An Immersed Interface Method for Viscous Incompressible Flows Involving Rigid and Flexible Boundaries

D.V. Le[1], B.C. Khoo[1,2], and J. Peraire[1,3]

[1]Singapore-MIT Alliance

[2]National University of Singapore, Mechanical Engineering Department

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

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. 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 the singular forces is determined by solving a small system of equations iteratively 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

In this paper, we present a novel numerical method for solving viscous, incompressible flow problems involving moving interfaces and rigid boundaries. One of the challenges of these problems is that the fluid motion, the flexible interface motion and the interaction with the immersed rigid boundaries must be computed simultaneously. This is necessary to account for the complex interaction between the fluid and the immersed boundaries. An example of interface problems that we consider is shown in Fig. 1. Our algorithm solves the incompressible Navier-Stokes equations formulated in primitive variables. In a 2-dimensional bounded domain Ω that contains a material interface Γ(t), we consider the incompressible Navier-Stokes equations, written as

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

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

where $\boldsymbol{u}$ is the fluid velocity, p the pressure, and μ the viscosity of the fluid. Here, we assume that fluid density $\rho \equiv 1$ and the viscosity μ is constant over the whole domain. The effect of the material interface $\Gamma(t)$ immersed in the fluid results in a singular force $\boldsymbol{F}$ which has the form

$$\boldsymbol{F}(\boldsymbol{x}, t) = \int_{\Gamma(t)} \boldsymbol{f}(s, t)\delta(\boldsymbol{x} - \boldsymbol{X}(s, t))ds \ , \tag{3}$$

where $\boldsymbol{X}(s, t)$ is the arc-length parametrization of $\Gamma(t)$, s is the arc-length, $\boldsymbol{x} = (x, y)$ is spatial position, and $\boldsymbol{f}(s, t)$ is the force strength. Here, $\delta(\boldsymbol{x})$ is the two-dimensional Dirac function. The force strength at the immersed rigid boundary is determined to impose the no-slip condition at the rigid boundary. The force strength at the flexible interface 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. The motion of the interfaces satisfies

$$\frac{\partial}{\partial t}\boldsymbol{X}(s, t) = \boldsymbol{u}\left(\boldsymbol{X}, t\right) = \int_{\Omega} \boldsymbol{u}(\boldsymbol{x}, t)\delta(\boldsymbol{x} - \boldsymbol{X}(s, t))d\boldsymbol{x} \ . \tag{4}$$

In our algorithm, the Navier-Stokes equations are discretized using a standard finite difference method on a staggered Cartesian grid.

Methods utilizing a Cartesian grid for solving interface problems or problems with

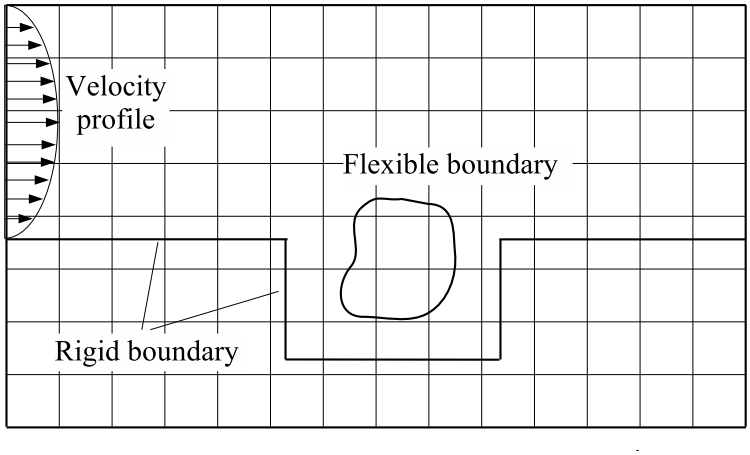


Figure 1: A typical domain in which the Navier-Stokes equations are solved. The flexible interface and the rigid boundary are immersed in a uniform Cartesian grid.

complex geometry have become popular in recent years. Existing Cartesian grid methods for interface problems can be categorized into two general groups: methods that determine the jump conditions across the interface and incorporate them into the finite difference scheme and methods that smooth out the singular force before it is applied to the fluid. Our method which is based on the immersed interface method originally proposed by LeVeque and Li [23, 24] falls into the first group. The immersed boundary method introduced by Peskin [29] belongs to the second group. Peskin's immersed boundary method has proven to be a very useful method for modelling fluid-structure interaction involving large geometry variations. This method was originally developed to study the fluid dynamics of blood flow in the human heart [28]. The original method has been developed further and has been applied to many biological problems including platelet aggregation [12, 13, 36], the deformation of red blood cell in a shear flow [10], the swimming of bacterial organisms and others [9, 11]. This method has also been applied to handle problems with rigid boundaries [17, 34]. In order to deal with rigid boundaries, Lai and Peskin [17] proposed to evaluate the force density using an expression of the form,

$$\boldsymbol{f}(s,t) = \kappa(\boldsymbol{X}^e(s) - \boldsymbol{X}(s,t)), \tag{5}$$

where κ is a constant, $\kappa \gg 1$, and $\boldsymbol{X}^e$ is the arc-length parametrization of the required boundary position. The forcing term in equation (5) is a particular case of the feedback forcing formulation proposed by Goldstein et al. [15] with $\beta = 0$. In [15],

the force is expressed as

$$\boldsymbol{f}(s,t) = \alpha \int_0^t \boldsymbol{U}(s,t')dt' + \beta \boldsymbol{U}(s,t) \tag{6}$$

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. [34] 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 $\boldsymbol{X}$ as,

$$\boldsymbol{f}(\boldsymbol{X}) = \frac{\partial \boldsymbol{u}(\boldsymbol{X})}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u}(\boldsymbol{X}) + \nabla p(\boldsymbol{X}) - \mu \nabla^2 \boldsymbol{u}(\boldsymbol{X}) \ . \tag{7}$$

The immersed boundary method uses a set of control points to represent the interface. The force densities are computed at these control points and are spread to the Cartesian grid points by a discrete representation of the delta function. Once the force densities are computed at the control points and spread to the grid, the Navier-Stokes equations with the forcing terms are then solved for pressure and velocity at Cartesian grid points. This velocity field is interpolated using the discrete delta function to find the velocity at the control points. More details on the immersed boundary method can be found in [29] and the references therein.

The immersed boundary method has several attractive features: the method is simple to implement, it can handle complex geometries easily and it can use standard regular Cartesian grid Navier-Stokes solvers. However, since the immersed boundary 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 general problems.

In contrast, the Immersed Interface Method (IIM) can avoid smearing sharp interfaces and maintains second-order accuracy by incorporating the known jumps into the finite difference scheme near the interface. The singular force $\boldsymbol{F}$ along the immersed boundaries results in solutions to the Navier-Stokes equations which may be non-smooth across the interface, i.e., there may be jumps in pressure and in the derivatives of both pressure and velocity at the interface. An essential ingredient of the immersed interface method is the relation between the jumps in the solutions and their derivatives, and the applied singular forces. The basic idea of our immersed interface method is to discretize the Navier-Stokes equations on a uniform Cartesian

grid and to account for the singular forces by explicitly incorporating the jumps in the solutions and their derivatives into the difference equations. The main advantage of the IIM is that the solutions of the Navier-Stokes equations on a uniform mesh can be done very efficiently with the use of fast solvers, and at the same time, complex geometrical changes can be handled in a rather seamless manner. The drawback of this method is that a special discretization for the Navier-Stokes equations near the immersed boundaries needs to be performed to maintain both accuracy and stability. The IIM was originally proposed by LeVeque and Li [23] for solving elliptic equations, and later extended to Stokes flow with elastic boundaries or surface tension [24]. The method was developed further for the Navier-Stokes equations in Li et al. [25], Lee [22] and Le et al. [19] for problems with flexible boundaries. The method was also used by Calhoun [6] and Li et al. [26] for solving the two-dimensional streamfunction-vorticity equations on irregular domains. In [6], to obtain the jumps in the streamfunction and vorticity, the method requires solving a small linear system of equations. For a stationary rigid boundary, the coefficient matrix is generated once and is factorized using LU decomposition. At each timestep, only the right-hand side vector is formed and the jumps are found via back substitution. For a moving rigid boundary, the linear system of equations must be generated at every timestep. This approach is impractical for moving rigid boundary because forming the linear system of equations for the jumps in the streamfunction and vorticity at each timestep is computationally prohibitive. To avoid generating the coefficient matrix explicitly, Li et al. [26] suggested to use the generalized minimal residual (GMRES) method [32] to solve the Schur complement system. However, this approach has not been applied for moving geometry. In [31], a Cartesian grid method for modelling multiple moving objects in incompressible viscous flow is considered. Instead of using a linear system to couple all the variables involved as in [6, 26], the authors compute the jumps in streamfunction and vorticity in separate steps. The jumps in streamfunction associated with the no-penetration condition are computed using a superposed homogenous solution. Boundary vorticity is calculated to impose the no-slip condition by interpolation. This approach greatly reduces the computational cost for problems with moving rigid boundaries.

In the present work, we focus on presenting an immersed interface method for solving

problems with immersed rigid boundaries. The method presented in this paper is based on that presented in Le et al. [20] and is an extension of our earlier work for problems with flexible boundaries [19]. Our approach uses the immersed interface method to solve the incompressible Navier-Stokes equations formulated in primitive variables. In [19], 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 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 into the finite difference discretization to obtain sharp interface resolution. Fast solvers from the FISHPACK software library [1] are used to solve the resulting discrete systems of equations. A contribution of this paper is the introduction of a novel numerical algorithm for the incompressible Navier-Stokes equations in the presence of rigid boundaries. Another contribution is that both flexible and rigid boundaries can be considered simultaneously. This contribution is significant because most of the current methods can only handle flexible boundaries and the rigid boundary is required to be aligned with the computational grid.

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 in detail 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

We have already mentioned that when singular forces are applied on a material interface, the solutions of the Navier-Stokes equations may be non-smooth or discontinuous

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

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

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

The jump, $[\,\cdot\,]$, denotes the difference between the value of its argument outside and inside the interface, and (ξ, η) are the rectangular coordinates associated with the directions of $\boldsymbol{n}$ and $\boldsymbol{\tau}$ respectively. Figure 2 illustrates a typical domain with an immersed flexible boundary and a local coordinate system. In order to construct the appropriate finite difference formulas we will also require the jumps in the second derivatives of velocity and pressure which can be obtained by differentiating the above expressions as,

$$\begin{aligned} [\mu \boldsymbol{u}_{\eta\eta}] &= \kappa f_2 \boldsymbol{\tau}, \qquad\qquad [\mu \boldsymbol{u}_{\xi\eta}] = -\frac{\partial f_2}{\partial \eta} \boldsymbol{\tau} - \kappa f_2 \boldsymbol{n}, \\ [\mu \boldsymbol{u}_{\xi\xi}] &= -[\mu \boldsymbol{u}_{\eta\eta}] + [p_\xi] \boldsymbol{n} + [p_\eta] \boldsymbol{\tau} + [\boldsymbol{u}_\xi] \boldsymbol{u} \cdot \boldsymbol{n} \end{aligned} \tag{10}$$

$$\begin{aligned} [p_{\eta\eta}] &= \frac{\partial^2 f_1}{\partial \eta^2} - \kappa [p_\xi], \quad [p_{\xi\eta}] = \frac{\partial^2 f_2}{\partial \eta^2} + \kappa [p_\eta], \\ [p_{\xi\xi}] &= -[\nabla \cdot (\boldsymbol{u} \cdot \nabla \boldsymbol{u})] - [p_{\eta\eta}] \ . \end{aligned} \tag{11}$$

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). The proof for expressions (8)-(11) can be found in detail in [23, 24, 25, 21]. We note that from expressions (8)-(11) 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 have,

$$\begin{aligned} [\boldsymbol{u}_x] &= [\boldsymbol{u}_\xi] n_1 + [\boldsymbol{u}_\eta] \tau_1 \\ [\boldsymbol{u}_y] &= [\boldsymbol{u}_\xi] n_2 + [\boldsymbol{u}_\eta] \tau_2 \\ [\boldsymbol{u}_{xx}] &= [\boldsymbol{u}_{\xi\xi}] n_1^2 + 2[\boldsymbol{u}_{\xi\eta}] n_1 \tau_1 + [\boldsymbol{u}_{\xi\eta}] \tau_1^2 \\ [\boldsymbol{u}_{yy}] &= [\boldsymbol{u}_{\xi\xi}] n_2^2 + 2[\boldsymbol{u}_{\xi\eta}] n_2 \tau_2 + [\boldsymbol{u}_{\xi\eta}] \tau_2^2 \ , \end{aligned}$$

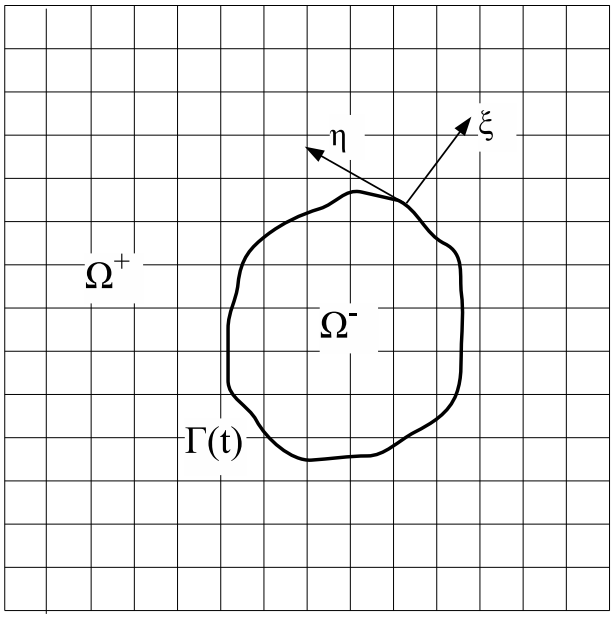


Figure 2: A typical domain with an immersed flexible boundary. The local coordinate system (ξ, η). The domain Ω^+ and Ω^- are divided by a closed curve $\Gamma(t)$.

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 $x = \alpha$, $x_i \leq \alpha < x_{i+1}, x_i \in \Omega^-, x_{i+1} \in \Omega^+$, then the following approximations hold for a piecewise twice differentiable function $v(x)$:

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

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

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

$$v_{xx}(x_{i+1}) = \frac{v_{i+2} - 2v_{i+1} + v_i}{h^2} + \frac{1}{h^2} \sum_{m=0}^{2} \frac{(h^-)^m}{m!} [v^{(m)}] + O(h) \tag{15}$$

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 x direction. The jumps in v and its derivatives are defined

as

$$[v^{(m)}]_\alpha = \lim_{x\to\alpha, x\in\Omega^+} v^{(m)}(x) - \lim_{x\to\alpha, x\in\Omega^-} v^{(m)}(x) \tag{16}$$

in short, $[\cdot] = [\cdot]_\alpha$, and $v^{(0)} = v$. See Weigmann and Bube [37] for more details on the proof. Note that if the interface cuts a grid line between two grid points $x_i \in \Omega^+$ and $x_{i+1} \in \Omega^-$, the above expressions need to be modified by changing the sign of the second terms from the right-hand sides of (12)–(15).

