Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 225 (2007) 1137–1158
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

An adaptive mesh redistribution method for the incompressible mixture flows using phase-field model

Zhijun Tan [a], K.M. Lim [b], B.C. Khoo [a,b,*]

[a] Singapore-MIT Alliance, 4 Engineering Drive 3, National University of Singapore, Singapore 117576, Singapore
[b] Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore

Received 16 May 2006; received in revised form 12 January 2007; accepted 19 January 2007
Available online 2 February 2007

Abstract

A phase field model which describes the motion of mixtures of two incompressible fluids is presented by Liu and Shen [C. Liu, J. Shen, A phase field model for the mixture of two incompressible fluids and its approximation by a Fourier-spectral method, Phys. D 179 (2003) 211–228]. The model is based on an energetic variational formulation. In this work, we develop an efficient adaptive mesh method for solving a phase field model for the mixture flow of two incompressible fluids. It is a coupled nonlinear system of Navier–Stokes equations and Allen–Cahn phase equation (phase-field equation) through an extra stress term and the transport term. The numerical strategy is based on the approach proposed by Li et al. [R. Li, T. Tang, P.-W. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, J. Comput. Phys. 170 (2001) 562–588] to separate the mesh-moving and PDE evolution. In the PDE evolution part, the phase-field equation is numerically solved by a conservative scheme with a Lagrange multiplier, and the coupled incompressible Navier–Stokes equations are solved by the incremental pressure-correction projection scheme based on the semi-staggered grid method. In the mesh-moving part, the mesh points are iteratively redistributed by solving the Euler–Lagrange equations with a parameter-free monitor function. In each iteration, the pressure and the phase are updated on the resulting new grid by a conservative-interpolation formula, while the velocity is re-mapped in a non-conservative approach. A simple method for preserving divergence-free is obtained by projecting the velocity onto the divergence-free space after generating the new mesh at the last iterative step. Numerical experiments are presented to demonstrate the effectiveness of the proposed method for solving the incompressible mixture flows.

AMS classification: 65M06; 35Q35; 80A22

Keywords: Moving mesh method; Finite volume method; Navier–Stokes equations; Projection method; Phase-field equations; Cahn–Hilliard equation; Allen–Cahn equation

* Corresponding author. Address: Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore. Tel.: +65 68742889; fax: +65 67791459.
E-mail addresses: smatz@nus.edu.sg (Z. Tan), limkm@nus.edu.sg (K.M. Lim), mpekbc@nus.edu.sg (B.C. Khoo).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2007.01.019

1. Introduction

In this paper, we shall discuss the class of adaptive grid methods often called moving mesh methods (or dynamic methods in contrast to the static methods) for solving time-dependent PDEs. These methods involve the solution of the underlying PDE for the physical problem in conjunction with a so-called moving mesh PDE for the mesh itself. Adaptive mesh redistribution methods have important applications in a variety of physical and engineering areas such as solid and fluid dynamics, combustion, heat transfer, material science and others. The physical phenomena in these areas may develop dynamically singular or nearly singular solutions in fairly localized region. The numerical investigation of these physical problems may require extremely fine meshes 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 can improve the accuracy of the numerical approximation and correspondingly decrease the computational cost.

Numerical study of free boundaries can be grouped broadly into two categories. One is to solve sharp-interface problems in which one or more variables (or their derivatives) are typically discontinuous across an interface, and the other is to solve a system of parabolic equations in which the interface is specified by a level set of one of the variables. The latter approach, also called phase field approach, has two appealing features: (i) a broad spectrum of distinct problems that can be studied by means of a single set of equations, and (ii) the interface in these problems does not need to be tracked explicitly.

Phase field models are an increasingly popular choice for modeling the motion of multi-phase fluids (see [1] for a recent review). In the phase-field model, sharp fluid interfaces are replaced by thin but nonzero thickness transition regions where the interfacial forces are smoothly distributed. The basic idea is to introduce a conserved order parameter (e.g., mass concentration) that varies continuously over thin interfacial layers and is mostly uniform in the bulk phases. These models allow topological changes of the interface [28] and have many advantages in numerical simulations of the interfacial motion [12]. Thus, it is also known as the diffuse-interface model. More precisely, in this work, a phase-field variable ϕ is introduced, which can be thought of as the volume fraction, to demarcate the two species and indicate the location of the interface. A mixing energy is defined based on ϕ which, through a convection–diffusion equation, governs the evolution of the interfacial profile. The phase-field method can be viewed as a physically motivated level-set method, and Lowengrub and Truskinovsky [28] have argued for the advantage of using a physically determined ϕ profile instead of an artificial smoothing function for the interface. When the thickness of the interface approaches zero, the diffuse-interface model becomes asymptotically identical to a sharp-interface level-set formulation. It also reduces properly to the classical sharp-interface model in general. Recently, many researchers have employed the phase field approach in various fluid environments [19,25,27,30]. Based on an energetic variational formulation, Liu and Shen [25,26] employed a phase field model to describe the mixture of two incompressible Newtonian fluids. The mixing energy studied is related to the usual Ginzburg–Laudau model for phase evolutions.

Most numerical methods used to solve the phase-field equations have employed stationary uniform meshes, see examples in [9,20,26,38,41]. It is well known the importance that the diffused interface be well resolved if the correct dynamics are to be reproduced. As the phase interface moves in time it is clear that an efficient numerical procedure to solve the phase-field equations should incorporate some form of mesh adaptation as necessitated by the phase field being invariably away from the interface region. There have been two approaches in doing this. One is to use the local mesh refinement method, i.e. *h*-method (see e.g. [8,31]). The adaptive mesh is generated by adding or removing grid points to achieve a desired level of accuracy, which allows a systematic error analysis. However, the local mesh refinement approach requires complicated data structures and technically complex methods/means to communicate information among different levels of refinement. The other is to use moving mesh methods in our paper, i.e. *r*-method, which requires less complicated data structures than the local mesh refinement methods. The algorithm includes two independent parts: mesh-redistribution and PDE evolution. The second part is independent of the first, which can be any of the standard codes for the given PDEs. In the mesh redistribution approach, the adaptive method can keep the total number of grid points unchanged, and cluster more grid points in areas with singularities or large solu-

tion gradients. In the past two decades the moving mesh methods have been proven very useful for time-dependent problems with localized singularities, see e.g. [2,3,7,13,14,23,34,35]. The basic idea of moving mesh method is to construct a transformation from a logical domain (or called computational domain) to the physical domain. A fixed mesh is given on the logical domain, and the transformation is realized by solving moving mesh PDEs or minimization problems for a mesh functional, see e.g. [10,17,24]. Computational cost of moving mesh methods can be efficiently minimized with locally varying time steps [35]. Recently, Mackenzie and Robertson have put forward a simple moving mesh strategy for interface propagation problems. They also developed a moving mesh method for the one-dimensional phase-change problems modeled by the phase-field equations [29]. In another recent work, Becket et al. [4] developed a moving finite element method for the solution of the two-dimensional phase-field equations. Tan et al. [34] develop a simple moving mesh method for one- and two-dimensional phase-field equations. One should note, however, that the phase-field model employed in this paper is different from those in [4,29,34]; it models a specific type of the mixture of two incompressible fluids. Very recently, Zhang and Tang [44] did a similar work on adaptive moving mesh methods of the phase field.

The main objective of this work is to develop an efficient adaptive moving mesh method to solve a phase field model for the mixture flow of two incompressible fluids. Our approach is based on the strategy proposed in [23] by decoupling the mesh motion and the PDE evolution. This approach requires using an interpolation to transform the information from the old mesh to the new mesh. 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 equations involving monitor functions. This approach allows fast solution solvers such as multi-grid methods to solve for the resulting system. Again, solution interpolations are employed so that time-independent mappings at each time can be utilised.

This paper is organized as follows. In Section 2, we describe a phase field model for the mixture of two incompressible fluids. Sections 3 and 4 describe the used numerical method. The 2D moving mesh technique will be described in Section 5. Numerical results are included in Section 6. Some concluding remarks will be made in Section 7.

2. A phase field model for the mixture of two incompressible fluids

Let Ω_p be a two-dimensional physical domain filled with two incompressible fluids separated by a free moving interface. As in [26], we introduce a phase function $\phi(\mathbf{x},t)$ to identify the two fluids ($\{\mathbf{x}:\phi(\mathbf{x},t)=1\}$ is occupied by fluid 1 and $\{\mathbf{x}:\phi(\mathbf{x},t)=-1\}$ by fluid 2), and consider the following Ginzburg–Landau type of mixing energy:

$$W(\phi,\nabla\phi)=\int_{\Omega_p}\left[\frac{1}{2}|\nabla\phi|^2+\frac{1}{4\eta^2}(\phi^2-1)^2\right]\mathrm{d}\mathbf{x}. \tag{2.1}$$

The first part of the energy represents the hydrophillic interaction between the molecules and the second part the hydrophobic interactions. It is the competition between these two parts of the energy that gives the hydrostatic configuration of the interface (see for instance [26]). The constant η can be viewed as the ratio between these two parts of energies. The interface is represented by $\{\mathbf{x}:\phi(\mathbf{x},t)=0\}$ with a transition layer of thickness η.

The dynamics of ϕ can be driven by either Allen–Cahn or Cahn–Hillard types of gradient flow, depending on the choice of dissipative mechanism. The former leads to the Allen–Cahn equation:

$$\phi_t+\mathbf{u}\cdot\nabla\phi=-\gamma\frac{\delta W}{\delta\phi}=\gamma(\Delta\phi-h(\phi)), \tag{2.2}$$

while the latter leads to the Cahn–Hillard equation:

$$\phi_t+\mathbf{u}\cdot\nabla\phi=\nabla\cdot\left(\gamma\nabla\frac{\delta W}{\delta\phi}\right)=-\gamma\Delta(\Delta\phi-h(\phi)), \tag{2.3}$$

where $h(\phi)=\mathscr{H}'(\phi)$ and $\mathbf{u}=(u,v)$ is the velocity vector of the fluids. Here $\mathscr{H}(\phi)=(|\phi|^2-1)^2/4\eta^2$ is the usual double-well potential. The parameter γ represents the elastic relaxation time. As $\gamma\to 0$, the limiting ϕ satisfies the transport equation, which is equivalent to the mass transport equation (for incompressible fluids).

Hence this formulation can also be viewed as the link (relaxation) between a mass average (in the kinetic energy) and a volume average (in the elastic energy) between the two species.

In this paper, we will use the Allen–Cahn dynamics since its numerical treatment is simpler than that of the Cahn–Hilliard type which involves fourth-order differential operators. It is noted that the solution ϕ of (2.2) does not preserve overall volume fraction. So we introduce a Lagrange multiplier $\xi(t)$ into the Allen–Cahn model to conserve the volume as a constraint [40]. More precisely, the modified Allen–Cahn equation reads like follows:

$$\phi_t + \mathbf{u}\cdot\nabla\phi = \gamma(\Delta\phi - h(\phi) + \xi(t)), \tag{2.4}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_p}\phi\,\mathrm{d}\mathbf{x} = 0. \tag{2.5}$$

We now describe the governing equations for the fluid flow. The momentum equation takes the form, which can be derived by the least action principle [25,26]:

$$\rho(\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u}) = -\nabla p + \nabla\cdot\sigma, \tag{2.6}$$

where ρ is the density, p is the pressure and σ is the deviatoric stress tensor that includes the viscous tensor and the induced elastic stress tensor. When we take into account the competition between the kinetic energy and the elastic energy (i.e., the mixing energy), we find:

$$\sigma = v[\nabla\mathbf{u} + (\nabla\mathbf{u})^{\mathrm{T}}] - \lambda(\nabla\phi\otimes\nabla\phi), \tag{2.7}$$

where v is the dynamic viscosity coefficient, the term $\nabla\phi\otimes\nabla\phi$ is the usual tensor product, i.e. $(\nabla\phi\otimes\nabla\phi)_{ij} = \nabla_i\phi\nabla_j\phi$, and λ corresponds to the surface tension.

