

An Immersed Interface Method for the Incompressible Navier-Stokes Equations in Irregular Domains

Duc-Vinh Le*, Boo Cheong Khoo*[†], Jaime Peraire*[‡]

*Singapore-MIT Alliance

[†]Department of Mechanical Engineering, National University of Singapore

[‡]Department of Aeronautics and Astronautics, Massachusetts Institute of Technology

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 at the immersed boundary. The 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 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.

I. 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

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

$$\nabla \cdot \mathbf{u} = 0 \quad (2)$$

with boundary and initial conditions

$$\mathbf{u}|_{\partial\Omega} = \mathbf{u}_b \quad (3)$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0 \quad (4)$$

where \mathbf{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 \mathbf{F} has the form

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

where $\mathbf{X}(s, t)$ is the arc-length parameterization of Γ , s is the arc-length, $\mathbf{x} = (x, y)$ is spatial position, and $\mathbf{f}(s, t)$ 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 cyclic reduction algorithm which has a complexity $O(N \log N)$, 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 ([10], [11], [15]). In order to deal with rigid boundaries, Lai and Peskin [11] propose to evaluate the force density using an expression of the form,

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

where κ is a constant, $\kappa \gg 1$, and \mathbf{X}^e is the arc-length parameterization 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. [12] with $\beta = 0$. In [12], the force is expressed as

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

where \mathbf{U} is the velocity at the control points, and α and β are chosen to be negative and large enough so that \mathbf{U} will stay close to zero. Lima E Silva et al. [15] proposed an alternative model to compute the force density \mathbf{f} based upon the evaluation of the various terms in the momentum equation (1) at the control points. The force density \mathbf{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 order of the meshwidth and it 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 a 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 [3] for solving elliptic equations, and later extended to Stokes flows by LeVeque et al. [4]. The method was developed further for the Navier-Stokes equations in Li et al. [6], Lee [9] and Le et al. [16] for problems with flexible boundaries. The method was also used by Calhoun [8] and Li et al. [7] for solving the two-dimensional streamfunction-vorticity equations in irregular domains. In [7], [8] 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. [13] and Udaykumar et al. [14] 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 works presented in Le et al. [16] 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 equations formulated in primitive variables. In [16], the singular force \mathbf{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 present work, the singular force at the immersed boundary is determined to impose the no-slip boundary condition at the 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 in the finite difference discretization to obtain sharp interface resolution. Fast solvers from Fishpack software library [18] are used to solve the resulting discrete systems of equations.

The remainder of the paper is organized as follows. In section II, we present the relations that must be satisfied along the immersed boundary between the singular force \mathbf{f} and the jumps in the velocity and pressure and their derivatives. In section III, we describe the generalized finite difference approximations to the solution derivatives, which incorporate solution jumps. In section IV, we present details of the numerical algorithm. In section V, some numerical examples are presented and finally, some conclusions and suggestions for future work are given in section VI.

II. JUMP CONDITIONS ACROSS THE INTERFACE

Let \mathbf{n} and $\boldsymbol{\tau}$ be the unit outward normal and tangential vectors to the interface, respectively. The normal, $f_1 = \mathbf{f}(s, t) \cdot \mathbf{n}$, and tangential, $f_2 = \mathbf{f}(s, t) \cdot \boldsymbol{\tau}$, components of the force density, can be related to the jump conditions for pressure and velocity as follows (see [4], [6], [9] for details),

$$[\mathbf{u}] = \mathbf{0}, \quad [\mu \mathbf{u}_\xi] = -f_2 \boldsymbol{\tau}, \quad [\mathbf{u}_\eta] = \mathbf{0} \quad (8)$$

$$[p] = f_1, \quad [p_\xi] = \frac{\partial f_2}{\partial s}, \quad [p_\eta] = \frac{\partial f_1}{\partial s}. \quad (9)$$

$$[\mu \mathbf{u}_{\eta\eta}] = \kappa f_2 \boldsymbol{\tau}, \quad [\mu \mathbf{u}_{\xi\eta}] = -\frac{\partial f_2}{\partial \eta} \boldsymbol{\tau} - \kappa f_2 \mathbf{n}, \quad (10)$$

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

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 \mathbf{n} and $\boldsymbol{\tau}$, respectively. Here, κ is the signed value of the curvature of the interface (i.e. we assume that $\mathbf{n} \times \boldsymbol{\tau} = \mathbf{k} \equiv \text{constant}$, so that \mathbf{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,

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

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

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

III. 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 $v(x)$:

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

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

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

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

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

$$[v^{(m)}]_\alpha = \lim_{x \rightarrow \alpha^+} v^{(m)}(x) - \lim_{x \rightarrow \alpha^-} v^{(m)}(x) \quad (11)$$

in short, $[\cdot] = [\cdot]_\alpha$, and $v^{(0)} = v$. See Wiegmann and Bube [5] for more details.

