A HIGH-ORDER IMMERSED-BOUNDARY METHOD FOR SIMULATION OF INCOMPRESSIBLE FLOWS

BY

Chi Zhu

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Mechanical Engineering

August, 2014

Nashville, Tennessee

Approved:

Haoxiang Luo, Ph.D.

D. Greg Walker, Ph.D.

Prodyot K. Basu, Ph.D.

ACKNOWLEDGMENTS

I have had two wonderful years at Vanderbilt University. I would like to use this chance to express my gratitude to those who have helped me during this period of time.

First, I want to thank my advisor, Dr. Haoxiang Luo. His support and wealth of knowledge guided me through the obstacles in my research. He was also willing to share his experience as a graduate student with me to help me choose the best career path. When I was making one of hardest decisions in my life, switching from the PhD program to the Master program, he was there providing suggestions and support. I really appreciate this. For the same reason, I would like to thank Dr. Jon Edd, Dr. Deyu Li and Dr. Douglas LeVan for their great advice.

As a foreigner, it was not easy for me to fit in the life here. But I was lucky to have Suzanne Weiss and my friend Qian Zhang with me all the way. Moreover, I owe my gratitude to all my previous and current labmates Bo Yin, Hu Dai, Fangbao Tian, Guibo Li, Jialei Song, Shiyuan Chang and Casey Brock. They not only helped me with my research, but also made my life easier.

Being a teaching assistant for two years was a wonderful experience. I am sincerely appreciative to the Mechanical Engineering Department for the finical support. Also, thanks all my course supervisors, Dr. Ahad Nasab, Dr. Greg Walker, Dr. Jason Valentine, and Dr. Amrutur Anilkumar, for their help.

I also want to thank my committee members, Dr. Haoxiang Luo, Dr. Greg Walker and Dr. Prodyot Basu for their critiques and time commitment.

Last but not least, I offer my most sincere thanks to my parents and my sister. Their love and support means the world to me!

CONTENTS

Page

ACKNOWLEDGMENTS . . . . . **ii**

LIST OF FIGURES . . . . . **iv**

Chapter

I Introduction **1**
- 1.1 Background 1
- 1.2 Objectives 5
- 1.3 Outline 5

II Numerical approach **7**
- 2.1 Governing equations and the time-marching scheme 7
- 2.2 Compact finite-difference scheme 8
- 2.3 Rate of convergence 10
- 2.4 Immersed-boundary treatment 11
- 2.5 A high-order method for the pressure Poisson equation 15

III One-dimensional tests **21**
- 3.1 One-dimensional advection-diffusion equation 21
- 3.2 One-dimensional Poisson equation 27
- 3.3 Conclusion 31

IV Two-dimensional numerical tests **32**
- 4.1 Least squares treatment 32
- 4.2 A high-order Poisson solver for the interior points 35
- 4.3 Kovasznay flow 36
- 4.4 Flow past a circular cylinder 37
- 4.5 Conclusion 42

V Conclusions **43**
- 5.1 Summary of present work 43
- 5.2 Contributions of present work 43
- 5.3 Directions for future work 44
  - 5.3.1 Improve some details of the program 44
  - 5.3.2 Utilize the program to solve flapping wing problems 45

REFERENCES **50**

LIST OF FIGURES

Figure **Page**

1.1 A 2D schematic showing a solid body immersed in the fluid region. . . . 2
2.1 A 2D schematic describing the previous second-order ghost-cell approach for treating the immersed boundary [9, 17]. 11
2.2 A 2D schematic describing the current least squares method for the immersed boundary. 12
3.1 1D schematic describing the extrapolation method. 23
3.2 Steady state solution of the 1D advection-diffusion equation with N=99, and compact finite-difference scheme along with quadratic polynomial extrapolation. 24
3.3 Convergence rate of the 1D advection-diffusion equation test. Interior nodes: second-order central-difference scheme. Immersed boundary: quadratic polynomial extrapolation. 25
3.4 Convergence rate of the 1D advection-diffusion equation test. Interior nodes: compact finite-difference scheme. Immersed boundary: N/A (body-fitted grid and the one-sided scheme are used). 25
3.5 Convergence rate of the 1D advection-diffusion equation test. Interior nodes: compact finite-difference scheme. Immersed boundary: quadratic polynomial extrapolation, $m = 2$. 26
3.6 Convergence rate of the 1D advection-diffusion equation test. Interior nodes: compact finite-difference scheme. Immersed boundary: cubic polynomial extrapolation, $m = 3$. 26
3.7 Solution of the 1D Poisson equation with N=99, compact finite-difference scheme and cubic polynomial extrapolation. 28
3.8 The reference case, of which the rate of convergence is shown. Interior nodes: compact finite-difference scheme. Immersed boundary: N/A (body-fitted grid). 29
3.9 Convergence rate the 1D Poisson equation test. Interior nodes: compact finite-difference scheme. Immersed boundary: cubic polynomial extrapolation, $m = 3$. 29
3.10 Convergence rate of the 1D Poisson equation test. Interior nodes: compact finite-difference scheme. Immersed boundary: extrapolation with a polynomial of fifth degree, $m = 5$. 31
4.1 The computational domain used to test the least squares immersed-boundary treatment; the circle surrounding the grey area defines the immersed boundary. 33
4.2 Rate of convergence of the least squares interpolation of function (4.1) when the Dirichlet boundary condition is applied at the interior boundary points. 34
4.3 Rate of convergence of the least squares interpolation of function (4.1) when the Neumann boundary condition is applied at the interior boundary points. 35

