SIAM J. SCI. COMPUT.
Vol. 26, No. 3, pp. 1036–1056

MOVING MESH FINITE ELEMENT METHODS FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS*

YANA DI†, RUO LI†, TAO TANG‡, AND PINGWEN ZHANG†

Abstract. This work presents the first effort in designing a moving mesh algorithm to solve the incompressible Navier–Stokes equations in the primitive variables formulation. The main difficulty in developing this moving mesh scheme is how to keep it divergence-free for the velocity field at each time level. The proposed numerical scheme extends a recent moving grid method based on harmonic mapping [R. Li, T. Tang, and P. W. Zhang, *J. Comput. Phys.*, 170 (2001), pp. 562–588], which decouples the PDE solver and the mesh-moving algorithm. This approach requires interpolating the solution on the newly generated mesh. Designing a divergence-free-preserving interpolation algorithm is the first goal of this work. Selecting suitable monitor functions is important and is found challenging for the incompressible flow simulations, which is the second goal of this study. The performance of the moving mesh scheme is tested on the standard periodic double shear layer problem. No spurious vorticity patterns appear when even fairly coarse grids are used.

Key words. moving mesh method, Navier–Stokes equations, divergence-free-preserving interpolation, incompressible flow

AMS subject classifications. 65M50, 65N22, 76D05

DOI. 10.1137/030600643

1. Introduction.

Many of the modern high resolution methods for solving incompressible flows use the upwind Godunov approach combined with Chorin's projection technique [7]; see, e.g., Bell, Colella, and Glaz [1], Brown and Minion [3], E and Shu [9], LeVeque [20], and Lopez and Shen [25]. Because the Godunov upwinding approach stabilizes the computed flows for cell Reynolds numbers where a strictly centered finite difference scheme would produce spurious oscillations and often instability, these Godunov-type methods enable us to make simulations in situations where it is not possible to carefully resolve the smallest scales everywhere. With currently available computing machines, such underresolution is often unavoidable. It is noted that the Godunov-type methods use exact or approximate Riemann solvers that greatly complicate the upwind algorithms, making them difficult to implement and to generalize to more complex systems. To improve this, Kupferman and Tadmor [26] proposed a second order difference method for incompressible flows. Their method is based on an extension of the classic Lax–Friedrichs scheme introduced for hyperbolic conservation laws [27] and a new discrete Hodge projection. Other state-of-the-art numerical methods for solving incompressible flow problems include the discontinuous Galerkin method (see, e.g., [24]) and high order schemes (see, e.g., [4, 11, 21]).

*Received by the editors August 8, 2003; accepted for publication (in revised form) May 3, 2004; published electronically February 3, 2005.
http://www.siam.org/journals/sisc/26-3/60064.html

†School of Mathematical Sciences, Peking University, 100871, Beijing, People's Republic of China (yndi@math.pku.edu.cn, rli@math.pku.edu.cn, pzhang@math.pku.edu.cn). The research of the first and second authors was supported in part by the Joint Applied Mathematics Research Institute between Peking University and Hong Kong Baptist University. The research of the fourth author was supported in part by the special funds for Major State Research Projects and National Science Foundation of China for Distinguished Young Scholars.

‡Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong (ttang@math.hkbu.edu.hk). The research of this author was supported in part by the Hong Kong Research Grants Council and the International Research Team on Complex System of Chinese Academy of Sciences.

In this work, we study numerical approximations for the incompressible Navier–Stokes equations by using the moving mesh finite element methods. Several moving mesh techniques have been introduced in the past, the most advocated method of which is based on solving elliptic PDEs first proposed by Winslow [36]. Winslow's formulation requires the solution of a nonlinear Poisson-like equation to generate a mapping from a regular domain in a parameter space Ω_c to an irregularly shaped domain in physical space Ω. By connecting points in the physical space corresponding to discrete points in the parameter space, the physical domain can be covered with a computational mesh suitable for the solution of finite difference/element equations. Brackbill and Saltzman [2] formulated the grid equations in a variational form to produce satisfactory mesh concentration while maintaining relatively good smoothness and orthogonality. Their approach has become one of the more popular methods used for mesh generation and adaptation. In [8], Dvinsky suggests that harmonic function theory may provide a general framework for developing useful mesh generators. His method can be viewed as a generalization and extension of Winslow's method. However, unlike most other generalizations which add terms or functionals to the basic Winslow grid generator, his approach uses a single functional to accomplish the adaptive mapping. The critical points of this functional are *harmonic maps.* Meshes obtained by Dvinsky's method enjoy desirable properties of harmonic maps, particularly regularity or smoothness [15, 30].

Motivated by the work of Dvinsky, a moving mesh finite element strategy based on harmonic mapping was proposed and studied by Li, Tang, and Zhang in [22]. The key idea of this strategy is to construct the harmonic map between the physical space and a parameter space by an iteration procedure. This procedure is simple, easy to program, and enables us to keep the map harmonic even after a long time of numerical integration. In our approach, the overall method contains two parts: a solution algorithm and a mesh selection algorithm. These two parts are independent in the sense that the change of the PDEs will affect the first part only.

In this work, using the framework introduced in [22], we develop a moving mesh scheme for solving the incompressible Navier–Stokes equations in the primitive variables formulation. To achieve our goal, the main effort is to design a divergence-free interpolation which is essential for the incompressible problems. By some careful analysis, we conclude that this can be done by solving linearized, inviscid Navier–Stokes-type equations. Some possible choices of the monitor function will be investigated, which are also essential for the incompressible flow computations.

This paper is organized as follows. In section 2, we briefly describe a standard mixed finite element method for solving the Navier–Stokes equations in the primitive variables formulation. In section 3, a moving mesh scheme for solving the Navier–Stokes equations is proposed. Since the monitor functions play important roles in the moving mesh implementation, their possible choices are discussed in section 4. Numerical experiments demonstrating the efficiency of the proposed numerical methods are carried out in section 5. Concluding remarks are given in the final section.

2. A mixed finite element method.

We consider a two-dimensional incompressible Navier–Stokes equation in primitive variables formulation,

$$\begin{cases} \partial_t \mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \nu\Delta\mathbf{u}, & \text{in } \Omega, \\ \nabla\cdot\mathbf{u} = 0, & \text{in } \Omega, \end{cases} \tag{2.1}$$

where $\mathbf{u} = (u, v)$ is the fluid velocity vector, p is the pressure, and ν is the kinematic viscosity. Without loss of generality, let Ω be the unit square $(0, 1) \times (0, 1)$. For ease

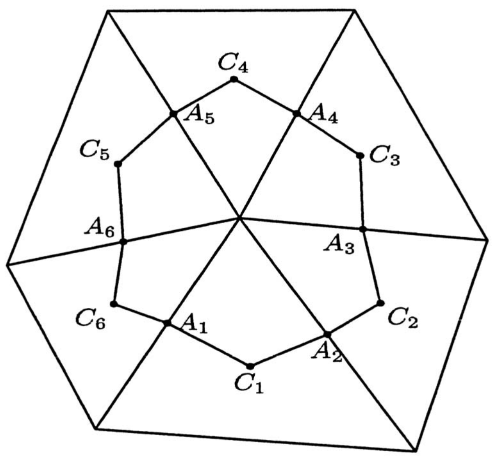

FIG. 1. *A typical element κ_n. Here A_i is the middle point of the corresponding edge, and C_i is the barycenter of the corresponding element.*

of illustration, we consider a well-known periodic double shear layer problem so the following periodic boundary condition is assumed:

$$\mathbf{u}(x,0;t)=\mathbf{u}(x,1;t), \qquad \mathbf{u}(0,y;t)=\mathbf{u}(1,y;t), \tag{2.2a}$$

$$\partial_{\mathbf{n}}\mathbf{u}(x,0;t)=\partial_{\mathbf{n}}\mathbf{u}(x,1;t), \qquad \partial_{\mathbf{n}}\mathbf{u}(0,y;t)=\partial_{\mathbf{n}}\mathbf{u}(1,y;t), \tag{2.2b}$$

$$p(x,0;t)=p(x,1;t), \qquad p(0,y;t)=p(1,y;t). \tag{2.2c}$$

Denote

$$\mathbf{V}=H^1(\Omega)^2\cap\{\mathbf{v}\mid \mathbf{v}\text{ satisfies (2.2a)–(2.2b)}\},$$
$$P=L_0^2(\Omega)\cap\{q\mid q\text{ satisfies (2.2c)}\}.$$

The classical variational formulation for the Navier–Stokes equations (2.1) reads as follows: Find a pair $(\mathbf{u},p)$ in $\mathbf{V}\times P$ such that

$$\begin{cases}(\partial_t\mathbf{u},\mathbf{v})+(\mathbf{u}\cdot\nabla\mathbf{u},\mathbf{v})=(p,\nabla\cdot\mathbf{v})-\nu(\nabla\mathbf{u},\nabla\mathbf{v}) & \forall\mathbf{v}\in\mathbf{V},\\ (q,\nabla\cdot\mathbf{u})=0 & \forall q\in P.\end{cases}\tag{2.3}$$