With same density (which is taken to be 1) and same viscosity constants, the system governing the mixture of two incompressible fluids can be written as follows:

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \mu\Delta\mathbf{u} + \lambda\nabla\cdot(\nabla\phi\otimes\nabla\phi) = \mathbf{g}(\mathbf{x}), \tag{2.8}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{2.9}$$

and

$$\phi_t + (\mathbf{u}\cdot\nabla)\phi = \gamma(\Delta\phi - h(\phi) + \xi(t)), \tag{2.10}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_p}\phi\,\mathrm{d}\mathbf{x} = 0, \tag{2.11}$$

where $\mathbf{g}$ is the external body force. The coupled nonlinear system (2.8)–(2.11) will be subjected to the initial conditions

$$\mathbf{u}|_{t=0} = \mathbf{u}_0, \quad \phi|_{t=0} = \phi_0,$$

and appropriate boundary conditions.

In the above system, the induced elastic stress $\nabla\phi\otimes\nabla\phi$ is due to the mixing of the different species. From this, we can see that $\Delta\phi\nabla\phi = \nabla\cdot(\nabla\phi\otimes\nabla\phi) - \nabla(\frac{1}{2}|\nabla\phi|^2)$ gives the corresponding elastic force, and the $\nabla(\frac{1}{2}|\nabla\phi|^2)$ term can be absorbed into the pressure in the computation.

3. Projection method

We employ a semi-staggered incremental pressure-correction projection scheme [16] for the discretization of the incompressible Navier–Stokes equations. We now describe our numerical approach. Given the velocity $\mathbf{u}^n$, the pressure p^n and the phase function ϕ_n at time level n, we obtain the velocity $\mathbf{u}^{n+1}$ and the p^{n+1} at time level $n+1$ using three stages. In the first stage, the momentum equations are solved for an approximate velocity field which is not generally divergence-free. In the second stage, we need to solve a Poisson-like equation with Neumann-type boundary conditions. In the third stage, the pressure and the velocity fields are corrected to satisfy the continuity equation. To summarize, the algorithm traces the following steps:

- Step 1: Compute an intermediate velocity field $\tilde{\mathbf{u}}^{n+1}$ by solving

$$\frac{\tilde{\mathbf{u}}^{n+1}-\mathbf{u}^n}{\Delta t^n}+(\mathbf{u}^n\cdot\nabla)\mathbf{u}^n=-\nabla p^n+\mu\Delta\tilde{\mathbf{u}}^{n+1}+\hbar(\phi^n),\quad \tilde{\mathbf{u}}^{n+1}|_{\partial\Omega_p}=0, \tag{3.1}$$

where $\hbar(\phi^n)=\lambda(\Delta\phi^n)\nabla\phi^n+\mathbf{g}(\mathbf{x})$.

- Step 2: Compute ψ^{n+1} by solving the Poisson-like equation

$$.\nabla\cdot\nabla\psi^{n+1}=\frac{1}{\Delta t^n}\nabla\cdot\tilde{\mathbf{u}}^{n+1},\quad \left.\frac{\partial\psi^{n+1}}{\partial n}\right|_{\partial\Omega_p}=0. \tag{3.2}$$

- Step 3: Once the potential ψ is obtained by solving Eq. (3.2), both the pressure and velocity field $(p^{n+1},\mathbf{u}^{\mathbf{n+1}})$ are updated as

$$\mathbf{u}^{n+1}=\tilde{\mathbf{u}}^{n+1}-\Delta t^n\nabla\psi^{n+1},\quad \mathbf{u}^{n+1}\cdot n|_{\partial\Omega_p}=0, \tag{3.3}$$

$$p^{n+1}=p^n+\psi^{n+1}-\mu\nabla\cdot\tilde{\mathbf{u}}^{n+1}. \tag{3.4}$$

To enhance stability, it is also noted that we advance the time stepping semi-implicity, i.e., the viscosity terms are treated implicitly while the nonlinear transport terms are treated explicitly in the above scheme.

3.1. Spatial discretization

Above, we discuss the temporal discretization of the Navier–Stokes equation. Now we describe the spatial discretization. Let $\mathbf{x}=(x,y)$ and $\boldsymbol{\xi}=(\xi,\eta)$ denote the physical and computational coordinates, respectively. A coordinate mapping from the computational domain Ω_c to the physical domain Ω_p is given by

$$x=x(\xi,\eta),\quad y=y(\xi,\eta), \tag{3.5}$$

and the inverse map is

$$\xi=\xi(x,y),\quad \eta=\eta(x,y). \tag{3.6}$$

Let the computational grid have unit spacing $(\Delta\xi=\Delta\eta=1)$ in the computational plane, so that integers (i,j) can be used for the discrete computational coordinates. In this work, we employ a semi-staggered grid method, i.e., the potential ψ, the pressure p and the phase ϕ are defined at the cell center, while the velocity $\mathbf{u}$ is defined at the cell corner. Fig. 1 shows a typical control cell for a semi-staggered grid in the physical domain (x,y) and in the computational domain (ξ,η). The volume flux is defined on its corresponding cell face. The mesh is defined by two-dimensional grid points $\mathbf{x}_{j,k}=(x_{j,k},y_{j,k})$, while the center $\bar{\mathbf{x}}_{j,k}=(\bar{x}_{j,k},\bar{y}_{j,k})$ of the control cell $A_{j+\frac{1}{2},k+\frac{1}{2}}$ is defined as the average of the coordinates of its four corners. For a variable φ, one may note that

$$\varphi_x=\frac{1}{J}[y_\eta\varphi_\xi-y_\xi\varphi_\eta],\quad \varphi_y=\frac{1}{J}[-x_\eta\varphi_\xi+x_\xi\varphi_\eta], \tag{3.7}$$

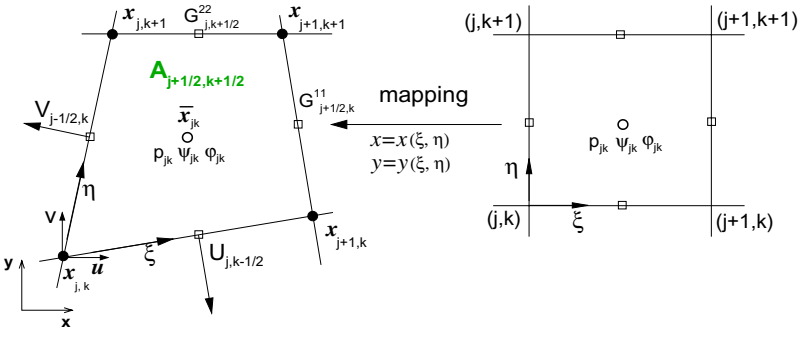


Fig. 1. A control volume of the semi-staggered grid and the mapping in two dimensions.

where $J = x_\xi y_\eta - x_\eta y_\xi$ is the Jacobian determinant of the coordinate transformation. Using the transform formula (3.7), all derivatives in the velocity gradient $\nabla\mathbf{u}$, the pressure gradient ∇p and the phase gradient $\nabla\phi$ are mapped into the logical domain which is equipped with a (fixed) uniform mesh. The derivatives in the logical domain are approximated using a central difference approach.

The viscous terms $\Delta\mathbf{u}$ and $\Delta\phi$ in the momentum equation are approximated using the similar discretized method with $\Delta\phi$ in the phase equation, except that the former are approximated on the cell vertex while the latter is approximated on the cell center, which is further discussed in Section 4.

In the first step of the projection method, the discretization for (3.1) leads to two large sparse positive definite linear systems. We employ a robust multi-grid method from Zeeuw [42] to solve them. This particular multi-grid method can efficiently handle the high-contrast variable coefficients introduced by the mesh map.

In the second step of the projection method, we use the finite volume method to solve the Poisson-like equation (3.2). Integrating $\nabla\cdot\mathbf{u}$ over the control volume $A_{j+\frac{1}{2},k+\frac{1}{2}}$, and using the divergence theorem gives rise to

$$\int_{A_{j+\frac{1}{2},k+\frac{1}{2}}} \nabla\cdot\mathbf{u}\,\mathrm{d}\mathbf{x} = \sum_{i=1}^{4}\int_{L^i}\mathbf{U}_{n^i}\,\mathrm{d}l \doteq U_{j+\frac{1}{2},k} - U_{j-\frac{1}{2},k} + V_{j,k+\frac{1}{2}} - V_{j,k-\frac{1}{2}}. \tag{3.8}$$

Here $\mathbf{U}_{n^i} = un_1^i + vn_2^i$, where $\mathbf{n}^i = (n_1^i, n_2^i)$ denotes the outward unit vector of the ith edge L_i of the control volume $A_{j+\frac{1}{2},k+\frac{1}{2}}$, $i = 1,\ldots,4$. The normal components of mass flux through the east and north faces of the cell are then given by

$$U_{j+\frac{1}{2},k} = (\hat{y}_\eta u - \hat{x}_\eta v)_{j+\frac{1}{2},k}, \quad V_{j,k+\frac{1}{2}} = (\hat{x}_\xi v - \hat{y}_\xi u)_{j,k+\frac{1}{2}},$$

where

$$(\hat{\mathscr{A}}_\eta)_{j-\frac{1}{2},k} = \mathscr{A}_{j,k+1} - \mathscr{A}_{j,k}, \quad (\hat{\mathscr{A}}_\xi)_{j,k-\frac{1}{2}} = \mathscr{A}_{j+1,k} - \mathscr{A}_{j,k}, \qquad \mathscr{A} = x \text{ or } y.$$

Mass is conserved for a given cell when its flux components satisfy the discrete analog of Eq. (2.9),

$$U_{j+\frac{1}{2},k} - U_{j-\frac{1}{2},k} + V_{j,k+\frac{1}{2}} - V_{j,k-\frac{1}{2}} = 0. \tag{3.9}$$

