COMMUNICATIONS IN COMPUTATIONAL PHYSICS
Vol. **3**, No. 3, pp. 679-703

Commun. Comput. Phys.
March 2008

An Adaptive Moving Mesh Method for Two-Dimensional Incompressible Viscous Flows

Zhijun Tan[1], K. M. Lim[2] and B. C. Khoo[1,2,*]

[1] *Singapore-MIT Alliance, 4 Engineering Drive 3, National University of Singapore, Singapore 117576, Singapore.*
[2] *Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore.*

Received 26 January 2007; Accepted (in revised version) 22 July 2007

Available online 30 October 2007

Abstract. In this paper, we present an adaptive moving mesh technique for solving the incompressible viscous flows using the vorticity stream-function formulation. The moving mesh strategy is based on the approach proposed by Li et al. [*J. Comput. Phys.*, 170 (2001), pp. 562–588] to separate the mesh-moving and evolving PDE at each time step. The Navier-Stokes equations are solved in the vorticity stream-function form by a finite-volume method in space, and the mesh-moving part is realized by solving the Euler-Lagrange equations to minimize a certain variation in conjunction with a more sophisticated monitor function. A conservative interpolation is used to redistribute the numerical solutions on the new meshes. This paper discusses the implementation of the periodic boundary conditions, where the physical domain is allowed to deform with time while the computational domain remains fixed and regular throughout. Numerical results demonstrate the accuracy and effectiveness of the proposed algorithm.

AMS subject classifications: 65M06, 65M50, 76D05

Key words: Moving mesh method, finite volume method, Navier-Stokes equations, vorticity stream-function, incompressible flow.

1 Introduction

Adaptive moving mesh methods have many important applications in various physical and engineering fields such as solid and fluid dynamics, combustion, heat transfer, material science etc. The physical phenomena in these mentioned areas may develop dynamically singular or nearly singular solutions in fairly localized region. A high fidelity numerical investigation of these physical problems may require extremely fine meshes

*Corresponding author. *Email addresses:* smatz@nus.edu.sg (Z. Tan), limkm@nus.edu.sg (K. M. Lim), mpekbc@nus.edu.sg (B. C. Khoo)

over a small portion of the physical domain to resolve the large solution variations. The use of globally refined uniform meshes becomes computationally wasteful when dealing with systems in two or higher dimensions. In multi-dimensional problems, developing an effective and robust adaptive mesh method becomes almost absolutely necessary. Successful implementation of the adaptive approaches not only produces a high mesh density in regions of large gradient to improve the accuracy of numerical solution, but also decreases the cost of numerical calculation in comparison with the uniform mesh. Currently, there has been much important progress made in adaptive moving mesh methods for partial differential equations, including the grid distribution approach based on the variational principle of Winslow [36], Brackbill et al. [5,7], Ren and Wang [26], and Tang and Tang [33]; the finite element methods of Miller and Miller [24], and Davis and Flaherty [10]; the moving mesh PDEs of Russell et al. [8], Li and Petzold [19], and Ceniceros and Hou [9]; and the moving mesh methods based on harmonic mapping of Dvinsky [14] and Li et al. [11, 21]; and others. Computational costs of moving mesh methods can be further reduced with locally varying time steps [31].

There are two main ways to generate an adaptive mesh, namely, local mesh refinements and moving mesh method. In local mesh refinement methods, the adaptive mesh is generated by adding or removing grid points based on the posteriori error of the numerical solution. Local refinement approach requires complicated data structures and technically complex methods to communicate information among different levels of refinement. In the moving mesh methods, the total number of the grid points is kept fixed. The grids are moved continuously in the whole solution domain to cluster grid points in regions where the solution has the larger variations. In the past two decades this numerical technique has been proven very successful for solving time-dependent problems whose solution has large gradient or discontinuities, see, e.g., [2–4,7,11,12,20,28,30,31]. In particular, Almgren et al. [1] introduced a adaptive projection method for the variable density incompressible Navier-Stokes equations on nested grids, while Di et al. [11] developed a moving mesh finite element methods for solving the incompressible Navier-Stokes equations in the primitive variables formulation and devised a divergence-free interpolation which is very essential for incompressible problems. Still, Ding and Shu [13] proposed a stencil adaptive algorithm for finite difference solution of incompressible viscous flows, and Min and Gibou [25] presented an unconditionally stable second-order accurate projection method for the incompressible Navier-Stokes equations on non-graded adaptive Cartesian grids. The latter employed quadtree and octree data structures as an efficient means to represent the grid.

One main difficulty in solving the incompressible viscous flows is the divergence-free constraint of the velocity field. There are two popular approaches to handle the divergence-free constraint in the incompressible Navier-Stokes equations. One is to use projection technique. This technique is commonly used in many incompressible Navier-Stokes solvers [1, 25, 29]. However, in general, the pressure Poisson solver of projection step will be time consuming on unstructured grids or adaptive grids. The other is to introduce the stream function, see [15,22]. One possible disadvantage of this approach is

that the order of spatial derivatives increases by one, which reduces the order of accuracy by one, but its implementation is very convenient and robust on an adaptive moving mesh. In this paper, we will employ this approach.

The main objective of this work is to develop an efficient adaptive moving mesh method to solve incompressible viscous flows in the vorticity stream-function formulation. Our moving mesh approach is based on the strategy proposed in [20] by decoupling the mesh motion and the PDE evolution. The solution-adaptive mesh is obtained by solving a set of nonlinear elliptic PDEs for the mesh map. A conservative interpolation is used to obtain the approximate solution on the resulting new meshes. The given PDEs are next advanced one time step based on a second-order finite volume approach. In our adaptive algorithm, both the mesh generation and the PDE evolution are solved in the computational domain. We first transform the governing equations into the computational domain by a local (time-independent) mapping. The mapping is obtained via the moving mesh approach, namely by solving the Euler-Lagrange equation to minimize a certain variation involving monitor functions.

This paper is organized as follows. In the next section we introduce the vorticity stream-function formulation for incompressible viscous flows. The corresponding moving mesh methods are described in Section 3. In Section 4, numerical results for 2D problems will be presented and discussed. Some concluding remarks will be made in the final section.

2 Vorticity stream-function formulation

We consider the two-dimensional incompressible Navier-Stokes equations in the traditional primitive variable (velocity-pressure) formulation:

$$\mathbf{u}_t+(\mathbf{u}\cdot\nabla)\mathbf{u}=-\nabla p+\mu\Delta\mathbf{u}, \tag{2.1}$$

$$\nabla\cdot\mathbf{u}=0, \tag{2.2}$$

where $\mathbf{u}=(u,v)^T=\mathbf{u}(x,y,t)$ is the fluid velocity vector with components u and v in the horizontal and vertical directions, respectively; $p=p\ (x,y,t)$ is the fluid pressure; and μ is the (assumed constant) kinematic viscosity. Partly in order to avoid handling directly the pressure variable, an alternative formulation using stream-function and vorticity has been used by researches. In 2D, we write this system of equations in the vorticity stream-function formulation by taking the curl on both sides of (2.1). Thus we have the following system of scalar equations:

$$\omega_t+(\mathbf{u}\cdot\nabla)\omega=\mu\Delta\omega, \tag{2.3}$$

$$-\Delta\psi=\omega, \tag{2.4}$$

where $\omega=v_x-u_y$ is the vorticity. The velocity $\mathbf{u}=(u,v)$ is determined by the stream function ψ as follows:

$$u=\psi_y, \qquad v=-\psi_x. \tag{2.5}$$

Due to the incompressibility of the flow, Eq. (2.3) can be written in an equivalent form,

$$\omega_t + \nabla \cdot (\mathbf{u}\omega) = \mu \Delta \omega. \tag{2.6}$$

This formulation has been used successfully by a large number of researchers over the past years to test new methods for the numerical solutions of a variety of fluid flow problems.

3 2D moving mesh technique

The basic idea of the moving mesh method is to *relocate* grid points in a mesh having a fixed number of nodes in such a way that the nodes remain concentrated in regions of rapid variation of the solution. The principal ingredient of the moving mesh methods is the so-called equidistribution principle. In 1D, it involves selecting mesh points such that some measure of the solution such as arclength or computed error is equalized over each subinterval. This measure is often connected to an indicator function called monitor function.

With a given numerical scheme like that provided in Section 3.3 , we can advance the numerical solution one time step to $t = t_{n+1}$. The following strategy is employed to carry out the grid restructuring as detailed in [30] and produce (briefly) below:

Algorithm 1

a. Solve the mesh redistributing equation (a generalized Laplacian equation) by one Gauss-Seidel iteration, to get $\mathbf{x}^{(k),n}$;

b. Interpolate the approximate solutions on the new grid $\mathbf{x}^{(k),n}$;

c. Obtain a weighted average of the locally calculated monitor at each computational cell and the surrounding monitor values;

d. The iteration procedure (a)-(c) on grid-motion and solution-interpolation is continued until there is no significant change in calculating the new grid from one iteration to the next.

Further discourse are divided into four subsections, namely mesh generation based on the variational approach, monitor functions, 2D PDE evolution and solution procedure.

3.1 Mesh generation

The mesh is generated using variational approach. Let $\mathbf{x} = (x,y)$ and $\boldsymbol{\xi} = (\xi,\eta)$ denote the physical and computational coordinates. A coordinate mapping from the computational domain Ω_c to the physical domain Ω_p is given by

$$x = x(\xi,\eta), \quad y = y(\xi,\eta), \tag{3.1}$$

and the inverse map is

$$\xi=\xi(x,y),\quad \eta=\eta(x,y). \tag{3.2}$$

The specific map is obtained by minimizing of a mesh adaptation functional of the form

$$E[\xi,\eta]=\frac{1}{2}\int_{\Omega_p}(\nabla\xi^T G_1^{-1}\nabla\xi+\nabla\eta^T G_2^{-1}\nabla\eta)dxdy, \tag{3.3}$$

where G_1 and G_2 are given symmetric positive definite matrices called monitor functions. In general, monitor functions depend on the underlying solution to be adapted and its derivatives. More terms can be added to the functional (3.3) to control other aspects of the adaptive mesh such as orthogonality and mesh alignment with a given vector field [5,7].

