A WEAK FORMULATION FOR THE MULTIPHASE STOKES FLOW PROBLEM WITHOUT BODY FITTING GRIDS

Ningchen Ying, Songming Hou, Shingyu Leung & Hongkai Zhao

Abstract

We develop an effective interface tracking method to simulate the incompressible Stokes flow with moving interfaces. The Stokes equations are first rewritten into a system of elliptic equations with singular sources which can be efficiently solved by a simple weak formulation proposed in [**11**]. The key idea is to first split the solution into a singular part and a regular part additively. The singular part captures the interface conditions, while the regular part approximates the equations in the whole domain, which can be solved by the standard finite element formulation. We carefully design numerical methods to interpolate the velocity to the moving interface. Numerical tests are carried out to demonstrate the accuracy and other properties of our method.

1. Introduction

In this paper we develop an interface tracking method for solving the incompressible Stokes flow problem with moving interfaces. The method is developed based on a uniform triangular mesh and does not require mesh adaptivity. In general, it can also be extended to a nonuniform triangle mesh. One main ingredient of the method is an elliptic interface solver which gives the instantaneous velocity and pressure in the flow. There are many possible choices. For example, we can apply the immersed boundary method (IBM) proposed in [**38**, **39**], the immersed interface method (IIM) in [**26**], the ghost fluid method (GFM) in [**5**] and an extension in [**34**, **35**], and a weak formulation solver developed in [**10**] and later extended to [**13**, **9**, **12**, **11**, **54**, **58**, **57**, **56**, **55**].

The IBM is a diffuse interface non-body-fitted solver focusing on the information near the boundary. It considers only the singular source on the interface but extends their influence to the near boundary cells by using a smooth discrete delta function. Due to the discrete delta function, it is in general of first order accuracy on the interface. Some high-order IBMs have been designed in [**20**, **4**] and some high-order discrete delta functions have been developed in [**49**, **50**, **51**]. The IBM was first applied to the moving interface problems in [**53**]. Later, with the high-order schemes designed, [**46**, **21**, **18**, **17**, **3**] have applied IBM to several moving interface problems, including the Stokes flow and the Navier-Stokes flow problems. These works have also provided a solution to handle the topological changes during the interface moving in two-dimensional case in [**18**]. Recently, the IBM has been extended to three-dimensional cases in [**14**, **19**, **43**]. A summary of IBM and its applications can be found in [**39**].

Similar to IBM, the IIM is also a non-body-fitted solver while it focuses on the sharp interface. In IIM, the finite difference scheme is used to discretize the interface problem by modifying the information at irregular grid nodes, which enforces the discrete elliptic operator satisfying the conditions on both sides of the interface. This idea is proposed by fitting the interface jump conditions at interface grid nodes with local Taylor expansions of the elliptic operator near the interface. The IIM is of second-order accuracy for both irregular and regular area and preserves the jump conditions between inside interface area and outside interface area. [**30**, **31**, **1**, **33**] have also proposed some more efficient and robust versions of IIM. The IIM can also be extended to moving interface problem. The first application is to solve the Stokes Flow problem in [**28**, **27**, **29**]. Later, the IIM has been successfully applied to the interface related problems including [**6**, **32**, **48**].

Some modified IBM and IIM have been proposed to solve the interface elliptic problems, such as the decomposed immersed interface method [**37**] and ghost-cell immersed boundary method [**52**]. There are also series of papers which combine two methods, IBM and IIM, to solve the moving interface problems in [**16**, **15**]. Furthermore, more methods using Cartesian grids take use of the phase field method [**2**] and the capacitance matrix method in [**40**].

The elliptic interface solver in this paper is based on a weak formulation to solve the elliptic interface problems developed in [**10**, **11**]. It is of second order accuracy and takes advantage of non-body fitted numerical method for elliptic interface problem with discontinuous jump conditions and singular source terms. The main idea is to decompose the solution into two parts, a singular part and a regular part. The regular part fits the elliptic equation without any jump conditions. While the singular part is explicitly constructed by the general finite element base function to recover the discontinuous properties inside and outside the interface. We simply treat the singular part as correction term mentioned in other elliptic interface solver. Since the singular part can be calculated directly, the formulation can be viewed as a type of unfitted finite element method with general matrix coefficient. With this elliptic interface solver, we follow the idea of [**27**] and solve the Stokes flow problem separately by decomposing it into three elliptic equations. During the procedure, we modify the velocities equations with variational form which improves the calculations and does not require the derivatives of pressure.

Most methods we discussed above require one to solve a linear system $A\mathbf{x} = \mathbf{b}$. The efficiency of the overall numerical approach for the moving interface problem, therefore, depends heavily on two main issues. One is whether the method requires the reconstruction of A as the interface evolves. The second one is whether an efficient solver exists for the linear problem which explores the structure of the matrix. General FEM methods, including the FEM implementation of the IIM, unfortunately require the construction of the matrix A at each time step since the basis function depends explicitly on the location where the moving interface crosses the mesh.

Comparing the elliptic solver we are using based on [**11**] with the one from the FDM based IIM, we find that both methods modify only the right hand side vector $\mathbf{b}$ at different time steps as the interface evolves if there is no jump in the diffusion coefficients. For different interface structure, the IIM

uses the Taylor expansion to approximate the local geometry and fit the jump conditions by incorporating it explicitly to the **b**, while the current method uses the same base functions in all mesh. It imposes the jump conditions by extracting the singular parts of the solution in those interface elements which can then be absorbed into the right hand side coefficient vector **b** right away. However, because we are using the weak formulation, we are able to apply integration-by-parts in the proposed formulation to replace ∇p when solving for the moving velocity in the domain. This provides a simple yet accurate way to update the velocity, which is not straight-forward in the FDM based approach.

The rest of the paper is organized as follows. In Section 2, we give the background of the Stokes flow with elastic boundary. In Section 3, we summarize the elliptic solver proposed in [**11**]. We then modify the Stokes flow and rewrite the system to fit the elliptic solver. The details are given in Section 4. The numerical implementation details and some numerical experiments are presented in Sections 5 and 6, respectively, followed by a comparison with other methods and also a conclusion.

2. The Mathematical Model

2.1. The Stokes Flow. We solve the two dimensional model of Stokes flow given by

$$\nabla p = \nu \Delta \mathbf{u} + \mathbf{F}(\mathbf{x}, t), \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

where $\mathbf{u}$ is the velocity, p is the fluid pressure, ν is the fluid viscosity, and $\mathbf{F}$ is the boundary force such as an elastic force or surface tension. With the incompressibility condition, we can easily decouple the Stokes equation (1-2) into three Poisson equations:

$$\begin{aligned} \Delta p &= \nabla \cdot \mathbf{F}\,, \\ \Delta u &= p_x - F_1\,, \\ \Delta v &= p_y - F_2\,, \end{aligned}$$

where $\mathbf{F} = (F_1, F_2)$; F_1 and F_2 are the components of the force in x and y directions. Here, we assume that the fluid viscosity ν is constant. For convenience, we take $\nu = 1$ in this work.

The moving boundary is parameterized by Lagrangian variables $\mathbf{X}(s,t)$ which represent the points at t. The variable s is the arc-length parameter. Then we can write down the force distribution using the two dimensional Dirac function:

$$\mathbf{F}(\mathbf{x}, t) = \int_{\Gamma(t)} \mathbf{f}(s,t)\delta(\mathbf{x} - \mathbf{X}(s,t))ds\,.$$

We consider a moving interface problem which simulates the evolution of elastic membrane, where the force density $\mathbf{f}$ can be given by

$$\mathbf{f}(s,t) = \frac{\partial}{\partial s}(T(s,t)\tau(s,t))\,.$$

The restoring force of the stretched boundary is set as force for an elastic boundary. If s_0 is the arc-length measure along the unstretched boundary, then there is a continuous mapping between s and s_0 given by $s_0 = \psi(s)$. From the generalized Hooke's law, the tension $T(s,t)$ is therefore given by

$$T(s,t) = T_0\left(\left|\frac{\partial \mathbf{X}(s,t)}{\partial s_0}\right| - 1\right) = T_0\left(\left|\frac{\partial \mathbf{X}(s,t)}{\partial s}\right| / |\psi'(s)| - 1\right),$$

where T_0 is the tension coefficient which describes the properties of the elastic band. Here we assume it is uniform along the band. The larger the chosen T_0 is, the stiffer the elastic band and the larger force will be generated. The tangent of the interface is given by $\tau(s,t)$, where

$$\tau(s,t) = \frac{\partial \mathbf{X}(s,t)}{\partial s} \Big/ \left|\frac{\partial \mathbf{X}(s,t)}{\partial s}\right|.$$

Therefore the force density $\mathbf{f}$ can be obtained by

$$\mathbf{f}(s,t) = (\partial T/\partial s)\tau(s,t) + T\kappa\mathbf{n}, \tag{3}$$

where $\mathbf{n}$ is the normal direction and κ is the curvature, which is defined by $\partial\tau/\partial s = \kappa\mathbf{n}$. If the tension T linearly depends on $\left|\frac{\partial \mathbf{X}(s,t)}{\partial s_0}\right|$ (i.e. the tension depends on the arc-length linearly), we can simply obtain the following force-density model

$$\mathbf{f}(s,t) = \gamma\frac{\partial^2}{\partial s^2}\mathbf{X}(s,t), \tag{4}$$

where γ is the surface tension coefficient.

