An immersed interface method for the incompressible Navier–Stokes equations in irregular domains

D.V. Le[a], B.C. Khoo[a,b], J. Peraire[a,c,*]

[a] Singapore-MIT Alliance
[b] Mechanical Engineering Department, National University of Singapore, Kent Ridge Road, Singapore
[c] Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

We present an immersed interface method for the incompressible Navier–Stokes equations capable of handling rigid immersed boundaries. The immersed boundary is represented by a set of Lagrangian control points. In order to guarantee that the no-slip condition on the boundary is satisfied, singular forces are applied on the fluid. These forces are related to the jumps in pressure and the jumps in the derivatives of both pressure and velocity, and are interpolated using cubic splines. The strength of the singular forces is determined by solving a small system of equations at each time step. The Navier–Stokes equations are discretized on a staggered Cartesian grid by a second-order accurate projection method for pressure and velocity.

Keywords: Immersed interface method; Navier–Stokes equations; Cartesian grid method; Finite difference; Fast Poisson solvers; irregular domains

1. Introduction

This paper considers the immersed interface method (IIM) for the incompressible Navier–Stokes equations in general domains involving rigid boundaries. In a 2-dimensional bounded domain Ω that contains a rigid interface Γ, we consider the incompressible Navier–Stokes equations, written as

$$\boldsymbol{u}_t + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p = \mu\Delta\boldsymbol{u} + \boldsymbol{F} \tag{1}$$

$$\nabla\cdot\boldsymbol{u} = 0 \tag{2}$$

with boundary and initial conditions

$$\boldsymbol{u}|_{\partial\Omega} = \boldsymbol{u}_b \tag{3}$$

$$\boldsymbol{u}(x,0) = \boldsymbol{u}_0 \tag{4}$$

where $\boldsymbol{u}$ is the fluid velocity, p the pressure, and μ the viscosity of the fluid. Here, we simply assume that the density, $\rho \equiv 1$, and the viscosity, μ, are constant. The singular force $\boldsymbol{F}$ has the form

$$\boldsymbol{F}(\boldsymbol{x},t) = \int_\Gamma \boldsymbol{f}(s,t)\delta(\boldsymbol{x}-\boldsymbol{X}(s,t))ds \tag{5}$$

where $\boldsymbol{X}(s,t)$ is the arc-length parameterization of Γ, s is the arc-length, $\boldsymbol{x} = (x, y)$ is the spatial position, and $\boldsymbol{f}(s,t)$ is the force density. The Navier–Stokes equations are discretized using finite differences on a staggered Cartesian grid. The main features of our method are:

- It is a Cartesian grid method; the method does not require complex mesh generation.
- It is second-order accurate for velocities.
- The Poisson-like equations resulting at each time step are solved using a Fast Fourier Transform algorithm $O(NlogN)$, where N is the number of degrees of freedom.

Methods utilizing a Cartesian grid for solving interface problems or problems with complex geometry have become popular in recent years. One of the most successful Cartesian grid methods is Peskin's immersed boundary (IB) method [1,2,3]. In order to deal with rigid boundaries, Lai et al. [2] proposed to evaluate the force density using an expression of the form

$$\boldsymbol{f}(s,t) = \kappa(\boldsymbol{X}^e(s) - \boldsymbol{X}(s,t)) \tag{6}$$

where κ is a constant, $\kappa \gg 1$, and $\boldsymbol{X}^e$ is the arc length

* Corresponding author. Tel.: +1 (617) 253 1981; Fax: +1 (617) 258 5143; E-mail: peraire@mit.edu

Computational Fluid and Solid Mechanics 2005
K.J. Bathe (Editor)

parametrization of the required boundary position. The forcing term in Eq. (6) is a particular case of the feedback forcing formulation proposed by Goldstein et al. [4] with $\beta = 0$. In [4], the force is expressed as

$$\boldsymbol{f}(s,t) = \alpha \int_0^t \boldsymbol{U}(s,t')dt' + \beta \boldsymbol{U}(s,t) \tag{7}$$

where $\boldsymbol{U}$ is the velocity of the boundary, and α and β are chosen to be negative and large enough so that $\boldsymbol{U}$ will stay close to zero. Lima E Silva et al. [3] proposed an alternative model to compute the force density $\boldsymbol{f}$ based upon the evaluation of the various terms in the momentum equation (1) at the control points. The force density $\boldsymbol{f}$ is calculated by computing all the Navier–Stokes terms at the control points.