Expressions involving two or more interface crossings could also be derived, see for example [37]. Finally, we also require centered and backwards approximations for $v(t^{n+1/2})$. These approximations are required when the interface crosses a grid point over the time interval considered. Thus assuming that the interface crosses a grid point at time τ, $t^{n-1} \le \tau < t^{n+1}$, we have,

a) when $t^n \le \tau < t^{n+1/2}$,

$$v(t^{n+1/2}) = \frac{1}{2}(v^n + v^{n+1}) + \frac{1}{2}[v]_\tau + O(\triangle t) \tag{17}$$

b) when $t^{n+1/2} \le \tau < t^{n+1}$,

$$v(t^{n+1/2}) = \frac{1}{2}(v^n + v^{n+1}) - \frac{1}{2}[v]_\tau + O(\triangle t) \tag{18}$$

and,

a) when $t^{n-1} \le \tau < t^n$

$$v(t^{n+1/2}) = \frac{3}{2}v^n - \frac{1}{2}v^{n-1} - \frac{1}{2}[v]_\tau + O(\triangle t) \tag{19}$$

b) when $t^n \le \tau < t^{n+1/2}$

$$v(t^{n+1/2}) = \frac{3}{2}v^n - \frac{1}{2}v^{n-1} + [v]_\tau + O(\triangle t)\ . \tag{20}$$

Here, $[v]_\tau$ denotes the jump in time of a function $v(x,t)$ at a particular grid point and is only non zero when the interface crosses the grid point at time τ. The jump in time is defined as

$$[v(t)]_\tau = \lim_{t\to\tau^+} v(t) - \lim_{t\to\tau^-} v(t)\ . \tag{21}$$

It is easy to see that $[v]_\alpha = \pm[v]_\tau$, where $[.]_\alpha$ denotes spatial jump as defined in (16) and the sign depends on the motion of the interface. For example we use a plus sign when the grid point moves from the inside of the interface to the outside of the interface, i.e. from Ω^- to Ω^+, and a minus sign when the grid point moves from the outside of the interface to the inside of the interface, i.e. from Ω^+ to Ω^-.

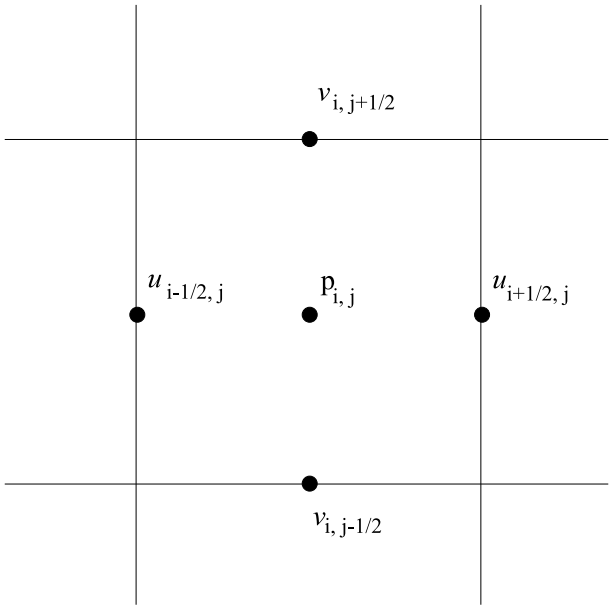


Figure 3: The MAC staggered grid in two dimensions.

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. [4]. 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 marker-and-cell (MAC) staggered grid analogous to that in Kim et al. [16]. The ENO third-order upwind scheme is used for the advective terms (Shu and Osher [33]). Figure 3 illustrates the MAC staggered grid. 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. The main advantage of the MAC mesh is that boundary conditions for pressure are not required explicitly. The main disadvantages of the MAC grid are that the implementation is more complicated than that on the non-staggered grids and some of the velocity components are not defined on the boundaries. The procedure of the pressure-increment projection method for problems with immersed interfaces is the same as that for non-interface

problems. For problems with immersed interface, however, the discretizations for the Navier-Stokes equations at all grid points near the interface need to be modified to account for the jump conditions across the interface of the solutions. Here we renew the pressure increment method for the case involving immersed interfaces. Given the velocity $\boldsymbol{u}^n$, and the pressure $p^{n-1/2}$, we compute the velocity $\boldsymbol{u}^{n+1}$ and pressure $p^{n+1/2}$ at the next time step in three steps:

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

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\triangle t} = -\overline{(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n+\frac{1}{2}}} - \overline{\nabla p^{n+\frac{1}{2}}} + \mu\overline{\nabla^2\boldsymbol{u}^{n+\frac{1}{2}}} + \boldsymbol{C}_1 \tag{22}$$

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

where the advective term is extrapolated using the formula,

$$\overline{(\boldsymbol{u}.\nabla\boldsymbol{u})^{n+\frac{1}{2}}} = \frac{3}{2}(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^n - \frac{1}{2}(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n-1} + \boldsymbol{C}_2 + \gamma_1[\boldsymbol{u}\cdot\nabla\boldsymbol{u}]_\tau \ , \tag{23}$$

the diffusion term is approximated implicitly as,

$$\overline{\nabla^2\boldsymbol{u}^{n+1/2}} = \frac{1}{2}(\nabla_h^2\boldsymbol{u}^* + \nabla_h^2\boldsymbol{u}^n) + \boldsymbol{C}_3 + \gamma_2[\nabla_h^2\boldsymbol{u}]_\tau \ , \tag{24}$$

and the pressure gradient is approximated simply as,

$$\overline{\nabla p^{n+\frac{1}{2}}} = G^{MAC}p^{n-\frac{1}{2}} + \boldsymbol{C}_4 + \gamma_3[\nabla p]_\tau \ . \tag{25}$$

The MAC gradient operators are defined as

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

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

$$\nabla_h^2\phi^{n+1} = \frac{D^{MAC}\boldsymbol{u}^*}{\triangle t} + C_5 \ , \tag{26}$$

with boundary condition

$$\boldsymbol{n}\cdot\nabla\phi^{n+1}|_{\partial\Omega} = 0 \ . \tag{27}$$

The MAC divergence operator is defined as

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

Step 3: Update pressure and velocity field

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \Delta t G^{MAC}\phi^{n+1} + \boldsymbol{C}_6 \tag{28}$$

$$p^{n+1/2} = p^{n-1/2} + \phi^{n+1} - \frac{\mu}{2}\left(D^{MAC}\boldsymbol{u}^*\right) + C_7 \tag{29}$$

Here, $[\cdot]_\tau$ denotes a jump in time and is only non zero when the interface crosses the grid point over the time interval considered. The coefficients $\gamma_i, i = 1, 2, 3$ correspond to the first order corrections in time. The coefficient γ_1 is determined from expressions (19), (20) and the coefficient γ_2 is determined from expressions (17), (18). The coefficient γ_3 is only nonzero when the interface crosses the grid point over the time interval $[t^{n-1/2}, t^{n+1/2}]$, and has the value of 1. Once again, at the interface $[\cdot]_\tau = \pm[\cdot]_\alpha$, where $[\cdot]_\alpha$ denotes spatial jump and the sign depends on the motion of the interface. The operator ∇_h^2 is the standard five point central difference operator and $\boldsymbol{C}_i, i = 2, \ldots, 7$, are the spatial correction terms which are only non-zero at the points near the interface. The constant $\boldsymbol{C}_1$ is the correction term for the discretization of $\partial\boldsymbol{u}/\partial t$ and is only nonzero at a particular grid point which the interface crosses over the time interval $[t^n, t^{n+1}]$.

In our projection method, we need to solve two Helmholtz equations for $\boldsymbol{u}^*$ in (22) and one Poisson equation for ϕ^{n+1} in (26). Since the correction terms in (22) and (26) only affect the right-hand side of the discrete systems for the Helmholtz and Poisson equations, we can take advantage of the fast solvers from FISHPACK [1] to solve these equations. In fact, the FISHPACK software library provides two subroutines HWSCRT() and HSTCRT() for solving the Helmholtz equations on the non-staggered and staggered Cartesian grids, respectively. The unknowns in HWSCRT() are defined at the corners of the cell and the unknowns in HSTCRT() are defined at the center of the cell as in Figure 3. The velocity field that we define in the MAC grid does not actually satisfy exactly the requirement of these subroutines but a simple combination of the two subroutines will be adequate for our purposes.

4.2 Correction terms

In this section, we will show how to evaluate the correction terms $\boldsymbol{C}_i, i = 1, \ldots, 7$ generated in section 4.1. Let's define $C\{u\}$ as a correction term for a quantity u. For

example, from (12) we can write

$$C\{u_x(x_i)\} = -\frac{1}{2h}\left([u] + h^+[u_x] + \frac{(h^+)^2}{2}[u_{xx}]\right) \tag{30}$$

Then the correction terms $\boldsymbol{C}_1$-C_7 are evaluated as follows:

$$\boldsymbol{C}_1 = -C\{\boldsymbol{u}_t\} \tag{31}$$

$$\boldsymbol{C}_2 = \frac{3}{2}C\{(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^n\} - \frac{1}{2}C\{(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n-1}\} \tag{32}$$

$$\boldsymbol{C}_3 = \frac{1}{2}\left(C\{\nabla^2\boldsymbol{u}^*\} + C\{\nabla^2\boldsymbol{u}^n\}\right) \tag{33}$$

$$\boldsymbol{C}_4 = C\{\nabla p^{n-\frac{1}{2}}\} \tag{34}$$

$$C_5 = \frac{C\{\nabla\cdot\boldsymbol{u}^*\}}{\triangle t} - C\{\nabla^2 p^{n+\frac{1}{2}}\} + C\{\nabla^2 p^{n-\frac{1}{2}}\} + \gamma_3[\nabla p]_\tau \tag{35}$$

$$\boldsymbol{C}_6 = -\triangle t\left(C\{\nabla p^{n+\frac{1}{2}}\} - C\{\nabla p^{n-\frac{1}{2}}\}\right) \tag{36}$$

$$C_7 = -\frac{\mu}{2}C\{\nabla\cdot\boldsymbol{u}^*\} \tag{37}$$

All the correction terms are included at least to first order accuracy. As explained in [23], the overall second order accuracy of the scheme is maintained provided only the singular points are treated with a first order scheme. This can be intuitively understood by noticing that when the mesh is refined, the area of the domain represented by these points is reduced.

We note that the correction term $C\{\boldsymbol{u}_t\}$ in (31) is only nonzero at the grid points crossed by the interface between time level n and time level $n+1$. Assume that the interface crosses a grid point (i,j) at time τ, $t^n \le \tau \le t^{n+1}$, the correction term for $\boldsymbol{u}_t$ at this point is given by

$$C\{\boldsymbol{u}_t\} = -\frac{1}{\triangle t}\left([\boldsymbol{u}]_\tau + (t^n - \tau)[\boldsymbol{u}_t]_\tau\right) \tag{38}$$

if $t^n \le \tau \le t^{n+1/2}$ and

$$C\{\boldsymbol{u}_t\} = -\frac{1}{\triangle t}\left([\boldsymbol{u}]_\tau + (t^{n+1} - \tau)[\boldsymbol{u}_t]_\tau\right) \tag{39}$$

if $t^{n+1/2} \le \tau \le t^{n+1}$.

Since the velocity is continuous across the interface, we have $[\boldsymbol{u}]_\tau = 0$. Also, by differentiating $[\boldsymbol{u}] = \boldsymbol{0}$ we obtain

$$[\boldsymbol{u}_t] = -[\boldsymbol{u}\cdot\nabla\boldsymbol{u}] = \pm[\boldsymbol{u}_t]_\tau \tag{40}$$

In (33), (35) and (37), we use the jump conditions for $\boldsymbol{u}^{n+1}$ to approximate the jump conditions for $\boldsymbol{u}^*$ as we expect that $\boldsymbol{u}^*$ is a good approximation for $\boldsymbol{u}^{n+1}$. This is one of the reasons why we choose to implement the pressure increment projection method in which $\boldsymbol{u}^*$ is computed to be a good approximation for $\boldsymbol{u}^{n+1}$. To evaluate the correction term $C\{\nabla^2\boldsymbol{u}^*\}$ of (33) at a point (i,j) in Fig. 4 we need to compute $[\boldsymbol{u}^*_x]$, $[\boldsymbol{u}^*_{xx}]$ at the intersection point α and $[\boldsymbol{u}^*_y]$, $[\boldsymbol{u}^*_{yy}]$ at β using the force strength at time level $n+1$. The correction term $C\{\nabla^2\boldsymbol{u}^*\}$ is calculated as follows

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

and $\nabla^2\boldsymbol{u}^*$ is approximated at the point (i,j) as

$$\nabla^2\boldsymbol{u}^*(i,j) = \frac{\boldsymbol{u}^*_{i+1,j} + \boldsymbol{u}^*_{i-1,j} + \boldsymbol{u}^*_{i,j+1} + \boldsymbol{u}^*_{i,j-1} - 4\boldsymbol{u}^*_{i,j}}{h^2} + C\{\nabla^2\boldsymbol{u}^*\}_{i,j} + O(h) .$$

Similarly, we can compute other correction terms in (33)–(37).

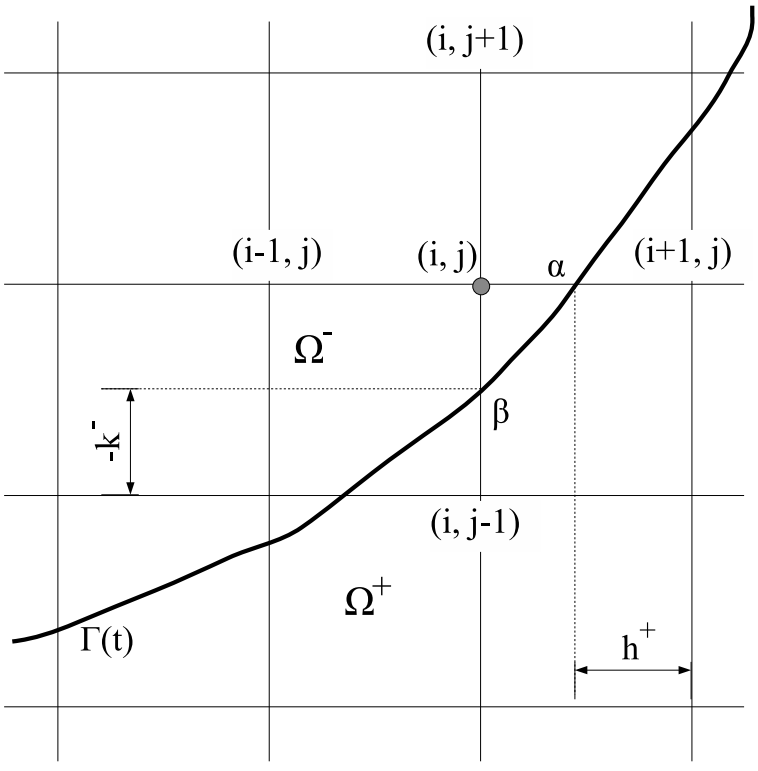


Figure 4: Interface and mesh geometry near the grid point (i,j).

