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

by

Duc Vinh Le

B.Eng., Aeronautical Engineering, HCMC University of Technology (2001)

SUBMITTED TO THE SMA OFFICE IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN HIGH PERFORMANCE COMPUTATION FOR ENGINEERING SYSTEMS
(HPCES)
AT THE
SINGAPORE-MIT ALLIANCE
June 2005

Author .....
HPCES Programme
June 03, 2005

Certified by .....
Prof. Khoo Boo Cheong
SMA Fellow, NUS
Thesis Advisor

Certified by .....
Prof. Jaime Peraire
SMA Fellow, MIT
Thesis Advisor

Accepted by .....
Prof. Khoo Boo Cheong
Programme Co-Chair
HPCES Programme

Accepted by .....
Prof. Jaime Peraire
Programme Co-Chair
HPCES Programme

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

by

Duc Vinh Le

Submitted to the SMA Office
on June 03, 2005, in Partial Fulfillment of the
Requirements for the degree of
Doctor of Philosophy in High Performance Computation for Engineering Systems

Abstract

We present an immersed interface method for the incompressible Navier-Stokes equations capable of handling rigid and flexible boundaries. The immersed boundaries are represented by a number of Lagrangian control points. In order to guarantee that the no-slip condition on the rigid 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 at the rigid boundary is determined by solving a small system of equations at each time step. For flexible boundary, the forces that the boundary exerts on the fluid are computed from the constitutive relation of the flexible boundary and are applied to the fluid through the jump conditions. The position of the flexible boundary is updated implicitly using a quasi-Newton method (BFGS) within each timestep. 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, Immersed boundary method, Navier-Stokes equations, Cartesian grid method, finite difference, fast Poisson solvers, irregular domains, fluid-membrane interaction, viscous flows, incompressible flows.

Thesis Advisors:

1. Assoc. Prof. Khoo Boo Cheong, SMA Fellow, NUS
2. Prof. Jaime Peraire, SMA Fellow, MIT

Acknowledgments

I would like to express my sincerest appreciation to my research advisors, Jaime Peraire and Khoo Boo Cheong, for their patience, encouragement, and generous support. Jaime's clarity and insight into many problems numerically and physically was the key factor that helped me to start and finish my research project. Boo Cheong gave me the freedom to explore my ideas while providing me with focus to make progress towards a degree. With their financial support, I was able to spend the second semester stay at MIT.

I would like to thank Li Zhilin of the North Carolina State University (NCSU) for many useful discussions during his visit at NUS and during my visit at NCSU. I also would like to thank Thay Tong at the HCMC University of Technology for encouraging me to join the HPCES programme.

Finally, I would like to thank my family, especially my parents and Nhu Tram for their wonderful support during my four years living abroad.

This work is supported by the Singapore-MIT Alliance programme. I would like to thank the SMA office for their financial support for my first semester stay at MIT.

Contents

1 Introduction **17**

1.1 Immersed Boundary Method and Immersed Interface Method 18

1.1.1 Immersed Boundary Method 18

1.1.2 Immersed Interface Method 21

1.2 Existing methods for solving the Navier-Stokes equations with immersed interfaces 22

1.2.1 Finite Element Method 22

1.2.2 Boundary Integral Method 23

1.2.3 Ghost Fluid Method 23

1.3 Outline of this thesis 24

2 An Immersed Interface Method for the Navier-Stokes equations with deformable interfaces **27**

2.1 Jump conditions across the interface 28

2.2 Generalized finite difference formulas 30

2.3 Projection methods 34

2.3.1 Pressure-free projection method 36

2.3.2 Pressure-increment projection method 37

2.3.3 Projection method for the immersed interface method 39

2.3.4 Correction terms 42

2.4 Implicit scheme for moving interface 44

2.5 Implementation 47

2.6 Numerical results 49

2.6.1 Forced flow 49
2.6.2 Rotational flow 50
2.6.3 Surface tension 52
2.6.4 Elastic membrane 53

3 An Immersed Interface Method for the Navier-Stokes equations with rigid boundaries 65

3.1 Introduction 65
3.2 Singular force evaluation 68
3.3 Implementation 71
3.4 Numerical results 73
3.4.1 Rotational flow 73
3.4.2 Flow past a circular cylinder 74
3.4.3 Flow past a flat plate 79
3.4.4 Flow past several cylinders 82
3.4.5 Grooved channel flow 83
3.4.6 Flow past a moving circular cylinder 83

4 An Immersed Interface Method for solving viscous, incompressible flows involving rigid and flexible boundaries 91

4.1 Introduction 91
4.2 Implementation 93
4.3 Numerical results 96
4.3.1 Grooved channel flow with an immersed elastic membrane . . 96
4.3.2 Flow in a constriction with immersed elastic membranes . . . 98

5 Conclusions and future work 109

5.1 Conclusions 109
5.2 Future work 110

A Jump conditions across an immersed interface 113

List of Figures

1-1 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. 20

1-2 Spreading singular force at a control point to nearby grid points and interpolating velocity at another control point. 20

1-3 Shock-induced collapse of a cylindrical air cavity in water. 1-3(a) Pressure field at time t = 2.36 μs, 1-3(b) Pressure field at time t = 3.34 μs. 25

1-4 Interaction of a planar shock wave in air with a water column. Air is treated as compressible fluid and water is treated as incompressible fluid. Shock Mach speed M = 1.3. 1-4(a) Pressure field at time t = 6.42 μs, 1-4(b) Pressure field at time t = 20.2 μs. 25

2-1 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 $\Gamma(t)$. 29

2-2 The interface cuts a grid line between two grid points at $x = \alpha$. 2-2(a) $x_i \in \Omega^-$ and $x_{i+1} \in \Omega^+$. 2-2(b) $x_i \in \Omega^+$ and $x_{i+1} \in \Omega^-$. 31

2-3 Two interfaces cut a grid line between two grid points x_{i-1} and x_{i+1}. 2-3(a) $x_{i-1} \in \Omega^+$, $x_i \in \Omega^-$ and $x_{i+1} \in \Omega^+$. 2-3(b) $x_{i-1} \in \Omega^-$, $x_i \in \Omega^+$ and $x_{i+1} \in \Omega^-$. 32

2-4 Two interfaces cut a grid line between two grid points x_i and x_{i+1}. 2-4(a) $x_i \in \Omega^+$ and $x_{i+1} \in \Omega^+$. 2-4(b) $x_i \in \Omega^-$ and $x_{i+1} \in \Omega^-$. 32

2-5 The MAC staggered grid in two dimensions. 39

2-6 Interface and mesh geometry near the grid point (i, j). 44

2-7 Velocity field at time t = 10 with a 64×64 grid, $\mu = 0.02$, $f_1 = 0$, $f_2 = 10\mu$. 2-7(a) The plot of the x component of velocity field at time t = 10. 2-7(b) The plot of velocity field at time t = 10. 51

2-8 Pressure is smooth over the domain since the force is only along the tangential direction of the interface. 51

2-9 The location of the interface at different times. 54

2-10 The evolution of r_x and r_y with $\mu = 0.1$ and $\gamma = 10$. The interface oscillates as it converges to the equilibrium state 55

2-11 The evolution of r_x and r_y with $\mu = 1$ and $\gamma = 1$. The interface relaxes gradually to the equilibrium state without oscillations. 55

2-12 The pressure distribution at different times. 56

2-13 The error in the location of the interface at t = 1 as measured in: 2-13(a) L_2 norm, 2-13(b) maximum norm. Viscosity, $\mu = 0.1$ and surface tension constant, $\gamma = 10$. 57

2-14 Initial, unstretched and equilibrium positions of the elastic membrane. 57

2-15 Number of BFGS iterations performed at each timestep to run the simulations to $t = 10$. 2-15(a): $\Delta t = \Delta x/5$, 2-15(b): $\Delta t = 2\Delta x/5$. . . 59

2-16 The evolution of r_x and r_y with $\mu = 0.1$ and $T_0 = 10$. The interface oscillates as it converges to the equilibrium state. 59

2-17 The velocity fields at different times. Simulation is performed on a 64×64 grid with 48 control points and $\Delta t = \Delta x/5$. 60

2-18 The pressure distribution at different times. Simulation is performed on a 64×64 grid with 48 control points and $\Delta t = \Delta x/5$. 61

2-19 Grid refinement analysis for studying the conservation of the area enclosed by the membrane at $t = 0.5$. 62

2-20 The evolution of r_x and r_y with $\mu = 0.02$ and $T_0 = 10$. The interface oscillates as it converges to the equilibrium state. 62

2-21 The evolution of r_x and r_y with $\mu = 1$ and $T_0 = 10$. The interface relaxes gradually to the equilibrium state without oscillations. 63

2-22 A cross section of u, the x component of the velocity field at $t = 0.2$ and $y = -0.207$. It is continuous but non-smooth across the interface. 63

3-1 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 $\omega = 2$. 3-1(a) Plot of the x component of velocity field. 3-1(b) Plot of velocity field. 74

3-2 Streamlines for Re = 20 and Re = 40. 76

3-3 Pressure fields for Re = 20 and Re = 40. 77

3-4 Lift Coefficients at Re = 50. 77

3-5 Pressure fields for Re = 100, Re = 200 and Re = 300. 80

3-6 Drag Coefficients for Re = 100 and Re = 200. 81

3-7 Lift Coefficients for Re = 100 and Re = 200. 81

3-8 Pressure field at t = 10, $\triangle t = \triangle x/4$. (a) Re = 20, (b) Re = 50, (c) Re = 100, (d) Re = 1000, (e) Re = 5000. 82

3-9 Flow past three cylinders. Streamline plots for Re = 100 at different times. 84

3-10 Flow past three cylinders. Pressure contours for Re = 100 at different times. 85

3-11 Flow past seven cylinders. Streamlines and pressure contours at Re = 100. 86

3-12 Velocity fields for Re = 100, Re = 500 and Re = 3000. 87

3-13 Pressure fields for Re = 100, Re = 500 and Re = 3000. 88

3-14 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. 90

3-15 Streamlines for moving cylinder at Re = 40. 90

4-1 Initial position of an elastic membrane in the simulation of elastic membrane in a groove. 97

4-2 Positions of the elastic membrane and velocity fields at different times. The fluid flow moves the membrane out of the groove. 99
4-3 Positions of the elastic membrane and velocity fields at different times. The membrane rotates inside the groove. 100
4-4 Positions of the elastic membrane and velocity fields at different times. The high flow rate moves the membrane out of the groove even though the membrane initially lies deep inside the groove. 101
4-5 Positions of the elastic membrane and velocity fields at different times. With a small groove, the membrane only rotates inside the groove even with the high flow rate. 102
4-6 Initial position of an elastic membrane in the simulation of an elastic membrane squeezing through a constriction. 103
4-7 A single elastic membrane with stiffness constant $T_0 = 2$ squeezes through a constriction with aspect ratio of 1.3. 104
4-8 A single elastic membrane with stiffness constant $T_0 = 2$ squeezes through a constriction with aspect ratio of 1.88. 105
4-9 Computational domain for studying the interaction between several elastic membranes at the entrance to a constriction. Initial positions of elastic membranes in the 3-membrane simulations. 105
4-10 The positions of the elastic membranes and velocity fields at different times. Simulations have been performed for Re = 5, stiffness constant $T_0 = 4$ and $\Delta t = \Delta x/7.5$. 106
4-11 The positions of the elastic membranes and velocity fields at different times. Simulations have been performed for Re = 5, stiffness constant $T_0 = 8$ and $\Delta t = \Delta x/15$. 107

B-1 Velocity at a control point is interpolated from the velocity at the four neighbor grid points using modified bilinear interpolation. 118

List of Tables

2.1 Grid refinement analysis for the forced flow problem is performed at time $t = 10$. Viscosity $\mu = 1$ and $\triangle t = \triangle x$. 50

2.2 Grid refinement analysis for the rotational flow problem performed at $t = 10$, $\triangle t = \triangle x/4$ and $\mu = 0.02$. 52

3.1 The grid refinement analysis for the rotational flow problem with $\mu = 0.02$, $\triangle t = \triangle x/4$, at t = 10. 73

3.2 Length of the recirculation zone, Angle of Separation and Drag Coefficient for Re = 20 and Re = 40 78

3.3 Drag Coefficients for Re = 100 and Re = 200 79

3.4 Lift Coefficients for Re = 100 and Re = 200 79

3.5 Strouhal numbers for Re = 80, 100, 200 and 300 79

3.6 Summary results for moving cylinder at Re = 40, compared against stationary cylinder at steady state. 89

Chapter 1

Introduction

In this thesis, 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. 1-1. 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

$$\boldsymbol{u}_t + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p = \mu \triangle \boldsymbol{u} + \boldsymbol{F} \tag{1.1}$$

$$\nabla \cdot \boldsymbol{u} = 0 \tag{1.2}$$

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

$$\boldsymbol{F}(\boldsymbol{x}, t) = \int_{\Gamma(t)} \boldsymbol{f}(s, t)\delta(\boldsymbol{x} - \boldsymbol{X}(s, t))ds \ , \tag{1.3}$$

where $\boldsymbol{X}(s,t)$ is the arc-length parametrization of $\Gamma(t)$, s is the arc-length, $\boldsymbol{x} = (x,y)$ is the 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

$$\frac{\partial}{\partial t}\boldsymbol{X}(s,t) = \boldsymbol{u}\left(\boldsymbol{X},t\right) = \int_{\Omega} \boldsymbol{u}(\boldsymbol{x},t)\delta(\boldsymbol{x}-\boldsymbol{X}(s,t))d\boldsymbol{x}\ . \tag{1.4}$$

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

1.1 Immersed Boundary Method and Immersed Interface Method

Methods utilizing a Cartesian grid for solving interface problems or problems with 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 [35, 36] falls into the first group. The immersed boundary method introduced by Peskin [45] belongs to the second group.

1.1.1 Immersed Boundary Method

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 [44]. The original method has been developed further and has been applied

to many biological problems including platelet aggregation [22, 23, 58], the deformation of red blood cell in a shear flow [19], the swimming of bacterial organisms and others [18, 20]. This method has also been applied to handle problems with rigid boundaries [30, 51]. More details on the immersed boundary method can be found in [45] and the references therein.

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,

$$\boldsymbol{F}(\boldsymbol{x}(i,j),t) = \sum_{k=1}^{N} \boldsymbol{f}_k(t) D_h(\boldsymbol{x}(i,j) - \boldsymbol{X}_k(t)) \triangle s \ , \tag{1.5}$$

where $\boldsymbol{f}_k(t)$ is the force density at the control point $\boldsymbol{X}_k$, $\boldsymbol{x}(i,j)$ and $\boldsymbol{F}(\boldsymbol{x}(i,j))$ are the coordinate of grid point (i,j) and the force at that point, respectively. $D_h(\boldsymbol{x})$ is a two-dimensional discrete delta function,

$$D_h(\boldsymbol{x}) = \delta_h(x)\delta_h(y), \tag{1.6}$$

where $\delta_h(r)$ is a one-dimensional discrete delta function. A typical example of $\delta_h(r)$ is

$$\delta_h(r) = \begin{cases} \dfrac{1}{4h}[1 + cos(\dfrac{\pi r}{2h})], & |r| \leq 2h \\ 0, & \text{otherwise.} \end{cases} \tag{1.7}$$

where h is the grid size.

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 to find the velocity at the control points, $\boldsymbol{U}_k^{n+1}$. Finally, the position of the interface is advanced as,

$$\boldsymbol{X}_k^{n+1} = \boldsymbol{X}_k^n + \triangle t \boldsymbol{U}_k^{n+1} \ . \tag{1.8}$$

Figure 1-2 illustrates the process of spreading the force density at a control point to the nearby Cartesian grid points and the process of interpolating the velocity at the

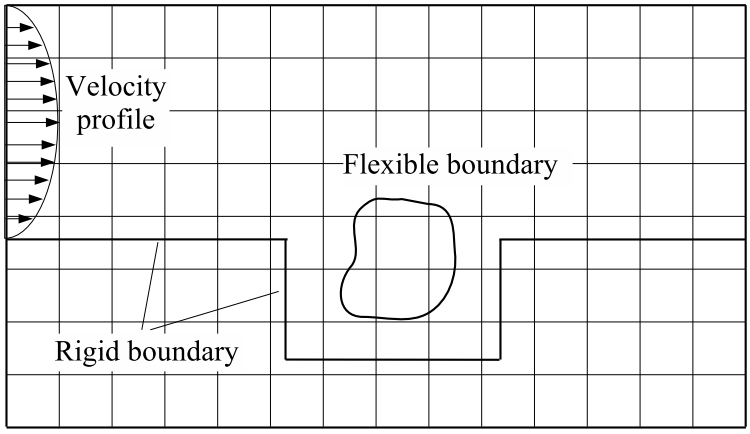


Figure 1-1: A typical domain in which the Navier-Stokes equations are solved. The flexible interface and the rigid boundary are immersed in a uniform Cartesian grid.

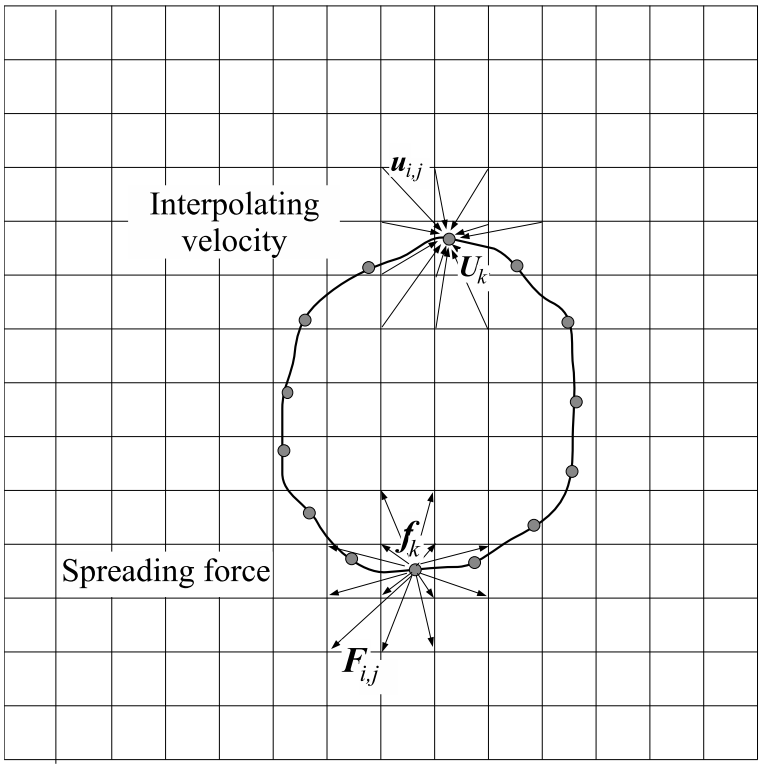


Figure 1-2: Spreading singular force at a control point to nearby grid points and interpolating velocity at another control point.

control point from the underlying velocity field. 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.

1.1.2 Immersed Interface Method

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 of 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 [35] for solving elliptic equations, and later extended to Stokes flow with elastic boundaries or surface tension [36]. The method was developed further for the Navier-Stokes equations in Li et al. [38], Lee [34] and Le et al. [31] for problems with flexible boundaries. The method was also used by Calhoun [13] and Li et al. [39] for solving the two-dimensional streamfunction-vorticity equations on irregular domains. Other immersed interface methods have

been developed for solving different PDEs, see [37, 49, 60, 61] for examples.
A contribution of this thesis 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 finite element method can handle both flexible and rigid boundaries but a mesh generation which is computationally expensive needs to be done many times to adapt to complex geometry changes.
In the next section, we briefly describe some other numerical methods for solving the viscous incompressible Navier-Stokes equations with immersed interfaces.

1.2 Existing methods for solving the Navier-Stokes equations with immersed interfaces

There are other methods for solving the Navier-Stokes equations with immersed boundaries. Each method has its own advantages and drawbacks. Below, a brief summary of the finite element methods, boundary integral methods and the ghost fluid methods are given to compare with the fixed Cartesian grid methods.

1.2.1 Finite Element Method

Finite element methods (FEMs) have been used extensively by researchers for solving viscous flow problems in irregular regions and fluid-structure interactions. Recent developments of the FEMs for general fluid flows with structure interactions can be found in [5]. The main advantage of the FEM is the capability of handling very complex geometries by using adaptive unstructured meshes. Adaptive unstructured mesh procedures can be used to refine the elements near the immersed boundaries where the solutions need to be resolved. However, time-dependent mesh generation, which is computationally expensive, cannot be avoided for moving boundary problems. This

makes the fixed Cartesian grid methods attractive for problems with moving immersed boundaries. Recently, the Extended Immersed Boundary Method (EIBM) using FEM has been developed by Wang and Liu [59] to handle submerged elastic bodies. As in Peskin's IB method, in EIBM, fluid-structure interaction problems are solved by using independent grids for the submerged solid and the surrounding fluid. Based on this method, the Immersed Finite Element Method (IFEM) has been developed by Zhang et al. [64]. In this method, the background fluid mesh does not have to follow the motion of the flexible fluid-structure interfaces and thus it is possible to assign sufficient elements within the region around which the immersed boundaries occupy and move.

1.2.2 Boundary Integral Method

Although the boundary integral method cannot be applied to the Navier-Stokes equations directly, several researchers have attempted to use it to solve for the linearized Navier-Stokes equations [1, 7, 26]. The results reported in these articles show that boundary integral formulations can be used to couple the governing equations for the fluid and immersed boundaries. Other advantages of the boundary integral method include the capability of handling complex geometries and the use of fast solution methods such as Fast Multipole Methods [27]. In [7, 42], in order to account for the jumps in the solutions across the interface, the authors use Taylor series expansion to express the jumps as source terms. This idea appears to be very similar to the idea of the immersed interface method [35]. The main disadvantage of the boundary integral methods is the inability to handle non-linear equations. From the numerical perspective the resulting integrals often become nearly singular which makes them difficult to evaluate accurately for points close to the boundary.

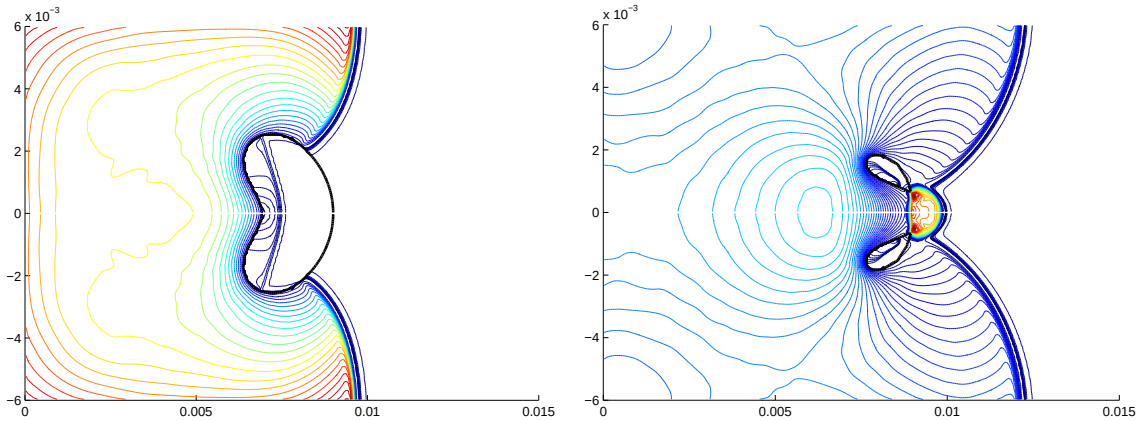
1.2.3 Ghost Fluid Method

The Ghost Fluid Method (GFM) was developed by Fedqiw and Aslam [21] to capture the boundary conditions at a contact discontinuity in the inviscid compressible

Euler equations. An extension of the GFM was also developed to solve multiphase incompressible flows [28, 43]. The GFM used a level set function to implicitly represent the interface which separates domains for the two separate fluids. The GFM captures the appropriate jump conditions at the interface by constructing ghost fluid properties and nodes based on the level set function. The GFM defines a ghost cell at every point in the computational domain. In this way, each grid point will contain the mass, momentum, and energy for the real fluid that exists at that point and a ghost mass, momentum and energy for the other fluid that does not really exist at that point. Once the ghost cells are defined, standard methods can be used to update the Euler equations at every grid point for both fluids without special concern for the interface. Since both fluids are solved for at every grid point, only the solutions of the appropriate fluid are chosen based on the sign of the level set function. In fact, the entire method relies on the ability to produce ghost cells that satisfy the appropriate boundary conditions for the governing equations. We have applied this method to simulate the shock-induced collapse of a cylindrical air cavity in water and the interaction of a planar shock wave in air with a water column. The results in Fig. 1-3 and Fig. 1-4 show that the GFM is a good method for solving multiphase compressible and incompressible flows. However, this method is only first order accurate. The level set representation of the contact discontinuity has the powerful advantage of automatically handling topological changes, but it does not appear to be adequate to represent certain types of material interfaces such as elastic membranes.

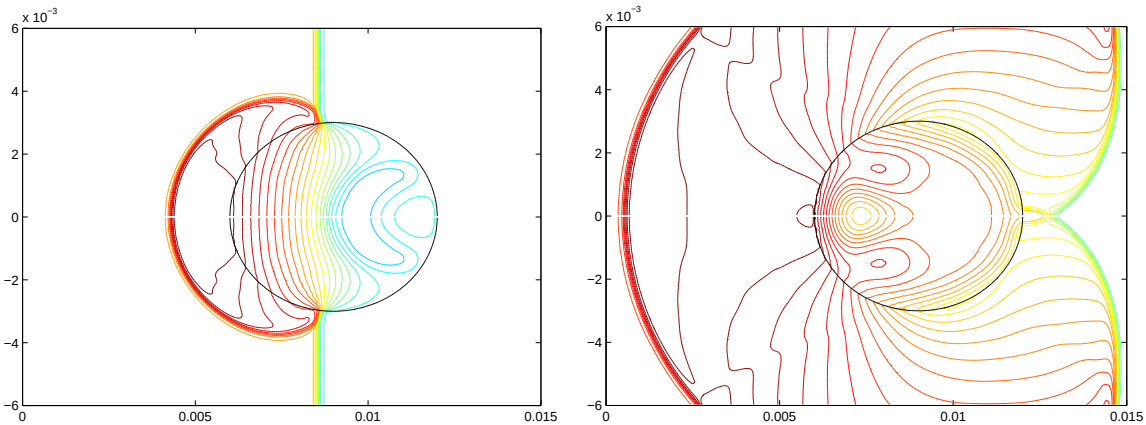
1.3 Outline of this thesis

This thesis is organized as follows. In Chapter 2, we describe an algorithm for solving the Navier-Stokes equations with deformable interfaces. We present the relations that must be satisfied along the immersed boundary between the singular force and the jumps in the velocity and pressure and their derivatives. We describe the generalized finite difference approximations to the solution derivatives, which incorporate solution jumps. In our algorithm, a projection method is employed for the discretization of



(a) (b)

Figure 1-3: Shock-induced collapse of a cylindrical air cavity in water. 1-3(a) Pressure field at time t = 2.36 μs, 1-3(b) Pressure field at time t = 3.34 μs.



(a) (b)

Figure 1-4: Interaction of a planar shock wave in air with a water column. Air is treated as compressible fluid and water is treated as incompressible fluid. Shock Mach speed M = 1.3. 1-4(a) Pressure field at time t = 6.42 μs, 1-4(b) Pressure field at time t = 20.2 μs.

the Navier-Stokes equations. Several versions of the projection method are reviewed in this chapter. In Chapter 3, we introduce an algorithm for solving the Navier-Stokes equations with rigid boundaries. We describe in detail how to impose the boundary conditions at the rigid boundaries immersed in the fluid flow. In Chapter 4, we combine the algorithms described in Chapter 2 and Chapter 3 to present an algorithm which is able to handle general viscous flow problems with deformable interfaces and rigid boundaries. Several examples are shown in each chapter to demonstrate the capability of our algorithms. Finally, some conclusions and suggestions for future work, are given in Chapter 5.

Chapter 2

An Immersed Interface Method for the Navier-Stokes equations with deformable interfaces

In this chapter, we consider flexible interfaces which are governed by either surface tension or elastic forces. The immersed interface method (IIM) that we present in this chapter is similar to the immersed interface method for the Navier-Stokes equations in [38] and [34], but also incorporates some differences. In [38], the level set method is used to represent the interface. This has the advantage of simplifying the algorithm but does not appear to be adequate to represent certain types of interfaces such as elastic membranes. In [34], the interface is tracked explicitly in a Lagrangian manner, the singular force $\boldsymbol{F}$ is split into components tangential and normal to the interface. The normal component is then incorporated into jump conditions for pressure across the interface. The tangential component is spread to the nearby Cartesian grid points using the discrete delta function as in the immersed boundary method [45]. Spreading the tangential force to the nearby Cartesian grid points has the effect of smoothing out the jumps in the derivatives of pressure and the jumps in the derivatives of velocity. These jumps are clearly not smooth in the case of elastic membrane problems. Our implementation of the IIM uses a set of control points to represent the interface and incorporates the entire singular force into jump conditions for pressure and the

derivatives of pressure and velocity. In this way, our algorithm can successfully capture all the jumps in the solutions and their derivatives. We also discuss the case when the interface crosses a grid point. This is important because grid crossing happens repeatedly when considering deformable interfaces and because jump conditions in time need to be taken into account to avoid errors which could accumulate in time. Our approach also includes the jump conditions for the second derivatives of pressure which are not used in [38] and [34] to improve the accuracy of the method.