Once the force density is obtained at the boundary, the immersed boundary method uses a discrete delta function to spread the force density to the nearby Cartesian grid points. Since the IB method uses the discrete delta function approach, it smears out sharp interface to a thickness of the order of the meshwidth and is only first-order accurate for problems with non-smooth but continuous solutions.

In contrast, the immersed interface method (IIM) can avoid this smearing 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 et al. [5] for solving elliptic equations, and later extended to Stokes flows by Le Veque et al. [6]. The method was developed further for the Navier–Stokes equations in Li et al. [7], Lee [8] and Le et al. [9] for problems with flexible boundaries. The method was also used by Calhoun [10] and Li et al. [11] for solving the two-dimensional streamfunction-vorticity equations in irregular domains. In [10,11] the no-slip boundary conditions are imposed directly by determining the correct jump conditions for streamfunction and vorticity.

Another Cartesian grid approach has been presented by Ye et al. [12] and Udaykumar et al. [13] using finite volume techniques. They reshaped the immersed boundary cells and use a polynomial interpolating function to approximate the fluxes and gradients on the faces of the boundary cells while preserving second-order accuracy.

In this paper, we extend our earlier work, presented in Le et al. [9] for problems with deformable boundaries, to solve problems with rigid immersed boundaries. Our approach uses the immersed interface method to solve the incompressible Navier–Stokes formulated in primitive variables. In [9], the singular force $\boldsymbol{f}$ is computed based on the configuration of the interface, i.e. the interface is assumed to be governed by either surface tension, or by an elastic membrane. In the present work, the singular force at the immersed boundary is determined to impose the no-slip condition at a rigid boundary. At each time step, the singular force is computed implicitly by solving a small, dense, linear system of equations. Having computed the singular force, we then compute the jump in pressure and jumps in the derivatives of both pressure and velocity. The jumps in the solution and its derivatives are incorporated into the finite difference discretization to obtain sharp interface resolution. Fast solvers from FISHPACK [14] are used to solve the resulting discrete systems of equations.

The remainder of the paper is organized as follows. In section 2, we present the relations that must be satisfied along the immersed boundary between the singular force $\boldsymbol{f}$ 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 solution jumps. In section 4, we present details of the numerical algorithm. In section 5, some numerical examples are presented and, finally, some conclusions and suggestions for future work are given in section 6.

2. Jump conditions across the interface

Let $\boldsymbol{n}$ and $\boldsymbol{\tau}$ be the unit outward normal and tangential vectors to the interface, respectively. The normal, $f_1 = \boldsymbol{f}(s,t) \cdot \boldsymbol{n}$, and tangential, $f_2 = \boldsymbol{f}(s,t) \cdot \boldsymbol{\tau}$, components of the force density, can be related to the jumps of pressure and velocity as follows (see [6,7,8] for details):

$$[\boldsymbol{u}] = \boldsymbol{0}, \qquad [\mu \boldsymbol{u}_\xi] = -f_2 \boldsymbol{\tau}, \qquad [\boldsymbol{u}_n] = \boldsymbol{0} \tag{8}$$

$$[p] = f_1, \qquad [p_\xi] = \frac{\partial f_2}{\partial s}, \qquad [p_\eta] = \frac{\partial f_1}{\partial s} \tag{9}$$

$$[\mu \boldsymbol{u}_{\eta\eta}] = \kappa f_2 \boldsymbol{\tau}, \qquad [\mu \boldsymbol{u}_{\xi\eta}] = -\frac{\partial f_2}{\partial \eta}\boldsymbol{\tau} - \kappa f_2 \boldsymbol{n} \tag{10}$$

$$[\mu \boldsymbol{u}_{\xi\xi}] = -[\mu \boldsymbol{u}_{\eta\eta}] + [p_\xi]\boldsymbol{n} + [p_\eta]\boldsymbol{\tau} + [\boldsymbol{u}_\xi]\boldsymbol{u} \cdot \boldsymbol{n}$$