4.3 Singular force evaluation

Assume that the singular force $\boldsymbol{f}$ is known at the rigid boundary. The velocity field $\boldsymbol{u}^{n+1}$ at all grid points can be computed via the projection method introduced in the previous chapter. Equation (22) is first solved for the intermediate velocity $\boldsymbol{u}^*$. The pressure increment ϕ^{n+1} is then determined by solving Eqn (26). Finally the velocity field is updated using Eqn (28). Having solved for $\boldsymbol{u}^{n+1}$ at the grid points, we now compute the velocity at the rigid boundary. In our method, we use a set of control points to represent the rigid boundary. The velocity at the control points, $\boldsymbol{U_k}$, is interpolated from the velocity at the grid points. Thus, we can write

$$\boldsymbol{U_k} = \boldsymbol{U}(\boldsymbol{X}_k) = \mathcal{B}(\boldsymbol{u}^{n+1}) \ , \tag{41}$$

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. The explicit form of $\boldsymbol{U_k}$ can be found in Appendix A. 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,

$$\begin{aligned}
\text{Eqn (22)} \quad &\rightarrow \quad \boldsymbol{H}\boldsymbol{u}^* = \boldsymbol{C} + \boldsymbol{B}_1\boldsymbol{f} \\
\text{Eqn (26)} \quad &\rightarrow \quad \boldsymbol{L}\boldsymbol{\phi}^{n+1} = \boldsymbol{D}\boldsymbol{u}^* + \boldsymbol{B}_2\boldsymbol{f} \\
\text{Eqn (28)} \quad &\rightarrow \quad \boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \boldsymbol{G}\boldsymbol{\phi}^{n+1} + \boldsymbol{B}_3\boldsymbol{f} \\
\text{Eqn (41)} \quad &\rightarrow \quad \boldsymbol{U_k} = \boldsymbol{M}\boldsymbol{u}^{n+1} + \boldsymbol{B}_4\boldsymbol{f}
\end{aligned}$$

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

$$\begin{aligned}
\boldsymbol{U_k} = \ & \boldsymbol{M}\big(\boldsymbol{H}^{-1}\boldsymbol{C} - \boldsymbol{G}\boldsymbol{L}^{-1}\boldsymbol{D}\boldsymbol{H}^{-1}\boldsymbol{C}\big) \\
& + \Big(\boldsymbol{M}\big(\boldsymbol{H}^{-1}\boldsymbol{B}_1 - \boldsymbol{G}\boldsymbol{L}^{-1}\boldsymbol{D}\boldsymbol{H}^{-1}\boldsymbol{B}_1 - \boldsymbol{G}\boldsymbol{L}^{-1}\boldsymbol{B}_2 + \boldsymbol{B}_3\big) + \boldsymbol{B}_4\Big)\boldsymbol{f} \ .
\end{aligned} \tag{42}$$

For convenience, we can write (42) as

$$\boldsymbol{U_k} = \boldsymbol{U}_k^0 + \boldsymbol{A}\boldsymbol{f} \ , \tag{43}$$

where $\boldsymbol{U}_k^0$ is simply the velocity at the control points obtained by solving Eqns (22), (26), (28) and (41) 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 $\boldsymbol{Af}$ is the velocity at the control points obtained by solving the following equations:

$$\frac{\boldsymbol{u}_f^*}{\triangle t} = \frac{\mu}{2}\nabla^2 \boldsymbol{u}_f^*, \qquad \boldsymbol{u}_f^*|_{\partial\Omega} = 0 \tag{44}$$

$$\nabla^2 \phi_f^{n+1} = \frac{\nabla \cdot \boldsymbol{u}_f^*}{\triangle t}, \qquad \boldsymbol{n} \cdot \nabla \phi_f^{n+1}|_{\partial\Omega} = 0 \tag{45}$$

$$\boldsymbol{u}_f^{n+1} = \boldsymbol{u}_f^* - \Delta t \nabla \phi_f^{n+1} \tag{46}$$

$$\boldsymbol{Af} = \mathcal{B}(\boldsymbol{u}_f^{n+1}) \tag{47}$$

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

Equation (43) 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

$$\boldsymbol{Af} = \boldsymbol{U}_p - \boldsymbol{U}_k^0 \tag{48}$$

In this way, the singular force is solved to impose exactly the no-slip boundary condition at the interface. The coefficient matrix $\boldsymbol{A}$ can be computed explicitly at each timestep from (44)–(47). We solve Eqns (44)–(47) $2N_b$ times, i.e., one for each column. Each time, the force strength $\boldsymbol{f}$ is set to zero except for the entry corresponding to the column we want to calculate which is set to one. Note that the matrix $\boldsymbol{A}$ depends on the location of the interface and the timestep Δt.

For static geometry, we will have the same matrix $\boldsymbol{A}$ at every timestep if we use the same Δt at every timestep. Therefore the matrix $\boldsymbol{A}$ is computed once and is factorized and stored. 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 (48) is then solved at each timestep for the singular force $\boldsymbol{f}$ via back substitution. Finally, we solve Eqs (22)–(29) to obtain $\boldsymbol{u}^{n+1}$ and $p^{n+1/2}$. Note that the same Δt as that used in computing $\boldsymbol{A}$ is used at every timestep of the simulation. 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 can choose the pressure inside the interface 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 $\boldsymbol{A}$ corresponding to

that control point, thus making the problem solvable. Or we can use singular value decomposition (SVD) method to solve the singular system of equations (48). The solution obtained via SVD method is the least-square solution which has the shortest length. We prefer the SVD method since the right-hand side of (48) may not lie in the range of $\boldsymbol{A}$ and hence the exact solution cannot be obtained.

For moving geometry, the coefficient matrix must be regenerated for every timestep. The computational cost associated with this is prohibitive. To avoid generating $\boldsymbol{A}$, we employ GMRES method to solve (48) iteratively. Each iteration of GMRES method requires a matrix-vector product which can be found by solving (44)–(47). In each matrix-vector product, we have to solve two Helhmoltz equations (44) and a Poisson equation (45). Therefore, our algorithm for solving the problems with moving boundary is only effective if the GMRES method takes a few iterations to converge. For a closed immersed boundary, the linear system of equations (48) is singular. A version of GMRES method for singular linear system of equations is required. We employed the GMRES method presented in [5] which used the incremental condition estimation (ICE) [2] to monitor the conditioning of the upper Hessenberg matrix.

4.4 Implementation

In this section we describe a basic implementation of our algorithm for the Navier-Stokes equations with immersed rigid boundary. One of the typical problems with immersed rigid boundary is the flow over a stationary circular cylinder. To start our computation we use a set of control points to represent the rigid boundary and compute the coefficient matrix as mentioned in the previous section. For the cylinder problem, this matrix is singular. We factorize the coefficient matrix using singular value decomposition as,

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \ , \tag{49}$$

where $\boldsymbol{U} = [u_1, \ldots, u_N]$ and $\boldsymbol{V} = [v_1, \ldots, v_N]$ are orthogonal matrices and $\boldsymbol{\Sigma} = diag(\sigma_1, \ldots, \sigma_N)$ is a diagonal matrix whose elements are the singular values of the original matrix such that

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_N \geq 0 \ .$$

Since $\boldsymbol{A}$ is singular it has at least one singular value equals to zero. We store $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{\Sigma}$ for solving the singular force at every timestep. Having $\boldsymbol{A}$, at each timestep, given the velocity field, $\boldsymbol{u}^n$ and pressure field $p^{n-1/2}$, our algorithm of finding $\boldsymbol{u}^{n+1}$, $p^{n+1/2}$ and the singular force to impose the no-slip condition at the rigid boundary can be summarized as follows:

Step 1: Compute the right-hand side of (48) by computing $\boldsymbol{U}_k^0$.

- Set $\boldsymbol{f} = 0$. Solve (22), (26) and (28) for the velocity at all grid points.
- Interpolate the velocity at the control points $\boldsymbol{U}_k^0$ as in (41).
- Compute the right-hand side vector $\boldsymbol{b} = \boldsymbol{U}_p - \boldsymbol{U}_k^0$.

Step 2: Compute the singular force by solving (48) using SVD method.

- If $\boldsymbol{A}$ is nonsingular, then the force $\boldsymbol{f}$ can be written in terms of the SVD as

$$\boldsymbol{f} = \sum_{i=1}^{N} \frac{u_i^T \boldsymbol{b}}{\sigma_i} v_i \ .$$

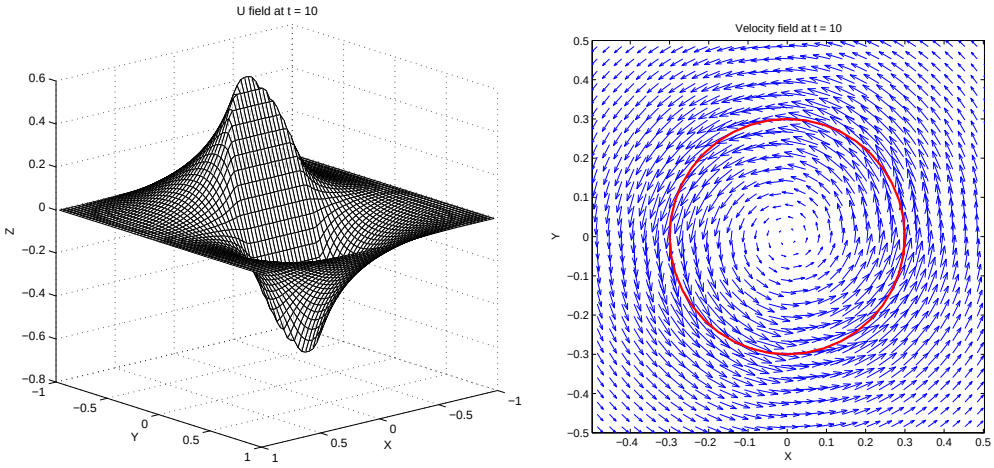
- If $\boldsymbol{A}$ is singular and has only one zero singular value, then the force $\boldsymbol{f}$ can be computed as,

$$\boldsymbol{f} = \sum_{i=1}^{N-1} \frac{u_i^T \boldsymbol{b}}{\sigma_i} v_i \ . \tag{50}$$

Step 3: Compute $\boldsymbol{u}^{n+1}$ and $p^{n+1/2}$ using the projection method.
For moving geometry, we still have the same algorithm except that the GMRES solver is applied to solve (48) iteratively at each time step. Thus we do not need to form the coefficient matrix explicitly.

5 Numerical results

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



(a) (b)

Figure 5: Velocity field at time t = 10 with a 64×64 grid, $\mu = 0.02$, $\triangle t = \triangle x/4$. The immersed boundary rotates with angular velocity $\omega = 2$. 5(a) Plot of the x component of velocity field. 5(b) Plot of velocity field.

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 Figure 5. 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 $\Omega = [0, 3] \times [0, 1.5]$. We use this problem as a benchmark test for our algorithm. 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, $Re = \frac{U_\infty d}{\mu}$. 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

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

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

we specify the velocity corresponding to the freestream velocity, and a homogeneous Neumann boundary condition is applied at the top, bottom and exit boundaries. For all these simulations we first use the free stream velocity as the initial velocity and the initial pressure is set to zero over the computational domain. Then the force at the cylinder interface is determined such that there is no flow inside the cylinder and the pressure is a particular constant inside the cylinder. After the first timestep, the flow evolves naturally and satisfies the no-slip boundary condition.

Once the velocity field and pressure field have been computed, the drag and lift coefficients and the Strouhal number can be computed from the force at the control points.

The drag coefficient is defined as

$$C_D = \frac{D}{\frac{1}{2}\rho U_\infty^2 d} \ . \tag{51}$$

The drag can be computed from the force along the cylinder interface as,

$$D = -\int_\Gamma f_x \mathrm{d}s \ , \tag{52}$$

where f_x is the x component of the singular force.

The lift coefficient is defined as

$$C_L = \frac{L}{\frac{1}{2}\rho U_\infty^2 d} \ . \tag{53}$$

The lift can be computed from the force along the cylinder interface as,

$$L = -\int_\Gamma f_y \mathrm{d}s \ , \tag{54}$$

where f_y is the y component of the singular force.

The Strouhal number is defined as,

$$St = \frac{fd}{U_\infty} \ , \tag{55}$$

where f is the vortex shedding frequency, is one of the key quantities that characterizes the vortex shedding process. This coefficient can be obtained using the Fast Fourier Transform of the periodic variation of the lift coefficient [34]. Finally, the dimensionless time is defined as

$$T = \frac{U_\infty t}{d} \ . \tag{56}$$

Fig. 6 shows the streamlines for $Re = 20$ and $Re = 40$. For these low Reynolds numbers, the wake formed behind the cylinder gradually attains a steady symmetric state. Once the flow has reached the steady state the drag coefficients, the length of the recirculation zone and the angle of separation are computed and are compared with established results in Table 2. The results obtained by our method are compared to the numerical simulations [6, 8, 14, 31] as well as experimental results [7, 35]. It is found that our results are in good agreement with other numerical simulations and experimental results. For $Re = 20$ our drag coefficient is very closed to other numerical results but it is about 8% lower than the experimental measurement of Tritton [35]. For $Re = 40$ our drag coefficient is about 5% higher than the experimentally determined value [35]. Fig. 7 shows a plot of the pressure fields for $Re = 20$ and $Re = 40$. The pressure patterns are symmetric about the streamwise axis.

Between $Re = 40$ and $Re = 50$ we expect to see a transition to instability. Fig. 8 shows that our algorithm are able to detect the onset of an instability by Reynolds number 50. It has been reported that the wake behind the cylinder first becomes unstable at a critical Reynolds number of about $Re = 46 \pm 1$ [39]. Above this Reynolds number the cylinder wake instability rises and grows in time and leads to Karman vortex shedding. These behaviors are shown in the numerical simulations for $Re =$ 80, 100, 200 and 300. Note that in all these simulations we do not need to artificially perturb the flow field to initiate the unsteady behavior. Fig. 9 shows the pressure fields at $Re = 100$, 200 and 300. The instabilities and vortex shedding can be visualized from this figure. In Table 3 and Table 4, the drag and lift coefficients at Re = 100 and Re = 200 are compared to other numerical simulations. For Re =

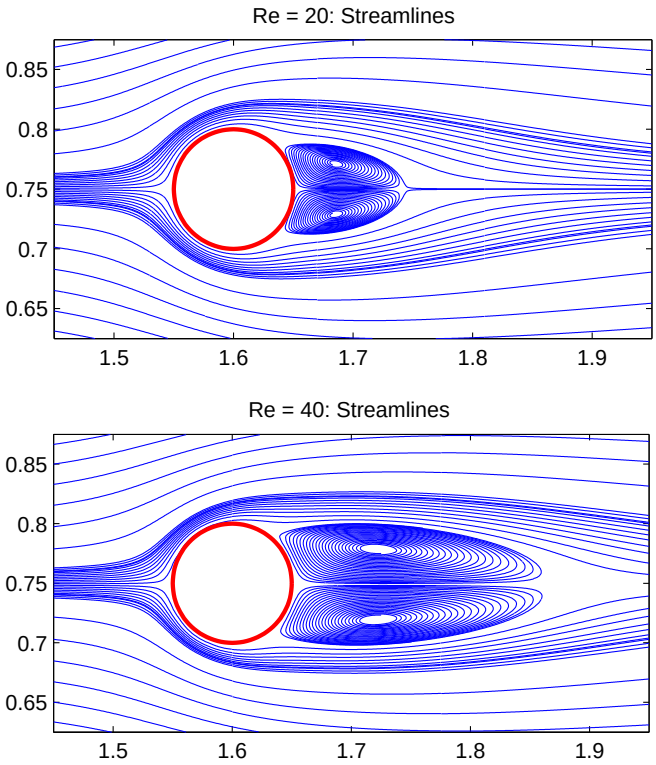


Figure 6: Streamlines for Re = 20 and Re = 40.

