Contemporary Mathematics

Moving Mesh Methods for Computational Fluid Dynamics

Tao Tang

Abstract. In this paper we will discuss a class of adaptive grid methods called moving mesh method (MMM). Some recent progress of the moving mesh methods will be reviewed. In particular, we review their applications to computational fluid dynamics.

Contents

1. Introduction

In the past two decades, several adaptive strategies have been used for computational fluid dynamics (CFD) problems. We classify them as follows:

- h-method. The h-method involves automatic refinement or coarsening of the spatial mesh based on a posteriori error estimates or error indicators. The overall method contains two independent parts, i.e. a solution algorithm and a mesh selection algorithm.
- p-method. The p-method involves the adaptive enrichment of the polynomial order.
- r-method. The r-method is also known as moving mesh method (MMM). It *relocates* grid points in a mesh having a fixed number of nodes in such

1991 *Mathematics Subject Classification.* 65M93, 35L64, 76N10.

Key words and phrases. Moving mesh methods, computational fluid dynamics, partial differential equation .

a way that the nodes remain concentrated in regions of rapid variation of the solution.

In this article, some recent results on moving mesh methods in CFD will be presented. The moving mesh methods require 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. The key ingredients of the moving mesh methods include:

- **Mesh equations.** The mesh equations determine a one-to-one mapping from a regular domain in a parameter space to an irregularly shaped domain in physical space. 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/difference/element equations. Choosing suitable mesh equations and solving them efficiently are very crucial for an effective moving mesh method;
- **Monitor function.** A monitor function is used to guide the mesh redistribution. It may depend on the solution arclength (in 1D), curvature, and a posteriori errors. In practice, local (spatial) smoothing of the monitor function is necessary, see, e.g., [23, 51];
- **Interpolations.** If the mesh equations are time-dependent and are solved simultaneously with the given differential equations, then interpolation of dependent variables from the old mesh to the new mesh is unnecessary. Otherwise, some kind of interpolation is required to pass the solution information on the old mesh to the newly generated mesh.

In using moving mesh methods for solving CFD problems, extra attentions have to be paid to guarantee that the important properties for the physical solutions will not be lost after moving the grids. For example, to solve the incompressible Navier-Stokes equations in the primitive variables formulation, one of the main difficulties in developing a moving mesh scheme is how to keep the divergence-free property for the velocity field.

The paper is organized as follows. In Section 2, we will review some results on the interpolation-free moving mesh methods. Sections 3 and 4 describe moving mesh methods using interpolations with two different types of discretization approach. The applications of the methods introduced in Sections 3 and 4 are reported in Sections 5 and 6. The concluding remarks will be given in Section 7.

2. Interpolation free MMM

Among moving mesh methods, the moving finite element method of Miller [71, 72] and the moving finite difference method of Dorfi & Drury [34] have aroused considerable interest. Some earlier works were reviewed by Hawken et al. in [45]. In these moving finite element approaches, the mesh equation and the original differential equation are often solved simultaneously for the physical solution and the mesh. Consequently, interpolation of dependent variables from the old mesh to the new mesh is unnecessary. In this sense, we call this class of methods *interpolation-free* moving mesh methods.

The principal ingredients of the interpolation-free moving mesh methods include:

- **Equidistribution principle.** It is first introduced by de Boor [30] for solving boundary value problems for ordinary differential equations. It involves selecting mesh points such that some measure of the solution error is equalized over each subinterval. It has turned out to be an excellent principle for formulating moving mesh equations.
- **Moving mesh equation.** The major role of the moving mesh equation is to concentrate sufficient points in regions of rapid variation of the solution. A satisfactory mesh equation should be simple, easy to program, and reasonably insensitive to the choice of its adjustable parameters. As compared with the problem of discretizing the underlying physical equation, this task is purely artificial. That is, the construction of a moving mesh equation cannot be guided completely by physical arguments and must rely on some numerical principles.
- **The method of lines (MOL) approach.** Most moving mesh codes are designed with the method of lines approach. In MOL, the partial differential equations are first discretized by a finite difference or a finite element method. The resulting equations form a system of ordinary differential equations (ODEs) or differential algebraic equations (DAEs). Then existing ODE or DAE softwares can be used for the time integration. After coupling the moving mesh equation with the original physical equations, the system of ODEs usually becomes strongly stiff because of the irregularity of the grid and the large time scale in controlling the mesh moving. This is why most of the moving mesh codes use a stiff ODE solver for time integration.

2.1. Moving mesh finite element approaches. The moving finite element method (MFEM) uses a very natural and elegant formulation to control mesh movement. The solution and mesh are both obtained by a process closely associated with equidistribution of one error measure: the residual of the original equation written in finite element form. The MFEM can be strongly linked to some equidistribution principles when it is applied to parabolic differential equations. Adjerid & Flaherty [1, 2, 5] introduced an error estimate to handle mesh movement based on an equidistribution principle. In [3, 4, 15, 16, 46], several other moving mesh methods are developed, based directly on equidistribution principles. Nevertheless, the constructions are very different, and in their final forms the moving mesh equations appear to be quite different from each other. In [85], a moving grid finite-element method using grid deformation is proposed and studied.

In [73], some results on adaptive methods in CFD were reviewed, where an *h*-*r* scheme was applied to shock computations (see also [37]).

The two basic types of methods, location based and velocity based, generally involve respectively computing x by minimizing a variational form or computing the mesh velocity $v = x_t$ using a Lagrangian like formulation. For the location based methods, several variational forms were reviewed. One common type of method involves solving the variational problem using a steepest descent method to introduce the time derivative for grid movement, see, e.g., [10–12, 17].

2.2. Moving mesh finite difference approaches. Several moving mesh finite difference algorithms have been developed in past years. One was given by Verwer and his research group at CWI (see, e.g., [40, 99]). Their moving mesh technique is due to the moving finite difference method proposed by Dorfi & Drury [34]. The spatial discretization is done by a second order nonlinear Galerkin-based method, which is studied extensively in Skeel & Berzins [87]. Further works along this line have been done by Zegeling [33, 103, 104, 106].

The moving finite difference algorithms are also developed by Russell and his research group, see., e.g., [49, 50, 82, 83]. In particular, the moving mesh equation is written as moving mesh partial differential equations (MMPDEs) based on the equidistribution principle. The physical equations are basically discretized using central differencing. Some practical aspects of formulation and solution of moving mesh partial differential equations are reviewed in Huang [47]. The stability for this type of moving mesh methods was studied in [64]. One of the successful applications of the moving mesh PDE approach is given in [21] to solve (one-dimensional) PDEs with blow up solutions (see also [65]).

2.3. Some discussions on interpolation-free MMMs. It is observed from many numerical experiments that the interpolation-free moving mesh methods enjoy several numerical advantages:

- They help the central difference to work for problems with large solution gradients or discontinuities, e.g., in resolving small-scale structures in Boussinesq convection [25];
- They reduce the time variation to allow larger time step, due to the use of the mesh velocity [82];
- They do not need interpolations between regriddings – unlike static methods, interpolation of dependent variables from the old mesh to the new mesh is unnecessary. This is one of the major advantages of this approach;
- They can automatically detect, resolve, and track steep wave fronts and moving boundaries.

The disadvantages of this class of moving mesh methods include:

- moving mesh method increases the stiffness of the system so it requires implicit time integration to work efficiently;
- choosing time scale τ in equidistributing moving mesh methods is highly heuristic.

As in many other moving mesh codes, there are several parameters that should be specified by the user in the setting-up routine. Most of them are insensitive to different applications and can be chosen by default. The most critical parameter in equidistributing mesh moving strategies is the time scale τ. Many heuristic guidelines for how to choose τ have been proposed in the past decade, but none of them seems to work for all of the applications. This is especially true for explicit integration, where a small τ can result in mesh crossing and failures in integrations. For implicit integration methods, changing τ can result in a new evaluation of the Jacobian for the Newton iteration.

3. MMM using interpolation I: finite element approach

This section describes moving finite element algorithms proposed in [61, 62]. The moving finite element method proposed in [61] contains two independent 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, which is different with most of the interpolation-free moving mesh methods. The algorithm proposed in [61] keeps the advantages of the moving mesh method (e.g., keep the number of nodes unchanged) and of the h-method (e.g., the two parts in the code are independent of each other). The simplicity and reliability of this approach were demonstrated by a number of numerical examples in two and three space dimensions, as to be reported in Section 5.

To completely specify the coordinate transformation, the moving mesh methods must be supplemented with suitable boundary conditions. In theory, as pointed out in [19, 22], there are a number of ways to redistribute the mesh points on the boundary, such as using homogeneous Neumann boundary conditions, extrapolating the interior mesh points to the boundary, and relocating the mesh points by solving a lower-dimensional moving-mesh PDE. However, these methods seem quite inefficient if 3D problems are being considered. In [62], we presented a moving mesh method, which is based on the minimization of the mesh energy, for solving problems in two and three space dimensions. In the mesh-restructuring step, we solve an *optimization problem* with some appropriate constraints, which is in contrast to the traditional method of solving the Euler-Lagrange equation directly. The key idea of this approach is to treat the interior and boundary grids as a whole, rather than considering them separately. Therefore, the new solution algorithm also provides an alternative boundary grid re-distribution technique, which turns out to be useful in solving 3D problems. We point out that there are other mesh movement approaches that can deal with boundary and interior points in a unified manner. For example, to solve variational problems the approach by Tourigny & Hülsemann [97] is dimension independent.

3.1. Harmonic mapping. Let Ω and Ω_c be compact Riemannian manifolds of dimension n with metric tensors d_{ij} and $r_{\alpha\beta}$ in some local coordinates $\vec{x}$ and $\vec{\xi}$, respectively. Following Dvinsky [36] and Brackbill [19], we define the energy for a map $\vec{\xi} = \vec{\xi}(\vec{x})$ as

$$E(\vec{\xi}) = \frac{1}{2} \int \sqrt{d}\, d^{ij} r_{\alpha\beta} \frac{\partial \xi^\alpha}{\partial x^i} \frac{\partial \xi^\beta}{\partial x^j} d\vec{x}\,, \tag{3.1}$$

where $d = \det(d_{ij})$, $(d_{ij}) = (d^{ij})^{-1}$, and the standard summation convention is assumed. The Euler-Lagrange equations, whose solution minimizes the above energy, with an Euclidean metric are given by

$$\frac{\partial}{\partial x^i} \sqrt{d}\, d^{ij} \frac{\partial \xi^k}{\partial x^j} = 0\,. \tag{3.2}$$

We emphasize that $d = \det(d_{ij}) = 1/\det(d^{ij})$. For ease of notation, we let $G^{ij} = \sqrt{d}\, d^{ij}$. The inverse of (G^{ij}) is called *monitor functions*. Therefore, the Euler-Lagrange equations, with Euclidean metric for the logical domain Ω_c, are given by

$$\frac{\partial}{\partial x^i}\left(G^{ij}\frac{\partial \xi^k}{\partial x^j}\right) = 0, \tag{3.3}$$

and the corresponding mesh energy is of the form

$$E(\vec{\xi}) = \sum_k \int_\Omega G^{ij}\frac{\partial \xi^k}{\partial x^i}\frac{\partial \xi^k}{\partial x^j}d\vec{x}\,. \tag{3.4}$$

Solutions to (3.3) are harmonic functions giving a continuous, one-to-one mapping with continuous inverse, which is differentiable and has a non-zero Jacobian. A detailed description for solving (3.3), as well as how to interchange its dependent and independent variables, can be found in Li et. al. [57,61].

In [36], Dvinsky suggests that harmonic function theory may provide a general framework for developing useful mesh generators. A good feature of the adaptive methods based on harmonic mapping is that existence, uniqueness and non-singularity for the continuous map can be guaranteed from the theory of harmonic maps: The existence and uniqueness of harmonic maps are established by Hamilton [43] and Schoen & Yau [84].

3.2. The moving mesh algorithm. In [62], a moving finite element method is proposed, which re-distributes the interior and boundary grids *simultaneously*. For simplicity, assume that the domain Ω is a polyhedron. Let Γ_i and $\Gamma_{c,i}$ denote the corresponding edges of Ω and Ω_c, respectively. The geometrical constraint to the boundary grid movement is to keep the geometrical character of the physical domain unchanged. This implies that the vertices (edges) of the physical domain will be mapped to the corresponding vertices (edges) of the computational domain. Therefore, it is reasonable to consider the following mapping set from $\partial\Omega$ to $\partial\Omega_c$,

$$\begin{aligned} K \;=\; \{\xi_b \in C^0(\partial\Omega) \;\Big|\; & \xi_b : \partial\Omega \to \partial\Omega_c\,; \\ & \xi_b|_{\Gamma_i} \text{is a linear segment and strictly increasing.}\} \end{aligned} \tag{3.5}$$

One example of such a map is illustrated in Fig. 1. It follows from the theory of Eell & Sampson [38] that for every $\xi_b \in \mathbf{K}$ there exists a unique $\xi : \Omega \to \Omega_c$, such that $\xi|_{\partial\Omega} = \xi_b$ and ξ is the extreme of the functional (3.4). Let us denote the mapping as $\xi = P(\xi_b)$, and consider the optimization problem

$$\begin{array}{rl} \min & E(P(\xi_b)) \\ s.t. & \xi_b \in \mathbf{K} \end{array} \tag{3.6}$$

where the functional E is defined by (3.4). Since E is convex and P is linear, it is shown in [62] that $E(P(\cdot))$ is a convex functional. Therefore, the optimization problem (3.6) has a unique solution in a closed subset of $\mathbf{K}$.