2.2. Jump Conditions Across the Interface. Because of the singular forces on the interface, the solution to the Stokes flow will be non-smooth or discontinuous. Applying the unit normal vector $\mathbf{n}$ and unit tangent vector τ on the force density $\mathbf{f}(s,t) = (f_1, f_2)$, we can obtain the force density $\mathbf{f}(s,t) = (\hat{f}_1, \hat{f}_2)$ on normal and tangential components.

$$\begin{aligned}\hat{f}_1(s,t) &= \mathbf{f}(s,t)\cdot\mathbf{n} = f_1(s,t)\cos(\theta) + f_2(s,t)\sin(\theta)\\ \hat{f}_2(s,t) &= \mathbf{f}(s,t)\cdot\tau = -f_1(s,t)\sin(\theta) + f_2(s,t)\cos(\theta),\end{aligned}$$

where θ is the angle between the x-axis and the outward normal direction. Both $\hat{f}_1$ and $\hat{f}_2$ can be related with the jump conditions for pressure and velocity as follows:

$$\begin{array}{ccc}[p](s) = \hat{f}_1(s,t), & [p_{\mathbf{n}}](s) = \frac{\partial \hat{f}_2}{\partial s}(s,t), & [p_\tau](s) = \frac{\partial \hat{f}_1}{\partial s}(s,t),\\ [u](s) = 0, & [u_{\mathbf{n}}](s) = \hat{f}_2(s,t)\sin(\theta), & [u_\tau](s) = 0,\\ [v](s) = 0, & [v_{\mathbf{n}}](s) = -\hat{f}_2(s,t)\cos(\theta), & [v_\tau](s) = 0.\end{array} \tag{5}$$

These jump conditions were derived in [**28**]. Here, we just use the same notation for clarity. The notation $[\cdot]$ is the jump denoting the difference of value outside and inside the interface Γ. Similar jump conditions were also used in [**32**] for solving Navier-Stokes equations with the moving interface problem.

3. A Weak Formulation of the Elliptic Solver

In order to solve the serial Poisson equations, we need to construct the Poisson solver which works with explicit or implicit interface representation. In this work, we follow the elliptic interface solver based on a weak formulation developed in [**11**]. We briefly discuss the method here for completeness and refer interested readers to the reference thereafter.

3.1. The Weak Formulation. Consider an open boundary domain $\Omega \subset \mathbb{R}^d$. Let Γ be the interface of co-dimension one, which divides the domain Ω into two parts, the inside domain Ω^- and the outside domain Ω^+. In other words, Ω can be separated into three parts $\Omega = \Omega^- \cup \Omega^+ \cup \Gamma$. Here, we assume that the boundary $\partial\Omega$ and the boundary of the subdomains $\partial\Omega^{\pm}$ are Lipschitz continuous. Then, we can also obtain Γ is Lipschitz continuous. A unit outward normal can be defined a.e. on Γ.

With these settings, we seek for the solution to the following elliptic equation with piecewise smooth variable coefficient

$$-\nabla \cdot (\beta(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \mathbf{x} \in \Omega \setminus \Gamma,$$

where $\mathbf{x} = (x_1, \cdots, x_d)$ denotes the spatial variables and ∇ is the spatial gradient operator. The coefficient $\beta(\mathbf{x})$ is assumed to be a matrix that is uniformly elliptic and continuously differentiable on each subdomain. However, it may be discontinuous across the interface Γ. The source term $f(\mathbf{x})$ is assumed to be in $L^2(\Omega)$.

The jump conditions in the solution and the flux across the interface Γ can be given by

$$[u]_\Gamma(\mathbf{x}) \equiv u^+(\mathbf{x}) - u^-(\mathbf{x}) = a(\mathbf{x}), \tag{6}$$

$$[(\beta\nabla u)\cdot\mathbf{n}]_\Gamma(\mathbf{x}) \equiv \mathbf{n}\cdot(\beta^+(\mathbf{x})\nabla u^+(\mathbf{x}) - \beta^-(\mathbf{x})u^-(\mathbf{x})) = b(\mathbf{x}), \tag{7}$$

where $a(\mathbf{x})$ and $b(\mathbf{x})$ are given along the interface Γ and "$\pm$" represents the limitations from the subdomains $\Omega^{\pm}$. Specifically, we use the Dirichlet boundary condition on the outer domain boundary

$$u(\mathbf{x}) = g(\mathbf{x}), \mathbf{x} \in \partial\Omega, \tag{8}$$

where g is a given function on the out boundary $\partial\Omega$.

The key idea of this weak formulation solver is to decompose the solution $u(\mathbf{x})$ into two parts:

$$u(\mathbf{x}) = u_r(\mathbf{x}) + u_s(\mathbf{x}),$$

where $u_r(\mathbf{x})$ is the regular part of solution, which is achieved without the jump conditions with first derivatives across the interface Γ, and $u_s(\mathbf{x})$ is the singular part which captures those jump conditions of $u(\mathbf{x})$ across the interface. The definition of $u_s(\mathbf{x})$ satisfies

$$\begin{aligned} u_s(\mathbf{x}) &= 0, \mathbf{x} \in \partial\Omega, \\ [u_s](\mathbf{x}) &= a(\mathbf{x}), \mathbf{x} \in \Gamma, \\ [\nabla u_s \cdot \beta \cdot \mathbf{n}](\mathbf{x}) &= b(\mathbf{x}) - [\nabla u_r \cdot \beta \cdot \mathbf{n}], \mathbf{x} \in \Gamma. \end{aligned} \tag{9}$$

Remark 3.1. *Generally, the regular part of decomposition $u_r(\mathbf{x})$ needs not to satisfy some homogeneous jump conditions. Instead, we implement it with $[u_r] = 0$ and $[\nabla u_r \cdot \mathbf{n}] = 0$, and then couple u_s and u_r by equation (9). If we can linearly construct $u_s(\mathbf{x}) \in H^1(\Omega^+) \cup H^1(\Omega^-)$ by $u_r(\mathbf{x})$ (i.e. $u_s = \mathcal{L}(u_r)$), we can write down a weak formulation to find $u_r(\mathbf{x}) \in H^1(\Omega)$, which satisfies the original elliptic problem*

$$\int_\Omega \nabla u_r \cdot \beta \cdot \nabla \phi d\mathbf{x} + \int_{\Omega^+\cup\Omega^-} \nabla \mathcal{L}(u_r) \cdot \beta \cdot \nabla \phi d\mathbf{x} = \int_\Omega f\phi d\mathbf{x} - \int_\Gamma b(\mathbf{x})\phi ds$$
$$u_r(\mathbf{x}) = g(\mathbf{x}), \mathbf{x} \in \partial\Omega,$$

for all test function $\phi(\mathbf{x}) \in H_0^1(\Omega)$. Then we can obtain the solution of original elliptic problem simply by $u = u_r + u_s$.

Remark 3.2. *If we have $\beta(\mathbf{x})$ being continuous across the interface, which means*

$$[\nabla u_r \cdot \beta \cdot \mathbf{n}]_\Gamma (\mathbf{x}) = 0\,,$$

the singular part u_s is independent with regular part u_r and the above weak formulation can be reduced to a simper form, by finding the $u_r(\mathbf{x}) \in H^1(\Omega)$, which satisfies the original elliptic problem as follows

$$\int_\Omega \nabla u_r \cdot \beta \cdot \nabla \phi d\mathbf{x} + \int_{\Omega^+\cup\Omega^-} \nabla u_s \cdot \beta \cdot \nabla \phi d\mathbf{x} = \int_\Omega f\phi d\mathbf{x} - \int_\Gamma b(\mathbf{x})\phi ds$$
$$u_r(\mathbf{x}) = g(\mathbf{x}), \mathbf{x} \in \partial\Omega,$$

for all test function $\phi(\mathbf{x}) \in H_0^1(\Omega)$.

3.2. Constructing the Jump Function. Consider a triangle intersecting with the interface, as shown in Figure 1. Let $\phi_A(\mathbf{x})$, $\phi_B(\mathbf{x})$, $\phi_C(\mathbf{x})$ be the linear base functions defined on the triangle. The singular part u_s^h is constructed as linear function separately in $\triangle ADE$ and $BCED$ with condition $u_s^h(\mathbf{x}) = 0$ on A, B and C, which means it will be in the form as

$$u_s^h = \begin{cases} c_2\phi_B(\mathbf{x}) + c_3\phi_C(\mathbf{x}), & \mathbf{x} \in \triangle AED \\ c_1\phi_A(\mathbf{x}), & \mathbf{x} \in BCED \end{cases}. \tag{10}$$

We enforce the jump in u at the two end points D and E, and the flux jump at the middle point M. This leads to a set of equations on c_1, c_2 and c_3 given by

$$\begin{bmatrix} -\phi_A(D) & 1-\phi_A(D) & 0 \\ -\phi_A(E) & 0 & 1-\phi_A(E) \\ -\mathbf{n}\cdot\beta^-\cdot\nabla\phi_A & \mathbf{n}\cdot\beta^+\cdot\nabla\phi_B & \mathbf{n}\cdot\beta^+\cdot\nabla\phi_C \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} a(D) \\ a(E) \\ \tilde{b}(M) \end{bmatrix}$$

Denote the coefficient matrix to be P. We have the following theorems.

