Journal of Computational Physics 228 (2009) 6855–6881

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

An immersed interface method for Stokes flows with fixed/moving interfaces and rigid boundaries

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

ARTICLE INFO

Article history:
Received 16 November 2008
Received in revised form 23 April 2009
Accepted 10 June 2009
Available online 17 June 2009

Keywords:
Incompressible Stokes equations
Singular force
Immersed interface method
CG-Uzawa method
Deformable interface
Front tracking
Irregular domains
Rigid boundaries

ABSTRACT

We present an immersed interface method for solving the incompressible steady Stokes equations involving fixed/moving interfaces and rigid boundaries (irregular domains). The fixed/moving interfaces and rigid boundaries are represented by a number of Lagrangian control points. In order to enforce the prescribed velocity at the rigid boundaries, singular forces are applied on the fluid at these boundaries. The strength of singular forces at the rigid boundary is determined by solving a small system of equations. For the deformable interfaces, the forces that the interface exerts on the fluid are calculated from the configuration (position) of the deformed interface. The jumps in the pressure and the jumps in the derivatives of both pressure and velocity are related to the forces at the fixed/moving interfaces and rigid boundaries. These forces are interpolated using cubic splines and applied to the fluid through the jump conditions. The positions of the deformable interfaces are updated implicitly using a quasi-Newton method (BFGS) within each time step. In the proposed method, the Stokes equations are discretized via the finite difference method on a staggered Cartesian grid with the incorporation of jump contributions and solved by the conjugate gradient Uzawa-type method. Numerical results demonstrate the accuracy and ability of the proposed method to simulate incompressible Stokes flows with fixed/moving interfaces on irregular domains.

1. Introduction

The low Reynolds number flow with interfaces in complex geometries is of interest in many engineering and physiological applications, for example, multi-phase flows in various fluid components within fuel cell, biological cell trapping and manipulation in microfluidic device, and droplet motion in confined geometries. Flow problems involving deformable interfaces and complex geometries often pose numerical difficulties and challenges in computational fluid dynamics. One of the difficulties and challenges in these problems is that the fluid motion, the motion of the deformable interface and the interaction with the rigid boundaries must be computed simultaneously. This is necessary in order to account for the complex interaction between the fluid, the interfaces and the rigid boundaries. The other difficulty is the accuracy of the fluid domain computation, and this can be improved partially by implementing moving mesh techniques as in [10,47]. Fig. 1 shows an illustration of such flow problems involving the rigid boundary and fixed/deformable interface embedded in a uniform Cartesian grid.

* Corresponding author. Address: Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore. Tel.: +65 65162889; fax: +65 67791459.
E-mail addresses: smatz@nus.edu.sg (Z. Tan), mpelimkm@nus.edu.sg (K.M. Lim), mpekbc@nus.edu.sg (B.C. Khoo).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.06.005

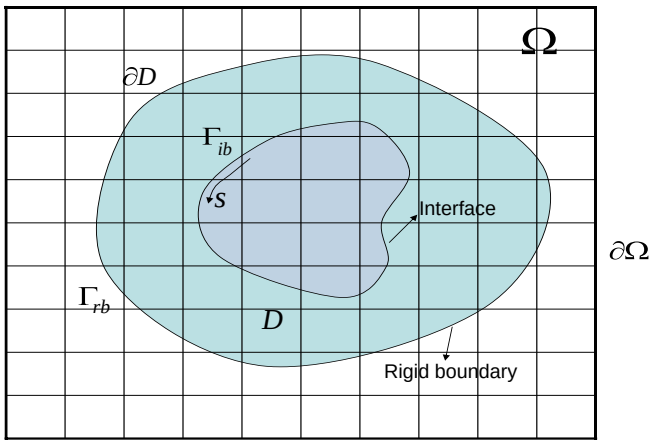


Fig. 1. A typical irregular domain D involving fluid interface, immersed in a simpler regular domain with a uniform Cartesian grid.

For the treatment of interaction between the fluid motion and rigid boundaries, traditional methods for simulating the flow with rigid boundaries include the body-fitted or unstructured grid methods. In this approach, the flow equations are discretized on a curvilinear grid that conforms to the immersed boundary so that the boundary conditions can be imposed easily. However, this method requires robust grid generation to account for the complexity of the immersed boundaries, and the computational cost caused and memory requirements of these methods are generally high.

In comparison, solving the governing equations on a Cartesian grid has the advantages of retaining the simplicity of the flow equations on the Cartesian coordinates and fast solvers can be used. One of the most successful Cartesian grid methods is Peskin's immersed boundary method [35,34], which provides an alternative approach to the treatment of the rigid boundaries. Despite the existing disadvantage that immersed boundary method is essentially first-order accurate for problems with non-smooth but continuous solutions since it smears out sharp interface to a thickness of order of the meshwidth. The method has been used in a wide variety of applications due to its ease of implementation, particularly in biofluid dynamics problems where complex geometries and immersed elastic boundaries are present. Examples of applications include the deformation of red blood cell in a shear flow [12], swimming of organisms [15], platelet aggregation [16,17,52], cochlear dynamics [4], biofilm processes [11], wood pulp fiber dynamics [39]. A summary of the development of the immersed boundary method can be found in [35].

Based on the same Cartesian grid framework, the immersed interface method (IIM), improves on the immersed boundary method by not smearing out sharp interfaces and maintains second-order accuracy by incorporating the known jumps into the finite difference scheme near the interface. The IIM was originally proposed by LeVeque and Li [27] for solving elliptic equations, and was later extended to Stokes flow with elastic boundaries or surface tension [26]. In [26,31], the IIM was based on the standard grid (not MAC grid) and employed the solution of three Poisson equations : one pressure Poisson equation and two velocity Poisson equations. There appears to be some difficulties in the Dirichlet boundary condition treatment and therefore bi-periodic boundary for the velocity and pressure are assumed. In addition, only regular domains are considered. The method was developed further for the Navier–Stokes equations in [29,24,25,55] for problems with flexible boundaries. In [43], Sethian and Shan presented a numerical algorithm for solving partial differential equations on irregular domains with moving interfaces for application to the electrodeposition process. We refer the interested readers to the recently published book by Li and Ito [28] and the references therein.

Still, the interest in IIM has led to several works for the simulation of fluid flows on irregular domains [6,30,40,32]. In [6,30], the no-slip boundary conditions were imposed directly by determining the correct jump conditions for the streamfunction and vorticity. In [40], a Cartesian grid method is developed, with an underlying regular Cartesian grid employed to solve the system using a streamfunction-vorticity formulation and discontinuities representing the embedded objects. The no-penetration condition for the moving geometry is satisfied by superposing a homogenous solution to the Poisson's equation for the streamfunction. The no-slip condition is satisfied by generating vorticity on the surfaces of the objects. Linnick and Fasel [32] presented a high-order modified IIM for the 2D-unsteady-incompressible Navier–Stokes equations in stream function-vorticity formulation. The method employs explicit fourth-order Runge–Kutta time integration scheme, fourth-order compact finite-differences for computation of spatial derivatives, and a nine-point-fourth-order compact discretization of the Poisson equation for computation of the stream function. Rutka [41] developed the explicit immersed interface method (EJIIM) for two-dimensional Stokes flows on irregular domain. The EJIIM introduces unknown jumps in the solution and its derivatives up to second-order along the interface. The augmented system of equations using EJIIM in [41] is larger due to the many unknowns are introduced as augmented variables, which increases the computational cost. In [7], a new biharmonic solver has been applied to solve the incompressible Stokes flow on an irregular domain. Taira and colonius [46] present an immersed boundary projection method, where the boundary force is applied along the immersed boundary to satisfy

the no-slip condition. In Cortez [9] developed a method of regularized Stokeslets based on the smoothing of force. Other related works on Stokes flows but completely different approaches include the boundary integral method (BIM) [5,37] and boundary element method (BEM) [19,53]. Biros et al. [5] proposed the embedded boundary integral (EBI) method for Stokes equations with distributed forces in complex geometries. The method uses an integral formulation to compute the jumps of the velocity and its derivatives at the interface and expresses the jumps as a source term at some grid points close to the interface. For a detailed presentation of the theory of the BIM for Stokes flows, we refer the interested readers to Pozrikidis [37] and the references therein. It is worth mentioning that there are yet some similarities for solving immersed boundary or domain boundary problem between the present IIM and the BIM. Compared with our approach, the BIM requires the Green function and it needs to set-up the system of equations. As such, more sophisticated level of mathematics for integral formulation and the development of accurate quadratures of integrals with singular kernels are (absolutely) required. From the numerical perspective or implementation, the resulting integrals become nearly singular, which make them extremely difficult to evaluate accurately for points close to the boundary. If information is required at a large number of internal points in a large expanse, the BIM would need highly accurate interpolation scheme. Also the BIM cannot be applied to solve for the Navier–Stokes equations directly. When the BIM is used to solve non-linear equation, the expensive volume integration and evaluation of interior values are involved. However, our method is deemed for more simpler to implement by only using the simple finite difference scheme with the incorporation of correction terms. While the current IIM is of particular interest to steady Stokes flows with interfaces and irregular domains in the present work, our method can be also applied potentially to solve for other types of fluid flow (including the unsteady Stokes flow, steady Navier–Stokes flow and unsteady Navier–Stokes flow) involving interfaces and irregular domains based on only a simple extension from the present Stokes solver to the efficient generalized Stokes solver [14,38,21] in a fairly straightforward manner. For example, for the case of unsteady Navier–Stokes equations, the generalized Stokes equations can be obtained via a semi-implicit temporal discretization using the Crank–Nicolson scheme for the viscous terms and the Adams–Bashforth scheme for the convective terms at each time step. As such, our approach is very flexible for different types of fluid flow problems. Another advantage to our approach is the capability to treat the incompressible viscous flow with interfaces and irregular domains simultaneously as will be seen from our numerical results, in this sense, which is very significant in providing a way to get a second-order IIM using a MAC finite difference scheme for the broad area of fluid-structure interaction studies. It is noted that most works on flow problems with irregular domains are focused on Navier–Stokes flows based on streamfunction-vorticity formulation. However, works on IIM for solving steady Stokes flow on irregular domain are still relatively very few. Moreover, all the above mentioned works and others found in the literature on solving the Stokes flow problems using the IIM are largely limited to treating either only interfaces on regular domains [26,31,22,23] or only irregular domains with no interfaces involved [7,41]. To the best of our knowledge, there seems to be no other work reported by far on implementing the IIM for the incompressible steady Stokes flow involving both (flexible) interfaces and rigid boundaries. The present implementation of the current approach for such flow problems is both new and non-trivial. It may be noted that even in the case of only regular domains involved, the literature on IIM for the steady Stokes flows with Dirichlet boundary conditions appears to be also rather limited. Most work on IIM using uniform Cartesian grids for Stokes flow is based on solving three Poisson problems on a regular non-staggered grid via implementing the periodic boundary conditions [26,31].

It should be mentioned that there are other Cartesian grid methods such as ghost-cell finite difference approach [3,49] and cut-cell finite-volume approach [51,56,20]. In the ghost-cell approach, the ghost cells which are defined as cells interior to the body and have at least one neighbor in the fluid are determined and the required ghost-cell values are extrapolated to impose the boundary condition implicitly. In the cut-cell approach, cells in the Cartesian grid that are cut by the boundary are identified and reshaped.

In this work, we present an IIM with second-order accuracy for solving the incompressible viscous Stokes flows in the presence of both fixed/moving interfaces and irregular domains. The method combines the IIM with a front tracking representation of the interface/boundary on a uniform Cartesian grid. In the proposed method, singular forces at the rigid boundary are introduced to enforce the prescribed velocity condition at the rigid boundary and then determined by solving a small, dense linear system of equations via the LU method. The advantage of such an approach of implicit-forcing for complex boundary is that it imposes exactly the prescribed velocity condition at the rigid boundary and avoid the need for very small time steps. The forces associated with the deformable interface are computed from the configuration (position) of the interface and applied to the fluid through the jump conditions. Once the forces at the rigid boundaries are computed and the forces at the fixed/moving interfaces are prescribed/computed, the jumps in pressure and the jumps in the derivatives of velocity and pressure are related to these forces which are interpolated using cubic splines. For the deformable interface problem on irregular domains, the positions of the deformable interfaces are updated implicitly within each time step. The Stokes equations with primitive variables are discretized on a staggered Cartesian grid by a second-order MAC finite difference scheme and solved by the conjugate gradient Uzawa-type method. The jumps in the solution and its derivatives are incorporated into the finite difference discretization to obtain a sharp interface resolution. Fast solvers from the FISHPACK software library [1] have been used to solve the resulting discrete systems of Poisson equations. The numerical results show that the overall scheme is second-order accurate for the velocity and nearly second-order accurate for the pressure. The capability of the proposed method to simulate the incompressible steady Stokes flow with fixed/moving interfaces and irregular domains is demonstrated by testing a number of problems, including rotational flow on irregular domain, the relaxation of an elastic membrane placed on irregular domain, deformation of a drop between two concentrically rotating cylinders, an elastic membrane in a contraction flow and the motion of an elastic membrane in a groove.

This paper is organized as follows. In Section 2, the model of incompressible Stokes flows with interfaces on irregular domains is described. In Section 3, we present the jump relations along the immersed interface via the singular force **f** and the jumps in the velocity and pressure and their derivatives. The numerical algorithm and numerical implementation are presented in Sections 4 and 5, respectively. In Section 6, several numerical examples are presented. Some concluding remarks will be made in Section 7.