Assume the domain Ω is triangulated into a triangle mesh $\mathcal{T}_h$ and the elements of the triangulation are denoted as κ; see Figure 1 for a typical mesh setting. Let $\mathbf{V}_h$ and P_h be two finite element spaces with triangulation parameter h such that

$$\mathbf{V}_h\subset\mathbf{V},\qquad P_h\subset P.$$

Then (2.3) can be approximated as follows: Find a pair $(\mathbf{u}_h,p_h)\in\mathbf{V}_h\times P_h$ such that

$$\begin{cases}(\partial_t\mathbf{u}_h,\mathbf{v}_h)+(\mathbf{u}_h\cdot\nabla\mathbf{u}_h,\mathbf{v}_h)=(p_h,\nabla\mathbf{v}_h)-\nu(\nabla\mathbf{u}_h,\nabla\mathbf{v}_h) & \forall\mathbf{v}_h\in\mathbf{V}_h,\\ (q_h,\nabla\cdot\mathbf{u}_h)=0 & \forall q_h\in P_h.\end{cases}\tag{2.4}$$

Let V be the subspace of $H^1(\Omega)$ satisfying the periodic boundary condition, which is a component of space $\mathbf{V}$. Each velocity component is then approximated piecewise linearly on every triangle element, which forms a continuous finite element space $V_h \subset V$. Denote $\mathbf{V}_h = (V_h)^2$. This is an order-one approximation for the velocity. For the pressure p, we adopt the piecewise constant finite element space $P_h \subset P$ on the dual mesh of $\mathcal{T}_h$. Thus, the a priori error estimates of such an approximation are known (see, e.g., [13, 14]),

$$\|\mathbf{u} - \mathbf{u}_h\|_{1,\Omega} + \|p - p_h\|_{0,\Omega} \leq C_1 h (|\mathbf{u}|_{2,\Omega} + |p|_{1,\Omega}). \tag{2.5}$$

Furthermore, if Ω is convex, then the above result can be further improved to

$$\|\mathbf{u} - \mathbf{u}_h\|_{0,\Omega} \leq C_2 h^2 (|\mathbf{u}|_{2,\Omega} + |p|_{1,\Omega}). \tag{2.6}$$

In the temporal direction, a multistep Runge–Kutta scheme will be employed. According to E and Liu [11], for a convection-dominated problem (which is our interest), the order of the Runge–Kutta scheme should be at least three in order to guarantee the numerical stability. In this work, a three-step Runge–Kutta scheme is used, $\forall (\mathbf{v}_h, q_h) \in \mathbf{V}_h \times P_h$:

1. Stage 1:

$$\begin{cases} \left(\dfrac{\mathbf{u}_h^1 - \mathbf{u}_h^{(n)}}{\Delta t/3}, \mathbf{v}_h \right) + (\mathbf{u}_h^{(n)} \cdot \nabla \mathbf{u}_h^{(n)}, \mathbf{v}_h) = (p_h^1, \nabla \mathbf{v}_h) - \nu (\nabla \mathbf{u}_h^1, \nabla \mathbf{v}_h), \\ (\nabla \cdot \mathbf{u}_h^1, q_h) = 0. \end{cases} \tag{2.7}$$

2. Stage 2:

$$\begin{cases} \left(\dfrac{\mathbf{u}_h^2 - \mathbf{u}_h^{(n)}}{\Delta t/2}, \mathbf{v}_h \right) + (\mathbf{u}_h^1 \cdot \nabla \mathbf{u}_h^1, \mathbf{v}_h) = (p_h^2, \nabla \mathbf{v}_h) - \nu (\nabla \mathbf{u}_h^2, \nabla \mathbf{v}_h), \\ (\nabla \cdot \mathbf{u}_h^2, q_h) = 0. \end{cases} \tag{2.8}$$

3. Stage 3:

$$\begin{cases} \left(\dfrac{\mathbf{u}_h^{(n+1)} - \mathbf{u}_h^{(n)}}{\Delta t}, \mathbf{v}_h \right) + (\mathbf{u}_h^2 \cdot \nabla \mathbf{u}_h^2, \mathbf{v}_h) = (p_h^{(n+1)}, \nabla \mathbf{v}_h) - \nu (\nabla \mathbf{u}_h^{(n+1)}, \nabla \mathbf{v}_h), \\ (\nabla \cdot \mathbf{u}_h^{(n+1)}, q_h) = 0. \end{cases} \tag{2.9}$$

It is noted that the above scheme needs only one set of intermediate variables. Moreover, the viscosity terms are treated implicitly and the nonlinear terms are treated explicitly. It is known (see, e.g., [11]) that for problems with very small viscosities (again, as is the case of our interest) an explicit Runge–Kutta treatment for the viscosity terms is acceptable for time marching. However, if there is a thin internal layer, as in our numerical examples, then the mesh needs to be highly adapted to resolve such a layer, which will affect the choice of time step restriction of an explicit method. To be more specific, let us take a one-dimensional viscous Burgers equation with layer width $\mathcal{O}(\epsilon)$ as an example. Roughly speaking, the time-step restriction should satisfy $\Delta t \sim \min\{\Delta t_{\text{CFL}}, \Delta t_{\text{vis}}\}$, where Δt_{CFL} is the standard CFL condition and Δt_{vis} is the viscous time step in the layer regions. The viscous time step is defined by $\Delta t_{\text{vis}} \sim \Delta x^2/\epsilon$, where Δx is the mesh diameter in the layer region. For the viscous Burgers equation with layer size $\mathcal{O}(\epsilon)$, this implies that

$$\Delta x \sim c_1 \epsilon. \tag{2.10}$$

It follows that if the proportional constant c_1 is sufficiently small, then the time-step restriction of an explicit method will be determined by the viscous time step rather than the CFL condition. In moving mesh computations, the proportional constant c_1 in (2.10) may be very small due to the mesh-moving effect. As a result, with an explicit scheme the (very small) viscous time step has to be used. In this case, the methods of handling the extremely small time steps include the use of locally varying time steps (see, e.g., [31]) or an implicit treatment of the viscous terms. In this work, the latter method is employed, which is quite standard in handling the incompressible Navier–Stokes equations; see, e.g., [12, 19].

3. A moving mesh strategy. At time $t = t_{n+1}$, a finite element solution $(\mathbf{u}_h^{(n+1)}, p_h^{(n+1)})$ is obtained using the method described in the last section. Now the question is how to obtain a new mesh $\mathcal{T}_h^{(n+1)}$ using this new solution and the old mesh $\mathcal{T}_h^{(n)}$. To this end, we extend the method proposed in [22, 23] to deal with the incompressible flow problems in this section.

We follow the framework given in [22, 23] but highlight the main differences and difficulties due to the incompressibility constraint. Roughly speaking, the mesh generation scheme consists of the following three steps:

- Step 1: Solve the elliptic system

$$\nabla_{\vec{x}} \left(m \nabla_{\vec{x}} \vec{\xi} \right) = 0 \tag{3.1}$$

together with the boundary conditions

$$\vec{\xi}(0,y) + (1,0)^T = \vec{\xi}(1,y), \qquad \vec{\xi}(x,0) + (0,1)^T = \vec{\xi}(x,1), \tag{3.2a}$$

$$\partial_{\mathbf{n}}\vec{\xi}(0,y) = \partial_{\mathbf{n}}\vec{\xi}(1,y), \qquad \partial_{\mathbf{n}}\vec{\xi}(x,0) = \partial_{\mathbf{n}}\vec{\xi}(x,1), \tag{3.2b}$$

where the function m in (3.1) is a monitor function which is, in general, dependent on the solution $(\mathbf{u}_h^{(n+1)}, p_h^{(n+1)})$. We assume m is given here; its choice is crucial to the moving mesh method and is discussed in section 5.

- Step 2: Denote the initial (fixed uniform) mesh in the logical domain as $\mathcal{T}_c$ (with nodes $\mathcal{A}^{(0)}$) and the new logical mesh obtained by solving (3.1)–(3.2) as $\mathcal{T}_c^*$ (with nodes $\mathcal{A}^*$). Their difference,

$$\delta\mathcal{A} = \mathcal{A}^{(0)} - \mathcal{A}^*, \tag{3.3}$$

is used to determine the displacement δX_i in the physical domain. Then select a suitable ratio-parameter μ, and move the old mesh in the physical domain to a new one by using

$$X_i^{(n+1)} = X_i^{(n)} + \mu \delta X_i. \tag{3.4}$$

- Step 3: Update the numerical solution on the new mesh $X_i^{(n+1)}$ using the solution $(\mathbf{u}_h^{(n+1)}, p_h^{(n+1)})$ and the meshes $X_i^{(n)}, X_i^{(n+1)}$. It is important to remain divergence-free in this updating (interpolation) step.

3.1. Step 1: Obtain new logical mesh. In [22], the elliptic system (3.1) with a Dirichlet boundary condition is solved. Since the present problem has a periodic boundary condition (3.2), we first introduce a coordinate transformation,

$$\vec{\eta} = \vec{\xi} - \vec{x}. \tag{3.5}$$

With this transformation, (3.1) becomes

$$\nabla_{\vec{x}} \left(m \nabla_{\vec{x}} \vec{\eta} \right) = \nabla_x m, \tag{3.6}$$

and the boundary condition (3.2) becomes

$$\vec{\eta}(0, y) = \vec{\eta}(1, y), \qquad \vec{\eta}(x, 0) = \vec{\eta}(x, 1), \tag{3.7a}$$

$$\partial_{\mathbf{n}} \vec{\eta}(0, y) = \partial_{\mathbf{n}} \vec{\eta}(1, y), \qquad \partial_{\mathbf{n}} \vec{\eta}(x, 0) = \partial_{\mathbf{n}} \vec{\eta}(x, 1). \tag{3.7b}$$

The above system can be written in a weak formulation as follows: Find $\vec{\eta} \in \mathbf{V}$ such that

$$\int_\Omega m \nabla_{\vec{x}} \vec{\eta} \nabla_{\vec{x}} \vec{\zeta} d\vec{x} = \int_\Omega m \nabla_{\vec{x}} \vec{\zeta} d\vec{x} \qquad \forall \vec{\zeta} \in \mathbf{V}. \tag{3.8}$$

The solution of this problem is unique subject to a constant vector. The weak formulation for the above problem reads as follows: Find $\vec{\eta}_h \in \mathbf{V}_h$ such that

$$\int_\Omega m \nabla_{\vec{x}} \vec{\eta}_h \nabla_{\vec{x}} \vec{\zeta}_h d\vec{x} = \int_\Omega m \nabla_{\vec{x}} \vec{\zeta}_h d\vec{x} \qquad \forall \zeta_h \in \mathbf{V}_h. \tag{3.9}$$

The solution of the above system is not unique in $\mathbf{V}_h$, which will be handled by removing one row and one column from the resulting matrix and moving the corresponding contributions to the right-hand side of the linear system. This implies that one of node-points of the mesh is kept fixed. This fixed point can be chosen arbitrarily. In our computation, it is chosen randomly. The resulting linear system is solved using a Bi-CGSTAB solver [35] preconditioned with an incomplete LU decomposition [29].