In this work, the adaptive mesh is determined by the corresponding Euler-Lagrange equations :

$$\nabla\cdot(G_1^{-1}\nabla\xi)=0,\quad \nabla\cdot(G_2^{-1}\nabla\eta)=0. \tag{3.4}$$

One of the simplest choices of monitor functions is $G_1=G_2=MI$, where I is the identity matrix and M is a positive weight function. One typical choice of the weight function is $M=\sqrt{1+|\nabla u|^2}$, where u is a solution of the underlying PDEs. This choice of the monitor function corresponds to Winslows variable diffusion method [36]:

$$\nabla\cdot\left(\frac{1}{M}\nabla\xi\right)=0,\quad \nabla\cdot\left(\frac{1}{M}\nabla\eta\right)=0. \tag{3.5}$$

(3.4) gives the coordinate transformation for the mesh generation and adaptation. Grid generation is basically to obtain the curvilinear coordinate system (3.1) from the above elliptic system (3.4). Usually, after solving the system (3.4) for $\boldsymbol{\xi}(\mathbf{x})$, we find the inverse map to obtain $\mathbf{x}(\boldsymbol{\xi})$, which is rather expensive. Certainly, we can directly solve the corresponding equations on the computational domain Ω_c by interchanging the dependent and independent variables in (3.4). However, the obtained equations are usually complicated and massive computations are required. An alternative approach, as suggested by Ceniceros and Hou [9], is to consider a functional defined in the computational domain directly:

$$\tilde{E}[x,y]=\frac{1}{2}\int_{\Omega_c}(\tilde{\nabla}^T xG_1\tilde{\nabla}x+\tilde{\nabla}^T yG_2\tilde{\nabla}y)d\xi d\eta, \tag{3.6}$$

to replace the convectional (3.3), where G_1 and G_2 are again the monitor functions and $\tilde{\nabla}=(\partial_\xi,\partial_\eta)^T$. The corresponding Euler-Lagrange equations are then of the form

$$\tilde{\nabla}\cdot(G_1\tilde{\nabla}x)=0,\qquad \tilde{\nabla}\cdot(G_2\tilde{\nabla}y)=0. \tag{3.7}$$

If we take the monitor function with the simplest form $G_1=G_2=MI$, then the Eq. (3.7) is reduced to

$$\tilde{\nabla}\cdot(M\tilde{\nabla}x)=0,\qquad \tilde{\nabla}\cdot(M\tilde{\nabla}y)=0. \tag{3.8}$$

Therefore, the mesh distribution in the physical space can be directly obtained by solving (3.7), which is much simpler than the conventional variational approach (3.3). However,

the system (3.7) can lead to degenerating grids in some concave regions [14]. The original system (3.4) is more accurate and reliable than the simple one (3.7) even though it is more complicated. In the current paper, all numerical examples have simple geometry, so Eq. (3.7) will be used for the mesh generation.

Next, we specially discuss an implementation of the periodic boundary conditions for the Euler-Lagrange equations, since periodic boundary conditions are perhaps the most commonly used in incompressible flow simulations. Suppose the computational domain is a unit square. Though periodic boundary conditions can be understood in a straightforward way on the unit square, it is not immediately clear how the resulting adaptive mesh assumes the periodic properties. In fact, it is the displacement of the moving grid point from its inverse image on the regular grid, i.e. $\mathbf{x}(\boldsymbol{\xi})$-$\boldsymbol{\xi}$, that satisfies the periodic boundary conditions on the unit square, i.e.,

$$\mathbf{x}(\boldsymbol{\xi}+(k,l))=\mathbf{x}(\boldsymbol{\xi})+(k,l), \qquad \text{integer pair}(k,l). \tag{3.9}$$

Interestingly, this condition does not require that the mapping $\mathbf{x}(\boldsymbol{\xi})$ maps a unit square onto a unit square. In other words, with periodic boundary conditions, the physical domain may not turn out to be a square even though the computational domain is. This can be seen in numerical examples later on. The condition (3.9) guarantees that the periodic copies of Ω_p (non-square) cover the whole two-dimensional space as effectively as periodic copies of the unit square. In particular, it is easy to verify that (3.9) implies that the area of the physical domain Ω_p is the same as that of Ω_c. With periodic boundary conditions on $\mathbf{X}=\mathbf{x}(\boldsymbol{\xi})$-$\boldsymbol{\xi}$, Eq. (3.8) becomes

$$\tilde{\nabla}\cdot(M\tilde{\nabla}X)+M_\xi=0, \qquad \tilde{\nabla}\cdot(M\tilde{\nabla}Y)+M_\eta=0. \tag{3.10}$$

In our computation, we use Gauss-Seidel (GS) iteration to approximate the solution of the above system (3.8) or (3.10). The iteration is continued until there is no significant change in calculating new grids from one iteration to the next. In practice, a few iterations are required at each time level, so the cost for generating new mesh is not expensive.

After generating the new mesh at each iterative step according to the monitor function, we need to remap the approximate solutions onto the newly resulting mesh $\{\mathbf{x}_{j,k}\}$ from the old mesh $\{\tilde{\mathbf{x}}_{j,k}\}$. Many remapping schemes have been suggested, such as the non-conservative one for the nonlinear Hamilton-Jacobi equation [34] and the conservative remapping scheme for the hyperbolic conservation laws [33]. Recently Zhang [37] presented a new conservative remapping, which may be more accurate and robust than the Tang and Tang's interpolation scheme [33]. However, it remains to be seen the implementation for higher-dimensions. In our computations, the remapping procedure of the vorticity ω can be realized by using the conservative interpolation technique proposed by Tang and Tang [33], which is

$$\begin{aligned}\left|\tilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}\right|\tilde{\omega}_{j+\frac{1}{2},k+\frac{1}{2}}=&\left|A_{j+\frac{1}{2},k+\frac{1}{2}}\right|\omega_{j+\frac{1}{2},k+\frac{1}{2}}-\left[(c^2_{\vec{n}}\omega)_{j+1,k+\frac{1}{2}}+(c^4_{\vec{n}}\omega)_{j,k+\frac{1}{2}}\right]\\&-\left[(c^3_{\vec{n}}\omega)_{j+\frac{1}{2},k+1}+(c^1_{\vec{n}}\omega)_{j+\frac{1}{2},k}\right],\end{aligned} \tag{3.11}$$

where $c^l_{\vec{n}} := c^x n^l_x + c^y n^l_y$ with mesh velocity $(c^x, c^y) = (x-\tilde{x}, y-\tilde{y})$ and the unit outward normal direction $\vec{n}^l = (n^l_x, n^l_y)$ on the corresponding surface of the control volume $A_{j+\frac{1}{2},k+\frac{1}{2}}$ for $l = 1, 2, 3, 4$. More detailed explanation can be found in [32]. The above formula is obtained using the classical perturbation theory. It is obvious that the discretization form (3.11) satisfies the mass-conservation in the following discrete sense:

$$\sum_{j,k} |\tilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}| \tilde{\omega}_{j+\frac{1}{2},k+\frac{1}{2}} = \sum_{j,k} |A_{j+\frac{1}{2},k+\frac{1}{2}}| \omega_{j+\frac{1}{2},k+\frac{1}{2}},$$

where $|A_{j+\frac{1}{2},k+\frac{1}{2}}|$ and $|\tilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}|$ mean the areas of the corresponding control cells. Some discussions of the properties of this conservative interpolation can be found in [33].

In order to obtain smoother transitions in the mesh, rather than merely using equations (3.8), an additional filter is applied to the monitor functions. Instead of working with M_{ij}, the smoothed values

$$\begin{aligned} \bar{M}_{i,j} \leftarrow & \frac{4}{16} M_{i,j} + \frac{2}{16} (M_{i+1,j} + M_{i-1,j} + M_{i,j+1} + M_{i,j-1}) \\ & + \frac{1}{16} (M_{i-1,j-1} + M_{i-1,j+1} + M_{i+1,j-1} + M_{i+1,j+1}) \end{aligned}$$

are being used in the mesh equations.

3.2 Monitor functions

The monitor function is one of the most important elements in the adaptive moving mesh algorithms. It is very important to choose a suitable monitor function; otherwise satisfactory adaptations can not be obtained no matter how good a moving mesh algorithm is. For problems with free interfaces, the singularity often occurs around the interface where more grid points are required. Away from the interface, it is suggested that the grids should be as uniform as possible. Appropriate choice of the monitor will generate grids with good quality in terms of smoothness, skewness, and aspect ratio. There are several possible choices of the monitor function for our problems.

An often seen, and probably the most basic choice (see [33]) to detect regions with high spatial activity is conventionally the arclength-type monitor function:

$$M = \sqrt{1 + \alpha |\nabla \omega|^2}. \tag{3.12}$$

Here the parameter α is an 'adaptivity'-parameter which controls the amount of adaptivity. For $\alpha = 0$, we have $M = 1$, representing a uniform mesh. Higher values of $\alpha > 0$ allow for more adaptivity. However, α is problem-dependent: in general, there is no straightforward rule on how to choose this parameter. In many cases the monitor functions involve some user-defined parameters which have to be obtained by doing several initial experiments.

A more sophisticated monitor function dealing with this issue involves a *time-dependent* parameter that is chosen automatically. Huang and Russell [17] and Huang and Sun [18] generalize this monitor function with a parameter β that controls the ratio of points in critical parts, and it reads

$$M=(1-\beta)\alpha(t)+\beta\|\nabla\omega\|_2, \quad \text{with} \quad \alpha(t)=\iint_{\Omega_c}\|\nabla\omega\|_2\,d\xi d\eta. \tag{3.13}$$

Here β is still a user-defined parameter, but the user does not strictly have to set this parameter. Following the approach of Huang and Russell [17], it can be shown that for the monitor (3.13), β is indeed the ratio of points in critical parts:

$$\beta=\frac{\int_{\Omega_p}\beta\|\nabla\omega\|_2\,dx}{\int_{\Omega_p}(1-\beta)\alpha(t)+\beta\|\nabla\omega\|_2\,dx}. \tag{3.14}$$

In this work, we will use the above monitor function. In our computations, for simplicity, we take the fixed choice of $\beta=0.5$; hence, approximately half of the mesh points is located in critical parts of the domain.

3.3 2D PDE evolution

3.3.1 The vorticity-transport equation

To demonstrate the principal ideas for the PDE evolution, we consider the following 2D convection-diffusion PDE system:

$$\omega_t+f(\omega)_x+g(\omega)_y=\mu\Delta\omega, \qquad (x,y)\in\Omega_p, \tag{3.15}$$

where Ω_p is the physical domain, and ω denotes the vorticity. For our problems, $f(\omega)=u\omega$, $g(\omega)=v\omega$. To allow flexibility in the handling complex geometry and use of fast solution solvers, we first transform the underlying PDEs using the coordinate transform

$$x=x(\xi,\eta),\ y=y(\xi,\eta) \quad \text{and} \quad \xi=\xi(x,y), \quad \eta=\eta(x,y), \tag{3.16}$$

where (x,y) and (ξ,η) are the physical and computational coordinates, respectively. Next, we solve the resulting equations in the computational domain with a (fixed) uniform mesh. The cell-centered finite volume method will be employed to solve the transformed PDEs. Note that

$$\omega_x=\frac{1}{J}\left[(y_\eta\omega)_\xi-(y_\xi\omega)_\eta\right],$$
$$\omega_y=\frac{1}{J}\left[-(x_\eta\omega)_\xi+(x_\xi\omega)_\eta\right],$$

where $J=x_\xi y_\eta - x_\eta y_\xi$ is the Jacobian of the coordinate transformation. With the above formulas, the underlying equation (3.15) becomes:

$$\omega_t+\frac{1}{J}F(\omega)_\xi+\frac{1}{J}G(\omega)_\eta=\mathcal{R},\quad (\xi,\eta)\in\Omega_c, \tag{3.17}$$

where

$$F(\omega)=y_\eta f(\omega)-x_\eta g(\omega),\quad G(\omega)=x_\xi g(\omega)-y_\xi f(\omega),\quad \mathcal{R}=\mu\Delta\omega.$$

It is noted that the transformation (3.16) is time-independent and therefore the time derivative in (3.15) is not transformed to a moving frame. However, the meshes in the physical domain are indeed time-dependent, which is realized by using the solution-dependent monitor functions.