The jump, $[\cdot]$, denotes the difference between the value of its argument outside and inside the interface, and (ξ, η) are the coordinates associated with the directions of $\boldsymbol{n}$ and $\boldsymbol{\tau}$, respectively. Here, κ is the signed valued of the curvature of the interface (i.e. we assume that $\boldsymbol{n} \times \boldsymbol{\tau} = \boldsymbol{k} \equiv$ constant, so that $\boldsymbol{n}$ can point either towards, or outwards from, the center of curvature). From expressions (8)–(10) the values of the jumps of the first and second derivatives of velocity and pressure with respect to the (x, y) coordinates are easily obtained by a simple coordinate transformation. For instance, we write

$$[\boldsymbol{u}_x] = [\boldsymbol{u}_\xi]n_1 + [\boldsymbol{u}_\eta]\tau_1$$

$$[\boldsymbol{u}_{yy}] = [\boldsymbol{u}_{\xi\xi}]n_2^2 + 2[\boldsymbol{u}_{\xi\eta}]n_2\tau_2 + [\boldsymbol{u}_{\xi\eta}]\tau_2^2$$

where $\boldsymbol{n} = (n_1, n_2)$ and $\boldsymbol{\tau} = (\tau_1, \tau_2)$, are the Cartesian components of the normal and tangential vectors to the interface at the point considered.

3. Generalized finite difference formulas

From Taylor series expansions, it is possible to show that if the interface cuts a grid line between two grid points at $x = \alpha$, $x_i \leq \alpha < x_{i+1}$, then the following approximations hold for a piecewise twice differentiable function $v(x)$:

$$v_x(x_i) = \frac{v_{i+1} - v_{i-1}}{2h} - \frac{1}{2h}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}[v^{(m)}] + O(h^2) \tag{11}$$

$$v_x(x_{i+1}) = \frac{v_{i+2} - v_i}{2h} - \frac{1}{2h}\sum_{m=0}^{2}\frac{(h^-)^m}{m!}[v^{(m)}] + O(h^2) \tag{12}$$

$$v_{xx}(x_i) = \frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} - \frac{1}{h^2}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}[v^{(m)}] + O(h) \tag{13}$$

$$v_{xx}(x_{i+1}) = \frac{v_{i+2} - 2v_{i+1} + v_i}{h^2} + \frac{1}{h^2}\sum_{m=0}^{2}\frac{(h^-)^m}{m!}[v^{(m)}] + O(h) \tag{14}$$

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

$$[v^{(m)}]_\alpha = \lim_{x\to\alpha^+} v^{(m)}(x) - \lim_{x\to\alpha^-} v^{(m)} \tag{15}$$

in short, $[\cdot] = [\cdot]_\alpha$, and $v^{(0)} = v$. See Wiegmann et al. [15] for more details.

4. Numerical algorithm

4.1. Projection method

We employ a pressure-increment projection algorithm for the discretization of the Navier–Stokes equations. This projection algorithm is analogous to that presented in Brown et al. [16]. It leads to second order accuracy for both velocity and pressure provided all the spatial derivatives are approximated to second-order accuracy. The spatial discretization is carried out on a standard MAC staggered grid analogous to that in Kim et al. [17]. The ENO second-order upwind scheme is used for the advective terms [18]. With the MAC mesh, the pressure field is defined at the cell center where the continuity equation is enforced. The velocity fields u and v are defined at the vertical edges and horizontal edges, respectively. Given the velocity $\boldsymbol{u}^n$, and the pressure $p^{n-1/2}$, we compute the velocity $\boldsymbol{u}^{n+1}$ and pressure $p^{n+1/2}$ in three steps:

Step 1: Compute an intermediate velocity field $\boldsymbol{u}^*$ by solving

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n+\frac{1}{2}} - \nabla p^{n-\frac{1}{2}} + \mu\nabla^2\boldsymbol{u}^{n+\frac{1}{2}} \tag{16}$$
$$\boldsymbol{u}^*|_{\partial\Omega} = \boldsymbol{u}_b^{n+1}$$

where the advective term is extrapolated using the formula

$$(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n+\frac{1}{2}} = \frac{3}{2}(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^n - \frac{1}{2}(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n-1} \tag{17}$$

the diffusion term is approximated implicitly as

$$\nabla^2\boldsymbol{u}^{n+1/2} = \frac{1}{2}(\nabla_h^2\boldsymbol{u}^* + \nabla_h^2\boldsymbol{u}^n) + \boldsymbol{C}_1 \tag{18}$$

and the pressure gradient term is given by

$$\nabla p^{n-\frac{1}{2}} = G^{MAC}p^{n-\frac{1}{2}} + \boldsymbol{C}_2 \tag{19}$$