2.1 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

$$[\boldsymbol{u}] = \boldsymbol{0}, \qquad [\mu \boldsymbol{u}_\xi] = -f_2 \boldsymbol{\tau}, \qquad [\boldsymbol{u}_\eta] = \boldsymbol{0} \tag{2.1}$$

$$[p] = f_1, \qquad [p_\xi] = \frac{\partial f_2}{\partial s}, \qquad [p_\eta] = \frac{\partial f_1}{\partial s} \ . \tag{2.2}$$

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 2-1 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,

$$\begin{aligned} &[\mu \boldsymbol{u}_{\eta\eta}] = \kappa f_2 \boldsymbol{\tau}, \qquad [\mu \boldsymbol{u}_{\xi\eta}] = -\frac{\partial f_2}{\partial \eta} \boldsymbol{\tau} - \kappa f_2 \boldsymbol{n}, \\ &[\mu \boldsymbol{u}_{\xi\xi}] = -\left[\mu \boldsymbol{u}_{\eta\eta}\right] + [p_\xi] \boldsymbol{n} + [p_\eta] \boldsymbol{\tau} + [\boldsymbol{u}_\xi] \boldsymbol{u} \cdot \boldsymbol{n} \end{aligned} \tag{2.3}$$

$$
\begin{aligned}
&[p_{\eta\eta}] = \frac{\partial^2 f_1}{\partial \eta^2} - \kappa[p_\xi], \quad [p_{\xi\eta}] = \frac{\partial^2 f_2}{\partial \eta^2} + \kappa[p_\eta], \\
&[p_{\xi\xi}] = -\left[\nabla \cdot (\boldsymbol{u} \cdot \nabla \boldsymbol{u})\right] - [p_{\eta\eta}] \ .
\end{aligned} \tag{2.4}
$$

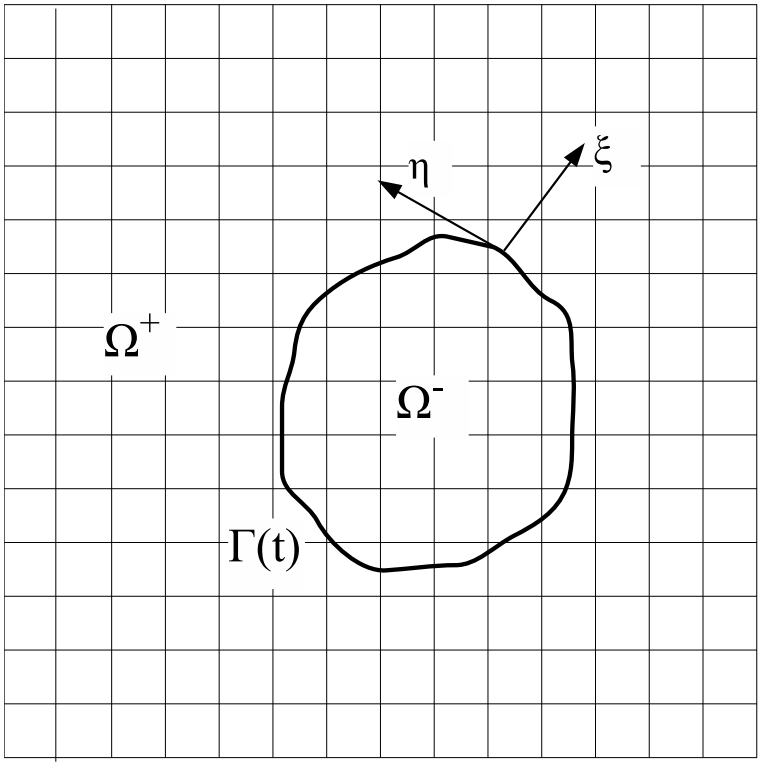


Figure 2-1: A typical domain with an immersed flexible boundary. The local coordinate system (ξ, η). The domain Ω^+ and Ω^- are divided by a closed curve $\Gamma(t)$.

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 detailed proof for expressions (2.1)-(2.4) can be found in [35, 36, 38, 33] and a sketch of the proof can be found in Appendix A. We note that from expressions (2.1)-(2.4) 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,

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

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.

2.2 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 $v(x)$:

$$v_x(x_i) = \frac{v_{i+1} - v_{i-1}}{2h} - \frac{1}{2h}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}[v^{(m)}] + O(h^2) \tag{2.5}$$

$$v_x(x_{i+1}) = \frac{v_{i+2} - v_i}{2h} - \frac{1}{2h}\sum_{m=0}^{2}\frac{(h^-)^m}{m!}[v^{(m)}] + O(h^2) \tag{2.6}$$

$$v_{xx}(x_i) = \frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} - \frac{1}{h^2}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}[v^{(m)}] + O(h) \tag{2.7}$$

$$v_{xx}(x_{i+1}) = \frac{v_{i+2} - 2v_{i+1} + v_i}{h^2} + \frac{1}{h^2}\sum_{m=0}^{2}\frac{(h^-)^m}{m!}[v^{(m)}] + O(h) \tag{2.8}$$

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

$$[v^{(m)}]_\alpha = \lim_{x\to\alpha, x\in\Omega^+} v^{(m)}(x) - \lim_{x\to\alpha, x\in\Omega^-} v^{(m)}(x) \tag{2.9}$$

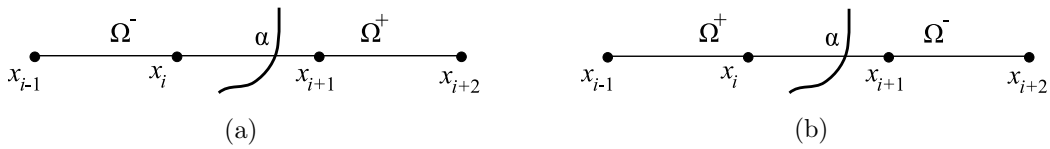


Figure 2-2: The interface cuts a grid line between two grid points at $x = \alpha$. 2-2(a) $x_i \in \Omega^-$ and $x_{i+1} \in \Omega^+$. 2-2(b) $x_i \in \Omega^+$ and $x_{i+1} \in \Omega^-$.

in short, $[\cdot] = [\cdot]_\alpha$, and $v^{(0)} = v$. See Weigmann and Bube [61] for more details on the proof. Figure 2-2 shows the interface cutting a grid line between two grid points x_i and x_{i+1}.

Note that if the interface cuts a grid line between two grid points $x_i \in \Omega^+$ and $x_{i+1} \in \Omega^-$ as shown in Fig. 2-2(b), the above expressions need to be modified as,

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

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

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

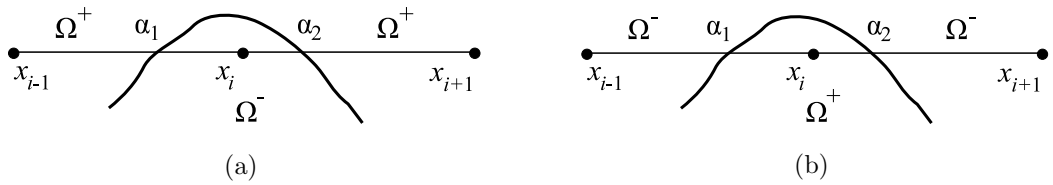
$$v_{xx}(x_{i+1}) = \frac{v_{i+2} - 2v_{i+1} + v_i}{h^2} - \frac{1}{h^2}\sum_{m=0}^{2}\frac{(h^-)^m}{m!}[v^{(m)}] + O(h)$$

The above expressions can be generalized for the case in which we have two interfaces cutting the grid line between x_{i-1} and x_{i+1}. In such cases we need to consider two possibilities:

a) when $x_{i-1} \le \alpha_1 < x_i \le \alpha_2 < x_{i+1}$, if $x_{i-1} \in \Omega^+$, $x_i \in \Omega^-$, $x_{i+1} \in \Omega^+$ as shown in Fig. 2-3(a), we have

$$v_x(x_i) = \frac{v_{i+1} - v_{i-1}}{2h} + \frac{1}{2h}\sum_{m=0}^{2}\frac{(h_1^-)^m[v^{(m)}]_{\alpha_1} - (h_2^+)^m[v^{(m)}]_{\alpha_2}}{m!} + O(h^2) \tag{2.10}$$

$$v_{xx}(x_i) = \frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} - \frac{1}{h^2}\sum_{m=0}^{2}\frac{(h_1^-)^m[v^{(m)}]_{\alpha_1} + (h_2^+)^m[v^{(m)}]_{\alpha_2}}{m!} + O(h) \tag{2.11}$$



(a) (b)

Figure 2-3: Two interfaces cut a grid line between two grid points x_{i-1} and x_{i+1}. 2-3(a) $x_{i-1} \in \Omega^+$, $x_i \in \Omega^-$ and $x_{i+1} \in \Omega^+$. 2-3(b) $x_{i-1} \in \Omega^-$, $x_i \in \Omega^+$ and $x_{i+1} \in \Omega^-$.

(a) (b)

Figure 2-4: Two interfaces cut a grid line between two grid points x_i and x_{i+1}. 2-4(a) $x_i \in \Omega^+$ and $x_{i+1} \in \Omega^+$. 2-4(b) $x_i \in \Omega^-$ and $x_{i+1} \in \Omega^-$.

and if $x_{i-1} \in \Omega^-$, $x_i \in \Omega^+$, $x_{i+1} \in \Omega^-$ as shown in Fig. 2-3(b), we have

$$v_x(x_i) = \frac{v_{i+1} - v_{i-1}}{2h} - \frac{1}{2h}\sum_{m=0}^{2} \frac{(h_1^-)^m [v^{(m)}]_{\alpha_1} - (h_2^+)^m [v^{(m)}]_{\alpha_2}}{m!} + O(h^2)$$

$$v_{xx}(x_i) = \frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} + \frac{1}{h^2}\sum_{m=0}^{2} \frac{(h_1^-)^m [v^{(m)}]_{\alpha_1} + (h_2^+)^m [v^{(m)}]_{\alpha_2}}{m!} + O(h)$$

where $h_1^+ = x_i - \alpha_1$, $h_1^- = x_{i-1} - \alpha_1$, $h_2^+ = x_{i+1} - \alpha_2$, $h_2^- = x_i - \alpha_2$.

b) when $x_i \le \alpha_1 < \alpha_2 < x_{i+1}$, if $x_i \in \Omega^+$, $x_{i+1} \in \Omega^+$ as shown in Fig. 2-4(a), we have

$$v_x(x_i) = \frac{v_{i+1} - v_{i-1}}{2h} + \frac{1}{2h}\sum_{m=0}^{2} \frac{(h_1^+)^m [v^{(m)}]_{\alpha_1} - (h_2^+)^m [v^{(m)}]_{\alpha_2}}{m!} + O(h^2) \tag{2.12}$$

$$v_x(x_{i+1}) = \frac{v_{i+2} - v_i}{2h} + \frac{1}{2h}\sum_{m=0}^{2} \frac{(h_1^-)^m [v^{(m)}]_{\alpha_1} - (h_2^-)^m [v^{(m)}]_{\alpha_2}}{m!} + O(h^2) \tag{2.13}$$

$$v_{xx}(x_i) = \frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} + \frac{1}{h^2}\sum_{m=0}^{2} \frac{(h_1^+)^m [v^{(m)}]_{\alpha_1} - (h_2^+)^m [v^{(m)}]_{\alpha_2}}{m!} + O(h) \tag{2.14}$$

$$v_{xx}(x_{i+1}) = \frac{v_{i+2} - 2v_{i+1} + v_i}{h^2} - \frac{1}{h^2}\sum_{m=0}^{2} \frac{(h_1^-)^m [v^{(m)}]_{\alpha_1} - (h_2^-)^m [v^{(m)}]_{\alpha_2}}{m!} + O(h) \tag{2.15}$$

and if $x_i \in \Omega^-$, $x_{i+1} \in \Omega^-$ as shown in Fig. 2-4(b), we have

$$v_x(x_i) = \frac{v_{i+1} - v_{i-1}}{2h} - \frac{1}{2h}\sum_{m=0}^{2} \frac{(h_1^+)^m[v^{(m)}]_{\alpha_1} - (h_2^+)^m[v^{(m)}]_{\alpha_2}}{m!} + O(h^2)$$

$$v_x(x_{i+1}) = \frac{v_{i+2} - v_i}{2h} - \frac{1}{2h}\sum_{m=0}^{2} \frac{(h_1^-)^m[v^{(m)}]_{\alpha_1} - (h_2^-)^m[v^{(m)}]_{\alpha_2}}{m!} + O(h^2)$$

$$v_{xx}(x_i) = \frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} - \frac{1}{h^2}\sum_{m=0}^{2} \frac{(h_1^+)^m[v^{(m)}]_{\alpha_1} - (h_2^+)^m[v^{(m)}]_{\alpha_2}}{m!} + O(h)$$

$$v_{xx}(x_{i+1}) = \frac{v_{i+2} - 2v_{i+1} + v_i}{h^2} + \frac{1}{h^2}\sum_{m=0}^{2} \frac{(h_1^-)^m[v^{(m)}]_{\alpha_1} - (h_2^-)^m[v^{(m)}]_{\alpha_2}}{m!} + O(h)$$

where $h_1^+ = x_{i+1} - \alpha_1$, $h_1^- = x_i - \alpha_1$, $h_2^+ = x_{i+1} - \alpha_2$, $h_2^- = x_i - \alpha_2$. Fig. 2-3 and Fig.2-4 show the finite difference stencils from which the generalized finite difference formulas are derived for the case in which two interfaces cut a grid line between two grid points x_{i-1} and x_{i+1}.

Expressions involving three or more interface crossings could also be derived but we have not found it necessary for our applications. 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, $t^{n-1} \leq \tau < t^{n+1}$, we have,

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

$$v(t^{n+1/2}) = \frac{1}{2}(v^n + v^{n+1}) + \frac{1}{2}[v]_\tau + O(\triangle t) \tag{2.16}$$

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

$$v(t^{n+1/2}) = \frac{1}{2}(v^n + v^{n+1}) - \frac{1}{2}[v]_\tau + O(\triangle t) \tag{2.17}$$

and,

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

$$v(t^{n+1/2}) = \frac{3}{2}v^n - \frac{1}{2}v^{n-1} - \frac{1}{2}[v]_\tau + O(\triangle t) \tag{2.18}$$

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

$$v(t^{n+1/2}) = \frac{3}{2}v^n - \frac{1}{2}v^{n-1} + [v]_\tau + O(\triangle t) \ . \tag{2.19}$$

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

$$[v(t)]_\tau = \lim_{t \to \tau^+} v(t) - \lim_{t \to \tau^-} v(t) \ . \tag{2.20}$$

It is easy to see that $[v]_\alpha = \pm[v]_\tau$, where $[.]_\alpha$ denotes spatial jump as defined in (2.9) and the sign depends on the motion of the interface. In particular, 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^-.

2.3 Projection methods

In this section, we describe a second order projection method for the incompressible Navier-Stokes equations with immersed interfaces. But first of all, we briefly describe some traditional versions of the projection methods for non interface problems. Our purpose is not to review the literature but to describe some features of these methods from which our projection method is developed. In a 2-dimensional bounded domain Ω, we consider the incompressible Navier-Stokes equations, written as

$$\boldsymbol{u}_t + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p = \mu \triangle \boldsymbol{u} \tag{2.21}$$

$$\nabla \cdot \boldsymbol{u} = 0 \tag{2.22}$$

with Dirichlet boundary conditions,

$$\boldsymbol{u}|_{\partial\Omega} = \boldsymbol{u}_b \ . \tag{2.23}$$

The extension to incorporate Neumann boundary conditions can be done without complications.

Projection methods have been used extensively for viscous incompressible flows since the first projection method [14] appeared three and a half decades ago. Since then, several versions of the projection methods have been proposed [4, 6, 10, 29, 41, 53]. In these methods, a semi-implicit time discretization of (2.21) and (2.22) is formed as,

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\triangle t} + [(\boldsymbol{u} \cdot \nabla)\boldsymbol{u}]^{n+1/2} + \nabla p^{n+1/2} = \frac{\mu}{2}(\nabla^2 \boldsymbol{u}^{n+1} + \nabla^2 \boldsymbol{u}^n) \tag{2.24}$$

$$\nabla \cdot \boldsymbol{u}^{n+1} = 0 \tag{2.25}$$

with boundary condition

$$\boldsymbol{u}^{n+1}|_{\partial\Omega} = \boldsymbol{u}_b^{n+1} \ . \tag{2.26}$$

The notation $\boldsymbol{u}^{n+1}$ is used to represent the approximation to $\boldsymbol{u}(t^{n+1})$. These equations are solved numerically in three steps. In the first step, the momentum equation (2.24) with an approximate pressure field is advanced to compute an intermediate velocity $\boldsymbol{u}^*$ which does not, in general, satisfy the continuity equation (2.25). In the second step, a Poisson equation is then solved to enforce the divergence free constraint (2.25). In the third step, the pressure field $p^{n+1/2}$ and the velocity field $\boldsymbol{u}^{n+1}$ are updated. Existing projection methods of this kind can be placed into two categories: pressure-free projection methods and pressure-increment projection methods [10]. The pressure-free projection methods set the pressure field in the momentum equations to zero and compute the new pressure by solving a Poisson equation. The pressure-increment projection methods use the pressure obtained at the previous time step as an approximate pressure field in the momentum equations and then solve a Poisson equation to obtain a pressure correction. Both of these projection methods are second order accuracy for velocity and pressure provided all the spatial derivatives are approximated

to second order accuracy and appropriate intermediate velocity boundary conditions are applied.

2.3.1 Pressure-free projection method

This method was originally proposed by Kim and Moin [29] and has since been extensively used for solving viscous incompressible flows. In this method, the momentum equations are first solved for the intermediate velocity without the pressure gradient term,

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\triangle t} = -\left(\boldsymbol{u} \cdot \nabla \boldsymbol{u}\right)^{n+\frac{1}{2}} + \mu \nabla^2 \boldsymbol{u}^{n+\frac{1}{2}} \ , \tag{2.27}$$

where the advective term is extrapolated using Adams-Bashforth formula,

$$\left(\boldsymbol{u}.\nabla \boldsymbol{u}\right)^{n+\frac{1}{2}} = \frac{3}{2}\left(\boldsymbol{u} \cdot \nabla \boldsymbol{u}\right)^{n} - \frac{1}{2}\left(\boldsymbol{u} \cdot \nabla \boldsymbol{u}\right)^{n-1} \ , \tag{2.28}$$

and the diffusion term is approximated implicitly using Crank-Nicolson formula,

$$\nabla^2 \boldsymbol{u}^{n+1/2} = \frac{1}{2}(\nabla^2 \boldsymbol{u}^* + \nabla^2 \boldsymbol{u}^n) \ . \tag{2.29}$$

A Poisson equation is then solved for a new variable ϕ which is used to enforce the divergence-free constraint on the velocity and update the pressure field at the next time step,

$$\nabla^2 \phi^{n+1} = \frac{\nabla \cdot \boldsymbol{u}^*}{\triangle t} \ , \tag{2.30}$$

$$\boldsymbol{n} \cdot \nabla \phi^{n+1}|_{\partial\Omega} = \frac{\boldsymbol{n} \cdot (\boldsymbol{u}^* - \boldsymbol{u}^{n+1})|_{\partial\Omega}}{\triangle t} \ . \tag{2.31}$$

The velocity and pressure fields are then updated as,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \Delta t \nabla \phi^{n+1} \ , \tag{2.32}$$

$$p^{n+1/2} = \phi^{n+1} - \frac{\mu}{2}\left(\nabla \cdot \boldsymbol{u}^*\right) \ . \tag{2.33}$$

The Poisson equation (2.30) is in fact obtained by substituting (2.32) into the momentum equation (2.27) and dropping the high order terms. The above scheme is

defined on the standard marker-and-cell (MAC) staggered grid using finite volume technique with standard central differences for the viscous term and pressure gradient term. A high order upwind scheme is used for the discretization of the nonlinear advective term. Kim and Moin noted that the boundary conditions for the intermediate velocity fields in fractional step methods are generally a source of ambiguity. At each time step, the boundary conditions for $\boldsymbol{u}^*$ are unknown, only the boundary conditions for $\boldsymbol{u}^{n+1}$ are given. And hence, they suggested using (2.32) to define the boundary conditions for the intermediate velocity $\boldsymbol{u}^*$. Since $\nabla\phi^{n+1}$ is not known when solving for $\boldsymbol{u}^*$, $\nabla\phi^n$ is suggested to replace $\nabla\phi^{n+1}$, i.e. $\boldsymbol{u}^*|_{\partial\Omega} = \boldsymbol{u}^{n+1}|_{\partial\Omega} + \Delta t\nabla\phi^n|_{\partial\Omega}$. And this appears to be adequate to obtain second order accuracy for velocities. Brown et al. [10] comment that the absence of the pressure gradient term in (2.27) could be considered appealing since it prohibits errors in pressure gradient from contributing to errors in the momentum equations.

2.3.2 Pressure-increment projection method

One of the well-known pressure-increment projection methods is that of Bell et al. [6] which has been applied for various types of physical problems such as incompressible viscous flows and reacting flows. In this method, the momentum equations are solved for the intermediate velocity $\boldsymbol{u}^*$ with $\nabla p^{n-\frac{1}{2}}$ as the approximation to the pressure gradient term,

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\triangle t} = -\left(\boldsymbol{u}\cdot\nabla\boldsymbol{u}\right)^{n+\frac{1}{2}} - \nabla p^{n-\frac{1}{2}} + \mu\nabla^2\boldsymbol{u}^{n+\frac{1}{2}} \tag{2.34}$$

$$\boldsymbol{u}^*|_{\partial\Omega} = \boldsymbol{u}_b^{n+1} \tag{2.35}$$

where the advective term and the diffusion term are approximated using the Adams-Bashforth formula and the Crank-Nicolson formula as in (2.28) and (2.29), respectively.

A pressure-increment is then computed to enforce the continuity equation (2.25) and to correct the velocity fields at the next time step. In order to do that, one first solves

for a scalar variable ϕ satisfying

$$\nabla^2 \phi^{n+1} = \frac{\nabla \cdot \boldsymbol{u}^*}{\triangle t} \ , \tag{2.36}$$

$$\boldsymbol{n} \cdot \nabla \phi^{n+1}|_{\partial\Omega} = 0 \ , \tag{2.37}$$

and then, the velocity and pressure fields are updated as,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \Delta t \nabla \phi^{n+1} \ , \tag{2.38}$$

$$p^{n+1/2} = p^{n-1/2} + \phi^{n+1} - \frac{\mu}{2} \left(\nabla \cdot \boldsymbol{u}^* \right) \ . \tag{2.39}$$

In fact, in [6] the last term on the right-hand side of (2.39) is not included. This omission results in lowering the accuracy of pressure and an inaccurate pressure gradient at the boundary [10]. Therefore, Brown et al. [10] suggested using (2.39) for updating the pressure to improve the order of accuracy of the pressure. The boundary condition (2.37) for ϕ^{n+1} is actually derived from (2.38) and the boundary condition for the intermediate normal velocity. We note that the boundary condition (2.35) for the intermediate tangential velocity is not consistent with (2.38). However, if $\boldsymbol{u}^*$ is a good approximation to $\boldsymbol{u}^{n+1}$, then the boundary condition (2.35) should suffice to get second order accuracy. More accurate boundary conditions for the intermediate velocity are suggested in [10] as,

$$\boldsymbol{n} \cdot \boldsymbol{u}^*|_{\partial\Omega} = \boldsymbol{n} \cdot \boldsymbol{u}_b^{n+1} \tag{2.40}$$

$$\boldsymbol{\tau} \cdot \boldsymbol{u}^*|_{\partial\Omega} = \boldsymbol{\tau} \cdot (\boldsymbol{u}_b^{n+1} + \Delta t \nabla \phi^n|_{\partial\Omega}) \tag{2.41}$$

In [10], the above scheme is defined on a uniform non-staggered grid. All the spatial derivatives are approximated using standard central differences. The pressure-increment projection method can achieve second order accuracy for both velocity and pressure provided that appropriate boundary conditions for $\boldsymbol{u}^*$ and the consistent equation (2.39) for updating $p^{n+\frac{1}{2}}$ are used.

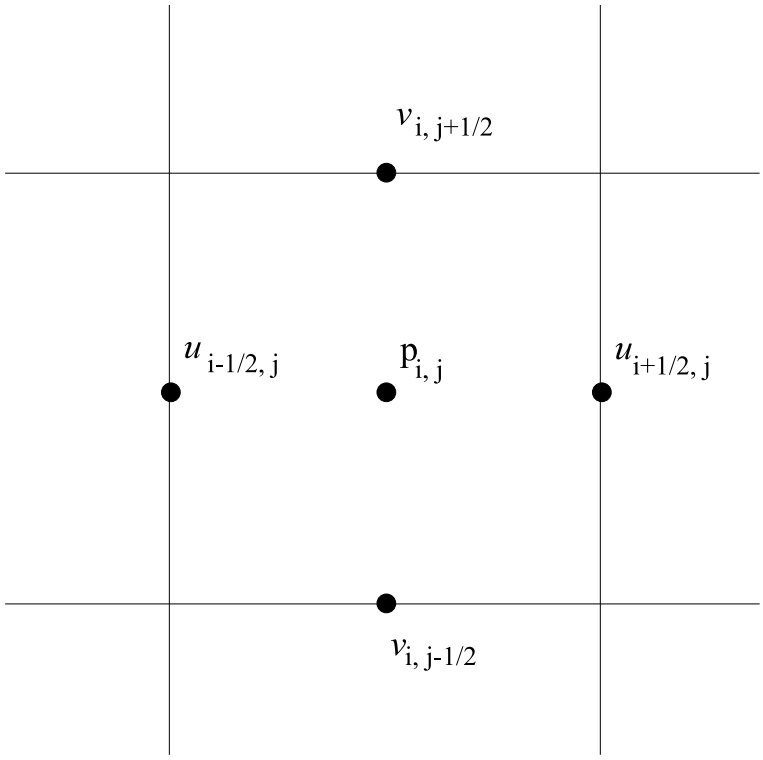


Figure 2-5: The MAC staggered grid in two dimensions.

2.3.3 Projection method for the immersed interface method

For viscous flow problems with immersed interfaces, 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. [10]. 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. [29]. The ENO third-order upwind scheme is used for the advective terms (Shu and Osher [50]). Figure 2-5 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. On the other hand, the implementation of the MAC scheme on the non-staggered grids introduces some complications. For

instance, some of the velocity components are not defined on the boundaries of the domain. The pressure-increment projection procedure described in section 2.3.2 is essentially unchanged by the presence of the immersed interfaces. 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 review 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

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\triangle t} = -\overline{(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n+\frac{1}{2}}} - \overline{\nabla p^{n+\frac{1}{2}}} + \mu\overline{\nabla^2\boldsymbol{u}^{n+\frac{1}{2}}} + \boldsymbol{C}_1 \tag{2.42}$$

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

where the advective term is extrapolated using the formula,

$$\overline{(\boldsymbol{u}.\nabla\boldsymbol{u})^{n+\frac{1}{2}}} = \frac{3}{2}(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^n - \frac{1}{2}(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n-1} + \boldsymbol{C}_2 + \gamma_1[\boldsymbol{u}\cdot\nabla\boldsymbol{u}]_\tau \ , \tag{2.43}$$

the diffusion term is approximated implicitly as,

$$\overline{\nabla^2\boldsymbol{u}^{n+1/2}} = \frac{1}{2}(\nabla_h^2\boldsymbol{u}^* + \nabla_h^2\boldsymbol{u}^n) + \boldsymbol{C}_3 + \gamma_2[\nabla_h^2\boldsymbol{u}]_\tau \ , \tag{2.44}$$

and the pressure gradient is approximated simply as,

$$\overline{\nabla p^{n+\frac{1}{2}}} = G^{MAC}p^{n-\frac{1}{2}} + \boldsymbol{C}_4 + \gamma_3[\nabla p]_\tau \ . \tag{2.45}$$

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

$$\nabla_h^2 \phi^{n+1} = \frac{D^{MAC}\boldsymbol{u}^*}{\triangle t} + C_5 \ , \tag{2.46}$$

with boundary condition

$$\boldsymbol{n} \cdot \nabla \phi^{n+1}|_{\partial\Omega} = 0 \ . \tag{2.47}$$

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

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \Delta t G^{MAC}\phi^{n+1} + \boldsymbol{C}_6 \tag{2.48}$$

$$p^{n+1/2} = p^{n-1/2} + \phi^{n+1} - \frac{\mu}{2}\left(D^{MAC}\boldsymbol{u}^*\right) + C_7 \tag{2.49}$$

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 (2.18), (2.19) and the coefficient γ_2 is determined from expressions (2.16), (2.17). 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, in such cases, 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 and are calculated using the generalized finite difference formulas of the type introduced in section 2.2. 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 (2.42) and one Poisson equation for ϕ^{n+1} in (2.46). Since the correction terms in (2.42) and

(2.46) 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 [2] 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 2-5. 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 is adequate for our purposes.