4.4 Rate of convergence of the new Poisson solver with correction terms for solving Eq. (4.4). 36
4.5 (a) Flow pattern of Kovasznay flow at Re=40. (b) L_1, L_2 and L_∞ norms of the error for the streamwise velocity u and transverse velocity v versus the grid spacing. 38
4.6 Convergence of the compact scheme combined with a previous second-order immersed-boundary method where a linear extrapolation is used to treat the immersed boundary. Plotted are the L_1 and L_2 norms of the error for the streamwise velocity u and transverse velocity v versus the grid spacing. 39
4.7 Error distribution of different velocity components on the 160×160 grid. (a) u-velocity; (b) v-velocity. The results from the 640×640 grid are chosen as references. 40
4.8 Convergence of the high-order immersed-boundary method, where the least squares method is applied at the immersed boundary. Plotted are the L_1, L_2 and L_∞ norms of the error for the streamwise velocity u and transverse velocity v versus the grid spacing. 41
5.1 Rate of convergence of the incomplete least squares interpolation of function (4.1) when the Dirichlet boundary condition is applied at the interior boundary points. 46
5.2 Rate of convergence of the incomplete least squares interpolation of function (4.1) when the Neumann boundary condition is applied at the interior boundary points. 46
5.3 A schematic of the rigid wing during hovering flight. 47
5.4 Vorticity field of the domain with a flapping wing in one cycle. (a) $t = T$; (b) $t = \frac{5T}{4}$; (c) $t = \frac{6T}{4}$; (d) $t = \frac{7T}{4}$; (e) $t = 2T$. 49

CHAPTER I

INTRODUCTION

1.1 Background

Fluid-structure interaction problems are very common in nature and in engineering applications. Examples include blood flow, biological flying/swimming, morphing airplanes, and parachute deployment, etc. One major obstacle of simulating such problems is how to handle the fluid/solid interface. The most common method for such problems is called Arbitrary Lagrangian-Eulerian (ALE) method, in which the governing equations are discretized on body-fitted grids that conform to the instantaneous geometry of the solid surface. However, this method may require massive computational time spent on coordinate transformation and grid-regenerating [1]. Worse still, for complex geometries and moving boundaries, quality control of the body-fitted mesh often becomes a significant issue. A particular example is the unsteady aerodynamics involved in the flapping wings of insects, birds, bats, and more recently, the biomimetic unmanned aerial vehicles. Such wings typically involve complex kinematics, and the wing shape is time-dependent and consists of important three-dimensional features due to active and/or passive flexibility of the wing structure. Even though the Reynolds numbers of these wings are several orders of magnitude lower than that of typical aircraft, the numerical modeling of their aerodynamics often requires direct numerical simulations (DNS) and is thus highly demanding. For this type of flow, the immersed-boundary method has been popularly used [2, 3]. Here, we give a brief review of the immersed-boundary method. Take the simulation of flow around a solid body shown in Fig. 1.1 for example. There are two boundaries defining the computational domain, the exterior boundary, which surrounds the fluid region, and the interior boundary, which encloses

