Available online at www.sciencedirect.com
ScienceDirect
Computers & Fluids 37 (2008) 99–106
computers & fluids
www.elsevier.com/locate/compfluid

A simple implementation of the immersed interface methods for Stokes flows with singular forces

Ming-Chih Lai *, Hsiao-Chieh Tseng

Department of Applied Mathematics, National Chiao Tung University, 1001, Ta Hsueh Road, Hsinchu 300, Taiwan

Received 11 November 2006; received in revised form 15 April 2007; accepted 27 April 2007
Available online 21 June 2007

Abstract

In this paper, we introduce a simple version of the immersed interface method (IIM) for Stokes flows with singular forces along an interface. The numerical method is based on applying the Taylor's expansions along the normal direction and incorporating the solution and its normal derivative jumps into the finite difference approximations. The fluid variables are solved in a staggered grid, and a new accurate interpolating scheme for the non-smooth velocity has been developed. The numerical results show that the scheme is second-order accurate.

1. Introduction

The fluid problems with interfaces have many applications in science and engineering. For example, Peskin's formulation for the blood flow in the heart valve leaflet (interface) involves exerting singular forces along the leaflet into the blood (fluid) [19]. Another example is the Hele-Shaw flow which can be formed by pumping a more viscous fluid into a less viscous surrounding fluid in two parallel thin plates. The shape of the interface is known to exhibit a fingering phenomenon. The governing equations for the above problems involve solving elliptic partial differential equations with possible discontinuous coefficients or with possible jump conditions for the unknown and its derivative across the interface. Several classes of practical finite difference methods have been developed in the past three decades. Those methods provide different treatments near the interface while keep the standard difference discretization away from the interface. We summarize those different techniques as follows.

The immersed boundary (IB) method of Peskin [20] is the first Cartesian grid method to provide a simple solution to the fluid problems interacting with complicated interfaces. The method is to treat the interface as a singular force (source) generator along which the force can be transmitted into the fluid smoothly. More precisely, the method uses the Lagrangian markers to track the interface and affects the fluid in Eulerian grid via a smooth version of discrete delta function. Since the equations are solved on a Cartesian grid, many fast fluid solvers can be applied easily. For instance, the original IB method exploits Fast Fourier Transform (FFT) in solving the Stokes equations in rectangular domain. The method is easy to implement but it is only first-order accurate. Besides, the solution is smoothing out across the interface. Although there are different variants of formally second-order schemes recently [9,7], the overall accuracy can be increased only for the sufficiently smooth problems. Despite this first-order accuracy issue, the IB method nowadays is getting more attention on applying to the flow interacting with rigid boundaries which has important applications in computational fluid dynamics. The reason for that is quite simple since it does not need to handle grid generations for unsteady problems which can save the computational effort significantly.

Unlike the IB method to have numerical smearing near the interface, the immersed interface method was invented to capture the solution and its derivative jumps sharply.

* Corresponding author.
E-mail address: mclai@math.nctu.edu.tw (M.-C. Lai).

0045-7930/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2007.04.003

The idea is to incorporate those jump conditions into the finite difference discretization by modifying the difference approximations near the interface. Mayo [18] used the idea to solve the Poisson equation in an irregular domain based on an integral equation. Later, LeVeque and Li applied the similar idea to an elliptic interface problem with discontinuous coefficients and singular sources [13]. Since the problem is more difficult to solve than the Poisson problem, they introduce the local coordinates and derive more interface relations so that the finite difference scheme can match up to all second-order partial derivatives. Their method appears to be the first finite difference method that is second-order accurate for the problem. Dumett and Keener has extended the method to the anisotropic case of elliptic interface problems [6]. There are other applications to the fluid problems [12,16,15,23], just to name a few. We refer the interested readers to Li's recent review article [10] and the newly published book [11].

There are several methods sharing the similar spirit as IIM in literature. For example, the boundary condition capturing method proposed by Liu et al. [17] can be implemented to solve the elliptic interface problems dimension by dimension. The method captures the solution and its normal derivative jumps sharply while smoothing the tangential derivative. The method is easy to implement; however, it is only first-order accurate. There are other variants of IIM such as the explicit jump [24] and the decomposed immersed interface method [4]. Beale and Lai [2] applied the boundary integral method to compute the solution for the Laplace problem near the interface and used it to form the correction terms. This method has been further extended to the Stokes problem with singular forces [3].

In this paper, we extend the immersed interface method developed in [21] for Poisson problems to Stokes problems with singular forces along an interface. The numerical method is based on applying the Taylor's expansions along the normal direction and incorporating the solution and its normal derivative jumps into the finite difference approximations. Unlike the traditional IIM, the present fluid variables are solved in a staggered grid [8] instead. Another contribution of this work is to derive a new accurate interpolation scheme for the non-smooth velocity field. The numerical results show that the scheme is indeed second-order accurate.

The paper is organized as follows. In Section 2, we review the immersed interface methods for Poisson problems with an interface and introduce a new interpolation scheme from a grid function to the interface. Then the numerical scheme for Stokes flows with singular forces is discussed and the numerical results are shown in Section 3. Some conclusions are given in Section 4.