2.3.4 Correction terms

In this section, we will show how to evaluate the correction terms $\boldsymbol{C}_i, i = 1, \ldots, 7$ generated in section 2.3.3. Let's define $C\{u\}$ as a correction term for a quantity u. For example, from (2.5) we can write

$$C\{u_x(x_i)\} = -\frac{1}{2h}\left([u] + h^+[u_x] + \frac{(h^+)^2}{2}[u_{xx}]\right) \tag{2.50}$$

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

$$\boldsymbol{C}_1 = -C\{\boldsymbol{u}_t\} \tag{2.51}$$

$$\boldsymbol{C}_2 = \frac{3}{2}C\{(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^n\} - \frac{1}{2}C\{(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n-1}\} \tag{2.52}$$

$$\boldsymbol{C}_3 = \frac{1}{2}\left(C\{\nabla^2\boldsymbol{u}^*\} + C\{\nabla^2\boldsymbol{u}^n\}\right) \tag{2.53}$$

$$\boldsymbol{C}_4 = C\{\nabla p^{n-\frac{1}{2}}\} \tag{2.54}$$

$$C_5 = \frac{C\{\nabla\cdot\boldsymbol{u}^*\}}{\triangle t} - C\{\nabla^2 p^{n+\frac{1}{2}}\} + C\{\nabla^2 p^{n-\frac{1}{2}}\} + \gamma_3[\nabla p]_\tau \tag{2.55}$$

$$\boldsymbol{C}_6 = -\triangle t\left(C\{\nabla p^{n+\frac{1}{2}}\} - C\{\nabla p^{n-\frac{1}{2}}\}\right) \tag{2.56}$$

$$C_7 = -\frac{\mu}{2}C\{\nabla\cdot\boldsymbol{u}^*\} \tag{2.57}$$

All the correction terms are included at least to first order accuracy. As explained in [35], 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 (2.51) 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, $t^n \leq \tau \leq t^{n+1}$, the correction term for $\boldsymbol{u}_t$ at this point is given by

$$C\{\boldsymbol{u}_t\} = -\frac{1}{\triangle t}\left([\boldsymbol{u}]_\tau + (t^n - \tau)[\boldsymbol{u}_t]_\tau\right) \tag{2.58}$$

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

$$C\{\boldsymbol{u}_t\} = -\frac{1}{\triangle t}\left([\boldsymbol{u}]_\tau + (t^{n+1} - \tau)[\boldsymbol{u}_t]_\tau\right) \tag{2.59}$$

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

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

$$[\boldsymbol{u}_t] = -[\boldsymbol{u}\cdot\nabla\boldsymbol{u}] = \pm[\boldsymbol{u}_t]_\tau \tag{2.60}$$

In (2.53), (2.55) and (2.57), 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 (2.53) at a point (i,j) in Fig. 2-6 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

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

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

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

Note that $[\boldsymbol{u}^*] = \mathbf{0}$ since the velocity is continuous across the interface. Similarly, we can compute other correction terms in (2.53)–(2.57).

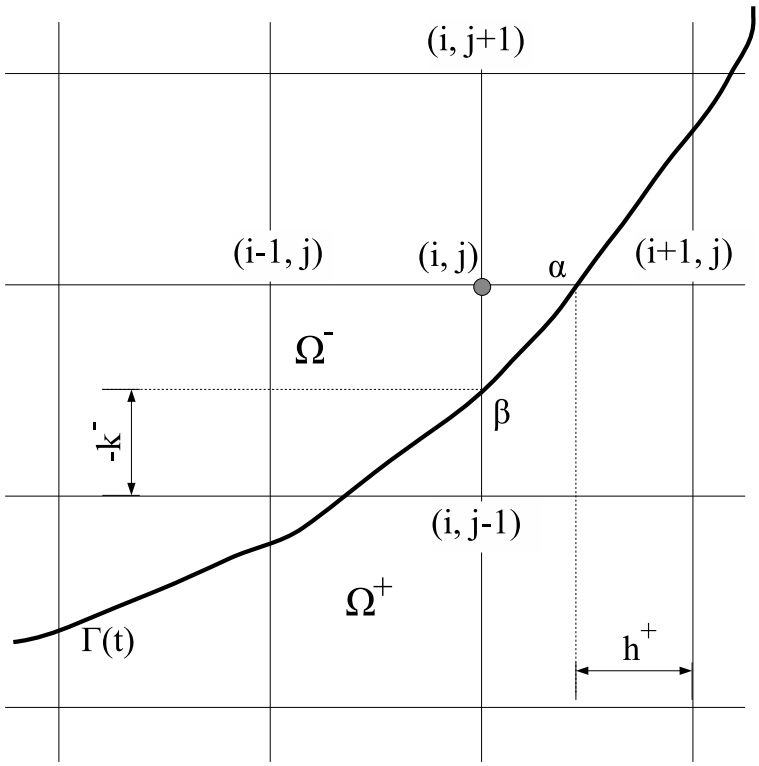


Figure 2-6: Interface and mesh geometry near the grid point (i,j).

2.4 Implicit scheme for moving interface

The location of the interface at time t^n is represented by a set of N control points $\boldsymbol{X}^n = (X^n_k, Y^n_k)$ for $k = 1, \ldots, N$. In [36], it was found that when the position of

the interface is advanced using an explicit method, the size of the timestep required for stability is very small. For this reason LeVeque et al. [36] recommend the use of an implicit scheme. Following [36], we update the position of the control points according to

$$\boldsymbol{X}^{n+1} = \boldsymbol{X}^{n} + \frac{1}{2}\triangle t \left(\boldsymbol{u}^{n}\left(\boldsymbol{X}^{n}\right) + \boldsymbol{u}^{n+1}\left(\boldsymbol{X}^{n+1}\right)\right) \tag{2.61}$$

Note that the velocity of a control point $\boldsymbol{u}\left(\boldsymbol{X}\right)$ is interpolated from the velocity at the Cartesian grid points using a modified bilinear interpolation derived in Appendix B. Equation (2.61) is implicit and couples the motion of the interface with the solution at all grid points. Therefore at each timestep, we need to solve a non-linear system of equations for the position of the control points of the form $g\left(\boldsymbol{X}^{n+1}\right) = 0$, where

$$g\left(\boldsymbol{X}\right) = \boldsymbol{X} - \boldsymbol{X}^{n} - \frac{1}{2}\triangle t \left(\boldsymbol{u}^{n}\left(\boldsymbol{X}^{n}\right) + \boldsymbol{u}^{n+1}\left(\boldsymbol{X}\right)\right) \tag{2.62}$$

Normally this non-linear system of equations is solved by a Newton's method in which the inverse Jacobian matrix is required. However, the evaluation of the Jacobian matrix

$$J\left(\boldsymbol{X}\right) = g'\left(\boldsymbol{X}\right) = \boldsymbol{I} - \frac{1}{2}\triangle t \boldsymbol{u}'\left(\boldsymbol{X}\right) \tag{2.63}$$

and its inverse is computationally expensive. Therefore a quasi-Newton method has been devised wherein the inverse Jacobian matrix J^{-1} is replaced by a suitable matrix B, which is easy to compute. The matrix B at iteration $(k+1)^{th}$, B_{k+1}, is updated from the matrix B_k at the previous iteration as follows

$$\begin{aligned} B_{k+1} = \gamma_k B_k &+ \left(1 + \gamma_k \theta_k \frac{q_k^T B_k q_k}{p_k^T q_k}\right) \frac{p_k p_k^T}{p_k^T q_k} - \gamma_k \frac{(1-\theta_k)}{q_k^T B_k q_k} B_k q_k \cdot q_k^T B_k \\ &- \frac{\gamma_k \theta_k}{p_k^T q_k}\left(p_k q_k^T B_k + B_k q_k p_k^T\right) \ , \end{aligned} \tag{2.64}$$

where

$$p_k := \boldsymbol{X}^{(k+1)} - \boldsymbol{X}^{(k)}, \qquad q_k := g^{(k+1)} - g^{(k)}$$

and the parameters $\gamma_k \geq 0$, $\theta_k \geq 0$.

The following special cases are contained in (2.64):

1. $\gamma_k \equiv 1, \theta_k \equiv 0$: the rank two method of Davidon, Fletcher and Powell (DFP method);

2. $\gamma_k \equiv 1, \theta_k \equiv 1$: the rank two method of Broyden, Fletcher, Goldfard and Shanno (BFGS method, see, for example [12]);

3. $\gamma_k = 1, \theta_k = p_k^T q_k/(p_k^T q_k - q_k^T B_k q_k)$: the symmetric, rank one method of Broyden.

Detailed discussions for these methods can be found in [52]. Practical experience indicates that the choice $\gamma_k \equiv 1, \theta_k \equiv 1$ (BFGS method) is good. It seems that, in our implementation, the BFGS method is slightly superior than the other methods. In practice, this approach requires only a few iterations as the solution at the previous timestep provides a very good initial guess for the iteration.

The BFGS algorithm can be summarized as follows:

1. Set $k := 0$, select $\boldsymbol{X}^{(0)}$ and a real positive definite matrix B_0 .

2. If $\|g^{(k)}\| < \epsilon$, stop. Else $d^{(k)} = -B_k g^{(k)}$.

3. Compute
$$\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} + \alpha_k d^{(k)} \ , \tag{2.65}$$
where in practice the constant α_k is chosen to be 1 for simplicity.

4. Update the inverse Jacobian approximation B_{k+1}, set $k := k+1$ and go to step 2. The inverse Jacobian approximation is updated as in(2.64) with $\gamma_k \equiv 1, \theta_k \equiv 1$.

At each time step the initial guess for $\boldsymbol{X}^{n+1}$ could be $\boldsymbol{X}^n$ or $2\boldsymbol{X^n} - \boldsymbol{X^{n-1}}$. The initial guess for the inverse Jacobian B_0 is the final approximate B obtained from the previous time step. At the very first time step, the initial guess for the inverse Jacobian B_0 is the identity matrix, which is a reasonable choice since $J = g(\boldsymbol{X})' = I + O(\Delta t)$.

2.5 Implementation

In this section, we describe the various steps required for the implementation of the immersed interface method for the Navier-Stokes equations. We consider the model problem used in [36, 55]. This consists of an immersed elastic band in a fluid with the same fluid properties on each side. The force strength exerted by the elastic band at $\boldsymbol{X}(s,t)$ is given as,

$$\boldsymbol{f}(s,t) = \frac{\partial}{\partial s}(T(s,t)\boldsymbol{\tau}(s,t)) \ , \tag{2.66}$$

where $T(s,t)$ is the tension of the band, defined by

$$T(s,t) = T_0 \left(\left| \frac{\partial \boldsymbol{X}(s,t)}{\partial s_0} \right| - 1 \right) \tag{2.67}$$

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

$$\boldsymbol{\tau}(s,t) = \frac{\partial \boldsymbol{X}}{\partial s} \Big/ \left| \frac{\partial \boldsymbol{X}}{\partial s} \right| \ . \tag{2.68}$$

Again, $\boldsymbol{X}(s,t)$ is the arc-length parametrization of the band and s_0 is the arc-length measured along the unstretched band. The scalar T_0 is the stiffness constant which describes the elastic property of the band.

At each time step, we use a set of control points $\boldsymbol{X}_k, k = 1, \ldots, N_b$ to represent the interface. The control points are connected by a closed curve which is a cubic spline. Based on this cubic spline, the intersections between the grid lines and the interface are determined. The jump conditions at the intersection points are then computed and used to evaluate the required correction terms. Before computing the jump conditions, the force strength at the intersection points needs to be calculated. To do so, we first compute the force strength at the control points using expression (2.66). And then, based on the force strength at the control points, a new cubic spline is determined to approximate the force strength along the interface. With this new cubic spline, the force strength and its derivatives can be calculated at the intersection points.

Having the jump conditions for pressure and velocity, we then apply the projection

method with the correction terms to advance for the velocity and pressure to the next time step. The location of the interface is also updated based on the surrounding fluid velocity. The implicit scheme for updating the location of the interface is used to increase the timestep. Since the viscous term is treated implicitly and the convection term is approximated explicitly, the time step is chosen to satisfy the CFL condition,

$$\max_{i,j}\left(\left|\frac{u_{i\pm1/2,j}}{\Delta x}\right|,\left|\frac{v_{i,j\pm1/2}}{\Delta y}\right|\right)\Delta t \leq 1\ . \tag{2.69}$$

In summary, given the location of the control points, $\boldsymbol{X}^n$, the velocity field, $\boldsymbol{u}^n$ and the pressure field $p^{n-1/2}$, the process of computing the new velocity $\boldsymbol{u}^{n+1}$, pressure field $p^{n+1/2}$ and the location of the control points $\boldsymbol{X}^{n+1}$ can be summarized as follows:

Step 1: Set $k := 0$ and make an initial guess for $\boldsymbol{X}^{n+1}$, i.e. $\boldsymbol{X}^{(0)}$ as

$$\boldsymbol{X}^{(0)} = 2\boldsymbol{X^n} - \boldsymbol{X^{n-1}}$$

Step 2:

- Compute the intersection points between the interface and the grid lines.
- Compute the force strength at the control points using expression (2.66). Interpolate the force strength using cubic splines.
- Compute the jump conditions for pressure and velocity at the intersection points.

Step 3:

- Compute the appropriate correction terms for spatial derivatives and temporal derivatives as described in section 2.3.4.
- Compute the intermediate velocity $\boldsymbol{u}^*$ by solving two Helmholtz equations,

$$\nabla^2\boldsymbol{u}^* - \frac{2}{\mu\triangle t}\boldsymbol{u}^* = \mathbf{RHS} \tag{2.70}$$

using fast solvers from FISHPACK [2].

- Solve the Poisson equation (2.46) for the pressure increment ϕ^{n+1} and update the velocity and pressure field as in (2.48) and (2.49), respectively.
- Compute the velocity at the control points, $\boldsymbol{u}^{n+1}(\boldsymbol{X}^{(k)})$ by interpolating from the velocity at the surrounding grid points using the modified bilinear interpolation (see appendix B).

Step 4:

- Evaluate

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

- If $\|g^{(k)}\| < \epsilon$ then $\boldsymbol{X}^{n+1} = \boldsymbol{X}^{(k)}$ and stop the iteration. Otherwise, update $\boldsymbol{X}^{(k+1)}$ and the inverse Jacobian matrix B_{k+1} using BFGS algorithm described in section 2.4. Set $k := k + 1$ and go to step 2.

2.6 Numerical results

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

2.6.1 Forced flow

We first test the order of convergence of our projection algorithm for a non interface problem by considering a forced flow problem. In this problem, the source term is added to the right hand side of the Navier-Stokes equations so that the exact solution is

$$\begin{aligned} u &= -\cos(t)\sin^2(\pi x)\sin(2\pi y) \\ v &= \cos(t)\sin(2\pi x)\sin^2(\pi y) \\ p &= -\frac{\partial \psi}{\partial t} + \mu\Delta\psi \ , \end{aligned}$$

N	$\|u-u_e\|_\infty$	Order	$\|v-v_e\|_\infty$	Order	$\|p-p_e\|_\infty$	Order
64	7.1102×10^{-4}		7.1102×10^{-4}		9.7405×10^{-4}	
128	1.7735×10^{-4}	2.003	1.7735×10^{-4}	2.003	2.4480×10^{-4}	1.992
256	4.4283×10^{-5}	2.002	4.4283×10^{-5}	2.002	6.1800×10^{-5}	1.986

Table 2.1: Grid refinement analysis for the forced flow problem is performed at time $t = 10$. Viscosity $\mu = 1$ and $\triangle t = \triangle x$.

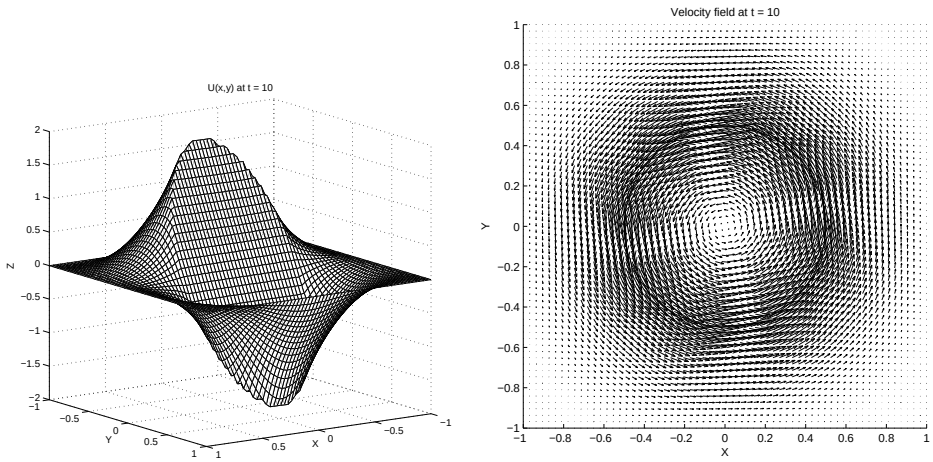
where the function ψ is defined as,

$$\psi = \left[(x-x^2)\sin(\pi x)(y-y^2)\sin(\pi y) - 16/\pi^6\right]\cos t\ .$$

This problem is taken from [38]. The computational domain is the unit square $\Omega = [0,1]\times[0,1]$. Table 2.1 shows the grid refinement analysis performed at time $t = 10$ and viscosity $\mu = 1$. The error in both pressure and velocity is determined in the infinity norm. In this problem, the pressure is not uniquely defined and hence the pressure has been adjusted by a constant to minimize the error in the infinity norm. Table 2.1 shows that our projection method can achieve second order accuracy for both the pressure and velocity. In [38], the order of convergence for pressure of this problem is only nearly second order because the accuracy of the pressure might be deteriorated due to the numerical boundary layer. Here, we use a consistent boundary condition for the intermediate tangential velocity (2.41) and it helps to improve the order of accuracy for the pressure.

2.6.2 Rotational flow

In this example, we consider a fixed interface problem with nonsmooth velocity. The interface is a circle $r = \frac{1}{2}$ located at the center of the computational domain $[-1,\ 1]\times[-1,\ 1]$. The normal and tangential stresses are $f_1 = 0$ and $f_2 = 10\mu$, respectively, and the viscosity is $\mu = 0.02$. The initial velocity and pressure are taken to be zero on the square domain. Since $f_1 = 0$ and f_2 are constant, the pressure and its derivatives are continuous across the interface. On the other hand, the derivatives of the velocity are discontinuous. This can be observed in Fig. 2-7(a) and Fig. 2-8. Figure 2-7(b) shows the steady state solution which corresponds to a rigid body motion inside the



(a) (b)

Figure 2-7: Velocity field at time t = 10 with a 64×64 grid, $\mu = 0.02$, $f_1 = 0$, $f_2 = 10\mu$. 2-7(a) The plot of the x component of velocity field at time t = 10. 2-7(b) The plot of velocity field at time t = 10.

interface. Table 2.2 shows the grid refinement analysis of our immersed interface method for the rotational flow problem. The order of convergence for both velocity and pressure is second order. Note that the error is measured up to the boundary, i.e. the error at the boundary is also taken into account.

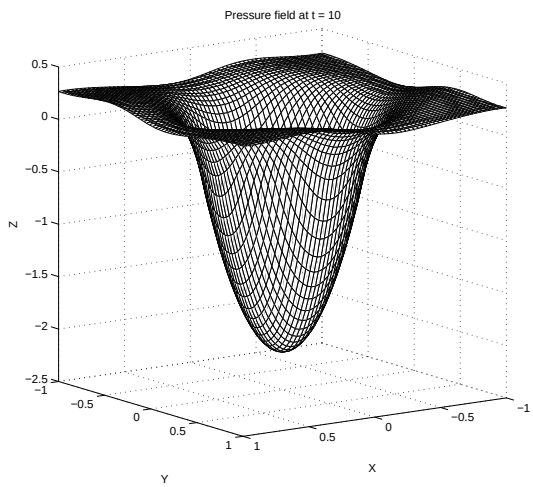


Figure 2-8: Pressure is smooth over the domain since the force is only along the tangential direction of the interface.

N	$\|u_N - u_{512}\|_\infty$	Order	$\|v_N - v_{512}\|_\infty$	Order	$\|p_N - p_{512}\|_\infty$	Order
64	2.4382×10^{-3}		2.1654×10^{-3}		3.1318×10^{-3}	
128	5.6889×10^{-4}	2.0996	5.1120×10^{-4}	2.0818	8.0890×10^{-4}	1.9530
256	1.1539×10^{-4}	2.3016	1.1260×10^{-4}	2.1829	1.9039×10^{-4}	2.0870

Table 2.2: Grid refinement analysis for the rotational flow problem performed at $t = 10$, $\triangle t = \triangle x/4$ and $\mu = 0.02$.

2.6.3 Surface tension

In this example, we consider an unsteady deformable interface problem with surface tension [36]. The initial interface is an ellipse with major and minor axes a = 0.75, b = 0.5, respectively. The force strength is now given by

$$\boldsymbol{f}(s,t) = \gamma \frac{\partial^2}{\partial s^2} \boldsymbol{X}(s,t) \ , \tag{2.71}$$

which can be seen to be at all points normal to the interface. The initial velocity and pressure are set to zero. The computational domain is $[-1.5,\ 1.5]\times[-1.5,\ 1.5]$ with $\rho = 1$ and $\mu = 0.1$ throughout the domain. The center of the ellipse is located at $(0,0)$. In our test, we take $\gamma = 10$ and $\boldsymbol{u}|_{\partial\Omega} = \mathbf{0}$. In Fig. 2-9, we plot the location of the interface obtained at different times with a 64×64 grid and 48 control points to represent the interface. The evolution of the major and minor axes is shown in Fig. 2-10. The interface oscillates as it settles down to the equilibrium state. Fig. 2-11 shows the evolution of the major and minor axes of the ellipse when $\mu = 1$ and $\gamma = 1$. In this case, the Reynolds number is smaller and the interface relaxes gradually to the equilibrium state without oscillations. In the equilibrium state, the interface is a circle, the velocity is zero everywhere and the pressure has two different constant values inside and outside the interface as show in Fig. 2-12. We also perform a grid refinement analysis to study the error in the location of the interface. The error in the location of the interface is measured by the error in the position of the control points. We compute the error in both L_2 and maximum norms as

$$\|e\|_2 = \sqrt{\frac{1}{N_b}\sum_{i=1}^{N_b}\left(X_i^{(\mathrm{ref})} - X_i^{(N)}\right)^2 + \left(Y_i^{(\mathrm{ref})} - Y_i^{(N)}\right)^2} \ , \tag{2.72}$$

$$\|e\|_\infty = \max_{1\le i\le N_b} \sqrt{\left(X_i^{(\mathrm{ref})} - X_i^{(N)}\right)^2 + \left(Y_i^{(\mathrm{ref})} - Y_i^{(N)}\right)^2}\ , \tag{2.73}$$

respectively, where N_b is the number of control points, $(X_i^{(\mathrm{ref})}, Y_i^{(\mathrm{ref})})$ is the location of a control point i^{th} obtained on the finest grid and $X_i^{(N)}, Y_i^{(N)}$ is the location of the control point i^{th} obtained on a coarser $N \times N$ grid. In Fig. 2-13, we plot the error in both the L_2 and the maximum norms at $t = 1$ for the problem with $\mu = 0.1$ and $\gamma = 10$. The finest grid is 480×480 and the coarser grids are 120×120, 160×160, 200×200, 240×240 and 320×320. We use the same (large) number of control points, $N_b = 120$, in all grids to allow for an easy comparison of the results.

2.6.4 Elastic membrane

In this example, we consider a deformable interface problem which involves an elastic membrane [36]. The initial interface is an ellipse with major and minor axes a = 0.75, b = 0.5, respectively. The ellipse is located at the center of the computational domain. The force strength exerted by the elastic membrane at $\boldsymbol{X}(s,t)$ is given as,

$$\boldsymbol{f}(s,t) = \frac{\partial}{\partial s}(T(s,t)\boldsymbol{\tau}(s,t))\ ,$$

where $T(s,t)$ is the tension of the membrane at this point, defined by

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

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

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

Here, s_0 is the arc-length measured along the unstretched membrane which is the dotted line in Fig. 2-14 with radius $r_0 = 0.5$. Because of the incompressibility condition, we expect that the membrane will relax to the equilibrium state (the dash-dot line in Fig. 2-14) with radius $r = \sqrt{ab} \approx 0.61237$. We start our simulation by setting the initial velocity and pressure fields to zero. The computational domain is

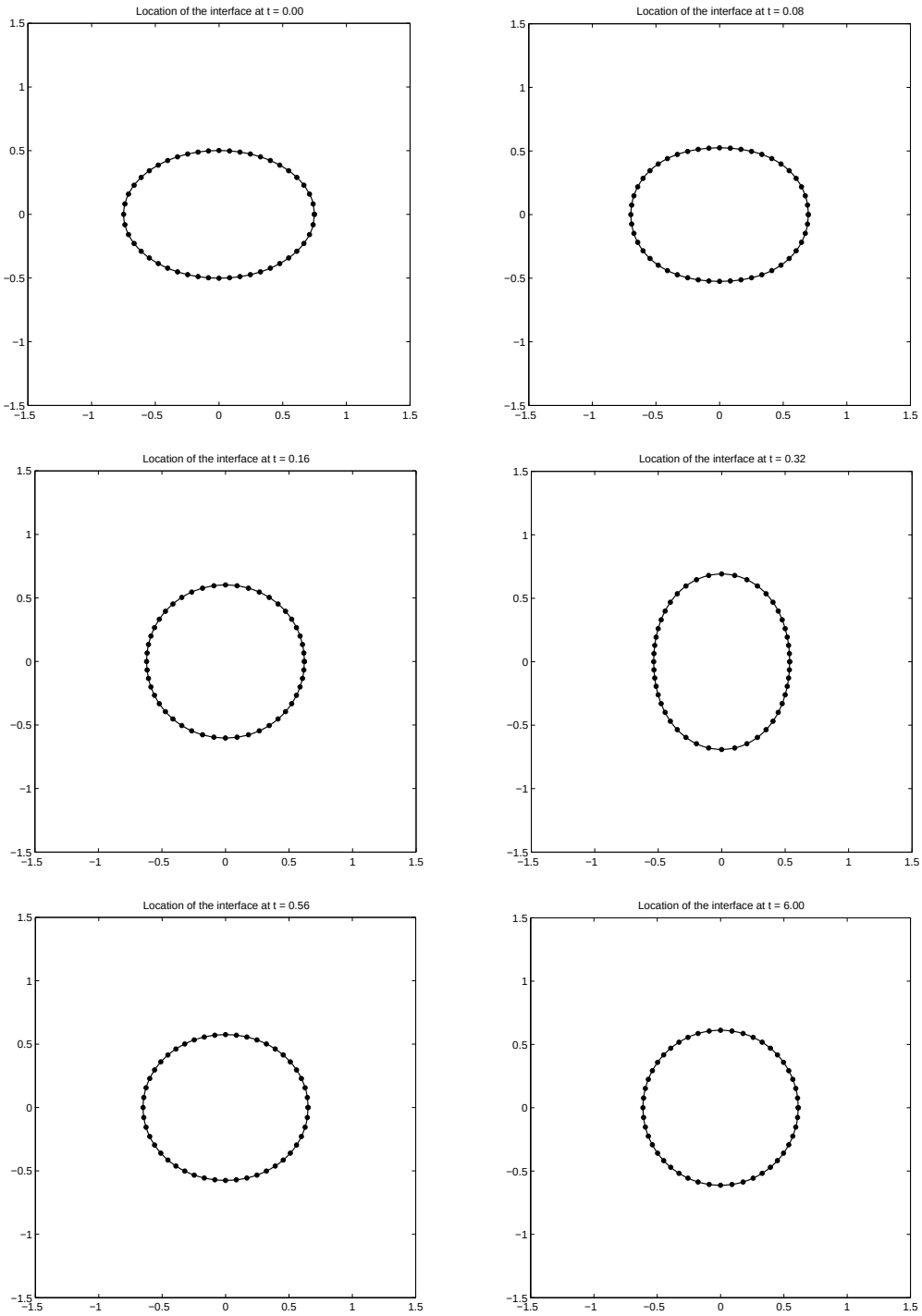


Figure 2-9: The location of the interface at different times.