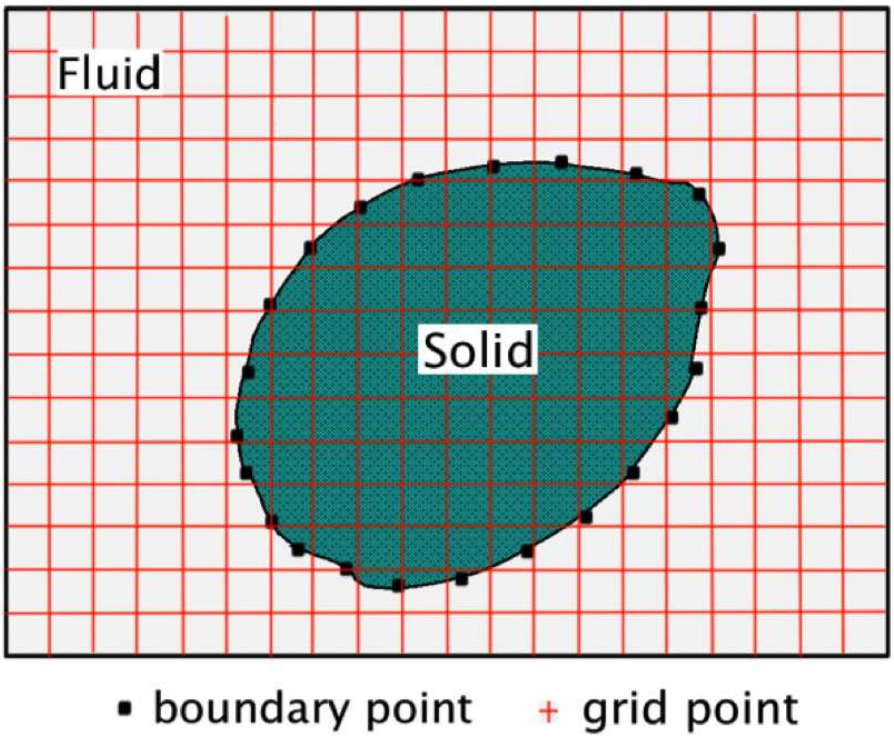


Figure 1.1: A 2D schematic showing a solid body immersed in the fluid region.

the solid body. The fluid/solid interface could be represented and traced explicitly by a set of Lagrangian points, or implicitly by approaches like level-set method [4]. In this work, we will use the Lagrangian points approach, as it is more commonly used in fluid-structure interaction problems. In addition to the interface points, a structured grid, typically a Cartesian grid, is defined on the whole domain, on which the governing equation will be discretized and solved in an Eulerian fashion.

Since the immersed boundary generally does not coincide with the grid, the boundary conditions at the interior boundary should be included in an unconventional way. These boundary conditions are imposed through adding a source term or forcing function in the governing equations around the immersed boundary. According to Mittal & Iaccarino's definition in [3], the immersed-boundary method can be divided into two categories based on how these source terms are applied. In the so-called "continuous forcing approach", the source term usually takes the form of a continuous distribution

function defined in a narrow region surrounding the physical boundary [2, 5], making the boundary conditions "diffused" into the fluid region. The other type is called "discrete forcing approach", because the forcing terms are incorporated, after the governing equations are discretized, at selected nodes only. In this type of implementations of immersed-boundary method, the computational stencil around the immersed boundary is modified according to the required boundary conditions, so that the interface still remains "sharp", without any ambiguity. One major advantage of the "sharp" interface is that high-order local accuracy can be achieved around the immersed boundary. There are several implementations of the sharp-interface approach. One of them is the cut-cell finite-volume approach, which is designed to meet the conservation laws for the cells around the immersed boundary. This is achieved by employing a finite-volume scheme to treat cells cut through by the immersed boundary, while the bulk fluid region is still discretized by the finite-difference scheme. Some of the work using this method are Ye *et al.* [6] and Udaykumar *et al.* [7]. Another method that falls into this category is the ghost-cell finite-difference approach. The ghost cell refers to a cell that is inside the solid region but is also included in the computational stencil. The source terms are added implicitly through a local flow reconstruction near the immersed boundary. Usually, the reconstruction is realized by building a polynomial using the nodal values in the fluid region and the boundary conditions at the interface as the input. Examples of previous work that employed this method are Tseng *et al.* [8], Mittal *et al.* [9] and Yang *et al.* [10].

Generally speaking, based on a fixed, structured grid (typically a Cartesian grid), the immersed-boundary method can deal with complex and moving boundaries without the need to regenerate the mesh. In addition, computing can be done efficiently on the grid, thanks to the simple mesh structure. However, most of existing immersed-boundary methods are of second-order accuracy. For many applications, and even in the case of low-Re flapping wing aerodynamics, Reynolds numbers can be well beyond the limit for increasing the resolution in the simulation. In such situations, using a high-order

approach could greatly extend the capability of the immersed-boundary.

There have been several recent efforts in developing high-order immersed-boundary methods. For example, in the work of Seo *et al.* [11] the compact finite-difference scheme and a high-order immersed-boundary method are combined to solve the linearized compressible equations. Laizet *et al.* [12] managed to solve the incompressible flow with the help of compact schemes and spectral methods. However, the immersed-boundary method they applied is still second-order accurate. Zeng *et al.* [13] used a third-order upwind finite-difference scheme along with a high-order immersed-boundary method to solve inviscid flows. In their method, the flow variables at the ghost cells are determined by coupling the interpolation and the state equation. An overall third-order accuracy is achieved for 2D advection problems. Zhou *et al.* [14] developed a so-called high-order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. In this method, they introduced multiple ghost nodes along each spatial direction to treat the jump conditions across the immersed interface. Their method can be combined with an explicit high-order finite-difference method to discretize the entire domain. Gibou & Fedkiw [15] developed a fourth-order immersed-boundary method for the Laplace and heat equations, in which the Laplacian is discretized using a five-point finite-difference stencil. To treat the immersed-boundary, one-dimensional polynomial extrapolations are used to determine the values of the variables at the two layers of ghost nodes. Finally, Linnick & Fasel [16] presented a fourth-order immersed-boundary method based on the compact scheme. To treat the boundary, they introduced high-order correction terms for the jump conditions across the interface. Since the method is applied to two-dimensional incompressible Navier-Stokes equations in the streamfunction-vorticity formulation, they avoided solving the Poisson equation with the high-order approach.