IV. NUMERICAL ALGORITHM

A. 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. [1]. It leads to a 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 and Moin [2]. The ENO second-order upwind scheme is used for the advective terms (Shu and Osher [17]). 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 \mathbf{u}^n , and the pressure $p^{n-1/2}$, we compute the velocity \mathbf{u}^{n+1} and pressure $p^{n+1/2}$ in three steps:

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

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -(\mathbf{u} \cdot \nabla \mathbf{u})^{n+\frac{1}{2}} - \nabla p^{n-\frac{1}{2}} + \mu \nabla^2 \mathbf{u}^{n+\frac{1}{2}} \quad (12)$$

$$\mathbf{u}^*|_{\partial\Omega} = \mathbf{u}_b^{n+1}$$

where, the advective term is extrapolated using the formula,

$$(\mathbf{u} \cdot \nabla \mathbf{u})^{n+\frac{1}{2}} = \frac{3}{2}(\mathbf{u} \cdot \nabla \mathbf{u})^n - \frac{1}{2}(\mathbf{u} \cdot \nabla \mathbf{u})^{n-1} \quad (13)$$

the diffusion term is approximated implicitly as,

$$\nabla^2 \mathbf{u}^{n+1/2} = \frac{1}{2}(\nabla_h^2 \mathbf{u}^* + \nabla_h^2 \mathbf{u}^n) + \mathbf{C}_1. \quad (14)$$

and the pressure gradient term is given by

$$\nabla p^{n-\frac{1}{2}} = G^{MAC} p^{n-\frac{1}{2}} + \mathbf{C}_2 \quad (15)$$

The MAC gradient operators are defined as

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

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

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

$$\nabla^2 \phi^{n+1} = \frac{\nabla \cdot \mathbf{u}^*}{\Delta t}, \quad \mathbf{n} \cdot \nabla \phi^{n+1}|_{\partial\Omega} = 0 \quad (16)$$

This is accomplished by solving the discrete system

$$\nabla_h^2 \phi^{n+1} = \frac{D^{MAC} \mathbf{u}^*}{\Delta t} + \mathbf{C}_3 \quad (17)$$

where the MAC divergence operator is defined as follows

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

Step 3: Update pressure and velocity field according to,

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t G^{MAC} \phi^{n+1} + \mathbf{C}_4 \quad (18)$$

$$p^{n+1/2} = p^{n-1/2} + \phi^{n+1} - \frac{\mu}{2} (D^{MAC} \mathbf{u}^*) + \mathbf{C}_5 \quad (19)$$

The operators ∇_h and ∇_h^2 are the standard three point central difference operators and $\mathbf{C}_i, i = 1, \dots, 5$, 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 \mathbf{u}^* in Eq (12) and one Poisson equation for ϕ^{n+1} in Eq (17). Since the correction terms only affect the right-hand sides of the discrete systems, we can take advantage of the fast solvers from Fishpack [18] to solve these equations.

B. Singular force evaluation

Having solved for \mathbf{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, \mathbf{U}_k , is interpolated from the velocity at the grid points. Thus, we can write

$$\mathbf{U}_k = \mathbf{U}(\mathbf{X}^k) = \mathcal{B}(\mathbf{u}^{n+1}) \quad (20)$$

where \mathcal{B} is the bilinear interpolation operator which includes the appropriate correction terms which 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 \mathbf{u}^{n+1} and \mathbf{U}_k , can be written symbolically as,

$$\text{Eq (12)} \quad \rightarrow \quad \mathbf{H}\mathbf{u}^* = \mathbf{C} + \mathbf{B}_1 \mathbf{f}$$

$$\text{Eq (17)} \quad \rightarrow \quad \mathbf{L}\phi^{n+1} = \mathbf{D}\mathbf{u}^* + \mathbf{B}_2 \mathbf{f}$$

$$\text{Eq (18)} \quad \rightarrow \quad \mathbf{u}^{n+1} = \mathbf{u}^* - \mathbf{G}\phi^{n+1} + \mathbf{B}_3 \mathbf{f}$$

$$\text{Eq (20)} \quad \rightarrow \quad \mathbf{U}_k = \mathbf{M}\mathbf{u}^{n+1} + \mathbf{B}_4 \mathbf{f}$$