The MAC gradient operators are defined as

$$(G_x^{MAC}p)_{i+\frac{1}{2},j} = \frac{p_{i+1,j} - p_{i,j}}{\Delta x}, \quad (G_y^{MAC}p)_{i,j+\frac{1}{2}} = \frac{p_{i,j+1} - p_{i,j}}{\Delta y}$$

Step 2: Compute a pressure update ϕ^{n+1} by solving the Poisson equation

$$\nabla^2\phi^{n+1} = \frac{\nabla\cdot\boldsymbol{u}^*}{\Delta t}, \qquad \boldsymbol{n}\cdot\nabla\phi^{n+1}|_{\partial\Omega} = 0 \tag{20}$$

This is accomplished by solving the discrete system

$$\nabla_h^2\phi^{n+1} = \frac{D^{MAC}\boldsymbol{u}^*}{\Delta t} + C_3 \tag{21}$$

where the MAC divergence operator is defined as follows

$$(D^{MAC}\boldsymbol{u})_{i,j} = \frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\Delta x} + \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\Delta y}$$

Step 3: Update the pressure and velocity field according to

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \Delta t G^{MAC}\phi^{n+1} + \boldsymbol{C}_4 \tag{22}$$

$$p^{n+1/2} = p^{n-1/2} + \phi^{n+1} - \frac{\mu}{2}(D^{MAC}\boldsymbol{u}^*) + C_5 \tag{23}$$

The operators ∇_h and ∇_h^2 are the standard three-point central difference operators and $\boldsymbol{C}_i$, $i = 1, \ldots, 5$, are the correction terms, which are only non-zero at the points near the interface and are calculated using generalized finite difference formulas of the type introduced in the previous section. This method requires solving two

Helmholtz equations for $\boldsymbol{u}^*$ in Eq. (16) and one Poisson equation for ϕ^{n+1} in Eq. (21). Since the correction terms only affect the right-hand sides of the discrete systems, we can take advantage of the fast solvers from Fishpack [14] to solve these equations.

4.2. Singular force evaluation

Having solved for $\boldsymbol{u}^{n+1}$ at the grid points, we now compute the velocity at the interface. In our method, we use a set of control points to represent the interface. The velocity at the control points, $\boldsymbol{U_k}$, is interpolated from the velocity at the grid points. Thus, we can write

$$\boldsymbol{U}_k = \boldsymbol{U}(\boldsymbol{X}^k) = \mathcal{B}(\boldsymbol{u}^{n+1}) \tag{24}$$

where $\mathcal{B}$ is the bilinear interpolation operator, which includes the appropriate correction terms that are required to guarantee second-order accuracy when the derivatives of the velocity are discontinuous.

In summary, the equations that need to be solved in order to calculate $\boldsymbol{u}^{n+1}$ and $\boldsymbol{U_k}$, can be written symbolically as

$$\begin{aligned}
\text{Eq(16)} \quad &\rightarrow \quad \boldsymbol{H}\boldsymbol{u}^* = \boldsymbol{C} + \boldsymbol{B}_1\boldsymbol{f} \\
\text{Eq(21)} \quad &\rightarrow \quad \boldsymbol{L}\phi^{n+1} = \boldsymbol{D}\boldsymbol{u}^* + \boldsymbol{B}_2\boldsymbol{f} \\
\text{Eq(22)} \quad &\rightarrow \quad \boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \boldsymbol{G}\phi^{n+1} + \boldsymbol{B}_3\boldsymbol{f} \\
\text{Eq(24)} \quad &\rightarrow \quad \boldsymbol{U_k} = \boldsymbol{M}\boldsymbol{u}^{n+1} + \boldsymbol{B}_4\boldsymbol{f}
\end{aligned}$$

