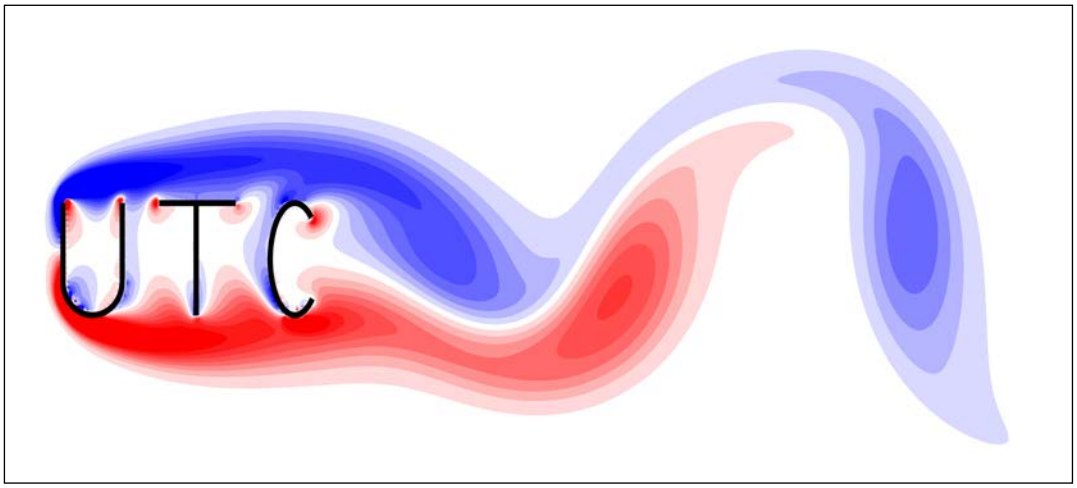
Computational fluid-structure interaction with the moving immersed boundary method

Shang-Gui Cai

Par **Shang-Gui CAI**

Computational fluid-structure interaction with the moving immersed boundary method

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



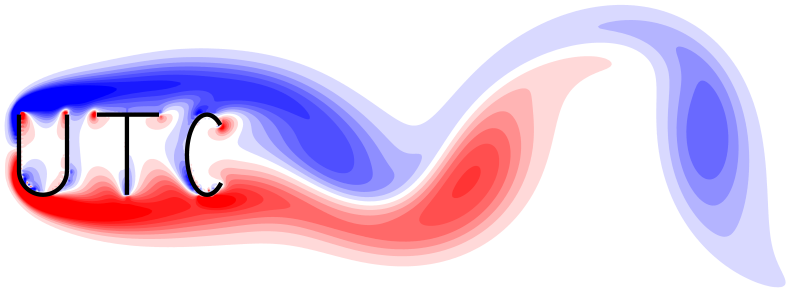
Soutenue le 30 mai 2016
Spécialité : Mécanique Avancée

D2276

Shang-Gui CAI

Computational fluid-structure interaction with the moving immersed boundary method

Thèse présentée pour l'obtention du grade de Docteur de Sorbonne Universités, Université de Technologie de Compiègne.



Membres du Jury :

Y. Hoarau	Professeur, Université de Strasbourg	Rapporteur
P. Pimenta	Professeur, University of São Paulo	Rapporteur
H. Naceur	Professeur, Université de Valenciennes, UVHC	Examinateur
J. Favier	Maître de Conférences, Aix-Marseille Université	Examinateur
A. Ibrahimbegovic	Professeur, Université de Technologie de Compiègne	Examinateur
E. Lefrançois	Professeur, Université de Technologie de Compiègne	Examinateur
P. Villon	Professeur, Université de Technologie de Compiègne	Invité
A. Ouahsine	Professeur, Université de Technologie de Compiègne	Directeur
H. Smaoui	Chargé de Recherche, HDR, UTC-CEMERA	Co-directeur

Computational fluid-structure interaction with the moving immersed boundary method

Shang-Gui CAI

Thesis submitted for the degree of Doctor

Laboratory Roberval
Sorbonne Universités, Université de Technologie de Compiègne
Compiègne, France

Defended - May 30, 2016

I would like to dedicate this thesis to my loving family.

Acknowledgements

First of all, I would like to express my profound gratitude to my supervisor Prof. Abdellatif OUAHSINE, for his continuous and patient guidance of my Ph.D. study during the past three and a half years. He has been always encouraging me and keeping me motivated along this academic journey. I am extremely thankful for the resources he has introduced to me, which has helped me move forward greatly.
My sincere thanks goes to my co-supervisor Dr. Hassan SMAOUI, who has been very supportive to me in developing numerical methods. I enjoyed stimulating discussions with him on fluid mechanics and mathematics. His immense knowledge and rigorous attitude have inspired me to pursue high-quality research.
My research experience would not have been this fruitful without the help from my laboratory members, especially, Prof. Adnan IBRAHIMBEGOVIC, Prof. Pierre VILLON, Prof. Alain RASSINEUX, Prof. Emmanuel LEFRANCOIS and Dr. Pierre FEISSEL, thanks to their precious and helpful comments on my research. I was fortunate to be involved in the research group LHN, receiving very useful suggestions and help from Dr. Philippe SERGENT, Dr. Nicolas HUYBRECHTS, Dr. Sami KAIDI, Dr. Shengcheng JI, Florian LINDE, Peng DU and Hanqi ZUO.
I also would like to acknowledge the members in the Laboratory BMBI, namely Prof. Dominique BARTHÈS-BIESEL, Dr. Anne-Virginie SALSAC and Dr. Badr KAOUI for interesting discussions on the immersed boundary method.
Special thanks goes to Prof. Yannick HOARAU (Université de Strasbourg) for his kind help in developing immersed boundary method and for the access of high performance computing services in Strasbourg. Appreciation also goes to Prof. Hermann G. MATTHIES (Technische Universität Braunschweig), Dr. Julien FAVIER (Aix-Marseille Université), Prof. Paulo PIMENTA (University of São Paulo) and Prof. Hakim NACEUR (Université de Valenciennes et du Hainaut-Cambrésis) for their valuable opinions.
I am grateful for the financial support provided by the China Scholarship Council (CSC) and the UT-INSA program to study in France. The high performance computing platform PILCAM2 at UTC is also greatly acknowledged. I thank all my friends here for making my life very enjoyable during my study.
Finally, I would like to thank my family for all unconditional love and encouragement in writing this thesis and in my life.

Abstract

In this thesis a novel non-body conforming mesh formulation is developed, called the moving immersed boundary method (MIBM), for the numerical simulation of fluid-structure interaction (FSI). The primary goal is to enable solids of complex shape to move arbitrarily in an incompressible viscous fluid, without fitting the solid boundary motion with dynamic meshes.

This novel method enforces the no-slip boundary condition exactly at the fluid-solid interface with a boundary force, without introducing any artificial constants to the rigid body formulation. As a result, large time step can be used in current method. To determine the boundary force more efficiently in case of moving boundaries, an additional moving force equation is derived and the resulting system is solved by the conjugate gradient method. The proposed method is highly portable and can be integrated into any fluid solver as a plug-in.

In the present thesis, the MIBM is implemented in the fluid solver based on the projection method. In order to obtain results of high accuracy, the rotational incremental pressure correction projection method is adopted, which is free of numerical boundary layer and is second order accurate. To accelerate the calculation of the pressure Poisson equation, the multi-grid method is employed as a preconditioner together with the conjugate gradient method as a solver. The code is further parallelized on the graphics processing unit (GPU) with the CUDA library to enjoy high performance computing.

At last, the proposed MIBM is applied to the study of two-way FSI problem. For stability and modularity reasons, a partitioned implicit scheme is selected for this strongly coupled problem. The interface matching of fluid and solid variables is realized through a fixed point iteration. To reduce the computational cost, a novel efficient coupling scheme is proposed by removing the time-consuming pressure Poisson equation from this fixed point interaction. The proposed method has shown a promising performance in modeling complex FSI system.

Contents

List of figures

List of tables

Nomenclature

List of abbreviations

AB2	Second order Adams-Bashforth
ALE	Arbitrary Lagrangian-Eulerian
AM	Adams-Moulton
BDF	Backward differentiation formula
CCS	Compressed Column Storage
CFD	Computational fluid dynamics
CFL	Courant–Friedrichs–Lewy
CG	Conjugate gradient
CN	Crank-Nicolson
CRS	Compressed Row Storage
DNS	Direct numerical simulation
FD/DLM	Fictitious domain/distributed Lagrange multiplier method
FSI	Fluid-structure interaction
GAMG	Geometric algebraic multi-grid
GPU	Graphics processing unit
IBM	Immersed boundary method
IBPM	Immersed boundary projection method
IC	Incomplete Cholesky
IFEM	Immersed finite element method
IIM	Immersed interface method
ILU	Incomplete LU
MG	Multi-grid
MIBM	Moving immersed boundary method
MPI	Message passing interface
NSE	Navier-Stokes equations
PPE	Pressure Poisson equation
RK	Runge–Kutta

List of symbols

Variables

t	Time
$\boldsymbol{\sigma}$	Fluid stress tensor
$\boldsymbol{\varepsilon}$	Fluid strain tensor
$\mathbf{u}$	Fluid velocity
ρ	Fluid density
p	Fluid pressure
ϕ	Fluid pseudo pressure
μ	Fluid dynamic viscosity
ν	Fluid kinematic viscosity
ω	Vorticity
ψ	Streamfunction
$\mathbf{U}_b$	Solid boundary velocity
$\mathbf{F}$	Boundary force in Lagrangian frame
$\mathbf{f}$	Boundary force in Eulerian frame
$\mathbf{g}$	Gravity vector
Re	Reynolds number
St	Strouhal number
KC	Keulegan-Carpenter number
$\mathbf{u}^*$	Fluid velocity from explicit prediction
$\hat{\mathbf{u}}$	Fluid velocity from implicit viscous prediction
$\tilde{\mathbf{u}}$	Fluid velocity from immersed boundary forcing
$\mathbf{u}^{n+1}$	Fluid velocity from projection
$\mathbf{v}_s$	Solid translational velocity
m_s	Solid mass
ρ_s	Solid density
I_s	Moment of inertial
$\boldsymbol{\omega}_s$	Solid angular velocity
$\mathbf{x}_s$	Solid position vector at the interface
$\mathbf{x}_c$	Solid gravity center position vector
$\mathbf{r}$	Solid position vector at the interface from the mass center
$\boldsymbol{\theta}_c$	Solid rotational angle
$\mathbf{n}$	Unit normal vector
$\boldsymbol{\tau}$	Unit tangent vector
ε_P	Stiffness constant for particle-particle collision
ε_W	Stiffness constant for particle-wall collision

Operators

$\mathcal{L}$	Laplacian
$\mathcal{N}$	Convective
$\mathcal{D}$	Divergence
$\mathcal{G}$	Gradient
$\mathcal{C}$	Curl
$\mathcal{R}$	Rotation
$\mathcal{M}$	Moving force
$\mathcal{T}$	Interpolation
$\mathcal{S}$	Spreading

Domains

Ω_f	Fluid domain
Ω_s	Solid domain
$\overline{\Omega}$	Combined fluid and solid domain
Γ_N	Neumann boundary
Γ_D	Dirichlet boundary
$\partial\Omega_i$	Fluid-structure interaction interface

Introduction

1.1 Motivation and background

The fluid-structure interaction (FSI) problem is of considerable scientific and technological interest in a broad spectrum of disciplines, such as aerodynamics, biology, hydrodynamics, civil engineering, etc. It occurs in a large scope of scales, ranging from the airplanes to the microcapsules, from the ships to the blood flow in arteries. The fundamental of FSI is the interaction between the solid structure and its internal or surrounding fluid flow. Due to the strong non-linearity and the multidisciplinary nature, the study of FSI problem is quite challenging.

For most FSI applications the analytical solutions are impossible to obtain, and the experiments are generally limited in scope. Owing to the increasingly growing powers of modern computers, numerical simulation becomes an important tool to investigate the fundamental physics of FSI. It reduces or avoids the expensive experiments and provides new insights into this complicated problem.

Due to such potential benefits, the development and application of numerical techniques for FSI simulations have gained a lot of popularities over the past decades. However numerical modeling of FSI problems is not an easy task. First, the motion and the deformation of solid structures are the results of the surrounding fluid stress, while the fluid flow in return is influenced by the solid movement. Therefore, a fully coupled problem is formulated, which is difficult to solve. Secondly, the interface of FSI is often complex and time-dependent. Representing such a complex interface accurately and efficiently is also difficult.

Accuracy, robustness and efficiency are the most important characteristics to examine the numerical methods. The present thesis concentrates on a new formulation for general FSI problems. Particular interest is devoted to the interaction between the incompressible fluid flow with rigid solids undergoing arbitrarily large displacements.

1.2 Objective of this thesis

The objective of this thesis is to develop a versatile numerical method for the FSI simulations, which mainly includes

- A high order accurate fluid Navier-Stokes solver;
- An efficient method for tackling the complex moving interface;
- A strongly coupled scheme for the fluid-structure interaction;
- A parallel algorithm to enable high performance computing.

1.3 Outline of this thesis

This thesis is organized as follows:

– **Chapter 2** gives a literature review of the incompressible fluid solvers and the numerical methods for the fluid-structure interaction.

– **Chapter 3** presents the projection method for the numerical solution of the incompressible fluid Navier-Stokes equations. High order time accurate scheme is employed to discretize the governing equations. How to preserve this temporal accuracy after the time splitting is discussed in detail. The staggered mesh is presented to decouple the pressure and velocity in space. Efficient linear system solvers and preconditioners are introduced for solving the discretized equations. Two parallel computing methods (CPU and GPU) are considered to accelerate the calculations. Numerical examples are performed in the end of this chapter to validate the current method.

– **Chapter 4** is devoted to a novel implicit immersed boundary method, called the moving immersed boundary method (MIBM), for the flows with complex moving boundaries. The novel method is an extension of previous explicit immersed boundary methods, to achieve better accuracy and higher efficiency. To impose the desired boundary condition at the interface, an additional moving force equation is derived and solved with the conjugate gradient method in this chapter. Comparison with previous models are made and numerical simulations are performed to demonstrate the accuracy and the efficiency of the proposed method.

– **Chapter 5** presents a partitioned strongly coupled scheme for the fluid-structure interaction. The fluid Navier-Stokes equations are linked to the rigid body motion equations via the moving immersed boundary method. This coupling is done implicitly through a fixed-point iteration. To save computational time, a new coupling scheme is proposed to prevent from solving the pressure Poisson equation repeatedly at each time step. The advantage of current partitioned strongly coupled method is highlighted. Numerous simulations with simple and complex solid geometries are performed to validate the proposed method.

– **Chapter 6** gives some conclusions and suggestions for the future work.

State of the art review

2.1 Introduction

This chapter is dedicated to a literature overview of the numerical methods related to this thesis, which consists of two main topics:

- **Numerical simulation of incompressible fluid flows**. The fluid solvers reviewed here are classified into two groups: the primitive variable formulation and the non-primitive variable formulation. The first group includes the penalty method, the artificial compressibility method and the SIMPLE-type methods. The second group contains the streamfunction-vorticity formulation, the velocity-vorticity formulation and the lattice Boltzmann method (LBM).
- **Numerical solution of fluid-structure interactions**. Five different methods are discussed, namely the boundary element method (BEM), the arbitrary Lagrangian-Eulerian method (ALE), the meshfree methods, the extended finite element method (XFEM) and the overset grid/Chimera method.

2.2 Fluid governing equations

To begin with, we will derive the governing equations for the fluid flow. The fluid motion respects the Newton's law. Under the continuum assumption the fluid flow is governed by the Navier-Stokes equations (NSE), which read

$$\frac{\partial}{\partial t}(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = \nabla \cdot \boldsymbol{\sigma} \tag{2.1a}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.1b}$$

where $\mathbf{u}(\mathbf{x}, t)$ is the fluid velocity vector, ρ the fluid density, $\boldsymbol{\sigma}$ the fluid stress tensor. Here $\otimes$ designates the tensor product. (2.1a) and (2.1b) represent the conservation

of momentum and mass respectively. Under the Stokes assumption for a Newtonian fluid, the fluid stress tensor is given by

$$\boldsymbol{\sigma} = (-p + \lambda \nabla \cdot \mathbf{u})\mathbf{I} + 2\mu\boldsymbol{\varepsilon} \tag{2.2}$$

where p designates the pressure, μ the dynamic viscosity, λ the bulk viscosity and $\mathbf{I}$ the identity tensor. The strain tensor $\boldsymbol{\varepsilon}$ is written as

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^{\mathrm{T}}) \tag{2.3}$$

For the incompressible fluid (ρ = const), the governing equations become

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} \tag{2.4a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2.4b}$$

where $\nu = \mu/\rho$ is the kinematic viscosity. This form of NSE is usually termed as the primitive variable formulation, namely with the velocity $\mathbf{u}$ and the pressure p.

For studying physical problems in different scales, the aforementioned equations are normalized to a dimensionless form. Considering L, U the reference length and velocity, we have the following dimensionless quantities

$$\mathbf{x}^* = \frac{\mathbf{x}}{L}, \quad \mathbf{u}^* = \frac{\mathbf{u}}{U}, \quad t^* = \frac{t}{L/U}, \quad p^* = \frac{p}{\rho U^2} \tag{2.5}$$

Substituting these dimensionless variables into (2.4), we obtain

$$\left(\frac{U}{L/U}\right)\frac{\partial \mathbf{u}^*}{\partial t^*} + \left(\frac{U^2}{L}\right)\nabla \cdot (\mathbf{u}^* \otimes \mathbf{u}^*) = -\left(\frac{\rho U^2}{\rho L}\right)\nabla p^* + \left(\frac{\nu U}{L^2}\right)\nabla^2 \mathbf{u}^* \tag{2.6a}$$

$$\left(\frac{U}{L}\right)\nabla \cdot \mathbf{u}^* = 0 \tag{2.6b}$$

which can be rearranged to

$$\frac{\partial \mathbf{u}^*}{\partial t^*} + \nabla \cdot (\mathbf{u}^* \otimes \mathbf{u}^*) = -\nabla p^* + \frac{1}{Re}\nabla^2 \mathbf{u}^* \tag{2.7a}$$

$$\nabla \cdot \mathbf{u}^* = 0 \tag{2.7b}$$

where $Re = UL/\nu$ designates the Reynolds number. By assuming that the problem is appropriately scaled, the subscript is dropped for convenience. Therefore, the

non-dimensional incompressible viscous fluid NSE is written as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = -\nabla p + \frac{1}{Re}\nabla^2 \mathbf{u} \tag{2.8a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2.8b}$$

or in a non-conservative form as

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2 \mathbf{u} \tag{2.9a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2.9b}$$

Directly solving the above NSE is very difficult for the following reasons:

- The equations are no-linear;
- There is no explicit equation for the pressure;
- The pressure and the velocity are coupled through the continuity (incompressibility or divergence-free) condition (2.9b);
- The solution of the pressure is not unique and is determined up to an additive constant, since the pressure presents itself in the momentum equation (2.9a) in a gradient form (Gresho & Sani, 1987; Gresho, 1991).

2.3 Incompressible fluid solvers

2.3.1 Non-primitive variable formulations

Considering the difficulties in solving the NSE (2.9) directly, formulations that automatically satisfy the divergence free condition (2.9b) are favored in the literature.

2.3.1.1 Streamfunction-vorticity formulation

In the streamfunction-vorticity method, we apply the curl operator to the NSE (2.9a)

$$\nabla \times (\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}) = \nabla \times (-\nabla p + \nu \nabla^2 \mathbf{u}) \tag{2.10}$$

then the vorticity transport equation (VTE) is obtained

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \nabla^2 \omega \tag{2.11}$$

In two dimensions, the vorticity ω is a scalar field, defined by

$$\omega = \nabla \times \mathbf{u} = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \tag{2.12}$$

For satisfying the divergence-free condition automatically, the streamfunction ψ is introduced

$$\frac{\partial \psi}{\partial y} = u, \qquad \frac{\partial \psi}{\partial x} = -v \tag{2.13}$$

Substitution of (2.13) into (2.12) yields an elliptic equation for the streamfunction

$$\nabla^2 \psi = -\omega \tag{2.14}$$

As a consequence, the pressure has been eliminated as a dependent variable completely. The resulting two equations (2.14) and (2.11) are coupled through the velocity, which is actually the derivative of the streamfunction ψ.

The solution is then as follows. At each time step, given the initial velocity field, we can obtain the initial vorticity using (2.12). With appropriate boundary conditions, we advance the vorticity transport equation (2.11). Once the vorticity is calculated, we update the streamfunction by solving the Poisson equation (2.14). The velocity field can be obtained by the definition of streamfunction (2.13). If necessary, the pressure can still be recovered.

The streamfunction-vorticity method is very attractive for two-dimensional simulations, since it reduces the number of variables from three (u, v, p) to two (ψ, ω). But its extension to three-dimension can lose this advantage. In three dimensions, the vorticity and streamfunction become three-component vectors. This requires six partial differential equations instead of four in the primitive variable formulation.

Moreover, the boundary conditions in the streamfunction-vorticity formulation are not straightforward compared to the primitive variable formulation. For complicated geometries, the vorticity at sharp corners is singular, which makes its evaluation very difficult. Another problem with this formulation lies in the Poisson equation of the streamfunction. The value of the streamfunction at solid or symmetric boundaries can be calculated only if the velocity is known, which however in return depends on the streamfunction (Gresho, 1991; Ferziger & Peric, 2002; McDonough, 2007).

2.3.1.2 Velocity-vorticity formulation

Another method using the vorticity transport equation to solve NSE (2.9) is the velocity-vorticity formulation, which can be summarized as

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \nabla^2 \omega \tag{2.15a}$$

$$\omega = \nabla \times \mathbf{u} \tag{2.15b}$$

$$\nabla^2 \mathbf{u} = -\nabla \times \omega \tag{2.15c}$$

where the last equation is a Poisson equation for the velocity, which is obtained by taking the curl to the second equation and using the identity $\nabla^2 \mathbf{u} = \nabla(\nabla \cdot \mathbf{u}) - \nabla \times \nabla \times \mathbf{u}$ along with incompressibility condition (2.9b). This formulation is usually discretized in a Lagrangian frame in the particle (vortex) method (Chorin, 1973; Koumoutsakos & Leonard, 1995; Cottet & Koumoutsakos, 2000). Except for the streamfunction, this formulation inherits other drawbacks of the streamfunction-vorticity formulation.

2.3.1.3 Lattice Boltzmann method (LBM)

Instead of dealing with macroscopic Navier-Stokes equations, the lattice Boltzmann method (LBM) is based on mesoscopic kinetic equations. LBM was originated from the Ludwig Boltzmann's kinetic theory. The essential idea is that the fluids are considered as a collection of small particles moving with random motions. The momentum and energy exchange are accomplished through particle streaming and billiard-like collision. This process is governed by the Boltzmann transport equation

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla f = \Omega(f) \tag{2.16}$$

where $f(\mathbf{x}, t)$ represents the particle distribution function, $\boldsymbol{\xi}$ the particle velocity and $\Omega(f)$ the collision operator which is responsible for the rate of change of the particle distribution. LBM simplifies this idea by reducing the number of particles and confining them to the lattice nodes. The lattice Boltzmann equation (LBE) is written as

$$f_i(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega_i \tag{2.17}$$

where f_i represents the particle density distribution function along ith direction and $\mathbf{e}_i$ is the discrete velocity. The lattice speed is defined as $c = \Delta x / \Delta t$, where Δx and Δt are the space and time increments respectively.
The macroscopic fluid variables, such as the density ρ and momentum $\rho \mathbf{u}$, can be

calculated at each node by

$$\rho = \sum_i f_i, \quad \rho \mathbf{u} = \sum_i f_i \mathbf{e}_i \tag{2.18}$$

The pressure is computed directly from the density by the isothermal equation of state

$$p = c_s^2 \rho \tag{2.19}$$

The fluid kinematic viscosity ν is recovered by

$$\nu = \frac{2\tau - 1}{6} \frac{\Delta x^2}{\Delta t} \tag{2.20}$$

LBM is found very efficient in fluid flow simulations. This is mainly due to the fact that it does not involve the pressure Poisson equation and the pressure is recovered by the equation of state. The flow is simply simulated by the streaming and collision processes. This explicit time scheme makes it ideal for parallel computing. The LBM has been successfully implemented in the commercial software (e.g. PowerFlow, XFlow) and open source libraries (e.g. Palabos, OpenLB) for real applications.

However, owing to the fact that LBM is a physically based method, it contains some compressible effects in incompressible fluid flow simulations. It should also be pointed out that the lattice Boltzmann equation can recover the incompressible Navier-Stokes equations using the Chapman–Enskog expansion to the second order in space and time (Chen & Doolen, 1998) with the following condition

$$Ma = \frac{u_{\max}}{c} \ll 1 \tag{2.21}$$

where Ma is the Mach number and $u_{\max}$ is the maximum fluid velocity.

Moreover, the boundary and initial conditions in LBM are treated differently from the Navier-Stokes based methods, since the primitive variable in LBE is the particle distribution function. The fluid density and velocity can be determined uniquely from this distribution function but not vice versa. How to transform the initial and boundary conditions from macroscopic variables to particle distribution functions is a central issue in LBM developments. Accurate initial and boundary conditions are crucial in LBM for simulating fluid flows (Guo & Shu, 2013).

2.3.2 Primitive variable formulations

The fluid solvers using the $\mathbf{u} - p$ formulation do not have the shortcomings of the non-primitive variable formulations and thus are summarized here.

2.3.2.1 Penalty method

Considering that the pressure is not included in the continuity equation and only appears in the momentum equation in a gradient form. There is no unique solution for the pressure field and it is solved up to an additive constant. To overcome these difficulties, the penalty method introduces a small term to contain the pressure in the continuity equation (Témam, 1968), namely

$$\varepsilon p + \nabla \cdot \mathbf{u} = 0 \tag{2.22}$$

where ε is an artificial small constant. Hence the pressure is computed by

$$p = -\frac{1}{\varepsilon} \nabla \cdot \mathbf{u} \tag{2.23}$$

Substituting (2.23) into (2.9a) yields an equation for the velocity

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \frac{1}{\varepsilon} \nabla (\nabla \cdot \mathbf{u}) + \frac{1}{Re} \nabla^2 \mathbf{u} \tag{2.24}$$

So one can solve (2.24) for the velocity and compute the pressure afterwards from (2.23).

The penalty formulation can only recover to the original NSE when ε approaches zero, otherwise the continuity equation is violated. Moreover, the appropriate value of ε is not obvious. In practice it is chosen as small as numerically possible, but too small value can make the system ill-conditioned.

2.3.2.2 Artificial compressibility method

The artificial compressibility method, first proposed by Chorin (1967), is very similar to the penalty method, which adds a time derivative of pressure to the continuity equation

$$\frac{1}{\beta} \frac{\partial p}{\partial t} + \nabla \cdot \mathbf{u} = 0 \tag{2.25}$$

where $\beta = \rho c^2$. Here ρ, c designate the fluid density and the speed of sound respectively. Larger value of β means that the fluid is more incompressible, but too large β can make the equations very stiff numerically.

The two equations (2.9a) and (2.25) are advanced explicitly in time in Chorin (1967). Explicit scheme is easy to be implemented but places a severe restriction on the size of the time step. Since the time step in this case is limited by the inverse of c^2, for nearly incompressible flows ($c \to \infty$) only an extremely small time step is allowed. Since the artificial compressibility method is intended for steady flows, implicit method should be employed and the resulting linear system can be solved by iterative methods.

2.3.2.3 SIMPLE-type methods

The SIMPLE-type methods, such as SIMPLE, SIMPLER, SIMPLEC, PISO and others, are frequently used in the commercial and open source software (e.g. FLUENT, CFX, OpenFOAM, etc.). The fundamental idea of the SIMPLE-type methods is using a predictor-corrector procedure. Although SIMPLE was initially aimed for steady flow and PISO was originally used for unsteady compressible flow, they can be extended to both steady and unsteady incompressible flow simulations. Following Ni & Abdou (2004), the NSE is discretized in one cell as follows

$$\left(\frac{1}{\Delta t}\mathbf{I} + \mathbf{A}_P^n\right)\mathbf{u}_P^{n+1} = \sum_M \mathbf{A}_M^n \mathbf{u}_M^{n+1} - \mathbf{G}_P(p^{n+1}) + \mathbf{S}^n \tag{2.26a}$$

$$\mathbf{D}_P \mathbf{u}_P^{n+1} = 0 \tag{2.26b}$$

where the superscript $n+1$ and n represent the current time level and the past time level respectively. The subscript M denotes the neighbour nodes surrounding the pole node P. $\mathbf{G}_P$ and $\mathbf{D}_P$ represent the gradient and divergence operators associated with the node P respectively. The explicit terms in the discretized momentum equation, including the explicit temporal term $\mathbf{u}_P^n/\Delta t$, are incorporated into $\mathbf{S}^n$. The coefficient $\mathbf{A}^n$ contains the contribution of the implicit terms (The superscript n indicates the contribution from the linearly implicit convection $\mathbf{u}^n \cdot \nabla \mathbf{u}^{n+1}$). For steady flow simulations, the temporal term $\mathbf{u}_P^{n+1}/\Delta t$ is dropped in (2.26a).

(*i*) SIMPLE algorithm

The algorithm of SIMPLE (Semi-Implicit Method for Pressure Linked Equations) was originally proposed by Patankar & Spalding (1972). Given a guessed pressure field p^*, it predicts a velocity field by

$$\left(\frac{1}{\Delta t}\mathbf{I} + \mathbf{A}_P^n\right)\mathbf{u}_P^* = \sum_M \mathbf{A}_M^n \mathbf{u}_M^* - \mathbf{G}_P(p^*) + \mathbf{S}^n \tag{2.27}$$

By subtracting (2.27) from (2.26a), a velocity difference equation is then acquired

$$\left(\frac{1}{\Delta t}\mathbf{I}+\mathbf{A}_P^n\right)\left(\mathbf{u}_P^{n+1}-\mathbf{u}_P^*\right)=\sum_M \mathbf{A}_M^n\left(\mathbf{u}_M^{n+1}-\mathbf{u}_M^*\right)-\mathbf{G}_P\left(p'\right) \tag{2.28}$$

Donate $\mathbf{u}'_P = \mathbf{u}_P^{n+1} - \mathbf{u}_P^*$, $p' = p^{n+1} - p^*$ the velocity and pressure corrections, therefore

$$\left(\frac{1}{\Delta t}\mathbf{I}+\mathbf{A}_P^n\right)\mathbf{u}'_P=\sum_M \mathbf{A}_M^n\mathbf{u}'_M-\mathbf{G}_P\left(p'\right) \tag{2.29}$$

The main approximation of the SIMPLE algorithm is neglecting the term $\sum_M \mathbf{A}_M^n\mathbf{u}'_M$ in (2.29). By applying the continuity condition (2.26b) to (2.29), a pressure correction equation is then obtained

$$\mathbf{D}_P\left[(\mathbf{I}+\mathbf{A}_P^n\Delta t)^{-1}\mathbf{G}_P(p')\right]=\left(\frac{1}{\Delta t}+\mathbf{A}_P^n\right)\mathbf{D}_P\mathbf{u}_P^* \tag{2.30}$$

Once the pressure correction equation is solved, the velocity and pressure are corrected by

$$\mathbf{u}'_P=-\Delta t(\mathbf{I}+\mathbf{A}_P^n\Delta t)^{-1}\mathbf{G}_P(p') \tag{2.31}$$

$$p^{n+1}=p^*+p' \tag{2.32}$$

$$\mathbf{u}_P^{n+1}=\mathbf{u}_P^*+\mathbf{u}'_P \tag{2.33}$$

Owing to the omission of neighbouring term, $\mathbf{u}^{n+1}$ is not divergence-free with just one single correction. Thus this predictor-corrector procedure is repeated at each time step with (2.27), (2.30), (2.31), (2.32), (2.33) by setting $p^* = p^{n+1}$ at the end of each sub-iteration until the convergence is achieved.

It should be noted that the pressure correction equation (2.30) in SIMPLE algorithm is susceptible to divergence (Versteeg & Malalasekera, 2007). A remedy is to use under-relaxation for this iterative process. However the optimum choice of the under-relaxation factor is *ad hoc* and depends on the flow. Too large value may lead to oscillation or even divergence. If the value is overly small, the convergence becomes extremely slow.

(*ii*) SIMPLER algorithm

Considering that the omission of the term $\sum_M \mathbf{A}_M^n\mathbf{u}'_M$ in the SIMPLE algorithm requires under-relaxation to ensure the convergence, Patankar (1980) later proposed the SIMPLER (SIMPLE-Revised) algorithm. In this improved version, a pressure equation is used instead of the pressure correction equation. But the velocity correction is retained for correcting the velocity.

To illustrate the SIMPLER algorithm, the momentum equation (2.26a) is rearranged as

$$\mathbf{u}_P^{n+1} = \Delta t \left(\mathbf{I} + \mathbf{A}_P^n \Delta t\right)^{-1} \left(\sum_M \mathbf{A}_M^n \mathbf{u}_M^{n+1} + \mathbf{S}^n\right) - \Delta t \left(\mathbf{I} + \mathbf{A}_P^n \Delta t\right)^{-1} \mathbf{G}_P(p^{n+1}) \tag{2.34}$$

Given a guessed field $\mathbf{u}^*$, a velocity is predicted to

$$\hat{\mathbf{u}}_P = \Delta t \left(\mathbf{I} + \mathbf{A}_P^n \Delta t\right)^{-1} \left(\sum_M \mathbf{A}_M^n \mathbf{u}_M^* + \mathbf{S}^n\right) \tag{2.35}$$

Applying the incompressibility condition (2.26b) to (2.34) yields an equation of the pressure

$$\mathbf{D}_P \left[(\mathbf{I} + \mathbf{A}_P^n \Delta t)^{-1} \mathbf{G}_P(p^{n+1})\right] = \frac{1}{\Delta t} \mathbf{D}_P \hat{\mathbf{u}}_P \tag{2.36}$$

Therefore the velocity is corrected by

$$\mathbf{u}'_P = -\Delta t (\mathbf{I} + \mathbf{A}_P^n \Delta t)^{-1} \mathbf{G}_P(p^{n+1}) \tag{2.37}$$

$$\mathbf{u}_P^{n+1} = \hat{\mathbf{u}}_P + \mathbf{u}'_P \tag{2.38}$$

Since $\hat{\mathbf{u}}_P$ is estimated by $\mathbf{u}_M^*$ not $\mathbf{u}_M^{n+1}$ in (2.36), this predictor-corrector procedure needs to be repeated at every time step until convergence with (2.35), (2.36), (2.37), (2.38) by setting $\mathbf{u}^* = \mathbf{u}^{n+1}$ at the end of each sub-iteration.

(***iii***) **SIMPLEC algorithm**

The SIMPLEC (SIMPLE-Consistent) algorithm introduced by Van Doormaal & Raithby *et al.* (1984) follows the same steps as the SIMPLE algorithm. But instead of omitting the terms $\sum_M \mathbf{A}_M^n \mathbf{u}'_M$ in the derivation of the pressure correction equation, SIMPLEC approximates them with $\sum_M \mathbf{A}_M^n \mathbf{u}'_P$, namely

$$\left(\frac{1}{\Delta t}\mathbf{I} + \mathbf{A}_P^n - \sum_M \mathbf{A}_M^n\right) \mathbf{u}'_P = -\mathbf{G}_P\left(p'\right) \tag{2.39}$$

Applying the incompressibility condition (2.26b) gives

$$\mathbf{D}_P \left[(\mathbf{I} + \mathbf{A}_P^n \Delta t - \sum_M \mathbf{A}_M^n \Delta t)^{-1} \mathbf{G}_P(p')\right] = \frac{1}{\Delta t} \mathbf{D}_P \mathbf{u}_P^* \tag{2.40}$$

Once the pressure correction equation is solved, the velocity correction is obtained

by using

$$\mathbf{u}'_P = -\Delta t \left(\mathbf{I} + \mathbf{A}_P^n \Delta t - \sum_M \mathbf{A}_M^n \Delta t \right)^{-1} \mathbf{G}_P \left(p' \right) \tag{2.41}$$

The velocity and pressure are updated by

$$p^{n+1} = p^* + p' \tag{2.42}$$

$$\mathbf{u}_P^{n+1} = \mathbf{u}_P^* + \mathbf{u}'_P \tag{2.43}$$

Again this procedure is iterated for convergence at each time step, namely repeating (2.27), (2.40), (2.41), (2.42), (2.43) with $p^* = p^{n+1}$ at the end of every sub-iteration.

(***iv***) **PISO algorithm**

Contrary to previous SIMPLE, SIMPLER, SIMPLEC methods, the PISO algorithm (Pressure Implicit with Split Operator) originally developed by Issa (1985) is non-iterative and aimed for transient flows. At every time step, the PISO algorithm only involves one predictor step and two corrector steps.
In PISO method, the same predictor is used as in the SIMPLE method, for predicting the velocity by a guessed pressure field. The first corrector follows

$$\frac{1}{\Delta t} \mathbf{u}_P^{**} = \sum_M \mathbf{A}_M^n \mathbf{u}_M^* - \mathbf{G}_P(p^*) + \mathbf{S}^n \tag{2.44}$$

A new pressure is determined such that the intermediate velocity is divergence-free $\mathbf{D}_P \mathbf{u}_P^{**} = 0$. The corresponding pressure p^* then is

$$\mathbf{D}_P \mathbf{G}_P(p^*) = \mathbf{D}_P \left[\sum_M \mathbf{A}_M^n \mathbf{u}_M^* + \mathbf{S}^n \right] \tag{2.45}$$

The PISO method enhances the SIMPLE method with a further corrector step

$$\frac{1}{\Delta t} \mathbf{u}_P^{***} = \sum_M \mathbf{A}_M^n \mathbf{u}_M^{**} - \mathbf{G}_P(p^{**}) + \mathbf{S}^n \tag{2.46}$$

By enforcing $\mathbf{D}_P \mathbf{u}_P^{***} = 0$, the pressure p^{**} is calculated by

$$\mathbf{D}_P \mathbf{G}_P(p^{**}) = \mathbf{D}_P \left[\sum_M \mathbf{A}_M^n \mathbf{u}_M^{**} + \mathbf{S}^n \right] \tag{2.47}$$

The actual realization of the PISO algorithm follows (2.27), (2.45), (2.44), (2.47) and (2.46) at each time step without repetition. Issa (1985) indicated that two cor-

rector steps are sufficient for approximating the final fields with $p^{n+1} = p^{**}$ and $\mathbf{u}^{n+1} = \mathbf{u}^{***}$, since the splitting errors for the velocity and pressure are of $O(\Delta t^4)$ and $O(\Delta t^3)$ respectively.
Due to its non-iterative feature, the PISO algorithm is more efficient than the SIMPLE, SIMPLER and SIMPLEC methods for transient flows. But the time step size should be kept small to maintain stability, since it is not a fully implicit method. A combined PIMPLE (merged PISO-SIMPLE) algorithm, is promoted in the openFOAM library in order to enlarge the time step.

2.4 Fluid-structure interaction methods

2.4.1 Boundary element method (BEM)

The boundary integral method or boundary element method (BEM) provides an efficient numerical tool of high precision for solving linear homogeneous partial differential equations using Greens' functions (Pozrikidis, 1992, 2002). Instead of discretizating the governing equations on the whole physical domain, the BEM formulates the equations into a boundary integral form and only solves for the boundary distribution of the unknown function or one of its derivative. As a result, the BEM decreases the geometric dimension of the physical problem by one and consequently reduces the number of unknowns. The BEM is widely used in the modeling of capsule motions in flow with very small Reynolds number. However, the method is only valid for infinite and semi-infinite domains and for flows where the inertial effects are negligible, i.e. Stokes flow. Therefore, it is extremely difficult to be applied to complicated geometries at moderate and high Reynolds number flow regime.

2.4.2 Arbitrary Lagrangian–Eulerian method (ALE)

For complicated flows, one can discretize the Navier-Stokes equations directly in the full physical space and solve them with sophisticated techniques, such as the finite element method, the finite volume method, etc.
The coordinate system is commonly described with two formulations, i.e. the Lagrangian description and the Eulerian description. In the Lagrangian formulation, the coordinate system moves with the fluid. Thus it is very easy to track the material interface. However it is essentially used for closed domains. Otherwise, any fluid particle quitting the domain should be replaced by a new one (Lefrançois & Boufflet, 2003). Another drawback of the purely Lagrangian method is that it suffers a

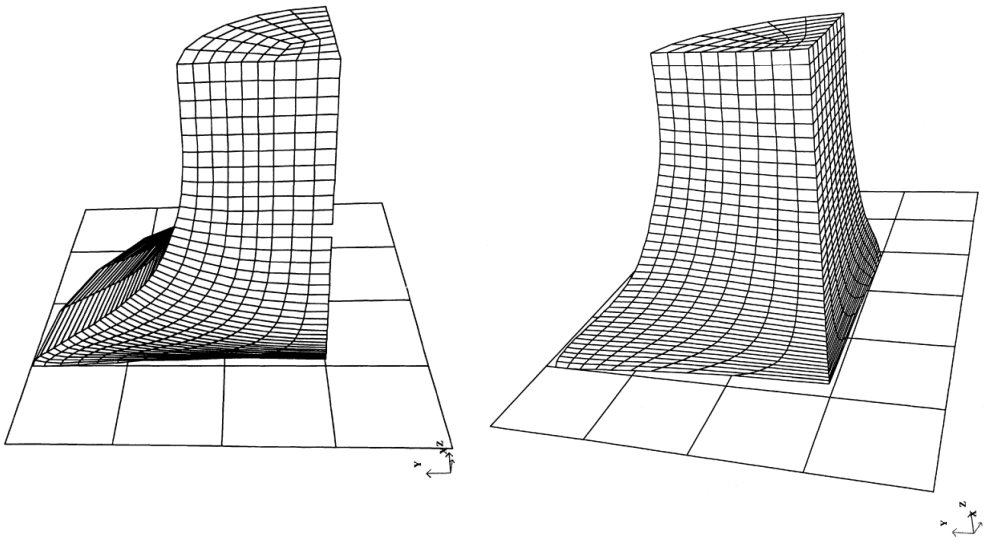
severe mesh distortion in the computational domain. The excessive mesh distortion problem can be easily handled in the Eulerian formulation, where the coordinate system is stationary and the fluid particles pass through a fixed region of space. The major disadvantage of the purely Eulerian method is that it is unable to define sharp interfaces.

The arbitrary Lagrangian–Eulerian (ALE) method provides a hybrid description of the purely Lagrangian method and the purely Eulerian method (Hu *et al.*, 1992; Hu, 1996; Hu *et al.*, 2001). The coordinate system in ALE method is associated with a moving imaginary mesh, allowing a smooth transition between the Lagrangian method and the Eulerian method. ALE method enables us to use the Lagrangian method in the zones where the mesh motion is small and to apply the Eulerian method in the zones where it is not possible for the mesh to follow the fluid motion (see Figure 2.1 for example, where the ALE mesh undergoes less distortions than the Lagrangian mesh). The unsteady incompressible NSE in an ALE frame of reference is written as

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} - \mathbf{u}_m) \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u} \tag{2.48a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2.48b}$$

where $\mathbf{u}_m$ represents the imaginary mesh velocity.



(a) Lagrangian formulation (b) ALE formulation

Figure 2.1: Mesh in Lagrangian and ALE formulations (Souli *et al.*, 2000).

Many successful FSI applications with the ALE method can be found in Souli *et al.* (2000), Duarte *et al.* (2004) and Kassiotis *et al.* (2011). However establishing a body-fitted grid of high quality is not an easy task, especially for the structured mesh. The unstructured mesh is much easier to fit the complex geometries compared to the structured one. This process requires the user's experience and usually takes most of the Computational Fluid Dynamic (CFD) time.

This body-conforming mesh method hits a bottleneck when applied to FSI problems, as the solid boundary changes over time. User intervention must be eliminated and the automatic meshing is sought. For small geometrical changes, the deforming mesh method is often adopted by moving mesh points to accommodate the boundary motion. However for significant shape changes, simple mesh motion will lead to extremely twisted meshes and finally breaks down the calculation. In this case the mesh topology, resolution and connectivity should be adapted (Jasak, 2009). This can be realised by re-meshing along with data mapping. The data mapping is used for transferring the flow fields from the old mesh to the new one, which however introduces numerical errors. Frequent re-meshing is certainly undesirable since it slows down the computation. Even though the ALE approach reduces the mesh distortion compared to the purely Lagrangian method, it is still difficult to preserve the mesh quality when the solid boundary undergoes large displacements or deformations.

2.4.3 Meshfree methods

The meshfree (or meshless) methods circumvent the mesh distortion problems by discretizating the computational domain with only a set of nodal points without any connections between nodes. The first meshfree method is introduced by Gingold & Monaghan (1977) and Lucy (1977), called the smoothed-particle hydrodynamics (SPH) method. Later other variants emerged such as the diffuse element method (DEM) (Nayroles *et al.*, 1992), the reproducing kernel particle method (RKPM) (Liu *et al.*, 1995a,b), the partition of unity method (PUM) (Babuska & Melenk, 1997), the element free Galerkin method (EFGM) (Belytschko *et al.*, 1994), etc.

The SPH method was initially developed for astrophysical problems and later extended to FSI problems. Since the SPH method describes the fluid motion in a Lagrangian framework, it is therefore very easy to simulate the FSI problems with the free surface (Antoci *et al.*, 2007; Canelas *et al.*, 2015). The SPH method has been implemented in the open source software SPHysics and the GPU-accelerated library DualSPHysics (Crespo *et al.*, 2015). Successful applications can be found

towards the simulation of wave impact on the coastal structures (Altomare *et al.*, 2015).
However the meshfree methods are not always superior to the mesh-based methods. First, the imposition of essential (Dirichlet) boundary conditions is quite difficult for meshfree methods, since they lack Kronecker delta property. Incorrect treatment may degrade the convergence. Improvements can be made by using the penalty method, Lagrange multiplier method, the augmented Lagrange multiplier method and the Nitsche's method. Moreover the calculation in meshfree methods is generally much slower than that in the mesh-based methods (Li & Liu, 2002; Fries & Matthies, 2004; Nguyen *et al.*, 2008; Monaghan, 2012).

2.4.4 Extended finite element method (XFEM)

The extended finite element method (XFEM) was initially developed by Belytschko and co-workers (Belytschko & Black, 1999; Moës *et al.*, 1999) for modeling crack growth. The XFEM method enriches the shape functions to encompass the discontinuities, consequently alleviating the computational costs and projection errors associated with re-meshing. It has been successfully extended to simulate fluid-structure interaction problems (Wagner *et al.*, 2001; Gerstenberger & Wall, 2008). The XFEM relies on one specific space discretization, however the finite volume discretization is more frequently used in CFD than the finite element method for the reason of conservation properties and stabilities.

2.4.5 Overset grid/Chimera method

The overset grid method or Chimera method was pioneered by Benek *et al.* (1983) and has subsequently been extended to various applications (Deloze *et al.*, 2012; Li *et al.*, 2012; Shen *et al.*, 2015). The overset grid method decomposes the complex geometrical configuration into a set of simple, overlapping subdomains, such that a boundary-conforming mesh can be established very easily on each subdomain, as shown in Figure 2.2. Consequently, the governing equations are solved independently on each subdomain and the subdomains are connected through the specification of interfacial boundary conditions.
The overset grid method is powerful since it allows for efficient clustering grids near solid walls. This facilitates resolving the very thin boundary layers at high Reynolds number. However a major difficulty in the implementation of overset grid method is to specify boundary conditions for all flow variables at interfaces between adjacent subdomains. Interpolations are generally used but special attention should be paid

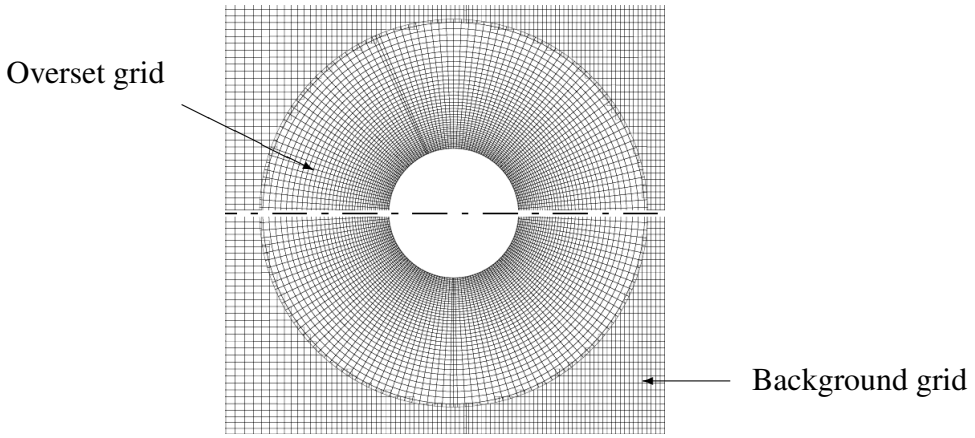


Figure 2.2: An illustrative example for the overset grid method (Deloze *et al.*, 2012).

to the global conservation of the flow quantities (Tang *et al.*, 2003). Moreover, body fitted grids are still used in the overset grid method. This is clearly inconvenient for the cases with deforming bodies, where deforming-mesh and re-meshing are required.

2.5 Concluding remarks

In this chapter, some principal methods have been reviewed towards the numerical simulation of incompressible fluid flows and the fluid-structure interactions.

In incompressible fluid flow simulations, the major difficulty is that the pressure and the velocity are coupled by the continuity condition. The primitive variable formulations decouple the pressure and the velocity, by either violating the continuity equation (the Penalty method and the artificial compressibility method) or iterating the two fields (the SIMPLE, SIMPLER, SIMPLC methods). The non-primitive variable formulations are very fast, since the pressure and the velocity have already been decoupled before they are solved. But the imposition of boundary conditions is not quite straightforward and sometimes difficult. In Chapter 3 the non-iterative projection method will be presented for the pressure-velocity decoupling, which is very efficient and does not contain any artificial constant.

For the fluid-structure interaction simulations, the BEM is highly accurate and efficient but only for Stokes flows and specific domains. The body-conforming mesh method, ALE, can be applied to Navier-Stokes equations and general domains. However it is limited to small displacements and deformations due to the mesh distortion. The meshfree methods circumvent the mesh distortion difficulty but they have problems of their own. The XFEM is confined to one space discretization,

thus it lacks generality. The overset grid/Chimera method is suitable for rigid body simulations at high Reynolds number, but it is difficult to ensure the conservation of flow quantities between adjacent meshes. In Chapter 4, a non-body conforming mesh method, the moving immersed boundary method (MIBM), will be proposed to simulate Navier-Stokes flows in general domains with arbitrary solid motions.

Projection method for simulating incompressible fluid flow

3.1 Introduction

In this chapter we will present the projection method for the pressure-velocity decoupling in the numerical simulation of incompressible fluid flows.
The projection method splits the Navier-Stokes equations into two independent equations for the pressure and the velocity and solves them separately. Whereas the overall time accuracy could be decreased due to this time splitting. Various methods will be presented in detail to the recovery of original time accuracy.
The time-discrete equations are then discretized on a staggered mesh in space. We will compare different solvers for the solution of the resulting systems. The Parallel computing is also considered in order to accelerate the calculation.

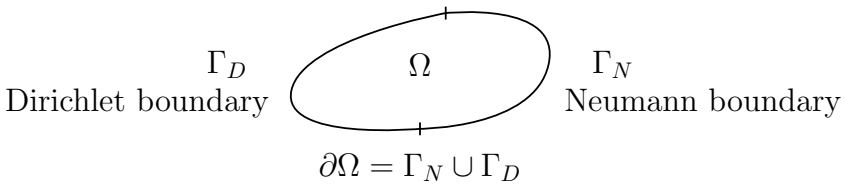


Figure 3.1: A schematic view of the fluid domain.

To begin with, we recall the incompressible Navier-Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u} \tag{3.1a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{3.1b}$$

$$\mathbf{u}|_\Gamma = 0 \tag{3.1c}$$

$$\mathbf{u}(t = 0) = \mathbf{u}_0 \tag{3.1d}$$

where initial and boundary conditions are assigned to the equations in order to close the mathematical problem. We donate $\Omega \subset \mathbb{R}^d$ ($d = 2$, or 3) the fluid domain and its

boundary $\Gamma = \Gamma_N \cup \Gamma_D = \partial\Omega$, as shown in Figure 3.1. For simple discussion, here a homogeneous Dirichlet boundary condition ($\Gamma_N = \emptyset$) is assumed for the velocity.

3.2 Derivation of the pressure Poisson equation

A mathematical observation towards the NSE (3.1) is that the pressure plays as a Lagrange multiplier to enforce the incompressibility (or continuity, divergence-free) constraint (3.1b) on the velocity field, since the continuity equation does not contain the pressure explicitly. To derive an equation for the pressure, we apply the divergence operator to (3.1a). Using the incompressibility condition (3.1b), the pressure Poisson equation (PPE) is obtained

$$\nabla^2 p = -\nabla \cdot (\mathbf{u} \cdot \nabla \mathbf{u}) \tag{3.2}$$

However, the boundary condition for the derived PPE is rather ambiguous. Taking the inner product of (3.1a) with the unit normal vector $\mathbf{n}$ and the tangent vector $\boldsymbol{\tau}$ at Γ, we obtain

$$\frac{\partial p}{\partial \mathbf{n}}|_\Gamma = \frac{1}{Re}\mathbf{n} \cdot \nabla^2 \mathbf{u} \quad \text{or} \quad \frac{\partial p}{\partial \boldsymbol{\tau}}|_\Gamma = \frac{1}{Re}\boldsymbol{\tau} \cdot \nabla^2 \mathbf{u} \tag{3.3}$$

So both the Dirichlet and Neumann boundary conditions seem plausible for the pressure. For the most natural choice with the Neumann boundary condition, it is difficult to enforce this condition accurately and maintain the consistency between this condition with the PPE in a discrete setting (E & Liu, 2003).

Moreover, solely replacing (3.1b) with this PPE does not result in a system equivalent to NSE (3.1). Subtracting (3.2) from the divergence of (3.1a) gives

$$\nabla \cdot \left(\frac{\partial \mathbf{u}}{\partial t} - \frac{1}{Re}\nabla^2 \mathbf{u} \right) = 0 \tag{3.4}$$

After commuting the operator, we obtain a transient equation for $\Theta \equiv \nabla \cdot \mathbf{u}$

$$\frac{\partial \Theta}{\partial t} - \frac{1}{Re}\nabla^2 \Theta = 0 \tag{3.5}$$

As indicated by Gresho & Sani (1987) and Johnston & Liu (2004), to obtain an equivalent system to (3.1) additional conditions need to be enforced with (3.5). Some choices can be made as follows:

(***i***) Enforce a divergence-free boundary condition, either Dirichlet or Neumann type. Since Θ is initially zero, it will remain zero ($\Theta \equiv 0$ for $t \geqslant 0$). The additional

conditions are

$$\Theta|_\Gamma = 0 \quad \text{or} \quad \frac{\partial \Theta}{\partial \mathbf{n}}|_\Gamma = 0 \quad \text{for } t > 0 \tag{3.6a}$$

$$\Theta(t = 0) = 0 \tag{3.6b}$$

In this case, a system equivalent to (3.1) consists of (3.1a), (3.2), (3.1c) and (3.6).

(***ii***) Retain the viscosity term in the PPE

$$\nabla^2 p = -\nabla \cdot (\mathbf{u} \cdot \nabla \mathbf{u}) + \frac{1}{Re} \nabla^2 (\nabla \cdot \mathbf{u}) \tag{3.7}$$

Then the additional conditions to be satisfied are

$$\frac{\partial \Theta}{\partial t} = 0 \quad \text{for } t > 0 \tag{3.8a}$$

$$\Theta(t = 0) = 0 \tag{3.8b}$$

In this case, a system equivalent to (3.1) consists of (3.1a), (3.7), (3.1c) and (3.8). However, a third order derivative is formulated in (3.7), which increases the complexity of discretization.

(***iii***) Write the viscous term in rotational form

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p - \frac{1}{Re} \nabla \times \nabla \times \mathbf{u} \tag{3.9}$$

where $\nabla^2 \mathbf{u} = \nabla(\nabla \cdot \mathbf{u}) - \nabla \times \nabla \times \mathbf{u}$ and the incompressibility condition (3.1b) are considered. In this case a system equivalent to (3.1) involves (3.9), (3.2), (3.8) and (3.1c). This scheme is well suited for the explicit treatment of the viscous term. However, implicit treatment would result in a coupled system involving all the components of $\mathbf{u}$, which is costly.

(***iv***) Enforce the following Neumann boundary condition for the pressure

$$\frac{\partial p}{\partial \mathbf{n}} = -\frac{1}{Re} \mathbf{n} \cdot (\nabla \times \nabla \times \mathbf{u}) \quad \text{on } \Gamma \tag{3.10}$$

Then a system equivalent to (3.1) includes (3.1a), (3.2), (3.10) and (3.1c). To prove this, we take the normal component of the momentum equation (3.1a) and use the identity $\nabla^2 \mathbf{u} = \nabla(\nabla \cdot \mathbf{u}) - \nabla \times \nabla \times \mathbf{u}$ for the viscous term, which yields

$$\frac{\partial p}{\partial \mathbf{n}} = -\frac{1}{Re} \mathbf{n} \cdot (\nabla \times \nabla \times \mathbf{u}) + \frac{\partial (\nabla \cdot \mathbf{u})}{\partial \mathbf{n}} \quad \text{on } \Gamma \tag{3.11}$$

Subtracting (3.11) from (3.10) gives

$$\frac{\partial \Theta}{\partial \mathbf{n}} = 0 \quad \text{on } \Gamma \tag{3.12}$$

which goes back to the Neumann boundary condition of (3.6).
This boundary condition is first introduced by Orszag *et al.* (1986) and is referred as an accurate pressure boundary condition to assess the accuracy of the pressure-velocity decoupling methods presented in this thesis.

3.3 Time discretization

The governing equations (3.1) are discretized in time as follows

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \mathbf{u}^p \cdot \nabla \mathbf{u}^q = -\nabla p^{n+1} + \frac{1}{Re} \nabla^2 \mathbf{u}^k \tag{3.13a}$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0 \tag{3.13b}$$

$$\mathbf{u}^{n+1}|_\Gamma = 0 \tag{3.13c}$$

where p, q, k are the time index for the convection and diffusion terms, respectively. The pressure is treated implicitly in order to ensure the divergence-free condition at the new time level. Some choices of these parameters can be made as follows

- Implicit formulation:
  - Fully implicit formulation (non-linear) : $p = q = k = n + 1$;
  - Linearly implicit formulation: $p = n$, $q = k = n + 1$;
- Explicit formulation: $p = q = k = n$;
- Semi-implicit formulation: $p = q = n$, $k = n + 1$.

(*i*) Implicit formulation

If we set $p = q = k = n + 1$, the discretized equations are fully implicit and always stable, which indicates that large time steps are allowed with this implicit formulation. However it requires non-linear iterations. This could be expensive in computation and the convergence is not always ensured.
One way to prevent non-linear iterations is to linearise the equations by setting $p = n$ (Braza *et al.*, 1986; Dong & Shen, 2010). This results in a non-symmetric coefficient matrix for the velocity, which can be solved with sophisticated Krylov

solvers like the bi-conjugate gradient stabilized (Bi-CGSTAB) method or the generalized minimal residual (GMRES) method. However, the velocity coefficient matrix needs to be re-computed every time step. It becomes very costly when the grid number increases. Moreover, larger time step can cause bigger truncation errors in the solution.

(*ii*) Explicit formulation

If we set $p = q = k = n$, the discretized equations are explicit and no iterations are needed for the velocity. Explicit formulation seems to be very efficient. However, the time step should be kept small enough to maintain stability. The constraints on the time step in two dimensions are the diffusive stability condition

$$\Delta t \leqslant \frac{Re}{2}\left(\frac{1}{\Delta x_{\min}^2}+\frac{1}{\Delta y_{\min}^2}\right)^{-1} \tag{3.14}$$

and the convective stability condition of the usual CFL (Courant–Friedrichs–Lewy) type

$$\Delta t \leqslant \min\left\{\frac{\Delta x_{\min}}{u_{\max}}, \frac{\Delta y_{\min}}{v_{\max}}\right\} \tag{3.15}$$

Those conditions are obtained through the stability analysis (see Appendix A for details). It is easy to see that the diffusive constraint is very severe, since reducing the mesh size by half requires a four times smaller time step. It becomes more severe as the dimension increases. At low Reynolds number regime, the time constraint due to (3.14) dominates (3.15). It might be thought that for moderate to high Reynolds number flows, the diffusive stability condition is less restrictive. However in practice, the grid spacing is usually kept small for high Reynolds number flows to capture the small turbulence. This is true when performing the direct numerical simulation (DNS) or the large eddy simulation (LES) (Fadlun *et al.*, 2000).