Eliminating \mathbf{u}^* , ϕ^{n+1} and \mathbf{u}^{n+1} from the above equations, we can compute the velocity \mathbf{U}_k at the control points as follows,

$$\mathbf{U}_k = \mathbf{M}(\mathbf{H}^{-1}\mathbf{C} - \mathbf{G}\mathbf{L}^{-1}\mathbf{D}\mathbf{H}^{-1}\mathbf{C}) + \left(\mathbf{M}(\mathbf{H}^{-1}\mathbf{B}_1 - \mathbf{G}\mathbf{H}^{-1}\mathbf{B}_1 - \mathbf{G}\mathbf{L}^{-1}\mathbf{B}_2 + \mathbf{B}_3) + \mathbf{B}_4 \right) \mathbf{f}$$

For convenience, we can write the above equation as

$$\mathbf{U}_k = \mathbf{U}_k^0 + \mathbf{A}\mathbf{f} \quad (21)$$

where \mathbf{U}_k^0 is simply the velocity at the control points obtained by solving Eqs (12)–(20) with $\mathbf{f} = \mathbf{0}$, given \mathbf{u}^n and $p^{n-1/2}$. \mathbf{A} is a $2N_b \times 2N_b$ matrix, where N_b is the number of control points. The vector $\mathbf{A}\mathbf{f}$ is the velocity at the control points obtained by solving the following equations

$$\frac{\mathbf{u}_f^*}{\Delta t} = \frac{\mu}{2} \nabla^2 \mathbf{u}_f^*, \quad \mathbf{u}_f^*|_{\partial\Omega} = 0 \quad (22)$$

$$\nabla^2 \phi_f^{n+1} = \frac{\nabla \cdot \mathbf{u}_f^*}{\Delta t}, \quad \mathbf{n} \cdot \nabla \phi_f^{n+1}|_{\partial\Omega} = 0 \quad (23)$$

$$\mathbf{u}_f^{n+1} = \mathbf{u}_f^* - \Delta t \nabla \phi_f^{n+1} \quad (24)$$

$$\mathbf{A}\mathbf{f} = \mathcal{B}(\mathbf{u}_f^{n+1}) \quad (25)$$

with \mathbf{f} being the singular force at the immersed boundary. Eq (21) can be used to determine the singular force if we know the prescribed velocity \mathbf{U}_p at the immersed boundary. Thus, the singular force at the control points can be computed by solving

$$\mathbf{A}\mathbf{f} = \mathbf{U}_p - \mathbf{U}_k^0 \quad (26)$$

The matrix \mathbf{A} is computed once and stored. We solve Eqs (22)–(25) $2N_b$ times, i.e., one for each column. Each time, the force strength \mathbf{f} is set to zero except for the entry in the column we want to calculate which is set to one. Once the matrix \mathbf{A} has been calculated, only the right hand side $\mathbf{U}_p - \mathbf{U}_k^0$, needs to be computed at each timestep. The resulting small system of equations (26) is then solved at each timestep for the singular force \mathbf{f} . Finally, we solve Eqs (12)–(19) to obtain \mathbf{u}^{n+1} and $p^{n+1/2}$.