Suppose there exists a velocity $(\tilde{u},\tilde{v})$ with corresponding mass flux $(\widetilde{U},\widetilde{V})$ that does not satisfy (3.9). Adding the gradient of a scalar potential, we obtain Eq. (3.3). After adjustment of $(\tilde{u},\tilde{v})$ by Eq. (3.3), the mass-flux components through the east and north faces of the cell become

$$U_{j+\frac{1}{2},k} = \widetilde{U}_{j+\frac{1}{2},k} + \Delta t(G^{11}\psi_\xi)_{j+\frac{1}{2},k} + \Delta t(G^{12}\psi_\eta)_{j+\frac{1}{2},k}, \tag{3.10}$$

$$V_{j,k+\frac{1}{2}} = \widetilde{V}_{j,k+\frac{1}{2}} + \Delta t(G^{21}\psi_\xi)_{j,k+\frac{1}{2}} + \Delta t(G^{22}\psi_\eta)_{j,k+\frac{1}{2}}, \tag{3.11}$$

where

$$G^{11}_{j+\frac{1}{2},k} = [J^{-1}(\hat{x}_\eta\breve{x}_\eta + \hat{y}_\eta\breve{y}_\eta)]_{j+\frac{1}{2},k}, \quad G^{12}_{j+\frac{1}{2},k} = [J^{-1}(\hat{x}_\eta\breve{x}_\xi + \hat{y}_\eta\breve{y}_\xi)]_{j+\frac{1}{2},k},$$
$$G^{21}_{j,k+\frac{1}{2}} = [J^{-1}(\breve{x}_\eta\hat{x}_\xi + \breve{y}_\eta\hat{y}_\xi)]_{j,k+\frac{1}{2}}, \quad G^{22}_{j,k+\frac{1}{2}} = [J^{-1}(\breve{x}_\xi\hat{x}_\xi + \breve{y}_\xi\hat{y}_\xi)]_{j,k+\frac{1}{2}},$$

are Christoffel symbols which measure the degree of deformation of a coordinate system. The ξ- and η-derivatives of $\breve{x}$ and $\breve{y}$ are approximated as follows:

$$(\breve{\mathscr{A}}_\xi)_{j-\frac{1}{2},k} = \bar{\mathscr{A}}_{j,k} - \bar{\mathscr{A}}_{j-1,k}, \quad (\breve{\mathscr{A}}_\xi)_{j,k-\frac{1}{2}} = \frac{1}{4}(\bar{\mathscr{A}}_{j+1,k} + \bar{\mathscr{A}}_{j+1,k-1} - \bar{\mathscr{A}}_{j-1,k} - \bar{\mathscr{A}}_{j-1,k-1}),$$
$$(\breve{\mathscr{A}}_\eta)_{j,k-\frac{1}{2}} = \bar{\mathscr{A}}_{j,k} - \bar{\mathscr{A}}_{j,k-1}, \quad (\breve{\mathscr{A}}_\eta)_{j-\frac{1}{2},k} = \frac{1}{4}(\bar{\mathscr{A}}_{j,k+1} + \bar{\mathscr{A}}_{j-1,k+1} - \bar{\mathscr{A}}_{j,k-1} - \bar{\mathscr{A}}_{j-1,k-1}),$$
$$J_{j+\frac{1}{2},k} = (\breve{x}_\xi\breve{y}_\eta - \breve{x}_\eta\breve{y}_\xi)_{j+\frac{1}{2},k}, \quad J_{j,k+\frac{1}{2}} = (\breve{x}_\xi\breve{y}_\eta - \breve{x}_\eta\breve{y}_\xi)_{j,k+\frac{1}{2}}, \quad \mathscr{A} = x \text{ or } y,$$

where $\bar{\mathscr{A}}_{j,k}$ means the average of $\mathscr{A}$ for the four corners of the control cell $A_{j+\frac{1}{2},k+\frac{1}{2}}$. Substituting Eqs. (3.10) and (3.11) into (3.9), we obtain the discrete form of the Poisson-like equation (3.2) in computational coordinates,

$$\left[(G^{11}\psi_\xi)_{j+\frac{1}{2},k}-(G^{11}\psi_\xi)_{j-\frac{1}{2},k}\right]-\left[(G^{12}\psi_\eta)_{j+\frac{1}{2},k}-(G^{12}\psi_\eta)_{j-\frac{1}{2},k}\right]-\left[(G^{21}\psi_\xi)_{j,k+\frac{1}{2}}-(G^{21}\psi_\xi)_{j,k-\frac{1}{2}}\right]$$
$$+\left[(G^{22}\psi_\eta)_{j,k+\frac{1}{2}}-(G^{22}\psi_\eta)_{j,k-\frac{1}{2}}\right]$$
$$=\frac{1}{\Delta t}\left(\widetilde{U}_{j+\frac{1}{2},k}-\widetilde{U}_{j-\frac{1}{2},k}+\widetilde{V}_{j,k+\frac{1}{2}}-\widetilde{V}_{j,k-\frac{1}{2}}\right), \tag{3.12}$$

where

$$(\psi_\xi)_{j-\frac{1}{2},k}=\psi_{j,k}-\psi_{j-1,k},(\psi_\eta)_{j-\frac{1}{2},k}=\frac{1}{4}(\psi_{j,k+1}+\psi_{j-1,k+1}-\psi_{j,k-1}-\psi_{j-1,k-1}),$$
$$(\psi_\eta)_{j,k-\frac{1}{2}}=\psi_{j,k}-\psi_{j,k-1},(\psi_\xi)_{j,k-\frac{1}{2}}=\frac{1}{4}(\psi_{j+1,k}+\psi_{j+1,k-1}-\psi_{j-1,k}-\psi_{j-1,k-1}).$$

When Eq. (3.12) is satisfied, it is not difficult to find that the velocity field is divergence-free in the sense of (3.9). We refer the reader to [5,21,22] for more details. In our computations, the pure homogeneous Neumann condition is imposed on the boundary. The Neumann condition is incorporated naturally in Eq. (3.12) as addressed in [5]. The algebraic equations arising from Poisson discretizations (3.12) are solved by the SOR method. The efficiency of SOR method depends on the determination of the optimum relaxation parameter ω_{opt}, generally. In this paper, we do not consider any special technique for determining ω_{opt}. For simplicity, in our paper, the relaxation parameter is obtained via several numerical experiments at some different time levels, and a value in the range 1.93–1.95 is found to give optional results for our adaptive grid. In our actual computations, we choose the same value 1.94 as the relaxation parameter. It is found that the maximum iteration times required is less than 300 with a tolerance of 10^{-8} at all the time levels, more often, the number of SOR iterations is much less than 300. So the SOR method is still satisfactory for our problems. However more efficient iterative solvers, such as the special preconditioned conjugate gradient scheme in [5] or multi-grid method in [22], could be easily introduced to solve the resulting linear system. Finally, in the last step of the projection method, all the equations are also transformed to the computational domain and solved with a uniform mesh.

4. Phase evolution

We derive easily from (2.10), (2.11) the Lagrange multiplier $\xi(t)=\frac{1}{|\Omega_p|}\int_{\Omega_p}h(\phi)\,dx$, where $|\Omega_p|$ denotes the area of the solution domain Ω_p. For incompressible flow, the phase equation (Eq. (2.10)) is equivalent to the following form:

$$\phi_t+(u\phi)_x+(v\phi)_y=\gamma(\Delta\phi-h(\phi)+\xi(t)), \tag{4.1}$$

with the boundary condition $\frac{\partial\phi}{\partial n}|_{\partial\Omega_p}=0$. The time-discretized form of Eq. (4.1) is

$$\frac{\phi^{n+1}-\phi^n}{\Delta t^n}+(u^{n+1}\phi^n)_x+(v^{n+1}\phi^n)_y=\gamma[\Delta\phi^{n+1}-h(\phi^n)+\xi(t^n)]. \tag{4.2}$$

To demonstrate just the principal idea for the phase ϕ evolution, denoting $s(\phi)=\gamma(\xi(t)-h(\phi))$, we rewrite Eq. (4.1) in the following form:

$$\phi_t+f(\phi)_x+g(\phi)_y=\gamma\Delta\phi+s(\phi),\quad (x,y)\in\Omega_p. \tag{4.3}$$

Again we transform the underlying PDEs using the coordinate transformation (3.5), and then solve the resulting equations in the computational domain equipped with a (fixed) uniform mesh. The cell-centered finite volume method will be employed to solve the transformed PDEs. Note that

$$\phi_x=\frac{1}{J}[(y_\eta\phi)_\xi-(y_\xi\phi)_\eta],\quad \phi_y=\frac{1}{J}[-(x_\eta\phi)_\xi+(x_\xi\phi)_\eta],$$

where $J=x_\xi y_\eta-x_\eta y_\xi$ is the Jacobian of the coordinate transformation. With the above formulas, the underlying equation (4.3) becomes:

$$\phi_t+\frac{1}{J}F(\phi)_\xi+\frac{1}{J}G(\phi)_\eta=\mathscr{R}+s(\phi),\quad (\xi,\eta)\in\Omega_c, \tag{4.4}$$

where

$$F(\phi) = y_\eta f(\phi) - x_\eta g(\phi), \quad G(\phi) = x_\xi g(\phi) - y_\xi f(\phi), \quad \mathscr{R} = \gamma\Delta\phi.$$

Here Ω_c is the computational domain with a uniform grid (ξ_j, η_k). The key point here is to obtain the transformations for $\Delta\phi$. Note that

$$\phi_{xx} = \frac{1}{J}[(J^{-1}y_\eta^2\phi_\xi)_\xi - (J^{-1}y_\xi y_\eta\phi_\eta)_\xi - (J^{-1}y_\xi y_\eta\phi_\xi)_\eta + (J^{-1}y_\xi^2\phi_\eta)_\eta], \tag{4.5}$$

$$\phi_{yy} = \frac{1}{J}[(J^{-1}x_\eta^2\phi_\xi)_\xi - (J^{-1}x_\xi x_\eta\phi_\eta)_\xi - (J^{-1}x_\xi x_\eta\phi_\xi)_\eta + (J^{-1}x_\xi^2\phi_\eta)_\eta], \tag{4.6}$$

where $J = x_\xi y_\eta - x_\eta y_\xi$ is again the Jacobian determinant of the coordinate transformation. We denote the phase ϕ at the cell center as $\phi_{j,k}$, at the right face center $\phi_{j+\frac{1}{2},k}$, and at the top face center $\phi_{j,k+\frac{1}{2}}$. Applying the symmetric discretizations as in [33,34] to approximate those terms on the right-hand side of (4.5) and (4.6), we are able to approximate the Laplacian $\Delta\phi$ at cell center $(\xi_{j+\frac{1}{2}}, \eta_{k+\frac{1}{2}})$ by

$$(\Delta\phi)_{j,k} = \frac{1}{J_{jk}}\sum_{l=-1}^{1}\sum_{m=-1}^{1} C_{jk}^{lm}\phi_{j+l,k+m}, \tag{4.7}$$

where

$$C_{jk}^{\pm1,-1} = \pm\frac{1}{4}[(J^{-1}y_\xi y_\eta)_{j\pm1,k} + (J^{-1}y_\xi y_\eta)_{j,k-1} + (J^{-1}x_\xi x_\eta)_{j\pm1,k} + (J^{-1}x_\xi x_\eta)_{j,k-1}], \tag{4.8a}$$

$$C_{jk}^{0,\pm1} = (J^{-1}y_\xi^2)_{j,k\pm\frac{1}{2}} + (J^{-1}x_\xi^2)_{j,k\pm\frac{1}{2}}, \tag{4.8b}$$

$$C_{jk}^{\pm1,0} = (J^{-1}y_\eta^2)_{j\pm\frac{1}{2},k} + (J^{-1}x_\eta^2)_{j\pm\frac{1}{2},k}, \tag{4.8c}$$

$$\begin{aligned} C_{jk}^{0,0} = &-(J^{-1}y_\eta^2)_{j+\frac{1}{2},k} - (J^{-1}y_\eta^2)_{j-\frac{1}{2},k} - (J^{-1}y_\xi^2)_{j,k+\frac{1}{2}} - (J^{-1}y_\xi^2)_{j,k-\frac{1}{2}} - (J^{-1}x_\eta^2)_{j+\frac{1}{2},k} - (J^{-1}x_\eta^2)_{j-\frac{1}{2},k} \\ &- (J^{-1}x_\xi^2)_{j,k+\frac{1}{2}} - (J^{-1}x_\xi^2)_{j,k-\frac{1}{2}}, \end{aligned} \tag{4.8d}$$

$$C_{jk}^{-1,1} = \frac{1}{4}(J^{-1}y_\xi y_\eta)_{j-1,k} + \frac{1}{4}(J^{-1}y_\xi y_\eta)_{j,k+1} + \frac{1}{4}(J^{-1}x_\xi x_\eta)_{j-1,k} + \frac{1}{4}(J^{-1}x_\xi x_\eta)_{j,k+1}, \tag{4.8e}$$

$$C_{jk}^{1,1} = -\frac{1}{4}(J^{-1}y_\xi y_\eta)_{j,k+1} - \frac{1}{4}(J^{-1}y_\xi y_\eta)_{j+1,k} - \frac{1}{4}(J^{-1}x_\xi x_\eta)_{j+1,k} - \frac{1}{4}(J^{-1}x_\xi x_\eta)_{j,k+1}, \tag{4.8f}$$