Theorem 3.1 ([**11**])**.** *If (i) β is a symmetric positive definite matrix and has no jump across the interface, or (ii) $\beta > 0$ is a scalar and the triangle is non-obtuse, then the coefficient matrix P is non-singular.*

Theorem 3.2 ([**11**])**.** *The constructed piecewise linear function $u_r + u_s$ with proper extension (defined above) approximates the piecewise smooth function w to the second order, i.e., $\forall \mathbf{x} \in \triangle ABC$, $|\psi^\pm(\mathbf{x}) - u^\pm| = O(h^2)$ where h denotes the size of the non-obtuse and shape regular triangle $\triangle ABC$.*

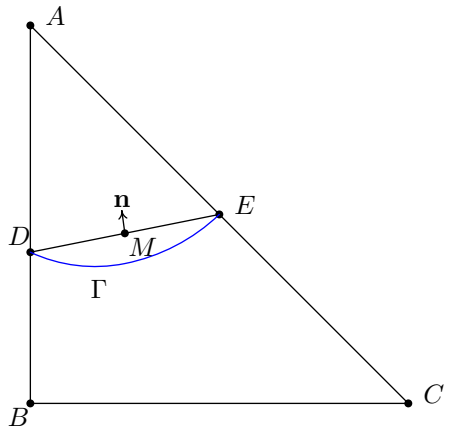


Figure 1. Interface triangle. $\overline{DE}$ approximates the interface in the triangle. M is the middle point of $\overline{DE}$, and $\mathbf{n}$ is the unit normal to $\overline{DE}$.

Here we only introduce the most general case as discussed in [**13**, **9**, **11**]. There are indeed some special cases, in which interface cut the triangle in different ways. For example, [**54**] has discussed all the possible situations which have not been listed here.

4. Coupling the Elliptic Solver to the Stokes Flow

Here, we first rewrite the Poisson equations in the Stokes flow into weak formulations that fit the elliptic solver developed in [**11**].

4.1. Variational Forms for the Pressure Poisson Equation. The Poisson equation for the pressure is

$$\Delta p = \nabla \cdot \mathbf{F}\,.$$

We will solve the Poisson equation in constant media ($\beta(\mathbf{x}) = 1$). By using the formula introduced in the last section, we can obtain the variational form of Pressure Poisson equation as follows:

$$\int_\Omega \nabla p_r \cdot \nabla \phi d\mathbf{x} = -\int_{\Omega^+\cup\Omega^-} \nabla p_s \cdot \nabla \phi d\mathbf{x} + \int_\Gamma \frac{\partial \hat{f}_2}{\partial \mathbf{n}} ds\,, \tag{11}$$

where p_r and p_s represent the regular part and singular part of pressure. The function $\hat{f}_2$ is the force density in tangent direction, which is given when interface is fixed.

4.2. A Variational Form for the Velocity Poisson Equations.

4.2.1. A General Approach to the Velocity Poisson Equations. Once the pressure is calculated from (11), we can obtain the velocities of x and y components through the two elliptic equations. In order to solve the Velocity Poisson equations, we need to obtain p_x and p_y for all grid points using the previous pressure information p_{ij} on the grids. However, it is not an easy job

due to the discontinuity of pressure. The work in [**29**] has proposed a local interpolation method to get p_x and p_y on the grids. The basic idea is as follows: at a regular grid point, the derivative of the pressure is computed using simple central differences. At an irregular mesh point, on the other hand, we apply a finite difference formula based on only data points from the same side of the interface. In case when both neighbors are from the opposite side, we apply an interpolation formula as follows:

$$(p_x^{\pm})_{ij} = \frac{p_{ij} - p_{lj} \mp [p] \mp [p_x](x_l - X_k^*) \mp [p_y](y_j - Y_k^*)}{(x_i - x_l)}. \tag{12}$$

The sign in the equation depends on the side of the interface where (x_i, y_j) is located. The point (X_k^*, Y_k^*) is the control point closest to the target grid point (x_i, y_j), and x_l is one of the neighbors closer to X_k^*. The jump conditions $[p]$, $[p_x]$ and $[p_y]$ can be obtained from the calculations in Section 2.2.

By applying Taylor expression to (12), it is easy to prove that the interpolation designed above is of first order accuracy in irregular meshes. However, for Finite Element methods, when we derive the variational form for weak solution, we need to calculate the integrals of p_x and p_y inside the meshes. For regular meshes, it is easy to extend. When integrating the irregular meshes, not only the values on grid points are needed, but also the values on intersections between interface and grid lines are necessary. One way to get these values is to use the piece-wise linear or high order nonlinear interpolation with jump conditions as constraints, yet the accuracy may be lost and the calculation time in each time steps will increase. The method we want to propose in the following is a simple modification which fixes the problem by using a mesh integration coupled with jump conditions without an explicit dependence on p_x and p_y.

4.2.2. A Modified Form to the Velocity Poisson Equations of the Stokes Flow. The Poisson equation for the velocity in the x component is given by

$$\Delta u = \frac{\partial p}{\partial x} - F_1 \,.$$

However, since F_1 is defined only on the interface (i.e. in Ω^+ and Ω^-), we can calculate the Poisson equations separately without adding the extra force F_1,

$$\int_{\Omega^+\cup\Omega^-} \Delta u \phi d\mathbf{x} - \int_{\Omega^+\cup\Omega^-} \frac{\partial p}{\partial x}\phi d\mathbf{x} = 0 \,. \tag{13}$$

In each domain, Ω^+ and Ω^-, we apply integration by parts which leads to

$$\begin{aligned} \int_{\Omega^+} \Delta u\phi d\mathbf{x} &= -\int_\Gamma (\nabla u^+ \cdot \mathbf{n}) ds - \int_{\Omega^+} \nabla u \cdot \phi d\mathbf{x} \\ \int_{\Omega^-} \Delta u\phi d\mathbf{x} &= \int_\Gamma (\nabla u^- \cdot \mathbf{n}) ds - \int_{\Omega^-} \nabla u \cdot \phi d\mathbf{x} \,, \end{aligned}$$

where $\mathbf{n} = [\cos\theta, \sin\theta]$ and

$$\begin{aligned} \int_{\Omega^+} \frac{\partial p}{\partial x}\phi d\mathbf{x} &= -\int_\Gamma ([p^+, 0]^T \cdot \mathbf{n})\phi ds + \int_{\Omega^+} \left[\frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}\right] \cdot [p, 0]^T d\mathbf{x} \\ \int_{\Omega^-} \frac{\partial p}{\partial x}\phi d\mathbf{x} &= \int_\Gamma ([p^-, 0]^T \cdot \mathbf{n})\phi ds + \int_{\Omega^-} \left[\frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}\right] \cdot [p, 0]^T d\mathbf{x} \,. \end{aligned}$$

Adding these terms together, Eq.(13) can be rewritten to the following equation with jump conditions,

$$\int_{\Omega^+\cup\Omega^-}(-\nabla u\cdot\nabla\phi+p\frac{\partial\phi}{\partial x})d\mathbf{x}+\int_\Gamma[p]\cos(\theta)\phi ds-\int_\Gamma\left[\frac{\partial u}{\partial\mathbf{n}}\right]\phi ds=0\,.$$

By the jump conditions

$$\left[\frac{\partial u}{\partial\mathbf{n}}\right]=[p]\cos(\theta)-f_1=\hat{f}_2\sin(\theta)\,,$$

we finally obtain the following new weak formulation for Velocity Poisson equation without p_x

$$\int_\Omega\nabla u_r\cdot\nabla\phi d\mathbf{x}=-\int_{\Omega^+\cup\Omega^-}\nabla u_s\cdot\nabla\phi d\mathbf{x}+\int_\Omega p\frac{\partial\phi}{\partial x}d\mathbf{x}+\int_\Gamma f_1\phi ds\,.$$

Similarly, for the y component, we have

$$\int_\Omega\nabla v_r\cdot\nabla\phi d\mathbf{x}=-\int_{\Omega^+\cup\Omega^-}\nabla v_s\cdot\nabla\phi d\mathbf{x}+\int_\Omega p\frac{\partial\phi}{\partial y}d\mathbf{x}+\int_\Gamma f_2\phi ds\,,$$

where f_1 and f_2 is the force density in x and y component and $p=p_r+p_s$ can be obtained by Eq.(11). In this work, we use regular triangles and therefore $\frac{\partial\phi}{\partial x}$ and $\frac{\partial\phi}{\partial y}$ can be easily obtained.