Based on the above discussion, we will solve the following constrained optimization problem:

$$\begin{array}{rl} \min & \displaystyle\sum_k \int_\Omega G^{ij}\frac{\partial \xi^k}{\partial x^i}\frac{\partial \xi^k}{\partial x^j}d\vec{x} \\ s.t. & \xi|_{\partial\Omega} = \xi_b \in \mathbf{K}\,. \end{array} \tag{3.7}$$

Note that the boundary values ξ_b are not fixed, instead they are unknowns in the same way as the interior points. This is one of the main differences between the present approach and the one proposed in [61] where a Dirichlet problem is solved

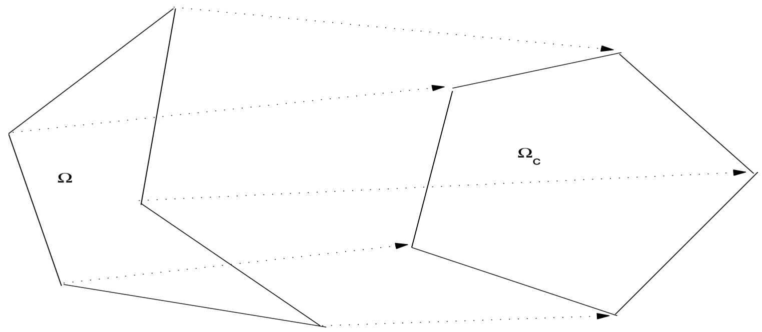


FIGURE 1. A map between $\partial\Omega$ and $\partial\Omega_c$.

for ξ. By solving the above problem, the meshes on the logical domain will be obtained at each iteration. The difference between the initial (fixed) mesh and the updated mesh in the logical domain will yield the grid re-distribution for both the interior *and* boundary grids (based on the formula (3.19) to be given in the next subsection). One advantage of this approach is that the overall moving mesh scheme can be easily implemented for 3D problems.

Once the initial mesh (in the logical domain) is given, it will be kept *unchanged* throughout the computation. This initial mesh in Ω_c, denoted by $\vec{\xi}^{(0)}$, is used as a reference grid only. After the solution u is computed at time level $t = t_n$, the inverse matrix of the monitor, G^{ij} (which in general depends on u), can be updated. By solving (3.7), we will obtain a mesh in the logical domain, denoted by $\vec{\xi}^*$. If the difference between this $\vec{\xi}^*$ and the initial mesh $\vec{\xi}^{(0)}$ is not small, we move the mesh in the physical space to obtain the updated values for u in the resulting new grid. This idea leads to the following algorithm.

ALGORITHM 1: Mesh-redistribution algorithm

- **(a):** solve the optimization problem (3.7) and compute the L^∞-difference between the solution of (3.7) and the fixed (initial) mesh in the logical domain. If the difference is smaller than a preassigned tolerance, then the mesh-redistribution at the given time level is complete. Otherwise, do
- **(b):** obtain the direction and the magnitude of the movement for $\vec{x}$ by using the difference obtained in part **(a)**, see (3.18) in Sect. 3.3, and then move the mesh based on (3.19);
- **(c):** update $\vec{u}$ on the new grid by solving a system of ODEs, see (3.25) in Sect. 3.3;
- **(d):** update the monitor function by using $\vec{u}$ obtained in part **(c)**, and go to part **(a)**.

In part **(a)**, a preassigned tolerance `TOL` is chosen so that the iteration is stopped when

$$\|\xi^* - \xi^{(0)}\|_{L^\infty} \le \texttt{TOL}\,. \tag{3.8}$$

The iteration above determines *progressively* better locations of the mesh grids in the physical domain. Typically about one to two iterations are required. The parts (b) and (c) above have been discussed in detail in Li et. al. [61]. We refer the reader to that paper for the algorithm details, although some necessary details will be mentioned in the next subsection.

After the interior and boundary grid points are well re-distributed based on the solution at $t = t_n$, we can use some appropriate numerical methods to solve the underlying PDEs at $t = t_{n+1}$ on the updated mesh in the physical space.

3.3. Mesh-redistribution and solution-updating. This subsection is to give some necessary details for Algorithm 1, the mesh-redistribution algorithm. Let us discretize the optimization problem (3.7) in the linear finite element space. The triangulation of the physical domain is $\mathcal{T}$, with T_i as its elements, and X_i as its nodes. The corresponding triangulation on the computational domain is $\mathcal{T}_c$, with $T_{i,c}$ as its elements, and $\mathcal{A}_i$ as its nodes. The linear finite element space on the mesh is denoted as $H_h^1(\Omega)$. If the basis function on node X_i is denoted by ϕ^i, then ξ can be approximated by $\xi_i\phi^i$ (here the standard summation convention is assumed). The coordinates of X_i are $(X_i^1\ X_i^2)^T$. Let the inner nodes be indexed from 1 to N_{inner} and the boundary nodes be indexed from $N_{inner}+1$ to N. The coordinates of the nodes $\mathcal{A}_i$ in the computational domain are denoted as $(\mathcal{A}_i^1\ \mathcal{A}_i^2)^T$. Denote $X = (X^1\ X^2)^T$, $\mathcal{A} = (\mathcal{A}^1\ \mathcal{A}^2)^T$, where $X^k = (X_1^k\ \cdots\ X_N^k)^T$, $\mathcal{A}^k = (\mathcal{A}_1^k\ \cdots\ \mathcal{A}_N^k)^T$, $k = 1, 2$. The objective function in (3.7) is approximated by

$$\sum_k \int_\Omega G^{ij} \frac{\partial \phi^\alpha}{\partial x^i} \frac{\partial \phi^\beta}{\partial x^j} d\vec{x}\, \xi_\alpha^k \xi_\beta^k. \tag{3.9}$$

As for the boundary points, we recall the assumption that ξ map a boundary (linear) segment L on $\partial\Omega$ to a linear segment L_c on $\partial\Omega_c$. This gives that

$$< \mathcal{A}_i, \mathbf{n}^i >= b^i \tag{3.10}$$

where $<>$ denotes the standard inner product, $\mathbf{n}^i$ is the normal direction of a fixed segment of the boundary of Ω_c and b^i is a given number. Since each $X_i \in \Gamma_i$ is mapped to a known segment of $\Gamma_{i,c}$, the relevant $\mathbf{n}^i$ and b^i are determined.

3.3.1. *Mesh-redistribution.* We now discuss the part (b) of Algorithm 1. First a linear system for $\mathcal{A}$ will be formed to determine the motion of the computational grids. Denote

$$H = \left(\int_\Omega G^{ij} \frac{\partial \phi^\alpha}{\partial x^i} \frac{\partial \phi^\beta}{\partial x^j} d\vec{x} \right)_{1\le\alpha,\beta\le N.} \tag{3.11}$$

We further split the matrix H into the following form:

$$\begin{array}{cccc} H = & \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix} & & \begin{array}{l} \leftarrow 1 \text{ to } N_{inner} \text{ row} \\ \leftarrow N_{inner}+1 \text{ to } N \text{ row} \end{array} \\ & \begin{array}{cc} \uparrow & \uparrow \\ 1 \text{ to } N_{inner} \text{ column} & N_{inner}+1 \text{ to } N \text{ column} \end{array} & & \end{array}$$

Correspondingly, we use $\mathcal{A}_{inner}$ and $\mathcal{A}_{bound}$ to denote the interior and boundary part of the node coordinates, respectively. Then the objective function is given by

$$\begin{pmatrix} \mathcal{A}^{1,T} & \mathcal{A}^{2,T} \end{pmatrix} \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \begin{pmatrix} \mathcal{A}^1 \\ \mathcal{A}^2 \end{pmatrix}. \tag{3.12}$$

Recall the earlier assumption that ξ maps a (linear) boundary segment L on $\partial\Omega$ to a linear segment L_c on $\partial\Omega_c$. This assumption leads to the following linear system

$$\begin{pmatrix} 0 & A_{12} & 0 & A_{22} \end{pmatrix} \begin{pmatrix} \mathcal{A}^1_{inner} \\ \mathcal{A}^1_{bound} \\ \mathcal{A}^2_{inner} \\ \mathcal{A}^2_{bound} \end{pmatrix} = \vec{b}, \tag{3.13}$$

where the matrices A_{12} and A_{22}, based on (3.10), are the entries of the unit normal of the boundary segments.

With the above preparation, the optimization problem (3.7) is equivalent to solving the following linear system:

$$\begin{pmatrix} H_{11} & H_{12} & 0 & 0 & 0 \\ H_{21} & H_{22} & 0 & 0 & A'_{12} \\ 0 & 0 & H_{11} & H_{12} & 0 \\ 0 & 0 & H_{21} & H_{22} & A'_{22} \\ 0 & A_{12} & 0 & A_{22} & 0 \end{pmatrix} \begin{pmatrix} \mathcal{A}^1_{inner} \\ \mathcal{A}^1_{bound} \\ \mathcal{A}^2_{inner} \\ \mathcal{A}^2_{bound} \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vec{b} \end{pmatrix} \tag{3.14}$$

where λ is the Lagrange multiplier. As the number of points in the computational domain increases, solving the above system efficiently becomes crucial. In our computations, two methods were tested: the first one uses BiCG with an LU preconditioner to solve the system (3.14), and the second one is somehow more efficient but not an exact method. Here we briefly outline the idea of the second method. It is to decouple the above system to the following forms:

$$\begin{pmatrix} H_{22} & 0 & A'_{12} \\ 0 & H_{22} & A'_{22} \\ A_{12} & A_{22} & 0 \end{pmatrix} \begin{pmatrix} \mathcal{A}^1_{bound} \\ \mathcal{A}^2_{bound} \\ \lambda \end{pmatrix} = \begin{pmatrix} -H_{21}\mathcal{A}^1_{inner} \\ -H_{21}\mathcal{A}^2_{inner} \\ \vec{b} \end{pmatrix} \tag{3.15}$$

and

$$\begin{pmatrix} H_{11} & 0 \\ 0 & H_{11} \end{pmatrix} \begin{pmatrix} \mathcal{A}^1_{inner} \\ \mathcal{A}^2_{inner} \end{pmatrix} = - \begin{pmatrix} H_{12}\mathcal{A}^1_{bound} \\ H_{12}\mathcal{A}^2_{bound} \end{pmatrix}. \tag{3.16}$$

The system (3.15) is small and is solved efficiently by using BiCG and GMRES. The system (3.16) is symmetric and positive definite and is solved by using a multigrid solver. Since the accuracy of the system is not very crucial (the solution of the system (3.14) is for the location of mesh but not for the physical solution), a non-exact but efficient algorithm is more useful. Therefore, we prefer to use the second method in obtaining the approximate solutions of the system (3.14).

After obtaining the solution of (3.14), we can obtain a new logical mesh $\mathcal{T}_c^*$ with nodes $\mathcal{A}^*$. For a given element E in $\mathcal{T}$, 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

$$\begin{aligned} &\begin{pmatrix} \mathcal{A}^{*,1}_{E_1} - \mathcal{A}^{*,1}_{E_0} & \mathcal{A}^{*,1}_{E_2} - \mathcal{A}^{*,1}_{E_0} \\ \mathcal{A}^{*,2}_{E_1} - \mathcal{A}^{*,2}_{E_0} & \mathcal{A}^{*,2}_{E_2} - \mathcal{A}^{*,2}_{E_0} \end{pmatrix} \begin{pmatrix} \dfrac{\partial x^1}{\partial \xi^1} & \dfrac{\partial x^1}{\partial \xi^2} \\ \dfrac{\partial x^2}{\partial \xi^1} & \dfrac{\partial x^2}{\partial \xi^2} \end{pmatrix} \\ &= \begin{pmatrix} X^1_{E_1} - X^1_{E_0} & X^1_{E_2} - X^1_{E_0} \\ X^2_{E_1} - X^2_{E_0} & X^2_{E_2} - X^2_{E_0} \end{pmatrix}. \end{aligned} \tag{3.17}$$

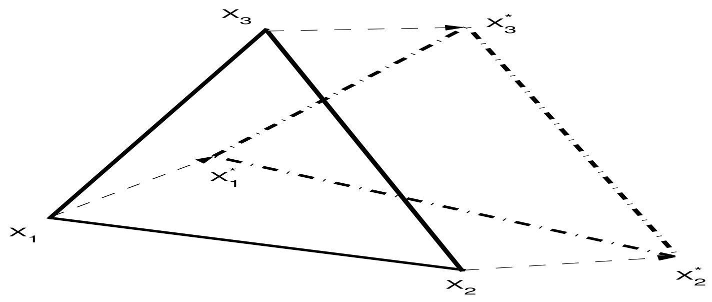


FIGURE 2. A demonstration of the element motion.

Solving (3.17) gives $\partial\vec{x}/\partial\xi$ in E. If we take the volume of the element as the weight, the weighted average error of X at the i-th node is defined by

$$\delta X_i = \frac{\displaystyle\sum_{E\in T_i} |E| \left.\frac{\partial\vec{x}}{\partial\xi}\right|_{in\ E} \delta\mathcal{A}_i}{\displaystyle\sum_{E\in T_i} |E|} \tag{3.18}$$

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 $\mathcal{A}^*$). It can be shown that the above volume weighted average converges to a smooth solution in measure when the size of mesh goes to 0. The location of the nodes in the new mesh $\mathcal{T}^*$ on the physical domain is taken as

$$X^* = X + \tau\delta X \tag{3.19}$$

where τ is a parameter in $[0, 1]$ and is used to prevent mesh tangling. Some possible choices for τ can be found in [61, 62]. The motion of one element based on (3.19) is illustrated in Fig. 2.