Here, the above ξ- and η-derivatives of x and y are approximated by the standard central difference. Readers can refer to [33,34] for more details. One should note that the discretization of the above Laplacian operator is different from that in the pressure Poisson equation of the previous section. Next we solve (4.4) by a finite volume approach. Denote the control cell $[\xi_j, \xi_{j+1}] \times [\eta_k, \eta_{k+1}]$ by $\tilde{B}_{j+\frac{1}{2},k+\frac{1}{2}}$ and the cell average values by

$$\bar{\phi}_{j,k}^n = \frac{1}{\Delta\xi\,\Delta\eta}\int_{\tilde{B}_{j+\frac{1}{2},k+\frac{1}{2}}} \phi(\xi,\eta,t^n)\,\mathrm{d}\xi\,\mathrm{d}\eta.$$

For ease of notations, below we will drop the top bar for $\bar{\phi}$. We shall discretize the convection term explicitly, whereas treat the diffusion terms implicitly. This gives rise to

$$\frac{\phi_{j,k}^{n+1} - \phi_{j,k}^n}{\Delta t^n} + \lambda_{j,k}\left(\bar{F}_{j+\frac{1}{2},k}^n - \bar{F}_{j-\frac{1}{2},k}^n\right) + \mu_{j,k}\left(\bar{G}_{j,k+\frac{1}{2}}^n - \bar{G}_{j,k-\frac{1}{2}}^n\right) = \mathfrak{R}_{j,k}, \tag{4.9}$$

where

$$\lambda_{j,k} = \frac{1}{\Delta\xi J_{j,k}}, \quad \mu_{j,k} = \frac{1}{\Delta\eta J_{j,k}}, \quad \mathfrak{R}_{j,k} = \gamma(\Delta\phi^{n+1})_{j,k} + s(\phi_{j,k}^n).$$

Here $s(\phi_{j,k}^n) = \gamma\xi(t^n) - \gamma h(\phi_{j,k}^n)$, and $\xi(t^n) = \frac{1}{|\Omega_p|}\sum_{j,k}|A_{j+\frac{1}{2},k+\frac{1}{2}}^n|h(\phi_{j,k}^n)$, where $|A_{j+\frac{1}{2},k+\frac{1}{2}}|$ means the area of the corresponding control cell. The one-dimensional Lax–Friedrichs numerical flux

$$\bar{f}(a,b) = \frac{1}{2}[f(a) + f(b) - \max_\phi\{|f_\phi|\}\cdot(b-a)], \tag{4.10}$$

where the maximum is taken between a and b, will be applied to $\bar{F}$, $\bar{G}$ in the ξ-, η-direction, respectively:

$$\bar{F}_{j-\frac{1}{2},k} = \bar{F}\left(\phi^-_{j-\frac{1}{2},k}, \phi^+_{j-\frac{1}{2},k}\right), \quad \bar{G}_{j,k-\frac{1}{2}} = \bar{G}\left(\phi^-_{j,k-\frac{1}{2}}, \phi^+_{j,k-\frac{1}{2}}\right). \tag{4.11}$$

In order to compute (4.11), a piecewise linear approximation will be used:

$$\phi^-_{j-\frac{1}{2},k} = \phi_{j-1,k} + \frac{\Delta\xi}{2} q_{j-1,k}, \quad \phi^+_{j-\frac{1}{2},k} = \phi_{j,k} - \frac{\Delta\xi}{2} q_{j,k},$$

$$q_{j,k} = (\operatorname{sign}(q^-_{j,k}) + \operatorname{sign}(q^+_{j,k})) \frac{|q^+_{j,k} q^-_{j,k}|}{|q^+_{j,k}| + |q^-_{j,k}|},$$

$$q^-_{j,k} = \frac{\phi_{j,k} - \phi_{j-1,k}}{\Delta\xi}, \quad q^+_{j,k} = \frac{\phi_{j+1,k} - \phi_{j,k}}{\Delta\xi}.$$

5. Moving mesh methods

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 the numerical scheme (3.1)–(3.4) and (4.9), we can advance the numerical solution one time step to $t = t_{n+1}$. Then the following strategy is employed to carry out the grid restructuring [34]:

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.

The key ingredients of our moving mesh methods consist of three parts: mesh equations, monitor functions and interpolation. We shall describe the necessary details in the following.

5.1. Mesh-redistribution

The mesh is generated via 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

$$x = x(\xi, \eta), \quad y = y(\xi, \eta), \tag{5.1}$$

and the inverse map is

$$\xi = \xi(x, y), \quad \eta = \eta(x, y). \tag{5.2}$$

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

$$E[\xi, \eta] = \frac{1}{2} \int_{\Omega_p} (\nabla\xi^T G_1^{-1} \nabla\xi + \nabla\eta^T G_2^{-1} \nabla\eta)\, \mathrm{d}x\, \mathrm{d}y, \tag{5.3}$$

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 (5.3) to control other aspects of the adaptive mesh such as orthogonality and mesh alignment with a given vector field [7,6].

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

$$\nabla \cdot (G_1^{-1}\nabla\xi) = 0, \quad \nabla \cdot (G_2^{-1}\nabla\eta) = 0. \tag{5.4}$$

One of the simplest choices of monitor function is $G_1 = G_2 = \omega I$, where I is the identity matrix and ω is a positive weight function. One typical choice of the weight function is $\omega = \sqrt{1 + |\nabla u|^2}$, where u is a solution of the underlying PDEs. This choice of the monitor function corresponds to Winslows variable diffusion method [39]:

$$\nabla \cdot \left(\frac{1}{\omega}\nabla\xi\right) = 0, \quad \nabla \cdot \left(\frac{1}{\omega}\nabla\eta\right) = 0. \tag{5.5}$$

(5.4) gives the coordinate transformation in mesh generation and adaptation. Grid generation is basically to obtain the curvilinear coordinate system (5.1) from the above elliptic system (5.4). Usually, after solving the system (5.4) for $\boldsymbol{\xi}(\mathbf{x})$, we find the inverse map to obtain $\mathbf{x}(\boldsymbol{\xi})$, which is expensive. Certainly, we can directly solve the corresponding equations on the computational domain Ω_c by interchanging the dependent and independent variables in (5.4). However, the obtained equations are complicated and massive computations are required. An alternative approach, as suggested by Ceniceros and Hou [11], is to consider a functional defined in the computational domain directly:

$$\tilde{E}[x, y] = \frac{1}{2}\int_{\Omega_c} (\tilde{\nabla}^T x G_1 \tilde{\nabla} x + \tilde{\nabla}^T y G_2 \tilde{\nabla} y)\, \mathrm{d}\xi\, \mathrm{d}\eta, \tag{5.6}$$

to replace the convectional (5.3), 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

$$\tilde{\nabla} \cdot (G_1\tilde{\nabla} x) = 0, \quad \tilde{\nabla} \cdot (G_2\tilde{\nabla} y) = 0. \tag{5.7}$$

If we take the monitor function with the simplest form $G_1 = G_2 = \omega I$, then Eq. (5.7) is reduced to

$$\tilde{\nabla} \cdot (\omega\tilde{\nabla} x) = 0, \quad \tilde{\nabla} \cdot (\omega\tilde{\nabla} y) = 0. \tag{5.8}$$

Therefore, the mesh distribution in the physical space can be directly obtained by solving (5.7), which is much simpler than the conventional variational approach (5.3). However, the system (5.7) can produce degenerated grids in some concave regions [15]. The original system (5.4) is more accurate and reliable than the simpler version (5.7) even though it is more complicated. In this paper, all numerical examples have relatively simple geometry, so Eq. (5.7) can be used for the mesh generation.

In our computation, we use an Gauss–Seidel iteration method [36] to solve for the mesh-moving equation (5.5) or (5.8). For example, the iteration method for (5.8) is written as follows:

$$\alpha_{j+\frac{1}{2},k}(\mathbf{x}^{[v]}_{j+1,k} - \mathbf{x}^{[v+1]}_{j,k}) - \alpha_{j-\frac{1}{2},k}(\mathbf{x}^{[v+1]}_{j,k} - \mathbf{x}^{[v+1]}_{j-1,k}) + \beta_{j,k+\frac{1}{2}}(\mathbf{x}^{[v]}_{j,k+1} - \mathbf{x}^{[v+1]}_{j,k}) - \beta_{j,k-\frac{1}{2}}(\mathbf{x}^{[v+1]}_{j,k} - \mathbf{x}^{[v+1]}_{j,k-1}) = 0 \tag{5.9}$$

for $1 \leqslant j \leqslant N_\xi$ and $1 \leqslant k \leqslant N_\eta$, $v = 0, 1, \ldots,$ where

$$\alpha_{j+\frac{1}{2},k} = \omega(u^{[v]}_{j\pm\frac{1}{2},k}) = \omega\left(\frac{1}{2}\left(u^{[v]}_{j\pm\frac{1}{2},k+\frac{1}{2}} + u^{[v]}_{j\pm\frac{1}{2},k-\frac{1}{2}}\right)\right),$$

$$\beta_{j+\frac{1}{2},k} = \omega(u^{[v]}_{j,k\pm\frac{1}{2}}) = \omega\left(\frac{1}{2}\left(u^{[v]}_{j+\frac{1}{2},k\pm\frac{1}{2}} + u^{[v]}_{j-\frac{1}{2},k\pm\frac{1}{2}}\right)\right).$$

The iteration is continued until there is no significant change in calculating the 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.