Given a partition of the physical domain Ω_p, $\{A_{j+\frac{1}{2},k+\frac{1}{2}}|j,k\in\mathbb{Z}\}$, and a uniform partition of the computational or logical domain Ω_c with unit step sizes (i.e., $\Delta\xi=\Delta\eta=1$), together with a partition of the time interval $[0,T]$, $\{t_n=t_{n-1}+\Delta t_n|\Delta t_n>0,n\in\mathbb{Z}\}$. Here $A_{j+\frac{1}{2},k+\frac{1}{2}}$ is a quadrangle with four corners $\mathbf{x}_{j,k}$, $\mathbf{x}_{j+1,k}$, $\mathbf{x}_{j+1,k+1}$, and $\mathbf{x}_{j,k+1}$. The ξ- and η-derivatives of x and y are approximated at the midpoints of the cell edges and the cell center points as follows:

$$\begin{aligned}
&(\mathcal{Z}_\xi)_{j+\frac{1}{2},k}=\mathcal{Z}_{j+1,k}-\mathcal{Z}_{j,k},\\
&(\mathcal{Z}_\xi)_{j,k+\frac{1}{2}}=\frac{1}{4}(\mathcal{Z}_{j+1,k}+\mathcal{Z}_{j+1,k+1}-\mathcal{Z}_{j-1,k}-\mathcal{Z}_{j-1,k+1}),\\
&(\mathcal{Z}_\eta)_{j,k+\frac{1}{2}}=\mathcal{Z}_{j,k+1}-\mathcal{Z}_{j,k},\\
&(\mathcal{Z}_\eta)_{j+\frac{1}{2},k}=\frac{1}{4}(\mathcal{Z}_{j,k+1}+\mathcal{Z}_{j+1,k+1}-\mathcal{Z}_{j,k-1}-\mathcal{Z}_{j+1,k-1}),\\
&(\mathcal{Z}_\xi)_{j+\frac{1}{2},k+\frac{1}{2}}=\frac{1}{2}(\mathcal{Z}_{j+1,k}+\mathcal{Z}_{j+1,k+1}-\mathcal{Z}_{j,k}-\mathcal{Z}_{j,k+1}),\\
&(\mathcal{Z}_\eta)_{j+\frac{1}{2},k+\frac{1}{2}}=\frac{1}{2}(\mathcal{Z}_{j,k+1}+\mathcal{Z}_{j+1,k+1}-\mathcal{Z}_{j,k}-\mathcal{Z}_{j+1,k}),\qquad \mathcal{Z}=x\quad\text{or}\quad y.
\end{aligned}$$

In order to obtain the transformed PDEs, the key point here is to obtain the transformations for $\Delta\omega$. Note that

$$\omega_{xx}=\frac{1}{J}\Big[(J^{-1}y_\eta^2\omega_\xi)_\xi-(J^{-1}y_\xi y_\eta\omega_\eta)_\xi-(J^{-1}y_\xi y_\eta\omega_\xi)_\eta+(J^{-1}y_\xi^2\omega_\eta)_\eta\Big], \tag{3.18}$$

$$\omega_{yy}=\frac{1}{J}\Big[(J^{-1}x_\eta^2\omega_\xi)_\xi-(J^{-1}x_\xi x_\eta\omega_\eta)_\xi-(J^{-1}x_\xi x_\eta\omega_\xi)_\eta+(J^{-1}x_\xi^2\omega_\eta)_\eta\Big], \tag{3.19}$$

where $J=x_\xi y_\eta - x_\eta y_\xi$ is the Jacobian of the coordinate transformation. In our computations, the vorticity ω is defined at cell centers. The following symmetric discretizations

at $(\xi_{j+\frac{1}{2}},\eta_{k+\frac{1}{2}})$ will be used to approximate terms on the right-hand side of (3.18):

$$\begin{aligned}\Big[(J^{-1}y_\eta^2\omega_\xi)_\xi\Big]_{j+\frac{1}{2},k+\frac{1}{2}} &= (J^{-1}y_\eta^2)_{j+1,k+\frac{1}{2}}(\omega_{j+\frac{3}{2},k+\frac{1}{2}}-\omega_{j+\frac{1}{2},k+\frac{1}{2}})\\ &\quad -(J^{-1}y_\eta^2)_{j,k+\frac{1}{2}}(\omega_{j+\frac{1}{2},k+\frac{1}{2}}-\omega_{j-\frac{1}{2},k+\frac{1}{2}}),\end{aligned}\tag{3.20a}$$

$$\begin{aligned}\Big[(J^{-1}y_\xi^2\omega_\eta)_\eta\Big]_{j+\frac{1}{2},k+\frac{1}{2}} &= (J^{-1}y_\xi^2)_{j+\frac{1}{2},k+1}(\omega_{j+\frac{1}{2},k+\frac{3}{2}}-\omega_{j+\frac{1}{2},k+\frac{1}{2}})\\ &\quad -(J^{-1}y_\xi^2)_{j+\frac{1}{2},k}(\omega_{j+\frac{1}{2},k+\frac{1}{2}}-\omega_{j+\frac{1}{2},k-\frac{1}{2}}),\end{aligned}\tag{3.20b}$$

$$\begin{aligned}\Big[-(J^{-1}y_\xi y_\eta\omega_\eta)_\xi\Big]_{j+\frac{1}{2},k+\frac{1}{2}} &= \frac{-1}{4}(J^{-1}y_\xi y_\eta)_{j+\frac{3}{2},k+\frac{1}{2}}(\omega_{j+\frac{3}{2},k+\frac{3}{2}}-\omega_{j+\frac{3}{2},k-\frac{1}{2}})\\ &\quad +\frac{1}{4}(J^{-1}y_\xi y_\eta)_{j-\frac{1}{2},k+\frac{1}{2}}(\omega_{j-\frac{1}{2},k+\frac{3}{2}}-\omega_{j-\frac{1}{2},k-\frac{1}{2}}),\end{aligned}\tag{3.20c}$$

$$\begin{aligned}\Big[-(J^{-1}y_\xi y_\eta\omega_\xi)_\eta\Big]_{j+\frac{1}{2},k+\frac{1}{2}} &= \frac{-1}{4}(J^{-1}y_\xi y_\eta)_{j+\frac{1}{2},k+\frac{3}{2}}(\omega_{j+\frac{3}{2},k+\frac{3}{2}}-\omega_{j-\frac{1}{2},k+\frac{3}{2}})\\ &\quad +\frac{1}{4}(J^{-1}y_\xi y_\eta)_{j+\frac{1}{2},k-\frac{1}{2}}(\omega_{j+\frac{3}{2},k-\frac{1}{2}}-\omega_{j-\frac{1}{2},k-\frac{1}{2}}).\end{aligned}\tag{3.20d}$$

The terms on the right-hand side of (3.19) can be approximated similarly. Having the above approximations, we are able to approximate the Laplacian $\Delta\omega$ by

$$(\Delta\omega)_{j+\frac{1}{2},k+\frac{1}{2}} = \frac{1}{J_{j+\frac{1}{2},k+\frac{1}{2}}}\sum_{l=-1}^{1}\sum_{m=-1}^{1}C^{lm}_{j+\frac{1}{2},k+\frac{1}{2}}\omega_{j+\frac{1}{2}+l,k+\frac{1}{2}+m},\tag{3.21}$$

where

$$\begin{aligned}C^{\pm1,-1}_{j+\frac{1}{2},k+\frac{1}{2}} &= \pm\frac{1}{4}\Big[(J^{-1}y_\xi y_\eta)_{j+\frac{1}{2}\pm1,k+\frac{1}{2}}+(J^{-1}y_\xi y_\eta)_{j+\frac{1}{2},k-\frac{1}{2}}\\ &\quad +(J^{-1}x_\xi x_\eta)_{j+\frac{1}{2}\pm1,k+\frac{1}{2}}+(J^{-1}x_\xi x_\eta)_{j+\frac{1}{2},k-\frac{1}{2}}\Big],\end{aligned}\tag{3.22a}$$

$$C^{0,\pm1}_{j+\frac{1}{2},k+\frac{1}{2}} = (J^{-1}y_\xi^2)_{j+\frac{1}{2},k+\frac{1}{2}\pm\frac{1}{2}}+(J^{-1}x_\xi^2)_{j+\frac{1}{2},k+\frac{1}{2}\pm\frac{1}{2}},\tag{3.22b}$$

$$C^{\pm1,0}_{j+\frac{1}{2},k+\frac{1}{2}} = (J^{-1}y_\eta^2)_{j+\frac{1}{2}\pm\frac{1}{2},k+\frac{1}{2}}+(J^{-1}x_\eta^2)_{j+\frac{1}{2}\pm\frac{1}{2},k+\frac{1}{2}},\tag{3.22c}$$

$$\begin{aligned}C^{0,0}_{j+\frac{1}{2},k+\frac{1}{2}} &= -(J^{-1}y_\eta^2)_{j+1,k+\frac{1}{2}}-(J^{-1}y_\eta^2)_{j,k+\frac{1}{2}}-(J^{-1}y_\xi^2)_{j+\frac{1}{2},k+1}-(J^{-1}y_\xi^2)_{j+\frac{1}{2},k}\\ &\quad -(J^{-1}x_\eta^2)_{j+1,k+\frac{1}{2}}-(J^{-1}x_\eta^2)_{j,k+\frac{1}{2}}-(J^{-1}x_\xi^2)_{j+\frac{1}{2},k+1}-(J^{-1}x_\xi^2)_{j+\frac{1}{2},k},\end{aligned}\tag{3.22d}$$

$$\begin{aligned}C^{-1,1}_{j+\frac{1}{2},k+\frac{1}{2}} &= \frac{1}{4}(J^{-1}y_\xi y_\eta)_{j-\frac{1}{2},k+\frac{1}{2}}+\frac{1}{4}(J^{-1}y_\xi y_\eta)_{j+\frac{1}{2},k+\frac{3}{2}}\\ &\quad +\frac{1}{4}(J^{-1}x_\xi x_\eta)_{j-\frac{1}{2},k+\frac{1}{2}}+\frac{1}{4}(J^{-1}x_\xi x_\eta)_{j+\frac{1}{2},k+\frac{3}{2}},\end{aligned}\tag{3.22e}$$

$$\begin{aligned}C^{1,1}_{j+\frac{1}{2},k+\frac{1}{2}} &= -\frac{1}{4}(J^{-1}y_\xi y_\eta)_{j+\frac{1}{2},k+\frac{3}{2}}-\frac{1}{4}(J^{-1}y_\xi y_\eta)_{j+\frac{3}{2},k+\frac{1}{2}}\\ &\quad -\frac{1}{4}(J^{-1}x_\xi x_\eta)_{j+\frac{3}{2},k+\frac{1}{2}}-\frac{1}{4}(J^{-1}x_\xi x_\eta)_{j+\frac{1}{2},k+\frac{3}{2}}.\end{aligned}\tag{3.22f}$$