3.3.2. *Solution-updating.* After the mesh-redistribution in the physical domain Ω, we need to update the solution $\vec{u}$ on the new mesh. Each element of $\mathcal{T}$ with X as its nodes corresponds uniquely to an element of $\mathcal{T}^*(\tau)$ with $X + \tau\delta X$ as its nodes. There is also an affine map between the two elements. By combining all those affine maps from each element of $\mathcal{T}^*(\tau)$ to $\mathcal{T}$, we obtain a map from Ω_c to Ω piecewise affine. The *surface of $\vec{u}$ on Ω will not move*, though the nodes of the mesh may be moved to new locations. Then $\vec{u}$, as the function of $\vec{x}$ at time t_n, is independent of the parameter τ. That is

$$\frac{\partial\vec{u}}{\partial\tau} = 0\,. \tag{3.20}$$

During the mesh redistribution, $\vec{u}$ is expressed as $\vec{u} = \vec{u}(\vec{x}, \tau)$. In the finite element space, $\vec{u}$ is expressed as

$$\vec{u} = U_i(\tau)\Phi^i(\vec{x}, \tau) \tag{3.21}$$

where $\Phi^i(\vec{x},\tau)$ is the basis function of the finite element space at its node $X_i+\tau\delta X_i$. Direct computation gives

$$\frac{\partial \Phi^i(\vec{x},\tau)}{\partial \tau} = -\frac{\partial \Phi^i(\vec{x},\tau)}{\partial x^j}(\delta\vec{x})_j\,, \tag{3.22}$$

where $\delta\vec{x} := \delta X_i \Phi^i$. Differentiating $\vec{u}$ with respect to τ gives

$$\begin{aligned} 0 = \frac{\partial \vec{u}}{\partial \tau} &= \frac{\partial U_i}{\partial \tau}\Phi^i(\vec{x},\tau) + U_i(\tau)\frac{\partial \Phi^i}{\partial \tau} \\ &= \frac{\partial U_i}{\partial \tau}\Phi^i(\vec{x},\tau) - U_i(\tau)\frac{\partial \Phi^i}{\partial x^j}(\delta\vec{x})_j\,. \end{aligned} \tag{3.23}$$

Using the expression for $\vec{u}$ in the finite element space, i.e. (3.21), we obtain from the above result that

$$\frac{\partial U_i}{\partial \tau}\Phi^i(\vec{x},\tau) - \nabla_{\vec{x}}\vec{u}\delta\vec{x} = 0\,. \tag{3.24}$$

Then a semi-discrete system for updating $\vec{u}$ follows from the above result.

$$\int_\Omega \left\{ \frac{\partial U_i}{\partial \tau}\Phi^i(\vec{x},\tau) - \nabla_{\vec{x}}\vec{u}\delta\vec{x} \right\} v d\vec{x} = 0 \qquad \forall v \in V_{\mathcal{T}}(\Omega)\,.$$

By letting v be the basis function of $V_{\mathcal{T}}(\Omega)$, i.e. $v = \Phi^j(\vec{x},\tau)$, we obtain a system of (linear) ODEs for U_i:

$$\int_\Omega \Phi^i\Phi^j d\vec{x}\,\frac{\partial U_i}{\partial \tau} = \int_\Omega \frac{\partial \Phi^i}{\partial x^k}(\delta\vec{x})_k \Phi^j d\vec{x}\, U_i(\tau)\,, \qquad 1 \le j \le N. \tag{3.25}$$

The above ODE system can be solved by a 3-stage Runge-Kutta scheme. This procedure, based on the fact that the surface of $\vec{u}$ in Ω is unchanged, provides a solution-updating formula.

The above numerical procedure has now been extended to solve several classes of problems such as the incompressible Navier-Stokes equations [32], incompressible interface problems [31], and elliptic optimal control problems [59]. One of the primary features of the numerical scheme is that the mesh redistribution part and the PDE evolution part are separated. As a result, the whole moving mesh algorithm can be packed in a black box which requires the following inputs: the current numerical solution of the underlying PDEs, the algorithm for solving the mesh PDEs, and an interpolation algorithm. Such a black box has been implemented in the adaptive finite element package AFEPack which is available at `http://circus.math.pku.edu.cn/AFEPack`, see also [58].

4. MMM using interpolation II: finite volume approach

4.1. 1D Algorithm. Let x and ξ denote the physical and computational coordinates, respectively, which are (without loss of generality) assumed to be in $[a,b]$ and $[0,1]$, respectively. A one–to–one coordinate transformation between these domains is denoted by

$$\begin{aligned} x &= x(\xi), \qquad && \xi \in [0,1], \\ x(0) &= a, && x(1) = b. \end{aligned} \tag{4.1}$$

The 1D Euler-Lagrange equation has the form

$$(\omega^{-1}\xi_x)_x = 0. \tag{4.2}$$

Using the above equation we can obtain the conventional 1D *equidistribution principle*: ωx_ξ=constant or equivalently

$$(\omega x_\xi)_\xi = 0. \tag{4.3}$$

Both equations (4.2) and (4.3) have the same form and therefore solving any one of them will end up with the desired mesh map $x = x(\xi)$. However, the situation is different in the 2D case where we will choose to solve equations of the form (4.3), as will be described in the next subsection.

Our solution procedure is based on two independent parts: a mesh-redistribution algorithm and a solution algorithm. The first part will be based on an iteration procedure using (4.3) for mesh-motion and (4.5) below for solution updating. The second part will be independent of the first one, and it can be any of the standard codes for solving the given PDEs. The solution procedure can be illustrated by the following flowchart:

ALGORITHM 2: 1D finite-volume moving mesh algorithm.

Step 1: Given a uniform (fixed) partition of the logical domain Ω_c, use the equidistribution principle (4.3) to generate an initial partition $x_j^{[0]} := x_j$ of the physical domain Ω_p. Then compute the grid values $u_{j+\frac{1}{2}}^{[0]}$ based on the cell average for the initial data $u(x, 0)$.

Step 2: Move grid $\{x_j^{[\nu]}\}$ to $\{x_j^{[\nu+1]}\}$ by solving (4.3) numerically, say with one or more Gauss-Seidel iterations. Then compute $\{u_{j+\frac{1}{2}}^{[\nu+1]}\}$ on the new grid based on the interpolation formula (4.5) to be given below. Repeat the updating procedure for a fixed number of iterations or until $\|x^{[\nu+1]} - x^{[\nu]}\| \le \epsilon$.

Step 3: Evolve the underlying PDEs using a high-resolution finite volume method on the mesh $\{x_j^{[\nu+1]}\}$ to obtain the numerical approximations $u_{j+\frac{1}{2}}^{n+1}$ at the time level t_{n+1}.

Step 4: If $t_{n+1} \le T$, then let $u_{j+\frac{1}{2}}^{[0]} := u_{j+\frac{1}{2}}^{n+1}$ and $x_j^{[0]} := x_j^{[\nu+1]}$ and go to **Step 2**.

After obtaining the new grid $\{\widetilde{x}_j\}$, we need to update u at the grid point $\widetilde{x}_{j+\frac{1}{2}} = (\widetilde{x}_j + \widetilde{x}_{j+1})/2$ based on the knowledge of $\{x_{j+\frac{1}{2}}, \widetilde{x}_{j+\frac{1}{2}}, u_{j+\frac{1}{2}}\}$. In [94], a conservative second-order interpolation formula is introduced to update u on the new grid. Below we will briefly derive it using the classical perturbation method. Assume the difference between $\tilde{x}_{j+\frac{1}{2}}$ and $x_{j+\frac{1}{2}}$ is small. Let $\widetilde{u}_{j+\frac{1}{2}}$ and $u_{j+\frac{1}{2}}$ be cell averages of the solution $u(x)$ over the intervals $[\widetilde{x}_j, \widetilde{x}_{j+1}]$ and $[x_j, x_{j+1}]$, respectively. If $\widetilde{x} = x - c(x)$ with a small displacement $c(x)$, i.e. $|c(x)| \ll 1$, then we have

$$\begin{aligned} \int_{\widetilde{x}_j}^{\widetilde{x}_{j+1}} \tilde{u}(\widetilde{x})\, d\widetilde{x} &= \int_{x_j}^{x_{j+1}} u(x - c(x))(1 - c'(x))\, dx \\ &\approx \int_{x_j}^{x_{j+1}} (u(x) - c(x)u_x(x))(1 - c'(x))\, dx \approx \int_{x_j}^{x_{j+1}} (u(x) - (cu)_x)\, dx \\ &= \int_{x_j}^{x_{j+1}} u(x)\, dx - ((cu)_{j+1} - (cu)_j) \end{aligned} \tag{4.4}$$

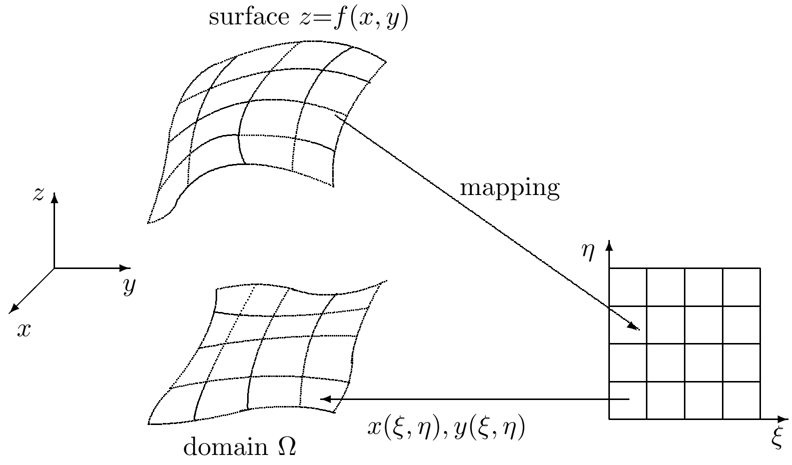


FIGURE 3. The coordinates on the surface of a function $z=f(x,y)$.

where we have neglected higher order terms, and $(cu)_j$ denotes the value of cu at the j-th cell interface. The following conservative-interpolation formula follows from (4.4):

$$\Delta\widetilde{x}_{j+\frac{1}{2}}\widetilde{u}_{j+\frac{1}{2}} = \Delta x_{j+\frac{1}{2}}u_{j+\frac{1}{2}} - \big((cu)_{j+1} - (cu)_j\big), \tag{4.5}$$

where $\Delta\widetilde{x}_{j+\frac{1}{2}} = \widetilde{x}_{j+1} - \widetilde{x}_j$ and $c_j = x_j - \widetilde{x}_j$. Note that the above solution-updating method guarantees the conservation of mass in the following sense:

$$\sum_j \Delta\widetilde{x}_{j+\frac{1}{2}}\widetilde{u}_{j+\frac{1}{2}} = \sum_j \Delta x_{j+\frac{1}{2}}u_{j+\frac{1}{2}}. \tag{4.6}$$

The linear flux cu in (4.5) will be approximated by some upwinding numerical flux, see [94].

In 1D, it can be shown that the underlying numerical approximation obtained in the mesh-redistribution part satisfies the desired TVD property [94], which guarantees that the numerical solution at any time level is TVD provided that the PDE solver in the first part satisfies such a property.

4.2. 2D Algorithm. One of the advantages of the adaptive mesh methods described in the last section is that they can be naturally extended to 2D. In the following, we briefly discuss this extension. We begin by extending the conservative-interpolation formula (4.5). In two dimensions, the coordinates and mapping relationships are shown in Fig. 3. Without loss of generality, assume the logical domain $\bar{\Omega}_c = \{(\xi,\eta)|0 \le \xi \le 1, 0 \le \eta \le 1\}$ is covered by the square mesh:

$$\left\{(\xi_j,\eta_k)\,\Big|\,\xi_j = j/(J_x+1),\, \eta_k = k/(J_y+1);\; 0 \le j \le J_x+1, 0 \le k \le J_y+1\right\}.$$

