determined as a family of integral curves of the direction field $\boldsymbol{v}(\boldsymbol{x},t)$, its existence and smoothness depend solely on the smoothness of $\boldsymbol{v}(\boldsymbol{x},t)$. Moreover, since $\boldsymbol{v}(\boldsymbol{x},t)$ is the solution of a system of elliptic PDEs—the Euler–Lagrange equation (16) of the quadratic functional (11)—it is smooth if $\partial\Omega$, ρ, w, and $\boldsymbol{u}$ are smooth. The choices of these functions are discussed in the next two sections.

4. Relation to and comparison with other moving mesh methods. Before discussing the implementation of the general moving mesh strategy described above, to which we refer hereafter as simply the *GCL method*, it is instructive to compare it with some of the existing moving mesh methods.

The Lagrangian method. The Lagrangian method, or the method of characteristics, has been widely used for simulation of incompressible fluid dynamics problems. Its primary feature is the use of the so-called Lagrangian coordinates defined by

$$\text{(26)}\qquad \boldsymbol{x}_t = \boldsymbol{u}_f(\boldsymbol{x},t),$$

where $\boldsymbol{u}_f$ is the flow velocity. In this coordinate system every mesh point represents a particle, so there are no convection terms in the governing equations, and the nonsingularity of the coordinate transformation from the Lagrangian coordinates to the Euler coordinates is guaranteed by the incompressibility condition of the fluid. However, as is well known, the mesh generated in the Lagrangian coordinates is often too skewed to be useful in simulating the diffusion process. This inadequacy has been a driving force behind the development of hybrid methods such as the particle-in-cell method [12].

The close relation between the GCL and Lagrangian methods is made clear by comparing (23) with (26). In fact, the GCL method can be regarded as a generalization of the Lagrangian method when choosing the control vector field $\boldsymbol{u}$ to be the flow velocity $\boldsymbol{u}_f$. As a special case, taking $\rho =$ constant (no adaption), for incompressible fluid flow where $\nabla \cdot \boldsymbol{u}_f = 0$ we have $\phi =$ constant, and a pure Lagrangian method (26) results.

The deformation map method. The deformation map is introduced by Moser [20] and Dacorogna and Moser [7] in their study of volume elements of a compact Riemannian manifold to prove the existence of a C^1 diffeomorphism with specified Jacobian. The mapping has been adopted by Liao and coworkers [2, 18, 23] to generate adaptive moving meshes. In our notation, this mapping $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t) : \Omega_c \equiv \Omega \to \Omega$ is determined from the system of equations [23]

$$\begin{cases} \boldsymbol{x}_t = \frac{1}{\rho(\boldsymbol{x},t)} \nabla \phi(\boldsymbol{x}, t) & \text{in } \Omega, \\ \Delta \phi = -\frac{\partial \rho}{\partial t} & \text{in } \Omega, \\ \frac{\partial \phi}{\partial n} = 0 & \text{on } \partial\Omega. \end{cases} \tag{27}$$

It is easy to see that (27) corresponds to the system (23)—the GCL method formulation which involves the potential function ϕ in the case where $\boldsymbol{u} = 0$ and $w = \rho$.

They also use the alternative formulation [2]

$$\begin{cases} \boldsymbol{x}_t = \frac{\boldsymbol{\nu}(\boldsymbol{x},t)}{\rho(\boldsymbol{x},t)} & \text{in } \Omega, \\ \nabla \cdot \boldsymbol{\nu} = -\frac{\partial \rho}{\partial t} & \text{in } \Omega, \\ \nabla \times \boldsymbol{\nu} = 0 & \text{on } \partial\Omega. \end{cases} \tag{28}$$

Note that letting $\boldsymbol{\nu} = \rho \boldsymbol{v}$, this becomes the div-curl system (25) with $\boldsymbol{u} = 0$ and $w = \rho$.

The equation for $\boldsymbol{x}_t$ in both cases is nonlinear, and for simplicity explicit time integration schemes are used in [2, 23]. For (27), the solution of the potential equation for ϕ is straightforward. However, it can be difficult to solve (27) with an implicit integrator because interpolation of the potential function ϕ is required at points other than grid nodes, and the flux-free condition cannot generally be preserved. For (28), the div-curl system is solved using a least squares approach. In the next section, we return to these issues when we discuss several numerical approaches to solve the systems.

While these two formulas using the deformation map are mathematically equivalent to (25) and are the special case of the GCL method with $w = \rho$, $\boldsymbol{u} = 0$, they are not necessarily the best choice. In fact, from (25) we see that the mesh velocity is generally not irrotational, viz.,

$$\nabla \times (\rho \boldsymbol{v}) = 0 \quad \text{or} \quad \nabla \times \boldsymbol{v} = -\frac{1}{\rho} \nabla \rho \times \boldsymbol{v}, \tag{29}$$

and our limited experience indicates that an irrotational mesh velocity field can often be preferable because more regular grids are produced (see the next section).

The moving finite element method. The MFE method developed by Miller and Miller [21, 22] also generates a moving mesh through a mesh velocity $\boldsymbol{x}_t$. Specifically, for a given time-dependent physical problem

$$\frac{\partial u}{\partial t} = \mathcal{L} u,$$

where $\mathcal{L}$ is a spatial differential operator, the continuous version of the MFE determines a solution $u(\boldsymbol{x}(\boldsymbol{\xi},t),t)$ and $\boldsymbol{x}_t(\boldsymbol{\xi},t)$ by minimizing the residual in the least squares sense, viz.,

$$\min_{\boldsymbol{x}_t,\frac{Du}{Dt}} I\left[\boldsymbol{x}_t,\frac{Du}{Dt}\right] \equiv \int_\Omega \left(\frac{Du}{Dt}-\nabla u\cdot\boldsymbol{x}_t-\mathcal{L}u\right)^2 W\,d\boldsymbol{x},$$

where the weight function $W=1$ for the classical version of MFE [21, 22] and $W=1/(1+|\nabla u|^2)$ for the gradient weighted MFE (GWMFE) [4, 5]. The functional derivative of $I[\boldsymbol{x}_t,\frac{Du}{Dt}]$ with respect to $\boldsymbol{x}_t$ can become singular and regularization is needed in practice.

While the formulation of MFE appears to be quite distinct from the GCL method, they share common features: both obtain the mesh equation by minimizing least squares functionals with respect to mesh velocity $\boldsymbol{x}_t$, and both have an inherent relation to the Lagrangian method [1].

The MMPDE method. There is another large class of adaptive mesh methods which contrast with those above. For these, the coordinate transformation is determined by minimizing a functional which involves the mesh transformation $\boldsymbol{x}$ directly instead of its velocity field. One basic approach is to define the inverse mapping $\boldsymbol{\xi}=\boldsymbol{\xi}(\boldsymbol{x},t):\Omega\to\Omega_c$ at each time level as the minimizer of the functional

$$I[\boldsymbol{\xi}]=\int_\Omega \sum_i (\nabla\xi^i)^T G^{-1}\nabla\xi^i\,d\boldsymbol{x}, \tag{30}$$

where the monitor function, G, is a symmetric positive definite matrix connecting mesh properties to the physical solution; see [3, 27]. A strategy of building in the mesh velocity $\boldsymbol{x}_t$, which is used by Huang, Ren, and Russell [13] and Huang and Russell [14, 15], is the so-called moving mesh PDE (MMPDE) approach. For it, one desires to move the mesh in a direction of decreasing $I[\boldsymbol{\xi}]$, e.g., so that

$$\frac{\partial\boldsymbol{\xi}}{\partial t}=-\frac{1}{\tau}\frac{\delta I}{\delta\boldsymbol{\xi}} \quad \text{or} \quad \frac{\partial\xi^i}{\partial t}=\frac{1}{\tau}\nabla\cdot(G^{-1}\nabla\xi^i), \tag{31}$$

where $\tau>0$ is the parameter used for adjusting the time scale of mesh movement. Interchanging the roles of the dependent and independent variables, a set of quasi-linear parabolic PDEs, or MMPDEs, is obtained for $\boldsymbol{x}_t(\boldsymbol{\xi},t)$.

