Computers & Fluids 38 (2009) 1973–1983

Contents lists available at ScienceDirect

Computers & Fluids

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

A fast immersed interface method for solving Stokes flows on irregular domains

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 14 October 2008
Received in revised form 7 April 2009
Accepted 16 June 2009
Available online 21 June 2009

ABSTRACT

We present a fast immersed interface method for solving the steady Stokes flows involving the rigid boundaries. The immersed rigid boundary is represented by a set of Lagrangian control points. In order to enforce the prescribed velocity at the rigid boundary, singular forces at the rigid boundary are applied on the fluid. The forces are related to the jumps in pressure and the jumps in the derivatives of both pressure and velocity, and are approximated using the cubic splines. The strength of singular forces is determined by solving a small system of equations via the GMRES method. The Stokes equations are discretized using finite difference method with the incorporation of jump conditions on a staggered Cartesian grid and solved by the conjugate gradient Uzawa-type method. Numerical results demonstrate the accuracy and ability of the proposed method to simulate Stokes flows on irregular domains.

1. Introduction

In many of the applications in engineering and mathematical modelling in biology, it is often necessary to solve the Stokes equations on irregular domains. This paper considers the viscous incompressible steady Stokes flows for two dimensional problems involving rigid boundaries. In a two dimensional bounded domain Ω that contains a immersed rigid boundary Γ, we consider the steady Stokes equations formulated in the velocity–pressure variables, written as

$$\nabla p = \mu\Delta\mathbf{u} + \mathbf{F}(\mathbf{x}) + \mathbf{g}(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{1.1}$$

$$\nabla \cdot \mathbf{u} = 0, \quad \mathbf{x} \in \Omega, \tag{1.2}$$

with boundary conditions

$$\mathbf{u}|_{\partial\Omega} = \mathbf{u}_b, \tag{1.3}$$

where $\mathbf{u} = (u, v)^T$ is the fluid velocity, p is the fluid pressure, 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, and the effect of the rigid boundary Γ immersed in the fluid results in a singular force $\mathbf{F}$ which has the form

$$\mathbf{F}(\mathbf{x}) = \int_\Gamma \mathbf{f}(s)\delta(\mathbf{x} - \mathbf{X}(s))\, ds. \tag{1.4}$$

Here $\mathbf{X}(s)$ is the arc-length parametrization of the rigid boundary Γ, s is the arc-length, $\mathbf{f} = (f_1, f_2)^T$ is the force density, and $\delta(\cdot)$ is the Dirac delta function defined in the distribution sense. The rigid boundary Γ is immersed into a simpler domain and separates the fluid region into two parts. Eq. (1.2) together with the Dirichlet condition equation (1.3) leads to the compatibility condition that $\mathbf{u}_b$ must satisfy

$$\int_{\partial\Omega} \mathbf{u}_b \cdot \mathbf{n}_b\, dS = 0, \tag{1.5}$$

where $\mathbf{n}_b$ is the outer unit normal to $\partial\Omega$. To determine p uniquely we may impose some additional condition, such as $\int_\Omega p\, dx = 0$, see [4,13,34].

Throughout this paper, we 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.

For the class of problems considered in the present work, the immersed rigid boundary is fixed and the velocity on the rigid boundary is specified, so the forcing term is used to exactly enforce specified velocity at the rigid boundary. Conventional methods for solving the Stokes equations with immersed rigid boundaries include the body-fitted or structured grid approach. In this approach, the Stokes equations are discretized on a curvilinear grid that conforms to the immersed boundary and hence the boundary conditions can be imposed easily. The disadvantage of this method is that robust grid generation is required to account for the complexity of the immersed boundaries.

An alternative approach for solving complex viscous flows is the Cartesian grid method which solves the governing equations on a Cartesian grid and has the advantages of retaining the simplicity of the Stokes equations on the Cartesian coordinates and enabling the use of fast solvers. One of the most successful Cartesian grid methods is Peskin's immersed boundary method [30]. This method was originally developed to study the fluid dynamics of blood flow

* 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).

0045-7930/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2009.06.004

in the human heart [29]. The method was developed further and has been applied to many biological problems involving flexible boundaries [16,43]. The immersed boundary method has also been applied to handle problems with immersed boundaries [18,38]. In order to avoid using very small time step, Mohd-Yusof [27] and Fadlun et al. [15] proposed a direct forcing formulation. This forcing is direct in the sense that the exact velocity is imposed directly on the rigid boundary through an interpolation procedure. Lima E. Silva et al. [38] proposed a different approach to compute the forcing term **f** based on the evaluation of the various terms in the momentum (Navier–Stokes) equations at the rigid boundary. Another similar approach that combines the original immersed boundary method with the direct and explicit forcing was introduced by Uhlmann [42] for the simulation of particulate flows. The forcing term at the boundary is evaluated based on the desired velocity at the boundary which is simply given by the rigid-body motion and a preliminary velocity obtained explicitly without the application of a forcing term.

Once the forcing term is obtained at the boundary, the immersed boundary method (IBM) uses a discrete delta function to spread the force density to the nearby Cartesian grid points. Since the immersed boundary method smears out sharp interface to a thickness of order of the mesh width and it is only first-order accurate for problems with non-smooth but continuous solutions. In contrast, the immersed interface method (IIM) can avoid 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 [22] for solving elliptic equations, and later extended to Stokes flow with elastic boundaries or surface tension [21] on regular domain. We refer the interested readers to the recently published book by Li and Ito [23]. The method was developed further for the Navier–Stokes equations in [19,20,24] for problems with flexible boundaries. Recently, the immersed interface method has been employed to solve for viscous flows with static rigid immersed boundaries [5,6,19,25,33]. In [6,25], the no-slip boundary conditions are imposed directly by determining the correct jump conditions for the stream function and vorticity. In [33], a Cartesian grid method for modelling multiple moving objects in incompressible viscous flow is considered. Biros et al. [5] presented 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 express the jumps as a source term at some grid points close to the interface. Le et al. [19] has presented an immersed interface method for the incompressible Navier–Stokes equations involving rigid and flexible boundaries. In [19], the immersed boundaries are represented by a set of Lagrangian controls points. The strength of singular forces is determined to impose the no-slip condition at the boundary by solving a small system of equations at each time step. In [8], a new biharmonic solver has been applied to solve the incompressible Stokes flow on an irregular domain. Recently, Rutka [34] 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 up to second order derivatives along the interface, in contrast to our approach in this paper, while the augmented system of equations using EJIIM in [34] will be much larger because more unknowns are introduced as augmented variables.

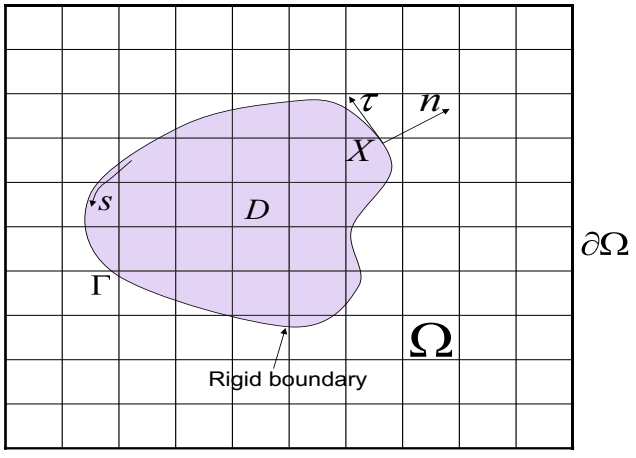

Fig. 1. A typical irregular domain D is immersed in a simpler rectangular domain Ω with a uniform Cartesian grid. We use $\mathbf{n}$ and $\boldsymbol{\tau}$ to denote the unit outward normal and tangential directions of the rigid boundary Γ, respectively.

Another Cartesian grid approach has been presented by Ye et al. [46] and Udaykumar et al. [41] using a finite volume technique. They reshaped the immersed boundary cells and used a polynomial interpolating function to approximate the fluxes and gradients on the faces of the boundary cells while preserving second-order accuracy.

Finally, it should be noted that there are other methods to solve Stokes flow on irregular domains, such as the method of fundamental solutions (MFS). The method has been used to solve Stokes problems in multidimensional interior or exterior flow fields in the past years. Alves and Silvestre [2], Young et al. [47,49] and Chen et al. [9] solved 2D and 3D interior problems by MFS based on Stokeslets. The same idea is applied by Tsai et al. [40] to 3D exterior flows. In [48], Young et al. used the dual-potential of the velocity potential and the stream function vector for both 2D and 3D Stokes equations.