5. Numerical Algorithm

The numerical methods to simulate the interface Stokes interface problem can be summarized as follows. All detailed descriptions of each step will be given in the corresponding subsections.

1) (Section 5.1) Collect the control points or the foot points. Then calculate the intersection points between interface and triangle elements. Generate a periodic cubic spline by the control points $\{X_k^n, Y_k^n\}$ to compute the forces and jump conditions at the intersection points.
2) (Section 5.2) Calculate the pressure and velocities on the grids by solving the three Poisson equations with weak formulation finite element method.
3) (Section 5.3) Interpolate the velocities on the grids to get the velocities on the control points.
4) (Section 5.4) Track the interface by updating the control points.

5.1. Interface Representation. At any time step t_n, the interface is represented by a given finite set of control points or foot points. Many methods can be used to represent the interface, such as the spline, the grid-based particle method (GBPM) [**24**, **25**, **23**], the Cell-based particle method (CBPM) [**8**]. In this work, however, we consider only a set of points which is represented by periodic cubic spline. The Lagrangian points $\{X_k^n, Y_k^n\}$, for $k=1,2,\cdots,(N_b-1)$ represent the moving interface at t_n.

We use a periodic cubic spline $\mathbf{X}(s,t)=(X(s,t),Y(s,t))$ which interpolates all control points to express the moving interface at $t_n=n\Delta t$, where s is the arclength parameter of interface. Because of the cubic spline representation, we

can easily calculate the force acting along the interface. First, we can compute $\partial \mathbf{X}/\partial s$ at every control points. Since the relationship between control points and the corresponding points on the unstretched interface is given, we can easily obtain the surface tension $T(s, t^n)$. By multiplying the tension with the tangent vector, we have the tangent force $w(s, t^n) = T(s, t^n)\tau(s, t^n)$. As mentioned in [**28**], we do not differentiate $w(s, t^n)$ to get the force density along the interface directly, but use its value at control points to generate a new periodic cubic spline and get the first derivative.

5.2. Discretization of the Weak Formulation. We define a cell K as an interface triangle if two of vertexes belong to different subdomains and the interface across such a triangle K. In the interface cell, we denote $K = K^+ \cup K^-$. K^+ and K^- are separated by a straight line segment Γ_K which approximate the interface Γ. Vertexes of K^+ are located in $\Omega^+ \cup \Gamma$ and vertexes of K^- are located in $\Omega^- \cup \Gamma$ similarly. We use K^+ and K^- to approximate the regions $K \cap \Omega^+$ and $K \cap \Omega^-$. $|K^+|$ and $|K^-|$ represent the area of K^+ and K^-.

One useful approximation in the finite element implementation to the weak formulation is the integration of a function inside the interface cell. For an interface triangle, we approximate $\int_K f\phi d\mathbf{x}$ by

$$\int_{K^+} f\phi d\mathbf{x} + \int_{K^-} f\phi d\mathbf{x}\,.$$

To evaluate $\int_{K^+} f\phi d\mathbf{x}$ in interface triangle, we first cut K^+ into two triangles if K^+ is not a triangle. Then on each triangle, we use a second order accurate numerical quadrature. Evaluation of $\int_{K^-} f\phi d\mathbf{x}$ can be done similarly. In [**10**], a second order accurate midpoint rule was used to evaluate integrals on interface triangles by

1) cutting the non-triangle part into two triangles (which gives 3 triangles in total, $\triangle\, p_1p_4p_5$, $\triangle\, p_4p_2p_5$ and $\triangle\, p_5p_2p_3$ as shown in Figure 2);
2) getting the midpoints of each triangle (e.g. for $\triangle\, p_1p_4p_5$, we have m_1, m_2, and m_3);
3) calculating the integration with basis function separately by

$$\frac{f(m_1)\phi(m_1) + f(m_2)\phi(m_2) + f(m_3)\phi(m_3)}{3} \times \text{area}\,.$$

For the static case, the function f is given in the calculation domain and we can get $f(p)$ directly. For general moving interface problem, however, we only have values of f on grid points. So, in this case we should interpolate the value of f separately with the jump condition carefully incorporated if f is discontinuous.

In particular, we approximate $f(p)$ using a piecewise linear function along the side (intersected by the interface) of an interface triangle and incorporating the interface conditions at the intersection. We may use a high order approximation of $f(p)$ at interface triangles, which will involve more neighboring vertices, and incorporate jump conditions at the intersection.

Other integrations in an interface triangle are

$$\int_K p\frac{\partial \phi}{\partial x} d\mathbf{x} \text{ and } \int_K p\frac{\partial \phi}{\partial y} d\mathbf{x}$$

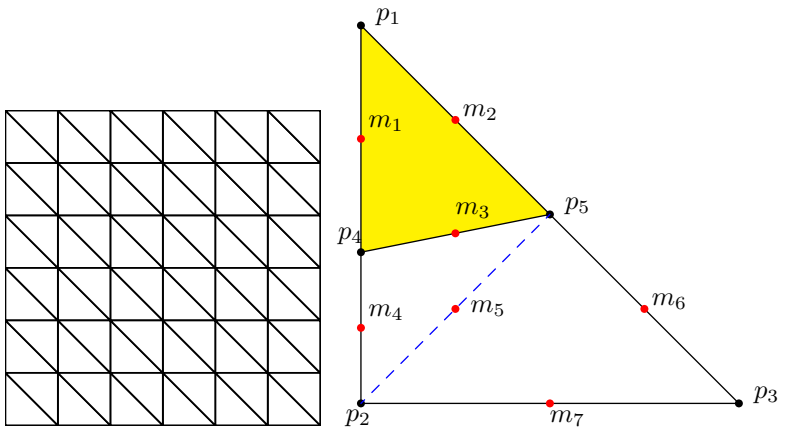


Figure 2. An illustration of a simple uniform triangulation (Left) and the partition of an interface triangle (Right). Right: The line segment p_4p_5 is the approximation of interface. The points $m_1, m_2, \cdots, m_7$ denote the middle points of any segments. The segment p_2p_5 is a virtual cut of the non-triangular part.

when solving the velocity Poisson equations. With a uniform triangulation, the terms $\frac{\partial \phi}{\partial x}$ and $\frac{\partial \phi}{\partial y}$ are constant. Therefore, we only need to give an approximation to the the average value of p inside the interface cell. For the regular part of p, $\int_K p_r d\mathbf{x}$ can be represented as the average of p on three vertexes. For the singular part, we can get $\int_K p_s d\mathbf{x}$ by the linear combination of the base function in (10), i.e.

$$\begin{aligned}\int_K p_s d\mathbf{x} \;=\; & C_{p_1}\left[\frac{\text{area}(\triangle\, p_1p_2p_3)}{3} - (1+\phi_{p_1}(p_4)+\phi_{p_1}(p5))\frac{\text{area}(\triangle\, p_1p_4p_5)}{3}\right] \\ & + \left[C_{p_2}(\phi_{p_2}(p_4)+\phi_{p_2}(p5)) + C_{p_3}(\phi_{p_3}(p_4)+\phi_{p_3}(p5))\right]\frac{\text{area}(\triangle\, p_1p_4p_5)}{3}.\end{aligned}$$

Our method can be easily extended to a general triangular mesh. In Example 6.2, we apply our method to a triangular mesh with adaptivity near the interface. The discretized Poisson equation is solved by FFT [**47**, **53**] on rectangular grids and by the algebraic multi-grid method [**45**, **41**, **42**] on a triangular mesh.

5.3. Interpolating the Velocity at the Control Points. Since we have the velocities in the x- and the y-components on the grids, we interpolate them from grid points to interface points. In other words, we need to find U_k and V_k at all control points (X_k, Y_k). However, since the velocity is not smooth across the interface, we need to carefully use the interface jump conditions to fix the interpolation procedure.

Because of the proposed weak formulation, a simple way to interpolate the velocities on boundary is to reuse the coefficients from regular-singular splitting of the solution. In particular, taking a typical point $\mathbf{x}_k$ on the interface as an example, we describe the way to interpolate the grid values U to obtain the

value U_k at $\mathbf{x} = \mathbf{x}_k$. The first step is to determine the corresponding triangle containing the location $\mathbf{x}_k$. If we use the GBPM/CBPM, such triangle is known automatically since the control point is the representative of a certain cell. Otherwise, a searching method has to be implemented. Let $\phi_A(\mathbf{x})$, $\phi_B(\mathbf{x})$, and $\phi_C(\mathbf{x})$ be the linear basis functions defined on the triangle. The function at the control point $\mathbf{x}_k$ can then be computed by

$$U_k = u_r^h(\mathbf{x}_k) + u_s^h(\mathbf{x}_k)$$

with the regular part u_r^h constructed using the linear combination of three vertexes as in the general finite element method, i.e.

$$u_r^h(\mathbf{x}) = \phi_A(\mathbf{x})U_A + \phi_B(\mathbf{x})U_B + \phi_C(\mathbf{x})U_C\,,$$

while the singular part u_s^h is constructed as a linear function separately using Eq.(10).