Unlike the above three types of methods (the Lagrangian, MFE, and GCL methods), these methods employ a functional to determine the mapping itself rather than its mesh velocity $\boldsymbol{x}_t$, and as such the resulting coordinate transformation can be more or less viewed as a quasi-equilibrium state. A more obvious advantage is that, if the mesh solution drifts away from equilibrium, there is a stabilizing effect; in addition, skewness and smoothness of the mesh transformation can be easily controlled because it satisfies a basically elliptic system of equations at each time instant. In contrast, for the Lagrangian-type methods where the mesh is determined by minimizing a functional involving the mesh speed, the mesh speed instead of the coordinate transformation itself obeys an elliptic system of PDEs at each time level. A key question is whether or not this is more likely to produce unstable mesh movement and generate skewed meshes.

5. Implementation. A large variety of implementations of the GCL method are possible. We restrict attention to those given in section 3.

Among the most straightforward implementations is the first one based on the Euler–Lagrange equation (22). Any standard spatial discretization method can be used to discretize (22) on a physical mesh, and the resulting system of ODEs

$$A(\boldsymbol{X})\boldsymbol{X}_t = \boldsymbol{B}(\boldsymbol{X}), \tag{32}$$

where $\boldsymbol{X}$ is the vector of mesh nodes, can be integrated explicitly or implicitly. Unfortunately, with this implementation it may not be easy to realize the boundary conditions numerically.

One can avoid this difficulty by instead using (24) for a direct minimization method, i.e., discretize the functional (11) directly, then minimize with respect to the discrete velocity $\boldsymbol{v}_j(t) = \boldsymbol{v}(\boldsymbol{x}_j, t)$, and finally replace $\boldsymbol{v}_j$ with $(d\boldsymbol{x}_j/dt)$. However, the main difficulty with this method comes from the highly nonlinear nature of the resulting system (32), which can make its solution prohibitively expensive to compute and therefore impractical.

The second approach is based on the system (23) involving the potential function ϕ. For this method, the elliptic equation for the potential function is solved on the physical mesh, and a new mesh is then obtained by integrating the mesh equation $\boldsymbol{x}_t = \boldsymbol{v}$. As mentioned in the previous section, an explicit integration scheme has been used in [2, 23]. A possible problem with explicit integration is a severe stability restriction on the time step. On the other hand, an implicit integration scheme is not without problems. It requires interpolation of ϕ at nongrid points. The boundary condition, defined in terms of $\nabla\phi$, is not generally preserved by an interpolation method, and as a result boundary points can move out of the physical domain. The main advantage of this method is that it can be efficiently implemented in parallel, since only a Poisson equation (for the potential function) needs to be solved at each time step, and the mesh equation can be solved for the coordinates of each node independently.

The third approach is designed to avoid the problem of boundary points moving out of the physical domain during the implicit integration. It is based upon (24), or directly minimizing the functional (11) to obtain the mesh velocity field before integrating the mesh equation (12) for $\boldsymbol{x}$. Since the boundary condition (8) is defined directly in terms of the mesh velocity, it can be preserved when interpolating $\boldsymbol{v}$ in the implicit integration. The method is slightly more expensive to implement than the second one since a system of three PDEs must be solved for the three velocity components.

We have tested straightforward implementations of all three of these methods, and on the basis of this limited experience, we prefer the third in terms of overall performance. For this reason, we give only a detailed description of the FEM implementation of it.

Finite element discretization of the functional $\boldsymbol{I}[\boldsymbol{v}]$. We use an s-stage explicit or implicit Runge–Kutta method to integrate the mesh equation (12). This requires determining $\boldsymbol{v}(\boldsymbol{x}, t_{n,m}), 1 \le m \le s$, where $t_{n,m}$ is the intermediate time level at the mth stage of the nth step. Below we consider how to determine $\boldsymbol{v}(\boldsymbol{x}, t_{n,m})$ at a specific time $t_{n,m} \in [t_n, t_{n+1}]$.

Recall that $\boldsymbol{v}$ is the minimizer of the functional in (11) subject to boundary condition (8). While in principle the functional or least squares system at $t_{n,m}$ can be discretized over any mesh, for accuracy sake it is preferable to use an adaptive

mesh $\Omega_h(t_n)$ associated with the monitor function ρ. For this, assume that $\Omega_h(t_n)$ is composed of N elements $K(t_n)$, which are either triangles or quadrilaterals. Let $\phi_j(\boldsymbol{x})$ be the nodal basis function (viz., "hill function") associated with node j of the mesh $\Omega_h(t_n)$. More precisely, let $K(t_n)$ be an element with node j as one of its vertices, and let $F_{K(t_n)}$ be the affine mapping from a standard reference element $\hat{K}$ onto $K(t)$. If node j of $K(t_n)$ corresponds to node j' of $\hat{K}$, then $\phi_j(\boldsymbol{x})$ is defined on $K(t_n)$ as

$$\phi_j(\boldsymbol{x}) = \hat{\phi}_{j'}(F^{-1}_{K(t_n)}(\boldsymbol{x})),$$

where $\hat{\phi}_{j'}$ is the basis function associated with node j' of $\hat{K}$. Note that by definition $\phi_j(\boldsymbol{x})$ depends on the mesh $\Omega_h(t_n)$.

Let $\boldsymbol{v}_j$ be the value of the vector field $\boldsymbol{v}$ at node j of the mesh $\Omega_h(t_n)$. Approximating $\boldsymbol{v}(t_n)$ by the piecewise linear function

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

and substituting into the variation of $I[\boldsymbol{v}]$,

$$\delta I = \int_\Omega \left\{ \left[\nabla \cdot (\rho \boldsymbol{v}) + \frac{\partial \rho}{\partial t} \right] \nabla \cdot (\rho \delta \boldsymbol{v}) + \left(\frac{\rho}{w} \right)^2 [\nabla \times w(\boldsymbol{v} - \boldsymbol{u})] \cdot [\nabla \times (w \delta \boldsymbol{v})] \right\},$$

we obtain

$$\begin{aligned} \delta I = & \sum_j \int_\Omega \left\{ \nabla \cdot (\rho \phi_j \boldsymbol{v}_j) \nabla \cdot (\rho \delta \boldsymbol{v}) + \left(\frac{\rho}{w} \right)^2 [\nabla \times (w \phi_j \boldsymbol{v}_j)] \cdot [\nabla \times (w \delta \boldsymbol{v})] \right\} \\ & - \int_\Omega \left\{ -\frac{\partial \rho}{\partial t} \nabla \cdot (\rho \delta \boldsymbol{v}) + \left(\frac{\rho}{w} \right)^2 [\nabla \times (w \boldsymbol{u})] \cdot [\nabla \times (w \delta \boldsymbol{v})] \right\}. \end{aligned} \tag{33}$$

Letting $\delta I = 0$ and taking $\delta \boldsymbol{v} = \boldsymbol{e}_k \phi_i, k = 1, 2, 3, i = 1, \ldots, N$, where $\{\boldsymbol{e}_k, k = 1, 2, 3\}$ are the unit vectors in $\Re^3$, we obtain the system of algebraic equations

$$S\boldsymbol{V} = \boldsymbol{F}, \tag{34}$$

where

$$S = (s_{ij}), \quad \boldsymbol{V} = \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_N \end{pmatrix}, \quad \boldsymbol{F} = \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \vdots \\ \boldsymbol{f}_N \end{pmatrix},$$

and s_{ij} and $\boldsymbol{f}_i$ are defined by

$$\begin{aligned} \boldsymbol{a}^T s_{ij} \boldsymbol{b} &= \int_\Omega \left\{ \nabla \cdot (\rho \phi_j \boldsymbol{b}) \nabla \cdot (\rho \phi_i \boldsymbol{a}) + \left(\frac{\rho}{w} \right)^2 [\nabla \times (w \phi_j \boldsymbol{b})] \cdot [\nabla \times (w \phi_i \boldsymbol{a})] \right\}, \\ \boldsymbol{a}^T \boldsymbol{f}_i &= \int_\Omega \left\{ -\frac{\partial \rho}{\partial t} \nabla \cdot (\rho \phi_i \boldsymbol{a}) + \left(\frac{\rho}{w} \right)^2 [\nabla \times (w \boldsymbol{u})] \cdot [\nabla \times (w \phi_i \boldsymbol{a})] \right\} \end{aligned} \tag{35}$$

for all $\boldsymbol{a}, \boldsymbol{b} \in \Re^3$. Note that S is symmetric positive definite.