To our knowledge, there are few works on IIM solving the incompressible steady Stokes equations on irregular domains in the literature. Most work of IIM by finite difference scheme are based on solving a Poisson solver three times on a regular domain with periodic boundary conditions. As such, one main objective of the present work is to develop an efficient IIM with second-order accuracy for solving steady Stokes flows involving the irregular domains. Note that the current approach shares a common feature with the approach of Le et al. [19] in terms of introducing the singular force as an unknown, whereas their method is used to solve only the Navier–Stokes equation. The approach of using the method in [19] directly to get the steady state solution for stationary Stokes flows is generally impractical and much more expensive. The application of the current IIM with the introduction of singular force to steady Stoke flows on irregular domain is, however, new. Such an implementation of IIM for steady Stokes flows in this work is non-trivial. The current IIM is mainly based on a fast and efficient iterative Stokes solver and therefore there is no need to use the projection method. (On the other hand, while the current IIM is of particular interest to steady Stokes flows as in this work, the present method can be also applied to solve for the general Navier–Stokes equation on irregular domains based on the extension to the efficient generalized Stokes solver [14,35] in a fairly straightforward manner.) In implementation, the present method combines the IIM with a front tracking representation of the interface on a uniform Cartesian grid. The singular force at the rigid boundary is determined to enforce the prescribed velocity at the rigid boundary. The singular force is then computed implicitly by solving a small, dense linear system of equations via the GMRES iterative method. Once the force is computed, we next compute the jump in pressure and jumps in the derivatives of both pressure and velocity. The Stokes equations are discretized on a staggered Cartesian grid by a second order finite difference method 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] are then 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 rest of the paper is organized as follows. In Section 2, we present the jump relations along the immersed interface via the singular force $\mathbf{f}$ and the jumps in the velocity and pressure and their derivatives. In Section 3, we describe the generalized finite difference approximations to the solution derivatives, which incorporate the solution jumps. 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. Jump conditions across the interface

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, respectively. The jump of an arbitrary function $q(\mathbf{X})$ across Γ at $\mathbf{X}$ is denoted by

$$[q] = \lim_{\epsilon \to 0^+} q(\mathbf{X} + \epsilon \mathbf{n}) - \lim_{\epsilon \to 0^+} q(\mathbf{X} - \epsilon \mathbf{n}). \tag{2.1}$$

Denoting (ξ, η) the local coordinates associated with the directions of $\mathbf{n}$ and $\boldsymbol{\tau}$, respectively, we have the following jump conditions for the velocity and pressure across the interface (see [19,20] for details):

$$[\mathbf{u}] = \mathbf{0}, \quad [\mathbf{u}_\eta] = \mathbf{0}, \quad [\mathbf{u}_\xi] = -\frac{1}{\mu}\hat{f}_2 \boldsymbol{\tau}, \quad [\mathbf{u}_{\eta\eta}] = \frac{1}{\mu}\kappa \hat{f}_2 \boldsymbol{\tau}, \tag{2.2}$$

$$[\mathbf{u}_{\xi\eta}] = -\frac{1}{\mu}\frac{\partial \hat{f}_2}{\partial \eta}\boldsymbol{\tau} - \frac{1}{\mu}\kappa \hat{f}_2 \mathbf{n}, \quad [\mathbf{u}_{\xi\xi}] = -[\mathbf{u}_{\eta\eta}] + \frac{1}{\mu}[p_\xi]\mathbf{n} + \frac{1}{\mu}[p_\eta]\boldsymbol{\tau} - \frac{1}{\mu}[\mathbf{g}], \tag{2.3}$$

$$[p] = \hat{f}_1, \quad [p_\xi] = [\mathbf{g}] \cdot \mathbf{n} + \frac{\partial \hat{f}_2}{\partial \eta}, \quad [p_\eta] = \frac{\partial \hat{f}_1}{\partial \eta}, \tag{2.4}$$

$$[p_{\eta\eta}] = \frac{\partial^2 \hat{f}_1}{\partial \eta^2} - \kappa[p_\xi], \quad [p_{\xi\eta}] = \frac{\partial([\mathbf{g}] \cdot \mathbf{n})}{\partial \eta} + \frac{\partial^2 \hat{f}_2}{\partial \eta^2} + \kappa[p_\eta], \quad [p_{\xi\xi}] = [\nabla \cdot \mathbf{g}] - [p_{\eta\eta}]. \tag{2.5}$$

Here, $\hat{f}_1$ and $\hat{f}_2$ are the components of the force density in the normal and tangential directions of the interface, denoting $\hat{\mathbf{f}} = \left(\hat{f}_1, \hat{f}_2\right)$, and κ is the signed valued of the curvature of the interface. In this work, we shall incorporate the jump conditions of higher-order spatial derivatives for the pressure into the finite difference scheme as compared to the literature. By differentiating the first equality twice and the second equality once in (2.4) along the tangential direction of the interface, we can get the first equality and second equality of (2.5), respectively. The third equality of (2.5) is derived by applying the divergence operator to Eq. (1.1) in the local coordinate. We note that from expressions (2.2)–(2.5) the values of the jumps of the first and second derivatives of velocity and pressure taken with respect to the (x, y) coordinates are easily obtained by a simple coordinate transformation. For instance, we have

$$[\mathbf{u}_x] = [\mathbf{u}_\xi]n_1 + [\mathbf{u}_\eta]\tau_1, \quad [\mathbf{u}_y] = [\mathbf{u}_\xi]n_2 + [\mathbf{u}_\eta]\tau_2, \tag{2.6}$$

$$[\mathbf{u}_{xx}] = [\mathbf{u}_{\xi\xi}]n_1^2 + 2[\mathbf{u}_{\xi\eta}]n_1\tau_1 + [\mathbf{u}_{\eta\eta}]\tau_1^2, \tag{2.7}$$

$$[\mathbf{u}_{yy}] = [\mathbf{u}_{\xi\xi}]n_2^2 + 2[\mathbf{u}_{\xi\eta}]n_2\tau_2 + [\mathbf{u}_{\eta\eta}]\tau_2^2. \tag{2.8}$$

3. Generalized finite difference formulas

One of the basic components for determining the correction terms in the next section is the generalized finite difference formulas. We shall briefly review the generalized finite difference formulas in this section. Here, we show two particular generalized finite difference formulas for demonstration. Assume that the interface 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 $q(x)$:

$$q_x(x_i) = \frac{q_{i+1} - q_{i-1}}{2h} - \frac{1}{2h}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}[q^{(m)}]_\alpha + O(h^2), \tag{3.1a}$$

$$q_{xx}(x_i) = \frac{q_{i+1} - 2q_i + q_{i-1}}{h^2} - \frac{1}{h^2}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}[q^{(m)}]_\alpha + O(h), \tag{3.1b}$$

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

$$[q^{(m)}]_\alpha = \lim_{x \to \alpha, x \in \Omega^+} q^{(m)}(x) - \lim_{x \to \alpha, x \in \Omega^-} q^{(m)}(x), \tag{3.2}$$

in short, $[\cdot] = [\cdot]_\alpha$, and $q^{(0)} = q$. Note that if the interface 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 crossings could also be derived, we refer the readers to [45] for details. From Eqs. (3.1a) and (3.1b) the correction terms for $q_x(x_i)$ and $q_{xx}(x_i)$ can be defined as

$$C\{q_x(x_i)\} = -\frac{1}{2h}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}\left[q^{(m)}\right],$$

$$C\{q_{xx}(x_i)\} = -\frac{1}{h^2}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}\left[q^{(m)}\right].$$

Thus, the finite difference approximation near the interface, for the derivatives of a function q, includes the standard central difference terms plus the additional correction terms.

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 irregular 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 [39]. We use a uniform MAC grid with mesh width $h = \Delta x = \Delta y$ in the computation. With the MAC mesh, the pressure field is defined at the cell center (i, j), where $i \in \{1, 2, \ldots, N_x\}$ and $j \in \{1, 2, \ldots, N_y\}$. The velocity fields u and v are defined at the vertical edges and horizontal edges of a cell, respectively. The pressure and the velocity components u and v are arranged as in Fig. 2. An advantage of the staggered grid is that there is no need for pressure boundary conditions while dealing only with the derivative of pressure since the pressure nodes are at the cell center. Based on the MAC grid, the irregular grid points can be distinguished as in [50].