5.2. Solution interpolation on the new mesh

After generating the new mesh at each iterative step according to the monitor function, we need to pass the solution information from the old mesh $(x_{j,k}, y_{j,k})$ to the newly obtained mesh $(\tilde{x}_{j,k}, \tilde{y}_{j,k})$. Many interpolating schemes have been suggested, such as the non-conservative one for the nonlinear Hamilton–Jacobi equation

[37] and the conservative interpolation scheme for the hyperbolic conservation laws [36]. Recently, Zhang [43] presented a new conservative interpolation, which may be more accurate and robust than the Tang and Tang's interpolation scheme [36]. However, it remains to be seen the implementation for higher-dimensions. In our computations, we need to update the velocity, the pressure and the phase on the resulting new mesh based on the information on the old mesh. For the pressure p and phase ϕ, they can be realized by using the following conservative interpolation technique proposed by Tang and Tang [36]:

$$\left|\tilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}\right|\tilde{\varphi}_{j+\frac{1}{2},k+\frac{1}{2}} = \left|A_{j+\frac{1}{2},k+\frac{1}{2}}\right|\varphi_{j+\frac{1}{2},k+\frac{1}{2}} - \left[(c^x\varphi)_{j+1,k+\frac{1}{2}} - (c^x\varphi)_{j,k+\frac{1}{2}}\right] - \left[(c^y\varphi)_{j+\frac{1}{2},k+1} - (c^y\varphi)_{j+\frac{1}{2},k}\right], \tag{5.10}$$

where $c^x_{j,k} = x_{j,k} - \tilde{x}_{j,k}$, $c^y_{j,k} = y_{j,k} - \tilde{y}_{j,k}$, and $\varphi = \phi$ or p. The above formula is obtained using the classical perturbation theory. It is obvious that the discretization form (5.10) satisfies the mass-conservation in the following discrete sense:

$$\sum_{j,k}\left|\tilde{A}_{j+\frac{1}{2},k+\frac{1}{2}}\right|\tilde{\varphi}_{j+\frac{1}{2},k+\frac{1}{2}} = \sum_{j,k}\left|A_{j+\frac{1}{2},k+\frac{1}{2}}\right|\varphi_{j+\frac{1}{2},k+\frac{1}{2}}, \quad \varphi = \phi \text{ or } p,$$

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 theoretical properties of this conservative interpolation can be found in [36].

Similarly, after obtain the new mesh $(\tilde{x}_{j,k}, \tilde{y}_{j,k})$, we still need to update the velocity on the resulting new grid. The velocity is not a conservation variable and there is no need to perform the above conservative interpolation. However, interpolation for this variable plays an important role in our moving mesh method too. The velocity field on the new mesh can be extrapolated as in [37,22] via

$$\mathbf{u}(\tilde{\mathbf{x}}) = \mathbf{u}(\mathbf{x}) + (\tilde{\mathbf{x}} - \mathbf{x}) \cdot \nabla\mathbf{u}. \tag{5.11}$$

This step can also be interpreted from the composite rule of derivatives

$$\frac{d\mathbf{u}}{dt} = \frac{\partial\mathbf{u}}{\partial t} + \mathbf{v} \cdot \nabla\mathbf{u}, \tag{5.12}$$

where $\mathbf{v}$ is the mesh motion speed. Eq. (5.11) can be realized in our paper by using the following non-conservative second-order interpolation formula [37]:

$$\tilde{\mathbf{u}}_{j,k} = \mathbf{u}_{j,k} + (\mathbf{u}_x)_{j,k}(\tilde{x}_{j,k} - x_{j,k}) + (\mathbf{u}_y)_{j,k}(\tilde{y}_{j,k} - y_{j,k}) = \mathbf{u}_{j,k} - (c^\xi)_{j,k}(\mathbf{u}_\xi)_{j,k} - (c^\eta)_{j,k}(\mathbf{u}_\eta)_{j,k}, \tag{5.13}$$

where

$$(c^\xi)_{j,k} = \frac{1}{J_{j,k}}[x_\eta(y - \tilde{y}) - y_\eta(x - \tilde{x})]_{j,k}, \quad (c^\eta)_{j,k} = \frac{1}{J_{j,k}}[y_\eta(x - \tilde{x}) - x_\eta(y - \tilde{y})]_{j,k}.$$

We refer the readers to [37] for the detailed interpolation technique. In general, the updated velocity on new mesh is strictly not divergence-free. On the other hand, a simple method for preserving divergence-free can be obtained by projecting velocity into the divergence-free space after generating the new mesh at the last iterative step, using the same procedure (with $\Delta t^n = 1$) as for the Step 2 of the projection method for the incompressible Navier–Stokes equations, see Eqs. (3.2) and (3.3). This projection method is applied to obtain the divergence-free velocity in our paper. Another approach is given in [13] and can be implemented by solving a linear convection equation to obtain the divergence-free interpolation. Special attention has to be taken to make sure that all the computations above are done in a computational domain with a uniform mesh.

5.3. 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 cannot 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 most basic choice (see [36]) to detect regions with high spatial activity is conventionally the arclength-type monitor function:

$$\omega = \sqrt{1 + \alpha|\nabla\phi|^2}, \tag{5.14}$$

here the parameter α is an 'adaptivity'-parameter which controls the amount of adaptivity. For $\alpha = 0$, we have $\omega = 1$, 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

$$\omega = (1-\beta)\alpha(t) + \beta\|\nabla\phi\|_2, \quad \text{with } \alpha(t) = \int\int_{\Omega_c} \|\nabla\phi\|_2 \,\mathrm{d}\xi\,\mathrm{d}\eta. \tag{5.15}$$

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 monitor (5.15), β is indeed the ratio of points in critical parts:

$$\beta = \frac{\int_{\Omega_p} \beta\|\nabla\phi\|_2 \,\mathrm{d}x}{\int_{\Omega_p} (1-\beta)\alpha(t) + \beta\|\nabla\phi\|_2 \,\mathrm{d}x}. \tag{5.16}$$

In this work, we will use the above improved 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.

In order to obtain smoother transitions in the mesh, rather than merely using Eqs. (5.8), an additional filter is applied to the monitor functions. Instead of working with ω_{ij}, the smoothed values

$$\bar{\omega}_{i,j} \leftarrow \frac{4}{16}\omega_{i,j} + \frac{2}{16}(\omega_{i+1,j} + \omega_{i-1,j} + \omega_{i,j+1} + \omega_{i,j-1}) + \frac{1}{16}(\omega_{i-1,j-1} + \omega_{i-1,j+1} + \omega_{i+1,j-1} + \omega_{i+1,j+1})$$

are being used in the mesh equations.

5.4. Solution procedure

Our solution procedure consists of two independent parts: PDE evolution and mesh-redistribution. In the previous sections, we describe the actual numerical discretization. Given the velocity $\mathbf{u}^n$ and the phase function ϕ^n at time t^n, the outline of our scheme can be illustrated by the following steps:

1. *Solve the Navier–Stokes equations to obtain* $\mathbf{u}^{n+1}$. Use a semi-staggered incremental pressure-correction projection method as described in Section 3.
2. *Solve the phase equation to* ϕ^{n+1}. This is done by using the conservative finite volume methods as described in Section 4 and the new velocity field $\mathbf{u}^{n+1}$.
3. *Mesh motion and solution interpolation.* Compute the monitor function, evolve/move the mesh and then update the velocity, pressure and phase function on the new mesh, as described in Section 5.

The last step above is in fact an iteration step and in general requires a few iterations at each time step. However, in our numerical computations, 1–4 iterations are sufficient to obtain a satisfactory mesh at each time level except at the initial stage. The projection method is performed to obtain a divergence-free velocity at the end of the iterative mesh redistribution. At each time step, we need to solve a second-order elliptic equation (3.1) for $\mathbf{u}^{n+1}$, a Poisson-like equation (3.12) for ψ^{n+1}, and a second-order elliptic equation (4.9) for

ϕ^{n+1}. They can be efficiently solved by via a robust multi-grid method developed by Zeeuw [42] or the SOR method.

6. Numerical examples

In this section, we will apply our numerical method to solve for several problems. The first one is to consider the surface tension effects, the second one is to consider both the surface tension effects and Allen–Cahn dissipation, and the last one is the coalescence of two kissing bubbles. In the numerical examples, for the purpose of comparison with the results in [26], we use the following physical parameters unless otherwise specified:

$$\eta = 0.02, \quad \lambda = 0.1, \quad \mu = 0.1, \quad \gamma = 0.1.$$

We recall that η is the capillary width (mixing region) of the fluids, λ/η is the surface tension constant, μ is the viscosity and γ is the "elastic" relaxation time. In all computations, the initial velocity and pressure are taken to be zero on the square domain $[0, 2\pi] \times [0, 2\pi]$ while the initial condition for ϕ is specified in each example. All computations are carried out on PC Pentium 4 with 3.00 GHz.

Example 6.1 (*Surface tension effects*). This test exhibits the surface tension effects of the model. We start with a square bubble which sits at the center of the domain. The length of side of the square bubble is 2. We take the phase ϕ inside and outside the bubble to be 1 and -1, respectively. Due to the surface tension, the square bubble quickly deforms into a circular bubble. In fact, if we choose $\lambda = 0$ (i.e., no fluid in the system), the bubble will not deform.

In Figs. 2 and 3, we show the contour plots of the phase, interface positions and the corresponding adaptive meshes at times $t = 0.1, 0.4, 0.7$, and 2.5, obtained using a 65^2 moving grid. From these figures, it can be seen that, as expected, due to the surface tension, the square bubble quickly deforms into a circular bubble. It is found that our moving mesh results are in a very good agreement with the results of Liu and Shen [26] obtained by the Fourier-spectral method. In [26], the problem is computed with 128^2 mesh points, while it can be resolved well with using just only 65^2 mesh points in our method. Note also that the choice of the parameter $\beta = 1$ leads (as expected) to a mesh that approximately places half of the mesh points within the

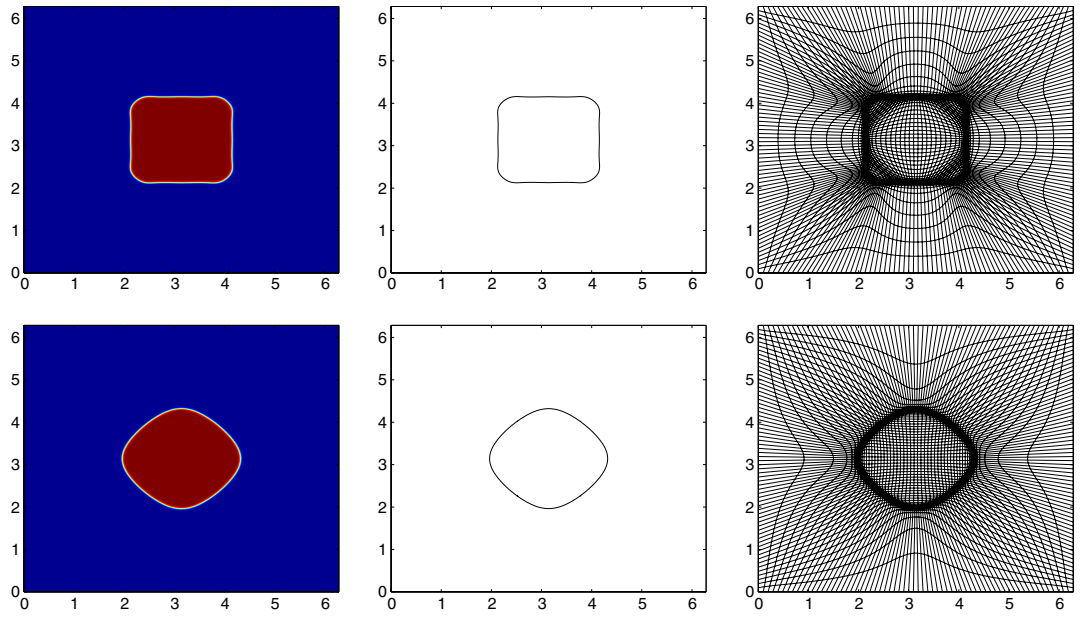


Fig. 2. Example 6.1: Phase evolution (left), interface predictions (middle) and grid (right) with a 65^2 moving grid at $t = 0.1$ and 0.4.