1.2 Objectives

In this work, we aim to develop an efficient high-order program to solve incompressible flow problems with immersed boundaries by combining the compact scheme and a compatible high-order immersed-boundary method. The compact scheme is chosen because it can provide better resolution at small spatial scales, due to the high accuracy, low-dispersion, and low-dissipation properties. The compact scheme also utilizes shorter stencils than some other high-order schemes, e.g., the explicit finite-difference method, and it is based on structured grids and can thus take advantage of efficient computations. For the immersed-boundary treatment, an extended idea based on the second-order ghost-cell method described in Mittal *et al.* [9] and Luo *et al.* [17] is adopted. Specifically, the second-order forcing scheme based on linear interpolation/extrapolation is replaced by the least squares method in Luo *et al.* [18] to ensure the accuracy of the overall program.

1.3 Outline

The structure of this thesis is organized as follows.

Chapter 1 introduces the background information, giving a brief overview of the field to provide context to the study. Here, various studies by other researchers are also reported. In this introduction, we intend to detail the background of the present study and also to clarify the objectives of this work as well as the organization of the thesis.

Chapter 2 elaborates the theories behind this study. We first introduce the mathematical model of the fluid problem solved in the work and the structure of the program. Thereafter, the compact scheme that is used to discretize the bulk flow region is detailed. Next, a least-squares based high-order immersed-boundary method is presented. Finally, an efficient, high-order Poisson solver developed as part of this study is described.

Chapter 3 consists of several fundamental one-dimensional studies. The high-order

immersed-boundary method is implemented to solve one-dimensional problems to test its accuracy and stability. Both an advection-diffusion equation and a Poisson equation are solved in these tests. The results are compared with both results from a second-order method and the exact solutions.

Chapter 4 provides several tests of two-dimensional implementation of the high -order immersed-boundary method. The Navier-Stokes equations are solved in this chapter. In particular, Kovasznay flow is used to assess the performance of the program when there is no immersed boundary present. Flow past a circular cylinder case is solved to evaluate the performance of the complete 2D program.

Chapter 5 serves as a summary to the thesis. In this chapter the overall conclusions and contributions of current work and suggested future work are presented.

CHAPTER II

NUMERICAL APPROACH

This chapter is used to elaborate the numerical methods used in the work. In section 2.1, we describe the governing equations as well as a step-by-step description of how the three-step projection method is combined with a fourth-order Runge-Kutta method for time advancement. Section 2.2 explains the compact schemes used to discretize the bulk flow region. Section 2.3 includes both a brief introduction of the original second-order immersed-boundary treatment and a detailed description of the high-order one. In section 2.4, a high-order, efficient pressure Poisson solver is presented.

2.1 Governing equations and the time-marching scheme

The following viscous, incompressible, unsteady Navier-Stokes equations are solved in our study:

$$\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V}\cdot\nabla)\mathbf{V} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{V} \tag{2.1}$$

$$\nabla\cdot\mathbf{V} = 0 \tag{2.2}$$

where $\mathbf{V}$ is the velocity vector, p is the pressure with the density incorporated , and Re represents the Reynolds number. A three-step projection method is used to decouple the pressure and velocity components in the temporal discretization.

$$\frac{\mathbf{V}^* - \mathbf{V}^n}{\Delta t} + (\mathbf{V}\cdot\nabla\mathbf{V})^n = \frac{1}{Re}(\nabla^2\mathbf{V})^n \tag{2.3}$$

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t}\nabla\mathbf{V}^* \tag{2.4}$$

$$\mathbf{V}^{n+1} = \mathbf{V}^* - \Delta t\nabla p^{n+1} \tag{2.5}$$

where n denotes nth time step, and a star (*) represents the intermediate velocity. To ensure the overall accuracy, a fourth-order Runge-Kutta (RK) method [19] is applied for time marching. The procedure is:

1) For the first sub-step, $i = 1$,

   $\mathbf{V}^{(1)} = \mathbf{V}^n$

   $\mathbf{k}^{(1)} = -(\mathbf{V}^{(1)} \cdot \nabla)\mathbf{V}^{(1)} + \frac{1}{Re}\nabla^2\mathbf{V}^{(1)}$,

   where n means nth time step, and (i) represents the RK step level.

2) For the RK sub-step $i = 2$ to 4, we use the following projection method to get $(\mathbf{V}^{(i)}, p^{(i)})$:

   $\mathbf{V}^* = \mathbf{V}^{(i-1)} + \alpha_i \Delta t \mathbf{k}^{(i-1)}$

   $\nabla^2 p^{(i)} = \frac{1}{\alpha_i \Delta t}\nabla \cdot \mathbf{V}^*$

   $\mathbf{V}^{(i)} = \mathbf{V}^* - \alpha_i \Delta t \nabla p^{(i)}$

   $\mathbf{k}^{(i)} = -(\mathbf{V}^{(i)} \cdot \nabla)\mathbf{V}^{(i)} + \frac{1}{Re}\nabla^2\mathbf{V}^{(i)}$,

   where $\alpha_i = \frac{1}{2}, \frac{1}{2}$, and 1 for $i = 2, 3$, and 4, respectively.