4.1. Fast Stokes solver with IIM-based

Discretization of Eqs. (1.1)–(1.3) by MAC finite difference scheme leads to the following linear system:

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

$$D^{\mathrm{MAC}} \mathbf{u} = C_2 - \widehat{C}_2, \quad \mathbf{u}|_{\partial\Omega} = \mathbf{u}_b. \tag{4.2}$$

The above discretization of the Stokes equations at those grid points near the interface has been modified to account for the jump

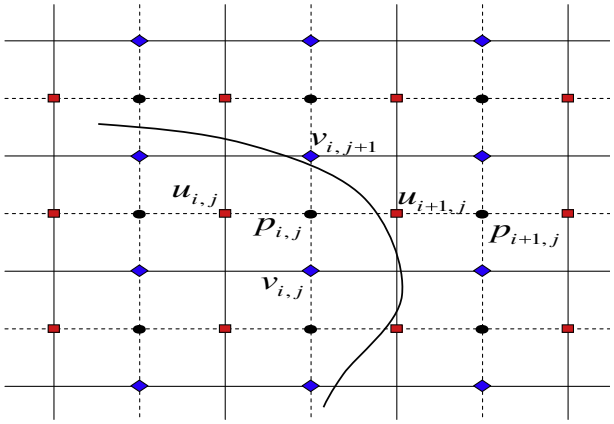

Fig. 2. A diagram of the interface cutting through a staggered grid with a uniform mesh width h, where the velocity component u is at the left-right face of the cell and v is at the top-bottom face, and the pressure is at the cell center.

conditions across the interface due to the presence of singular forces at the interface. The coefficients $\mathbf{C}_1$ and C_2 are the spatial correction terms added to the finite difference equations at the points near the interface to improve the accuracy of the local finite difference approximations. In order to satisfy the discrete compatibility condition corresponding to (1.5) to thereby ensure the solvability of system equations (4.1) and (4.2), we employed a solvable perturbed system with similar approach as in [21] via perturbing C_2 to $C_2 - \widehat{C}_2$ on the right hand of Eq. (4.2), where $\widehat{C}_2$ is the mean value of the correction term C_2. We refer the readers to [21] for details. The above correction terms can be computed by using the generalized finite difference formulas in the previous section if we know the jumps in the solution and their derivatives. We will review how to compute the correction terms in the next section. 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

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

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

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

and system (4.4) can be solved using some fast solvers, for example, the PCG method [13,31], the PMINRES method [13,31], the FFT-based method [12], multigrid method [13,28,31], and so on. In this work, we shall employ the CG-Uzawa method. The Uzawa procedure for problems with immersed interfaces is analogous to the fast iterative method presented in [37,39] and consists of two steps:

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

Here, the $D^{\mathrm{MAC}}\Delta_h^{-1}G^{\mathrm{MAC}}$ 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 with $D^{\mathrm{MAC}}\Delta_h^{-1}G^{\mathrm{MAC}}$ 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 take advantage of the fast solvers from FISHPACK [1] to solve these Poisson equations very efficiently. 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 Poisson solver from FISHPACK is $\mathcal{O}(M\log(M))$, where M is the number of interior grid points of the embedded domain. Note that the present CG method converges fast since the number of iterations in the CG method is very small and are essentially independent of the mesh size, which can be seen from the various numerical examples in Section 6. As such, the present Stokes solver is fast and efficient as also discussed in [37,39].

4.2. Correction terms calculation

In this section, we will illustrate how to evaluate the above correction terms $\mathbf{C}_1$ and C_2. The correction terms $\mathbf{C}_1$ and C_2 are evaluated as follows:

$$\mathbf{C}_1 = \mu(C\{\Delta\mathbf{u}\}) - C\{\nabla p\}, \tag{4.5a}$$

$$C_2 = -C\{\nabla\cdot\mathbf{u}\}. \tag{4.5b}$$

We note that all the correction terms are evaluated at least to first order accuracy. This is sufficient to guarantee second order accuracy globally since our numerical scheme is second order away from the boundary and only the irregular points near the boundary are treated with a first order scheme. To evaluate the correction term $C\{\Delta\mathbf{u}\}$ of (4.5a) 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 with the grid lines, and $[\mathbf{u}_y]$ and $[\mathbf{u}_{yy}]$ at β using the force strength. 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

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

Similarly, we can compute for the other correction terms in (4.5a) and (4.5b) as in [19].

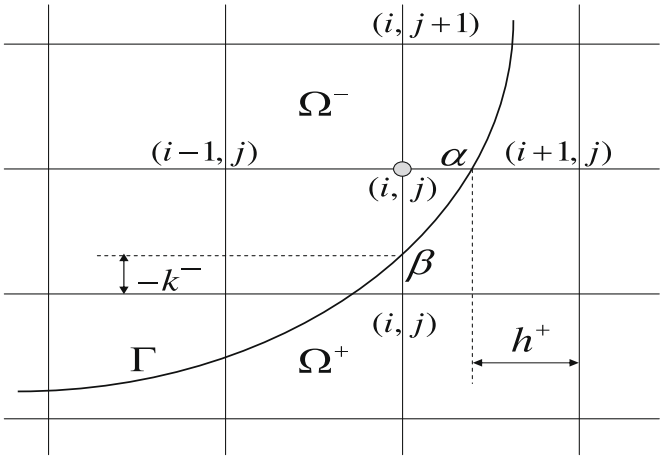

Fig. 3. Interface and mesh geometry near the grid point (i,j).

4.3. Determinant of singular force at control points

Assuming that the singular force **f** at the rigid boundary is known, the velocity field **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, $\mathbf{U}_k$, can be interpolated from the velocity **u** at the grid points as shown in [19]. Thus, we can write

$$\mathbf{U}_k = \mathbf{U}(\mathbf{X}_k) = \mathscr{B}(\mathbf{u}), \tag{4.6}$$

where $\mathscr{B}$ is the interpolation operator which includes the appropriate correction terms required to guarantee at least second order accuracy when the derivatives of the velocity are discontinuous. Two possible ways are to use the third-order accurate least square interpolation scheme in [23] or the modified bilinear interpolation with jump conditions in [19]. In order to save computational cost, we employ the latter in this work. Since the relationships between the singular forces and the jumps in the solution or its derivatives are linear and all the equations solved are linear, we can write the velocity at the rigid boundary as,

$$\mathbf{U}_k = \mathbf{U}_k^0 + \mathbf{A}\mathbf{f}, \tag{4.7}$$

where $\mathbf{U}_k^0$ corresponds to the velocity at the control points obtained by solving Eqs. (1.1) and (1.2) with $\mathbf{f} = 0$. **A** is a $2N_b \times 2N_b$ matrix, where N_b is the number of control points. The vector **Af** is the velocity at the control points obtained by solving the following equations:

$$\nabla_h p_{\mathbf{f}} = \mu \Delta_h \mathbf{u}_{\mathbf{f}} + \overline{\mathbf{C}_1}, \tag{4.8}$$

$$\nabla_h \cdot \mathbf{u}_{\mathbf{f}} = \overline{C}_2, \quad \mathbf{u}_{\mathbf{f}}|_{\partial\Omega} = 0, \tag{4.9}$$

$$\mathbf{A}\mathbf{f} = \mathscr{B}(\mathbf{u}_{\mathbf{f}}), \tag{4.10}$$

with **f** being the singular force at the rigid boundary. Here, $\overline{\mathbf{C}}_1$ and $\overline{C}_2$ are the correction terms which take into account the effect of the singular force **f** at the rigid boundary. From Eq. (4.7), with the prescribed velocity $\mathbf{U}_p$ at the rigid boundary, the singular force **f** at the rigid boundary is determined by solving

$$\mathbf{A}\mathbf{f} = \mathbf{U}_p - \mathbf{U}_k^0. \tag{4.11}$$

Note that the matrix **A** depends on the location of the boundary. In the present work, the boundary is fixed, therefore, the matrix **A** can be formed and factorized. In order to compute the coefficients of **A**, one can solve Eqs. (4.8)–(4.10) for $2N_b$ times, i.e., once for each column. Each time, the singular force **f** is set to zero except for the entry corresponding to the column we want to calculate, which is set to one. Once the matrix **A** has been calculated, the terms on the right-hand side, $\mathbf{U}_p - \mathbf{U}_k^0$, can be computed. The resulting small system of Eq. (4.11) is then solved for **f** via back substitution. Note that the matrix is singular for a closed immersed boundary since an arbitrary constant can be added to the normal component of the forces. We can add the constraint that the normal component of the forces sum to zero or the normal force at one of control points is set to zero. In actual computation, we can use the GMRES method to solve the system of Eq. (4.11), which only requires the matrix vector multiplication and is more efficient than forming the matrix explicitly for the present problem. In our simulations, only a few iterations are needed in the GMRES method, so the algorithm is efficient.

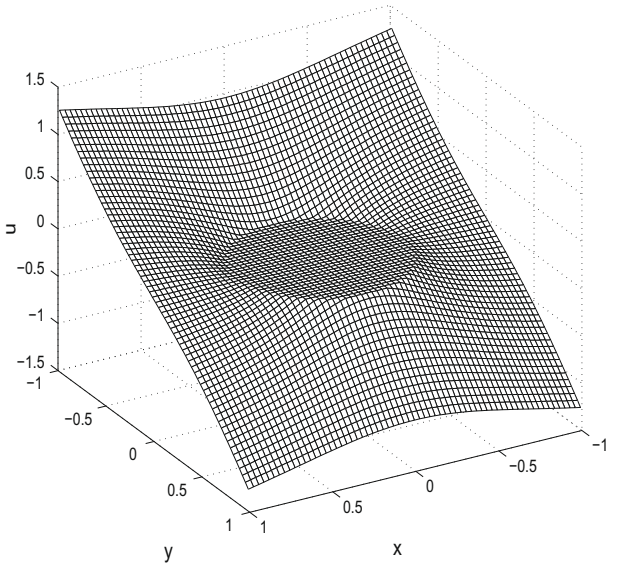

Fig. 4. The *x*-component of velocity field **u** for Example 6.1.