The numerical approximations to $x = x(\xi,\eta)$ and $y = y(\xi,\eta)$ are denoted by $x_{j,k} = x(\xi_j,\eta_k)$ and $y_{j,k} = y(\xi_j,\eta_k)$, respectively. Let $A_{j+\frac{1}{2},k+\frac{1}{2}}$ and $\widetilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}$ be control volumes with four vertices $(x_{j+p,k+q}, y_{j+p,k+q})$ and $(\widetilde{x}_{j+p,k+q}, \widetilde{y}_{j+p,k+q})$, $0 \le p,q \le 1$, respectively. Assume that $\widetilde{u}_{j+\frac{1}{2},k+\frac{1}{2}}$ and $u_{j+\frac{1}{2},k+\frac{1}{2}}$ are cell averages of $u(x,,y)$ over $\widetilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}$ and $A_{j+\frac{1}{2},k+\frac{1}{2}}$, respectively. As in the 1D case, we use the perturbation method to evaluate the numerical approximation on the resulting new grids $(\widetilde{x}_{j,k}, \widetilde{y}_{j,k})$. If $(\widetilde{x}, \widetilde{y}) = (x - c^x(x,y),\, y - c^y(x,y))$ where we assume the speeds (c^x, c^y) have small amplitude, then it can be shown

$$\begin{aligned}\int_{\widetilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}} \widetilde{u}(\widetilde{x},\widetilde{y})\, d\widetilde{x}d\widetilde{y} &\approx \int_{A_{j+\frac{1}{2},k+\frac{1}{2}}} u(x,y)\, dxdy \\ &-\Big[(c_n u)_{j+1,k+\frac{1}{2}} + (c_n u)_{j,k+\frac{1}{2}}\Big] - \Big[(c_n u)_{j+\frac{1}{2},k+1} + (c_n u)_{j+\frac{1}{2},k}\Big],\end{aligned} \tag{4.7}$$

where we have neglected higher order terms, $c_n := c^x n_x + c^y n_y$ with (n_x, n_y) the unit normal, and $(c_n u)_{j,k+\frac{1}{2}}$ and $(c_n u)_{j+\frac{1}{2},k}$ denote the values of $c_n u$ at the corresponding surfaces of the control volume $A_{j+\frac{1}{2},k+\frac{1}{2}}$. From (4.7), we obtain a conservative-interpolation:

$$\begin{aligned}|\widetilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}|\widetilde{u}_{j+\frac{1}{2},k+\frac{1}{2}} &= |A_{j+\frac{1}{2},k+\frac{1}{2}}|u_{j+\frac{1}{2},k+\frac{1}{2}} \\ &-\Big[(c_n u)_{j+1,k+\frac{1}{2}} + (c_n u)_{j,k+\frac{1}{2}}\Big] - \Big[(c_n u)_{j+\frac{1}{2},k+1} + (c_n u)_{j+\frac{1}{2},k}\Big],\end{aligned} \tag{4.8}$$

where $|\widetilde{A}|$ and $|A|$ denote the areas of the control volumes $\widetilde{A}$ and A, respectively. It can be verified that the above solution-updating scheme satisfies mass-conservation:

$$\sum_{j,k} |\widetilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}|\widetilde{u}_{j+\frac{1}{2},k+\frac{1}{2}} = \sum_{j,k} |A_{j+\frac{1}{2},k+\frac{1}{2}}|u_{j+\frac{1}{2},k+\frac{1}{2}}. \tag{4.9}$$

The solution procedure of our adaptive mesh strategy for two-dimensional hyperbolic problems is almost the same as ALGORITHM 2 provided in Section 4.1. Some details of the steps used for our 2D algorithm are given below.

ALGORITHM 3: 2D finite-volume moving mesh algorithm.

Step i: Give an initial partition $\vec{z}^{[0]}_{j,k} = \big(x^{[0]}_{j,k}, y^{[0]}_{j,k}\big) := (x_{j,k}, y_{j,k})$ of the physical domain Ω_p and a uniform (fixed) partition of the logical domain Ω_c, and compute grid values $u^{[0]}_{j+\frac{1}{2},k+\frac{1}{2}}$ by cell averaging the initial data $u(x,y,0)$ over the control volume $A_{j+\frac{1}{2},k+\frac{1}{2}}$.

Step ii: For $\nu = 0, 1, 2, \ldots$, do the following:

(a). Move grid $\vec{z}^{[\nu]}_{j,k} = \{(x^{[\nu]}_{j,k}, y^{[\nu]}_{j,k})\}$ to $\vec{z}^{[\nu+1]}_{j,k} = \{(x^{[\nu+1]}_{j,k}, y^{[\nu+1]}_{j,k})\}$ by solving $\vec{z}_\tau = (\omega\vec{z}_\xi)_\xi + (\omega\vec{z}_\eta)_\eta$ with the conventional explicit scheme. This step can be also done by solving $(\omega\vec{z}_\xi)_\xi + (\omega\vec{z}_\eta)_\eta = 0$ with the following Gauss-Seidel iteration:

$$\begin{aligned}&\alpha_{j+\frac{1}{2},k}\big(\vec{z}^{[\nu]}_{j+1,k} - \vec{z}^{[\nu+1]}_{j,k}\big) - \alpha_{j-\frac{1}{2},k}\big(\vec{z}^{[\nu+1]}_{j,k} - \vec{z}^{[\nu+1]}_{j-1,k}\big) \\ &\quad + \beta_{j,k+\frac{1}{2}}\big(\vec{z}^{[\nu]}_{j,k+1} - \vec{z}^{[\nu+1]}_{j,k}\big) - \beta_{j,k-\frac{1}{2}}\big(\vec{z}^{[\nu+1]}_{j,k} - \vec{z}^{[\nu+1]}_{j,k-1}\big) = 0,\end{aligned}$$

for $1 \le j \le J_x$ and $1 \le k \le J_y$, where

$$\begin{aligned}\alpha_{j\pm\frac{1}{2},k} &= \omega\big(u^{[\nu]}_{j\pm\frac{1}{2},k}\big) = \omega\big(\tfrac{1}{2}(u^{[\nu]}_{j\pm\frac{1}{2},k+\frac{1}{2}} + u^{[\nu]}_{j\pm\frac{1}{2},k-\frac{1}{2}})\big), \\ \beta_{j,k\pm\frac{1}{2}} &= \omega\big(u^{[\nu]}_{j,k\pm\frac{1}{2}}\big) = \omega\big(\tfrac{1}{2}(u^{[\nu]}_{j+\frac{1}{2},k\pm\frac{1}{2}} + u^{[\nu]}_{j-\frac{1}{2},k\pm\frac{1}{2}})\big).\end{aligned}$$

(b). Compute $\{u^{[\nu+1]}_{j+\frac{1}{2},k+\frac{1}{2}}\}$ on the new grid using the conservative-interpolation (4.8). The approximations for c^x, c^y etc are direct extensions of those defined for the 1D case.

(c). Repeat the updating procedure (a) and (b) for a fixed number of iterations (say, 3 or 5) or until $\|\vec{z}^{[\nu+1]} - \vec{z}^{[\nu]}\| \le \epsilon$.

Step iii: Evolve the underlying PDEs using 2D high-resolution finite volume methods on the mesh $\{(x_{j,k}^{[\nu+1]}, y_{j,k}^{[\nu+1]})\}$ to obtain the numerical approximations $u_{j+\frac{1}{2},k+\frac{1}{2}}^{n+1}$ at the time level t_{n+1}.

Step iv: If $t_{n+1} \le T$, then let $u_{j+\frac{1}{2},k+\frac{1}{2}}^{[0]} := u_{j+\frac{1}{2},k+\frac{1}{2}}^{n+1}$ and $(x_{j,k}^{[0]}, y_{j,k}^{[0]}) := (x_{j,k}^{[\nu+1]}, y_{j,k}^{[\nu+1]})$, and go to Step ii.

The above numerical procedures have now been extended to solve several classes of problems such as the nonlinear Hamilton-Jacobi equations [96], convection-dominated problems [107,108], phase-field equations [92]. Recently, Zegeling solved the resistive MHD models [105] and van Dam simulated the traffic flow models [98] using a similar approach as described in this section.

5. Application I: incompressible flow simulations

5.1. Navier-Stokes equations. There have been some efforts in solving the incompressible Navier-Stokes (NS) problems using the moving grid methods, see, e.g., [22,86], but most of these works solve the NS equations in the streamfunction-vorticity formulation. In this subsection, we describe the work of [32] which presents the first effort in designing moving mesh algorithm to solve the incompressible NS equations in the primitive variables formulation. The proposed numerical scheme extends the framework presented in Section 3. The main difficulty in developing this moving mesh scheme is how to keep the divergence-free property for the velocity field at each time level. By some careful analysis, we conclude that this can be done by solving a linearized inviscid Navier-Stokes-type equations.

We consider a two-dimensional incompressible Navier-Stokes equations in primitive variables formulation

$$\begin{cases} \partial_t \mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} &= -\nabla p + \nu\Delta\mathbf{u}, \quad \text{in } \Omega \\ \nabla\cdot\mathbf{u} &= 0, \qquad\qquad\qquad \text{in } \Omega \end{cases} \tag{5.1}$$

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

$$\mathbf{u}(x,0;t) = \mathbf{u}(x,1;t), \qquad \mathbf{u}(0,y;t) = \mathbf{u}(1,y;t), \tag{5.2a}$$

$$\partial_{\mathbf{n}}\mathbf{u}(x,0;t) = \partial_{\mathbf{n}}\mathbf{u}(x,1;t), \qquad \partial_{\mathbf{n}}\mathbf{u}(0,y;t) = \partial_{\mathbf{n}}\mathbf{u}(1,y;t), \tag{5.2b}$$

$$p(x,0;t) = p(x,1;t), \qquad p(0,y;t) = p(1,y;t). \tag{5.2c}$$

Denote

$$\mathbf{V} = H^1(\Omega)^2 \cap \{\mathbf{v} \mid \mathbf{v} \text{ satisfies} (5.2a) - (5.2b)\}$$
$$P = L_0^2(\Omega) \cap \{q \mid q \text{ satisfies} (5.2\text{c})\}.$$

The classical variational formulation for the NS equations (5.1) reads: Find a pair $(\mathbf{u}, p)$ in $\mathbf{V}\times P$ such that

$$\begin{cases} (\partial_t\mathbf{u}, \mathbf{v}) + (\mathbf{u}\cdot\nabla\mathbf{u}, \mathbf{v}) &= (p, \nabla\cdot\mathbf{v}) - \nu(\nabla\mathbf{u}, \nabla\mathbf{v}) \quad \forall\mathbf{v}\in\mathbf{V}, \\ (q, \nabla\cdot\mathbf{u}) &= 0 \qquad\qquad\qquad\qquad\qquad \forall q\in P. \end{cases} \tag{5.3}$$

Assume the domain Ω is triangulated into a triangle mesh $\mathcal{T}_h$, and the elements of the triangulation are denoted as κ. 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 (5.3) can be approximated as: Find a pair $(\mathbf{u}_h, p_h) \in \mathbf{V}_h \times P_h$ such that

$$\left\{ \begin{array}{rcll} (\partial_t \mathbf{u}_h, \mathbf{v}_h) + (\mathbf{u}_h \cdot \nabla \mathbf{u}_h, \mathbf{v}_h) & = & (p_h, \nabla \mathbf{v}_h) - \nu(\nabla \mathbf{u}_h, \nabla \mathbf{v}_h) & \forall \mathbf{v}_h \in \mathbf{V}_h\,, \\ (q_h, \nabla \cdot \mathbf{u}_h) & = & 0 & \forall q_h \in P_h\,. \end{array} \right. \tag{5.4}$$

Denote the subspace of $H^1(\Omega)$ satisfying the periodic boundary condition as V, 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 $\mathcal{T}_h$. In the temporal direction, a 3-step Runge-Kutta scheme is used to solve (5.4).

5.1.1. *Divergence-free interpolation.* It is demonstrated in [32] that a divergence-free-preserving interpolation scheme can be 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 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 point-wise 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

$$x(\tau) = x^{\text{old}} + \tau(x^{\text{new}} - x^{\text{old}}), \qquad \tau \in [0,1] \tag{5.5}$$

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

$$x_i(\tau) = X_i + \tau(X_i^* - X_i), \qquad \tau \in [0,1]. \tag{5.6}$$

With the linear transformation (5.5), the corresponding basis function and the point-wise value can be defined by $\phi^i(\tau) := \phi^i(x(\tau))$ and $u_i(\tau) := u_i(x(\tau))$.

We 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

$$\partial_\tau u_h = 0. \tag{5.7}$$

By direct computation we obtain

$$\frac{\partial \phi^i}{\partial \tau} = -\nabla_{\vec{x}} \phi^i \cdot \delta\vec{x}, \tag{5.8}$$

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

$$(\partial_\tau u_h - \nabla_{\vec{x}} u_h \cdot \delta\vec{x},\, \psi) = 0. \tag{5.9}$$

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

$$\mathbf{W}_h = \mathbf{V}_h \cap \{\mathbf{u}_h \,|\, \nabla \cdot \mathbf{u}_h = 0\}. \tag{5.10}$$

Then the problem (5.9) becomes the following: Find $\mathbf{w}_h \in \mathbf{W}_h$ such that

$$(\partial_\tau \mathbf{w}_h - \nabla_{\vec{x}} \mathbf{w}_h \cdot \delta\vec{x},\, \mathbf{z}_h) = 0, \qquad \forall \mathbf{z}_h \in \mathbf{W}_h. \tag{5.11}$$

The above result implies that

$$\partial_\tau \mathbf{w}_h - \nabla_{\vec{x}} \mathbf{w}_h \cdot \delta\vec{x} \in {\mathbf{W}_h}^{\perp} \tag{5.12}$$

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 [42] that if the solution domain Ω is simply-connected then

$${\mathbf{W}_h}^{\perp} = \{\nabla q \,|\, q \in H^1(\Omega)\}. \tag{5.13}$$

Using the above two results, we can show that solving (5.11) is equivalent to finding $(\mathbf{u}_h, p_h) \in \mathbf{V}_h \times P_h$ such that

$$(\partial_\tau \mathbf{u}_h - \nabla_{\vec{x}} \mathbf{u}_h \cdot \delta\vec{x}, \mathbf{v}_h) = (p_h, \nabla \mathbf{v}_h), \qquad \forall \mathbf{v}_h \in \mathbf{V}_h \tag{5.14a}$$

$$(\nabla_{\vec{x}} \cdot \mathbf{u},\, q_h) = 0, \qquad \forall q_h \in P_h. \tag{5.14b}$$

A 3-step Runge-Kutta scheme is applied to the temporal discretization of (5.14), see [32]. 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 re-distribution. The reason allowing the large time step is due to the fact that the convection speed in (5.14), namely $\delta\vec{x}$, is very small. The speed for most of nodes is as small as $\mathcal{O}(h)$.

5.1.2. *Monitor functions.* There are several possible choices of the monitor function for the incompressible Navier-Stokes approximations, including those based on vorticity (see, e.g., [22]) and density gradient (see, e.g., [25]).