(*iii*) Semi-implicit formulation

In present thesis, a compromised solution is employed by using a semi-implicit scheme, namely the diffusive term is treated implicitly ($k = n + 1$) for stability and the convective term is treated explicitly ($p = q = n$) for simplicity. By doing so, only the CFL condition needs to be satisfied for the overall scheme

$$\Delta t \leqslant \min\left\{\frac{\Delta x_{\min}}{u_{\max}}, \frac{\Delta y_{\min}}{v_{\max}}\right\} \tag{3.16}$$

Note that the CFL condition is in general a necessary but not a sufficient condition for the NSE.

3.4 High order scheme

Previous time discretization is only first order accurate in time. A large class of time stepping schemes can be employed for achieving high order accuracy for the semi-implicit formulation, such as

- Convective term: the Runge–Kutta methods (RK), the Adams-Bashforth methods (AB) or the extrapolation (EP);
- Diffusive term: the backward difference formula (BDF) or the Adams-Moulton methods (AM);

The second order time stepping scheme is frequently used in the literature, for example

- The RK3 for the convection and the Crank-Nicolson (CN or AM2) for the diffusion (Fadlun *et al.*, 2000; Uhlmann, 2005; Kempe & Fröhlich, 2012a). With this combination a relative larger CFL number is found (CFL $< \sqrt{3}$);
- The AB2 for the convection and the CN for the diffusion (Kim & Moin, 1985; Perot, 1993; Johnston & Liu, 2004);
- The BDF2 for the diffusion and EP2 for the convection (Guermond & Shen, 2003b).

The difference between the RK methods and the AB methods is that the former requires more calculations within each time step but less storages, while the latter needs more storages but less calculations. In the present work, the explicit second order Adams-Bashforth (AB2) scheme with the implicit second order Crank-Nicolson (CN) scheme are used, which yields a second order time accurate scheme as follows

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t}+\left[\frac{3}{2}\mathcal{N}(\mathbf{u}^{n})-\frac{1}{2}\mathcal{N}(\mathbf{u}^{n-1})\right]=-\mathcal{G}p^{n+1}+\frac{1}{2Re}\mathcal{L}(\mathbf{u}^{n+1}+\mathbf{u}^{n})+\mathcal{O}(\Delta t^{2}) \tag{3.17a}$$

$$\mathcal{D}\mathbf{u}^{n+1}=0 \tag{3.17b}$$

$$\mathbf{u}^{n+1}|_{\Gamma}=0 \tag{3.17c}$$

where $\mathcal{L}, \mathcal{N}, \mathcal{G}, \mathcal{D}$ represent the discrete Laplacian, convective, gradient, divergence operators, respectively.

3.5 Projection methods

In the present thesis, the projection method (or fractional step method, time splitting method) is employed for the pressure-velocity decoupling, which is non-iterative and based on a mathematical foundation. The evolution of the velocity only consists of two substeps at each time level, i.e., the prediction and the projection. We will present this method in detail in the following contents along with the error analysis.

3.5.1 Helmholtz-Hodge decomposition

The projection method is based on the mathematical foundation of Helmholtz-Hodge decomposition, which indicates that any smooth vector $\mathbf{v}$ can be decomposed into a divergence-free component $\mathbf{v}_d$ and a curl-free component $\mathbf{v}_c$ (see Figure 3.2)

$$\mathbf{v} = \mathbf{v}_d + \mathbf{v}_c \tag{3.18}$$

where

$$\nabla \cdot \mathbf{v}_d = 0 \tag{3.19a}$$

$$\nabla \times \mathbf{v}_c = 0 \tag{3.19b}$$

and the curl-free component can be further expressed as a gradient of a potential field

$$\mathbf{v}_c = \nabla \phi \tag{3.20}$$

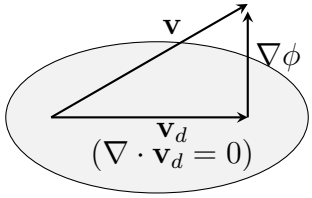


Figure 3.2: Illustration of Helmholtz-Hodge decomposition.

The divergence-free component $\mathbf{v}_d$ can be obtained by taking the divergence operator to (3.18) while using (3.19a) and (3.20)

$$\nabla^2 \phi = \nabla \cdot \mathbf{v} \tag{3.21a}$$

$$\mathbf{v}_d = \mathbf{v} - \nabla\phi \tag{3.21b}$$

In projection methods, $\mathbf{v}$ is usually obtained in the prediction step and then projected into the divergence-free field in the projection step.

3.5.2 First order accurate pressure-correction method

The first projection method is proposed by Chorin (1968) and Témam (1969), which is performed in two substeps:

• Step 1, prediction for $\hat{\mathbf{u}}^{n+1}$

$$\frac{\hat{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t} + \left[\frac{3}{2}\mathcal{N}(\mathbf{u}^n) - \frac{1}{2}\mathcal{N}(\mathbf{u}^{n-1})\right] = \frac{1}{2Re}\mathcal{L}(\hat{\mathbf{u}}^{n+1} + \mathbf{u}^n) \tag{3.22a}$$

$$\hat{\mathbf{u}}^{n+1}|_\Gamma = 0 \tag{3.22b}$$

where the physical boundary condition is assigned to the intermediate velocity.

• Step 2, projection for $\mathbf{u}^{n+1}$ and p^{n+1}

$$\frac{\mathbf{u}^{n+1} - \hat{\mathbf{u}}^{n+1}}{\Delta t} = -\mathcal{G}p^{n+1} \tag{3.23a}$$

$$\mathcal{D}\mathbf{u}^{n+1} = 0 \tag{3.23b}$$

$$\mathbf{u}^{n+1} \cdot \mathbf{n}|_\Gamma = 0 \tag{3.23c}$$

By applying the divergence operator to (3.23a) and using the incompressibility condition, step 2 can be written as

$$\mathcal{L}p^{n+1} = \frac{1}{\Delta t}\mathcal{D}\hat{\mathbf{u}}^{n+1} \tag{3.24a}$$

$$\frac{\partial p^{n+1}}{\partial \mathbf{n}}|_\Gamma = 0 \tag{3.24b}$$

$$\mathbf{u}^{n+1} = \hat{\mathbf{u}}^{n+1} - \Delta t\mathcal{G}p^{n+1} \tag{3.24c}$$

Note that (3.24c) can be written as

$$\mathbf{u}^{n+1} = P_H\hat{\mathbf{u}}^{n+1} \tag{3.25}$$

where P_H is the L^2 projection onto the H space of solenoidal vector fields

$$H = \left\{\mathbf{u} \in L^2(\Omega)^d : \nabla \cdot \mathbf{u} = 0, \mathbf{u} \cdot \mathbf{n}|_\Gamma = 0\right\} \tag{3.26}$$

This scheme is very attractive for its efficiency in simulating large scale problems.

At each time step, only a sequence of decoupled elliptic equations for the pressure and velocity need to be solved.

However the overall scheme is found to be first order accurate after the time splitting, even though a second order scheme is applied to the original system (3.17). To study the splitting error, we sum up the two split equations (3.22a) and (3.23a) and compare to the original system (3.17a). The splitting error is found to be (Perot, 1993)

$$\frac{1}{2Re}\mathcal{L}(\hat{\mathbf{u}}^{n+1} - \mathbf{u}^{n+1}) = \frac{\Delta t}{2Re}\mathcal{L}\mathcal{G}p^{n+1} \tag{3.27}$$

which is due to the time splitting scheme with the implicit treatment of the viscous term. As mentioned before, explicit treatment of the viscous term would require a very small time step to maintain stability. This error is irreducible, hence using a higher-order time stepping scheme will not improve the overall accuracy. Furthermore, an artificial homogeneous Neumann boundary condition (3.24b) is enforced on the pressure, compared to the exact pressure boundary condition (3.10). This artificial Neumann boundary condition induces a numerical boundary layer that prevents the scheme to be fully first order (E & Liu, 1995; Guermond *et al.*, 2006). In the review paper of Guermond *et al.* (2006) this first order accurate method is termed as the non-incremental pressure-correction method and satisfies the following error estimates:

$$\begin{aligned} &||\mathbf{u} - \mathbf{u}_{\Delta t}||_{l^\infty(L^2(\Omega)^d)} + ||\mathbf{u} - \hat{\mathbf{u}}_{\Delta t}||_{l^\infty(L^2(\Omega)^d)} \leqslant c(\mathbf{u}, p, T)\Delta t \\ &||p - p_{\Delta t}||_{l^\infty(L^2(\Omega)^d)} + ||\mathbf{u} - \hat{\mathbf{u}}_{\Delta t}||_{l^\infty(H^1(\Omega)^d)} \leqslant c(\mathbf{u}, p, T)\Delta t^{1/2} \end{aligned} \tag{3.28}$$

where ($\mathbf{u}_{\Delta t}$, $p_{\Delta t}$) represents the numerical solution for the semi-discrete projection method.

3.5.3 Formally second order accurate pressure-correction method

By observing that the pressure gradient is not used in (3.22a), we then add an known old value of the pressure gradient in the prediction step to obtain a formally second order accurate scheme. A large class of projection methods can be classified into this group, such as the methods of Goda (1979), Van Kan (1986), Braza *et al.* (1986) and Bell *et al.* (1989). The formally second order method can be expressed as

• Step 1, prediction for $\hat{\mathbf{u}}^{n+1}$

$$\frac{\hat{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t} + \left[\frac{3}{2}\mathcal{N}(\mathbf{u}^n) - \frac{1}{2}\mathcal{N}(\mathbf{u}^{n-1})\right] = -\mathcal{G}p^n + \frac{1}{2Re}\mathcal{L}(\hat{\mathbf{u}}^{n+1} + \mathbf{u}^n) \tag{3.29a}$$

$$\hat{\mathbf{u}}^{n+1}|_\Gamma = 0 \tag{3.29b}$$

where the physical boundary condition is still assigned to the intermediate velocity.

• Step 2, projection for $\mathbf{u}^{n+1}$ and p^{n+1}

$$\frac{\mathbf{u}^{n+1} - \hat{\mathbf{u}}^{n+1}}{\Delta t} = -\mathcal{G}\phi^{n+1} \tag{3.30a}$$

$$\mathcal{D}\mathbf{u}^{n+1} = 0 \tag{3.30b}$$

$$\mathbf{u}^{n+1} \cdot \mathbf{n}|_\Gamma = 0 \tag{3.30c}$$

where ϕ represents the pseudo pressure. By applying the divergence operator to (3.30a) along with the divergence-free condition, the Step 2 can be written as

$$\mathcal{L}\phi^{n+1} = \frac{1}{\Delta t}\mathcal{D}\hat{\mathbf{u}}^{n+1} \tag{3.31a}$$

$$\frac{\partial \phi^{n+1}}{\partial \mathbf{n}}|_\Gamma = 0 \tag{3.31b}$$

$$\mathbf{u}^{n+1} = \hat{\mathbf{u}}^{n+1} - \Delta t \mathcal{G}\phi^{n+1} \tag{3.31c}$$

where the homogeneous Neumann boundary condition is enforced on ϕ. The final pressure is then updated by

$$p^{n+1} = p^n + \phi^{n+1} \tag{3.32}$$

To study the splitting error, we sum up (3.29a) and (3.30a) and then compare to the original system (3.17a). Since the pseudo pressure is the approximation of $\phi^{n+1} = p^{n+1} - p^n = \Delta t \partial p / \partial t$, the splitting error is second order in time (Perot, 1993; Armfield & Street, 2002)

$$\frac{1}{2Re}\mathcal{L}(\hat{\mathbf{u}}^{n+1} - \mathbf{u}^{n+1}) = \frac{\Delta t}{2Re}\mathcal{L}\mathcal{G}\phi^{n+1} = \frac{\Delta t^2}{2Re}\mathcal{L}\mathcal{G}\frac{\partial p}{\partial t} \tag{3.33}$$

However the artificial Neumann boundary condition of ϕ^{n+1} (3.31b) implies that

$$\frac{\partial p^{n+1}}{\partial \mathbf{n}}|_\Gamma = \frac{\partial p^n}{\partial \mathbf{n}}|_\Gamma = \cdots = \frac{\partial p^0}{\partial \mathbf{n}}|_\Gamma \tag{3.34}$$

is enforced on the final pressure. This pressure boundary condition is not physical compared to the exact pressure boundary condition (3.10). Hence it introduces a numerical boundary layer that prevents the scheme to be fully second order. This error is irreducible, hence using a higher order time stepping scheme will not improve the overall accuracy. This formally second order scheme is called the standard incremental pressure-correction scheme in Guermond *et al.* (2006) and

satisfies the following error estimates:

$$\begin{aligned} &||\mathbf{u}-\mathbf{u}_{\Delta t}||_{l^2(L^2(\Omega)^d)}+||\mathbf{u}-\hat{\mathbf{u}}_{\Delta t}||_{l^2(L^2(\Omega)^d)}\leqslant c(\mathbf{u},p,T)\Delta t^2 \\ &||p-p_{\Delta t}||_{l^\infty(L^2(\Omega)^d)}+||\mathbf{u}-\hat{\mathbf{u}}_{\Delta t}||_{l^\infty(H^1(\Omega)^d)}\leqslant c(\mathbf{u},p,T)\Delta t \end{aligned} \tag{3.35}$$

3.5.4 Second order accurate pressure-correction method

Timmermans *et al.* (1996) proposed a slightly modified algorithm to achieve a second order accurate scheme. By considering the identity $\nabla^2\mathbf{u} = \nabla(\nabla\cdot\mathbf{u}) - \nabla\times\nabla\times\mathbf{u}$, the error term (3.33) can be rewritten as

$$\frac{1}{2Re}\mathcal{L}(\hat{\mathbf{u}}^{n+1}-\mathbf{u}^{n+1}) = \frac{1}{2Re}\mathcal{G}(\mathcal{D}\hat{\mathbf{u}}^{n+1}) \tag{3.36}$$

where $\nabla\times\nabla\times\hat{\mathbf{u}}^{n+1} = \nabla\times\nabla\times\mathbf{u}^{n+1}$ is used, which can be verified by the Helmholtz-Hodge decomposition. Now the error term in this form is absorbed into the pressure

$$p^{n+1} = p^n + \phi^{n+1} - \frac{1}{2Re}\mathcal{D}\hat{\mathbf{u}}^{n+1} \tag{3.37}$$

As a result, only an inexact tangential boundary condition is applied to the velocity due to the splitting, i.e.,

$$\mathbf{u}^{n+1}\cdot\boldsymbol{\tau} \neq 0 \tag{3.38}$$

The pressure boundary condition is found to be consistent with (3.10). Inserting (3.37) into (3.29a) and applying the boundary conditions gives

$$0 = \mathbf{n}\cdot\mathcal{G}p^n = \mathbf{n}\cdot\mathcal{G}\left[p^{n+1}-\phi^{n+1}+\frac{1}{2Re}\mathcal{D}\hat{\mathbf{u}}^{n+1}\right] \tag{3.39}$$

Considering the homogeneous Neumann boundary conditions (3.31b) for ϕ, we have

$$\frac{\partial p^{n+1}}{\partial\mathbf{n}} = -\frac{1}{2Re}\mathbf{n}\cdot\mathcal{G}(\mathcal{D}\hat{\mathbf{u}}^{n+1}) \quad \text{on } \Gamma \tag{3.40}$$

where $\mathbf{n}\cdot\mathcal{G}p^{n+1} \equiv \partial p^{n+1}/\partial\mathbf{n}$. We donate $\mathcal{RR}$ the operator of $\nabla\times\nabla\times$ and use the identity $\mathcal{L}\mathbf{u} = \mathcal{G}(\mathcal{D}\mathbf{u}) - \mathcal{RR}\mathbf{u}$ again

$$\frac{\partial p^{n+1}}{\partial\mathbf{n}} = -\frac{1}{2Re}\mathbf{n}\cdot(\mathcal{L}\hat{\mathbf{u}}^{n+1}+\mathcal{RR}\hat{\mathbf{u}}^{n+1}) \quad \text{on } \Gamma \tag{3.41}$$

To recall that the physical boundary conditions are also assigned to the intermediate velocity and $\mathcal{RR}\hat{\mathbf{u}}^{n+1} = \mathcal{RR}\mathbf{u}^{n+1}$, then we obtain

$$\frac{\partial p^{n+1}}{\partial\mathbf{n}} = -\frac{1}{2Re}\mathbf{n}\cdot\mathcal{RR}\mathbf{u}^{n+1} \quad \text{on } \Gamma \tag{3.42}$$

which is consistent with (3.10) for the second order time discretization.

An alternative view of this method is that it solves a different form of the momentum equation, namely

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p - \frac{1}{Re} \nabla \times \nabla \times \mathbf{u} \tag{3.43}$$

In Guermond *et al.* (2006) this method is termed as the rotational incremental pressure-correction method, because the operator $\nabla \times \nabla \times$ plays a key role. The method satisfies the following error estimates

$$\begin{aligned} &||\mathbf{u} - \mathbf{u}_{\Delta t}||_{l^2(L^2(\Omega)^d)} + ||\mathbf{u} - \hat{\mathbf{u}}_{\Delta t}||_{l^2(L^2(\Omega)^d)} \leqslant c\Delta t^2 \\ &||\mathbf{u} - \mathbf{u}_{\Delta t}||_{l^2(H^1(\Omega)^d)} + ||\mathbf{u} - \hat{\mathbf{u}}_{\Delta t}||_{l^2(H^1(\Omega)^d)} + ||p - p_{\Delta t}||_{l^2(L^2(\Omega)^d)} \leqslant c\Delta t^{3/2} \end{aligned} \tag{3.44}$$

where ($\mathbf{u}_{\Delta t}$, $p_{\Delta t}$) is the numerical solution for the semi-discrete projection method.

Moreover it is possible to generalize this method to r-th order, which can be summarized as

• Step 1, prediction for $\hat{\mathbf{u}}^{n+1}$

$$\frac{\hat{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t} + \mathcal{N}(\mathbf{u}^\star) = -\mathcal{G}p^n + \frac{1}{Re}\mathcal{L}\hat{\mathbf{u}}^\star \tag{3.45a}$$

$$\hat{\mathbf{u}}^{n+1}|_\Gamma = 0 \tag{3.45b}$$

where $\mathcal{N}(\mathbf{u}^\star)$ and $\mathcal{L}\hat{\mathbf{u}}^\star$ are the non-linear terms with the r-th order Adams-Bashforth methods and the linear terms with the r-th order Adams-Moulton methods respectively, i.e.,

$$\mathcal{N}(\mathbf{u}^\star) = \sum_{k=0}^{r-1} \gamma_k \mathcal{N}(\mathbf{u}^{n-k}), \quad \mathcal{L}\hat{\mathbf{u}}^\star = \alpha_k \mathcal{L}\hat{\mathbf{u}}^{n+1} + \sum_{k=0}^{r-2} \beta_k \mathcal{L}\mathbf{u}^{n-k} \tag{3.46}$$

where α_k, β_k, γ_k are the suitable weights, for example

$$\mathcal{N}(\mathbf{u}^\star) = \begin{cases} \mathcal{N}(\mathbf{u}^n) & \text{for } r = 1 \text{ (Forward Euler)} \\ \frac{3}{2}\mathcal{N}(\mathbf{u}^n) - \frac{1}{2}\mathcal{N}(\mathbf{u}^{n-1}) & \text{for } r = 2 \text{ (AB2)} \\ \frac{23}{12}\mathcal{N}(\mathbf{u}^n) - \frac{16}{12}\mathcal{N}(\mathbf{u}^{n-1}) + \frac{5}{12}\mathcal{N}(\mathbf{u}^{n-2}) & \text{for } r = 3 \text{ (AB3)} \\ \frac{55}{24}\mathcal{N}(\mathbf{u}^n) - \frac{59}{24}\mathcal{N}(\mathbf{u}^{n-1}) + \frac{37}{24}\mathcal{N}(\mathbf{u}^{n-2}) - \frac{9}{24}\mathcal{N}(\mathbf{u}^{n-3}) & \text{for } r = 4 \text{ (AB4)} \end{cases} \tag{3.47}$$

and

$$\mathcal{L}\hat{\mathbf{u}}^{\star} = \begin{cases} \mathcal{L}\hat{\mathbf{u}}^{n+1} & \text{for } r = 1 \text{ (Backward Euler)} \\ \frac{1}{2}\mathcal{L}\hat{\mathbf{u}}^{n+1} + \frac{1}{2}\mathcal{L}\mathbf{u}^{n} & \text{for } r = 2 \text{ (Crank-Nicolson)} \\ \frac{5}{12}\mathcal{L}\hat{\mathbf{u}}^{n+1} + \frac{8}{12}\mathcal{L}\mathbf{u}^{n} - \frac{1}{12}\mathcal{L}\mathbf{u}^{n-1} & \text{for } r = 3 \text{ (AM3)} \\ \frac{9}{24}\mathcal{L}\hat{\mathbf{u}}^{n+1} + \frac{19}{24}\mathcal{L}\mathbf{u}^{n} - \frac{5}{24}\mathcal{L}\mathbf{u}^{n-1} + \frac{1}{24}\mathcal{L}\mathbf{u}^{n-2} & \text{for } r = 4 \text{ (AM4)} \end{cases} \tag{3.48}$$

• Step 2, projection for $\mathbf{u}^{n+1}$ and p^{n+1}

$$\frac{\mathbf{u}^{n+1} - \hat{\mathbf{u}}^{n+1}}{\Delta t} = -\mathcal{G}\phi^{n+1} \tag{3.49a}$$

$$\mathcal{D}\mathbf{u}^{n+1} = 0 \tag{3.49b}$$

$$\mathbf{u}^{n+1} \cdot \mathbf{n}|_{\Gamma} = 0 \tag{3.49c}$$

Applying the divergence operator to (3.49a) with the incompressibility condition leads to

$$\mathcal{L}\phi^{n+1} = \frac{1}{\Delta t}\mathcal{D}\hat{\mathbf{u}}^{n+1} \tag{3.50a}$$

$$\frac{\partial \phi^{n+1}}{\partial \mathbf{n}}|_{\Gamma} = 0 \tag{3.50b}$$

$$\mathbf{u}^{n+1} = \hat{\mathbf{u}}^{n+1} - \Delta t \mathcal{G}\phi^{n+1} \tag{3.50c}$$

$$p^{n+1} = p^{n} + \phi^{n+1} - \frac{\chi}{Re}\mathcal{D}\hat{\mathbf{u}}^{n+1} \tag{3.50d}$$

where $\chi = 1, 1/2, 5/12, 9/24$ for the first order Adams-Moulton scheme (backward Euler), the second order Adams-Moulton scheme (Crank-Nicolson), the third order Adams-Moulton scheme (AM3) and the fourth order Adams-Moulton scheme (AM4) respectively.

3.5.5 Issues of the projection methods

(***i***) Controversy may arise over which velocity is correct in the projection methods. The final velocity $\mathbf{u}^{n+1}$ is divergence-free but does not satisfy the correct tangential boundary condition, while the intermediate velocity $\hat{\mathbf{u}}^{n+1}$ satisfies the boundary condition but is not divergence-free. From the accuracy point of view, both velocities yield the same error estimates (Guermond *et al.*, 2006). In practice, by simple algebraic substitutions the velocity $\mathbf{u}^{n+1}$ can be completely avoided for computation, which is usually the case in the velocity-correction projection methods of Guermond

& Shen (2003b). The properties of the two velocity fields are summarized as follows

$$\begin{aligned} &\nabla \cdot \mathbf{u}^{n+1} = 0, \quad \mathbf{u}^{n+1} \cdot \mathbf{n}|_\Gamma = 0, \quad \mathbf{u}^{n+1} \cdot \boldsymbol{\tau} \neq 0 \\ &\nabla \cdot \hat{\mathbf{u}}^{n+1} \neq 0, \quad \hat{\mathbf{u}}^{n+1} \cdot \mathbf{n}|_\Gamma = 0, \quad \hat{\mathbf{u}}^{n+1} \cdot \boldsymbol{\tau} = 0 \end{aligned} \tag{3.51}$$

(***ii***) Another important issue is that a Poisson equation is solved with the Neumann boundary condition applied to all the boundaries at each time step. For a solution to exist, the compatibility condition or the solvability condition must be satisfied. For example, considering the Poisson equation

$$\begin{aligned} \nabla^2 \phi &= f \qquad \text{in } \Omega \\ \text{with} \quad \nabla \phi \cdot \mathbf{n} &= 0 \quad \text{on } \Gamma \end{aligned} \tag{3.52}$$

To ensure the above problem has solutions, the following condition is needed

$$\int_\Omega f dV = \int_\Omega \nabla^2 \phi dV = \int_\Gamma \nabla \phi \cdot \mathbf{n} dS = 0 \tag{3.53}$$

where the divergence theorem is used. It is easy to confirm that this condition is indeed satisfied in the projection methods, since

$$\int_\Omega \frac{1}{\Delta t} \nabla \cdot \hat{\mathbf{u}}^{n+1} dV = \int_\Gamma \hat{\mathbf{u}}^{n+1} \cdot \mathbf{n} dS = 0 \tag{3.54}$$

However the solution is not unique (up to a constant) for the Poisson equation with all Neumann boundary conditions. To obtain a unique solution, we set a fixed value (e.g. zero) at one node to remove the zero eigenvalue.

(***iii***) Although as mentioned before the scheme is stable under the necessary CFL condition, it can suffer numerical instabilities when decreasing the time step arbitrarily (Guermond & Quartapelle, 1998). The time step is limited by a lower bound of $\Delta t \geqslant c \Delta x^{l+1}$ provided that equal orders of interpolation are employed for the pressure and the velocity, where c is a constant and l is the interpolation order of velocity ($l = 2$ in the present work). Therefore, a relative larger time step is usually selected in the present work based on the CFL condition.

3.5.6 Comparison with other projection methods

The problem associated with the projection methods lies on the fact that boundary conditions are needed for the intermediate velocity and the pressure, which however are quite ambiguous. Inconsistent boundary conditions can significantly decrease the accuracy of the original system and lead to inconsistent solutions.

According to E & Liu (1995) and Brown *et al.* (2001), improvements towards the fractional step methods can be made in three different ways: Using a pressure incremental formulation, which is just described formerly; Applying an accurate boundary condition for the intermediate velocity; Employing a correct pressure boundary condition. One can also avoid the use of fractional step method, therefore the splitting error and the inconsistent boundary conditions will not be generated. In the following contents these methods are summarised.

(*i*) projection method with accurate intermediate velocity boundary condition

Kim & Moin (1985) introduced a second order fractional step scheme by deriving an accurate boundary condition for the intermediate velocity. This method can be summarized as

• Step 1, prediction for $\hat{\mathbf{u}}^{n+1}$ with a non-homogeneous boundary condition

$$\frac{\hat{\mathbf{u}}^{n+1}-\mathbf{u}^{n}}{\Delta t}+\left[\frac{3}{2}\mathcal{N}(\mathbf{u}^{n})-\frac{1}{2}\mathcal{N}(\mathbf{u}^{n-1})\right]=\frac{1}{2Re}\mathcal{L}(\hat{\mathbf{u}}^{n+1}+\mathbf{u}^{n}) \tag{3.55a}$$

$$\hat{\mathbf{u}}^{n+1}|_{\Gamma}=\Delta t\mathcal{G}\phi^{n}|_{\Gamma} \tag{3.55b}$$

Note that the correct boundary condition for the intermediate velocity requires the pressure value. To avoid evaluating the pressure, ϕ is used instead of p, and the boundary condition is still second order accurate since $p=\phi+O(\Delta t/Re)$.

• Step 2, projection for $\mathbf{u}^{n+1}$ and p^{n+1}

$$\frac{\mathbf{u}^{n+1}-\hat{\mathbf{u}}^{n+1}}{\Delta t}=-\mathcal{G}\phi^{n+1} \tag{3.56a}$$

$$\mathcal{D}\mathbf{u}^{n+1}=0 \tag{3.56b}$$

$$\mathbf{u}^{n+1}\cdot\mathbf{n}|_{\Gamma}=0 \tag{3.56c}$$

which is realised by

$$\mathcal{L}\phi^{n+1}=\frac{1}{\Delta t}\mathcal{D}\hat{\mathbf{u}}^{n+1} \tag{3.57a}$$

$$\frac{\partial\phi^{n+1}}{\partial\mathbf{n}}|_{\Gamma}=\frac{\partial\phi^{n}}{\partial\mathbf{n}}|_{\Gamma} \tag{3.57b}$$

$$\mathbf{u}^{n+1}=\hat{\mathbf{u}}^{n+1}-\Delta t\mathcal{G}\phi^{n+1} \tag{3.57c}$$

The pressure is eliminated from the time evolution completely. Therefore, this scheme is named as the pressure-free projection method in Brown *et al.* (2001). The pressure

can be related to ϕ^{n+1} using

$$p^{n+1} = \phi^{n+1} - \frac{\Delta t}{2Re}\mathcal{L}\phi^{n+1} \tag{3.58}$$

which can be interpreted as a time-centered value in order to obtain a second order accurate pressure. As indicated by Guermond *et al.* (2006), the scheme of Kim & Moin (1985) is equivalent to the rotational incremental pressure-correction scheme after appropriate change of variables. However this scheme has been only successfully implemented with spectral or finite difference approximations. It is quite inconvenient for the finite element discretization, as it involves the trace of a gradient as a Dirichlet condition in the prediction step.

(*ii*) projection method with accurate pressure boundary condition

Orszag *et al.* (1986) and Karniadakis *et al.* (1991) proposed the high-order splitting schemes based on the accurate pressure boundary condition. This strategy is also adopted in other velocity-correction projection methods (Guermond & Shen, 2003b; Guermond *et al.*, 2006). The main idea of the velocity-correction schemes is to switch the role of the pressure and the velocity, namely, the projection in performed in the first substep and the viscous term is treated implicitly in the second substep. Similar to the pressure-correction schemes, the velocity-correction methods can be implemented in a non-incremental, standard incremental or rotational incremental way (Guermond *et al.*, 2006). In Guermond & Shen (2003b) and Guermond *et al.* (2006) the BDF scheme is used for the discretization, with no doubt we can rewrite the rotational incremental velocity-correction method with the second order the Adams-Bashforth scheme and the Crank-Nicolson scheme as follows

• Step 1, projection for $\mathbf{u}^{n+1}$ and p^{n+1}

$$\frac{\hat{\mathbf{u}}^{n+1/2} - \hat{\mathbf{u}}^{n}}{\Delta t} + \left[\frac{3}{2}\mathcal{N}(\hat{\mathbf{u}}^{n}) - \frac{1}{2}\mathcal{N}(\hat{\mathbf{u}}^{n-1})\right] = -\frac{1}{Re}\left[\frac{3}{2}\mathcal{R}\mathcal{R}\hat{\mathbf{u}}^{n} - \frac{1}{2}\mathcal{R}\mathcal{R}\hat{\mathbf{u}}^{n-1}\right] \tag{3.59a}$$

$$\frac{\mathbf{u}^{n+1} - \hat{\mathbf{u}}^{n+1/2}}{\Delta t} = -\mathcal{G}p^{n+1} \tag{3.59b}$$

$$\mathcal{D}\mathbf{u}^{n+1} = 0 \tag{3.59c}$$

$$\mathbf{u}^{n+1} \cdot \mathbf{n}|_{\Gamma} = 0 \tag{3.59d}$$

which is solved with the pressure Poisson equation in the form

$$\mathcal{L}p^{n+1} = \frac{1}{\Delta t}\mathcal{D}\hat{\mathbf{u}}^{n+1/2} \tag{3.60a}$$

$$\frac{\partial p^{n+1}}{\partial \mathbf{n}}|_\Gamma = \frac{1}{\Delta t}\mathbf{n}\cdot\hat{\mathbf{u}}^{n+1/2}|_\Gamma = -\mathbf{n}\cdot\frac{1}{Re}\left[\frac{3}{2}\mathcal{R}\mathcal{R}\hat{\mathbf{u}}^{n} - \frac{1}{2}\mathcal{R}\mathcal{R}\hat{\mathbf{u}}^{n-1}\right]|_\Gamma \tag{3.60b}$$

$$\mathbf{u}^{n+1} = \hat{\mathbf{u}}^{n+1/2} - \Delta t\mathcal{G}p^{n+1} \tag{3.60c}$$

where the pressure boundary condition is consistent with (3.10) to the second order accuracy.

• Step 2, correction for $\hat{\mathbf{u}}^{n+1}$

$$\frac{\hat{\mathbf{u}}^{n+1} - \mathbf{u}^{n+1}}{\Delta t} = \frac{1}{2Re}\left(\mathcal{L}\hat{\mathbf{u}}^{n+1} + \mathcal{L}\hat{\mathbf{u}}^{n}\right) + \frac{1}{Re}\left[\frac{3}{2}\mathcal{R}\mathcal{R}\hat{\mathbf{u}}^{n} - \frac{1}{2}\mathcal{R}\mathcal{R}\hat{\mathbf{u}}^{n-1}\right] \tag{3.61a}$$

$$\hat{\mathbf{u}}^{n+1}|_\Gamma = 0 \tag{3.61b}$$

Since the velocity-correction schemes can be viewed as the dual class of the pressure-correction methods, the velocity-correction schemes share almost the same advantages and disadvantages with the pressure-correction counterparts. But the evaluation of the pressure condition (3.60b) is not as convenient as in the pressure-correction projection methods.

(*iii*) Generalized block LU decomposition method

Perot (1993) proposed a fractional step method based on the approximate block LU decomposition. In this method, the boundary conditions on the intermediate variables and the pressure are not required. The overall scheme is solved in an algebraic manner. We rewrite the NSE (3.17) as

$$\begin{pmatrix} \mathcal{A} & \mathcal{G} \\ \mathcal{D} & 0 \end{pmatrix}\begin{pmatrix} \mathbf{u}^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} r^n \\ 0 \end{pmatrix} + \begin{pmatrix} bc_1 \\ bc_2 \end{pmatrix} \tag{3.62}$$

where

$$\mathcal{A} = \frac{1}{\Delta t}I - \frac{1}{2Re}\mathcal{L} \tag{3.63}$$

$$r^n = \frac{1}{\Delta t}\mathbf{u}^n + \frac{1}{2Re}\mathcal{L}\mathbf{u}^n - \left[\frac{3}{2}\mathcal{N}(\mathbf{u}^n) - \frac{1}{2}\mathcal{N}(\mathbf{u}^{n-1})\right] \tag{3.64}$$

and bc_1, bc_2 are the boundary conditions generated from the Laplacian and divergence operators respectively. Solving this system directly is very difficult. Perot (1993) applied the block LU decomposition to an approximate system as follows

$$\begin{pmatrix} \mathcal{A} & (\mathcal{A}\mathcal{B})\mathcal{G} \\ \mathcal{D} & 0 \end{pmatrix}\begin{pmatrix} \mathbf{u}^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} r^n \\ 0 \end{pmatrix} + \begin{pmatrix} bc_1 \\ bc_2 \end{pmatrix} \tag{3.65}$$

and then using the Schur Complement gives

$$\begin{pmatrix} \mathcal{A} & 0 \\ \mathcal{D} & -\mathcal{D}\mathcal{B}\mathcal{G} \end{pmatrix} \begin{pmatrix} I & \mathcal{B}\mathcal{G} \\ 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{u}^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} r^n \\ 0 \end{pmatrix} + \begin{pmatrix} bc_1 \\ bc_2 \end{pmatrix} \tag{3.66}$$

which is commonly written as

$$\mathcal{A}\hat{\mathbf{u}}^{n+1} = r^n + bc_1 \tag{3.67}$$

$$\mathcal{D}\mathcal{B}\mathcal{G}p^{n+1} = \mathcal{D}\hat{\mathbf{u}}^{n+1} - bc_2 \tag{3.68}$$

$$\mathbf{u}^{n+1} = \hat{\mathbf{u}}^{n+1} - \mathcal{B}\mathcal{G}p^{n+1} \tag{3.69}$$

where $\mathcal{B}$ is an approximate inverse of $\mathcal{A}^{-1}$, which determines the overall accuracy. If $\mathcal{B} = \mathcal{A}^{-1}$, the block LU decomposition is equivalent to the Uzawa method. However, the Uzawa method involves nested iteration and many non-zero values enter to $\mathcal{A}^{-1}$ even though $\mathcal{A}$ is sparse. Therefore, Uzawa method is usually avoided in computation. To properly approximate $\mathcal{A}^{-1}$, we first consider the identity for general matrix C and D

$$(C+D)^{-1} = C^{-1} - C^{-1}DC^{-1} + C^{-1}DC^{-1}DC^{-1} - C^{-1}DC^{-1}DC^{-1}DC^{-1} + \cdots \tag{3.70}$$

then we have

$$\begin{aligned} \mathcal{A}^{-1} &= \left[\frac{1}{\Delta t}I + (-\frac{1}{2Re}\mathcal{L})\right]^{-1} \\ &\approx \Delta t I - \Delta t I(-\frac{1}{2Re}\mathcal{L})\Delta t I + \Delta t I(-\frac{1}{2Re}\mathcal{L})\Delta t I(-\frac{1}{2Re}\mathcal{L})\Delta t I - \cdots \\ &= \Delta t I + \frac{\Delta t^2}{2Re}\mathcal{L} + \frac{\Delta t^3}{(2Re)^2}\mathcal{L}^2 + \cdots \end{aligned} \tag{3.71}$$

$\mathcal{B}$ takes the N-th order approximation of $\mathcal{A}^{-1}$

$$\mathcal{A}^{-1} \approx \mathcal{B} = \Delta t I + \frac{\Delta t^2}{2Re}\mathcal{L} + \frac{\Delta t^3}{(2Re)^2}\mathcal{L}^2 + \cdots + \frac{\Delta t^N}{(2Re)^{N-1}}\mathcal{L}^{N-1} = \sum_{j=1}^{N} \frac{\Delta t^j}{(2Re)^{j-1}}\mathcal{L}^{j-1} \tag{3.72}$$

Therefore the splitting error for the inexact factorization depends on the approximation order of $\mathcal{B}$. For example, if $N = 1$ is used the scheme is only first order accurate, and if $N = 2$ is chosen, the scheme is second order accurate.

The generalized block LU decomposition method has gained a lot of popularities in the literature as it does not involve any artificial pressure boundary condition

explicitly. However, the method is only as accurate as the standard incremental pressure-correction scheme and enforces weakly an artificial boundary condition (Guermond *et al.*, 2006).

(***iv***) **Exact fractional step method**

Chang *et al.* (2002) proposed an exact fractional step method by using the discrete streamfunction s as the unknown variable, hence this method is also termed as the discrete streamfunction or then nullspace approach. The discrete streamfunction s is defined as

$$\mathbf{u} = \mathcal{C}s \tag{3.73}$$

where $\mathcal{C}$ is a curl operator, which is constructed with column vectors corresponding to the basis of the nullspace of $\mathcal{D}$. Therefore the operator $\mathcal{D}$ and $\mathcal{C}$ enjoy the following relation

$$\mathcal{D}\mathcal{C} = 0 \tag{3.74}$$

which implies that the divergence-free condition is satisfied automatically in this method because $\mathcal{D}\mathbf{u} = \mathcal{D}\mathcal{C}s = 0$. Note that the divergence operator is the negative transpose of the gradient operator in the staggered mesh, therefore the gradient operator is the null space of another matrix operator $\mathcal{R}$, since $(\mathcal{D}\mathcal{C})^{\mathrm{T}} = \mathcal{C}^{\mathrm{T}}\mathcal{D}^{\mathrm{T}} = -\mathcal{R}\mathcal{G} = 0$, where $\mathcal{R}$ is the rotation matrix operator.
By pre-multiplying the momentum equation with $\mathcal{C}^{\mathrm{T}}$, the pressure term is then eliminated totally, since $\mathcal{C}^{\mathrm{T}}\mathcal{G}p = -(\mathcal{D}\mathcal{C})^{\mathrm{T}}p = 0$. Finally, the system is reduced to a single equation for the discrete streamfunction

$$\mathcal{C}^{\mathrm{T}}\mathcal{A}\mathcal{C}s^{n+1} = \mathcal{R}(r^n + bc_1) \tag{3.75}$$

where bc_1 represents the boundary conditions for the momentum equation.
Once the discrete streamfunction is solved, the velocity can be recovered by the definition $\mathbf{u} = \mathcal{C}s$. Even it shares some similarities with the streamfunction-vorticity method, this method is more attractive for the reason that it contains only one variable. As a result in two dimensions only one component needs to be solved. In three dimensions, three components of the streamfunction are solved instead of six components in the streamfunction-vorticity method.
Of course, since it uses the discrete streamfunction as the primary unknown, enforcing the boundary conditions is not straightforward, just like other non-primitive variable formulations.

(*v*) Gauge method

The gauge method is developed by E & Liu (2003) through the gauge transformation. By replacing the pressure with a gauge variable ϕ and introducing an auxiliary field $\mathbf{a} = \mathbf{u} + \nabla\phi$, the fluid momentum equation is changed to

$$\frac{\partial \mathbf{a}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \frac{1}{Re} \nabla^2 \mathbf{a} \tag{3.76}$$

with

$$\nabla^2 \phi = \nabla \cdot \mathbf{a} \tag{3.77}$$

Through comparing to (3.1), ϕ can be related to the pressure by

$$p = \frac{\partial \phi}{\partial t} - \frac{1}{Re} \nabla^2 \phi \tag{3.78}$$

By doing so, we can take advantage of the gauge freedom to assign simple and specific boundary conditions for the gauge field and the auxiliary variable, either

$$\frac{\partial \phi}{\partial \mathbf{n}} = 0, \quad \mathbf{a} \cdot \mathbf{n} = 0, \quad \mathbf{a} \cdot \boldsymbol{\tau} = \frac{\partial \phi}{\partial \boldsymbol{\tau}} \quad \text{on } \Gamma \tag{3.79}$$

or

$$\phi = 0, \quad \mathbf{a} \cdot \mathbf{n} = \frac{\partial \phi}{\partial \mathbf{n}}, \quad \mathbf{a} \cdot \boldsymbol{\tau} = 0 \quad \text{on } \Gamma \tag{3.80}$$

Using the second order semi-implicit discretization and the Neumann boundary conditions, the gauge formulation can be written as

$$\frac{\mathbf{a}^{n+1} - \mathbf{a}^n}{\Delta t} + \left[\frac{3}{2} \mathcal{N}(\mathbf{u}^n) - \frac{1}{2} \mathcal{N}(\mathbf{u}^{n-1}) \right] = \frac{1}{2Re} \mathcal{L}(\mathbf{a}^{n+1} + \mathbf{a}^n) \tag{3.81a}$$

$$\mathbf{a}^{n+1} \cdot \mathbf{n} = 0, \quad \mathbf{a}^{n+1} \cdot \boldsymbol{\tau} = \frac{\partial \phi^n}{\partial \boldsymbol{\tau}} \quad \text{on } \Gamma \tag{3.81b}$$

$$\mathcal{L}\phi^{n+1} = \nabla \cdot \mathbf{a}^{n+1} \tag{3.82a}$$

$$\frac{\partial \phi^{n+1}}{\partial \mathbf{n}} = 0 \quad \text{on } \Gamma \tag{3.82b}$$

$$\mathbf{u}^{n+1} = \mathbf{a}^{n+1} - \nabla \phi^{n+1} \tag{3.83}$$

It is easy to see that the gauge method is not a fractional step method. Therefore the splitting error and the numerical boundary layer that happen in some fractional

step methods will not be generated in the gauge method. Indeed full accuracy has been proved in E & Liu (2003), for example the second order accuracy for the velocity and pressure in this case. Notice that the explicit boundary conditions are used in (3.81b), but this will not influence the accuracy as indicated by Wang & Liu (2000). Another method proposed by Guermond & Shen (2003a), the consist splitting scheme, is found to be equivalent to the gauge method after appropriate change of variables. This method is quite attractive for its accuracy, however, it is not as stable as the fractional step method when including open boundary conditions (Guermond *et al.*, 2006).

(***vi***) **Method based on the explicit treatment of the pressure**

Usually the pressure is discretized implicitly in order to ensure the velocity to be divergence-free at the new time level. For this reason, the decoupling between the pressure and velocity often requires time splitting, but this could result in inconsistency. Contrarily, Johnston & Liu (2004) observed that if the PPE is solved correctly the divergence-free condition can be fulfilled accurately. This allows to treat the pressure term explicitly in the momentum equation and hence the dynamic and kinematic equations are decoupled completely. The most important feature is that no additional time step constraints are established besides the standard CFL condition, provided that the viscous term is treated implicitly and the convection term is treated explicitly. Employing the AB2 discretization for both the convection and pressure terms and the CN discretization for the viscous term, gives

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t}+\left[\frac{3}{2}\mathcal{N}(\mathbf{u}^{n})+\frac{3}{2}\mathcal{G}p^{n}-\frac{1}{2}\mathcal{N}(\mathbf{u}^{n-1})-\frac{1}{2}\mathcal{G}p^{n-1}\right]=\frac{1}{2Re}\mathcal{L}(\mathbf{u}^{n+1}+\mathbf{u}^{n}) \tag{3.84a}$$

$$\mathbf{u}^{n+1}|_{\Gamma}=0 \tag{3.84b}$$

Then the PPE (3.2) is solved with the exact pressure boundary condition

$$\mathcal{L}p^{n+1}=-\mathcal{D}\left(\mathcal{N}(\mathbf{u}^{n+1})\right) \tag{3.85a}$$

$$\frac{\partial p^{n+1}}{\partial \mathbf{n}}|_{\Gamma}=-\frac{1}{Re}\mathbf{n}\cdot\mathcal{R}\mathcal{R}\mathbf{u}^{n+1}|_{\Gamma} \tag{3.85b}$$

This method also features the non fractional step method, therefore all flow variables achieve full second order accuracy in time. However, it is inconvenient to evaluate the boundary condition (3.85b).

3.5.7 Summary

Various variants of the projection methods have been discussed for improving the overall time accuracy. Almost each method has its virtues and drawbacks. The rotational incremental projection methods should be always preferred over the standard incremental and the non-incremental versions, for both the pressure correction and the velocity correction cases. Compared to the rotational velocity correction scheme, the rotational pressure correction scheme is relatively easy to be implemented, as the operator $\mathcal{RR}$ is in fact not involved in the calculation. Therefore it is chosen as the fluid solver in the present thesis. Its space discretization and validation will be given in the following sections.

3.6 Space discretization

3.6.1 Staggered grid

For convenience, we rewrite the NSE as follows

$$\frac{\partial u}{\partial t} + \frac{\partial (u^2)}{\partial x} + \frac{\partial (vu)}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{3.86a}$$

$$\frac{\partial v}{\partial t} + \frac{\partial (uv)}{\partial x} + \frac{\partial (v^2)}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{3.86b}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{3.86c}$$

To discretize the above set of equations, numerous techniques can be employed, such as the finite difference method, finite volume method, finite element method, spectral method, meshless method, etc. The finite difference method is the simplest but it has a lot of difficulties in dealing with complicated domains. In the present work, we will show how this method can handle complicated geometries with the aide of the immersed boundary method.

Therefore, the equations are simply discretized in a regular Cartesian grid using finite difference approximations for the first and second derivatives. However, using a collocated grid where all the variables are defined in the same positions will arise an instability from the odd-even decoupling or spurious pressure mode. This odd-even decoupling comes from the fact that the pressure depends on the first derivatives of the velocity and vice versa. If central differences are used, the pressure at odd or even nodes depends on the velocity at even or odd nodes. This is similar for the velocity. Hence this decoupling leads to a checker-board effect.

To avoid the instability we employ the MAC staggered grid of Harlow & Welch (1965), as shown in Figure 3.3. On the staggered grid all the scalar variables are stored in the cell centers like the pressure, while the velcotiy components are placed at cell faces. Consequently, not all extremal grid points lie on the boundary domain. For example the vertical boundaries have no v component and the horizontal boundaries have no u component. To solve this problem, extra boundary cells or ghost cells, are introduced, which are depicted in the shaded zones in Figure 3.3. If the collocated grid is preferred, special interpolation schemes are needed, such as the Rhie-Chow interpolation (Rhie & Chow, 1983) which is widely used in the commercial software (e.g. FLUENT) and open source codes (e.g. OpenFOAM).
However in the present work, the MAC staggered grid is preferred, as we would like to emphasize the non-body conforming features of the immersed boundary method. In the staggered settings, the immersed solid boundary grid can never coincident with the fluid grid, since the three variables are defined separately in space.

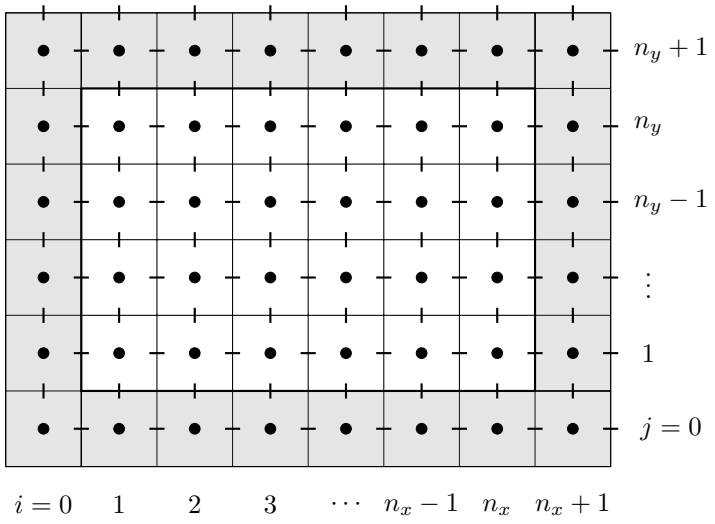


Figure 3.3: The MAC staggered grid arrangement for fluid cells of number $n_x \times n_y$, where the pressure and the velocity components are stored in different locations (Harlow & Welch, 1965). The circles represent the pressure; The horizontal and vertical lines designate the velocity components u and v respectively. The ghost cells are depicted by the shaded zones. Note that the boundary points in the four corners are never used.

3.6.2 Approximation of derivatives

In equation (3.86) the u component is calculated at the midpoint of the right edge of cell (i, j) for $i = 1, \cdots, n_x - 1$ and $j = 1, \cdots, n_y$, shown in Figure 3.4a. Using

central differences, we have

$$\left[\frac{\partial(u^2)}{\partial x}\right]_{i,j} = \frac{1}{\Delta x}\left(\frac{(u_{i,j}+u_{i+1,j})}{2}\frac{(u_{i,j}+u_{i+1,j})}{2} - \frac{(u_{i-1,j}+u_{i,j})}{2}\frac{(u_{i-1,j}+u_{i,j})}{2}\right) \tag{3.87}$$

$$\left[\frac{\partial(vu)}{\partial y}\right]_{i,j} = \frac{1}{\Delta y}\left(\frac{(v_{i,j}+v_{i+1,j})}{2}\frac{(u_{i,j}+u_{i,j+1})}{2} - \frac{(v_{i,j-1}+v_{i+1,j-1})}{2}\frac{(u_{i,j-1}+u_{i,j})}{2}\right) \tag{3.88}$$

$$\left[\frac{\partial^2 u}{\partial x^2}\right]_{i,j} = \frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2} \tag{3.89}$$

$$\left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j} = \frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\Delta y^2} \tag{3.90}$$

$$\left[\frac{\partial p}{\partial x}\right]_{i,j} = \frac{p_{i+1,j}-p_{i,j}}{\Delta x} \tag{3.91}$$

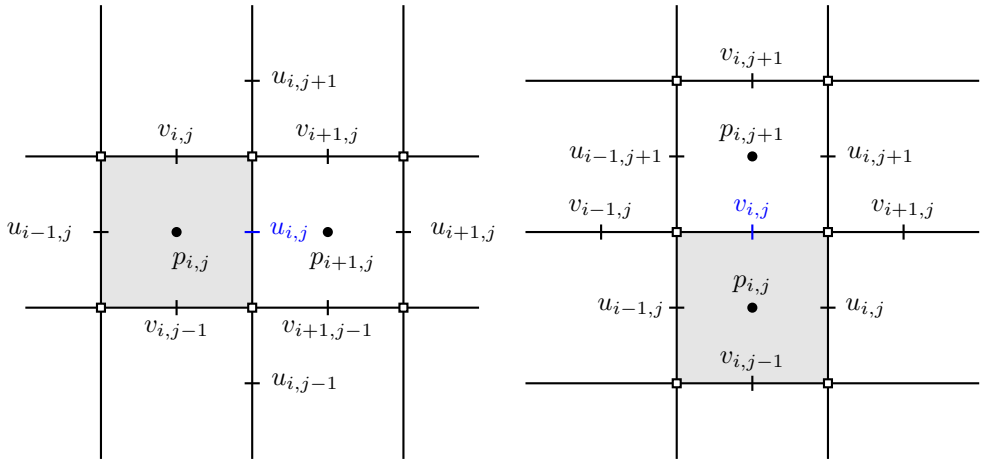
where Δx and Δy are the mesh sizes at x and y direction respectively. However, at high Reynolds numbers or high velocities where convection dominates diffusion, the central differences are unstable when the grid spacing is chosen too coarse. This can cause unphysical oscillations in the solution. The reason for the instability lies in the fact that the diffusion influences the flow along its gradients in all directions, whereas the convection affects the flow only in the flow direction. If the grid size is too large, certain properties of the continuous equation are no longer maintained by the discrete equation. Consequently, central differences introduce a stringent upper limit to the mesh size (Griebel, 1998; Versteeg & Malalasekera, 2007).

This instability can be avoided by using the upwind difference, namely one-side difference. However, this lowers the approximation accuracy to be first order in space and results in extra numerical diffusion or artificial viscosity which actually aides the stability. One compromised solution is to use a weighted average of the central difference and the upwind difference

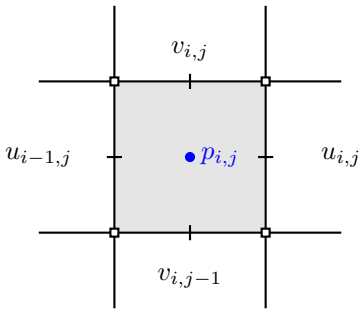
$$\gamma \cdot \text{upwind difference} + (1-\gamma)\cdot \text{central difference} \tag{3.92}$$

where γ is the weight coefficient. For $\gamma = 0$ we recover the central difference and for $\gamma = 1$ the upwind difference is used. In present work, we use the following expression

$$\gamma = \min\left\{\alpha \cdot \max\left(\frac{u_{\max}\Delta t}{\Delta x}, \frac{v_{\max}\Delta t}{\Delta y}\right), 1\right\} \tag{3.93}$$



(a) u (b) v



(c) p

Figure 3.4: Grid stencil for u, v, p at cell (i, j), which is depicted by the shaded zone.

where the factor $\alpha = 1.2$ is taken from experience (Seibold, 2008). With this hybrid scheme, the convection terms are finally discretized as

$$\left[\frac{\partial(u^2)}{\partial x}\right]_{i,j} = \frac{1}{\Delta x}\left(\frac{(u_{i,j}+u_{i+1,j})}{2}\frac{(u_{i,j}+u_{i+1,j})}{2} - \frac{(u_{i-1,j}+u_{i,j})}{2}\frac{(u_{i-1,j}+u_{i,j})}{2}\right) + \gamma\frac{1}{\Delta x}\left(\frac{|u_{i,j}+u_{i+1,j}|}{2}\frac{(u_{i,j}-u_{i+1,j})}{2} - \frac{|u_{i-1,j}+u_{i,j}|}{2}\frac{(u_{i-1,j}-u_{i,j})}{2}\right) \tag{3.94}$$

$$\left[\frac{\partial(vu)}{\partial y}\right]_{i,j} = \frac{1}{\Delta y}\left(\frac{(v_{i,j}+v_{i+1,j})}{2}\frac{(u_{i,j}+u_{i,j+1})}{2} - \frac{(v_{i,j-1}+v_{i+1,j-1})}{2}\frac{(u_{i,j-1}+u_{i,j})}{2}\right) + \gamma\frac{1}{\Delta y}\left(\frac{(|v_{i,j}+v_{i+1,j}|}{2}\frac{(u_{i,j}-u_{i,j+1})}{2} - \frac{(v_{i,j-1}+v_{i+1,j-1})}{2}\frac{(u_{i,j-1}-u_{i,j})}{2}\right) \tag{3.95}$$

Even though other high-order schemes can be used for the same purpose, such as the second order upwind, QUICK (Quadratic Upstream Interpolation for Convective Kinematics) and TVD (Total variation diminishing) schemes, the hybrid scheme is

adopted in the present study as it provides good results.

Similarly, the v component is computed at the midpoint of the upper edge of cell (i,j) for $i = 1, \cdots, n_x$ and $j = 1, \cdots, n_y - 1$, shown in Figure 3.4b. Using the hybrid scheme for the convection terms and central difference for the others, we obtain

$$\left[\frac{\partial(uv)}{\partial x}\right]_{i,j} = \frac{1}{\Delta x}\left(\frac{(u_{i,j}+u_{i,j+1})}{2}\frac{(v_{i,j}+v_{i+1,j})}{2} - \frac{(u_{i-1,j}+u_{i-1,j+1})}{2}\frac{(v_{i-1,j}+v_{i,j})}{2}\right) \gamma\frac{1}{\Delta x}\left(\frac{|u_{i,j}+u_{i,j+1}|}{2}\frac{(v_{i,j}-v_{i+1,j})}{2} - \frac{|u_{i-1,j}+u_{i-1,j+1}|}{2}\frac{(v_{i-1,j}-v_{i,j})}{2}\right) \tag{3.96}$$

$$\left[\frac{\partial(v^2)}{\partial y}\right]_{i,j} = \frac{1}{\Delta y}\left(\frac{(v_{i,j}+v_{i,j+1})}{2}\frac{(v_{i,j}+v_{i,j+1})}{2} - \frac{(v_{i,j-1}+v_{i,j})}{2}\frac{(v_{i,j-1}+v_{i,j})}{2}\right) + \gamma\frac{1}{\Delta y}\left(\frac{|v_{i,j}+v_{i,j+1}|}{2}\frac{(v_{i,j}-v_{i,j+1})}{2} - \frac{|v_{i,j-1}+v_{i,j}|}{2}\frac{(v_{i,j-1}-v_{i,j})}{2}\right) \tag{3.97}$$

$$\left[\frac{\partial^2 v}{\partial x^2}\right]_{i,j} = \frac{v_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{\Delta x^2} \tag{3.98}$$

$$\left[\frac{\partial^2 v}{\partial y^2}\right]_{i,j} = \frac{v_{i,j+1} - 2v_{i,j} + v_{i,j-1}}{\Delta y^2} \tag{3.99}$$

$$\left[\frac{\partial p}{\partial y}\right]_{i,j} = \frac{p_{i,j+1} - p_{i,j}}{\Delta y} \tag{3.100}$$

The continuity equation is evaluated at the centroid of cell (i,j) with central difference, as illustrated in Figure 3.4c

$$\frac{u_{i,j} - u_{i-1,j}}{\Delta x} + \frac{v_{i,j} - v_{i,j-1}}{\Delta y} = 0 \tag{3.101}$$

The other flow variables, such as the streamfunction ψ and vorticity ω, can be also defined on the staggered mesh. To avoid tedious interpolations, these variables are simply evaluated in the upper right corner of cell (i,j) instead of the cell centers. Considering $\partial\psi/\partial y = u$, we compute the discrete streamfunction by

$$\psi_{i,j} = \psi_{i,j-1} + u_{i,j}\Delta y \quad \text{for } i = 1, \cdots, n_x,\ j = 1, \cdots, n_y \tag{3.102}$$

where $\psi_{i,0} = 0$. And the discrete vorticity $\omega_{i,j}$ is calculated by

$$\omega_{i,j} = \frac{v_{i+1,j} - v_{i,j}}{\Delta x} - \frac{u_{i,j+1} - u_{i,j}}{\Delta y} \quad \text{for } i = 1, \cdots, n_x,\ j = 1, \cdots, n_y \tag{3.103}$$

3.7 Implementation of boundary conditions

To solve the discretized equations, boundary conditions are needed at the physical boundaries. In the present work, five boundary conditions are considered, which are inflow, no-slip, free-slip, outflow and periodic boundary conditions.

(***i***) Inflow boundary conditions $\mathbf{u} = \mathbf{U}_\infty$: The velocities are specified explicitly at the boundaries. For the normal velocity, the values are fixed directly on the boundary lines. For the tangent component, we average the values on both sides.

$$\text{Left:} \qquad u_{0,j} = U_\infty, \qquad v_{0,j} = 2V_\infty - v_{1,j} \tag{3.104a}$$

$$\text{right:} \qquad u_{n_x,j} = U_\infty, \qquad v_{n_x+1,j} = 2V_\infty - v_{n_x,j} \tag{3.104b}$$

for $j = 0, \cdots, n_y + 1$ and

$$\text{Bottom:} \qquad u_{i,0} = 2U_\infty - u_{i,1}, \qquad v_{i,0} = V_\infty \tag{3.104c}$$

$$\text{Top:} \qquad u_{i,n_y+1} = 2U_\infty - u_{i,n_y}, \qquad v_{i,n_y} = V_\infty \tag{3.104d}$$

for $i = 0, \cdots, n_x + 1$.