5. Numerical implementation

In this section, we shall describe a basic implementation of our algorithm. The singular force **f** to enforce the prescribed velocity $\mathbf{U}_p$ at the immersed boundary can be summarized as follows:

Algorithm. The implementation of the IIM on irregular domains

Step 1: Compute the right-hand side of (4.11) by calculating $\mathbf{U}_p - \mathbf{U}_k^0$.
- Set $\mathbf{f} = 0$, and solve (4.1) and (4.2) for the velocity at all the grid points.
- Interpolate the velocity at the control points $\mathbf{U}_k^0$ as in (4.6).
- Compute the right-hand side vector $\mathbf{b} = \mathbf{U}_p - \mathbf{U}_k^0$.

Step 2: Compute the singular force **f** by solving (4.11) using the GMRES method.

Step 3: Compute **u** and *p* using the CG-Uzawa method with incorporating some appropriate correction terms as described in Section 4.1.

6. Numerical examples

In this section, several numerical experiments are carried out to demonstrate the capabilities and the accuracy of our proposed algorithm in this work. All the simulations are done on a Pentium 4 PC with 3.00 GHz. The stop tolerance for the CG method and the GMRES method are set as 10^{-8}.

Example 6.1 (*Circular flow*). We start our numerical tests by checking the accuracy of the algorithm. In this example, there is provided exact solution [24] and the exact velocity and pressure are given by

$$u(x,y) = \begin{cases} \left(\frac{y}{r} - \frac{y}{r_0}\right) & r > r_0, \\ 0 & r \leqslant r_0, \end{cases} \tag{6.1}$$

$$v(x,y) = \begin{cases} \left(-\frac{x}{r} + \frac{x}{r_0}\right) & r > r_0, \\ 0 & r \leqslant r_0, \end{cases} \tag{6.2}$$

$$p(x,y) = \begin{cases} \cos(\pi x)\cos(\pi y) & r > r_0, \\ 0 & r \leqslant r_0, \end{cases} \tag{6.3}$$

where $r = \sqrt{x^2 + y^2}$. The interface is the circle with the radius r_0. 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 [24]. The external force term **g** is

Table 1
Grid refinement analysis for Example 6.1.

N	$\|E_{\mathbf{u}}\|_\infty$	Order	$\|E_p\|_\infty$	Order	N_{CG}	N_{GMRES}	CPU (s)
32	1.3433E−03	–	6.9325E−03	–	11	22	0.56
64	3.3438E−04	2.01	1.8447E−03	1.91	11	19	1.81
128	7.4946E−05	2.16	5.3715E−04	1.78	12	17	7.38
256	1.8199E−05	2.04	1.4493E−04	1.89	13	15	31.86

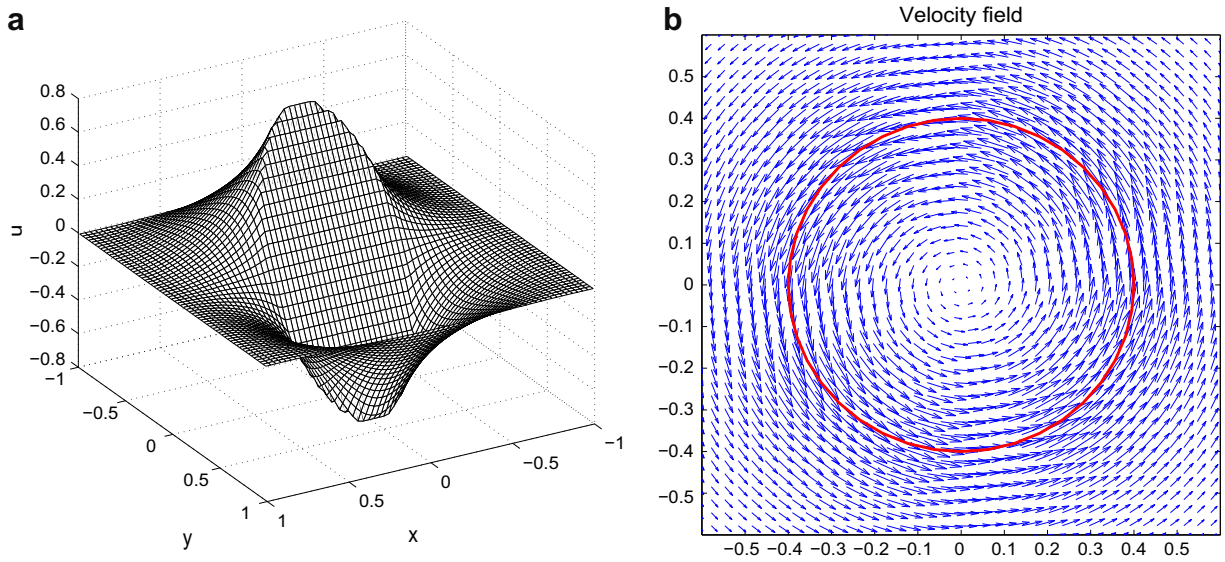

Fig. 5. (a) The x-component of velocity field $\mathbf{u}$ and (b) the velocity field $\mathbf{u}$ for Example 6.2.

computed to satisfy the Stokes equation (1.1), which also has a finite jump across the interface.

In the computations, we use the exterior geometry of the circle in the domain $[-1,1]\times[-1,1]$ and the radius of the circle is taken as $r_0 = 0.5$. The prescribed velocity at the circular interface is no-slip condition. The simulation is performed with a 64×64 grid and $\mu = 0.1$. In Fig. 4, we present the plots 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

$$\text{order} = \frac{\log(\|E_u(N)\|_\infty / \|E_u(2N)\|_\infty)}{\log 2}. \tag{6.4}$$

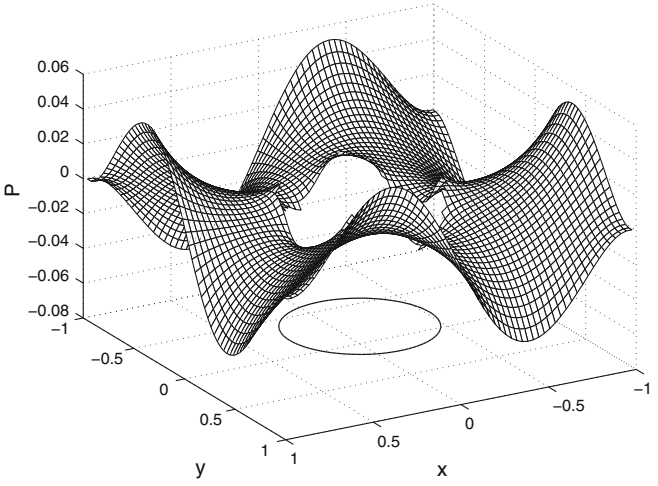

Fig. 6. Pressure distribution for Example 6.2.

Table 2
Grid refinement analysis for Example 6.2.

N	$\|E_{\mathbf{u}}\|_\infty$	Order	$\|E_p\|_\infty$	Order	N_{CG}	N_{GMRES}	CPU (s)
64	1.5703E−03	–	4.2703E−03	–	7	10	0.67
128	3.7398E−04	2.07	1.1763E−03	1.86	7	9	2.65
256	9.2208E−05	2.02	3.4015E−04	1.79	8	9	10.83

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

$$\|E_u(N)\|_\infty = \max_{ij} |U_{ij} - u(x_i, y_j)|, \tag{6.5}$$

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 and the seventh column shows the number of the GMRES iterations, which indicates that a limited number of iterations are needed in CG and GMRES and the number of iterations is almost independent of the mesh size. The CPU time in seconds is listed in the eighth column, which shows that the present method is reasonably fast.

Example 6.2 (*Rotational flow*). In this example, we consider a fixed interface problem with no exact solution taken from [19]. The interface is a circle with radius $r = 0.4$, which is located at the center of the square domain $[-1,1]\times[-1,1]$. On the boundary of Ω, we set the no-slip boundary conditions. We prescribe the interface to rotate with angular velocity $\omega = 2$.

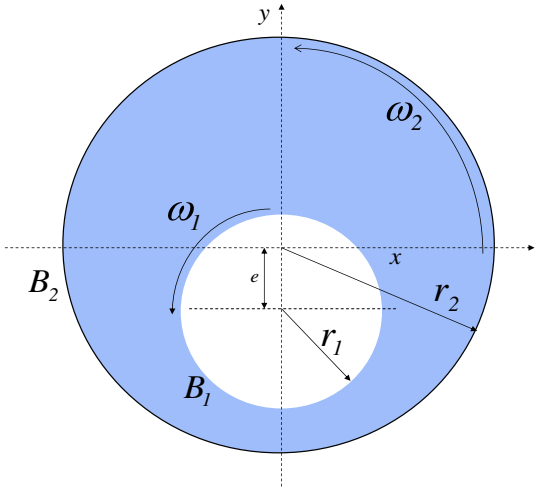

Fig. 7. Schematic diagram of the geometry and fluid domain between eccentric rotating cylinders. ω_1 and ω_2 represent the angular velocity of the inner and outer boundaries, respectively.