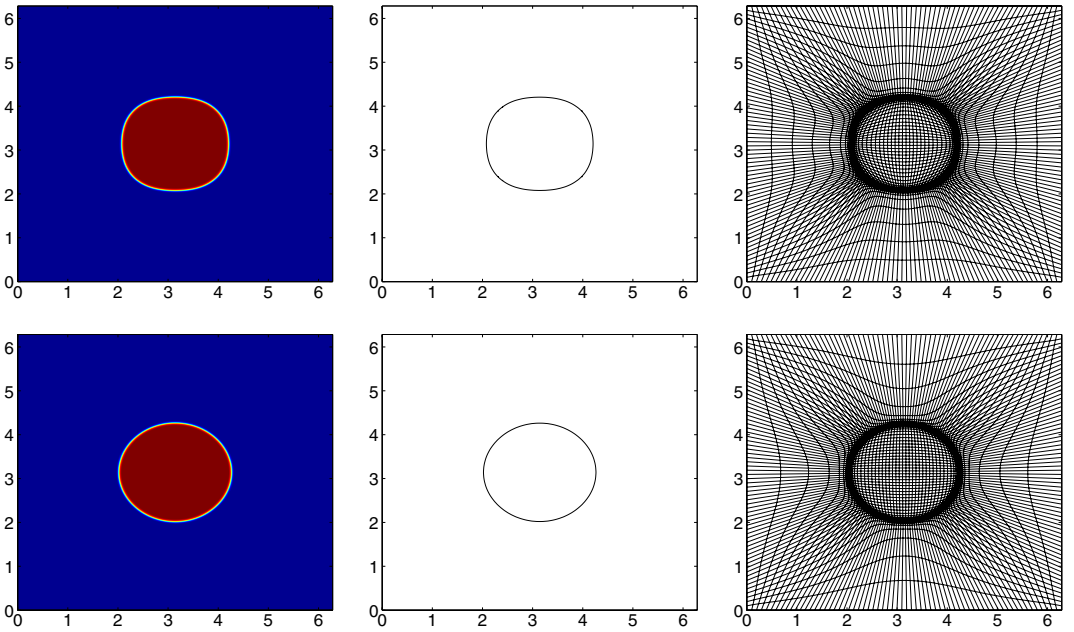


Fig. 3. Example 6.1: Phase evolution (left), interface predictions (middle) and grid (right) with a 65^2 moving grid at $t = 0.7$, and 2.5.

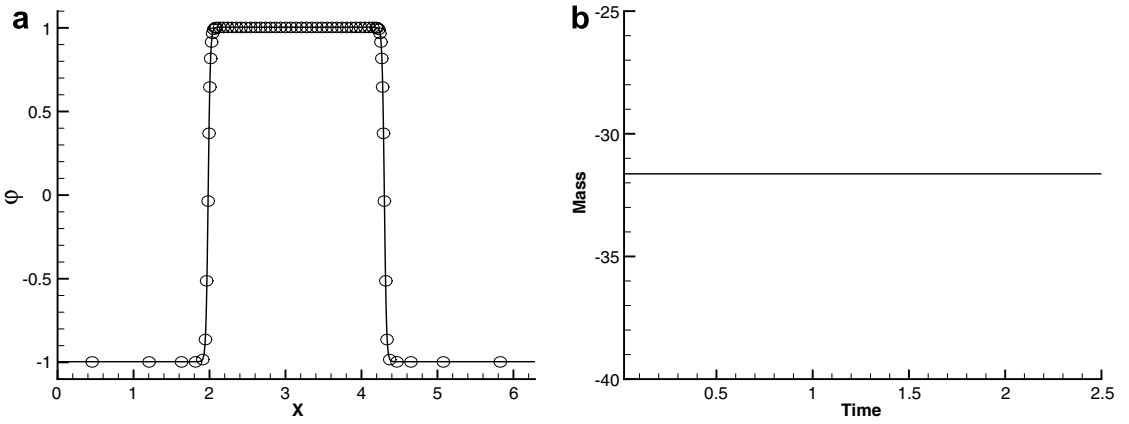


Fig. 4. Example 6.1: Left is adaptive mesh solution for the phase ϕ at $t = 0.4$ with $N_x = N_y = 65$ along the axis $y = \pi$ and the solid line is obtained on a uniform mesh with 257^2 grid points. Right is the plot of the overall volume fraction versus time to depict mass conservation.

interface region. There is no exact analytical solution for this example. To test for accuracy, the problem is computed with the same method on a 257^2 uniform grid. It is observed that the moving mesh results on the coarse grid still agree very well with the finer uniform mesh results and these are graphically indistinguishable. For quantitative comparison, in Fig. 4a, we show the computed phase variable profile along the axis $y = \pi$ at $t = 0.4$, obtained using a 65^2 moving grid and a 257^2 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.

In Table 1, the computing CPU times at different time level (t) using moving mesh method on a 65^2 moving grid and uniform mesh method on a 257^2 uniform grid are listed. The results show that 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, comparing with corresponding uniform mesh method.

Table 1
Example 6.1: Comparison of the CPU time in seconds for the different mesh methods

Schemes	$t = 0.2$	$t = 0.4$	$t = 0.6$
Moving mesh method	151.28	294.11	435.74
Uniform mesh method	2853.74	5822.07	8718.14

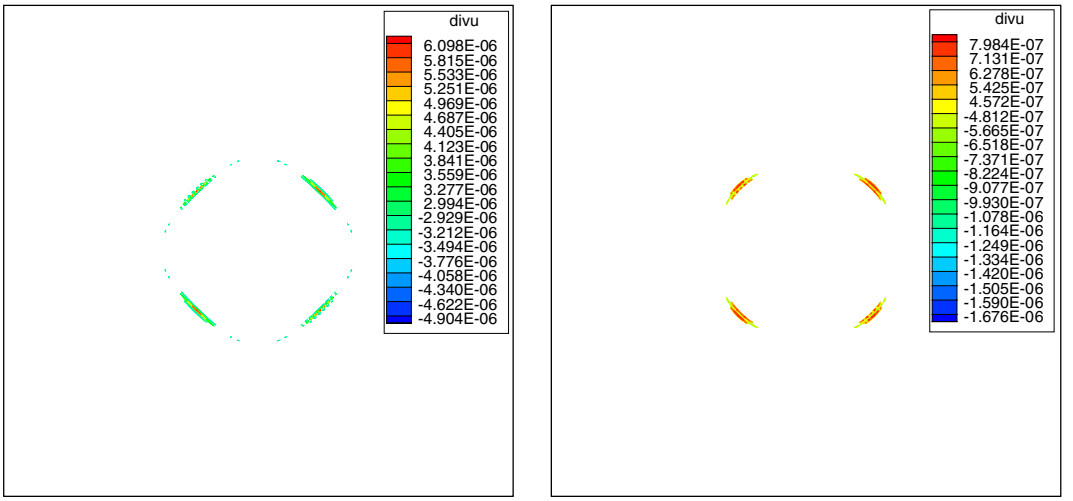


Fig. 5. Example 6.1: The divergence of velocity field at $t = 0.4$ (left) and $t = 0.7$ (right).

In Fig. 4b, we present the plot of the total system "mass" (i.e., $\int_\Omega \phi \, dx$) versus time. It is found that the total system "mass" is conserved very well for all times; this shows the following conservation of total system "mass":

$$\frac{d}{dt} \int_\Omega \phi(x,t) \, dx = 0. \tag{6.1}$$

Hence, the overall volume fraction of the bubble is preserved very well. One may also note that if we start with a circular bubble, the bubble is not deformed and the volume of the bubble is preserved naturally.

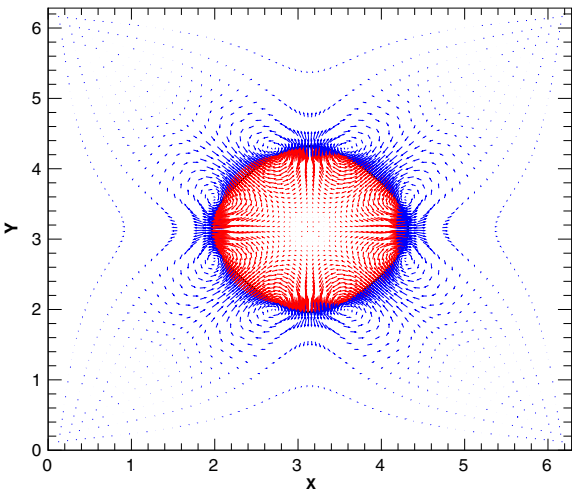


Fig. 6. Velocity field at $t = 0.4$ for Example 6.1.

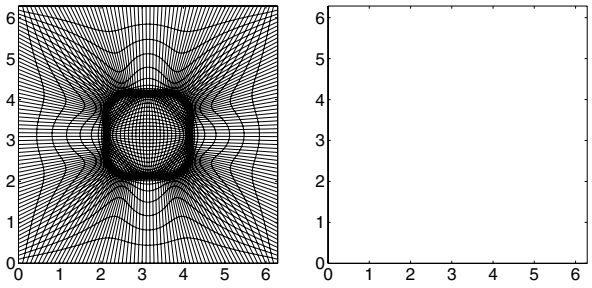


Fig. 5 shows the discrete velocity-divergence at $t = 0.4$ and 0.7 obtained with a 65^2 moving grid. As expected, the divergence-free requirement is largely satisfied; the divergence-free quantity is kept quite small around the interface between the two fluids. Finally, we also show the velocity field at $t = 0.4$ in Fig. 6.

Example 6.2 (*Surface tension effects and Allen–Cahn dissipation*). In this test, we choose the same initial condition as in Example 6.1 but we do not introduce the Lagrange multiple $\xi(t)$ into the Allen–Cahn model. As expected, again we notice that more grid points are clustered in the neighborhood of the interface. The square bubble still deforms into a circular bubble while the size of the bubble shrinks (eventually it shrinks to zero) due to the dissipative mechanism in the Allen–Cahn system. Notice that in both Examples 6.1 and 6.2, the shape of the bubble vibrates tangentially before it becomes a circular bubble (the preferred configuration). This tangential vibration is attributed to the so-called T-modes of the spheric normal modes [32]. It illustrates that the phase field model presented by Liu and Shen [26] captures another important physical aspect of the surface tension.