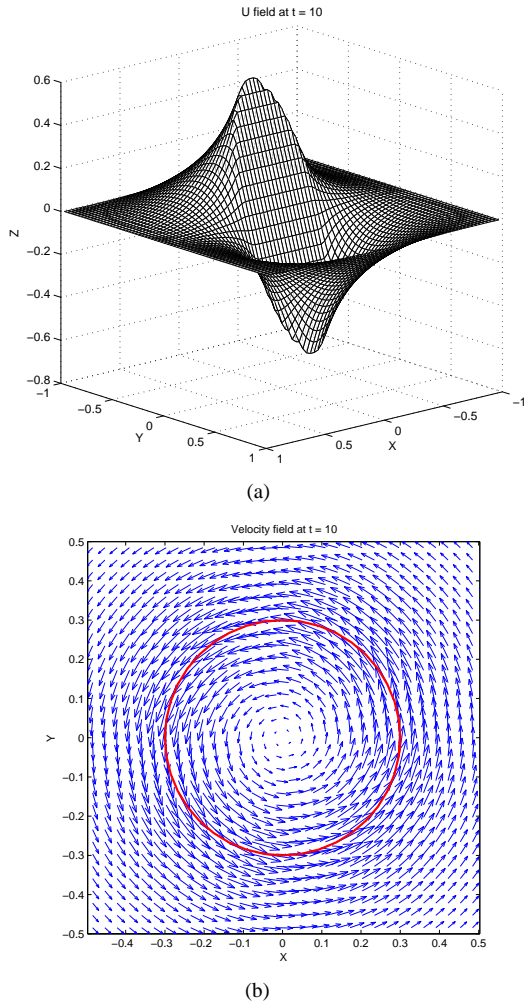


Fig. 1: Velocity field at time $t = 10$ with a 64×64 grid, $\mu = 0.02$, $\Delta t = \Delta x/4$. The immersed boundary rotates with angular velocity $\omega = 2$. Fig. 1(a) Plot of the x component of velocity field. Fig. 1(b) Plot of velocity vector field.

V. NUMERICAL RESULTS

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

A. Rotational flow

In this problem, the interface is a circle with radius $r = 0.3$ embedded in 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 Figure 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 I show that the velocity is second order accurate and pressure is nearly second order accurate.

B. Flow past a circular cylinder

In this example, we simulate an unsteady flow past a circular cylinder immersed in a rectangular domain $\Omega = [0, 3] \times [0, 1.5]$.

N	N_b	$\ E(\mathbf{u})\ _\infty$	order	$\ E(\mathbf{u})\ _2$	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

TABLE I: The grid refinement analysis for the rotational flow problem with $\mu = 0.02$, $\Delta t = \Delta x/4$, at $t = 10$.

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 freestream velocity is set to unity, $U_\infty = 1$. The viscosity is determined by the Reynolds number, Re_d . Simulations have been performed at $Re_d = 20, 40, 80, 100, 200$ and 300 on 512×256 computational mesh. We use 40 control points to represent the circular cylinder. At the inflow boundary we specify the velocity corresponding to the freestream velocity, and a homogeneous Neumann boundary condition is applied at the top, bottom and exit boundaries. The pressure is set to zero at the exit boundary. The pressure field plots are shown in Figure 2. The results appear to be in good agreement with the other numerical simulations and experimental results.

It is important to note that the matrix \mathbf{A} , for a closed immersed boundary, is singular. This happens because the pressure inside the close 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 one column and row of the matrix \mathbf{A} corresponding to that control point, thus making the problem solvable.

C. 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 $Re_L = 20, 50, 100, 1000$ and 5000 on a 256×128 computational mesh. We use 8 control 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 Figure 3.

VI. CONCLUSION

We have presented a formally second order accurate immersed interface method for the solution of the incompressible Navier-Stokes equations in irregular domain. The implementation has been tested with three examples involving rotational flow and unsteady flows over a circular cylinder and over a flat plate. Numerical experiments have shown that our method can handle problems with rigid boundaries. We plan to incorporate the current approach with our earlier works [16] for problems with deformable interfaces to deal with deformable boundaries in irregular domain problems and contact problems.