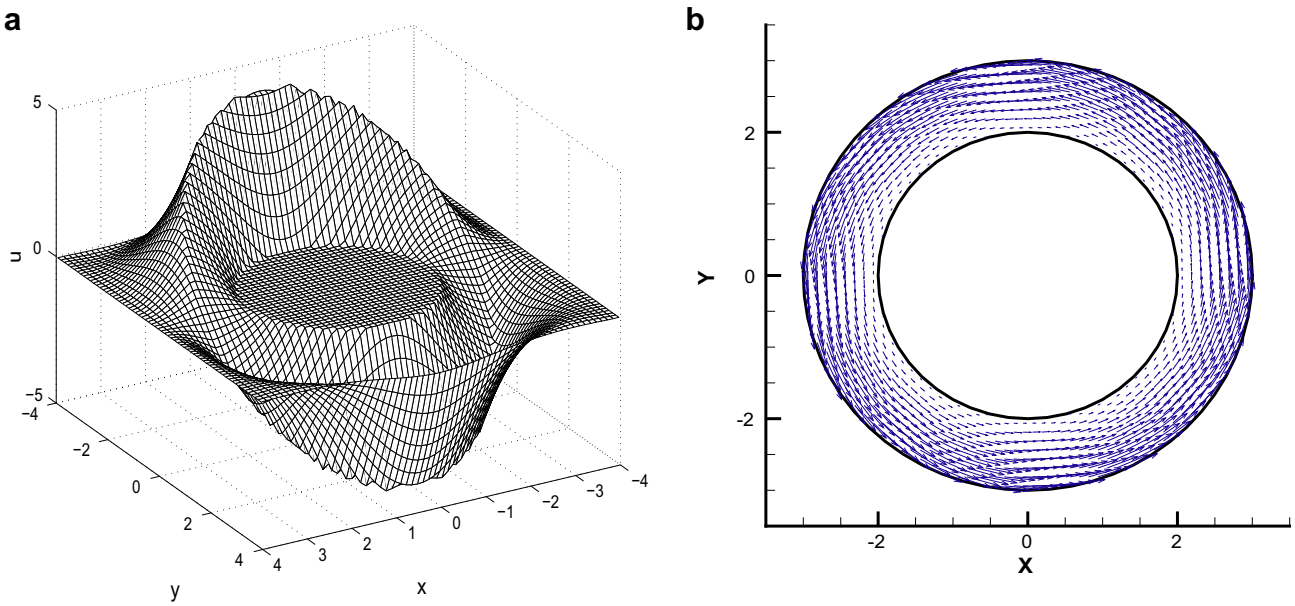

Fig. 8. (a) The x-component of velocity field $\mathbf{u}$ and (b) the velocity field between concentric cylinders for Example 6.3.

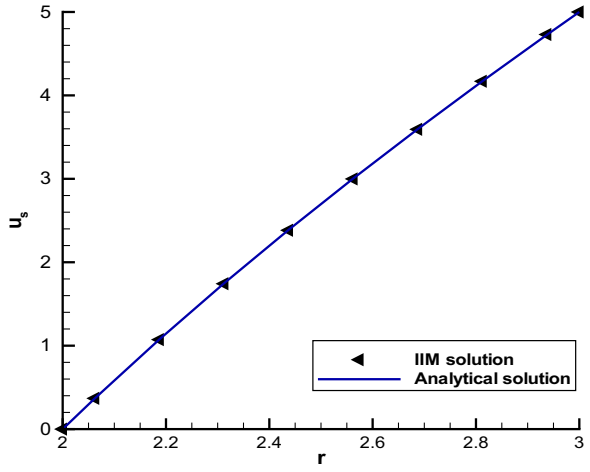

Fig. 9. Circumferential velocity solution for the flow between the concentric cylinders.

Table 3
Configuration of eccentric rotating cylinders for Example 6.3.

	r_1	r_2	e	ω_1	ω_2
Case 1	0.5	1	0.25	2	0
Case 2	0.5	1	0.25	0	2
Case 3	0.5	1	0.25	−2	1
Case 4	0.5	1	0.35	−2	−1

In the computations, we use a 64×64 grid and set $\mu = 0.1$. The steady solution is shown in Fig. 5, which corresponds to a rigid body motion inside the interface. Fig. 5(a) and (b) shows the x-component of the velocity field $\mathbf{u}$ and the velocity field, respectively. The plot of the pressure is presented in Fig. 6. From these figures, we can observe that the velocity u is continuous but not smooth due to imposition of singular force at the rigid boundary, as expected, while the pressure is discontinuous. The sharp solution of the pressure is captured well by our method. Finally, we carry out a grid refinement analysis, 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 fairly small number of iterations and is almost independent of mesh size, and therefore it is very fast in terms of the CPU time. In particular, the CPU time is 0.67 s on a 64×64 grid, which can be compared to the computational cost of 26.03 s by using the method [19] directly to get the steady state solution for this same problem. From the comparison, it is clear that the computational cost is fairly low using the present method to solve steady Stokes flows directly.

Example 6.3 (*Flow between two cylinders*). In this example, we shall consider two-dimensional Stokes flow confined in the two cylinders of which at least one is allowed to rotate about its axis, thereby inducing a flow in the region between the cylinders. These flows are related to the lubrication studies of journal bearing, which was already studied by some researchers [3,10,11,34,44,48].

Our flow domain is the annular region as shown in Fig. 7. The inner and outer cylinders have radius r_1 and r_2, respectively. Their centers are at $(0,0)$ and $(0,-e)$, respectively, where e, $0 \leqslant e < r_2 - r_1$, is the distance between the centers of two cylinders. This configuration yields an eccentricity ratio $\lambda = e/(r_2 - r_1)$. In case of $e = 0$, it indicates that these two cylinders are concentric; otherwise they are eccentric. The parameters U_1 and U_2 are defined for the circumferential velocities of the outer and inner cylinders, respectively. The constant angular velocities along the inner and outer cylinders are denoted by ω_1 and ω_2, respectively. The signs of ω_1 and ω_2 indicate their respective senses of rotation (e.g. positive sign means anti-clockwise rotation). In particular, when no-slip stationary boundary condition is specified on the fluid–solid interface along the inner or outer cylinder, it indicates that the corresponding inner or outer cylinder is stationary ($\omega_1 = 0$ or $\omega_2 = 0$). In the computations, we use a 64×64 grid and set $\mu = 0.1$. We use 48 control points and 64 control points to represent the inner circular cylinder and the outer circular cylinder, respectively, unless it is stated otherwise.

We first consider the flow between two concentric cylinders as in [17,32], i.e., $e = 0$. In this study, the inner cylinder is kept sta-

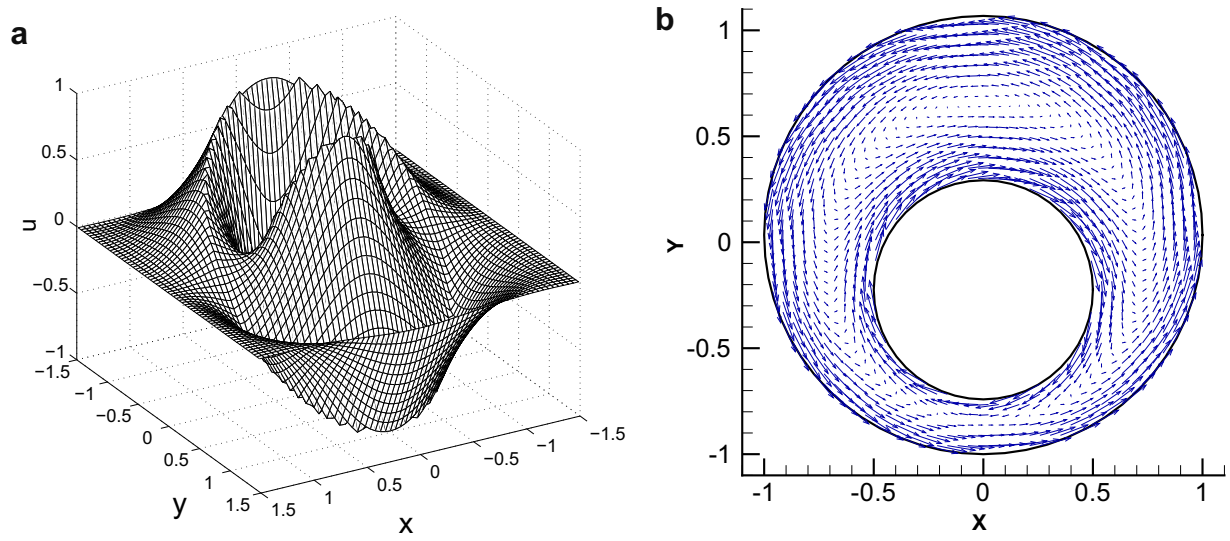

Fig. 10. (a) The x-component of velocity field $\mathbf{u}$ and (b) the velocity field $\mathbf{u}$ for case 3 listed in Table 3 for Example 6.3.

tionary, while the outer cylinder rotates at a constant angular velocity. We take the radius of the inner cylinder and the outer cylinder as $r_1 = 2$ and $r_2 = 3$, respectively, and take the circumferential velocity $U_s = 5$ along the outer cylinder. Fig. 8 shows the x-component of the velocity field and velocity vector between two concentric cylinders. The numerical result for the circumferential velocity is compared with analytical solution given in [32,36]. The results are found in very good agreement with analytical solution (see Fig. 9) and computed solution in [32].