3) At the end of a complete step:

   $\mathbf{V}^* = \mathbf{V}^n + \frac{\Delta t}{6}(\mathbf{k}^{(1)} + 2\mathbf{k}^{(2)} + 2\mathbf{k}^{(3)} + \mathbf{k}^{(4)})$

   $\nabla^2 p^{n+1} = \frac{1}{\Delta t}\nabla \cdot \mathbf{V}^*$

   $\mathbf{V}^{n+1} = \mathbf{V}^* - \Delta t \nabla p^{n+1}$.

2.2 Compact finite-difference scheme

The compact finite-difference scheme is one of the most widely used high-order schemes. Thanks to its low-dissipation and low-dispersion properties, the compact scheme can resolve very short length scales [20]. Compact schemes can have either symmetric or asymmetric stencils. What is more, they can be applied on both uniform grid [20] and non-uniform grid [21]. In our study, symmetric schemes are used to solve

the bulk flow and asymmetry schemes are used to treat the exterior boundaries. For the sake of simplicity, uniform Cartesian grids will be adopted in all the cases in this work.

The general form of the symmetric compact-scheme for the first-order derivative of function $f(x)$ is:

$$\beta f'_{i-2} + \alpha f'_{i-1} + f'_i + \alpha f'_{i+1} + \beta f'_{i+2} = c\frac{f_{i+3} - f_{i-3}}{6h} + b\frac{f_{i+2} - f_{i-2}}{4h} + a\frac{f_{i+1} - f_{i-1}}{2h} \tag{2.6}$$

where f_i is the nodal value at node i, and h is the grid spacing. Using the Taylor series to expand the equation and matching coefficients to different orders will lead to the relationship between a, b, c, and α, β. By satisfying different constraints, we can get finite-difference schemes of different order of accuracy. Some of the commonly used schemes are:

$$\frac{1}{4}f'_{i-1} + f'_i + \frac{1}{4}f'_{i+1} = \frac{3}{2}\frac{f_{i+1} - f_{i-1}}{2h} \qquad \text{(4th-order)} \tag{2.7}$$

$$\frac{1}{3}f'_{i-1} + f'_i + \frac{1}{3}f'_{i+1} = \frac{14}{9}\frac{f_{i+1} - f_{i-1}}{2h} + \frac{1}{9}\frac{f_{i+2} - f_{i-2}}{4h} \qquad \text{(6th-order)} \tag{2.8}$$

At the exterior boundary, say $i = 1$, the following one-sided compact scheme can be used:

$$f'_1 + 2f'_2 = \frac{1}{h}(-\frac{5}{2}f_1 + 2f_2 + \frac{1}{2}f_3) \qquad \text{(3rd-order)} \tag{2.9}$$

For the second-order derivatives, f'', we have similar equations:

$$\frac{1}{10}f''_{i-1} + f''_i + \frac{1}{10}f''_{i+1} = \frac{6}{5}\frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} \qquad \text{(4th-order)} \tag{2.10}$$

$$\frac{2}{11}f''_{i-1} + f''_i + \frac{2}{11}f''_{i+1} = \frac{12}{11}\frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} + \frac{3}{11}\frac{f_{i+2} - 2f_i + f_{i-2}}{4h^2} \quad \text{(6th-order)} \quad (2.11)$$

and at the exterior boundary, $i = 1$,

$$f''_1 + 11f''_2 = \frac{1}{h^2}(13f_1 - 27f_2 + 15f_3 - f_4) \qquad \text{(3rd-order)} \qquad (2.12)$$

2.3 Rate of convergence

Several tests will be presented in this thesis to show the feasibility and effectiveness of using the compact finite-difference scheme and the high-order immersed-boundary method to solve problems with interior boundaries. In order to compare the results in a more rigorous way, we choose problems with an exact solution to do most of these tests. Also, different error-norms are adopted to help quantitatively evaluate the performance of the simulations. Assume ϕ represents a general function. These norms are defined as:

$$L_1 = \frac{1}{N+1}\sum_{i=0}^{N}\left|\phi_{i,numerical} - \phi_{i,exact}\right|, \qquad (2.13)$$

$$L_2 = \sqrt{\frac{1}{N+1}\sum_{i=0}^{N}\left(\phi_{i,numerical} - \phi_{i,exact}\right)^2}, \qquad (2.14)$$

$$L_\infty = \max\left(\left|\phi_{i,numerical} - \phi_{i,exact}\right|\right). \qquad (2.15)$$

There norms can be put into two categories. L_1 and L_2 are good approximations of the global error, while L_∞ can capture the local error effectively, especially that around the immersed boundaries.