Next we solve (3.17) by a finite volume approach. Denote the control cell $[\xi_j,\xi_{j+1}]\times[\eta_k,\eta_{k+1}]$ by $B_{j+\frac{1}{2},k+\frac{1}{2}}$ and the cell average values by

$$\bar{\omega}^n_{j+\frac{1}{2},k+\frac{1}{2}}=\frac{1}{\Delta\xi\Delta\eta}\int_{B_{j+\frac{1}{2},k+\frac{1}{2}}}\omega(\xi,\eta,t_n)d\xi d\eta.$$

For ease of notation, below we will drop the top bar for $\bar{\omega}$. This gives rise to

$$\omega^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}}=\omega^n_{j+\frac{1}{2},k+\frac{1}{2}}-\gamma_{j,k}\left(\bar{F}^n_{j+1,k+\frac{1}{2}}-\bar{F}^n_{j,k+\frac{1}{2}}\right)-\tau_{j,k}\left(\bar{G}^n_{j+\frac{1}{2},k+1}-\bar{G}^n_{j+\frac{1}{2},k}\right)+\Re_{j+\frac{1}{2},k+\frac{1}{2}},\quad(3.23)$$

where

$$\gamma_{j,k}=\frac{\Delta t_n}{J_{j+\frac{1}{2},k+\frac{1}{2}}},\qquad\tau_{j,k}=\frac{\Delta t_n}{J_{j+\frac{1}{2},k+\frac{1}{2}}},\qquad\Re_{j+\frac{1}{2},k+\frac{1}{2}}=\Delta t_n\mu(\Delta\omega^n)_{j+\frac{1}{2},k+\frac{1}{2}}.$$

The one-dimensional Lax-Friedrichs numerical flux will be applied to $\bar{F}$ and $\bar{G}$ in the ξ-, η-direction, respectively. That is

$$\bar{F}_{j,k+\frac{1}{2}}=\frac{1}{2}\left[F(\omega^-_{j,k+\frac{1}{2}})+F(\omega^+_{j,k+\frac{1}{2}})-\max_\omega\{|F_\omega|\}\cdot(\omega^+_{j,k+\frac{1}{2}}-\omega^-_{j,k+\frac{1}{2}})\right],\quad(3.24)$$

$$\bar{G}_{j+\frac{1}{2},k}=\frac{1}{2}\left[G(\omega^-_{j+\frac{1}{2},k})+G(\omega^+_{j+\frac{1}{2},k})-\max_\omega\{|G_\omega|\}\cdot(\omega^+_{j+\frac{1}{2},k}-\omega^-_{j+\frac{1}{2},k})\right],\quad(3.25)$$

where the maximum is taken between $\omega^-_{j,k+\frac{1}{2}}$ and $\omega^+_{j,k+\frac{1}{2}}$ in (3.24), and the maximum is taken between $\omega^-_{j+\frac{1}{2},k}$ and $\omega^+_{j+\frac{1}{2},k}$ in (3.25). In order to compute (3.24) and (3.25), a piecewise linear approximation will be used:

$$\omega^\pm_{j,k+\frac{1}{2}}=\omega_{j\pm\frac{1}{2},k+\frac{1}{2}}\mp\frac{1}{2}s_{j\pm\frac{1}{2},k+\frac{1}{2}},\qquad\omega^\pm_{j+\frac{1}{2},k}=\omega_{j+\frac{1}{2},k\pm\frac{1}{2}}\mp\frac{1}{2}s_{j+\frac{1}{2},k\pm\frac{1}{2}},$$

$$s_{j+\frac{1}{2},k+\frac{1}{2}}=\left(\text{sign}(s^-_{j+\frac{1}{2},k+\frac{1}{2}})+\text{sign}(s^+_{j+\frac{1}{2},k+\frac{1}{2}})\right)\frac{|s^+_{j+\frac{1}{2},k+\frac{1}{2}}s^-_{j+\frac{1}{2},k+\frac{1}{2}}|}{|s^+_{j+\frac{1}{2},k+\frac{1}{2}}|+|s^-_{j+\frac{1}{2},k+\frac{1}{2}}|},$$

$$s^-_{j+\frac{1}{2},k+\frac{1}{2}}=\omega_{j+\frac{1}{2},k+\frac{1}{2}}-\omega_{j-\frac{1}{2},k+\frac{1}{2}},\qquad s^+_{j+\frac{1}{2},k+\frac{1}{2}}=\omega_{j+\frac{3}{2},k+\frac{1}{2}}-\omega_{j+\frac{1}{2},k+\frac{1}{2}}.$$

A system of semi-discretized equations can be obtained from (3.23), which will be solved by a 3-stage Runge-Kutta method proposed by Shu and Osher [27]. More precisely, for the ODE system $\omega'(t)=L(\omega)$ we use

$$\begin{aligned}\omega^{(1)}_{jk}&=\omega^n_{jk}+\Delta t_nL(\omega^n_{jk}),\\ \omega^{(2)}_{jk}&=\frac{3}{4}\omega^n_{jk}+\frac{1}{4}\left[\omega^{(1)}_{jk}+\Delta t_nL(\omega^{(1)}_{jk})\right],\\ \omega^{n+1}_{jk}&=\frac{1}{3}\omega^n_{jk}+\frac{2}{3}\left[\omega^{(2)}_{jk}+\Delta t_nL(\omega^{(2)}_{jk})\right].\end{aligned}$$

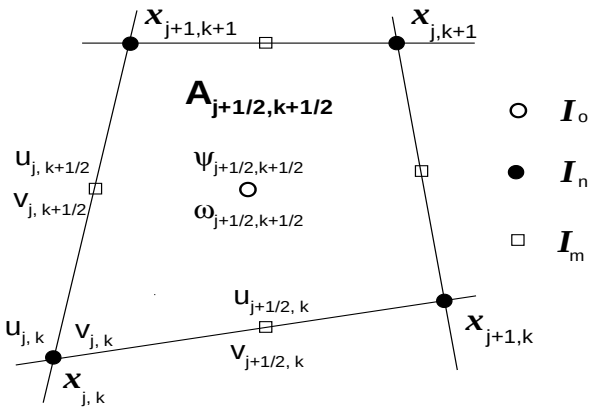

Figure 1: A control volume.

The above ODE solver satisfies the total variation non-increasing property.

The size of the time step used to advance the solution is restricted by two conditions necessary to guarantee stability of the method. The convective time step restriction is given by

$$\Delta t_{\text{CFL}} = \lambda_{\text{CFL}} \min_{j,k} \frac{J_{j+\frac{1}{2},k+\frac{1}{2}}}{|U_{j+\frac{1}{2},k+\frac{1}{2}}| + |V_{j+\frac{1}{2},k+\frac{1}{2}}|}.$$

Here $U_{j+\frac{1}{2},k+\frac{1}{2}} = (y_\eta u - x_\eta v)_{j+\frac{1}{2},k+\frac{1}{2}}$ and $V_{j+\frac{1}{2},k+\frac{1}{2}} = (x_\xi v - y_\xi u)_{j+\frac{1}{2},k+\frac{1}{2}}$, where $J_{j+\frac{1}{2},k+\frac{1}{2}}$ is the Jacobian of the coordinate transformation at $(\xi_{j+\frac{1}{2}}, \eta_{k+\frac{1}{2}})$ and λ_{CFL} is the standard CFL constant associated with the convection term. The viscous time step restriction is given by

$$\Delta t_{\text{vis}} = C_{\text{vis}} \frac{h^2}{2\mu}.$$

Here $h = \min_{j,k}(x_{j+1,k} - x_{j,k}, y_{j,k+1} - y_{j,k})$, where C_{vis} is the constant associated with the viscous term and taken as 0.5 here. The eventual restriction on the time step is then $\Delta t = \min\{\Delta t_{\text{CFL}}, \Delta t_{\text{vis}}\}$.

3.3.2 Computing the velocity

To compute the cell face values of the flow velocity (u,v) in (3.24) we need to solve first for the stream function ψ. In our computation, the vorticity ω and stream-function ψ are defined at the cell center, while the velocity $\mathbf{u}$ is defined at the cell corner, see Fig. 1, where $\mathbf{I}_o$ denotes the cell center points, $\mathbf{I}_n$ denotes the cell corner points, and $\mathbf{I}_m$ denotes the midpoints of the cell edges. Using the same approximation to the Laplacian operator provided in (3.21), the spatial discretization of $\Delta\psi$ in the stream-function Eq. (2.4) leads to a large sparse symmetric and positive definite linear system, which is solved by using the PCG method. After solving for ψ, we can compute the flow velocity from (2.5) by

using a second-order accurate central finite difference scheme as follows:

$$u=\psi_y=\frac{1}{\hat{J}}(-\hat{x}_\eta\hat{\psi}_\xi+\hat{x}_\xi\hat{\psi}_\eta),\qquad v=-\psi_x=\frac{1}{\hat{J}}(-\hat{y}_\eta\hat{\psi}_\xi+\hat{y}_\xi\hat{\psi}_\eta)\quad\text{on}\quad \mathbf{I}_n, \tag{3.26}$$

where

$$\begin{aligned}
&(\hat{\psi}_\xi)_{j,k}:=\frac{1}{2}(\psi_{j+\frac{1}{2},k+\frac{1}{2}}-\psi_{j-\frac{1}{2},k+\frac{1}{2}}+\psi_{j+\frac{1}{2},k-\frac{1}{2}}-\psi_{j-\frac{1}{2},k-\frac{1}{2}}),\\
&(\hat{\psi}_\eta)_{j,k}:=\frac{1}{2}(\psi_{j+\frac{1}{2},k+\frac{1}{2}}-\psi_{j+\frac{1}{2},k-\frac{1}{2}}+\psi_{j-\frac{1}{2},k+\frac{1}{2}}-\psi_{j-\frac{1}{2},k-\frac{1}{2}}),\\
&(\hat{\mathcal{Z}}_\xi)_{j,k}:=(\mathcal{Z}_{j+1,k}-\mathcal{Z}_{j-1,k})/2,\quad(\hat{\mathcal{Z}}_\eta)_{j,k}:=(\mathcal{Z}_{j,k+1}-\mathcal{Z}_{j,k-1})/2,\\
&\hat{J}_{j,k}=(\hat{x}_\xi\hat{y}_\eta-\hat{x}_\eta\hat{y}_\xi)_{j,k},\quad \mathcal{Z}=x\text{ or }y.
\end{aligned}$$