2. Governing equations

This papers concerns the viscous incompressible Stokes flows involving fluid interfaces and rigid boundaries in 2D. Without loss of generality, we assume below that only a fluid fixed/moving interface Γ_{ib} and a rigid boundary Γ_{rb} are involved, and extension to involve multiple fluid interfaces and multiple rigid boundaries is straightforward. In a two-dimensional computational domain $\Omega = [a,b] \times [c,d]$, we consider the steady Stokes equations formulated in the velocity–pressure variables, written as

$$\nabla p = \mu \Delta \mathbf{u} + \mathbf{F}(\mathbf{x},t) + \mathbf{g}(\mathbf{x},t), \quad \mathbf{x} \in \Omega, \tag{2.1}$$

$$\nabla \cdot \mathbf{u} = 0, \quad \mathbf{x} \in \Omega, \tag{2.2}$$

with boundary conditions

$$\mathbf{u}|_{\partial\Omega} = \mathbf{u}_b, \quad \mathbf{u}|_{\partial D} = \mathbf{u}_p, \tag{2.3}$$

where $\mathbf{u} = (u,v)^T$ is the fluid velocity, μ is the fluid viscosity, $\mathbf{x} = (x,y)$ is the Cartesian coordinate variable, $\mathbf{g}(\mathbf{x},t) = (g_1,g_2)^T$ is an external force which can be a function of time t, $\mathbf{u}_p$ is the prescribed velocity at the rigid boundary. The effects of the interface and rigid boundary immersed in the fluid result in the total singular forces **F** which are typically modeled as a Dirac delta function along the interface and boundary as follows:

$$\mathbf{F}(\mathbf{x},t) = \int_{\Gamma_{ib}} \mathbf{f}^{ib}(s,t)\delta(\mathbf{x} - \mathbf{X}^{ib}(s,t))ds + \int_{\Gamma_{rb}} \mathbf{f}^{rb}(s,t)\delta(\mathbf{x} - \mathbf{X}^{rb}(s,t))ds. \tag{2.4}$$

Here, $\mathbf{X}^{ib}(s,t)$ and $\mathbf{X}^{rb}(s,t)$ are the arc-length parametrization of the fluid interface Γ_{ib} and rigid boundary Γ_{rb}, respectively, s is the arc-length, $\mathbf{f}^{ib} = (f_1^{ib}, f_2^{ib})^T$ and $\mathbf{f}^{rb} = (f_1^{rb}, f_2^{rb})^T$ are the corresponding force densities along the fluid interface and rigid boundary, respectively, and $\delta(\cdot)$ is the Dirac delta function defined in the distribution sense. Eq. (2.2) together with the Dirichlet condition Eq. (2.3) leads to the compatibility condition that $\mathbf{u}_b$ must satisfy

$$\int_{\partial\Omega} \mathbf{u}_b \cdot \mathbf{n} dS = 0, \tag{2.5}$$

where **n** is the outer unit normal to $\partial\Omega$. The motion of the deformable interface satisfies

$$\frac{\partial \mathbf{X}^{ib}(s,t)}{\partial t} = \mathbf{u}(\mathbf{X}^{ib}(s,t),t) = \int_{\Omega} \mathbf{u}(\mathbf{x},t)\delta(\mathbf{x} - \mathbf{X}^{ib}(s,t))d\mathbf{x}. \tag{2.6}$$

We shall consider a moving interface problem which involves an elastic membrane, where the force strength $\mathbf{f}^{ib}$ exerted by elastic membrane on the fluid is given by

$$\mathbf{f}^{ib}(s,t) = \frac{\partial}{\partial s}(T(s,t)\boldsymbol{\tau}(s,t)), \tag{2.7}$$

with the tension $T(s,t)$ given by

$$T(s,t) = T_0\left(\left|\frac{\partial \mathbf{X}^{ib}(s,t)}{\partial s_0}\right| - 1\right). \tag{2.8}$$

Here, the tension coefficient T_0 is the stiffness constant which describes the elastic property of the membrane and s and s_0 are the arc-lengths measured along the current and undeformed configuration of the membrane. The vector tangential to Γ is given by $\boldsymbol{\tau}(s,t)$, where

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

Thus, the force density can be computed directly from the location $\mathbf{X}^{ib}$ of the deformable interface Γ_{ib}. An equivalent form of Eq. (2.7) is

$$\mathbf{f}^{ib}(s,t) = (\partial T/\partial s)\boldsymbol{\tau}(s,t) + T\kappa\mathbf{n}, \tag{2.9}$$

where κ is the curvature, defined by $\partial\boldsymbol{\tau}/\partial s = \kappa\mathbf{n}$. If we assume a stressed initial configuration and T depends linearly on $\left|\frac{\partial \mathbf{X}^{ib}(s,t)}{\partial s_0}\right|$, then we obtain the surface tension model from (2.7) as follows:

$$\mathbf{f}(s,t) = \gamma \frac{\partial^2}{\partial s^2}\mathbf{X}^{ib}(s,t), \tag{2.10}$$

where γ is the surface tension coefficient. This surface tension model is used in the numerical simulation of the Example 6.4 in this paper.

Throughout this paper, we shall assume that the fluid viscosity μ is constant over the whole domain. We refer the readers to Fig. 1 for an illustration of the problem.

3. Jump conditions across the interface/boundary

In order to implement the IIM, we need to know the jump conditions for the velocity and pressure and their spatial derivatives. Let $\mathbf{n} = (n_1, n_2)$ and $\boldsymbol{\tau} = (\tau_1, \tau_2)$ be the unit outward normal and tangential vectors to the interface/boundary, respectively. The jump of an arbitrary function $q(\mathbf{X})$ across the interface/boundary at $\mathbf{X}$ is denoted by

$$[q] = \lim_{\epsilon \to 0^+} q(\mathbf{X} + \epsilon\mathbf{n}) - \lim_{\epsilon \to 0^+} q(\mathbf{X} - \epsilon\mathbf{n}). \tag{3.1}$$

Denoting (ξ, η) the local coordinates associated with the directions of $\mathbf{n}$ and $\boldsymbol{\tau}$, respectively, we have the jump conditions for the velocity and pressure across the interface/boundary related to normal force and tangential force and express them in the local Cartesian coordinate as follows (see [29,24] for details):

$$[\mathbf{u}] = \mathbf{0}, \quad [\mathbf{u}_\eta] = \mathbf{0}, \quad [\mathbf{u}_\xi] = -\frac{1}{\mu}\hat{f}_2\boldsymbol{\tau}, \quad [\mathbf{u}_{\eta\eta}] = \frac{1}{\mu}\kappa\hat{f}_2\boldsymbol{\tau}, \tag{3.2}$$

$$[\mathbf{u}_{\xi\eta}] = -\frac{1}{\mu}\frac{\partial \hat{f}_2}{\partial \eta}\boldsymbol{\tau} - \frac{1}{\mu}\kappa\hat{f}_2\mathbf{n}, \quad [\mathbf{u}_{\xi\xi}] = -[\mathbf{u}_{\eta\eta}] + \frac{1}{\mu}[p_\xi]\mathbf{n} + \frac{1}{\mu}[p_\eta]\boldsymbol{\tau} - \frac{1}{\mu}[\mathbf{g}], \tag{3.3}$$

$$[p] = \hat{f}_1, \quad [p_\xi] = [\mathbf{g}]\cdot\mathbf{n} + \frac{\partial \hat{f}_2}{\partial \eta}, \quad [p_\eta] = \frac{\partial \hat{f}_1}{\partial \eta}, \tag{3.4}$$

$$[p_{\eta\eta}] = \frac{\partial^2 \hat{f}_1}{\partial \eta^2} - \kappa[p_\xi], \quad [p_{\xi\eta}] = \frac{\partial([\mathbf{g}]\cdot\mathbf{n})}{\partial \eta} + \frac{\partial^2 \hat{f}_2}{\partial \eta^2} + \kappa[p_\eta], \quad [p_{\xi\xi}] = [\nabla\cdot\mathbf{g}] - [p_{\eta\eta}]. \tag{3.5}$$

Here, $\hat{f}_1$ and $\hat{f}_2$ are the components of the force density in the normal and tangential directions of the interface/boundary such that $\hat{\mathbf{f}} = (\hat{f}_1, \hat{f}_2)$, and κ is the signed valued of the curvature of the interface/boundary. In this work, we shall incorporate the jump conditions of second-order spatial derivatives for the pressure into the finite difference scheme; this can be compared to the usual jump condition of first-order spatial derivatives for the pressure in the literature [31,29,26,24]. In order to be numerically useful, those jump conditions for the first and second derivatives of the velocity and pressure in (3.2)–(3.5) in the local coordinate are transformed into the jump relations in the Cartesian coordinate by a simple coordinate transformation [29]. As such, we have

$$[q_x] = [q_\xi]n_1 + [q_\eta]\tau_1, \quad [q_y] = [q_\xi]n_2 + [q_\eta]\tau_2, \tag{3.6}$$

$$[q_{xx}] = [q_{\xi\xi}]n_1^2 + 2[q_{\xi\eta}]n_1\tau_1 + [q_{\eta\eta}]\tau_1^2, \tag{3.7}$$

$$[q_{yy}] = [q_{\xi\xi}]n_2^2 + 2[q_{\xi\eta}]n_2\tau_2 + [q_{\eta\eta}]\tau_2^2, \quad q = \mathbf{u}, p. \tag{3.8}$$

4. Numerical algorithm

Our numerical algorithm is based on the conjugate gradient Uzawa-type algorithm for the discretization of the Stokes equations with special treatment at the grid points near the interface. The spatial discretization is carried out on a standard marker-and-cell (MAC) staggered grid similar to that found in Tau [48]. We use a uniform MAC grid with the spacing $h = \Delta x = \Delta y$ in the computation. With this MAC grid, the pressure is defined at p-mesh points $(x_{i+1/2}, y_{j+1/2}) = (a + (i - 1/2)h, b + (j - 1/2)h)$, where $i \in \{1, 2, \ldots, N_x\}$ and $j \in \{1, 2, \ldots, N_y\}$. The velocity components u and v are defined at u-mesh points $(x_i, y_{j+1/2}) = (a + (i - 1)h, b + (j - 1/2)h)$ and v-mesh points $(x_{i+1/2}, y_j) = (a + (i - 1/2)h, b + (j - 1)h)$, respectively. The arrangement of the pressure and velocity components is shown as in Fig. 2. An advantage of such a MAC grid is that there is no need for pressure boundary conditions while dealing with the derivative of pressure since the pressure nodes are at the cell center.

4.1. Stokes solver with correction terms

Discretization of Eqs. (2.1)–(2.3) by second-order MAC finite difference scheme leads to the following linear system

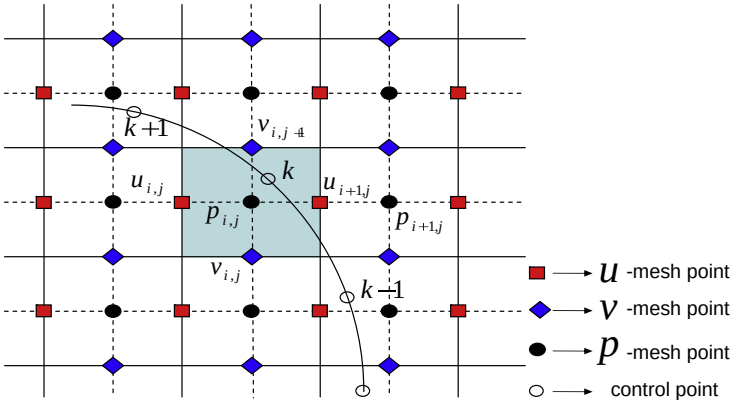


Fig. 2. A diagram of the interface cutting through a staggered grid with a uniform mesh width h, where the velocity component u is at the left-right face of the cell and v is at the top-bottom face, and the pressure is at the cell center.

$$G^{\mathrm{MAC}}p = \mu\Delta_h\mathbf{u} + \mathbf{g}(\mathbf{x}) + \boldsymbol{C}_1, \tag{4.1}$$

$$D^{\mathrm{MAC}}\mathbf{u} = C_2, \quad \mathbf{u}|_{\partial\Omega} = \mathbf{u}_b. \tag{4.2}$$

The above discretization of the Stokes equations at those grid points near the interface and rigid boundary has been modified to account for the jump conditions across the fixed/moving interface and rigid boundary due to the presence of singular forces at the fixed/moving interface and rigid boundary. The coefficients $\boldsymbol{C}_1$ and C_2 are the spatial correction terms added to the finite difference equations at the points near the interface and rigid boundary to improve the accuracy of the local finite difference approximations and will be evaluated later. In order to satisfy the discrete compatibility condition corresponding to (2.5) to thereby ensure the solvability of system Eqs. (4.1) and (4.2), we employed a solvable perturbed system with similar approach as in [26] via perturbing C_2 to $C_2 - \widehat{C}_2$ on the right hand of Eq. (4.2). Here $\widehat{C}_2$ is the mean value of the correction term C_2. We refer the readers to [26] for details. In the above expressions, Δ_h is the standard central difference operator, and G^{MAC} and D^{MAC} are the MAC gradient and divergence operators, respectively. These operators are defined as

$$\begin{aligned} &\Delta_h\mathbf{u}_{i,j} = \frac{\mathbf{u}_{i+1,j} + \mathbf{u}_{i-1,j} + \mathbf{u}_{i,j+1} + \mathbf{u}_{i,j-1} - 4\mathbf{u}_{ij}}{h^2}, \quad (G^{\mathrm{MAC}}p)_{i,j} = \left(\frac{p_{i+1,j} - p_{i,j}}{h}, \frac{p_{i,j+1} - p_{i,j}}{h}\right), \\ &(D^{\mathrm{MAC}}\mathbf{u})_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{h} + \frac{v_{i,j+1} - v_{i,j}}{h}. \end{aligned} \tag{4.3}$$

Denoting $\mathbf{G}_1 = \mathbf{g}(\mathbf{x}) + \boldsymbol{C}_1$ and $G_2 = C_2 - \widehat{C}_2$ as the right-hand equivalent of Eqs. (4.1) and (4.2), the linear system (4.1) and (4.2) can be written in the matrix–vector form as

$$\begin{pmatrix} -\mu\Delta_h & G^{\mathrm{MAC}} \\ D^{\mathrm{MAC}} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{G}_1 \\ G_2 \end{pmatrix}. \tag{4.4}$$

Currently, there exists some fast solvers to solve system (4.4), for example, the PCG method [13,36], the PMINRES method [13,36], the FFT-based method [8], multigrid method [33,13,36], and so on. In this work, we shall employ the CG-Uzawa method. The Uzawa procedure for problems with interfaces/boundaries is analogous to the fast iterative method presented in [48,44] and consists of two steps:

Step 1 : Solve $\mathcal{M}p = \mu G_2 + D^{\mathrm{MAC}}\Delta_h^{-1}\mathbf{G}_1$ to obtain the pressure p.
Step 2 : Solve $\mu\Delta_h\mathbf{u} = \mathbf{G}_1 - G^{\mathrm{MAC}}p$ to obtain the velocity $\mathbf{u}$.

Here, the $\mathcal{M} = D^{\mathrm{MAC}}\Delta_h^{-1}G^{\mathrm{MAC}}$ is the Schur complement of system (4.4). In *Step 1*, the system is solved by the conjugate gradient method (CG) in this work. In the CG method, each matrix–vector product of $\mathcal{M}p$ requires one application of Δ_h^{-1} which corresponds to solving one Poisson equation, and which can be solved by several efficient methods, for example, ICCG method, the FFT method and multigrid method. In the present work, we can take advantage of the fast solvers from FISHPACK [1] to solve these Poisson equations. Once the pressure is obtained, the velocity field $\mathbf{u}$ can be again solved by the fast solvers from FISHPACK [1] via *Step 2*. The computational complexity for the fast Poisson solver from FISHPACK is $\mathcal{O}(\mathrm{M}\log(\mathrm{M}))$, where M is the number of interior grid points of the embedded domain. It turns out that for some properly chosen operators the condition number of the Schur complement matrix $\mathcal{M}$ is bounded independently of the mesh size as discussed in [48,44,14], so the CG algorithm converges rapidly. The number of iterations in the CG method is reasonably small and are independent of the mesh size for the present numerical examples in Section 6. As such, the present Stokes solver is fast and efficient as discussed in [48,44].