After obtaining the solution of (3.9), the numerical solution for (3.1)–(3.2) can be obtained using the transformation (3.5): $\vec{\xi}_h = \vec{\eta}_h + \vec{x}_h$.

3.2. Step 2: Mesh-motion in physical domain. We first introduce some notation. The triangulation of the physical domain is $\mathcal{T}_h$. The ith node is denoted by X_i, and the set of elements containing the ith node is denoted by T_i. The corresponding notations on the computational domain are $\mathcal{T}_c$, $\mathcal{A}_i$, and $T_{i,c}$, respectively. The coordinates of the nodes $\mathcal{A}_i$ in the computational domain are denoted by $(\mathcal{A}_i^1, \mathcal{A}_i^2)^T$. After finishing Step 1 in the last subsection, a new logical mesh $\mathcal{T}_c^*$ with nodes $\mathcal{A}_i^*$ is obtained. For a given element $E \in \mathcal{T}_h$, with X_{E_k}, $0 \le k \le 2$, as its vertices, the piecewise linear map from $V_{\mathcal{T}_c^*}(\Omega_c)$ to $V_{\mathcal{T}}(\Omega)$ has constant gradient on E and satisfies the following linear system:

$$\begin{aligned} &\begin{pmatrix} \mathcal{A}_{E_1}^{*,1} - \mathcal{A}_{E_0}^{*,1} & \mathcal{A}_{E_2}^{*,1} - \mathcal{A}_{E_0}^{*,1} \\ \mathcal{A}_{E_1}^{*,2} - \mathcal{A}_{E_0}^{*,2} & \mathcal{A}_{E_2}^{*,2} - \mathcal{A}_{E_0}^{*,2} \end{pmatrix} \begin{pmatrix} \dfrac{\partial x^1}{\partial \xi^1} & \dfrac{\partial x^1}{\partial \xi^2} \\ \dfrac{\partial x^2}{\partial \xi^1} & \dfrac{\partial x^2}{\partial \xi^2} \end{pmatrix} \\ &= \begin{pmatrix} X_{E_1}^1 - X_{E_0}^1 & X_{E_2}^1 - X_{E_0}^1 \\ X_{E_1}^2 - X_{E_0}^2 & X_{E_2}^2 - X_{E_0}^2 \end{pmatrix}. \end{aligned} \tag{3.10}$$

Solving the above linear system gives $\partial\vec{x}/\partial\xi$ in E. If we take the volume of the element as the weight, the weighted average displacement of X at the ith node is defined by

$$\delta X_i = \frac{\displaystyle\sum_{E\in T_i} |E| \left.\frac{\partial\vec{x}}{\partial\xi}\right|_{in\ E} \delta\mathcal{A}_i}{\displaystyle\sum_{E\in T_i} |E|}, \tag{3.11}$$

where $|E|$ is the volume of the element E, and $\delta\mathcal{A} = \mathcal{A}^{(0)} - \mathcal{A}^*$ is the difference between the fixed mesh $\mathcal{T}_c$ (with nodes $\mathcal{A}^{(0)}$) and the logical mesh $\mathcal{T}_c^*$ (with nodes $\mathcal{A}^*$). We point out that once the initial mesh (in the logical domain) is given, it will be kept *unchanged* throughout the computation. In other words, the initial mesh in the logical domain Ω_c is used as a reference mesh.

To avoid mesh tangling, a scale-parameter μ is used so that the nodes in the new mesh $\mathcal{T}^*$ on the physical domain are taken as

$$X_i^* = X_i + \mu\delta X_i. \tag{3.12}$$

A simple method for choosing μ was proposed in [22, 23]. Here we give a more accurate alternative. Given a triangle element E_i with vertex coordinates $\vec{x}_0$, $\vec{x}_1$, and $\vec{x}_2$, as well as the corresponding moving vectors $\delta\vec{x_0}$, $\delta\vec{x_1}$, and $\delta\vec{x_2}$ given by (3.11), let the minimal positive root of the quadratic equation

$$\det\begin{pmatrix} 1 & 1 & 1 \\ \vec{x}_0 + \mu\delta\vec{x}_0 & \vec{x}_1 + \mu\delta\vec{x}_1 & \vec{x}_2 + \mu\delta\vec{x}_2 \end{pmatrix} = 0 \tag{3.13}$$

be μ_i^*. Then we set

$$\mu = \min(1, \mu_i^*/2). \tag{3.14}$$

It is obvious that such a choice of the scale-parameter can prevent mesh tangling.

Another issue for the mesh-motion in the physical domain is the boundary mesh redistribution. With a periodic boundary condition, the physical domain does not need to be fixed anymore (although the logical domain is kept fixed). We still use (3.12) to perform the boundary mesh redistribution, but special care has to be taken in computing δX_i in (3.11) when X_i belongs to the boundary. In this case, the neighboring points include not only those elements next to X_i but also those mirror points on the corresponding opposite boundary; i.e., the periodic structure of the physical domain should be reflected here. This procedure will keep the physical domain periodic, as will be seen in Figure 5.

3.3. Step 3: Divergence-free interpolation. In using the moving mesh method for incompressible flow simulations, it is essential to remain divergence-free in the interpolation step. Below we propose a divergence-free-preserving interpolation scheme, which is obtained by solving a simple convection equation whose convection speed is the same as the mesh-moving speed. Assume that a finite element solution u_h in a finite element space W_h is $u_h = u_i\phi^i$, where the standard summation convention is used and ϕ^i is the basis function of W_h. We introduce a virtual time variable τ and assume that in the mesh-moving process the basis function ϕ^i and the pointwise value u_i both depend on the virtual time variable τ, i.e., $\phi^i = \phi^i(\vec{x}, \tau)$, $u_i = u_i(\tau)$. To be more precise, we introduce a continuous transformation

$$x(\tau) = x^{\text{old}} + \tau(x^{\text{new}} - x^{\text{old}}), \qquad \tau \in [0, 1], \tag{3.15}$$

where $x^{\rm old}$ and $x^{\rm new}$ are two sets of coordinates in the physical domain, which in the discrete level satisfy $x_i^{\rm old} = X_i$ and $x_i^{\rm new} = X_i^*$. In particular, the change for the discrete nodes is given by

$$x_i(\tau) = X_i + \tau(X_i^* - X_i), \qquad \tau \in [0,1]. \tag{3.16}$$

With the linear transformation (3.15), the corresponding basis function and the pointwise value can be defined by $\phi^i(\tau) := \phi^i(x(\tau))$ and $u_i(\tau) := u_i(x(\tau))$.

Assume the solution curve $u_h = \phi^i(\tau)u_i(\tau)$, $\tau \in [0,1]$, is independent of τ; namely, it is unchanged in the mesh-moving process. This assumption leads to

$$\partial_\tau u_h = 0. \tag{3.17}$$

By direct computation we obtain

$$\frac{\partial \phi^i}{\partial \tau} = -\nabla_{\vec{x}} \phi^i \cdot \delta\vec{x}, \tag{3.18}$$

where $\delta\vec{x} := x^{\rm new} - x^{\rm old}$, which is well defined in the discrete level. It follows from the above two equations that $\forall \psi \in W_h$

$$\begin{aligned} 0 &= (\partial_\tau u_h,\, \psi) \\ &= (\phi^i \partial_\tau u_i + u_i \partial_\tau \phi^i,\, \psi) \\ &= (\phi^i \partial_\tau u_i - u_i \nabla_{\vec{x}} \phi^i \cdot \delta\vec{x},\, \psi) \\ &= (\partial_\tau u_h - \nabla_{\vec{x}} u_h \cdot \delta\vec{x},\, \psi). \end{aligned} \tag{3.19}$$

It is seen that (3.19) is in fact the discretization of the convection problem

$$\frac{\partial u}{\partial \tau} - \nabla u \cdot \delta\vec{x} = 0 \qquad \text{for} \quad u \in W_h. \tag{3.20}$$

We now apply this formulation to the velocity field of an incompressible flow by letting W_h be the divergence-free space

$$\mathbf{W}_h = \mathbf{V}_h \cap \{\mathbf{u}_h \mid \nabla \cdot \mathbf{u}_h = 0\}. \tag{3.21}$$

Then (3.19) becomes the following: Find $\mathbf{w}_h \in \mathbf{W}_h$ such that

$$(\partial_\tau \mathbf{w}_h - \nabla_{\vec{x}} \mathbf{w}_h \cdot \delta\vec{x},\, \mathbf{z}_h) = 0 \qquad \forall \mathbf{z}_h \in \mathbf{W}_h. \tag{3.22}$$