2. Poisson equations with jump discontinuities across the interface

In this section, we consider the Poisson problem on a computational domain $\Omega = [a,b]\times[c,d]$ with an immersed interface $\Gamma = \{\mathbf{X}(s) = (X(s), Y(s)), 0 \leqslant s < L\}$, where s is the arc-length parameter and $\mathbf{X}(0) = \mathbf{X}(L)$. The interface Γ divides the domain Ω into two regions,; namely inside (Ω^-) and outside (Ω^+) the interface. Across Γ, the solution and its normal derivative exhibit jump discontinuities. The Poisson problem can be written as

$$\Delta u = f \quad \text{in } \Omega, \tag{2.1}$$

$$[u](s) = g(s), \quad \left[\frac{\partial u}{\partial \mathbf{n}}\right](s) = h(s) \quad \text{on } \Gamma, \tag{2.2}$$

$$u = u_{\mathrm{b}} \quad \text{on } \partial\Omega, \tag{2.3}$$

where the right-hand side function f could be discontinuous across the interface Γ. The jumps $[u](s) = u^+ - u^-$ and $\left[\frac{\partial u}{\partial \mathbf{n}}\right](s) = \frac{\partial u^+}{\partial \mathbf{n}} - \frac{\partial u^-}{\partial \mathbf{n}}$ are defined as the difference of the limiting values from two different sides of the interface. Here, we also assume the interface parametrization $\mathbf{X}(s)$ and the jump discontinuity functions $g(s)$ and $h(s)$ are fairly smooth (say $C^2(\Gamma)$) on the interface.

2.1. An IIM to incorporate the jumps in the normal direction

In this subsection, we review the scheme developed in [21] to solve Eqs. (2.1)–(2.3). Before we proceed, let us lay out a uniform Cartesian grid in Ω with mesh width $h = \Delta x = \Delta y$, and let the discretized solution at the grid point $\mathbf{x}_{i,j} = (x_i, y_j) = (a + i\Delta x, c + j\Delta y)$ be denoted by $u_{i,j}$. As in [18], we classify the grid point as either a regular or irregular point. For a regular grid point, we mean that the interface does not cut through the standard five-point Laplacian of that point. On the other hand, if the five-point Laplacian of a grid point involves using the grid points inside and outside the interface simultaneously, then we call such point as an irregular point. Since the solution is smooth either inside or outside the interface, the five-point Laplacian of a regular point does not need to modify in order to have the second-order accuracy. However, it has to be modified at an irregular point since the solution is not smooth across the interface, and the modification depends on the jumps $[u]$ and $\left[\frac{\partial u}{\partial \mathbf{n}}\right]$. We explain how to modify the discrete Laplacian in the following.

Let $\mathbf{x}_{i,j}$ be an irregular grid point in Ω^- as shown in Fig. 1. The five-point Laplacian at $\mathbf{x}_{i,j}$ can be written as

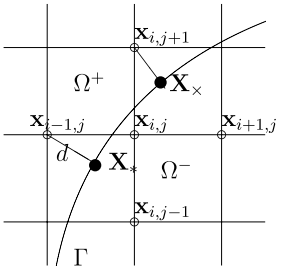

Fig. 1. The five-point Laplacian of the irregular point $\mathbf{x}_{i,j}$.

$$\begin{aligned}\Delta_h u(\mathbf{x}_{i,j}) &= \frac{u_{i-1,j}-2u_{i,j}+u_{i+1,j}}{h^2}+\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{h^2}\\ &= \frac{u^+_{i-1,j}-2u^-_{i,j}+u^-_{i+1,j}}{h^2}+\frac{u^+_{i,j+1}-2u^-_{i,j}+u^-_{i,j-1}}{h^2}\\ &= \frac{u^-_{i-1,j}-2u^-_{i,j}+u^-_{i+1,j}}{h^2}+\frac{u^-_{i,j+1}-2u^-_{i,j}+u^-_{i,j-1}}{h^2}\\ &\quad+\frac{u^+_{i-1,j}-u^-_{i-1,j}}{h^2}+\frac{u^+_{i,j+1}-u^-_{i,j+1}}{h^2}\\ &= u_{xx}(\mathbf{x}_{i,j})+u_{yy}(\mathbf{x}_{i,j})+\mathcal{O}(h^2)+\frac{u^c_{i-1,j}}{h^2}+\frac{u^c_{i,j+1}}{h^2}\\ &= f_{i,j}+\frac{1}{h^2}\left(u^c_{i-1,j}+u^c_{i,j+1}\right)+\mathcal{O}(h^2). \end{aligned} \tag{2.4}$$

To derive the correction term $u^c_{i-1,j}$, we need to find the orthogonal projection of $\mathbf{x}_{i-1,j}$ at the interface (say $\mathbf{X}_* = \mathbf{X}(s_*)$ in Fig. 1), and then apply the Taylor's expansions along the normal direction at $\mathbf{X}_*$. (Note that, there are two different techniques of finding the orthogonal projection of a grid point on the interface based on the interface representation; namely, the arc-length parametrization and the level set representation. We refer the interested readers to the review article [10].) So we have

$$\begin{aligned}u^c_{i-1,j} &= u^+_{i-1,j}-u^-_{i-1,j}\\ &= \left(u^+_* + d\frac{\partial u^+_*}{\partial \mathbf{n}}+\frac{d^2}{2}\frac{\partial^2 u^+_*}{\partial \mathbf{n}^2}+\mathcal{O}(h^3)\right)\\ &\quad-\left(u^-_* + d\frac{\partial u^-_*}{\partial \mathbf{n}}+\frac{d^2}{2}\frac{\partial^2 u^-_*}{\partial \mathbf{n}^2}+\mathcal{O}(h^3)\right)\\ &= [u]_{\mathbf{X}_*}+d\left[\frac{\partial u}{\partial \mathbf{n}}\right]_{\mathbf{X}_*}+\frac{d^2}{2}\left[\frac{\partial^2 u}{\partial \mathbf{n}^2}\right]_{\mathbf{X}_*}+\mathcal{O}(h^3). \end{aligned} \tag{2.5}$$

The value d is the signed distance between the grid point $\mathbf{x}_{i-1,j}$ and the orthogonal projection $\mathbf{X}_*$. Notice that, if the grid point $\mathbf{x}_{i-1,j}$ is inside the interface, then d must be negative.