4.2. Calculation of correction terms

One of the basic components for determining the correction terms is the generalized finite difference formulas. We shall briefly review the generalized finite difference formulas in this section. Here, we show four particular generalized finite difference formulas for demonstration. Assume that the interface/boundary cuts a grid line between two grid points at $x = \alpha, x_i \leqslant \alpha < x_{i+1}, x_i \in \Omega^-, x_{i+1} \in \Omega^+$, where Ω^- and Ω^+ denote the region inside and outside the interface, respectively. Then, the following approximations hold for a piecewise twice differentiable function $w(x)$:

$$w_x(x_i) = \frac{w_{i+1} - w_{i-1}}{2h} - \frac{1}{2h}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}[w^{(m)}]_\alpha + O(h^2), \tag{4.5a}$$

$$w_x(x_{i+1}) = \frac{w_{i+2} - w_i}{2h} - \frac{1}{2h}\sum_{m=0}^{2}\frac{(h^-)^m}{m!}[w^{(m)}] + O(h^2), \tag{4.5b}$$

$$w_{xx}(x_i) = \frac{w_{i+1} - 2w_i + w_{i-1}}{h^2} - \frac{1}{h^2}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}[w^{(m)}]_\alpha + O(h), \tag{4.5c}$$

$$w_{xx}(x_{i+1}) = \frac{w_{i+2} - 2w_{i+1} + w_i}{h^2} + \frac{1}{h^2}\sum_{m=0}^{2}\frac{(h^-)^m}{m!}[w^{(m)}] + O(h), \tag{4.5d}$$

where $w^{(m)}$ denotes the mth derivative of $w, w_i = w(x_i), h^+ = x_{i+1} - \alpha, h^- = x_i - \alpha$ and h is the mesh width in x-direction. The jump in w and its derivatives are defined as

$$[w^{(m)}]_\alpha = \lim_{x\to\alpha, x\in\Omega^+} w^{(m)}(x) - \lim_{x\to\alpha, x\in\Omega^-} w^{(m)}(x); \tag{4.6}$$

in short, $[\cdot] = [\cdot]_\alpha$, and $w^{(0)} = w$. Note that if the interface/boundary cuts a grid line between two grid points $x_i \in \Omega^+$ and $x_{i+1} \in \Omega^-$, these expressions need to be modified by changing the sign of the second terms on the respective right-hand sides. Expressions involving two or more interface/boundary crossings could also be derived, we refer the readers to [54] for details. From Eqs. (4.5a) and (4.5c) the correction terms for $w_x(x_i)$ and $w_{xx}(x_i)$ can be defined as

$$C\{w_x(x_i)\} = -\frac{1}{2h}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}[w^{(m)}], \tag{4.7}$$

$$C\{w_{xx}(x_i)\} = -\frac{1}{h^2}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}[w^{(m)}]. \tag{4.8}$$

Thus, the finite difference approximation near the interface/boundary, for the derivatives of a function q, includes the standard central difference terms plus the additional correction terms. Accordingly, the correction terms $\mathbf{C}_1$ and C_2 are evaluated as follows:

$$\mathbf{C}_1 = \mu(C\{\Delta\mathbf{u}\}) - C\{\nabla p\}, \tag{4.9a}$$

$$C_2 = -C\{\nabla\cdot\mathbf{u}\}. \tag{4.9b}$$

We note that all the correction terms are evaluated at least to first-order accuracy, i.e., the local truncation error of the present scheme is $O(h^2)$ at an regular grid point away from the interface/boundary, while $O(h)$ at an irregular grid point near the interface/boundary. Despite the first-order truncation errors at those irregular points, the overall accuracy is still second-order since the number of irregular grid points is much less than the total number of grid points as also shown by other works [2,28]. To evaluate the correction term $C\{\Delta\mathbf{u}\}$ of (4.9a) at an irregular point (i,j) as depicted in Fig. 3, we need to compute $[\mathbf{u}_x]$ and $[\mathbf{u}_{xx}]$ at the intersection point α of the interface/boundary with the grid lines, and $[\mathbf{u}_y]$ and $[\mathbf{u}_{yy}]$ at β of the interface/boundary with the grid lines. The correction term $C\{\Delta\mathbf{u}\}$ is calculated as follows:

$$C\{\Delta\mathbf{u}\}_{i,j} = -\frac{[\mathbf{u}] + h^+[\mathbf{u}_x]_\alpha + \frac{(h^+)^2}{2}[\mathbf{u}_{xx}]_\alpha}{h^2} - \frac{[\mathbf{u}] + k^-[\mathbf{u}_y]_\beta + \frac{(k^-)^2}{2}[\mathbf{u}_{yy}]_\beta}{h^2},$$

where $h^+ = x_{i+1} - x_\alpha, k^- = y_{j-1} - y_\beta$, and x_α and y_β are the x-coordinate of the intersection point α and the y-coordinate of the intersection point β as shown in Fig. 3, respectively. $\Delta\mathbf{u}$ is approximated at the irregular point (i,j) as

$$\Delta\mathbf{u}(i,j) = \Delta_h\mathbf{u}_{ij} + C\{\Delta\mathbf{u}\}_{i,j} + O(h).$$

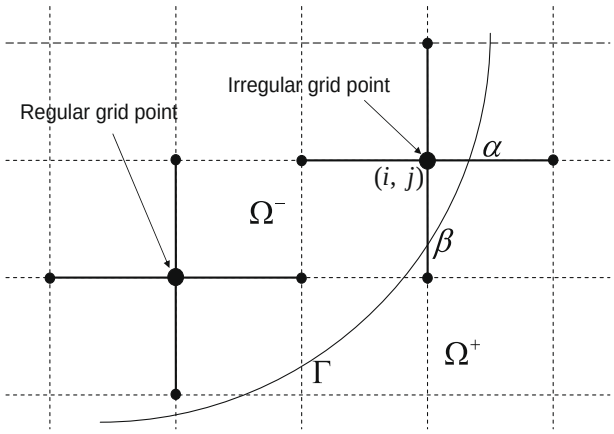


Fig. 3. Interface/boundary and mesh geometry near the irregular grid point (i,j).

Similarly, we can compute for the other correction terms in (4.9a) and (4.9b) as follows

$$C\{\nabla p\}_{i,j} = \left(-\frac{[p] + h^+[p_x]_\alpha + \frac{(h^+)^2}{2}[p_{xx}]_\alpha}{h}, \frac{[p] + k^-[p_y]_\beta + \frac{(k^-)^2}{2}[p_{yy}]_\beta}{h} \right),$$

$$C\{\nabla \cdot \mathbf{u}\}_{i,j} = -\frac{[u] + h^+[u_x]_\alpha + \frac{(h^+)^2}{2}[u_{xx}]_\alpha}{h} + \frac{[v] + k^-[v_y]_\beta + \frac{(k^-)^2}{2}[v_{yy}]_\beta}{h}.$$

Note above intersection points α and β at which the jumps for u, v and p correspond to those of the interface/boundary with the grid lines for u-mesh, v-mesh and p-mesh.

4.3. Imposing the rigid boundary condition

In the present work, the rigid boundary is immersed in a rectangular computational domain. The prescribed velocity condition at the rigid boundary is imposed by applying an appropriate singular force at the control points representing the rigid boundary. Assuming that the singular force $\mathbf{f}^{rb}$ at the rigid boundary is known, the velocity field $\mathbf{u}$ at all the grid points can be computed via the CG-Uzawa method as discussed in Section 4.1. The velocity at the control points of the rigid boundary, $\mathbf{U}_k^{rb,\mathbf{f}^{rb}}$ (since it depends on $\mathbf{f}^{rb}$ and can be taken as a function of $\mathbf{f}^{rb}$), can be interpolated from the velocity $\mathbf{u}$ at the grid points. Here, we can write

$$\mathbf{U}_k^{rb,\mathbf{f}^{rb}} = \mathbf{U}(\mathbf{X}_k^{rb}) = \mathcal{B}(\mathbf{u}), \tag{4.10}$$

where $\mathcal{B}$ is the modified bilinear interpolation operator with jump conditions to guarantee second-order accuracy when the derivatives of the velocity are discontinuous and its explicit form is shown in [24]. Since the relationships between the singular forces and the jumps in the solution or its derivatives are linear and all the discreted equations are linear, therefore we simply write the velocity at the control points of the rigid boundary as,

$$\mathbf{U}_k^{rb,\mathbf{f}^{rb}} = \mathbf{U}_k^{rb,0} + \mathbf{A}\mathbf{f}^{rb}, \tag{4.11}$$

where $\mathbf{U}_k^{rb,0}$ corresponds to the velocity at the control points of the rigid boundary obtained by solving Eqs. (2.1) and (2.2) with $\mathbf{f}^{rb} = 0$. $\mathbf{A}$ is a $2N_b \times 2N_b$ matrix, where N_b is the number of control points at the rigid boundary. The vector $\mathbf{A}\mathbf{f}^{rb}$ is the velocity at the control points of the rigid boundary obtained by solving the following equations:

$$\nabla_h p_{\mathbf{f}^{rb}} = \mu \Delta_h \mathbf{u}_{\mathbf{f}^{rb}} + \overline{\mathbf{C}}_1, \tag{4.12}$$

$$\nabla_h \cdot \mathbf{u}_{\mathbf{f}^{rb}} = \overline{C}_2, \quad \mathbf{u}_{\mathbf{f}^{rb}}|_{\partial\Omega} = 0, \tag{4.13}$$

$$\mathbf{A}\mathbf{f}^{rb} = \mathcal{B}(\mathbf{u}_{\mathbf{f}^{rb}}), \tag{4.14}$$

with $\mathbf{f}^{rb}$ being the singular force at the rigid boundary. Here, $\overline{\mathbf{C}}_1$ and $\overline{C}_2$ are the correction terms which only take into account the effect of the singular force $\mathbf{f}^{rb}$ at the rigid boundary. From Eq. (4.11), with the prescribed velocity $\mathbf{U}_p^{rb}$ at the rigid boundary, the singular force $\mathbf{f}^{rb}$ at the rigid boundary is determined by solving

$$\mathbf{A}\mathbf{f}^{rb} = \mathbf{U}_p^{rb} - \mathbf{U}_k^{rb,0}. \tag{4.15}$$

Eq. (4.15) can be solved via using the GMRES method [42]. Each GMRES iteration requires one vector–matrix product of $\mathbf{Af}^{rb}$ with a known $\mathbf{f}^{rb}$. Note that the matrix $\mathbf{A}$ depends on the location of the rigid boundary. In the present work, the rigid boundary is static, therefore, we can form the coefficient matrix $\mathbf{A}$ explicitly and solve Eq. (4.15) directly. In order to compute the coefficients of $\mathbf{A}$, we solve Eqs. (4.12)–(4.14) for $2N_b$ times, i.e. once for each column. Each time, the singular force $\mathbf{f}^{rb}$ is set to zero except for the entry corresponding to the column we want to calculate, which is set to one. Once the matrix $\mathbf{A}$ has been calculated, the terms on the right hand side, $\mathbf{U}_p^{rb} - \mathbf{U}_k^{rb,0}$, can be computed. The resulting small system of Eq. (4.15) is then solved for $\mathbf{f}^{rb}$ via back substitution. Finally, we solve Eqs. (4.1) and (4.2) to obtain $\mathbf{u}$ and p. In actual computation, we use the LU method to solve the system of Eq. (4.15) for flows involving moving interfaces and irregular domains.

4.4. Advancing the deformable interface

For the deformable interface problem, the location of the interface $\mathbf{X}^{ib}$ is updated based on the surrounding fluid velocity. To overcome the strict limit of very small time steps with explicit method and increase the stability of the current method, the updated location of the deformable interface is advanced in time in an implicit manner, according to

$$\mathbf{X}^{ib,n+1} = \mathbf{X}^{ib,n} + \frac{1}{2}\Delta t\Big(\mathbf{u}^n(\mathbf{X}^{ib,n}) + \mathbf{u}^{n+1}(\mathbf{X}^{ib,n+1})\Big). \tag{4.16}$$

The new positions of the control points $\mathbf{X}^{ib,n+1}$ are determined by solving a nonlinear system of equations

$$Q(\mathbf{X}^{ib,n+1}) = 0, \tag{4.17}$$

where

$$Q(\mathbf{X}) = \mathbf{X} - \mathbf{X}^{ib,n} - \frac{1}{2}\Delta t\Big(\mathbf{u}^n(\mathbf{X}^{ib,n}) + \mathbf{u}^{n+1}(\mathbf{X})\Big).$$

The BFGS method [45] which is a quasi-Newton method is employed to solve the nonlinear system of Eq. (4.16) iteratively to calculate the location of the deformable interface. For more details on the IIM for deformable interfaces, see [26,24,25]. In each iteration of the BFGS method, we need to solve the system of Eq. (4.15) for the singular force $\mathbf{f}^{rb}$ to impose the prescribed velocity at the rigid boundary thereby ensuring the boundary condition for the velocity is exactly satisfied. This is necessary because the velocity field and pressure field are updated at every iterations of the BFGS method. In the numerical tests, it takes only a few iterations for the BFGS method.

5. Numerical implementation

In this section, we describe a basic implementation of our algorithm for the steady Stokes equations with fixed/moving interfaces and rigid boundaries. To start our procedure we compute the coefficient matrix as mentioned in the previous section. We then factorize the coefficient matrix using LU decomposition and store the $\mathbf{L}$ and $\mathbf{U}$ matrices. First, given the singular force $\mathbf{f}^{ib}$ at the fixed interface, our algorithm for finding $\mathbf{u}$, p and the singular force $\mathbf{f}^{rb}$ at the rigid boundary to satisfy the prescribed velocity condition at the rigid boundary can be summarized as follows:

Algorithm 1 (*IIM with fixed interfaces on irregular domains*).

Step 1: Compute the right hand side of (4.15) by calculating $\mathbf{U}_p^{rb} - \mathbf{U}_k^{rb,0}$.
- Set $\mathbf{f}^{rb} = 0$, and solve (4.1) and (4.2) for the velocity at all the grid points, with incorporation of the correction terms which only take into account the contribution of $\mathbf{f}^{ib}$ at the fixed interface.
- Interpolate the velocity at the control points $\mathbf{U}_k^{rb,0}$ as in (4.10).
- Compute the right hand side vector $\mathbf{b} = \mathbf{U}_p^{rb} - \mathbf{U}_k^{rb,0}$.

Step 2: Compute the singular force $\mathbf{f}^{rb}$ by solving (4.15) using the LU method or the GMRES method.

Step 3: Compute $\mathbf{u}$ and p using the CG-Uzawa method with the incorporation of correction terms which take into account the contributions of both $\mathbf{f}^{ib}$ at the fixed interface and $\mathbf{f}^{rb}$ at the rigid boundary.

Next we turn our attention to the implementation of the immersed interface method for the incompressible Stokes equations with moving interfaces and rigid boundaries. Given the location of the control points $\mathbf{X}^{\mathbf{n}}$, the velocity $\mathbf{u}^n$ and pressure p^n, the algorithm for computing the velocity $\mathbf{u}^{n+1}$ that satisfies the prescribed velocity condition at the rigid boundary, pressure p^{n+1} and the location of the control points $\mathbf{X}^{n+1}$ can be described as follows:

Algorithm 2 (*IIM with moving interfaces on irregular domains*).