(***ii***) No-slip boundary conditions $\mathbf{u} = \mathbf{U}_b$: The Dirichlet boundary conditions are set similarly.

$$\text{Left:} \qquad u_{0,j} = U_b, \qquad v_{0,j} = 2V_b - v_{1,j} \tag{3.105a}$$

$$\text{right:} \qquad u_{n_x,j} = U_b, \qquad v_{n_x+1,j} = 2V_b - v_{n_x,j} \tag{3.105b}$$

for $j = 0, \cdots, n_y + 1$ and

$$\text{Bottom:} \qquad u_{i,0} = 2U_b - u_{i,1}, \qquad v_{i,0} = V_b \tag{3.105c}$$

$$\text{Top:} \qquad u_{i,n_y+1} = 2U_b - u_{i,n_y}, \qquad v_{i,n_y} = V_b \tag{3.105d}$$

for $i = 0, \cdots, n_x + 1$.

(***iii***) Free-slip or symmetric boundary conditions: In this case, the fluid is allowed to flow freely in the tangent direction but vanishes normal to the boundary. Therefore,

for stationary left and right boundaries, we set $u = 0$ and $\partial v/\partial x = 0$.

$$\text{Left:} \quad u_{0,j} = 0, \quad v_{0,j} = v_{1,j}, \quad \text{for } j = 0, \cdots, n_y + 1 \tag{3.106a}$$

$$\text{right:} \quad u_{n_x,j} = 0, \quad v_{n_x+1,j} = v_{n_x,j}, \quad \text{for } j = 0, \cdots, n_y + 1 \tag{3.106b}$$

For stationary bottom and top boundaries, we require $\partial u/\partial y = 0$ and $v = 0$.

$$\text{Bottom:} \quad u_{i,0} = u_{i,1}, \quad v_{i,0} = 0, \quad \text{for } i = 0, \cdots, n_x + 1 \tag{3.107a}$$

$$\text{Top:} \quad u_{i,n_y+1} = u_{i,n_y}, \quad v_{i,n_y} = 0, \quad \text{for } i = 0, \cdots, n_x + 1 \tag{3.107b}$$

(***iv***) Outflow or open boundary conditions: A large class of flows involve physically unbounded domain, but the domain needs to be truncated artificially to a finite size in order to perform numerical simulations. Imposing boundary conditions on these boundaries is a challenging and open issue, since neither the pressure nor the velocity is known physically or mathematically. Various types of open boundary conditions and the comparison can be found in Jin & Braza (1993), Sani & Gresho (1994), Dong *et al.* (2014), Dong (2015) and Dong & Shen (2015). In fact the types and the ease of implementation of open boundary conditions are closely associated with the spatial discretization. In the present study, two types of boundary conditions are employed, namely the Neumann boundary condition ($\partial \mathbf{u}/\partial \mathbf{n} = 0$) and the convective boundary condition ($\partial \mathbf{u}/\partial t + \tilde{U}\partial \mathbf{u}/\partial \mathbf{n} = 0$), where $\tilde{U}$ can be taken from either the average value of the normal velocity at the outflow boundary or just the inflow velocity. The convective boundary condition is found to be more effective for allowing the generated vortices to leave the computational domain freely.

For the Neumann type, the normal derivatives of both velocity components are set to zero, which indicates that the total velocity does not change along the normal direction. Therefore, the velocities on the boundaries can be realized by setting the velocities at the boundary equal to the neighbouring values inside the domain.

$$\text{Left:} \quad u_{0,j} = u_{1,j}, \quad v_{0,j} = v_{1,j}, \quad \text{for } j = 0, \cdots, n_y + 1 \tag{3.108a}$$

$$\text{right:} \quad u_{n_x,j} = u_{n_x-1,j}, \quad v_{n_x+1,j} = v_{n_x,j}, \quad \text{for } j = 0, \cdots, n_y + 1 \tag{3.108b}$$

$$\text{Bottom:} \quad u_{i,0} = u_{i,1}, \quad v_{i,0} = v_{i,1}, \quad \text{for } i = 0, \cdots, n_x + 1 \tag{3.108c}$$

$$\text{Top:} \quad u_{i,n_y+1} = u_{i,n_y}, \quad v_{i,n_y} = v_{i,n_y-1}, \quad \text{for } i = 0, \cdots, n_x + 1 \tag{3.108d}$$

The convective boundary conditions can be easily constructed by using the Neumann boundary condition.

(***v***) Periodic boundary conditions: The flow quantities at these boundaries share the same values. At left and right boundaries, the velocities and the pressure are set by

$$u_{0,j} = u_{n_x-1,j}, \qquad v_{0,j} = v_{n_x-1,j}, \qquad v_{1,j} = v_{n_x,j} \tag{3.109a}$$

$$u_{n_x,j} = u_{1,j}, \qquad v_{n_x+1,j} = v_{2,j}, \qquad p_{1,j} = p_{n_x,j} \tag{3.109b}$$

for $j = 0, \cdots, n_y + 1$. At the bottom and top boundaries,

$$v_{i,0} = v_{i,n_y-1}, \qquad u_{i,0} = u_{i,n_y-1}, \qquad u_{i,1} = u_{i,n_y} \tag{3.110a}$$

$$v_{i,n_y} = v_{i,1}, \qquad u_{i,n_y+1} = u_{i,2}, \qquad p_{i,1} = p_{i,n_y} \tag{3.110b}$$

for $i = 0, \cdots, n_x + 1$. Note that in this discrete settings, the domain boundaries overlap by one cell width which contrasts with the continuous case. Therefore, the physical domain must be chosen one cell width longer (Griebel, 1998).

3.8 Solving linear system

The discretization of governing equations results in a series of linear systems to be solved. Figure 3.5 shows the typical matrix pattern after the finite difference discretization. These matrices are generally sparse and banded. For instance, for a 100×100 grid, only 0.05% of the matrix elements are non-zero. Storing and computing such matrices directly would cost huge memories and computational resources.

Therefore, it is more efficient to compress the matrix and to store only the non-zero elements. But accessing the individual elements becomes more complicated and additional informations are needed to recover the original matrix.

3.8.1 Sparse matrix storage

Many storage formats exist for the sparse matrix, such as the COO (Coordinate), CRS (Compressed Row Storage), CCS (Compressed Column Storage), DIA (Diagonal), ELL (ELLPACK) and HYB (Hybrid ELL with COO). The non-zero elements are extracted into an array and auxiliary arrays are introduced to describe the locations of the non-zero elements. In present work, the CRS and CCS formats are used.

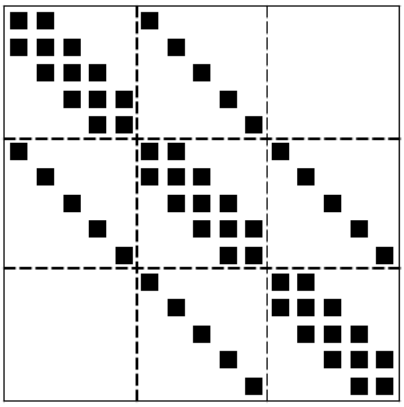


Figure 3.5: The matrix pattern resulting from finite difference discretization of a 5×3 physical cells. Squares represent the non-zero entries.

For instance, given the matrix

$$\begin{bmatrix} 10 & 11 & 0 & 12 \\ 0 & 13 & 14 & 0 \\ 0 & 0 & 15 & 0 \\ 16 & 17 & 0 & 18 \end{bmatrix} \tag{3.111}$$

The CRS format stores it with three following arrays

$$\begin{aligned} \text{row offsets} &= [0 \quad 3 \quad 5 \quad 6 \quad 9] \\ \text{values} &= [10 \quad 11 \quad 12 \quad 13 \quad 14 \quad 15 \quad 16 \quad 17 \quad 18] \\ \text{column indices} &= [0 \quad 1 \quad 2 \quad 1 \quad 2 \quad 2 \quad 0 \quad 1 \quad 3] \end{aligned}$$

and the corresponding CCS format yields

$$\begin{aligned} \text{column offsets} &= [0 \quad 2 \quad 5 \quad 7 \quad 9] \\ \text{values} &= [10 \quad 16 \quad 11 \quad 13 \quad 17 \quad 14 \quad 15 \quad 12 \quad 18] \\ \text{row indices} &= [0 \quad 3 \quad 0 \quad 1 \quad 3 \quad 1 \quad 2 \quad 0 \quad 3] \end{aligned}$$

The most attracting feature of CRS and CCS formats is the ease of storing the matrix transpose and manipulating operations between the original matrix and its transpose. They are frequently used in solving the moving force equation derived in Chapter 4.

3.8.2 Linear system solvers

To solve the linear system $Ax = b$, many methods can be employed, either directly or iteratively. Direct methods are extremely inefficient for large scale problems. In present work, we use the iterative methods. Following Barrett *et al.* (1994), we classify the iterative methods into two groups:

- Stationary methods:
  - Jacobi;
  - Gauss-Seidel;
  - SOR (Successive Over-Relaxation);
- Non-stationary methods:
  - Conjugate Gradient (CG);
  - Conjugate Gradient Squared (CGS);
  - BiConjugate Gradient (BiCG);
  - Bi-conjugate Gradient Stabilized (Bi-CGSTAB);
  - Generalized Minimal Residual (GMRES);
  - Minimum Residual (MINRES) and Symmetric LQ (SYMMLQ);
  - Conjugate Gradient on the Normal Equations: CGNE and CGNR;
  - Quasi-Minimal Residual (QMR);
  - Richardson iteration;
  - Chebyshev iteration.

To compare the efficiency of above methods, we perform several tests by solving the two dimensional PPE with Neumann boundary conditions on different grids. Table 3.1 shows the comparison of stationary and non-stationary methods. Clearly, the non-stationary methods converge much more faster compared to the stationary methods. For the stationary methods, the convergence speed follows SOR, Gauss-Seidel, Jacobi. The Gauss-Seidel method is about two times quicker than the Jacobi method. The SOR method reduces the computational time by two orders of magnitude. However, when increasing the system dimension, the computational time with the three stationary methods grows tremendously. This test is performed on the HP Z420 workstation, with the CPU Intel Xeon E5-1607 3.00 GHz with 16 GB of RAM.

The non-stationary methods, also termed as the Krylov subspace methods (Saad, 2003), are generally suitable for solving large systems. We perform the same test on the HP Z420 workstation and compare the process time of various Krylov methods in Table 3.2. Apparently, the CG method shows the best convergence speed. The BiCG and BiCGSTAB methods are twice times slower, which is reasonable according their construction but they works well with non-symmetric matrices. The algorithm for the CG method is shown in algorithm 1.

Algorithm 1: Conjugate gradient method

Initialize: $r_0 = b - Ax_0$, $p_0 = r_0$
for $k = 0$ to $k_{\max}$ **do**
  $\alpha_k = (r_k, r_k)/(Ap_k, p_k)$
  $x_{k+1} = x_k + \alpha_k p_k$
  $r_{k+1} = r_k - \alpha_k A p_k$
  $\beta_k = (r_{k+1}, r_{k+1})/(r_k, r_k)$
  $p_{k+1} = r_{k+1} + \beta_k p_k$
  if $||r_{k+1}|| <$ tolerance **then**
    break
  else
    $k = k + 1$
  end
end

	10×10			20×20		
	Iter.	Time (s)	Speed-up	Iter.	Time (s)	Speed-up
Jacobi	16243	1.492×10^{0}	1.00	78226	1.153×10^{2}	1.00
Gauss-Seidel	8411	8.233×10^{-1}	1.81	40564	6.618×10^{1}	1.74
SOR	330	3.396×10^{-2}	43.93	1229	1.914×10^{0}	60.24
CG	43	1.934×10^{-4}	7714.58	94	1.221×10^{-3}	94430.79
CGNR	73	6.443×10^{-4}	2315.69	307	6.947×10^{-3}	16597.09
CGNE	70	4.631×10^{-4}	3221.76	302	5.781×10^{-3}	19944.65

Table 3.1: Comparison of stationary and non-stationary solvers on different grids of 10×10 and 20×20. The tolerance is set to 1×10^{-10}.

	100×100			400×400		
	Iter.	Time (s)	Speed-up	Iter.	Time (s)	Speed-up
CG	489	6.506×10^{-1}	1.00	1978	4.087×10^{1}	1.00
Bi-CGSTAB	373	1.267×10^{0}	0.51	1707	9.197×10^{1}	0.44
MINRES	484	7.222×10^{-1}	0.90	2241	5.168×10^{1}	0.79
SYMMLQ	489	7.368×10^{-1}	0.88	2786	6.287×10^{1}	0.65
BiCG	489	1.297×10^{0}	0.50	1978	8.448×10^{1}	0.48

Table 3.2: Comparison of Krylov subspace solvers on different grids of 100×100 and 400×400. The tolerance is set to 1×10^{-10}.

3.8.3 Preconditioning

The convergence rate of iterative methods relies on distribution of eigenvalues of the coefficient matrix A. A simple indicator, known as the condition number, is defined by

$$\text{cond}(A) = ||A||\ ||A^{-1}|| \tag{3.112}$$

In general the pressure Poisson equation possesses a high condition number and the system is ill-conditioned. The high condition number indicates that a small error in b may cause a large error in x. Hence iterative methods are easy to diverge sometimes. To reduce the condition number, preconditioning is needed. We introduce a preconditioner M that approximates the coefficient matrix A, such that

$$M^{-1}Ax = M^{-1}b \tag{3.113}$$

has the same solution as the original system but with a more favourable condition number for $M^{-1}A$. However, adding preconditioners is a trade-off since it increases extra cost for their construction and application (see the preconditioned Conjugate Gradient method in algorithm 2 for example).

Several preconditioners can be employed, such as the Jacobi (or diagonal) preconditioner, the ILU (Incomplete LU factorization) preconditioner, the IC (Incomplete Cholesky factorization) preconditioner and the multigrid preconditioner. Table 3.3 shows the efficiency of different preconditioners. As discussed previously, adding preconditioner can increase the computational time. This can be demonstrated by using the diagonal preconditioner, as it contributes little towards the convergence rate. The ILU and IC outperforms the others in this test. The multigrid method is originally used as the solver, but it can achieve a better performance when used as a preconditioner combined with the Krylov subspace solvers. Even thought the

Algorithm 2: Preconditioned conjugate gradient method

Initialize: $r_0 = b - Ax_0$, solve $Mz_0 = r_0$, $p_0 = r_0$
for $k = 0$ to $k_{\max}$ **do**
  $\alpha_k = (r_k, z_k)/(Ap_k, p_k)$
  $x_{k+1} = x_k + \alpha_k p_k$
  $r_{k+1} = r_k - \alpha_k A p_k$
  solve $Mz_{k+1} = r_{k+1}$
  $\beta_k = (r_{k+1}, z_{k+1})/(r_k, z_k)$
  $p_{k+1} = z_{k+1} + \beta_k p_k$
  if $||r_{k+1}|| <$ tolerance **then**
    break
  else
    $k = k + 1$
  end
end

multigrid method, specifically the geometric algebraic multigrid (GAMG) method, is less efficient than the incomplete factorization methods in this steady test, it is found to be much more efficient in the unsteady projection method. Since in the projection method, once the fluid coefficient matrices are formulated, the corresponding preconditioners are constructed only in the beginning and maintained during the whole calculation. This can significantly improve the efficiency, which is shown in Table 3.5 with respect to the application time.

	100×100			400×400		
	Iter.	Time (s)	Speed-up	Iter.	Time (s)	Speed-up
None	489	6.506×10^{-1}	1.00	1978	4.087×10^{1}	1.00
Diagonal	490	7.232×10^{-1}	0.90	2415	5.211×10^{1}	0.78
ILU	145	2.728×10^{-1}	2.38	587	1.521×10^{1}	2.69
IC	145	1.771×10^{-1}	3.67	587	1.544×10^{1}	2.65
GAMG	11	4.947×10^{-1}	2.45	15	2.921×10^{1}	1.40

Table 3.3: Comparison of preconditioners along with the CG solver on different grids of 100×100 and 400×400. The tolerance is set to 1×10^{-10}. This test is also performed on the HP Z420 workstation.

3.8.4 Parallel computing

In the past years the increasing power of computers has greatly accelerated the numerical simulations. Numerous high-performance computing resources are available nowadays. To further improve this work, the programs are extended to allow parallel computing. First we integrate our methods into the well-known PETSc (Portable, Extensible Toolkit for Scientific Computation) software, which employs the MPI (Message Passing Interface) for communications between the CPU cores. Highly efficient Krylov subspace solvers and preconditioners have already been implemented.

In the second model, we parallelize the programs on the GPU (Graphics Processing Unit). In fact the CPU consists of a few cores optimized for sequential serial task, while the GPU has massive smaller cores, which can be extremely efficient for handling multiple tasks simultaneously. Therefore, we send the parallelable and computational intensive parts of the application to the GPU and run the remainders on the CPU. From the practical point of view, the hybrid CPU/GPU model runs significantly fast.

Currently two languages are available for GPU programming: Nvidia CUDA and OpenCL. In the present work, we use the Nvidia CUDA C++ language for accelerating our work. The linear system is solved by the CUDA CUSP library where the calculation and data are distributed to the GPU cores by the THRUST library.

Table 3.4 illustrates the performances of the two models in the test of solving the PPE with the CG solver on a 400×400 grid. This test is done on the calculation platform PILCAM2 with the CPU Intel Xeon X7542 2.67 GHz of 24×6 cores and the graphic card of Quadroplex 2200 S4 with 240 cores. The process time decreases by half when we double the cores of CPU from 1 to 16. Approximately 1.5-2.8 times' acceleration have been found when the multigrid method is applied as a preconditioner for the CPU parallelization. The parallelization of GPU greatly accelerates the calculation up to 40 times when the multigrid preconditioner is used. Different preconditioners, such as the diagonal preconditioner, the approximate inverse (AINV) preconditioner (Chow & Saad, 1998) and the multigrid preconditioner are compared in Table 3.5. The version of PETSc library tested here is 3.5.1 and the version of CUSP library is 0.2.0.

Parallelization	Cores	CG		CG+MG	
		Time (s)	Speed-up	Time (s)	Speed-up
CPU	1	63.00	1.00	25.25	1.00
	2	30.68	2.05	10.59	2.38
	4	15.26	4.13	4.56	5.23
	8	7.79	8.09	2.13	11.84
	16	4.20	15.00	1.10	22.92
	20	8.88	7.09	1.11	22.71
GPU	240	10.36	6.08	0.63	40.08

Table 3.4: Time consummation and speed-up of the CPU and GPU parallelization for solving the PPE on a 400×400 grid. The tolerance is set to 1×10^{-10}.

Preconditioner	Construction time (s)	Application time (s)	Speed-up
None	0	10.36	1.00
Diagonal	≈ 0	10.87	0.95
AINV	1.26	6.52	1.59
GAMG	0.51	0.12	86.33

Table 3.5: Comparison of different preconditioners in GPU parallelization with the CUSP library, where the CG solver is used for solving the PPE on a 400×400 grid. The tolerance is set to 1×10^{-10}.

3.9 Numerical validation

In order to validate the current method, various numerical simulations are performed in this section. The results are compared against the analytical solution or the well established experimental and numerical data sets. To facilitate the accuracy study, the error norms, for example $||e_u||$ for the velocity component u, on a $n_x \times n_y$ grid are computed by

$$||e_u||_2 = \left[\frac{1}{n_x n_y} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} (u_{i,j} - u_{i,j}^{\text{ref}})^2 \right]^{1/2} \tag{3.114}$$

$$||e_u||_1 = \frac{1}{n_x n_y} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} |u_{i,j} - u_{i,j}^{\text{ref}}| \tag{3.115}$$

$$||e_u||_\infty = \max \left\{ |u_{i,j} - u_{i,j}^{\text{ref}}| \right\} \tag{3.116}$$

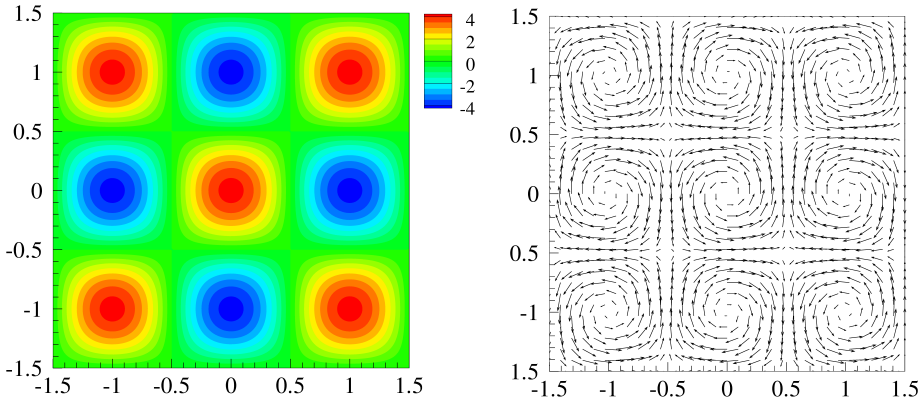
for $i = 1, \ldots, n_x, j = 1, \ldots, n_y$, where u^{ref} represents the reference value obtained either from an exact solution or a refined solution and $u_{i,j}$ is the numerical solution.

3.9.1 Taylor-Green vortices for convergence study

To validate previous error estimations for the projection methods, we consider the two-dimensional unsteady case of an array of decaying vortices with the following analytical solution (Kim *et al.*, 2001)

$$\begin{aligned} u &= -cos(\pi x)\ sin(\pi y)\ e^{-2\pi^2 t/Re} \\ v &= sin(\pi x)\ cos(\pi y)\ e^{-2\pi^2 t/Re} \\ p &= -\frac{1}{4}\left(cos(2\pi x) + sin(2\pi y)\right) e^{-4\pi^2 t/Re} \end{aligned} \tag{3.117}$$

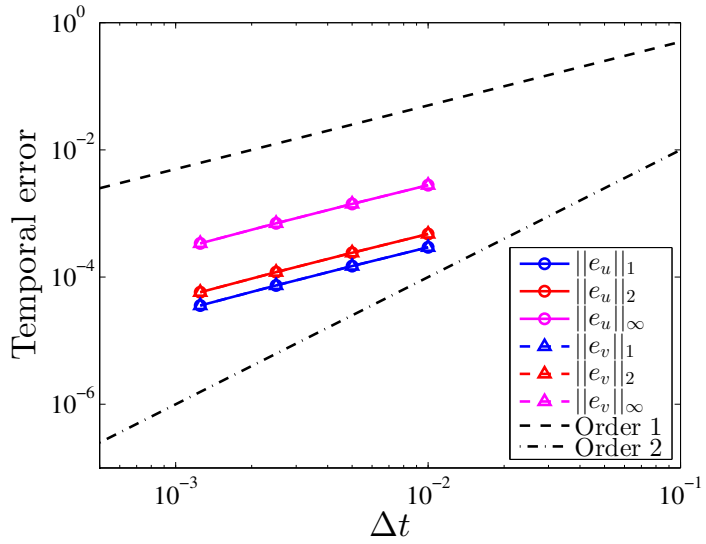
This simulation is performed on a square domain $\Omega = [-1.5, 1.5] \times [-1.5, 1.5]$. The Reynolds number Re is prescribed to 10. The initial and boundary conditions are provided by the exact solution. We advance the equations for $0 \leqslant t \leqslant 0.2$. The computed vorticity and velocity fields at $t = 0.2$ are shown in Figure 3.6.



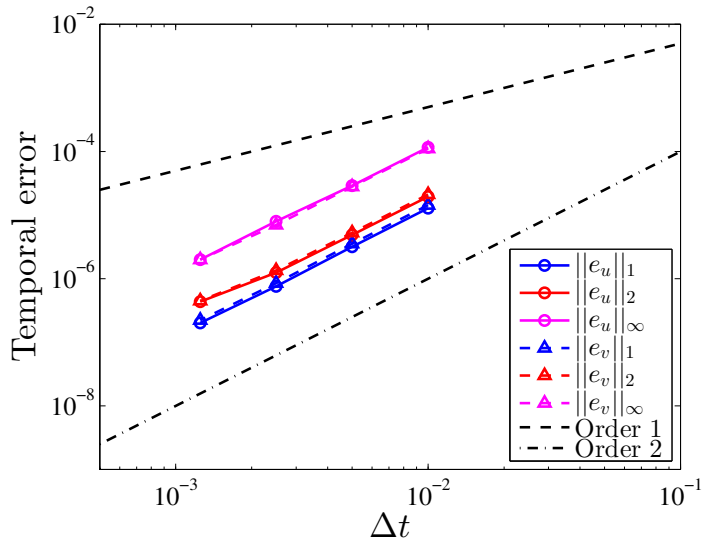
(a) Vorticity (b) Velocity vector

Figure 3.6: The computed vorticity and velocity fields of the decaying vortices problem at $t = 0.2$.

To study the splitting error on the temporal accuracy, we compare the results at $t = 0.2$ to a reference solution obtained with a very fine time step $\Delta t = 1 \times 10^{-4}$, and the spatial resolution of $\Delta x = \Delta y = 9.375 \times 10^{-3}$. The L_1, L_2, L_∞ error norms are displayed in Figure 3.7 on a log-log plot. Figure 3.7b shows a second order temporal accuracy for all the components of the fluid velocity, conforming previous error estimation analysis for the rotational incremental pressure-correction projection method. Figure 3.7a displays that the non-incremental pressure-correction method (Chorin-Témam projection method) only gives a first order accuracy, which is due to the irreducible time splitting error of order one.
We also expect the second order accuracy in space since the second order central



(a)



(b)

Figure 3.7: Temporal convergence of the decaying vortices problem: (a) results of the non-incremental pressure-correction scheme (Chorin-Témam projection method); (b) results of the rotational incremental pressure-correction scheme.

differencing scheme is used for all the derivatives in this case. We fix the time step $\Delta t = 1 \times 10^{-4}$ and vary the computational grids (20×20, 40×40, 80×80, and 160×160). The error is computed by comparing the results to the analytical solution. The spatial error is shown in Figure 3.8, showing a second order spatial accuracy for the velocity fields.

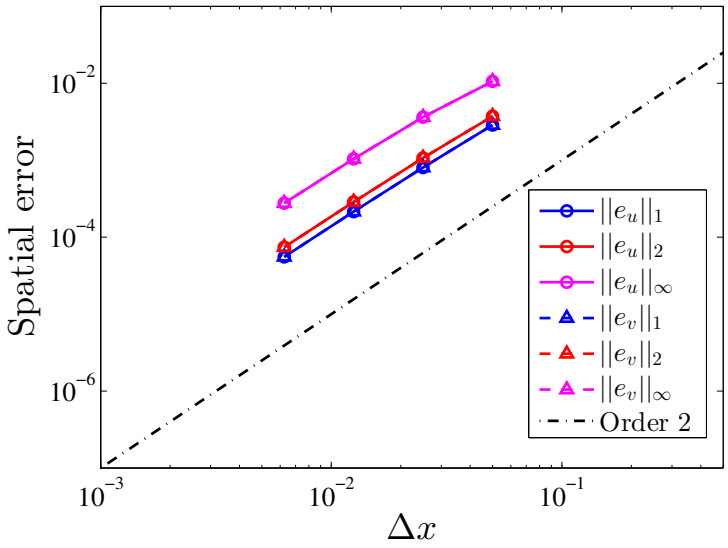


Figure 3.8: Spatial convergence of the decaying vortices problem.

3.9.2 Kovasznay flow for stability study

The steady Kovasznay flow is considered here as another test to study the stability of the projection schemes (Dong & Shen, 2010). The exact solution for the velocity and pressure fields is given by

$$\begin{aligned} u &= 1 - e^{\lambda x} cos(2\pi y) \\ v &= \frac{\lambda}{2\pi} e^{\lambda x} sin(2\pi y) \\ p &= \frac{1}{2}(1 - e^{2\lambda x}) \end{aligned} \tag{3.118}$$

where $\lambda = Re/2 - \sqrt{Re^2/4 + 4\pi^2}$ and Re is prescribed to 40. The computational domain is selected to be $[-0.5, 1.0] \times [-0.5, 0.5]$ and the boundary conditions are provided by the exact solution. The computed final field is shown in Figure 3.9.

We compare the stability of the explicit scheme and the semi-implicit scheme in Figure 3.10 and Figure 3.11. In the explicit scheme, both the convection and diffusion terms are treated explicitly. The scheme is known to be free of splitting error but with a severe limitation on the time step. Figure 3.10 shows that on the coarse mesh the time step is at most 0.003, and a higher value of 0.0032 results in divergence after a few time steps. In this case the diffusive constraint dominates the CFL constraint, therefore the time step is limited by the diffusive constraint. Refining the mesh by half requires a four times smaller time step for convergence, which confirms the diffusive stability constraint.

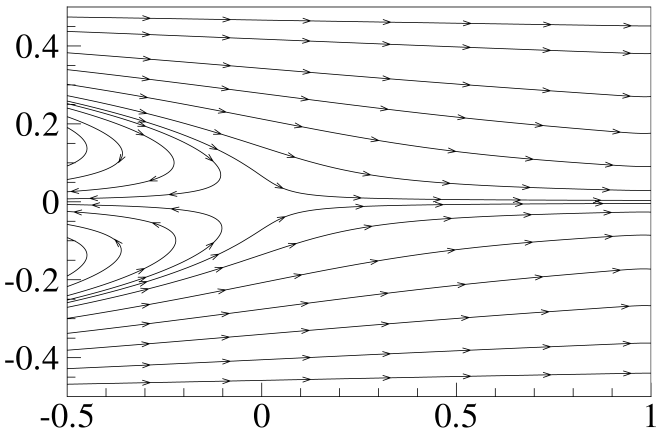
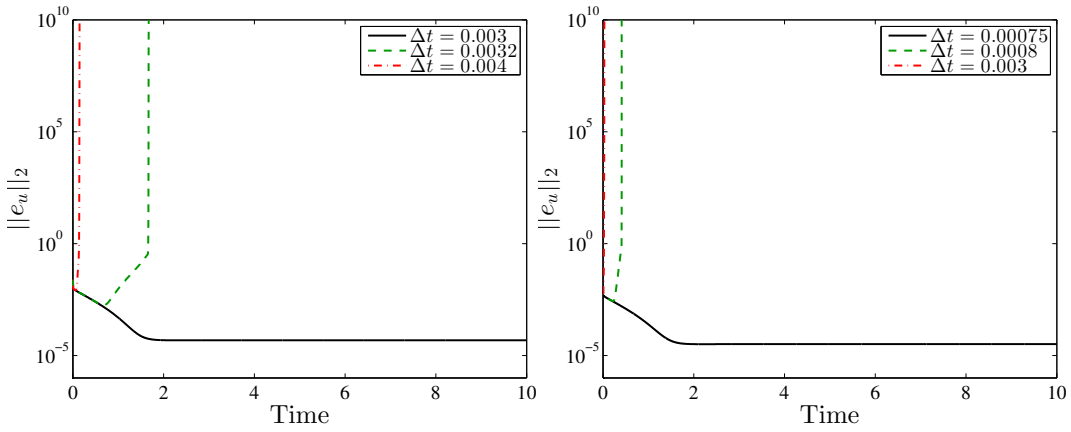


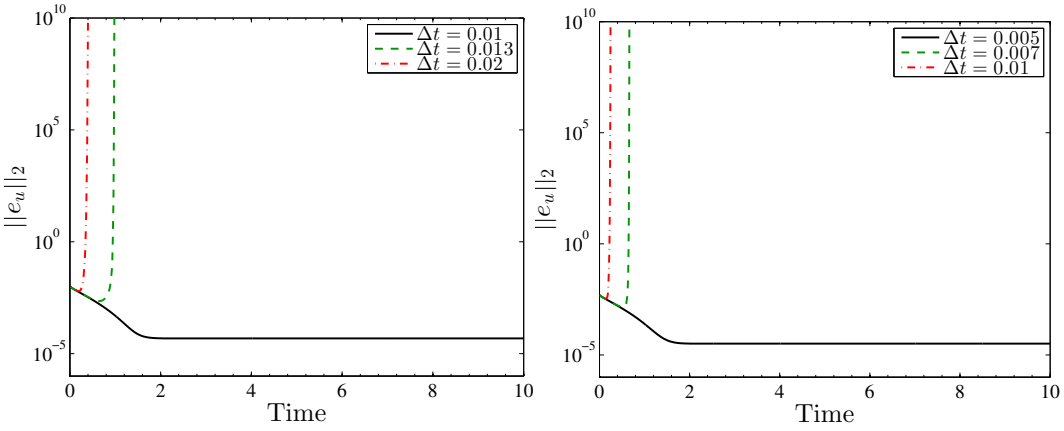
Figure 3.9: Computed streamlines of the Kovasznay flow problem.

The semi-implicit scheme circumvents this severe diffusive constraint by treating the diffusion term implicitly. The convection term is still treated explicitly for simplicity, leaving a usual CFL constraint. Figure 3.11 shows that a much larger time step can be used on the coarse mesh with the semi-implicit scheme compared to the explicit scheme. Refining the mesh by half only requires a two times smaller time step for convergence, which is controlled by the CFL constraint. Therefore, in the present thesis the semi-implicit scheme is used.



(a) $h = 1/40$ (b) $h = 1/80$

Figure 3.10: Time histories of the x-component velocity L_2 norm with different time step sizes using the explicit scheme: (a) the coarse mesh; (b) the fine mesh.



(a) $h = 1/40$ (b) $h = 1/80$

Figure 3.11: Time histories of the x-component velocity L_2 norm with different time step sizes using the semi-implicit scheme: (a) the coarse mesh; (b) the fine mesh.

3.9.3 Lid-driven cavity flow

In this case, the lid-driven cavity flow is simulated as a standard test. The physical configuration and boundary conditions are shown in Figure 3.12. The flow is driven by the top wall moving at a constant velocity. Inside the cavity several standing vortices exist, whose position and size mainly depend on the Reynolds number. At $Re = 1$ the flow is symmetric with two vortices at the corners. Increasing the Reynolds number the main vortex center moves toward the downstream corner. At Reynolds number above 1000, a third corner vortex is formed at the upper left corner (see Figure 3.13). A uniform grid of 128×128 is employed in this test, as used in Ghia *et al.* (1982).

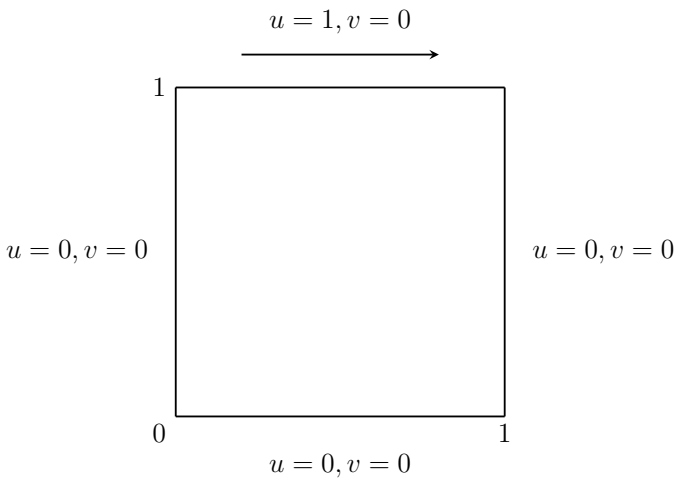
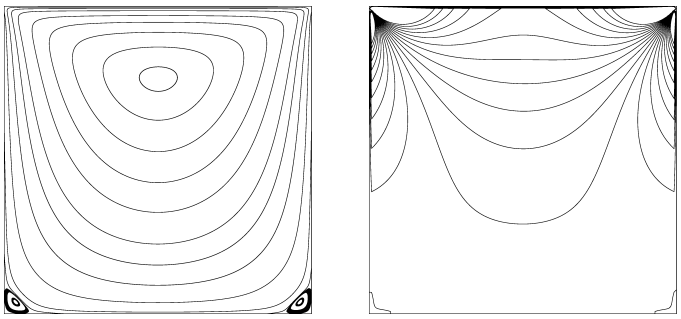
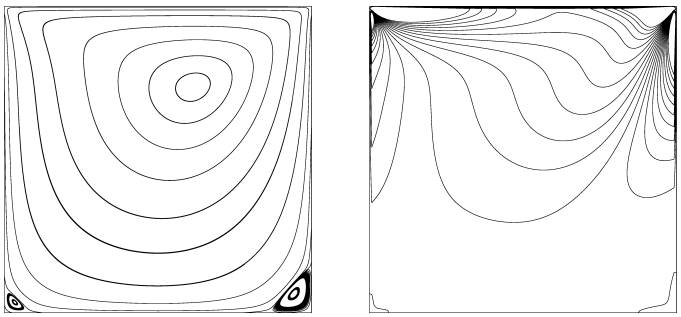


Figure 3.12: Problem configuration of the lid-driven cavity flow.

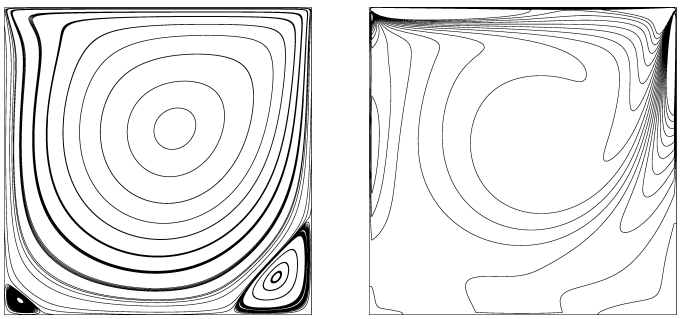
To assess the accuracy of present results, the velocity components along the horizontal and vertical centerlines are plotted in Figure 3.14 and compared to the well established work of Ghia *et al.* (1982). Table 3.6 displays the stream function, vorticity values and the coordinates at the center of primary vortex. Those values agree well with results of Ghia *et al.* (1982), Kim & Moin (1985) and Bruneau *et al.* (2006).



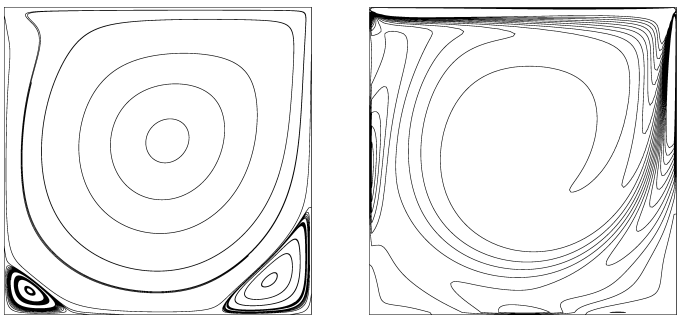
(a) $Re = 1$



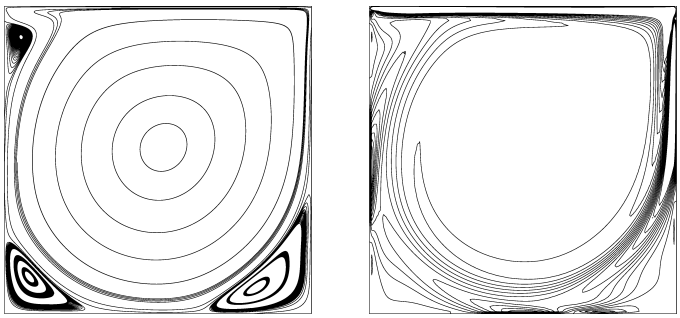
(b) $Re = 100$



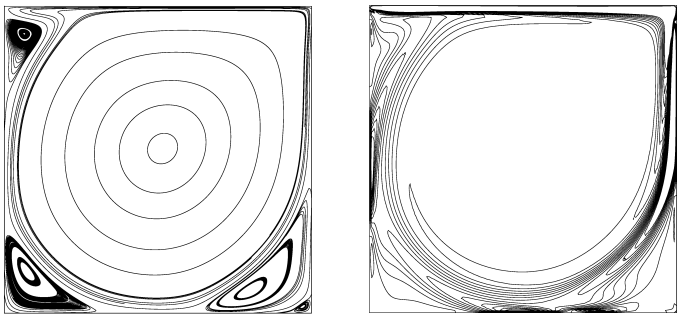
(c) $Re = 400$



(d) $Re = 1000$

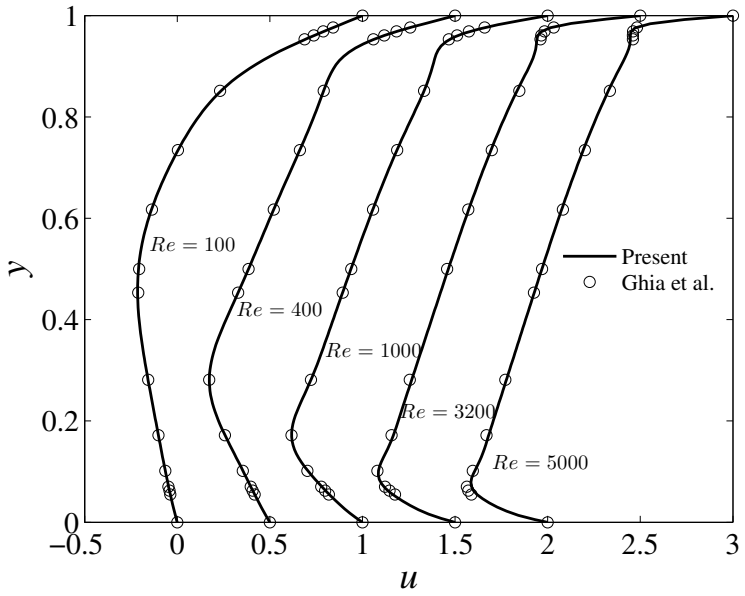


(e) $Re = 3200$

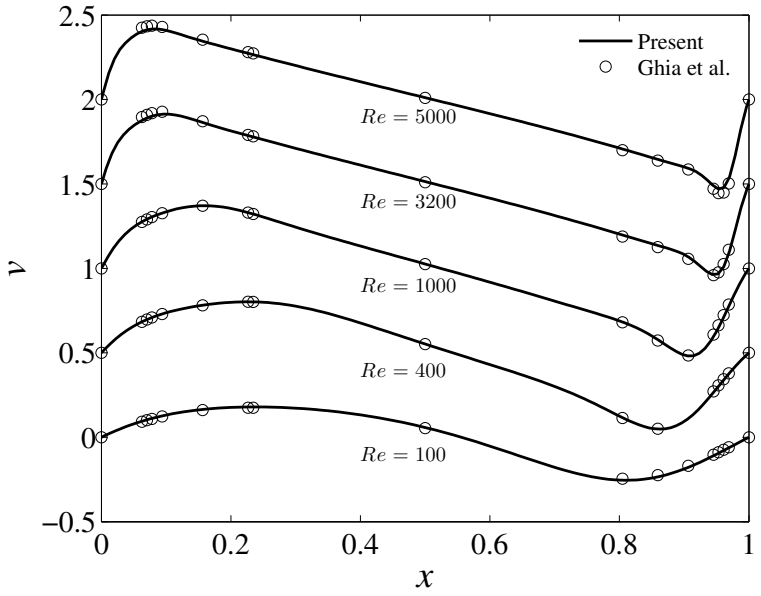


(f) $Re = 5000$

Figure 3.13: Computed streamlines (left) and vorticity contours (right) at different Reynolds number ranging from 1 to 5000. The contour levels are set from -10 to 10 with an increment of 1.



(a)



(b)

Figure 3.14: Comparison of velocity profiles at different Reynolds number ranging from 100 to 5000: (a) u along the line $x = 0.5$ and (b) v along the line $y = 0.5$. The curves are shifted for visualisation.

		Re				
		100	400	1000	3200	5000
ψ_c	**Present**	-0.103	-0.113	-0.118	-0.117	-0.115
	Ghia *et al.* (1982)	-0.103	-0.114	-0.118	-0.115	-0.119
	Kim & Moin (1985)	-0.103	-0.112	-0.116	-0.119	-0.112
	Bruneau *et al.* (2006)	-	-	-0.118	-	-0.117
ω_c	**Present**	-3.165	-2.288	-2.046	-1.890	-1.829
	Ghia *et al.* (1982)	-3.166	-2.295	-2.050	-1.989	-1.860
	Kim & Moin (1985)	-3.177	-2.260	-2.026	-1.901	-1.812
	Bruneau *et al.* (2006)	-	-	-2.051	-	-1.860
x_c	**Present**	0.616	0.555	0.531	0.518	0.515
	Ghia *et al.* (1982)	0.617	0.555	0.531	0.517	0.512
	Bruneau *et al.* (2006)	-	-	0.469	-	0.484
y_c	**Present**	0.737	0.606	0.566	0.541	0.536
	Ghia *et al.* (1982)	0.734	0.606	0.563	0.547	0.535
	Bruneau *et al.* (2006)	-	-	0.563	-	0.539

Table 3.6: Stream function, vorticity values and coordinates of primary vortex center at different Reynolds numbers.

3.9.4 Backward-facing step flow

In the final example, the flow over a backward-facing step at different Reynolds number is simulated. The flow domain is taken as $[0, 30h] \times [0, 2h]$ where h is the step height, as shown in Figure 3.15. A parabolic velocity profile is prescribed at the inflow boundary on top of the step. Outflow boundary condition is assigned at the downstream boundary. No-slip wall condition is applied to the other boundaries. A 101×101 grid is employed in the present test as used in Kim & Moin (1985).

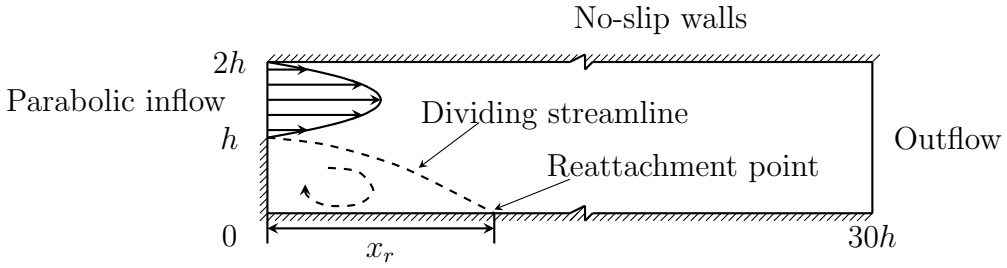
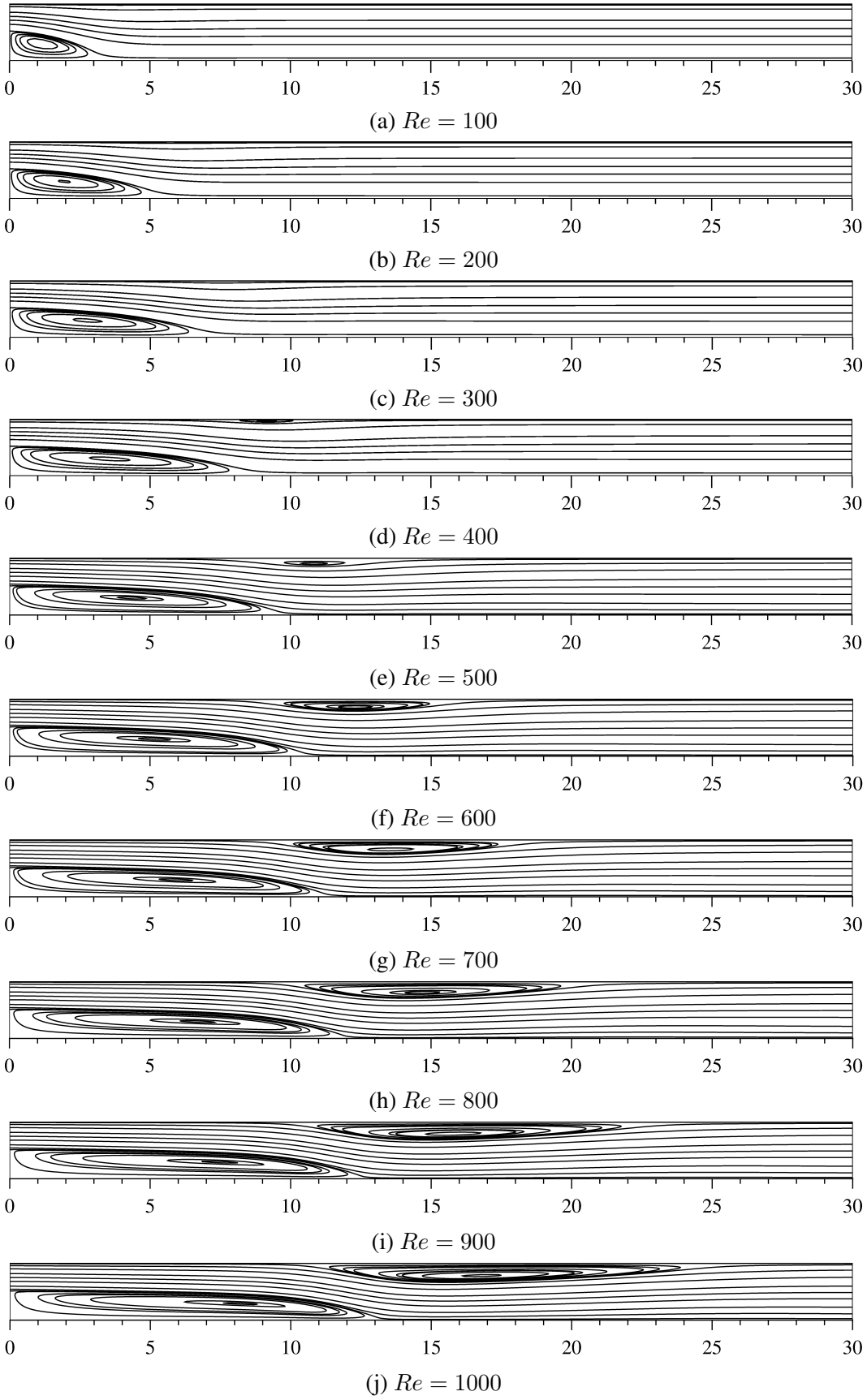


Figure 3.15: Sketch of flow over a backward facing step.

Figure 3.16 shows the streamlines at different Reynolds number ranging from 100 to 1000. At $Re = 100$ a small eddy is formed behind the step. The eddy size increases and the main flow is drawn downward when increasing the Reynolds number. From about $Re = 400$ a second eddy is generated at the upper boundary, whose size grows considerably from $Re = 400$ to 1000. The dimensionless reattachment length x_r/h is plotted as a function of the Reynolds number in Figure 3.17. To reveal the accuracy of present method, the results are compared to the well documented benchmark of Armaly *et al.* (1983), the data from Kim & Moin (1985) with the projection method using an accurate boundary condition for the intermediate velocity, and the computation with the streamfunction-vorticity method Erturk (2008). The reattachment length agrees well with the experimental data from Armaly *et al.* (1983) up to around $Re = 500$. Increasing the Reynolds number, the results start to deviate from the experimental data. Refining the mesh shows no differences. This is due to the three-dimensional effect at this Reynolds number, as pointed out by Armaly *et al.* (1983). The same trend has been found to the results of Kim & Moin (1985) and Erturk (2008), indicating that this discrepancy is not caused by numerical errors. The overlapped curves in Figure 3.17 demonstrates a very good agreement of the present method with other methods of Kim & Moin (1985) and Erturk (2008) towards the accuracy.



(a) $Re = 100$

(b) $Re = 200$

(c) $Re = 300$

(d) $Re = 400$

(e) $Re = 500$

(f) $Re = 600$

(g) $Re = 700$

(h) $Re = 800$

(i) $Re = 900$

(j) $Re = 1000$

Figure 3.16: Stream function contours at various Reynolds numbers from 100 to 1000 for the problem of flow over a backward facing step.

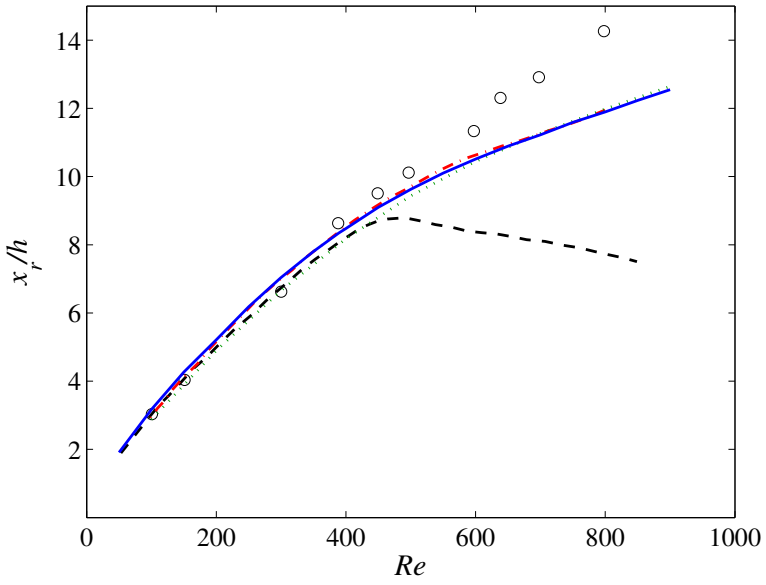


Figure 3.17: Comparison of reattachment length as a function of Reynolds number. "○", experimental data from Armaly *et al.* (1983); "- - - -", computation of Armaly *et al.* (1983); "— · —·", numerical data from Kim & Moin (1985); "· · · ·", computation of Erturk (2008); "——", present results.

3.10 Concluding remarks

In this chapter, we have presented various projection methods as the incompressible fluid solver. At each time step, the velocity and pressure fields are solved sequentially without iterations between the two unknowns. The classical projection method of Chorin-Témam decreases the original second order temporal accuracy to be only first order (see Figure 3.7a). To prevent the accuracy loss after the time splitting, we have employed the rotational incremental pressure-correction projection method as our fluid solver. The numerical tests have shown the second accuracy for all the velocity components (see Figure 3.7b). The implementation of this projection method is very simple and no numerical boundary layers will be introduced during the calculation.

We have discretized the fluid solver on a staggered mesh to avoid checker-board effects. Second order spatial accuracy is also obtained with current fluid solver (see Figure 3.8). Various boundary conditions have been implemented and highly efficient Krylov subspace solvers have been tested for solving the resulting systems. Two parallel algorithms are integrated into the fluid solver to enable high performance computing, by using the PETSc library for the parallelization of CPUs

and the CUSP library for the parallelization of GPUs. The GPU model has shown a promising performance in terms of the speed-up.
The present fluid solver has been validated with a couple of canonic tests, such as the lid-driven cavity flow and the backward-facing step flow. This solver will be served as a basic framework for the immersed boundary method in the next chapter.

Moving immersed boundary method (MIBM)

4.1 Introduction

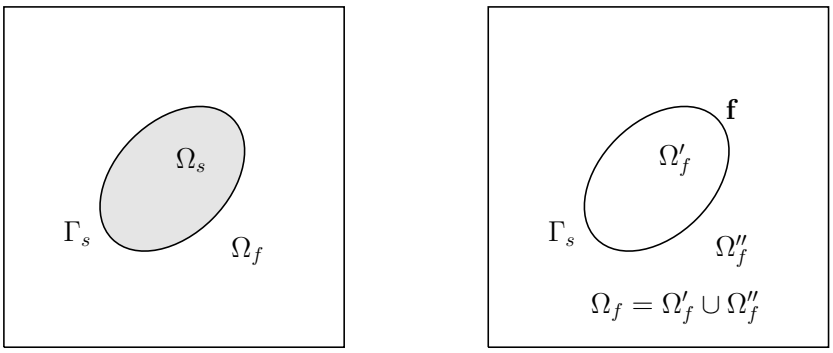
In this chapter we propose a new immersed boundary method, the moving immersed boundary method (MIBM), for the flow simulation with prescribed solid motions. The main goal of the novel method is to enforce exactly the boundary condition at the immersed interface.

We first systematically investigate the existing immersed boundary methods (IBM) in the literature, in order to highlight the improvements and the advantages of our method. Section 4.4 gives the details about the derivation of the MIBM and the comparison of performance with other immersed boundary methods. We then discuss different interpolation schemes and functions for the transfer of flow quantities between the fluid and the solid meshes in Section 4.5. The proposed method is validated in Section 4.7 with kinds of numerical examples.

4.2 Mathematical formulation and discretization

The fundamental idea of the IBM is to replace the solid domain with the surrounding fluid, as shown in Figure 4.1. The influences of the immersed objects on the fluid are reproduced by a boundary force. This force is added into the fluid momentum equation as a source term to take effects.

Therefore, the fluid mesh does not need to conform to the solid geometry. The two physical domains can use two independent meshes with their own favorite discretization, namely the Eulerian discretization for the fluid and the Lagrangian discretization for the solid. This greatly accelerates the solution procedure and eliminates issues associated with re-meshing or deforming-mesh, such as the mesh distortions and the mesh interpolation errors, which are the bottlenecks in the body-conforming mesh method (e.g. ALE).



(a) Original domain (b) Immersed domain

Figure 4.1: Illustration of the immersed boundary method.

The mathematical formulation of the immersed boundary method is written as

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2 \mathbf{u} + \mathbf{f} \tag{4.1a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{4.1b}$$

$$\int_{\Omega_f} \mathbf{u}(\mathbf{x}, t)\delta(\mathbf{x} - \mathbf{X}(s, t))d\mathbf{x} = \mathbf{U}_b \tag{4.1c}$$

$$\mathbf{f}(\mathbf{x}, t) = \int_{\Gamma_s} \mathbf{F}(s, t)\delta(\mathbf{x} - \mathbf{X}(s, t))ds \tag{4.1d}$$

in the domain Ω_f, where

- $\mathbf{U}_b$ is the solid boundary velocity;
- δ designates the Dirac delta function;
- $\mathbf{X}(s, t)$ represents the solid boundary position;
- $\mathbf{F}(s, t)$ is the boundary force defined on the Lagrangian position $\mathbf{X}(s, t)$;
- $\mathbf{f}(\mathbf{x}, t)$ is the corresponding force on the Eulerian frame.

It is easy to see that the boundary force acts as a Lagrange multiplier to satisfy the no-slip wall condition (4.1c) at the immersed interface. This boundary force also has a physical meaning that it represents the force exerted on the fluid by the solid. It is an unknown that needs to be solved along with the velocity field. How it is evaluated differs one immersed boundary method from another and determines the accuracy of the overall scheme. Recent reviews towards the IBM can be found in Iaccarino & Verzicco (2003), Mittal & Iaccarino (2005) and Sotiropoulos & Yang (2014).

By employing previous time discretization schemes for the NSE (3.17) and using the implicit scheme for the boundary force, the immersed boundary method can be written as

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t}+\left[\frac{3}{2}\mathcal{N}(\mathbf{u}^{n})-\frac{1}{2}\mathcal{N}(\mathbf{u}^{n-1})\right]=-\mathcal{G}p^{n+1}+\frac{1}{2Re}\mathcal{L}(\mathbf{u}^{n+1}+\mathbf{u}^{n})+\mathcal{S}(\mathbf{F}^{n+1}) \tag{4.2a}$$

$$\mathcal{D}\mathbf{u}^{n+1}=0 \tag{4.2b}$$

$$\mathcal{T}(\mathbf{u}^{n+1})=\mathbf{U}_{b}^{n+1} \tag{4.2c}$$

where $\mathcal{T}$ and $\mathcal{S}$ are the interpolation and spreading operators respectively

$$\mathcal{S}_{\mathbf{X}^{n+1}}(\mathbf{F}^{n+1})=\int_{\Gamma_s}\mathbf{F}^{n+1}\delta(\mathbf{x}-\mathbf{X}^{n+1})ds \tag{4.3}$$

$$\mathcal{T}_{\mathbf{X}^{n+1}}(\mathbf{u}^{n+1})=\int_{\Omega_f}\mathbf{u}^{n+1}\delta(\mathbf{x}-\mathbf{X}^{n+1})d\mathbf{x} \tag{4.4}$$

The above equations are solved on a very simple Cartesian mesh, as shown in Figure 4.2. The immersed boundary is represented with a set of Lagrangian points, where we define the boundary force for the presence of the embedded solid. As the fluid variables are defined separately in space in the staggered mesh, the boundary force points can never coincide with the underlying fluid mesh.