It is not difficult to find that the above discrete velocity field is divergence-free in the sense of

$$0=(\nabla\cdot\mathbf{u})=\frac{1}{\hat{J}}\Big(\overline{(y_\eta u-x_\eta v)_\xi}+\overline{(x_\xi v-y_\xi u)_\eta}\Big)\quad\text{on}\quad \mathbf{I}_o, \tag{3.27}$$

where

$$\begin{aligned}
\Big(\overline{(y_\eta u-x_\eta v)_\xi}\Big)_{j+\frac{1}{2},k+\frac{1}{2}}:=\frac{1}{2}\Big(&(\hat{y}_\eta u-\hat{x}_\eta v)_{j+1,k+1}-(\hat{y}_\eta u-\hat{x}_\eta v)_{j,k+1}\\
&+(\hat{y}_\eta u-\hat{x}_\eta v)_{j+1,k}-(\hat{y}_\eta u-\hat{x}_\eta v)_{j,k}\Big),\\
\Big(\overline{(x_\xi v-y_\xi u)_\eta}\Big)_{j+\frac{1}{2},k+\frac{1}{2}}:=\frac{1}{2}\Big(&(\hat{x}_\xi v-\hat{y}_\xi u)_{j+1,k+1}-(\hat{x}_\xi v-\hat{y}_\xi u)_{j,k+1}\\
&+(\hat{x}_\xi v-\hat{y}_\xi u)_{j+1,k}-(\hat{x}_\xi v-\hat{y}_\xi u)_{j,k}\Big),
\end{aligned}$$

and

$$\hat{J}_{j+\frac{1}{2},k+\frac{1}{2}}=|A_{j+\frac{1}{2},k+\frac{1}{2}}|.$$

We then evaluate the normal velocities on the cells′ edges ($\mathbf{I}_m$) by simple second order averaging:

$$\mathbf{u}_{j,k+\frac{1}{2}}=\frac{1}{2}(\mathbf{u}_{j,k}+\mathbf{u}_{j,k+1}),\quad \mathbf{u}_{j+\frac{1}{2},k}=\frac{1}{2}(\mathbf{u}_{j,k}+\mathbf{u}_{j+1,k}).$$

Remark 3.1. Note that there are some variants to computing the velocity field $\mathbf{u}$. For example, the velocity $\mathbf{u}$ is defined at the cell edges and then approximated directly on $\mathbf{I}_m$ with a central difference approach, i.e.

$$u=\psi_y=\frac{1}{J}(-x_\eta\psi_\xi+x_\xi\psi_\eta),\qquad v=-\psi_x=\frac{1}{J}(-y_\eta\psi_\xi+y_\xi\psi_\eta)\quad\text{on}\quad \mathbf{I}_m, \tag{3.28}$$

where $J = x_\xi y_\eta - x_\eta y_\xi$, and the geometric quantities x_ξ, x_η, y_ξ, y_η and physical quantities ψ_ξ, ψ_η on $\mathbf{I}_m$ are approximated by centered finite differences. However the computed velocity from (3.28) is not divergence-free in the discrete sense of (3.27) or

$$0 \neq (\nabla \cdot \mathbf{u}) = \frac{1}{J}\Big((y_\eta u_\xi - y_\xi u_\eta v) + (-x_\eta u_\xi + x_\xi u_\eta)\Big) \quad \text{on} \quad \mathbf{I}_o. \tag{3.29}$$

The reason is that the identity $\psi_{xy} = \psi_{yx}$ is not always satisfied in the discrete sense on a non-uniform mesh, which depends on the specific implementation of the discretization scheme.

3.4 Solution procedure

Our solution procedure consists of two independent parts: evolution of the governing equations and an iterative mesh redistribution. The first part is a finite volume method, see Section 3.3. In each iteration of the second part, mesh points are first redistributed by the Gauss-Seidel method based on a more sophisticated monitor function, and then vorticity ω is updated on the resulting new meshes by the conservative interpolation formula (3.11), see Section 3.1. The algorithm ensures that the velocity field is divergence-free in each control volume $A_{j+\frac{1}{2},k+\frac{1}{2}}$. The overall solution procedure can be illustrated by the following flowchart:

Algorithm 2

Step 1 Provide an initial adaptive mesh $\mathbf{x}^n_{j,k}$ based on the initial function, $n \geq 0$.

Step 2 Advance the solution one time step Δt_n based on a appropriate numerical scheme.

- **a.** Compute ψ^n using the PCG iterative method via solving the stream function equation, Eq. (2.4), then compute $\mathbf{u}^n$ from (3.26) using the central finite difference method.
- **b.** Compute vorticity ω^{n+1} from (3.23) for solving the vorticity transport equation, Eq. (2.3).

Step 3 Carry out the grid restructuring as provided by **Algorithm 1** to move the mesh and then update the vorticity on the new mesh.

Step 4 If $t \geq T$, then save the result and stop. Otherwise, go to **Step** 2 for the next cycle.

The third step above is in fact an iteration step and in general requires a few iterations at each time step. However, in our numerical computations, $1 \sim 4$ iterations are sufficient to obtain a satisfactory mesh at each time level except at the initial stage where the number of iterations depends on the degree of singularity of the initial data. The conservative interpolation (3.11) is carried out after each GS iteration step.

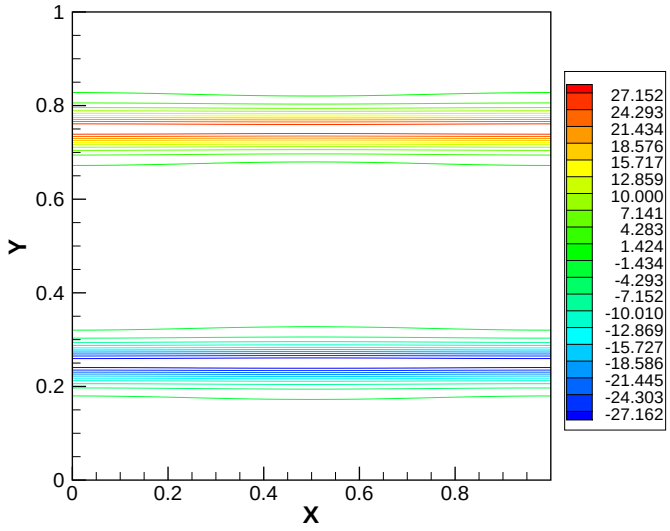

Figure 2: Example 4.1: Initial vorticity distribution for the shear layer problem.

4 Numerical examples

In this section, two numerical experiments are presented to demonstrate the efficiency of the proposed adaptive moving mesh method procedure. All the computations are carried out on a personal computer with Pentium 4 (3.0GHz) and Fortran compiler.

Example 4.1. Double shear flow

Consider a double shear layer governed by the Navier-Stokes equations (2.1)-(2.2), in a one unit periodic domain, subject to the initial conditions

$$u_0(x,y) = \begin{cases} \tanh(\rho_s(y-0.25)) & \text{for} \quad y \leq 0.5, \\ \tanh(\rho_s(0.75-y)) & \text{for} \quad y > 0.5, \end{cases} \tag{4.1}$$

$$v_0(x,y) = \delta_s \sin(2\pi x), \tag{4.2}$$

with $\rho_s = 30$ and $\delta_s = 0.05$. The velocity $\mathbf{u}$, the stream function ψ and the vorticity ω are subjected to periodic boundary conditions. The above parameter ρ_s determines the slope of the shear layer, and δ_s represents the size of the perturbation. This is the same problem studied by Brown and Minion [6], and represents a shear layer which is their "thick" case (Fig. 2). In our computations, the viscosity is set to $\mu = 10^{-4}$. When the mesh is not fine enough, there are spurious vorticity appearing in the numerical result [6]. The calculations are run to $t = 1.2$ with a constant CFL of 0.6. In Figs. 3 and 4, we show the contour plots of the vorticity and the corresponding adaptive meshes at times $t = 0.8$ and 1.2, obtained using a 80^2 moving grid. From these figures, it can be seen that the initial layer rolls up in time into strong vortical structures. The contour plots for the vorticity of the moving mesh solutions are of very good quality. These results compare very well

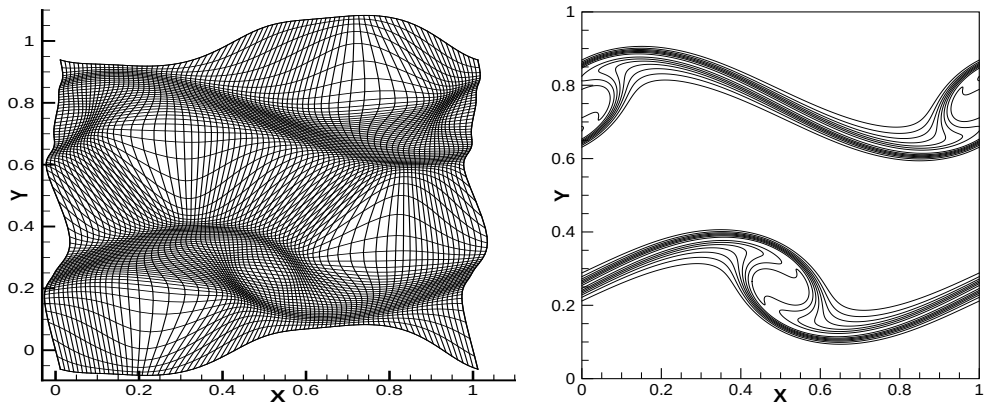

Figure 3: Example 4.1: The adaptive mesh and the contour of vorticity for the shear layer problem at $t=0.8$, obtained using a 80^2 moving grid.