Time integration of $\boldsymbol{x_t = v(x,t)}$. While an explicit method is straightforward to implement for the time discretization of the ODE system (12), we employ implicit methods because of their superior stability properties. In particular, we use the second-order singly diagonally implicit Runge–Kutta method SDIRK2 and solve the resulting systems of linear equations with the preconditioned BiCGStab2 [10, 26] with a level-1 fill-in ILU preconditioner. During the integration of the mesh equation $\boldsymbol{x}_t = \boldsymbol{v}$, piecewise linear (or bilinear) interpolation is used to obtain values of $\boldsymbol{v}(\boldsymbol{x}, t)$ at points other than the grid points of $\Omega_h(t_n)$.

Choices for functions $\boldsymbol{w}$, $\boldsymbol{\rho}$, and $\boldsymbol{u}$. As mentioned in the previous section, there are two more or less obvious choices for the weight function, $w = \rho$ and $w = 1$. The former corresponds to the deformation map method and generally does not result in an irrotational mesh velocity field (even when $\boldsymbol{u} = 0$). The latter results in an irrotational mesh velocity field when $\boldsymbol{u} = 0$, i.e.,

$$\nabla \times \boldsymbol{v} = 0. \tag{36}$$

Of course, numerous other options are also possible, but their mathematical and/or physical significance is unclear to us.

The choice of the control vector field $\boldsymbol{u}$ is very problem dependent. For fluid dynamics problems, a good choice of it can be the flow velocity, since this is likely to reduce the magnitude of the convection term. However, when mesh adaption is allowed (i.e., ρ is not constant), grid movement due to the mesh adaption may increase the convection term and make $\boldsymbol{u}$ more difficult to choose. While this issue certainly warrants further investigation, we have found that generally speaking, when physical intuition for choosing $\boldsymbol{u}$ is not available, the best option is simply to choose $\boldsymbol{u} = 0$.

Our choice of the monitor function ρ is primarily motivated by the fact that (3) is a generalization of the equidistribution principle. In fact, (3) implies that ρ plays the role of a density function, i.e., the larger ρ, the denser the mesh. We can compute ρ using the solution gradient or an error indicator such as an interpolation error estimator. At the same time, the overall mesh density can also be controlled by limiting the ratio $(\max \rho)/(\min \rho)$. Finally, it is worth mentioning that ρ can be chosen as a physical variable which is positive and whose integral over the domain is conserved. An example is the (real) density function in compressible fluid dynamics.

6. Numerical experiments. To demonstrate various features of the moving mesh method based on the GCL, in this section we present some two-dimensional numerical results for which the monitor function is given. The method is also forced (numerically) to satisfy the compatibility condition (5). Specifically, given a density function $d = d(x, y, t)$ for the desired mesh adaption, we define the normalized monitor function

$$\rho(x, y, t) = \frac{d(x, y, t)}{\int_\Omega d(\tilde{x}, \tilde{y}, t) d\tilde{x} d\tilde{y}}, \tag{37}$$

where the integral is calculated with a midpoint rule. Unless stated otherwise, we use a 40×40 uniform rectangular mesh as the initial mesh in the computations. The control vector field is taken as $\boldsymbol{u} = 0$ in all of the examples but Example 5. The function

$$E(x, y, t) = \rho(x, y, t) J(x, y, t)$$

is used to measure the degree to which the computed mesh satisfies the equidistribution relation (3). Recall that in theory $E(x, y, t)$ is time independent, and thus remains constant. In fact, $E = 1$ initially for all the examples here, so it can be expected that $E(x, y, t)$ will be close to one for an accurate mesh.

Example 1. For our first example, an adaptive mesh will be generated for the density function

$$d(x,y) = 1 + A\exp\left(-50\left|\left(x-\frac{1}{2}\right)^2 + \left(y-\frac{1}{2}\right)^2 - \left(\frac{1}{4}\right)^2\right|\right),$$

where A is a parameter modulating the ratio of the largest cell size to the smallest one. Loosely speaking, the mesh should be about $1+A$ times denser around the circle $(x-\frac{1}{2})^2 + (y-\frac{1}{2})^2 = (\frac{1}{4})^2$ than in other regions.

Using the GCL method, the time-dependent monitor function is defined by

$$d(x,y,t) = 1 + tA\exp\left(-50\left|\left(x-\frac{1}{2}\right)^2 + \left(y-\frac{1}{2}\right)^2 - \left(\frac{1}{4}\right)^2\right|\right),$$

so it changes continuously from the constant 1 at $t = 0$ to $d(x, y)$ at $t = 1$. Three parameter values $A = 5, 10$, and 20 are considered. The weight function is taken as $w = 1$, so the mesh velocity field is irrotational.

Figure 1 shows the computed adaptive meshes and $E(x, y, t)$ at $t = 1$ for the cases $A = 5$ and 20 with time step $\delta t = 0.1$. Note that the mesh is concentrated in the proper region, and E is around 1 for both cases. In fact, at $t = 1$ we have $0.9645 \le E \le 1.0774$ for $A = 5$ and $0.7555 \le E \le 1.1417$ for the more difficult case $A = 20$. The deviation of E from 1 generally depends on the sizes of the time step and the mesh cells as well as the value of parameter A (which determines the ratio of the largest-to-smallest cell size). The larger the time step size, or the coarser the mesh, or the the bigger the value of A, the larger the discretization error and consequently the larger the deviation of E from 1. In Table 1 we list the maximum deviation $\|E-1\|_\infty$ for the case $A = 10$ and for various sizes of time step and mesh. It is clear that E approaches 1 when δt is reduced and the mesh size is increased. Table 2 lists the maximum deviation for $A = 5, 10$, and 20 with a 40×40 mesh and various values of δt. Not surprisingly, the size of δt required to generate satisfactory adaptive meshes (e.g., having E within $\pm 20\%$ of 1) depends upon the size of the parameter A. Consequently, to closely approximate the generalized equidistribution relation (3), the number of time integration steps should be proportional to the ratio of the largest-to-smallest cell size—a large step size δt can lead to poor results. This can be seen in Figure 2 where $\delta t = 0.2$ is used for the cases $A = 10$ and 20.

Computations done with the weight function $w = \rho$ give nearly the same results for this example. This is because the mesh movement (starting from a uniform initial mesh) is approximately in the radial direction of the circle (which is identical to the direction of $\nabla\rho$), and therefore we roughly have $\nabla\rho \times \boldsymbol{x}_t = 0$ or $\nabla \times \boldsymbol{x}_t = 0$ (see (29)). In other words, for this example the mesh velocity is irrotational even when $w = \rho$ is used.