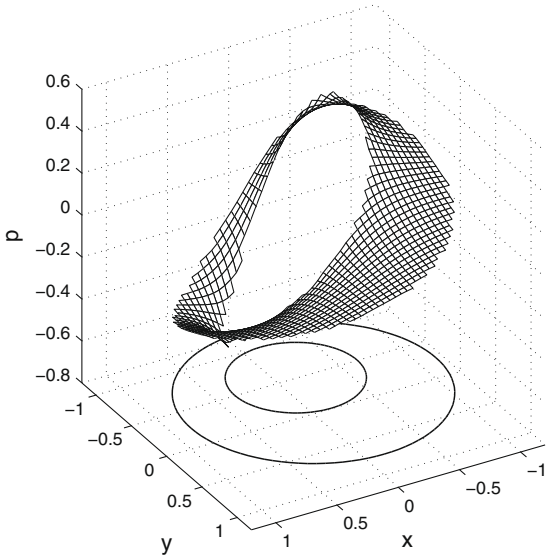

Fig. 11. Pressure distribution for case 3 listed in Table 3 for Example 6.3.

Next, we consider the flow between two eccentric cylinders, i.e., $e \neq 0$. The flow is simulated under the setting summarized in Table 3. In particular, for case 3, we show the x-component of the velocity $\mathbf{u}$ and the velocity field in Fig. 10(a) and (b), respectively. The corresponding pressure distribution is present in Fig. 11 for this case. We shall next consider the flow of different cases. The streamlines patterns are created by the rotation of the inner and outer cylinders. In Fig. 12, we show streamlines for the four different cases of steady boundary motion. The comparisons of the corresponding pressure contours for the four different flows are shown in Fig. 13. Note that the flow separates into several regions with the streamline pattern depicting the general demarcation of the vortex and the main flow; further detailed discussions are provided below. From these figures, it is also observed that the flow patterns are symmetric with respect to the y-axis passing through the centers of both cylinders due to the reversibility of the Stokes flows, and a single recirculating eddy appears in the interior of the annular region.

Close examination reveals that different types of recirculation region are formed which are closely related to the rotation modes of the cylinders. In case 1, corresponding to inner cylinder rotation

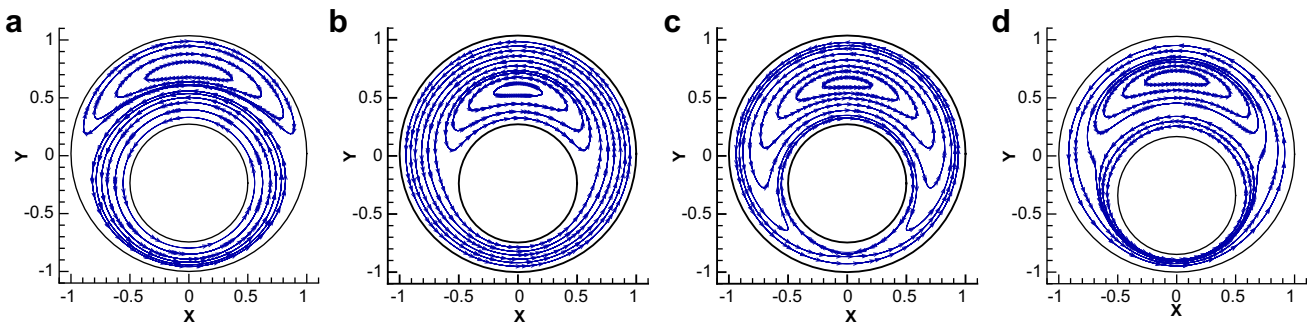

Fig. 12. Streamlines for cases 1–4 in Table 3 corresponding to (a–d), respectively for Example 6.3.

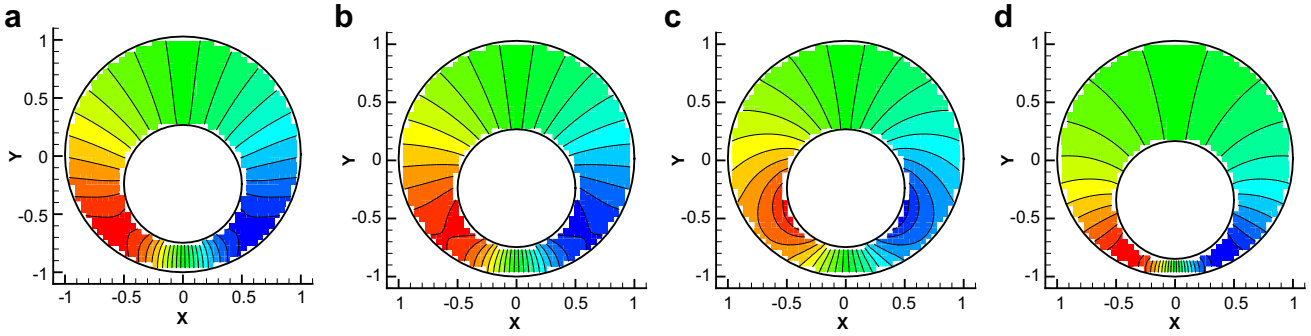

Fig. 13. Pressure contours corresponding to Fig. 12 for Example 6.3.

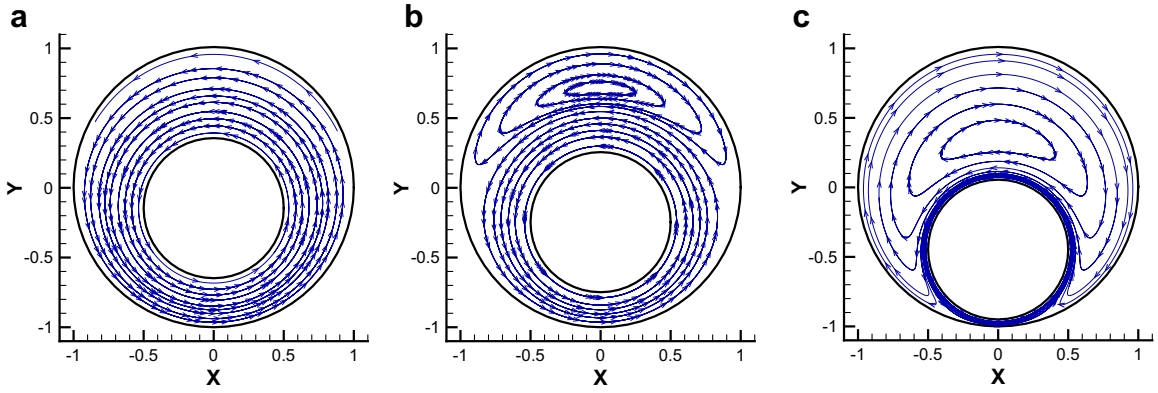

Fig. 14. Streamlines for the eccentricity ratio (a) $\lambda = 0.3$; (b) $\lambda = 0.5$; and (c) $\lambda = 0.9$ for Example 6.3.

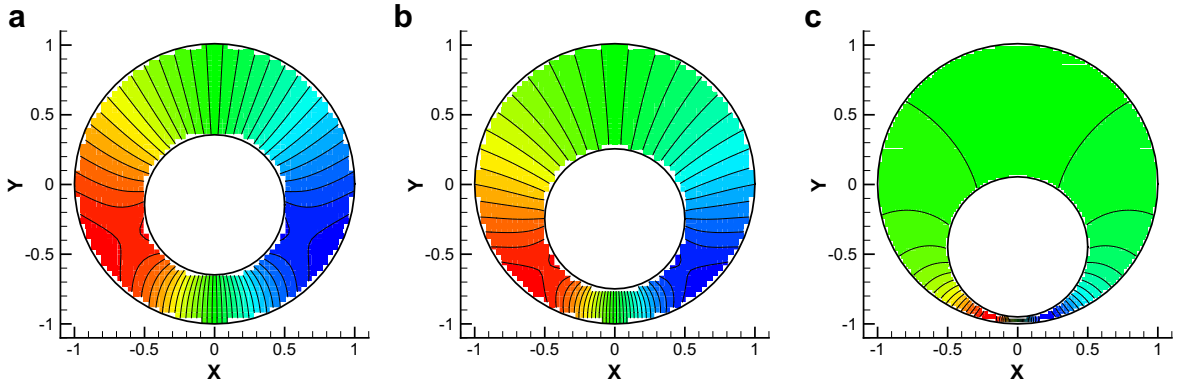

Fig. 15. Pressure contours corresponding to Fig. 14 for Example 6.3.