Step 1: Set $k := 0$, make an initial guess for $\mathbf{X}^{ib,n+1}$, i.e. $\mathbf{X}^{ib,(0)}$ as $\mathbf{X}^{ib,(0)} = 2\mathbf{X}^{ib,n} - \mathbf{X}^{ib,n-1}$ and set the inverse Jacobian $\mathbf{B}_0^{n+1} = \mathbf{B}_k^n$. At the first time step, the inverse Jacobian is initialized to the identity matrix $\mathbf{I}$.

Step 2:

- Compute the force strength $\mathbf{f}^{ib}$ at the deformable interface using expression (2.7).
- Compute the force strengths $\mathbf{f}^{rb}$ at the rigid boundary to impose the prescribed velocity condition at the rigid boundary, i.e., calculate the right hand side vector of (4.15), and then solve for the small system of Eq. (4.15) to obtain the singular force $\mathbf{f}^{rb}$ at the rigid boundary as in Algorithm 1.

Step 3:

- Employ the CG-Uzawa method as described in Section 4.1 to obtain the velocity field $\mathbf{u}^{n+1}$ and pressure field p^{n+1}. This step involves computing the appropriate correction terms for the spatial derivatives as described in Section 4.2.
- Compute the velocity $\mathbf{u}^{n+1}(\mathbf{X}^{ib,(k)})$ at control points $\mathbf{X}^{ib,(k)}$, which is interpolated from the velocity $\mathbf{u}^{n+1}$ at the surrounding grid points.

Step 4:

- Evaluate $Q(\mathbf{X}^{ib,(k)})$.
- If $\|Q^{(k)}\| < \varepsilon$ then $\mathbf{X}^{ib,n+1} = \mathbf{X}^{ib,(k)}$ and stop the iteration. Otherwise, update $\mathbf{X}^{ib,(k+1)}$ and the inverse Jacobian matrix B_{k+1}^{n+1} [45]. Set $k = k+1$ and go to **Step 2**.

6. Numerical examples

In this section, several numerical examples are carried out to demonstrate the capabilities of our proposed algorithm in this work. All the simulations are done on a Laptop PC with 1.6 GHz.

Example 6.1. In the first example, we start our numerical tests by checking the accuracy of the algorithm. In this example, there is a provided exact solution [29] and the exact velocity and pressure are given by

$$u = \begin{cases} \frac{y}{4}, & x^2+y^2 < 1, \\ \frac{y}{4}(x^2+y^2), & x^2+y^2 \geqslant 1, \end{cases} \tag{6.1}$$

$$v = \begin{cases} -\frac{x}{4}(1-x^2), & x^2+y^2 < 1, \\ -\frac{xy^2}{4}, & x^2+y^2 \geqslant 1, \end{cases} \tag{6.2}$$

$$\tilde{p} = \begin{cases} (-\frac{3}{4}x^3+\frac{3}{8}x)y, & x^2+y^2 < 1, \\ 0, & x^2+y^2 \geqslant 1, \end{cases} \tag{6.3}$$

$$p = \tilde{p} - mean(\tilde{p}), \tag{6.4}$$

where $mean(\tilde{p})$ is the average of $\tilde{p}$. The external force term $\mathbf{g} = (g_1, g_2)^T$ is derived directly from the exact solution by satisfying the Stokes equation as follows

$$g_1 = \begin{cases} (-\frac{9}{4}x^2+\frac{3}{8})y, & x^2+y^2 < 1, \\ -2\mu y, & x^2+y^2 \geqslant 1, \end{cases} \tag{6.5}$$

$$g_2 = \begin{cases} -\frac{3}{4}x^3+\frac{3}{8}x-\frac{3\mu}{2}x, & x^2+y^2 < 1, \\ \frac{\mu}{2}x, & x^2+y^2 \geqslant 1. \end{cases} \tag{6.6}$$

Thus, external force term $\mathbf{g}$ has a finite jump across the interface. And the singular force terms in the normal direction and tangential direction are

$$\hat{f}_1 = \left(\frac{3}{4}\cos^3\theta - \frac{3}{8}\cos\theta\right)\sin\theta, \tag{6.7}$$

$$\hat{f}_2 = \frac{1}{2}\mu \tag{6.8}$$

calculated from (3.3) and (3.5), respectively, where θ is the angle between the x-axis and the normal direction at the point of the interface. It is easy to verify that the velocity satisfies the incompressibility constraint, and it is continuous but has a finite jump in the normal derivative across the interface [29].

Here, we use the circular geometry with the radius $R_s = 2$. The prescribed velocity at the boundary of the circular geometry (i.e., rigid boundary) is found by exact solution. The simulation is performed with a 64×64 grid and $\mu = 0.1$. The outer rigid boundary and inner interface are presented by 64 control points and 40 control points, respectively. In Fig. 4, we present the plot for the computed u-component velocity. We perform the grid refinement analysis to determine the order of convergence of the algorithm. The order of accuracy is estimated as

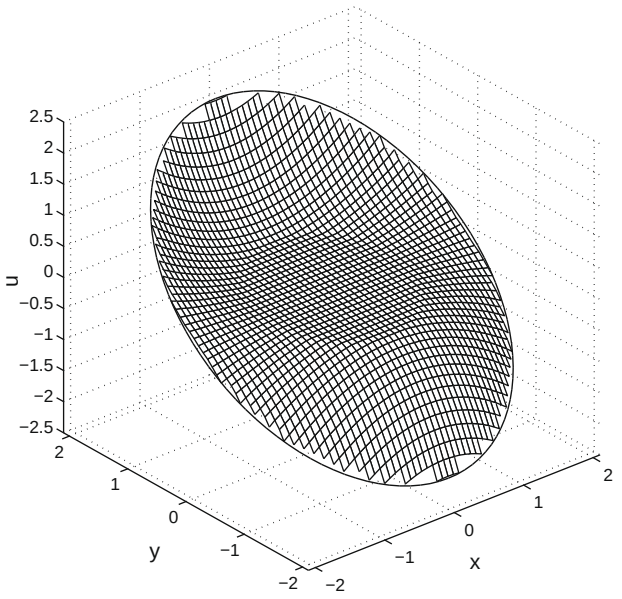


Fig. 4. For Example 6.1. The x-component of velocity field $\mathbf{u}$.

$$\text{order} = \frac{\log(\|E_u(N)\|_\infty / \|E_u(2N)\|_\infty}{\log 2}. \tag{6.9}$$

Here, $\|E_u(N)\|_\infty$ is the maximum error

$$\|E_u(N)\|_\infty = \max_{i,j} |U_{ij} - u(x_i, y_j)|, \tag{6.10}$$

where $u(x_i, x_j)$ is the exact solution at (x_i, x_j) and U_{ij} is the numerical solution.

The result of the convergence rate analysis is shown in Table 1. From Table 1, one can easily see that the velocity is second-order accurate, and the pressure is nearly second-order accurate. The sixth column shows the number of average CG iterations, which indicates that a limited number of iterations are needed and the number of iterations is almost independent of the mesh size. The CPU time in seconds is listed in the last column, which shows that the present method is efficient. The corresponding condition number of matrix $\mathcal{M}$ in Eqs. (4.1) and (4.2) with homogeneous Dirichlet boundary conditions is 9.08 ($N = 512$).

Example 6.2 (*Rotational flow on irregular domain*). In the second example, we consider the rotational flow problem involving a fixed interface (where forces are prescribed) and rigid boundary (where velocity is prescribed). In this example, we use a 64×64 grid on a computation domain of $[-1,1] \times [-1,1]$ and set $\mu = 0.1$ unless it is stated otherwise.

We first consider the external irregular domain as in Fig. 5(a). In this case, the rigid boundary ∂D is a circle with radius $R_s = 0.7$; the involved fixed interface is also a circle but with radius $r = 0.3$, which is located at the center of the circular domain. We set 48 and 40 control points on the rigid boundary and inner interface in the simulation, respectively. Along the inner interface, the normal and tangential forces are $f_1 = 0$ and $f_2 = -10\mu$, respectively. At the rigid boundary, we first prescribe the rigid boundary to rotate with a constant angular velocity $\omega = 1$. The x-component of the velocity field $\mathbf{u}$ and velocity field $\mathbf{u}$ are shown in Fig. 6(a) and (b), respectively. The steady-state motion is a anti-clockwise rotation along the outer rigid boundary and a clockwise rotation along the inner interface. Next we set the no-slip boundary conditions at the rigid boundary, i.e., $\omega = 0$. The steady solution is shown in Fig. 7, which corresponds to a clockwise rigid body motion inside the interface. Fig. 7(a) and (b) show the x-component of the velocity field $\mathbf{u}$ and velocity field $\mathbf{u}$, respectively. From these figures, we can observe that the velocity is continuous but not smooth across the interface due to the singular forces at the interface as expected.

Table 1
Grid refinement analysis for Example 6.1.

N	$\|E_\mathbf{u}\|_\infty$	Order	$\|E_p\|_\infty$	Order	N_{iter}	CPU (s)
32	8.8190E−03	–	4.6174E−02	–	11	0.66
64	2.0429E−03	2.11	9.3114E−03	2.31	12	1.77
128	5.3611E−04	1.93	2.9261E−03	1.67	13	6.93
256	1.2768E−04	2.07	8.5203E−04	1.78	13	33.23

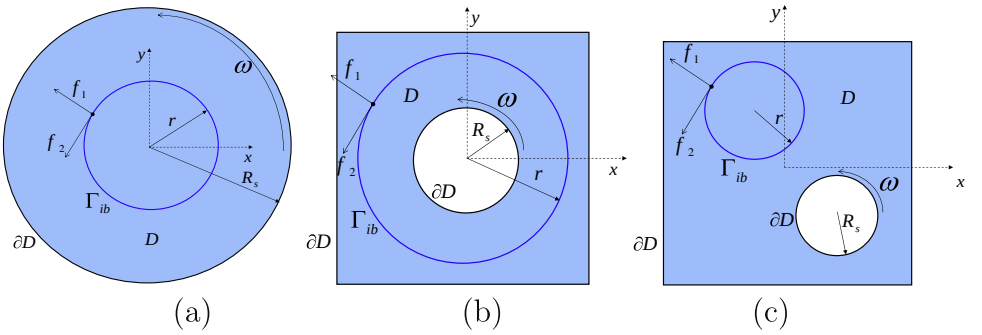


Fig. 5. The domain of the simulation and the geometry of rotational flow.

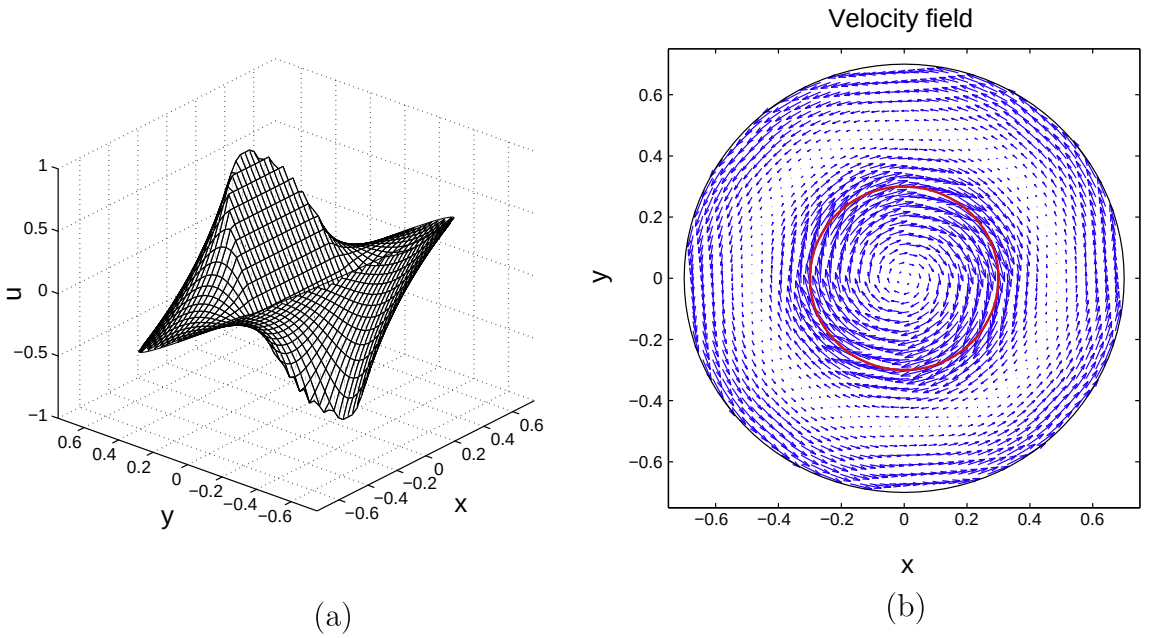


Fig. 6. For Example 6.2. (a) The x-component of the velocity field $\mathbf{u}$, and (b) velocity field $\mathbf{u}$ with $\omega = 1, \mu = 0.1, f_1 = 0$ and $f_2 = -10\mu$.

In the second case, we consider the internal irregular domain as in Fig. 5(b). The rigid boundary is inside the interface as shown in Fig. 5(b); the rigid boundary ∂D is a circle with radius $R_s = 0.3$ while the involved fixed interface is a circle with radius $r = 0.7$, located at the center of the square domain. We use 40 and 48 control points to represent the rigid boundary and outer interface, respectively. Along the outer interface, the normal and tangential forces are $f_1 = 0$ and $f_2 = -5\mu$, respectively. At the rigid boundary, we first set the no-slip boundary conditions. Fig. 8(a) and (b) show the x-component of the velocity field $\mathbf{u}$ and velocity field, respectively. The motion of the steady solution is a simple clockwise rotation along the outer interface. Next we prescribe the inner rigid boundary to rotate with a angular velocity $\omega = 2$. The x-component of the velocity field $\mathbf{u}$ and velocity field are shown in Fig. 9(a) and (b), respectively. The motion of the steady solution is a clockwise rotation along the outer interface and an anti-wise rotation along the inner boundary.