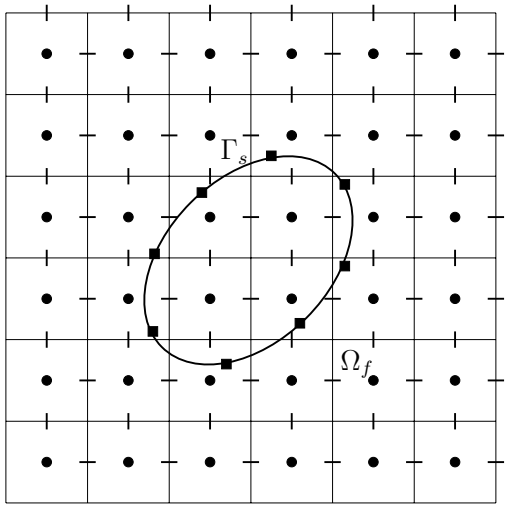


Figure 4.2: Schematic view of the immersed boundary method in a two-dimensional computational domain Ω_f. The immersed boundary Γ_s is represented by the Lagrangian marker "■", where the boundary force $\mathbf{F}(s,t)$ is defined.

4.3 Evolution of immersed boundary methods

4.3.1 Continuous forcing methods

(*i*) Deformable boundary

The immersed boundary method was first introduced by Peskin (1972a,b) for simulating blood flows through a beating heart. In the original method, the immersed elastic membrane is represented by a series of massless Lagrangian markers where the boundary force is evaluated by using the constitutive law

$$\mathbf{F}(\mathbf{X},t) = \frac{\partial(T\boldsymbol{\tau})}{\partial s} \tag{4.5}$$

where T is the fibre tension and $\boldsymbol{\tau}$ is the unit tangent to the fibre. This boundary force is then distributed to the surrounding Eulerian fluid cells with a discrete delta function δ_h

$$\mathbf{f}(\mathbf{x},t) = \int_s \mathbf{F}(\mathbf{X},t)\delta_h(\mathbf{x}-\mathbf{X}(s,t))ds \tag{4.6}$$

In the original method, the force is incorporated into the Navier-Stokes solver before discretization. Thus this method is classified into the continuous forcing category in the review paper of Mittal & Iaccarino (2005).

Despite its efficiency, the method suffers several drawbacks. First the method smears out the sharp interface and decreases the spatial accuracy to be first-order for non-smooth solutions (Peskin, 2002); Secondly the mass conservation is lost at a rate proportional to the pressure difference across the interface (Peskin, 1993, 2002); The explicit time-stepping scheme in the immersed boundary forcing part results in a severe restriction on the time step (Peskin, 2002); At high Reynolds number this method is less desirable compared to the body-conforming mesh method because of the diffused interface (Peskin, 2002; Mittal & Iaccarino, 2005).

To recover the second order spatial accuracy and maintain a sharp interface, LeVeque & Li (1994) proposed the immersed interface method (IIM) for elliptic equations. It is further extended to the Navier-Stokes equations by imposing proper jump conditions on the pressure to eliminate the discrete dipole of the forcing term (Li & Lai, 2001; Lee & LeVeque, 2003). By employing the finite element technique, the original method has been extended to the immersed finite element method (Wang & Liu, 2004; Zhang *et al.*, 2004; Wang *et al.*, 2009) and the immersed continuum method (Wang, 2007; Wang *et al.*, 2009) for more general solid problems, by replacing the membrane with solid structures that occupy finite volumes.

However, the previous methods are only well suited for elastic boundaries. When applied to rigid bodies, the stiffness is chosen to be very large. Too large value can cause instabilities during the calculation. Therefore, the compromise must be taken between extremely small time steps for negligible deformations or larger time steps with mild deformations.

(*ii*) Rigid boundary

Considering that the constitutive laws are generally not well posed in the rigid limit, Beyer & LeVeque (1992) and Lai & Peskin (2000) used a spring to attach the solid to an equilibrium location $\mathbf{X}^e(s,t)$ with a restoring force given by

$$\mathbf{F}(\mathbf{X},t) = \kappa(\mathbf{X}^e(s,t) - \mathbf{X}(s,t)) \tag{4.7}$$

where κ is the stiffness coefficient. However, to impose the boundary condition accurately, the value of κ needs to be very large as well. This can lead to stiff equations and numerical difficulties. If a smaller value of κ is used, the solid can excessively deviate from its equilibrium location.

Alternatively, Goldstein *et al.* (1993) and Saiki & Biringen (1996) employed an feedback forcing strategy by using

$$\mathbf{F}(\mathbf{X},t) = -\alpha \int_0^t (\mathbf{U}(\mathbf{X},\tau) - \mathbf{U}_b(\mathbf{X},\tau))\, d\tau - \beta\,(\mathbf{U}(\mathbf{X},t) - \mathbf{U}_b(\mathbf{X},t)) \tag{4.8}$$

where $\alpha \gg 1$ and $\beta \gg 1$ are some large artificial constants whose dimensions are $1/T^2$ and $1/T$. This approach behaves as a system of springs and dampers to correct $\mathbf{U}(\mathbf{X},t)$ to $\mathbf{U}_b(\mathbf{X},t)$ in a feedback manner. The major shortcomings of the feedback forcing approach are that big values of α and β can cause stiff equations and the explicit forcing scheme results in a severe stability constraint with the $\mathrm{CFL} = \mathrm{O}(10^{-3} - 10^{-2})$ (Goldstein *et al.*, 1993; Fadlun *et al.*, 2000).

It is worth noting that other variants in this class have been proposed in the literature for tackling rigid boundaries. For example, in Angot *et al.* (1999) and Khadra *et al.* (2000) the flow is assumed in a porous medium and the Navier-Stokes/Brinkman equations are solved with the extra force term of $\mu/K\mathbf{u}$, with K being the permeability defined as infinity for the fluid and zero for the solid. In practice, a large value of K is chosen for the fluid and smoothed over the interface, which however can lead to inaccuracies. Contrarily, Glowinski *et al.* (1994) constructed the fictitious domain method or the distributed Lagrange multiplier method (FD/DLM), which considers the solid as a fluid subject to a rigidity constraint.

To sum up, the continuous forcing immersed boundary method is attractive for elastic boundaries, since it has a physical basis and is easy to be implemented. To apply to rigid bodies, several remedies have been proposed. However, most of them rely on artificial constants, which are *ad hoc* and can cause numerical instabilities.

4.3.2 Discrete forcing methods

(*i*) Direct forcing method

To overcome the drawbacks of previous continuous forcing immersed boundary methods in simulating rigid boundaries, Mohd-Yosuf (1997) and Fadlun *et al.* (2000) proposed the direct forcing immersed boundary method via modifying the discrete momentum equation. This method is classified into the discrete forcing group in Mittal & Iaccarino (2005). For simple discussion, the discrete fluid momentum equation is rewritten as

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t}=\mathrm{RHS}^{n+1}+\mathbf{f}^{n+1} \tag{4.9}$$

where RHS contains the viscous, convective and pressure gradient terms. To yield a correct boundary condition at the interface Γ_s, the force term is then determined as

$$\mathbf{f}^{n+1}|_{\Gamma_s}=\left(-\mathrm{RHS}^{n+1}+\frac{\mathbf{U}_b^{n+1}-\mathbf{u}^{n}}{\Delta t}\right)|_{\Gamma_s} \tag{4.10}$$

which is zero elsewhere. By doing so, the desired velocity value is enforced directly without any dynamical process. No more free constants are needed in this setting. Most importantly, no additional constraints are introduced to the time step.
Unlike Peskin (1972b, 2002), the direct forcing method of Mohd-Yosuf (1997) and Fadlun *et al.* (2000) employs a local velocity reconstruction to enforce the boundary condition. Fadlun *et al.* (2000) compared several interpolation schemes and achieved a second order accuracy using a linear interpolation. In Figure 4.3, the velocity u_i at the first grid point outside the solid is obtained by linearly interpolating the velocity V at the boundary and the velocity u_{i+1} at the second grid point.

This direct forcing immersed boundary method was made popular by the work of Uhlmann (2005) for moving rigid boundaries. Uhlmann (2005) observed strong oscillations of the boundary force with the local velocity reconstruction approach and attributed this problem to insufficient smoothing. To obtain smooth results, Uhlmann (2005) suggested to evaluate the force on the Lagrangian locations and

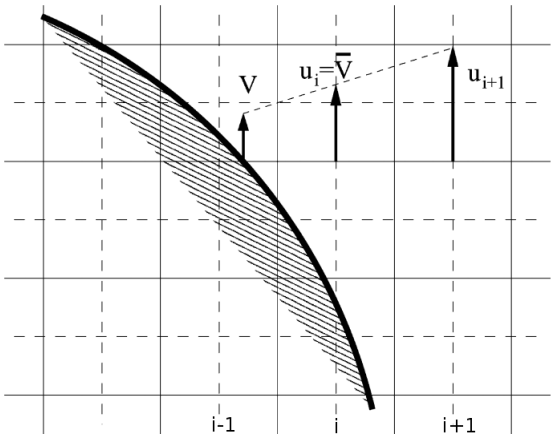


Figure 4.3: Local velocity interpolation scheme of Fadlun *et al.* (2000).

distribute the force to the fluid cells using the discrete delta function (see Figure 4.4), which inherits from the method of Peskin (1972b, 2002). Using the rotational incremental pressure-correction projection method as the fluid solver, the direct forcing immersed boundary method of Uhlmann (2005) can be summarized as

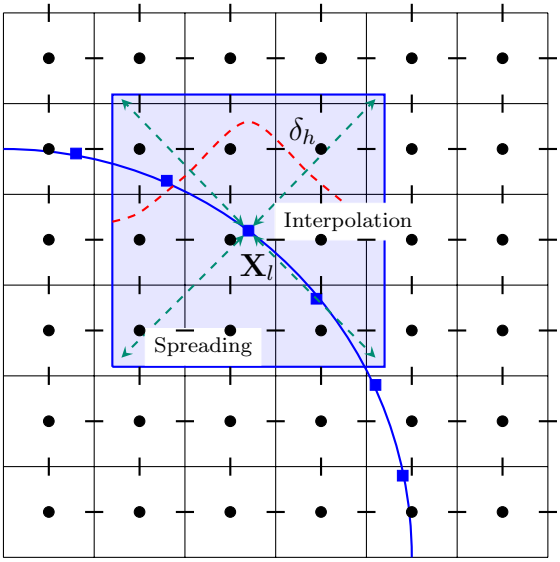


Figure 4.4: Illustration of interpolation and spreading procedures of Uhlmann (2005) and Kempe & Fröhlich (2012a) with a discrete delta function. "■" represents the solid node.

(1) Velocity prediction of all the explicit terms in the discretized momentum equation

$$\mathbf{u}^* = \mathbf{u}^n + \Delta t \left\{ -\left[\frac{3}{2}\mathcal{N}(\mathbf{u}^n) - \frac{1}{2}\mathcal{N}(\mathbf{u}^{n-1})\right] - \mathcal{G}p^n + \frac{1}{2Re}\mathcal{L}\mathbf{u}^n \right\} \tag{4.11a}$$

(2) Interpolate the predicted fluid velocity into the immersed interface with a dis-

crete delta function

$$\mathbf{U}^*(\mathbf{X}_l) = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} \mathbf{u}^* \delta_h(\mathbf{x}_{i,j} - \mathbf{X}_l) h^2 \tag{4.11b}$$

(3) Evaluate the boundary force on the Lagrangian locations

$$\mathbf{F}^{n+1}(\mathbf{X}_l) = \frac{\mathbf{U}_b^{n+1}(\mathbf{X}_l) - \mathbf{U}^*(\mathbf{X}_l)}{\Delta t} \tag{4.11c}$$

(4) Spread the boundary force to the surrounding fluid cells using the same discrete delta function

$$\mathbf{f}^{n+1}(\mathbf{x}_{i,j}) = \sum_{l=1}^{n_b} \mathbf{F}^{n+1}(\mathbf{X}_l) \delta_h(\mathbf{x}_{i,j} - \mathbf{X}_l) \Delta V_l \tag{4.11d}$$

where $\Delta V_l \approx h^2$ is surface associated with the element $\mathbf{X}_l$ by setting the arc-length δs approximate the uniform mesh width h, as shown in Figure 4.5.

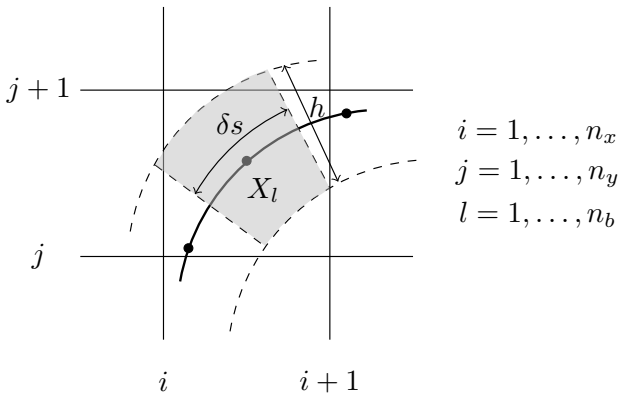


Figure 4.5: The lth element $\mathbf{X}_l$ and its associated surface $\Delta V_l \approx h^2$ marked by a shaded zone.

(5) Correct the fluid velocity with the boundary force to account for the immersed objects

$$\tilde{\mathbf{u}}^{n+1} = \mathbf{u}^* + \Delta t \mathbf{f}^{n+1} \tag{4.11e}$$

(6) Implicit treatment of the viscous term

$$\frac{1}{\Delta t}\hat{\mathbf{u}}^{n+1} - \frac{1}{2Re}\mathcal{L}\hat{\mathbf{u}}^{n+1} = \frac{1}{\Delta t}\tilde{\mathbf{u}}^{n+1} \tag{4.11f}$$

(7) Project the fluid velocity into the divergence-free field and update the pressure

$$\mathcal{L}\phi^{n+1} = \frac{1}{\Delta t}\mathcal{D}\hat{\mathbf{u}}^{n+1} \tag{4.11g}$$

$$\mathbf{u}^{n+1} = \hat{\mathbf{u}}^{n+1} - \Delta t\mathcal{G}\phi^{n+1} \tag{4.11h}$$

$$p^{n+1} = p^n + \phi^{n+1} - \frac{1}{2Re}\mathcal{D}\hat{\mathbf{u}}^{n+1} \tag{4.11i}$$

Here $\mathbf{u}^*$, $\tilde{\mathbf{u}}^{n+1}$, $\hat{\mathbf{u}}^{n+1}$, $\mathbf{u}^{n+1}$ represent the fluid velocities at different stages in the fractional step method, i.e.,

- $\mathbf{u}^*$ at the prediction step with explicit terms;
- $\tilde{\mathbf{u}}^{n+1}$ at the immersed boundary forcing step;
- $\hat{\mathbf{u}}^{n+1}$ at the viscous prediction step;
- $\mathbf{u}^{n+1}$ at the projection step.

This explicit scheme is favored in the literature as it is computationally inexpensive. However, the correct boundary condition can never be achieved on the immersed interface neither for the final velocity $\mathbf{u}^{n+1}$ nor for the intermediate velocity $\hat{\mathbf{u}}^{n+1}$ fields. To enforce the boundary condition exactly, iteration of the whole system is needed to have $\mathbf{U}^{n+1}$ instead of $\mathbf{U}^*$ in the right hand side of (4.11c). This certainly loses the advantage of the fractional step method. But it implies one way of reducing the error by choosing the closest value to the final velocity in the immersed boundary forcing step.

Kempe & Fröhlich (2012a) suggested to perform the viscous prediction step first and then use the intermediate velocity $\hat{\mathbf{u}}^{n+1}$ to compute the boundary force. To further improve the accuracy, a forcing loop is added in the immersed boundary forcing step. This additional loop is performed with 3 iterations without convergence. Even though the error is considerably reduced, the method of Kempe & Fröhlich (2012a) is still explicit. The exact no-slip boundary condition can never be satisfied. The method of Kempe & Fröhlich (2012a) is summarized as

(1) Prediction of the explicit terms

$$\mathbf{u}^* = \mathbf{u}^n + \Delta t\left\{-\left[\frac{3}{2}\mathcal{N}(\mathbf{u}^n) - \frac{1}{2}\mathcal{N}(\mathbf{u}^{n-1})\right] - \mathcal{G}p^n + \frac{1}{2Re}\mathcal{L}\mathbf{u}^n\right\} \tag{4.12a}$$

(2) Implicit viscous step

$$\frac{1}{\Delta t}\hat{\mathbf{u}}^{n+1} - \frac{1}{2Re}\mathcal{L}\hat{\mathbf{u}}^{n+1} = \frac{1}{\Delta t}\mathbf{u}^* \tag{4.12b}$$

(3) Immersed boundary forcing loop

Loop for k = 1 to 3 with $\hat{\mathbf{u}}^{(0)} = \hat{\mathbf{u}}^{n+1}$

$$\hat{\mathbf{U}}^{(k)}(\mathbf{X}_l) = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} \hat{\mathbf{u}}^{(k-1)}\delta_h(\mathbf{x}_{i,j} - \mathbf{X}_l)h^2 \tag{4.12c}$$

$$\mathbf{F}^{(k)}(\mathbf{X}_l) = \frac{\mathbf{U}_b^{n+1}(\mathbf{X}_l) - \hat{\mathbf{U}}^{(k)}(\mathbf{X}_l)}{\Delta t} \tag{4.12d}$$

$$\mathbf{f}^{(k)}(\mathbf{x}_{i,j}) = \sum_{l=1}^{n_b} \mathbf{F}^{(k)}(\mathbf{X}_l)\delta_h(\mathbf{x}_{i,j} - \mathbf{X}_l)\Delta V_l \tag{4.12e}$$

$$\tilde{\mathbf{u}}^{(k)} = \hat{\mathbf{u}}^{(k)} + \Delta t\mathbf{f}^{(k)} \tag{4.12f}$$

$$\hat{\mathbf{u}}^{(k)} = \tilde{\mathbf{u}}^{(k)} \tag{4.12g}$$

End loop

(4) Projection and update of the final fields

$$\mathcal{L}\phi^{n+1} = \frac{1}{\Delta t}\mathcal{D}\tilde{\mathbf{u}}^{n+1} \tag{4.12h}$$

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1} - \Delta t\mathcal{G}\phi^{n+1} \tag{4.12i}$$

$$p^{n+1} = p^n + \phi^{n+1} - \frac{1}{2Re}\mathcal{D}\hat{\mathbf{u}}^{n+1} \tag{4.12j}$$

If full convergence of the forcing loop is required more iterations are needed, such as the multidirect forcing scheme of Luo *et al.* (2007) and Breugem (2012). However, the convergence rate of this iteration becomes very slow after several iterations. Therefore the number of iteration of the forcing loop is usually kept low for the computational efficiency.

(*ii*) Immersed boundary projection method (IBPM)

To impose the no-slip boundary condition accurately, Taira & Colonius (2007) proposed the immersed boundary projection method (IBPM). By considering that the boundary force is a Lagrange multiplier for satisfying the no-slip boundary condition on the immersed interface, the immersed boundary method combines the two Lagrange multipliers, the pressure and the boundary force, into a modified Poisson equation solved with the projection method of Perot (1993). The immersed

boundary projection method is written algebraically as

$$\begin{bmatrix} \mathcal{A} & \mathcal{G} & \mathcal{S} \\ \mathcal{D} & 0 & 0 \\ \mathcal{T} & 0 & 0 \end{bmatrix} \begin{pmatrix} \mathbf{u}^{n+1} \\ p^{n+1} \\ \mathbf{f}^{n+1} \end{pmatrix} = \begin{pmatrix} r^n \\ 0 \\ \mathbf{U}_b^{n+1} \end{pmatrix} + \begin{pmatrix} bc_1 \\ bc_2 \\ 0 \end{pmatrix} \tag{4.13}$$

where $\mathcal{A} = \frac{1}{\Delta t}[I - \frac{\Delta t}{2Re}\mathcal{L}]$. Note that we have $\mathcal{S} = \mathcal{T}^{\mathrm{T}}$ and $\mathcal{D} = -\mathcal{G}^{\mathrm{T}}$, the above system is simplified to

$$\begin{bmatrix} \mathcal{A} & \mathcal{Q} \\ \mathcal{Q}^{\mathrm{T}} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{u}^{n+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \tag{4.14}$$

where $\lambda = [p \quad \mathbf{F}^{n+1}]^{\mathrm{T}}$, $\mathcal{Q} = [\mathcal{G}, \mathcal{S}^{\mathrm{T}}]$ and r_1, r_2 include the boundary conditions, the explicit terms in the discretized momentum equation and the no-slip boundary condition at the immersed interface. This system is solved by an approximate block LU decomposition based on the Schur Complement

$$\begin{bmatrix} \mathcal{A} & 0 \\ \mathcal{Q}^{\mathrm{T}} & -\mathcal{Q}^{\mathrm{T}}\mathcal{B}\mathcal{Q} \end{bmatrix} \begin{bmatrix} I & \mathcal{B}\mathcal{Q} \\ 0 & I \end{bmatrix} \begin{pmatrix} \mathbf{u}^{n+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} + \begin{pmatrix} -\frac{\Delta t^N}{2^N}\mathcal{L}^N\mathcal{Q}\lambda \\ 0 \end{pmatrix} \tag{4.15}$$

where $\mathcal{B}$ is the N-th order Taylor series expansion of $\mathcal{A}^{-1}$

$$\mathcal{B} = \Delta t I + \frac{\Delta t^2}{2Re}\mathcal{L} + \frac{\Delta t^3}{(2Re)^2}\mathcal{L}^2 + \cdots + \frac{\Delta t^N}{(2Re)^{N-1}}\mathcal{L}^{N-1} = \sum_{j=1}^{N} \frac{\Delta t^j}{(2Re)^{j-1}}\mathcal{L}^{j-1} \tag{4.16}$$

In practice, this algebraic equation is usually solved in a fractional manner

$$\mathcal{A}\hat{\mathbf{u}}^{n+1} = r_1 \tag{4.17a}$$

$$\mathcal{Q}^{\mathrm{T}}\mathcal{B}\mathcal{Q}\lambda = \mathcal{Q}^T\hat{\mathbf{u}}^{n+1} - r_2 \tag{4.17b}$$

$$\mathbf{u}^{n+1} = \hat{\mathbf{u}}^{n+1} - \mathcal{B}\mathcal{Q}\lambda \tag{4.17c}$$

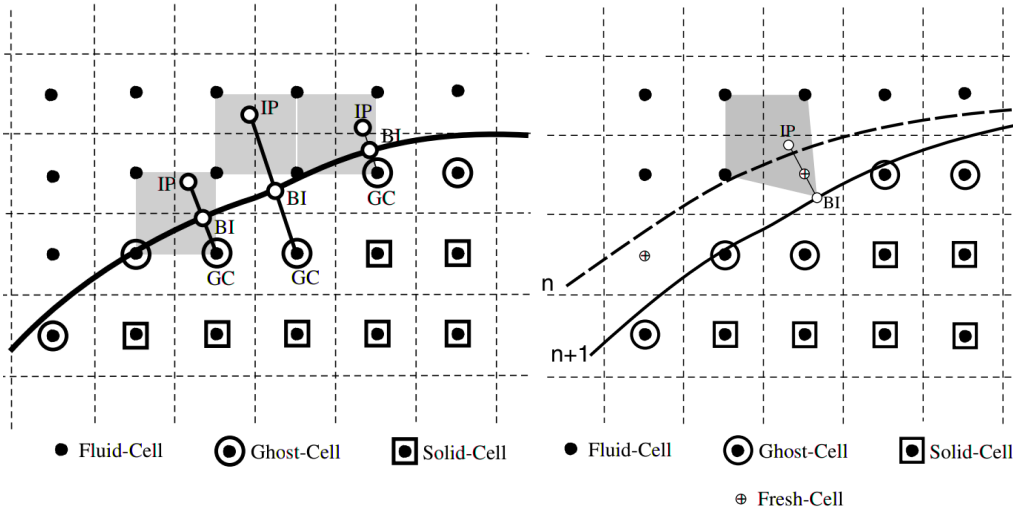
which just follows the fractional step method: prediction and projection. $N = 3$ is suggested in Taira & Colonius (2007) for achieving positive-definiteness of the modified Poisson equation (4.17b).

The immersed boundary projection method is very accurate since it enforces the divergence free condition and the no-slip boundary condition simultaneously in the projection step. In spite of the mathematical rigour and completeness, the immersed boundary method is less efficient for the following reasons. First a large matrix is created towards the modified Poisson equation, which requires more storage and

iterations for a given tolerance. Since the boundary force is a vector, the dimension of the modified Poisson equation increases significantly for three dimensional simulations. Secondly the condition number of the original PPE is undermined significantly due to the singular property of the interpolation and distribution functions, which can lead to convergence problems (Ji *et al.*, 2012). Moreover for moving boundaries, the coefficient matrix of the modified Poisson equation is re-computed and its preconditioner is re-evaluated at every time step, which increases the computational time.

(***iii***) **Ghost-cell immersed boundary method**

Tseng & Ferziger (2003) and Mittal *et al.* (2008) extended the work of Mohd-Yosuf (1997) and Fadlun *et al.* (2000) via a ghost-cell approach. The ghost-cell IBM attempts to achieve a sharp representation of the immersed boundary and to preserve a higher order spatial accuracy. The ghost-cell is defined inside the immersed boundary, such that each ghost-cell has at least one neighbor in the fluid (see Figure 4.6a). The local flow variable is then expressed in terms of polynomial (linear or quadratic) and the ghost-cell value is weighted by the neighboring nodes values.



(a) (b)

Figure 4.6: Ghost-cell methodology of Mittal *et al.* (2008): (a) Identification of the ghost-cell (GC), the image point (IP), and the boundary intercept (BI); (b) The emerged fresh cells due to boundary motion.

The ghost-cell IBM directly imposes the boundary condition at the precise location of the immersed boundary, hence it is suitable for high Reynolds number flow simulations. However, numerical instability may arise when the boundary point is

close to the fluid nodes, caused by large negative weighting coefficients. The image method is used to circumvent this problem in Tseng & Ferziger (2003) and Mittal *et al.* (2008), as shown in Figure 4.6a. The flow variable is evaluated at the image point (IP) and transfered to the ghost cell (GC) by extrapolation. Another issue with the ghost-cell method is the so-called "fresh-cell" problem (Mittal & Iaccarino, 2005; Mittal *et al.*, 2008). The fresh-cells are in the solid at one time step and emerge into the fluid at the next time step because of the boundary motion (see Figure 4.6b). This leads to a temporal discontinuity on the fresh-cells. Mittal *et al.* (2008) handled this problem by interpolating the fluid velocity from neighboring nodes. Further, the identification of the boundary intercept (BI) point is complicated for complex geometries in the ghost-cell IBM, although conceptually simple. Incorrect BI can lead to divergence (Mittal *et al.*, 2008).

(*iv*) **Cut-cell immersed boundary method**

None of the immersed boundary methods we have discussed so far satisfies the underlying conservation laws in the vicinity of the immersed boundary. Ye *et al.* (1999) proposed the cut-cell immersed boundary method on a collocated (non-staggered) grid using the finite volume discretization for the conservation of mass and momentum.

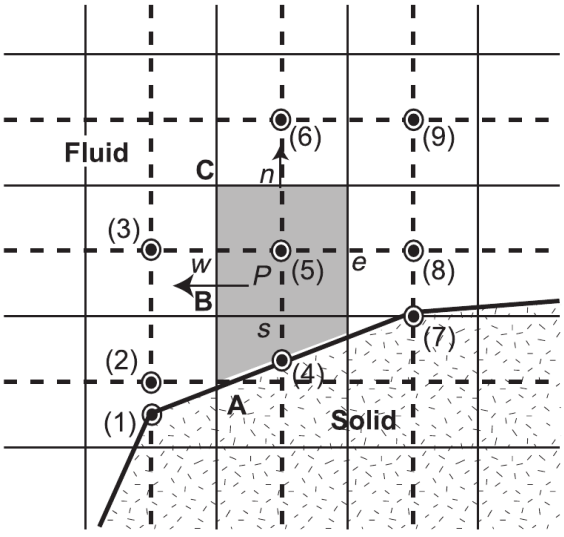


Figure 4.7: Cut-cell immersed boundary method of Ye *et al.* (1999). The fluid cells are reshaped as shown in the shaded zone.

In the cut-cell IBM, the fluid cells that are cut by the immersed boundary are identified, and the intersection of the boundary with these cells is calculated. Next the cut-cells whose center lies in the fluid side are reshaped or absorbed into neighboring

cells otherwise, as shown in Figure 4.7. This method retains the second order spatial accuracy and yields a sharp interface. Moving boundaries have also been considered in (Udaykumar, 2001). However, the extension to three dimensions is extremely difficult. This due to the fact that the cut-cell procedure often yields complex polyhedral cells, which makes the discretization of the Navier-Stokes equations on these cells very complicated (Mittal & Iaccarino, 2005).
Cheny & Botella (2010) extended the collocated cut-cell IBM to the MAC staggered grid, called the LS-STAG method, by using the level-set function to represent the irregular boundary. Super-linear convergence is found in this method and moving boundaries can be also handled.
The cut-cell IBM has a clear advantage over other methods for the conservation properties near the immersed boundary. But it comes at a price of increased complexity in implementation, especially in three dimensions. Similar to previous ghost-cell IBM, the fresh-cells are non-trivial in this sharp interface method (Mittal & Iaccarino, 2005).

4.3.3 Non-primitive variable immersed boundary methods

It is worth noticing that the immersed boundary methods have also been successfully integrated into the fluid solvers in non-primitive variable formulations. Ren *et al.* (2012) implemented the immersed boundary method in the streamfunction-vorticity formulation. The desired boundary condition is achieved by the velocity and vorticity correction. Mimeau *et al.* (2015) proposed a penalization immersed boundary method in the velocity-vorticity fluid solver, where the fluid is assumed to be porous medium and the Brinkman-Navier-Stokes equations are solved. The immersed boundary method based on the discrete streamfunction fluid solver was reported in Colonius & Taira (2008) and Wang & Zhang (2011). The combined lattice Boltzmann method and immersed boundary method (LBM-IBM) has gained a lot of popularities in Huang *et al.* (2007), Dupuis *et al.* (2008), and Favier *et al.* (2014). The enforced boundary condition scheme in LBM-IBM has been considered in Wu & Shu (2009), Wu & Shu (2010) and Wang *et al.* (2015). As mentioned previously, despite the efficiency, the non-primitive variable formulations can suffer from the difficulty of the boundary condition impositions.

4.3.4 Summary

Various immersed boundary methods have been investigated systematically in this subsection. The first group of continuous forcing methods are suitable for deformable

boundaries but become numerically instable when extended to rigid boundaries. The second group of discrete forcing methods do not need artificial constants to enforce the boundary condition at rigid boundaries, and they are stable with larger time steps. In the second IBM category, the sharp interface methods provides a better boundary representation and a higher accuracy but increases the complexity of implementation, compared to the diffused interface methods. Whereas the diffused interface methods (e.g. the direct forcing method) are more easily to work with moving boundary problems. There is no need to consider the cell identification and the cell-trimming encountered by the sharp interface methods. Hence the diffused interface methods are suitable for the simulation of fluid-structure interactions.

4.4 Moving immersed boundary method (MIBM)

Inspired by the direct forcing IBM of Uhlmann (2005) and Kempe & Fröhlich (2012a), the multidirect forcing IBM of Luo *et al.* (2007) and Breugem (2012), and the IBPM of Taira & Colonius (2007), we present a novel moving immersed boundary method is this subsection. We hope the new method can maintain the efficiency of the direct forcing IBM, but with an improved accuracy like that in the multidirect forcing IBM and the IBPM.

4.4.1 Derivation of the moving force equation

To this end, we first take the immersed boundary forcing part from the explicit method of Kempe & Fröhlich (2012a) for consideration, i.e., (4.12c), (4.12d), (4.12e) and (4.12f). By dropping the temporal superscript $n+1$ for convenience, the immersed boundary forcing part is written as

$$\hat{\mathbf{U}} = \mathcal{T}\hat{\mathbf{u}} \tag{4.18a}$$

$$\mathbf{F} = \frac{\mathbf{U}_b - \hat{\mathbf{U}}}{\Delta t} \tag{4.18b}$$

$$\mathbf{f} = \mathcal{S}\mathbf{F} \tag{4.18c}$$

$$\tilde{\mathbf{u}} = \hat{\mathbf{u}} + \Delta t\mathbf{f} \tag{4.18d}$$

We require that the interpolated velocity satisfies the no-slip wall boundary condition on the immersed interface after the immersed boundary forcing, namely $\mathcal{T}\tilde{\mathbf{u}} = \mathbf{U}_b$, then

$$\mathcal{T}(\hat{\mathbf{u}} + \Delta t\mathbf{f}) = \mathbf{U}_b \tag{4.19}$$

Substituting (4.18c) into (4.19) gives

$$\mathcal{T}(\hat{\mathbf{u}} + \Delta t \mathcal{S}\mathbf{F}) = \mathbf{U}_b \tag{4.20}$$

which can be rearranged in order to separate the boundary force

$$(\mathcal{T}\mathcal{S})\mathbf{F} = \frac{\mathbf{U}_b - \mathcal{T}\hat{\mathbf{u}}}{\Delta t} \tag{4.21}$$

We donate $\mathcal{M} = \mathcal{T}\mathcal{S}$ the moving force coefficient matrix, since $\mathcal{M}$ is a function of the solid position. As the solid moves it changes its value, so that the force is redistributed just like the boundary force moves. The moving force equation can be rewritten in a more concise form

$$\mathcal{M}\mathbf{F} = \mathbf{F}^e \tag{4.22}$$

where $\mathbf{F}^e = (\mathbf{U}_b - \mathcal{T}\hat{\mathbf{u}})/\Delta t$ is exactly the explicit forcing value used by Kempe & Fröhlich (2012a).

Compared to the modified Poisson equation in the IBPM of Taira & Colonius (2007), the moving force equation (4.22) is much smaller in size and easier to work with. At each dimension (x or y), the size of the moving force coefficient matrix is $n_b \times n_b$ since $\mathcal{T} \in \mathbb{R}^{n_b \times n_x n_y}$ and $\mathcal{S} \in \mathbb{R}^{n_x n_y \times n_b}$. Therefore, for moving boundaries, its update is computational less expensive than the modified Poisson equation.

Note that $\mathcal{S} = (\Delta V_l/h^2)\mathcal{T}^{\mathrm{T}}$ if the same function is used for interpolation and spreading, where $\Delta V_l/h^2 \approx 1$ is the volume ratio between the fluid and the solid cell. As a result, the moving force coefficient matrix $\mathcal{M} = (\Delta V_l/h^2)\mathcal{T}\mathcal{T}^{\mathrm{T}}$ is symmetric. It is also found that $\mathcal{M}$ is positive-definite irrespective of the time step and the approximation order as in the IBPM Taira & Colonius (2007). Moreover, the moving force equation is well conditioned. It converges very quickly by using the conjugate gradient method.

The matrix $\mathcal{M} \in \mathbb{R}^{n_b \times n_b}$ is in general sparse with most non-zero elements near the diagonal position, thus it can be stored with any sparse matrix format like the CRS in our present work. As for $\mathcal{T}$ and $\mathcal{S}$, they are stored in the CRS and CCS formats separately, which allows to manipulate their product very easily. Another reason is that CCS can store the matrix transpose directly from CRS without changing the vector values. Figure 4.8 shows the moving force matrix pattern resulted from the lid-driven cavity flow with an immersed cylinder case ($n_b = 51$) in Section 4.7.2.

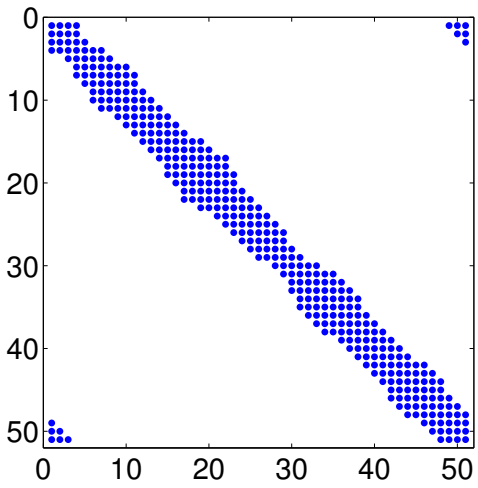


Figure 4.8: Moving force coefficient matrix pattern.

4.4.2 Implementation in the projection method

In previous subsection, we have derived an additional moving force equation for imposing the no-slip wall condition implicitly. Based on that, we can construct a novel immersed boundary method, termed as the moving immersed boundary method (MIBM) in this thesis. The greatest advantage of the moving force equation is that it can be easily integrated into any fluid solver as a plug-in to play a role. In the present work, we demonstrate this flexibility by incorporating the moving force equation into the rotational incremental pressure-correction projection method. For the sake of simplicity, we rewrite the governing equations (4.2) as

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t} = \mathcal{H} + \mathcal{P} + \mathcal{F} \tag{4.23a}$$

$$\mathcal{D}\mathbf{u}^{n+1} = 0 \tag{4.23b}$$

$$\mathcal{T}\mathbf{u}^{n+1} = \mathbf{U}_b^{n+1} \tag{4.23c}$$

where $\mathcal{H}$, $\mathcal{P}$ and $\mathcal{F}$ are the operators defined as

$$\mathcal{H} := -\left[\frac{3}{2}\mathcal{N}(\mathbf{u}^{n}) - \frac{1}{2}\mathcal{N}(\mathbf{u}^{n-1})\right] + \frac{1}{2Re}\mathcal{L}(\mathbf{u}^{n+1}+\mathbf{u}^{n}) - \mathcal{G}p^{n} \tag{4.24a}$$

$$\mathcal{P} := -\mathcal{G}\phi^{n+1} \tag{4.24b}$$

$$\mathcal{F} := \mathcal{S}\mathbf{F}^{n+1} \tag{4.24c}$$

To decouple the momentum equation (4.23a) from the divergence free condition

(4.23b) and the no-slip wall condition on the interface (4.23c), we perform the following operator splitting algorithm:
(1) Prediction step by ignoring the immersed objects

$$\frac{\hat{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t} = \mathcal{H}(\hat{\mathbf{u}}^{n+1}) \tag{4.25}$$

(2) Immersed boundary forcing step for satisfying the no-slip wall condition on the interface

$$\frac{\tilde{\mathbf{u}}^{n+1} - \hat{\mathbf{u}}^{n+1}}{\Delta t} = \mathcal{F} \tag{4.26a}$$

$$\mathcal{T}\tilde{\mathbf{u}}^{n+1} = \mathbf{U}_b^{n+1} \tag{4.26b}$$

Applying (4.26b) to (4.26a) gives the moving force equation that we have defined previously

$$\mathcal{M}\mathbf{F}^{n+1} = \frac{\mathbf{U}_b^{n+1} - \mathcal{T}\hat{\mathbf{u}}^{n+1}}{\Delta t} \tag{4.27a}$$

Once the boundary force is determined, we correct the fluid velocity with

$$\tilde{\mathbf{u}}^{n+1} = \hat{\mathbf{u}}^{n+1} + \Delta t \mathcal{S}\mathbf{F}^{n+1} \tag{4.27b}$$

(3) Projection step for obtaining the divergence free velocity $\mathbf{u}^{n+1}$ and the final pressure p^{n+1}

$$\frac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}^{n+1}}{\Delta t} = \mathcal{P} \tag{4.28a}$$

$$\mathcal{D}\mathbf{u}^{n+1} = 0 \tag{4.28b}$$

Applying the divergence operator to (4.28a) and using the divergence free condition (4.28b) gives

$$\mathcal{L}\phi^{n+1} = \frac{1}{\Delta t}\mathcal{D}\tilde{\mathbf{u}}^{n+1} \tag{4.29a}$$

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1} - \Delta t \mathcal{G}\phi^{n+1} \tag{4.29b}$$

In the rotational incremental pressure-correction projection method, the final pressure is advanced by

$$p^{n+1} = p^n + \phi^{n+1} - \frac{1}{2Re}\mathcal{D}\hat{\mathbf{u}}^{n+1} \tag{4.30}$$

The overall scheme still follows the regular fractional step method, and the moving

immersed boundary forcing can be viewed as another operator splitting. The global structure of the MIBM is shown in Figure 4.9.

In the moving immersed boundary method, the coefficient matrix of the pressure Poisson equation is unchanged during the calculation. Therefore, convergence problems will not occur. Compared to the modified Poisson equation in the IBPM (Taira & Colonius, 2007), the moving force equation is much easier to work with. Even though the interface velocity condition is enforced before the projection step, we have found that the velocity on the immersed boundary is essentially unchanged after the projection step. The same observation has also been made by Kempe & Fröhlich (2012a) and Fadlun *et al.* (2000).

It is worth noting that the present moving immersed boundary method recovers to the explicit method of Kempe & Fröhlich (2012a) with one iteration in the forcing loop, if $\mathcal{M}$ is set to the identity matrix. However it is not the case, hence our method is implicit.

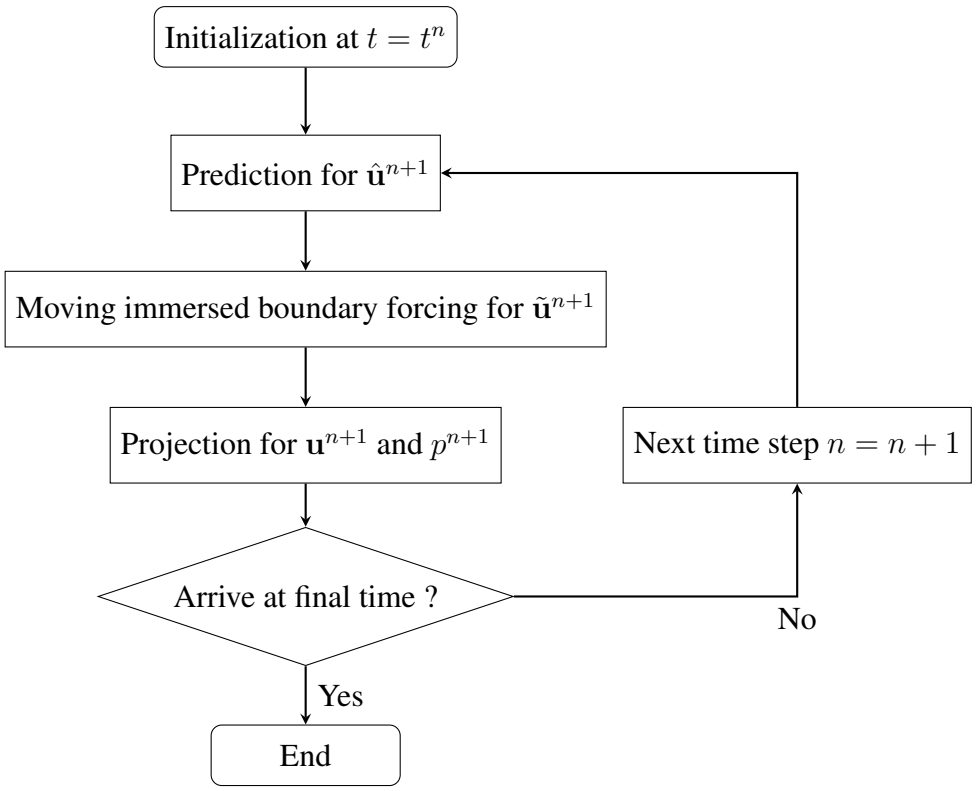


Figure 4.9: Global structure of the moving immersed boundary method.

4.5 Interpolation techniques

Since the Lagrangian markers for representing the immersed boundary are not coincident with the underlying fluid Eulerian grids, interpolation scheme is needed for the transfer of quantities between the two meshes. Many different techniques are available and they can be broadly classified into two groups: (a) Schemes that use a local velocity reconstruction; (b) Schemes that spread the boundary force to the fluid cells over the vicinity of the immersed boundary.

The first group includes the direct forcing IBM of Mohd-Yosuf (1997) and Fadlun *et al.* (2000) and the ghost-cell IBM of Tseng & Ferziger (2003) and Mittal *et al.* (2008). No spurious spreading of the boundary force into the fluid will occur. High degree of accuracy can be obtained with certain local reconstruction schemes. However in case of moving boundaries, spurious oscillating can occur towards the force and deteriorates the solution in the direct forcing IBM. Special care has also to be taken about the fresh cells in the ghost-cell IBM.

Current method is categorized into the second group. The boundary force is distributed by using a smoothed kernel function, such as the discrete delta function (Peskin, 1972b; Beyer & LeVeque, 1992; Lai & Peskin, 2000; Uhlmann, 2005; Taira & Colonius, 2007; Kempe & Fröhlich, 2012a), the reproducing kernel particle method (RKPM) (Wang & Liu, 2004; Zhang *et al.*, 2004; Pinelli *et al.*, 2010), the moving least squares method (MLS) (Vanella & Balaras, 2009) and the radial basis function (RBF) (Toja-Silva *et al.*, 2014). Even though the interface is diffused, they are relatively easy to incorporate the boundary motion. As indicated by Beyer & LeVeque (1992), original accuracy of the fluid solver can be retained if the discrete delta function is carefully chosen.

We assume that the two-dimensional δ_h is given by a product of one-variable functions which scale with the mesh size h as follows

$$\delta_h(\mathbf{x}) = \frac{1}{h^2}\phi(\frac{x}{h})\phi(\frac{y}{h}) \tag{4.31}$$

Donate $r = x/h$ or y/h, the discrete delta functions commonly have the following properties:

- ϕ has narrow compact support so that the evaluation is cheap and better boundary resolution is obtained. Donate $2M$ the support width, thus

$$\phi(r) \neq 0, \quad \text{only if} \quad |r| \leqslant Mh \tag{4.32}$$

- ϕ is second-order accurate for smooth fields.

- ϕ satisfies the moment conditions, which guaranties the mass, force ($m = 0$, i.e. partion of unity) and torque identities ($m = 1$) between Eulerian and Lagrangian grids. If ϕ meets the moment conditions up to order $m - 1$, it is then of moment order m.

$$\sum_j \phi(r - j) = 1 \quad \text{for all real } r \quad \text{if } m = 0 \tag{4.33}$$

$$\sum_j (r - j)^m \phi(r - j) = 0 \quad \text{for all real } r \quad \text{if } m > 0 \tag{4.34}$$

The most simplest form can be the 2-point hat function ϕ_1 (Beyer & LeVeque, 1992)

$$\phi_1(r) = \begin{cases} 1 - |r| & |r| < 1 \\ 0 & 1 \leqslant |r| \end{cases} \tag{4.35}$$

which however is not smooth in the support domain. A smoother version is provided in Peskin (2002) with four points width

$$\phi_2(r) = \begin{cases} \frac{1}{4}\left(1 + cos(\frac{\pi r}{2})\right) & |r| < 2 \\ 0 & 2 \leqslant |r| \end{cases} \tag{4.36}$$

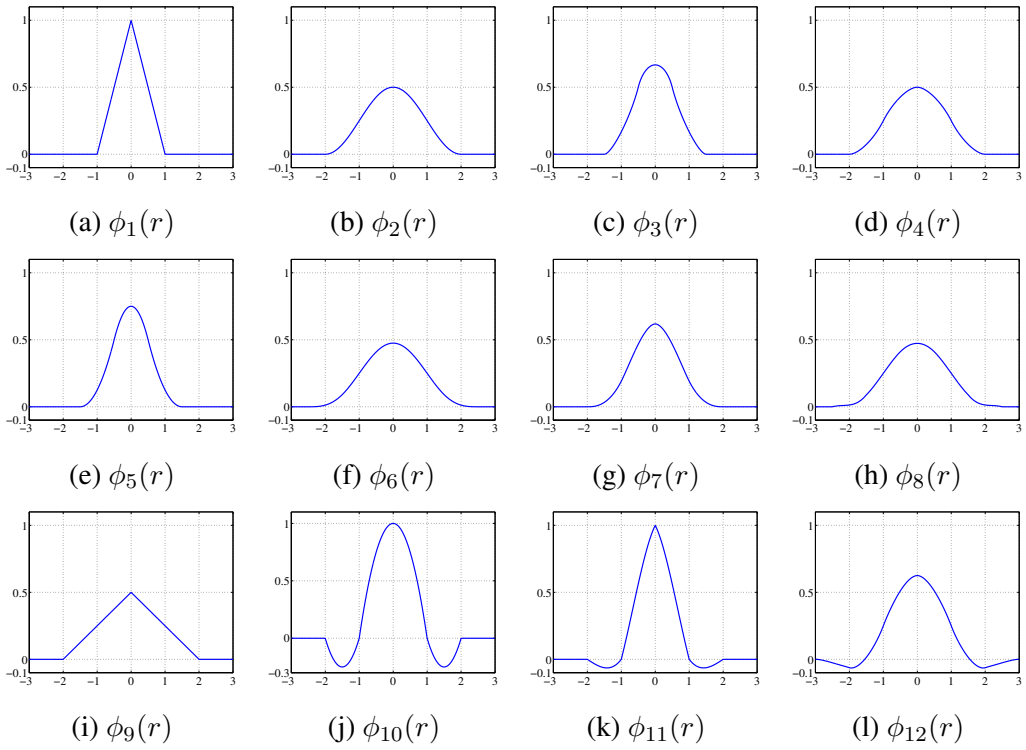
Roma *et al.* (1999) designed a 3-point-width function for the staggered mesh, which is widely used in the literature

$$\phi_3(r) = \begin{cases} \frac{1}{3}\left(1 + \sqrt{-3r^2 + 1}\right) & |r| < 0.5 \\ \frac{1}{6}\left(5 - 3|r| - \sqrt{-3(1 - |r|)^2 + 1}\right) & 0.5 \leqslant |r| < 1.5 \\ 0 & 1.5 \leqslant |r| \end{cases} \tag{4.37}$$

Another popular function of 4-point-width ϕ_4 constructed by Peskin (2002) is

$$\phi_4(r) = \begin{cases} \frac{1}{8}\left(3 - 2|r| + \sqrt{1 + 4|r| - 4r^2}\right) & |r| < 1 \\ \frac{1}{8}\left(5 - 2|r| + \sqrt{-7 + 12|r| - 4r^2}\right) & 1 \leqslant |r| < 2 \\ 0, & 2 \leqslant |r| \end{cases} \tag{4.38}$$

It is often observed that the above versions lead to non-physical oscillations to the boundary force in moving boundary simulations. Yang *et al.* (2009) attributed this spurious oscillation to the lack of certain moment conditions of the discrete



(a) $\phi_1(r)$ (b) $\phi_2(r)$ (c) $\phi_3(r)$ (d) $\phi_4(r)$

(e) $\phi_5(r)$ (f) $\phi_6(r)$ (g) $\phi_7(r)$ (h) $\phi_8(r)$

(i) $\phi_9(r)$ (j) $\phi_{10}(r)$ (k) $\phi_{11}(r)$ (l) $\phi_{12}(r)$

Figure 4.10: Plots of discrete delta functions.

delta function derivatives. A smoothing technique is then proposed to improve the moment condition

$$\phi^*(r) = \int_{r-0.5}^{r+0.5} \phi(r')dr' \tag{4.39}$$

Using this technique, the above discrete delta functions are smoothed as follows

$$\phi_5(r) = \begin{cases} \frac{3}{4} - r^2 & |r| < 0.5 \\ \frac{9}{8} - \frac{3}{2}|r| + \frac{1}{2}r^2 & 0.5 \leqslant |r| < 1.5 \\ 0 & 1.5 \leqslant |r| \end{cases} \tag{4.40}$$

which is the smoothed version of the 2-point hat function ϕ_1. The smoothed 4-point cosine function of ϕ_2 is given by

$$\phi_6(r) = \begin{cases} \frac{1}{4\pi}\left(\pi + 2sin\left(\frac{\pi}{4}(2r+1)\right) - 2sin\left(\frac{\pi}{4}(2r-1)\right)\right) & |r| < 1.5 \\ -\frac{1}{8\pi}\left(-5\pi + 2\pi|r| + 4sin\left(\frac{\pi}{4}(2|r|-1)\right)\right) & 1.5 \leqslant |r| < 2.5 \\ 0 & 2.5 \leqslant |r| \end{cases} \tag{4.41}$$

The smoothed 3-point version of ϕ_3 is

$$\phi_7(r) = \begin{cases} \frac{17}{48} + \frac{\sqrt{3}\pi}{108} + \frac{|r|}{4} - \frac{r^2}{4} + \frac{1-2|r|}{16}\sqrt{-12r^2+12|r|+1} \\ \qquad - \frac{\sqrt{3}}{12} arcsin\left(\frac{\sqrt{3}}{2}(2|r|-1)\right) & |r| < 1 \\ \frac{55}{48} - \frac{\sqrt{3}\pi}{108} - \frac{13|r|}{12} + \frac{r^2}{4} + \frac{2|r|-3}{48}\sqrt{-12r^2+36|r|-23} \\ \qquad + \frac{\sqrt{3}}{36}\arcsin\left(\frac{\sqrt{3}}{2}(2|r|-3)\right) & 1 \leqslant |r| < 2 \\ 0 & 2 \leqslant |r| \end{cases} \tag{4.42}$$

The smoothed 4-point version of ϕ_4 is

$$\phi_8(r) = \begin{cases} \frac{3}{8} + \frac{\pi}{32} - \frac{r^2}{4} & |r| < 0.5 \\ \frac{1}{4} + \frac{1-|r|}{8}\sqrt{-2+8|r|-4r^2} \\ \qquad - \frac{1}{8} arcsin\left(\sqrt{2}(|r|-1)\right) & 0.5 \leqslant |r| < 1.5 \\ \frac{17}{16} - \frac{\pi}{64} - \frac{3|r|}{4} + \frac{r^2}{8} + \frac{|r|-2}{16}\sqrt{-14+16|r|-4r^2} \\ \qquad + \frac{1}{16} arcsin\left(\sqrt{2}(|r|-2)\right) & 1.5 \leqslant |r| < 2.5 \\ 0 & 2.5 \leqslant |r| \end{cases} \tag{4.43}$$

Yang *et al.* (2009) and Beyer & LeVeque (1992) compared the hat function with a wider support of 4-point-width

$$\phi_9(r) = \begin{cases} 0.5 - 0.25|r| & |r| < 2 \\ 0 & 2 \leqslant |r| \end{cases} \tag{4.44}$$

Considering that the velocity usually has a kink at the interface where one or more derivatives are discontinuous, Beyer & LeVeque (1992) constructed the ϕ_{10} by using linear extrapolation on each side of the interface

$$\phi_{10}(r) = \begin{cases} 1 - r^2 & |r| < 1 \\ 2 - 3|r| + r^2 & 1 \leqslant |r| < 2 \\ 0 & 2 \leqslant |r| \end{cases} \tag{4.45}$$

Griffith & Peskin (2005) improved the accuracy by using the smoothed delta func-

tions that satisfy four moment conditions, such as 4-point piecewise cubic function

$$\phi_{11}(r) = \begin{cases} 1 - \frac{1}{2}|r| - r^2 + \frac{1}{2}|r|^3 & |r| < 1 \\ 1 - \frac{11}{6}|r| + r^2 - \frac{1}{6}|r|^3 & 1 \leqslant |r| < 2 \\ 0 & 2 \leqslant |r| \end{cases} \tag{4.46}$$

and the 6-point delta function

$$\phi_{12}(r) = \begin{cases} \frac{61}{112} - \frac{11}{42}|r| - \frac{11}{56}r^2 + \frac{1}{12}|r|^3 + \frac{\sqrt{3}}{336}(243 + 1584|r| \\ \quad - 748r^2 - 1560|r|^3 + 500r^4 + 336|r|^5 - 112r^6)^{1/2} & |r| < 1 \\ \frac{21}{16} + \frac{7}{12}|r| - \frac{7}{8}r^2 + \frac{1}{6}|r|^3 - \frac{3}{2}\phi_{12}(|r| - 1) & 1 \leqslant |r| < 2 \\ \frac{9}{8} - \frac{23}{12}|r| + \frac{3}{4}r^2 - \frac{1}{12}|r|^3 + \frac{1}{2}\phi_{12}(|r| - 2) & 2 \leqslant |r| < 3 \\ 0 & 3 \leqslant |r| \end{cases} \tag{4.47}$$

Those functions are displayed in Figure 4.10 and their properties are summarized in Table 4.1.

	Support width	Continuity	Moment condition
$\phi_1(r)$	2	C^0	1
$\phi_2(r)$	4	C^1	0
$\phi_3(r)$	3	C^1	1
$\phi_4(r)$	4	C^1	1
$\phi_5(r)$	3	C^1	1
$\phi_6(r)$	5	C^2	1
$\phi_7(r)$	4	C^2	1
$\phi_8(r)$	5	C^2	1
$\phi_9(r)$	4	C^0	1
$\phi_{10}(r)$	4	C^2	1
$\phi_{11}(r)$	4	C^3	4
$\phi_{12}(r)$	6	C^3	4

Table 4.1: Properties of various discrete delta functions.

4.6 Comparison with different immersed boundary methods

To demonstrate the accuracy and efficiency of present moving immersed boundary method, we perform the following test

$$\begin{aligned} &\text{Given} \quad u_0(x,y) = e^x cos\, y - 2, \qquad 0 \leqslant x, y \leqslant 1 \\ &\text{Find } F \text{ such that } u(x,y) = u_0(x,y) + \mathcal{S}F = U_b \quad \text{on } \Gamma_s \end{aligned}$$

where Γ_s is represented by a circle with a radius of 0.2 centered at $(0.52, 0.54)$ and $U_b = 0$. The domain is covered by 64×64 nodes with around 81 Lagrangian points defining the circle surface. Δt is set to 1.

Algorithm 3: Explicit direct forcing IBM of Uhlmann (2005).

$U = \mathcal{T} u_0$
$F = (U_b - U)/\Delta t$
$u = u_0 + \mathcal{S}F$

Algorithm 4: Improved explicit forcing IBM of Kempe & Fröhlich (2012a).

$u^0 = u_0$
for $k = 0$ to 2 **do**
    $U^{(k)} = \mathcal{T} u^{(k)}$
    $F^{(k)} = (U_b - U^{(k)})/\Delta t$
    $u^{(k+1)} = u^{(k)} + \mathcal{S}F^{(k)}$
    $k = k + 1$
end
$u = u^{(k+1)}$

In this test, the fluid equations are not solved and only the immersed boundary forcing part is considered. The initial field $u_0(x, y)$ can be seen as a predicted fluid velocity component in one direction. This test is to examine different immersed boundary forcing methods for imposing the desired velocity U_b at the interface Γ_s via a boundary force F. The present MIBM (shown in Algorithm 6) is compared to the explicit direct forcing IBM of Uhlmann (2005) (shown in Algorithm 3), the improved explicit direct forcing IBM of Kempe & Fröhlich (2012a) (shown

Algorithm 5: Iterative implicit IBM of Luo *et al.* (2007) and Breugem (2012).

$u^0 = u_0$
for $k = 0$ to $k_{\max}$ **do**
  $U^{(k)} = \mathcal{T} u^{(k)}$
  $F^{(k)} = (U_b - U^{(k)})/\Delta t$
  $u^{(k+1)} = u^{(k)} + \mathcal{S} F^{(k)}$
  if $||\mathcal{T} u^{(k+1)} - U_b|| <$ tolerance **then**
    break
  else
    $k = k + 1$
  end
end
$u = u^{(k+1)}$

Algorithm 6: Present implicit MIBM.

Construct $\mathcal{M} = (\Delta V_l / h^2) \mathcal{T} \mathcal{T}^{\mathrm{T}}$
$U = \mathcal{T} u_0$
$F^e = (U_b - U)/\Delta t$
Solve $\mathcal{M} F = F^e$
$u = u_0 + \mathcal{S} F$

in Algorithm 4), and the iterative implicit multidirect forcing IBM of Luo *et al.* (2007) and Breugem (2012) (shown in Algorithm 5).

Figure 4.11a displays the result of the explicit direct forcing IBM of Uhlmann (2005), where u is far away from zero over the immersed boundary compared to Figure 4.11c. The accuracy is improved after 3 iterations with the method of Kempe & Fröhlich (2012a), as shown in Figure 4.11b. Figure 4.11d reveals that the results are nearly the same for present method with the iterative implicit multidirect forcing IBM of Luo *et al.* (2007) and Breugem (2012).

Table 4.2 compares the computational time and velocity error on the interface in these immersed boundary methods. The error is measured in L_2-norm and the tolerance is 1×10^{-15}. The method of Uhlmann (2005) is the quickest due to its explicit nature, but it suffers a large error of 3.01×10^{-1} on the immersed interface. The forcing loop of Kempe & Fröhlich (2012a) reduces the error by a factor of 4 with 3 iterations. However, the error of 7.41×10^{-2} is still considered large.