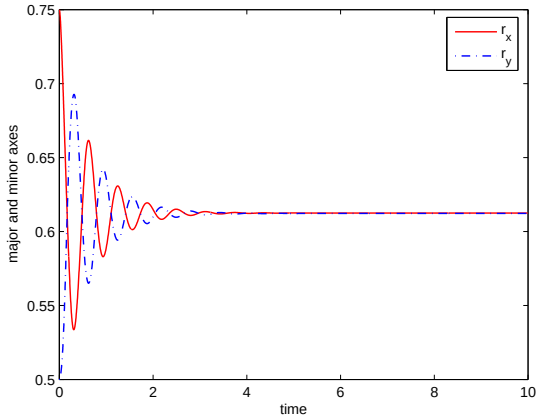


Figure 2-10: The evolution of r_x and r_y with $\mu = 0.1$ and $\gamma = 10$. The interface oscillates as it converges to the equilibrium state

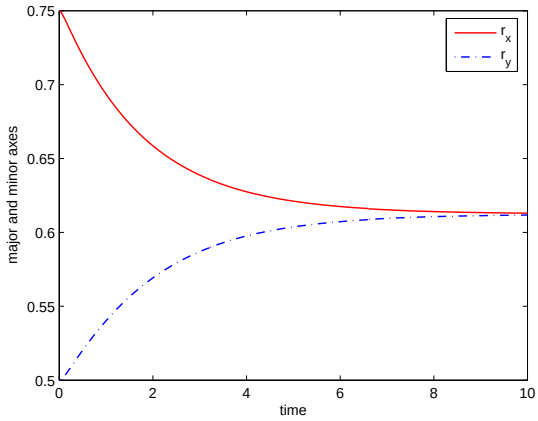


Figure 2-11: The evolution of r_x and r_y with $\mu = 1$ and $\gamma = 1$. The interface relaxes gradually to the equilibrium state without oscillations.

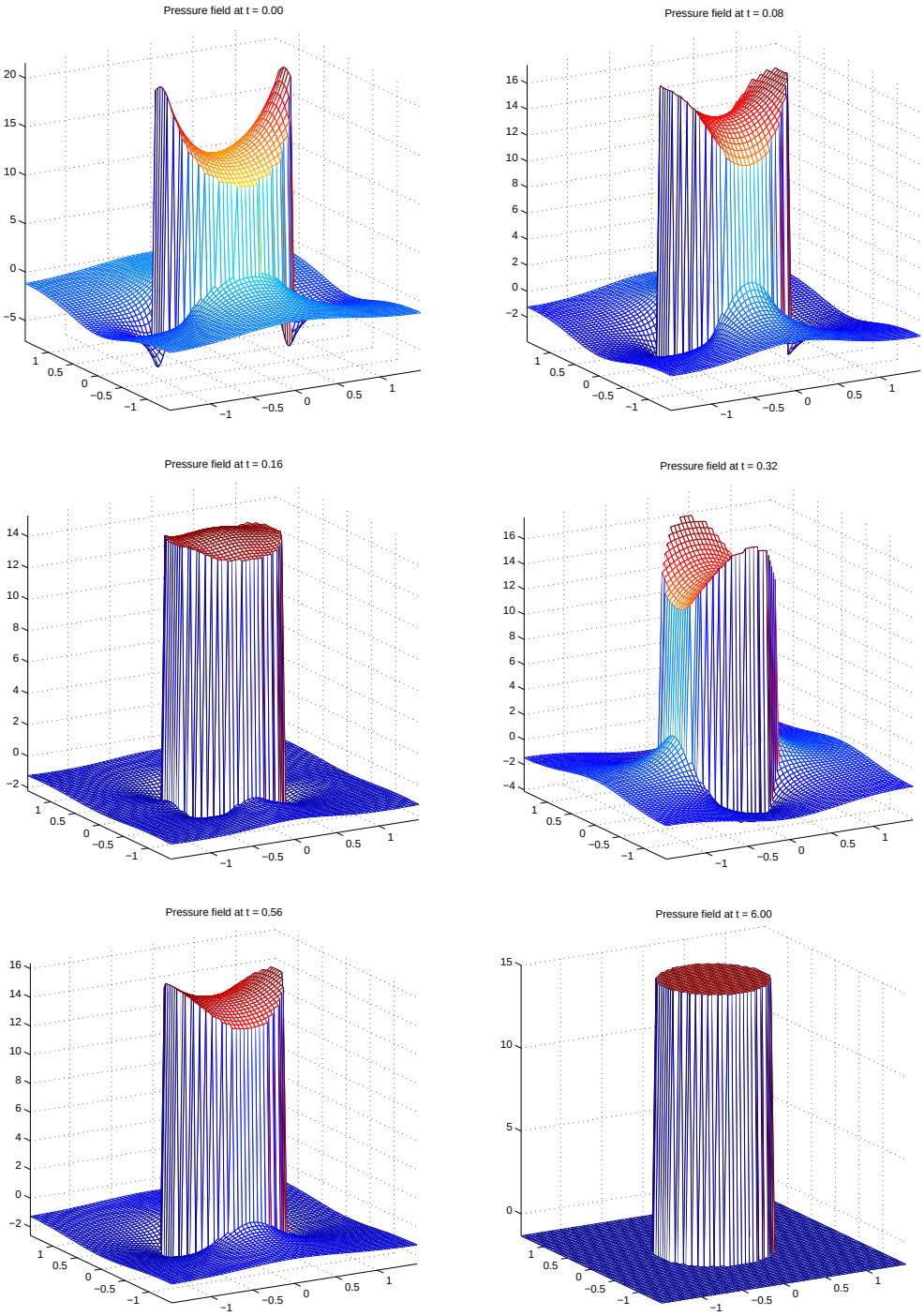
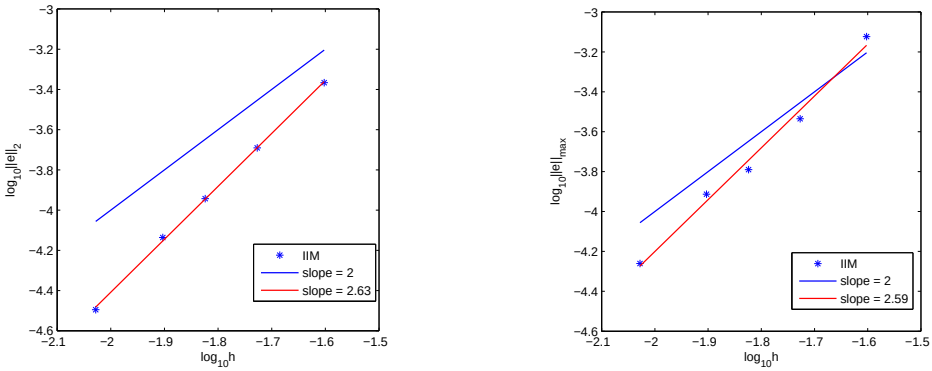


Figure 2-12: The pressure distribution at different times.



(a) (b)

Figure 2-13: The error in the location of the interface at t = 1 as measured in: 2-13(a) L_2 norm, 2-13(b) maximum norm. Viscosity, $\mu = 0.1$ and surface tension constant, $\gamma = 10$.

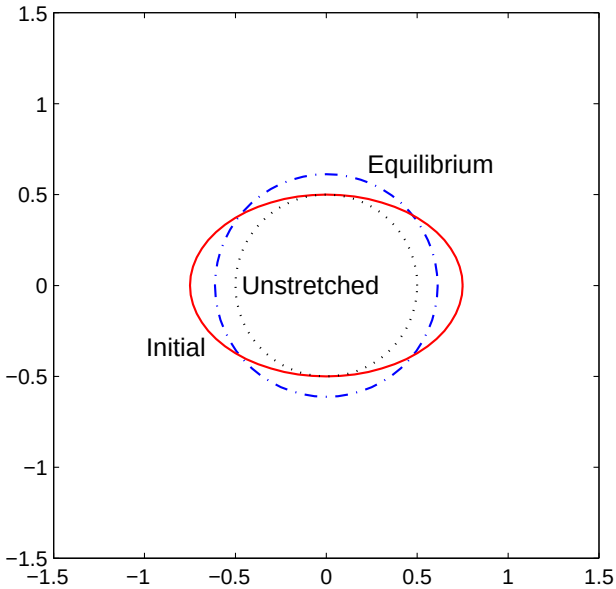
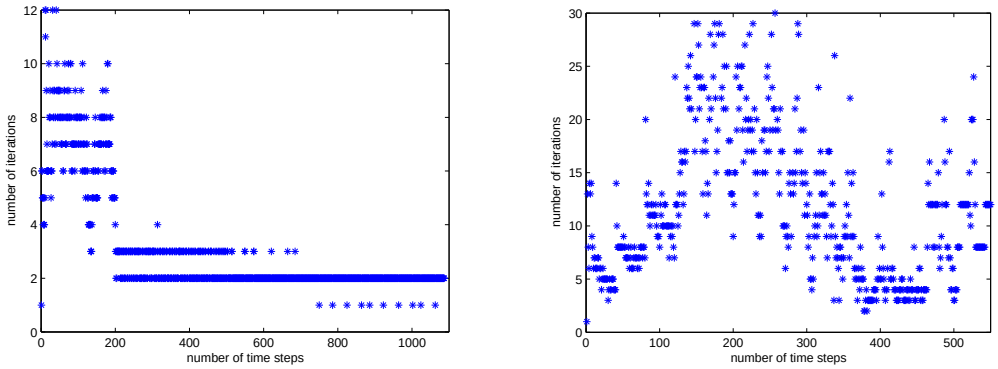


Figure 2-14: Initial, unstretched and equilibrium positions of the elastic membrane.

$[-1.5,\ 1.5] \times [-1.5,\ 1.5]$ with $\rho = 1$ and $\mu = 0.1$ throughout the domain. In our test, we take $T_0 = 10$ and $\boldsymbol{u}|_{\partial\Omega} = \mathbf{0}$. The simulation is performed with a 64×64 grid and 48 control points to represent the interface. The timestep in all simulations is $\Delta t = \Delta x/5$. We are actually able to increase the timestep further without loss of stability. However, we found that the number of iterations in solving (2.62) per timestep increases as the timestep increases. Therefore the total iterations, and hence the computational cost, for running the simulation to a particular time level may increase when using a larger timestep. For example, Fig. 2-15(a) and Fig. 2-15(b) show the number of iterations required at each timestep to run the simulations to $t = 10$ using $\Delta t = \Delta x/5$ and $\Delta t = 2\Delta x/5$, respectively. The tolerance that we use for the stopping criteria is $\epsilon = 5 \times 10^{-8}$. It is easy to see that the total number of iterations performed when using $\Delta t = 2\Delta x/5$ is much greater than that performed when using $\Delta t = \Delta x/5$.

The evolution of the major and minor axes is shown in Fig. 2-16. The interface oscillates as it settles down to the equilibrium state. Figure 2-17 and Figure 2-18 show the velocity fields and the pressure fields at different time levels. Since the fluid is incompressible, the area inside the membrane should be conserved. In Fig. 2-19, we perform a grid refinement analysis to study the conservation of the area enclosed by the interface. It could be seen that the area is conserved with second order accuracy. Note that we use the same number of control points, i.e. $N_b = 96$, for the different grids used in the grid refinement analysis.

Figure 2-20 shows the evolution of the major and minor axes of the elastic membrane when $\mu = 0.02$ and $T_0 = 10$. Since the Reynolds number is higher, the elastic membrane takes a longer time to oscillate before settling down to the equilibrium state. Fig. 2-21 shows the evolution of the major and minor axes of the ellipse when $\mu = 1$ and $T_0 = 10$. In this case, the interface relaxes gradually to the equilibrium state without oscillations. In the equilibrium state, the interface is a circle, the velocity is zero everywhere and the pressure has two different constant values inside and outside the interface. We note that the behavior of the elastic membrane is almost the same as the interface with surface tension considered in the previous example. However,



(a) (b)

Figure 2-15: Number of BFGS iterations performed at each timestep to run the simulations to $t = 10$. 2-15(a): $\Delta t = \Delta x/5$, 2-15(b): $\Delta t = 2\Delta x/5$.

the tangential force along the elastic membrane is different from zero. This tangential force results in a jump in the derivatives of the velocity which does not appear at the interface with surface tension. In Fig. 2-22, we plot a cross section of the u component of the velocity field at $t = 0.2$ and $y = -0.207$. It is clear that the u component of the velocity field at this position is continuous but not smooth across the interface.

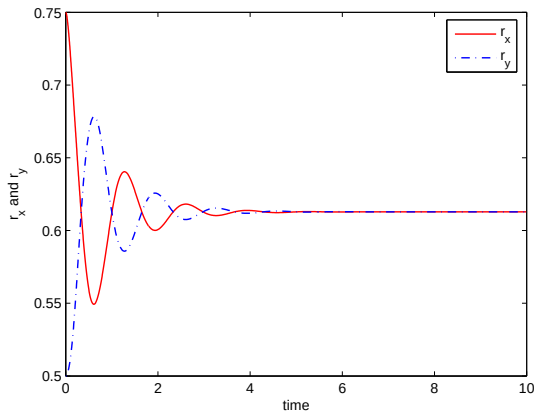


Figure 2-16: The evolution of r_x and r_y with $\mu = 0.1$ and $T_0 = 10$. The interface oscillates as it converges to the equilibrium state.

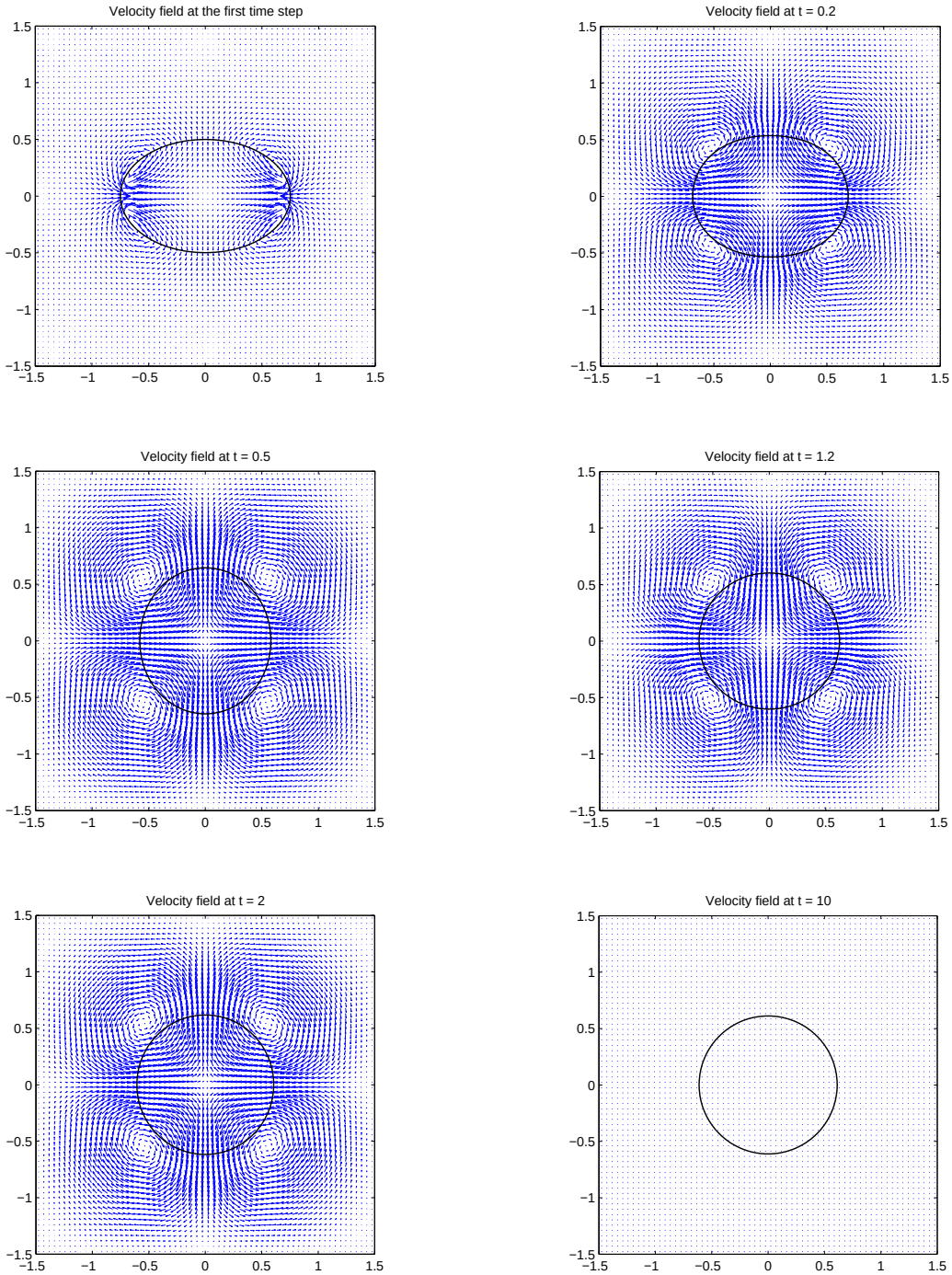


Figure 2-17: The velocity fields at different times. Simulation is performed on a 64×64 grid with 48 control points and $\Delta t = \Delta x/5$.

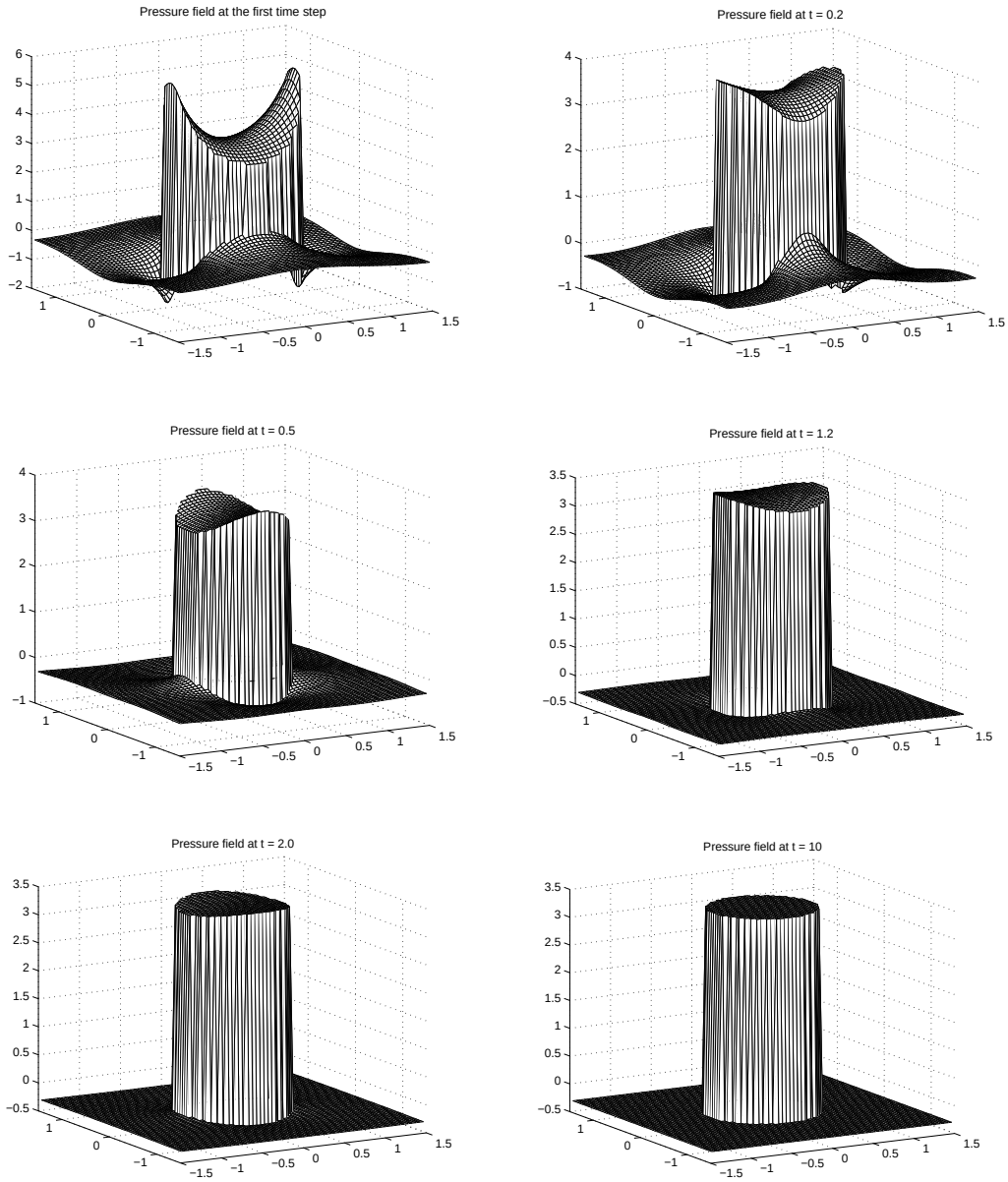


Figure 2-18: The pressure distribution at different times. Simulation is performed on a 64×64 grid with 48 control points and $\Delta t = \Delta x/5$.

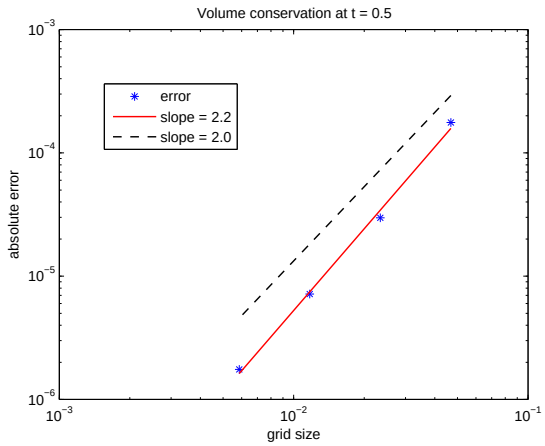


Figure 2-19: Grid refinement analysis for studying the conservation of the area enclosed by the membrane at $t = 0.5$.

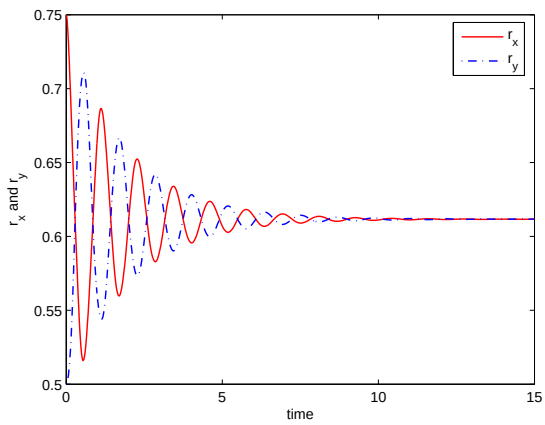


Figure 2-20: The evolution of r_x and r_y with $\mu = 0.02$ and $T_0 = 10$. The interface oscillates as it converges to the equilibrium state.

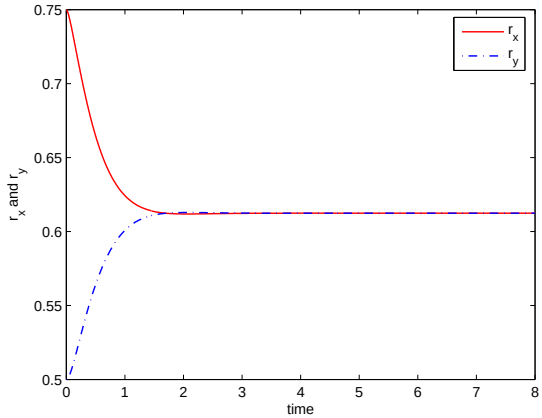


Figure 2-21: The evolution of r_x and r_y with $\mu = 1$ and $T_0 = 10$. The interface relaxes gradually to the equilibrium state without oscillations.

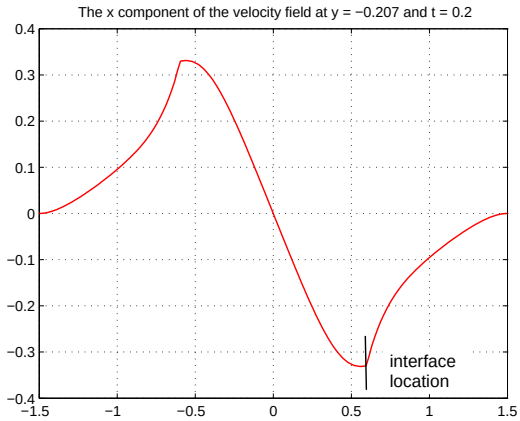


Figure 2-22: A cross section of u, the x component of the velocity field at $t = 0.2$ and $y = -0.207$. It is continuous but non-smooth across the interface.

Chapter 3

An Immersed Interface Method for the Navier-Stokes equations with rigid boundaries

3.1 Introduction

This chapter considers the immersed interface method for the incompressible Navier-Stokes equations in general domains involving rigid boundaries. In a 2-dimensional bounded domain Ω that contains a rigid boundary $\Gamma(\mathrm{t})$, the incompressible Navier-Stokes equations introduced in Chapter 1 are written as

$$\boldsymbol{u}_t + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p = \mu \triangle \boldsymbol{u} + \boldsymbol{F} \tag{3.1}$$

$$\nabla \cdot \boldsymbol{u} = 0 \tag{3.2}$$

Conventional methods for the Navier-Stokes equations with rigid immersed boundaries include the body-fitted or structured grid approach. In this approach, the Navier-Stokes equations are discretized on a curvilinear grid that conforms to the immersed boundary and hence the boundary conditions can be imposed easily. The main disadvantage of this method is that robust grid generation is required to account

for the complexity of the immersed boundary. Besides, this method is computationally expensive for general motions of the rigid boundary. The finite element method with unstructured meshes is also an effective approach for simulating flows with complex stationary immersed boundaries. However, this method is also computationally expensive for moving boundaries since the unstructured mesh must be regenerated at each time step. Generating a mesh at each timestep can be partially avoided by using a moving mesh finite element method [17].

An alternative approach for solving complex viscous flows is the Cartesian grid method, which has become popular in recent years. This method solves the governing equations on a Cartesian grid and retains the simplicity of the Navier-Stokes equations on the Cartesian coordinates. In addition, this method has the advantage of enabling the use of fast solvers. The main disadvantage of the Cartesian grid method is that no grid refinement is performed in the area near the immersed boundaries where the solutions need to be resolved. One of the most successful Cartesian grid methods is Peskin's immersed boundary (IB) method [30, 45, 51]. In order to deal with rigid boundaries, Lai and Peskin [30] proposed to evaluate the force density using an expression of the form,

$$\boldsymbol{f}(s,t) = \kappa(\boldsymbol{X}^e(s) - \boldsymbol{X}(s,t)), \tag{3.3}$$

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 (3.3) is a particular case of the feedback forcing formulation proposed by Goldstein et al. [25] with $\beta = 0$. In [25], the force is expressed as

$$\boldsymbol{f}(s,t) = \alpha \int_0^t \boldsymbol{U}(s,t')dt' + \beta \boldsymbol{U}(s,t) \tag{3.4}$$

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. [51] proposed an alternative model to compute the force density $\boldsymbol{f}$ based upon the evaluation of the various terms in the momentum equation (3.1) 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,

$$\boldsymbol{f}(\boldsymbol{X}) = \frac{\partial \boldsymbol{u}(\boldsymbol{X})}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u}(\boldsymbol{X}) + \nabla p(\boldsymbol{X}) - \mu \nabla^2 \boldsymbol{u}(\boldsymbol{X}) \ . \tag{3.5}$$

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. As mentioned in Chapter 1, the immersed boundary method 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 addition, using the feed back forcing formulas (3.3), (3.4) or the expression (3.5) to compute the force at the rigid boundary explicitly results in the restriction of the timestep.

Recently, the immersed interface method [35] has been employed to solve for viscous flows with static rigid immersed boundaries [13, 32, 39, 47]. Again, the immersed interface method can avoid smearing out sharp interfaces and maintains second-order accuracy by incorporating the known jumps into the finite difference scheme near the interface. In [13, 39] the no-slip boundary conditions are imposed directly by determining the correct jump conditions for streamfunction and vorticity. In [13], 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 boundaries 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. [39] suggested to use the generalized minimal residual (GMRES) method [48] to solve the Schur complement system. However, this approach has not been applied for moving geometry. In [47], 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 [13, 39], 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.