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. [80]:

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

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

$$G_2(v_h) = \sqrt{1 + \alpha\eta^2(v_h)}. \tag{5.15}$$

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 introduce a scaling and a larger power $\beta > 2$ in the monitor:

$$G_3(v_h) = \sqrt{1 + \alpha\Big[\eta(v_h)/\max\eta(v_h)\Big]^{\beta}}. \tag{5.16}$$

The above monitor has been found appropriate in our numerical experiments.

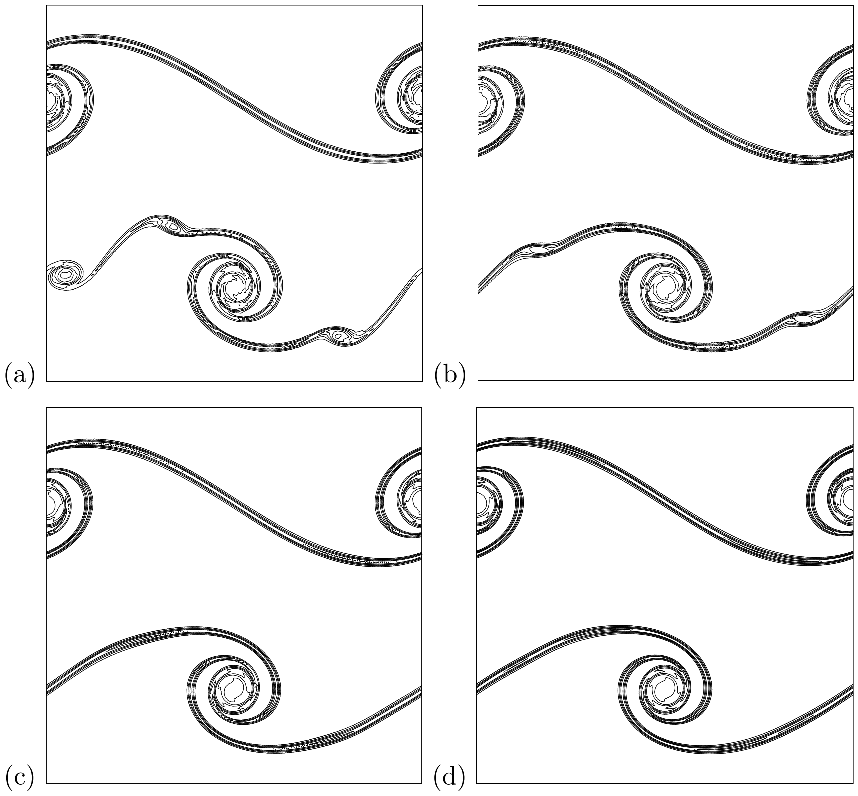


FIGURE 4. Example 5.1: vorticity contours at $t = 0.8$ obtained by using moving mesh methods with resolution (b): 80×80, (c): 100×100, and (d): 160×160. A 160×160 static-mesh solution is included in (a).

5.1.3. *Numerical experiments.*

EXAMPLE 5.1. *Consider a double shear layer governed by the Navier-Stokes equations* (5.1)*, in a unit 1-periodic domain, subject to the initial conditions:*

$$
\begin{aligned}
u_0(x,y) &= \begin{cases} \tanh(\rho(y-0.25)) & \text{for } y \le 0.5; \\ \tanh(\rho(0.75-y)) & \text{for } y > 0.5; \end{cases} \\
v_0(x,y) &= \delta \sin(2\pi x).
\end{aligned}
\tag{5.17}
$$

This problem is a canonical test problem for a scheme's accuracy and resolution in incompressible flows. Brown & Minion [20] performed for this problem a systematic comparison 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 observed in [20] that these underresolved solutions, although convergent when the grid is refined, contain spurious *nonphysical* vortices that are artifacts of the underresolution.

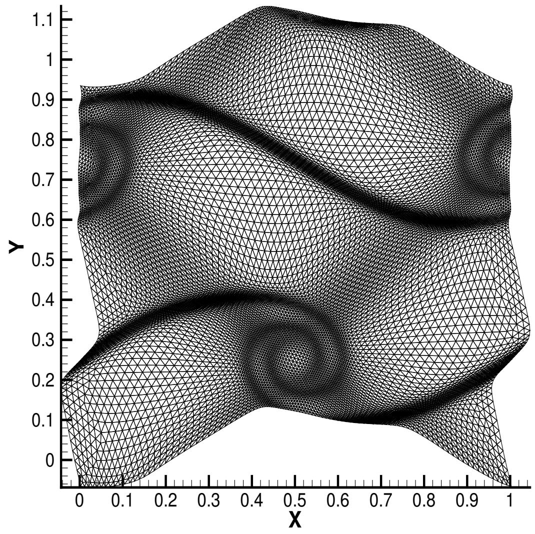


FIGURE 5. Example 5.1. Thin shear layer problem: the mesh at $t = 0.8$ with a 100×100-mesh.

In (5.17), 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 [32], the perturbation size used is $\delta = 0.05$, but the shear layer width is varied to study the effect of the layer resolution on the computations. As in [20], the double layer shear is called a *thick layer problem* when $\rho = 30$ and $\nu = 10^{-4}$ and a *thin layer problem* when $\rho = 100$ and $\nu = 0.5 * 10^{-4}$. It was demonstrated in [20] that the thin layer problem is a challenging one: If the mesh is not fine enough then spurious vortices will be generated in the numerical solutions. It was concluded in [20] that with the second-order Godunov-projection methods about 512×512 cells have to be used in order to obtain reasonable resolutions.

In the thin-layer calculations, we tested the monitor functions (5.15) and (5.16). Our goal is to use about 150×150 cells to resolve the thin-layer problem. However, it is found that the monitor (5.15) does not perform well to achieve this goal, while the monitor (5.16) works well. Three mesh resolutions are used: 80×80, 100×100 and the 160×160. The parameters (α, β) in (5.16) used for the three meshes are (5,2), (5,3), and (8,4), respectively.

Figure 4 shows 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 significant from the finest-mesh reference solutions given in [20]. 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 a single roll-up, as shown in Figures 4 (c) and (d). The adaptive grid with a 100×100-mesh is shown in Figure 5. 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.

5.2. Incompressible two-phase flows with moving mesh. In [31], a coupled moving mesh and level set method for computing incompressible two-phase flow with surface tension is presented. The level set approach of [76] and the moving mesh method for incompressible flow simulations of [32] are combined to compute incompressible two-phase flow with surface tension. The flow we consider has discontinuous density and viscosity, and is characterized by large density and viscosity ratios at the free surface, e.g. air and water. Our goal is to achieve higher resolution of the free surface with a minimum of additional expense.

As an example, we compute the interaction of two fluid bubbles of the same density under the influence of gravity. The fluid is set at rest initially. The viscosity for the fluid inside and out the two bubbles is equal to $\mu = 0.00025$ and 0.0005, respectively. The surface tension is set to zero. The initial positions of the two bubbles correspond to two circles, with the lower one centered at $(0.5, 0.35)$ with radius 0.1 and the upper one centered at $(0.5, 0.65)$ with radius 0.15. We take the density inside the two bubbles to be 1 and 10, respectively.

In Fig. 6, we plot the numerical solutions together with the corresponding meshes at $t = 0.3$, 0.4 and 0.5, obtained by using a 80^2 grid. The desired effect of the mesh adaptivity can be clearly seen in this figure. In [26], the level set method together with a second-order projection scheme was used to study the merging of two bubbles with the above parameters. The overall agreement between our coarse mesh results and the fine mesh results of [26] is found very satisfactory.

6. Application II: hyperbolic conservation laws

There have existed several moving mesh methods for solving hyperbolic conservation laws. The earliest work in this direction may be due to Yanenko et al. [102]. Harten & Hyman [44] proposed to move the grid at an adaptive speed in each time step to improve the resolution of shocks and contact discontinuities. After their works, many other moving mesh methods for hyperbolic problems have been proposed in the literature, including Li & Petzold [63], Azarenok et al. [6, 7], Stockie et al. [91], Lipnikov & Shashkov [67], Liu et al. [68] (for steady state Euler flow calculations), and Tang & Tang [93–95]. Other moving mesh applications include Pen [77] who solve cosmological astrophysical fluid problems.

6.1. Finite-volume approach.

Example 6.1. **The double-Mach reflection problem.** This problem was studied extensively in Woodward & Colella [101] and later by many others. We use exactly the same setup as in [101], i.e. the same initial and boundary conditions and same solution domain $\Omega_p = [0, 4] \times [0, 1]$.

The above problem was computed using the moving mesh scheme described in Section 4, see also [94]. As in [101], only the results in $[0, 3] \times [0, 1]$ are displayed. In Fig. 7, the adaptive meshes with $(J_x, J_y) = (160, 40)$ and $(320, 80)$ are displayed, while the corresponding contours of density are displayed in Fig. 8. By comparing the density plots, it is found the adaptive computation results with

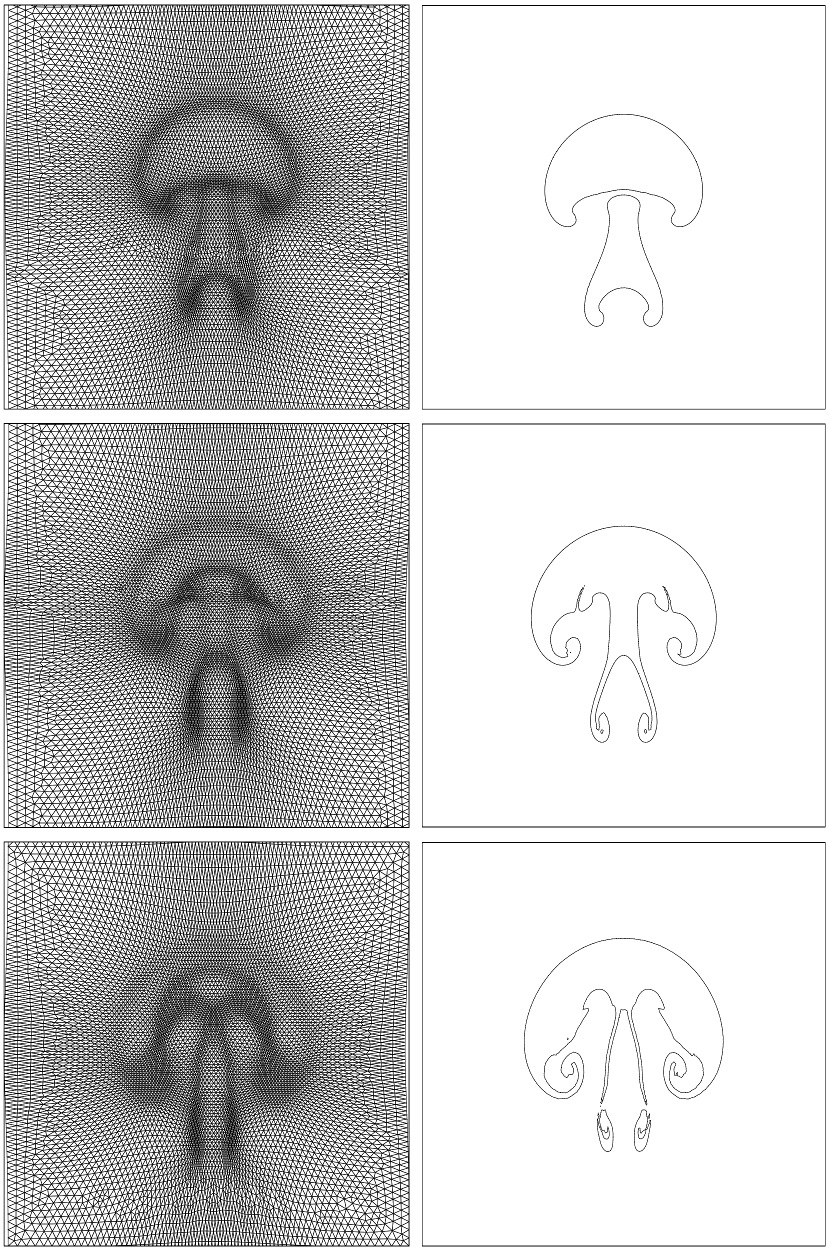


FIGURE 6. Merging of two bubbles in moving 80×80 mesh: meshes and solutions at $t = 0.3, 0.4$ and 0.5 (from top to bottom).

$(J_x, J_y) = (320, 80)$ have similar resolution to the results obtained by the second-order discontinuous Galerkin method with $(J_x, J_y) = (960, 240)$ (p.214, [29]) and by the second-order central scheme with $(J_x, J_y) = (960, 240)$ (p.67, [29]).

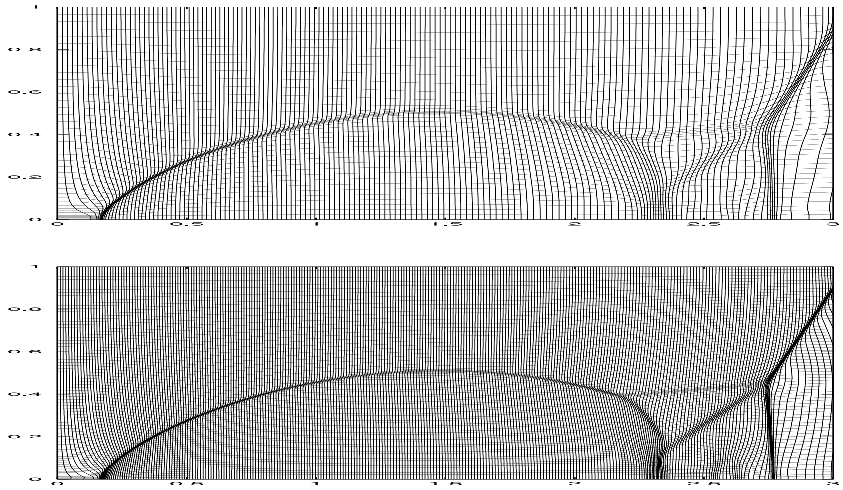


FIGURE 7. Example 6.1: 2D double Mach reflection at $t = 0.2$: the contours of meshes. From top to bottom: $(J_x, J_y) = (160, 40)$ and $(320, 80)$.