The above result implies that

$$\partial_\tau \mathbf{w}_h - \nabla_{\vec{x}} \mathbf{w}_h \cdot \delta\vec{x} \in {\mathbf{W}_h}^\perp, \tag{3.23}$$

where the right-hand side of the above equation denotes the orthogonal space of $\mathbf{W}_h$ in L^2. It follows from Theorem 2.7 of [13] that if the solution domain Ω is simply connected, then

$$ {\mathbf{W}_h}^\perp = \{\nabla q \mid q \in H^1(\Omega)\}. \tag{3.24}$$

Using the above two results, we can show that solving (3.22) is equivalent to finding $(\mathbf{u}_h, p_h) \in \mathbf{V}_h \times P_h$ such that

$$(\partial_\tau \mathbf{u}_h - \nabla_{\vec{x}} \mathbf{u}_h \cdot \delta\vec{x},\, \mathbf{v}_h) = (p_h,\, \nabla \mathbf{v}_h) \quad \forall \mathbf{v}_h \in \mathbf{V}_h, \tag{3.25a}$$

$$(\nabla_{\vec{x}} \cdot \mathbf{u},\, q_h) = 0 \quad \forall q_h \in P_h. \tag{3.25b}$$

It can be further concluded that solving (3.25) is equivalent to solving the following system:

$$\frac{\partial \mathbf{u}}{\partial \tau} - \nabla_{\vec{x}} \mathbf{u} \cdot \delta \vec{x} = -\nabla p, \tag{3.26a}$$

$$\nabla_{\vec{x}} \cdot \mathbf{u} = 0, \tag{3.26b}$$

in some appropriate space. In practice, any appropriate numerical methods for solving (3.26) can be used to realize the solution redistribution, although (3.25) is the most straightforward approach in the present setting. The initial value for (3.25) and (3.26) is the Navier–Stokes solutions at $t = t_{n+1}$ obtained by using the mesh at $t = t_n$.

Again, a three-step Runge–Kutta scheme similar to (2.7)–(2.9) is applied for the temporal discretization:

1. Stage 1:

$$\begin{cases} \left(\dfrac{\mathbf{u}_h^1 - \mathbf{u}_h^{(n)}}{\Delta\tau/3}, \mathbf{v}_h \right) - (\delta\vec{x} \cdot \nabla \mathbf{u}_h^{(n)}, \mathbf{v}_h) = (p_h^1, \nabla \mathbf{v}_h), \\ (\nabla \cdot \mathbf{u}_h^1, q_h) = 0, \end{cases} \tag{3.27}$$

2. Stage 2:

$$\begin{cases} \left(\dfrac{\mathbf{u}_h^2 - \mathbf{u}_h^{(n)}}{\Delta\tau/2}, \mathbf{v}_h \right) - (\delta\vec{x} \cdot \nabla \mathbf{u}_h^1, \mathbf{v}_h) = (p_h^2, \nabla \mathbf{v}_h), \\ (\nabla \cdot \mathbf{u}_h^2, q_h) = 0, \end{cases} \tag{3.28}$$

3. Stage 3:

$$\begin{cases} \left(\dfrac{\mathbf{u}_{h,*}^{(n)} - \mathbf{u}_h^{(n)}}{\Delta\tau}, \mathbf{v}_h \right) - (\delta\vec{x} \cdot \nabla \mathbf{u}_h^2, \mathbf{v}_h) = (p_{h,*}^{(n)}, \nabla \mathbf{v}_h), \\ (\nabla \cdot \mathbf{u}_{h,*}^{(n)}, q_h) = 0, \end{cases} \tag{3.29}$$

where $\mathbf{u}_h^{(n)}$ and p_h are the numerical solution of the Navier–Stokes equations at $t = t_{n+1}$ obtained using the mesh at $t = t_n$, and $\mathbf{u}_{h,*}^{(n)}$ and $p_{h,*}^{(n)}$ are the desired updated solution at $t = t_{n+1}$ on the new mesh. Again, the periodic boundary condition is applied to the above scheme. In our computations, the virtual time step $\Delta\tau$ is taken as 1. In other words, we only use one marching step to realize the solution redistribution. The reason for allowing the large time step is that the convection speed in (3.25) or (3.26), namely $\delta\vec{x}$, is very small. The speed for most of the nodes is as small as $\mathcal{O}(h)$.

3.4. Spatial smoothing. In practice, it is common to use some temporal or spatial *smoothing* on the monitor function or directly on the mesh to obtain smoother meshes. One of the reasons for doing this is to avoid producing very singular meshes. Several smoothing techniques have been proposed before. In this work, the smoothing technique proposed in [22] is used.

- First, we interpolate the monitor function m from $L^2(\Omega)$ into $H_{1,h}(\Omega)$, namely from piecewise constant to piecewise linear, by the following formula:

$$(\pi_h m)|_{\text{at } P} = \frac{\displaystyle\sum_{P \text{ is vertex of } e} m|_e \; |e|}{\displaystyle\sum_{P \text{ is vertex of } e} |e|}. \tag{3.30}$$

- Second, we project it back into $L^2(\Omega)$ by the following formula:

$$m|_{\text{on } e} = \frac{1}{d+1} \sum_{P \text{ is vertex of } e} (\pi_h m)_{\text{at } P}, \tag{3.31}$$

  where d is the dimension of Ω.

Our numerical experiments have shown that the total number of spatial smoothings is proportional to the total number of elements used. For example, with a 40×40 mesh, 2–4 smoothings have to be used, while for an 80×80 mesh, 5–8 smoothings are needed. One criterion used to determine these numbers is to require that the maximum value of the monitor function is less than a given (large) number at each time step, which can be achieved by several spatial smoothings.

4. Monitor functions. It is very important to choose a suitable monitor function; otherwise satisfactory adaptations cannot be obtained no matter how good a moving mesh algorithm is. There are several possible choices of the monitor function for the incompressible Navier–Stokes approximations. Let

$$m = 1/G,$$

where the scalar function m is used in (3.1). In previous computations [5, 6], there have been several suggestions for the choice of G. One is based on the vorticity

$$G_0 = \sqrt{1 + \alpha|\omega|^\beta}, \tag{4.1}$$

where $\omega = \nabla \times \mathbf{u}$ is the vorticity and $\alpha > 0$ and $\beta > 0$ are user-defined constants. In [5], it was demonstrated via numerical experiments that $\beta = 4$ gives good adaptation results. We point out that in many cases the monitor functions involve some user-defined parameters which have to be determined experimentally. This is the case for the choice of $\beta = 4$ in [5] as well as for all of the numerical examples in the next section. Obviously, some theoretical study on how to choose the parameters (or the general forms of the monitor function) would be very useful. There have been some efforts in this direction recently. For example, Huang and Sun [17] proposed two types of monitor functions based on the asymptotic estimates of interpolation errors. Their work seems useful in developing parameter-free monitor functions. It is obvious that having more robust parameter-free monitors can make the moving mesh approach more attractive.

Another natural choice for G is based on the gradient of the solution variables

$$G_1(q) = \sqrt{1 + \alpha|\nabla q|^\beta}, \tag{4.2}$$

where q can be density for the Boussinesq equations [6, 10] or velocity for the gas dynamics problems [32, 33]. For the test problems proposed by Brown and Minion [3], considered in our numerical experiment section, the first velocity component u is discontinuous. Therefore, it is natural to use the gradient of u in the monitor function. However, as demonstrated at the end of section 5, although some desired adaptation effects can be observed, the overall performance using the monitor $G_1(u)$ is not satisfactory.

For a piecewise linear approximation v_h to a function v, the following formal a posteriori formula is adopted to approximate the computational error (see, e.g., [28]):

$$|v - v_h|_{1,\Omega} \backsim \eta(v_h) := \sqrt{\sum_{l:\ \text{interior edge}} \int_l [\nabla v_h \cdot \mathbf{n}_l]|_l{}^2 dl},$$

where $[\,\cdot\,]|_l$ denotes the jump along the edge l,

$$[\mathbf{v}]|_l = \mathbf{v}|_{l^-} - \mathbf{v}|_{l^+}.$$

It is natural to equally redistribute the numerical errors $\eta(v_h)$ in each element, which can be done by choosing the monitor function G as

$$G_2(v_h) = \sqrt{1 + \alpha\eta^2(v_h)}. \tag{4.3}$$

It is found in the numerical computations that the error η is very small in most parts of the solution domain, which makes the choice of the parameter α difficult. To overcome this difficulty, we use a scaling and a larger power $\beta > 2$ in the monitor

$$G_3(v_h) = \sqrt{1 + \alpha\Big[\eta(v_h)/\max\eta(v_h)\Big]^{\beta}}. \tag{4.4}$$

The above monitor has been found appropriate in our numerical experiments, which can handle not only the so-called thick layer problems but also the thin layer problems.

5. Numerical results.

5.1. Accuracy test.

Example 5.1. The Navier–Stokes equations (2.1) are defined in the box $[0, 2\pi] \times [0, 2\pi]$ with periodic boundary conditions on both directions. The initial conditions

$$u(x, y, 0) = -\cos(x)\sin(y), \qquad v(x, y, 0) = \sin(x)\cos(y) \tag{5.1}$$

give the following exact solutions for the velocity field:

$$u(x, y, t) = -\cos(x)\sin(y)e^{-2\nu t}, \qquad v(x, y, t) = \sin(x)\cos(y)e^{-2\nu t}. \tag{5.2}$$

This example is used to check the accuracy of our moving mesh method when the underling solutions are smooth. By considering the error estimates (2.5) and (2.6) which hold for static (uniform) mesh computation, it is hoped that the errors for a moving mesh algorithm are of the same convergence rate, namely, second order for the velocity and first order for the pressure.