The iterative implicit multiforcing IBM of Luo *et al.* (2007) and Breugem (2012) is required to converge towards the machine precision, but it takes approximately 606

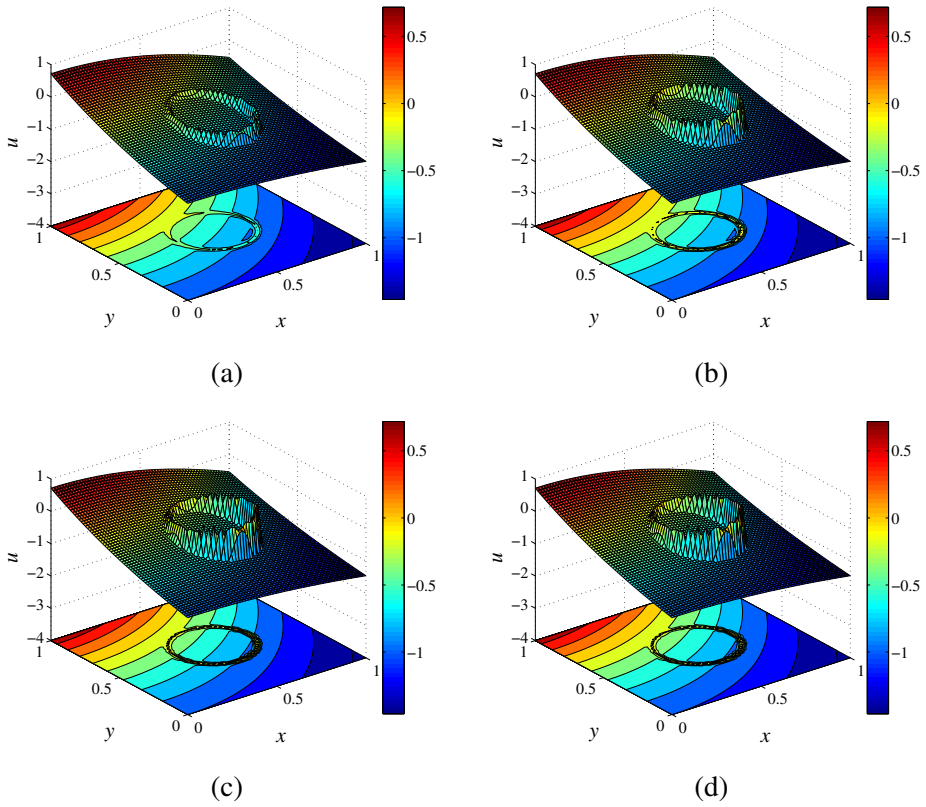


Figure 4.11: Contour of the scalar field after the immersed boundary forcing: (a) with the explicit direct forcing IBM of Uhlmann (2005); (b) with the improved explicit direct forcing IBM of Kempe & Fröhlich (2012a); (c) with the iterative implicit multidirect forcing IBM of Luo *et al.* (2007) and Breugem (2012); (d) with present MIBM.

times more additional computational effort than the explicit method of Uhlmann (2005). Actually, the convergence rate in the iterative implicit method decreases dramatically after about 10 iterations, as shown in Figure 4.12. In order to reduce the error to 1×10^{-6}, around 1000 iterations are needed. 4443 iterations are needed for the machine precision. Therefore the iteration number in the iterative implicit multiforcing IBM is often kept low in practice (Breugem, 2012).

The present MIBM converges to the same machine precision only with 60 iterations by using the conjugate gradient solver. The computation is not increased considerably compared to the explicit method of Uhlmann (2005), as we can see that the present method only takes twice the amount of computational time of the direct forcing IBM of Uhlmann (2005). It worth noticing that present method is almost as efficient as the explicit direct forcing IBM of Kempe & Fröhlich (2012a).

	Time (s)	Iter.	Error
Uhlmann (2005)	6.02×10^{-3}	1 (fixed)	3.01×10^{-1}
Kempe & Fröhlich (2012a)	1.71×10^{-2}	3 (fixed)	7.41×10^{-2}
Luo *et al.* (2007) and Breugem (2012)	3.65×10^{1}	4443	9.96×10^{-16}
Present	1.33×10^{-2}	60	8.29×10^{-16}

Table 4.2: Comparison of the computational time and the velocity error with various immersed boundary methods. A fixed iteration is used in the methods of Uhlmann (2005) and Kempe & Fröhlich (2012a), while the other methods are solved until convergence under a tolerance of 1×10^{-15}.

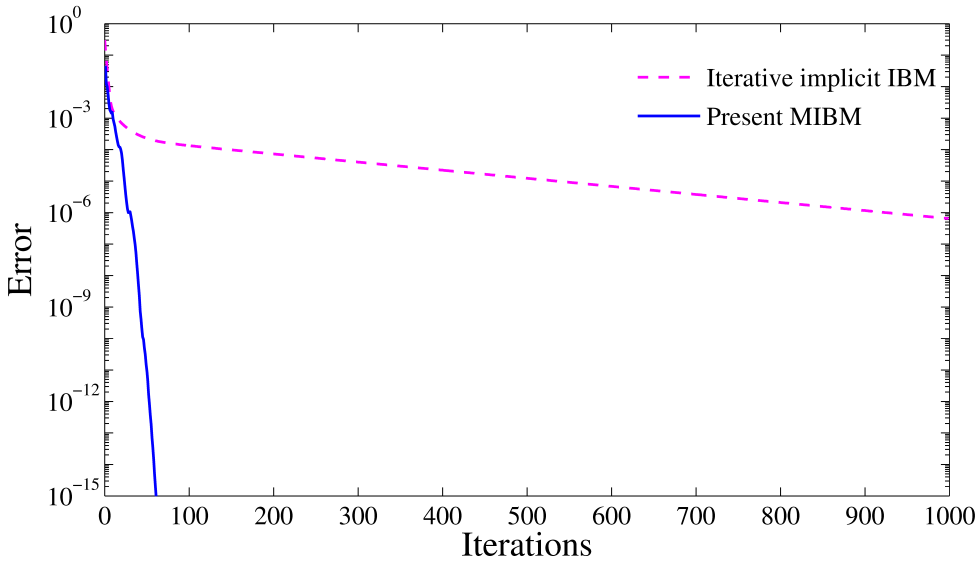


Figure 4.12: Comparison of convergence between present MIBM with the iterative implicit multidirect forcing IBM of Luo *et al.* (2007) and Breugem (2012).

4.7 Numerical examples

4.7.1 Convergence test

In order to assess the accuracy of present immersed boundary method, we re-consider the two-dimensional decaying vortices problem with an cylinder immersed in the flow. This case was also considered by Kim *et al.* (2001) and Uhlmann (2005). The initial and boundary conditions for the flow domain are taken from the following analytical solution

$$
\begin{aligned}
u(x,y,t) &= -cos(\pi x)sin(\pi y)e^{-2\pi^2 t/Re} \\
v(x,y,t) &= sin(\pi x)cos(\pi y)e^{-2\pi^2 t/Re} \\
p(x,y,t) &= -\frac{1}{4}(cos(2\pi x) + sin(2\pi y))e^{-4\pi^2 t/Re}
\end{aligned}
\tag{4.48}
$$

The immersed cylinder with a radius of unity is fixed at the origin of the computational domain $[-1.5, 1.5] \times [-1.5, 1.5]$. The Reynolds number Re is prescribed to be 10. The time-dependent velocity boundary conditions at the embedded cylinder surface are enforced by the present immersed boundary method. A small time step is used with $\Delta t = 0.001$ to ensure that the temporal truncation error is negligible compared to the spatial one. The equations are advanced for $0 \leqslant t \leqslant 0.2$. Then the computed field inside the embedded cylinder at $t = 0.2$ is compared to the analytical solution.

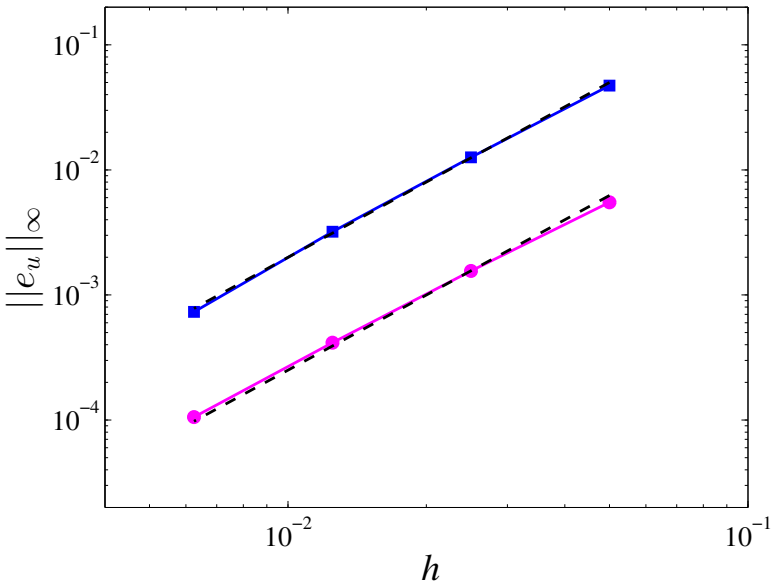


Figure 4.13: Maximum error of the velocity field u at $t = 0.2$ as a function of mesh width h, for the Taylor-Green vortices problem: "■" with the embedded cylinder, "●" without the embedded cylinder. The dashed lines are of second order.

Figure 4.13 shows the variation of the maximum error of velocity field in u as a function of the mesh width h. The errors are taken from two situations: with the immersed cylinder and without the immersed cylinder, i.e. without the immersed boundary method. It is demonstrated from Figure 4.13 that the proposed method retains the second order accuracy (with a slope of 2), which corresponds to the interpolation properties of the discrete delta function for smooth fields.

4.7.2 Lid-driven cavity flow with an embedded cylinder

In this test, we compare the accuracy of present non-body conforming MIBM with the body-conforming mesh method. The domain configuration and the boundary conditions are taken the same as in the lid-driven cavity flow case, except that we set up a fixed cylinder in the domain center. In order to compare with Vanella & Balaras (2009), the diameter of the cylinder is set to $D = 0.4L$ with L being the cavity length. The Reynolds number is 1000 in this study based on the cavity length. Figure 4.14 shows the mesh setup for present immersed boundary approach and the body-conforming mesh method. A uniform mesh of 200×200 is employed in the immersed boundary method, and the same mesh size is used for the body-conforming mesh method for comparison.

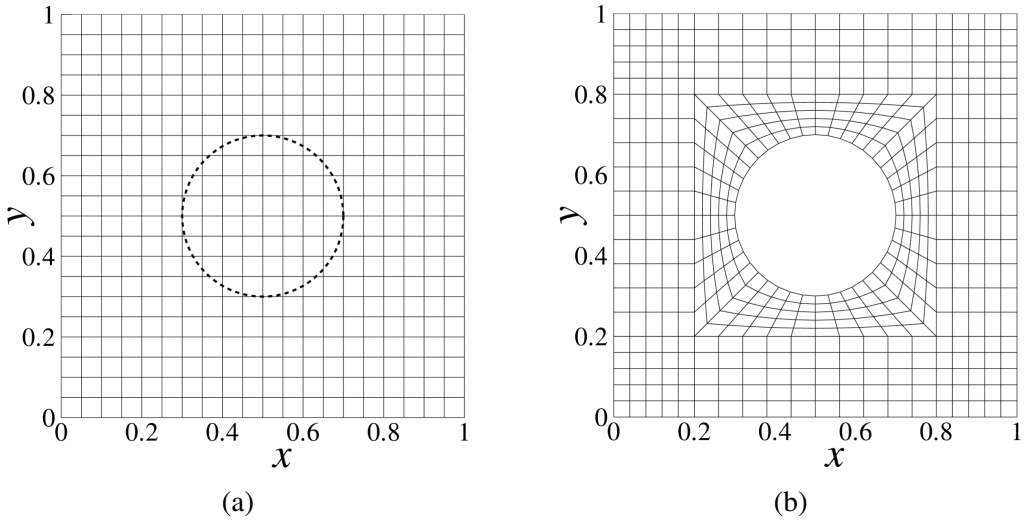


Figure 4.14: Mesh setup for the cavity flow with a fixed cylinder: (a) Immersed boundary approach (the embedded cylinder is represented by the thick dashed line); (b) Body-conforming mesh method. For better visualization, only 20×20 meshes are shown here.

The flow reaches a final steady state as the time advances. The convergence criterion is set to $||\mathbf{u}^{n+1} - \mathbf{u}^{n}||_2/||\mathbf{u}^{n+1}||_2 < 1 \times 10^{-8}$. Figure 4.15 shows the vorticity contours and streamlines for the flow at $Re = 1000$, which are similar to the results of Vanella & Balaras (2009). As we can see, three vortices emerge in the flow. One at the upper right position of the cylinder and two near the bottom at the left and the right corners. It is noteworthy that the upper vortex is generated by the presence of the fixed cylinder. The flow fields outside the cylinder are essentially the same for both the immersed boundary method and the body-conforming mesh method. The only difference is that there is a flow inside the cylinder in the immersed boundary method, which however is the key idea of the immersed boundary method to replace

the solid domain with fluid. The velocity component u at the vertical midline $x = 0.5$ and the velocity component v at the horizontal midline $y = 0.5$ are plotted in Figure 4.16. The velocity profiles of both methods match pretty well. The location of the three vortices centers are also listed in Table 4.3. Very close results have been obtained.

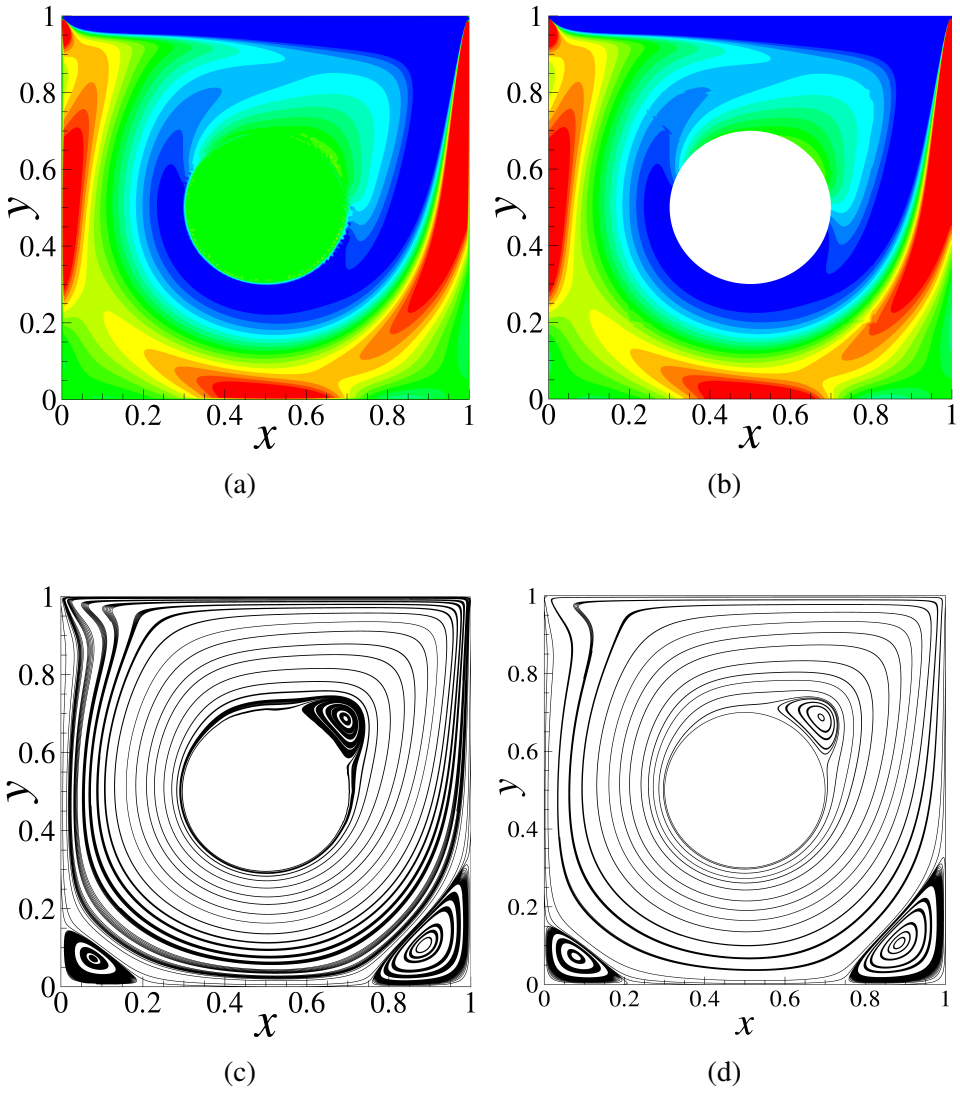
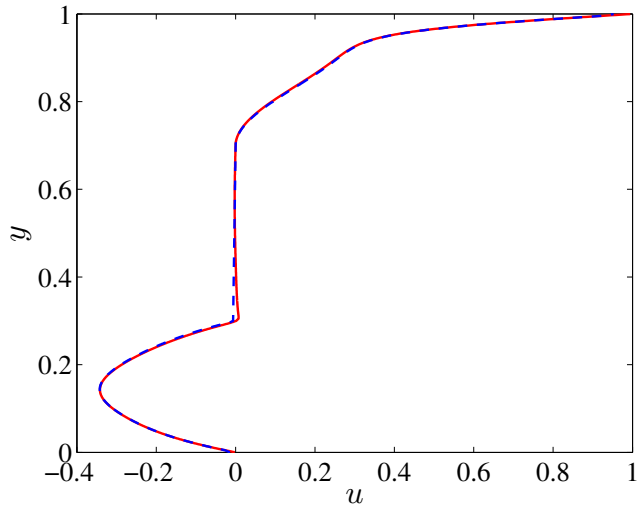
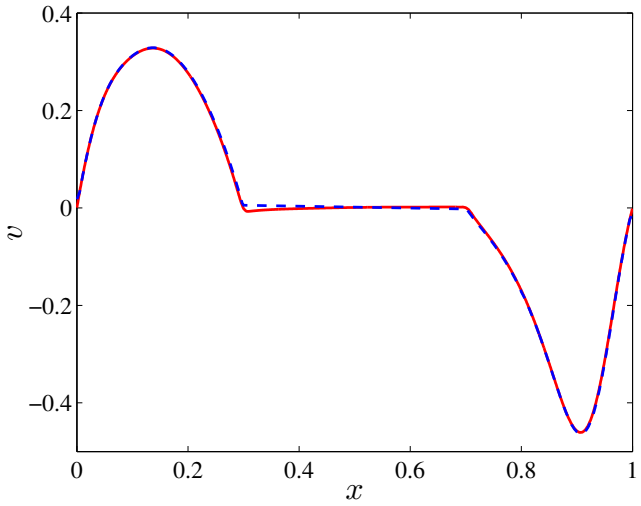


Figure 4.15: Vorticity contours and streamlines of the lid-driven cavity flow with a cylinder at $Re = 1000$, where the vorticity contour value is varied from -3 (blue) to 3 (red) with an increment of 0.4. Results of present MIBM are listed on the left; Results of the body-conforming mesh method are on the right.



(a)



(b)

Figure 4.16: Comparison of velocity profiles of the lid-driven cavity flow with a cylinder at $Re = 1000$: (a) Distribution of velocity component u along $x = 0.5$; (b) Distribution of velocity component v along $y = 0.5$. "- - - -", result of body-conforming mesh method; "—", result of present MIBM.

	(x_1, y_1)	(x_2, y_2)	(x_3, y_3)
Present	$(0.6942, 0.6881)$	$(0.0789, 0.0720)$	$(0.8852, 0.1063)$
Body-conforming mesh method	$(0.6906, 0.6872)$	$(0.0791, 0.0721)$	$(0.8849, 0.1063)$

Table 4.3: Comparison of vortices center positions for the proposed immersed boundary method and the body-conforming mesh method, where (x_1, y_1), (x_2, y_2), (x_3, y_3) represent the centers at the upper right to the cylinder, at the lower left corner and at the lower right corner respectively.

4.7.3 Flow over a stationary circular cylinder

The flow past a stationary circular cylinder is considered as a canonical test case to validate current method, since a great amount of experimental and numerical studies at different Reynolds numbers are available for comparison. The flow characteristics depend on the Reynolds number $Re = u_\infty D/\nu$, based on the inflow velocity u_∞, the cylinder diameter $D = 1$ and the fluid kinematic viscosity ν.

The simulation is performed in a rectangular domain, where the fluid flows from the left to the right (see Figure 4.17). At left boundary, a uniform velocity of $u_\infty = 1$ is imposed; The free slip boundary conditions are applied at lateral boundaries; At outlet, the convective boundary condition $\partial \mathbf{u}/\partial t + u_\infty \partial \mathbf{u}/\partial x = 0$ is employed for reducing the reflection effects because of the finite artificially truncated domain. The cylinder is placed at the center of the computational domain. The fluid domain is covered with a uniform mesh, and the cylinder surface is represented by a set of uniformly distributed Lagrangian points with $\delta s \approx h$. For comparison the drag and lift coefficients are defined as

$$C_D = \frac{F_D}{\frac{1}{2}\rho u_\infty^2 D}, \qquad C_L = \frac{F_L}{\frac{1}{2}\rho u_\infty^2 D} \tag{4.49}$$

where the fluid density ρ is set to 1 here. F_D, F_L are the drag and lift forces on the cylinder exerted by the fluid, calculated by

$$\begin{pmatrix} F_D \\ F_L \end{pmatrix} = -\sum_{l=1}^{n_b} \mathbf{F}(\mathbf{X}_l) \Delta V_l \tag{4.50}$$

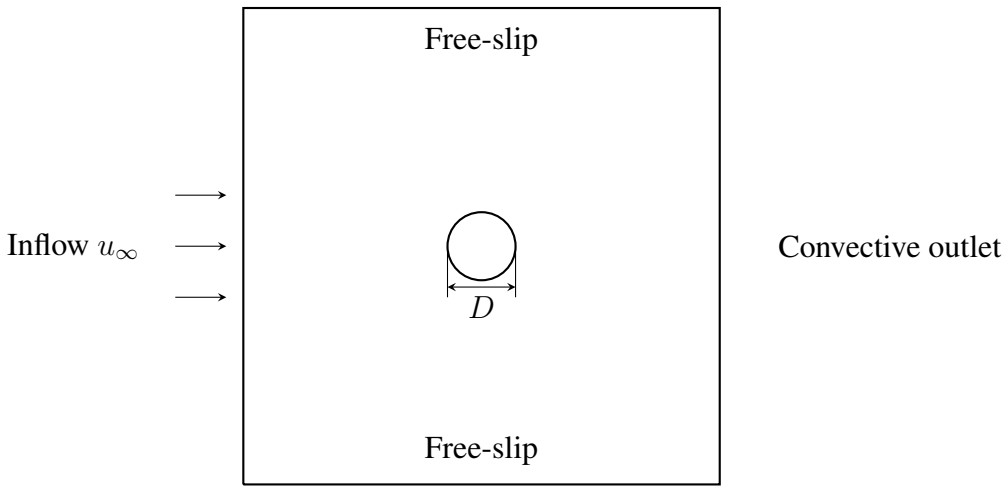


Figure 4.17: Sketch of the flow over a stationary circular cylinder.

(***i***) $Re = 30, 40$

The flow presents a steady state at low Reynolds numbers $Re = 30, 40$ with a recirculating region in the wake of the cylinder. The wake dimensions are described by the length of the wake l, the stream-wise distance a from the vortex center to the nearest point at the cylinder surface, the cross-wise distance b between two vortices centers, and the angle θ of flow separation, as shown in Figure 4.18. The computations are performed under different mesh resolutions to check the grid sensitivity. Various domain sizes are also considered to ensure that the boundary confinement effect does not influence the solution. We select the time step such that the CFL condition is satisfied, and the current method yields a stable solution even with a Courant number close to one. The 3-point-width discrete delta function ϕ_3 of Roma *et al.* (1999) is used for the interpolation and spreading.

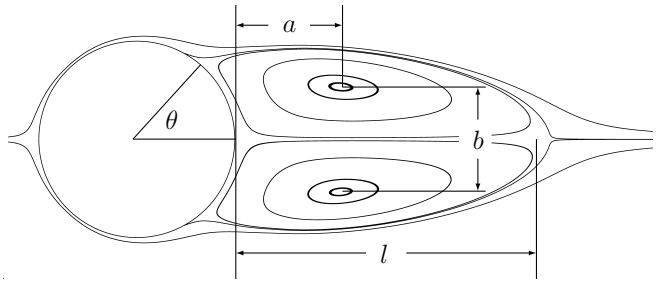
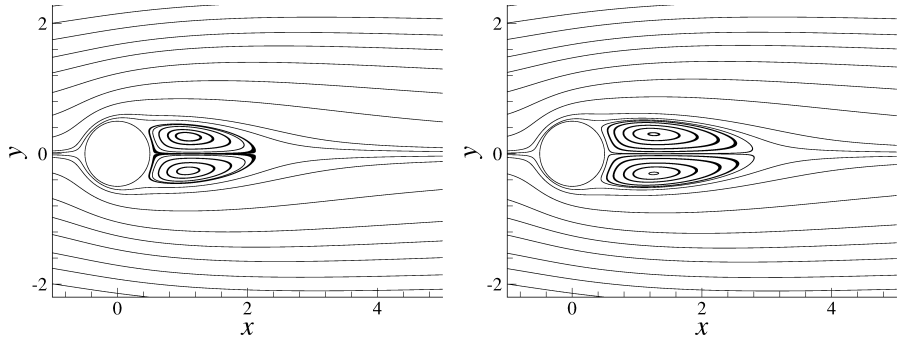


Figure 4.18: Definition of the characteristic wake dimensions for the steady flow over a stationary circular cylinder.

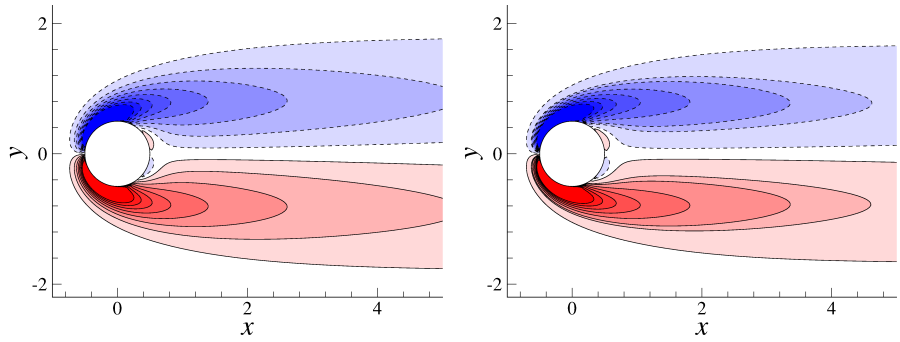
The streamlines, vorticity and pressure contours are shown in Figure 4.19, which are in close agreement with those reported in the literature. Table 4.4 compares the wakes dimensions and the drag coefficient against other numerical and experimental results. Good agreements have been obtained. It can be concluded from Table 4.4 that narrow domain size leads to a larger value of the drag coefficient, which was also observed in Uhlmann (2005), Lai & Peskin (2000) and Taira & Colonius (2007). For example at $Re = 30$ the drag coefficient for the domain $\Omega = 30D \times 30D$ is 3% higher than the value with the largest domain. By enlarging the domain size to $\Omega = 40D \times 40D$, the drag coefficient is reduced by 2%. This confinement effect is due to the finite distance of the lateral boundaries treated as slip walls. The time histories of the drag and lift coefficients are shown in Figure 4.20.

Figure 4.23 shows the distribution of wall vorticity ω_z and wall pressure coefficient C_P along the immersed cylinder surface at $Re = 40$. The wall pressure coefficient is defined as $C_P = (p - p_\infty)/(\frac{1}{2}\rho u_\infty^2)$, where p_∞ is the free-stream pressure.

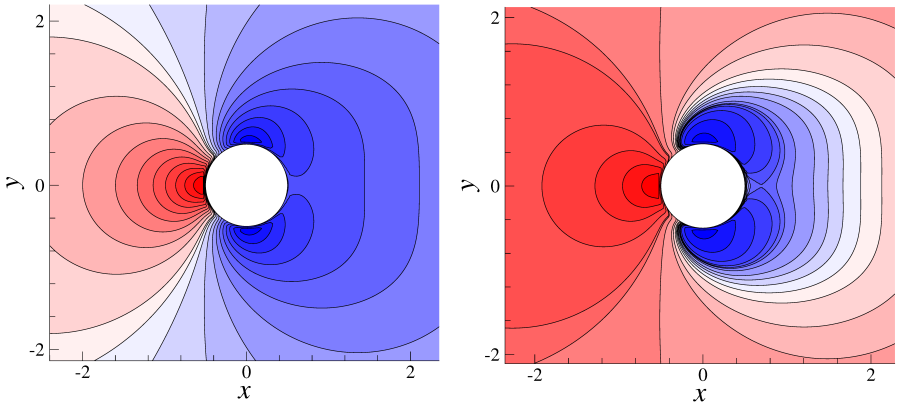
The results with present method is found to be satisfactory compared to the well-established work of Braza *et al.* (1986), especially for the stagnation and base region.



(a) Streamlines



(b) Vorticity

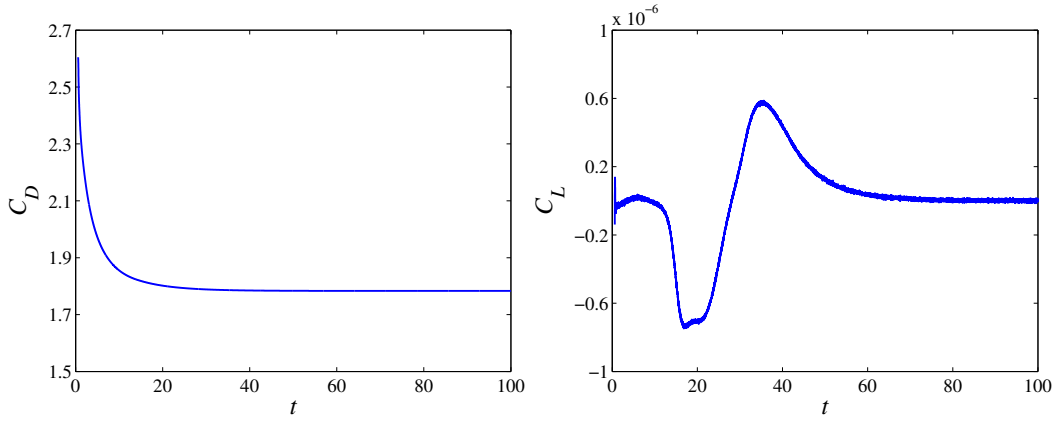


(c) Pressure

Figure 4.19: Streamlines, vorticity and pressure contours for the steady-state flow around a circular cylinder at $Re = 30$ (left) and $Re = 40$ (right). The contour level for the vorticity field are from -3 to 3 with an increment of 0.4, where dashed line represents the negative value.

		l/D	a/D	b/D	$\theta(^\circ)$	C_D
$Re = 30$	Coutanceau & Bouard (1977)†	1.55	0.54	0.54	50.0	-
	Tritton (1959)†	-	-	-	-	1.74
	Pinelli *et al.* (2010)	1.70	0.56	0.52	48.1	1.80
	Toja-Silva *et al.* (2014)	1.71	0.56	0.53	47.9	1.78
	Present ($\Omega = 30D \times 30D$, $h = 0.04D$)	1.66	0.58	0.52	45.0	1.78
	Present ($\Omega = 30D \times 30D$, $h = 0.029D$)	1.64	0.58	0.53	49.9	1.78
	Present ($\Omega = 30D \times 30D$, $h = 0.02D$)	1.64	0.58	0.52	46.5	1.78
	Present ($\Omega = 40D \times 40D$, $h = 0.029D$)	1.65	0.57	0.53	47.4	1.75
	Present ($\Omega = 60D \times 60D$, $h = 0.029D$)	1.64	0.57	0.53	49.8	1.73
$Re = 40$	Coutanceau & Bouard (1977)†	2.13	0.76	0.59	53.8	-
	Tritton (1959)†	-	-	-	-	1.59
	Wang & Zhang (2011)	2.36	0.72	0.6	53.8	1.54
	Taira & Colonius (2007)	2.30	0.73	0.60	53.7	1.54
	Present ($\Omega = 30D \times 30D$, $h = 0.04D$)	2.38	0.77	0.59	52.0	1.58
	Present ($\Omega = 30D \times 30D$, $h = 0.029D$)	2.34	0.76	0.62	54.5	1.58
	Present ($\Omega = 30D \times 30D$, $h = 0.02D$)	2.36	0.77	0.60	53.1	1.59
	Present ($\Omega = 40D \times 40D$, $h = 0.029D$)	2.36	0.75	0.62	52.1	1.56
	Present ($\Omega = 60D \times 60D$, $h = 0.029D$)	2.34	0.76	0.62	54.5	1.54

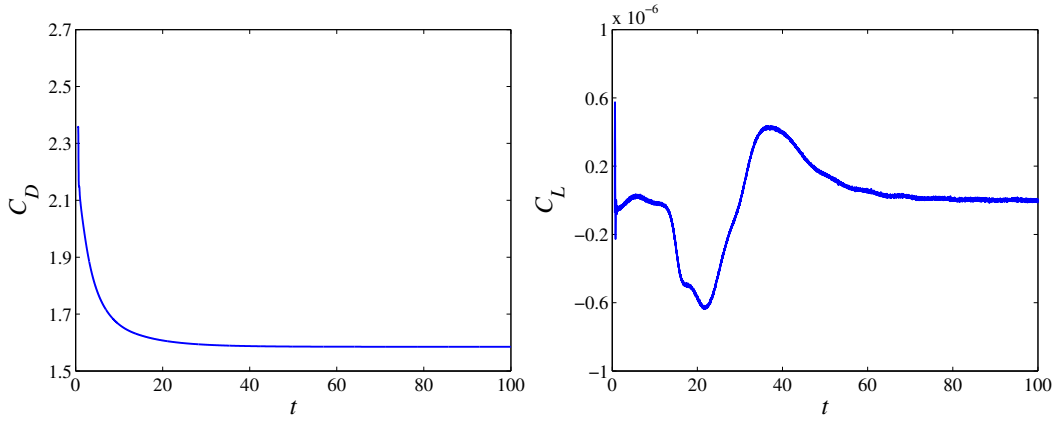
Table 4.4: Comparison of the steady-state wake dimensions and the drag coefficient for the flow over a stationary cylinder at $Re = 30, 40$. The experimental results are marked with (†). ϕ_3 is used for interpolation and spreading.



(a) $Re = 30$

(ii) $Re = 100, 200$

Increasing the Reynolds number to $Re = 100$ and 200, the flow becomes unsteady and periodic shedding of vortices is found. The well-known von Kármán vortex



(b) $Re = 40$

Figure 4.20: Drag and lift coefficients versus time for flow over a stationary cylinder at (a) $Re = 30$ and (b) $Re = 40$.

street is shown in figure 4.21. Figure 4.22 shows the corresponding pressure field. The time evolution of the drag and the lift coefficients at $Re = 100$ and $Re = 200$ are plotted in figure 4.24. It should be pointed out that the oscillating frequency of the drag is twice that of the lift, which is in fact the vortex shedding frequency f_s, which has also been noticed in Lai & Peskin (2000). The Strouhal number $St = Df_s/u_\infty$ as well as the coefficients of drag, lift are summarized in Table 4.5. For comparison, we list the well established experimental results of Williamson (1989) and the numerical results with the body-fitted mesh methods of Braza *et al.* (1986) and Liu *et al.* (1998). Results with other IBM variants are also included, e.g. the explicit direct forcing immersed boundary method of Uhlmann (2005), the vortex penalization method of Mimeau *et al.* (2015), the immersed interface method of Xu & Wang (2006), the iterative direct forcing immersed boundary method of Ji *et al.* (2012) and the immersed boundary projection method of Taira & Colonius (2007). Good agreement has been obtained towards the flow quantities.

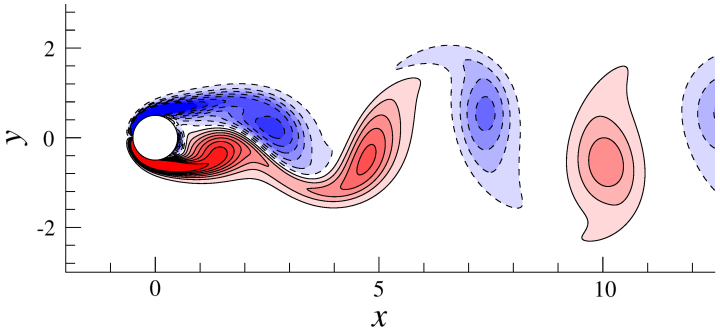
From Table 4.5 we can see that the current method yields an over-prediction of the mean drag coefficient only with 2% error (compared to Liu *et al.* (1998)), while Uhlmann (2005) over-predicted the mean drag coefficient value by approximately 11%. In this case we consider the same parameters as used in Uhlmann (2005), namely the computational domain $\Omega = 30D \times 30D$ with the mesh resolution $h = 0.029D$. This improved accuracy can be attributed to the exact imposition of the no-slip boundary condition at the interface in current method. This error is further reduced to 1% when we use an enlarged domain of $\Omega = 40D \times 40D$ while a relative large error of 8% is still found in Uhlmann (2005).

Compared to the vortex penalization method of Mimeau *et al.* (2015) and the im-

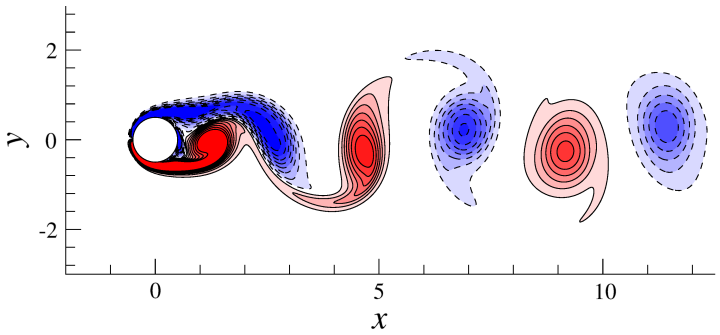
mersed interface method of Xu & Wang (2006), our implementation does not introduce artificial constants and thus is much suitable for flow with rigid bodies. Our results are very close to those of the iterative direct forcing immersed boundary method of Ji *et al.* (2012) and the immersed boundary projection method of Taira & Colonius (2007). However, our method is non-iterative compared to Ji *et al.* (2012). The original system is unchanged and only a small system is solved additionally at each time step. Therefore the current method can be more efficient than Taira & Colonius (2007).

The time-averaged values of the wall vorticity ω_z and the wall pressure coefficient C_P are shown in Figure 4.23 for $Re = 100$. Good agreements have been found compared to the results of Braza *et al.* (1986).

The effects of different discrete delta functions on the results are also tested in Table 4.6 for $Re = 100$, 200. The domain of $\Omega = 30D \times 30D$ is used and the mesh resolution is set to $h = 0.029D$.

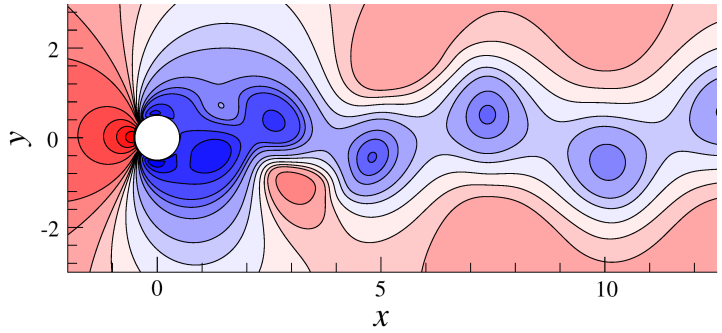


(a) $Re = 100$

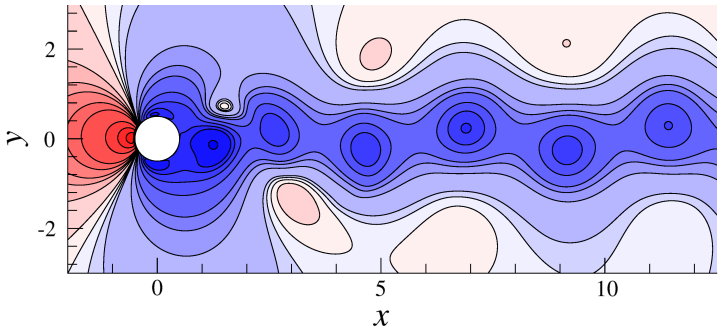


(b) $Re = 200$

Figure 4.21: Instantaneous vorticity contours of flow over a circular cylinder at (a) $Re = 100$ and (b) $Re = 200$, where the contour level is set from -3 to 3 with an increment of 0.4.

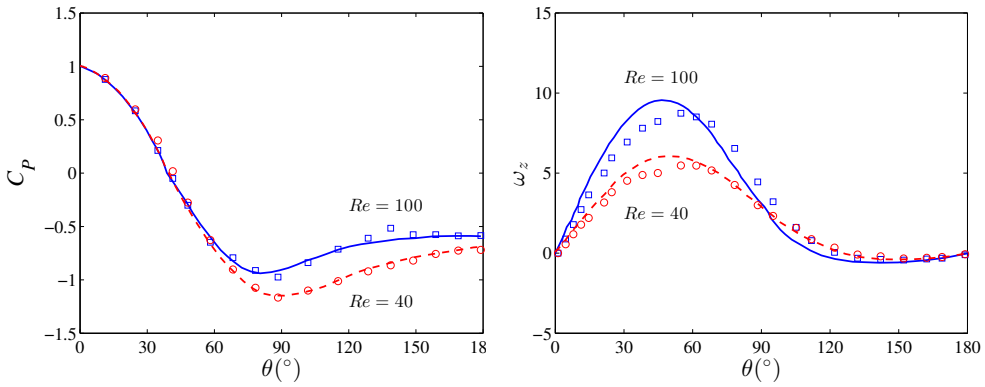


(a) $Re = 100$



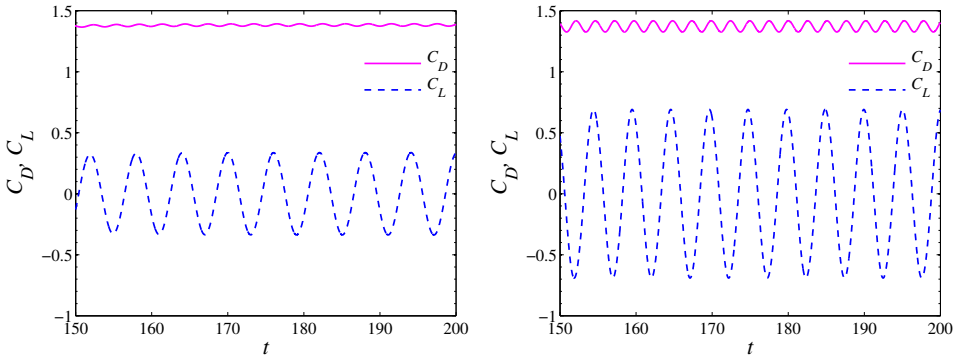
(b) $Re = 200$

Figure 4.22: Instantaneous pressure contours of flow over a circular cylinder at (a) $Re = 100$ and (b) $Re = 200$.



(a) (b)

Figure 4.23: Flow variables on the immersed cylinder surface at $Re = 40$ and $Re = 100$ as a function of the angle θ: (a) wall pressure coefficient C_P; (b) wall vorticity ω_z, where $\theta = 0^\circ$ and $\theta = 180^\circ$ correspond to the stagnation point and the base point, respectively. The values of C_P and ω_z at $Re = 100$ are time-averaged. The lines of "——" and "- - - -" represent the results from Braza *et al.* (1986), and present results are marked with "□" and "○".



(a) (b)

Figure 4.24: Time evolution of drag and lift coefficients for the flow over a stationary cylinder problem at (a) $Re = 100$ and (b) $Re = 200$.

		$\overline{C}_D$	C'_D	C'_L	St
$Re = 100$	Williamson (1989)†	-	-	-	0.164
	Uhlmann (2005)	1.453	$\pm$0.011	$\pm$0.339	0.169
	Ji *et al.* (2012)	1.376	$\pm$0.010	$\pm$0.339	0.169
	Braza *et al.* (1986)	1.359	$\pm$0.019	$\pm$0.293	0.16
	Liu *et al.* (1998)	1.350	$\pm$0.012	$\pm$0.339	0.165
	Mimeau *et al.* (2015)	1.40	$\pm$0.010	$\pm$0.32	0.165
	Xu & Wang (2006)	1.423	$\pm$0.013	$\pm$0.34	0.171
	Present ($\Omega = 30D \times 30D$, $h = 0.04D$)	1.380	$\pm$0.010	$\pm$0.343	0.166
	Present ($\Omega = 30D \times 30D$, $h = 0.029D$)	1.377	$\pm$0.010	$\pm$0.337	0.166
	Present ($\Omega = 30D \times 30D$, $h = 0.02D$)	1.379	$\pm$0.010	$\pm$0.346	0.166
	Present ($\Omega = 40D \times 40D$, $h = 0.029D$)	1.366	$\pm$0.010	$\pm$0.342	0.166
	Present ($\Omega = 60D \times 60D$, $h = 0.029D$)	1.353	$\pm$0.010	$\pm$0.335	0.166
$Re = 200$	Williamson (1989)†	-	-	-	0.197
	Taira & Colonius (2007)	1.35	$\pm$0.048	$\pm$0.68	0.196
	Ji *et al.* (2012)	1.354	$\pm$0.044	$\pm$0.682	0.20
	Braza *et al.* (1986)	1.386	$\pm$0.040	$\pm$0.766	0.20
	Liu *et al.* (1998)	1.31	$\pm$0.049	$\pm$0.69	0.192
	Mimeau *et al.* (2015)	1.44	$\pm$0.05	$\pm$0.75	0.200
	Xu & Wang (2006)	1.42	$\pm$0.04	$\pm$0.66	0.202
	Present ($\Omega = 30D \times 30D$, $h = 0.04D$)	1.355	$\pm$0.042	$\pm$0.677	0.200
	Present ($\Omega = 30D \times 30D$, $h = 0.029D$)	1.365	$\pm$0.044	$\pm$0.696	0.200
	Present ($\Omega = 30D \times 30D$, $h = 0.02D$)	1.374	$\pm$0.046	$\pm$0.705	0.200
	Present ($\Omega = 40D \times 40D$, $h = 0.029D$)	1.358	$\pm$0.044	$\pm$0.682	0.200
	Present ($\Omega = 60D \times 60D$, $h = 0.029D$)	1.345	$\pm$0.043	$\pm$0.682	0.200

Table 4.5: Comparison of the drag, lift coefficients and the Strouhal number for the flow around a stationary cylinder at $Re = 100, 200$. The experimental results are marked with (†). ϕ_3 is used for interpolation and spreading.

		$\overline{C}_D$	C'_D	C'_L	St
$Re = 100$	ϕ_1	1.388	± 0.010	± 0.346	0.166
	ϕ_2	1.390	± 0.011	± 0.345	0.164
	ϕ_3	1.377	± 0.010	± 0.339	0.166
	ϕ_4	1.379	± 0.011	± 0.343	0.166
	ϕ_5	1.379	± 0.010	± 0.341	0.166
	ϕ_6	1.379	± 0.011	± 0.343	0.166
	ϕ_7	1.378	± 0.010	± 0.341	0.166
	ϕ_8	1.378	± 0.011	± 0.342	0.166
	ϕ_9	1.388	± 0.013	± 0.344	0.164
	ϕ_{10}	1.362	± 0.010	± 0.324	0.170
	ϕ_{11}	1.383	± 0.010	± 0.343	0.168
	ϕ_{12}	1.387	± 0.011	± 0.365	0.166
$Re = 200$	ϕ_1	1.391	± 0.047	± 0.709	0.198
	ϕ_2	1.387	± 0.047	± 0.702	0.193
	ϕ_3	1.365	± 0.044	± 0.696	0.200
	ϕ_4	1.358	± 0.045	± 0.688	0.195
	ϕ_5	1.370	± 0.045	± 0.693	0.198
	ϕ_6	1.355	± 0.045	± 0.686	0.195
	ϕ_7	1.362	± 0.045	± 0.688	0.195
	ϕ_8	1.354	± 0.045	± 0.685	0.195
	ϕ_9	1.365	± 0.046	± 0.685	0.193
	ϕ_{10}	1.382	± 0.046	± 0.692	0.202
	ϕ_{11}	1.390	± 0.047	± 0.710	0.198
	ϕ_{12}	1.404	± 0.051	± 0.730	0.195

Table 4.6: Effects of different discrete delta functions on the drag, lift coefficients and the Strouhal number for the flow around a stationary cylinder at $Re = 100$ and 200.

(***iii***) $Re = 1000$

We further extend our method to higher Reynolds number flow $Re = 1000$. At this regime, the convection effects become predominant and the boundary layer thickness decreases, which is estimated by $\delta \approx D/\sqrt{Re} = 0.032$. To capture the thin boundary layer, a fine grid resolution of $h = 0.01D$ is taken, as also used in Mittal *et al.* (2008) and Apte *et al.* (2009). To compare with Mittal *et al.* (2008), the computational domain is chosen to be $[-20D, 20D] \times [-20D, 20D]$. The 2-point-width hat function ϕ_1 is employed for the interpolation and spreading, as it provides the narrowest support domain and thus the force is less diffused. Note that the grid resolution is only marginal for resolving the boundary layer at this Reynolds number. Nevertheless, the results are satisfactory and the essential features of the

flow are well captured. Figure 4.25 shows the instantaneous vorticity field. The the drag and lift coefficients are plotted in Figure 4.26 and compared in Table 4.7, showing a good agreement.

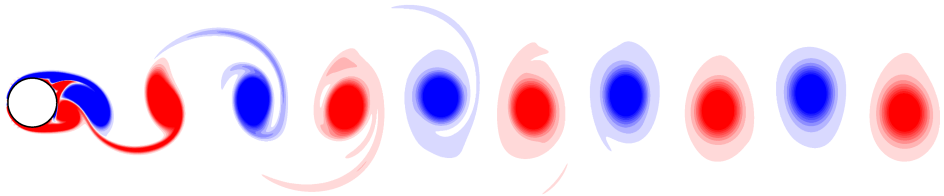


Figure 4.25: Instantaneous vorticity field of flow over a stationary cylinder at $Re = 1000$.

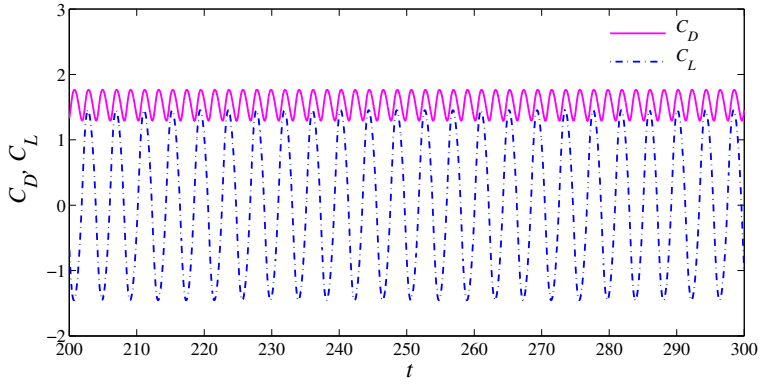


Figure 4.26: Time evolution of drag and lift coefficients for the flow over a stationary cylinder problem at $Re = 1000$.

		$\overline{C}_D$	C'_D	C'_L	St
$Re = 1000$	Mittal & Kumar (2001)	1.48	$\pm$0.21	$\pm$1.65	0.250
	Apte *et al.* (2009)	1.50	-	-	0.238
	Mittal *et al.* (2008)	1.48	-	-	-
	Mimeau *et al.* (2015)	1.51	$\pm$0.23	$\pm$1.54	0.245
	Present	1.55	$\pm$0.22	$\pm$1.46	0.240

Table 4.7: Comparison of the drag, lift coefficients and the Strouhal number for the flow around a stationary cylinder at $Re = 1000$. ϕ_1 is used for interpolation and spreading.

4.7.4 In-line oscillating circular cylinder in a fluid at rest

We consider the flow induced by an oscillating circular cylinder as another test, in order to demonstrate the ability of our method for handling moving boundaries. The motion of the cylinder is described by a simple harmonic oscillation as follows

$$x(t) = -A\,sin(2\pi ft) \tag{4.51}$$

where $x(t)$ is the streamwise location of the cylinder center. A and f are the amplitude and the frequency of oscillation, respectively. Two key parameters determine the flow characteristics: the Reynolds number $Re = U_{max}D/\nu$ and the Keulegan–Carpenter number $KC = U_{max}/fD$, where U_{max} is the maximum velocity of the oscillating cylinder, ν is the kinematic viscosity and D is the diameter of the cylinder. The KC number can be related to the Strouhal number by $KC = 1/St$. In the present study, Re is set to 100 and KC is set to 5, corresponding to the LDA experiments and the numerical simulations of Dütsch *et al.* (1998).

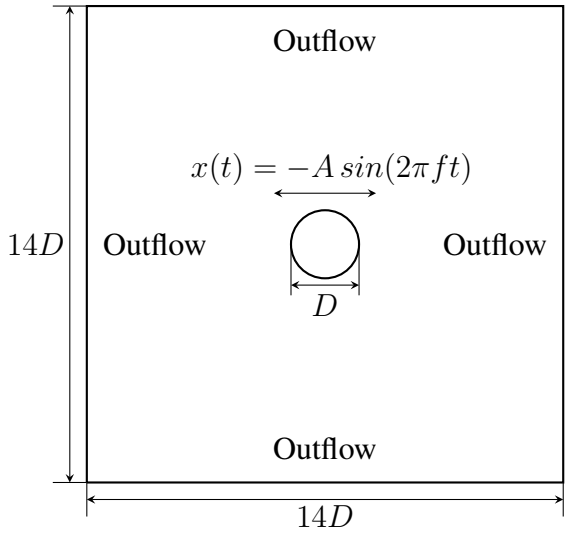
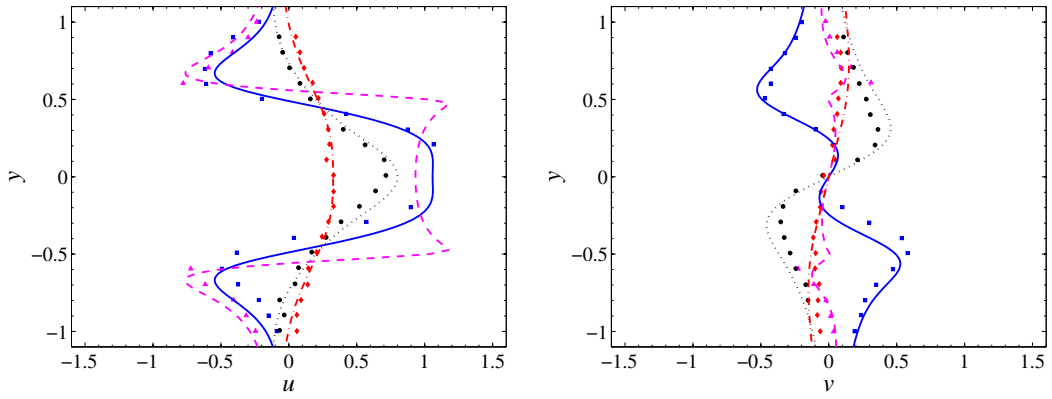


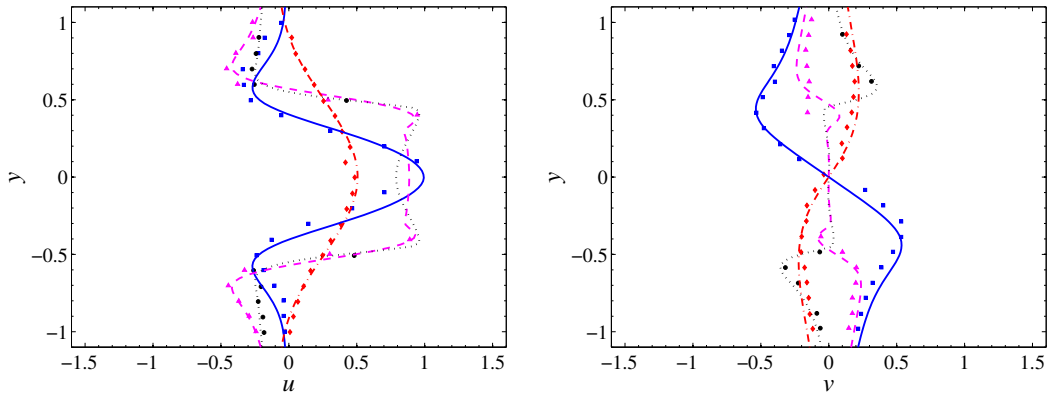
Figure 4.27: Sketch of the oscillating circular cylinder in a fluid at rest.

The computational domain is chosen to be $14D \times 14D$, as shown in Figure 4.27. The cylinder is initially located at the center of the computational domain. The outflow boundary condition $\partial\mathbf{u}/\partial\mathbf{n} = 0$ is applied at the domain contours. A uniform mesh of 560×560 is adopted for the fluid domain and the cylinder is represented by 126 points, such that $\delta_s \approx h$. The transient no-slip velocity boundary condition at the cylinder surface is enforced by present MIBM at each time level

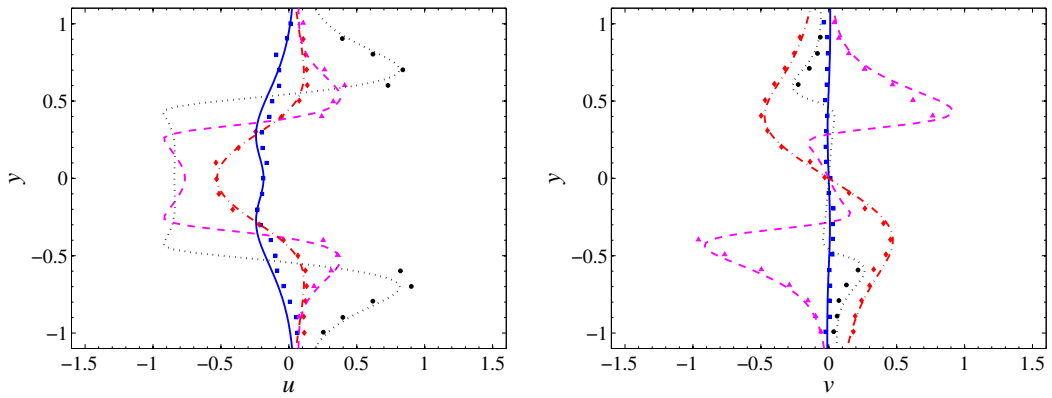
$$u(t) = -2\pi fA\,cos(2\pi ft) \tag{4.52}$$



(a) $\phi = 180°$

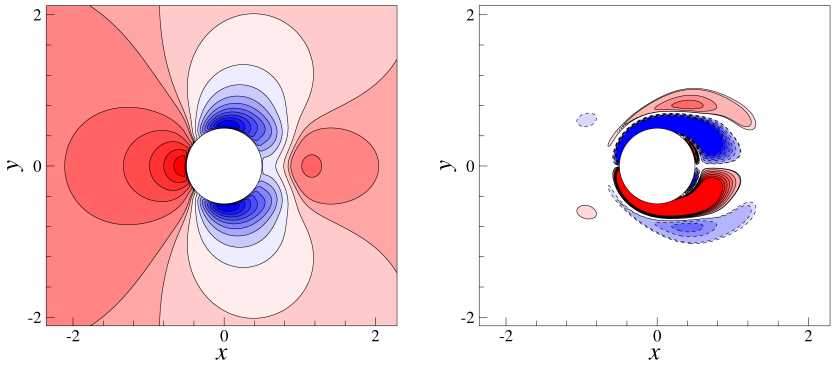


(b) $\phi = 210°$

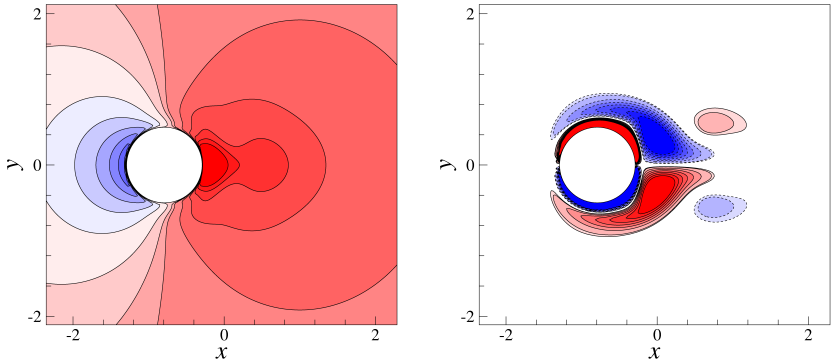


(c) $\phi = 330°$

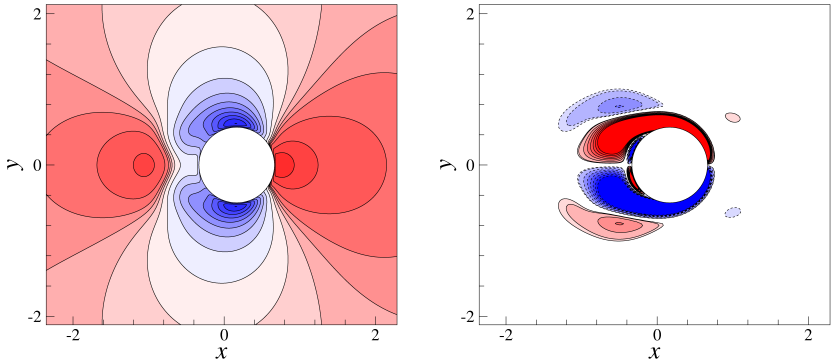
Figure 4.28: Comparison of the velocity profiles u (left) and v (right) at four different cross-sections and three phase positions: (a) $\phi = 180°$, (b) $\phi = 210°$, (c) $\phi = 330°$. The experimental results of Dütsch *et al.* (1998) are marked with "■" at $x = -0.6D$, "▲" at $x = 0D$, "●" at $x = 0.6D$, "♦" at $x = 1.2D$. The present results are represented by "——" at $x = -0.6D$, "- - - -" at $x = 0D$, "· · · ·" at $x = 0.6D$, "— · —·" at $x = 1.2D$.