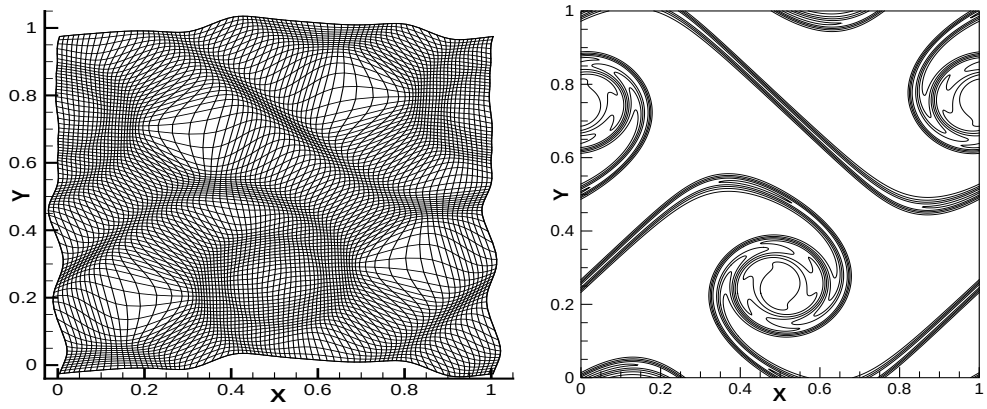

Figure 4: Same as Fig. 3, except with $t=1.2$.

with those obtained via the Godunov-projection method in [6] and those obtained using the moving finite element method in [11].

Note also that the choice of the parameter $\beta=1$ leads (as expected) to a mesh that approximately places half of the mesh points to be clustered inside the shear layer and the roll-ups where the solutions have large solution variations. The layout of the mesh plots is quite interesting, see Figs. 3 and 4. It should be pointed out that the spatial domain is allowed to deform for this problem with time as in [11], though the physical domain in the experiments is very simple. Due to the periodic nature of the solutions, it is natural to deform the physical domain as discussed in Section 3.1. For comparison, adaptive mesh without deformation of the domain is presented in Fig. 5. We can see clearly from Figs. 3 and 4 how the resulting adaptive meshes assumes the periodic properties. In this problem, large gradient solutions are developed close to the boundaries. As a consequence, the motion of the boundary points is of great importance to improve the quality of the

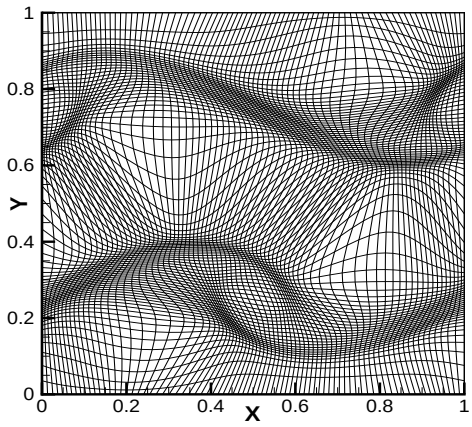

Figure 5: Adaptive mesh for the shear layer problem at $t=0.8$ without deformation of the domain.

adaptive mesh. The implementation of the periodic boundary condition for the mesh moving can also avoid using the 1D boundary grid redistribution method of [28,31]. The moving meshes are generated in the periodic setting as discussed in Section 3.1. Using the periodic characteristics, both the mesh and the corresponding solutions can be easily mapped back to the unit domain. It is noted that with periodic boundary conditions Brackbill and Saltzman [7] also obtained an "irregular" mesh layout very similar to those in Figs. 3 and 4.

Next, we make some comments on two possible monitor functions discussed in Section 3.2. As we mentioned in Section 3.2, the adaptivity parameter α in the monitor function (3.12) is problem-dependent, and there is no straightforward rule on how to choose this parameter. On the other hand, α is a constant, and the solution profile might change significantly through time. For this problem, we found $\alpha=10$ a suitable value from several experiments carried out. In our computations, the choice of the monitor function (3.13) overcomes the above disadvantages, and this type of the monitor function involves a *time-dependent* parameter that is chosen automatically. For comparison, in Fig. 6, we show adaptive meshes, magnified around the point $(0.5,0.25)$, with the monitor function (3.12) and the monitor function (3.13), respectively. We observe from Fig. 6 (right) that too many mesh grids have been clustered inside the roll-ups with using the monitor function (3.12), and the monitor function (3.13) seems more appropriate.

In Fig. 7, we plot the time history of the total energy (square of the L_2 norm of the velocity $\mathbf{u}$) and total enstrophy (square of the L_2 norm of the vorticity ω). We can see from Fig. 7 that the total energy decays rather than stays at a constant due to numerical dissipation, and the total enstrophy decreases with time. The dissipation of the energy is relatively smaller.

There is no exact analytical solution for this example. To test the accuracy, the problem is computed with the same method on much finer 320^2 uniform grid. It is observed that the moving mesh results on the coarse grid agree very well with the finer uniform mesh

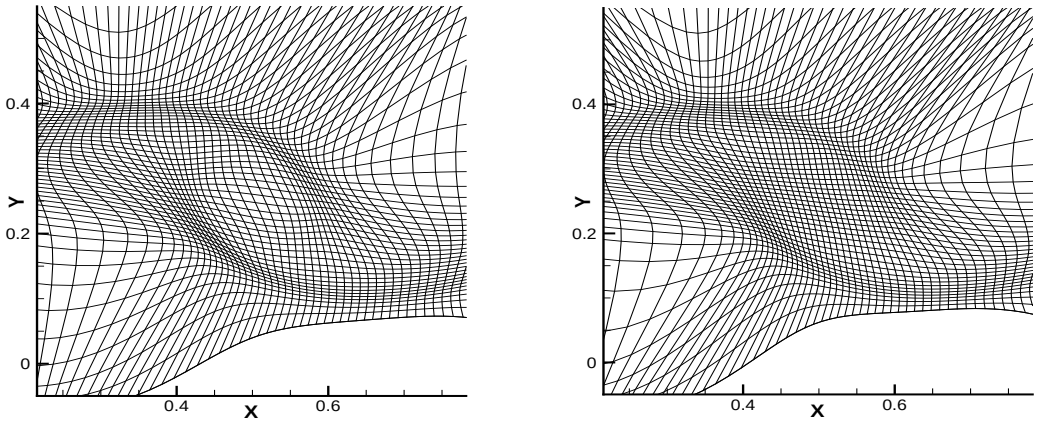

Figure 6: Adaptive meshes, magnified around the point $(0.5,0.25)$, for the shear layer problem at $t=0.8$ with the monitor function (3.12) (left) and the monitor function (3.13) (right), respectively.

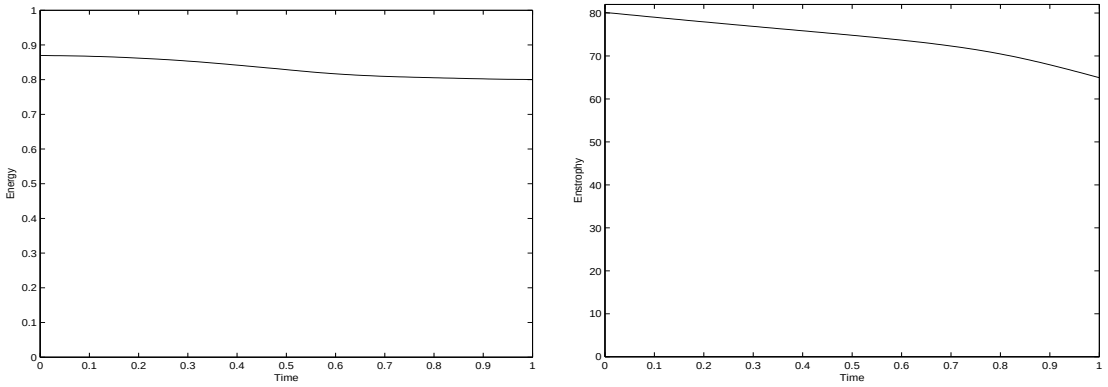

Figure 7: The time history of energy (square of the L_2 norm of the velocity $\mathbf{u}$) and total enstrophy (square of the L_2 norm of the vorticity ω).

results. For quantitative comparison, Figs. 8 (a) and 8 (b) show the computed vorticity profile and the velocity component u along the line $x=0.5$ at $t=0.8$, respectively, obtained using a 80^2 moving grid and a 320^2 uniform grid. In this plot, the solid line represents the numerical results obtained on the uniform mesh for reference. We can see their good agreement. It demonstrates the accuracy and efficiency of our moving mesh method.

In Table 1, the computing CPU times at different time level (t) using moving mesh method on a 80^2 moving grid and uniform mesh method on a 320^2 uniform grid are listed. The results show that both methods produce almost the same numerical results, but our proposed algorithm takes much less CPU time than the uniform mesh method. It demonstrates that our moving mesh method has great advantage in conserving computational resource and/or CPU time, compared to the corresponding uniform mesh method. Finally, we also show the velocity field at $t=1.2$ in Fig. 9.

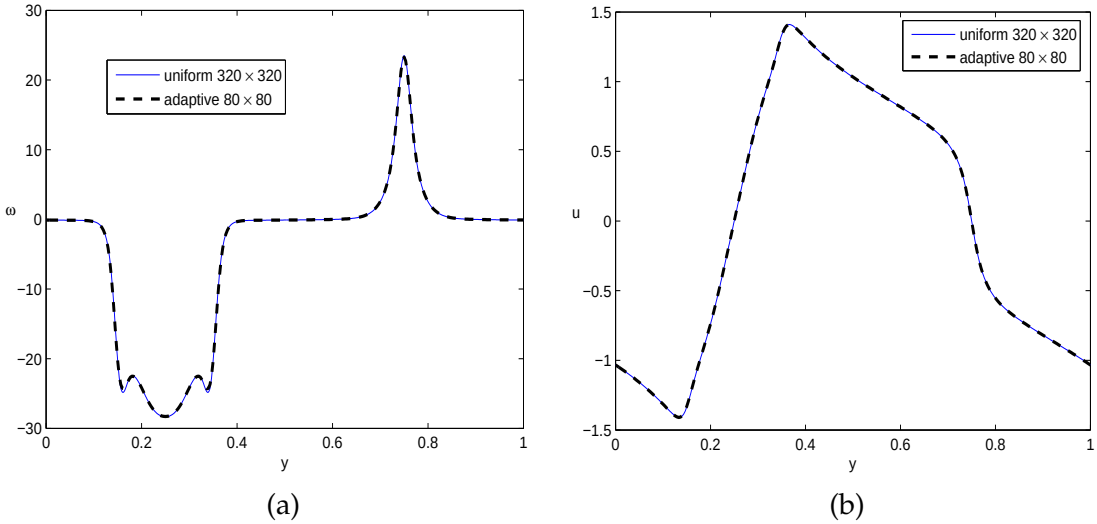

Figure 8: Example 4.1: Vorticity profile (a) and velocity component u profile (b) at $t=0.8$ with $N_x=N_y=80$ along vertical centerline. The solid line is obtained on a uniform mesh with 320^2 grid points.

Table 1: Example 4.1: Comparison of the CPU time in Seconds for the different mesh methods.