Accordingly, the rate of convergence can be defined in terms of different error-

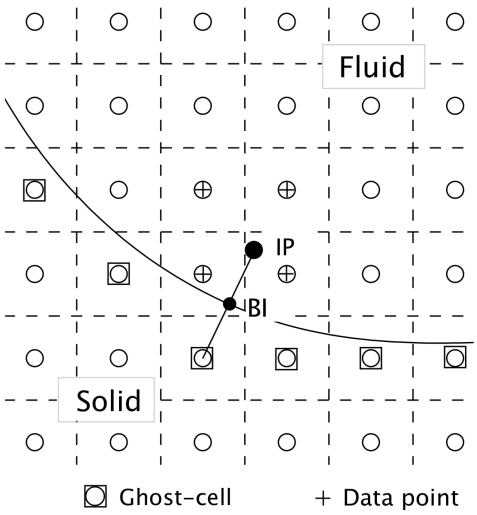


Figure 2.1: A 2D schematic describing the previous second-order ghost-cell approach for treating the immersed boundary [9, 17].

norms:

$$rate = -\frac{log(err_1/err_2)}{log(N_1/N_2)}, \tag{2.16}$$

where err_1 and err_2 are the error-norms of tests with $(N_1 + 1)$ nodes and $(N_2 + 1)$ nodes. Usually, a log-log plot of error-norms versus grid spacing will be provided instead of the numeric rate of convergence. According to Eq. (2.16), the slope of the plot equals the rate of convergence.

2.4 Immersed-boundary treatment

Here we extend a previous sharp-interface, second-order immersed boundary method by Mittal *et al.* [9] and Luo *et al.* [17] to higher-order accuracy. The previous low-order, ghost-cell method, shown in Fig. 2.1, is divided into the following four steps:

1) Identify the so-called “ghost cells" (GC), which refer to solid nodes that are in-

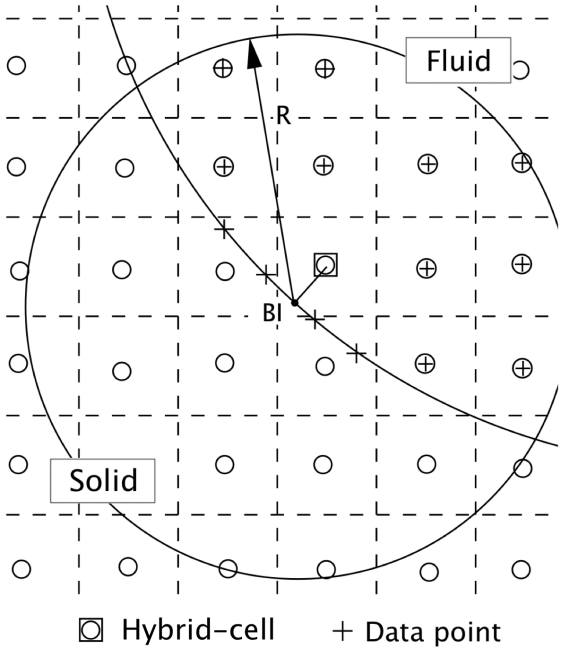


Figure 2.2: A 2D schematic describing the current least squares method for the immersed boundary.

cluded in the finite-difference stencils of fluid nodes in the vicinity of the immersed boundary.

2) Find the corresponding image point (IP) in the fluid for each GC, which makes the line connecting an IP and a GC perpendicular to the immersed boundary; the intercept point is called the boundary intercept (BI).

3) Locate the four nodal points surrounding the IP, and use a bilinear interpolation to get the value at the IP.

4) Determine the value at the GC through a second-order extrapolation using the values at the IP and the boundary condition at the BI.

To extend the boundary treatment to higher order, we adopt the least squares method developed by Luo *et al.* [18]. Some modifications are made during the implementation. First, instead of extrapolating the functions at the ghost cells outside of the fluid domain, we directly interpolate at the first fluid node next to the solid boundary. These nodes

will be called hybrid cells, since they are under the direct influence of both the fluid region and the solid region. Second, in the previous second-order method an image point is required along with the body intercept to determine the flow variable at the ghost cell. In the current method, the boundary conditions will be incorporated through including some boundary points in the least square scheme. Thus, the image point is no longer needed.

To ensure at least fourth-order accuracy of the new interpolation method, a multi-dimensional, third-degree polynomial, Φ, will be introduced to estimate the general function ϕ around the boundary intercept point (BI) at (x_0,y_0)

$$\phi(\hat{x},\hat{y}) \approx \Phi(\hat{x},\hat{y}) = \sum_{j=0}^{3}\sum_{i=0}^{3} c_{ij}\hat{x}^i\hat{y}^j, \qquad i+j \le 3, \tag{2.17}$$

where $\hat{x} = x - x_0$ and $\hat{y} = y - y_0$ are local coordinates, and c_{ij}, of which the total number is ten, are the coefficients to be determined. These coefficients will be calculated by the least squares method. First, sufficient number of nodes ($N \ge 10$) will be selected by a circle centered at BI with radius R, as is shown in Fig. 2.2. There are two types of data points will be used to do the interpolation. One of them is the fluid nodes that reside in the circle, and the other type is the boundary points that lie on the interface, through which the boundary conditions will be applied. Then, c_{ij} are determined so that the following error function