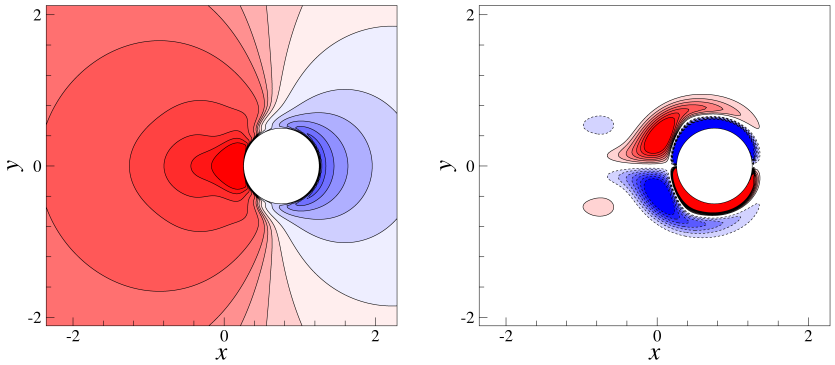
(a) $\phi = 0°$



(b) $\phi = 96°$



(c) $\phi = 192°$



(d) $\phi = 288°$

Figure 4.29: Pressure and vorticity contours at four different phases: (a) $\phi = 0°$; (b) $\phi = 96°$; (c) $\phi = 192°$; (d) $\phi = 288°$.

Figure 4.28 shows the profiles of the velocity components u and v at four different stream-wise locations ($x = -0.6D$, $0D$, $0.6D$, $1.2D$) for three phase ($\phi = 2\pi ft = 180°$, $210°$, $330°$). The experimental results of Dütsch *et al.* (1998) by LDA measurements are also plotted for comparison. The velocity profiles outside the cylinder agree well those of Dütsch *et al.* (1998). The only discrepancy is the velocity inside the cylinder. Since the present IBM treats the solid domain as fluid, the velocity is non-zero inside the cylinder. From Figure 4.28 we can see that this treatment, however, does not influence the flow field outside the solid. Various internal treatments of the body have been discussed in the work of Iaccarino & Verzicco (2003), such as applying the force inside the body and thus changing the velocity distribution. Iaccarino & Verzicco (2003) also concluded that for direct forcing IBM, there is essentially no difference. Therefore, for simple implementation we just leave the interior of the solid free to develop a flow without imposing anything.

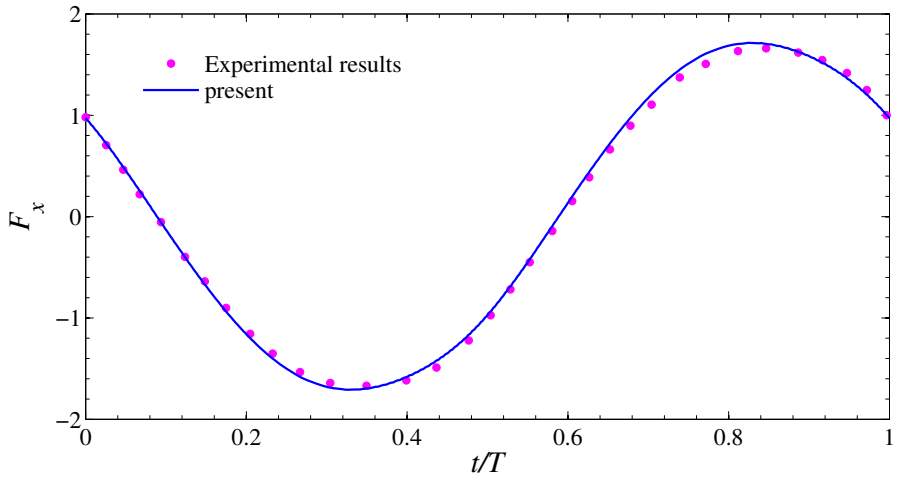


Figure 4.30: Comparison of the in-line force F_x in a period at $Re = 100$, $KC = 5$. The experimental results are taken from Dütsch *et al.* (1998).

The pressure and vorticity contours at four different phases ($\phi = 2\pi ft = 0°$, $96°$, $192°$, $288°$) are shown in Figure 4.29, where two counter-rotating vortices are formulated during the oscillation. The vortices contours are drawn from -3 to 3 with an increment of 0.4, which display the same structure as in Dütsch *et al.* (1998). The time history of the in-line force F_x in the oscillatory direction is shown in one period in Figure 4.30, which agrees very well with the experimental results of Dütsch *et al.* (1998). It should be noted that no spurious oscillation has been found towards the in-line force in this case.

4.7.5 Transverse oscillation of a circular cylinder in a free-stream

Next, we consider the transversely oscillating cylinder subject to a free-stream, as shown in Figure 4.31. This case has been extensively studied in the development of immersed boundary methods (e.g. Uhlmann, 2005 and Yang *et al.*, 2009), as it provides a good test to examine the effects of the discrete delta functions on the boundary force in moving boundary situations. The cylinder moves in the cross-wise direction in a sinusoidal fashion as

$$y(t) = A\,sin(2\pi ft) \tag{4.53}$$

where $y(t)$ is the cross-wise position of the cylinder center. The amplitude is set to $A = 0.2D$ with $D = 1$ being the cylinder diameter. The Reynolds number is set to $Re = 185$ based on the cylinder diameter, in order to compare with Uhlmann (2005) and Yang *et al.* (2009). The frequencies of $f = 0.8f_s$ and $1.0f_s$ are considered in this test, where f_s is the natural shedding frequency for the stationary cylinder ($f_s = 0.195$ for $Re = 185$).

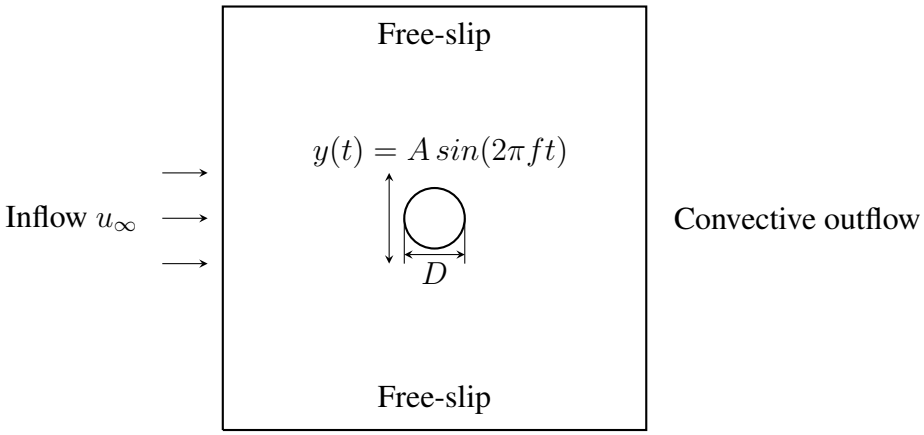
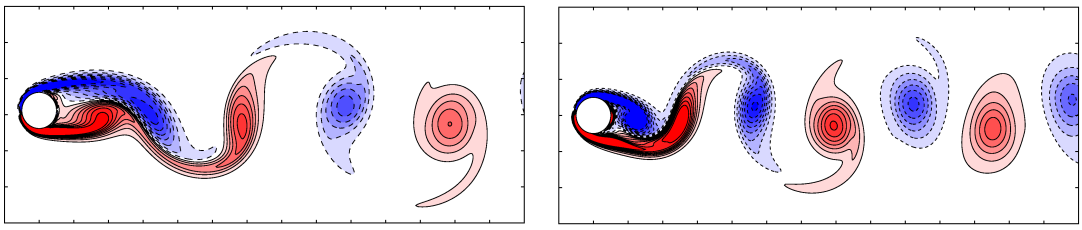


Figure 4.31: Sketch of the transversely oscillating circular cylinder in free-stream.

The computational domain is $[-15D, 15D] \times [-15D, 15D]$ covered with a uniform mesh of 1024×1024. The resulting mesh resolution is $0.029D$, which is based on the mesh independence study of the stationary cylinder case. Around 108 points are used for describing the cylinder surface such that $\delta_s \approx h$. The boundary condition at the inlet is set to be uniform ($u = u_\infty = 1, v = 0$). The free slip boundary condition ($\partial u/\partial y = 0, v = 0$) is applied at lateral boundaries. The convective boundary condition ($\partial \mathbf{u}/\partial t + u_\infty \partial \mathbf{u}/\partial x = 0$) is specified at outlet to allow vorticity to freely exit the flow domain.

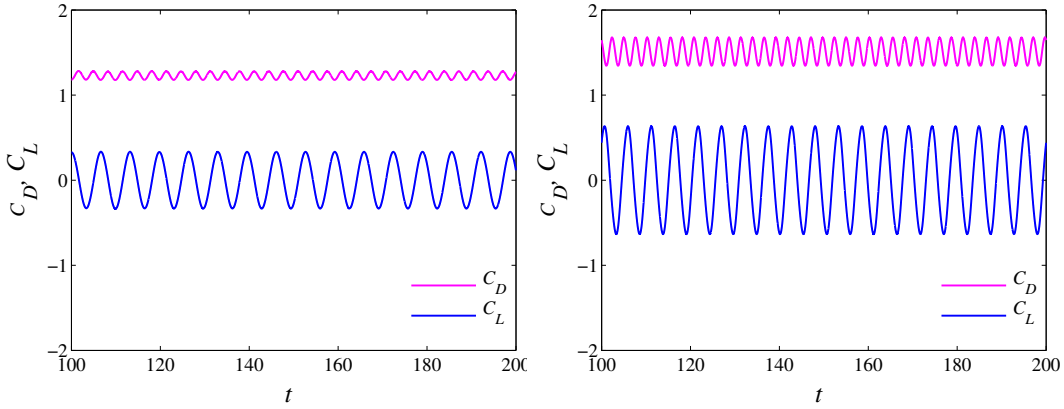
The instantaneous vorticity fields are shown in Figure 4.32, showing a periodic vortex shedding. The time histories of the drag and the lift coefficients are plotted

in Figure 4.33. Table 4.8 compares the mean, rms drag and lift coefficients with the literature. Good agreement has been found.



(a) (b)

Figure 4.32: Instantaneous vorticity fields for the transversely oscillating circular cylinder problem at $Re = 185$ for (a) $f/f_0 = 0.8$ and (b) $f/f_0 = 1.0$.



(a) (b)

Figure 4.33: Time history of the drag and the lift coefficients of the transversely oscillating circular cylinder. (a) $f/f_0 = 0.8$ and (b) $f/f_0 = 1.0$.

The drag and lift coefficients in one period are plotted in Fig 4.34 as a function of the cylinder's position. With the discrete delta function ϕ_4 of 4-point width of Peskin (2002), oscillations have been observed towards the drag and lift coefficients, which is also reported in Uhlmann (2005) and Yang *et al.* (2009). These oscillations reduce as the mesh number increases. By employing a smoothed version ϕ_8 of Yang *et al.* (2009), the oscillations are significantly diminished with the same mesh resolution. Hence we confirm the conclusion of Yang *et al.* (2009) that increasing the moment condition of the discrete delta function can reduce this spurious oscillation.

		$\overline{C}_D$	$(C_D^{'})_{rms}$	$(C_L^{'})_{rms}$
$f/f_s = 0.8$	**Present**	1.229	0.036	0.235
	Guilmineau & Queutey (2002)	1.194	0.038	0.074
	Kim & Choi (2006)	1.235	0.037	0.068
	Uhlmann (2005)	1.380	-	0.176
	Yang *et al.* (2009)	1.290	0.043	0.070
$f/f_s = 1.0$	**Present**	1.511	0.117	0.442
	Guilmineau & Queutey (2002)	1.506	0.134	0.420
	Kim & Choi (2006)	1.537	0.140	0.376

Table 4.8: Comparison of the mean, rms drag and lift coefficients for the cylinder oscillating transversely in a free-stream.

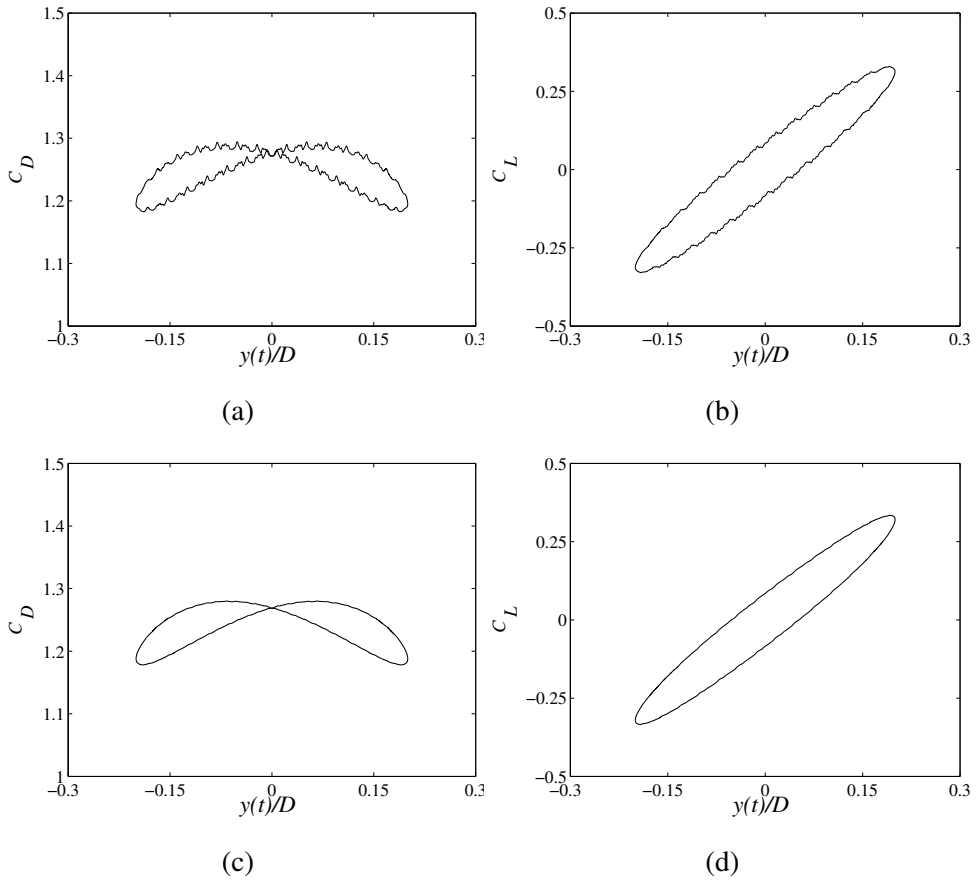


Figure 4.34: Influences of different discrete delta functions on the drag and lift coefficients, for the transversely oscillating cylinder in a uniform cross-flow problem at $Re = 185$ with $f/f_0 = 0.8$. (a) and (b) use the 4-point discrete delta function ϕ_4 of Peskin (2002); (c) and (d) use the corresponding smoothed version ϕ_8 proposed by Yang *et al.* (2009).

4.7.6 Flow around a flapping wing

In this example, we investigate the flow induced by a flapping wing, in order to demonstrate the ability of current method for handling non-circular object in both translational and rotational motions. The configuration of this problem is shown in Figure 4.35. The hovering wing is a geometrical 2D ellipse with major axis c (chord length) and minor axis b. The aspect ratio is defined as $e = c/b$. The wing is initially located at the origin with an angle of attack of θ_0, then shifts along a stroke plane inclined at an angle β. The translational and rotational motions of the hovering wing are described as follows

$$A(t) = \frac{A_0}{2}\left[cos(\frac{2\pi t}{T}) + 1\right] \tag{4.54}$$

$$\theta(t) = \theta_0\left[1 - sin(\frac{2\pi t}{T} + \phi_0)\right] \tag{4.55}$$

where A_0 is the translational amplitude, $2\theta_0$ the rotational amplitude, T the flapping period and ϕ_0 the phase difference. The chord length c and the maximum velocity $U_{\max} = \pi A_0/T$ along the flapping path are used as the length and the velocity scales, respectively. The Reynolds number is defined as $Re = U_{\max}c/\nu$. We employ the same parameters as used in Wang (2000), Xu & Wang (2006) and Yang & Stern (2012): $c = 1, e = 4$, $A_0 = 2.5c$, $\theta_0 = \pi/4$, $T = \pi A_0/c$, $\beta = \pi/3$, $\phi_0 = 0$, $Re = 157$.

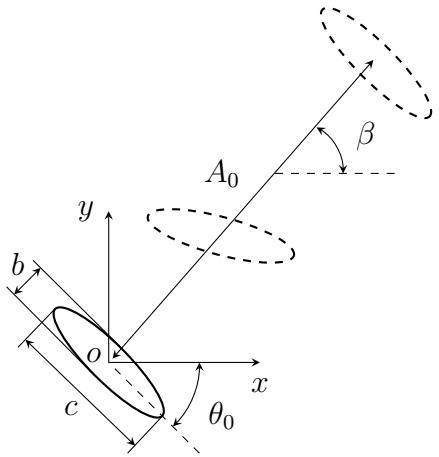


Figure 4.35: Configuration for flow over a flapping wing.

As suggested by Yang & Stern (2012), this simulation is performed on a large square domain of $[-24c, 24c] \times [-24c, 24c]$ to obtain a better periodicity for the results. A uniform mesh of 2400×2400 is employed to cover the computational domain and the mesh spacing around the wing is $0.02c$. A larger time step is selected in the

present study ($\Delta t = 0.01$) based on the CFL number ($\text{CFL}_{\max} = 0.72$), while $\Delta t = 0.001$ is used in the immersed interface method of Xu & Wang (2006) to reduce the body shape distortion. The time-dependent no-slip wall conditions on the wing surfaces are enforced by present immersed boundary method. The smoothed discrete delta function ϕ_7 is used for interpolation and spreading.

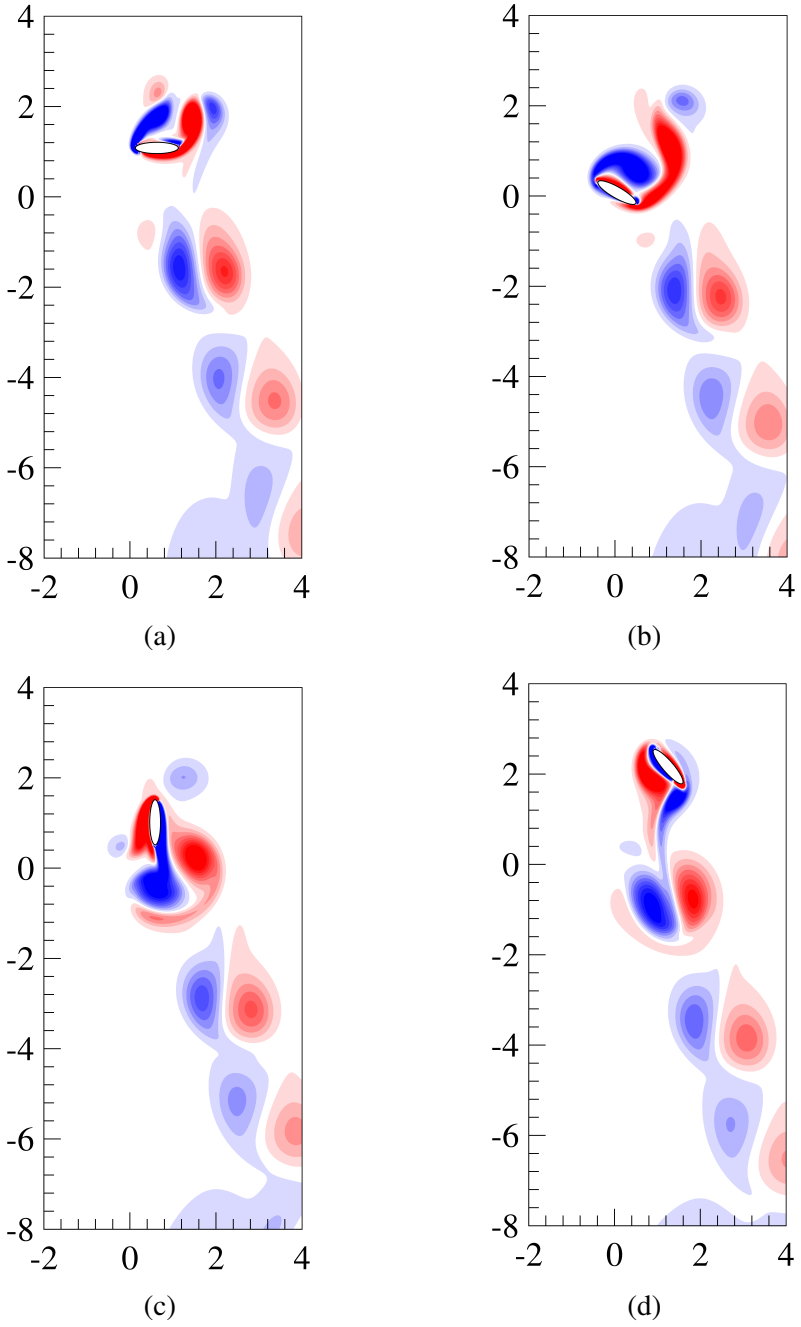
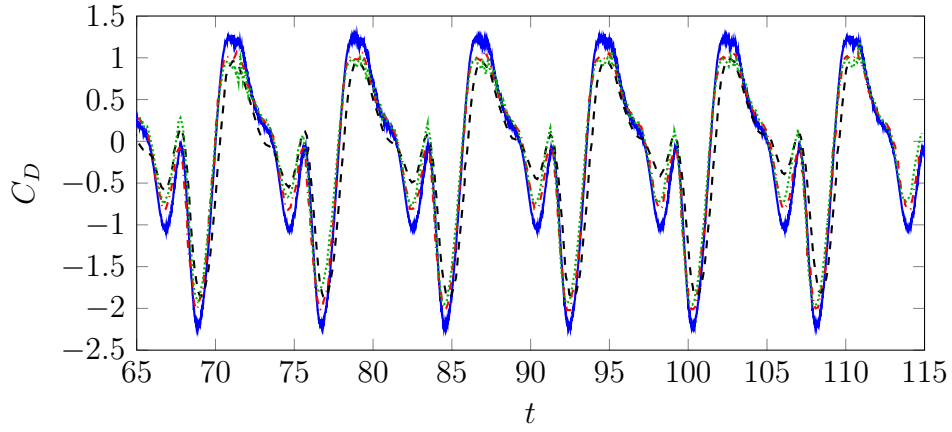
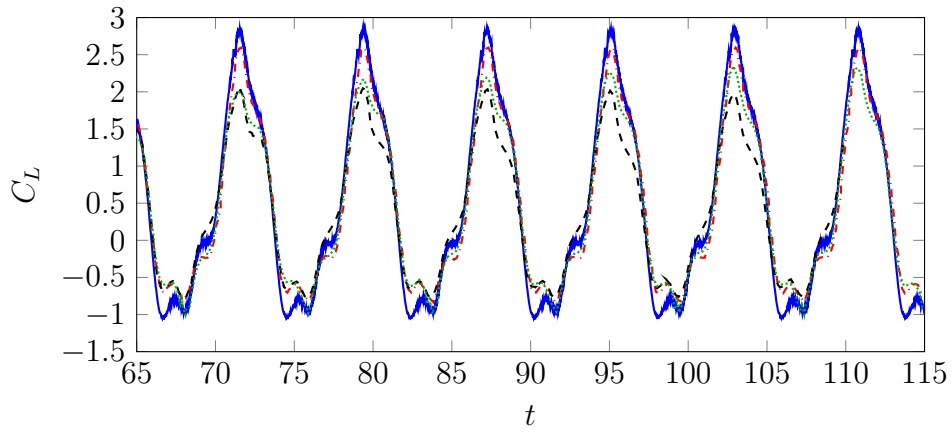


Figure 4.36: Snapshots of the vorticity fields around a flapping wing at $Re = 157$ for four different positions: (a) $t = 0.25T$; (b) $t = 0.44T$; (c) $t = 0.74T$; (d) $t = 0.99T$.

Figure 4.36 shows the vorticity fields near the flapping wing in one flapping period



(a)



(b)

Figure 4.37: Time history of drag and lift coefficients for flow around a flapping wing at $Re = 157$. "—" for present method; "- · - ·" for the immersed boundary method of Yang & Stern (2012); "- - - -" for the body-conforming mesh method of Wang (2000); "· · · ·" for the immersed interface method of Xu & Wang (2006).

at four different positions, which are very similar to those given in Wang (2000), Xu & Wang (2006) and Yang & Stern (2012). A pair of leading and trailing edge vortices of opposite rotation is formed into a dipole. The dipole moves downward, generating the lift of the wing. The vortices shed from the wing by the self-induced flow, without interfering the new vortices in the next cycle.

The time history of the drag and lift coefficients are plotted in Figure 4.37 and compared to the results of Wang (2000) and Yang & Stern (2012). Good agreement has been found. Note that in order to maintain the shape of the rigid body in IIM of Xu & Wang (2006), a feedback control technique is employed and the time step is kept small to reduce the shape distortion. The present immersed boundary method is found to be much more satisfactory, since no additional springs for feedback control are needed and the no-slip boundary condition is exactly imposed at the interface.

4.8 Concluding remarks

In this chapter we have developed a novel implicit immersed boundary method, termed as the moving immersed boundary method (MIBM). The present MIBM is an extension of the direct forcing IBM of Uhlmann (2005) and Kempe & Fröhlich (2012a), the multidirect forcing IBM of Luo *et al.* (2007) and Breugem (2012), and the immersed boundary projection method (IBPM) of Taira & Colonius (2007). Inspired by the IBPM, we also consider the boundary force as a Lagrange multiplier in the MIBM to satisfy the no-slip boundary condition at the interface. However, the boundary force and the pressure are formulated into different linear systems and solved separately. This allows us to retain the efficiency of the direct forcing IBMs. Although our results are identical to the multidirect forcing IBM, the MIBM is non-iterative and much fast. No artificial constants and additional time step constraint are introduced compared to the continuous forcing immersed boundary variants.

In MIBM we have derived an additional moving force equation for the boundary force, to impose the no-slip boundary condition accurately. The coefficient matrix is constructed to be symmetric and positive-definite, such that the conjugate gradient method can solve it quickly. The size of the moving force coefficient matrix is usually much smaller than that of the fluid matrices. Hence the MIBM is very suitable for moving boundary problems. The overall method still follows the traditional fractional step algorithm, namely the prediction, the immersed boundary forcing and the projection. This allows us to integrate the MIBM into any fluid solver easily as a plug-in.

The MIBM has been well validated with a series of numerical examples with prescribed solid motions. The spatial accuracy of current method is found to be second order accurate for smooth fields, which is in accord with the theory. The accuracy of present MIBM is in a good agreement with the body-conforming mesh method. Various discrete delta function have been tested and used to smooth the boundary force. The MIBM proposed here will be extended to solve fluid-structure interaction problems in the next chapter.

Modeling fluid-structure interaction with MIBM

5.1 Introduction

In previous chapter we have developed the moving immersed boundary method and applied to the fluid flows over stationary and forced oscillating objects. In this chapter we will couple the fluid Navier-Stokes equations with the rigid body motion equations via the moving immersed boundary method for studying two way fluid-structure interactions.

We will first recast the fluid-structure interaction problem into the immersed boundary formulation in Section 5.2.2. To obtain stable solutions and reduce the computational cost, we construct a novel strongly coupled approach in Section 5.4 by taking advantage of the projection method and the moving immersed boundary method. Next we consider the particulate flow simulation by taking the particle collisions into consideration. We then demonstrate the proposed method with various numerical examples in Section 5.6.

5.2 Mathematical formulation

5.2.1 Governing equations

Figure 5.1 illustrates the physical domain of a general FSI problem, where the fluid and the rigid body occupy the domain Ω_f and Ω_s respectively. The interaction takes place at the their common boundary $\partial\Omega_i = \Omega_f \cap \Omega_s$. The whole system is subjected to the gravitational acceleration $\mathbf{g}$.

Note that the symbol $\mathbf{u}$ is often referred to the displacement in solid mechanics, while it is frequently used as the velocity in the fluid dynamics. In this chapter we will redefine the symbols for the fluid and the solid variables. In following context the subscript f will be used for the fluid variables and the subscript s for the solid

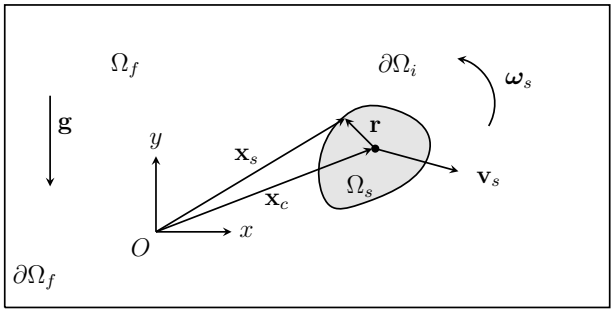


Figure 5.1: Schematic representation of the fluid-structure interaction domain.

variables. With these new notations, the Navier-Stokes equations can be rewritten as

$$\frac{\partial \mathbf{v}_f}{\partial t} + \nabla \cdot (\mathbf{v}_f \otimes \mathbf{v}_f) = \nabla \cdot \boldsymbol{\sigma}_f + \mathbf{g} \tag{5.1a}$$

$$\nabla \cdot \mathbf{v}_f = 0 \tag{5.1b}$$

where $\mathbf{v}_f$ is the fluid velocity vector and the fluid stress tensor $\boldsymbol{\sigma}_f$ is given by

$$\boldsymbol{\sigma}_f = -\frac{p}{\rho_f}\mathbf{I} + \nu(\nabla \mathbf{v}_f + (\nabla \mathbf{v}_f)^{\mathrm{T}}) \tag{5.1c}$$

where p is the fluid pressure, ρ_f the fluid density, ν the fluid kinematic viscosity. Appropriate initial and boundary conditions are assumed to the fluid Navier-Stokes equations to ensure that the problem is well posed.
The solid equations are usually described in a Lagrangian framework in the material domain Ω_s. The motion of the rigid body is governed by the Newton-Euler equations, which can be written as

$$m_s \frac{d\mathbf{v}_s}{dt} = \rho_f \int_{\partial\Omega_i} \boldsymbol{\sigma}_f \cdot \mathbf{n} ds + m_s(1 - \frac{\rho_f}{\rho_s})\mathbf{g} \tag{5.2a}$$

$$I_s \frac{d\boldsymbol{\omega}_s}{dt} = \rho_f \int_{\partial\Omega_i} \mathbf{r} \times (\boldsymbol{\sigma}_f \cdot \mathbf{n})\, ds \tag{5.2b}$$

where

- m_s represents the solid mass;
- ρ_s is the solid density;
- I_s designates the moment of inertia;
- $\mathbf{v}_s$ is the translational velocity of solid;

- $\boldsymbol{\omega}_s$ is the angular velocity of solid;
- $\mathbf{r}$ represents the position vector of the surface point with respect to the solid mass center ($\mathbf{r} = \mathbf{x}_s - \mathbf{x}_c$);
- $\mathbf{x}_s$ is the solid position vector at the surface;
- $\mathbf{x}_c$ is the solid gravity center vector;
- $\mathbf{n}$ represents the outward-pointing normal vector to the surface $\partial\Omega_i$.

The position of the rigid body can be obtained by the integration of the following kinematic equations, i.e.,

$$\frac{d\mathbf{x}_c}{dt} = \mathbf{v}_s \tag{5.3a}$$

$$\frac{d\boldsymbol{\theta}_c}{dt} = \boldsymbol{\omega}_s \tag{5.3b}$$

where $\boldsymbol{\theta}_c$ designates the rotation angle of the solid mass center.
On the fluid-structure interface $\partial\Omega_i$ the following no-slip boundary condition

$$\mathbf{v}_f = \mathbf{v}_s + \boldsymbol{\omega}_s \times \mathbf{r} \tag{5.4}$$

needs to be satisfied in order to take the fluid-structure interaction into account.

5.2.2 Immersed boundary formulation

The immersed boundary method approximates the above fluid-structure interaction problem by replacing the solid domain with the surrounding fluid. To account for the presence of the immersed solid, a boundary force $\mathbf{f}$ is introduced and added into the fluid momentum equation. Therefore the fluid is simply simulated in a fixed domain $\overline{\Omega} = \Omega_f(t) \cup \Omega_s(t)$ irrespective to the movement of the immersed solid. The incompressible fluid Navier-Stokes equations (5.1) in the immersed boundary formulation can be written as

$$\frac{\partial \mathbf{v}_f}{\partial t} + \nabla \cdot (\mathbf{v}_f \otimes \mathbf{v}_f) = -\frac{1}{\rho_f}\nabla p + \nu \nabla^2 \mathbf{v}_f + \mathbf{f} \quad \text{in } \overline{\Omega} \tag{5.5a}$$

$$\nabla \cdot \mathbf{v}_f = 0 \quad \text{in } \overline{\Omega} \tag{5.5b}$$

$$\mathbf{v}_f = \mathbf{v}_s + \boldsymbol{\omega}_s \times \mathbf{r} \quad \text{on } \partial\Omega_i \tag{5.5c}$$

where the effect of gravity in the fluid momentum equation is from now on incorporated into the pressure. The boundary force $\mathbf{f}$ here acts as a Lagrange multiplier

for satisfying the internal interface velocity condition (5.5c) from the mathematical point of view. The force $\mathbf{f}$ has also a physical signification that it represents the contact force between the fluid and solid.

By considering that the direct evaluation of the surface integrals in (5.2) would require considerable numerical efforts, Uhlmann (2005) proposed a more efficient way of evaluating the hydrodynamic force and the torque by using the Cauchy's principle as follows

$$\rho_f \int_{\partial\Omega_i} \boldsymbol{\sigma}_f \cdot \mathbf{n} ds = -\rho_f \int_{\Omega_s} \mathbf{f} dV + \frac{d}{dt} \int_{\Omega_s} \rho_f \mathbf{v}_f dV \tag{5.6a}$$

$$\rho_f \int_{\partial\Omega_i} \mathbf{r} \times (\boldsymbol{\sigma}_f \cdot \mathbf{n})\, ds = -\rho_f \int_{\Omega_s} \mathbf{r} \times \mathbf{f} dV + \frac{d}{dt} \int_{\Omega_s} \rho_f \mathbf{r} \times \mathbf{v}_f dV \tag{5.6b}$$

Uhlmann (2005) further assumed that the fluid inside $\partial\Omega_i$ follows the rigid body motion, thus

$$\frac{d}{dt} \int_{\Omega_s} \rho_f \mathbf{v}_f dV = \frac{\rho_f m_s}{\rho_s} \frac{d\mathbf{v}_s}{dt} \tag{5.7a}$$

$$\frac{d}{dt} \int_{\Omega_s} \rho_f \mathbf{r} \times \mathbf{v}_f dV = \frac{\rho_f I_s}{\rho_s} \frac{d\boldsymbol{\omega}_s}{dt} \tag{5.7b}$$

Substituting (5.6), (5.7) into (5.2) yields

$$m_s \frac{d\mathbf{v}_s}{dt} = -\frac{\rho_f}{1 - \rho_f/\rho_s} \int_{\Omega_s} \mathbf{f} dV + m_s \mathbf{g} \tag{5.8a}$$

$$I_s \frac{d\boldsymbol{\omega}_s}{dt} = -\frac{\rho_f}{1 - \rho_f/\rho_s} \int_{\Omega_s} \mathbf{r} \times \mathbf{f} dV \tag{5.8b}$$

However this scheme has a singularity problem when the fluid density is equal to the solid density. Kempe & Fröhlich (2012a) recognized this problem and proposed an alternative solution to calculate the volume integrals. For example in two dimensions

$$\int_{\Omega_s} \mathbf{v}_f dV = \sum_{i,j} {\mathbf{v}_f}^{i,j} \alpha^{i,j} \Delta x \Delta y \tag{5.9}$$

where $\alpha^{i,j}$ is the solid volume fraction in an fluid cell. In Kempe & Fröhlich (2012a), a level-set function Φ is employed to approximate the volume fraction

$$\alpha^{i,j} = \frac{\sum_{m=1}^{4} -\Phi_m H(-\Phi_m)}{\sum_{m=1}^{4} |\Phi_m|} \tag{5.10}$$

where H is the Heaviside step function. Kempe & Fröhlich (2012a) advances this equation using an explicit scheme. However implicit implement can be difficult,

since the fluid velocity should be evaluated simultaneously.
In the present work, we omit the fluid motion inside the solid and will show that the results are not essentially changed. By doing so, we avoid the singularity problem when $\rho_f/\rho_s = 1.0$ encountered by Uhlmann (2005). Now the entire fluid-structure interaction problem in the immersed boundary formulation can be summarized as

$$\frac{\partial \mathbf{v}_f}{\partial t} + \nabla \cdot (\mathbf{v}_f \otimes \mathbf{v}_f) = -\frac{1}{\rho_f}\nabla p + \nu \nabla^2 \mathbf{v}_f + \mathbf{f} \quad \text{in } \overline{\Omega} \tag{5.11a}$$

$$\nabla \cdot \mathbf{v}_f = 0 \quad \text{in } \overline{\Omega} \tag{5.11b}$$

$$m_s \frac{d\mathbf{v}_s}{dt} = -\rho_f \int_{\Omega_s} \mathbf{f} dV + m_s(1 - \frac{\rho_f}{\rho_s})\mathbf{g} \tag{5.11c}$$

$$I_s \frac{d\boldsymbol{\omega}_s}{dt} = -\rho_f \int_{\Omega_s} \mathbf{r} \times \mathbf{f} dV \tag{5.11d}$$

$$\mathbf{v}_f = \mathbf{v}_s + \boldsymbol{\omega}_s \times \mathbf{r} \quad \text{on } \partial\Omega_i \tag{5.11e}$$

5.3 Coupling methods

In general, the numerical procedures for solving the two-way fluid-structure interaction can be broadly classified into two groups: (***i***) the monolithic (direct) approach and (***ii***) the partitioned (segregated) approach (see Figure 5.2).

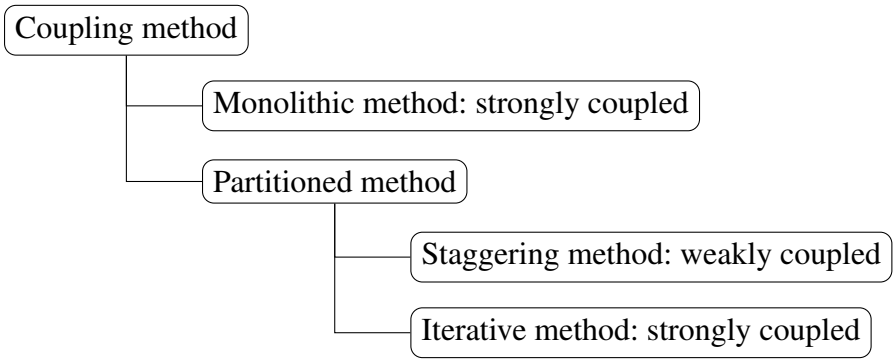


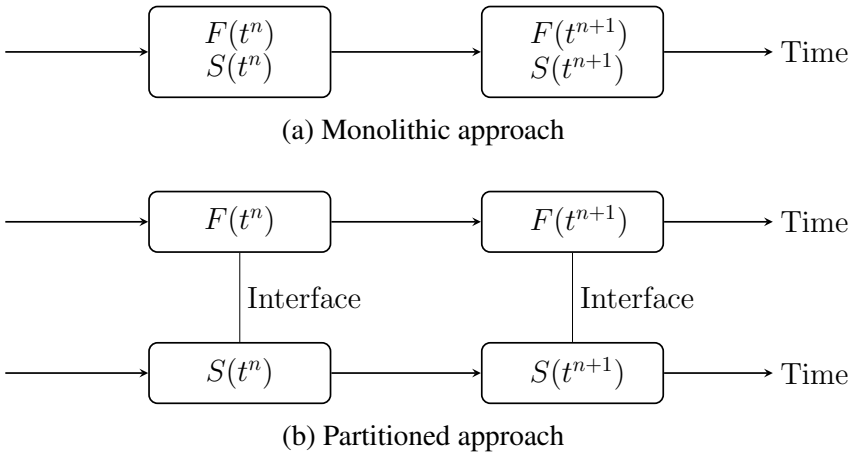
Figure 5.2: Fluid-structure coupling methods.

(*i*) Monolithic method

In the monolithic method, the subproblems are discretized in time and space in the same way with no distinction between the fluid and the solid. The resulting large system is solved simultaneously by a unified algorithm, as illustrated in Figure 5.3a. A typical choice is to use the stabilized finite element method for the fluid combined with the standard finite element method for the solid. Another option is to employ

the finite volume method for both subproblems, which however is less frequently used in the literature.

The monolithic approach is very advantageous for such a multidisciplinary problem, both from the stability and accuracy point of view. The interface condition is enforced directly, thus the monolithic approach is strongly coupled by construction. However, it requires considerable efforts and expertise on the reformulations. Moreover it is difficult to maintain state-of-the-art schemes for each subproblem, and to devise efficient global preconditioners.



(a) Monolithic approach

(b) Partitioned approach

Figure 5.3: Illustration of the monolithic approach and the partitioned approach (F: fluid solver; S: solid solver).

(*ii*) Partitioned method

On the contrary, the partitioned method formulates the fluid equations and the solid equations into different systems and solves them separately. The two physical domains exchange data at the interface regularly through additional coupling schemes. This procedure is shown in Figure 5.3b. The partitioned method has gained a lot of popularities in recent years for the following reasons: It allows the two subproblems to chose their own favorite discretization; Sophisticated solvers or software that have already been developed for each subproblem can be used directly in the partitioned method.

The partitioned approach can be further classified into two groups based on the additional coupling scheme, namely the explicit coupling (weakly or loosely coupled) and the implicit coupling (strongly or tightly coupled) (Matthies & Steindorf, 2002, 2003; Matthies *et al.*, 2006; Hou *et al.*, 2012), as shown in Figure 5.2. The weak coupling is very efficient since it solves the two subproblems only once in each time step in a sequential manner. Consider the example in Figure 5.4, the fluid

equations are solved at t^{n+1} with the solid information at t^n, and the solid equations are then computed at t^{n+1} with the updated fluid stresses. In spite of the efficiency, the interface conditions can never be fulfilled in the weak coupling calculation. As a result, spurious energy is generated at the interface and leads to unstable solutions, which is often called the added mass effect. The stability criterion mainly depends on the density ratio between the solid and the fluid. Below a certain threshold the calculation diverges intermediately for any chosen time step for incompressible fluids (Causin *et al.*, 2005; Förster *et al.*, 2007; Kassiotis *et al.*, 2011).

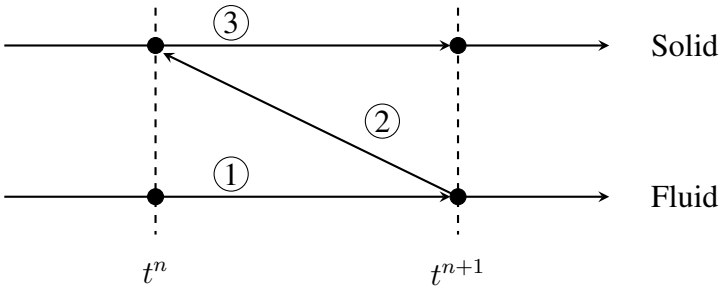


Figure 5.4: Explicit coupling algorithm for fluid-structure interaction.

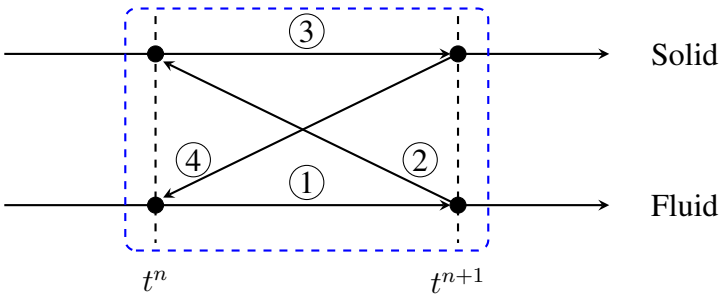


Figure 5.5: Iterative implicit coupling algorithm for fluid-structure interaction.

The interface conditions can be satisfied by employing an iterative approach between the fluid equations and the solid equations (see Figure 5.5), which implies the implicit coupling (strongly or tightly coupled) method. The strongly coupled method can be comparable to the monolithic method. The iteration is often performed either with fixed-point methods or Newton–Raphson methods. The Gauss-Seidel iterations are frequently used in the fixed-point methods (Kassiotis *et al.*, 2011), but they are often slow and sometimes lead to divergence. Using the Aitken relaxation method or steepest descent method can accelerate the calculation (Küttler & Wall, 2008), but convergence is not always guaranteed. Song *et al.* (2013) enhanced the convergence of this iterative process by employing an added mass compensation technique, which is proven to be unconditionally stable. The Newton-

Raphson methods improve the convergence of the FSI sub-iterations and possess a quadratic convergence rate. However they need to compute the Jacobian matrix, which is rather costly. Various methods have been developed to approximate the Jacobian matrix (Matthies & Steindorf, 2002, 2003; Matthies *et al.*, 2006).

5.4 MIBM for strongly coupled FSI

5.4.1 Numerical discretization

The governing equation (5.11) can be discretized as

Fluid:

$$\frac{{\mathbf{v}_f}^{n+1} - {\mathbf{v}_f}^{n}}{\Delta t} + \mathbf{C} = -\frac{1}{\rho_f}\mathcal{G}p^{n+1} + \frac{\nu}{2}\mathcal{L}({\mathbf{v}_f}^{n+1} + {\mathbf{v}_f}^{n}) + \mathbf{f}^{n+1} \tag{5.12a}$$

$$\mathcal{D}{\mathbf{v}_f}^{n+1} = 0 \tag{5.12b}$$

$${\mathbf{v}_f}^{n+1} = {\mathbf{v}_s}^{n+1} + {\boldsymbol{\omega}_s}^{n+1} \times \mathbf{r}^{n+1} \quad \text{on} \;\; \partial\Omega_i^{n+1} \tag{5.12c}$$

where

$$\mathbf{C} = \frac{3}{2}\mathcal{N}({\mathbf{v}_f}^{n}) - \frac{1}{2}\mathcal{N}({\mathbf{v}_f}^{n-1}) \tag{5.12d}$$

$$\mathbf{f}^{n+1} = \mathcal{S}\mathbf{F}^{n+1} \tag{5.12e}$$

Solid:

$$m_s\frac{{\mathbf{v}_s}^{n+1} - {\mathbf{v}_s}^{n}}{\Delta t} = -\rho_f \int_{\Omega_s} \mathbf{f}^{n+1} dV + m_s(1 - \frac{\rho_f}{\rho_s})\mathbf{g} \tag{5.13a}$$

$$I_s\frac{{\boldsymbol{\omega}_s}^{n+1} - {\boldsymbol{\omega}_s}^{n}}{\Delta t} = -\rho_f \int_{\Omega_s} \mathbf{r} \times \mathbf{f}^{n+1} dV \tag{5.13b}$$

$$\frac{{\mathbf{x}_c}^{n+1} - {\mathbf{x}_c}^{n}}{\Delta t} = {\mathbf{v}_s}^{n+1} \tag{5.13c}$$

$$\frac{{\boldsymbol{\theta}_c}^{n+1} - {\boldsymbol{\theta}_c}^{n}}{\Delta t} = {\boldsymbol{\omega}_s}^{n+1} \tag{5.13d}$$

Uhlmann (2005) assumed that the total amount of force and torque are not changed during the transfer between Lagrangian and Eulerian locations. Hence the solid

equations can be solved as follows

$$m_s \frac{{\mathbf{v}_s}^{n+1} - {\mathbf{v}_s}^{n}}{\Delta t} = -\rho_f \sum_{l=1}^{n_b} \mathbf{F}^{n+1} \Delta V_l + m_s (1 - \frac{\rho_f}{\rho_s}) \mathbf{g} \tag{5.14a}$$

$$I_s \frac{{\boldsymbol{\omega}_s}^{n+1} - {\boldsymbol{\omega}_s}^{n}}{\Delta t} = -\rho_f \sum_{l=1}^{n_b} \mathbf{r} \times \mathbf{F}^{n+1} \Delta V_l \tag{5.14b}$$

Therefore, provided the boundary force, the solid equations can be easily solved. For the fluid part, we employ previous operator splitting scheme as

(1) Prediction step for ${\hat{\mathbf{v}}_f}^{n+1}$

$$\frac{{\hat{\mathbf{v}}_f}^{n+1} - {\mathbf{v}_f}^{n}}{\Delta t} + \mathbf{C} = -\frac{1}{\rho_f} \mathcal{G} p^n + \frac{\nu}{2} \mathcal{L} ({\hat{\mathbf{v}}_f}^{n+1} + {\mathbf{v}_f}^{n}) \tag{5.15}$$

(2) Immersed boundary forcing step for imposing the no-slip wall condition at the interface

$$\frac{{\tilde{\mathbf{v}}_f}^{n+1} - {\hat{\mathbf{v}}_f}^{n+1}}{\Delta t} = \mathcal{S} \mathbf{F}^{n+1} \tag{5.16a}$$

$$\mathcal{T} {\tilde{\mathbf{v}}_f}^{n+1} = {\mathbf{v}_s}^{n+1} + {\boldsymbol{\omega}_s}^{n+1} \times \mathbf{r}^{n+1} \quad \text{on} \ \ \partial \Omega_i^{n+1} \tag{5.16b}$$

Using the moving force matrix notation defined previously, we obtain

$$\mathcal{M} \mathbf{F}^{n+1} = \frac{{\mathbf{v}_s}^{n+1} + {\boldsymbol{\omega}_s}^{n+1} \times \mathbf{r}^{n+1} - \mathcal{T} {\hat{\mathbf{v}}_f}^{n+1}}{\Delta t} \tag{5.17a}$$

$${\tilde{\mathbf{v}}_f}^{n+1} = {\hat{\mathbf{v}}_f}^{n+1} + \Delta t \mathcal{S} \mathbf{F}^{n+1} \tag{5.17b}$$

where the solid velocity and position are not known in advance and should be evaluated along with the fluid fields simultaneously. The coupling strategies will be presented in the next section.

(3) Projection step for obtaining a divergence free velocity ${\mathbf{v}_f}^{n+1}$

$$\frac{{\mathbf{v}_f}^{n+1} - {\tilde{\mathbf{v}}_f}^{n+1}}{\Delta t} = -\mathcal{G} \phi^{n+1} \tag{5.18a}$$

$$\mathcal{D} {\mathbf{v}_f}^{n+1} = 0 \tag{5.18b}$$

which is often performed as

$$\mathcal{L} \phi^{n+1} = \frac{1}{\Delta t} \mathcal{D} {\tilde{\mathbf{v}}_f}^{n+1} \tag{5.19a}$$

$${\mathbf{v}_f}^{n+1} = {\tilde{\mathbf{v}}_f}^{n+1} - \Delta t \mathcal{G} \phi^{n+1} \tag{5.19b}$$

The final pressure is advanced by

$$p^{n+1} = p^n + \phi^{n+1} - \frac{\nu}{2}\mathcal{D}\hat{\mathbf{v}}_f{}^{n+1} \tag{5.20}$$

5.4.2 Novel strongly coupled scheme

In this subsection, we will propose a novel coupling approach for the FSI based on the extended MIBM. In viewing of previous discussions, the partitioned strongly coupled method is preferred in the present work for the reason of modularity, stability and accuracy.

Mathematically speaking, the FSI problem of (5.11) is accomplished by the Lagrange multiplier method to joint two subdomains on a common boundary. The unknowns of the entire system are ($\mathbf{v}_f{}^{n+1}$, p^{n+1}) for the fluid, ($\mathbf{x}_s{}^{n+1}$, $\dot{\mathbf{x}}_s{}^{n+1}$) for the solid, and $\mathbf{F}^{n+1}$ for the interface condition. The conventional strongly coupled method iterates those variables at each time step in order to match the interface condition, as shown in Figure 5.5. Even it leads to accurate and stable solutions, solving implicit coupling usually exhibits a prohibitive computational cost. This becomes more unaffordable in ALE formulation, since the mesh and the associated matrices have to be updated per time step.

Fernandez *et al.* (2007) proposed an efficient coupling method by taking advantage of the projection method. The prediction (ALE-advection-viscous) step is ruled out from the inner loop to reduce computational cost, so that the mesh and the associated matrices are computed once at each time step. Finally only the projection step is coupled implicitly to ensure stability. Fernandez *et al.* (2007) has shown that this semi-implicit scheme is stable for a reasonable range of the discretization parameters, compared to the explicit coupling approach.

We extend this idea to the immersed boundary method. However the projection step usually is the most time-consuming part in the projection method. In spite of various methods (e.g. Aitken relaxation) are available to accelerate the coupling procedure, the computation cost still remains high. We also notice that in (5.14) the solid is coupled to the fluid by the boundary force not the pressure, and in (5.17) the boundary force is determined by the solid velocity and position.

Therefore, we can move out the time-consuming projection step from the inner iteration and replace it with the immersed boundary forcing step. This novel coupling scheme is illustrated in Figure 5.6. We would like to emphasize that unlike the semi-implicit scheme of Fernandez *et al.* (2007), present method is strongly coupled. The reason is that in ALE the prediction step requires the new fluid domain (solid position) and the interface velocity (solid velocity), while in IBM the

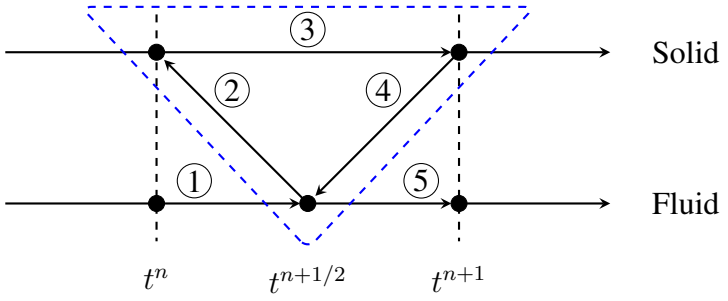


Figure 5.6: Novel implicit coupling algorithm for fluid-structure interaction.

Algorithm 7: Novel implicit coupling scheme

Given: ${\mathbf{v}_f}^n$, p^n, ${\mathbf{x}_s}^n$, ${\dot{\mathbf{x}}_s}^n$;
(Fluid) Predict the velocity ${\tilde{\mathbf{v}}_f}^{n+1}$ using (5.15);
Initialize values: $(\cdot)^{k=0,n+1} = (\cdot)^n$, where $(\cdot)$ includes $\mathbf{x}_s, \dot{\mathbf{x}}_s, \mathbf{F}$;
for $k = 0$ to $k_{\max}$ **do**
    (Fluid) Construct or update the interpolation operator matrix $\mathcal{I}({\mathbf{x}_s}^{k,n+1})$ and the moving force coefficient matrix $\mathcal{M}({\mathbf{x}_s}^{k,n+1})$;
    (Fluid) Interpolate the fluid velocity $\mathcal{I}({\mathbf{x}_s}^{k,n+1}){\tilde{\mathbf{v}}_f}^{n+1}$;
    (Interface) Solve the moving force equation (5.17a) for $\mathbf{F}^{k+1,n+1}$ with ${\dot{\mathbf{x}}_s}^{k,n+1}$;
    (Solid) Compute the solid equations for $({\mathbf{x}_s}^{k+1,n+1}, {\dot{\mathbf{x}}_s}^{k+1,n+1})$ using (5.14);
    if $||(\cdot)^{k+1,n+1} - (\cdot)^{k,n+1}||/||(\cdot)^{k+1,n+1}|| <$ tolerance **then**
        $(\cdot)^{n+1} = (\cdot)^{k+1,n+1}$;
        break;
    else
        $k = k + 1$;
    end
end
(Fluid) Correct the fluid velocity to ${\hat{\mathbf{v}}_f}^{n+1}$ with $\mathbf{F}^{n+1}$ using (5.17b);
(Fluid) Solve the pressure Poisson equation and compute the final velocity ${\mathbf{v}_f}^{n+1}$ and the pressure p^{n+1} using (5.19a), (5.19b), (5.20).

prediction step is performed on a stationary combined domain and the boundary force is not incorporated in this step for the solid effects. The prediction step will not change the value if placed in the inner iteration and can be moved out. Note that the moving force equation is considered as an implicit equation for the no-slip boundary condition for the fluid at the interface. Therefore the implicit coupling of the immersed boundary forcing step with the solid equations features an overall strongly coupled method.

Note that the moving force equation is a non-linear equation, because the interpolation, spreading operators and the boundary force are functions of the solid position

$\mathbf{x}_s{}^{n+1}$. Hence $\mathcal{M}^{n+1}\mathbf{F}^{n+1} = \mathcal{I}(\mathbf{x}_s{}^{n+1})\mathcal{S}(\mathbf{x}_s{}^{n+1})\mathbf{F}(\mathbf{x}_s{}^{n+1})$. We can linearize this equation by treating the moving force coefficient matrix explicitly $\mathcal{M}(\mathbf{x}_s{}^{n})$, but this will decrease the overall accuracy due to the time lag, as indicated by Lacis *et al.* (2016) in the immersed boundary projection method. In order to preserve a high accuracy, fully implicit implementation of the moving force equation is considered in Algorithm 7.

5.5 Particulate flow modeling

5.5.1 Introduction

Particulate flows are of considerable interest in many engineering applications, such as the fluidized bed, the blood flow and the sediment transport near river beds. The traditional body-conforming mesh methods meets bottlenecks in those simulations, as the topological changes of the fluid domain are so significant that it is extremely difficult to adapt the mesh to fit the new domain. The deforming-mesh can lead to inaccurate solutions and potentially catastrophic results. During approach of two particles, the mesh cells in the gap are compressed. If the contact occurs, those cells will collapse. In fact even before touching, the calculation explodes due to these compressed meshes with negative volumes.

On the opposite direction for rebound, the meshes are expanded. If previously there are no cell layers between the particles, i.e. they are strictly touched, the two particles will always stick together or the particle will stick to the wall. Because they share the same node that can not be divided. If they are not strictly touched, namely there are few cell layers between them during contact, the meshes can get hugely expanded when the solid movement is large.

In those situations re-meshing can be helpful by re-distributing the nodes and re-ordering their connectivities. The unstructured mesh is more suitable for this purpose compared to the structured one, since it is hard to fit arbitrary domain topology with the structured mesh without user intervention. Sophisticated algorithms are available to compute the unstructured mesh automatically, such as the Delaunay method and the advancing front method. However, re-meshing has to map the flow fields from the old mesh to the new mesh while deforming-mesh does not need to. This procedure can introduce undesired numerical errors. In strongly coupled FSI, it becomes a heavy burden as the mesh solver is called and the fluid matrices are computed every inner iteration in one time step. Computational cost can be saved by employing a local re-meshing, where only several nodes are added and removed

in the zone of interest. In any case maintaining a good mesh quality dynamically has always been a central issue in the body-conforming mesh methods.

The immersed boundary method has successfully circumvented the dynamic mesh problem, showing a great capacity in modeling particulate flows (Glowinski *et al.*, 1999, 2001; Uhlmann, 2005; Wan & Turek, 2006; Kempe & Fröhlich, 2012a). In this section we will extend the MIBM to such flow simulations. However the IBM may have a problem when the particles come close or a particle is close to a wall. This is because the partition of unity property of the discrete delta function is violated in these situations (Kempe & Fröhlich, 2012a).

Figure 5.7a shows the grid stencil of IBM when two particles are close. The two particles will spread two different forces to the same Eulerian points. It is rather ambiguous about which force value should be taken at the overlapping points (Vanella & Balaras, 2009). Therefore the sum of the discrete delta functions at the Eulerian grid is no longer equal to one. When a particle comes to a wall, as shown in Figure 5.7b, the force is spread to wall nodes and hence is lost with respect to the fluid. On the other hand, it is not clear which velocity should be assigned to the wall nodes in order to obtain a correct interpolated velocity.