In Eq. (2.5), the jumps $[u]_{\mathbf{X}_*}$ and $\left[\frac{\partial u}{\partial \mathbf{n}}\right]_{\mathbf{X}_*}$ are simply given from Eq. (2.2); however, the second normal derivative jump $\left[\frac{\partial^2 u}{\partial \mathbf{n}^2}\right]_{\mathbf{X}_*}$ is not known. To find the second normal derivative jump, one can first rewrite the Laplace operator in Eq. (2.1) on the interface as in [22]

$$\frac{\partial^2 u}{\partial \mathbf{n}^2}(\mathbf{X}(s))+\kappa(\mathbf{X}(s))\frac{\partial u}{\partial \mathbf{n}}(\mathbf{X}(s))+\frac{\partial^2 u}{\partial s^2}(\mathbf{X}(s))=f(\mathbf{X}(s)), \tag{2.6}$$

where the value $\kappa(\mathbf{X}(s))$ is the local curvature of the interface. Then, we can easily compute the second normal derivative jump at $\mathbf{X}_*$ by

$$\left[\frac{\partial^2 u}{\partial \mathbf{n}^2}\right]_{\mathbf{X}_*} = [f]_{\mathbf{X}_*}-\kappa_{\mathbf{X}_*}\left[\frac{\partial u}{\partial \mathbf{n}}\right]_{\mathbf{X}_*}-\frac{\partial^2}{\partial s^2}[u]_{\mathbf{X}_*}. \tag{2.7}$$

Finally, the correction term can be approximated by

$$\begin{aligned}u^c_{i-1,j} &= [u]_{\mathbf{X}_*}+d\left[\frac{\partial u}{\partial \mathbf{n}}\right]_{\mathbf{X}_*}+\frac{d^2}{2}\left[\frac{\partial^2 u}{\partial \mathbf{n}^2}\right]_{\mathbf{X}_*}\\ &= [u]_{\mathbf{X}_*}+d\left[\frac{\partial u}{\partial \mathbf{n}}\right]_{\mathbf{X}_*}+\frac{d^2}{2}\left([f]_{\mathbf{X}_*}-\kappa_{\mathbf{X}_*}\left[\frac{\partial u}{\partial \mathbf{n}}\right]_{\mathbf{X}_*}-\frac{\partial^2}{\partial s^2}[u]_{\mathbf{X}_*}\right). \end{aligned} \tag{2.8}$$

The correction term $u^c_{i,j+1}$ can be evaluated in a similar way.

It is worth mentioning that the correction term $u^c_{i-1,j}$ must be computed up to third-order accurate since it appears in the discrete Laplacian term, that is, a term of $\mathcal{O}(1/h^2)$. Thus, the local truncation error at an irregular point is $\mathcal{O}(h)$. It has been proved by different authors [3,11] that despite the fact of the first-order truncation errors at those irregular points (but second-order accurate at regular points), the overall accuracy is still second-order. Since the correction terms are only involved some modifications in the right-hand side of the finite difference equations, the resultant linear system (2.4) can be efficiently solved by a fast Poisson solver such as Fishpack [1]. The computational cost spent on the modifications at irregular grid points is just a small portion of that for a fast Poisson solver.

2.2. An interpolation scheme from a grid function to the interface

In the Stokes problems, once we have obtained the velocity field on the grid, we need to find the velocity on the interface. Thus, a comparably accurate interpolation formula from a grid function to the interface markers needs to be developed. When the velocity field is smooth, the standard bi-linear interpolation can be used to achieve the second-order accuracy. However, if the velocity is not smooth across the interface ($[u]=0$, $\left[\frac{\partial u}{\partial \mathbf{n}}\right]\neq 0$, as in the next section), the bi-linear formula needs to be modified in order to have second-order accurate interpolations. In the following, we derive such modification for the case that even the jump of u is not zero. Notice that, this new scheme differs from the one used in [14] where the authors used a linear combination of the three nearby grid values with a correction term that consists of the function jump and the partial derivative jumps in both x- and y-directions at the interpolated marker. Here, however, we simply modify the standard bi-linear interpolation formula (involving four neighboring grid values) with the incorporation of the function and its normal derivative jumps which are given directly from the problem. The detail can be explained as follows.

Suppose we want to interpolate the value of u^-_{I} at a marker $\mathbf{X}_{\mathrm{I}} = (X_{\mathrm{I}}, Y_{\mathrm{I}}) = (x_{i,j}+\alpha h, y_{i,j}+\beta h)$ from the inside of the interface using surrounding values $u_{i,j}, u_{i+1,j}, u_{i,j+1}$ and $u_{i+1,j+1}$ as illustrated in Fig. 2. Let C^-_{I} be the interpolating correction term that should be added to the standard bi-linear formula so that the interpolation accuracy is second-order. That is

$$u^-_{\mathrm{I}} = u_{\mathrm{BI}}+C^-_{\mathrm{I}}+\mathcal{O}(h^2), \tag{2.9}$$

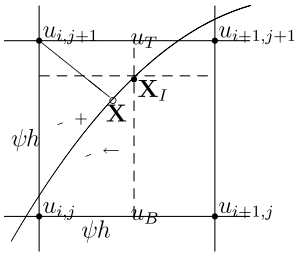

Fig. 2. A diagram of the interpolation from a grid function to an interface marker X_I.