Eliminating $\boldsymbol{u}^*$, ϕ^{n+1} and $\boldsymbol{u}^{n+1}$ from the above equations, we can compute the velocity $\boldsymbol{U_k}$ at the control points, as follows:

$$\begin{aligned}
\boldsymbol{U_k} = {} & \boldsymbol{M}(\boldsymbol{H}^{-1}\boldsymbol{C} - \boldsymbol{G}\boldsymbol{L}^{-1}\boldsymbol{D}\boldsymbol{H}^{-1}\boldsymbol{C}) \\
& + \left(\boldsymbol{M}(\boldsymbol{H}^{-1}\boldsymbol{B}_1 - \boldsymbol{G}\boldsymbol{H}^{-1}\boldsymbol{B}_1 - \boldsymbol{G}\boldsymbol{L}^{-1}\boldsymbol{B}_2 + \boldsymbol{B}_3) + \boldsymbol{B}_4\right)\boldsymbol{f}
\end{aligned} \tag{25}$$

For convenience, we can write Eq. (25) as

$$\boldsymbol{U_k} = \boldsymbol{U}_k^0 + \boldsymbol{A}\boldsymbol{f} \tag{26}$$

where $\boldsymbol{U}_k^0$ is simply the velocity at the control points obtained by solving Eqs. (16)–(24) with $\boldsymbol{f} = \boldsymbol{0}$, given $\boldsymbol{u}^n$ and $p^{n-1/2}$. $\boldsymbol{A}$ is a $2N_b \times 2N_b$ matrix, where N_b is the number of control points. The vector $\boldsymbol{A}\boldsymbol{f}$ is the velocity at the control points obtained by solving the following equations:

$$\frac{\boldsymbol{u}_f^*}{\Delta t} = \frac{\mu}{2}\nabla^2\boldsymbol{u}_f^*, \qquad \boldsymbol{u}_f^*|_{\partial\Omega} = 0 \tag{27}$$

$$\nabla^2\phi_f^{n+1} = \frac{\nabla\cdot\boldsymbol{u}_f^*}{\Delta t}, \qquad \boldsymbol{n}\cdot\nabla\phi_f^{n+1}|_{\partial\Omega} = 0 \tag{28}$$

$$\boldsymbol{u}_f^{n+1} = \boldsymbol{u}_f^* - \Delta t\nabla\phi_f^{n+1} \tag{29}$$

$$\boldsymbol{A}\boldsymbol{f} = \mathcal{B}(\boldsymbol{u}_f^{n+1}) \tag{30}$$

with $\boldsymbol{f}$ being the singular force at the immersed boundary.

Equation (26) can be used to determine the singular force if we know the prescribed velocity $\boldsymbol{U}_p$ at the immersed boundary. Thus, the singular force at the control points can be computed by solving

$$\boldsymbol{A}\boldsymbol{f} = \boldsymbol{U}_p - \boldsymbol{U}_k^0 \tag{31}$$

The matrix $\boldsymbol{A}$ is computed once and stored. We solve Eqs. (27)–(30) $2N_b$ times, i.e. once for each column. Each time, the force strength $\boldsymbol{f}$ is set to zero except for the entry in the column we want to calculate which is set to one. Once the matrix $\boldsymbol{A}$ has been calculated, only the right hand side, $\boldsymbol{U}_p - \boldsymbol{U}_k^0$, needs to be computed at each timestep. The resulting small system of equations (31) is then solved at each timestep for the singular force $\boldsymbol{f}$. Finally, we solve Eqs. (16)–(23) to obtain $\boldsymbol{u}^{n+1}$ and $p^{n+1/2}$.

5. Numerical results

In this section we present the numerical results for three problems which involve immersed boundaries.

5.1. Rotational flow

In this problem, the interface is a circle with radius r = 0.3 embedded in a square domain $[-1, 1] \times [-1, 1]$. We prescribe the interface to rotate with angular velocity $\omega = 2$. We set $\mu = 0.02$ and consider the solution when $t = 10$. The velocity field is shown in Fig. 1. We carried out a grid refinement analysis, using a reference grid of 512×512, to determine the order of convergence of the algorithm. The results in Table 1 show that the velocity is second-order accurate and the pressure is nearly second-order accurate.