The moving mesh scheme described in the last section is applied to this test problem. The monitor function used is the gradient-based monitor (4.2), where q is chosen as the velocity vector and the parameters (α, β) are chosen as 5 and 2, respectively. The l_2-errors for the numerical velocity and vorticity with $\nu = 0.05$ are listed in Table 1. It is observed that a second order accuracy for velocity and first order accuracy for the vorticity are obtained. Since the divergence-free requirement is important in the computation, its errors and convergence rate are also listed in Table 1. It is seen that a first order rate of convergence is achieved for the velocity divergence.

It is noted that the exact solution (5.2) is time separable, and as a result the moving mesh is time independent. Namely, the initial uniform mesh will be moved to a nonuniform one at $t = 0$, which will remain almost the same at the later time. In Figure 2, the streamline and the moving mesh are plotted at $t = 1$. It is seen that this mesh is different with the initial uniform mesh.

TABLE 1
Example 5.1: *Accuracy check for the moving mesh solutions.*

Accuracy of velocity						
Mesh	l_2-error ($t = 1$)	Order	l_2-error ($t = 2$)	Order	l_2-error ($t = 3$)	Order
20×20	1.16e-3		1.63e-3		1.78e-3	
40×40	3.85e-4	2.02	4.13e-4	1.98	4.60e-4	1.96
80×80	8.74e-5	2.14	1.04e-4	1.98	1.19e-4	1.96

Accuracy of vorticity						
Mesh	l_2-error ($t = 1$)	Order	l_2-error ($t = 2$)	Order	l_2-error ($t = 3$)	Order
20×20	1.66e-1		1.50e-1		1.36e-1	
40×40	8.40e-2	0.98	7.60e-2	0.98	6.88e-2	0.99
80×80	4.22e-2	0.99	3.82e-2	0.99	3.45e-2	0.99

Accuracy of divergence						
Mesh	l_2-error ($t = 1$)	Order	l_2-error ($t = 2$)	Order	l_2-error ($t = 3$)	Order
20×20	4.77e-2		4.33e-2		3.93e-2	
40×40	2.31e-2	1.04	2.01e-2	1.05	1.89e-2	1.05
80×80	1.10e-2	1.07	9.97e-3	1.07	9.01e-3	1.07

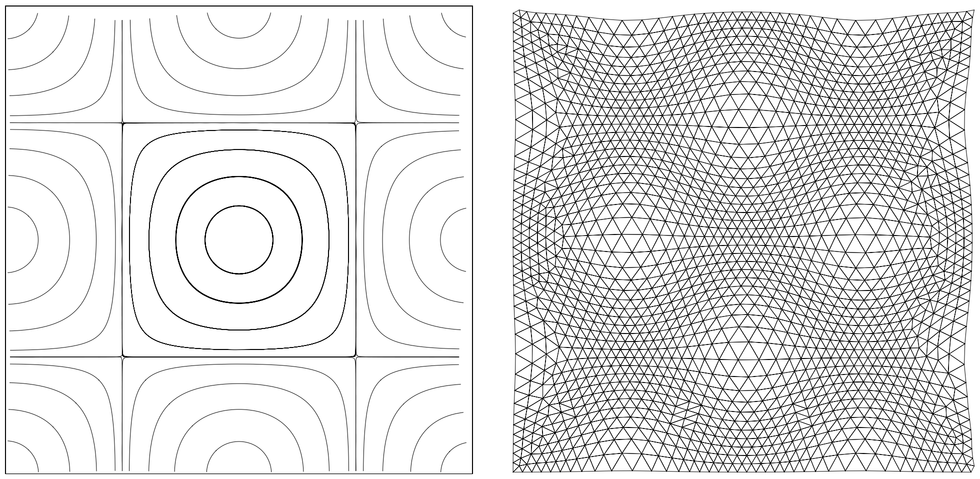

FIG. 2. *Example* 5.1: *Streamfunction (left) and mesh (right) at* $t = 1$.

5.2. Double shear flow.

Example 5.2. Consider a double shear layer governed by the Navier–Stokes equations (2.1), in a unit one periodic domain, subject to the initial conditions

$$u_0(x,y) = \begin{cases} \tanh(\rho(y-0.25)) & \text{for } y \le 0.5, \\ \tanh(\rho(0.75-y)) & \text{for } y > 0.5, \end{cases} \tag{5.3}$$
$$v_0(x,y) = \delta \sin(2\pi x).$$

This problem is a canonical test problem for a scheme's accuracy and resolution in incompressible flows. Brown and Minion [3] performed a systematic comparison for this problem between a number of schemes, concentrating on the effect of underresolution. They demonstrated that a Godunov-projection method performs as well as an accurate central difference method in cases where the smallest flow scales are well resolved. However, in underresolved cases where centered methods compute solutions badly polluted with mesh-scale oscillations, the Godunov-projection method sometimes gives smooth, apparently physical solutions. It is found in [3] that these underresolved solutions, although convergent when the grid is refined, contain spurious *nonphysical* vortices that are artifacts of the underresolution. It seems necessary that powerful numerical methods have to be used to resolve the smallest flow scales, which can be done by increasing the accuracy order of the numerical method (see, e.g., [11, 24]) or by using an adaptive grid method. The goal of adaptive grid methods is to resolve small flow scale by clustering more grid points in the smallest scale areas.

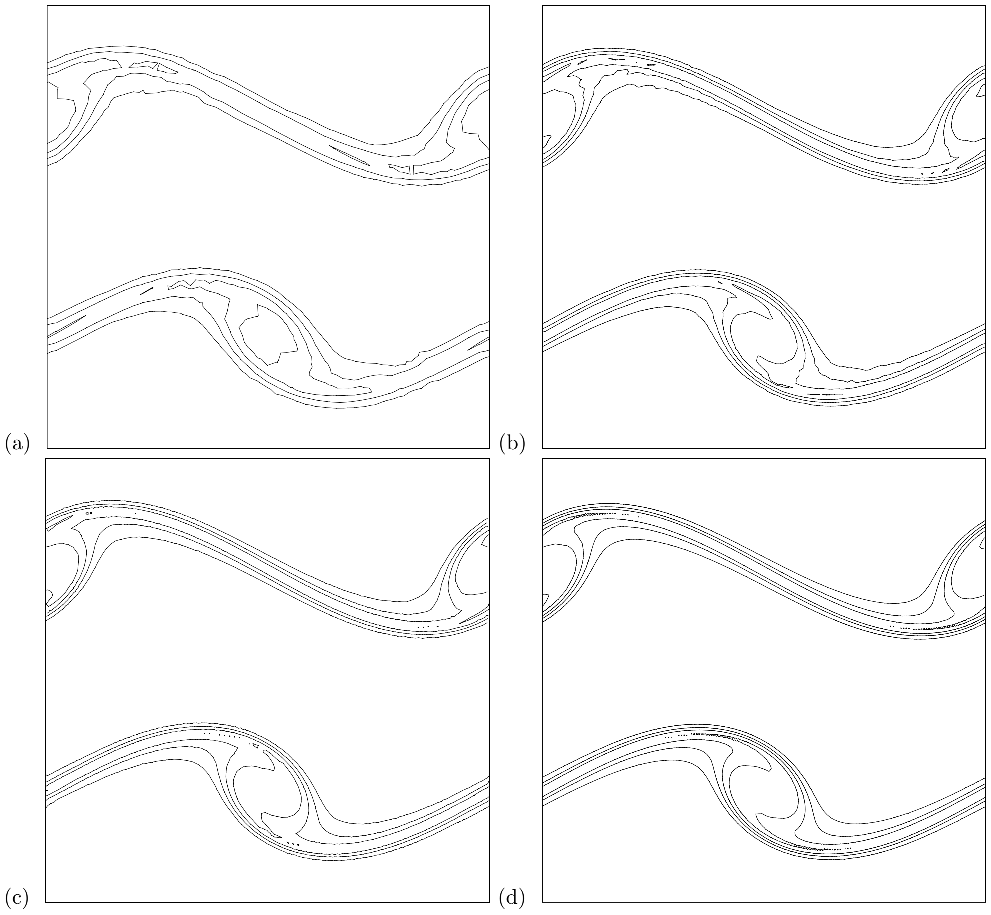

FIG. 3. *Thick shear layer problem: Vorticity contours at $t = 0.8$, obtained by using a 40×40 mesh (left) and an 80×80 mesh (right). Top: Static mesh results. Bottom: Moving mesh results. Layer width parameter $\rho = 30$; viscosity $\nu = 1/10{,}000$.*

In (5.3), the parameter ρ determines the slope of the shear layer, and δ represents the size of the perturbation. The initial layer rolls up in time into strong vortical structures. In our computations, the perturbation size used is $\delta = 0.05$, but the shear layer width is varied in order to study the effect of the layer resolution on the computations.

5.2.1. Thick layer problem. As in [3], the double layer shear is called a *thick layer problem* when $\rho = 30$ and $\nu = 10^{-4}$. In this computation, we use the monitor function of the form (4.3). Two meshes are used: a 40×40 mesh and an 80×80 mesh.