In the third case that the rigid boundary is outside the interface as shown in Fig. 5 (c), the rigid boundary ∂D is a circle with radius $R_s = 0.3$ and its center at $(0.4, -0.4)$; while the involved fixed interface is a circle with radius $r = 0.5$, which is located at (-0.3, 0.3). We use 40 and 48 control points to represent the rigid boundary and interface, respectively. Along the interface, the normal and tangential forces are $f_1 = 0$ and $f_2 = -5\mu$, respectively. At the rigid boundary, we set the no-slip boundary conditions. Fig. 10(a) and (b) show the x-component of the velocity $\mathbf{u}$ and velocity field, respectively. The motion of the steady solution is a simple clockwise rotation along the interface. Again we can observe from these figures that the velocity is continuous but not smooth across the interface as expected due to the singular forces at the interface. Finally, we carry out a grid refinement analysis for Fig. 5(a) with $\omega = 1$, using a referenced grid of 512×512, to determine the order of the convergence of the algorithm. The results in Table 2 indicate that the velocity is second-order accurate and the pressure is nearly second-order accurate. We can also see from this table that the present method for this example requires a

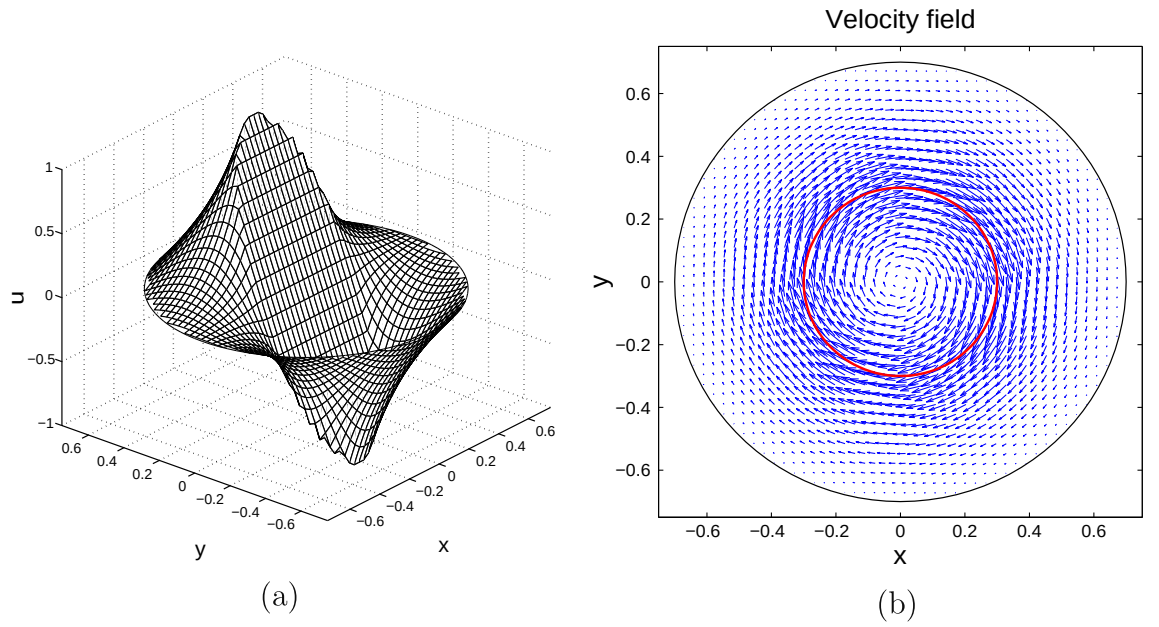


Fig. 7. For Example 6.2. (a) The x-component of the velocity field $\mathbf{u}$, and (b) velocity field $\mathbf{u}$ with $\omega = 0, \mu = 0.1, f_1 = 0$ and $f_2 = -10\mu$.

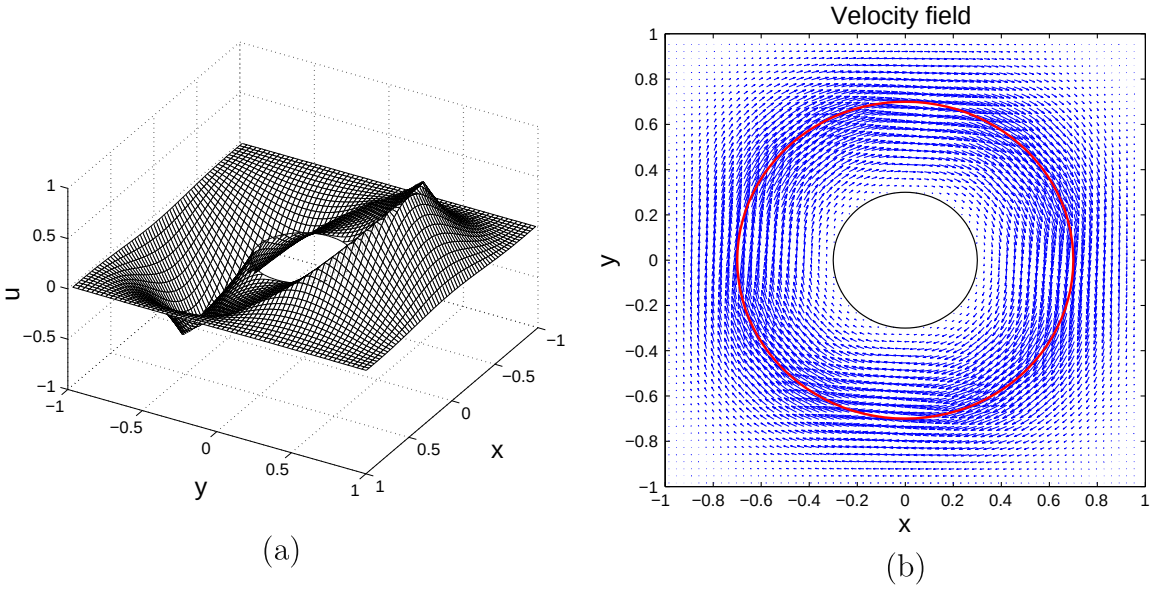


Fig. 8. For Example 6.2. (a) The x-component of velocity field $\mathbf{u}$, and (b) the velocity field $\mathbf{u}$ with $\omega = 0$ and $\mu = 0.1, f_1 = 0$ and $f_2 = -5\mu$.

small number of iterations and is almost independent of mesh size, and therefore it is very fast in terms of the CPU time. Again the corresponding condition number of matrix $\mathcal{M}$ in Eqs. (4.1) and (4.2) is 9.08 with no-slip boundary conditions for this case ($N = 512$).

Example 6.3 (*Relaxation of elastic membrane on irregular domain*). In the third example, we consider a deformable interface problem which involves an elastic membrane on the irregular domain. On the regular domain, this problem has been already used by Tu and Peskin [50] to test their immersed boundary method, by LeVeque and Li [26] to test their IIM for Stokes flows, and by Lee and LeVeque [25] to test the IIM for Navier–Stokes equations. For our problem to be studied, the fluid domain D considered is irregular as in Fig. 11 where the rigid boundary is a circle with radius $R_s = 1$. The initial state of membrane (the solid line in Fig. 11, labeled "Initial") is an ellipse with the semi-major and semi-minor axes $a = 0.75, b = 0.5$, respectively, and the ellipse is located at the center of the circular domain. The unstretched state of membrane (the dashed line in Fig. 11, labeled "Resting") is a circle with radius $r_0 = 0.5$. The tension coefficient T_0 is set to 1 in this example.

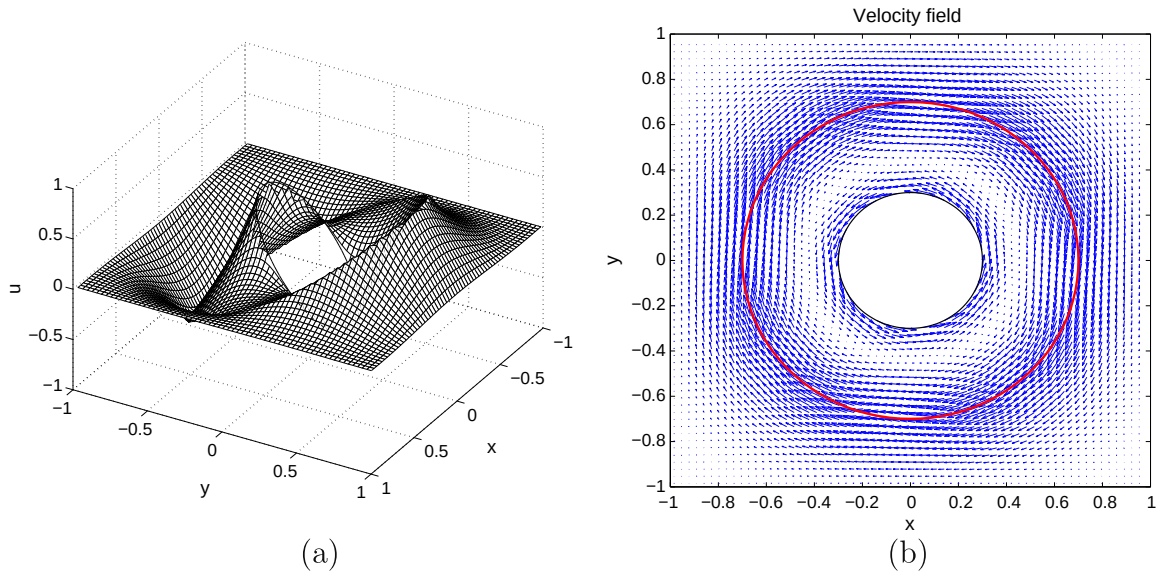


Fig. 9. For Example 6.2. (a) The x-component of velocity field $\mathbf{u}$, and (b) the velocity field $\mathbf{u}$ with $\omega = 2$ and $\mu = 0.1, f_1 = 0$ and $f_2 = -5\mu$.

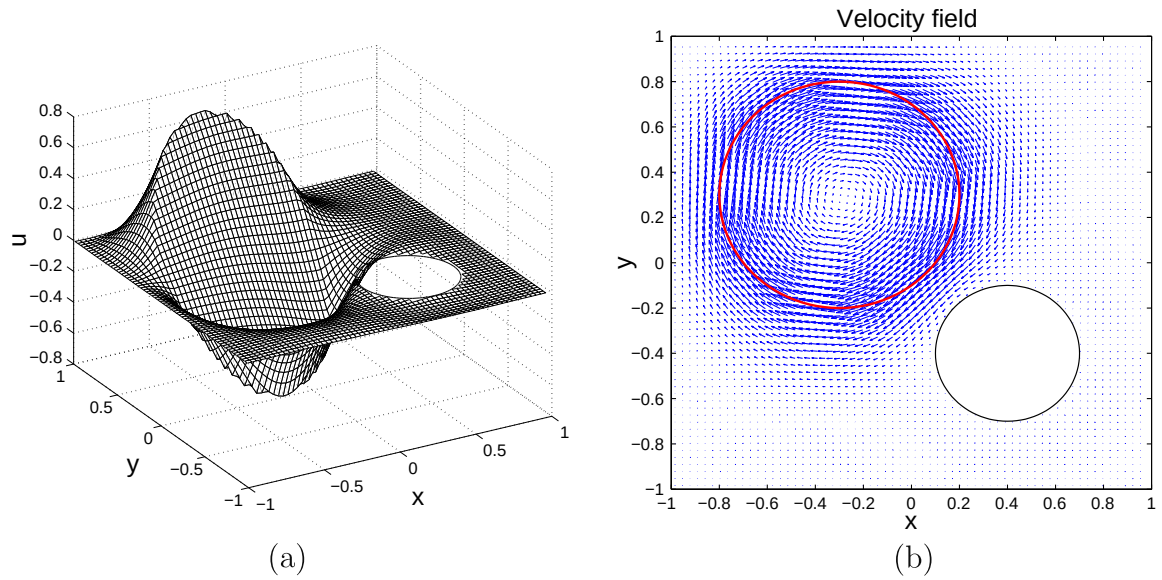


Fig. 10. For Example 6.2. (a) The x-component of the velocity field $\mathbf{u}$, and (b) velocity field $\mathbf{u}$ with $\omega = 0$ and $\mu = 0.1, f_1 = 0$ and $f_2 = -5\mu$.

Table 2
Convergence analysis for the first case of Example 6.2 with $\omega = 1, \mu = 0.1$.

N	$\|E_{\mathbf{u}}\|_\infty$	Order	$\|E_p\|_\infty$	Order	N_{iter}	CPU (s)
64	2.1032E−03	–	3.6211E−03	–	8	0.94
128	5.1142E−04	2.04	1.0691E−03	1.76	9	3.65
256	1.2697E−04	2.01	3.3134E−04	1.69	10	14.70

Due to the restoring force, the ellipse will converge to a circle (the dash-dot line in Fig. 11, labeled "Equilibrium") with radius $r_e = \sqrt{ab} \approx 0.61237$, which is larger than the unstretched interface but has the same area as the initial ellipse because of the incompressibility of the enclosed fluid. So the interface is still stretched at the equilibrium state. The prescribed veloc-

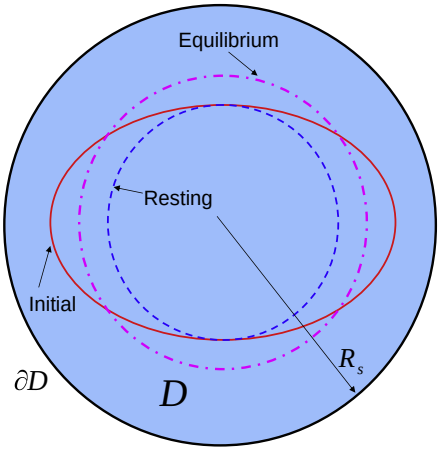


Fig. 11. The interface configurations at different states in a circular domain.

ity at the rigid boundary is set to zero in all the simulations for this example, i.e, $\mathbf{u}|_{\partial D} = \mathbf{0}$, unless otherwise stated. In this test, we perform the simulations with a 64×64 grid on a computational domain of $[-1.2, 1.2] \times [-1.2, 1.2]$, where 40 control points and 64 control points are used to represent the interface and rigid boundary.

We begin by computing the velocity and pressure at time $t = 0$ based on the initial elliptical interface, before the interface has moved. In Fig. 12(a) and (b), we show the x-component of the velocity field $\mathbf{u}$ and velocity field at $t = 0$, respectively. The pressure distributions at $t = 0$ and $t = 1$ are presented in Fig. 13(a) and in (b), respectively. As expected, from these figures, we can see that the velocity u is continuous but not smooth, while the pressure p are discontinuous across the interface. Fig. 14 shows this more clearly with the plot of cross section of u along the line $y = -0.431$ and the plots of cross section of p along the line $y = -0.019$ at $t = 0$ and $t = 40$. From Figs. 13 and 14(b) and (c), we can see clearly the discontinuities in the pressure are captured very sharply by our immerse interface method. The evolution of the semi-major and semi-minor axes with time is shown in Fig. 15. The interface relaxes gradually to the equilibrium state without oscillations and this equilibrium is then maintained as expected. With our method, the numerical equilibrium agrees very well with the true equilibrium. For example, at $t = 40$, the error between semi-major axis and the true equilibrium position r_e is only 1.0622e−5, and the error between semi-minor axis and r_e is only 9.4631e−6.

We also perform convergence analysis for the flow field. Since the analytic solution is not available, we measure the error in velocity and pressure using a reference solution that is obtained on the finest 512×512 grid. In Table 3, we show the convergence rate analysis at $t = 0$, and the expected second-order accuracy for the velocity and near second-order accuracy for the pressure are observed.