where u_{BI} is the bi-linear interpolation of u at X_I. To derive C_I^-, we have

$$\begin{aligned} u_{BI} &= \beta u_T + (1-\beta)u_B \\ &= \beta((1-\alpha)u^+_{i,j+1} + \alpha u^-_{i+1,j+1}) \\ &\quad + (1-\beta)((1-\alpha)u^-_{i,j} + \alpha u^-_{i+1,j}) \\ &= \beta((1-\alpha)u^-_{i,j+1} + \alpha u^-_{i+1,j+1}) + (1-\beta)((1-\alpha)u^-_{i,j} \\ &\quad + \alpha u^-_{i+1,j}) + \beta(1-\alpha)u^c_{i,j+1} \\ &= u_I^- + \beta(1-\alpha)u^c_{i,j+1}, \end{aligned}$$

where the correction $u^c_{i,j+1}$ can be derived exactly the same procedure as Eq. (2.5). However, since we only need to approximate the correction up to $\mathcal{O}(h^2)$, one can neglect the second normal derivative jump, that is

$$u^c_{i,j+1} = u^+_{i,j+1} - u^-_{i,j+1} = [u]_{\mathbf{X}_*} + d\left[\frac{\partial u}{\partial \mathbf{n}}\right]_{\mathbf{X}_*} + \mathcal{O}(h^2), \tag{2.10}$$

where $\mathbf{X}_*$ is the orthogonal projection of the grid point $\mathbf{x}_{i,j+1}$ on the interface. Therefore, the interpolating correction term can be derived as

$$C_I^- = -\beta(1-\alpha)\left([u]_{\mathbf{X}_*} + d\left[\frac{\partial u}{\partial \mathbf{n}}\right]_{\mathbf{X}_*}\right). \tag{2.11}$$

2.3. Numerical examples

We consider three different exact solutions (shown in Table 1) for Poisson equation in $\Omega = [-1,1]\times[-1,1]$ with jump discontinuities on an elliptical interface $\Gamma = \{\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1\}$ with $a = 0.8$, and $b = 0.2$. In [21], the similar examples were used but with a circle interface so that the orthogonal projection point can be found straightforwardly and the implementation is much simpler. In our present code, we have also implemented the uniformly chosen markers to construct the interface and thus from which to find the necessary orthogonal projections and the curvatures. Our numerical scheme and interface construction can be applied to any smooth curve in principle.

Table 2 shows the maximum errors of $N\times N$ grid points in Ω. For Examples 1 and 2, we can see that the present scheme is indeed second-order convergent. For Example 3, the discrete Laplacian approximates the Laplacian operator exactly so that the error is roughly equal to the machine precision. Those numerical results behave similarly as the ones obtained in [21] for the circular interface problems. Throughout this paper, all of numerical results were obtained by running our Fortran 90 code in a PC of Intel Pentium 4 (3.00 GHz) processor with 4 GB RAM. We also list the CPU time (s) in Table 2 for our computations. One can see that even a grid size 320×320, the overall CPU time takes less than a quarter of a second.

Once we obtain the solution on the grid, we can interpolate the value on the interface from inside by using the technique discussed before. Note that, the solution and its normal derivative are not continuous across the interface Γ. Table 3 shows the maximum errors of the interpolating solution (from inside) of Example 2 on the interface markers $(X_k, Y_k) = (0.8\cos\theta_k, 0.2\sin\theta_k), k = 0,1,\ldots,N-1$ with $\theta_k = k\Delta\theta$, $\Delta\theta = 2\pi/N$. One can see the present interpolating accuracy is also close to second-order.

Table 1
Three test examples for Poisson problem (2.1)–(2.3) with an elliptic interface $\frac{x^2}{(0.8)^2} + \frac{y^2}{(0.2)^2} = 1$ in $\Omega = [-1,1]\times[-1,1]$

	Example 1	Example 2	Example 3
u_-	1	$\exp(x)\cos(y)$	$x^2 - y^2$
u_+	$1 + \ln(2\sqrt{x^2+y^2})$	$\exp(x^2)\cos(y)$	0
f_-	0	0	0
f_+	0	$(1+4x^2)\exp(x^2)\cos(y)$	0

Table 2
The maximum errors for the examples of Poisson problem on the $N\times N$ grid points

N	$\|u - u_e\|_\infty$	Ratio	CPU time
Example 1			
40	2.1577E−03	–	0.015
80	6.3698E−04	3.39	0.016
160	1.7153E−04	3.71	0.046
320	4.0663E−05	4.22	0.219
Example 2			
40	1.1909E−03	3.71	0.015
80	3.0901E−04	3.85	0.016
160	7.8497E−05	3.94	0.078
320	1.9776E−05	3.97	0.235
Example 3			
40	5.5511E−16	–	0.015
80	7.9797E−16	–	0.016
160	2.6749E−15	–	0.047
320	1.5377E−14	–	0.172

Table 3
The interpolation errors for Example 2 of Poisson problem on the interface markers $(X_k, Y_k) = (0.8\cos\theta_k, 0.2\sin\theta_k), k = 0,1,\ldots,N-1$ with $\theta_k = k\Delta\theta$, $\Delta\theta = 2\pi/N$

N	$\|u^- - u_e^-\|_{\infty,\Gamma}$	Ratio
40	1.6036E−03	–
80	4.7394E−04	3.38
160	1.2650E−04	3.75
320	4.0435E−05	3.13