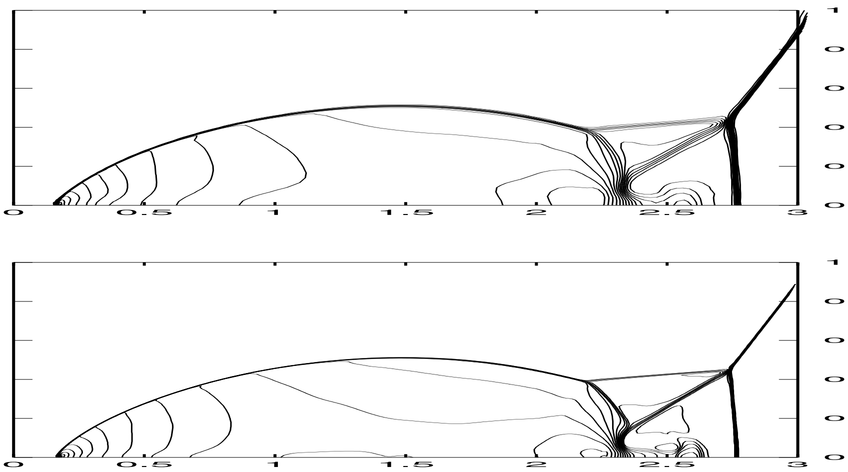


FIGURE 8. Same as Fig. 7, except for density contours.

EXAMPLE 6.2. Consider a spherical explosion between two parallel walls at z=0 and z=1. Initially the gas is at rest with parameters $(p, \rho)_{out}$=$(1, 1)$ everywhere except in a sphere centered at $(0, 0, 0.4)$ with radius 0.2. Inside the sphere $(p, \rho)_{in}$=$(5, 1)$, γ=1.4. This problem was proposed by LeVeque [56].

The above problem was computed using the moving mesh scheme proposed in [8]. The moving mesh method uses a second-order Godunov-type solver to solve the hyperbolic system. The main strategies of this approach are two folds. First, a second-order finite-volume flow solver is employed to update the flow parameters at the new time level directly on the adaptive grid without using interpolation. This PDE solver involves two level grids at t^n and t^{n+1}, so after mesh redistribution certain number of iterations is needed. The conservation equations are formulated and approximated on a time-dependent two-level grids. This is in contrast to the approach used in [61, 94] where the PDE solver involves one level grid but interpolation has to be used (see Section 4). Secondly, some *optimization problem* is solved in order to obtain the mesh, similar to that described in Section 3. The idea of this approach is to minimize some discrete Dirichlet's functional possessing the infinite barrier property, as suggested by Charakhch'yan & Ivanenko [27, 53].

For Example 6.2, the initial jump in pressure results in an outward moving shock, a contact discontinuity and an inward moving rarefaction wave. There occurs interactions among these waves and between the waves and the walls. The flow by t=0.7 consists of several shocks and strong contact/(tangential) discontinuity surrounding the low density region near the center. In Fig. 9 the adapted mesh with a 100×140 grid is shown. To compare with the fixed mesh solution, the pressure contours, computed on the 200×250 uniform and adapted meshes, are presented in Fig. 10. We see that the shock thickness obtained by using the moving mesh method is decreased significantly in comparison with the fixed mesh solution. More computational results and computational details can be found in [8].

6.2. Reactive flow calculations on moving meshes. Numerical simulation of reactive flow has been an important and active research direction. Numerical efforts using adaptive grid methods can be found in Oran & Boris' books [74, 75], see also the work of Fortov et al. [39] and Geßner & Kröner [41].

In [9], the method of calculating the system of gas dynamics equations coupled with the chemical reaction equation is considered. The flow parameters are updated in whole without splitting the system into a hydrodynamical part and an ODE part. The algorithm is based on the Godunov's scheme on deformed meshes with some modification to increase the scheme-order in time and space. To generate the moving adaptive mesh the variational approach is applied. At each time step the functional of smoothness, written on the graph of some control functions, is minimized. The grid-lines are condensed in the vicinity of the solution singularity and, thus, an adaptive mesh is generated. Strong grid-lines condensing allows placing several cells into the burning zone and, thus, resolving a fine structure of the reaction front, e.g. appearance of vortices.

The governing system of the differential equations relating to 2D reactive gas flow is

$$\frac{\partial \boldsymbol{\sigma}}{\partial t} + \frac{\partial \boldsymbol{a}}{\partial x} + \frac{\partial \boldsymbol{b}}{\partial y} = \boldsymbol{c} \,, \tag{6.1}$$

where

$$\boldsymbol{\sigma} = (\rho, \rho u, \rho v, E, \rho Z)^\top, \quad \boldsymbol{a} = (\rho u, \rho u^2{+}p, \rho uv, u(E{+}p), \rho uZ)^\top,$$
$$\boldsymbol{b} = (\rho v, \rho uv, \rho v^2{+}p, v(E{+}p), \rho vZ)^\top, \quad \boldsymbol{c} = (0, 0, 0, 0, -\rho K(T)Z)^\top,$$

with u and v the velocity components, $E = \rho[e + 0.5(u^2 + v^2)] + q_o\rho Z$ the total energy.

Solving the 1-D or 2-D gas dynamics equations with grid adaptation at each time step contains the following stages:

ALGORITHM 4: Coupled algorithm for reactive flow

(i) Generate the mesh at the next time level t^{n+1}.
(ii) Compute the gas dynamics values at time t^{n+1}.

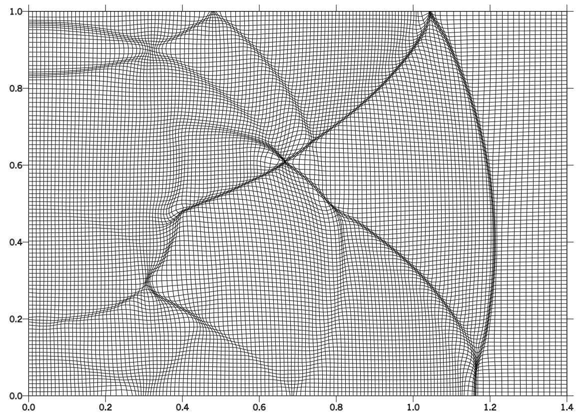


FIGURE 9. Example 6.2: adapted mesh 100×140.

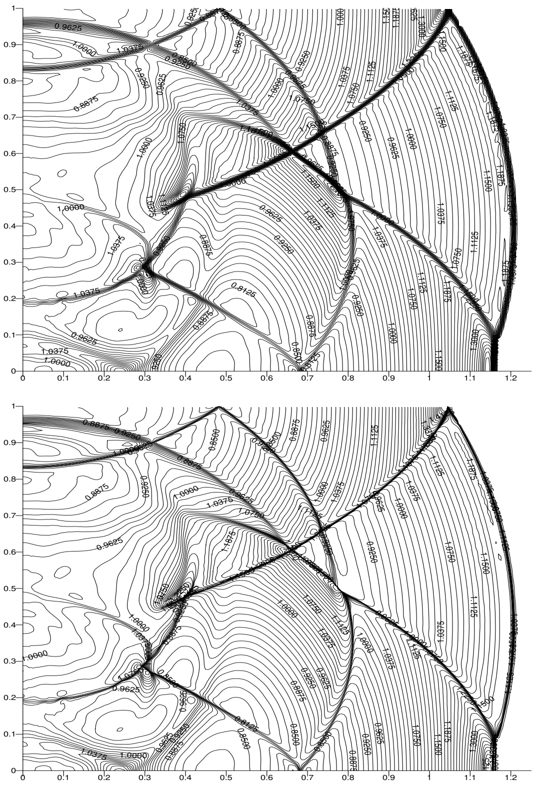
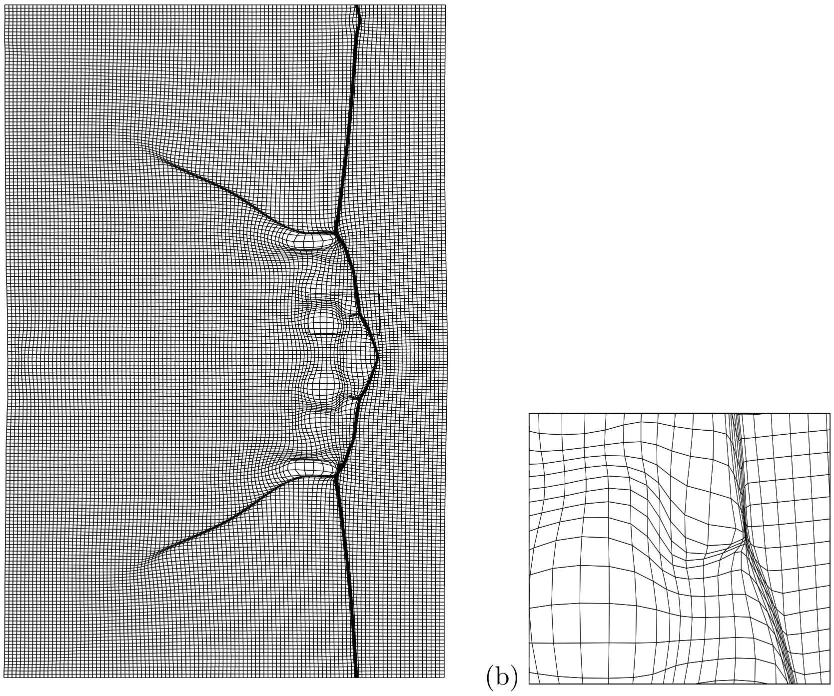


FIGURE 10. Example 6.2: pressure contours computed with a 200×250 uniform (top) and moving (bottom) meshes.

(iii) Make one iteration to compute the new grid coordinates $(x, y)_i$ at t^{n+1}.
(iv) Repeat steps (ii) and (iii) using a given number of iterations.
(v) Compute the final gas dynamics values at t^{n+1}.

It is pointed out that the above procedure does not require interpolation from the old mesh solution to the new mesh solution. The reason is that the numerical



(a) (b)

FIGURE 11. Reactive flow computations: adapted mesh (a) and close-up (b) at t=61.59.

scheme for obtaining the solution of 6.1 uses the meshes on levels t^n and t^{n+1}s by utilizing the integral conservation laws in $\mathbf{R}^3$ space (x, y, t), see, e.g., [7–9].

We now present the results of modeling the 2D unsteady detonation, when the cellular structure arises, as considered in [18]. In Fig. 11 the adapted mesh at t=61.59 is presented. We draw only 120 front cells in the x-direction. We observe that two main triple points are resolved by grid lines very accurately, see the fragment of the mesh in Fig. 11b. The burning zone takes 7−8 cells, conversely the shock wave, emanating from the triple points, takes only 3 cells. The thickness of the detonation wave on the adapted mesh is significantly smaller than that on the uniform mesh. The mesh also "feels" the triple points, arising due to instability in the solution. In Fig. 12 the pressure contours at t=61.59 are plotted, obtained on the uniform and adapted meshes. It is observed in the vicinity of the triple points the resolution is much higher with the adaptive grid, the difference can be better seen by comparing Fig. 12c and Fig. 12d. The details of the above computations can be found in [9].