Other interpolation methods might also be possible. For example, [**22**] has proposed a slightly more tedious bilinear interpolation method incorporating the jump condition near the interface. The approach is more natural for rectangular mesh. Nevertheless, we have also implemented that approach and found that the solutions obtained are of similar accuracy.

5.4. Updating the Interface. To evolve the interface, the simplest method is the forward Euler scheme where the control points are updated explicitly by

$$\begin{aligned} X_k^{n+1} &= X_k^n + \Delta t\, U_k^n \\ Y_k^{n+1} &= Y_k^n + \Delta t\, V_k^n\,, \end{aligned}$$

where U_k^n and V_k^n represent the local velocity in the x- and the y- components interpolated by the grid values as discussed in the previous section. However, because of the stability condition, the method requires a very small time-step. To relax such CFL condition, we recommend the following implicit Trapezoidal approach by solving

$$\mathbf{X}^{n+1} = \mathbf{X}^n + \frac{\Delta t}{2}\left[\mathbf{U}^n(\mathbf{X}^n) + \mathbf{U}^{n+1}(\mathbf{X}^{n+1})\right]\,, \tag{14}$$

where $\mathbf{U}^n(\mathbf{X}^n)$ defines the velocity of the interface at $\mathbf{X}^n$ and time t^n. The next position $\mathbf{X}^{n+1}$ is determined by solving the following system of nonlinear equations

$$g(\mathbf{X}) = \mathbf{X} - \mathbf{X}^n - \frac{\Delta t}{2}\left[\mathbf{U}^n(\mathbf{X}^n) + \mathbf{U}(\mathbf{X})\right]\,,$$

which can be solved using the Broyden-Fletcher-Goldfrab-Shanno (BFGS) method as described in [**7**, **44**]. The BFGS method is a quasi-Newton method used to solve the nonlinear system (14) iteratively to calculate the location of the moving interface. Within the iteration, it needs to solve the linear system three times for the singular force $\mathbf{f}$ to impose the prescribed velocity at the fixed interface at t^n to ensure the interface condition for the velocity is satisfied. This is necessary because the velocity field and pressure field are updated at every iterations of the BFGS method. We find that in most of the following numerical examples, we require only 2 to 3 iterations in each time-step since $\mathbf{X}^n$ provides a good initial guess for $\mathbf{X}^{n+1}$.

6. Numerical Experiments

6.1. Relaxation of an Elastic Elliptic Membrane. This example was first introduced in [**53**] for testing the performance of the IBM and was also used in testing the behavior of the IIM. The initial interface is an ellipse with the semi-major and the semi-minor axes given by $a = 0.75$ and $b = 0.5$, respectively. The ellipse is located at the center of the computational domain initially. The unstretched state of the membrane is a circle with radius $r_0 = 0.5$. Due to the restoring force on interface, the ellipse will converge to an equilibrium state given by a circle with radius $r = \sqrt{ab}$ which is larger than the unstretched circle because of the incompressibility of the inner interface fluid. The setup of this example is shown in Figure 3.

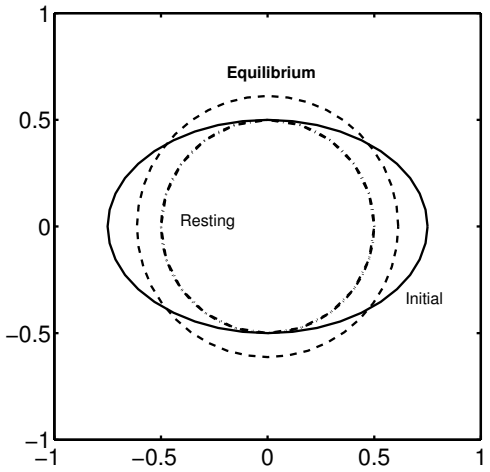


Figure 3. (Example 6.1) The interface at different states. The initial interface (Solid line) is given by an ellipse with $a = 0.75$ and $b = 0.5$. The equilibrium position (dashed line): the circle with $r = \sqrt{ab} \approx 0.6123$. The resting circle (dash-dot line): the circle with $r_0 = 0.5$.

Figure 4 shows the solutions at the initial time $t = 0$ before we advect the interface. To check the accuracy of the elliptic solver for the Stokes flow, we compare the solutions with the fine mesh solution. Table 1 shows the error in the computed pressure and the velocity at the initial time. We compute the solution using the fine mesh ($N = 320$) and regard it as the *exact* solution. The ratios in the errors are approximately 4, implying that the solutions are approximately second order accurate.

During the interface evolution, we compute the longest ($r_{\max}$) and the shortest ($r_{\min}$) distances from the control points to the origin of the computational domain at each time step and regard the solution as the equilibrium solution when $|r_{\max} - r_{\min}| < 10^{-5}$. Figure 5 shows the plot of $r_{\max}$ and $r_{\min}$ as a function of time. In the steady state when $r_{\max} \simeq r_{\min}$, the interface relaxes

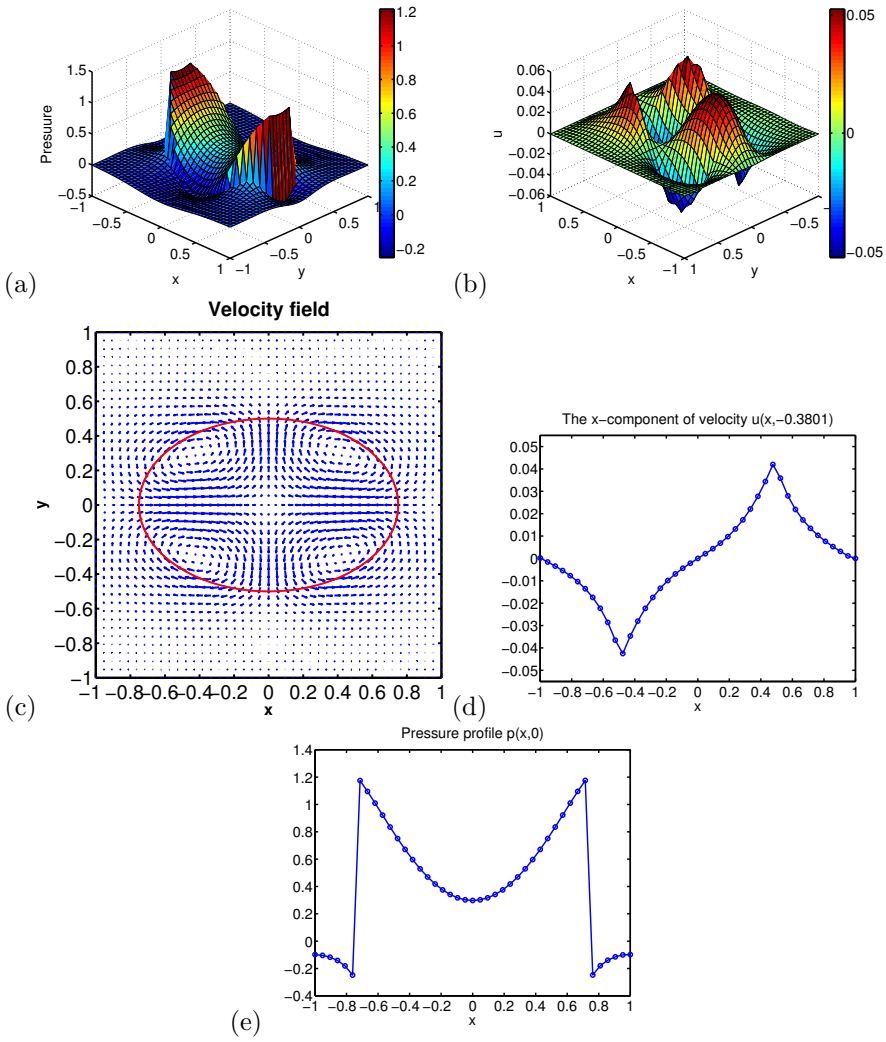


Figure 4. (Example 6.1) The computed solution at the initial time $t = 0$. (a) the pressure, (b) the velocity u, (c) the velocity field, (d) the velocity u along $y = -0.3801$, and (e) the pressure along $y = 0$.

N	$\|p_N - p_{320}\|_\infty$	order	$\|u_N - u_{320}\|_\infty$	order	$\|v_N - v_{320}\|_\infty$	order
40	1.9191e-03		7.2564e-04		3.9839e-04	
80	6.9554e-04	1.3796	1.7220e-04	2.1070	9.2078e-05	2.1633
160	1.6169e-04	2.151	4.2716e-05	2.0157	3.0093e-05	1.5299

Table 1. (Example 6.1) Errors in the pressure (p) and velocity (u and v) at the initial time $t = 0$ on the grid mesh.

to a circle with the pressure given by a piecewise constant function, as shown in Figure 6. In this equilibrium state, the flow is steady and so both u and v are zero. Table 2 shows the velocities at the steady state computed using different sets of mesh. Finally, we show in Figure 7 the convergence rate of $r_{\max}$ at $t = 0.5$. The rate is roughly 2.