3. Stokes flow with singular forces

In this section, we consider the following Stokes equations:

$$-\nabla p + \mu\Delta\mathbf{u} + \mathbf{f} + \mathbf{g} = 0 \quad \text{in } \Omega, \tag{3.1}$$

$$\nabla\cdot\mathbf{u} = 0 \quad \text{in } \Omega, \tag{3.2}$$

$$\mathbf{u} = \mathbf{u}_b \quad \text{on } \partial\Omega, \tag{3.3}$$

where $\mathbf{u} = (u(\mathbf{x}), v(\mathbf{x}))$ is the fluid velocity, $p = p(\mathbf{x})$ is the pressure, and μ is the constant viscosity. There are two different forces acting to the fluid; namely, the singular force $\mathbf{f}$ exerted by the immersed interface (immersed boundary) and the external force $\mathbf{g}$ (could be discontinuous). More precisely, the singular force $\mathbf{f}$ has a support only along the interface $\Gamma = \{\mathbf{X}(s) = (X(s), Y(s)), 0 \leqslant s \leqslant L\}$ and it has the form

$$\mathbf{f}(\mathbf{x}) = \int_\Gamma \mathbf{F}(s)\delta(\mathbf{x} - \mathbf{X}(s))\mathrm{d}s, \tag{3.4}$$

where δ is the 2D Dirac delta function. The boundary force $\mathbf{F}$ defined on the interface can be further decomposed to its normal ($\mathbf{n} = (n_1, n_2)$) and tangential ($\boldsymbol{\tau} = (\tau_1, \tau_2)$) directions as

$$\mathbf{F} = (F_1, F_2) = (\mathbf{F}\cdot\mathbf{n})\mathbf{n} + (\mathbf{F}\cdot\boldsymbol{\tau})\boldsymbol{\tau} = F_{\mathbf{n}}\mathbf{n} + F_\tau\boldsymbol{\tau}, \tag{3.5}$$

where $F_{\mathbf{n}}$ and F_τ represent the normal and tangential forces, respectively.

Since the force $\mathbf{f}$ has a delta function singularity along the interface, one can expect that the velocity and the pressure will not be smooth across the interface. In fact, the boundary force $\mathbf{F}$ plays an important role to the jump conditions of $\mathbf{u}$ and p, and their derivatives. In IIM, we usually reformulate the Eqs. (3.1)–(3.5) into a Stokes problem without the delta function force term but with some known jump conditions across the interface. Thus, the governing equations are summarized as follows [14]:

$$-\nabla p + \mu\Delta\mathbf{u} + \mathbf{g} = 0 \quad \text{in } \Omega, \tag{3.6}$$

$$\nabla\cdot\mathbf{u} = 0 \quad \text{in } \Omega, \tag{3.7}$$

$$[\mathbf{u}] = 0, \quad \mu\left[\frac{\partial\mathbf{u}}{\partial\mathbf{n}}\right] = -F_\tau\boldsymbol{\tau} \quad \text{on } \Gamma, \tag{3.8}$$

$$[p] = F_{\mathbf{n}}, \quad \left[\frac{\partial p}{\partial\mathbf{n}}\right] = \frac{\partial F_\tau}{\partial s} + [\mathbf{g}]\cdot\mathbf{n} \quad \text{on } \Gamma, \tag{3.9}$$

$$\mathbf{u} = \mathbf{u}_b \quad \text{on } \partial\Omega. \tag{3.10}$$

3.1. Numerical scheme

Unlike the traditional IIM implementation, we use a staggered grid [8] with a uniform mesh width $h = \Delta x = \Delta y$ for the fluid velocity and the pressure as in Fig. 3. By applying the divergence operator to Eq. (3.6) and using the incompressibility constraint of Eq. (3.7), the pressure satisfies the following equation:

$$\Delta p = \nabla\cdot\mathbf{g} \quad \text{in } \Omega, \tag{3.11}$$

$$[p] = F_{\mathbf{n}}, \quad \left[\frac{\partial p}{\partial\mathbf{n}}\right] = \frac{\partial F_\tau}{\partial s} + [\mathbf{g}]\cdot\mathbf{n} \quad \text{on } \Gamma. \tag{3.12}$$

This is exactly the type of Poisson equation that we have discussed in the previous section so the scheme discussed before can be applied directly. One remaining question is how to choose the pressure boundary condition on the computational boundary $\partial\Omega$. In this work, for testing purpose, the pressure boundary condition can be chosen either a Dirichlet (Examples 1 and 2 in next subsection) or a zero Neumann boundary condition (Example 3 in next subsection).

Once we solve the pressure, the velocity field can be obtained by solving other two Poisson-type equations with the interface as

$$\Delta\mathbf{u} = \frac{1}{\mu}(\nabla p - \mathbf{g}) \quad \text{in } \Omega, \tag{3.13}$$

$$[\mathbf{u}] = 0, \quad \left[\frac{\partial\mathbf{u}}{\partial\mathbf{n}}\right] = -\frac{1}{\mu}F_\tau\boldsymbol{\tau} \quad \text{on } \Gamma, \tag{3.14}$$

$$\mathbf{u} = \mathbf{u}_b \quad \text{on } \partial\Omega. \tag{3.15}$$

Once again, the numerical scheme in previous section can also be applied directly to find the velocity. However, in order to compute the pressure gradient at the right-hand side of Eq. (3.13) accurately, one has to take its jump across the interface into account. To derive the jumps $[p_x]$ and $[p_y]$, we first recall that the unit tangent vector can be written as $\boldsymbol{\tau} = (\tau_1, \tau_2) = (-n_2, n_1)$ at each point on Γ. Since