Example 2. For the second example, a moving mesh is generated for the time-dependent monitor function defined by (37) and the density function

$$d(x,y,t) = \begin{cases} 1 + 100(t+0.1)\exp(-50|(x-\frac{1}{2})^2 + (y-\frac{1}{2})^2 - 0.09|) \\ \qquad \text{for } -0.1 < t < 0, \\ 1 + 10\exp(-50|(x-\frac{1}{2}-t)^2 + (y-\frac{1}{2})^2 - 0.09|) \\ \qquad \text{for } t \ge 0. \end{cases}$$

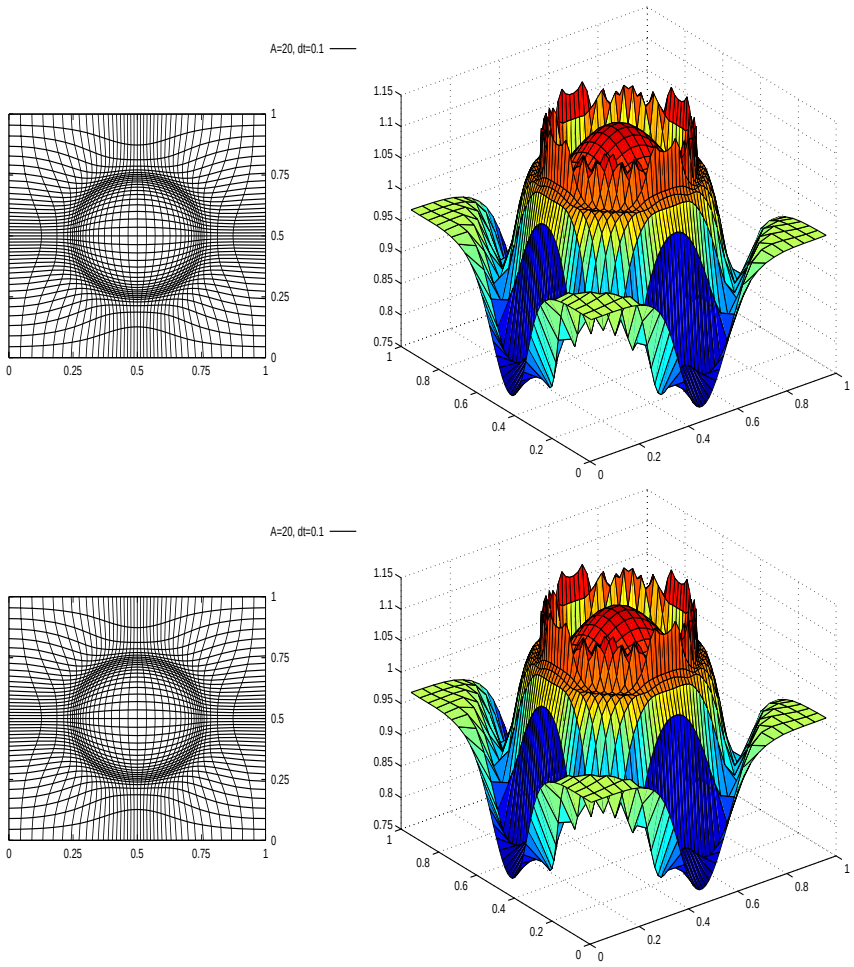

FIG. 1. *Example* 1. *Adaptive meshes and* E *for the case* $A = 5$ *(top) and* $A = 20$ *(bottom) with* 40×40 *mesh and* $\delta t = 0.1$.

TABLE 1

The maximum deviation $\|E - 1\|_\infty$ *at* $t = 1$ *for Example* 1 *with* $A = 10$, $w = 1$, $\boldsymbol{u} = 0$ *and for various sizes of time step and mesh.*

Mesh	$\delta t = 0.2$	$\delta t = 0.1$	$\delta t = 0.05$	$\delta t = 0.025$
20×20	1.0323	0.2294	0.1989	0.1802
40×40	0.1437	0.0954	0.0983	0.0945
80×80	0.2202	0.0445	0.0368	0.0335
160×160	0.2881	0.0257	0.0191	0.0161

TABLE 2

The maximum deviation $\|E-1\|_\infty$ *at* $t = 1$ *for Example* 1 *with a* 40×40 *mesh and for various values of* A *and time step.*

	$\delta t = 0.2$	$\delta t = 0.1$	$\delta t = 0.05$	$\delta t = 0.025$
$A = 5$	0.1006	0.0774	0.0662	0.0614
$A = 10$	0.1437	0.1049	0.0983	0.0945
$A = 20$	1.8845	0.2445	0.2385	0.2356

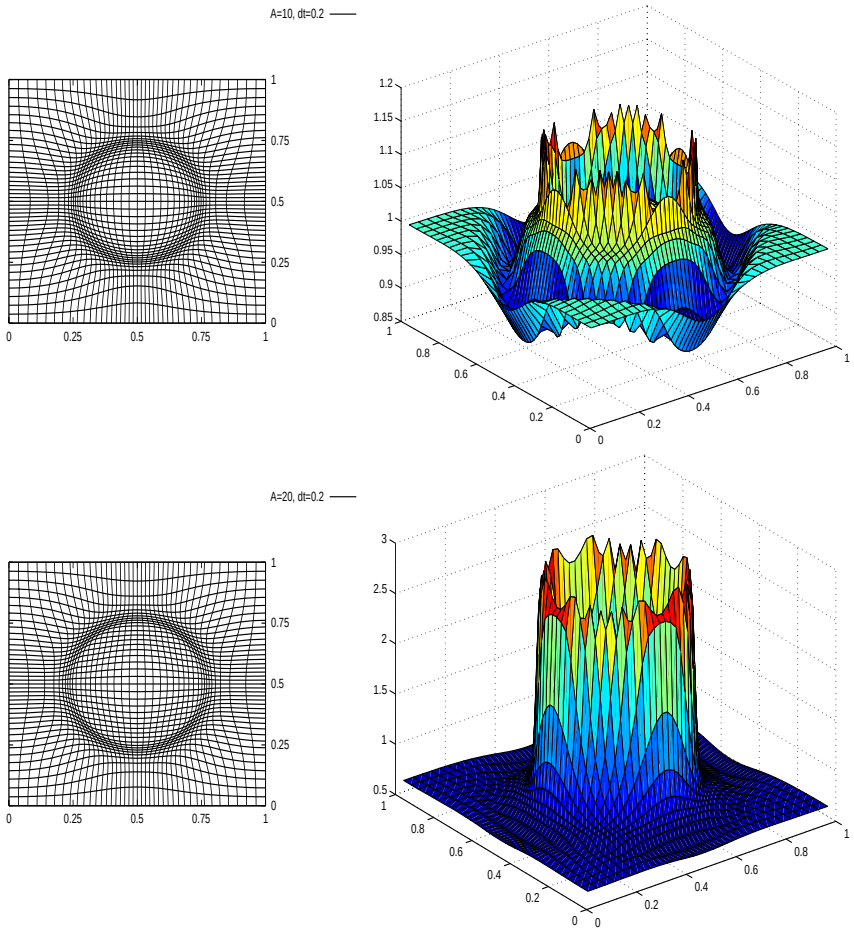

FIG. 2. *Example* 1. *Adaptive meshes and* E *for the case* $A = 10$ *(top) and* $A = 20$ *(bottom) with* 40×40 *mesh and* $\delta t = 0.2$.

This function is defined using two time phases, one from $t = -0.1$ to $t = 0$ and the other for $t > 0$. The purpose of the first phase is to produce an adaptive mesh for $t = 0$, starting from a uniform mesh at $t = -0.1$. For $t > 0$, the function simulates a circular peak which moves right at speed 1 and eventually leaves the domain while maintaining its shape.

We plot in Figure 3 the moving mesh and the distribution of E obtained for $w = 1$ and $\delta t = 0.01$. As expected, the mesh points are concentrated around the circle $(x - \frac{1}{2})^2 + (y - \frac{1}{2})^2 = 0.3^2$ at the beginning $(t = 0)$ and then follow the movement of the circular peak. The function E ranges from 0.6 to 1.4 during the course of mesh movement.