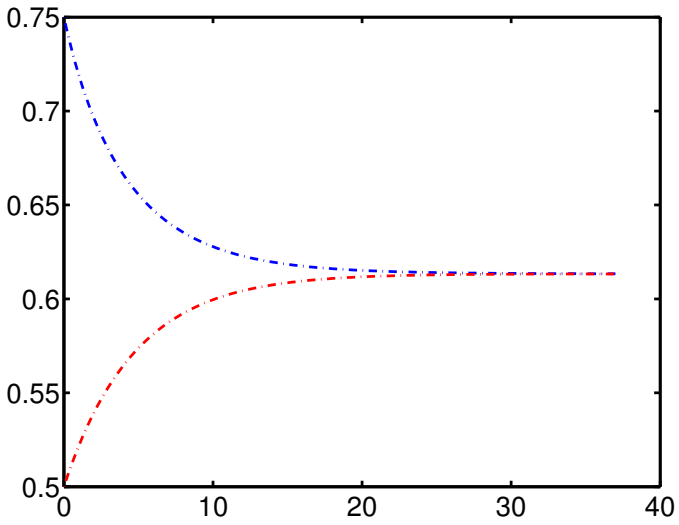
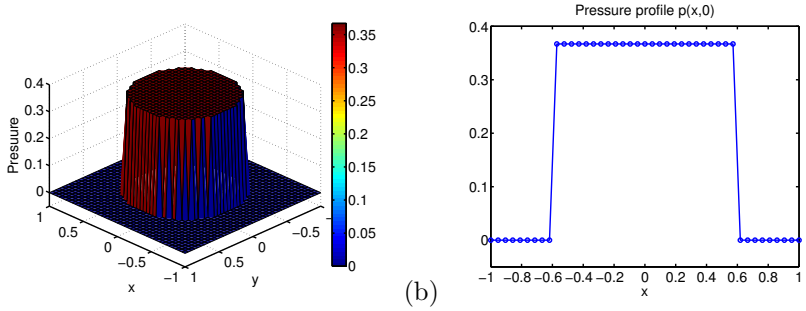


Figure 5. (Example 6.1) The longest ($r_{\max}$) and shortest ($r_{\min}$) distance of control points from the origin as a function of time t with $N = 161$. The blue line is the longest distance, while the red one is the shortest distance from the origin.



(a) (b)

Figure 6. (Example 6.1) The computed solution at time $t = 10$. (a) The pressure contour and (b) the pressure along $y = 0$.

6.2. Nonuniform Grid with Relaxation Elastic Membranes. Because of the simplicity of the elliptic solver, it is easy to implement on an adaptive

N	$\|E_u\|_\infty$	order	$\|E_v\|_\infty$	order
42	8.7465e-05		8.7465e-05	
82	2.4877e-05	1.8008	2.4877e-05	1.8008
162	6.9340e-06	1.8160	6.9340e-06	1.8160
322	1.6382e-06	2.1295	1.6382e-06	2.1295

Table 2. (Example 6.1) Errors in the steady state velocity (u, v). The initial interface is a circle with radius $r = \sqrt{ab}$ on the control points. The exact solution for this case is $u = v = 0$.

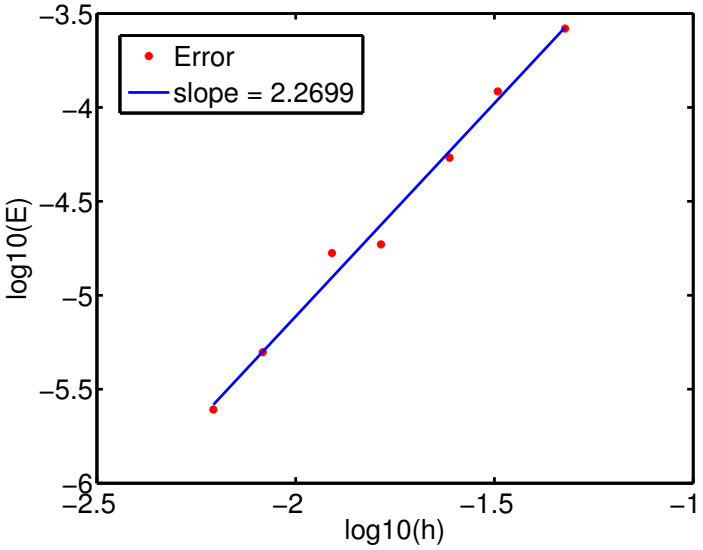


Figure 7. (Example 6.1) The convergence rate for $r_{\max}$ at $t = 0.5$. The slope is 2.2699.

mesh too. In this example, we repeat the previous example in the same calculation domain while replace the computational mesh by an adaptive triangular grid. Using a similar setting as in Example 6.1, the initial state of this simulation is an ellipse with the semi-major and semi-minor axes $a = 0.75$, $b = 0.5$. The tension coefficient T_0 is set to 1 in this case. The computational domain is $[-2, 2] \times [-2, 2]$. In the initial state, we refine the triangle mesh near the interface. Figure 8 shows the mesh size in different scales.

Here we use the algebraic multigrid to solve the linear system instead of FFT due to the nonuniform grid. Figure 9 shows the solutions at the initial time $t = 0$ before we advect the interface. Compared with the solution in Figure 4, we found that the two results have very similar pattern in both the pressure and the x-component velocity.

6.3. Relaxation of an Elastic Seven-folded Membrane. This example considers a more complicated interface given by $\rho = 0.6+0.2 \sin 7\theta$ representing

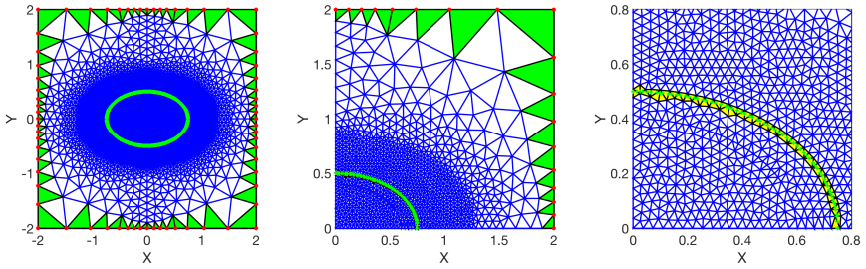


Figure 8. (Example 6.2) Meshes for initial state with different scales.

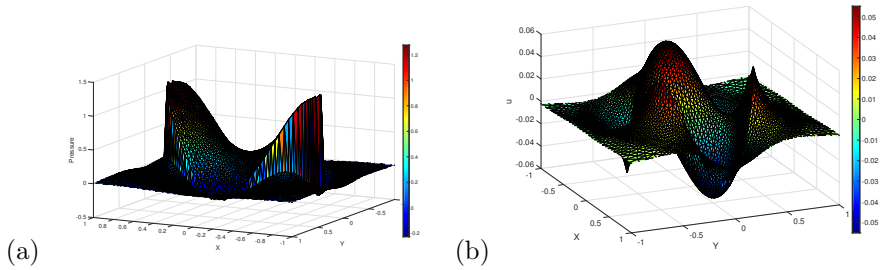


Figure 9. (Example 6.2) The computed solution at the initial time $t = 0$ with nonuniform grid. (a) The pressure, and (b) the velocity u.

a seven-folded shape with the unstretched interface given by a circle of radius $r_0 = 0.3$. The evolution of the membrane is shown in Figure 10. Figure 11 shows the velocity field and the pressure contours at the intermediate time $t = 0.2$.

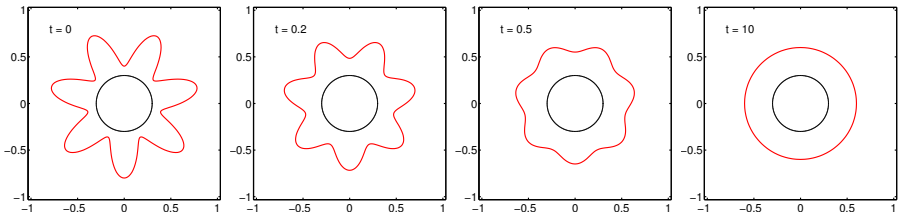


Figure 10. (Example 6.3) Dynamic of interface for the seven-folded shape relaxation. The red solid line represents the interface while the black solid line represents the interface at (a) $t = 0$, (b) $t = 0.2$, (c) $t = 0.5$, and (d) $t = 10$.