$$\epsilon = \sum_{n=1}^{N} w_n^2[\Phi(\hat{x}_n,\hat{y}_n) - \phi(\hat{x}_n,\hat{y}_n)]^2 \tag{2.18}$$

reaches its minimum. In Eq. (2.18), $(\hat{x}_n,\hat{y}_n)$ is the nth node, and w_n is the weight function, which is chosen basing on Li's work [22] and has the form:

$$w_n = \frac{1}{2}\left[1 + \cos\left(\frac{\pi d_n}{R}\right)\right]. \tag{2.19}$$

d_n is the distance between $(\hat{x}_n,\hat{y}_n)$ and (x_0, y_0).

If the Dirichlet boundary condition is applied at the interface, the weight matrix, $\mathbf{W}$, and Vandermonde matrix, $\mathbf{V}$, derived from Eq. (2.19) and Eq. (2.17), respectively, are:

$$\mathbf{W} = \begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_N \end{bmatrix} \tag{2.20}$$

$$\mathbf{V} = \begin{bmatrix} 1 & \hat{x}_1 & \hat{y}_1 & \hat{x}_1^2 & \hat{x}_1\hat{y}_1 & \hat{y}_1^2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \hat{x}_i & \hat{y}_i & \hat{x}_i^2 & \hat{x}_i\hat{y}_i & \hat{y}_i^2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \hat{x}_N & \hat{y}_N & \hat{x}_N^2 & \hat{x}_N\hat{y}_N & \hat{y}_N^2 & \cdots \end{bmatrix} \tag{2.21}$$

However, if the Neumann boundary condition,$\partial\phi/\partial n = g$, is provided at the immersed boundary point $(\hat{x}_B, \hat{y}_B)$, the $\mathbf{V}$ matrix will have the form:

$$\mathbf{V} = \begin{bmatrix} 1 & \hat{x}_1 & \hat{y}_1 & \hat{x}_1^2 & \hat{x}_1\hat{y}_1 & \hat{y}_1^2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \hat{x}_{N-1} & \hat{y}_{N-1} & \hat{x}_{N-1}^2 & \hat{x}_{N-1}\hat{y}_{N-1} & \hat{y}_{N-1}^2 & \cdots \\ 0 & \hat{x}_n & \hat{y}_n & 2\hat{x}_n\hat{x}_B & (\hat{x}_n\hat{y}_B + \hat{x}_B\hat{y}_n) & 2\hat{y}_n\hat{y}_B & \cdots \end{bmatrix} \tag{2.22}$$

where $(\hat{x}_n, \hat{y}_n)$ is the surface norm at $(\hat{x}_B, \hat{y}_B)$. Using the Vandermonde matrix, the coefficients c_{ij} can be calculated by:

$$c = \mathbf{V}^{\perp}\phi = (\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\phi \tag{2.23}$$

where an $\perp$ denotes pseudoinverse, which is a generalization of the inverse matrix.

2.5 A high-order method for the pressure Poisson equation

Typically solving the pressure Poisson equation (Eq. (2.4)) is the most time consuming part when simulating the incompressible flow. Because of the implicit nature of the derivatives in the compact scheme, additional challenges arise when it comes to solving the two- or three-dimensional Poisson equation. Previously, in order to apply the compact scheme, some researchers have added a pseudo-temporal term in the pressure Poisson equation [23, 24]. However, this method typically has a slow convergence speed. Some other researchers simply avoid this equation by solving a weakly compressible flow instead, and incorporate the equation of state [25]. Instead of applying the compact scheme directly, here we present an efficient, high-order Poisson solver developed based on the work of Singer *et al.* [26]. In this method, the discrete Laplacian maintains a three-point stencil in each direction, and therefore, the whole linear system can still take advantage of the high efficiency of tridiagonal solvers. The basic idea of this method is to attach some explicit correction terms to the standard second-order scheme so that we can achieve both high-order accuracy and the simplicity of the stencil.

To first illustrate this idea in 1D, we solve the equation

$$p''(x) = F(x), \tag{2.24}$$

We write down the standard second-order central difference scheme for the second derivative:

$$p''(x_i) \approx D_{xx}p(x_i) = \frac{p_{i+1} - 2p_i + p_{i-1}}{\Delta x^2}, \tag{2.25}$$

where $p''(x_i)$ and $p(x_i)$ represent the exact value of the second derivative and nodal value at x_i, and D_{xx} is the second-order numerical operator. We use p_i to represent $p(x_i)$. If function p is smooth enough, the error of Eq. (2.25) is

$$p''(x_i) = D_{xx}p(x_i) + O(\Delta x^2). \tag{2.26}$$

A more accurate approximation of Eq. (2.26) is

$$D_{xx}p(x_i) = p''(x_i) + \frac{\Delta x^2}{12}p^{(4)}(x_i) + O(\Delta x^4), \tag{2.27}$$

which can be easily shown using the Taylor expansion. Applying the operator D_{xx} to p'' term, we can get

$$p^{(4)}(x_i) = D_{xx}p''(x_i) + O(\Delta x^2). \tag{2.28}$$

Substituting Eq. (2.28) into Eq. (2.27) we have:

$$D_{xx}p(x_i) = (1 + \frac{\Delta x^2}{12}D_{xx})p''(x_i) + O(\Delta x^4). \tag{2.29}$$