In Figure 4 we display the corresponding results obtained with the weight function $w = \rho$. It is apparent that the moving mesh is considerably different from the previous one. The major influence of $w = \rho$ occurs near the top and bottom parts of the moving circle $(x - \frac{1}{2} - t)^2 + (y - \frac{1}{2})^2 = 0.3^2$. This can be explained as follows. The velocity field $\boldsymbol{x}_t$ is in the horizontal direction and $\nabla\rho$ is in the radial direction of the circle, so

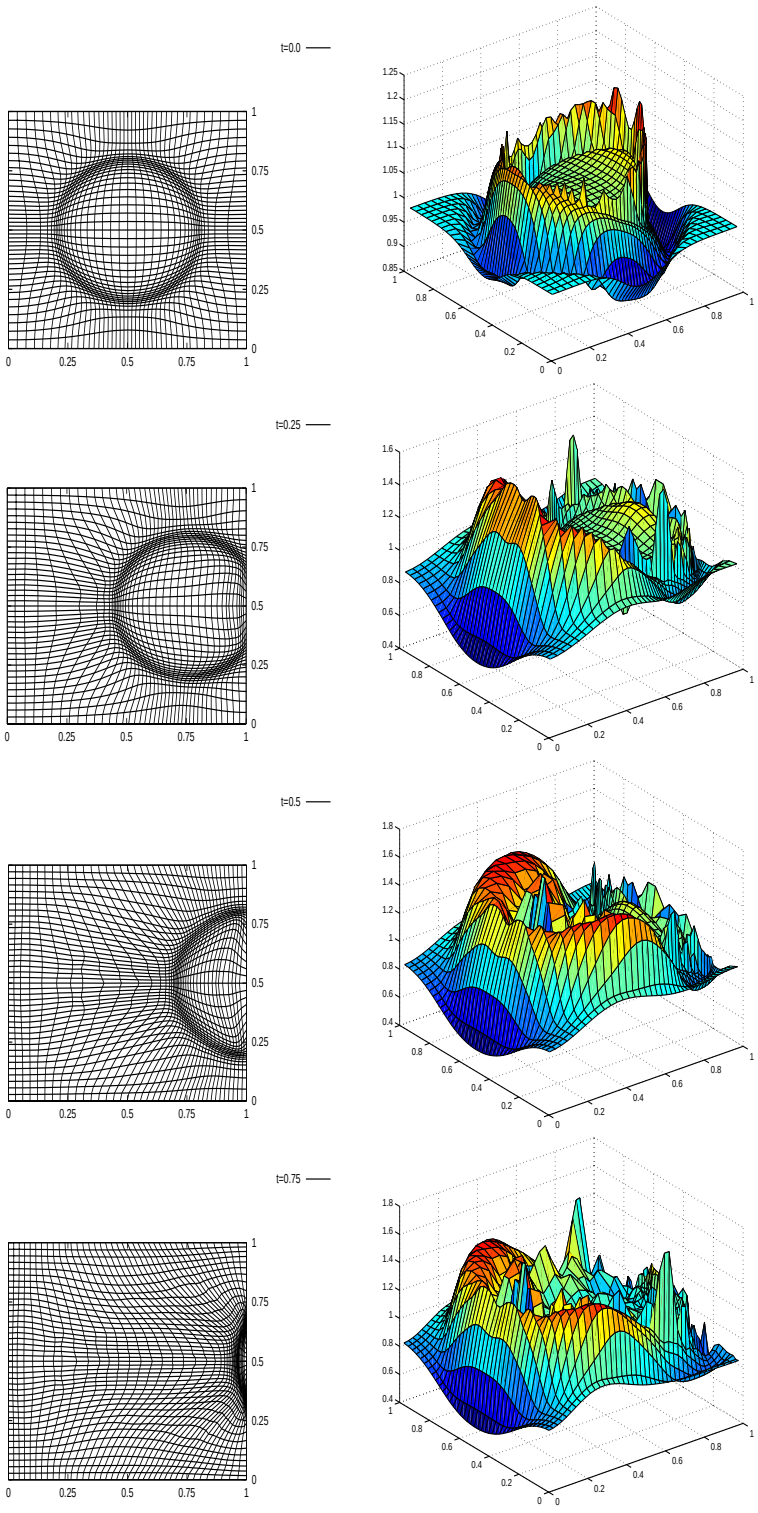

FIG. 3. *Example* 2. *Moving meshes and* E *at time* $t = 0, 0.25, 0.5, 0.75$ *(with* $w = 1$*).*

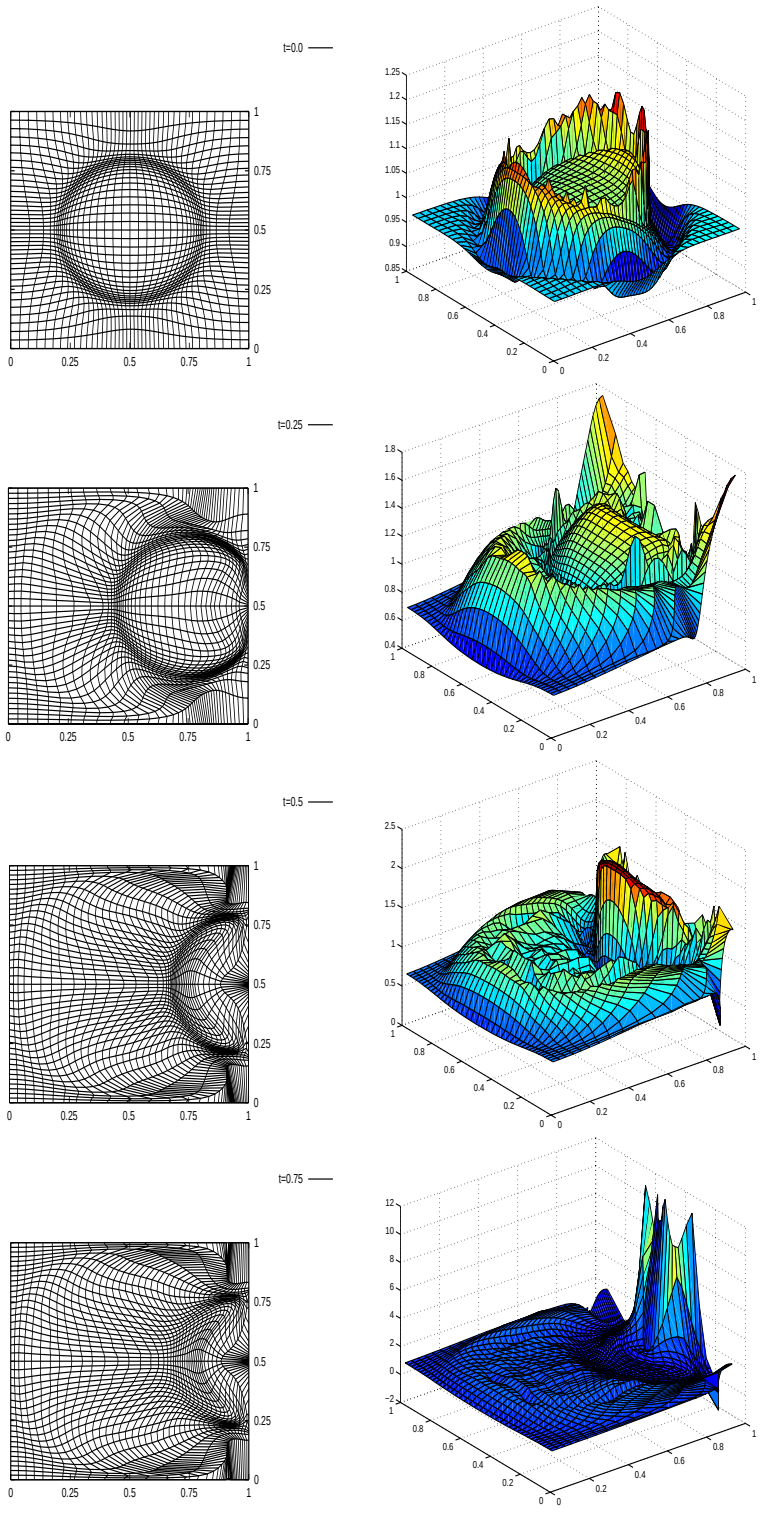

FIG. 4. *Example* 2. *Moving meshes and* E *at time* $t = 0, 0.25, 0.5, 0.75$ *(with* $w = \rho$*).*

$\nabla\rho \times \boldsymbol{x}_t$ vanishes at the left and right tips and takes extreme values at the top and bottom tips. From (29), the mesh velocity field is only rotation free at the left and right tips, and the mesh points rotate most at the top and bottom of the circle.