5.2. Flow past a circular cylinder

In this example, we simulate an unsteady flow past a circular cylinder immersed in a rectangular domain $\Omega = [0, 3] \times [0, 1.5]$. The cylinder has a diameter $d = 0.1$ and its center is located at (1.6, 0.75). The fluid density is $\rho = 1.0$ and the free stream velocity is set to unity, $U_\infty = 1$. The viscosity is determined by the Reynolds number, Re. Simulations have been performed at $Re = 20$, 40, 80, 100, 200 and 300 on a 512×256 computational mesh. We use 40 points to represent the circular cylinder. At the inflow boundary we specify the velocity corresponding to the free-stream velocity, and a homogeneous Neumann boundary condition is applied at the top, bottom and exit boundaries. The pressure is

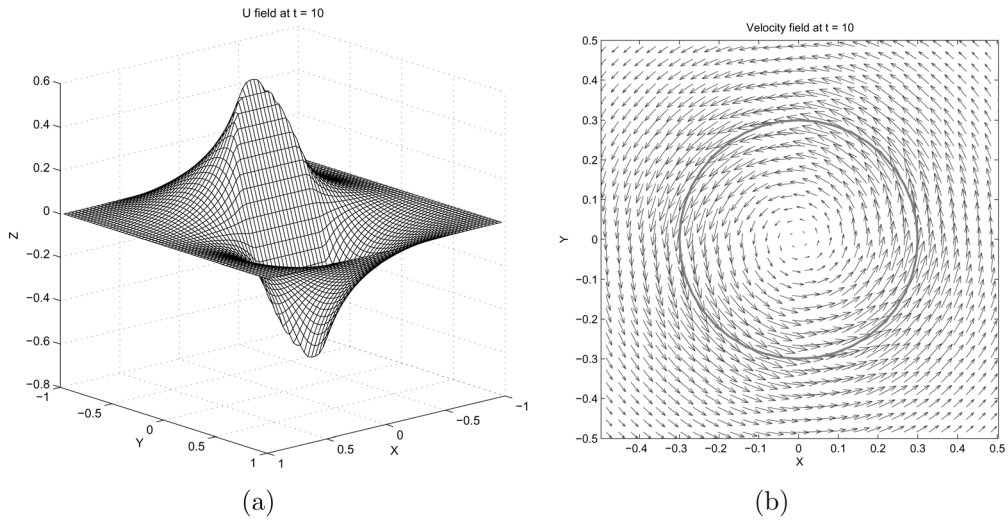

Fig. 1. Velocity field at time $t = 10$ with a 64×64 grid, $\mu = 0.02$, $\Delta t = \Delta x/4$. The immersed boundary rotates with angular velocity $\omega = 2$; (a) Plot of the x component of velocity field. (b) Plot of velocity vector field.

Table 1
The grid refinement analysis for the rotational flow problem with $\mu = 0.02$, $\Delta t = \Delta x/4$, at t = 10

N	N_b	$\|E(\boldsymbol{u})\|_\infty$	Order	$\|E(\boldsymbol{u})\|_2$	Order
64	40	1.8001×10^{-3}		1.6528×10^{-4}	
128	80	5.5145×10^{-4}	1.71	3.9239×10^{-5}	2.08
256	160	1.2755×10^{-4}	2.11	1.0021×10^{-5}	1.97
N	N_b	$\|E(p)\|_\infty$	Order	$\|E(p)\|_2$	Order
64	40	6.6995×10^{-3}		1.6014×10^{-3}	
128	80	1.5951×10^{-3}	2.07	4.7510×10^{-4}	1.75
256	160	5.7996×10^{-4}	1.46	1.5854×10^{-4}	1.58

set to zero at the exit boundary. The pressure field plots are shown in Fig. 2. These results appear to be in good agreement with the other numerical simulations and experimental results.

It is important to note that the matrix $\boldsymbol{A}$, for a closed immersed boundary, is singular. This happens because the pressure inside the closed boundary is not uniquely determined. We choose the pressure inside the cylinder such that there is no jump in pressure at one of the control points, i.e. the normal force at that point is set to zero. Therefore, we can eliminate the column and row of the matrix $\boldsymbol{A}$ corresponding to that control point, thus making the problem solvable.

5.3. Flow past a flat plate

In this example, we simulate an unsteady flow past a flat plate immersed in a rectangular domain $\Omega = [0, 3] \times [-0.75, 0.75]$. The flat plate whose length is $L = 0.1$ is oriented in the crossflow direction and located at $x = 1.40$. Simulations have been performed at $Re = 20$, 50, 100, 1000 and 5000 on a 256×128 computational mesh. We use eight points to represent the flat plate. The same boundary conditions as for the flow past a cylinder problem are applied in this problem. The pressure field plots are shown in Fig. 3.