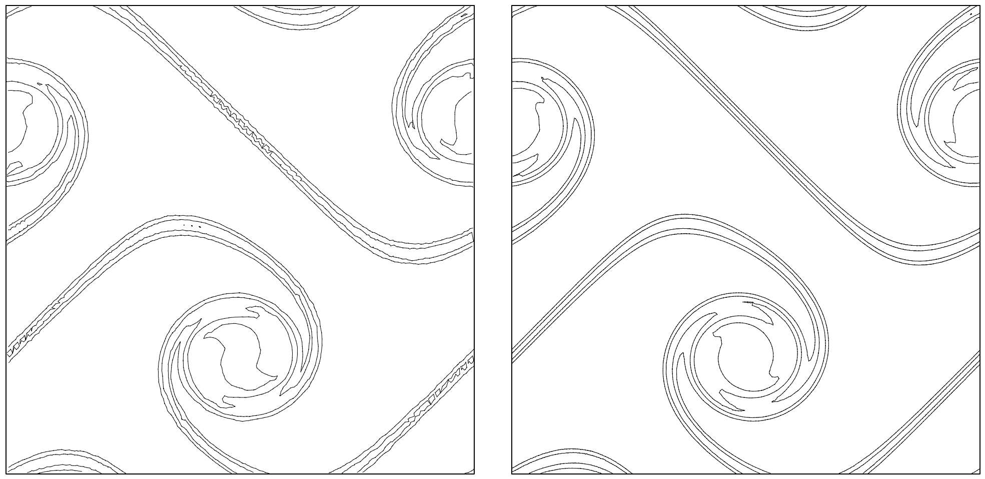

FIG. 4. *Vorticity contours for the thick shear layer at* $t = 1.2$*, obtained by using a* 40×40 *moving mesh (left) and an* 80×80 *moving mesh (right). Layer width parameter* $\rho = 30$*; viscosity* $\nu = 1/10{,}000$.

On both meshes, the parameter α is chosen as 10^4. In fact, it is not sensitive as long as it is about $\mathcal{O}(10^4)$. The large value of α can be reduced to $\mathcal{O}(10)$ when a scaling factor is used, as seen in (4.4). For example, with the 40×40 mesh, the feasible parameters of (α, β) are $(80, 2)$, and with the 80×80 mesh, the feasible parameters are $(40, 2)$. The moving mesh results with the monitor functions (4.3) and (4.4) are found in good agreement.

Figure 5 shows the adaptive meshes for the thick shear layer with 40×40 and 80×80 meshes at $t = 0.8$ and 1.2. It is expected that the meshes obtained by a good moving mesh method can represent some useful features of the numerical solutions. This is indeed the case by comparing Figures 3, 4, and 5. It also can be clearly observed that more grid points have been clustered in the regions where the solutions vary rapidly.

The layout of the mesh plots is quite interesting. Since the solution of our test problem is periodic, the moving meshes are also generated in the periodic setting; see section 3.2. Using the periodic natures, both the mesh and the corresponding solutions (velocity, vorticity, etc.) can be easily mapped back to the unit domain. It is noted that with periodic boundary conditions Brackbill and Saltzman [2] also obtained an "irregular" mesh layout very similar to those in Figure 5.

In closing this subsection, we demonstrate the convergence rate of our finite element method described in section 2 for the thick shear layer problem. Since the exact solution is not known, the errors are estimated by using the Richardson extrapolation method on the uniform mesh. This technique was proposed by [3]. To make the comparison meaningful, the same parameters of (α, β) are used in the monitor function (4.4), namely, $(\alpha, \beta) = (40, 2)$. Table 2 shows the convergence rate estimates for the velocities, and as expected the order of convergence is two.

5.2.2. Thin layer problem. We now consider Example 5.2 with a thinner shear layer: $\rho = 100$ and $\nu = 1/20{,}000$. It was demonstrated in [3] that this is a challenging problem. If the mesh is not fine enough, then spurious vortices will be generated in the numerical solutions. It was concluded in [3] that with the second order Godunov-

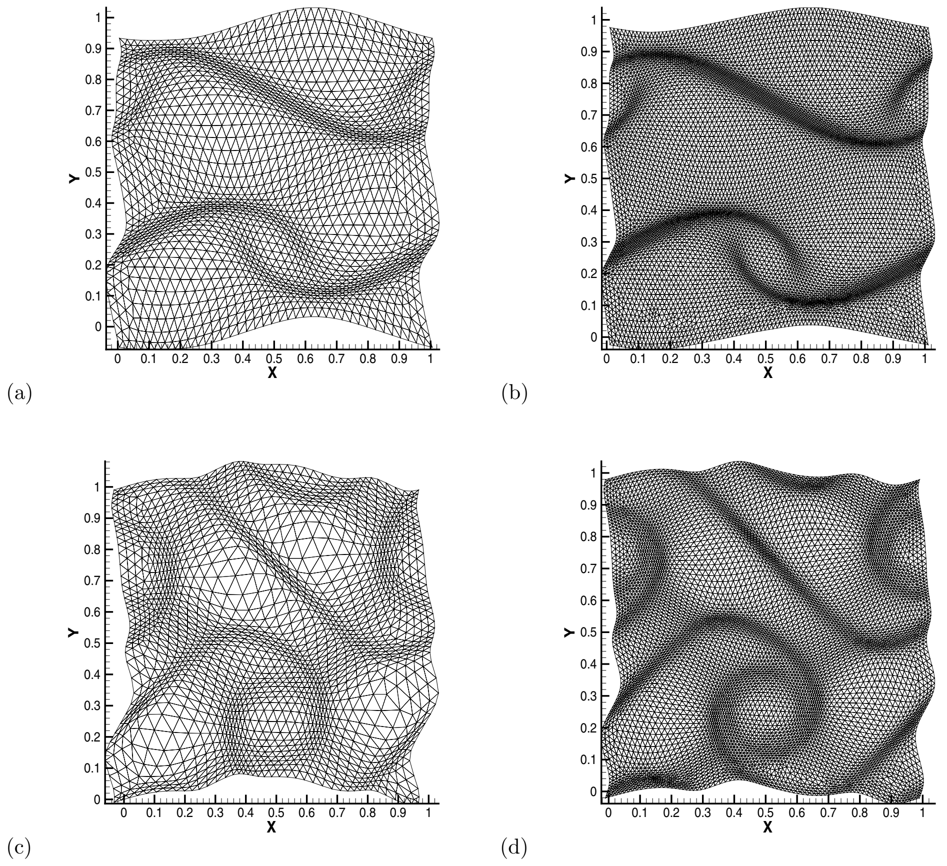

FIG. 5. *Thick shear layer problem: Adaptive meshes* 40×40 *(left) and* 80×80 *(right) mesh resolution. Top:* $t = 0.8$. *Bottom:* $t = 1.2$.

projection methods about 512×512 cells have to be used in order to obtain reasonable approximations.

In the thin layer calculations, we tested the monitor functions (4.3) and (4.4). Our goal is to use about 150×150 cells to resolve the thin layer problem. However, it is found that the monitor (4.3) does not perform well to achieve this goal, while the monitor (4.4) works well. Three mesh resolutions are used: 80×80, 100×100, and 160×160. The parameters (α, β) in (4.4) used for the three meshes are $(5, 2)$, $(5, 3)$, and $(8, 4)$, respectively.

Figure 6 shows the vorticity contours at $t = 0.8$ computed with the moving mesh method on the three mesh resolutions. For comparison, a uniform mesh solution with a 160×160 mesh is also included. It is clear that on the coarser grid the appearance of the moving mesh solution is distinct from the finest mesh reference solutions given in [3]. The 80×80 moving mesh solution and the 160×160 uniform mesh solution both give spurious vortices, developing additional roll-ups in the shear layer. This is clearly the underresolution effect. However, when the mesh is refined, the spurious vortices disappear and the numerical solutions converge to a double shear layer with

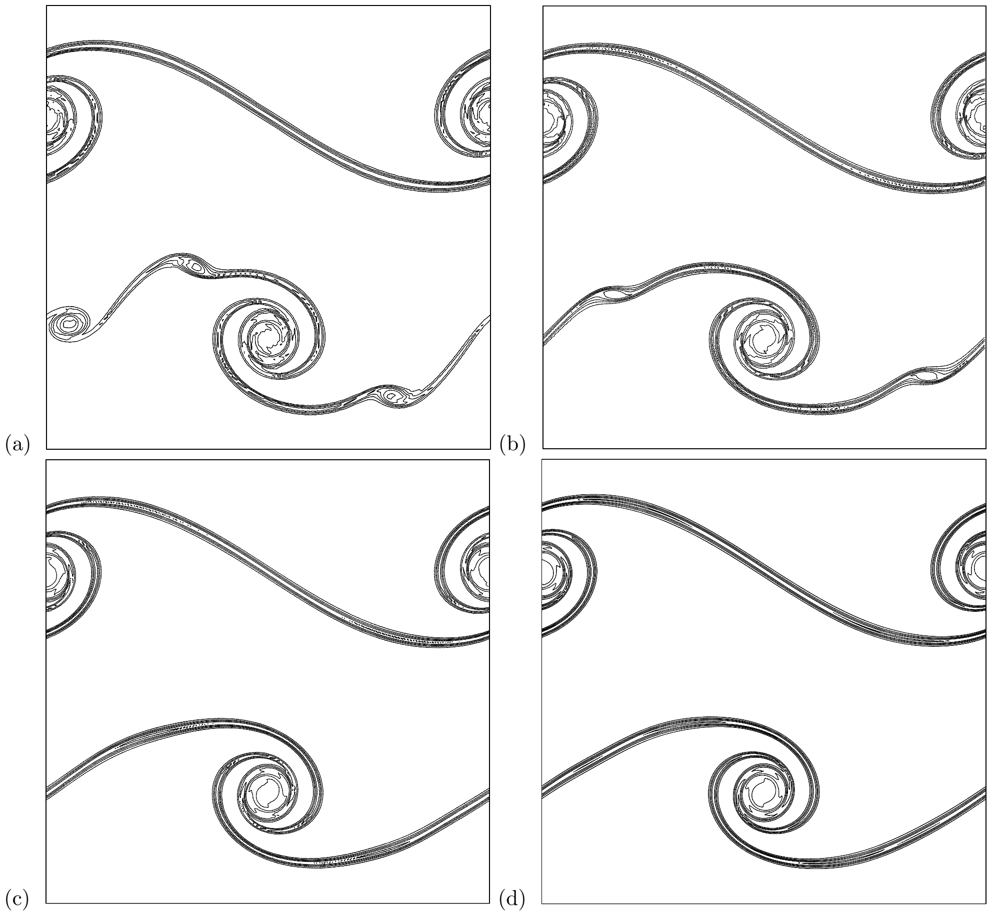

Fig. 6. *Thin shear layer problem: Vorticity contours at* $t = 0.8$ *obtained by using moving mesh methods with resolution* (b): 80×80, (c): 100×100, *and* (d): 160×160. *A* 160×160 *static mesh solution is included in* (a). *Layer width parameter* $\rho = 100$; *viscosity* $\nu = 1/20{,}000$. *Contours are from* -70 *to* 70 *by* 10.

a single roll-up, as shown in Figures 6 (c) and (d).