It is evident from this particular example that the moving meshes generated with $w = 1$ are less skewed than those obtained with $w = \rho$. Although it is in general hard to predict the precise influence caused by the choice $w = \rho$, it appears from our experience that $w = 1$, which produces an irrotational mesh velocity, will generally produce better behaved adaptive meshes.

It is interesting to note that the meshes obtained for this example are not smooth; see Figures 3 and 4. This appears to be an inherent feature of the GCL method, because the locations of the mesh points are not governed by either an elliptic or a parabolic PDE. The nonsmoothness can be seen more clearly in the next example.

Example 3. This example is chosen to demonstrate the effect of discretization error and the nonsmoothness of meshes generated by the GCL method. The density function is given as

$$d(x,y,t) = \begin{cases} 1 + 50(0.1+t)\exp(-50|y - \frac{1}{2}|) & \text{for } -0.1 < t < 0, \\ 1 + 5\exp(-50|y - \frac{1}{2} - \frac{1}{4}\sin(2\pi x)\sin(2\pi t)|) & \text{for } 0 \le t \le 1, \end{cases}$$

which simulates the motion of a periodic sine wave. We use $w = 1$ and $\delta t = 0.005$.

The moving mesh is plotted in Figure 5 for various time instants. Clearly, the mesh concentration captures the moving sine wave accurately. The function E ranges from 0.7 to 1.3 over the time period. Note that the monitor function ρ is periodic, and at time $t = 1$ it is restored to its original profile. However, the mesh does not return to its original form, mainly because of the spatial and temporal discretization errors. The nonsmoothness of the mesh is also noticeable.

Liu, Ji, and Liao [19] propose a "static mode of the moving mesh method" in an attempt to avoid this undesired effect of the discretization error. For any given time level $\hat{t}$, the idea is to fix the density function values as constant—its value at $\hat{t}$—and use continuation starting from a uniform mesh to produce the needed adaptive mesh at $\hat{t}$. This procedure preserves the periodicity of the monitor function and, as one may expect, generates a better adaptive moving mesh. Unfortunately, the procedure can turn out to be extremely costly. As we have seen in Example 1, a substantial number of continuation steps must be used to generate a reasonably accurate adaptive mesh, and this number increases with a correspondingly increased level of adaptivity (where the ratio of largest-to-smallest cell size grows).

Example 4. For this example the density function is given by

$$d(x,y,t) = \begin{cases} 1 + 50(0.1+t)\exp(-50|(x - \frac{3}{4})^2 + (y - \frac{1}{2})^2 - .01|) \\ \qquad \text{for } -0.1 < t < 0, \\ 1 + 5\exp(-50|(x - \frac{1}{2} - \frac{1}{4}\cos(2\pi t))^2 + (y - \frac{1}{2} - \frac{1}{4}\sin(2\pi t))^2 - .01|) \\ \qquad \text{for } t \ge 0. \end{cases}$$

For this monitor function, the largest values of ρ occur around a small circle which rotates about the point $(\frac{1}{2}, \frac{1}{2})$.

This is a very difficult test problem for many moving mesh methods, especially for ones with close similarity to the Lagrangian method. With these methods, as the mesh points and concentration follow the small moving circle rotating around the point $(\frac{1}{2}, \frac{1}{2})$, if some of the boundary mesh points stay fixed (as in the current case where the four corner points are fixed), then the mesh becomes more and more skewed and eventually singular. Unfortunately, the present moving mesh based on

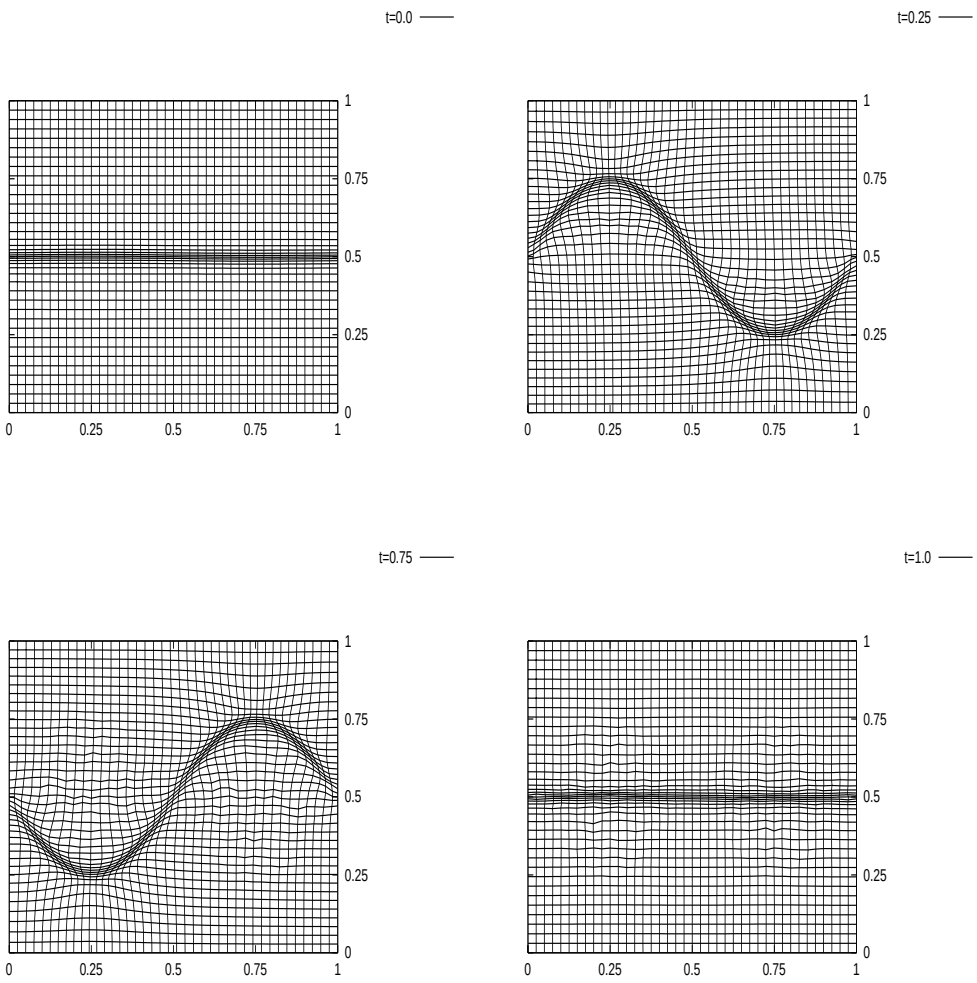

FIG. 5. *Example* 3. *Moving meshes at time* $t = 0, 0.25, 0.75, 1$.

the GCL also suffers from this difficulty. Although the GCL condition guarantees nonsingularity of the Jacobian of the coordinate transformation in the continuous case, it does not prevent the mesh from becoming increasingly skewed. Meanwhile, the points of a highly skewed mesh can easily tangle each other numerically. This is illustrated in Figure 6 for the case $w = 1$ and $\delta t = 0.005$.

The function E ranges from 0.7 to 1.3, but the moving mesh becomes more skewed with time. That is, even though the mesh becomes very skewed, the generalized equidistribution relation is closely satisfied.