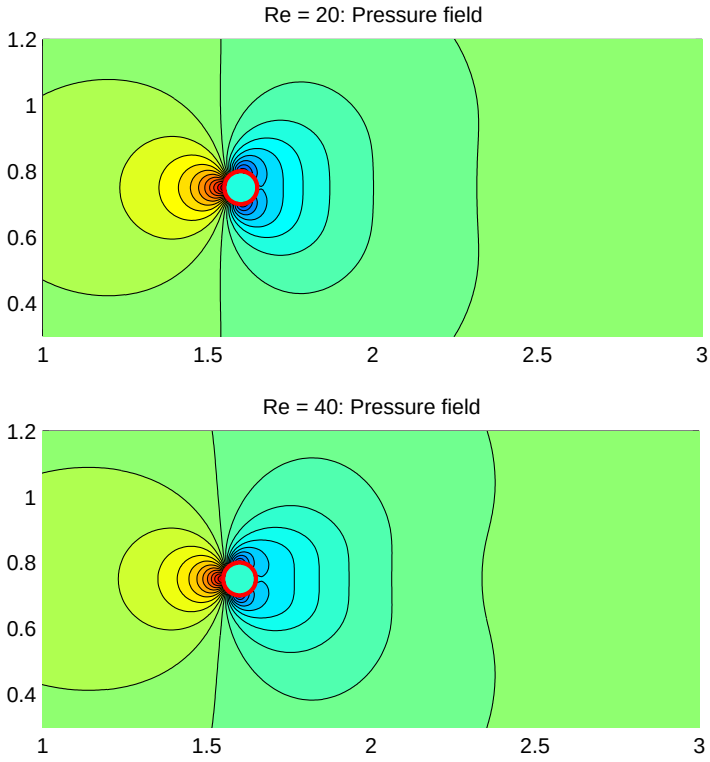


Figure 7: Pressure fields for Re = 20 and Re = 40.

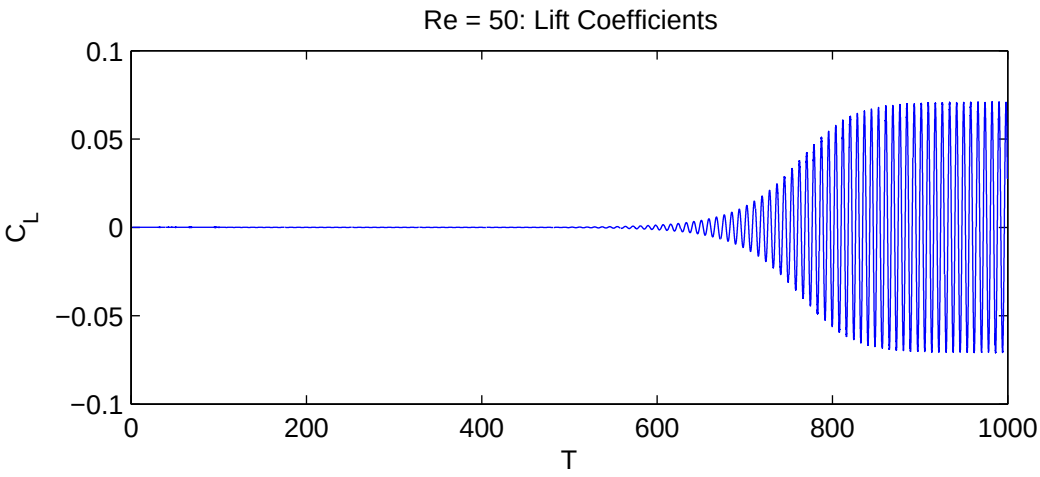


Figure 8: Lift Coefficients at Re = 50.

	$Re = 20$			$Re = 40$		
	L/d	θ	C_D	L/d	θ	C_D
Tritton [35]	–	–	2.22	–	–	1.48
Coutanceau and Bouard [7]	0.73	42.3^o	–	1.89	52.8^o	–
Fornberg [14]	0.91	–	2.00	2.24	–	1.50
Dennis and Chang [8]	0.94	43.7^o	2.05	2.35	53.8^o	1.52
Calhoun [6]	0.91	45.5^o	2.19	2.18	54.2^o	1.62
Russell and Wang [31]	0.94	43.3^o	2.13	2.29	53.1^o	1.60
Ye et al. [39]	0.92	–	2.03	2.27	–	1.52
Present	0.93	43.9^o	2.05	2.22	53.6^o	1.56

Table 2: Length of the recirculation zone, Angle of Separation and Drag Coefficient for Re = 20 and Re = 40

C_D	$Re = 100$	$Re = 200$
Braza et al. [3]	1.36 ± 0.015	1.40 ± 0.050
Liu at al. [27]	1.35 ± 0.012	1.31 ± 0.049
Calhoun [6]	1.33 ± 0.014	1.17 ± 0.058
Russell et al. [31]	1.38 ± 0.007	1.29 ± 0.022
Present	1.37 ± 0.009	1.34 ± 0.030

Table 3: Drag Coefficients for Re = 100 and Re = 200

100, the mean drag obtained by our algorithm is slightly greater than that computed by other researchers [3, 6, 27]. Our drag coefficient differs from that reported by them by $1\% - 3\%$. For Re = 200, our drag coefficient lies within the range of results reported in [3, 6, 27, 31]. Our value is about 15% higher than that in Calhoun [6] and 4% lower than the value obtained by Braza et al. [3]. In Table 4 it can be seen that the lift coefficient calculated by our method for $Re = 100$ is well within the range of the values obtained by other researchers. However our lift coefficient for $Re = 200$ is lower than their values. Fig. 10 and Fig. 11 show the variations in time of the drag coefficients and the lift coefficients, respectively. They also show how the vortex shedding develops to a periodic state in time at Re = 100 and Re = 200. The vortex shedding Strouhal number is computed for Re = 80, 100, 200 and 300 and is compared with established results in Table 5. Our computed Strouhal number obtained at Re = 80 comes out to be 0.15 which compares very well with the values obtained from experiment [38] and from numerical simulation [39]. At Re = 100 and Re = 200, our Strouhal numbers are in good agreement with those given in [6, 27, 31] and differ from the experimental results [38] by 1.8% and 1%, respectively. At Re = 300, our computed Strouhal number compares very well with the value obtained from experiment [38].

5.3 Flow past several cylinders

In this example, we consider an unsteady flow past several cylinders immersed in a rectangular domain $\Omega = [0, 3] \times [0, 1.5]$. This example shows the capability of handling multiple rigid boundaries of our algorithm. Simulation has been performed

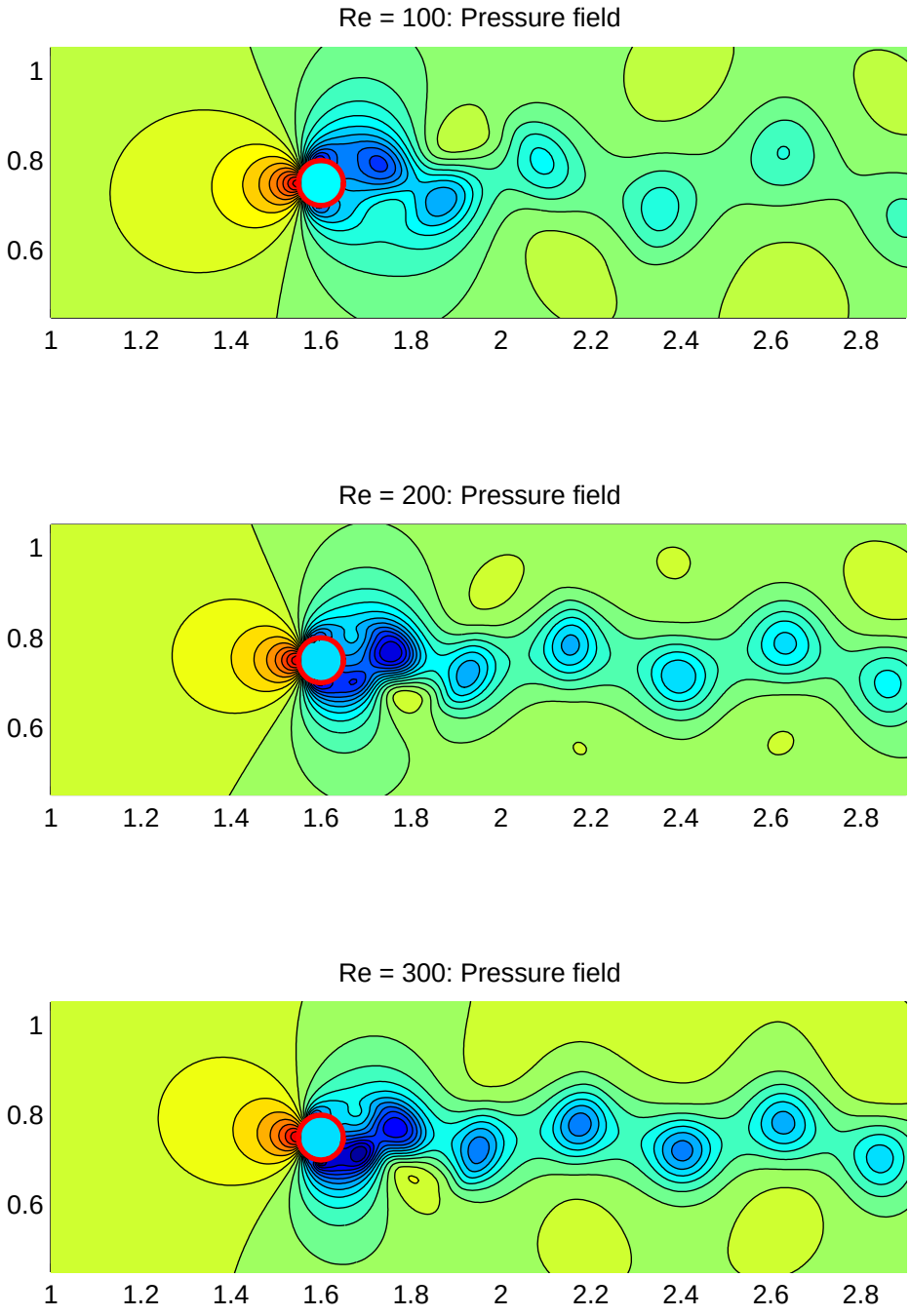


Figure 9: Pressure fields for Re = 100, Re = 200 and Re = 300.

C_L	$Re = 100$	$Re = 200$
Braza et al. [3]	± 0.250	± 0.75
Liu at al. [27]	± 0.339	± 0.69
Calhoun [6]	± 0.298	± 0.67
Russell et al. [31]	± 0.300	± 0.50
Present	± 0.323	± 0.43

Table 4: Lift Coefficients for Re = 100 and Re = 200

St	$Re = 80$	$Re = 100$	$Re = 200$	$Re = 300$
Ye et al. [39]	0.15	–	–	0.210
Williamson [38]	0.15	0.163	0.185	0.203
Liu at al. [27]	–	0.164	0.192	–
Calhoun [6]	–	0.175	0.202	–
Russell et al. [31]	–	0.169	0.195	–
Present	0.15	0.160	0.187	0.200

Table 5: Strouhal numbers for Re = 80, 100, 200 and 300

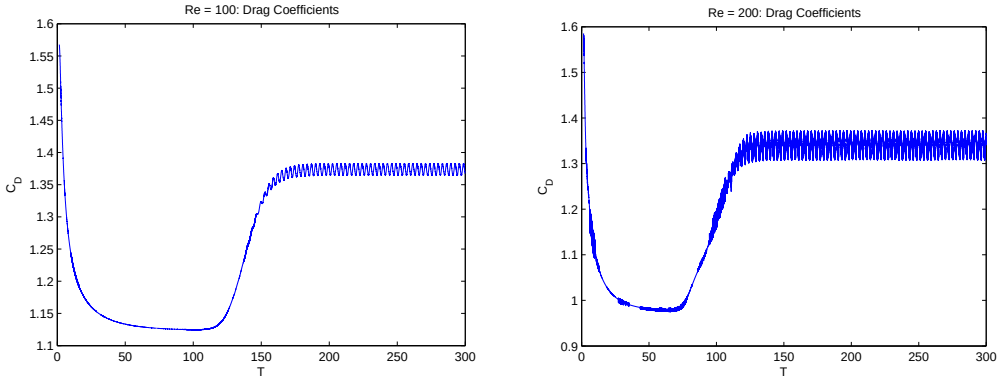


Figure 10: Drag Coefficients for Re = 100 and Re = 200.

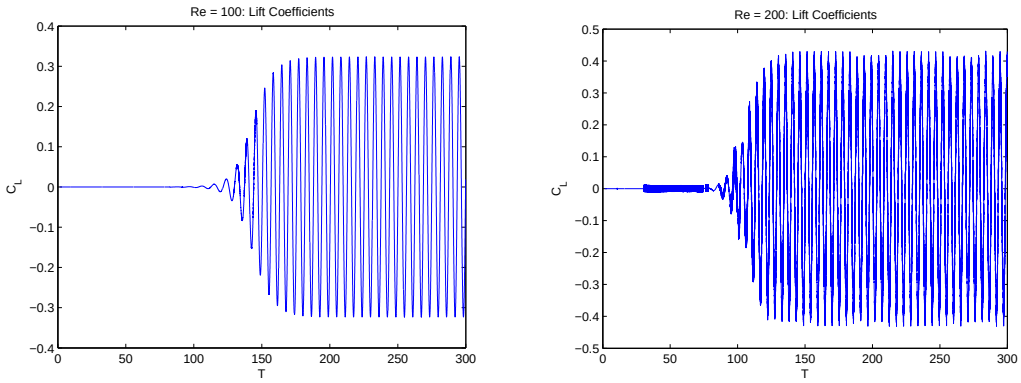


Figure 11: Lift Coefficients for Re = 100 and Re = 200.

for three cylinders immersed in the flow at Re = 100. All the cylinders have the same diameter of 0.1. We use 20 control points to represent each of the circular cylinders. The computational grid is 512×256 and the same boundary conditions as those for the flow past a single cylinder problem are applied. Fig. 12 and Fig. 13 show the streamlines and pressure contours for Re = 100 at different time levels. The vortex shedding is not vertically symmetric since the cylinders are not placed symmetrically.