6. Conclusion

We have presented a formally second-order accurate immersed interface method for the solution of the incompressible Navier–Stokes equations in irregular domains. The implementation has been tested with three examples involving a rotational flow, and unsteady flows over a circular cylinder and over a flat plate. Numerical

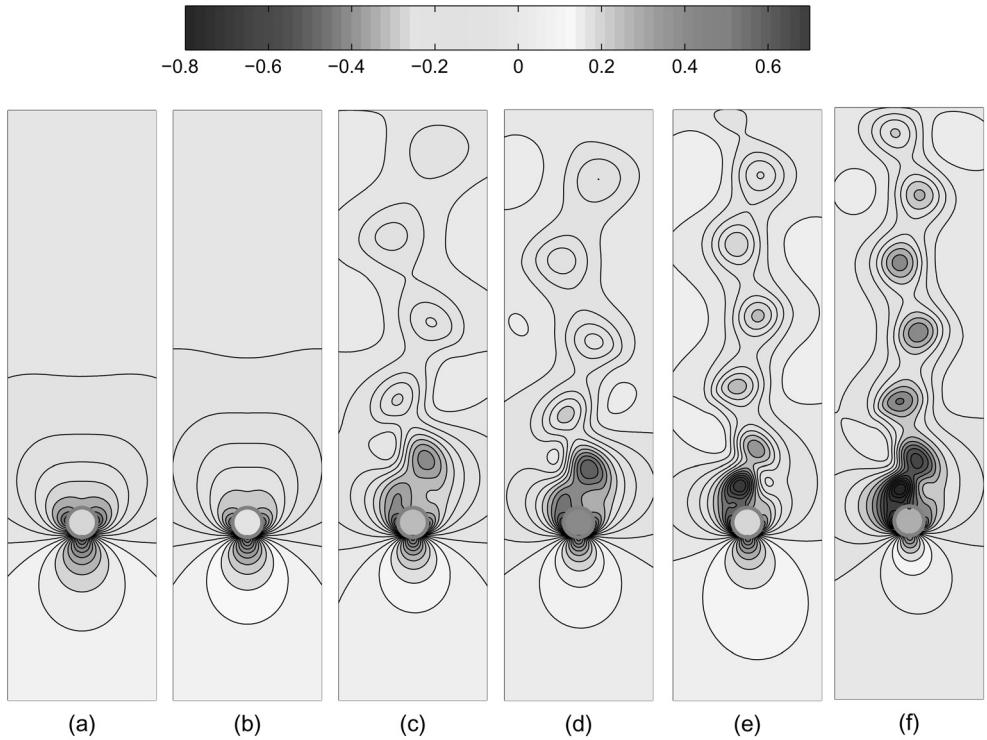

Fig. 2. Pressure field at $t = 10$, $\Delta t = \Delta x/4$. (a) $Re = 20$, (b) $Re = 40$, (c) $Re = 80$, (d) $Re = 100$, (e) $Re = 200$, (f) $Re = 300$.