Another Cartesian grid approach has been presented by Ye et al. [63] and Udaykumar et al. [56] using a finite volume technique. 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 chapter, we extend our earlier work, presented in the previous chapter for problems with deformable boundaries, to solve problems with rigid immersed boundaries. The method presented in this chapter is based on that presented in Le et al. [32]. Our approach uses the immersed interface method to solve the incompressible Navier-Stokes equations formulated in primitive variables. In the previous chapter, the singular force $\boldsymbol{f}$ was computed as a function of the configuration of the interface, i.e., the interface was moved with the fluid velocity and the singular force was computed from the constitutive equation of the membrane. In the present chapter, 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 in the finite difference discretization to obtain sharp interface resolution. Fast solvers from the FISHPACK software library [2] are used to solve the resulting discrete systems of equations.

3.2 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 (2.42) is first solved for the intermediate velocity $\boldsymbol{u}^*$. The pressure increment ϕ^{n+1} is then determined by solving Eqn (2.46). Finally the velocity field is updated using Eqn (2.48). 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

$$\boldsymbol{U_k} = \boldsymbol{U}(\boldsymbol{X}_k) = \mathcal{B}(\boldsymbol{u}^{n+1}) \ , \tag{3.6}$$

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 B.

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,

$$\begin{aligned}
\text{Eqn (2.42)} \quad &\rightarrow \quad \boldsymbol{H}\boldsymbol{u}^* = \boldsymbol{C} + \boldsymbol{B}_1\boldsymbol{f} \\
\text{Eqn (2.46)} \quad &\rightarrow \quad \boldsymbol{L}\boldsymbol{\phi}^{n+1} = \boldsymbol{D}\boldsymbol{u}^* + \boldsymbol{B}_2\boldsymbol{f} \\
\text{Eqn (2.48)} \quad &\rightarrow \quad \boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \boldsymbol{G}\boldsymbol{\phi}^{n+1} + \boldsymbol{B}_3\boldsymbol{f} \\
\text{Eqn (3.6)} \quad &\rightarrow \quad \boldsymbol{U_k} = \boldsymbol{M}\boldsymbol{u}^{n+1} + \boldsymbol{B}_4\boldsymbol{f}
\end{aligned}$$

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,

$$\begin{aligned}
\boldsymbol{U_k} = \boldsymbol{M}\big(\boldsymbol{H}^{-1}\boldsymbol{C} - \boldsymbol{G}\boldsymbol{L}^{-1}\boldsymbol{D}\boldsymbol{H}^{-1}\boldsymbol{C}\big) \\
+ \Big(\boldsymbol{M}\big(\boldsymbol{H}^{-1}\boldsymbol{B}_1 - \boldsymbol{G}\boldsymbol{L}^{-1}\boldsymbol{D}\boldsymbol{H}^{-1}\boldsymbol{B}_1 - \boldsymbol{G}\boldsymbol{L}^{-1}\boldsymbol{B}_2 + \boldsymbol{B}_3\big) + \boldsymbol{B}_4\Big)\boldsymbol{f} \ .
\end{aligned} \tag{3.7}$$

For convenience, we can write (3.7) as

$$\boldsymbol{U_k} = \boldsymbol{U}_k^0 + \boldsymbol{A}\boldsymbol{f} \ , \tag{3.8}$$

where $\boldsymbol{U}_k^0$ is simply the velocity at the control points obtained by solving Eqns (2.42),

(2.46), (2.48) and (3.6) 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:

$$\frac{\boldsymbol{u}_f^*}{\triangle t} = \frac{\mu}{2}\nabla^2 \boldsymbol{u}_f^*, \qquad \boldsymbol{u}_f^*|_{\partial\Omega} = 0 \tag{3.9}$$

$$\nabla^2 \phi_f^{n+1} = \frac{\nabla \cdot \boldsymbol{u}_f^*}{\triangle t}, \qquad \boldsymbol{n} \cdot \nabla \phi_f^{n+1}|_{\partial\Omega} = 0 \tag{3.10}$$

$$\boldsymbol{u}_f^{n+1} = \boldsymbol{u}_f^* - \Delta t \nabla \phi_f^{n+1} \tag{3.11}$$

$$\boldsymbol{Af} = \mathcal{B}(\boldsymbol{u}_f^{n+1}) \tag{3.12}$$

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

Equation (3.8) 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

$$\boldsymbol{Af} = \boldsymbol{U}_p - \boldsymbol{U}_k^0 \tag{3.13}$$

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 (3.9)–(3.12). We solve Eqns (3.9)–(3.12) $2N_b$ 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 (3.13) is then solved at each timestep for the singular force $\boldsymbol{f}$ via back substitution. Finally, we solve Eqs (2.42)–(2.49) 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. An alternative consists of using a singular value decomposition (SVD) method to solve the singular system of equations (3.13). 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 (3.13) may not lie exactly 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 (3.13) iteratively. Each iteration of GMRES method requires a matrix-vector product which can be found by solving (3.9)–(3.12). In each matrix-vector product, we have to solve two Helhmoltz equations (3.9) and a Poisson equation (3.10). 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, a version of GMRES method for singular linear systems of equations is required. We employed the GMRES method presented in [11] which uses the incremental condition estimation (ICE) [8] to monitor the conditioning of the upper Hessenberg matrix.

3.3 Implementation

In this section, we describe a basic implementation of our algorithm for the Navier-Stokes equations with immersed rigid boundaries. We described our approach for the problem of the flow past a circular cylinder. To start our procedure 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 a singular value decomposition as,

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \ , \tag{3.14}$$

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 (3.13) by computing $\boldsymbol{U}_k^0$.

- Set $\boldsymbol{f} = 0$. Solve (2.42), (2.46) and (2.48) for the velocity at all grid points.
- Interpolate the velocity at the control points $\boldsymbol{U}_k^0$ as in (3.6).
- Compute the right-hand side vector $\boldsymbol{b} = \boldsymbol{U}_p - \boldsymbol{U}_k^0$.

Step 2: Compute the singular force by solving (3.13) 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,

$$\boldsymbol{f} = \sum_{i=1}^{N-1} \frac{u_i^T \boldsymbol{b}}{\sigma_i} v_i \ . \tag{3.15}$$

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

Step 3: Compute $\boldsymbol{u}^{n+1}$ and $p^{n+1/2}$ using the projection method. This step is similar to step 3 in section 2.5.

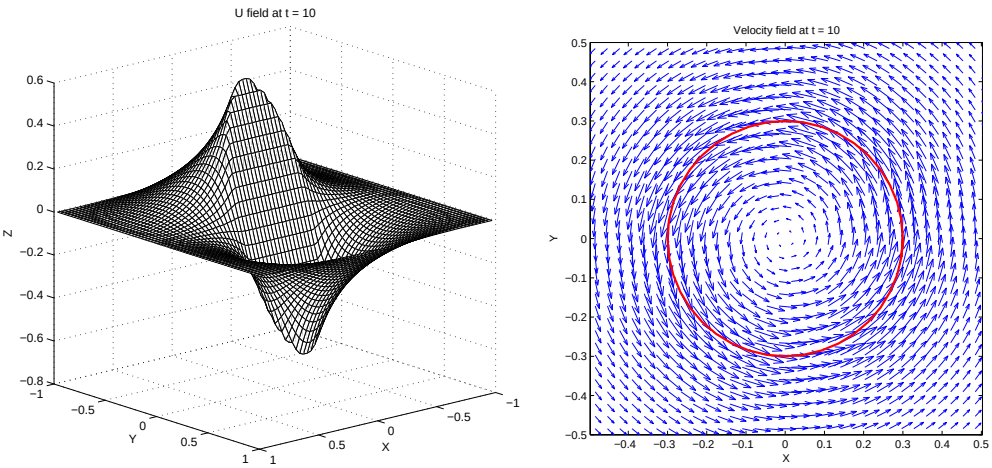
For moving geometry, we still have the same algorithm except that the GMRES solver is applied to solve (3.13) iteratively at each time step. Thus we do not need to form the coefficient matrix explicitly.

3.4 Numerical results

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

3.4.1 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 3-1. 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 3.1 show that the velocity is second order accurate and the pressure is nearly second order accurate.



(a) (b)

Figure 3-1: 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 $\omega = 2$. 3-1(a) Plot of the x component of velocity field. 3-1(b) Plot of velocity field.

3.4.2 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, $Re = \frac{\rho U_\infty d}{\mu}$. 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, 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 use the free stream velocity as the initial velocity and the initial pressure is set to zero. Then, the force at the cylinder interface is determined such that there is no flow inside the cylinder and the pressure is an arbitrary 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

$$C_D = \frac{D}{\frac{1}{2}\rho U_\infty^2 d} \ . \tag{3.16}$$

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

$$D = -\int_\Gamma f_x \mathrm{d}s \ , \tag{3.17}$$

where f_x is the x component of the singular force.

The lift coefficient is defined as

$$C_L = \frac{L}{\frac{1}{2}\rho U_\infty^2 d} \ . \tag{3.18}$$

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

$$L = -\int_\Gamma f_y \mathrm{d}s \ , \tag{3.19}$$

where f_y is the y component of the singular force.

The Strouhal number is defined as,

$$St = \frac{fd}{U_\infty} \ , \tag{3.20}$$

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 [51]. Finally, the dimensionless time is defined as

$$T = \frac{U_\infty t}{d} \ . \tag{3.21}$$

Fig. 3-2 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 coefficient, the length

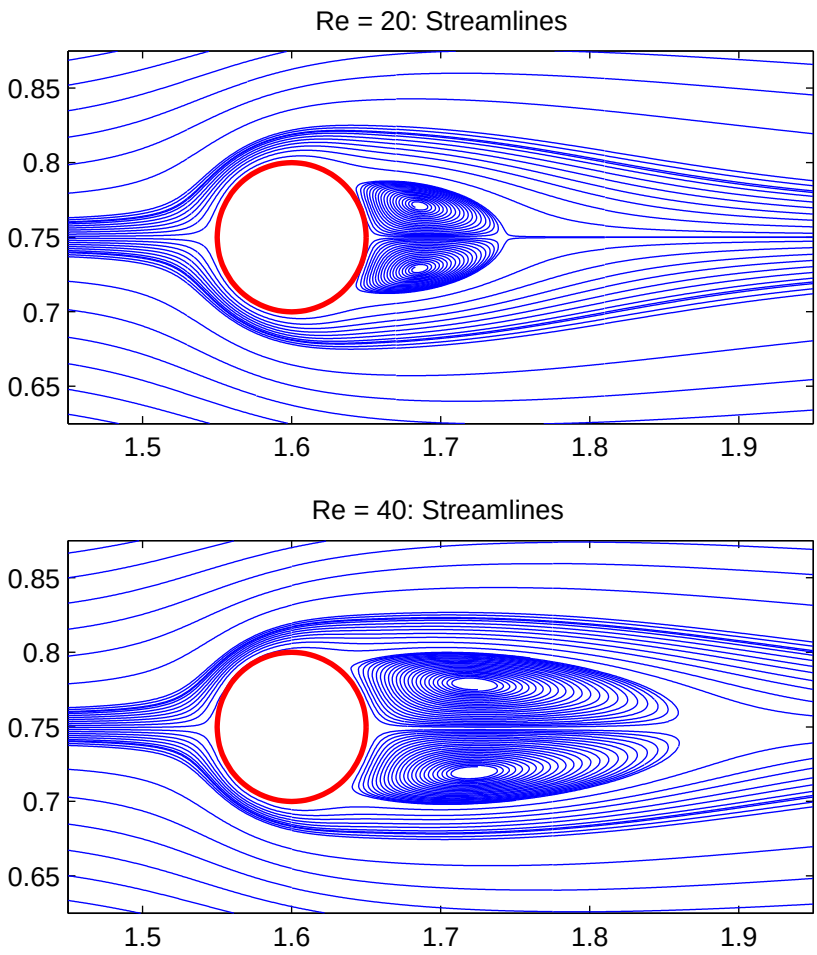


Figure 3-2: Streamlines for Re = 20 and Re = 40.

of the recirculation zone and the angle of separation are computed and are compared with established results in Table 3.2. The results obtained by our method are compared to the numerical simulations [13, 16, 24, 47] as well as experimental results [15, 54]. 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 close to other numerical results but it is about 8% lower than the experimental measurement of Tritton [54]. For $Re = 40$ our drag coefficient is about 5% higher than the experimentally determined value [54]. Fig. 3-3 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. 3-4

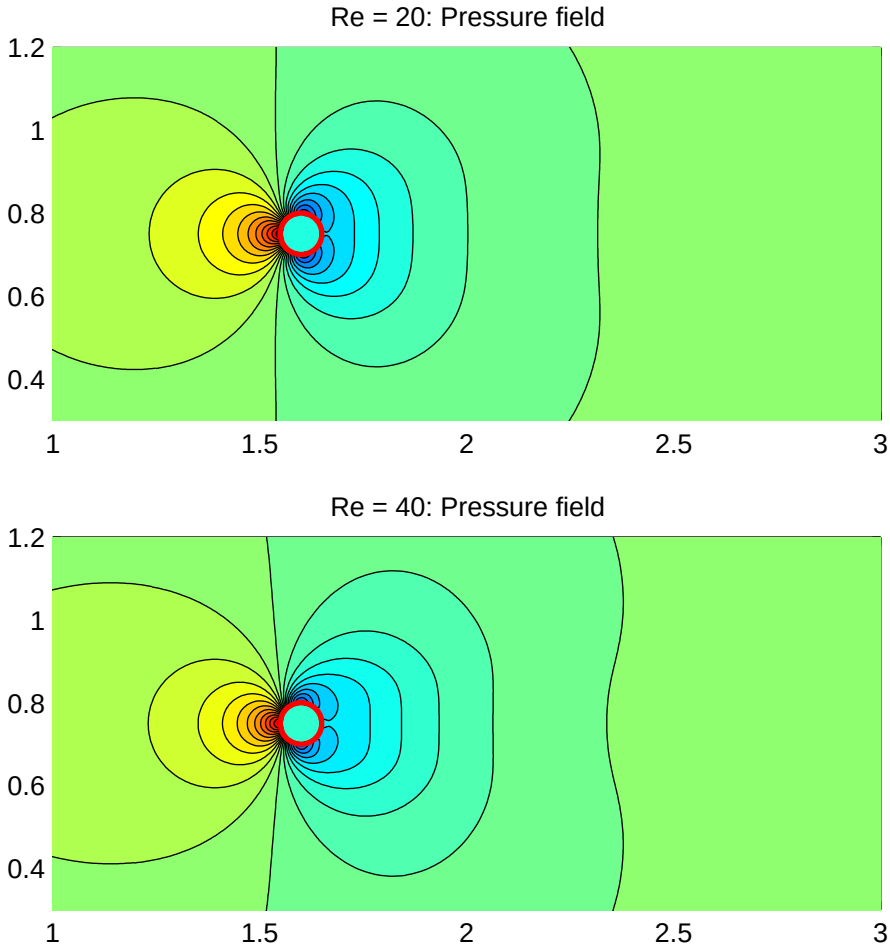


Figure 3-3: Pressure fields for Re = 20 and Re = 40.

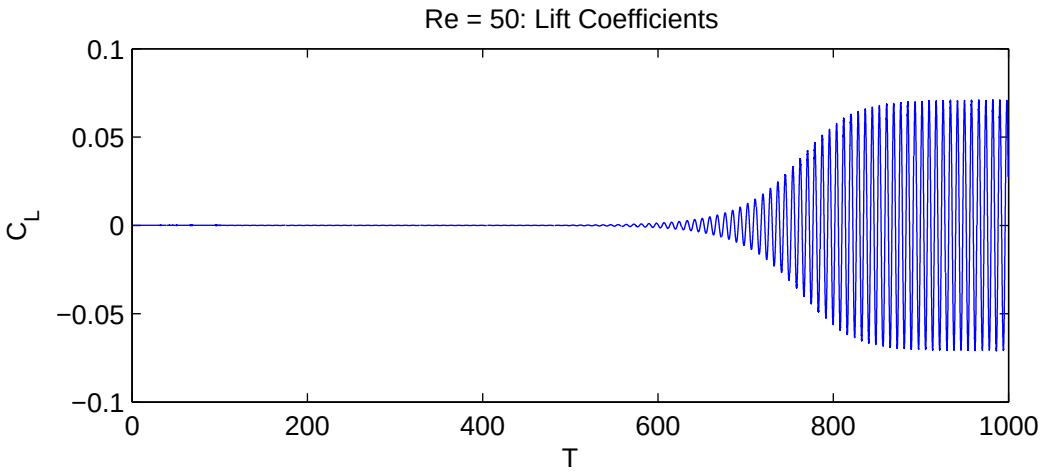


Figure 3-4: Lift Coefficients at Re = 50.

	$Re = 20$			$Re = 40$		
	L/d	θ	C_D	L/d	θ	C_D
Tritton [54]	–	–	2.22	–	–	1.48
Coutanceau and Bouard [15]	0.73	42.3^o	–	1.89	52.8^o	–
Fornberg [24]	0.91	–	2.00	2.24	–	1.50
Dennis and Chang [16]	0.94	43.7^o	2.05	2.35	53.8^o	1.52
Calhoun [13]	0.91	45.5^o	2.19	2.18	54.2^o	1.62
Russell and Wang [47]	0.94	43.3^o	2.13	2.29	53.1^o	1.60
Ye et al. [63]	0.92	–	2.03	2.27	–	1.52
Present	0.93	43.9^o	2.05	2.22	53.6^o	1.56

Table 3.2: Length of the recirculation zone, Angle of Separation and Drag Coefficient for Re = 20 and Re = 40

shows that our algorithm is able to detect the onset of an instability at a Reynolds number of about 50. It has been reported that the wake behind the cylinder first becomes unstable at a critical Reynolds number of about $Re = 46 \pm 1$ [63]. Above this Reynolds number the cylinder wake instability rises and grows in time and leads to Karman vortex shedding. This behavior is 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. 3-5 shows the pressure fields at Re = 100, 200 and 300. The instabilities and vortex shedding can be visualized from this figure. In Table 3.3 and Table 3.4, the drag and lift coefficients at Re = 100 and Re = 200 are compared to other numerical simulations. For Re = 100, the mean drag obtained by our algorithm is slightly greater than that computed by other researchers [9, 13, 40]. 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 [9, 13, 40, 47]. Our value is about 15% higher than that in Calhoun [13] and 4% lower than the value obtained by Braza et al. [9]. In Table 3.4, 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. 3-6 and Fig. 3-7 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

C_D	$Re = 100$	$Re = 200$
Braza et al. [9]	1.36 ± 0.015	1.40 ± 0.050
Liu at al. [40]	1.35 ± 0.012	1.31 ± 0.049
Calhoun [13]	1.33 ± 0.014	1.17 ± 0.058
Russell et al. [47]	1.38 ± 0.007	1.29 ± 0.022
Present	1.37 ± 0.009	1.34 ± 0.030

Table 3.3: Drag Coefficients for Re = 100 and Re = 200

C_L	$Re = 100$	$Re = 200$
Braza et al. [9]	± 0.250	± 0.75
Liu at al. [40]	± 0.339	± 0.69
Calhoun [13]	± 0.298	± 0.67
Russell et al. [47]	± 0.300	± 0.50
Present	± 0.323	± 0.43

Table 3.4: Lift Coefficients for Re = 100 and Re = 200

is compared with established results in Table 3.5. Our computed Strouhal number obtained at Re = 80 comes out to be 0.15 which compares very well with the values obtained from experiment [62] and from numerical simulation [63]. At Re = 100 and Re = 200, our Strouhal numbers are in good agreement with those given in [13, 40, 47] and differ from the experimental results [62] by 1.8% and 1%, respectively. At Re = 300, our computed Strouhal number compares very well with the value obtained from experiment [62].

3.4.3 Flow past a flat plate

In this example, we simulate an unsteady flow past a flat plate immersed in a rectangular domain $\Omega = [0, 3] \times [-0.75, 0.75]$. The flat plate, whose length is $L = 0.1$, is

St	$Re = 80$	$Re = 100$	$Re = 200$	$Re = 300$
Ye et al. [63]	0.15	–	–	0.210
Williamson [62]	0.15	0.163	0.185	0.203
Liu at al. [40]	–	0.164	0.192	–
Calhoun [13]	–	0.175	0.202	–
Russell et al. [47]	–	0.169	0.195	–
Present	0.15	0.160	0.187	0.200

Table 3.5: Strouhal numbers for Re = 80, 100, 200 and 300

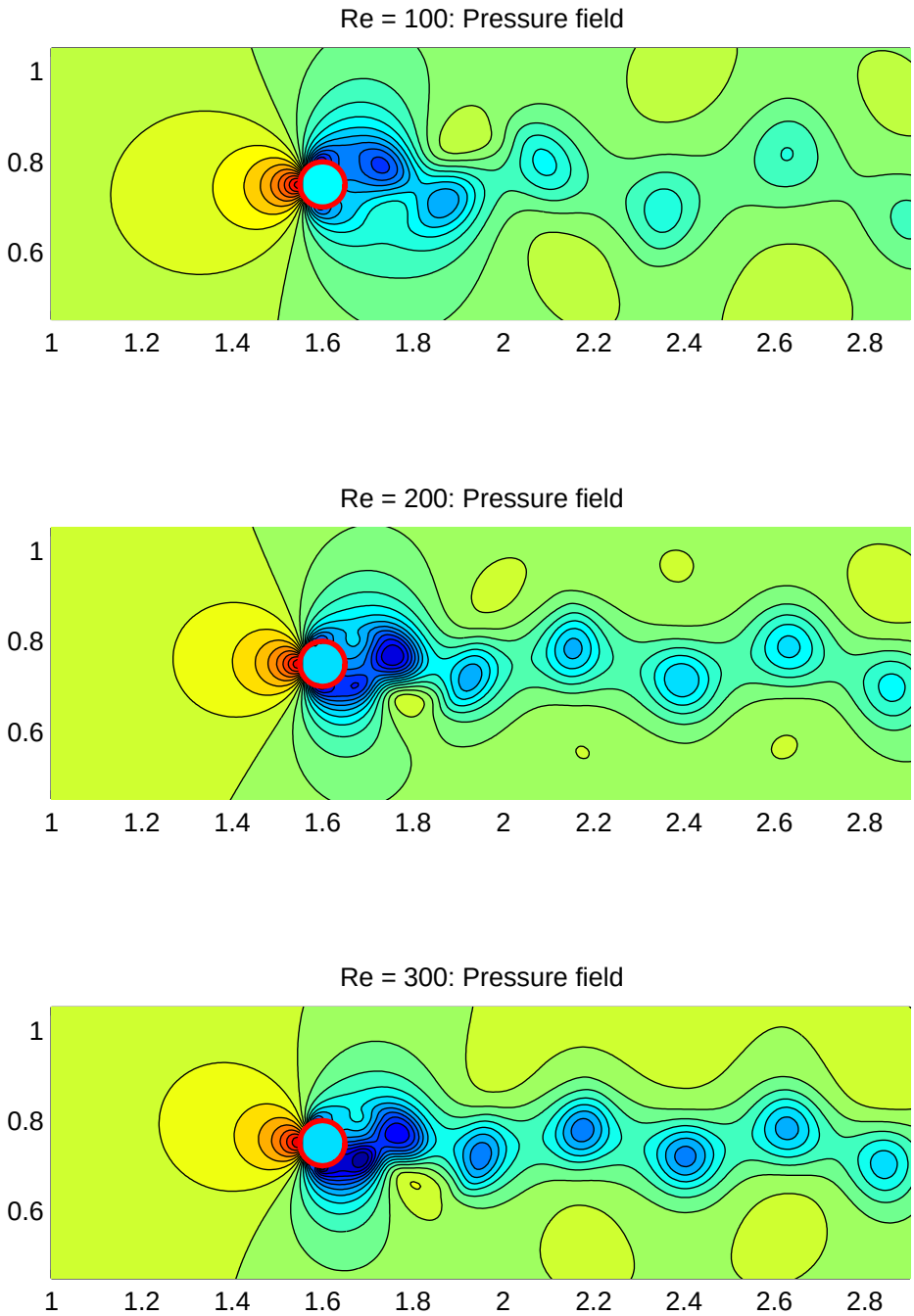


Figure 3-5: Pressure fields for Re = 100, Re = 200 and Re = 300.

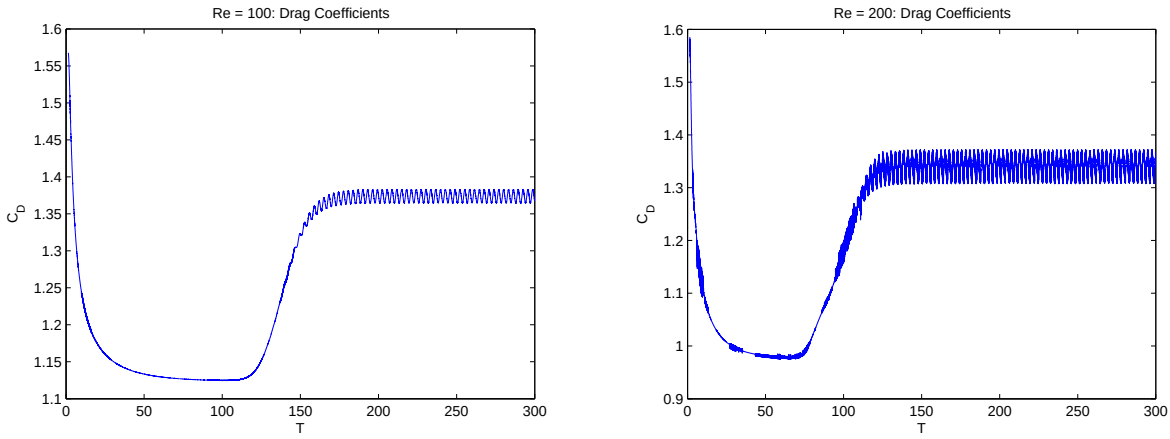


Figure 3-6: Drag Coefficients for Re = 100 and Re = 200.

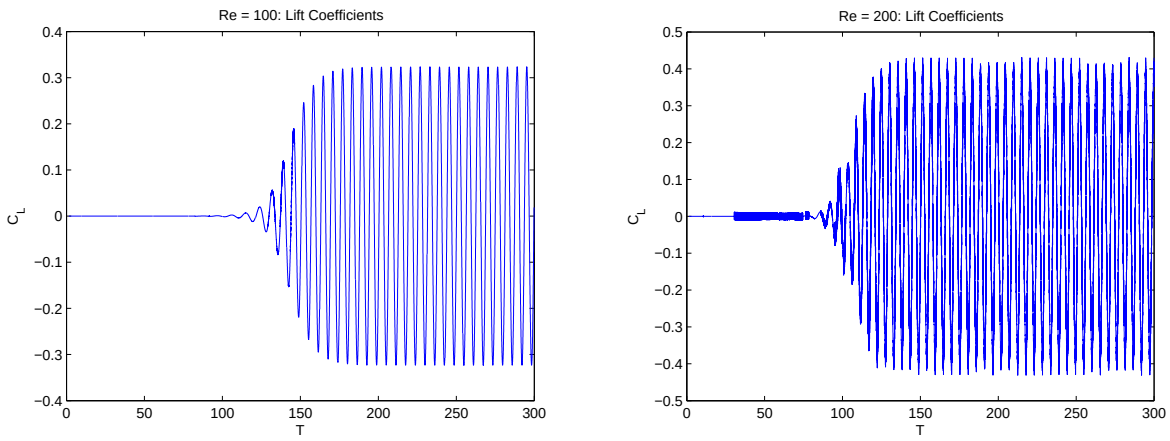


Figure 3-7: Lift Coefficients for Re = 100 and Re = 200.