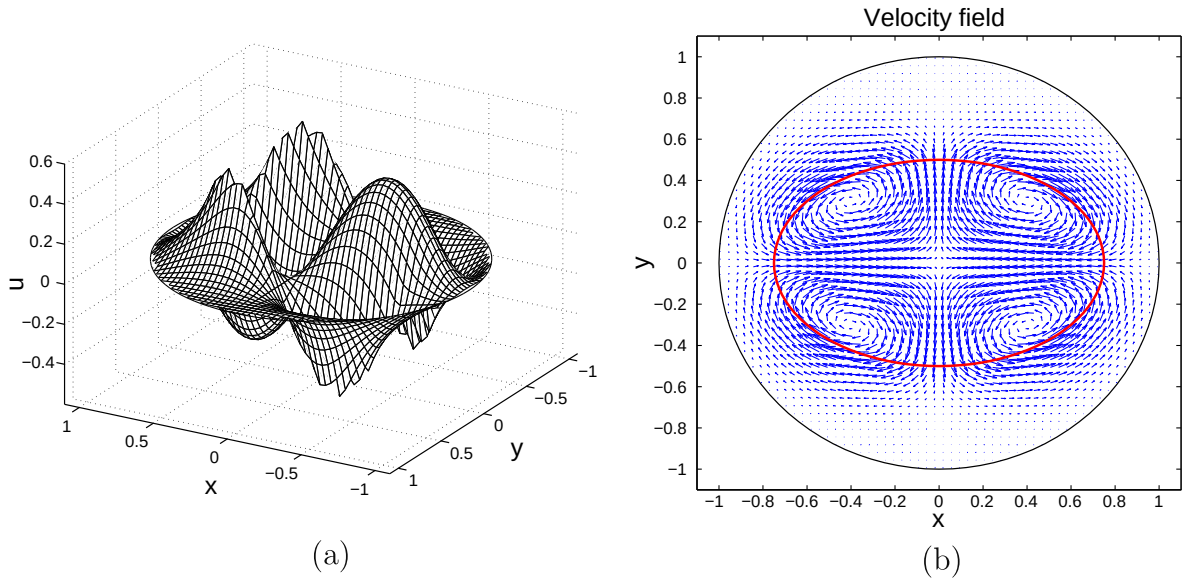


Fig. 12. For Example 6.3. (a) The x-component of the velocity field $\mathbf{u}$, and (b) velocity field $\mathbf{u}$ at $t = 0$ with $\mu = 0.1$ and $T_0 = 1$.

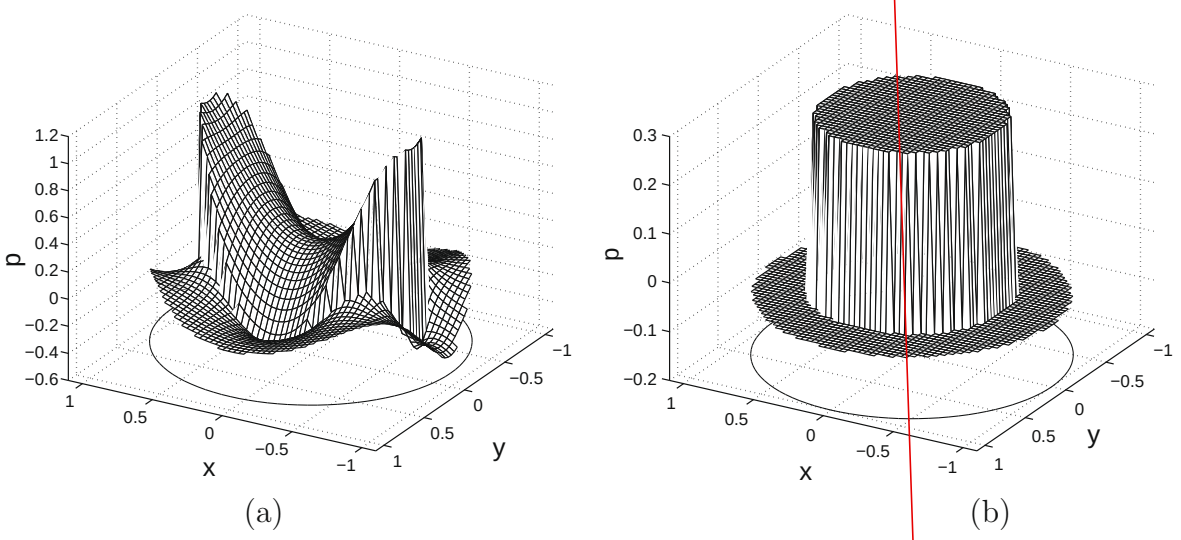


Fig. 13. For Example 6.3. The pressure distribution at $t = 0$ (a) and $t = 1$ (b) with $\mu = 0.1$ and $T_0 = 1$.

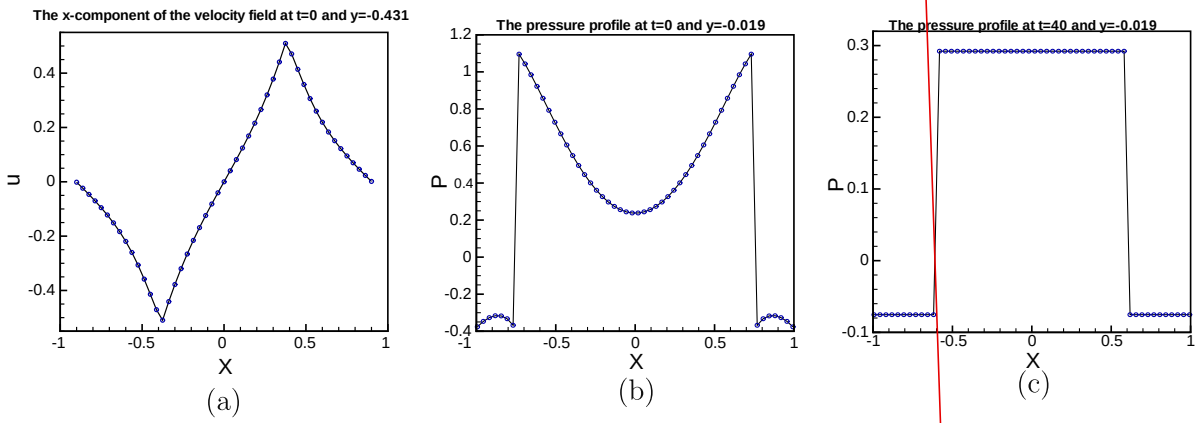


Fig. 14. For Example 6.3. (a) A slice of the u-component velocity along $y = -0.431$ and the pressure profile along $y = -0.019$ at (b) $t = 0$, and (c) $t = 40$ with $\mu = 0.1$ and $T_0 = 1$.

In the simulation, the conservation of the area is also found to be preserved very well. In Fig. 16(a), we present the plot of the absolute error in area versus time up to $t = 40$. In this figure, the maximum absolute error in area is 9.0591e−5 and

Table 3
Convergence analysis for Example 6.4 at $t = 0$ with $\mu = 0.1$ and $T_0 = 1$.

N	$\|E_{\mathbf{u}}\|_\infty$	Order	$\|E_p\|_\infty$	Order
32	2.9905E−03	–	2.5633E−02	–
64	7.0241E−04	2.09	4.8903E−03	2.39
128	1.7929E−04	1.97	1.5582E−03	1.65
256	4.3901E−05	2.03	4.6648E−04	1.74

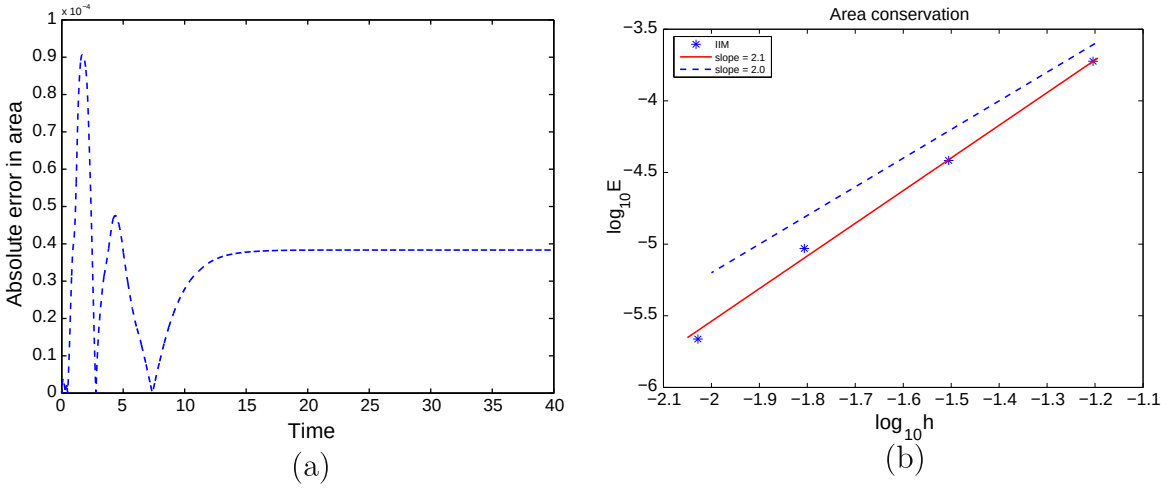


Fig. 16. For Example 6.3. (a) Plot of the absolute error in area versus time, and (b) grid refinement analysis for studying the conservation of the area enclosed by elastic membrane at the steady-state with $\mu = 0.1$ and $T_0 = 1$.

indicates fairly little leakage of about only 0.0077%. In Fig. 16(b), we perform a grid refinement analysis to study the conservation of the area enclosed by the elastic membrane. It could be seen from this figure that the area is conserved with second-order accuracy. Next, we also want to briefly discuss about the iterations in BFGS for this problem. In Fig. 17(b), we present the plot of the number of BFGS iterations versus time. From this figure, we can see that it takes only a few iterations at most time level, say about 1–3, to converge with a tolerance of 10^{-10} for the BFGS.

It is interesting to test further whether our method can handle flow on irregular domain with a more complicated initial interface inside such as the flower-shaped configuration found in the literature [26] employed to test for a stiff problem. In this test, the initial interface is given by $r(\theta) = 0.8 + 0.3\sin(7\theta)$ in polar coordinates in the circular domain with radius

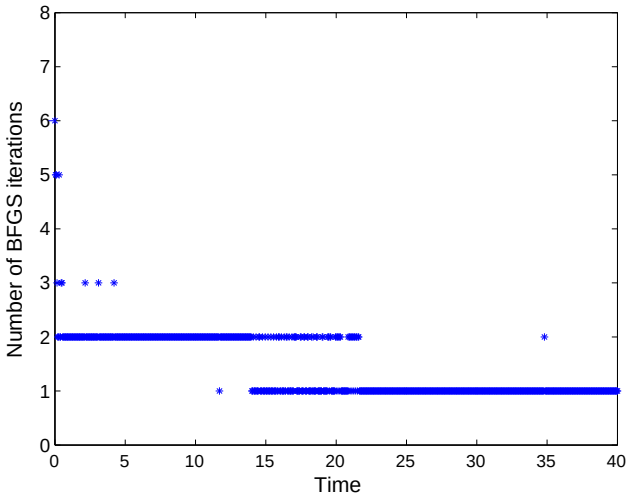


Fig. 17. Number of BFGS iterations with $\mu = 0.1$ and $T_0 = 1$.

$R_s = 1.3$. The unstretched interface is a circle with the radius $r_0 = 0.3$. The initial interface and unstretched interface are shown in Fig. 18 (solid line and dashdotdot line, respectively). In the computations, we use a 128×128 grid on a computation domain of $[-1.5, 1.5] \times [-1.5, 1.5]$ and set $\mu = 0.1$. We use 40 control points and 128 control points to represent the rigid boundary and interface, respectively. The tension coefficient T_0 is set to 0.5. In Fig. 18, we plot the configurations of interface at $t = 0, t = 0.05, t = 0.1$ and $t = 1.02$. At $t = 1.02$, the interface is almost in the equilibrium state depicted as a circle as in Fig. 18 (dotted line). The velocity field and pressure contour at $t = 0.1$ are plotted in Fig. 19. It is clear from Fig. 19(b) that our method can capture the highly localized discontinuous profile for the pressure.

Next, to demonstrate the capability of the present method to simulate Stokes flows with multiple interfaces and connected domains, the simulation is first applied to capture the relaxation evolution of an elastic membrane between two stationary eccentric cylinders. The inner and outer cylinders have radius $R_I = 0.3$ and $R_O = 1$, respectively. Their centers are at $(0,0)$ and $(0,0.4)$, respectively. The initial state of membrane is an ellipse with the semi-major and semi-minor axes $a = 0.5, b = 0.3$, respectively, and the ellipse center is located at $(0, -0.4)$. The unstretched state of membrane is a circle with radius $r_0 = 0.3$. The tension coefficient T_0 is set to 1. In the computations, we use a 80×80 grid on a computational domain of $[-1.2, 1.2] \times [-1.2, 1.2]$ and set $\mu = 0.1$. We use 40 and 64 control points to represent the inner/outer cylinder and interface, respectively. The velocity and pressure distributions at $t = 0.4$ and $t = 10$ are presented in Fig. 20. Our code is able to perform the simulation effectively.

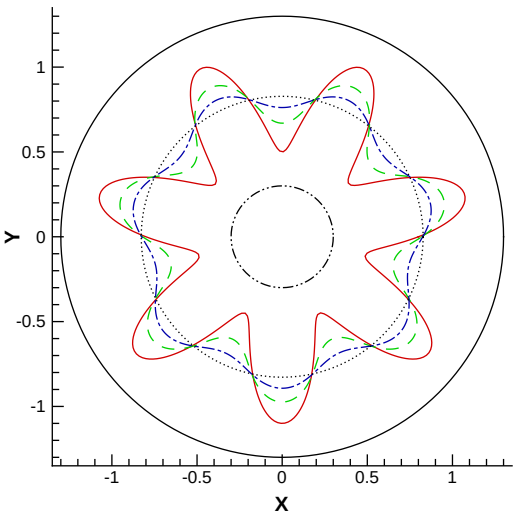
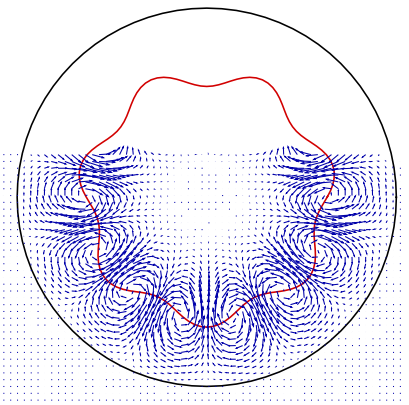
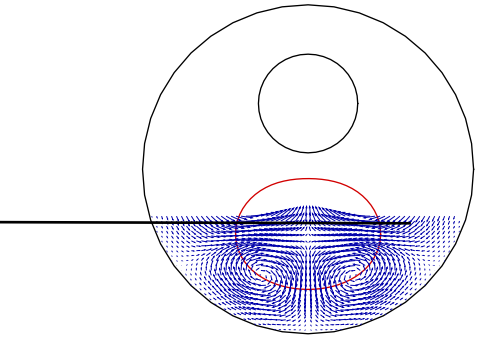


Fig. 18. For Example 6.3. The configurations of the interface at $t = 0$ (solid line), $t = 0.05$ (dashed line), $t = 0.1$ (dashdoted line) and $t = 1.02$ (dotted line), with $\mu = 0.1$ and $T_0 = 0.5$. The innermost dashdotdot circle shows the unstretched interface.