5.4 Flow past a moving circular cylinder

In this example, we show the capability of modelling moving rigid boundaries for our method. We simulate the flow past a moving cylinder which moves to the left at a velocity $U_\infty = -1$. The computational domain is $[0, 6] \times [-1.5, 1.5]$. The cylinder has a radius $r = 0.1$ and its center is initially located at (5.5, 0.0). At the left boundary we set the velocity to zero, and a homogeneous Neumann boundary condition is applied at the top, bottom and right boundaries. In the frame of reference that is attached to the moving cylinder, these boundary conditions are the same as those used for the stationary circular cylinder problem. Simulation has been performed for Re = 40.
In this example, to solve for the singular force at the moving boundary we do not have to generate a system of equations explicitly. But instead we solve for the force at the boundary iteratively via GMRES algorithm. Since the system of equations is singular, the convergence rate of the GMRES algorithm is very slow. However, we can use the incremental condition estimation (ICE) [2] to monitor the conditioning of the upper Hessenberg matrix and stop the iteration when the conditioner number

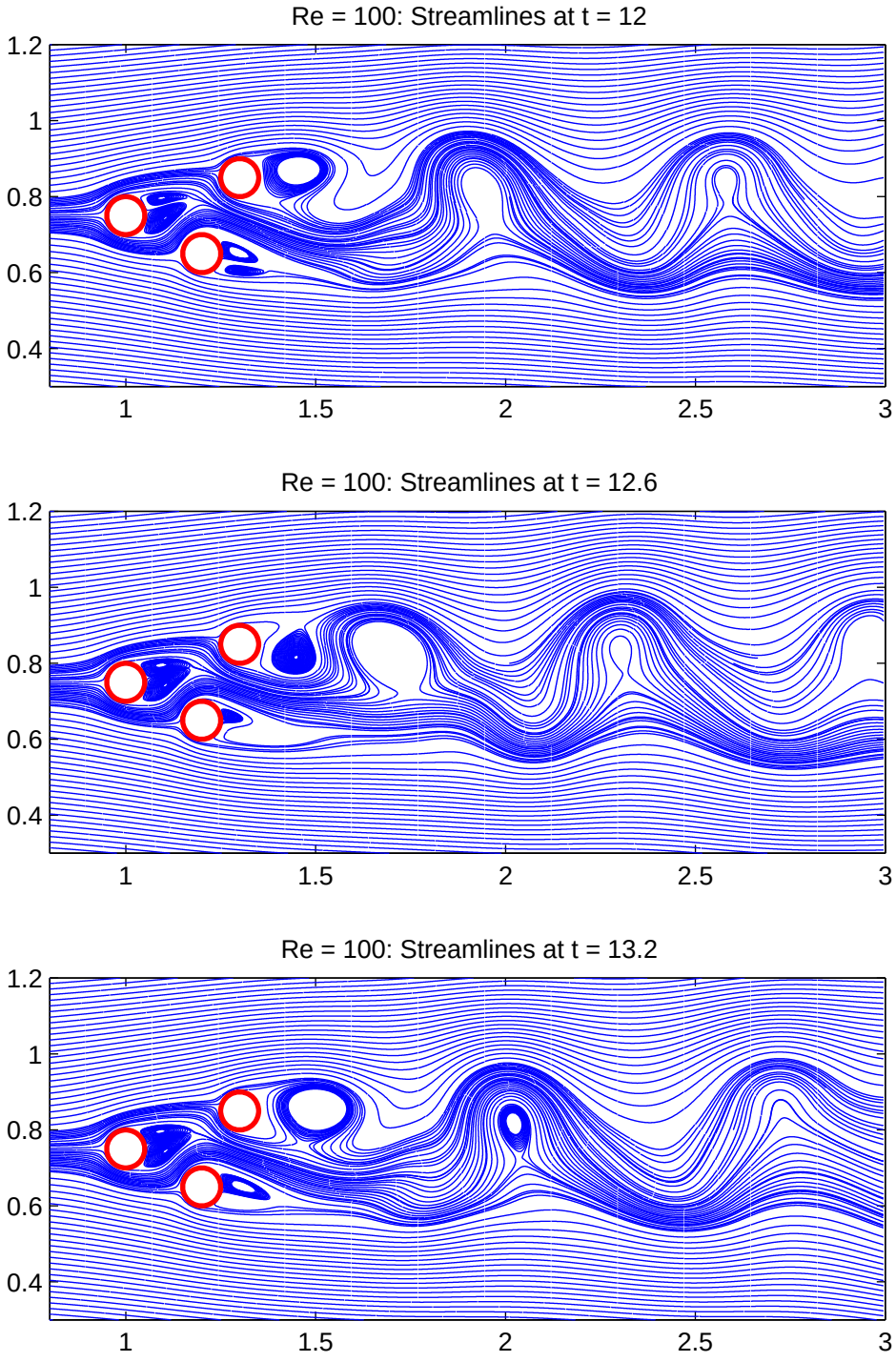


Figure 12: Flow past three cylinders. Streamline plots for Re = 100 at different time.

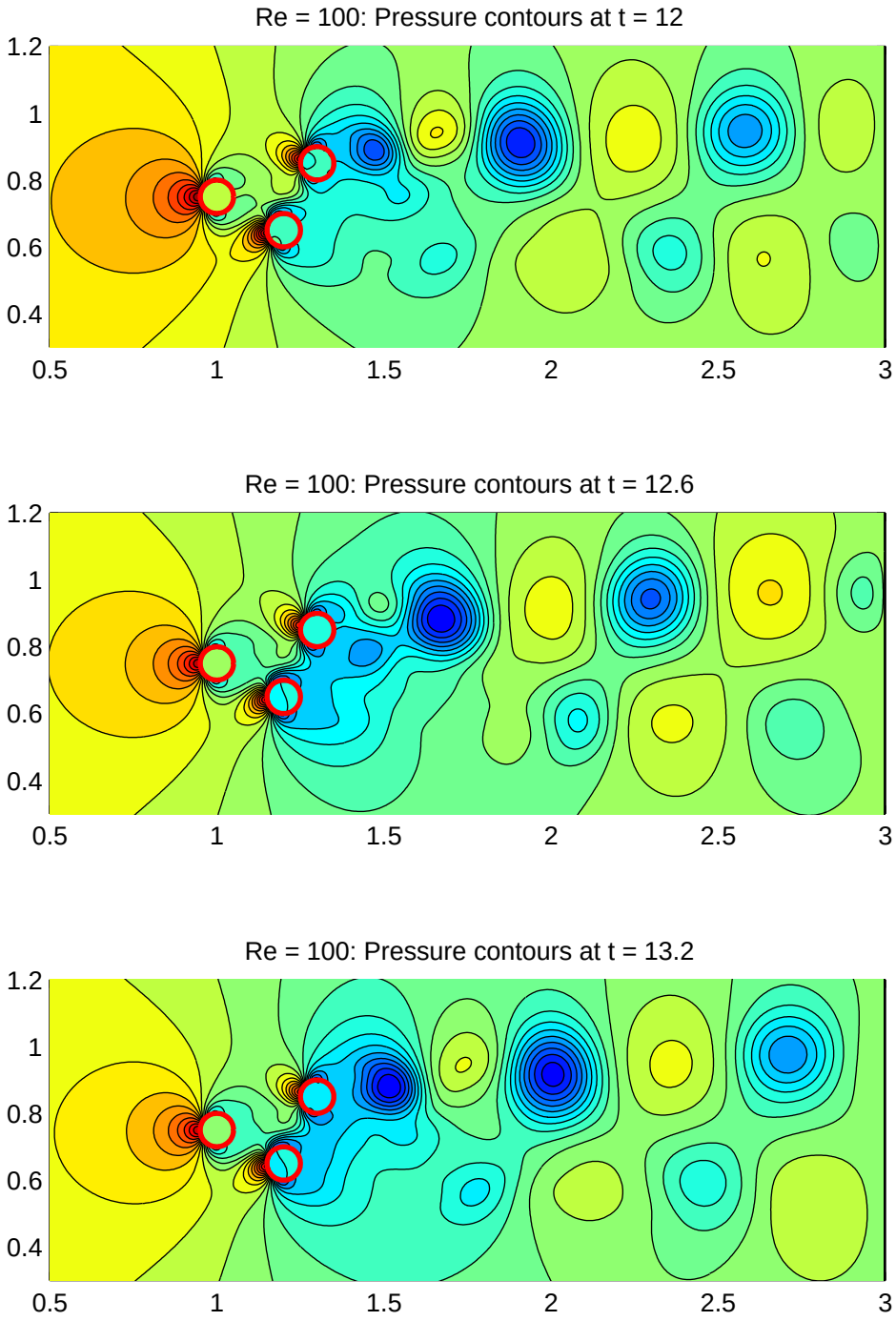


Figure 13: Flow past three cylinders. Pressure contours for Re = 100 at different time.

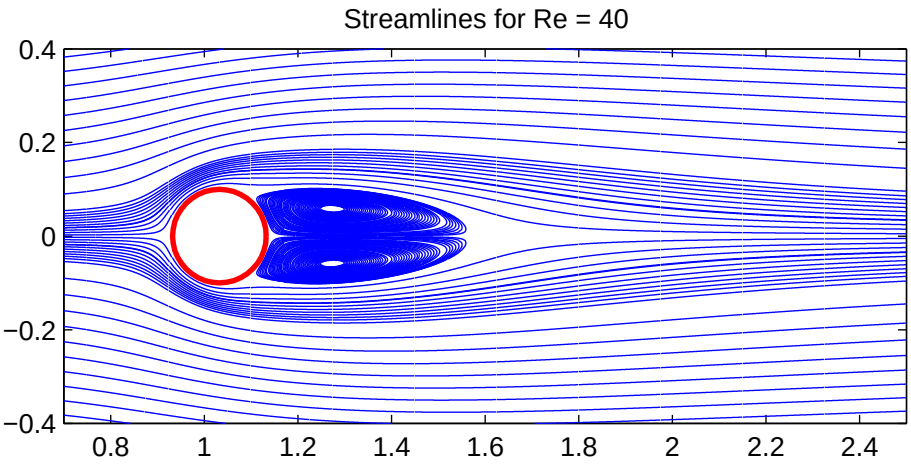


Figure 14: Streamlines for moving cylinder at Re = 40 in the frame of reference attached to the moving cylinder when the wake behind the cylinder is fully developed.

increases rapidly or when the residual does not change much. Numerical experiments show that the residual is small and decreases very little after 2-5 iterations. Hence, we can stop the GMRES iterative process after 2-5 iterations.

Figure 14 shows the streamlines plot for Re = 40 in the frame of reference attached to the moving cylinder when the wake behind the cylinder is fully developed. Figure 15 shows the streamlines plot at the same time level. Table 6 shows the results of the drag coefficient and the length of the recirculation zone at Re = 40. These results are compared to those obtained for the stationary cylinder. We can see that the length of the recirculation zone is in good agreement with that obtained for the stationary cylinder. The drag coefficient is about 7% higher than that obtained for the stationary cylinder. The error is a result of several reasons. Firstly, as the cylinder passing through the underlying grid, the topology changes and this may cause the noise in the force at the moving boundary. Secondly, as we terminate the GMRES algorithm in a few iterations, the truncated error in the residual may cause the inaccuracy in the singular force at the rigid boundary.

5.5 Motion of elastic membranes in irregular domains

In this section, we consider the immersed interface method for the incompressible Navier-Stokes equations in general domains involving immersed flexible and rigid

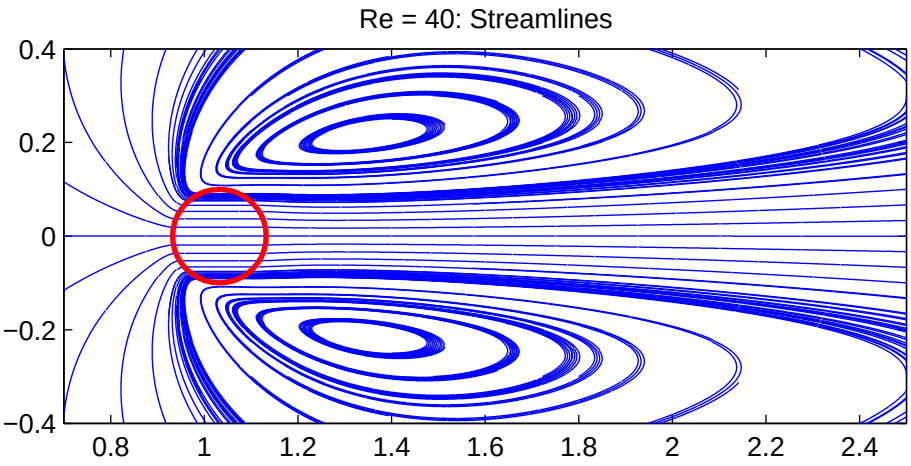


Figure 15: Streamlines for moving cylinder at Re = 40.

$Re = 40$	C_D	L/d
Moving Cylinder	1.67±0.01	2.15
Stationary Cylinder	1.56	2.20

Table 6: Summary results for moving cylinder at Re = 40, compared against stationary cylinder at steady state.

boundaries. The force strength exerted by the elastic membrane is given as,

$$\boldsymbol{f}(s,t) = \frac{\partial}{\partial s}(T(s,t)\boldsymbol{\tau}(s,t)) + \sigma\frac{\partial^2 \boldsymbol{X}}{\partial s^2} \ , \tag{57}$$

where $T(s,t)$ is defined as

$$T(s,t) = T_0\left(\left|\frac{\partial \boldsymbol{X}(s,t)}{\partial s_0}\right| - 1\right) \tag{58}$$

and $\boldsymbol{\tau}(s,t)$ is the unit tangential vector to the interface,

$$\boldsymbol{\tau}(s,t) = \frac{\partial \boldsymbol{X}}{\partial s} \Big/ \left|\frac{\partial \boldsymbol{X}}{\partial s}\right| \ . \tag{59}$$

Here, $\boldsymbol{X}(s,t)$ is the arc-length parametrization of the elastic membrane, s and s_0 are the arc-lengths measured along the current and initial configuration of the membrane, respectively. The scalar T_0 is the stiffness constant which describes the elastic property of the membrane. The scalar σ is the surface tension constant. The location of the flexible boundaries is advanced in time in an implicit manner,

$$\boldsymbol{X}^{n+1} = \boldsymbol{X}^n + \frac{1}{2}\triangle t\left(\boldsymbol{u}^n\left(\boldsymbol{X}^n\right) + \boldsymbol{u}^{n+1}\left(\boldsymbol{X}^{n+1}\right)\right) \tag{60}$$

The BFGS method which is a quasi-Newton method is employed to solve the nonlinear system of equations (60) iteratively to calculate the location of the flexible boundaries. For more details on the immersed interface method for flexible interfaces, see, for example [19, 18].

5.5.1 Grooved channel flow with an immersed elastic membrane

This example considers a Poiseuille flow between two walls, one of which has a groove perpendicular to the streamwise direction. An elastic membrane is immersed in the fluid inside the groove. Under the fluid flow, the elastic membrane circulates inside the groove or the flow moves it out of the groove depending on some parameters such as the location of the elastic membrane, the flow rate, the size of the groove, the stiffness of the membrane. In the numerical simulation, the gap between the walls is 0.2, the depth and the width of the groove are D and W, respectively. The velocity profile at the inflow boundary is parabolic with maximum velocity U_{max} and the viscosity $\mu = 0.02$. Figure 16 illustrates the geometry of the grooved channel and the initial position of the membrane inside the groove. The boundary conditions are inflow at the left boundary and outflow at the right boundary. The velocity is set to zero at the top and bottom boundaries. The no-slip boundary condition at the immersed rigid boundary is enforced by imposing an appropriate singular force at the rigid boundary.

In all simulations presented in this example, a computational domain of $[0, 1.5] \times$

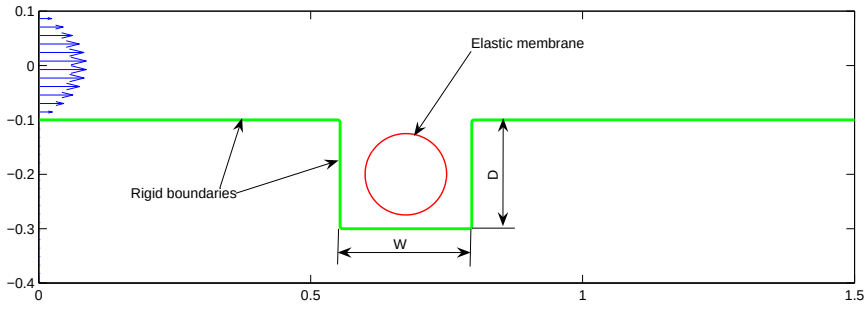


Figure 16: Initial position of an elastic membrane in the simulation of elastic membrane in a groove.