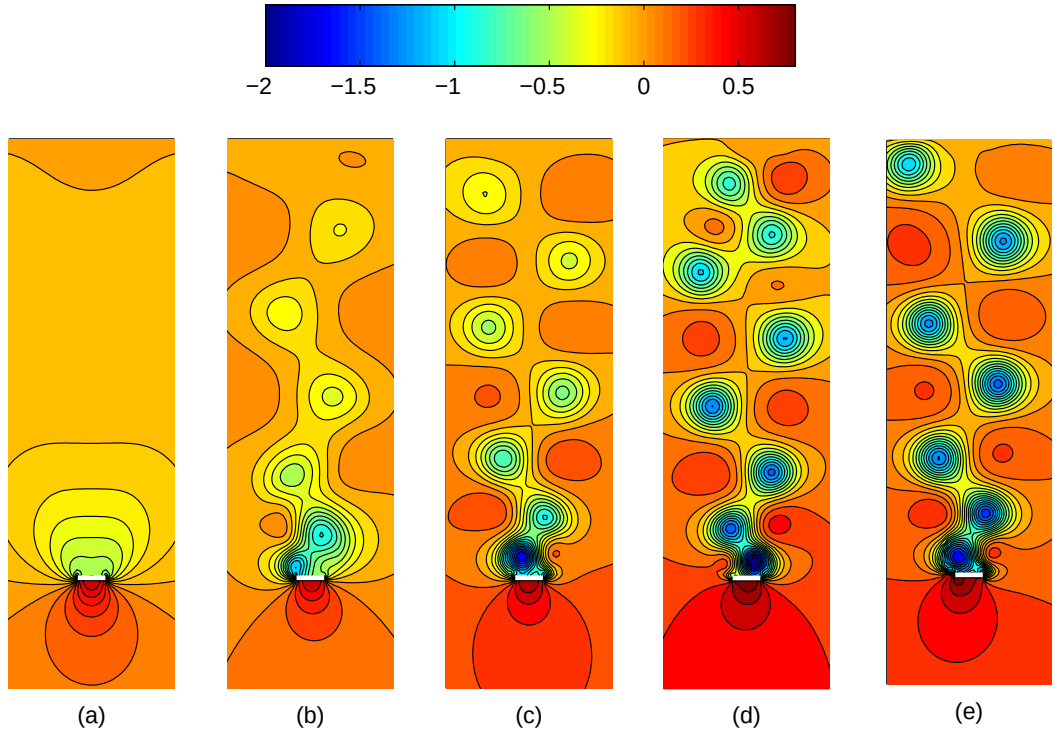


Figure 3-8: Pressure field at t = 10, $\triangle t = \triangle x/4$. (a) Re = 20, (b) Re = 50, (c) Re = 100, (d) Re = 1000, (e) Re = 5000.

oriented in the crossflow direction and is located at $x = 1.40$. Simulations have been performed at $Re = 20$, 50, 100, 1000 and 5000 on a 256×128 computational mesh. We use 8 points to represent the flat plate. The same boundary conditions as those for the flow past a cylinder problem are applied in this problem. The pressure field plots are shown in Fig. 3-8.

3.4.4 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 ability of our algorithm to handle multiple rigid boundaries. Simulations have been performed for three cylinders and an array of seven 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. For the problem with three cylinders, Fig. 3-9 and Fig. 3-10 show the streamlines and pressure contours for Re = 100 at different time levels. The vortex shedding is not symmetric since the cylinders are not placed symmetrically. For the problem with seven cylinders which are placed symmetrically with respect to the x axis, the vortex shedding is symmetric. In Fig. 3-11, we plot the streamlines and pressure contours for the flow past the array of seven cylinders at $Re = 100$ and $t = 10$.

3.4.5 Grooved channel flow

This example considers a Poiseuille flow between two walls, one of which has grooves perpendicular to the streamwise direction. This example illustrates the ability of our algorithm to handle non-smooth boundaries. The presence of the grooves complicates the geometry and an analytical solution for this flow does not exist. The flow separates and starts to recirculate in the grooves even for small Reynolds number flows. In the numerical simulation, the gap between the walls is 0.2 and the depth of the grooves is 0.6. The velocity profile at the inflow boundary is parabolic with maximum velocity $U_{max} = 1$. The viscosity is chosen to obtain a desired Reynolds number. The Reynolds number is measured based on the depth of the grooves and U_{max}. Simulations are performed at Re = 100, 500 and 3000. Fig. 3-12 shows the velocity fields for different Reynolds numbers at $t = 10$. Pressure fields at $t = 10$ are shown in Fig. 3-13.

3.4.6 Flow past a moving circular cylinder

In this example, 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. The simulation has been performed

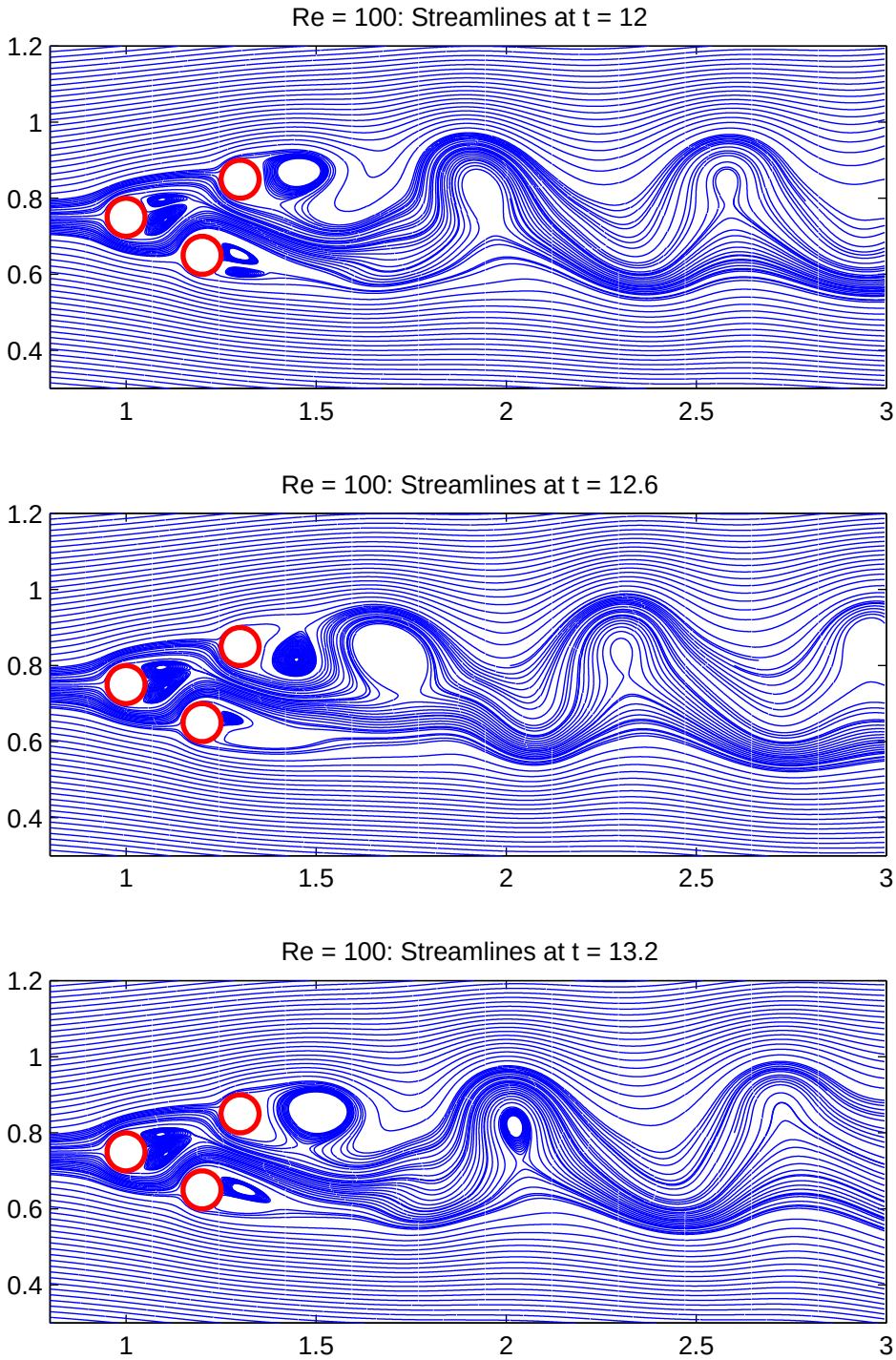


Figure 3-9: Flow past three cylinders. Streamline plots for Re = 100 at different times.

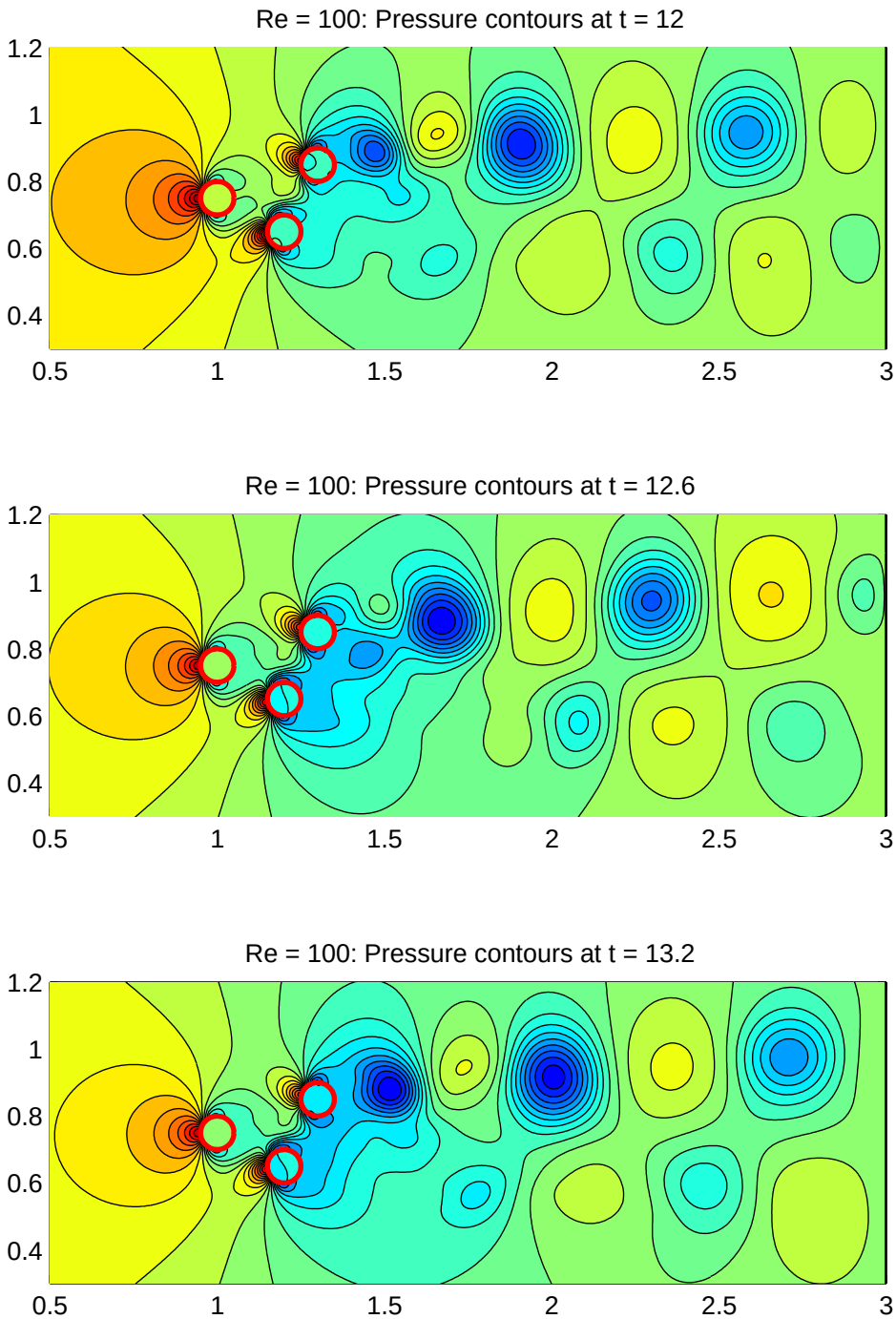


Figure 3-10: Flow past three cylinders. Pressure contours for Re = 100 at different times.

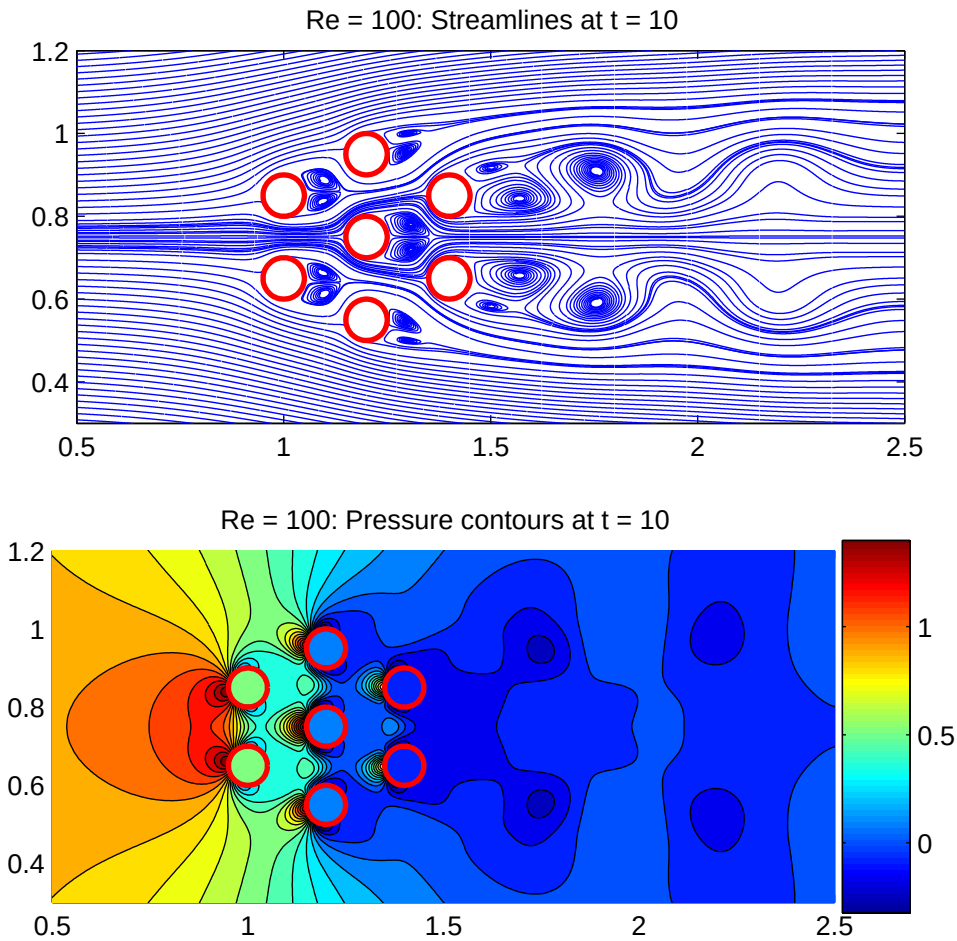


Figure 3-11: Flow past seven cylinders. Streamlines and pressure contours at Re = 100.

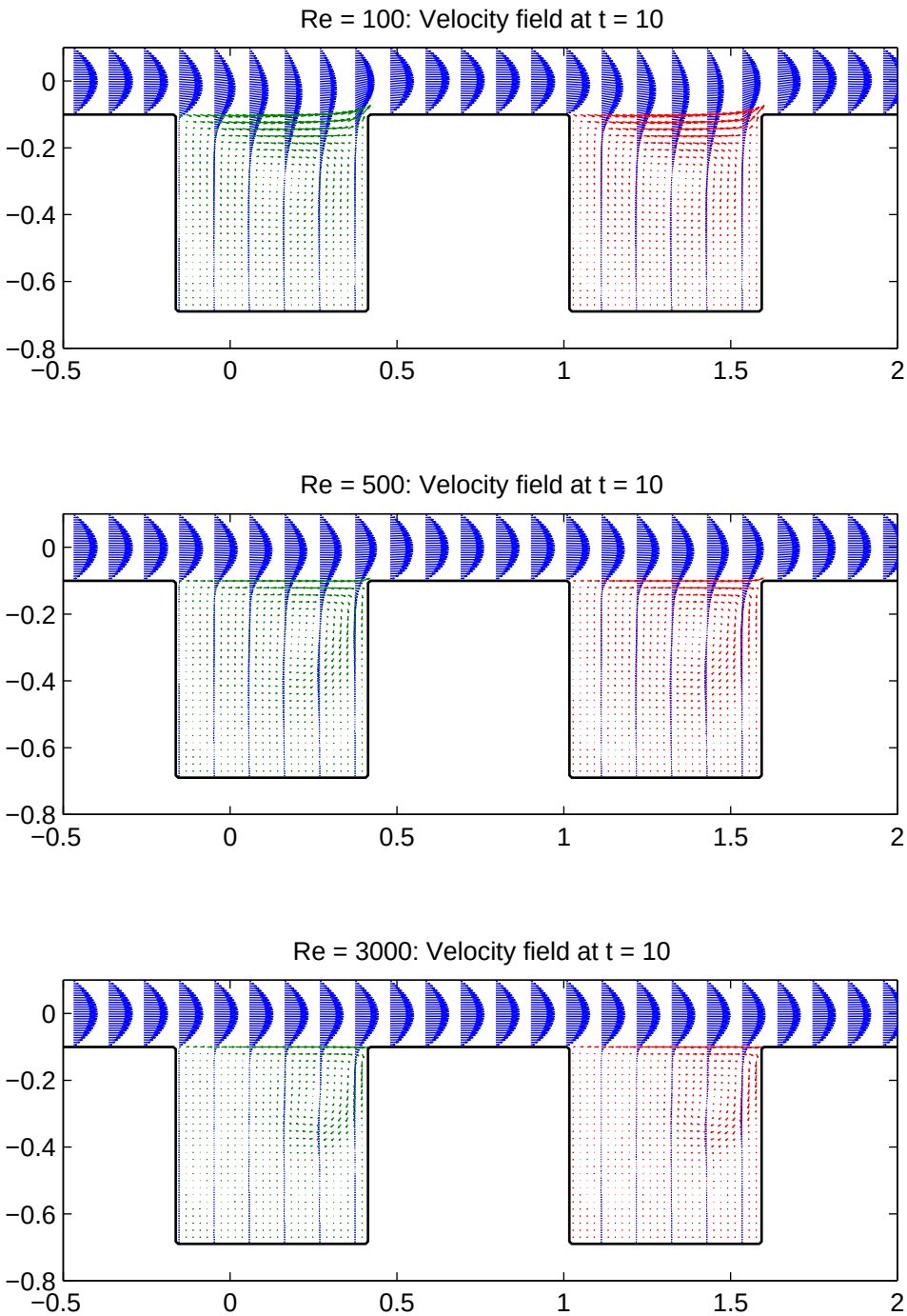


Figure 3-12: Velocity fields for Re = 100, Re = 500 and Re = 3000.

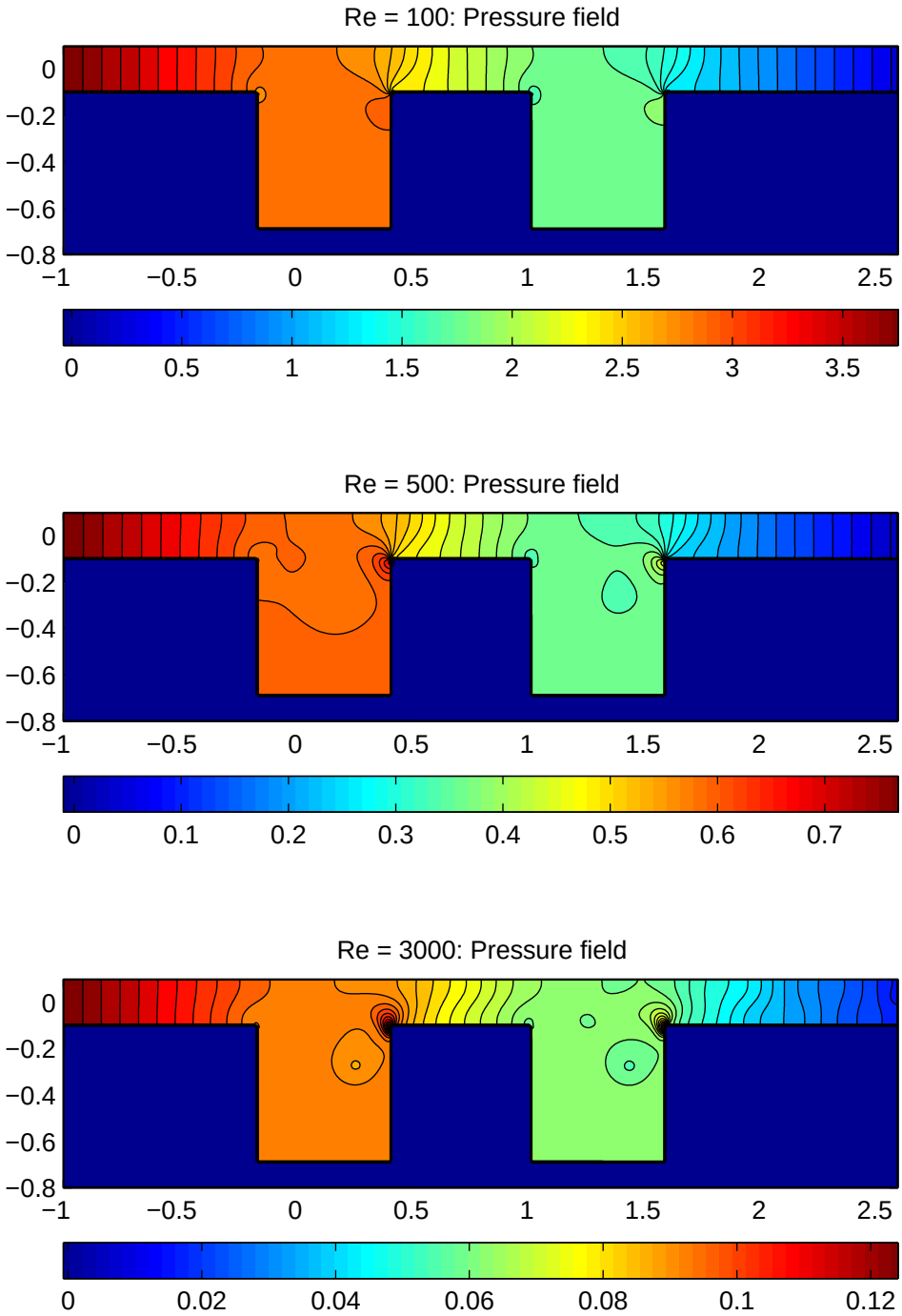


Figure 3-13: Pressure fields for Re = 100, Re = 500 and Re = 3000.

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

Table 3.6: Summary results for moving cylinder at Re = 40, compared against stationary cylinder at steady state.

for Re = 40.

In this example, to solve for the singular force at the moving boundary we do not generate a system of equations explicitly. Instead we solve for the force at the boundary iteratively using the GMRES algorithm. Since the system of equations is singular, the convergence rate of the GMRES algorithm is slow. However, we can use the incremental condition estimation (ICE) [8] to monitor the conditioning of the upper Hessenberg matrix and stop the iteration when the condition number 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 typically stop the GMRES iterative process after 2-5 iterations.

Figure 3-14 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 3-15 shows the streamlines plot at the same time level. Table 3.6 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. First, as the cylinder passing through the underlying grid, the topology changes and this may cause the noise in the force at the moving boundary. Second, 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.

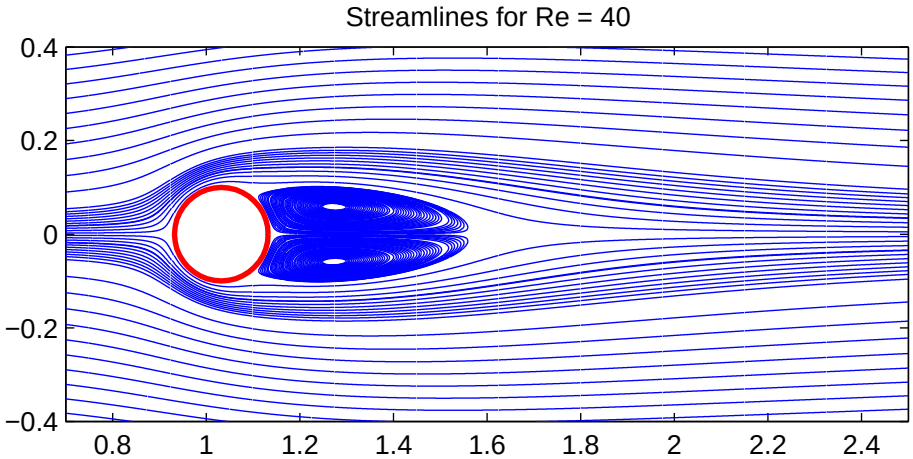


Figure 3-14: Streamlines for moving cylinder at Re = 40 in the frame of reference attached to the moving cylinder when the wake behind the cylinder is fully developed.

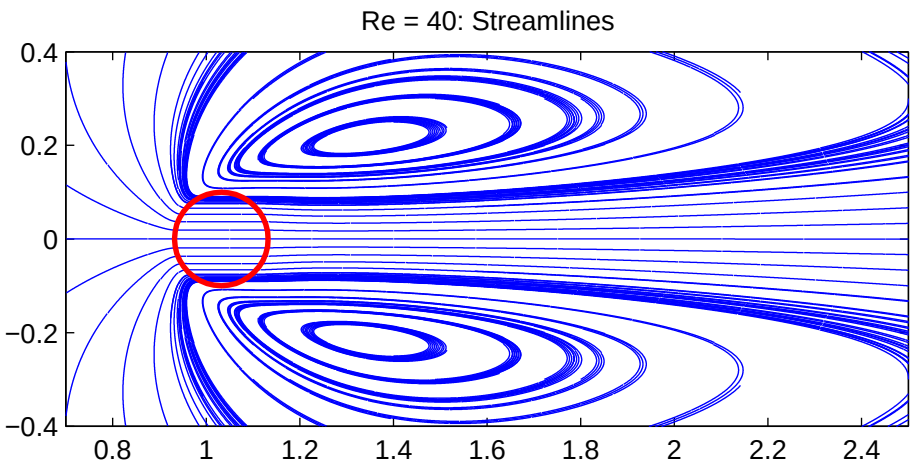


Figure 3-15: Streamlines for moving cylinder at Re = 40.

Chapter 4

An Immersed Interface Method for solving viscous, incompressible flows involving rigid and flexible boundaries

4.1 Introduction

This chapter considers the immersed interface method for the incompressible Navier-Stokes equations in general domains involving immersed flexible and rigid boundaries. The algorithm presented in this chapter is a combination of the algorithms introduced in the previous two chapters. This combination allows us to develop a robust algorithm for simulating the motion of the flexible interfaces in complicated irregular domains. Currently, most of the Cartesian grid methods can only handle flexible boundaries and the rigid boundary is required to be aligned with the computational grid [3, 46]. Agresar et al. [3] use a front-tracking method [57] and adaptive mesh refinement to perform simulation of a common cell-mechanics experiment (a cell-entry micropipet assay). In [46], the front-tracking method was also used to study the interactions between colloidal particles near the entrance to a cylindrical pore. In these

simulations, the micropipet and the cylindrical pore are represented by parallel walls which must align with the Cartesian grid. Thus, only vertical and horizontal rigid boundaries are allowed in [3, 46].

In our algorithm, arbitrary piecewise smooth rigid boundaries can be considered. The main advantage of our method is its ability to simulate the motion of two or more deformable boundaries in a domain with two or more immersed rigid boundaries. This capability allows us to solve for many problems and applications with complex geometry. In section 4.3, we show some numerical examples involving elastic membranes immersed in a grooved channel flow and in a flow with a constriction. The algorithm presented in this chapter can also be applied to the simulation of biological flow problems such as the flows in mechanical filters for biomolecule separation and the deformation of a cell in a single cell-trap. The process of cell separation involves complex fluid-structure interaction and deformation of cells in the mechanical cell filters which have complicated geometry. In our algorithm, a cell is modelled as a two-dimensional fluid body enclosed by an infinitesimally thin elastic membrane and surrounded by a fluid of identical properties. The force strength exerted by the elastic membrane is given as,

$$\boldsymbol{f}(s,t) = \frac{\partial}{\partial s}(T(s,t)\boldsymbol{\tau}(s,t)) + \sigma\frac{\partial^2 \boldsymbol{X}}{\partial s^2} \ , \tag{4.1}$$

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

$$T(s,t) = T_0\left(\left|\frac{\partial \boldsymbol{X}(s,t)}{\partial s_0}\right| - 1\right) \tag{4.2}$$

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

$$\boldsymbol{\tau}(s,t) = \frac{\partial \boldsymbol{X}}{\partial s}\Big/\left|\frac{\partial \boldsymbol{X}}{\partial s}\right| \ . \tag{4.3}$$

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 undeformed 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.

Rigid boundaries are modelled as immersed interfaces whose motions are prescribed. For static rigid boundaries, the prescribed velocity is set to zero. Singular forces need to be imposed at the rigid boundaries to enforce the no-slip boundary conditions. The singular forces at the rigid boundaries are obtained by solving the following small system of equations

$$\boldsymbol{A}\boldsymbol{f} = \boldsymbol{U}_p - \boldsymbol{U}_k^0 \tag{4.4}$$

where $\boldsymbol{U}_p$ is the prescribed velocity. This system of equations has been derived in section 3.2.