The meshes generated by the moving mesh methods at two different times are shown in Figure 7. Again it is observed that more grid points have been clustered inside the shear layer and the roll-ups where the solutions have large solution variations. Figure 8 shows the velocity-divergence at $t = 0.8$ obtained with a 160×160 mesh. As expected, the divergence-free requirement is satisfied away from the areas with large solution variation, and the overall divergence-free property is kept quite small.

Figure 9 shows the vorticity contours at a larger time, $t = 1.2$, with two mesh resolutions. The convergence of the moving mesh solutions is observed, and the solutions can be well compared with the finest mesh solutions in [3].

5.2.3. More on monitor functions. Before closing this section we make some comments on the other two possible monitors discussed in section 4, namely, the vorticity monitor (4.1) and the gradient-based monitor (4.2). Our numerical experiments

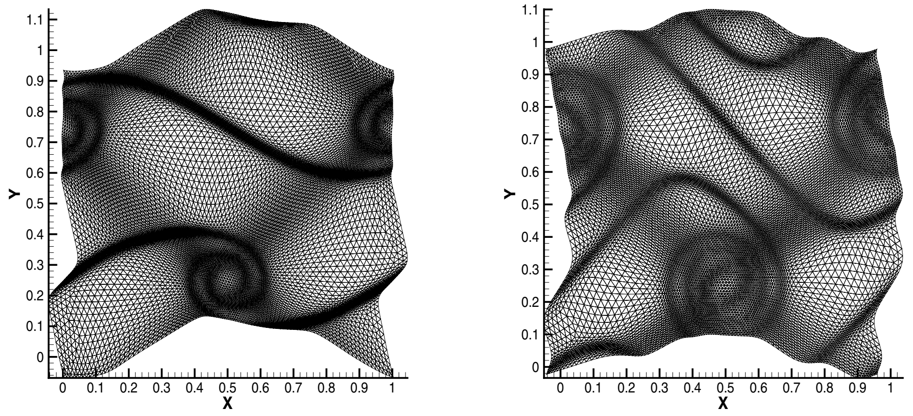

Fig. 7. *Thin shear layer problem: The mesh at* $t = 0.8$ *(left) and* $t = 1.2$ *(right) with a* 100×100 *mesh. Layer width parameter* $\rho = 100$*; viscosity* $\nu = 1/20{,}000$.

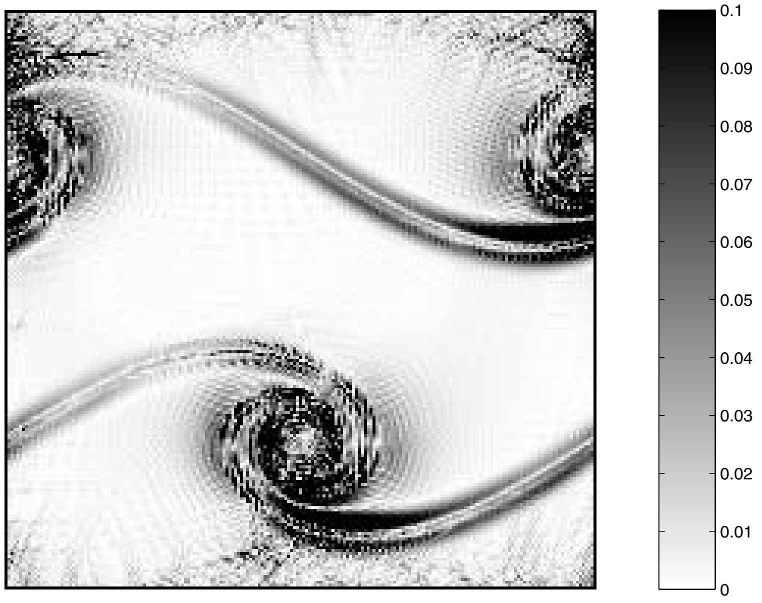

Fig. 8. *Thin shear layer problem: The divergence of the velocity field at* $t = 0.8$ *with a* 160×160 *mesh. Layer width parameter* $\rho = 100$*; viscosity* $\nu = 1/20{,}000$.

have shown that both of them are less satisfactory. Although they can improve the uniform mesh solutions, the improvement is not as significant as with the use of (4.3) and (4.4). To show an example, the thick layer problem is computed at $t = 0.8$ with a 40×40 mesh using the velocity-gradient monitor $G_1(u)$. The corresponding mesh and vorticity are shown in Figure 10. The moving mesh effect in this figure is clearly less satisfactory compared with that in Figure 3(c) (the solution) and Figure 5(a) (the mesh).

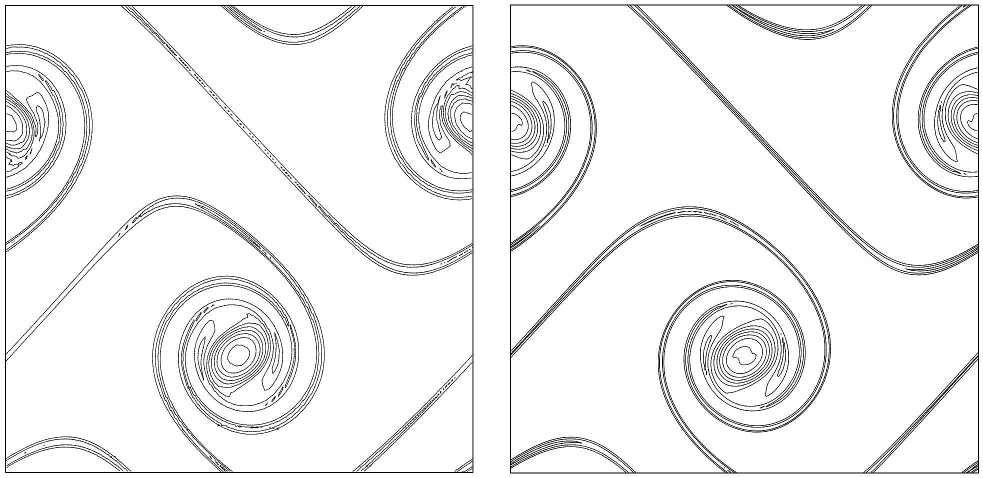

FIG. 9. *Thin shear layer problem: Vorticity contours at* $t = 1.2$ *with resolution* 100×100 *(left) and* 160×160 *(right). Layer width parameter* $\rho = 100$*; viscosity* $\nu = 1/20{,}000$*. Contours are from* -70 *to* 70 *by* 10.

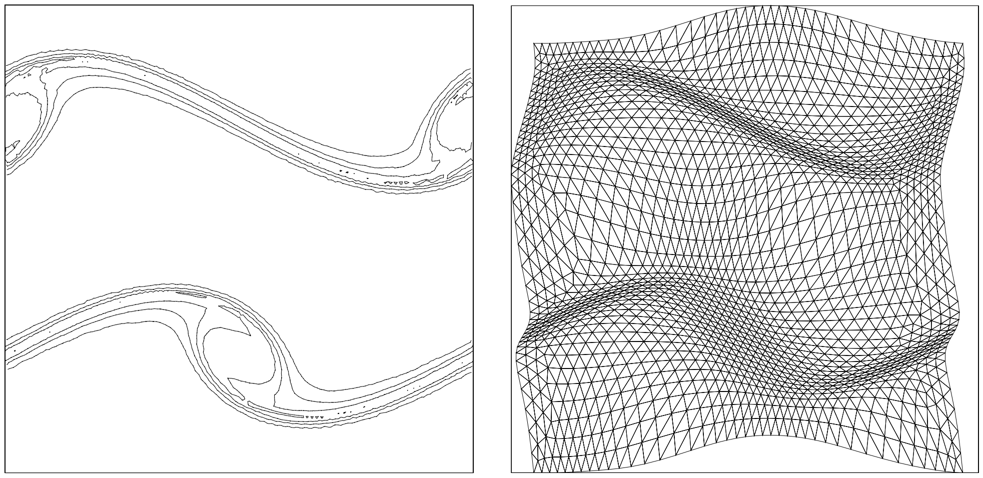

FIG. 10. *Thick shear layer problem: Vorticity (left) and mesh (right) contours at* $t = 0.8$*, obtained by using a* 40×40 *mesh with the velocity-gradient monitor* (4.2)*. The results indicate that this monitor is not as effective as the error-based monitor* (4.3) *or* (4.4).