$[-0.4, 0.1]$, a 384×128 grid and a circular membrane with diameter of 0.15 have

been used. This membrane has initially been pre-stretched from the rest state with a diameter of 0.12. We first consider the elastic membrane whose center is located at $(0.675, -0.18)$ inside the groove with $D = 0.2$ and $W = 0.25$. The stiffness constant of the membrane (T_0) of 1.5, the surface tension constant (σ) of 1.0 and the far-field maximum velocity (U_{max}) of 1.0 were specified. Figure 17 shows the positions of the elastic membrane and velocity fields at different time levels. Because of the high relative location of the membrane inside the groove, the flow moves the membrane out of the groove. However, if the membrane is located a bit lower inside the groove, i.e. the center of the membrane is $(0.675, -0.2)$, the membrane only circulates inside the groove under the same flow condition. Figure 18 shows the positions of the elastic membrane inside the groove and velocity fields at different time. A dot on the interface indicates the rotation of the membrane. In these simulations, the timestep Δt of $h/7.5$ has been used. The computational time is about 1.5 hours and 3 hours for the first and the second simulations, respectively. Note that all the simulations presented in this thesis are performed on an IBM Pentium IV 2.4 GHz.

We now keep the location of the membrane center at $(0.675, -0.2)$ and increase the flow rate by increasing the maximum far-field velocity, $U_{max} = 5$. Again, the fluid flow brings the membrane out of the groove. Figure 19 shows the deformation of the membrane under the high flow rate condition. Because of the high flow rate and the low stiffness constant of the membrane, there is significant deformation of the membrane when it tries to climb out of the groove. If one is interested in simulating the interactions of non-deformable, "solid" particles, a high stiffness constant and high surface tension constant would be assigned to increase the stiffness of the membranes. Under the same flow conditions and the same properties of the membrane, we may keep the membrane inside the groove by reducing the width of the groove. Figure 20 shows the rotation of the membrane inside the smaller groove with the width W of 0.2.

5.5.2 Flow in a constriction with immersed elastic membranes

This problem considers the motion of one or more membranes in a domain with a constriction. Figure 21 illustrates the geometry of the constriction and the initial

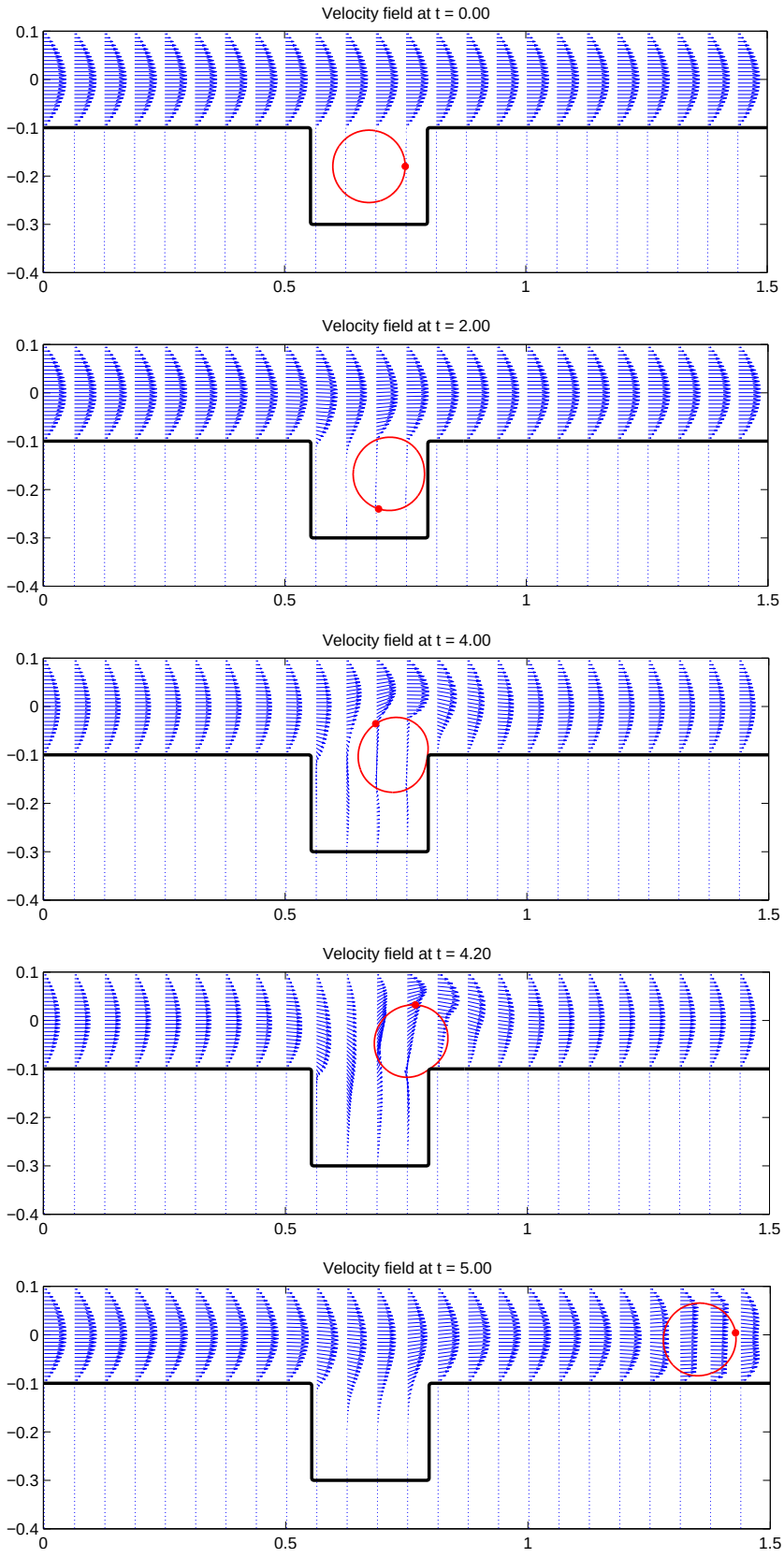


Figure 17: Positions of the elastic membrane and velocity fields at different time. The fluid flow moves the membrane out of the groove.

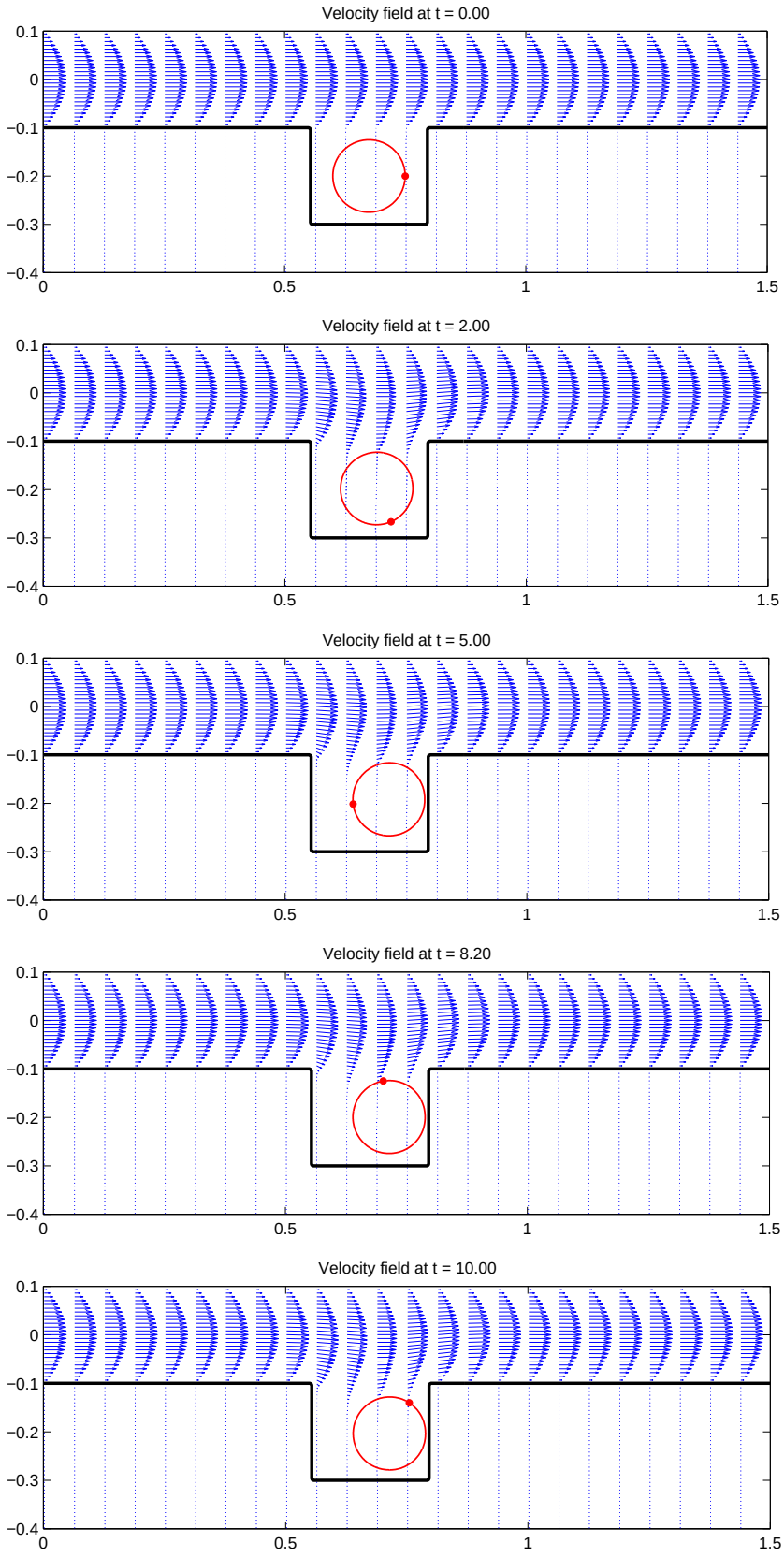


Figure 18: Positions of the elastic membrane and velocity fields at different time. The membrane circulates inside the groove.

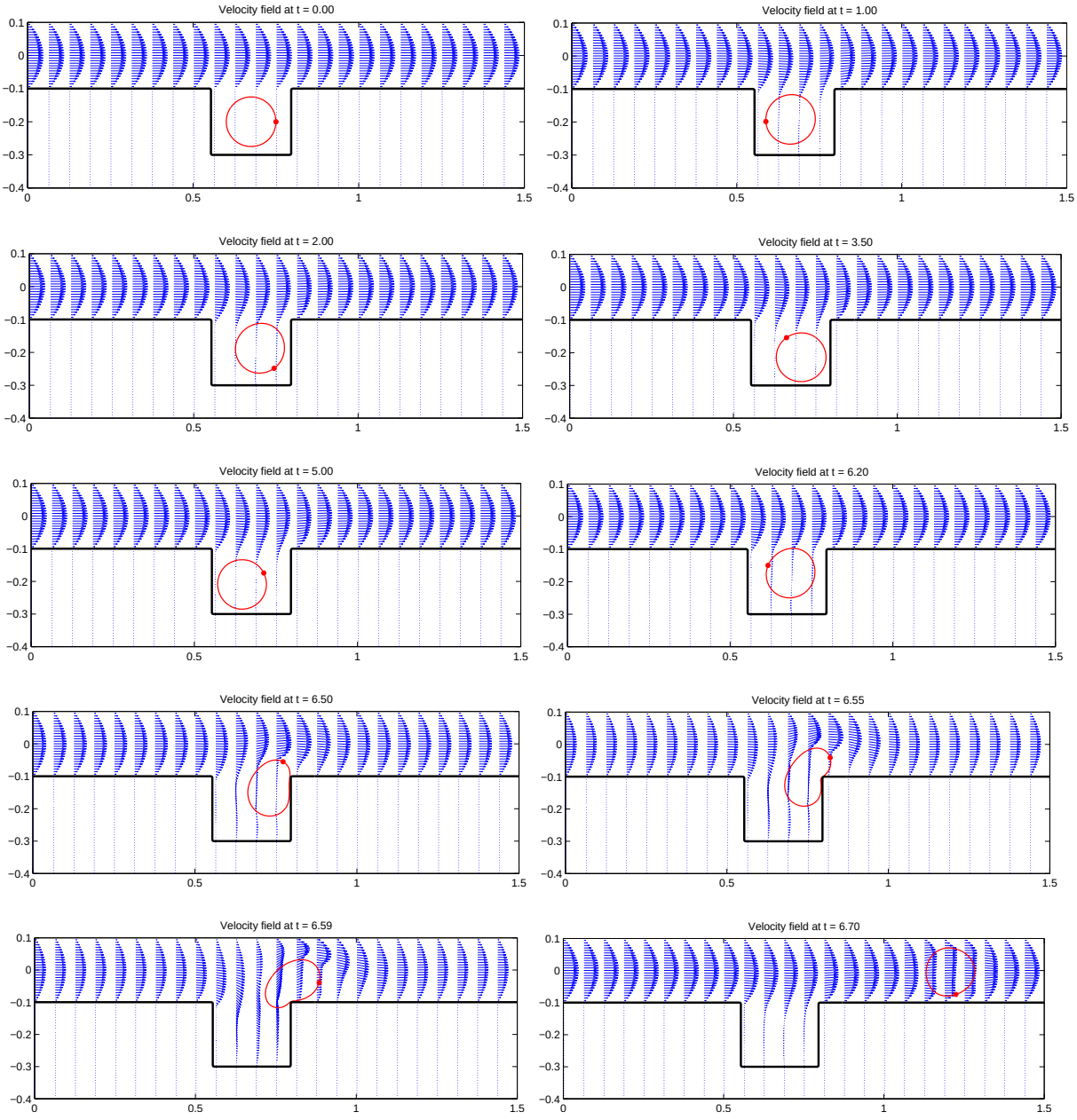


Figure 19: Positions of the elastic membrane and velocity fields at different time. The high flow rate moves the membrane out of the groove even though the membrane initially lies deeply inside the groove.