In Fig. 7, the interface positions and the corresponding adaptive meshes at times $t = 0.1$, 0.4, 0.5, and 2.0 are presented, obtained using a 65^2 moving grid. Again the overall agreement between our moving mesh results and the Fourier-spectral method results of Liu and Shen [26] who used a 128^2 uniform mesh is very satisfactory. The diffused interface is well resolved with far fewer grid points than those in [26]. The ability of moving mesh method to capture and follow the interface is clearly demonstrated in both Examples 6.1 and 6.2, which confirm the effectiveness of our moving mesh method.

Example 6.3 (*Coalescence of two kissing bubbles*). In the last example, we consider the coalescence of two kissing bubbles. We start with two circular kissing bubbles with the same radius $r = 0.25\pi$, of which the centers are located at $(0.75\pi, \pi)$ and $(1.25\pi, \pi)$ initially. The phase ϕ is taken to be 1 inside the two bubbles, while ϕ is taken to be -1 outside two bubbles. With time, the two bubbles coalesces into one big bubble, very similar to the phenomenon observed in [26]. This observation is a culmination of the combined surface tension and elastic effect from the phase equation.

In Fig. 8, we plot the numerical solutions, depicting the computed interface positions and the corresponding adaptive meshes at times $t = 0.0$, 0.2, 0.5, and 0.6, which were calculated using a 65^2 moving grid. On the one hand, as desired, it is seen from these figures that approximately half of the grid points are clustered around the interface where the phase function is zero; this increases the resolution of the numerical solutions. On the

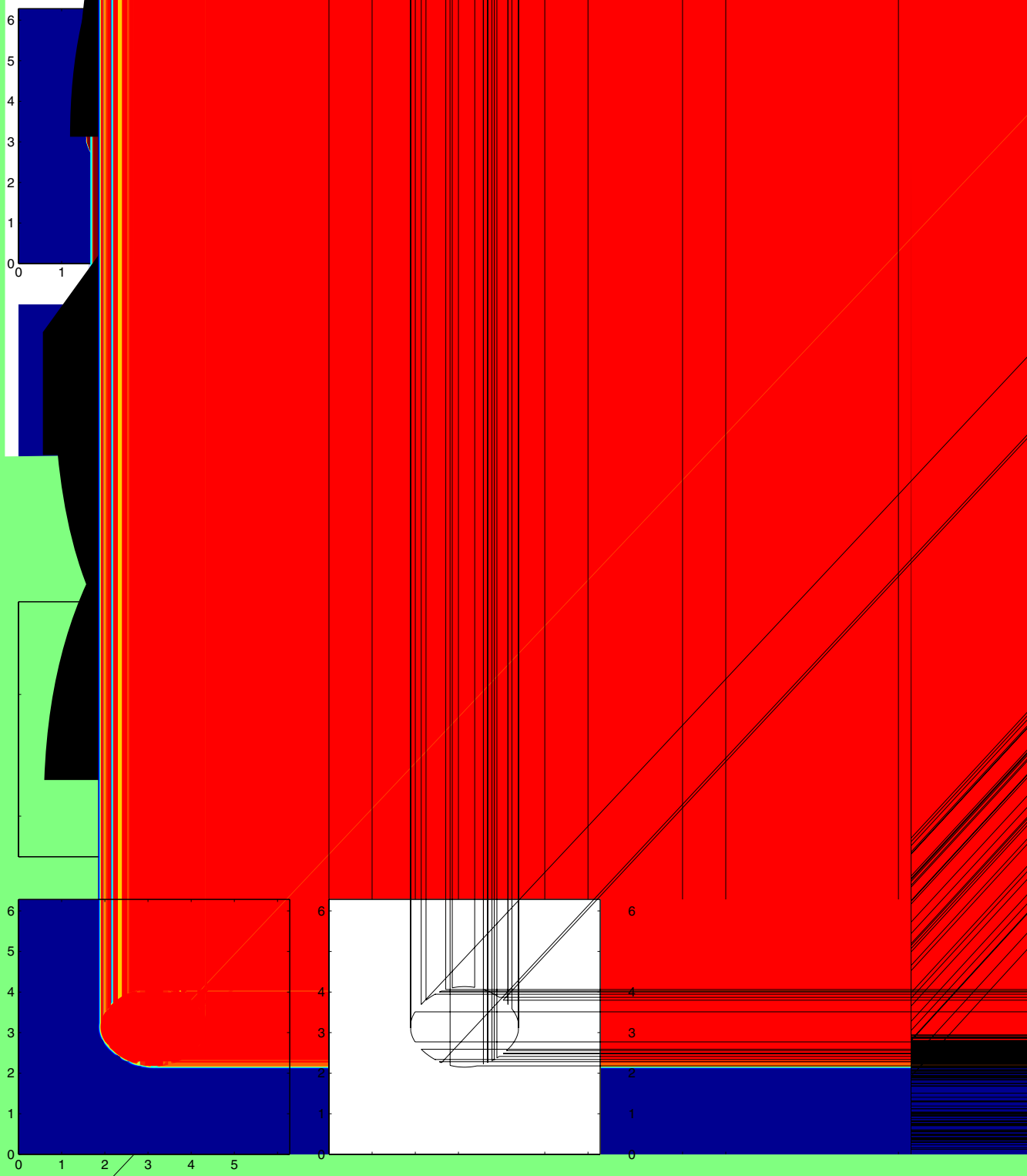


Fig. 8. Example 6.3: Phase evolution (left), interface predictions (middle) and grid (right) with a 65^2 moving grid at $t = 0.0$, 0.2, 0.5, and 0.6 (top to bottom, respectively).

other hand, the diffused interface is well resolved and again the computed interface positions agree well with the results by Fourier-spectral method of Liu and Shen [26]. Compared with the Fourier-spectral method of Liu and Shen [26] who used 128^2 mesh points, we employed only 65^2 mesh points. From this example, we can further see that the adaptive moving mesh does an excellent job of tracking the interface and has no difficulty in dealing with the change in topology, even when they merge. Likewise, the problem is also computed on a 257^2 uniform grid. It is found that the coarser moving mesh results agree very well with the finer uniform mesh results. In Fig. 9a, we show the computed phase variable profile along the axis $y = \pi$ at $t = 0.2$ using a 65^2 moving grid and a 257^2 uniform grid. It is clear that these results are in good quantitative agreement. Again this demonstrates the effectiveness of our moving mesh method and the ability of our method to handle singular topological changes.

Table 2 shows the computing CPU times at different time level (t) using the moving mesh method on a 65^2 moving grid and uniform mesh method on a 257^2 uniform grid. Again it is observed that our moving mesh

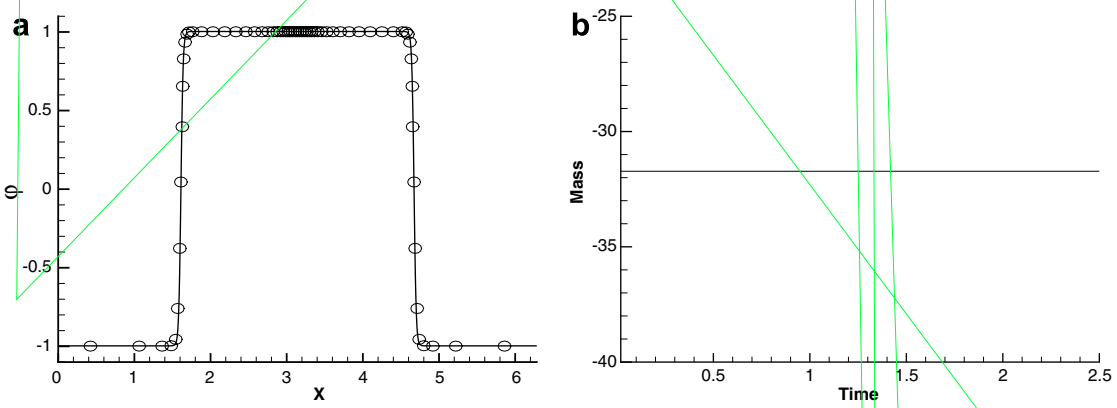
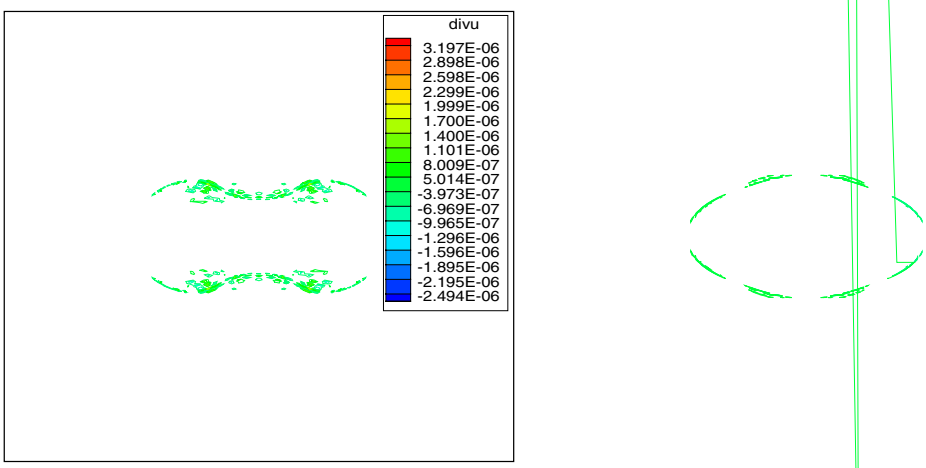


Fig. 9. Example 6.3: Left is adaptive mesh solution for the phase ϕ at $t = 0.2$ with $N_x = N_y = 65$ along the axis $y = \pi$ and the solid line is obtained on a uniform mesh with 257^2 grid points. Right is the plot of the overall volume fraction versus time to depict mass conservation.

Table 2
Example 6.3: Comparison of the CPU time in seconds for the different mesh methods

Schemes	$t = 0.2$	$t = 0.4$	$t = 0.6$
Moving mesh method	148.74	294.84	446.98
Uniform mesh method	2963.81	5815.46	8836.56



method has great advantage in CPU time saving, comparing with the corresponding uniform mesh method. It further demonstrates the performance of our proposed algorithm.

The overall volume fraction of the bubble is also preserved very well in this example, see Fig. 9b. In Fig. 10, the plot of the divergence of the velocity field is presented; it can be seen that the discrete velocity-divergence is very small even in the interface region, and the divergence-free property is well satisfied. The velocity field at $t = 0.3$ is given in Fig. 11.