Example 5. The final example illustrates the effect of the control vector field $\boldsymbol{u}$ on the adaptive moving mesh. As discussed in section 4, the GCL method reduces to a pure Lagrangian method when $\rho =$ constant and $\boldsymbol{u}$ is divergence free inside the domain and satisfies $\boldsymbol{u} \cdot \boldsymbol{n} = 0$ on the boundary. A special example is where the mesh velocity is given by the angular direction $\boldsymbol{u}$ for a unit disk Ω. However, the mesh movement is not so obvious if $\boldsymbol{u} \cdot \boldsymbol{n}$ is not zero on the boundary and/or some of the

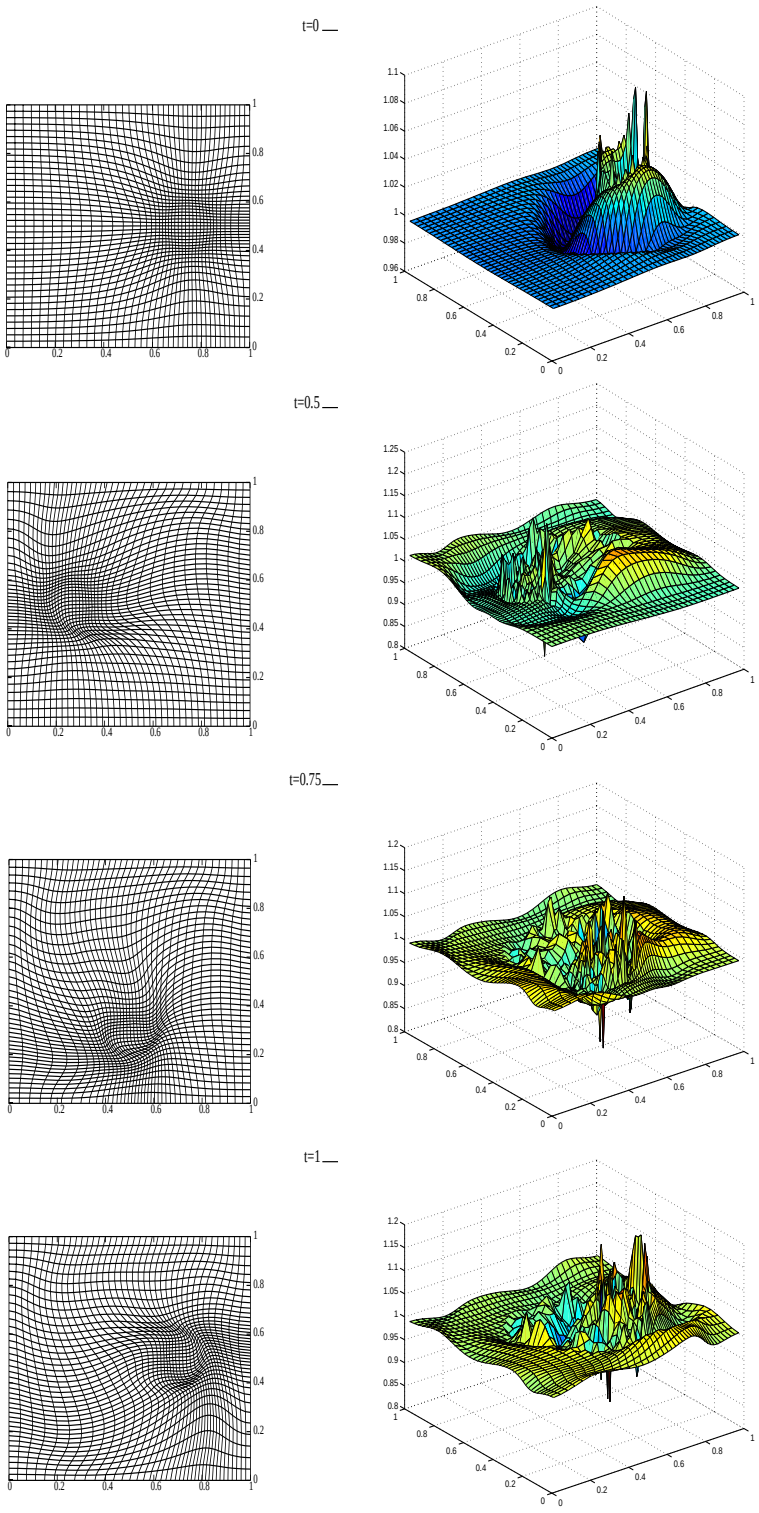

FIG. 6. *Example* 4. *Moving meshes and* E *at time* $t = 0, 0.5, 0.75, 1$.

boundary mesh points such as corner points stay fixed. To see this, we consider the case where $w = 1$, $\rho = 1$, and

$$\boldsymbol{u} = 2\pi \left(-y + \frac{1}{2},\ x - \frac{1}{2} \right).$$

The control vector field is a divergence-free rotation around the center of the domain, with angular speed 1. In Figure 7 we display the moving mesh and E at times $t = 0, 0.1, 0.2$ obtained with $\delta t = 0.01$. Away from the boundary, the influence of the boundary condition is small and the mesh points move at a speed close to $\boldsymbol{u}$. However, near the boundary, and especially on it, the boundary condition has a much more significant effect, and as a consequence the mesh points accumulate around the corners (where the corner points are fixed). The mesh stops moving at around $t = 0.2$ because too many points have accumulated near the corners. It is interesting to note that for the current case we nevertheless have $0.9 \leq E \leq 1.1$, indicating a high degree of satisfaction of the equidistribution relation (3) by the computed mesh.

7. Conclusions and comments. We present in this paper a moving mesh method based upon the geometric conservation law. The basic idea is to minimize a functional of the mesh speed $\boldsymbol{x}_t$ which involves its divergence and curl components. While the divergence part comes from the geometric conservation law and a prescribed monitor function, the curl of $\boldsymbol{x}_t$ is controlled by a given weight function w and a vector field $\boldsymbol{u}$. Using the Helmholtz decomposition theorem, we see why the minimizer of the functional (of the mesh speed) will drive mesh points in such a way that the rate of change of the cell volumes is proportional to the rate of change of the monitor function, causing the desired mesh adaption. A nice feature of this approach is that the cell volume can then be determined from the magnitude of the monitor function—a generalization of the well-known equidistribution principle.

Several different formulations of the basic mesh adaption method have been presented. The moving mesh method based upon Moser's deformation method which has been recently studied by Liao and his coworkers can be interpreted as a particular case of one of them with $w = \rho$ and $\boldsymbol{u} = 0$. In our somewhat limited experience, a more desirable choice of w from the numerical point of view appears to be $w = 1$, which results in an irrotational mesh velocity field, namely, $\nabla \times \boldsymbol{x}_t = 0$, and results in less skewed meshes. Other choices for w and $\boldsymbol{u}$ are yet to be investigated.

Several numerical examples are presented to demonstrate the performance of the moving mesh method (or, more precisely, of an implementation of the formulation which we prefer). It is shown that the cell volumes of the resulting adaptive meshes are usually within $\pm 30\%$ of the exact ones defined by the monitor function.

Like other moving mesh methods based on determining the mesh speed, the moving mesh method based on the GCL shares many common features with the Lagrangian method and consequently suffers from the same difficulty: a highly skewed mesh can often result when the mesh points follow the moving features in a monitor function. Though it is possible to use a brute force method to circumvent this difficulty, e.g., simply discard all the intermediate moving steps and calculate the adaptive mesh from the initial mesh, more efficient and effective approaches are clearly needed to correct meshes for which some control over skewness has been lost.

The GCL method shares with Miller's MFE the feature that they both minimize a functional with respect to the mesh velocity. But unlike the MFE, for which a singular mass matrix can result, the GCL method in theory leads to a well-defined PDE (although it may be quite expensive to solve).