6.4. Relaxation of Multiple Elastic Membranes. In this example, we study the motion of multiple interfaces under the Stokes flow in a connected domains. There are four ellipses located in symmetric places. The initial state of this simulation is an ellipse with the semi-major and semi-minor axes

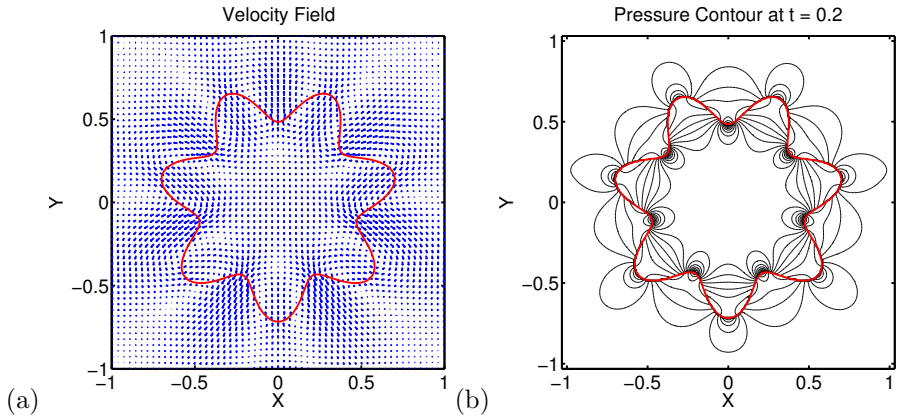


Figure 11. (Example 6.3) (a) The velocity field and (b) the pressure contour at $t = 0.2$.

$a = 0.5$, $b = 0.3$. The center of upper right ellipse is located at $(0.7, 0.7)$ and other ellipses are located symmetrically with two diagonal straight line $y = x$ and $y = -x$. All tension coefficient T_0 are set to 1 in this case. We select a domain $[-1.5, 1.5] \times [-1.5, 1.5]$ with 80×80 grid in the whole calculation domain. In Figure 12, we show the velocity field of multi-interface relaxation at initial time and pressure distributions from $t = 0$ to $t = 10$. When $t = 10$, the pressure distribution nearly reaches the equilibrium state. The pressure in Figure 12 (d) is nearly uniformly distributed. From Figure 12 (a), we can see the velocity field for Multi-interfaces relaxation problem is well captured. From Figure 12 (b)-(c), we can see the sharp pressure profiles along the moving interfaces.

6.5. Relaxation of an Elastic Membrane with Discontinuous Diffusion Coefficients. In this example, we modify the previous example by considering the relaxation of an elastic membrane with with discontinuous diffusion coefficients across the interface. We follow a similar setting as in Example 6.1 so that the initial state of this simulation is an ellipse with the semi-major and semi-minor axes given by $a = 0.75$ and $b = 0.5$, respectively. The tension coefficient T_0 is set to be 1 in this case, while

$$\nabla p = \nu(\nabla \cdot (\beta(\mathbf{x})\nabla \mathbf{u})) + \mathbf{F}(\mathbf{x}, t)\,,$$
$$\nabla \cdot \mathbf{u} = 0\,.$$

with $\beta(\mathbf{x})$ is a discontinuous function across the membrane and is given by

$$\beta(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in \Omega^- \\ \rho, & \mathbf{x} \in \Omega^+ \end{cases}$$

for some constant ρ.

Figure 13 shows the evolutions of the largest and the shortest distances from the origin along the elliptic membrane under different value of ρ. For ρ equals one, the setup reduces back to the original one in Example 6.1. We notice that as one increases the value of the diffusion coefficient of the interior fluid, it takes a longer time for the membrane to return to its equilibrium state.

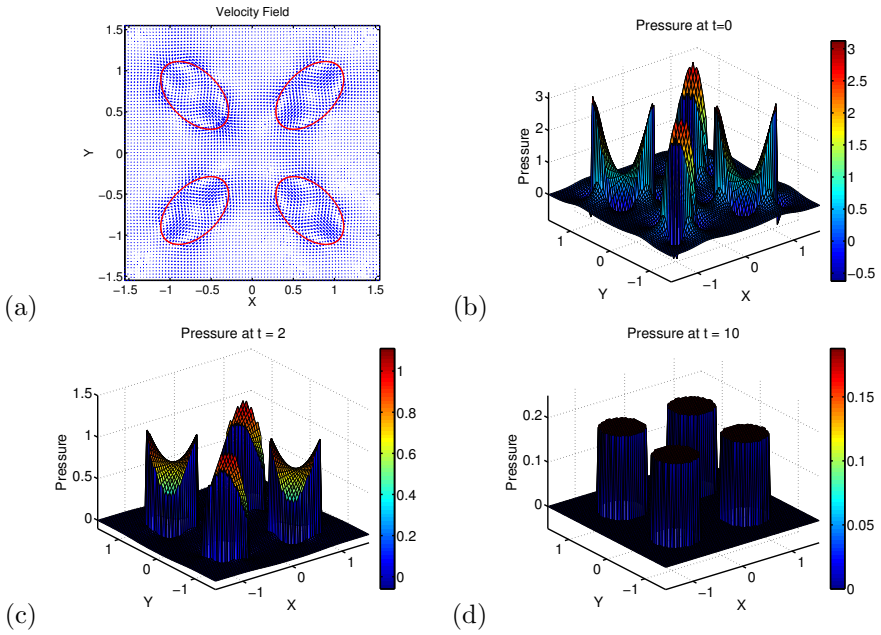


Figure 12. (Example 6.4) Multiple membranes under the Stokes flow. (a) The velocity field of four ellipses at the initial time $t = 0$. (b) The pressure in the computational domain at the initial time $t = 0$. (c) The pressure in the computational domain at $t = 2$. (d) The pressure in the computational domain at $t = 10$.

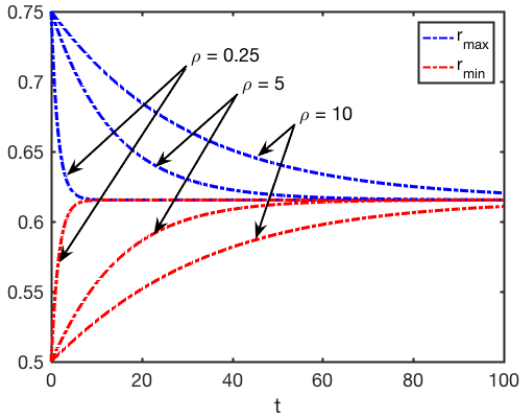


Figure 13. (Example 6.5) Convergence to steady state with different ρ with $N = 41$. The blue line is the longest distance, while the red one is the shortest distance from the origin.

7. Conclusion

We develop an efficient Stoke flow solver based on a weak formulation. Compared with the finite difference method using the IIM in computing the flow velocity on the interface, the proposed method does not require interpolating the derivatives of the pressure p_x and p_x at irregular grids with jump conditions. Because of the weak formulation, our method is easier to implement and is more accurate in evaluating the correction terms near the interface.

While the periodic cubic spline is used to represent the interface in the current work, some other interface representation methods such as the level set method [**36**], the GBPM [**24**, **25**, **23**] or the CBPM [**8**] can also be applied which might provide a more robust handling of topological changes or a better numerical accuracy. Although we consider only two dimensional flows for the moment, the weak formulation elliptic interface solver can be well-extended to three dimensional cases with a possible challenge in incorporating a good representation of a moving interface.

Acknowledgments. S. Hou's research is partially supported by the Walter Koss Endowed Professorship. This professorship is made available through the State of Louisiana Board of Regents Support Funds. The work of Leung was supported in part by the Hong Kong RGC grants 16303114 and 16309316. Research of Zhao is partially supported by NSF grant DMS-1418422.

References

[1] L. Adams and Z.L. Li. The immersed interface/multigrid methods for interface problems. *SIAM J. Sci. Comput.*, 24:463–479, 2002.

[2] J.W. Cahn and J.E. Hilliard. free energy of a nonuniform system. i. interfacial free energy. *J. Chem. Phys.*, 28:258–267, 1958.

[3] K.Y. Chen, K.A. Feng, Y. Kim, and M.C. Lai. A note on pressure accuracy in immersed boundary method for Stokes flow. *J. Comput. Phys.*, 230:4377–4383, 2011.

[4] R. Cortez and M. Minion. The blob projection method for immersed boundary problems. *J. Comput. Phys.*, 161:428–453, 2000.

[5] R.P. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows. *J. Comput. Phys.*, 79:12–49, 1999.

[6] A.L. Fogelson and J.P. Keener. Immersed interface methods for Neumann and related problems in two and three dimensions. *SIAM J. Sci. Comput.*, 22:1630–1654, 2000.

[7] P.E. Gill, W. Murray, and M.H. Wright. *Practical optimization*. Academic Press, 1981.

[8] S. Hon, S. Leung, and H.K. Zhao. A cell based particle method for modeling dynamic interfaces. *J. Comp. Phys.*, 272:279–306, 2014.

[9] S.M. Hou, Z.L. Li, L.Q. Wang, and W. Wang. A numerical method for solving elasticity equations with interfaces. *Comm. Comput. Phys.*, pages 595–612, 2012.

[10] S.M. Hou and X.D. Liu. A numerical method for solving variable coefficient elliptic equation with interfaces. *J. Comp. Phys.*, 202:411–445, 2005.

[11] S.M. Hou, P. Song, L.Q. Wang, and H.K. Zhao. A weak formulation for solving elliptic interface problems without body fitted grid. *J. Comp. Phys.*, 249(C):80–95, 2013.