TABLE 2

Example 5.2*:* l_2*-error and convergence rate for the moving mesh solutions of the velocity. In each case, the error was estimated using the Richardson extrapolation between two meshes (columns labeled, e.g.,* 20–40*). The convergence rate exponent was then estimated from these values by comparing adjacent error estimates (columns labeled "rate").*

Time	20–40	40–80	Rate	80–160	Rate	160–320	Rate
0.4	2.48e-2	5.88e-3	2.08	1.17e-3	2.32	3.90e-4	1.59
0.8	1.17e-1	3.00e-2	1.96	6.60e-3	2.19	1.72e-3	1.94
1.2	1.36e-1	3.00e-2	2.18	5.78e-3	2.38	1.62e-3	1.84

6. Concluding remarks.

This work presents an effective moving mesh algorithm for solving the incompressible Navier–Stokes equations in the primitive variables formulation. One of the main contributions is that we propose a (continuous) divergence-free interpolation which solves a linearized inviscid Navier–Stokes-type equation (3.26). A careful numerical study is presented for a double shear layer problem by using the proposed moving mesh algorithm.

Most of the solution contours are for the vorticity functions which are recovered by using the straightforward differentiation to the velocity. Since the flow field solver used is of second order accuracy, the computed vorticity is of first order accuracy only. Even with such a low accuracy, the plots for the vorticity of the moving mesh solutions are of very good quality. The extremely thin layer problems can be well resolved by a 160×160 mesh.

There are several ways to further improve the efficiency of the proposed scheme. One is to apply the moving mesh technique with some more efficient and robust finite element solvers for the incompressible Navier–Stokes equations, such as those developed in [12, 19]. It is useful to have some efficient solvers which are robust for a large Reynolds number and large ratio of the element volume.

Finally, we comment on the difference between the proposed method and the arbitrary Lagrangian–Eulerian (ALE) [16, 18] or deforming space-time mesh approaches [34]. In ALE representations, a grid is established where the cells can move and distort, but the nodes and cell interfaces are not constrained to move only at the local fluid velocity. Thus the grid can track the flow in a Lagrangian manner in some regions but allows the fluid to cross the grid in other regions. ALE is often very useful for solving flows with sharp interfaces between fluids with different properties. On the other hand, it often happens that the grid becomes so distorted in an ALE solution that further motion within the flow must be prohibited in extended regions [16]. Our method is based on an Eulerian setting, which solves the Navier–Stokes equations in a standard Eulerian system. Since the Lagrangian representation is not used, distorted grids are not seen in the moving mesh algorithms. In the deforming space-time mesh approaches [34], the finite element formulations of the governing equations are written over the space-time domain of the equation. Consequently, changes in the shape of the spatial domain due to the motion of the boundaries and interfaces are taken into account automatically. At each time step, the spatial domain occupied by the fluid changes its shape and some new elements/nodes may be generated, which are in contrast with the present moving mesh approach. It is pointed out that due to the periodic nature of the solutions the spatial domain seems deformable in our computations; see, e.g., Figure 3 and [2]. Even in this case, the overall area of the solution domain for the spatial domain is fixed. When the boundary conditions are nonperiodic (Dirichlet or Neumann), the geometry of the physical domain is unchanged. This is also different with the deformable space-time approach.

An animation related to the numerical simulations in this work can be viewed on the following website: http://www.math.hkbu.edu.hk/~ttang/MMmovie/NS.

Acknowledgments. We thank the referees for useful comments and suggestions. The major portion of this research was carried out while the first author was visiting the Hong Kong Baptist University.

REFERENCES

[1] J. Bell, P. Colella, and H. Glaz, *A second-order projection method for the incompressible Navier-Stokes equations*, J. Comput. Phys., 85 (1989), pp. 257–283.

[2] J. U. Brackbill and J. S. Saltzman, *Adaptive zoning for singular problems in two dimensions*, J. Comput. Phys., 46 (1982), pp. 342–368.

[3] D. L. Brown and M. L. Minion, *Performance of under-resolved two-dimensional incompressible flow simulations*, J. Comput. Phys., 122 (1995), pp. 165–183.

[4] A. Bruger, B. Gustafsson, P. Lotstedt, and J. Nilsson, *High Order Accurate Solution of the Incompressible Navier-Stokes Equations*, J. Comput. Phys., to appear, 2004.

[5] W. M. Cao, W. Z. Huang, and R. D. Russell, *An r-adaptive finite element method based upon moving mesh PDEs*, J. Comput. Phys., 149 (1999), pp. 221–244.

[6] H. D. Ceniceros and T. Y. Hou, *An efficient dynamically adaptive mesh for potentially singular solutions*, J. Comput. Phys., 172 (2001), pp. 609–639.

[7] A. J. Chorin, *Numerical solution of the Navier-Stokes equations*, Math. Comp., 22 (1968), pp. 745–762.

[8] A. S. Dvinsky, *Adaptive grid generation from harmonic maps on Riemannian manifolds*, J. Comput. Phys., 95 (1991), pp. 450–476.

[9] W. E and C.-W. Shu, *A numerical resolution study of high order essentially nonoscillatory schemes applied to incompressible flows*, J. Comput. Phys., 110 (1994), pp. 39–46.

[10] W. E and C.-W. Shu, *Small-scale structures in Boussinesq convection*, Phys. Fluids, 6 (1994), pp. 49–58.

[11] W. E and J.-G. Liu, *Essentially compact schemes for unsteady viscous incompressible flows*, J. Comput. Phys., 126 (1996), pp. 122–138.

[12] H. C. Elman, *Preconditioning for the steady-state Navier–Stokes equations with low viscosity*, SIAM J. Sci. Comput., 20 (1999), pp. 1299–1316.

[13] V. Girault and P.-A. Raviart, *Finite Element Methods for Navier-Stokes Equations, Theory and Algorithms*, Springer-Verlag, Berlin, 1986.

[14] M. Gunzburger, *Finite Element Methods for Viscous Incompressible Flows: A Guide to Theory, Practice and Algorithms*, Academic Press, Boston, 1989.

[15] R. Hamilton, *Harmonic Maps of Manifolds with Boundary*, Lecture Notes in Math. 471, Springer-Verlag, New York, 1975.

[16] 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.

[17] W. Huang and W. Sun, *Variational mesh adaptation.* II. *Error estimates and monitor functions*, J. Comput. Phys., 184 (2003), pp. 619–648.

[18] T. J. R. Hughes, W. K. Liu, and T. K. Zimmermann, *Lagrangian-Eulerian finite element formulation for incompressible viscous flows*, Comput. Methods Appl. Mech. Engrg., 29 (1981), pp. 329–349.

[19] D. Kay, D. Loghin, and A. Wathen, *A preconditioner for the steady-state Navier–Stokes equations*, SIAM J. Sci. Comput., 24 (2002), pp. 237–256.

[20] R. J. LeVeque, *High-resolution conservative algorithms for advection in incompressible flow*, SIAM J. Numer. Anal., 33 (1996), pp. 627–665.

[21] M. Li, T. Tang, and B. Fornberg, *A compact fourth-order finite difference scheme for the steady incompressible Navier-Stokes equations*, Internat. J. Numer. Methods Fluids, 20 (1995), pp. 1137–1151.

[22] R. Li, T. Tang, and P. W. Zhang, *Moving mesh methods in multiple dimensions based on harmonic maps*, J. Comput. Phys., 170 (2001), pp. 562–588.

[23] R. Li, T. Tang, and P. W. Zhang, *A moving mesh finite element algorithm for singular problems in two and three space dimensions*, J. Comput. Phys., 177 (2002), pp. 365–393.

[24] J.-G. Liu and C.-W. Shu, *A high-order discontinuous Galerkin method for 2D incompressible flows*, J. Comput. Phys., 160 (2000), pp. 577–596.

[25] J. Lopez and J. Shen, *An efficient spectral-projection method for the Navier-Stokes equations in cylindrical geometries.* I. *Axisymmetric cases*, J. Comput. Phys., 139 (1998), pp. 308–326.

[26] R. Kupferman and E. Tadmor, *A fast, high resolution, second-order central scheme for incompressible flows*, Proc. Natl. Acad. Sci. U.S.A., 94 (1997), pp. 4848–4852.

[27] H. Nessyahu and E. Tadmor, *Nonoscillatory central differencing for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 408–463.

[28] J.-F. Remacle, J. E. Flaherty, and M. S. Shephard, *An adaptive discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems*, SIAM Rev., 45 (2003), pp. 53–72.

[29] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.

[30] R. Schoen and S. T. Yau, *On univalent harmonic maps between surfaces*, Invent. Math., 44 (1978), pp. 265–278.

[31] Z.-J. Tan, Z.-R. Zhang, Y.-Q. Huang, and T. Tang, *Moving mesh methods with locally varying time steps*, J. Comput. Phys., 200 (2004), pp. 347–367.

[32] H. Z. Tang and T. Tang, *Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws*, SIAM J. Numer. Anal., 41 (2003), pp. 487–515.

[33] H. Z. Tang, T. Tang, and P. W. Zhang, *An adaptive mesh redistribution method for nonlinear Hamilton-Jacobi equations in two- and three-dimensions*, J. Comput. Phys., 188 (2003), pp. 543–572.

[34] T. E. Tezduyar, M. Behr, and J. Liou, *A new strategy for finite element computations involving moving boundaries and interfaces—the deforming-spatial-domain/space-time procedure.* I. *The concept and the preliminary tests*, Comput. Methods Appl. Mech. Engrg., 94 (1992), pp. 339–351.

[35] 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.

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