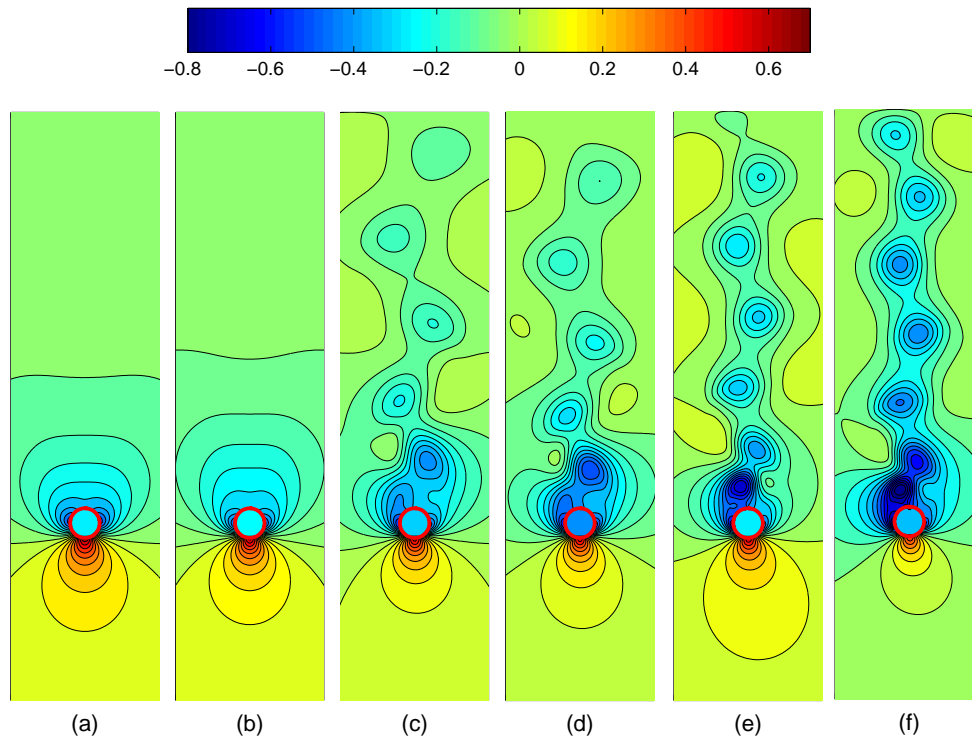


Fig. 2: Pressure field at $t = 10$, $\Delta t = \Delta x/4$. (a) $Re = 20$, (b) $Re = 40$, (c) $Re = 80$, (d) $Re = 100$, (e) $Re = 200$, (f) $Re = 300$.

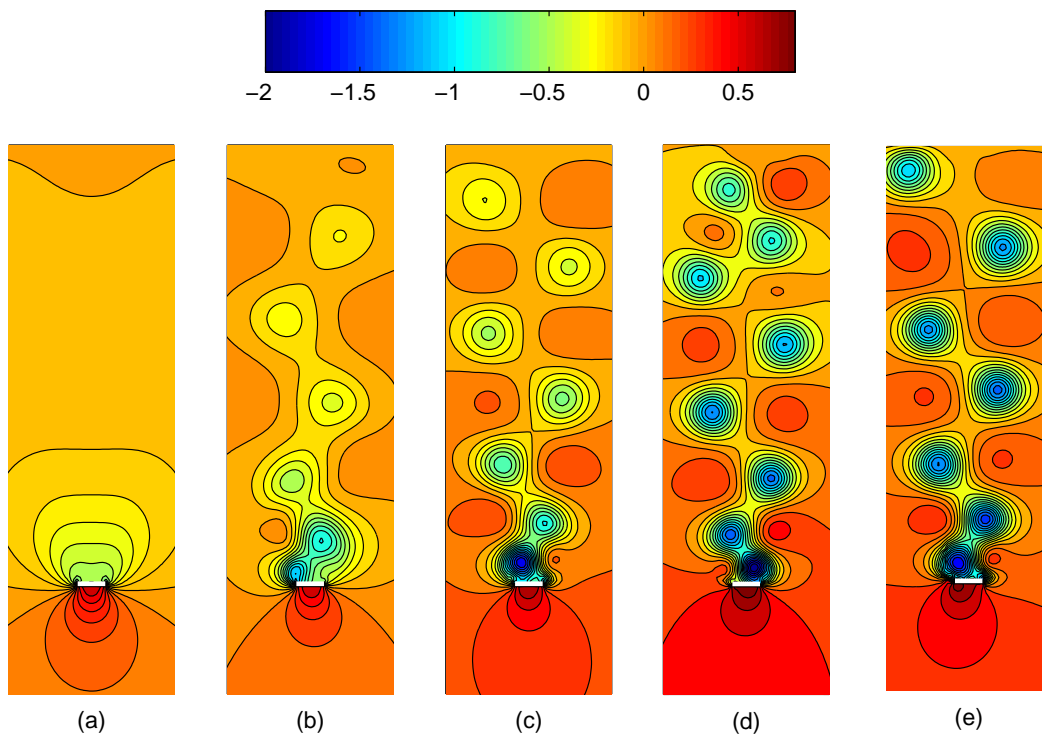


Fig. 3: Pressure field at $t = 10$, $\Delta t = \Delta x/4$. (a) $Re = 20$, (b) $Re = 50$, (c) $Re = 100$, (d) $Re = 1000$, (e) $Re = 5000$.

REFERENCES

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