$$\left[\frac{\partial p}{\partial\mathbf{n}}\right] = [p_x]n_1 + [p_y]n_2 = \frac{\partial F_\tau}{\partial s} + [\mathbf{g}]\cdot\mathbf{n}, \tag{3.16}$$

$$\frac{\partial}{\partial s}[p] = [p_x]\tau_1 + [p_y]\tau_2 = -[p_x]n_2 + [p_y]n_1 = \frac{\partial F_{\mathbf{n}}}{\partial s}, \tag{3.17}$$

we can easily obtain the pressure gradient jump $[\nabla p] = ([p_x], [p_y])$ as

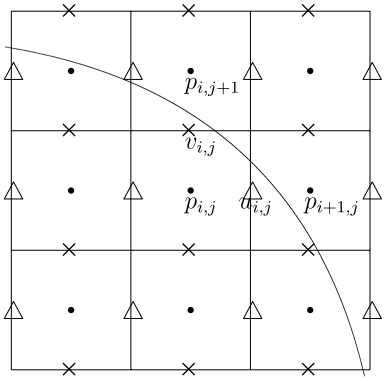

Fig. 3. A diagram of the interface cutting through a staggered grid with a uniform mesh width h in Ω. The pressure is located at the center of the cell, the velocity component u is at the left-right face, and v is at the top-bottom face of a cell.

$$[p_x] = n_1\left(\frac{\partial F_\tau}{\partial s} + [\mathbf{g}]\cdot\mathbf{n}\right) - n_2\frac{\partial F_\mathbf{n}}{\partial s}, \tag{3.18}$$

$$[p_y] = n_2\left(\frac{\partial F_\tau}{\partial s} + [\mathbf{g}]\cdot\mathbf{n}\right) + n_1\frac{\partial F_\mathbf{n}}{\partial s}. \tag{3.19}$$

We summarize our numerical algorithm as follows:

Step 1: Solve the pressure Poisson equation

$$\Delta_h p = \nabla\cdot\mathbf{g} + \frac{p^c}{h^2} \quad \text{in } \Omega \tag{3.20}$$

with modified right-hand side functions at those irregular points which the modifications are based on the jump conditions of the pressure. More precisely, the correction terms at those irregular points can be computed as

$$\begin{aligned} p^c &= [p] + \alpha\left[\frac{\partial p}{\partial \mathbf{n}}\right] + \frac{1}{2}\alpha^2\left([\Delta p] - \kappa\left[\frac{\partial p}{\partial \mathbf{n}}\right] - \frac{\partial^2[p]}{\partial s^2}\right) \\ &= F_\mathbf{n} + \alpha\left(\frac{\partial F_\tau}{\partial s} + [\mathbf{g}]\cdot\mathbf{n}\right) \\ &\quad + \frac{1}{2}\alpha^2\left([\nabla\cdot\mathbf{g}] - \kappa\left(\frac{\partial F_\tau}{\partial s} + [\mathbf{g}]\cdot\mathbf{n}\right) - \frac{\partial^2 F_\mathbf{n}}{\partial s^2}\right). \end{aligned} \tag{3.21}$$

Step 2: Solve the velocity equations

$$\Delta_h\mathbf{u} = \frac{1}{\mu}(\nabla_h p - \mathbf{g}) + \frac{\mathbf{u}^c}{h^2} \quad \text{in } \Omega, \tag{3.22}$$

$$\mathbf{u} = \mathbf{u}_\mathrm{b} \quad \text{on } \partial\Omega, \tag{3.23}$$

where the corrections are made only at irregular points as

$$\begin{aligned} \mathbf{u}^c &= [\mathbf{u}] + \alpha\left[\frac{\partial \mathbf{u}}{\partial \mathbf{n}}\right] + \frac{1}{2}\alpha^2\left([\Delta\mathbf{u}] - \kappa\left[\frac{\partial \mathbf{u}}{\partial \mathbf{n}}\right] - \frac{\partial^2[\mathbf{u}]}{\partial s^2}\right) \\ &= \frac{1}{\mu}\left(-\alpha F_\tau\boldsymbol{\tau} + \frac{1}{2}\alpha^2([\nabla p] - [\mathbf{g}] + \kappa F_\tau\boldsymbol{\tau})\right). \end{aligned} \tag{3.24}$$

Note that, the above pressure gradient jump has been derived from (3.18) and (3.19).

The remaining question is how to approximate the pressure gradient in Eq. (3.22). At those regular points, the derivatives p_x and p_y at the locations of $u_{i,j}$ and $v_{i,j}$ can be approximated by centered differences $(p_{i+1,j} - p_{i,j})/h$, and $(p_{i,j+1} - p_{i,j})/h$, respectively. However, a correction term must be added when the p-grid points straddle across the interface. Suppose the locations of $p_{i,j}$ and $u_{i,j}$ fall in the same side of the interface while the location of $p_{i+1,j}$ falls in another side of the interface as illustrated in Fig. 3, then we need to add a correction term $p^c_{i+1,j}$ of $p_{i+1,j}$ so that the approximation of p_x at $u_{i,j}$ grid can be computed by $((p_{i+1,j} + p^c_{i+1,j}) - p_{i,j})/h$. Note that, the correction term $p^c_{i+1,j}$ can be computed by the formula (3.21). On the other hand, if the locations of $p_{i+1,j}$ and $u_{i,j}$ fall in the same side of the interface while the location of $p_{i,j}$ falls in another side of the interface, then the approximation of p_x at $u_{i,j}$ grid can be computed by $(p_{i+1,j} - (p_{i,j} + p^c_{i,j}))/h$, where a correction term $p^c_{i,j}$ of $p_{i,j}$ must be added. The approximation of p_y at the $v_{i,j}$ grid point can be handled in a similar manner.