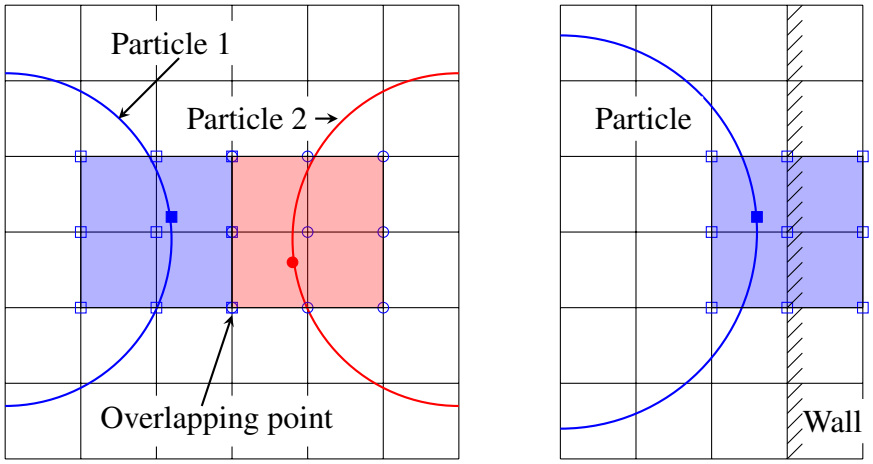


Figure 5.7: Grid stencils of IBM for collision modeling: (a) two close particles; (b) one particle close to a wall. The supported Eulerian points for interpolation and spreading are included by the shaded zone.

Collisions in particulate flows also pose other difficulties in numerical simulations. For example, during the collision of two particles, a very thin lubrication layer is formed. The fluid is squeezed out of their gap when they approach or is pushed back into the gap when they rebound. To fully resolve the flow in the narrow gap requires significantly refined spatial and temporal resolutions, which makes the computation

unaffordable for many particles (Kempe & Fröhlich, 2012b).

5.5.2 Collision model

In this work we employ the collision model of Glowinski *et al.* (1999, 2001), which is widely used in the literature (Feng & Michaelides, 2004; Uhlmann, 2005; Kempe & Fröhlich, 2012a; Favier *et al.*, 2014). This model attempts to prevent solid particles from overlapping by a repulsive force, when the gap between two particles exceeds a given threshold. The expression of the repulsive force for particle-particle collision is given by

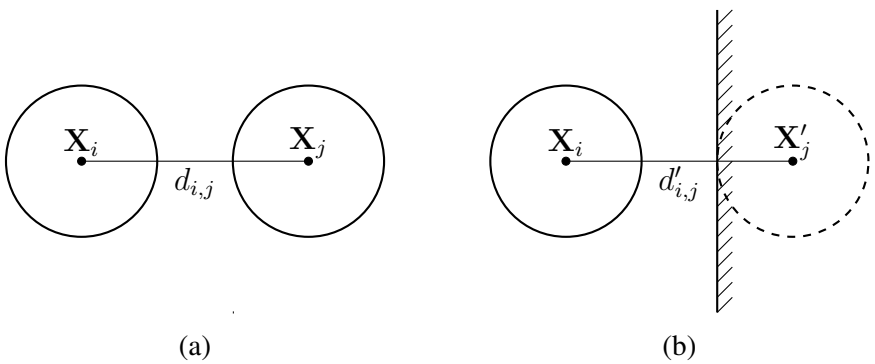
$$\mathbf{F}_{i,j}^{P} = \begin{cases} 0, & d_{i,j} \geqslant R_i + R_j + \xi_n \\ \dfrac{1}{\varepsilon_P}(\mathbf{X}_i - \mathbf{X}_j)(R_i + R_j + \xi_n - d_{i,j})^2, & d_{i,j} < R_i + R_j + \xi_n \end{cases} \tag{5.21}$$

where $d_{i,j} = |\mathbf{X}_i - \mathbf{X}_j|$ is the distance between the centers of the i-th and j-th particles, whose radius are R_i and R_j respectively (see Figure 5.8a). ξ_n is the force range or the threshold, which is often chosen to be $\xi_n = 2h \sim 3h$. ε_P is a small positive stiffness parameter.

Similarly the repulsive force between a particle and a wall is given by

$$\mathbf{F}_{i,j}^{W} = \begin{cases} 0, & d'_{i,j} \geqslant 2R_i + \xi_n \\ \dfrac{1}{\varepsilon_W}(\mathbf{X}_i - \mathbf{X}'_j)(2R_i + \xi_n - d'_{i,j})^2, & d'_{i,j} < 2R_i + \xi_n \end{cases} \tag{5.22}$$

where $d_{i,j} = |\mathbf{X}_i - \mathbf{X}'_j|$ is the distance between the centers of the i-th particle and an imaginary particle with respect to the wall, as shown in Figure 5.8b. ε_W is a another small positive stiffness parameter. Glowinski *et al.* (2001) provided the justification and discussed how to choose these stiffness values.



(a) (b)

Figure 5.8: Collision model of Glowinski *et al.* (1999, 2001): (a) particle-particle collision and (b) particle-wall collision.

Now the repulsive force is added as an external force into the total force in the particle motion equation

$$m_s \frac{d\mathbf{v}_s}{dt} = \rho_f \int_{\partial\Omega_i} \boldsymbol{\sigma}_f \cdot \mathbf{n} ds + m_s(1 - \frac{\rho_f}{\rho_s})\mathbf{g} + \mathbf{F}'_i \tag{5.23}$$

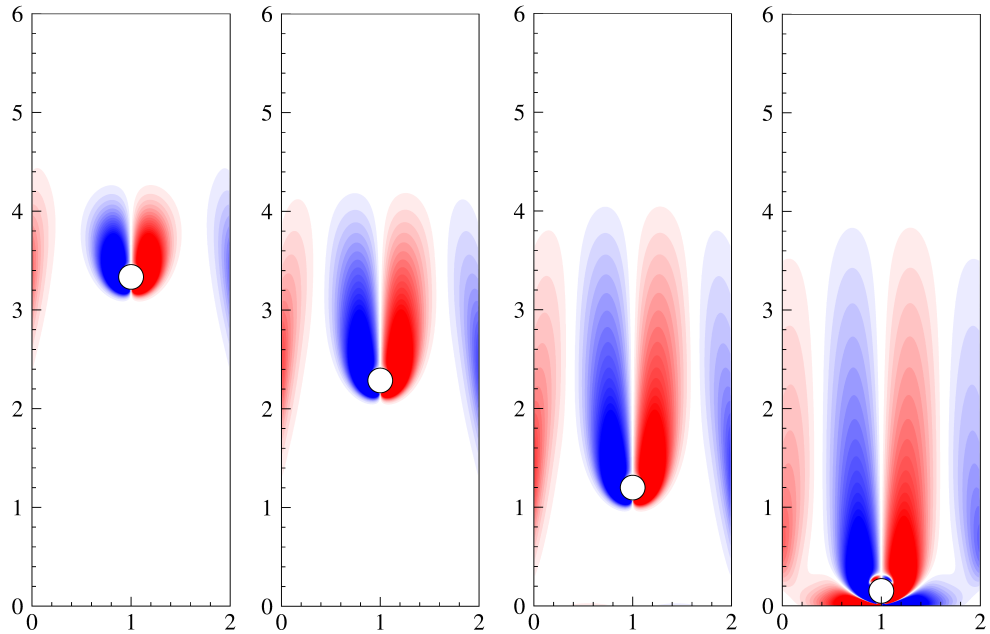
where

$$\mathbf{F}'_i = \sum_{j=1, j\neq i}^{N} \mathbf{F}^P_{i,j} + \sum_{j=1}^{4} \mathbf{F}^W_{i,j} \tag{5.24}$$

5.6 Numerical examples

5.6.1 Freely falling cylinder in a confined channel

We first investigate the motion of a circular cylinder falling freely between two parallel walls in a quiescent fluid. The width and height of the computational domain are chosen to be 2 cm and 6 cm. The circular cylinder with a diameter of $D = 0.25$ cm is released initially from the position (1 cm, 4 cm) and falls down because of gravity. The density of the cylinder and the surrounding fluid are $\rho_s = 1.25$ g/cm^3 and $\rho_f = 1.0$ g/cm^3 respectively. The fluid dynamic viscosity μ is set to 0.1 g/(cm·s).



(a) $t = 0.2$ s (b) $t = 0.4$ s (c) $t = 0.6$ s (d) $t = 0.8$ s

Figure 5.9: Vorticity fields at different times t = 0.2 s, 0.4 s, 0.6 s, 0.8 s for the freely falling cylinder in a confined channel problem. The contour levels are set from -15 (blue) to 15 (red) with an increment of 1.

The calculations are carried out on two different uniform meshes, i.e. $h = 1/48$ and $h = 1/96$, to check the mesh sensitivity. The time step is chosen to be $\Delta t = 0.001$ s and the resulting maximum CFL number is reported to be 0.46. The vorticity around the falling cylinder is shown in Figure 5.9 at different times t = 0.2 s, 0.4 s, 0.6 s, 0.8 s. The cylinder quickly reaches a uniform falling velocity until it hits the bottom of the channel. We plot the flow quantities as a function of time in Figure 5.10, including the longitude position y_c of the cylinder center, the vertical velocity v_c, the Reynolds number Re and the translational kinetic energy E_T. Here Re and E_T are defined as $Re = (\rho_s D\sqrt{u_c^2 + v_c^2})/\mu$ and $E_T = 0.5m_s(u_c^2 + v_c^2)$ respectively, where u_c is the horizontal velocity component. For comparison, the results of Wan & Turek (2006) are included in Figure 5.10, taken from $h = 1/96$. Good agreements have been obtained.

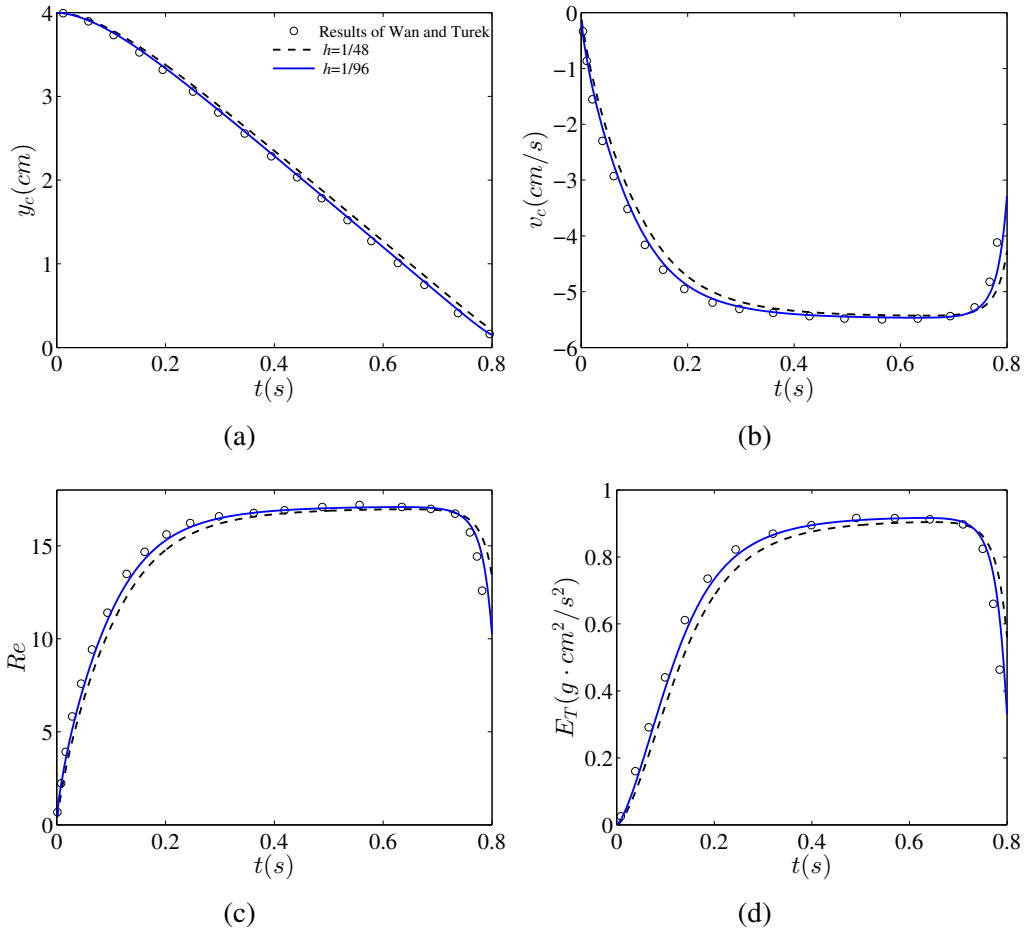
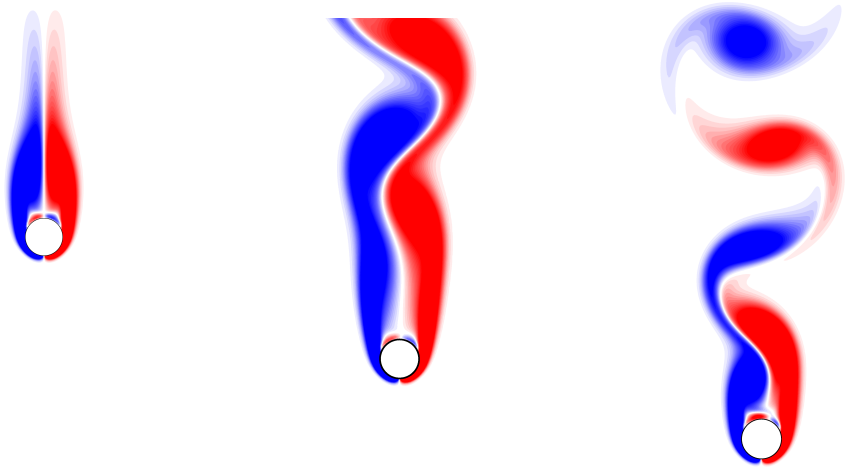


Figure 5.10: Time evolution of longitude position y_c, velocity v_c of the cylinder center, the Reynolds number Re and the translational kinetic energy E_T for the freely falling cylinder in a confined channel problem. "$\circ$", result of Wan & Turek (2006); "—", present result with $h = 1/96$; "- - - -", present result with $h = 1/48$.

5.6.2 Freely falling and rising cylinder in an open domain

Next we consider an object freely falling and rising in an open domain as another test. This phenomenon happens frequently in nature and a large amount of work can be found in the literature. Here we compare our numerical results with the data of Namkoong *et al.* (2008) and Lacis *et al.* (2016). Namkoong *et al.* (2008) performed the simulation using a body-fitted ALE formulation while Lacis *et al.* (2016) employed the immersed boundary projection method.

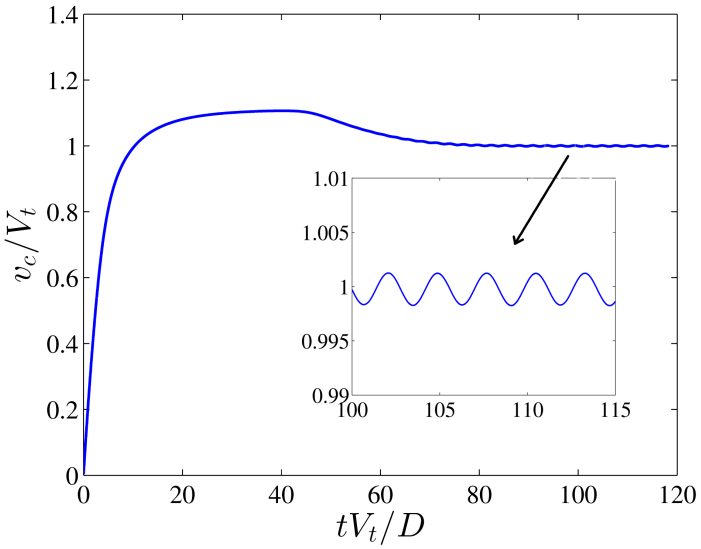
Two density ratios are considered in this study, i.e. $\rho_s/\rho_f = 1.01$ for the falling case and $\rho_s/\rho_f = 0.99$ for the rising simulation. A large computational domain is taken as $[-5D, 5D] \times [-70D, 70D]$ with free-slip boundary conditions applied at all the exterior boundaries, where $D = 0.5$ cm is the cylinder diameter. A uniform mesh is employed to cover the computational domain, and the mesh resolution is kept to $0.04D$ in order to compare with Lacis *et al.* (2016). Initially the cylinder is located at $\pm 65D$, depending on the situation ($65D$ for the falling case, $-65D$ for the rising case). The Reynolds number $Re = V_t D/\nu_f$ here is 156, where V_t is the terminal velocity. Note that the Reynolds number depends on the Galileo number $G = (|\rho_s/\rho_f - 1|gD^3)^{1/2}/\nu_f$ (the gravity force divided by the viscous force, $G = 138$) and the density ratio ρ_s/ρ_f.



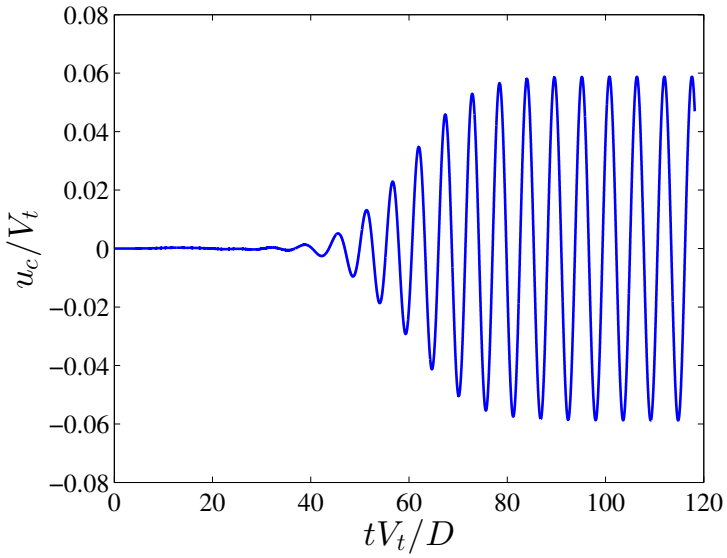
(a) $tV_t/D = 10$ (b) $tV_t/D = 50$ (c) $tV_t/D = 90$

Figure 5.11: Snapshots of vorticity fields for a freely falling cylinder in an open domain . The contour level is set from -6 (blue) to 6 (red) with an increment of 0.4.

The instantaneous vorticity fields are presented in Figure 5.11 for the falling cylinder case. Initially symmetric vortex pair forms behind the cylinder in the beginning of falling. After that the numerical error accumulates and breaks the symmetry. At around $tV_t/D = 40$, the flow becomes unsteady and periodic vortex shedding



(a)



(b)

Figure 5.12: Time histories of the vertical and horizontal velocity for the freely rising cylinder $\rho_s/\rho_f = 0.99$.

occurs. The time histories of the velocity of the cylinder are plotted in Figure 5.12. Table 5.1 shows the Strouhal number $St = fD/V_t$ (f is the shedding frequency) and the coefficients of drag and lift. Present results are compared to those of Namkoong *et al.* (2008) and Lacis *et al.* (2016). Good agreements have been obtained.

		C_D	max$\|C_L\|$	St
$\rho_s/\rho_f = 1.01$	**Present**	1.35	0.10	0.189
	Lacis *et al.* (2016)	1.29	0.14	0.17185
	Namkoong *et al.* (2008)	1.23	0.15	0.1684
$\rho_s/\rho_f = 0.99$	**Present**	1.35	0.10	0.189
	Lacis *et al.* (2016)	1.29	0.14	0.17188
	Namkoong *et al.* (2008)	-	-	0.1687

Table 5.1: The drag, lift coefficients and the Strouhal number for the freely falling and rising circular cylinder in an open domain.

5.6.3 Rotating cylinder in a shear flow

To further validate the angular velocity calculation, we consider a circular particle subjected to a shear flow between two walls, as shown in Figure 5.13. The computational domain is [0 cm, 6 cm] $\times$ [0 cm, 4 cm] with the cylinder located at (3 cm, 2 cm). The top and bottom walls are supposed to move horizontally with an opposite velocity $U_{\text{wall}} = \pm 0.02$ cm/s. Periodic boundary conditions are assigned at left and right boundaries. Hence without the particle, the flow would become a linear shear flow between the top and bottom walls with a uniform vorticity of -0.01 s^{-1}. If the particle size is small enough, the angular velocity of the particle is -5×10^{-3} rad/s (Wan & Turek, 2006). Three radius of the particle are considered here, i.e., $R = 0.2$ cm, 0.4 cm, 1.0 cm. The cylinder is allowed to rotate freely around its mass center. A uniform mesh of 300×200 is used to cover the computational domain. The fluid kinematic viscosity is set to 0.01 cm^2/s. The density ratio of the fluid and the solid is $\rho_f/\rho_s = 1.0$.

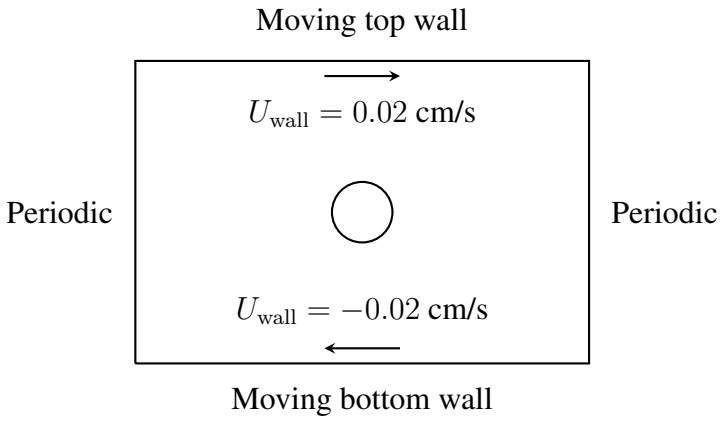
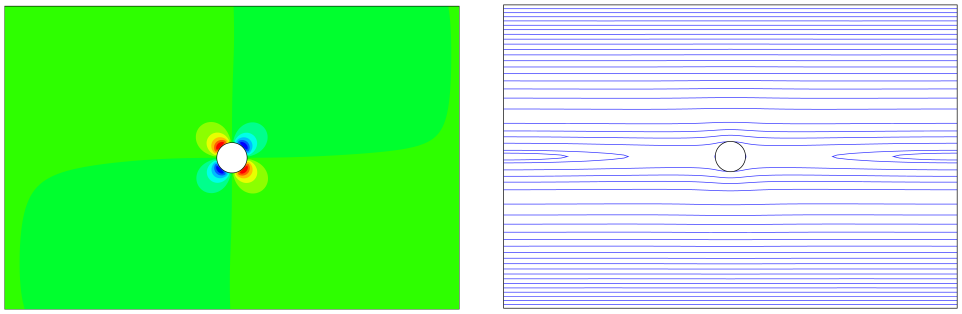
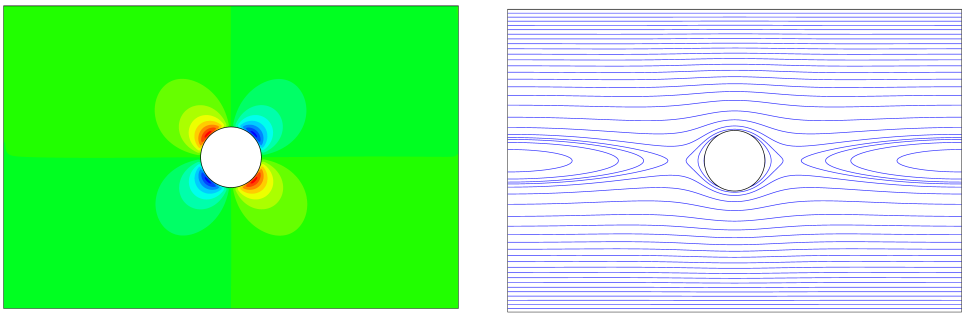


Figure 5.13: Sketch of the rotating cylinder in a shear flow.

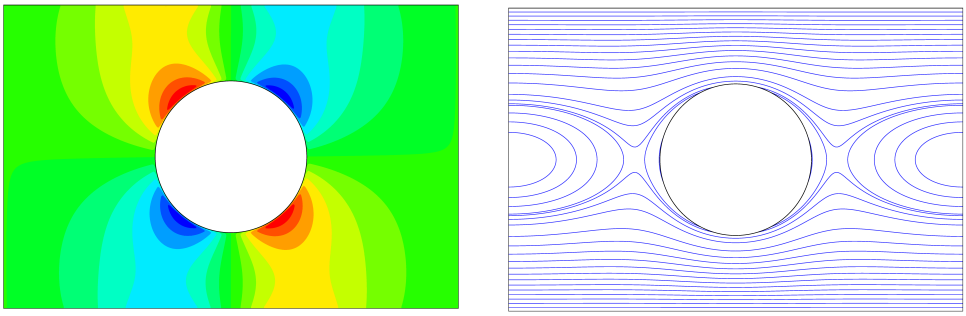
The flow fields of the pressure and the streamfunction are shown in Figure 5.14. The terminal angular velocity at the steady state is compared to the results of Wan & Turek (2006) in Table 5.2, showing a good agreement. As the radius of the circular particle decreases, the terminal angular velocity approaches the value of -5×10^{-3} rad/s.



(a) $R = 0.2$ cm



(b) $R = 0.4$ cm



(c) $R = 1.0$ cm

Figure 5.14: Pressure (left) and streamfunction (right) for a rotating cylinder in a shear flow.

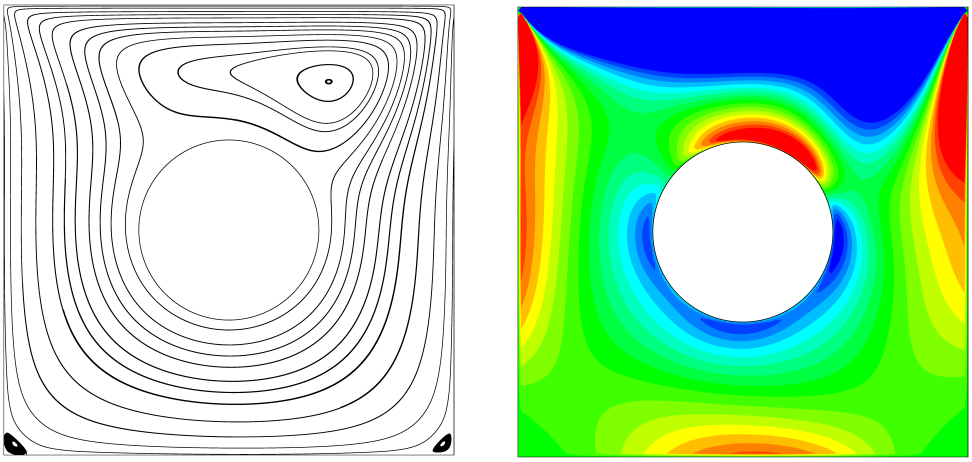
	$R = 0.2$ cm	$R = 0.4$ cm	$R = 1.0$ cm
Present	-4.9818×10^{-3}	-4.9167×10^{-3}	-4.3878×10^{-3}
Wan & Turek (2006)	-4.9584×10^{-3}	-4.8697×10^{-3}	-4.3148×10^{-3}

Table 5.2: Terminal angular velocity at steady state for a rotating cylinder in a shear flow.

5.6.4 Rotating cylinder in a lid-driven cavity flow

In this simulation, we place a cylinder in a lid-driven cavity flow, where the cavity length L is set to 1 cm and the cylinder radius R is 0.2 cm. The cylinder center is fixed at $(0.5L, 0.5L)$ and is free to rotate due to the hydrodynamic force. The moving wall velocity is 1 cm/s. The fluid density is $\rho_f = 1.0$ g/cm^3 and the solid density is $\rho_s = 1.1$ g/cm^3. The viscosity of the fluid is set to $\mu = 0.01$ g/(cm·s).

The calculation is performed on three different meshes, i.e., $h = 1/128$, $h = 1/256$ and $h = 1/512$. The time step is selected based on the CFL number. Figure 5.15 shows the vorticity contour and streamline at final steady state. Clearly three vortices are formed. The vortices center coordinates and the final angular velocity are reported in Table 5.3. We also plot the time history of the rotation angle and the angular velocity in Figure 5.15. Since there is no available experimental and computational data for comparison, we provide this data as a benchmark for future testing.



(a) (b)

Figure 5.15: Streamline (a) and vorticity contour (b) for the rotating cylinder in a lid-driven cavity flow.

	(x_1, y_1)	(x_2, y_2)	(x_3, y_3)	ω_s
$h = 1/256$	(0.7182,0.8263)	(0.0247,0.0246)	(0.9748,0.0253)	-0.4001
$h = 1/128$	(0.7221,0.8296)	(0.0247,0.0245)	(0.9752,0.0250)	-0.3994
$h = 1/64$	(0.7251,0.8320)	(0.0233,0.0230)	(0.9768,0.0237)	-0.3973

Table 5.3: Vortices positions and the final angular velocity of the rotating cylinder in a lid-driven cavity flow. (x_1, y_1) for the upper right vortex, (x_2, y_2) for the lower left vortex, (x_3, y_3) for the lower right vortex.

5.6.5 Elliptical particle sedimentation in a confined channel

In this example, we consider the sedimentation of an elliptical particle in a narrow channel, to demonstrate the ability of current FSI algorithm for handling non-circular object. This example was studied previously by Xia *et al.* (2009) for the boundary effects on the sedimentation mode. Five distinct modes of sedimentation have been found ranging from oscillating, tumbling along the wall, vertical sedimentation, horizontal sedimentation to an inclined mode. In their work, a multi-block lattice Boltzmann method is used and compared to the traditional ALE formulation.

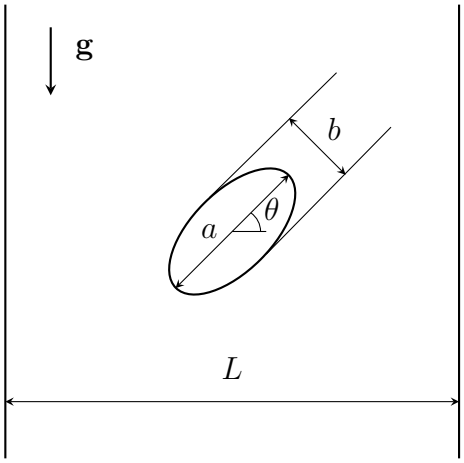


Figure 5.16: Computational domain of the elliptical particle sedimentation problem.

To compare with Xia *et al.* (2009), the computational domain is selected to be $[0, L] \times [0, 7L]$ with $L = 0.4$ cm, as shown in Figure 5.16. The aspect ratio of the ellipse is $\alpha = a/b = 2$, where a and b are the major and minor axes respectively. The blockage ratio is defined as $\beta = L/a = 4$. The density ratio is $\rho_s/\rho_f = 1.1$. The kinematic viscosity of fluid is set to $\nu = 0.01$ cm^2/s. The particle starts falling in a quiescent fluid from the centroid at $(0.5L, 6L)$ with an initial angle of $\pi/4$ to break the symmetry.

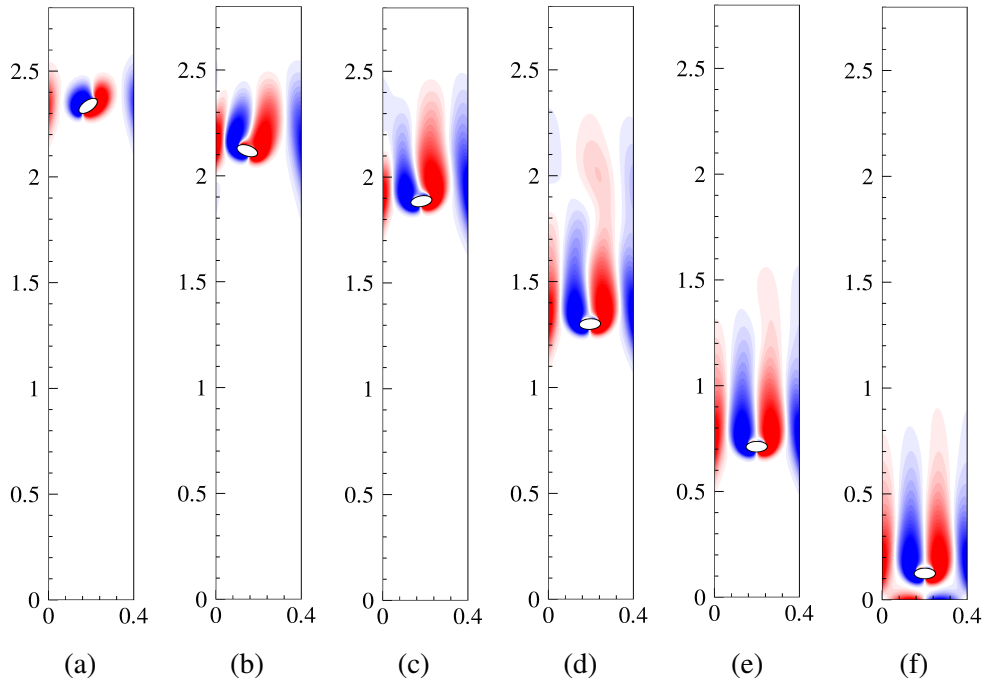


Figure 5.17: Vorticity fields at different times: (a) $t = 0.1$ s ; (b) $t = 0.3$ s; (c) $t = 0.5$ s; (d) $t = 1.0$ s; (e) $t = 1.5$ s; (f) $t = 2.0$ s. The contour levels are set from -15 to 15.

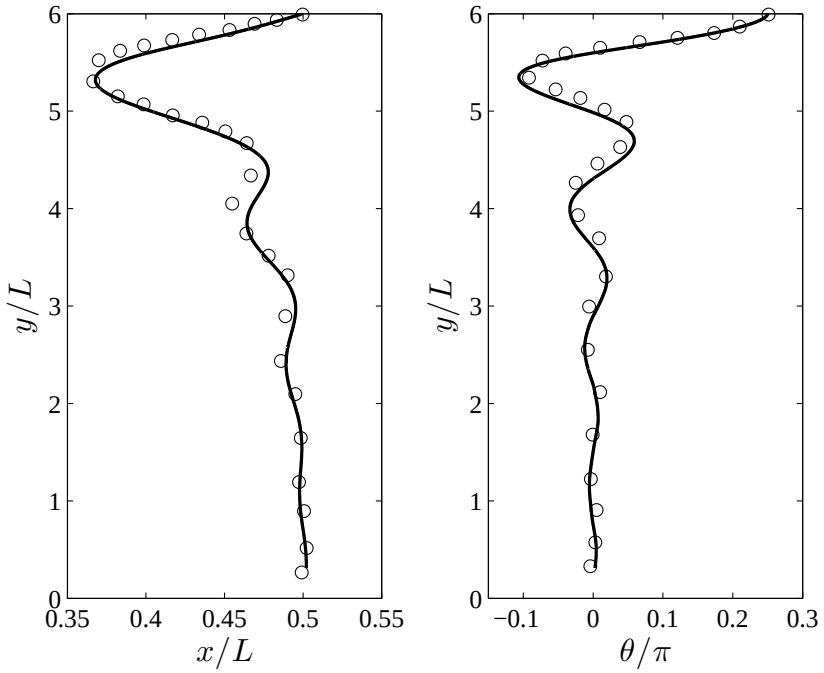


Figure 5.18: Particle trajectory and orientation of the elliptical particle. "—", present results; "∘", results of Xia *et al.* (2009).

No-slip boundary conditions are applied at four boundaries. A uniform mesh is employed with a gird resolution of 0.0027 cm. The time step is chosen such that the CFL condition is satisfied. Figure 5.17 shows the vorticity fields at different times at t = 0.1 s, 0.3 s, 0.5 s, 1.0 s, 1.5 s, 2.0 s. The trajectory and orientations are compared to the results of Xia *et al.* (2009) in Figure 5.18. Good agreements have been obtained.

5.6.6 Flow around a rotating NACA0012 airfoil

The rigid objects simulated in this chapter so far have been the circular or elliptical particles. In this example, we consider the incompressible viscous flow over a NACA0012 airfoil to demonstrate the capacity of current MIBM for handling complicated geometries. The shape of the NACA0012 airfoil is given by

$$Y = \pm 0.6 \cdot (0.2969\sqrt{X} - 0.1260X - 0.3516X^2 + 0.2843X^3 - 0.1015X^4) \quad (5.25)$$

where $X \in (0, 1.009)$ cm. The characteristic length, i.e. the airfoil length, is 1.009 cm. Following Glowinski *et al.* (2001) and Wan & Turek (2006), we select a computational domain of [-4 cm,16 cm] $\times$ [-2 cm, 2cm] with the airfoil centered at (0.42 cm,0), as shown in Figure 5.19.

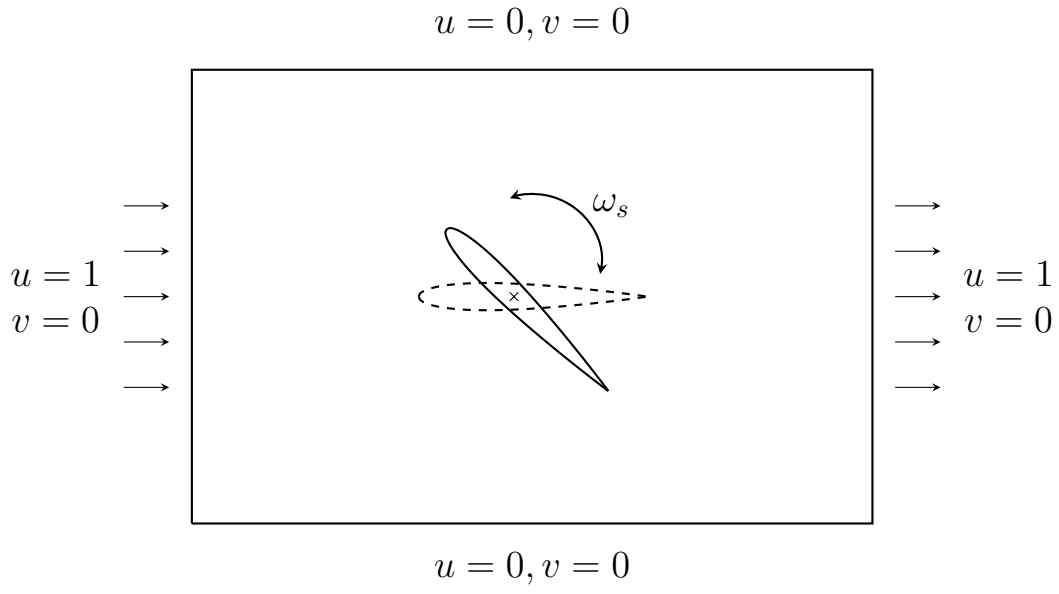
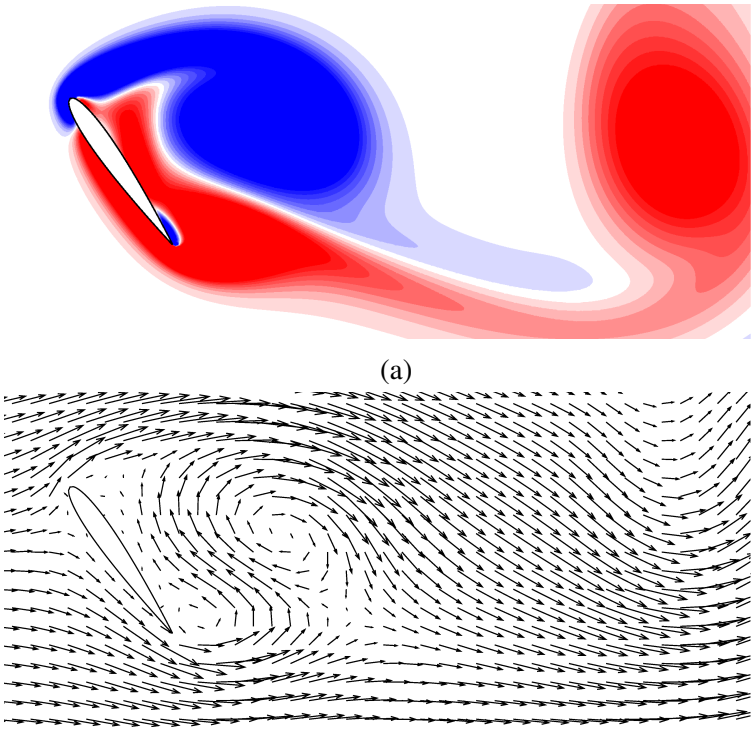


Figure 5.19: Computational domain of the flow past a rotating NACA0012 airfoil. The dashed lines represent the initial position of the airfoil.

The airfoil is fixed at its mass center and is free to rotate due to hydrodynamic forces. The density of fluid is taken as $\rho_f = 1.0$ g/cm^3 and the density of solid

is $\rho_s = 1.1$ g/cm^3 in this simulation. The viscosity of the fluid is $\nu_f = 0.01$ cm^2/s. Initial angular velocity and incident angle of the airfoil are set to zero. The boundary conditions of flow are given as $\mathbf{u} = (0, 0)$ at y=-2 cm, 2 cm and $\mathbf{u} = (1, 0)$ cm/s at x = -4 cm, 16 cm. Those boundary conditions are used in Glowinski *et al.* (2001) and are adopted here in order to compare the results of two methods. The Reynolds number is around 101 based on airfoil length and the maximum inflow speed.
This flow is quite challenging as the leading edge of the airfoil has very small radius of curvature. To resolve the flow near the leading edge, a good resolution of the Cartesian mesh is required. Two sets of grids are chosen here to test the grid sensitivity, namely $h = 1/96$ and $h = 1/64$. The same time step is used in both cases ($\Delta t = 0.002$ s). The resulting CFL numbers are 0.40 and 0.25 respectively.



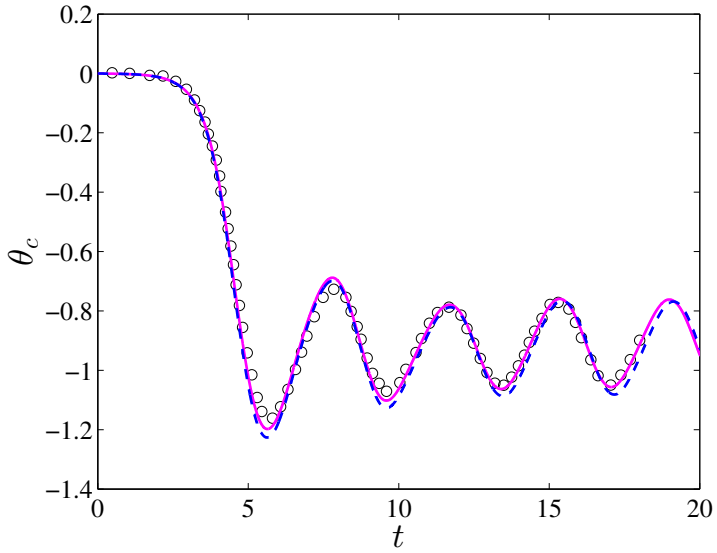
(a)

(b)

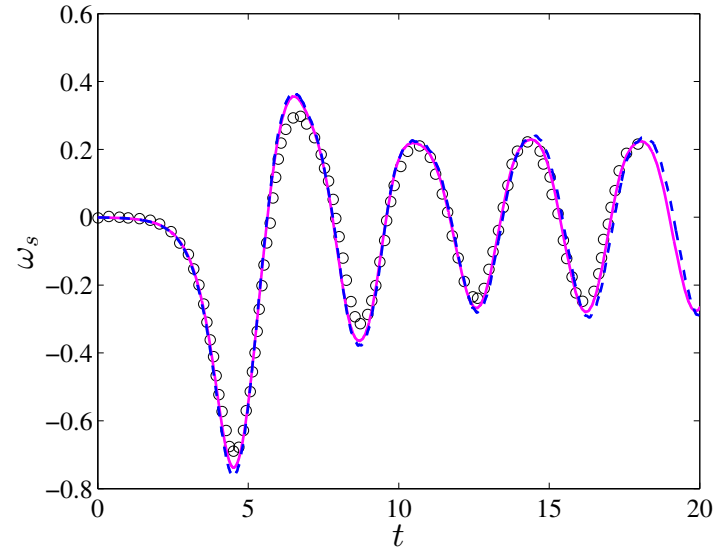
Figure 5.20: Instantaneous vorticity (a) and velocity (b) of the flow over a rotating NACA0012 airfoil.

The flow fields are shown in Figure 5.20. The airfoil keeps a stable position with its broadside perpendicular to the in-flow direction in the beginning and finally reaches a periodic motion of oscillation. The time histories of the rotational angle and the angular velocity are plotted in Figure 5.21. The results are pretty smooth. For comparison, we also include the results of Glowinski *et al.* (2001) in Figure 5.21 obtained by the distributed Lagrange multiplier method. Presents results match well those of Glowinski *et al.* (2001). We observe that the direction of oscillation

depends on the numerical errors. The oscillating direction changes over different calculation parameters. An opposite oscillating case can be found in Wan & Turek (2006). Present direction of oscillation is in accord with Glowinski *et al.* (2001).



(a)



(b)

Figure 5.21: Time histories of the angle (a) and the angular velocity (b) of the rotating NACA0012 airfoil. $h = 1/96$, solid line; $h = 1/64$, dashed line. The measures are in rad and rad/s respectively. The results of Glowinski *et al.* (2001) are marked with circles.

5.6.7 Drafting-kissing-tumbling process of two settling particles

This numerical test consists of two disks with identical density and radius falling freely in a viscous fluid due to gravitation. The two disks are initially at rest in a vertical channel, as shown in Figure 5.22. They are placed at the same horizontal position but have a vertical distance of one disk diameter. The leading (lower) disk creates a pressure drop in its wake, as it falls down. Consequently the trailing (upper) disk falls faster due to the reduced resistance from the fluid and catches up with the leading one. Once in contact, the two disks act as an elongated body falling in a viscous fluid. The elongated body has a tendency to rotate such that the broad side is perpendicular to the flow direction. But the assemblage of two disks are not stable and they separate at a later stage. This phenomenon is well documented as the drafting-kissing-tumbling in the literature (Glowinski *et al.*, 2001; Uhlmann, 2005; Wan & Turek, 2006).

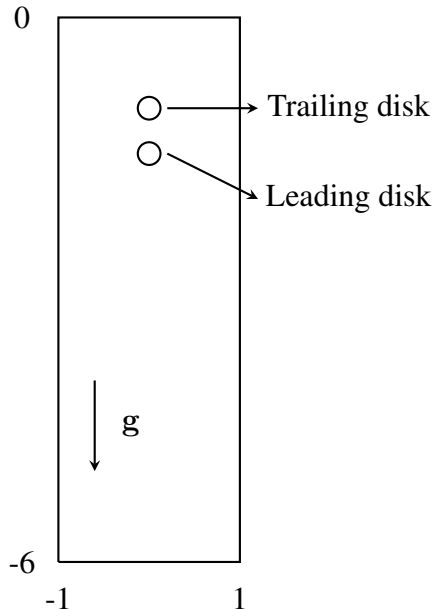


Figure 5.22: Computational domain of the drafting-kissing-tumbling case.

To compare with available data, the physical parameters of the problem are chosen as follows:

- The Computational domain is [-1 cm, 1 cm] $\times$ [-6 cm, 0 cm];
- The diameter of the disks is $D = 0.25$ cm;
- The initial location of the leading disk is (0, -2.0 cm) and (0, -1.5 cm) for the trailing disk;

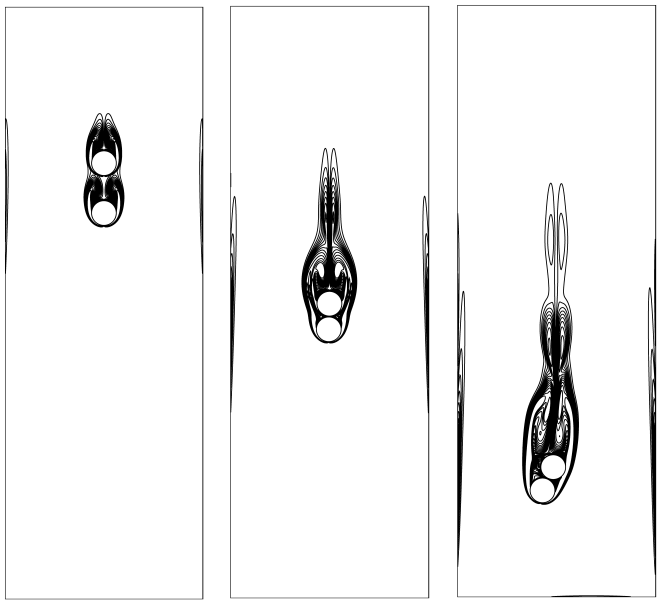


Figure 5.23: Snapshots of disk positions and vorticity contours at three different phases: the drafting (left), the kissing (middle) and the tumbling (right). The contour levels are set from -200 to 200 with 28 steps.

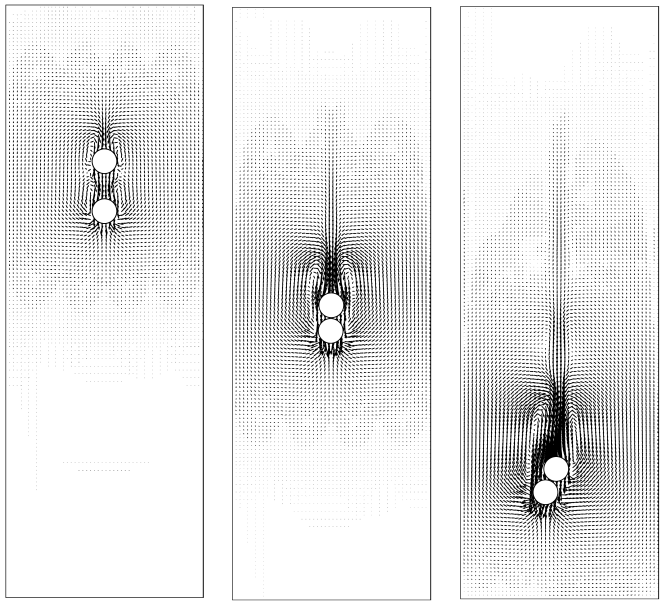


Figure 5.24: Snapshots of the velocity fields of the sedimentation of two disks at three different phases: the drafting (left), the kissing (middle) and the tumbling (right).

- The fluid density is $\rho_f = 1.0$ g/cm^3 and the solid density is $\rho_s = 1.5$ g/cm^3;
- The fluid viscosity is $\nu = 0.01$ cm^2/s;
- The gravitational acceleration is $\mathbf{g} = (0, -981)$ cm/s^2.

The maximum values of the Reynolds number are approximately 480 and 430, respectively. The computational parameters are the following:

- The collision parameter ε_P is 1×10^{-5} and ε_W is 5×10^{-6};
- The force range is $2h$;
- The mesh size is $h = 1/256$, i.e. $D/h = 64$;
- The time step is 1×10^{-4} s.

Figures 5.23 and 5.24 show the snapshots of vorticity and velocity fields. Good agreement has been obtained compared to Glowinski *et al.* (2001). For the vertical velocity, as shown in Figure 5.25, we observe a very close agreement with previous results up to the direct interaction of particles. Discrepancy however is found after the collision, which was also experienced by Uhlmann (2005) and Favier *et al.* (2014). This discrepancy was explained by the intensity of the repulsive force in Favier *et al.* (2014), which is a pre-chosen constant just to prevent from overlapping particles.

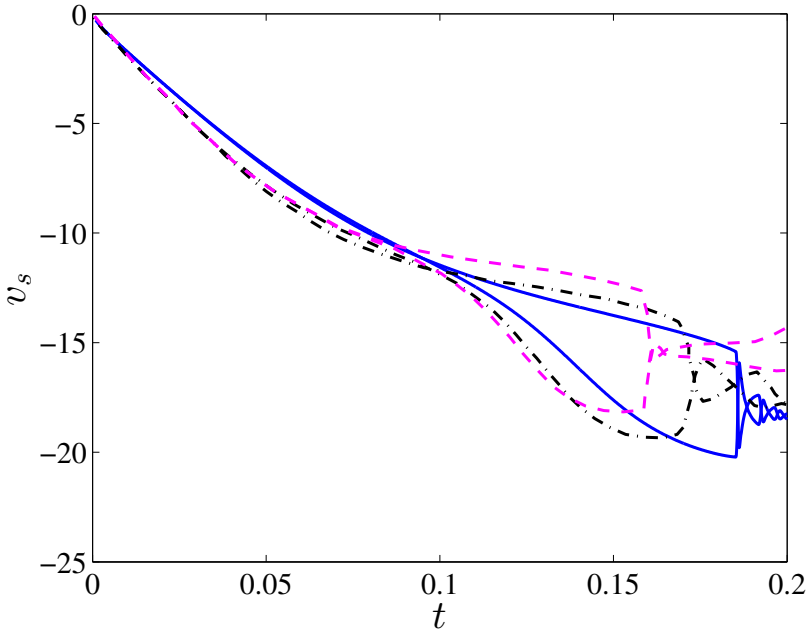


Figure 5.25: Time history of vertical velocity for the drafting-kissing-tumbling case. "—", present results; "- - - -", results of Glowinski *et al.* (2001); "· · · ·", results of Uhlmann (2005).

5.7 Concluding remarks

In this chapter, we have extended the moving immersed boundary method for the study of two-way fluid-structure interaction. To maintain the flexibility of discretization for each subproblem, the fluid and the solid have been solved differently in their own solvers. We note that this partitioned approach is not stable for certain situations. Therefore, we have employed the implicit coupling approach for the fluid-solid interface matching.

The interface matching process has been realised though a fixed point iteration in this chapter. We have considered that solving the pressure Poisson equation is very expensive during interface matching. To reduce the computational time, we have proposed a novel efficient coupling scheme by taking advantage of the projection method. The interface matching only consists of the moving force equation and the solid motion equation, while the pressure Poisson equation is released from this subiteration. The proposed method has been validated through a couple of numerical examples, ranging from circular disk to highly complicated NACA airfoil. The numerical results and the performance efficiency are very satisfactory.

We have also considered the particulate flow in this work, where the interaction happens not only between the fluid and the solid, but also among the particles. We have employed the potential repulsive model to prevent solid bodies from overlapping by using a repulsive force in the normal direction. We have tested this model with the classical drafting-kissing-tumbling case and the results are in good agreement with previous studies.

Conclusions and future work

6.1 Conclusions

A new formulation of the immersed boundary method, termed as the moving immersed boundary method (MIBM), has been developed in this thesis for the interaction of incompressible viscous fluid with solids undergoing large displacements. This novel method can be viewed as an extension of the direct forcing immersed boundary methods of Uhlmann (2005) and Kempe & Fröhlich (2012a), the multidirect forcing immersed boundary method of Luo *et al.* (2007) and Breugem (2012), and the immersed boundary projection method of Taira & Colonius (2007).

The projection/fractional step method has been employed as the fluid solver in present thesis in Chapter 3. The original projection method was found to be only first order time accurate (see Figure 3.7a). We have analyzed the splitting error and then introduced several improved versions to achieve a second order accuracy. The rotational incremental pressure correction projection method was found very satisfactory among the variants of projection methods. It is second order time accurate (see Figure 3.7b), free of artificial boundary layer and very simple to be implemented. It was discretized on a staggered mesh to avoid checker-board effects. Second order spatial accuracy was obtained (see Figure 3.8).

Further various linear system solvers and preconditioners have been tested for solving the discretized equations. Best performance has been obtained when using the conjugate gradient solver with the multi-grid preconditioner. To run the simulation in parallel on high performance computers, we have considered the CPU parallelization with MPI and the GPU parallelization with CUDA. The GPU algorithm has shown an outstanding performance (see Table 3.4).

We have implemented the novel MIBM in the framework of the fractional step method as another operator splitting in Chapter 4. An additional moving force equation has been derived in order to determine the boundary force. Compared to past methods, the MIBM was found very advantageous for the following numerical aspects:

- Accuracy. Compared to the explicit direct forcing IBM of Uhlmann (2005) and Kempe & Fröhlich (2012a), the no-slip boundary condition in MIBM is exactly satisfied at the fluid-solid interface (see Table 4.2).

- Efficiency. Compared to the multidirect forcing IBM of Luo *et al.* (2007) and Breugem (2012), the MIBM is non-iterative and hence efficient. The moving force equation in MIBM is formulated to be symmetric and positive-definite. Many efficient linear system solvers can be used to solve it quickly, like the conjugate gradient method in this thesis (see Figure 4.12 and Table 4.2). Compared to the IBPM of Taira & Colonius (2007), the size of the coefficient matrix is much smaller in MIBM. For moving boundaries, the computation of the coefficient matrix is much less expensive.

- Stability. The boundary force is determined implicitly without any constitutive laws for the rigid body formulation. Large time step hence is allowed in MIBM, compared to the continuous IBMs of Peskin (1972a,b), Goldstein *et al.* (1993) and Saiki & Biringen (1996). The coefficient matrix of the moving force equation is well conditioned and the system is easy to converge, compared to modified Poisson equation in the IBPM of Taira & Colonius (2007).

- Flexibility. Compared to the implicit IBPM of Taira & Colonius (2007), the implicit MIBM can be easily integrated into any fluid solver as a plug-in without reformulating the fluid system.

We have employed the discrete delta function for the field transfer in MIBM. Second order spatial accuracy was found for smooth fields (see Figure 4.13). Different discrete delta functions have been tested for their influences on the results, and we have found that the smoothed functions can effectively suppress unphysical oscillations (see Figure 4.34).
To deal with two-way fluid-structure interaction, the fluid Navier-Stokes equations were coupled to the rigid body dynamic equations through the MIBM in Chapter 5. We have selected the strongly coupled partitioned method for the interface matching with a fixed point iteration. To reduce the computational time, we have removed the pressure Poisson equation from this fixed point iteration. We have shown the ability of present method with numerous cases involving highly complicated solid geometries, as well as solid collisions.

6.2 Future work

Several possible improvements of the present immersed boundary method are summarized as follows.

- First the discrete delta function used in the present thesis will undermine the spatial accuracy of the original fluid solver, in case of flows with discontinuities at the interface. It would be desirable to improve this accuracy by considering alternative interpolation and spreading operators, such as the reproducing kernel particle method (Pinelli *et al.*, 2010), or the moving least square method (Vanella & Balaras, 2009).

- Secondly, many engineering FSI problems occur with the free surface, for example the ocean wave energy conversion. The present work can be extended to these simulations by encompassing the free surface models, such candidates can be the volume of fluid method, or the level set method, or the phase of field method. Three-dimensional implementation is also desirable for real applications with complicated body surface.

- Further, the finite element method can be incorporated into present immersed boundary method, for the simulation of fluid interaction with deformable structures. This has been considered in the immersed finite element method (Wang & Liu, 2004) and the immersed continuum method (Wang, 2007).

- Moreover, to consider multi-body interactions with complex geometry is desirable for the study of particle suspensions. This can be realized by coupling the current immersed boundary method with the discontinuous deformation analysis.

- Finally, the turbulence modeling of the FSI problems with the immersed boundary method is in urgent demand, especially for the RANS models (e.g., $k-\varepsilon$, $k-\omega$, v^2-f, etc.). The direct numerical simulation and the large eddy simulation are too expensive for general engineering problems.

With those improvements, it is hoped that the proposed immersed boundary method will be a competitive tool for solving complex FSI problems.

Bibliography

Altomare, C., Crespo, A.J.C., Domínguez, J.M., Gómez-Gesteira M., Suzuki T., and Verwaest T. (2015): Applicability of Smoothed Particle Hydrodynamics for estimation of sea wave impact on coastal structures. *Coastal Engineering*, Vol. 96, pp. 1-12.

Angot, P., Bruneau, C.H. and Fabrie, P. (1999): A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik*, Vol. 81, pp. 497-520.

Antoci, C. and Gallati, M. and Sibilla, S. (2007): Numerical simulation of fluid-structure interaction by SPH. *Computers & Structures*, Vol. 85, pp. 879-890.

Apte, S.V., Martin, M. and Patankar, N.A. (2009): A numerical method for fully resolved simulation (FRS) of rigid particle–flow interactions in complex flows. *Journal of Computational Physics*, Vol. 228, pp. 2712–2738.

Armaly, B.F., Durst, F., Pereira, J.C.F., and Schönung, B. (1983): Experimental and theoretical investigation of backward-facing step flow. *Journal of Fluid Mechanics*, Vol. 127, pp. 473-496.

Armfield, S. and Street, R. (2002): An analysis and comparison of the time accuracy of fractional-step methods for the Navier-Stokes equations on staggered grids. *International Journal for Numerical Methods in Fluids*, Vol. 38, pp. 255-282.