4.2 Implementation

The implementation for the algorithm presented in this chapter is a combination of the basic implementation for the algorithms introduced in the previous two chapters. Basically, flexible boundaries and rigid boundaries are represented by a number of control points. We use two sets of control points. One set is used to represent the flexible boundaries and the other represents the immersed rigid boundaries. The singular force at the flexible boundaries is computed based on the configuration of the flexible boundaries. The location of the flexible boundaries is advanced in time in an implicit manner. The BFGS method which is a quasi-Newton method is employed to solve the non-linear system of equations (2.61) iteratively to calculate the location of the flexible boundaries. In each iteration of the BFGS method, we need to solve the system of equations (4.4) for the singular force at the rigid boundaries to enforce the no-slip boundary conditions. This is necessary because the velocity field and pressure field are updated at every iterations of the BFGS method.

In summary, given the location of the flexible boundaries, $\boldsymbol{X}^n$, the singular force at the rigid boundaries, $\boldsymbol{f}^n$, the velocity field, $\boldsymbol{u}^n$ and the pressure field $p^{n-1/2}$, the process of computing the new velocity field $\boldsymbol{u}^{n+1}$ that satisfies the no-slip boundary conditions at the rigid boundaries, pressure field $p^{n+1/2}$ and the location of the flexible boundaries $\boldsymbol{X}^{n+1}$ can be summarized as follows:

Step 1: Set $k := 0$ and make an initial guess for $\boldsymbol{X}^{n+1}$, i.e. $\boldsymbol{X}^{(0)}$ as

$$\boldsymbol{X}^{(0)} = 2\boldsymbol{X^n} - \boldsymbol{X^{n-1}}$$

Step 2:

- Compute the force strength at the flexible boundaries using expression (4.1). Interpolate the force strength using cubic splines.
- Compute the force strength at the rigid boundaries to enforce the no-slip boundary conditions.
  - Calculate the right-hand side vector $\boldsymbol{U}_p - \boldsymbol{U}_k^0$ of (4.4).
  - Solve the small system of equations (4.4) to obtain the singular force at the rigid boundaries.
- Compute the jump conditions for pressure and velocity.

Step 3:

- Compute the appropriate correction terms for spatial derivatives and temporal derivatives as described in section 2.3.4.
- Employ the projection method described in section 2.3.3 to update the velocity $\boldsymbol{u}^{n+1}$ and pressure field $p^{n+1/2}$.
- Compute the velocity at the control points, $\boldsymbol{u}^{n+1}(\boldsymbol{X}^{(k)})$ by interpolating from the velocity at the surrounding grid points.

Step 4:

- Evaluate

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

- If $\|g^{(k)}\| < \epsilon$ then $\boldsymbol{X}^{n+1} = \boldsymbol{X}^{(k)}$ and stop the iteration. Otherwise, update $\boldsymbol{X}^{(k+1)}$ and the inverse Jacobian matrix B_{k+1} using BFGS algorithm mentioned in section 2.4. Set $k := k + 1$ and go to step 2.

Our implementation prohibits the intersection between two flexible boundaries or between a flexible boundary and rigid boundaries. This is enforced by defining a contact threshold to be a distance $1.5h$, where h is the mesh size. If a control point of a flexible boundary lies within a contact threshold of other flexible boundaries or rigid boundaries, we start to include a repulsive force into the total singular force at the flexible boundary. This repulsive force is applied in the outward normal direction to the rigid boundaries or other flexible boundaries. If the flexible membrane represents a particle with surface potential, the repulsive force can be understood as the electrostatic repulsion between two colloidal particles or between a particle and rigid boundaries. In [3], the velocities of interfacial points that, at their new location, lie within a contact threshold of other membranes or rigid boundaries are adjusted using the following rules: (i) if a point gets within a contact threshold of a rigid boundary, the component of its velocity normal to the boundary is set to zero; (ii) if two points (on separate membranes) get within the contact threshold of each other, the velocity of both points is set to the average of the two. In most problems, these velocity adjustments are typically very small and the errors introduced decrease with the size of the mesh and the size of the timestep. In cases in which many velocity adjustments are performed, these adjustments might significantly alter the volume of the bodies. A repulsive force was suggested to include into the total surface force to replace the velocity adjustment. In our algorithm, the expression for the repulsive force at a control point is

$$|\boldsymbol{f}_R(r)| = \begin{cases} C\left[1 - \left(\dfrac{r}{1.5h}\right)^n\right], & r \leq 1.5h \\ 0, & \text{otherwise,} \end{cases} \tag{4.5}$$

where r is a separation distance between a flexible membrane and other flexible membranes or rigid boundaries, C is a positive constant and n is a power index. It is not

very clear how to choose effective values for C and n. It can be construed that C and n are functions of the properties of the surface material but this topic is outside the scope of the present study. In our numerical experiments, a typical n is chosen within $[2, 4]$ and the constant C is chosen to have the same order of magnitude as the current force at the control point under consideration. In order to avoid a kink which may occur when adding the repulsive force to the singular force at the membrane, we distribute the repulsive force to the nearby control points of the same membrane using a Gaussian normal representation of the discrete delta function. Typically, we distribute the repulsive force to five control points including the control point under consideration and its four closest neighbors on the same membrane.

4.3 Numerical results

This section presents some numerical examples involving elastic membranes immersed in a grooved channel flow and in a flow through a constriction. These examples show the ability of handling rigid boundaries and flexible interfaces simultaneously for our algorithm. In addition, complex interactions between the fluid and flexible boundaries, interactions between flexible boundary and rigid boundary and interactions between multiple flexible boundaries are naturally considered.

4.3.1 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. Depending on the flow, the elastic membrane rotates inside the groove or the flow moves it out of the groove depending on some parameters such as the initial 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 4-1 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 $[0, 1.5] \times$

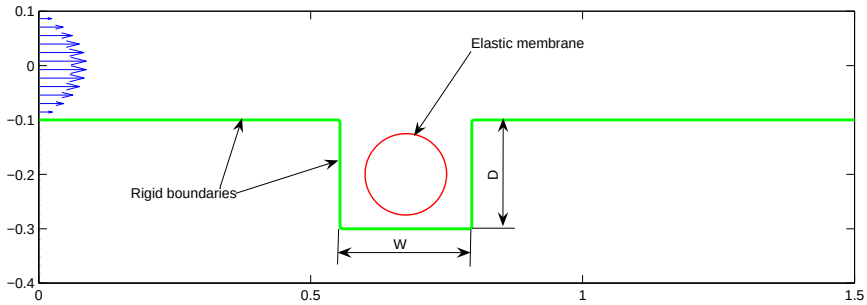


Figure 4-1: Initial position of an elastic membrane in the simulation of elastic membrane in a groove.

$[-0.4, 0.1]$, 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 undeformed 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 4-2 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 rotates inside the groove under the same flow condition. Figure 4-3 shows the positions of the elastic membrane inside the groove and velocity fields at different time. A solid circle on the interface corresponds to a material point and shows 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 4-4 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.

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 4-5 shows the rotation of the membrane inside the smaller groove with the width W of 0.2.

4.3.2 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 4-6 illustrates the geometry of the constriction and the initial position of a single membrane in front of the constriction. In all the simulations presented in this example, the 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 of $\sigma = 0$, 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 for the circular membrane whose center is located at $(0.37, 0.0)$. The elastic membrane is pre-stretched from the undeformed state with a diameter of 0.12. Two aspect ratios (ratio of the membrane

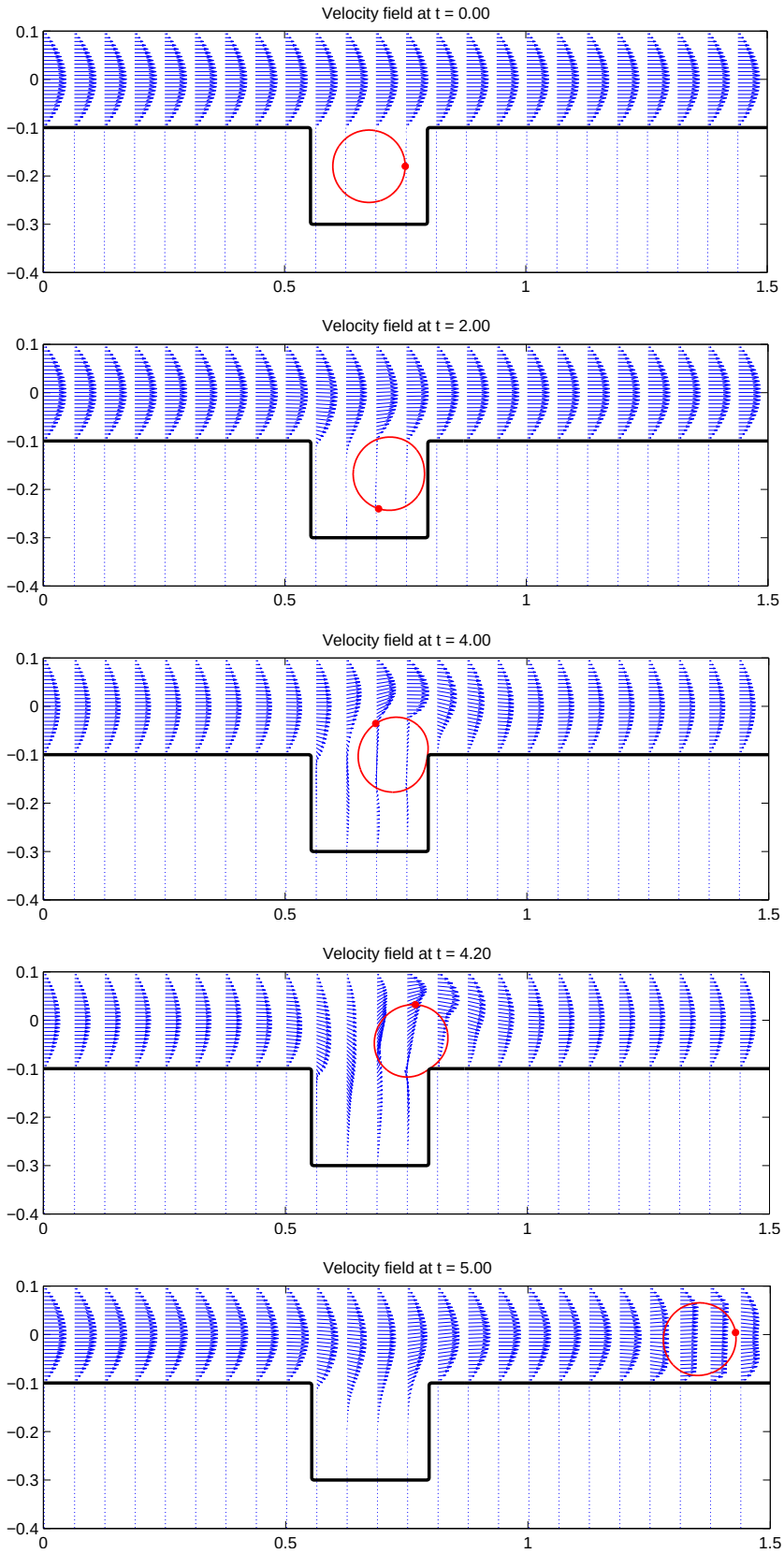


Figure 4-2: Positions of the elastic membrane and velocity fields at different times. The fluid flow moves the membrane out of the groove.

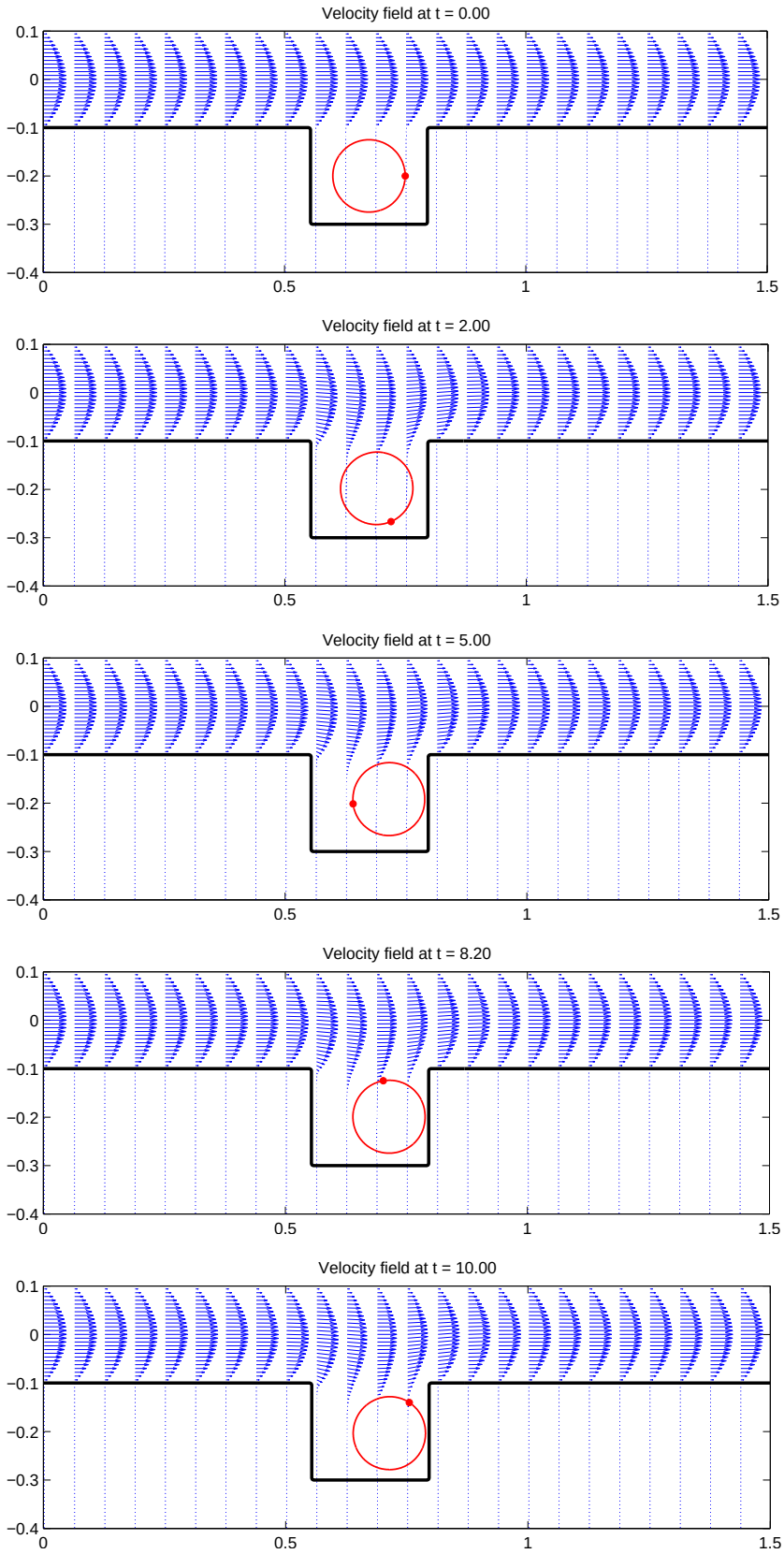


Figure 4-3: Positions of the elastic membrane and velocity fields at different times. The membrane rotates inside the groove.

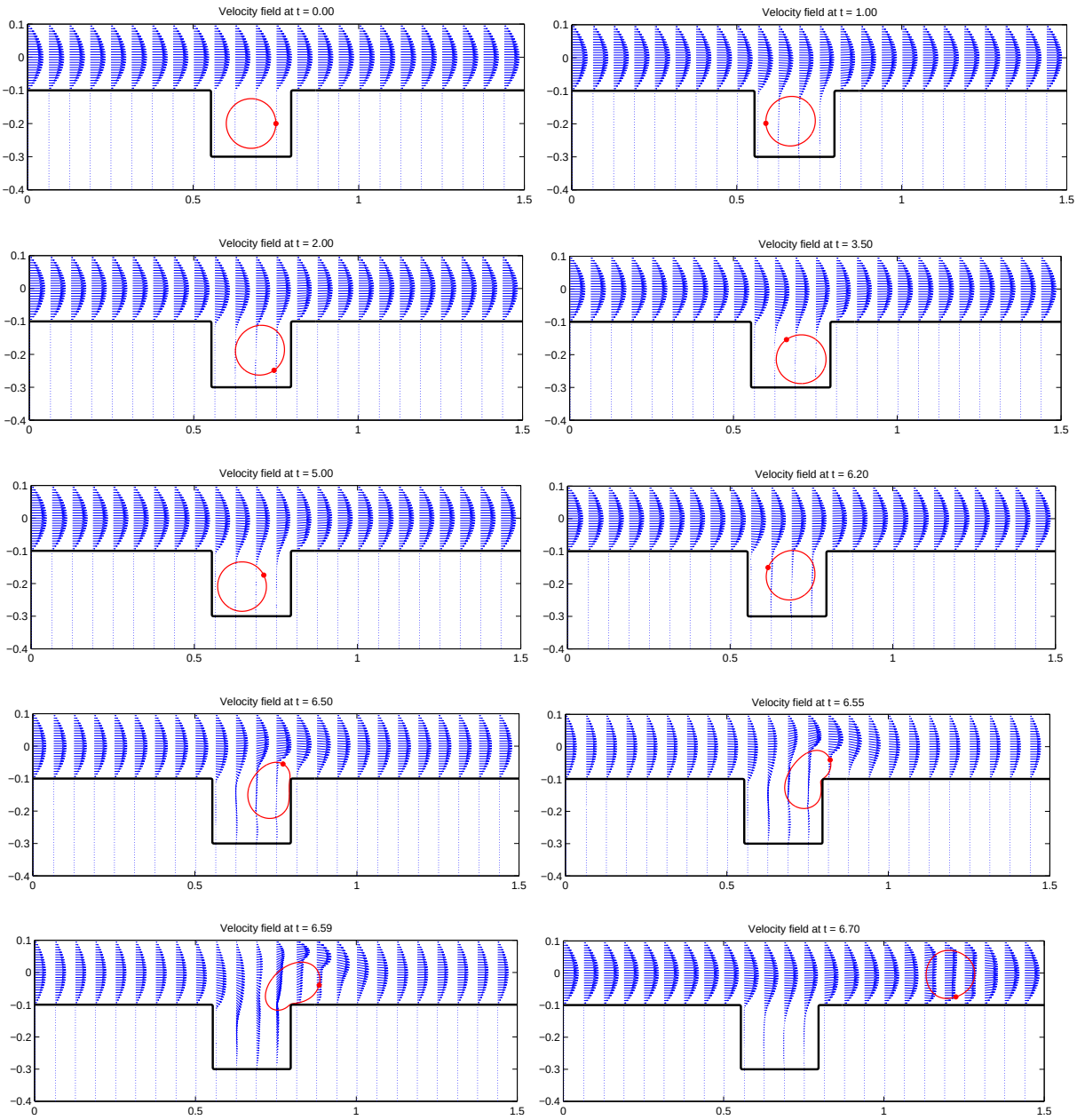


Figure 4-4: Positions of the elastic membrane and velocity fields at different times. The high flow rate moves the membrane out of the groove even though the membrane initially lies deep inside the groove.

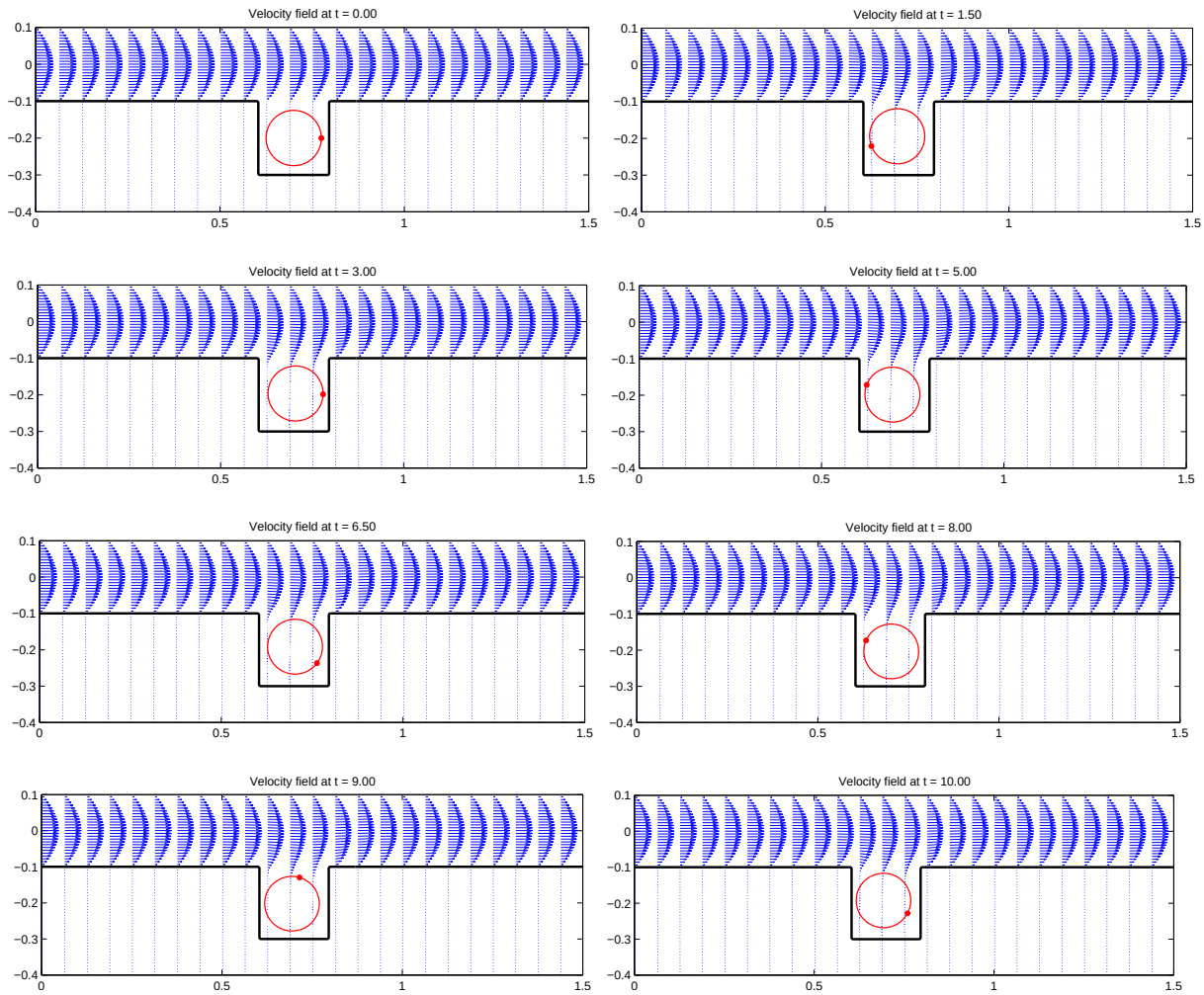


Figure 4-5: Positions of the elastic membrane and velocity fields at different times. With a small groove, the membrane only rotates inside the groove even with the high flow rate.

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 4-7 shows the locations of the elastic membrane and the corresponding velocity fields at different times with an aspect ratio of 1.3. The positions of the membrane squeezing through the smaller constriction are shown in Fig. 4-8. Fig. 4-8 shows that it takes a longer time for the membrane to squeeze through a smaller constriction.

We also performed a simulation for the motion of three membranes flowing through a constriction with an aspect ratio of 0.72. The three membranes have the same stiffness constant T_0 and diameter of 0.1. The geometry of the computational domain and the initial position of the membranes are illustrated in Fig. 4-9. Simulations have been performed for $T_0 = 4$ and $T_0 = 8$ at $Re = 5$. The Reynolds number is calculated based on the membrane diameter and the maximum far-field velocity U_{max}. Figure 4-10 and Figure 4-11 show the positions of the membranes and the corresponding velocity fields at different time for $T_0 = 4$ and $T_0 = 8$, respectively. From these figures, it can be seen that the membranes with smaller stiffness constants pass through the constriction faster than the membranes with higher stiffness constants. This happens because soft membranes can adapt to the change in geometry of the rigid boundaries better than stiff membranes.

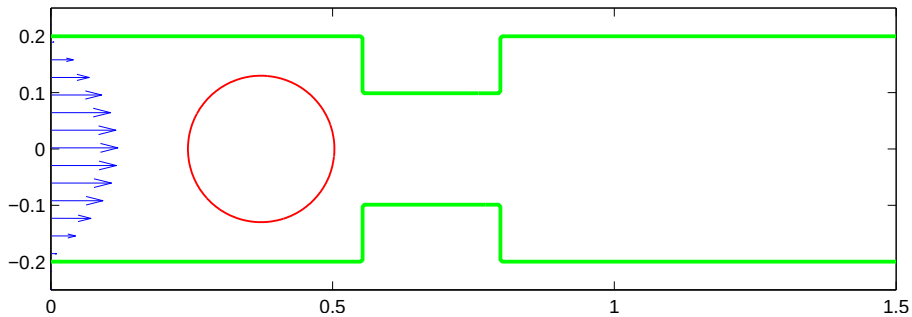


Figure 4-6: Initial position of an elastic membrane in the simulation of an elastic membrane squeezing through a constriction.

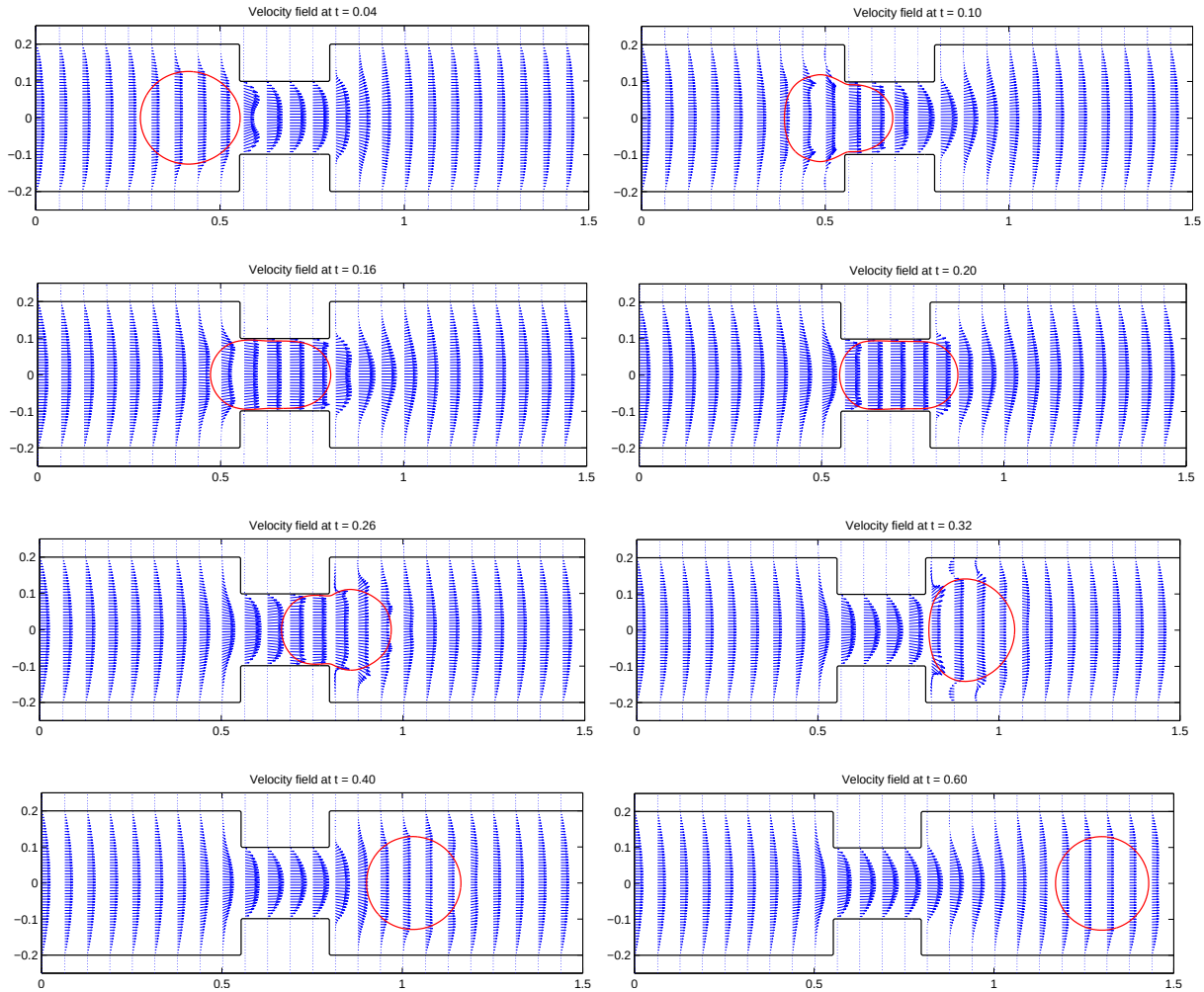


Figure 4-7: A single elastic membrane with stiffness constant $T_0 = 2$ squeezes through a constriction with aspect ratio of 1.3.