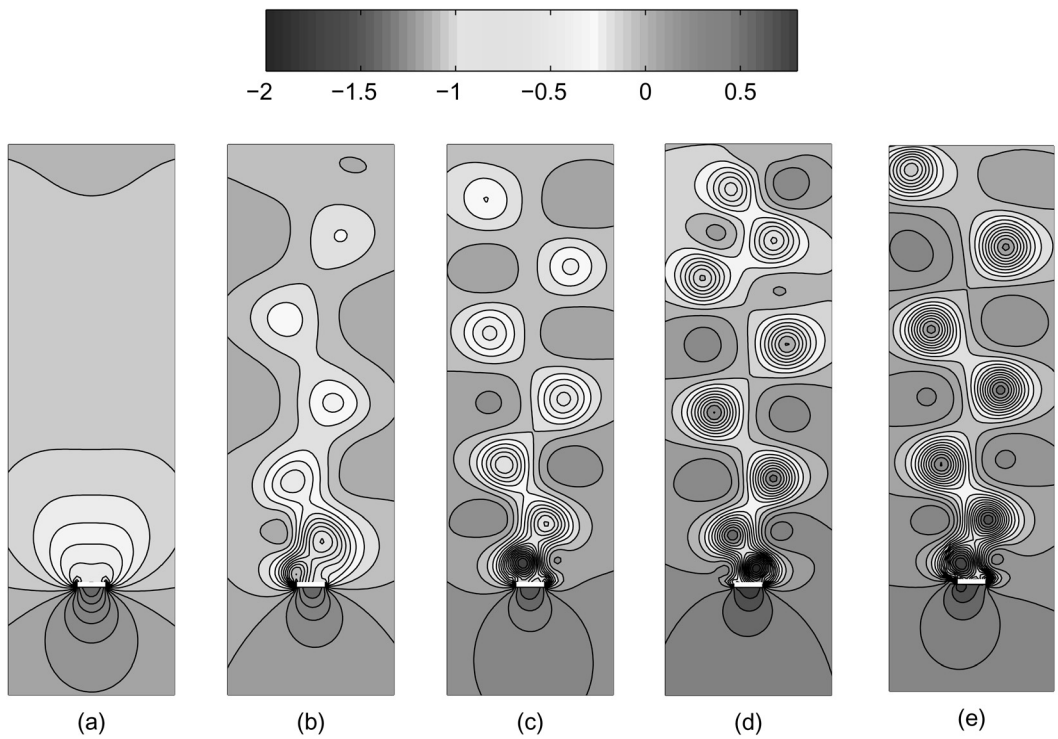

Fig. 3. Pressure field at $t = 10$, $\Delta t = \Delta x/4$. (a) $Re = 20$, (b) $Re = 50$, (c) $Re = 100$, (d) $Re = 1000$, (e) $Re = 5000$.

experiments have shown that our method can handle problems with rigid boundaries. We plan to combine the current algorithm with our earlier work [9] on problems involving deformable immersed interfaces. One of the issues that needs to be resolved is that of contact between moving deformable immersed interfaces.

References

[1] Peskin CS. The immersed boundary method. Acta Numerica 2002;161:479–517.
[2] Lai MC, Peskin CS. An immersed boundary method with formal second order accuracy and reduced numerical viscosity. J Comput Phys 2000;160:707–719.
[3] 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;189:351–370.
[4] Goldstein D, Handler R, Sirovich L. Modeling a no-slip flow with an external force field. J Comput Phys 1993; 105:354–366.
[5] LeVeque RJ, Li Z. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. SIAM J Numer Anal 1994;31:1019–1044.
[6] LeVeque RJ, Li Z. Immersed interface method for Stokes flow with elastic boundaries or surface tension. SIAM J Sci Comput 1997;18:709–735.
[7] Li Z, Lai MC. The immersed interface method for the Navier–Stokes equations with singular forces. J Comput Phys 2001;171:822–842.
[8] Lee L. An immersed interface method for the incompressible Navier–Stokes equations. Ph.D. thesis, University of Washington, DC, 2002.
[9] Le DV, Khoo BC, Peraire J. An immersed interface method for the incompressible Navier–Stokes equations. Presented at the SMA Symposium, Singapore, 2004.
[10] Calhoun D. A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions. J Comput Phys 2002;176:231–275.
[11] Li Z, Wang C. A fast finite difference method for solving Navier–Stokes equations on irregular domains. Comm Math Sci 2003;1(1):180–196.
[12] 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; 156:209–240.
[13] 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;174:345–380.
[14] Adams J, Swarztrauber P, Sweet R. FISHPACK: Efficient FORTRAN Subprograms for the Solution of Separable Eliptic Partial Differential Equations. National Center for Athmospheric Research, 1999, http://www.scd.ucar.edu/css/software/fishpack/
[15] Wiegmann A, Bube KP. The explicit-jump immersed interface method: finite difference methods for PDEs with piecewise smooth solutions. SIAM J Numer Anal 1997; 37(3):827–862.
[16] Brown DL, Cortez R, Minion ML. Accurate projection methods for the incompressible Navier–Stokes equations. J Comput Phys 2001;168:464–499.
[17] Kim J, Moin P. Application of a fractional step method to incompressible Navier–Stokes equations. J Comput Phys 1985;59:308–323.
[18] Shu CW, Osher S. Efficient implementation of essentially non-oscillatory shock capturing scheme, II. J Comput Phys 1989;83:32–78.