We shall in the following consider a moving interface problem on the square domain $[-1.5,1.5]\times[-1.5,1.5]$ with a circular hole and multiple interfaces consisting initially of four ellipses. The circular hole is located at the center of the domain and has radius $R_s = 0.3$. Each ellipse has same initial configuration with the semi-major and semi-minor axes $a = 0.5, b = 0.3$, respectively. The center of upper right ellipse is located at $(0.7,0.7)$, other ellipses are aligned symmetrically with each other along the line $y = x$ or $y = -x$. In Fig. 21, we show the velocity fields and pressure distributions at $t = 0.15$ and $t = 10$. As the four ellipses are symmetrically placed initially, we observe good preservation of the symmetries in the flow when the ellipses relax to form circles. Equilibrium is reached when the pressure is uniformly distributed. From Fig. 20(b) and (c) and Fig. 21(b) and (c), we can see clearly that the sharp profiles for the pressure are well captured by our method.

Example 6.4 (*Deformation of a drop between two cylinders*). In the fourth example, we shall study the motion and deformation of a liquid drop between two-dimensional Stokes flow confined in the two rotating cylinders. These flows are already studied in [18]. The set-up of the problem is as follows. The annular flow domain D is shown in Fig. 22, where ∂D represents the boundary of the mixer geometry. The radius of the inner cylinder and the radius of outer cylinder are R_I and R_O, respectively. The cylinders are concentric and their centers are at $(0,0)$. The constant angular velocities along the inner and outer cylinders are denoted by ω_I and ω_O, respectively. The signs of ω_I and ω_O indicate their respective senses of rotation (e.g. positive sign means anti-clockwise rotation). In this simulation considered, the two cylinders rotate in opposite directions. A circular viscous drop with radius r is placed at (x_0, y_0), and the boundary of the drop is represented by Γ_{ib}. We refer the readers to Fig. 22 for a sketch of the problem.

In the computations, the parameters $R_I = 0.5, R_O = 1, r = 0.125$, and $(x_0, y_0) = (0.71, 0)$ are taken in this example. We use a 128×128 grid on a computational domain of $[-1.2,1.2]\times[-1.2,1.2]$ and set $\mu = 0.1$. We use 40 control points to represent both the inner and outer circular cylinders and 48 control points to represent the boundary of the drop, respectively. We first take $\omega_I = 1$ and $\omega_O = -0.5$, that is, the inner cylinder rotates at a constant angular velocity of 1 in the anti-clockwise

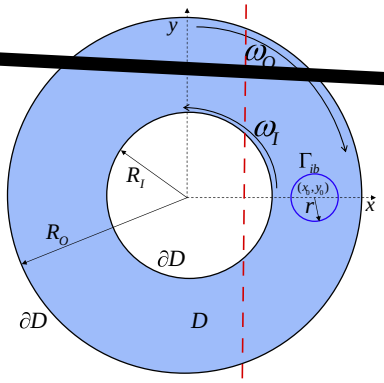
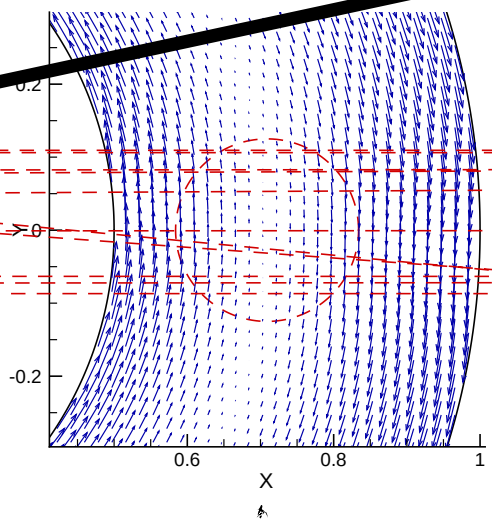


Fig. 22. Sketch of the deformation of a drop between two concentrically rotating cylinders.



ω_I ω_O

direction; whereas the outer cylinder rotates at a constant angular velocity of 0.5 in the clockwise direction. Fig. 23 shows the steady velocity field and streamlines around the referenced circular drop (the dashed line in Fig. 23) between the two concentrically rotating cylinders when no drop is involved (i.e., a fictitious drop). It can be seen from Fig. 23(a) that a stationary layer is located at some points in the gap due to two cylinders's rotating in opposite directions. That the streamlines are distributed concentrically is also observed in Fig. 23(b).

Next we want to study the deformation of drop placed between the cylinders by first taking $\omega_I = 1, \omega_O = -0.5$ and tension coefficient $\gamma = 1$. Fig. 24 shows the shapes of deformed drop at a time series of t = 0, 0.25, 0.5, 1.0 and 2.0, with a magnified view provided depicting more clearly the evolution or the surfaces. We see from Fig. 24(a) that the circular drop has only small deformation in this case. The flow structure around and inside the drop is of interest in this example. We show the corresponding plots of streamlines at time $t = 0, t = 0.5$ and $t = 2.0$ in Fig. 25 for this case, where the bold closed solid line represents the configuration of the drop. The flow features observed are fairly similar due to small deformation. However, on comparing the flow patterns with that in Fig. 23, we easily see from Fig. 25 that the flow patterns around and inside the referenced circular drop in Fig. 23 have been modified by the presence of the drop. Single eddies formed inside the drop and recirculating regions are observed in this figure. The drop rotates round the liquid inside and the fluid flow inside the drop moves in the clockwise direction. In Fig. 25, a closed streamline is nearly aligned with the drop interface, which indicates a nearly steady-state shape of drop is achieved at this moment.

Next, we keep the angular velocities of the inner and outer cylinders but reduce the tension coefficient to $\gamma = 0.1$. The pressure field is presented in Fig. 26(a). As expected, from this figure, we can see that the pressure is discontinuous across the drop interface, whereas the discontinuity in the pressure is still captured very sharply by our method. In Fig. 26(b), we

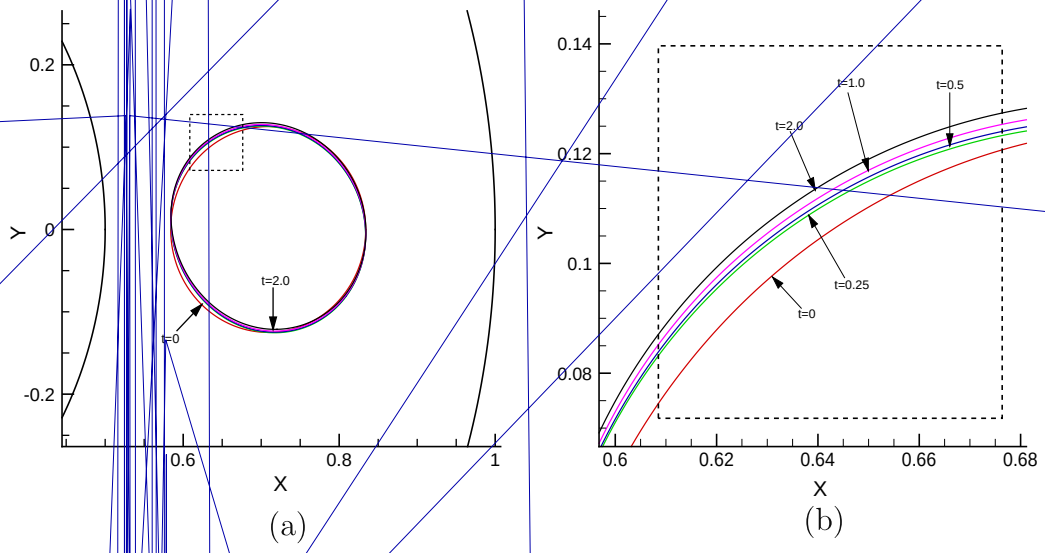
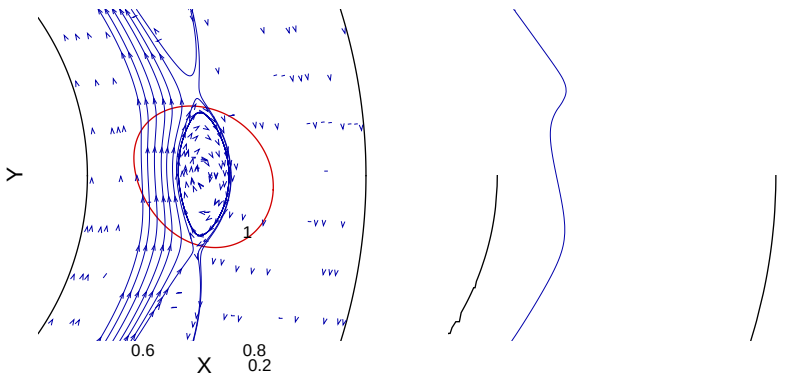


Fig. 24. For Example 6.4. (a) Configurations of deformed drops with circular initial shape at time t = 0, 0.25, 0.5, 1.0 and 2.0, $\omega_I = 1, \omega_O = -0.5$ and $\gamma = 1$. (b) The box region with the dashed line is magnified for (a).

show the shapes and positions of deformed drops at time t = 0, 0.25, 0.5, 1.0 and 2.0. It is found that the drop has significant deformation and inclined towards the boundary of the outer cylinder due to smaller tension coefficient as compared to the previous case. Since the surface tension decreases, the 'resistance' that the deformable drop presents to the flow becomes smaller, which in turn causes the drop to elongate. The corresponding plots of streamlines at time t = 0, t = 0.5 and t = 2.0 are presented in Fig. 27. Again we observe single eddies formed inside the drop and recirculating regions. The drop with significant deformation rotates round the liquid inside and the fluid flow inside the drop moves in the clockwise direction. Compared with those in Fig. 25, the flow patterns around the drop has also significant changes.

Finally, we keep this tension coefficient the same (i.e., $\gamma = 0.1$) but increase the flow rate by increasing the angular velocities of both the inner cylinder and outer cylinder to $\omega_I = 2, \omega_O = -1$, respectively. We present the pressure field in Fig. 28(a). It is again seen, as expected, that the pressure is discontinuous across the drop interface, whereas the sharp profile of the pressure is well captured by our method. The plot of the shapes and positions of deformed drops, at time t = 0, 0.25, 0.5, 1.0 and 2.0, is shown in Fig. 28(b). On comparing with Figs. 24(a) and 26(b), the drop is found to be deformed much more when the flow rate is increased. The drop becomes thinner and longer at the two ends. In Fig. 29, we present the corresponding plots of streamlines at time $t = 0, t = 0.5$ and $t = 2.0$. The flow patterns can be contrasted to those found in Figs. 25 and 27. It is interesting to observed in Fig. 29(c) that the flow separates at the regions with high curvature and two secondary eddies have formed.

Example 6.5 (*Elastic membrane through a constricted channel*). In the fifth example, as an application to simulate the similar biological processes like the motion of deformable cell in the micro-channel flow with low Reynolds number, here we shall



consider the motion of a membrane in an irregular channel with a constriction. Fig. 30 illustrates the geometry of the constricted channel and the initial position of a membrane. The geometric parameters W and H represent the width and height of constricted part, respectively, and L represents the distance between the inflow boundary and the entrance into constriction. (x_0, y_0) represents the initial position of the placed elastic membrane, and r represents the radius of elastic membrane. In the computation, the width of inlet of channel and the height of the constriction are 0.5 and 0.15, respectively, thus the standard 2.5:1 contraction geometry is being considered. We use a computational domain of $[0, 1.8] \times [-0.3, 0.3]$ and a 384×128 grid. The fluid viscosity $\mu = 0.1$ and the tension coefficient $T_0 = 1$ have been used. We specify a full developed parabolic velocity profile with $U_{\max} = 1$ at the inflow and outflow boundary and no-slip wall boundary is applied at the top and bottom boundaries. The no-slip boundary condition at the immersed rigid boundaries is enforced by imposing appropriate singular forces at the rigid boundaries.

In the simulation, $(x_0, y_0) = (0.37, 0.0)$ is taken, and the radius of initial elastic membrane is 0.15. The elastic membrane is pre-stretched from the undeformed state with a diameter of 0.1. We use 64 control points to represent the elastic membrane, and use 234 control points to present the rigid boundaries of the constricted channel. We take the time step $\Delta t = \Delta x/15$. Figs. 31 and 32 show the positions and shapes of the elastic membrane squeezing through the constriction and the corresponding velocity fields at a series of times. From these figures, we can clearly see how the deformed elastic membrane squeezes through the constriction. The elastic membrane undergoes large deformation while passing through the constriction. The interplay between elastic forces and viscous forces is evident. As the membrane passes through the constriction, viscous and elastic forces have comparable magnitudes. When the elastic membrane exits the contraction, the elastic forces cause the membrane to approach its circular shape (see Fig. 32).

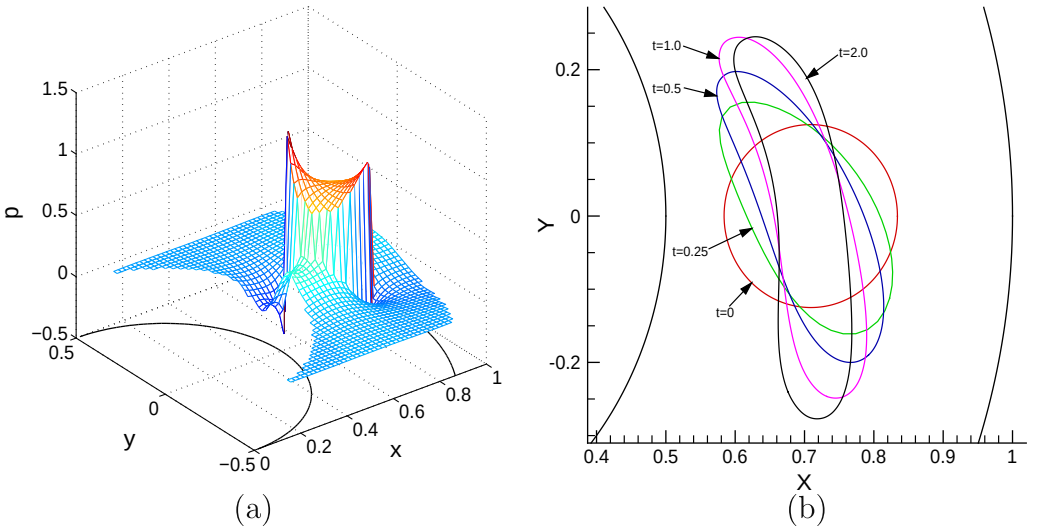
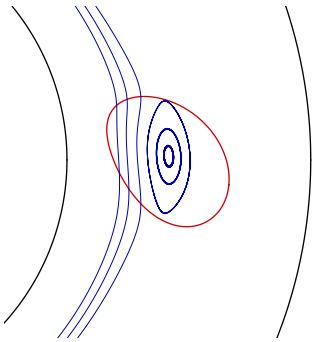
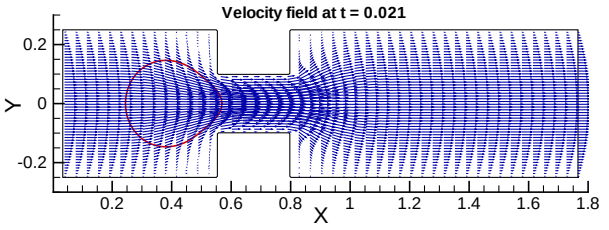
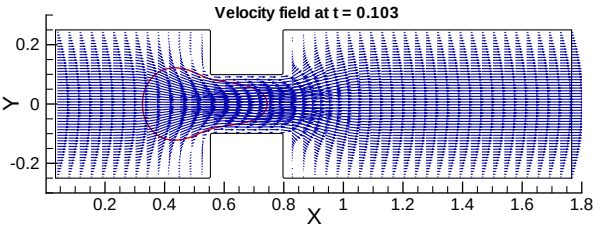


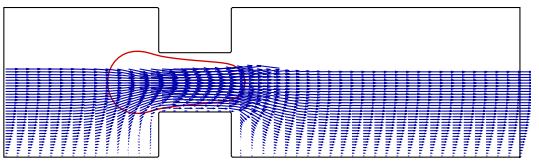
Fig. 28. For Example 6.4. (a) Pressure profile, and (b) configurations of deformed drops with circular initial shape at time $t = 0$, 0.25, 0.5, 1.0 and 2.0, $\omega_I = 2, \omega_O = -1$ and $\gamma = 0.1$.