Babuška, I. and Melenk, J.M. (1997): The partition of unity method. *International Journal for Numerical Methods in Engineering*, Vol. 40, pp. 727-758.

Barrett, R., Berry, M., Chan, T., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and Vorst, H. van der (1994): Templates for the solution of linear systems: Building blocks for iterative methods. *SIAM*, ISBN: 978-0-89871-328-2.

Bell, J.B., Colella, P., and Glaz, H.M. (1989): A second order projection method for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, Vol. 85, pp. 257-283.

Belytschko, T. and Black, T. (1999): Elastic crack growth in finite elements with minimal remeshing. *International Journal for Numerical Methods in Engineering*, Vol. 45, pp. 601-620.

Belytschko, T., Lu, Y.Y., and Gu, L. (1994): Element-free Galerkin methods. *International Journal for Numerical Methods in Engineering*, Vol. 37, pp. 229-256.

Benek, J., Steger, J., and Dougherty, F.C. (1983): A flexible grid embedding technique with application to the Euler equations. *AIAA Paper*, pp. 83-1944.

Beyer, R. P. and LeVeque, R. J. (1992): Analysis of a one-dimensional model for the immersed boundary method. *SIAM Journal on Numerical Analysis*, Vol. 29, pp. 332-364.

Braza, M., Chassaing, P., and Ha Minh, H. (1986): Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder. *Journal of Fluid Mechanics*, Vol. 165, pp. 79-130.

Breugem, W.-P. (2012): A second-order accurate immersed boundary method for fully resolved simulations of particle-laden flows. *Journal of Computational Physics*, Vol. 231, pp. 4469-4498.

Brown, D.L., Cortez, R., and Minion, M.L. (2001): Accurate projection methods for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, Vol. 168, pp. 464-449.

Bruneau, C.-H. and Saad, M. (2006): The 2D lid-driven cavity problem revisited. *Computers & Fluids*, Vol. 35, pp. 326-348.

Canelas, R.B., Dominguez, J.M., Crespo, A.J.C., Gómez-Gesteira, M., and Ferreira, R.M.L. (2015): A smooth Particle Hydrodynamics discretization for the modelling of free surface flows and rigid body dynamics. *International Journal for Numerical Methods in Fluids*, Vol. 78, pp. 581-593.

Causin, P., Gerbeau, J.F., and Nobile, F. (2005): Added-mass effect in the design of partitioned algorithms for fluid-structure problems. *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, pp. 4506-4527.

Chang, W., Giraldo, F., and Perot, B. (2002): Analysis of an exact fractional step method. *Journal of Computational Physics*, Vol. 180, pp. 183-199.

Chen, S. and Doolen, G.D. (1998): Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, Vol. 30, pp. 329-364.

Cheny, Y. and Botella, O. (2010): The LS-STAG method: A new immersed boundary/level-set method for the computation of incompressible viscous flows in complex moving geometries with good conservation properties. *Journal of Computational Physics*, Vol. 229, pp. 1043–1076.

Chorin, A. J. (1967): A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, Vol. 2, pp. 12-26.

Chorin, A. J. (1968): Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, Vol. 22, pp. 745-762.

Chorin, A. J. (1973): Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, Vol. 57, pp. 785-796.

Chow, E. and Saad, Y. (1998): Approximate inverse preconditioners via sparse-sparse iterations. *SIAM Journal on Scientific Computing*, Vol. 19, pp. 995-1023.

Colonius, T. and Taira, K. (2008): A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions. *Computer Methods in Applied Mechanics and Engineering*, Vol. 197, pp. 2131–2146.

Cottet, G.-H. and Koumoutsakos, P.D. (2000): Vortex Methods: Theory and Practice. *Cambridge University Press*, ISBN: 9780521061704.

Coutanceau, M. and Bouard, R. (1977): Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 1. Steady flow. *Journal of Fluid Mechanics*, Vol. 79, pp. 231-256.

Crespo, A.J.C., Domínguez, J.M., Rogers, B.D., Gómez-Gesteira, M., Longshaw, S., Canelas, R., Vacondio, R., Barreiro, A., and García-Feal, O. (2015): DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH). *Computer Physics Communications*, Vol. 187, pp. 204-216.

Deloze, T., Hoarau, Y., and Dušek, J. (2012): Transition scenario of a sphere freely falling in a vertical tube. *Journal of Fluid Mechanics*, Vol. 711, pp. 40-60.

Deriaz, E. (2012): Stability conditions for the numerical solution of convection-dominated problems with skew-symmetric discretizations. *SIAM Journal on Numerical Analysis*, Vol. 50, pp. 1058-1085.

Dong, S. (2015): A convective-like energy-stable open boundary condition for simulations of incompressible flows. *Journal of Computational Physics*, Vol.

302, pp. 300-328.

Dong, S., Karniadakis, G.E., and Chryssostomidis, C. (2014): A robust and accurate outflow boundary condition for incompressible flow simulations on severely-truncated unbounded domains. *Journal of Computational Physics*, Vol. 261, pp. 83-105.

Dong, S. and Shen, J. (2010): An unconditionally stable rotational velocity-correction scheme for incompressible flows. *Journal of Computational Physics*, Vol. 229, pp. 7013-7029.

Dong, S. and Shen, J. (2015): A pressure correction scheme for generalized form of energy-stable open boundary conditions for incompressible flows. *Journal of Computational Physics*, Vol. 291, pp. 254-278.

Duarte, F., Gormaz, R., and Natesan, S. (2004): Arbitrary Lagrangian-Eulerian method for Navier-Stokes equations with moving boundaries. *Computer Methods in Applied Mechanics and Engineering*, Vol. 193, pp. 4819-4836.

Dupuis, A., Chatelain, P. and Koumoutsakos, P. (2008): An immersed boundary-lattice-Boltzmann method for the simulation of the flow past an impulsively started cylinder. *Journal of Computational Physics*, Vol. 227, pp. 4486-4498.

Dütsch, H., Durst, F., Becker, S., and Lienhart, H. (1998): Low-Reynolds-number flow around an oscillating circular cylinder at low Keulegan-Carpenter numbers. *Journal of Fluid Mechanics*, Vol. 360, pp. 249-271.

E, W. and Liu, J.-G. (1995): Projection method I: Convergence and numerical boundary layers. *SIAM Journal on Numerical Analysis*, Vol. 32, pp. 1017-1057.

E, W. and Liu, J.-G. (2003): Gauge method for viscous incompressible flows. *Communications in Mathematical Sciences*, Vol. 1, pp. 317-332.

Erturk, E. (2008): Numerical solutions of 2-D steady incompressible flow over a backward-facing step, Part I: High Reynolds number solutions. *Computers & Fluids*, Vol. 37, pp. 633-655.

Fadlun, E.A., Verzicco, R., Orlandi, P., and Mohd-Yusof, J. (2000): Combined immersed boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics*, Vol. 161, pp. 35-60.

Favier, J., Revell, A. and Pinelli, A. (2014): A Lattice Boltzmann-Immersed Boundary method to simulate the fluid interaction with moving and slender flexible objects. *Journal of Computational Physics*, Vol. 261, pp. 145-161.

Feng, Z.-G. and Michaelides, E.E. (2004): The immersed boundary-lattice Boltzmann method for solving fluid–particles interaction problems. *Journal of Computational Physics*, Vol. 195, pp. 602-628.

Fernández, M.A., Gerbeau, J.-F. and Grandmont, C. (2007): A projection semi-implicit scheme for the coupling of an elastic structure with an incompressible fluid. *International Journal for Numerical Methods in Engineering*, Vol. 69, pp. 794-821.

Ferziger, J.H. and Peric, M. (2002): Computational methods for fluid dynamics. 3rd. *Springer*, ISBN 978-3-642-56026-2.

Förster, C., Wall, W.A., and Ramm, E. (2007): Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, Vol. 196, pp. 1278-1293.

Fries, T.-P. and Matthies, H. G. (2004): Classification and overview of meshfree methods. *Technical University Braunschweig*.

Gerstenberger, A. and Wall, W. A. (2008): An eXtended Finite Element Method-/Lagrange multiplier based approach for fluid-structure interaction. *Computer Methods in Applied Mechanics and Engineering*, Vol. 197, pp. 1699-1714.

Ghia, U., Ghia, K.N., and Shin, C.T. (1982): High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, Vol. 48, pp. 387-411.

Gingold, R. A. and Monaghan, J. J. (1977): Smoothed particle hydrodynamics - Theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, Vol. 181, pp. 375-389.

Glowinski, R., Pan, T.-W., and Périaux, J. (1994): A fictitious domain method for external incompressible viscous flow modeled by Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, Vol. 112, pp. 133-148.

Glowinski, R., Pan, T.-W., Hesla, T.I. and Joseph, D.D. (1999): A distributed Lagrange multiplier/fictitious domain method for particulate flows. *International Journal of Multiphase flow*, Vol. 25, pp. 755-794.

Glowinski, R., Pan, T.W., Hesla, T.I., Joseph, D.D., and Périaux, J. (2001): A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow. *Journal of*

Computational Physics, Vol. 169, pp. 363-426.

Goda, K. (1979): A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows. *Journal of Computational Physics*, Vol. 30, pp. 76-95.

Goldstein, D., Handler, R., and Sirovich, L. (1993): Modeling a no-slip flow boundary with an external force field. *Journal of Computational Physics*, Vol. 105, pp. 354-366.

Gresho, P.M. (1991): Incompressible fluid dynamics: Some fundamental formulation issues. *Annual Review of Fluid Mechanics*, Vol. 23, pp. 413-453.

Gresho, P.M. and Sani, R.L. (1987): On pressure boundary conditions for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, Vol. 7, pp. 1111-1145.

Griebel, M. (1998): Numerical simulation in fluid dynamics: A practical introduction. *SIAM*, ISBN: 978-0-89871-398-5.

Guermond, J.L., Minev, P., and Shen, J. (2006): An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, pp. 6011-6045.

Guermond, J.-L. and Quartapelle, L. (1998): On stability and convergence of projection methods based on pressure Poisson equation. *International Journal for Numerical Methods in Fluids*, Vol. 26, pp. 1039-1053.

Guermond, J.L. and Shen, J. (2003a): A new class of truly consistent splitting schemes for incompressible flows. *Journal of Computational Physics*, Vol. 192, pp. 262-276.

Guermond, J.L. and Shen, J. (2003b): Velocity-correction projection methods for incompressible flows. *SIAM Journal on Numerical Analysis*, Vol. 41, pp. 112-134.

Guilmineau, E. and Queutey, P. (2002): A numerical simulation of vortex shedding from an oscillating circular cylinder. *Journal of Fluids and Structures*, Vol. 16, pp. 773-794.

Guo, Z. and Shu, C. (2013): Lattice Boltzmann method and its applications in engineering. Vol. 3. Advances in computational fluid dynamics. *World scientific*.

Harlow, F.H. and Welch, J.E. (1965): Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids*, Vol.

8, pp. 2182-2189.

Hirsch, C. (2007): Numerical computation of internal and external flows. Volume 1 Fundamentals of numerical discretization. *Elsevier*, ISBN: 978-0-471-92385-5.

Hou, G., Wang, J., and Layton, A. (2012): Numerical methods for fluid-structure interaction - A review. *Communications in Computational Physics*, Vol. 12, pp. 337-377.

Hu, H.H., Joseph, D.D. and Crochet, M.J. (1992): Direct simulation of fluid particle motions. *Theoretical and Computational Fluid Dynamics*, Vol. 3, pp. 285-306.

Hu, H.H. (1996): Direct simulation of flows of solid-liquid mixtures. *International Journal of Multiphase Flow*, Vol. 22, pp. 335-352.

Hu, H.H., Patankar, N.A., and Zhu, M.Y. (2001): Direct Numerical Simulations of Fluid-Solid Systems Using the Arbitrary Lagrangian-Eulerian Technique. *Journal of Computational Physics*, Vol. 169, pp. 427-462.

Huang, W.-X., Shin, S.J., and Sung, H.J. (2007): Simulation of flexible filaments in a uniform flow by the immersed boundary method. *Journal of Computational Physics*, Vol. 226, pp. 2206-2228.

Iaccarino, G. and Verzicco, R. (2003): Immersed Boundary technique for turbulent flow simulations. *Applied Mechanics Reviews*, Vol. 56, pp. 331-347.

Issa, R.I. (1985): Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, Vol. 62, pp. 40-65.

Jasak, H. (2009): Dynamic mesh handling in OpenFOAM. *47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*.

Ji, C., Munjiza, A., and Williams, J. J. R. (2012): A novel iterative direct-forcing immersed boundary method and its finite volume applications. *Journal of Computational Physics*, Vol. 231, pp. 1797-1821.

Jin, G. and Braza, M. (1993): A nonreflecting outlet boundary condition for incompressible unsteady Navier-Stokes calculations. *Journal of Computational Physics*, Vol. 107, pp. 239-253.

Johnston, H. and Liu, J.-G. (2004): Accurate, stable and efficient Navier-Stokes solvers based on explicit treatment of the pressure term. *Journal of Computational Physics*, Vol. 199, pp. 221-259.

Karniadakis, G.E., Israeli M. and Orszag, S.A. (1991): High-order splitting methods for the incompressible Navier-Stokes equations. *Journal of Computational*

Physics, Vol. 97, pp. 414-443.

Kassiotis, C., Ibrahimbegovic, A., Niekamp, R., and Matthies, H. G. (2011): Nonlinear fluid-structure interaction problem. Part I: implicit partitioned algorithm, nonlinear stability proof and validation examples. *Computational Mechanics*, Vol. 47, pp. 305-323.

Kempe, T. and Fröhlich, J. (2012): An improved immersed boundary method with direct forcing for the simulation of particle laden flows. *Journal of Computational Physics*, Vol. 231, pp. 3663-3684.

Kempe, T. and Fröhlich, J. (2012): Collision modelling for the interface-resolved simulation of spherical particles in viscous fluids. *Journal of Fluid Mechanics*, Vol. 709, pp. 445-489.

Khadra, K., Angot, P., Parneix, S., and Caltagirone, J.P. (2000): Fictitious domain approach for numerical modeling of Navier-Stokes equations. *International Journal for Numerical Methods in Fluids* , Vol. 34, pp. 651-684.

Kim, J., Kim, D., and Choi, H. (2001): An immersed-boundary finite-volume method for simulations of flow in complex geometries. *Journal of Computational Physics*, Vol. 171, pp. 132-150.

Kim, D., and Choi, H. (2006): Immersed boundary method for flow around an arbitrarily moving body. *Journal of Computational Physics*, Vol. 212, pp. 662-680.

Kim, J. and Moin, P. (1985): Application of a fractional-step method to incompressible Navier-Stokes equations. *Journal of Computational Physics*, Vol. 59, pp. 308-323.

Koumoutsakos, P. and Leonard, A. (1995): High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *Journal of Fluid Mechanics*, Vol. 296, pp. 1-38.

Küttler, U. and Wall, W. A. (2008): Fixed-point fluid-structure interaction solvers with dynamic relaxation. *Computational Mechanics*, Vol. 43, pp. 61-72.

Lacis, U., Taira K. and Bagheri, S. (2016): A stable fluid-structure-interaction solver for low-density rigid bodies using the immersed boundary projection method. *Journal of Computational Physics*, Vol. 305, pp. 300-318.

Lai, M.-C. and Peskin, C.S. (2000): An immersed boundary method with formal second-order accuracy and reduced Numerical viscosity. *Journal of Computa-*

tional Physics, Vol. 160, pp. 705-719.

Lee, L. and LeVeque, R.J. (2003): An immersed interface method for incompressible Navier-Stokes equations. *SIAM Journal on Scientific Computing*, Vol. 25, pp. 832-856.

Lefrançois, E. and Boufflet, J.-P. (2003): An introduction to fluid-structure interaction: Application to the piston problem. *SIAM Review*, Vol. 52, pp. 747-767.

LeVeque, R.J. (2007): Finite difference methods for ordinary and partial differential equations: Steady-state and time-dependent problems. *SIAM*, ISBN: 978-0-89871-629-0.

LeVeque, R.J. and Li, Z. (1994): The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Journal on Numerical Analysis*, Vol. 31, pp. 1019-1044.

Li, S. and Liu, W.K. (2002): Meshfree and particle methods and their applications. *Applied Mechanics Reviews*, Vol. 55, pp. 1-34.

Li, Y., Paik, K.-J., Xing, T., and Carrica, P.M. (2012): Dynamic overset CFD simulations of wind turbine aerodynamics. *Renewable Energy*, Vol. 37, pp. 285-298.

Li, Z. and Lai, M.-C. (2001): The immersed interface method for the Navier-Stokes equations with singular forces. *Journal of Computational Physics*, Vol. 171, pp. 822-842.

Liu, C., Zheng, X., and Sung, C. H. (1998): Preconditioned Multigrid Methods for Unsteady Incompressible Flows. *Journal of Computational Physics*, Vol. 139, pp. 35-57.

Liu, W.K., Jun, S., Li, S., Adee, J., and Belytschko, T. (1995a): Reproducing kernel particle methods for structural dynamics. *International Journal for Numerical Methods in Engineering*, Vol. 38, pp. 1655-1679.

Liu, W.K., Jun, S., and Zhang, Y.F. (1995b): Reproducing kernel particle methods. *International Journal for Numerical Methods in Engineering*, Vol. 20, pp. 1081-1106.

Lucy, L. B. (1977): A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, Vol. 82, pp. 1013-1024.

Luo, K., Wang, Z., Fan, J., and Cen, K. (2007): Full-scale solutions to particle-laden

flows: Multidirect Forcing and immersed boundary method. *Physical Review E*, Vol. 76, p. 066709.

Matthies, H. G., Niekamp, R., and Steindorf, J. (2006): Algorithms for strong coupling procedures. *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, pp. 2028-2049.

Matthies, H. G. and Steindorf, J. (2002): Partitioned but strongly coupled iteration schemes for nonlinear fluid-structure interaction. *Computers & structures*, Vol. 80, pp. 1991-1999.

Matthies, H. G. and Steindorf, J. (2003): Partitioned strong coupling algorithms for fluid-structure interaction. *Computers & Structures*, Vol. 81, pp. 805-812.

McDonough, J.M. (2007): Lectures in computational fluid dynamics of incompressible flow: Mathematics, algorithms and implementations.

Mittal, S. and Kumar, V. (2001): Flow-induced vibrations of a light circular cylinder at Reynolds numbers 10^3 to 10^4. *Journal of Sound and Vibration*, Vol. 245, pp. 923-946.

Mittal, R., Dong, H., Bozkurttas, M., Najjar, F.M., Vargas, A., and von Loebbecke, A. (2008): A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *Journal of Computational Physics*, Vol. 227, pp. 4825-4852.

Mittal, R. and Iaccarino, G. (2005): Immersed boundary methods. *Annual Review of Fluid Mechanics*, Vol. 37, pp. 239-261.

Mimeau, C., Gallizio, F., Cottet, G.-H. and Mortazavi, I. (2015): Vortex penalization method for bluff body flows. *International Journal for Numerical Methods in Fluids*, Vol. 79, pp. 55-83.

Moës, N., Dolbow, J., and Belytschko, T. (1999): A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, Vol. 46, pp. 131-150.

Mohd-Yosuf, J. (1997): Combined immersed Boundary/B-spline methods for simulation of flow in complex geometries. *Annual Research Briefs*, pp. 317-327.

Monaghan, J. J. (2012): Smoothed particle hydrodynamics and its diverse applications. *Annual Review of Fluid Mechanics*, Vol. 44, pp. 323-346.

Namkoong, K., Yoo, J.Y., and Choi, H.G. (2008): Numerical analysis of two-dimensional motion of a freely falling circular cylinder in an infinite fluid.

Journal of Fluid Mechanics, Vol. 604, pp. 33-53.

Nayroles, B., Touzot, G., and Villon, P. (1992): Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics*, Vol. 10, pp. 307-318.

Nguyen, V.P., Rabczuk, T., Bordas, S., and Duflot, M. (2008): Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation*, Vol. 79, pp. 763-813.

Ni, M.-J. and Abdou, M.A. (2004): Temporal second-order accuracy of SIMPLE-type methods for incompressible unsteady Flows. *Numerical Heat Transfer, Part B*, Vol. 46, pp. 529-548.

Orszag, S.A., Israeli, M., and Deville, M.O. (1986): Boundary conditions for incompressible flows. *Journal of Scientific Computing*, Vol. 1, pp. 75-111.

Patankar, S.V. and Spalding, D.B. (1972): A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, Vol. 15, pp. 1787-1806.

Patankar, S.V. (1980): Numerical heat transfer and fluid flow. *CRC Press*, ISBN: 0-89116-522-3.

Perot, J.B. (1993): An Analysis of the Fractional Step Method. *Journal of Computational Physics*, Vol. 108, pp. 51-58.

Peskin, C.S. (1972a): Flow patterns around heart valves: A digital computer method for solving the equations of motion. *PhD thesis. Albert Einstein College of Medicine, Yeshiva University.*

Peskin, C.S. (1972b): Flow patterns around heart valves: A numerical method. *Journal of Computational Physics*, Vol. 10, pp. 252-271.

Peskin, C.S. (1993): Improved volume conservation in the computation of flows with immersed elastic boundaries. *Journal of Computational Physics*, Vol. 105, pp. 33-46.

Peskin, C.S. (2002): The immersed boundary method. *Acta Numerica*, Vol. 11, pp. 479-517.

Griffith, B.E. and Peskin, C.S. (2005): On the order of accuracy of the immersed boundary method: Higher order convergence rates for sufficiently smooth problems. *Journal of Computational Physics*, Vol. 208, pp. 75-105

Pinelli, A., Naqavi, I. Z., Piomelli, U., and Favier, J. (2010): Immersed-boundary

methods for general finite-difference and finite-volume Navier-Stokes solvers. *Journal of Computational Physics*, Vol. 229, pp. 9073-9091.

Pozrikidis, C. (1992): Boundary integral and singularity methods for linearized viscous flow. *Cambridge University Press*, ISBN: 0-521-40502-5.

Pozrikidis, C. (2002): A practical guide to boundary element methods with the software library BEMLIB. *CRC Press*, ISBN: 1-58488-323-5.

Pozrikidis, C. (2011): Introduction to theoretical and computational fluid dynamics. *Oxford University Press*, ISBN: 978-0-19-975207-2.

Rhie, C.M. and Chow, W.L. (1983): Numerical study of the turbulent flow past an isolated airfoil with trailing edge separation. *AIAA Journal*, Vol. 21, pp. 1525-1532.

Ren, W.W., Wu, J., Shu, C. and Yang, W.M. (2012): A stream function-vorticity formulation-based immersed boundary method and its applications. *International Journal for Numerical Methods in Fluids*, Vol. 70, pp. 627-645.

Roma, A. M., Peskin, C.S., and Berger, M. J. (1999): An adaptive version of the immersed boundary method. *Journal of Computational Physics*, Vol. 153, pp. 509-534.

Saad, Y. (2003): Iterative methods for sparse linear systems. Second edition. *SIAM*, ISBN: 978-0-89871-534-7.

Saiki, E.M. and Biringen, S. (1996): Numerical Simulation of a Cylinder in Uniform Flow: Application of a Virtual Boundary Method. *Journal of Computational Physics*, Vol. 123, pp. 450-465.

Sani, R.L. and Gresho, P. (1994): Resume and remarks on the open boundary condition minisymposium. *International Journal for Numerical Methods in Fluids*, Vol. 18, pp. 983-1008.

Seibold, B. (2008): A compact and fast Matlab code solving the incompressible Navier-Stokes equations on rectangular domains. *MIT*.

Shen, Z., Wan, D., and Carrica, P.M. (2015): Dynamic overset grids in OpenFOAM with application to KCS self-propulsion and maneuvering. *Ocean Engineering*, Vol. 108, pp. 287-306.

Song, M.D., Lefrançois, E., and Rachik, M. (2013): A partitioned coupling scheme extented to structures interacting with high-density fluid flows. *Computers & Fluids*, Vol. 84, pp. 190-202.

Sotiropoulos, F. and Yang, X. (2014): Immersed boundary methods for simulating fluid-structure interaction. *Progress in Aerospace Sciences*, Vol. 65, pp. 1-21.

Souli, M., Ouahsine, A., and Lewin, L. (2000): ALE formulation for fluid-structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, Vol. 190, pp. 659-675.

Taira, K. and Colonius, T. (2007): The immersed boundary method: A projection approach. *Journal of Computational Physics*, Vol. 225, pp. 2118-2137.

Tang, H.S., Jones, S.C., and Sotiropoulos, F. (2003): An overset-grid method for 3D unsteady incompressible flows. *Journal of Computational Physics*, Vol. 191, pp. 567-600.

Témam, R. (1968): Une méthode d'approximation de la solution des équations de Navier-Stokes. *Bulletin de la Société Mathématique de France*, Vol. 96, pp. 115-152.

Témam, R. (1969): Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (II). *Archive for Rational Mechanics and Analysis*, Vol. 33, pp. 377-385.

Thompson, H.D., Webb, B.W., and Hoffman, J.D. (1985): The cell Reynolds number myth. *International Journal for Numerical Methods in Fluids*, Vol. 5, pp. 305-310.

Timmermans, L.J.P., Minev, P.D., and Van De Vosse, F.N. (1996): An approximate projection scheme for incompressible flow using spectral elements. *International Journal for Numerical Methods in Fluids*, Vol. 22, pp. 673-688.

Toja-Silva, F., Favier, J., and Pinelli, A. (2014): Radial basis function (RBF)-based interpolation and spreading for the immersed boundary method. *Computers & Fluids*, Vol. 105, pp. 66-75.

Tritton, D.J. (1959): Experiments on the flow past a circular cylinder at low Reynolds numbers. *Journal of Fluid Mechanics*, Vol. 6, pp. 547-567.

Tseng, Y.-H. and Ferziger, J. (2003): A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics*, Vol. 192, pp. 593-623.

Udaykumar, H.S., Mittal, R., Rampunggoon, P. and Khanna, A. (2001): A sharp interface Cartesian grid method for simulating flows with complex moving boundaries. *Journal of Computational Physics*, Vol. 174, pp. 345-380.

Uhlmann, M. (2005): An immersed boundary method with direct forcing for the simulation of particulate flows. *Journal of Computational Physics*, Vol. 209, pp. 448-476.

Van Doormaal, J.P. and Raithby, G.D. (1984): Enhancements of the SIMPLE method for predicting incompressible fluid flows. *Numerical Heat Transfer*, Vol. 7, pp. 147-163.

Van Kan, J. (1986): A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, pp. 870-891.

Vanella, M. and Balaras, E. (2009): A moving-least-squares reconstruction for embedded boundary formulations. *Journal of Computational Physics*, Vol. 228, pp. 6617-6628.

Versteeg, H.K. and Malalasekera, W. (2007): An introduction to computational fluid dynamics: The finite volume method. Second edition. *Pearson Education*, ISBN: 978-0-13-127498-3.

Wagner, G.J., Moës, N., Liu, W.K. and Belytschko, T. (2001): The extended finite element method for rigid particles in Stokes flow. *International Journal for Numerical Methods in Engineering*, Vol. 51, pp. 293-313.

Wan, D. and Turek, S. (2006): Direct numerical simulation of particulate flow via multigrid FEM technique and the fictitious boundary method. *International Journal for Numerical Methods in Fluids*, Vol. 51, pp. 531-566.

Wang, C. and Liu, J.-G. (2000): Convergence of gauge method for incompressible flow. *Mathematics of Computation*, Vol. 69, pp. 1385-1407.

Wang, S. and Zhang, X. (2011): An immersed boundary method based on discrete stream function formulation for two- and three-dimensional incompressible flows. *Journal of Computational Physics*, Vol. 230, pp. 3479-3499.

Wang, Z.J. (2000): Two dimensional mechanism for insect hovering. *Physical Review letters*, Vol. 10, pp. 2216-2219.

Wang, X.S. (2007): An iterative matrix-free method in implicit immersed boundary/continuum methods. *Computers & Structures*, Vol. 85, pp. 739-748.

Wang, X.S., Zhang, L.T., and Liu, W.K. (2009): On computational issues of immersed finite element methods. *Journal of Computational Physics*, Vol. 228, pp. 2535-2551.

Wang, Y., Shu, C., Teo, C.J., Wu, J. (2015): An immersed boundary-lattice Boltzmann flux solver and its applications to fluid-structure interaction problems. *Journal of Fluids and Structures*, Vol. 54, pp. 440-465.

Wang, X. and Liu, W.K. (2004): Extended immersed boundary method using FEM and RKPM. *Computer Methods in Applied Mechanics and Engineering*, Vol. 193, pp. 1305-1321.

Williamson, C.H.K. (1989): Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers. *Journal of Fluid Mechanics*, Vol. 206, pp. 579-627.

Wu, J. and Shu, C.(2009): Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its applications. *Journal of Computational Physics*, Vol. 228, pp. 1963-1979.

Wu, J. and Shu, C.(2010): An improved immersed boundary-lattice Boltzmann method for simulating three-dimensional incompressible flows. *Journal of Computational Physics*, Vol. 229, pp. 5022-5042.

Xia, Z., Connington, K.W., Rapaka, S., Yue, P., Feng, J.J., and Chen, S. (2009): Flow patterns in the sedimentation of an elliptical particle. *Journal of Fluid Mechanics*, Vol. 625, pp. 249-272.

Xu, S and Wang, Z.J. (2006): An immersed interface method for simulating the interaction of a fluid with moving boundaries. *Journal of Computational Physics*, Vol. 216, pp. 454–493.

Yang, X., Zhang, X., Li, Z. and He, G.-W. (2009): A smoothing technique for discrete delta functions with application to immersed boundary in moving boundary simulations. *Journal of Computational Physics*, Vol. 228, pp. 7821-7836.

Yang, J. and Stern, F. (2012): A simple and efficient direct forcing immersed boundary framework for fluid-structure interactions. *Journal of Computational Physics*, Vol. 231, pp. 5029-5061.

Ye, T., Mittal, R., Udaykumar, H.S. and Shyy, W. (1999): An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. *Journal of Computational Physics*, Vol. 156, pp. 209-240.

Zhang, L., Gerstenberger, A., Wang, X., and Liu, W.K. (2004): Immersed finite element method. *Computer Methods in Applied Mechanics and Engineering*, Vol. 193, pp. 2051-2067.

Stability analysis

It is necessary to assess the stability of a numerical scheme in order to ensure that the calculation will not explode. The stability of a numerical scheme is closely linked to the numerical error. A finite difference scheme is stable if the error remains bounded as the calculation continues.

The von Neumann method provides a good tool for investigating the stability conditions of linear problems with constant coefficients (Hirsch, 2007). We first investigate the stability of simple equations, such as the diffusion equation, the convection equation and the transport (convection-diffusion) equation, and then extend the conclusion to the Navier-Stokes equations.

A.1 Von Neumann stability condition

Let u_i^n be the computed solution of the difference equation with a finite precision at time level n at node i, which can be decomposed into the Fourier series in space, on a interval $[-L, L]$, as

$$u_i^n = \sum_{m=-M}^{M} E_m^n e^{Ik \cdot i\Delta x} \tag{A.1}$$

where $I = \sqrt{-1}$, $M = L/\Delta x$. $k = \pi m/L$, E_m^n are the wave number and the amplitude of the k-th harmonic respectively. Therefore, the numerical scheme is stable if the amplitude of any harmonic E^n does not increase in time, i.e.,

$$|G| \equiv \left| \frac{E^{n+1}}{E^n} \right| \leqslant 1 \tag{A.2}$$

where $G \equiv E^{n+1}/E^n$ is the amplification factor.

A.2 Stability analysis for diffusion equation

A.2.1 Explicit FTCS scheme

First we consider the one-dimensional diffusion equation

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} \tag{A.3}$$

where a is a constant. Applying the explicit forward in time, central in space (FTCS) scheme gives

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = a\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{A.4}$$

Considering one single harmonic $E^n e^{Ik\cdot i\Delta x}$ for u_i^n, we have

$$\frac{E^{n+1} - E^n}{\Delta t}e^{Ik\cdot i\Delta x} = a\frac{E^n e^{Ik\cdot(i+1)\Delta x} - 2E^n e^{Ik\cdot i\Delta x} + E^n e^{Ik\cdot(i-1)\Delta x}}{\Delta x^2} \tag{A.5}$$

Dividing by $E^n e^{Ik\cdot i\Delta x}$,

$$\frac{G-1}{\Delta t} = a\frac{e^{Ik\Delta x} + e^{-Ik\Delta x} - 2}{\Delta x^2} \tag{A.6}$$

or

$$\begin{aligned} G &= 1 + \frac{a\Delta t}{\Delta x^2}(e^{Ik\Delta x} + e^{-Ik\Delta x} - 2) \\ &= 1 + \frac{a\Delta t}{\Delta x^2}(2cos(k\Delta x) - 2) \\ &= 1 + \frac{a\Delta t}{\Delta x^2}(-4sin^2(\frac{k\Delta x}{2})) \\ &= 1 - \frac{4a\Delta t}{\Delta x^2}sin^2(\frac{k\Delta x}{2}) \end{aligned} \tag{A.7}$$

The stability condition $|G| \leqslant 1$ implies that the FTCS scheme is stable under the condition

$$\frac{a\Delta t}{\Delta x^2} \leqslant \frac{1}{2} \tag{A.8}$$

or

$$\Delta t \leqslant \frac{\Delta x^2}{2a} \tag{A.9}$$

For the two-dimensional problem

$$\frac{\partial u}{\partial t} = a(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \tag{A.10}$$

The FTCS scheme yields

$$\frac{u_{i,j}^{n+1}-u_{i,j}^{n}}{\Delta t}=a(\frac{u_{i+1,j}^{n}-2u_{i,j}^{n}+u_{i-1,j}^{n}}{\Delta x^2}+\frac{u_{i,j+1}^{n}-2u_{i,j}^{n}+u_{i,j-1}^{n}}{\Delta y^2}) \tag{A.11}$$

Again we perform the same analysis as in the one-dimensional case and the amplification factor is found to be

$$\begin{aligned} G &= 1+\frac{a\Delta t}{\Delta x^2}(e^{Ik\Delta x}+e^{-Ik\Delta x}-2)+\frac{a\Delta t}{\Delta y^2}(e^{Ik\Delta y}+e^{-Ik\Delta y}-2) \\ &= 1-\frac{4a\Delta t}{\Delta x^2}sin^2(\frac{k\Delta x}{2})-\frac{4a\Delta t}{\Delta y^2}sin^2(\frac{k\Delta y}{2}) \end{aligned} \tag{A.12}$$

Requiring $|G|\leqslant 1$

$$-1\leqslant 1-\frac{4a\Delta t}{\Delta x^2}sin^2(\frac{k\Delta x}{2})-\frac{4a\Delta t}{\Delta y^2}sin^2(\frac{k\Delta y}{2})\leqslant 1 \tag{A.13}$$

then we have

$$\frac{a\Delta t}{\Delta x^2}sin^2(\frac{k\Delta x}{2})+\frac{a\Delta t}{\Delta y^2}sin^2(\frac{k\Delta y}{2})\leqslant\frac{1}{2} \tag{A.14}$$

Since

$$\frac{a\Delta t}{\Delta x^2}sin^2(\frac{k\Delta x}{2})+\frac{a\Delta t}{\Delta y^2}sin^2(\frac{k\Delta y}{2})\leqslant\frac{a\Delta t}{\Delta x^2}+\frac{a\Delta t}{\Delta y^2} \tag{A.15}$$

the stability condition now becomes

$$\frac{a\Delta t}{\Delta x^2}+\frac{a\Delta t}{\Delta y^2}\leqslant\frac{1}{2} \tag{A.16}$$

or

$$\Delta t\leqslant\frac{1}{2a}(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2})^{-1} \tag{A.17}$$

We can further extend to the N-dimensional problem, and the stability condition is

$$\sum_{p=1}^{N}\frac{a\Delta t}{\Delta x_p^2}\leqslant\frac{1}{2} \tag{A.18}$$

or

$$\Delta t\leqslant\frac{1}{2a}(\sum_{p=1}^{N}\frac{1}{\Delta x_p^2})^{-1} \tag{A.19}$$

If the same mesh size h is used in all directions, then we have

$$\Delta t \leqslant \frac{h^2}{2aN} \tag{A.20}$$

We can conclude that this stability condition becomes more restrictive as the dimension increases. Halving the mesh size means that the time step should be reduced by a factor of four in all cases.

A.2.2 Implicit BTCS scheme

This stability condition can be eliminated by using the backward in time, central in space (BTCS) scheme. The one-dimensional discretized equation is then written as

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = a\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} \tag{A.21}$$

The amplification factor is found to be

$$G = \left(1 + 4\frac{a\Delta t}{\Delta x^2} sin^2(\frac{k\Delta x}{2})\right)^{-1} \leqslant 1 \tag{A.22}$$

In other words, the BTCS implicit scheme is unconditionally stable. This conclusion holds for the N-dimension, since

$$G = \left(1 + \sum_{p=1}^{N} 4\frac{a\Delta t}{\Delta x_p^2} sin^2(\frac{k\Delta x_p}{2})\right)^{-1} \leqslant 1 \tag{A.23}$$

A.3 Stability analysis for convection equation

A.3.1 Explicit FTCS scheme

In this subsection we consider the one-dimensional convection equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{A.24}$$

where c is a constant here, which can be related to the convection velocity. The FTCS scheme yields

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} = 0 \tag{A.25}$$

The resulting amplification factor is

$$\begin{aligned} G &= 1 - \frac{c\Delta t}{2\Delta x}(e^{Ik\Delta x} - e^{-Ik\Delta x}) \\ &= 1 - I\frac{c\Delta t}{\Delta x}sin(k\Delta x) \end{aligned} \tag{A.26}$$

It is easy to find that

$$|G|^2 = 1 + \left(\frac{c\Delta t}{\Delta x}\right)^2 sin^2(k\Delta x) \geqslant 1 \tag{A.27}$$

Hence the FTCS scheme for the convection equation is always unstable. This conclusion holds for the N-dimension, since

$$G = 1 - I\sum_{p=1}^{N}\frac{c_{x_p}\Delta t}{\Delta x_p}sin(k\Delta x_p) \tag{A.28}$$

and

$$|G|^2 = 1 + \left[\sum_{p=1}^{N}\frac{c_{x_p}\Delta t}{\Delta x_p}sin(k\Delta x_p)\right]^2 \geqslant 1 \tag{A.29}$$

A.3.2 Explicit upwind scheme

To overcome this problem, we apply the explicit upwind scheme, specifically the explicit forward in time backward in space (FTBS) scheme in this case

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0 \tag{A.30}$$

The amplification factor now becomes

$$\begin{aligned} G &= 1 - \frac{c\Delta t}{\Delta x}(1 - e^{-Ik\Delta x}) \\ &= 1 + \frac{c\Delta t}{\Delta x}(cos(k\Delta x) - 1) - I\frac{c\Delta t}{\Delta x}sin(k\Delta x) \end{aligned} \tag{A.31}$$

Therefore,

$$\begin{aligned} |G|^2 &= 1 + 2\frac{c\Delta t}{\Delta x}\left(cos(k\Delta x) - 1\right) + \left(\frac{c\Delta t}{\Delta x}\right)^2 (cos(k\Delta x) - 1)^2 + \left(\frac{c\Delta t}{\Delta x}\right)^2 sin^2(k\Delta x) \\ &= 1 - 4\frac{c\Delta t}{\Delta x}(1 - \frac{c\Delta t}{\Delta x})sin^2(\frac{k\Delta x}{2}) \end{aligned} \tag{A.32}$$

Provided $c > 0$, the von Neumann stability condition $|G| \leqslant 1$ implies that

$$C = \frac{c\Delta t}{\Delta x} \leqslant 1 \tag{A.33}$$

where C is the Courant number. This condition means that the flow quantity can travel at most one grid spacing in a single time step, which is also known as the Courant–Friedrichs–Lewy (CFL) condition. Note that this condition only holds for $c > 0$. If c is negative, this constraint is no longer guaranteed and the FTBS scheme is unconditionally unstable. In that case the forward in time forward in space (FTFS) scheme should be used and a similar condition can be found

$$-1 \leqslant C < 0 \tag{A.34}$$

But the FTFS scheme is unstable when $c > 0$. In other words, we should apply the FTBS scheme when c is positive and use the FTFS scheme when c is negative. In both cases, the following condition should be satisfied

$$|C| \leqslant 1 \tag{A.35}$$

The upwind scheme is very effective for circumventing numerical oscillations, but it is only first order accurate in space and introduces numerical diffusion or artificial viscosity to the solution (LeVeque, 2007; Pozrikidis, 2011). This numerical diffusion is a non-physical damping that aides the stability, compared to the unconditionally unstable FTCS scheme. However it smears out the sharp gradients and distinguishes the result from the exact solution.

A.3.3 Implicit BTCS scheme

To achieve an unconditionally stable scheme, we turn to the implicit BTCS scheme, which yields

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{2\Delta x} = 0 \tag{A.36}$$

The corresponding amplification factor is

$$\begin{aligned} G &= \frac{1}{1 + ICsin(k\Delta x)} \\ &= \frac{1 - ICsin(k\Delta x)}{1 + C^2sin^2(k\Delta x)} \\ &= \frac{1}{1 + C^2sin^2(k\Delta x)} - I\frac{Csin(k\Delta x)}{1 + C^2sin^2(k\Delta x)} \end{aligned} \tag{A.37}$$

This unconditional stability is confirmed by considering that

$$|G|^2 = \frac{1}{1 + C^2 sin^2(k\Delta x)} \leqslant 1 \tag{A.38}$$

This conclusion holds for N-dimension, since

$$|G|^2 = \frac{1}{1 + \left(\sum_{p=1}^{N} C_{x_p} sin(k\Delta x_p)\right)^2} \leqslant 1 \tag{A.39}$$

A.4 Stability analysis for convection-diffusion equation

A.4.1 Explicit FTCS scheme

Now we consider the one-dimensional convection-diffusion (transport) equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = a\frac{\partial^2 u}{\partial x^2} \tag{A.40}$$

and we assume $c > 0$ henceforth. The amplification factor for the explicit FTCS scheme is

$$G = 1 - 2D\,(1 - cos(k\Delta x)) - ICsin(k\Delta x) \tag{A.41}$$

where

$$D = \frac{a\Delta t}{\Delta x^2} \tag{A.42}$$

The stability condition $|G| \leqslant 1$ gives

$$\begin{aligned} |G|^2 &= (1 - 2D(1 - cos(k\Delta x)))^2 + C^2 sin^2(k\Delta x) \\ &= (1 - 2D(1 - cos(k\Delta x)))^2 + C^2\left(1 - cos^2(k\Delta x)\right) \\ &= (1 - 2D(1 - q))^2 + C^2\left(1 - q^2\right) \\ &\leqslant 1 \end{aligned} \tag{A.43}$$

where $q = cos(k\Delta x)$. This is equivalent to finding C, D such that

$$\begin{aligned} y(q) &= (1 - 2D(1 - q))^2 + C^2\left(1 - q^2\right) - 1 \\ &= (4D^2 - C^2)q^2 + (4D - 8D^2)q + (4D^2 + C^2 - 4D) \\ &\leqslant 0 \end{aligned} \tag{A.44}$$

is satisfied for $-1 \leqslant q \leqslant 1$. Therefore we can find the following expression for C

and D (Thompson *et al.*, 1985; Hirsch, 2007; Pozrikidis, 2011)

$$C^2 \leqslant 2D \leqslant 1 \tag{A.45}$$

which corresponds to the following criteria

$$\Delta t \leqslant \frac{\Delta x^2}{2a} \tag{A.46}$$

and

$$\Delta t \leqslant \frac{2a}{c^2} \tag{A.47}$$

It is easy to find that the scheme becomes unconditionally unstable when a approaches zero. This is just the case for the FTCS scheme used in the pure convection equation. Hence the presence of diffusion stabilizes the discretization. The same conclusion has been found to the upwind scheme in the pure convection case where numerical diffusions are added automatically.

Most importantly, the CFL condition $C \leqslant 1$ can be implied by (A.45) but is only necessary and certainly not sufficient (Hirsch, 2007).

The condition (A.47) can be also rewritten as

$$Re_c C \leqslant 2 \tag{A.48}$$

where $Re_c \equiv C/D = c\Delta x/a$ is the cell Reynolds number if a refers to the viscosity. The cell Reynolds number is equivalent to the Péclet number if a takes the thermal diffusivity.

Next we consider the two-dimensional problems

$$\frac{\partial u}{\partial t} + c_x \frac{\partial u}{\partial x} + c_y \frac{\partial u}{\partial y} = a\frac{\partial^2 u}{\partial x^2} + a\frac{\partial^2 u}{\partial y^2} \tag{A.49}$$

By performing the stability analysis for FTCS scheme, the amplification factor is found to be

$$G = \left[1 + 2D_x\left(cos(\theta_x) - 1\right) + 2D_y\left(cos(\theta_y) - 1\right)\right] - I\left[C_x sin(\theta_x) + C_y sin(\theta_y)\right] \tag{A.50}$$

where $\theta_x = k\Delta x$ and $\theta_y = k\Delta y$. Assuming $\theta = \theta_x = \theta_y$, the von Neumann stability condition implies that

$$|G|^2 = \left[1 + 2(D_x + D_y)\left(cos\theta - 1\right)\right]^2 + (C_x + C_y)^2(1 - cos^2\theta) \leqslant 1 \tag{A.51}$$

is true for all values of θ. By analogy to (A.45), the resulting stability condition is

$$0 \leqslant (C_x + C_y)^2 \leqslant 2(D_x + D_y) \leqslant 1 \tag{A.52}$$

which indicates that

$$0 \leqslant 2(D_x + D_y) \leqslant 1 \tag{A.53}$$

and

$$0 \leqslant C_x^2/2D_x + C_y^2/2D_y \leqslant 1 \tag{A.54}$$

or in terms of time step

$$\Delta t \leqslant \left(\frac{2a}{\Delta x^2} + \frac{2a}{\Delta y^2} \right)^{-1} \tag{A.55}$$

and

$$\Delta t \leqslant \left(\frac{c_x^2}{2a} + \frac{c_y^2}{2a} \right)^{-1} \tag{A.56}$$

The generalization to N-dimension can be made analogously (Pozrikidis, 2011)

$$\Delta t \leqslant \left(\sum_{p=1}^{N} \frac{2a}{\Delta x_p^2} \right)^{-1} \quad \text{and} \quad \Delta t \leqslant \left(\sum_{p=1}^{N} \frac{c_{x_p}^2}{2a} \right)^{-1} \tag{A.57}$$

Notice that the condition is obtained by assuming $\theta_x = \theta_y$. This simplification leads to a necessary condition that is generally not sufficient (Thompson *et al.*, 1985).

A.4.2 Explicit FTBSCS scheme

Assuming c is positive, we use the explicit forward difference in time, backward difference for the convection and centered difference for the diffusion (FTBSCS), which gives

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_i^n - u_{i-1}^n}{\Delta x} = a\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{A.58}$$

Performing the von Neumann method, we derive the amplification factor

$$G = 1 - (C + 2D) + (C + 2D)cos\theta + ICsin\theta \tag{A.59}$$

To ensure stability, the following condition has to be satisfied

$$C^2 \leqslant C + 2D \leqslant 1 \tag{A.60}$$

If c is negative, the FTFSCS scheme should be used. For both cases, the method is stable if

$$C^2 \leqslant |C| + 2D \leqslant 1 \tag{A.61}$$

is satisfied. However, the numerical diffusion associated with the upwind difference can be significant and the overall method can be only first order accurate in space. The stability condition in two dimensions can be found in Pozrikidis (2011).

A.4.3 Explicit hybrid scheme

By combining the upwind scheme and central difference scheme for the convection, we obtain

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + \frac{1}{2}c\left[(1+\beta)\frac{u_i^n - u_{i-1}^n}{\Delta x} + (1-\beta)\frac{u_{i+1}^n - u_i^n}{\Delta x}\right] = a\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{A.62}$$

where β controls the weight between upwind difference and central difference. If $\beta = 0$, the FTCS scheme is recovered. Setting $\beta = 1$ when $c > 0$ and $\beta = -1$ when $c < 0$ we obtain the FTBSCS and FTFSCS schemes respectively. Performing the von Neumann analysis, we find the stability condition

$$C^2 \leqslant \beta C + 2D \leqslant 1 \tag{A.63}$$

which is consistent with previous results for $\beta = 0, -1, 1$ (Pozrikidis, 2011).

A.4.4 Implicit BTCS scheme

Similarly, we can expect the discretization to be unconditionally stable when using an implicit method. For the one-dimensional convection-diffusion equation, the corresponding amplification factor is

$$G = \frac{1}{1 - 2D\,(cos\theta - 1) + ICsin\theta} \tag{A.64}$$

whose magnitude is always smaller than 1

$$|G|^2 = \frac{1}{\left[1 + 4Dsin^2(\frac{\theta}{2})\right]^2 + C^2 sin^2\theta} \leqslant 1 \tag{A.65}$$

hence the method is unconditional stable (Pozrikidis, 2011). This conclusion holds for N-dimension, since

$$G = \frac{1}{1 - \sum_{p=1}^{N} \left[2D_{x_p}\left(cos\theta_{x_p} - 1\right)\right] + I\sum_{p=1}^{N} C_{x_p} sin\theta_{x_p}} \tag{A.66}$$

and it is easy to verify that its magnitude is smaller than 1

$$|G|^2 = \frac{1}{\left[1 + \sum_{p=1}^{N} 4D_{x_p} sin^2(\frac{\theta_{x_p}}{2})\right]^2 + \left(\sum_{p=1}^{N} C_{x_p} sin\theta_{x_p}\right)^2} \leqslant 1 \tag{A.67}$$

A.5 Stability analysis for Navier-Stokes equations

As demonstrated previously, the von Neumann stability analysis is very effective for linear equations. For non-linear problems, such as the Burgers' equation or the Navier-Stokes equations, it can be extremely difficult to derive the exact expression of stability. One solution is to linearise the equations and perform a local stability analysis. In general, the stability obtained in the linear cases is only a necessary, but not sufficient, condition for the corresponding non-linear equations (Hirsch, 2007; Deriaz, 2012).

Equation	Diffusion scheme		Convection scheme		Stability
	Time	Space	Time	Space	
Diffusion	Explicit	Centered			Conditionally stable
	Implicit	Centered			Unconditionally stable
Convection			Explicit	Centered	Unconditionally unstable
			Explicit	Upwind	Conditionally stable
			Implicit	Centered	Unconditionally stable
Transport	Explicit	Centered	Explicit	Centered	Conditionally stable
	Explicit	Centered	Explicit	Upwind	Conditionally stable
	Explicit	Centered	Explicit	Hybrid	Conditionally stable
	Implicit	Centered	Implicit	Centered	Unconditionally stable
NSE	Explicit	Centered	Explicit	Centered	Conditionally stable
	Implicit	Centered	Explicit	Centered	Conditionally stable
	Implicit	Centered	Implicit	Centered	Unconditionally stable

Table A.1: Stability conditions for different equations using different schemes.

Therefore, if the convective term is treated explicitly in NSE, it requires a necessary CFL condition to maintain stability. Recently a more restrictive stability condition than the CFL condition is derived by Deriaz (2012). If the explicit Euler scheme for the convection is used, the time step should be chosen smaller than $(h/u_{\max})^2$. Increasing the temporal order of the scheme will improve the stability. When the second order Runge–Kutta or the second order Adams–Bashforth scheme is applied to the convection, Δt needs to be smaller than $(h/u_{\max})^{4/3}$.
To sum up, Table A.1 lists all the stability conditions discussed previously for different equations using different schemes.

Code description

The code of this thesis is built from zero using the objective oriented language C++ based on the CUDA platform, with less than 10,000 lines in total. The global structure of the code is shown in Figure B.1. The version evolution is controlled by the "git" system, which is found more efficient than the "svn" software. Now the code is backed up to the website Bitbucket "https://bitbucket.org/".

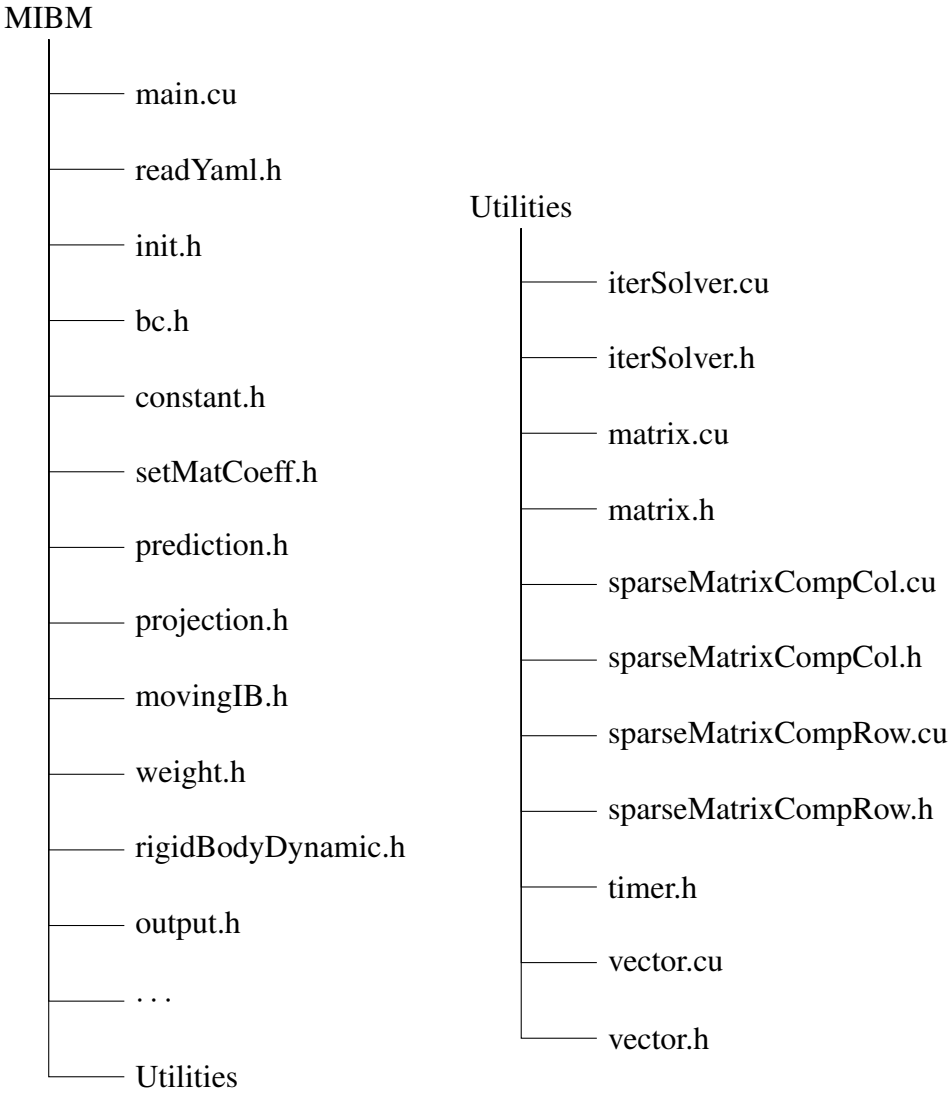


Figure B.1: Global structure of the code.

B.1 Input file formats: XML, YAML

In order to well organize the input data, we have considered the formats of XML (Extensible Markup Language) and YAML (YAML Ain't Markup Language). Compared to the pure text input, they are more human-readable and machine-readable. Moreover it is very convenient to add, remove or reorder entries with those formats without changing the code.

Listing B.1 shows a sample XML input file for the flow over a stationary cylinder case. The elements in the XML file are well documented and marked up with "<>" and "</>". The entries are read into the program through "pugixml version 1.2", a light-weight C++ XML library from "http://pugixml.org/".

The YAML format is data-oriented, rather than document markup. The same example in YAML format is shown in Listing B.2. Entries do not require enclosure in quotations. Therefore YAML is more compact and user-friendly compared to the XML format. In present work, we select the YAML format instead of XML for the input file. To read the YAML file format, the "yaml-cpp version 0.3.0" from "https://github.com/jbeder/yaml-cpp/releases/tag/release-0.3.0" is used.

Listing B.1: Short sample XML input file.

```
1  // Input file "flowOverCylinder.xml"
2  <?xml version="1.0" encoding="UTF-8" ?>
3  <case>
4      <caseName>flowOverCylinder</caseName>
5      <timeSettings>
6          <dt>0.1</dt>
7          <tf>500</tf>
8          <saveInterval>1000</saveInterval>
9      </timeSettings>
10 </case>
```

Listing B.2: Short sample YAML input file.

```
1  // Input file "flowOverCylinder.yaml"
2  - caseName : flowOverCylinder
3    timeSettings :
4        - dt : 0.01
5          tf : 500
6          saveInterval : 1000
```

B.2 Matrix manipulations and linear system solvers

Note that C++ does not provide matrix-vector manipulations by default. We build a library to allow for various operations between vector, matrix and sparse matrix, which turns out to be comparable to the open source libraries such as "IML++", "Armadillo", "Eigen", "Blitz", etc. We also would like to use MATLAB syntax in C++ for compact code programming and optimized execution performance. To this end, we redefine the C++ array indexing from $[i][j]$ to (i, j) through operator overloading. The most frequent operations are coded into the "utilities" package using template programming, such as "vector.cu", "matrix.cu", "sparseMatrixCompCol.cu" with CCS storage and "sparseMatrixCompRow.cu" with CRS storage. With this, we are able to construct the fluid matrix and the moving force coefficient matrix easily and freely.

Listing B.3 illustrates the main solution procedures of the immersed boundary forcing in MIBM, including interpolation, solution and spreading. To solve the linear systems, we build the solver "iterSolver.cu" that only contains matrix-vector manipulations. The conjugate gradient code for solving the moving force equation in CPU is shown in Listing B.4. The "iterSolver.cu" is also built with template programming, thus it can be used to any other linear systems directly without modifications.

Listing B.3: Solve moving force equation.

```
// Construct interpolation matrices Tu, Tv with CRS
(......)
// Construct spreading matrices Su, Sv with CCS
Su = Tu.transpose() * volumeFactor;
Sv = Tv.transpose() * volumeFactor;
// Construct moving force coefficient matrices
Mu = Tu * Su;
Mv = Tv * Sv;
// Given boundary velocities Uxd, Uyd and construct RHS vector
bu = ( Uxd - Tu * u ) / dt;
bv = ( Uyd - Tv * v ) / dt;
// Solve moving force equation
iterSolver.cg(Mu,fxb,bu,maxIter,tolerance);
iterSolver.cg(Mv,fyb,bv,maxIter,tolerance);
// Spread the boundary force
fx = Su * fxb;
fy = Sv * fyb;
```

Listing B.4: Conjugate gradient code.

```
// Template programming for conjugate gradient method
template<typename T>
int iterSolver<T>::cg(spMatCompRow<T>& A, vec<T>& x, vec<T>& b,
    int maxIter, double tol)
{
    flag = 1;
    x.zeros(b.rows());

    vec<T> r;
    vec<T> rn;
    vec<T> p;
    double alpha;
    double beta;

    r = b - A * x;
    p = r;

    if( b.norm("2") == 0 )
        return flag;

    for( iter=0; iter<maxIter; iter++ )
    {
        alpha = ( r * r ) / ( A * p * p );
        x += p * alpha;
        rn = r ;
        r -= A * p * alpha;
        beta = ( r * r ) / ( rn * rn );
        p = r + p * beta;

        resid = r.rms();
        if( resid < tol )
        {
            flag = 0;
            break;
        }
    }

    return flag;
}
```

B.3 Parallel computing with GPU

Present code is parallelized with CUDA library of version 3.2 for GPU computing on the HPC platform PILCAM2. Listing B.5 shows the solution procedure of the pressure Poisson equation in GPU. Usually we assemble the coefficient matrix and the RHS vector in CPU (host) and copy them to the GPU (device). The preconditioner and solver are called in the device for solving the linear system, with the help of "CUSP" library of version 0.2.0 from "https://code.google.com/archive/p/cusp-library/". Finally the solution is copied back from GPU to CPU. The "THRUST" library of version 1.3.0 from "https://thrust.github.io/" is used for parallelizing the data and the calculation.

Listing B.5: Parallel computing with GPU.

```
1  // Assemble coefficient matrix in the host
2  (......)
3  // Copy coefficient matrix and RHS vector from host to device
4  A_device = A_host;
5  b_device = b_host;
6  // GAMG preconditioner
7  cusp::precond::aggregation::smoothed_aggregation<int,float,cusp::
       device_memory> M(A_device);
8  // Set up convergence criterion
9  cusp::default_monitor<float> monitor(b_device,maxIter,tolerance);
10 // Solve the system by conjugate gradient method
11 cusp::krylov::cg(A_device,x_device,b_device,monitor,M);
12 // Copy the solution from device to host
13 x_host = x_device;
```