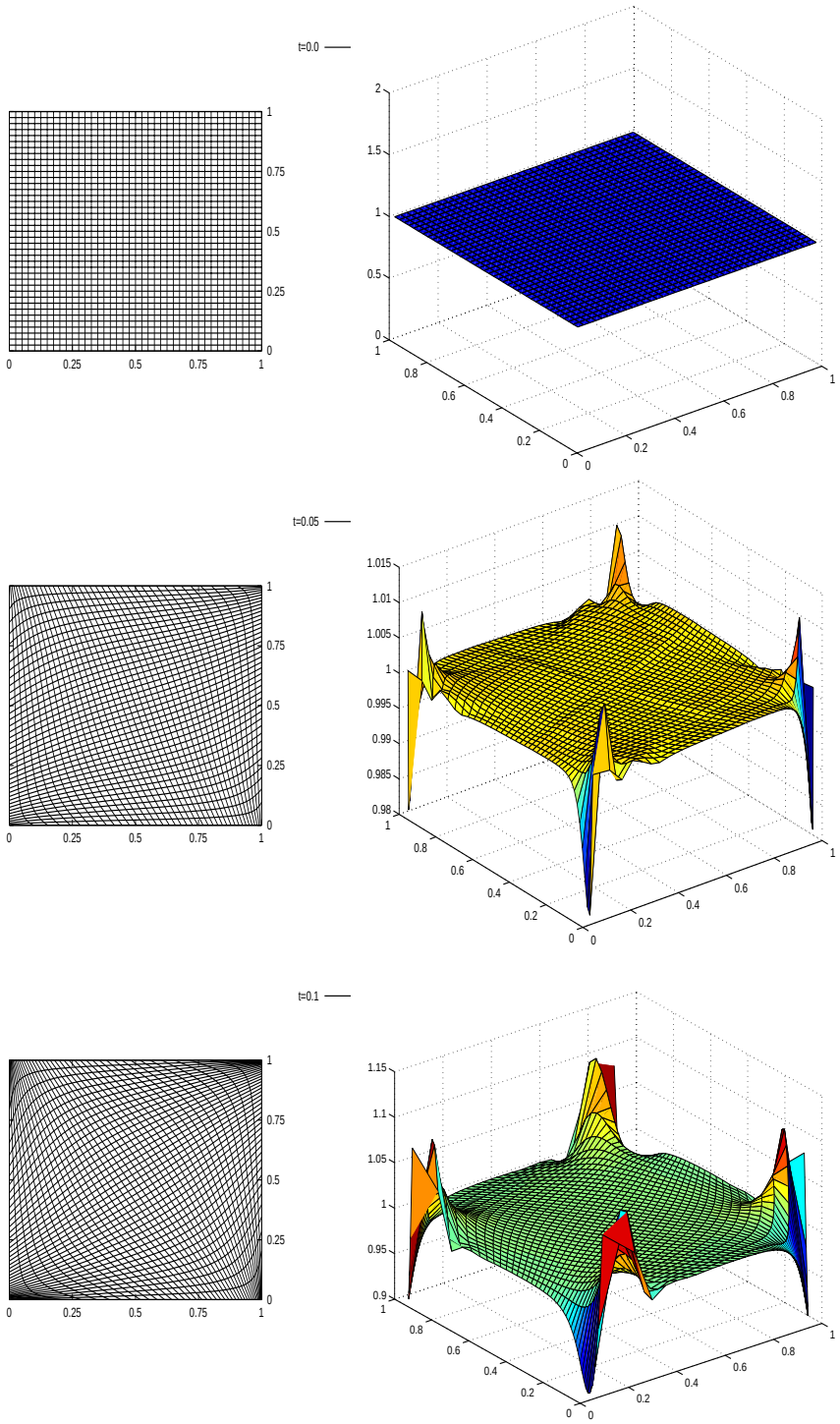

FIG. 7. *Example* 5. *Moving meshes and* E *at time* $t = 0, 0.05, 0.1$.

The GCL method is also compared with the MMPDE method. The MMPDE method involves a formulation in terms of an elliptic PDE. With the MMPDE approach, the mesh *locations* are obtained by minimizing a quadratic functional, and in this sense, the mesh locations are globally distributed. As a consequence, a smooth and less skewed mesh can often result. The GCL approach, on the other hand, by minimizing a quadratic functional determines the mesh speed instead of mesh location, and only the mesh speed is globally distributed. As demonstrated in the numerical results, generating nonsmooth, skew adaptive meshes can generally not be avoided, and the challenge of overcoming this difficulty remains.

Acknowledgment. The authors would like to thank the referees for their very constructive comments.

REFERENCES

[1] M. J. Baines, *Moving Finite Elements*, Oxford University Press, New York, 1994.
[2] P. Bochev, G. Liao, and G. Pena, *Analysis and computation of adaptive moving grids by deformation*, Numer. Methods Partial Differential Equations, 12 (1996), pp. 489–506.
[3] J. U. Brackbill and J. S. Saltzman, *Adaptive zoning for singular problems in two dimensions*, J. Comput. Phys., 46 (1982), pp. 342–368.
[4] N. N. Carlson and K. Miller, *Design and application of a gradient-weighted moving finite element code* I: In one dimension, SIAM J. Sci. Comput., 19 (1998), pp. 728–765.
[5] N. N. Carlson and K. Miller, *Design and application of a gradient-weighted moving finite element code* II: In two dimensions, SIAM J. Sci. Comput., 19 (1998), pp. 766–798.
[6] C. I. Christov, *Orthogonal coordinate meshes with manageable Jacobian*, in Numerical Grid Generation, J. F. Thompson, ed., North-Holland, New York, 1982, pp. 885–894.
[7] B. Dacorogna and J. Moser, *On a partial differential equation involving the Jacobian determinant*, Ann. Inst. H. Poincaré Anal. Non Linéaire, 7 (1990), pp. 1–26.
[8] I. Demirdžić and M. Perić, *Space conservation law in finite volume calculations of fluid flow*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 1037–1050.
[9] I. Demirdžić and M. Perić, *Finite volume method for prediction of fluid flow in arbitrarily shaped domains with moving boundaries*, Internat. J. Numer. Methods Fluids, 10 (1990), pp. 771–790.
[10] M. H. Gutknecht, *Variants of BICGSTAB for matrices with complex spectrum*, SIAM J. Sci. Comput., 14 (1993), 1020–1033.
[11] C. W. Hirt, A. A. Amsden, and J. L. Cook, *An arbitrary Lagrangian-Eulerian computing method for all flow speeds*, J. Comput. Phys., 14 (1974), pp. 227–253.
[12] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, McGraw-Hill, New York, 1981.
[13] W. Huang, Y. Ren, and R. D. Russell, *Moving mesh partial differential equations (MMPDEs) based on the equidistribution principle*, SIAM J. Numer. Anal., 31 (1994), pp. 709–730.
[14] W. Huang and R. D. Russell, *A high dimensional moving mesh strategy*, Appl. Numer. Math., 26 (1997), pp. 63–76.
[15] W. Huang and R. D. Russell, *Moving mesh strategy based on a gradient flow equation for two-dimensional problems*, SIAM J. Sci. Comput., 20 (1999), pp. 998–1015.
[16] P. M. Knupp, *Jacobian-weighted elliptic grid generation*, SIAM J. Sci. Comput., 17 (1996), pp. 1475–1490.
[17] P. M. Knupp and N. Robidoux, *A framework for variational grid generation: Conditioning the Jacobian matrix with matrix norms*, SIAM J. Sci. Comput., 21 (2000), pp. 2029–2047.
[18] G. J. Liao and D. Anderson, *A new approach to grid generation*, Appl. Anal., 44 (1992), pp. 285–298.
[19] F. Liu, S. Ji, and G. J. Liao, *An adaptive grid method and its application to steady Euler flow calculations*, SIAM J. Sci. Comput., 20 (1998), pp. 811–825.
[20] J. Moser, *On the volume elements of a manifold*, Trans. Amer. Math. Soc., 120 (1965), pp. 286–294.
[21] K. Miller and R. N. Miller, *Moving finite elements.* I, SIAM J. Numer. Anal., 18 (1981), pp. 1019–1032.
[22] K. Miller, *Moving finite elements.* II, SIAM J. Numer. Anal., 18 (1981), pp. 1033–1057.

[23] B. Semper and G. Liao, *A moving grid finite-element method using grid deformation*, Numer. Methods Partial Differential Equations, 11 (1995), pp. 603–615.

[24] P. D. Thomas and C. K. Lombard, *Geometric conservation law and its application to flow computations on moving grids*, AIAA J., 17 (1979), pp. 1030–1037.

[25] J. G. Trulio and K. R. Trigger, *Numerical Solution of the One-Dimensional Hydrodynamic Equations in an Arbitrary Time-Dependent Coordinate System*, Report UCLR-6522, Lawrence Radiation Laboratory, University of California, Berkeley, 1961.

[26] 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. Statist. Comput., 13 (1992), pp. 631–644.

[27] A. Winslow, *Numerical solution of the quasi-linear Poisson equation in a nonuniform triangle mesh*, J. Comput. Phys., 1 (1967), pp. 149–172.