Example 6.6 (*Example 6.6 Elastic membrane in a grooved channel*). In the final example, we shall consider the motion of a membrane in the flow of a grooved channel. Fig. 33 illustrates a sketch of the problem with the geometry of the grooved channel and the initial position of a membrane inside the groove. The geometric parameters W, D represent the width and depth of the groove, respectively, and L represents the distance between the inflow boundary and the groove. (x_0, y_0) is the initial position of the placed elastic membrane, and r represents the radius of elastic membrane. In the computation, the width of inlet of channel is 0.2. In this example, we use a 384×128 grid on a computational domain of $[0, 1.5] \times [-0.35, 0.15]$. We take $\mu = 0.1$ and the tension coefficient $T_0 = 1$. The inflow and outflow boundaries are assigned the full developed parabolic velocity profile with $U_{\max} = 1$. The velocity is set to zero at the top and bottom boundaries.







In the simulation, the parameters $(x_0, y_0) = (0.675, -0.16), W = 0.25$ and $D = 0.2$ are taken, and the diameter of initial elastic membrane is 0.15. The elastic membrane is pre-stretched from the undeformed circle with the radius $r_0 = 0.06$. We use 64 control points to represent the elastic membrane, and 192 control points to represent the rigid boundaries of the constricted channel. We take the time step $\Delta t = \Delta x/15$. In the simulation, we have prohibited the intersection between the deformable interface and rigid boundary by introducing a repulsive force into the total singular force at the deformable interface as in [24]. Figs. 34 and 35 show the positions and shapes of the elastic membrane and velocity fields at a series of times. From these figures, we can see that the flow induces the membrane to take large deformation and to move out of the groove due to the low tension coefficient and the relatively high initial location of the membrane inside the groove.

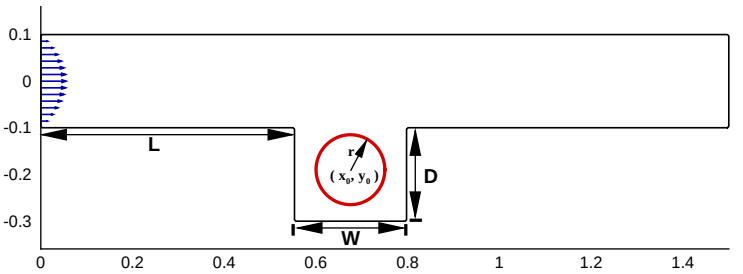


Fig. 33. An elastic membrane in a grooved channel.

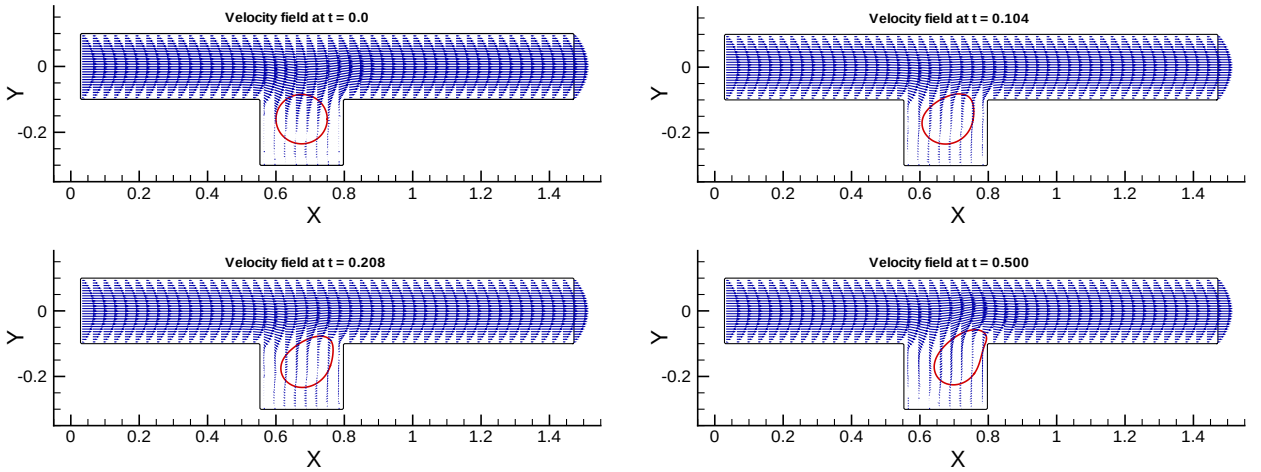
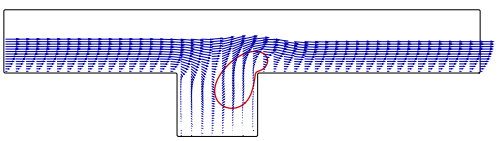


Fig. 34. The positions and shapes of the elastic membrane and velocity fields at different times.



7. Concluding remarks

In this paper, we have presented an immersed interface algorithm for solving the steady Stokes equations with fixed/moving interfaces on irregular domains. The method combines the IIM with a front tracking representation of the interface on a uniform Cartesian grid. The accuracy and ability of the proposed method to simulate in incompressible Stokes flows with fixed/moving interfaces on irregular domains are demonstrated by several examples. The grid convergence analysis shows that current algorithm can achieve second-order accurate for the velocity and nearly second-order accurate for the pressure. The current method can be applied to the simulation of biological flow problems like the flow with very low Reynolds number in mechanical filters for biomolecule separation and the deformation of a cell in a simple cell-trap. The present method can be also extended to solve for the general Navier–Stokes flows involving interfaces and irregular domains in a fairly straightforward manner. As a next step, we hope to extend the current algorithm to solve the Stokes flow problem involving deformable interfaces and moving rigid bodies simultaneously. In the future, we will also extend the present method to solve the incompressible two-phase Stokes flows involving rigid boundaries and deformable interfaces across which the viscosity is discontinuous. The results will be reported in the future.

References

[1] J. Adams, P. Swarztrauber, R. Sweet, FISHPACK: Efficient FORTRAN subprograms for the solution of separable eliptic partial differential equations, 1999. <http://www.scd.ucar.edu/css/software/fishpack/>.
[2] J.T. Beale, A.T. Layton, On the accuracy of finite difference methods for elliptic problems with interfaces, Commun. Appl. Math. Comput. Sci. 1 (2006) 91–119.
[3] P.A. Berthelsen, O.M. Faltinsen, A local directional ghost cell approach for incompressible viscous flow problems with irregular boundaries, J. Comput. Phys. 227 (2008) 4354–4397.
[4] R.P. Beyer, A computational model of the cochlea using the immersed boundary method, J. Comput. Phys. 98 (1992) 145–162.
[5] G. Biros, L. Ying, D. Zorin, A fast solver for the Stokes equations with distributed forces in complex geometries, J. Comput. Phys. 193 (2003) 317–348.
[6] D. Calhoun, A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions, J. Comput. Phys. 176 (2002) 231–275.
[7] G. Chen, Z. Li, P. Lin, A fast finite difference method for biharmonic equations on irregular domains, Adv. Comput. Math. 29 (2008) 113–133.
[8] B. Christoph, Domain imbedding methods for the Stokes equations, Numer. Math. 57 (1990) 435–451.
[9] R. Cortez, The method of regularized Stokeslets, SIAM J. Sci. Comput. 23 (2001) 1204–1225.
[10] 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.
[11] R. Dillon, L. Fauci, A. Fogelson, D. Gaver, Modeling biofilm processes using the immersed boundary method, J. Comput. Phys. 129 (1996) 57–73.
[12] C.D. Eggleton, A.S. Popel, Large deformation of red blood cell ghosts in a simple shear flow, Phys. Fluids 10 (1998) 1834–1845.
[13] H.C. Elman, Multigrid and Krylov subspace methods for the discrete Stokes equations, Int. J. Numer. Meth. Fluid 227 (1996) 755–770.
[14] H.C. Elman, Preconditioners for saddle point problems arising in computational fluid dynamics, Appl. Numer. Math. 43 (2002) 75–89.
[15] L.J. Fauci, C.S. Peskin, A computational model of aquatic animal locomotion, J. Comput. Phys. 77 (1988) 85–108.
[16] A.L. Fogelson, A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting, J. Comput. Phys. 1 (1984) 111–134.
[17] A.L. Fogelson, Continuum models of platelet aggregation: formulation and mechanical properties, SIAM J. Appl. Math. 52 (1992) 1089–1110.
[18] M. Giraldo, H. Power, W.F. Florez, Numerical simulation of the motion and deformation of a non-Newtonian shear-thinning drop suspended in a Newtonian circular Couette flow using DR-BEM, Eng. Anal. Bound. Elem. 33 (2009) 93–104.
[19] J.E. Gómez, H. Power, A multipole direct and indirect BEM for 2D cavity flow at low Reynolds number, Eng. Anal. Bound. Elem. 19 (1997) 17–31.
[20] H. Jia, F.-S. Lienb, E. Yeec, A robust and efficient hybrid cut-cell/ghost-cell method with adaptive mesh refinement for moving boundaries on irregular domains, Comput. Meth. Appl. Mech. Eng. 198 (2008) 432–448.
[21] G.M. Kobelkov, M.A. Olshanskii, Effective preconditioning of Uzawa type schemes for a generalized Stokes problem, Numer. Math. 86 (2000) 443–470.
[22] M.-C. Lai, H.-C. Tseng, A simple implementation of the immersed interface methods for Stokes flows with singular forces, Comput. Fluid 37 (2008) 99–106.
[23] A.T. Layton, An efficient numerical method for the two-fluid Stokes equations with a moving immersed boundary, Comput. Meth. Appl. Mech. Eng. 197 (2008) 2147–2155.
[24] D.V. Le, B.C. Khoo, J. Peraire, An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries, J. Comput. Phys. 220 (2006) 109–138.
[25] L. Lee, R.J. LeVeque, An immersed interface method for incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 25 (2003) 832–856.
[26] R.J. LeVeque, Z. Li, Immersed interface methods for Stokes flow with elastic boundaries or surface tension, SIAM J. Sci. Comput. 18 (1997) 709–735.
[27] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (1994) 1019–1044.
[28] Z. Li, K. Ito, The immersed interface method-numerical solutions of PDEs involving interfaces and irregular domains, SIAM Frontiers Appl. Math. (2006) 33.
[29] Z. Li, M.C. Lai, The immersed interface method for the Navier–Stokes equations with singular forces, J. Comput. Phys. 171 (2001) 822–842.
[30] Z. Li, C. Wang, A fast finite difference method for solving Navier–Stokes equations on irregular domains, Commun. Math. Sci. 1 (2003) 180–196.
[31] Z. Li, K. Ito, M.-C. Lai, An augmented approach for Stokes equations with a discontinuous viscosity and singular forces, Comput. Fluid 36 (2007) 622–635.
[32] M.N. Linnick, H.F. Fasel, A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains, J. Comput. Phys. 204 (2005) 157–192.
[33] C.W. Oosterlee, F.J.G. Lorenz, Multigrid methods for the Stokes system, Comput. Sci. Eng. 8 (2006) 34–43.
[34] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.
[35] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[36] J. Peters, V. Reichelt, A. Reusken, Fast iterative solvers for discrete Stokes equations, SIAM J. Sci. Comput. 27 (2005) 646–666.
[37] C. Pozrikidis, Boundary Integral and Singularity Methods for Linearized Viscous Flow, Cambridge University Press, Cambridge, 1992.
[38] V. Sarin, A. Sameh, An efficient iterative method for the generalized Stokes problem, SIAM J. Sci. Comput. 19 (1998) 206–226.
[39] J.M. Stockie, S.I. Green, Simulating the motion of flexible pulp fibres using the immersed boundary method, J. Comput. Phys. 147 (1998) 147–165.
[40] D. Russell, Z.J. Wang, A Cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow, J. Comput. Phys. 191 (2003) 177–205.

[41] V. Rutka, A staggered grid-based explicit-jump immersed interface method for two-dimensional Stokes flows, Int. J. Numer. Meth. Fluid 57 (2008) 1527–1543.
[42] Y. Sadd, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[43] J.A. Sethian, Y. Shan, Solving partial differential equations on irregular domains with moving interfaces, with applications to superconformal electrodeposition in semiconductor manufacturing, J. Comput. Phys. 227 (2008) 6411–6447.
[44] D. Shin, J.C. Strikwerda, Fast solvers for finite difference approximations for the Stokes and Navier–Stokes equations, J. Aust. Math. Soc. 38 (1996) 274–290.
[45] J. Stoer, R. Bulirsch, Introduction to Numerical Analysis, third ed., Springer-Verlag, 2002.
[46] K. Taira, T. Colonius, The immersed boundary method: a projection approach, J. Comput. Phys. 225 (2007) 2118–2137.
[47] Z.-J. Tan, K.M. Lim, B.C. Khoo, An adaptive mesh redistribution method for the incompressible mixture flows using phase-field model, J. Comput. Phys. 225 (2007) 1137–1158.
[48] E.Y. Tau, Numerical solution of the steady Stokes equations, J. Comput. Phys. 99 (1992) 190–195.
[49] Y.H. Tseng, J.H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, J. Comput. Phys. 192 (2003) 593–623.
[50] C. Tu, C.S. Peskin, Stability and instability in the computation of flows with moving immersed boundaries: a comparison of three methods, SIAM J. Sci. Stat. Comput. 13 (1992) 1361–1376.
[51] H.S. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A sharp interface Cartesian grid method for simulating flows with complex moving boundaries, J. Comput. Phys. 174 (2001) 345–380.
[52] N.T. Wang, A.L. Fogelson, Computational methods for continuum models of platelet aggregation, J. Comput. Phys. 151 (1999) 649–675.
[53] C. Wang, B.C. Khoo, An indirect boundary element method for three-dimensional explosion bubbles, J. Comput. Phys. 194 (2004) 451–480.
[54] A. Wiegmann, K.P. Bube, The explicit-jump immersed interface method: finite difference methods for PDEs with piecewise smooth solutions, SIAM J. Numer. Anal. 37 (2000) 827–862.
[55] S. Xu, Z.J. Wang, An immersed interface method for simulating the interaction of a fluid with moving boundaries, J. Comput. Phys. 216 (2006) 454–493.
[56] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundary, J. Comput. Phys. 156 (1999) 209–240.