Schemes	$t=0.2$	$t=0.5$	$t=0.8$
Moving mesh method	93.43	293.42	567.49
Uniform mesh method	1581.06	4510.82	8444.63

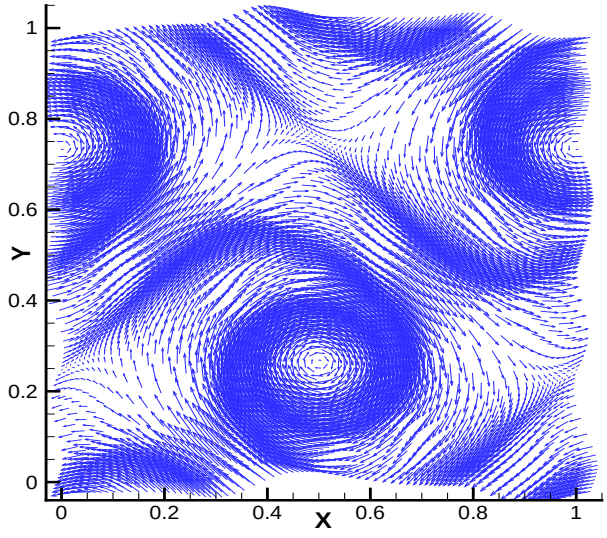

Figure 9: Velocity field at $t=1.2$ for Example 4.1.

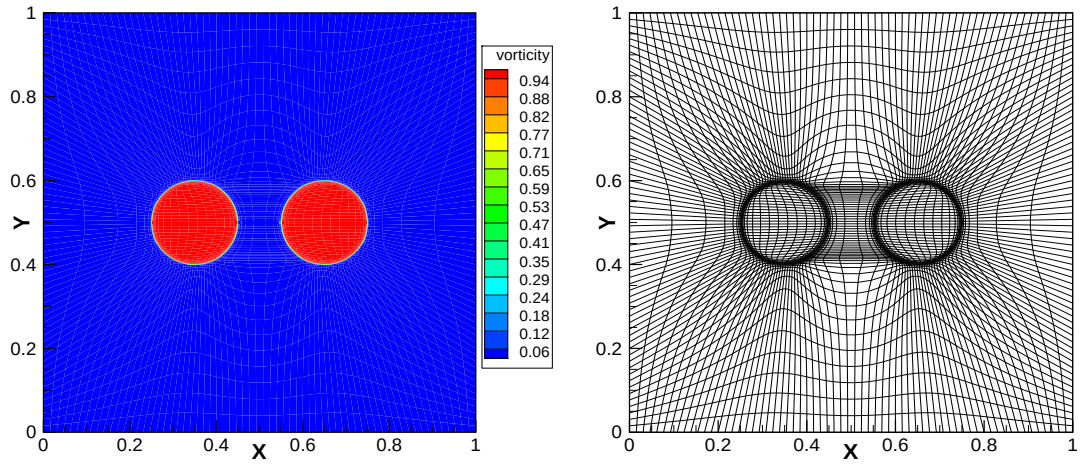

Figure 10: Initial vorticity distribution (left) and adaptive mesh redistribution (right) with a 64^2 moving grid.

Example 4.2. Co-rotating vortex merging

This example is to study a vortex merger problem and has been considered in [16, 23, 35]. The initial conditions consist of a pair of rotating circular vortices with the same radius $r_0 = 0.1$ and strength 1.0 placed in the unit square at $(0.35,0.5)$ and $(0.65,0.5)$. Denoting $x_1 = 0.35$, $x_2 = 0.65$, and $y_1 = y_2 = 0.5$, the profile for each vortex, centered at (x_i,y_i), is

$$\frac{1}{2}(1+\tanh(1000(0.1-r_i))),$$

where $r_i = \sqrt{(x-x_i)^2+(y-y_i)^2}$, $i = 1, 2$. In Fig. 10, we show the initial vorticity contour and the corresponding adaptive mesh. We can see from Fig. 10 (right) that more grid points are move to the initial vorticity region at the initial time. We also observe from this figure that there are still some mesh points which have been clustered between two circular vortexes. The possible reason is that the highly concentrated mesh is generated there due to the large singularity of the initial data. The initial velocity field is then obtained by solving the stream function associated with the given vorticity field. The boundary condition is subjected to the period boundary conditions. It is shown in [23, 35] that merger will occur when vortices are initially close enough (i.e., their distance of separation is smaller than some critical value). Due to the velocity field induced by each vortex, the vortex pair start to rotate around each other and then merge. In our computations, it is easily known from above initial conditions that the separated distance of two circular vortices is $d = 3.0r_0$ initially, which corresponds to the first case of $d/r_0 \leq 3.305$ reported in [35].

The calculations are run to $t = 30.0$ with a constant CFL of 0.6. The viscosity is set to $\mu = 10^{-6}$. In Figs. 11 and 12, the vorticity evolution, the contour plots of the vorticity and the corresponding adaptive meshes at times $t = 5.0$, 10.0, 15.0, 20.0, 25.0, and 30.0

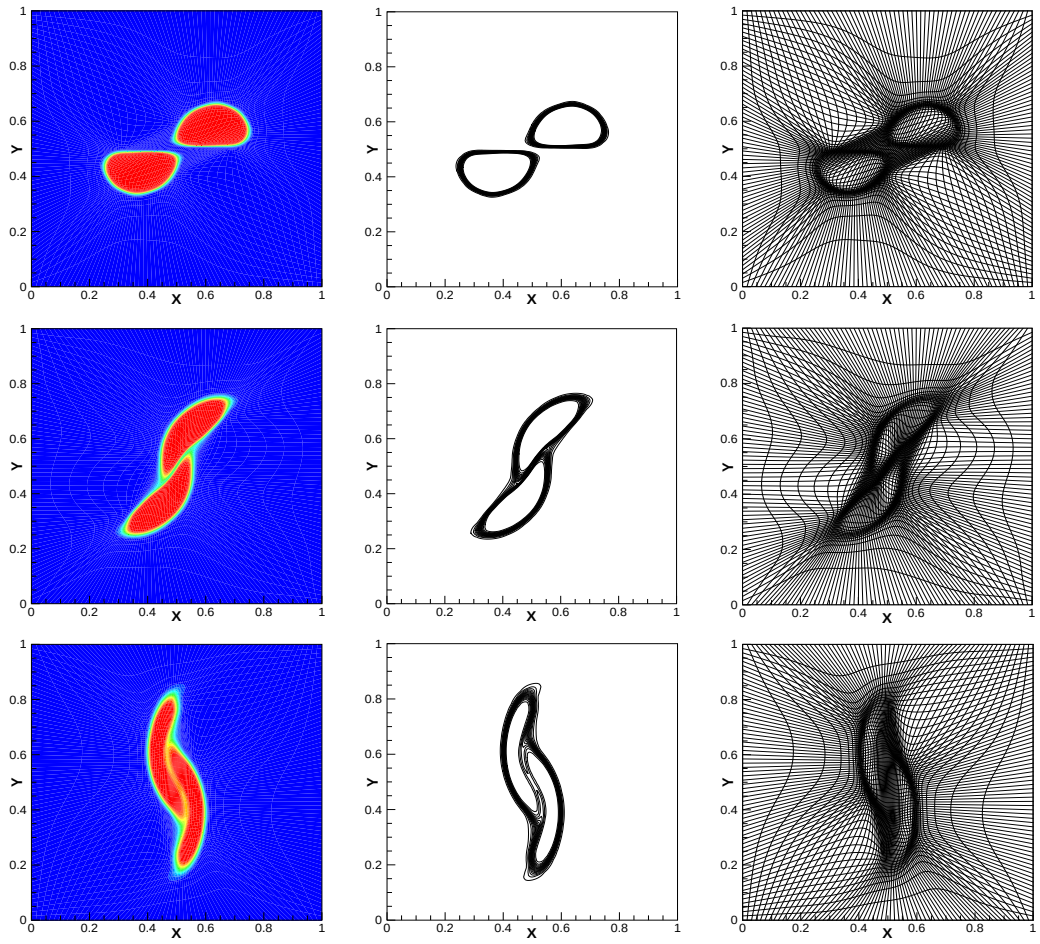

Figure 11: Example 4.2: Vorticity evolution (left), contours of vorticity (middle) and the adaptive meshes (right) with a 64^2 moving grid at $t=5.0$, 10.0, and 15.0 (top to bottom, respectively).

are presented; these are obtained using a 64^2 moving grid. We see clearly detailed vortex merger process: as time evolves, during the initial stages of the evolution of two vortices in the merger process, the vortices approach each other as they rotate about their center of vorticity and pass through a state very similar to two co-rotating vortices. These vortices subsequently merge, and simultaneously thin filaments of vorticity are ejected. The central vortex then consolidates into an ellipse (approximately), and the filaments are wrapped around the vortex with roll-up. Those results may be comparable to those found in [16, 23, 35]. As desired, it is also seen from these figures that approximately half of the grid points are clustered in the regions where the solutions vary rapidly; this increases the resolution of the numerical solutions. Again this demonstrates the effectiveness of our moving mesh method and the ability of our method to follow the evolving vortical structure.

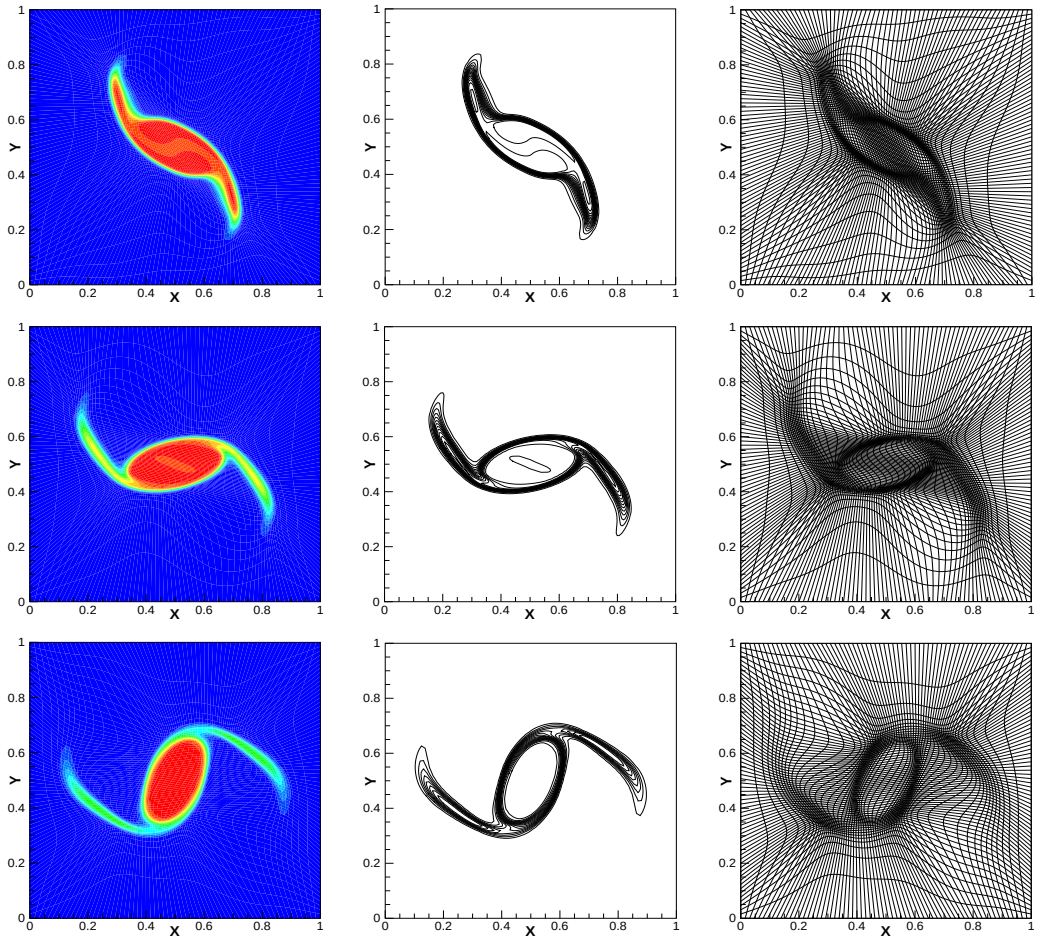

Figure 12: Same as Fig. 11, except with $t=20.0$ (top), $t=25.0$ (middle) and $t=30.0$ (bottom).