[12] S.M. Hou, L.Q. Wang, and W. Wang. A Numerical Method for solving the Elliptic Interface Problem with Multi-Domains and Triple Junction Points. *J. Comp. Math.*, 30(5):504–516, 2012.

[13] S.M. Hou, W. Wang, and L.Q. Wang. Numerical method for solving matrix coefficient elliptic equation with sharp-edged interface. *J. Comp. Phys.*, 229:7162–7179, 2010.

[14] W.F. Hu, Y. Kim, and M.C. Lai. An immersed boundary method for simulating the dynamics of three-dimensional axisymmetric vesicles in Navier-Stokes flows. *J. Comput. Phys.*, 257:670–686, 2014.

[15] W.F. Hu, M.C. Lai, Y. Seol, and Y.N. Young. Vesicle electrohydrodynamic simulations by coupling immersed boundary and immersed interface method. *J. Comput. Phys.*, 317:66–81, 2016.

[16] W.F. Hu, M.C. Lai, and Y.N. Young. A hybrid immersed boundary and immersed interface method for electrohydrodynamic simulations. *J. Comput. Phys.*, 282:47–61, 2015.

[17] Y. Kim and M.C. Lai. Simulating the dynamics of inextensible vesicles by the penalty immersed boundary method. *J. Comput. Phys.*, 229:4840–4853, 2010.

[18] Y. Kim, M.C. Lai, and C.S. Peskin. Numerical simulations of two-dimensional foam by the immersed boundary method. *J. Comput. Phys.*, 229:5194–5207, 2010.

[19] Y. Kim, M.C. Lai, C.S. Peskin, and Y. Seol. Numerical simulations of three-dimensional foam by the immersed boundary method. *J. Comput. Phys.*, 269:1–21, 2014.

[20] M.C. Lai and C.S. Peskin. An Immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *J. Comput. Phys.*, 160:705–719, 2000.

[21] M.C. Lai, Y.H. Tseng, and H. Huang. An immersed boundary method for interfacial flows with insoluble surfactant. *J. Comput. Phys.*, 227:7279–7293, 2008.

[22] D.V. Le, B.C. Khoo, and J. Peraire. An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries. *J. Comput. Phys.*, 220:109–138, 2006.

[23] S. Leung, J. Lowengrub, and H.K. Zhao. A grid based particle method for high order geometrical motions and local inextensible flows. *J. Comput. Phys.*, 230:2540–2561, 2011.

[24] S. Leung and H.K. Zhao. A Grid Based Particle Method for Moving Interface Problems. *J. Comput. Phys.*, 228:2993–3024, 2009.

[25] S. Leung and H.K. Zhao. A grid-based particle method for evolution of open curves and surfaces. *J. Comput. Phys.*, 228:7706–7728, 2009.

[26] R.J. LeVeque and Z.L. Li. The immersed interface method for elliptic equations with discontinous coefficients and singular sources. *SIAM J. Num. Anal.*, 31:1019–1044, 1994.

[27] R.J. LeVeque and Z.L. Li. Immersed interface methods for Stokes flow with elastic boundaries or surface tension. *SIAM J. Sci. Comput.*, 18:709–735, 1997.

[28] Z.L. Li. *The immersed interface method: A numerical approach for partial differential equations with interfaces*. PhD thesis, University of Washington, 1994.

[29] Z.L. Li. Immersed interface method for moving interface problems. *Numer. Algorith.*, 14:269–293, 1997.

[30] Z.L. Li. A fast iterative algorithm for elliptic interface problems. *SIAM J. Num. Anal.*, 25:230–254, 1998.

[31] Z.L. Li and K. Ito. Maximum principle preserving schemes for interface problems with discontinuous coefficients. *SIAM J. Sci. Comput.*, 23:330–361, 2001.

[32] Z.L. Li and M.C. Lai. The immersed interface method for the Navier–Stokes equations with singular forces. *J. Comput. Phys.*, 171:822–842, 2001.

[33] Z.L. Li, W.C. Wang, I.L. Chern, and M.C. Lai. New formulations for interface problems in polar coordinates. *SIAM J. Sci. Comput.*, 25:224–245, 2003.

[34] X.D. Liu, R.P. Fedkiw, and M. Kang. A boundary condition capturing method for Possion's equation on irregular domains. *J. Comput. Phys.*, 160:151–178, 2000.

[35] X.D. Liu and T.C. Sideris. Convergence of the ghost fluid method for ellipic equations with interfaces. *Math. Comp.*, 72:1731–1746, 2003.

[36] S. J. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.

[37] Berthelsen P.A. A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions. *J. Comput. Phys.*, 197:364–386, 2004.

[38] C.S. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 25:220–252, 1977.

[39] C.S. Peskin. The immersed boundary method. *Acta. Numer.*, 11:479–517, 2002.

[40] W. Proskurowski and O. Widlund. On the numerical solution of Helmholtz's equation by the capacitance matrix method. *Math. Comput.*, 30:433–468, 1976.

[41] John Ruge and Klaus Stüben. *Efficient solution of finite difference and finite element equations by algebraic multigrid AMG*. Gesellschaft f. Mathematik u. Datenverarbeitung, 1984.

[42] John W Ruge and Klaus Stüben. Algebraic multigrid. *Multigrid methods*, 3(13):73–130, 1987.

[43] Y. Seol, W.F. Hu, Y. Kim, and M.C. Lai. An immersed boundary method for simulating vesicle dynamics in three dimensions. *J. Comput. Phys.*, 322:125–141, 2016.

[44] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis, third ed.* Springer-Verlag, 2002.

[45] Klaus Stüben. Algebraic multigrid (amg): experiences and comparisons. *Applied mathematics and computation*, 13(3):419–451, 1983.

[46] S.W. Su, M.C. Lai, and C.A. Lin. An immersed boundary technique for simulating complex flows with rigid boundary. *J. Comput. Phys.*, 36:313–324, 2007.

[47] P.N. Swarztrauber. *Fast poisson solver, In G.H. Golub, editor, Studies of Numerical Analysis*, volume 24. The Mathematics Association of America, 1984, pp. 319-370.

[48] Z.J. Tan, K.M. Lim, and B.C. Khoo. An immersed interface method for Stokes flows with fixed/moving interfaces and rigid boundaries. *J. Comp. Phys.*, 228:6855–6881, 2009.

[49] A.K. Tornberg and B. Engquist. Regularization techniques for numerical approximation of PDEs with singularities. *J. Sci. Comput.*, 19:527–552, 2003.

[50] A.K. Tornberg and B. Engquist. Numerical approximations of singular source terms in differential equations. *J. Comput. Phys.*, 200:462–488, 2004.

[51] A.K. Tornberg, B. Engquist, and R. Tsai. Discretization of Dirac delta functions in level set methods. *J. Comput. Phys.*, 207:28–51, 2005.

[52] Y.H. Tseng and J.H. Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *J. Comput. Phys.*, 192:593–623, 2003.

[53] C. Tu and C.S. Peskin. Stability and instability in the computation of flows with moving immersed boundaries: a comparison of three methods. *SIAM J. Sci. Comput.*, 13:1361–1376, 1992.

[54] L.Q. Wang, S.M. Hou, and L.W. Shi. An improved non-traditional finite element formulation for solving the elliptic interface problems. *J. Comp. Math.*, 32:39–57, 2013.

[55] L.Q. Wang, S.M. Hou, and L.W. Shi. A numerical method for solving three-dimensional elliptic interface problems with triple junction points. *Advances in Computational Mathematics*, in press, 2017.

[56] L.Q. Wang, S.M. Hou, and L.W. Shi. A Weak Formulation for Solving the Elliptic Interface Problems with Imperfect Contact. *Advances in Applied Mathematics and Mechanics*, 9(5):1189–1205, 2017.

[57] L.Q. Wang, S.M. Hou, and L.W. Shi. An Improved non-traditional Finite Element Formulation for solving three dimensional Elliptic Interface Problems. *Computers & Mathematics with Applications*, 73:374–384, 2017.

[58] L.Q. Wang, S.M. Hou, L.W. Shi, and J. Solow. A Numerical Method for solving Elasticity Equations with Interface involving Multi-domain and Triple Junction Points. *Applied Mathematics and Computation*, 251:615–625, 2015.

Ningchen Ying,
Department of Mathematics,
the Hong Kong University of Science and Technology,
Clear Water Bay, Hong Kong.
E-mail address: mancying@ust.hk, nying@connect.ust.hk

Songming Hou,
Department of Mathematics & Statistics and Center of Applied Physics,
Louisiana Tech University,
Ruston, LA 71272, United States.
E-mail address: shou@latech.edu

Shingyu Leung,
Department of Mathematics,
the Hong Kong University of Science and Technology,
Clear Water Bay, Hong Kong.
E-mail address: masyleung@ust.hk

Hongkai Zhao,
Department of Mathematics,
University of California at Irvine,
Irvine, CA 92697-3875, United States.
E-mail address: zhao@math.uci.edu