3.2. Numerical results

In this subsection, we perform three different numerical tests to examine the accuracy of the present numerical schemes for Stokes problems. Throughout the following examples, the computational domain is chosen as $\Omega = [-2,2]\times[-2,2]$, and the interface is simply a circle with radius one so that it can be represented by $\Gamma = \{\mathbf{X}(\theta) = (\cos(\theta),\sin(\theta)), 0 \leqslant \theta < 2\pi\}$ where θ is the arc-length parameter. For simplicity, we also choose the viscosity $\mu = 1$.

Example 1 (*Normal force on an interface* [5]). In the first example, we consider the case in which the boundary force exerts only along the normal direction, and it has the form as

$$\mathbf{F}(\theta) = 2\sin(3\theta)\mathbf{X}(\theta). \tag{3.25}$$

For convenience, the pressure and the velocity are written in polar coordinates as

$$p(r,\theta) = \begin{cases} -r^3\sin(3\theta), & r < 1, \\ r^{-3}\sin(3\theta), & r > 1, \end{cases}$$

$$u(r,\theta) = \begin{cases} \frac{3}{8}r^2\sin(2\theta) + \frac{1}{16}r^4\sin(4\theta) \\ \quad -\frac{1}{4}r^4\sin(2\theta), & r < 1, \\ \frac{1}{8}r^{-2}\sin(2\theta) - \frac{3}{16}r^{-4}\sin(4\theta) \\ \quad +\frac{1}{4}r^{-2}\sin(4\theta), & r \geqslant 1, \end{cases}$$

$$v(r,\theta) = \begin{cases} \frac{3}{8}r^2\cos(2\theta) - \frac{1}{16}r^4\cos(4\theta) \\ \quad -\frac{1}{4}r^4\cos(2\theta), & r < 1, \\ \frac{1}{8}r^{-2}\cos(2\theta) + \frac{3}{16}r^{-4}\cos(4\theta) \\ \quad -\frac{1}{4}r^{-2}\cos(4\theta), & r \geqslant 1. \end{cases}$$

Example 2 (*Tangential force on an interface* [5]). In this example, we consider the case in which the boundary force exerts only along the tangential direction, and it has the form as

$$\mathbf{F}(\theta) = 2\sin(3\theta)\mathbf{X}'(\theta). \tag{3.26}$$

The pressure and the velocity are given by

$$p(r,\theta) = \begin{cases} -r^3\cos(3\theta), & r < 1, \\ -r^{-3}\cos(3\theta), & r > 1, \end{cases}$$

$$u(r,\theta) = \begin{cases} \frac{1}{8}r^2\cos(2\theta) + \frac{1}{16}r^4\cos(4\theta) \\ \quad -\frac{1}{4}r^4\cos(2\theta), & r < 1, \\ -\frac{1}{8}r^{-2}\cos(2\theta) + \frac{5}{16}r^{-4}\cos(4\theta) \\ \quad -\frac{1}{4}r^{-2}\cos(4\theta), & r \geqslant 1, \end{cases}$$

$$v(r,\theta) = \begin{cases} -\frac{1}{8}r^2\sin(2\theta) + \frac{1}{16}r^4\sin(4\theta) \\ \quad +\frac{1}{4}r^4\sin(2\theta), & r < 1, \\ \frac{1}{8}r^{-2}\sin(2\theta) + \frac{5}{16}r^{-4}\sin(4\theta) \\ \quad -\frac{1}{4}r^{-2}\sin(4\theta), & r \geqslant 1. \end{cases}$$

Table 4
The maximum errors for the examples of Stokes problem on $N \times N$ grid points

N	$\|u-u_e\|_\infty$	Ratio	$\|v-v_e\|_\infty$	Ratio	$\|p-p_e\|_\infty$	Ratio
Example 1						
32	2.9955E−03	–	9.5555E−03	–	1.4625E−02	–
64	7.4576E−04	4.02	2.1775E−03	4.39	3.2027E−03	4.57
128	2.1442E−04	3.48	5.4344E−04	4.01	8.2001E−04	3.91
256	4.8445E−05	4.43	1.3800E−04	3.94	1.9358E−04	4.24
Example 2						
32	9.3164E−03	–	5.5489E−03	–	1.7579E−02	–
64	2.2334E−03	4.17	9.8214E−04	5.65	3.5421E−03	4.96
128	4.5329E−04	4.93	2.6948E−04	3.64	9.5814E−04	3.70
256	1.2100E−04	3.75	6.8943E−05	3.91	2.1994E−04	4.36
Example 3						
32	9.9654E−03	–	9.3837E−03	–	2.5682E−02	–
64	2.7483E−03	3.63	1.8844E−03	4.98	7.2394E−03	3.55
128	5.2897E−04	5.20	4.4803E−04	4.21	1.8827E−03	3.85
256	1.4410E−04	3.67	1.2263E−04	3.65	4.7359E−04	3.98

Table 5
The interpolation errors for Example 3 of Stokes problem on the interface markers $(X_k, Y_k) = (\cos\theta_k, \sin\theta_k), k = 0, 1, \ldots, N-1, \quad \theta_k = k\Delta\theta, \Delta\theta = 2\pi/N$

N	$\|u-u_e\|_{\infty,\Gamma}$	Ratio	$\|v-v_e\|_{\infty,\Gamma}$	Ratio
32	1.0035E−02	–	1.0923E−02	–
64	2.3020E−03	4.36	2.9853E−03	3.66
128	4.5430E−04	5.07	6.8889E−04	4.33
256	1.2788E−04	3.55	1.8553E−04	3.71

After some careful calculations, one can obtain that the velocity and the pressure in above two examples satisfy the Stokes equations (3.6)–(3.10) without the external forcing term ($\mathbf{g} = \mathbf{0}$). The pressure and velocity boundary conditions for those two examples are all Dirichlet.