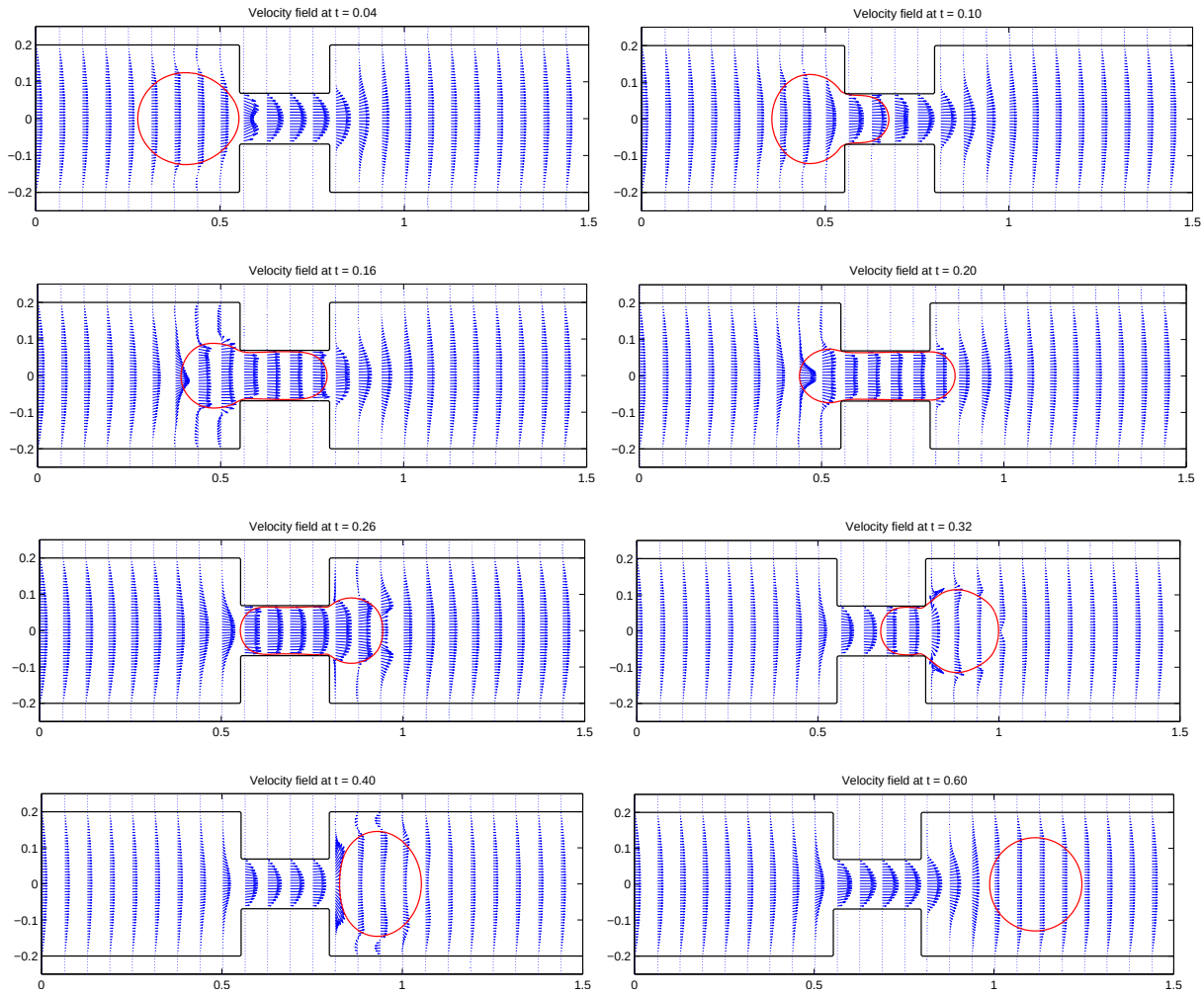


Figure 4-8: A single elastic membrane with stiffness constant $T_0 = 2$ squeezes through a constriction with aspect ratio of 1.88.

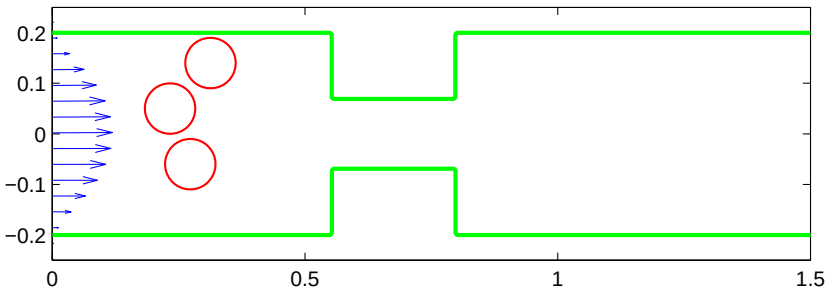


Figure 4-9: Computational domain for studying the interaction between several elastic membranes at the entrance to a constriction. Initial positions of elastic membranes in the 3-membrane simulations.

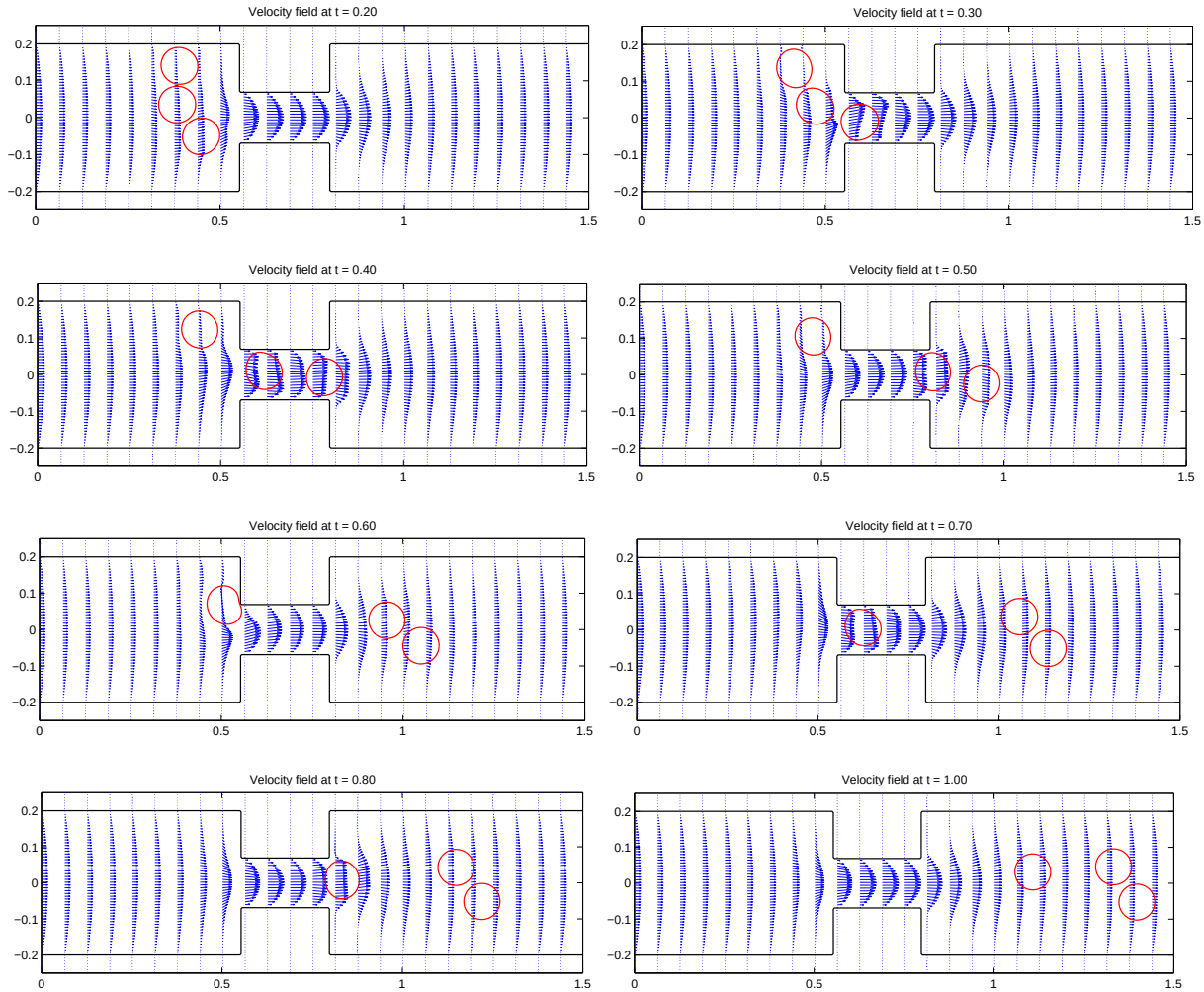


Figure 4-10: The positions of the elastic membranes and velocity fields at different times. Simulations have been performed for Re = 5, stiffness constant $T_0 = 4$ and $\Delta t = \Delta x/7.5$.

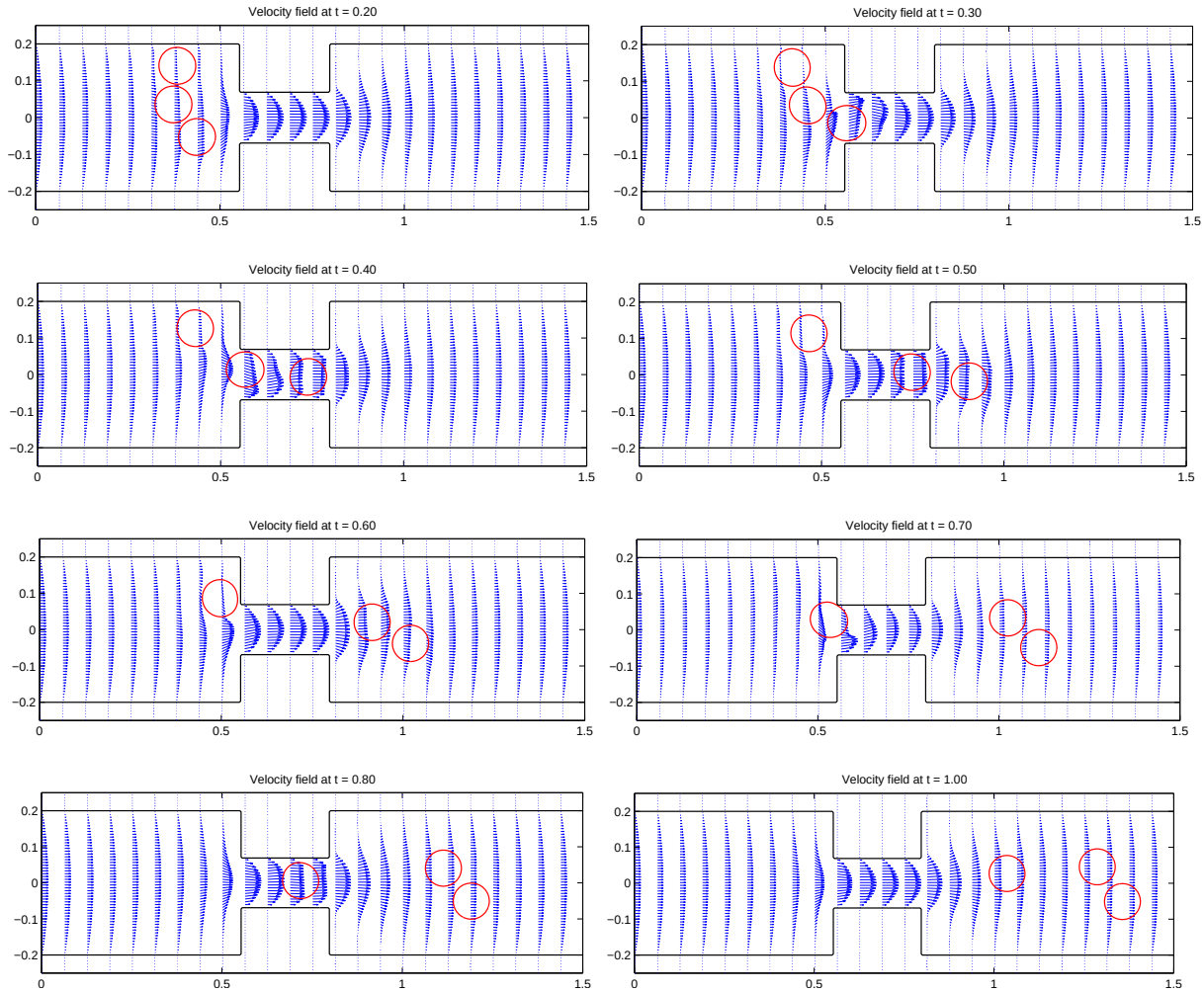


Figure 4-11: The positions of the elastic membranes and velocity fields at different times. Simulations have been performed for Re $= 5$, stiffness constant $T_0 = 8$ and $\Delta t = \Delta x/15$.

Chapter 5

Conclusions and future work

5.1 Conclusions

In this thesis, an immersed interface method for solving viscous, incompressible flows involving flexible interfaces and rigid boundaries has been described in detail. The contributions of this thesis include the following:

- An immersed interface method is developed for solving viscous, incompressible flows involving moving flexible interfaces. An entire singular force at the interface is incorporated into the jump conditions of the solutions and their derivatives. Our algorithm can capture not only the jump in pressure but also the jumps in the derivatives of pressure and velocity. The jump in the time derivative of the velocity is also taken into account to avoid errors which could accumulate in time.

- 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.

- An immersed interface method has been developed 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 for rotational flow and elastic band problems to validate our algorithm for flexible interfaces. Second order accuracy has been demonstrated thorough several computational examples. We also 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 boundaries are also considered to show the flexibility of our algorithm. Finally, some numerical results have been shown in Chapter 4 for problems involving motions of membranes in a grooved channel and through a constriction.

5.2 Future work

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.

In biological systems, viscosity is normally different at both sides of the interface. Therefore, we would like to extend our code to solve for problems with different viscosities. In this case, the jumps in pressure and its derivatives are coupled with the jumps in velocity derivatives. Hence a more careful study needs to be done to decouple these jumps in order to compute the correction terms.

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 one of the main motivations for carrying out the work presented in this thesis 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 [46].

Appendix A

Jump conditions across an immersed interface

The incompressible Navier-Stokes equations in a 2-dimensional bounded domain Ω that contains a material interface Γ(t) can be written as

$$\frac{D\boldsymbol{u}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \boldsymbol{F} \tag{A.1}$$

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

where

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + (\boldsymbol{u} \cdot \nabla) \tag{A.3}$$

is the material derivative and

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + \mu(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) \tag{A.4}$$

is the fluid stress tensor. The effect of the material interface $\Gamma(t)$ immersed in the fluid results in a singular force $\boldsymbol{F}$ which has the form

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

where $\boldsymbol{X}(s,t)$ is the arc-length parametrization of $\Gamma(t)$, s is the arc-length and $\boldsymbol{f}(s,t)$ is the force strength.

When the singular force is 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. And let (ξ,η) be the rectangular coordinates associated with the directions of $\boldsymbol{n}$ and $\boldsymbol{\tau}$, respectively. Then in the neighborhood of $(\xi,\eta)=(0,0)$ the interface Γ can be represented by $\xi=\chi(\eta)$ which satisfies $\chi(0)=0$, $\chi'(0)=0$ and $\chi''(0)=\kappa$, the curvature of Γ at $(0,0)$. 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

$$[\boldsymbol{u}]=\boldsymbol{0}, \qquad [\mu\boldsymbol{u}_\xi]=-f_2\boldsymbol{\tau}, \qquad [\boldsymbol{u}_\eta]=\boldsymbol{0}\ , \tag{A.6}$$

$$[p]=f_1, \qquad [p_\xi]=\frac{\partial f_2}{\partial s}, \qquad [p_\eta]=\frac{\partial f_1}{\partial s}\ , \tag{A.7}$$

$$\begin{aligned}
&[\mu\boldsymbol{u}_{\eta\eta}]=\kappa f_2\boldsymbol{\tau}, \qquad [\mu\boldsymbol{u}_{\xi\eta}]=-\frac{\partial f_2}{\partial\eta}\boldsymbol{\tau}-\kappa f_2\boldsymbol{n},\\
&[\mu\boldsymbol{u}_{\xi\xi}]=-\left[\mu\boldsymbol{u}_{\eta\eta}\right]+[p_\xi]\boldsymbol{n}+[p_\eta]\boldsymbol{\tau}+[\boldsymbol{u}_\xi]\boldsymbol{u}\cdot\boldsymbol{n}\ ,
\end{aligned} \tag{A.8}$$

$$\begin{aligned}
&[p_{\eta\eta}]=\frac{\partial^2 f_1}{\partial\eta^2}-\kappa[p_\xi], \quad [p_{\xi\eta}]=\frac{\partial^2 f_2}{\partial\eta^2}+\kappa[p_\eta],\\
&[p_{\xi\xi}]=-\left[\nabla\cdot(\boldsymbol{u}\cdot\nabla\boldsymbol{u})\right]-[p_{\eta\eta}]\ .
\end{aligned} \tag{A.9}$$

The jump, $[\,\cdot\,]$, denotes the difference between the value of its argument from the outside and that from the inside of the interface. The first equality of (A.6) means that the velocity is continuous across the interface. Differentiating $[\boldsymbol{u}]=\boldsymbol{0}$ along the tangential direction of Γ, i.e.,

$$\boldsymbol{0}=\frac{d[\boldsymbol{u}]}{d\eta}=\frac{\partial[\boldsymbol{u}]}{\partial\xi}\chi'(\eta)+\frac{\partial[\boldsymbol{u}]}{\partial\eta} \tag{A.10}$$

and using $\chi'(0)=0$, we get the third equality of the (A.6) which means that the tangential derivatives of the velocity are continuous.

The second equality of (A.6) and the first equality of (A.7) can be derived by balancing the force at the interface, i.e,

$$\boldsymbol{f} + \boldsymbol{t}^+ + \boldsymbol{t}^- = 0 \ , \tag{A.11}$$

where $\boldsymbol{t}^+ = \boldsymbol{\sigma}^+ \cdot \boldsymbol{n}^+$ and $\boldsymbol{t}^- = \boldsymbol{\sigma}^- \cdot \boldsymbol{n}^-$ are the fluid traction. Since $\boldsymbol{n}^+ = -\boldsymbol{n}^- = \boldsymbol{n}$, equation (A.11) can be rewritten as

$$\boldsymbol{f} + [\boldsymbol{\sigma}] \cdot \boldsymbol{n} = 0 \ . \tag{A.12}$$

Taking the normal and tangential components of (A.12) and using the jump conditions $[\boldsymbol{u}_\eta] = \boldsymbol{0}$ and $[\nabla \cdot \boldsymbol{u}] = 0$, we can easily derive $[p] = \boldsymbol{f} \cdot \boldsymbol{n}$ and $[\mu \boldsymbol{u}_\xi] = -(\boldsymbol{f} \cdot \boldsymbol{\tau})\boldsymbol{\tau}$. The second equality of (A.7) can be derived by applying the divergence operator to the momentum equation (A.1), see [36, 33]. Differentiating $[p] = [f_1]$ along the tangential direction of Γ and using $\chi'(0) = 0$, we get the third equality of (A.7). Differentiating (A.10) along the tangential direction of Γ, i.e.,

$$[\boldsymbol{u}_{\xi\xi}] \left(\chi'(\eta)\right)^2 + 2[\boldsymbol{u}_{\xi\eta}]\chi'(\eta) + [\boldsymbol{u}_\xi]\chi''(\eta) + [\boldsymbol{u}_{\eta\eta}] = \boldsymbol{0}$$

and using $\chi'(0) = 0$ and $\chi''(0) = \kappa$, we get the first equality of (A.8). Taking the derivative of the second equality of (A.6) with respect to the tangential direction and using $\frac{\partial \boldsymbol{\tau}}{\partial \eta} = \kappa \boldsymbol{n}$, we obtain the second equality of (A.8). The last equality of (A.8) can be found from the momentum equation (A.1) written in the local coordinate (ξ, η). Similarly, by differentiating $[p] = [f_1]$ along the tangential direction of Γ twice, we get the first equality of (A.9). The second equality of (A.9) is obtained by differentiating $[p_n] = \frac{\partial f_2}{\partial \eta}$ along the interface. The third equality of (A.9) is derived by applying the divergence operator to the momentum equation (A.1) in the local coordinate (ξ, η). More details on the proof of (A.6)–(A.9) can be found in [33, 35, 36].

Appendix B

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

$$\boldsymbol{U_k} = \boldsymbol{U}(\boldsymbol{X}_k) = \mathcal{B}(\boldsymbol{u}) \ , \tag{B.1}$$

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 B-1, the velocity at the control point $\boldsymbol{X}_k$ is interpolated from the velocity at four neighbor grid points as follows

$$\boldsymbol{U}_k = (1-\xi)(1-\eta)\boldsymbol{u}_1 + \boldsymbol{C}_1 + \xi(1-\eta)\boldsymbol{u}_2 + \boldsymbol{C}_2 + \xi\eta\boldsymbol{u}_3 + \boldsymbol{C}_3 + (1-\xi)\eta\boldsymbol{u}_4 + \boldsymbol{C}_4 \tag{B.2}$$

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

$$\boldsymbol{C}_1 = \begin{cases} h(1-\xi)(1-\eta)\big(\xi[\boldsymbol{u}_x] + \eta[\boldsymbol{u}_y]\big), & \boldsymbol{x}_1 \in \Omega^+ \\ 0, & \boldsymbol{x}_1 \in \Omega^-, \end{cases} \tag{B.3}$$

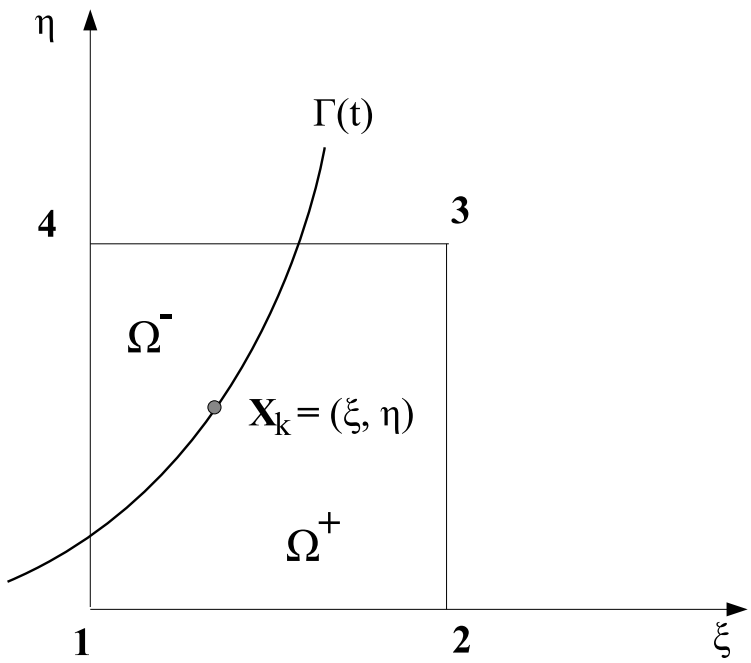


Figure B-1: Velocity at a control point is interpolated from the velocity at the four neighbor grid points using modified bilinear interpolation.

$$\boldsymbol{C}_2 = \begin{cases} -h\xi(1-\eta)\big((1-\xi)[\boldsymbol{u}_x] - \eta[\boldsymbol{u}_y]\big), & \boldsymbol{x}_2 \in \Omega^+ \\ 0, & \boldsymbol{x}_2 \in \Omega^-, \end{cases} \tag{B.4}$$

$$\boldsymbol{C}_3 = \begin{cases} -h\xi\eta\big((1-\xi)[\boldsymbol{u}_x] + (1-\eta)[\boldsymbol{u}_y]\big), & \boldsymbol{x}_3 \in \Omega^+ \\ 0, & \boldsymbol{x}_3 \in \Omega^-, \end{cases} \tag{B.5}$$

$$\boldsymbol{C}_4 = \begin{cases} h(1-\xi)\eta\big(\xi[\boldsymbol{u}_x] - (1-\eta)[\boldsymbol{u}_y]\big), & \boldsymbol{x}_4 \in \Omega^+ \\ 0, & \boldsymbol{x}_4 \in \Omega^-. \end{cases} \tag{B.6}$$

Bibliography

[1] Y. Achdou and O. Pironneau. A fast solver for Navier-Stokes equations in the laminar regime using mortar finite element and boundary element methods. *SIAM Journal on Numerical Analysis*, 32(4):985–1016, 1995.

[2] 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/.

[3] G. Agresar, J. J. Linderman, G. Tryggvason, and K. G. Powell. An adaptive, Cartesian, front-tracking method for the motion, deformation and adhesion of circulating cells. *J. Comput. Phys.*, 143:346–380, 1998.

[4] S. Armfield and R. Street. An analysis and comparison of the time accuracy of fractional step methods for the Navier-Stokes equations on staggered grids. *Int. J. for Numerical Methods in Fluids*, 38:255–282, 2002.

[5] K. J. Bathe and H. Zhang. Finite element developments for general fluid flows with structural interactions. *International Journal for Numerical Methods in Engineering*, 60:213–232, 2004.

[6] J. B. Bell, P. Collela, and H. M. Glaz. A second order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85:257, 1989.

[7] G. Biros, L. Ying, and D. Zorin. The embedded boundary integral equation solver for the incompressible Navier-Stokes equations. Technical report, Courant Institute, New York University, 2002.

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

[9] 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.

[10] 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.

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

[12] C. G. Broyden. The convergence of a class of double rank minimization algorithms. 1. General considerations, 2. The new algorithm. *J. Inst. Math. Appl.*, 6:76–90, 222–231, 1970.

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

[14] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comput.*, 22:745, 1968.

[15] 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.

[16] 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.

[17] Y. Di, R. Li, T. Tang, and P. Zhang. Moving mesh finite element methods for the incompressible Navier-Stokes equations. *SIAM J. Sci. Comput.*, 26(3):1036–1056, 2005.

[18] 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.

[19] 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.

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

[21] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152:457–492, 1999.

[22] 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.

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

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

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

[26] L. Greengard and M. C. Kropinski. An integral approach to the incompressible Navier-Stokes equations in two dimensions. *SIAM Journal on Scientific Computing*, 20(1):318–336, 1998.

[27] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73:325–348, 1987.

[28] M. Kang, R. Fedkiw, and X. D. Liu. A boundary condition capturing method for multiphase incompressible flow. *J. Sci. Comput.*, 15:323–360, 2000.

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

[30] 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.

[31] 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.

[32] 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.

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

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

[35] 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.

[36] 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.

[37] Z. Li. Immersed interface method for moving interface problem. *Numerical Algorithms*, 14:269–293, 1997.

[38] 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.

[39] 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.

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

[41] M. Liu, Y. X. Ren, and H. Zhang. A class of fully second order accurate projection methods for solving the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 200:325–346, 2004.

[42] A. Mayo. The fast solution of Poisson's and the biharmonic equations on irregular regions. *SIAM Journal on Numerical Analysis*, 21(2):285–299, 1984.

[43] D. Nguyen, R. Fedkiw, and M. Kang. A boundary condition capturing method for incompressible flame discontinuities. *J. Comput. Phys.*, 172:71–98, 2001.

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

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

[46] 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.

[47] 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.

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

[49] J. A. Sethian and A. Wiegmann. Structural boundary design via level set and immersed interface methods. *J. Comput. Phys.*, 163:489–528, 2000.

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

[51] 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.

[52] J. Stoer and R. Bulirsch. *Introduction to numerical analysis.* Springer-Verlag, third edition, 2002.

[53] E. Y. Tau. A second-order projection method for the incompressible Navier-Stokes equations in arbitrary domains. *J. Comput. Phys.*, 115:147–152, 1994.

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

[55] C. Tu and C. S. Peskin. Stability and instability in the computation of flows with moving immersed boundaries: a comparison of three methods. *SIAM J. Sci. Statist. Comput.*, 13:1361–1376, 1992.

[56] H. S. Udaykumar, R. Mittal, P. Rampunggoon, and A. Khanna. A sharp interface Cartesian grid method for simulating flows with complex moving boundaries. *J. Comput. Phys.*, (174):345–380, 2001.

[57] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible flows. *J. Comput. Phys.*, 100:25, 1992.

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

[59] X. Wang and W. K. Liu. Extended immersed boundary method using FEM and RKPM. *Comput. Methods Appl.Mech. Engrg.*, 193:1305–1321, 2004.

[60] A. Wiegmann and K.P. Bube. The immersed interface method for nonlinear differential equations with discontinuous coefficients and singular sources. *SIAM J. Numer. Anal.*, 35:177–200, 1998.

[61] 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.

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

[63] 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.

[64] L. Zhang, A. Gerstenberger, X. Wang, and W. K. Liu. Immersed finite element method. *Comput. Methods Appl.Mech. Engrg.*, 193:2051–2067, 2004.