(i.e., the outer cylinder is stationary while the inner cylinder rotates with a constant angular velocity), flow separates into two regions: one rotating with the inner cylinder and a vortex zone. The direction of the vortex is opposite to the direction of the inner cylinder's rotation, see Fig. 12(a). In case 2, corresponding to outer cylinder rotation (i.e., the inner cylinder is stationary while the outer cylinder rotates with a constant angular velocity), there are also two flow regions. The directions of the vortex and the outer cylinder's rotation are the same as shown in Fig. 12(b) in contrast to case 1. In case 3, corresponding to counter-rotation (i.e., the cylinders rotate in the opposite directions), there are three flow regions: one rotating with the inner cylinder, one rotating with the outer cylinder and a vortex zone. The direction of vortex is opposite to the direction of the inner cylinder's rotation, see Fig. 12(c). In case 4, corresponding to co-rotation (i.e., the cylinders rotates in the same directions), compared with case 3, the direction of vortex is the same as the direction of the inner cylinder's rotation for this case as shown in Fig. 12(d). The results obtained by our method can be compared with those in [6,34]. These results are also comparable to those found in [10,11,48]. All are found to be in good agreement.

Finally, we want to study the effect of eccentricity ratio on the flow patterns. In Figs. 14 and 15, we shows the comparison of streamlines and the corresponding pressure contours for a rotating inner ($\omega_1 = 1$) and a stationary outer cylinder at different values of eccentricity ratio, respectively. These results can be compared with those given in [7,17]. Again they are found to be in reasonably good agreement. When two cylinders are very close together (i.e., the

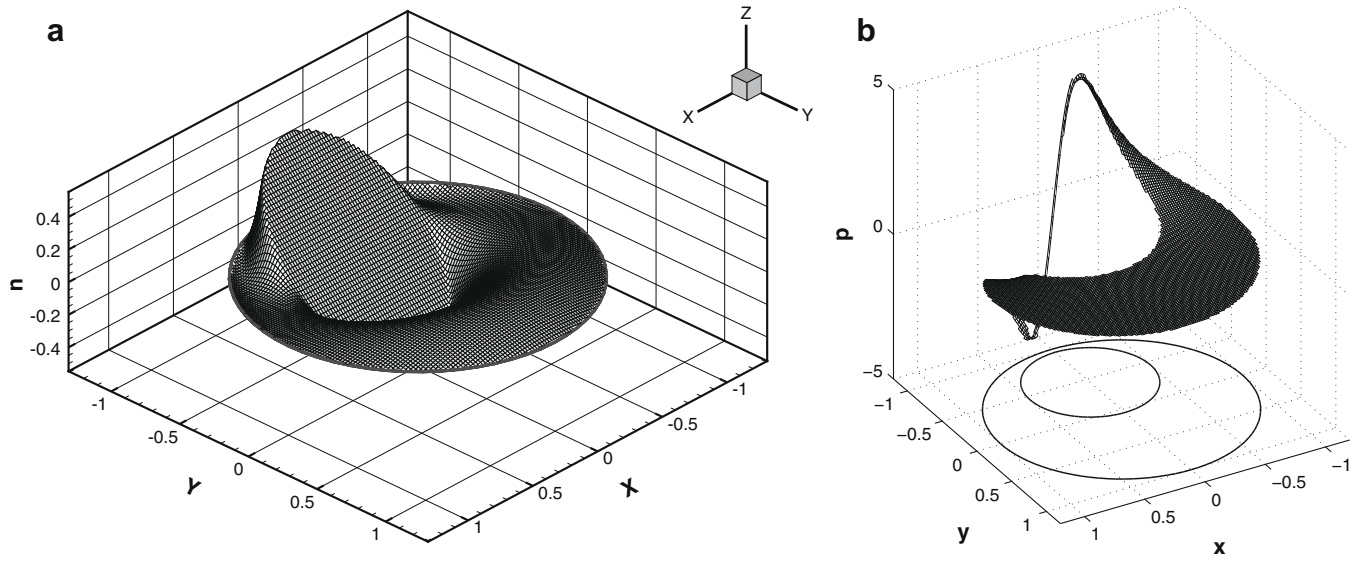

Fig. 16. (a) The x-component of velocity field $\mathbf{u}$ and (b) the pressure distribution with $\lambda = 0.9$ for Example 6.3.

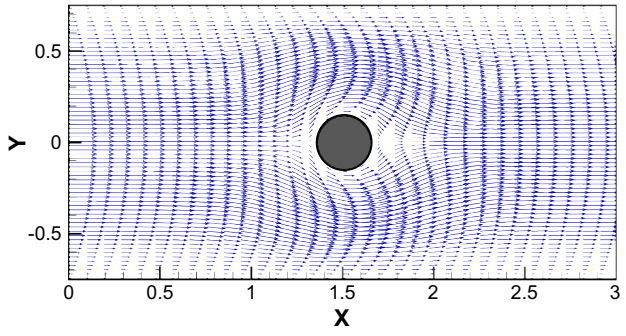

Fig. 17. Example 6.4 on flow past a circular cylinder. Vector plot of the velocity field.

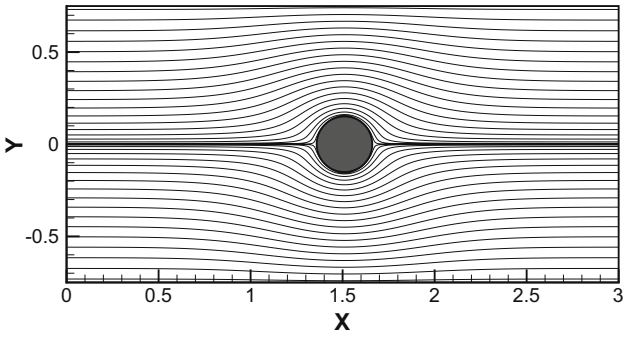

Fig. 18. Example 6.4 on flow past a circular cylinder. Streamlines.

eccentricity ratio λ is close to 1), we choose a finer grid size to guarantee that a few grids exist within this gap. In particular, for the eccentricity ratio $\lambda = 0.9$, we use a 128×128 grid. The x-component of the velocity $\mathbf{u}$ and the pressure distribution are presented in Fig. 16(a) and (b), respectively. From these figures, we can see that the non-smooth solution of the velocity and sharp interface solution of the pressure caused by the singularity at the rigid boundaries are captured very well by our method. This example demonstrates good capability of the present method to solve Stokes flow in multi-connected domains.

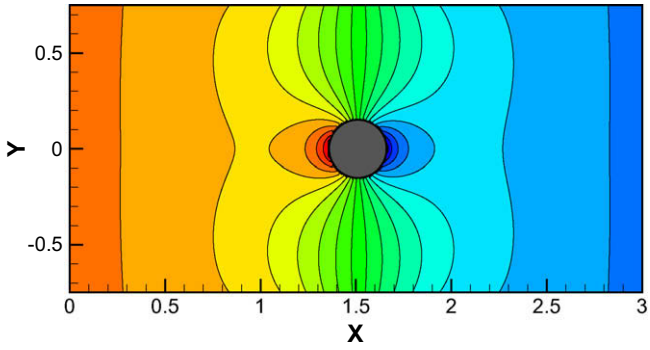

Fig. 19. Example 6.4 on flow past a circular cylinder. Pressure contours.

Example 6.4 (*Flow past a circular cylinder*). In this final example taken from [26], we simulate Stokes flows through a circular cylinder immersed in a rectangular domain $\Omega = [0,3] \times [-0.75, 0.75]$. The cylinder has a diameter $d = 0.3$ and its center is located at (1.5,0). At the inlet of the channel, the flow has a parabolic velocity profile with $U_{\max} = 1$, where $U_{\max}$ is the maximum value of the velocity. At the outlet of the channel, the same velocity profile is assumed, that is, the flow is assumed to have recovered from the disturbances by the cylinder placed in the middle section of the channel. On the upper and lower boundaries and all cylinder boundaries, no slip boundary conditions are assumed. We set the viscosity $\mu = 1$ and use 40 control points to represent the circular cylinder. Figs. 17 and 18 show the velocity vector and streamline plots within the fluid, respectively. From these figures it is observed that the flow is symmetric. The plot of the pressure contour is shown in Fig. 19. The pressure patterns are symmetric about the streamwise axis as expected. These results compare very well with that given in [26].