6.3. Moving mesh discontinuous Galerkin method. In [60], a moving mesh discontinuous Galerkin (DG) method is developed for solving the nonlinear conservation laws. In the mesh adaptation part, two issues have received much attention. One is about the construction of the monitor function which is used to guide the mesh redistribution. In [60], a heuristic posteriori error estimator is used in constructing the monitor function. The second issue is concerned with the solution interpolation which is used to interpolates the numerical solution from the

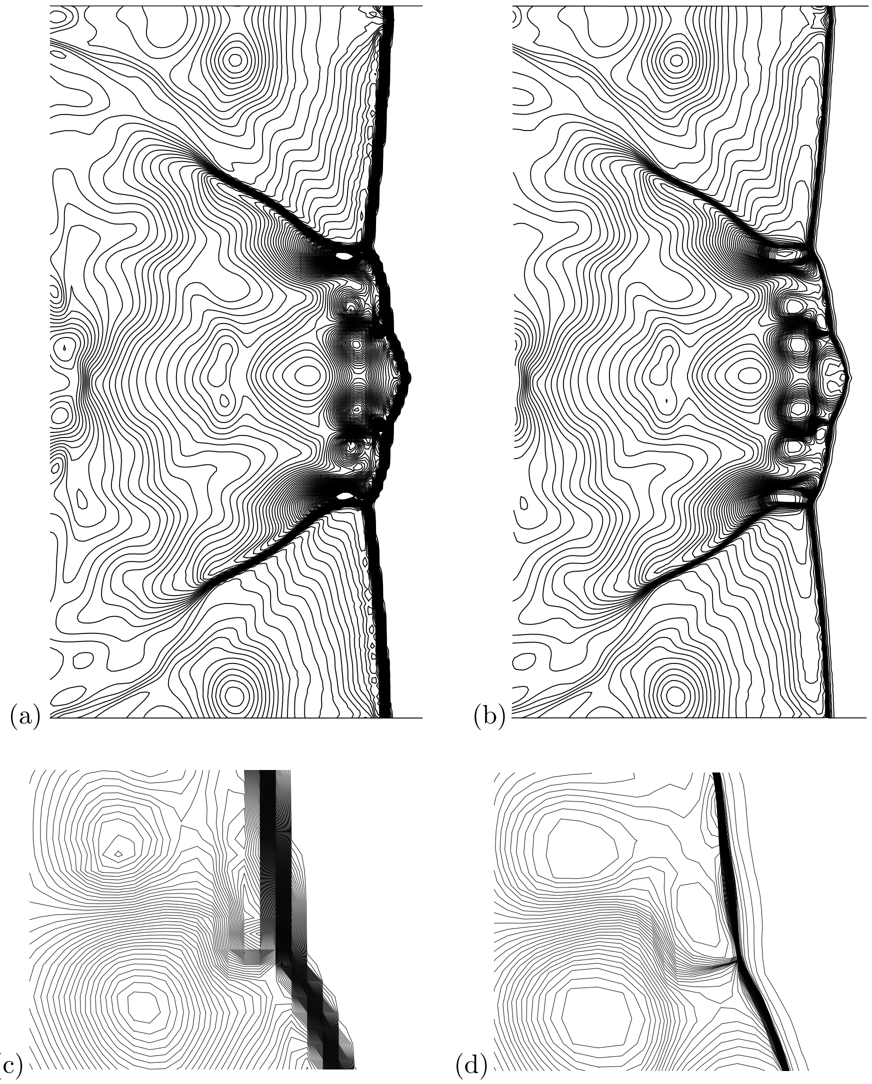


FIGURE 12. Reactive flow computations: pressure contours at t=61.59 computed on the uniform (a) and adapted (b) mesh. Close-up for the uniform (c) and adapted (d) mesh (domain, depicted in Fig 11b).

old mesh to the updated mesh. This is done by using a scheme that mimics the DG method for linear conservation laws. Appropriate limiters are used on seriously distorted meshes generated by the moving mesh approach to suppress the numerical oscillations.

As a numerical example, the double Mach problem Example 6.1 is re-computed using the moving mesh DG methods. We calculate this problem on 96×24 and 192×48 meshes to time $t = 0.2$. The mesh and density contours using the two meshes are plotted in Figs. 13 and 14, respectively. The density contour is plotted

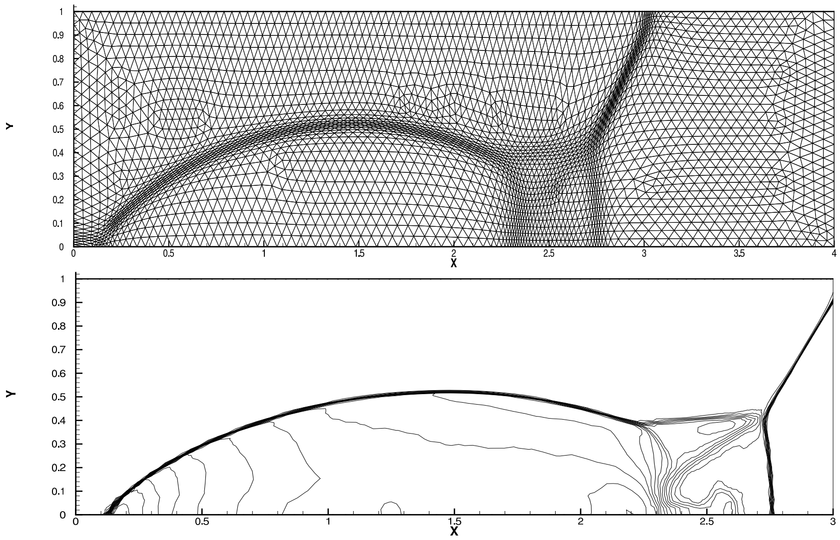


FIGURE 13. Double Mach reflection problem using moving mesh DG method: Mesh and density contour obtained on a 96×24 grid.

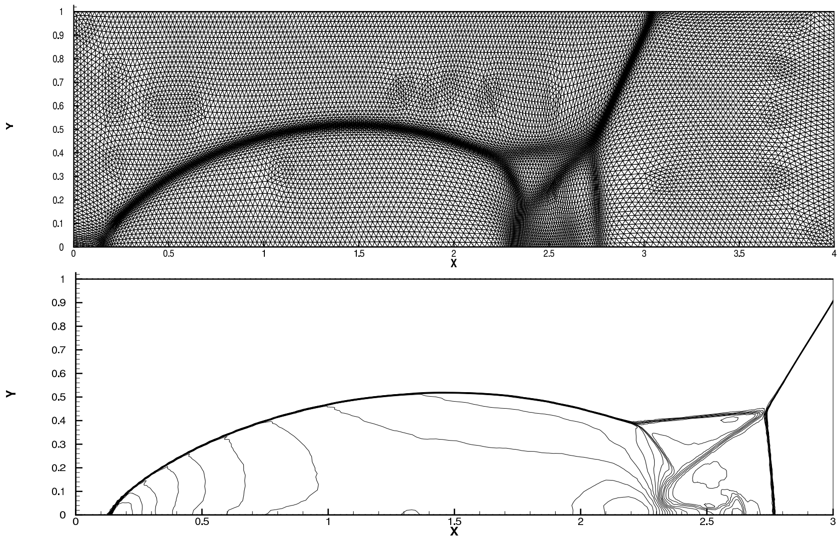


FIGURE 14. Same as Fig. 13, except on a 192×48 grid.

on $[0, 3] \times [0, 1]$. It can be seen that the mesh grid points are clustered in the regions where the shocks and the detailed structures are located.

7. Concluding remarks

In this article we have reviewed some recent efforts in computing CFD problems with moving mesh methods. Due to limitation of space, it is impossible to give a comprehensive review on this rapidly developing subject. In particular, we are unable to include many important applications of the moving mesh methods, such as computations for Schrödinger equations [24, 81, 100], for free-surface or moving boundary simulations [78, 90], for traffic flow simulations [98], and for phase-field equations [14, 70].

Moving grid methods have been generally less developed and less well-known than the other adaptive techniques. One possible reason for this is the lack of firmer theoretical foundations. In the past decade, there have been some efforts in understanding the theoretical backgrounds of the moving mesh methods, see, e.g., [13, 35, 66, 69, 79, 88, 89]. In particular, a clean uniform convergence rate is established by Kopteva et al. [54, 55] for two-point value ODEs using moving mesh methods with the arclength monitor. However, the extension to time-dependent problems does not seem simple.

Most recently, it is realized that the meshes may be designed by analyzing the interpolation error, see, e.g., [28, 48, 52]. The three key grid features or qualities which play a role in determining the interpolation error (e.g., in a finite element analysis setting) are (1) geometric quality in physical space, (2) alignment quality in physical space or isotropy in computational space, and (3) adaptive quality or level of equidistribution. From the way that terms representing these three features arise in the interpolation error bound, it was shown that the latter two are the most important [48, 52].

The following website contains some materials related to this work:

http://www.math.hkbu.edu.hk/∼ttang/MMmovie

We close this paper by quoting the following words from Oran & Boris (*Numerical Simulation of Reactive Flow*, 2nd ed., Cambridge University Press, 2001; p.183): "Adaptive griding techniques fall into two broad classes, *adaptive mesh redistribution* and *adaptive mesh refinement*, both contained in the acronym AMR. Techniques for adaptive mesh redistribution continuously reposition a fixed number of cells, and so they improve the resolution in particular locations of the computational domain. Techniques for adaptive mesh refinement add new cells as required and delete other cells that are no longer required. Because the great potential of AMR for reducing computational costs without reducing the overall level of accuracy, it is a forefront area in scientific computation".

Acknowledgments. I wish to thank my collaborators Dr. R. Li, Prof. H.-Z. Tang and Prof. Pingwen Zhang of Peking University, Dr. B. Azarenok of Russian Academy of Sciences, and Prof. Wenbin Liu of the University of Kent. Special thanks are due to my graduate students Z.-R. Zhang, Yana Di and Z.-J. Tan for their contributions to this project. I would also like to acknowledge valuable comments made by Arthur van Dam, Ruo Li, Paul Zegeling and Zhengru Zhang. The research of the author was partially supported by Hong Kong Research Grants Council (RGC HKBU2044/00P, 2083/01P, 2045/02P), Hong Kong Baptist University (FRG FRG/01-02/II-01, FRG/00-01/II-08, FRG/98-99/II-14), and International Research Team on Complex System, Chinese Academy of Sciences.

References

1. S. Adjerid and J. E. Flaherty, *A moving FEM with error estimation and refinenment for 1-D time dependent PDEs*, SIAM J. Numer. Anal. **23** (1986), 778–795.
2. ______, *A moving-mesh FEM with local refinement for parabolic PDEs*, Comput. Meth. Appl. Mech. Engrg. **55** (1986), 3–26.
3. D. A. Anderson, *Adaptive mesh schemes based on grid speeds*, AIAA Paper **83-1931** (1983), 311.
4. ______, *Equidistribution schemes, poisson generators, and adaptive grids*, Appl. Math. Comput. **24** (1987), 211–227.
5. D.C. Arney and J.E. Flaherty, *A two-dimensional mesh moving technique for time dependent partial differential equations*, J. Comput. Phys. **67** (1986), 124–144.
6. B.N. Azarenok, *Realization of a second-order Godunov's scheme*, Comput. Meth. in Appl. Mech. and Engin. **189** (2000), 1031–1052.
7. ______, *Variational barrier method of adaptive grid generation in hyperbolic problems of gas dynamics*, SIAM J. Numer. Anal. **40** (2002), 651–682.
8. 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.
9. B.N. Azarenok and T. Tang, *Reactive flow calculations on moving meshes*, 2005, To appear in J. Comput. Phys.
10. M.J. Baines, *Moving Finite Elements*, Oxford University Press, 1994.
11. ______, *Grid adaptation via node movement*, Appl. Numer. Math. **26** (1998), 77–96.
12. M.J. Baines, M.E. Hubbard, and P.K. Jimack, *A Lagrangian moving finite element method incorporating monitor functions*, Advances in Scientific Computing and Applications (Beijing/New York) (Y. Lu, W. Sun, and T. Tang, eds.), Science Press, 2004, pp. 32–44.
13. G. Beckett and J.A. Mackenzie, *Convergence analysis of finite-difference approximations on equidistributed grids to a singularly perturbed boundary value problem*, Appl. Numer. Math. **35** (2000), 209–131.
14. G. Beckett, J.A. Mackenzie, and M.L. Robertson, *A moving mesh finite element method for the solution of two-dimensional stephan problems*, J. Comput. Phys. **168** (2001), 500–518.
15. G. Beckett, J.A. Mackenzie, M.L. Robertson, and D.M. Sloan, *On the numerical solutions of one-dimensional PDEs using adaptive methods based on equidistribution*, J. Comput. Phys. **167** (2001), 372–392.
16. ______, *Computational solution of two-dimensional PDEs using adaptive methods based on equidistribution*, J. Comput. Phys. **182** (2002), 478–495.
17. K.W. Blake, *Moving mesh methods for nonlinear parabolic partial differential equations*, Ph.D. thesis, University of Reading, U.K., 2001.
18. A. Bourlioux and A. Majda, *Theoretical and numerical structure for unstable two-dimensional detonations*, Combustion and Flame **90** (1992), 211–229.
19. J.U. Brackbill and J.S. Saltzman, *Adaptive zoning for singular problems in two dimensions*, J. Comput. Phys. **46** (1982), 342–368.
20. D.L. Brown and M.L. Minion, *Performance of under-resolved two-dimensional incompressible flow simulations*, J. Comput. Phys. **122** (1995), 165–183.
21. C. Budd, W.-Z. Huang, and R.D. Russell, *Moving mesh methods for problems with blow-up*, SIAM J. Sci. Comput. **17** (1996), 305–327.
22. 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.
23. ______, *A study of monitor functions for two dimensional adaptive mesh generation*, SIAM J. Sci. Comput. **20** (1999), 1978–1994.
24. H.D. Ceniceros, *A semi-implicit moving mesh method for the focusing nonlinear Schrodinger equation*, Commun. Pure Appl. Anal. **1** (2002), 1–18.
25. H.D. Ceniceros and T.Y. Hou, *An efficient dynamically adaptive mesh for potentially singular solutions*, J. Comput. Phys. **172** (2001), 609–639.
26. Y.C. Chang, T.Y. Hou, B. Merriman, and S. Osher, *A level set formulation of Eulerian interface capturing methods for incompressible fluid flows*, J. Comput. Phys. **124** (1996), 449–464.
27. A.A. Charakhch'yan and S.A. Ivanenko, *Curvilinear grids of convex quadrilaterals*, USSR Comput. Math. Math. Phys. **2** (1988), 126–133.