Example 3 (*Normal and tangential forces on an interface*). In this example, we consider the boundary force that has both normal and tangential components as

$$\mathbf{F}(\theta) = 2\sin(3\theta)\mathbf{X}'(\theta) - \cos^3(\theta)\mathbf{X}(\theta). \tag{3.27}$$

The velocity components u and v are chosen exactly the same as in Example 2 while the pressure written in Cartesian coordinates ($x = r\cos\theta, y = r\sin\theta$) is

$$p(x,y) = \begin{cases} x^3 + \cos(\pi x)\cos(\pi y), & r < 1, \\ \cos(\pi x)\cos(\pi y), & r \geqslant 1. \end{cases}$$

The velocity and the pressure satisfy the Stokes equations (3.6)–(3.10) with a nonzero external force $\mathbf{g} \neq 0$ which can be calculated analytically. Here, we use the zero Neumann boundary condition (easily to be checked) for the pressure and the Dirichlet boundary conditions for the velocity.

Table 4 shows the maximum errors of the computed velocity u, v and the pressure p for those three examples. One can see from these numerical results that our scheme is indeed second-order accurate. In addition, using the computed values on the grid, we can interpolate the velocity from the grid to the interface. Table 5 shows the interpolation errors for the velocity field on the interface. One can also see that the interpolating accuracy is second-order on average.

4. Conclusion

In this paper, we introduce a simple version of the immersed interface methods (IIM) for Stokes flows with singular forces along an interface. The numerical method is based on applying the Taylor's expansions along the normal direction and incorporating the solution and its normal derivative jumps into the finite difference approximations. The fluid variables are solved in a staggered grid, and a new interpolation scheme for the non-smooth velocity has been developed. The numerical results show that the scheme is second-order accurate. In the future work, we shall develop a fourth-order accurate scheme to the present problems by taking more jump relations into account and extend the same techniques to the three-dimensional problems.

Acknowledgments

The authors were supported in part by MOE-ATU program and National Science Council of Taiwan under research grant NSC-94-2115-M-009-004.

References

[1] Adams J, Swarztrauber P, Sweet R. Fishpack—A package of Fortran subprograms for the solution of separable elliptic partial differential equations. Available from: http://www.netlib.org/fishpack.
[2] Beale JT, Lai M-C. A method for computing nearly singular integrals. SIAM J Numer Anal 2001;38:1902–25.
[3] Beale JT, Layton AT. On the accuracy of finite difference methods for elliptic problems with interfaces. Commun Appl Math Comput Sci 2006;1:91–119.
[4] Berthelsen PA. A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions. J Comput Phys 2004;197:364–86.

[5] Cortez R. The method of regularized stokeslets. SIAM J Sci Comput 2001;23:1204–25.
[6] Dumett M, Keener JP. A numerical method for solving anisotropic elliptic boundary value problems in 3D. SIAM J Sci Comput 2003;25:348–67.
[7] Griffith BE, Peskin CS. On the order of accuracy of the immersed boundary method: higher order convergence rates for sufficiently smooth problems. J Comput Phys 2005;208:75–105.
[8] Harlow FH, Welsh JE. Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. Phys Fluids 1965;8:2181–9.
[9] Lai M-C, Peskin CS. An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. J Comput Phys 2000;160:705–19.
[10] Li Z. An overview of the immersed interface method and its applications. Taiwan J Math 2003;7:1–49.
[11] Li Z, Ito K. The immersed interface method—numerical solutions of PDEs involving interfaces and irregular domains. SIAM Frontiers Appl Math 2006;33.
[12] Li Z, Lai M-C. The immersed interface method for the Navier–Stokes equations with singular forces. J Comput Phys 2001;171:822–42.
[13] LeVeque R, Li Z. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. SIAM J Numer Anal 1994;31:1019–44.
[14] LeVeque R, Li Z. Immersed interface method for Stokes flow with elastic boundaries or surface tension. SIAM J Sci Comput 1997;18:709–35.
[15] 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.
[16] Linnick MN, Fasel HF. A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains. J Comput Phys 2005;204:157–92.
[17] Liu X-D, Fedkiw RP, Kang M. A boundary condition capturing method for Poisson's equation on irregular domains. J Comput Phys 2000;160:151–78.
[18] Mayo A. The fast solution of Poisson's and the biharmonic equations on irregular regions. SIAM J Numer Anal 1984;21:285–99.
[19] Peskin CS. Numerical analysis of blood flow in the heart. J Comput Phys 1972;25:220–52.
[20] Peskin CS. The immersed boundary method. Acta Numer 2002:1–39.
[21] 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.
[22] Xu J, Zhao H-K. An Eulerian formulation for solving partial differential equations along a moving interface. J Sci Comput 2003;19:573–94.
[23] 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.
[24] Wiegmann A, Bube KP. The explicit-jump immersed interface method: Finite difference method for PDEs with piecewise smooth solutions. SIAM J Numer Anal 2000;37:827–62.