More computational details are given for this example. In Table 2, we show the largest and the smallest sizes and their ratio of the adaptive mesh at different time levels. From this table, we can see that at the initial time a quite large portion of the grid points is moved to the initial vorticity region due to the singularity of the initial data and make the minimum size of the cell volume very small, see also Fig. 10 (right). Then the ratios of maximum to minimum of the grid size in the x and y directions become bigger. In the late time, the evaluated ratio of largest to the smallest grid size in the y-direction is decreased, while the evaluated ratio of largest to the smallest grid size in the x-direction is firstly increased and then decreased. In Table 2, the maximum and minimum meshes are defined as

$$\min \Delta x = \min_{j,k}\{\Delta x_{j,k}\}, \qquad \max \Delta x = \max_{j,k}\{\Delta x_{j,k}\}$$

where $\Delta x_{j,k} = \max_{p,q\in\{1,2,3,4\}}\{|x_p - x_q|\}$. Here x_p and x_q are the four vertices of the con-

Table 2: Example 4.2: Adaptive mesh with $N_x = N_y = 64$.

		$t=0.0$	$t=5.0$	$t=10.0$	$t=15.0$
Δx	min Δx	9.24e-04	1.91e-03	1.39e-03	2.14e-03
	max Δx	1.48e-01	1.35e-01	1.39e-01	1.75e-01
	max/min	160.17	70.68	100.00	81.78
Δy	min Δy	8.63e-04	9.79e-04	3.47e-03	4.39e-03
	max Δy	1.32e-01	1.71e-01	1.33e-01	1.14e-01
	max/min	152.95	174.67	38.33	25.97
$\lvert A_{jk}\rvert$	min $\lvert A_{jk}\rvert$	3.51e-06	8.99e-06	9.56e-06	1.27e-05
	max $\lvert A_{jk}\rvert$	1.84e-03	2.22e-03	1.78e-03	2.37e-03
	max/min	524.22	246.94	186.19	186.61

trol cell $A_{j+\frac{1}{2},k+\frac{1}{2}}$, and $|A_{j,k}|$ denotes the areas of the cell $A_{j+\frac{1}{2},k+\frac{1}{2}}$ in this table. Similar definition is also used for max Δy and min Δy.

5 Concluding remarks

In this paper, we have presented an adaptive moving mesh technique applied to incompressible viscous flows in the vorticity stream-function formulation. Our moving mesh method is fairly simple to code. The numerical approach is based on the strategy proposed in [20] by decoupling the mesh-moving and PDE evolution at each time step. This approach requires using an interpolation to transform the information from the old mesh to the new mesh. In this work, we have used the second-order conservative interpolation proposed in [33] to update the vorticity ω on the resulting new grid. For the choice of the monitor function, we used an improved parameter-free monitor function [17,18]. The Navier-Stokes equations are numerically solved in the vorticity stream-function form by a finite-volume method. The physical domain is allowed to deforma in our computations and the computed velocity satisfies the discrete divergence constraint. The numerical results are in good agreement with the finer uniform mesh results and the computations found in [11,16,23]; this clearly demonstrates the accuracy and effectiveness of our proposed algorithm. To obtain the equivalent accuracy as for the much finer uniform mesh, our approach only needs much lesser grid points and greatly conserves the CPU time. Future work is to consider the problems with more complex geometry and the extension of our moving mesh algorithm to 3D problems.

References

[1] A. Almgren, J. Bell, P. Colella, L. Howell and M. Welcome, A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations, J. Comput. Phys., 142 (1998), 1-46.

[2] B. N. Azarenok, S. A. Ivanenko and T. Tang, Adaptive mesh redistribution method based on Godunov's scheme, Commun. Math. Sci., 1 (2003), 152-179.
[3] B. N. Azarenok and T. Tang, Second-order Godunov-type scheme for reactive flow calculations on moving meshes, J. Comput. Phys., 206 (2005), 48-80.
[4] G. Beckett, J. A. Mackenzie and M. L. Robertson, An r-adaptive finite element method for the solution of the two-dimensional phase-field equations, Commun. Comput. Phys., 1 (2006), 805-826.
[5] J. U. Brackbill, An adaptive grid with direction control, J. Comput. Phys., 108 (1993), 38-50.
[6] D. L. Brown and M. L. Minion, Performance of under-resolved two-dimensional incompressible flow simulations, J. Comput. Phys., 122 (1995), 165-183.
[7] J. U. Brackbill and J. S. Saltzman, Adaptive zoning for singular problems in two dimensions, J. Comput. Phys., 46 (1982), 342-368.
[8] 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), 221-244.
[9] H. D. Ceniceros and T. Y. Hou, An efficient dynamically adaptive mesh for potentially singular solutions, J. Comput. Phys., 172 (2001), 609-639.
[10] S. F. Davis and J. E. Flaherty, An adaptive finite element method for initial-boundary value problems for partial differential equations, SIAM J. Sci. Stat. Comput., 3 (1982), 6-27.
[11] Y. Di, R. Li, T. Tang and P.-W. Zhang, Moving mesh finite element methods for the incompressible Navier-Stokes equations, SIAM J. Sci. Comput., 26 (2005), 1036-1056.
[12] Y. Di and P.-W. Zhang, Moving mesh kinetic simulation for sheared rodlike polymers with high potential intensities, Commun. Comput. Phys., 1 (2006), 859-873.
[13] H. Ding and C. Shu, A stencil adaptive algorithm for finite difference solution of incompressible viscous flows, J. Comput. Phys., 214 (2006), 397-420.
[14] A. S. Dvinsky, Adaptive grid generation from harmonic maps on Riemannian manifolds, J. Comput. Phys., 95 (1991), 221-244.
[15] W. N. E and J.-G. Liu, Essentially compact schemes for unsteady viscous incompressible flows, J. Comput. Phys., 126 (1996), 122-138.
[16] L. H. Howell and J. B. Bell, An adaptive-mesh projection method for viscous incompressible flow, SIAM J. Sci. Comput., 18 (1997), 996-1013.
[17] W. Z. Huang and R. D. Russell, Moving mesh strategy based on a gradient flow equation for two-dimensional problems, SIAM J. Sci. Comput., 20 (1999), 998-1015.
[18] W. Z. Huang and W. Sun, Variational mesh adaptation II: error estimates and monitor functions, J. Comput. Phys., 184 (2003), 619-648.
[19] S. Li and L. Petzold, Moving mesh methods with upwinding schemes for time-dependent PDEs, J. Comput. Phys., 131 (1997), 368-377.
[20] R. Li, T. Tang and P.-W. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, J. Comput. Phys., 170 (2001), 562-588.
[21] 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), 365-393.
[22] J.-G. Liu and C.-W. Shu, A high-order discontinous Galerkin method for 2D incompressible flows, J. Comput. Phys., 160 (2000), 577-596.
[23] G. Maze, G. Lapeyre and X. Carton, Dynamics of a 2D vortex doublet under external deformation, Reg. Chaot. Dyn., 9 (2004), 477-497.
[24] K. Miller and R. N. Miller, Moving finite element methods I, SIAM J. Numer. Anal., 18 (1981), 1019-1032.
[25] C.-H. Min and F. Gibou, A second order accurate projection method for the incompressible

Navier-Stokes equations on non-graded adaptive grids, J. Comput. Phys., 219 (2006), 912-929.
[26] W. Q. Ren and X. P. Wang, An iterative grid redistribution method for singular problems in multiple dimensions, J. Comput. Phys., 159 (2000), 246-273.
[27] C.-W. Shu and S. Osher, Efficient implement of essentially non-oscillatory shock-wave schemes, II, J. Comput. Phys., 83 (1989), 32-78.
[28] Z.-J. Tan, Adaptive moving mesh methods for two-dimensional resistive magneto-hydrodynamic PDE models, Comput. Fluids, 36 (2007), 758-771.
[29] Z.-J. Tan, K. M. Lim and B. C. Khoo, An adaptive mesh redistribution method for the incompressible mixture flows using phase-field model, J. Comput. Phys., 2007, to appear.
[30] Z.-J. Tan, T. Tang and Z.-R. Zhang, A simple moving mesh method for one- and two-dimensional phase-field equations, J. Comput. Appl. Math., 190 (2006), 252-269.
[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), 347-367.
[32] H. Z. Tang, A moving mesh method for the Euler flow calculations using a directional monitor function, Commun. Comput. Phys., 1 (2006), 656-676.
[33] H. Z. Tang and T. Tang, Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws, SIAM J. Numer. Anal., 41 (2003), 487-515.
[34] H. Z. Tang, T. Tang and P.-W. Zhang, Adaptive mesh redistribution method for nonlinear Hamilton-Jacobi equations in two- and three-dimensions, J. Comput. Phys., 188 (2003), 543-572.
[35] D. W. Waugh, The efficiency of symmetric vortex merger, Phys. Fluids A, 4 (1992), 1745-1758.
[36] A. Winslow, Numerical solution of the quasi-linear Poisson equation in a nonuniform triangle mesh, J. Comput. Phys., 1 (1967), 149-172.
[37] Z.-R. Zhang, Moving mesh method with conservative interpolation based on L^2-Projection, Commun. Comput. Phys., 1 (2006), 930-944.