More computational details are given for Example 6.3. In Table 3, we show the largest and the smallest sizes and their ratios of the adaptive mesh at different time levels. From this table, we can perceive that at the initial time a quite large portion of the grid points is moved to the initial interface region due to the singularity of the initial data. Also notice how the ratios of maximum to minimum of the grid size in the x and y directions change differently with time. A possible reason is that the largest mesh is found near the boundary of domain while the smallest mesh is near the interface. The largest mesh nearest the boundary changes faster in the y-direction while the largest mesh nearest the boundary changes more slowly in the x-direction during the time evolution than that of the corresponding smallest mesh in the interface region (see Fig. 8). As a result, the evaluated ratio of the largest to the smallest grid size in the x-direction becomes some smaller while the same ratio in the y-direction becomes somewhat bigger. If we do not consider the meshes near the boundary, the trend for the change between the ratios of maximum to minimum for the grid size in the x and y directions will not be so clear. In Table 1, the maximum and minimum meshes are defined as

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

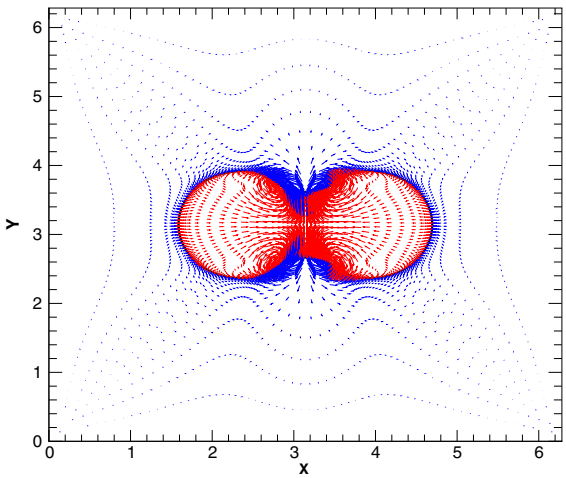


Fig. 11. Velocity field at $t = 0.2$ for Example 6.3.

Table 3
Example 6.3: adaptive mesh with $N_x = N_y = 65$

		$t = 0.0$	$t = 0.2$	$t = 0.3$	$t = 0.6$
Δx	min Δx	6.58e−03	9.26e−03	8.87e−03	8.91e−03
	max Δx	7.84e−01	7.82e−01	7.16e−01	4.34e−01
	max/min	119.15	84.45	80.72	48.71
Δy	min Δy	7.43e−03	8.91e−03	6.51e−03	1.08e−02
	max Δy	5.21e−01	6.73e−01	5.12e−01	1.12
	max/min	70.12	75.53	78.65	96.43
$\lvert A_{jk}\rvert$	min $\lvert A_{jk}\rvert$	2.53e−04	2.83e−04	3.26e−04	3.24e−04
	max $\lvert A_{jk}\rvert$	6.52e−02	6.46e−02	6.12e−02	9.64e−02
	max/min	257.71	228.27	187.73	297.53

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 control 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 the $\max \Delta y$ and $\min \Delta y$.

Finally, we want to briefly discuss about the time steps used in our computations. On the moving mesh methods for evolution, the selection of time step size is always an issue. For the computations reported in the paper, the time step is taken to be 10^{-3}. By comparing with the uniform mesh approach, to reach the same resolution the moving mesh method does not gain anything significant in time stepping but gains in using much less grid points in space. To increase the efficiency in time stepping, proper local time stepping techniques may be used [35].

7. Concluding remarks

In this paper, we have presented an efficient adaptive moving mesh technique for solving a phase field model for the mixture of two incompressible fluids. Our moving mesh method combined with phase field method is fairly simple to code. The numerical approach is based on the strategy proposed in [23] by decoupling the mesh-moving and PDE evolution at each time step. In this work, we have used the second-order conservative interpolation proposed in [36] to update the phase ϕ on the resulting new grid, while the velocity is re-mapped in a non-conservative approach. A simple method for preserving divergence-free is obtained by projecting velocity into the divergence-free space after generating new mesh at the last iterative step. For the choice of the monitor function, we used an improved parameter-free monitor function [17,18]. The phase-field equation is numerically solved by a conservative scheme, and the coupled incompressible Navier–Stokes equations are solved by the incremental pressure-correction projection scheme based on the semi-staggered grid method. The Lagrange multiplier $\xi(t)$ is introduced into the Allen–Cahn model to conserve the volume. The numerical results are in good agreement with the recent computations by Fourier-spectral method of Liu and Shen [26] and demonstrate the accuracy and effectiveness of our proposed algorithm. In order to obtain the same resolution, our approach needs much lesser grid points and save on computational cost. It is noted that this paper focused on the mixture flow of two incompressible fluids in the same density and viscosity. The scheme can be easily generalized to allow different densities (ρ_1 and ρ_2) and viscosities (μ_1 and μ_2) for the two fluids. In our future work, we will extend our moving mesh scheme to handle more general variations in the density and viscosity, and even some challenging 3D model problems.

Acknowledgement

The authors thank the reviewers' valuable suggestions during the revision of the paper.

References

[1] D.M. Anderson, G.B. McFadden, A.A. Wheeler, Diffuse-interface methods in fluid mechanics, Appl. Math. Lett. 30 (1998) 139–165.
[2] B.N. Azarenok, S.A. Ivanenko, T. Tang, Adaptive mesh redistribution method based on Godunov's scheme, Commun. Math. Sci. 1 (2003) 152–179.
[3] B.N. Azarenok, 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, 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] R.S. Bernard, H. Kapitza, How to discretize the pressure gradient for curvilinear MAC grids, J. Comput. Phys. 99 (1992) 288–298.
[6] J.U. Brackbill, An adaptive grid with direction control, J. Comput. Phys. 108 (1993) 38–50.
[7] J.U. Brackbill, J.S. Saltzman, Adaptive zoning for singular problems in two dimensions, J. Comput. Phys. 46 (1982) 342–368.
[8] R.J. Braun, B.T. Murray, Adaptive phase-field computations of dendritic growth, J. Cryst. Growth 174 (1997) 41–53.
[9] G. Caginalp, E.A. Socolovsky, Phase field computations of single-needle crystals, crystal growth and motion by mean curvature, SIAM J. Sci. Comput. 15 (1994) 106–126.
[10] W.M. Cao, W.Z. Huang, R.D. Russell, An *r*-adaptive finite element method based upon moving mesh PDEs, J. Comput. Phys. 149 (1999) 221–244.

[11] H.D. Ceniceros, T.Y. Hou, An efficient dynamically adaptive mesh for potentially singular solutions, J. Comput. Phys. 172 (2001) 609–639.
[12] Y.C. Chang, T.Y. Hou, B. Merriman, S. Osher, A level set formulation of Eulerian interface capturing methods for incompressible fluid flows, J. Comput. Phys. 124 (1996) 449–464.
[13] Y. Di, R. Li, T. Tang, P.-W. Zhang, Moving mesh finite element methods for the incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 26 (2005) 1036–1056.
[14] Y. Di, P.-W. Zhang, Moving mesh kinetic simulation for sheared rodlike polymers with high potential intensities, Commun. Comput. Phys. 1 (2006) 859–873.
[15] A.S. Dvinsky, Adaptive grid generation from harmonic maps on Riemannian manifolds, J. Comput. Phys. 95 (1991) 221–244.
[16] J.L. Guermond, J. Shen, On the error estimates of rotational pressure-correction projection methods, Math. Comp. 73 (2004) 1719–1737.
[17] W.Z. Huang, 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, W. Sun, Variational mesh adaptation II: error estimates and monitor functions, J. Comput. Phys. 184 (2003) 619–648.
[19] D. Jacqmin, Calculation of two-phase Navier–Stokes flows using phase-field modeling, J. Comput. Phys. 155 (1999) 96–127.
[20] A. Karma, W.-J. Rappel, Quantitative phase-field modeling of dendritic growth in two and three dimensions, Phys. Rev. E 57 (1998) 4323–4349.
[21] J. Li, Numerical resolution of Navier–Stokes equation with reconnection of interfaces, Volume tracking and application to atomization, Ph.D. Thesis, University of Paris VI, 1996.
[22] J. Li, M. Hesse, J. Ziegler, A.W. Woods, An arbitrary Lagrangian Eulerian method for moving-boundary problems and its application to jumping over water, J. Comput. Phys. 208 (2005) 289–314.
[23] R. Li, T. Tang, P.-W. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, J. Comput. Phys. 170 (2001) 562–588.
[24] R. Li, T. Tang, 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.
[25] C. Liu and S. Shkoller, Variational phase field model for the mixture of two fluids, preprint, 2001.
[26] C. Liu, J. Shen, A phase field model for the mixture of two incompressible fluids and its approximation by a Fourier-spectral method, Physica D 179 (2003) 211–228.
[27] C. Liu, S.J. Tavener, N.J. Walkington, A variational phase field model for Marangoni–Bénard convection with a deformable free surface, preprint, 2001.
[28] J. Lowengrub, L. Truskinovsky, Quasi-incompressible Cahn–Hilliard fluids and topological transitions, R. Soc. Lond. Proc. Ser. A Math. Phys. Eng. Sci. 454 (1998) 2617–2654.
[29] J.A. Mackenzie, M.L. Robertson, A moving mesh method for the solution of the one-dimensional phase-field equations, J. Comput. Phys. 181 (2002) 526–544.
[30] G.B. McFadden, A.A. Wheeler, D.M. Anderson, Thin interface asymptotics for an energy/entropy approach to phase-field models with unequal conductivities, Physica D 144 (2000) 154–168.
[31] N. Provatas, N. Goldenfeld, J. Dantzig, Efficient computation of dendritic microstructures using adaptive mesh refinement, Phys. Rev. Lett. 80 (1998) 3308–3311.
[32] F. Stacy, Physics of the Earth, second ed., Wiley, New York, 1977.
[33] Z.-J. Tan, Adaptive moving mesh methods for two-dimensional resistive magneto-hydrodynamic PDE models, Comput. Fluids 36 (2007) 758–771.
[34] Z.-J. Tan, T. Tang, Z.-R. Zhang, A simple moving mesh method for one- and two-dimensional phase-field equations, J. Comput. Appl. Math. 190 (2006) 252–269.
[35] Z.-J. Tan, Z.-R. Zhang, Y.-Q. Huang, T. Tang, Moving mesh methods with locally varying time steps, J. Comput. Phys. 200 (2004) 347–367.
[36] H.Z. Tang, T. Tang, Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws, SIAM J. Numer. Anal. 41 (2003) 487–515.
[37] H.Z. Tang, T. Tang, P.-W. Zhang, Adaptive mesh redistribution method for nonlinear Hamilton–Jacobi equations in two- and three-dimensions, J. Comput. Phys. 188 (2003) 543–572.
[38] S.-L. Wang, R.F. Sekerka, A.A. Wheeler, T. Murray, S.R. Coriell, R.J. Braun, G.B. McFadden, Thermodynamically-consistent phase field-models for solidification, Physica D 69 (1993) 189–200.
[39] A. Winslow, Numerical solution of the quasi-linear Poisson equation in a nonuniform triangle mesh, J. Comput. Phys. 1 (1967) 149–172.
[40] X. Yang, J.J. Feng, C. Liu, J. Shen, Numerical simulations of jet pinching-off and drop formation using an energetic variational phase-field method, J. Comput. Phys. 218 (2006) 417–428.
[41] P. Yue, J.J. Feng, C. Liu, J. Shen, A diffuse-interface method for simulating two-phase flows of complex fluids, J. Fluid Mech. 515 (2004) 293–317.
[42] P.M. De Zeeuw, Matrix-dependent prolongation and restrictions in a blackbox multigrid solver, J. Comput. Phys. 33 (1990) 1–27.

[43] Z.-R. Zhang, Moving mesh method with conservative interpolation based on L^2-projection, Commun. Comput. Phys. 1 (2006) 930–944.
[44] Z.-R. Zhang, H.Z. Tang, An adaptive phase field method for the mixture of two incompressible fluids, Comput. Fluids (in press).