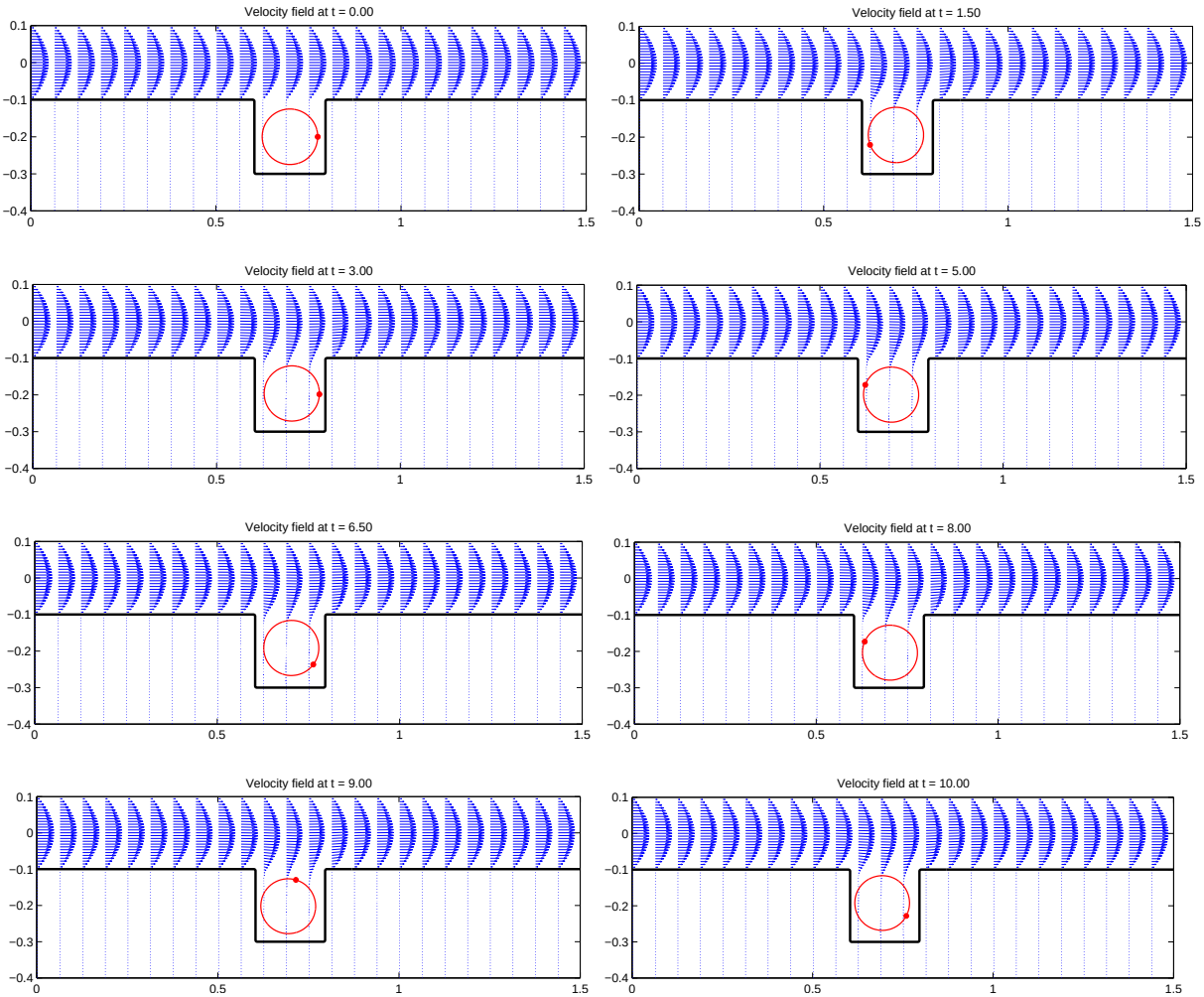


Figure 20: Positions of the elastic membrane and velocity fields at different time. With a small groove, the membrane only circulates inside the groove even with the high flow rate.

position of a single membrane in front of the constriction. In all the simulations presented in this example, a computational domain $[0, 1.5] \times [-0.25, 0.25]$, a 384×128 grid, a fluid viscosity of 0.02 and a surface tension constant $\sigma = 0$ have been used. The boundary conditions are inflow at the left boundary and outflow at the right boundary. A parabolic velocity profile with $U_{max} = 1$ is specified for the velocity at the inflow boundary. The velocity is set to zero at the top and bottom boundaries. The no-slip boundary condition at the immersed rigid boundaries is enforced by imposing appropriate singular forces at the rigid boundaries.

For simulations of a single membrane squeezing through a constriction, a diameter of 0.26, a stiffness constant (T_0) of 2.0 are specified to the circular membrane whose center is located at $(0.37, 0.0)$. The elastic membrane is pre-stretched from the rest state with a diameter of 0.12. Two aspect ratios (ratio of the membrane size to the constriction size) of 1.3 and 1.88 have been considered to investigate the motion of a single membrane through the constriction. We use 60 control points to represent the elastic membrane. We use 235 and 247 markers to represent the rigid boundaries of the constriction with aspect ratios of 1.3 and 1.88, respectively. Figure 22 shows the locations of the elastic membrane and the corresponding velocity fields at different time with an aspect ratio of 1.3. The positions of the membrane squeezing through the smaller constriction are shown in Fig. 23. Fig. 23 shows that it takes a longer time for the membrane to squeeze through a smaller constriction.

For multiple membrane simulations, we performed simulation for the motion of three membranes in the flow through a constriction with an aspect ratio of 0.72. Three membranes have the same stiffness constant T_0 of 4 and diameter of 0.1. The geometry of the computational domain and the initial position of the membranes are illustrated in Fig. 24. Simulations have been performed at $Re = 5$. The Reynolds number is calculated based on the membrane diameter and the maximum far-field velocity U_{max}. Fig. 25 shows the positions of the membranes and the corresponding velocity fields at different time levels.

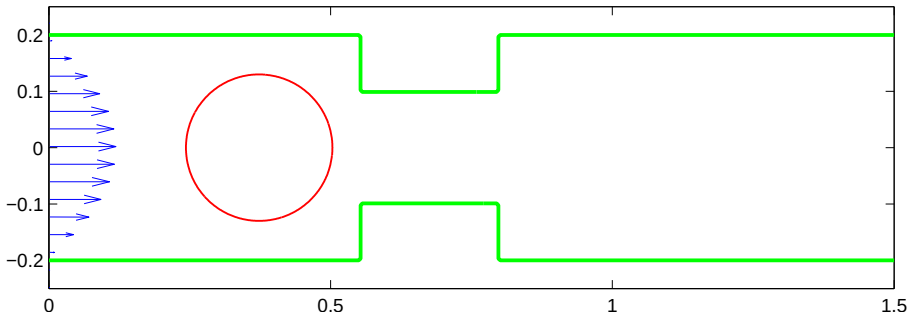


Figure 21: Initial position of an elastic membrane in the simulation of elastic membrane squeezing through a constriction.

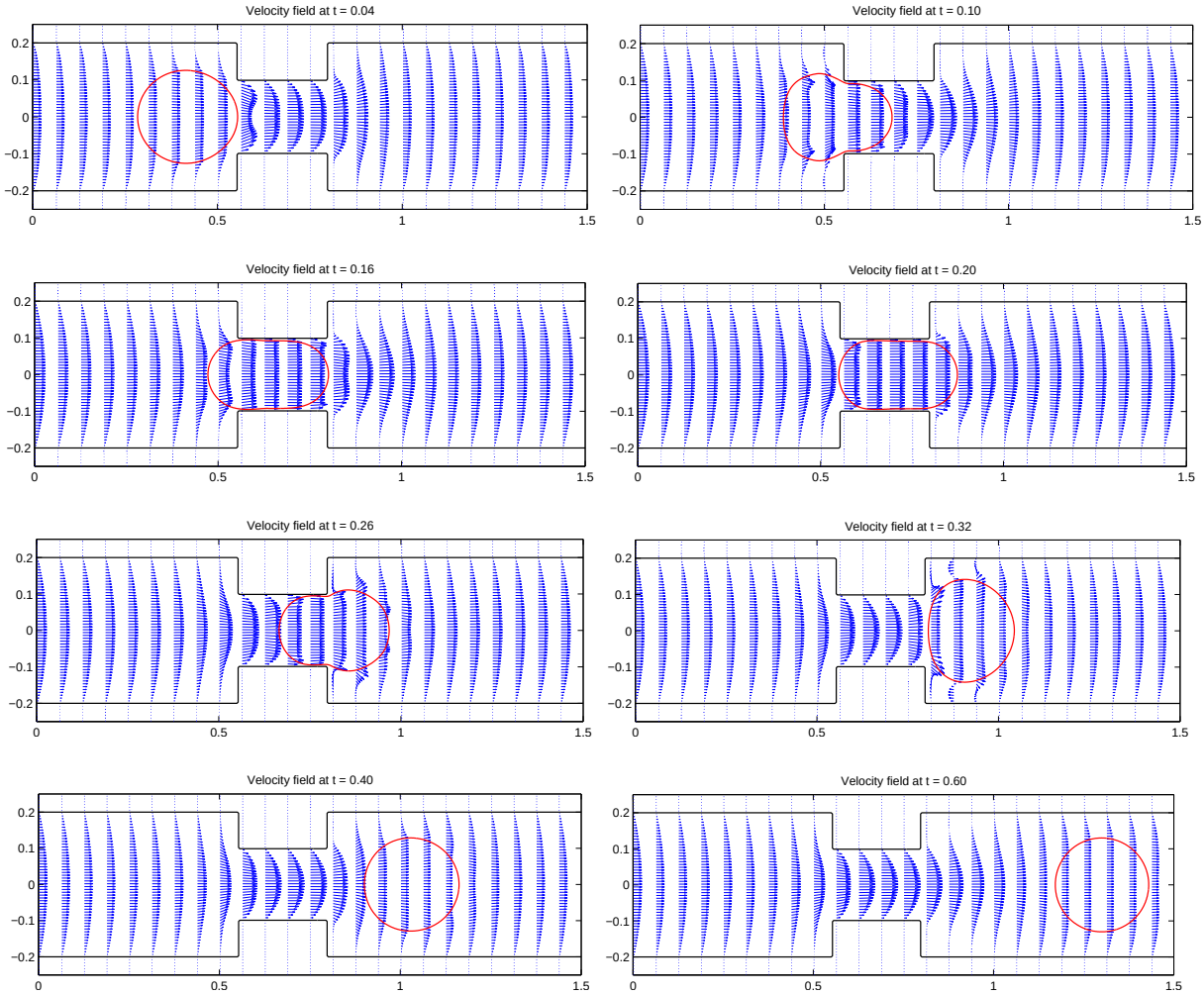


Figure 22: A single elastic membrane with stiffness constant $T_0 = 2$ squeezes through a constriction with aspect ratio of 1.3.

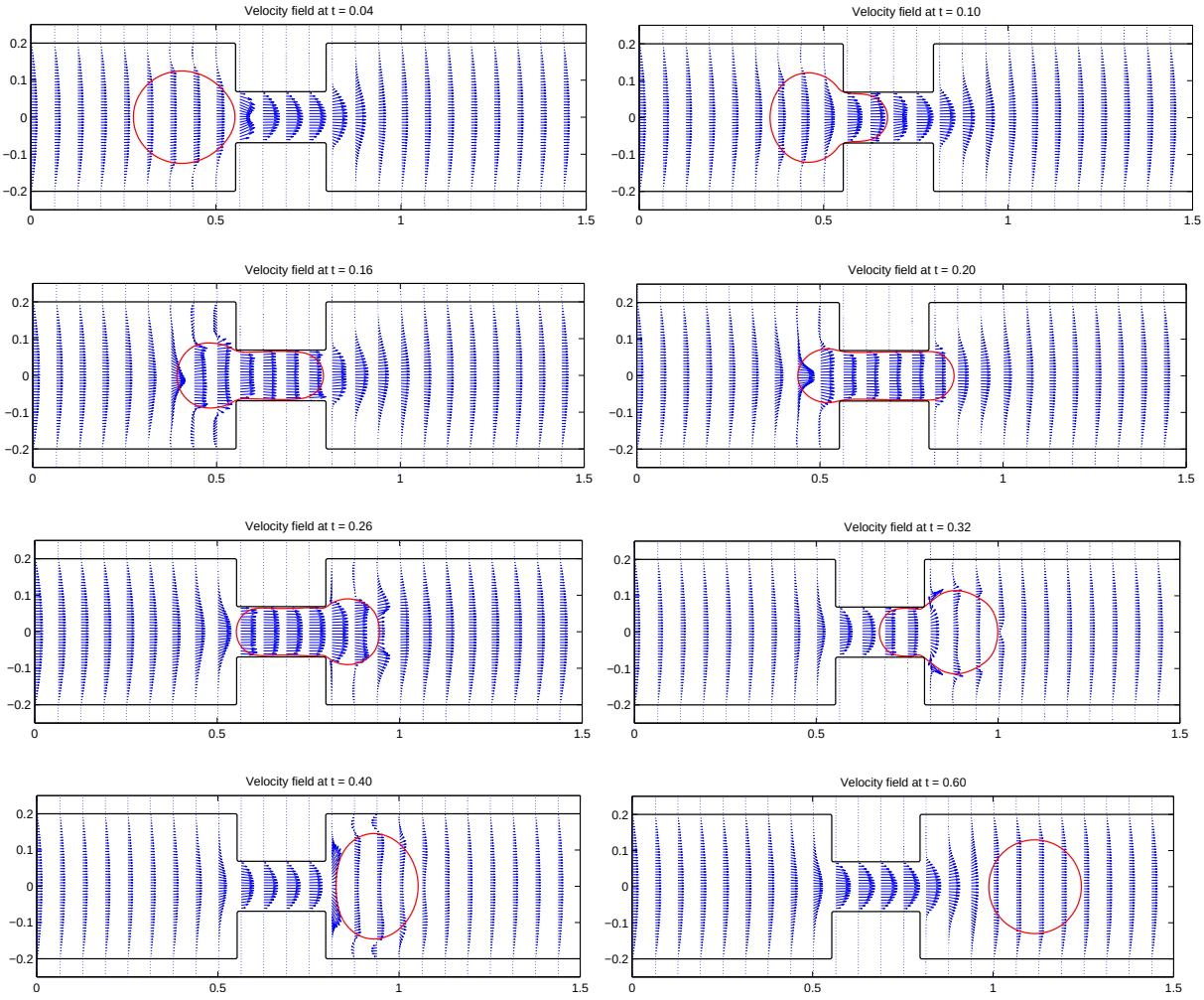


Figure 23: A single elastic membrane with stiffness constant $T_0 = 2$ squeezes through a constriction with aspect ratio of 1.88.

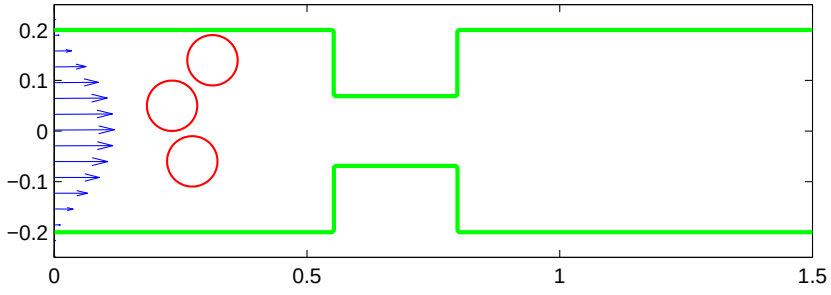


Figure 24: Computational domain for studying the interaction between elastic membranes at the entrance to a constriction. Initial positions of elastic membranes in the 3-membrane simulations.