28. L. Chen, P. Sun, and J. Xu, *Optimal anisotropic simplicial meshes for minimizing interpolation errors in l^p-norm*, 2004, To appear in Math. Comp.
29. B. Cockburn, C. Johnson, C.-W. Shu, and E. Tadmor, *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, vol. 1697, Springer, Berlin, New York, 1997, Lecture Notes in Math.
30. C. de Boor, *Good approximation by splines with variable knots II*, Springer Lecture Notes Series 363, Springer Verlag, Berlin, 1973.
31. Y. Di, R. Li, T. Tang, and P.W. Zhang, *Level set calculations of interface on a dynamically adaptive grid*, 2005, To be submitted.
32. ———, *Moving mesh finite element methods for the incompressible Navier-Stokes equations*, 2005, To appear in SIAM J. Sci. Comput.
33. A. Doelman, T.J. Kaper, and P.A. Zegeling, *Pattern formation in the 1-d gray-scott model*, Nonlinearity **10** (1997), 523–563.
34. E. A. Dorfi and L. O'c. Drury, *Simple adaptive grids for 1-d initial value problems*, J. Comput. Phys. **69** (1987), 175–195.
35. T. Dupont and Y. Liu, *Symmetric error estimates for moving mesh Galerkin methods for advection-diffusion equations*, SIAM J. Numer. Anal. **40** (2002), 914–927.
36. A.S. Dvinsky, *Adaptive grid generation from harmonic maps on Riemannian manifolds*, J. Comput. Phys. **95** (1991), 450–476.
37. M.G. Edwards, J.T. Oden, and L. Demkowicz, *An hr adaptive approximate Riemann solver for the Euler equations in two dimensions*, SIAM J. Sci. Comput. **14** (1993), 185–217.
38. J. Eell and J.H. Sampson, *Harmonic mapping of Riemannian manifolds*, Amer. J. Math. **86** (1964), 109–160.
39. V.E. Fortov, B. Goel, C.D. Munz, A.L. Ni, A.V. Shutov, and O.Yu. Vorobiev, *Numerical simulations of nonstationary fronts and interfaces by the godunov method in moving grids*, Nuclear Science and Engineering **123** (1996), 169–189.
40. R.M. Furzeland, J.G. Verwer, and P.A. Zegeling, *A numerical study of three moving gird methods for 1-d partial differential equations which are based on the method of lines*, J. Comput. Phys. **89** (1990), 349–399.
41. T. Geßner and D. Kröner, *Dynamic mesh adaptive for supersonic reactive flow*, Hyperbolic Problems: Theory, Numerics, Applications (Basel-Boston-Berlin) (H. Freistühler and G. Warnecke, eds.), Birkhäuser, 2001, pp. 415–424.
42. V. Girault and P.-A. Raviart, *Finite Element Methods for Navier-Stokes Equations, Theory and Algorithms*, Springer-Verlag, 1986.
43. R. Hamilton, *Harmonic Maps of Manifolds with Boundary*, vol. 471, Springer-Verlag, New York, 1975.
44. A. Harten and J.M. Hyman, *Self–adjusting grid methods for one–dimensional hyperbolic conservation laws*, J. Comput. Phys. **50** (1983), 235–269.
45. D.F. Hawken, J.J. Gottlieb, and J.S. Hansen, *Review of some adaptive node-movement techniques in finite-element and finite difference solutions of partial differential equations*, J. Comput. Phys. **95** (1991), 254–302.
46. R.G. Hindman and J. Spencer, *A new approach to truely adaptive grid generation*, AIAA Paper **83-0450** (1983), 1.
47. W.-Z. Huang, *Practical aspects of formulation and solution of moving mesh partial differential equations*, J. Comput. Phys. **171** (2001), 753–775.
48. ———, *Variational mesh adaptation: isotropy and equidistribution*, J. Comput. Phys. **174** (2001), 903–924.
49. W.-Z. Huang, Y. Ren, and R.D. Russell, *Moving mesh methods based on moving mesh partial differential equations*, J. Comput. Phys. **113** (1994), 279–290.
50. ———, *Moving mesh partial differential equations (MMPDEs) based on the equidistribution principle*, SIAM J. Numer. Anal. **31** (1994), 709–730.
51. W.-Z. Huang and R.D. Russell, *Analysis of moving mesh partial differential equations with spatial smoothing*, SIAM J. Numer. Anal. **34** (1997), 1106–1126.
52. W.-Z. Huang and W. Sun, *Variational mesh adaptation ii: Error estimates and monitor functions*, J. Comput. Phys. **184** (2003), 619–648.
53. P. Knupp, L. Margolin, and M. Shashkov, *Reference Jacobian optimization-based rezone strategies for arbitrary Lagrangian Eulerian methods*, J. Comput. Phys. **176** (2002), 93–128.

54. N. Kopteva, *Maximum norm a posteriori error estimates for a one-dimensional convection-diffusion problem*, SIAM J. Numer. Anal. **39** (2001), 423–441.
55. N. Kopteva and M. Stynes, *A robust adaptive method for quasi-linear one-dimensional convection-diffusion problem*, SIAM J. Numer. Anal. **39** (2001), 1446–1467.
56. J.O. Langseth and R.J. LeVeque, *A wave propagation method for 3d hyperbolic conservation laws*, J. Comput. Phys. **165** (2000), 126–166.
57. R. Li, *Moving mesh method and its application*, Ph.D. thesis, Peking University, School of Mathematical Sciences, 2001, (in Chinese).
58. ———, *On multi-mesh h-adaptive methods*, 2004, To appear in J. Sci. Comput.
59. R. Li, W.-B. Liu, H.-P. Ma, and T. Tang, *Adaptive finite element approximation for distributed elliptic optimal control problems*, SIAM J. Control Optim. **41** (2002), 1321–1349.
60. R. Li and T. Tang, *Moving mesh discontinuous Galerkin method for hyperbolic conservation laws*, 2004, Submitted to J. Sci. Comput. (under revision).
61. 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.
62. ———, *A moving mesh finite element algorithm for singular problems in two and three space dimensions*, J. Comput. Phys. **177** (2002), 365–393.
63. S. Li and L. Petzold, *Moving mesh method with upwinding schemes for time-dependent PDEs*, J. Comput. Phys. **131** (1997), 368–377.
64. S. Li, L. Petzold, and Y. Ren, *Stability of moving mesh systems of partial differential equations*, SIAM J. Sci. Comput. **20** (1998), 719–738.
65. K. Liang and P. Lin, *A splitting moving mesh method for 3-d quenching and blow-up problems*, 2005, Preprint.
66. T. Linß, *Uniform pointwise convergence of finite difference schemes using grid equidistribution*, Computing **66** (2001), 27–39.
67. K. Lipnikov and M. Shashkov, *Moving meshes for the Burgers equation*, 2003, Report LA-UR-03-7605, Los Alomas National Laboratory.
68. F. Liu, S. Ji, and G. Liao, *An adaptive grid method and its application to steady Euler flow calculations*, SIAM J. Sci. Comput. **20** (1998), 811–825.
69. Y. Liu, R.E. Bank, T.F. Dupont, S. Garcia, and R.F. Santos, *Symmetric error estimates for moving mesh mixed methods for advection-diffusion equations*, SIAM J. Numer. Anal. **40** (2003), 2270–2291.
70. J.A. Mackenzie and M.L. Robertson, *The numerical solution of one-dimensional phase change problems using an adaptive moving mesh method*, J. Comput. Phys. **161** (2000), 537–557.
71. K. Miller, *Moving finite element methods II*, SIAM J. Numer. Anal. **18** (1981), 1033–1057.
72. K. Miller and R.N. Miller, *Moving finite element methods I*, SIAM J. Numer. Anal. **18** (1981), 1019–1032.
73. T.J. Oden, *Progress in adaptive methods in computational fluid dynamics*, Adaptive Methods for Partial Differential Equations (Philadelphia) (J. E. Flaherty, P. J. Paslow, M. S. Shephard, and J. D. Vasilakis, eds.), 1989, pp. 206–252.
74. E. Oran and J.P. Boris, *Numerical Simulation of Reactive Flow*, Elsvier, New York, 1987.
75. ———, *Numerical simulation of reactive flow*, 2nd ed., Cambridge University Press, 2001.
76. S. Osher and J.A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys. **79** (1988), 12–49.
77. Ue-Li Pen, *A high-resolution adaptive moving mesh hydrodynamic algorithm*, Astrophysical J. (suppl. series) **115** (1998), 19–34.
78. B. Perot and R. Nallapati, *A moving unstructured staggered mesh method for the simulation of incompressible free-surface flows*, J. Comput. Phys. **184** (2003), 192–214.
79. Y. Qiu, D.M. Sloan, and T. Tang, *Numerical solution of a singularly perturbed two-point boundary value problem using equidistribution: analysis of convergence*, J. Comput. Appl. Math. **106** (2000), 121–143.
80. 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), 53–72.
81. 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.

82. Y. Ren, *Theory and computation of moving mesh methods for solving time-dependent partial differential equations*, Ph.D. thesis, Simon Fraser University, Dept. of Mathematics, Burnaby, Canada, 1991.
83. Y. Ren and R.D. Russell, *Moving mesh techniques based upon equidistribution, and their stability*, SIAM J. Sci. Statist. Comput. **13** (1992), 1265–1286.
84. R. Schoen and S.-T. Yau, *On univalent harmonic maps between surfaces*, Invent. Math. **44** (1978), 265–278.
85. B. Semper and G. Liao, *A moving grid finite-element method using grid deformation*, Numer. Meth. PDEs **11** (1995), 603–615.
86. C.Y. Shen and H.L. Reed, *Application of a solution-adaptive method to fluid flow: line and arclength approach*, Computers & Fluids **23** (1994), 373–395.
87. R.D. Skeel and M. Berzins, *A method for the spatial discretization of parabolic differential equations in one space variable*, SIAM J Sci. Statist. Comp. **11** (1990), 1–32.
88. J.H. Smith, *Analysis of moving mesh methods for dissipative partial differential equations*, Ph.D. thesis, Stanford University, Dept. Computer Science, 1996.
89. J.H. Smith and A.M. Stuart, *Analysis of continuous moving mesh equations*, 1996, Technical report, SCCM Program, Stanford University, Stanford, CA.
90. V. Sochnikov and S. Efrima, *Level set calculations of the evolution of boundaries on a dynamically adaptive grid*, Int. J. Numer. Meth. Eng. **56** (2003), 1913–1929.
91. J.M. Stockie, J.A. Mackenzie, and R.D. Russell, *A moving mesh method for one-dimensional hyperbolic conservation laws*, SIAM J. Sci. Comput. **22** (2001), 1791–1813.
92. Z.-J. Tan, T. Tang, and Z.-R. Zhang, *A simple moving mesh method for one- and two-dimensional phase-field equations*, 2005, To appear in J. Comput. Appl. Math.
93. H.Z. Tang, *Solution of the shallow-water equations using an adaptive moving mesh method*, Int. J. Numer. Meth. Fluids **44** (2004), 789–810.
94. 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.
95. ______, *Multi-dimensional moving mesh methods for shock computations*, Recent Advances in Scientific Computing and Partial Differential Equations (Providence, USA) (S.Y. Cheng, C.-W. Shu, and T. Tang, eds.), Contemporary Mathematics, vol. 330, American Mathematical Society, 2003, pp. 169–183.
96. 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), 543–572.
97. Y. Tourigny and F. Hülsemann, *A new moving mesh algorithm for the finite element solution of variational problems*, SIAM J. Numer. Anal. **35** (1998), 1416–1438.
98. A. van Dam, *A moving mesh finite volume solver for macroscopic traffic flow models*, 2002, MSc thesis, Mathematical Institute, Utrecht University, Netherlands.
99. J.G. Verwer, J.G. Blom, R.M. Furzeland, and P.A. Zegeling, *A moving-grid method for one-dimensional PDEs based on the method of lines*, Adaptive Methods for Partial Differential Equations (M. S. Shephard J. E. Flaherty, P.J. Paslow and J. D. Vasilakis, eds.), SIAM, Philadelphia, 1989, pp. 160–185.
100. D. Wang and X.P. Wang, *A three-dimensional adaptive method based on the iterative grid redistribution*, J. Comput. Phys. **199** (2004), 423–436.
101. P. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks*, J. Comput. Phys. **54** (1984), 115–173.
102. N.N. Yanenko, E.A. Kroshko, V.V. Liseikin, V.M. Fomin, V.P. Shapeev, and Y.A. Shitov, *Methods for the construction of moving grids for problems of fluid dynamics with big defromations*, Proceedings of the 5th Inter. Conf. on Numer. Meth. in Fluid Mechanics (A.I. van de Vooren and P.J. Zandbergen, eds.), Springer, 1976, Lecture Notes in Physics, vol. 59., pp. 454–459.
103. P.A. Zegeling, *R-refinement with finite elements or finite differences for evolutionary pde models*, Appl. Numer. Math. **26** (1998), 97–104.
104. ______, *Moving grid techniques*, Handbook of Grid Generation (B. K. Soni J. F. Thompson and N. P. Weatherill, eds.), CRC Press, 1999, pp. 37.1–37.18.
105. ______, *On resistive MHD models with adaptive moving meshes*, 2005, To appear in J. Sci. Comput.

106. P.A. Zegeling and R. Keppens, *Adaptive method of lines for magnetohydrodynamic pde models*, 2001, a chapter in the book Adaptive Method of Lines, CRC Press.
107. Z.-R. Zhang, *Moving mesh methods for convection-dominated equations and nonlinear conservation laws*, Ph.D. thesis, Hong Kong Baptist University, Dept of Mathematics, 2003.
108. Z.-R. Zhang and T. Tang, *An adaptive mesh redistribution algorithm for convection-dominated problems*, Commun. Pure Appl. Anal. **1** (2002), 341–357.

Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong & Institute of Computational Mathematics, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing 100080, China.

E-mail address: `ttang@math.hkbu.edu.hk, ttang@lsec.cc.ac.cn`