7. Concluding remarks

In this paper, we have presented an immersed interface algorithm for solving the incompressible steady Stokes equations on irregular domains. The method combines the immersed interface method with a front tacking representation of the boundary on a uniform Cartesian grid. The main advantage of the method is that the prescribed velocity condition at the rigid boundary is exactly

satisfied. The grid convergence analysis shows that current algorithm can achieve second order accuracy in both the velocity and pressure. We also show the capability of the present method to simulate the Stokes flow in multi-connected domains through the simulation of flow between two cylinders. It is rather straightforward to extend the current algorithm to solve for the problems with moving rigid geometry having prescribed velocity, which is our future work. Also as a next step, we will extend the present method to solve the incompressible Stokes flow problems involving deformable interfaces on irregular domains. As a very important application of that, in particular, we will study and simulate the realistic biological flow problems with low Reynolds number like the motion of the deformable cell and cell-trap simulation of red blood cell (RBC) in the complex micro-channel geometry. That will be reported in the near future.

Acknowledgements

The authors thank the referees for the valuable suggestions on the revision of the manuscript.

References

[1] Adams J, Swarztrauber P, Sweet R. FISHPACK: efficient FORTRAN subprograms for the solution of separable eliptic partial differential equations; 1999. Available from: http://www.scd.ucar.edu/css/software/fishpack/.
[2] Alves CJS, Silvestre AL. Density results using Stokeslets and a method of fundamental solutions for the Stokes equations. Eng Anal Bound Elem 2004;28:1245–52.
[3] Ballal BY, Rivlin RS. Flow of a Newtonian fluid between eccentric, rotating cylinders: inertial effects. Arch Rational Mech Anal 1976;62:237–94.
[4] Benzi M, Golub GH, Liesen J. Numerical solution of saddle point problems. Acta Numer 2005;14:1–137.
[5] Biros G, Ying L, Zorin D. A fast solver for the Stokes equations with distributed forces in complex geometries. J Comput Phys 2003;193:317–48.
[6] Calhoun D. A Cartesian grid method for solving the two-dimensional stream function–vorticity equations in irregular regions. J Comput Phys 2002;176:231–75.
[7] Chen JT, Hsiao CC, Leu SY. A new method for Stokes problems with circular boundaries using degenerate kernel and Fourier series. Int J Numer Meth Eng 2008;74:1955–87.
[8] Chen G, Li Z, Lin P. A fast finite difference method for biharmonic equations on irregular domains. Adv Comput Math 2008;29:113–33.
[9] Chen CW, Young DL, Tsai CC, Murugesan K. The method of fundamental solutions for inverse 2D Stokes problems. Comput Mech 2005;37:2–14.
[10] Chou MH. A multigrid finite difference approach to steady flow between eccentric rotating cylinders. Int J Numer Meth Fluids 2000;34:479–94.
[11] Chou MH. A multigrid pseudospectral method for steady flow computation. Int J Numer Meth Fluids 2003;43:25–42.
[12] Christoph B. Domain imbedding methods for the Stokes equations. Numer Math 1990;57:435–51.
[13] Elman HC. Multigrid and Krylov subspace methods for the discrete Stokes equations. Int J Numer Meth Fluids 1996;227:755–70.
[14] Elman HC. Preconditioners for saddle point problems arising in computational fluid dynamics. Appl Numer Math 2002;43:75–89.
[15] Fadlun EA, Verzicco R, Orlandi P. Combined immersed boundary finite-difference methods for three-dimensional complex flows simulations. J Comput Phys 2000;161:35–60.
[16] Fogelson AL. Continuum models of platelet aggregation: Formulation and mechanical properties. SIAM J Appl Math 1992;52:1089–110.
[17] Kim E. A mixed Galerkin method for computing the flow between eccentric rotation cylinders. Int J Numer Meth Fluids 1998;26:877–85.
[18] Lai M-C, Peskin CS. An immersed boundary method with formal second order accuracy and reduced numerical viscosity. J Comput Phys 2000;160:707–19.
[19] Le DV, Khoo BC, Peraire J. An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries. J Comput Phys 2006;220:109–38.
[20] Lee L, LeVeque RJ. An immersed interface method for incompressible Navier–Stokes equations. SIAM J Sci Comput 2003;25:832–56.
[21] LeVeque RJ, Li Z. Immersed interface methods for Stokes flow with elastic boundaries or surface tension. SIAM J Sci Comput 1997;18:709–35.
[22] LeVeque RJ, Li Z. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. SIAM J Numer Anal 1994;31:1019–44.
[23] Li Z, Ito K. The immersed interface method-numerical solutions of PDEs involving interfaces and irregular domains. SIAM Frontiers Appl Math 2006:33.
[24] Li Z, Lai MC. The immersed interface method for the Navier–Stokes equations with singular forces. J Comput Phys 2001;171:822–42.
[25] Li Z, Wang C. A fast finite difference method for solving Navier–Stokes equations on irregular domains. Commun Math Sci 2003;1:180–96.
[26] Liu YJ. A new fast multipole boundary element for solving 2-D stokes flow problems based on a dual formulation. Eng Anal Bound Elem 2008;32:139–51.
[27] Mohd-Yusof J. Combined immersed boundary/*B*-splines methods for simulations of flows in complex geometry, Annual Research Briefs, Center for Turbulence Research; 1997. p. 317–27.
[28] Oosterlee CW, Lorenz FJG. Multigrid methods for the Stokes system. Comput Sci Eng 2006;8:34–43.
[29] Peskin CS. Numerical analysis of blood flow in the heart. J Comput Phys 1977;25:220–52.
[30] Peskin CS. The immersed boundary method. Acta Numer 2002;11:479–517.
[31] Peters J, Reichelt V, Reusken A. Fast iterative solvers for discrete Stokes equations. SIAM J Sci Comput 2005;27:646–66.
[32] Phan A-V, Gray LJ, Kaplan T, Phan T-N. The boundary contour method for two-dimensional stokes flow and incompressible elastic materials. Comput Mech 2002;28:425–33.
[33] Russell D, Wang ZJ. A Cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow. J Comput Phys 2003;191:177–205.
[34] Rutka V. A staggered grid-based explicit jump immersed interface method for two-dimensional Stokes flows. Int J Numer Meth Fluids 2008;57:1527–43.
[35] Sarin V, Sameh A. An efficient iterative method for the generalized Stokes problem. SIAM J Sci Comput 1998;19:206–26.
[36] Schlichting H. Boundary-layer theory. New York: McGraw-Hill; 1987.
[37] Shin D, Strikwerda JC. Fast solvers for finite difference approximations for the Stokes and Navier–Stokes equations. J Aust Math Soc 1996;38:274–90.
[38] Lima E. Silva ALF, Silveira-Neto A, Damasceno JJR. Numerical simulation of two-dimensional flows over a circular cylinder using the immersed boundary method. J Comput Phys 2003;189:351–70.
[39] Tau EY. Numerical solution of the steady Stokes equations. J Comput Phys 1992;99:190–5.
[40] Tsai CC, Young DL, Lo DC, Wong TK. Method of fundamental solutions for three-dimensional Stokes flow in exterior field. J Eng Mech ASCE 2006;132:317–26.
[41] Udaykumar HS, Mittal R, Rampunggoon P, Khanna A. A sharp interface Cartesian grid method for simulating flows with complex moving boundaries. J Comput Phys 2001;174:345–80.
[42] Uhlmann M. An immersed boundary method with direct forcing for the simulation of particulate flows. J Comput Phys 2005;209:448–76.
[43] Wang NT, Fogelson AL. Computational methods for continuum models of platelet aggregation. J Comput Phys 1999;151:649–75.
[44] Wannier GH. A contribution to the hydrodynamics of lubrication. Q Appl Math 1950;8:1–32.
[45] Wiegmann A, Bube KP. The explicit-jump immersed interface method: finite difference methods for PDEs with piecewise smooth solutions. SIAM J Numer Anal 2000;37:827–62.
[46] Ye T, Mittal R, Udaykumar HS, Shyy W. An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundary. J Comput Phys 1999;156:209–40.
[47] Young DL, Chen CW, Fan CM, Murugesan K, Tsai CC. Method of fundamental solutions for Stokes flows in a rectangular cavity with cylinders. Eur J Mech B 2005;24:703–16.
[48] Young DL, Chiu CL, Fan CM, Tsai CC, Lin YC. Method of fundamental solutions for multidimensional Stokes equations by the dual-potential formulation. Eur J Mech B 2006;25:877–93.
[49] Young DL, Jane SJ, Fan CM, Murugesan K, Tsai CC. The method of fundamental solutions for 2D and 3D Stokes problems. J Comput Phys 2006;211:1–8.
[50] Xu S, Wang ZJ. An immersed interface method for simulating the interaction of a fluid with moving boundaries. J Comput Phys 2006;216:454–93.