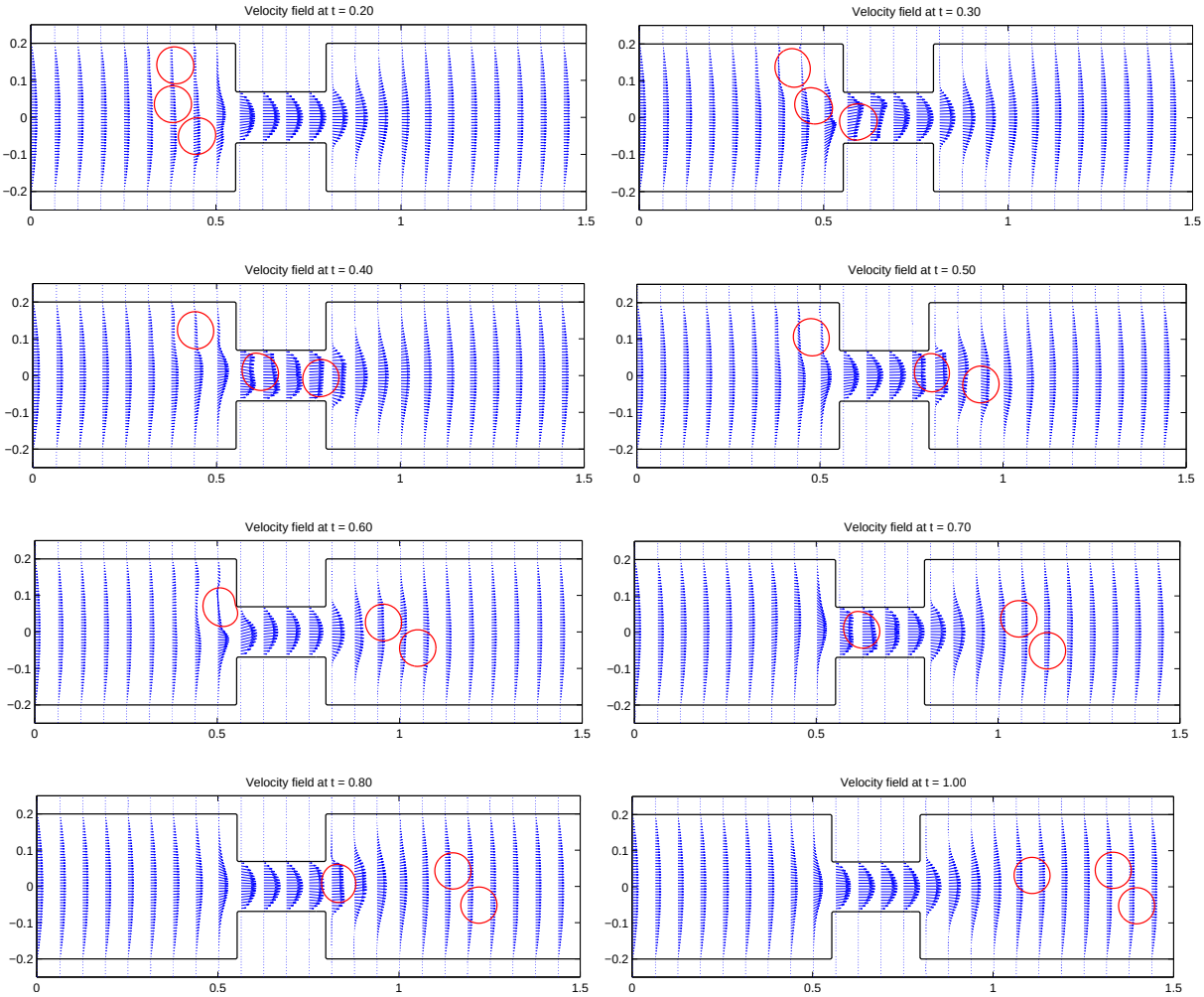


Figure 25: The positions of the elastic membranes and velocity fields at different time. Simulations have been performed for Re $= 5$, stiffness constant $T_0 = 4$ and $\Delta t = \Delta x/7.5$.

6 Conclusions

In this paper, an immersed interface algorithm is developed for solving the Navier-Stokes equations in complicated domains. It was shown that we can use the immersed interface method to handle viscous, incompressible flow problems involving rigid boundaries. In our algorithm, rigid boundaries are treated as immersed boundaries at which singular forces are imposed to enforce the no-slip conditions. This immersed interface algorithm has also been developed further for solving the Navier-Stokes equations with flexible interfaces and rigid boundaries. Numerical experiments have shown that our algorithm can handle complex fluid-membrane interactions, membrane-membrane interactions and the interactions between flexible boundaries and rigid boundaries simultaneously. Numerical simulations have been performed to reproduce some numerical results for the flow past a circular cylinder problem as a benchmark test for our method when dealing with rigid boundaries. It was shown that our numerical results are in good agreement with other numerical and experimental results in both steady and unsteady regimes. Moving rigid boundary is also considered to show different abilities of our algorithm. We also performed simulations for problems involving motions of membranes in a grooved channel and a constriction. These results are used to show the capability of handling flexible interfaces and rigid boundaries simultaneously of our method. We would like to extend our code to three dimensions to solve for more realistic problems such as biological flow problems. In 2D, the interface is discretized using a set of control points. In 3D, the interface is a surface and hence is discretized using triangular mesh. Singular forces are computed at the nodes of the triangulations and are used to compute the jump conditions of the solutions and their derivatives. A projection method is then employed to update the solutions in time and the extension to 3D of the projection method is straightforward. Another issue that needs to be resolved is the computation of the interaction forces realized when the two membranes approach each other or when a membrane comes close to rigid boundaries. Since our interest is the motion of deformable particles in biological flows, the colloidal interaction force between two particles or between a particle with rigid boundaries is a combination of Van de Waals attractive force, electrostatic repulsive force and short-ranged Born repulsive force [30].

Appendix

A Modified Bilinear Interpolation

In this section, we derive a bilinear interpolation formula to compute the velocity at a control point. The velocity at the control points, $\boldsymbol{U_k}$, is interpolated from the velocity at the nearby Cartesian grid points. Thus, we can write

$$\boldsymbol{U_k} = \boldsymbol{U}(\boldsymbol{X}_k) = \mathcal{B}(\boldsymbol{u}) \ , \tag{61}$$

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 Figure 26, the velocity at the control point $\boldsymbol{X}_k$

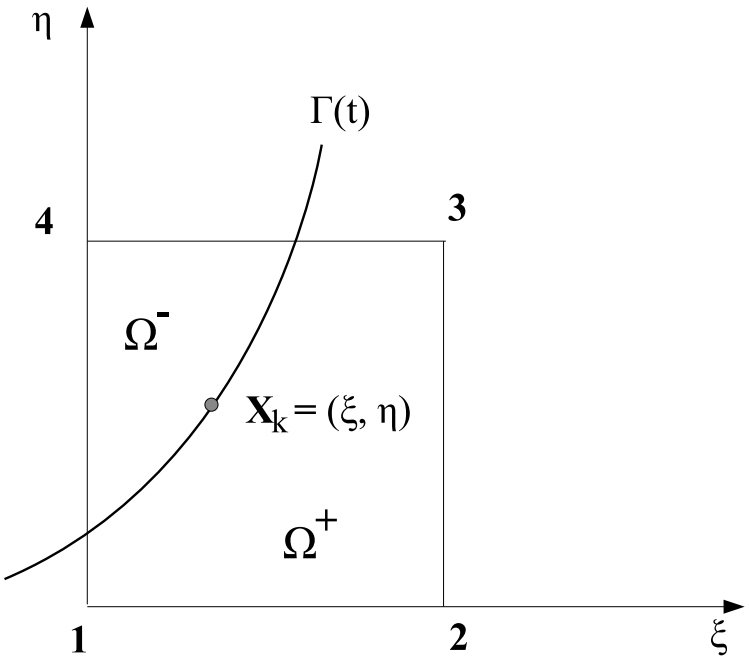


Figure 26: Velocity at a control point is interpolated from the velocity at the four neighbor grid points using modified bilinear interpolation.

is interpolated from the velocity at four neighbor grid points as follows

$$\boldsymbol{U}_k = (1-\xi)(1-\eta)\boldsymbol{u}_1 + \boldsymbol{C}_1 + \xi(1-\eta)\boldsymbol{u}_2 + \boldsymbol{C}_2 + \xi\eta\boldsymbol{u}_3 + \boldsymbol{C}_3 + (1-\xi)\eta\boldsymbol{u}_4 + \boldsymbol{C}_4 \tag{62}$$

where $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_4$ are correction terms, $\xi = \frac{X-x_1}{h}$, $\eta = \frac{Y-y_1}{h}$ and h is the grid size. Jump conditions $[\boldsymbol{u}_x]$ and $[\boldsymbol{u}_y]$ are required at the control point to compute the cor-

rection terms. The correction terms can be derived using Taylor series expansion and have the following forms:

$$\boldsymbol{C}_1 = \begin{cases} h(1-\xi)(1-\eta)\big(\xi[\boldsymbol{u}_x] + \eta[\boldsymbol{u}_y]\big), & \boldsymbol{x}_1 \in \Omega^+ \\ 0, & \boldsymbol{x}_1 \in \Omega^-, \end{cases} \tag{63}$$

$$\boldsymbol{C}_2 = \begin{cases} -h\xi(1-\eta)\big((1-\xi)[\boldsymbol{u}_x] - \eta[\boldsymbol{u}_y]\big), & \boldsymbol{x}_2 \in \Omega^+ \\ 0, & \boldsymbol{x}_2 \in \Omega^-, \end{cases} \tag{64}$$

$$\boldsymbol{C}_3 = \begin{cases} -h\xi\eta\big((1-\xi)[\boldsymbol{u}_x] + (1-\eta)[\boldsymbol{u}_y]\big), & \boldsymbol{x}_3 \in \Omega^+ \\ 0, & \boldsymbol{x}_3 \in \Omega^-, \end{cases} \tag{65}$$

$$\boldsymbol{C}_4 = \begin{cases} h(1-\xi)\eta\big(\xi[\boldsymbol{u}_x] - (1-\eta)[\boldsymbol{u}_y]\big), & \boldsymbol{x}_4 \in \Omega^+ \\ 0, & \boldsymbol{x}_4 \in \Omega^-. \end{cases} \tag{66}$$

References

[1] J. Adams, P. Swarztrauber, and R. Sweet. FISHPACK: Efficient FORTRAN subprograms for the solution of separable eliptic partial differential equations, 1999. Available on the web at http://www.scd.ucar.edu/css/software/fishpack/.

[2] C. H. Bischof. Incremental condition estimation. *SIAM J. Matrix Anal. Appl.*, 11:312–322, 1990.

[3] M. Braza, P. Chassaing, and H. Ha Minh. Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder. *J. Fluid. Mech.*, 165:79–130, 1986.

[4] D. L. Brown, R. Cortez, and M. L. Minion. Accurate projection methods for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 168:464–499, 2001.

[5] P. N. Brown and H. F. Walker. GMRES on (nearly) singular systems. *SIAM J. Matrix Anal. Appl.*, 18(1):37–51, 1997.

[6] D. Calhoun. A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions. *J. Comput. Phys.*, 176:231–275, 2002.

[7] M. Coutanceau and R. Bouard. Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 1. Steady flow. *J. Fluid. Mech.*, 79(2):231–256, 1977.

[8] S. C. R. Dennis and G. Chang. Numerical solutions for steady flow past a circular cylinder at Reynolds number up to 100. *J. Fluid. Mech.*, 42(3):471–489, 1970.

[9] R. Dillon, L. J. Fauci, and D. Graver. A microscale model of bacterial swimming, chemotaxis and substrate transport. *J. Theor. Biol.*, 177:325–340, 1995.

[10] C. D. Eggleton and A. S. Popel. Large deformation of red blood cell ghosts in a simple shear flow. *Phys. Fluids*, 10:1834–1845, 1998.

[11] L. Fauci and C. S. Peskin. A computational model of aquatic animal locomotion. *J. Comput. Phys*, 77:85–108, 1988.

[12] A. L. Fogelson. A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting. *J. Comput. Phys.*, 56:111–134, 1984.

[13] A. L. Fogelson. Continuum models of platelet aggregation: Formulation and mechanical properties. *SIAM J. Applied Math.*, 52:1089–1110, 1992.

[14] B. Fornberg. A numerical study of steady viscous flow past a circular cylinder. *J. Fluid. Mech.*, 98(4):819–855, 1980.

[15] D. Goldstein, R. Handler, and L. Sirovich. Modeling a no-slip flow with an external force field. *J. Comput. Phys.*, 105:354–366, 1993.

[16] J. Kim and P. Moin. Application of a fractional step method to incompressible Navier-Stokes equations. *J. Comput. Phys.*, 59:308–323, 1985.

[17] M. C. Lai and C. S. Peskin. An immersed boundary method with formal second order accuracy and reduced numerical viscosity. *J. Comput. Phys.*, 160:707–719, 2000.

[18] D.V. Le. *An immersed interface method for solving viscous incompressible flows involving rigid and flexible boundaries*. PhD thesis, Singapore-MIT Alliance, June 2005.

[19] D.V. Le, B.C. Khoo, and J. Peraire. An immersed interface method for the incompressible Navier-Stokes equations. Presented at the SMA Symposium, Singapore 2004.

[20] D.V. Le, B.C. Khoo, and J. Peraire. An immersed interface method for the incompressible Navier-Stokes equations in irregular domains. In K. J. Bathe, editor, *Proceedings of the Third M.I.T. Conference on Computational Fluid and Solid Mechanics*, pages 710–716. Elsevier Science, June 2005.

[21] L. Lee. *Immersed interface methods for incompressible flow with moving interfaces*. PhD thesis, University of Washington, 2002.

[22] L. Lee. An immersed interface method for incompressible Navier-Stokes equations. *SIAM J. Sci. Comput.*, 25(3):832–856, 2003.

[23] R. J. LeVeque and Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM J. Numer. Anal.*, 31:1019–1044, 1994.

[24] R. J. LeVeque and Z. Li. Immersed interface method for Stokes flow with elastic boundaries or surface tension. *SIAM J. Sci. Comput.*, 18(3):709–735, 1997.

[25] Z. Li and M.C. Lai. The immersed interface method for the Navier-Stokes equations with singular forces. *J. Comput. Phys.*, 171:822–842, 2001.

[26] Z. Li and C. Wang. A fast finite difference method for solving Navier-Stokes equations on irregular domains. *Comm. Math. Sci.*, 1(1):180–196, 2003.

[27] C. Liu, X. Sheng, and C. H. Sung. Preconditioned multigrid methods for unsteady incompressible flows. *J. Comput. Phys.*, 139:35–57, 1998.

[28] C. S. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 25:220–252, 1977.

[29] C. S. Peskin. The immersed boundary method. *Acta Numerica*, 11(2):479–517, 2002.

[30] V. Ramachandran, R. Venkaresan, G. Tryggvason, and H. S. Fogler. Low Reynolds number interactions between colloidal particles near the entrance to a cylindrical pore. *J. Colloid and Interface Science*, 229:311–322, 2000.

[31] D. Russell and Z. J. Wang. A Cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow. *J. Comput. Phys.*, 191:177–205, 2003.

[32] Y. Sadd. GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM J. Sci. Stst. Comput.*, 7:856–869, 1986.

[33] C.W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing scheme, II. *J. Comput. Phys.*, 83:32–78, 1989.

[34] A.L.F. Lima E. Silva, A. Silveira-Neto, and J.J.R Damasceno. Numerical simulation of two-dimensional flows over a circular cylinder using the immersed boundary method. *J. Comput. Phys.*, 189:351–370, 2003.

[35] D. J. Tritton. Experiments on the flow past a circular cylinder at low Reynolds numbers. *J. Fluid. Mech.*, 6(4):547–567, 1959.

[36] N. T. Wang and A. L. Fogelson. Computational methods for continuum models of platelet aggregation. *J. Comput. Phys*, 151:649–675, 1999.

[37] A. Wiegmann and K.P. Bube. The explicit-jump immersed interface method: Finite difference methods for PDEs with piecewise smooth solutions. *SIAM J. Numer. Anal.*, 37(3):827–862, 2000.

[38] C. H. K. Williamson. Vortex dynamics in the cylinder wake. *Ann. Rev. Fluid Mech.*, 28:477–539, 1996.

[39] T. Ye, R. Mittal, H.S. Udaykumar, and W. Shyy. An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundary. *J. Comput. Phys.*, 156:209–240, 1999.