This equation is essentially the same as the fourth-order compact scheme in Eq. (2.10). Using the original differential equation (Eq. (2.24)), Eq. (2.29) becomes

$$D_{xx}p(x_i) = F(x_i) + \frac{\Delta x^2}{12}D_{xx}F(x_i) + O(\Delta x^4). \tag{2.30}$$

or

$$D_{xx}p_i = F_i + \frac{\Delta x^2}{12}D_{xx}F_i + O(\Delta x^4).$$

This discretization is therefore fourth-order accurate, and the left side maintains the second-order discrete form. Compared with the second-order discretization, Eq. (2.30) has an extra term on its right-hand side, $\frac{\Delta x^2}{12}D_{xx}F(x_i)$, which is a high-order, explicit correction term.

Rearranging Eq. (2.29), we can get a fourth-order, implicit expression for $p''(x_i)$:

$$p''(x_i) = (1 + \frac{\Delta x^2}{12}D_{xx})^{-1}D_{xx}p(x_i) + O(\Delta x^4). \tag{2.31}$$

This expression will be use to derive the 2D formula.

For the 2D problem, we want to solve the following equation:

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = F(x, y). \tag{2.32}$$

Of course, there will also be proper boundary conditions attached. If the problem degrades to 1D, from Eq. (2.30) we already know the fourth-order discretization is

$$D_{xx} p_i = \frac{F_{i+1} + 10F_i + F_{i-1}}{12}. \tag{2.33}$$

The operator D_{xx} has the same definition as above. For the 2D problem, we use the expression in Eq. (2.31). Therefore, Eq. (2.32) becomes

$$(1 + \frac{\Delta x^2}{12} D_{xx})^{-1} D_{xx} p_{i,j} + (1 + \frac{\Delta y^2}{12} D_{yy})^{-1} D_{yy} p_{i,j} = F_{i,j}, \tag{2.34}$$

at point (i, j). Expanding this equation, we have:

$$(1 + \frac{\Delta y^2}{12} D_{yy}) D_{xx} p_{i,j} + (1 + \frac{\Delta x^2}{12} D_{xx}) D_{yy} p_{i,j} = (1 + \frac{\Delta x^2}{12} D_{xx})(1 + \frac{\Delta y^2}{12} D_{yy}) F_{i,j}. \tag{2.35}$$

Rearranging this equation, we get:

$$D_{xx} p_{i,j} + D_{yy} p_{i,j} = RHS_{i,j} - \frac{\Delta x^2 + \Delta y^2}{12} D_{xx} D_{yy} p_{i,j}, \tag{2.36}$$

where

$$RHS_{i,j} = F_{i,j} + \frac{\Delta x^2}{12} D_{xx} F_{i,j} + \frac{\Delta y^2}{12} D_{yy} F_{i,j} + \frac{\Delta x^2 \Delta y^2}{144} D_{xx} D_{yy} F_{i,j}.$$

Like its 1D counterpart, this discretization has also fourth-order accuracy.

As we can see, the left-hand side of the equation still have the same shape as the standard second-order discretization scheme. So all the efficient solvers developed for the previous scheme can be applied here with relative ease. Attention is needed for the terms on the right-hand side of Eq. (2.36). The last term in $RHS_{i,j}$ is at least one

magnitude smaller than any other terms in it, so we can ignore this term during actual implementation, which leads to further efficiency. To calculate $D_{xx}D_{yy}p_{i,j}$, we first define two expressions:

$$\delta_c = p_{i+1,j+1} + p_{i-1,j+1} + p_{i+1,j-1} + p_{i-1,j-1},$$
$$\delta_s = p_{i+1,j} + p_{i-1,j} + p_{i,j-1} + p_{i,j+1},$$

where c and s represent corner nodes and side nodes respectively. Then, we have

$$D_{xx}D_{yy}p_{i,j} = \frac{\delta_c - 2\delta_s + 4p_{i,j}}{\Delta x^2 \Delta y^2}. \tag{2.37}$$

This explicit formula involves nine nodes on the Cartesian grid.

After the right-hand side is explicitly calculated, Eq. (2.36) can be solved in the same manner as that for the standard second-order discretization. Therefore, the advantage of this method is that the structure of the original second-order program is retained. We only need to add some explicit correction terms, and the computational cost is not significantly increased. For the numerical tests described in this work, we will apply the ADI method to solve the linear system generated by discretizing the Poisson equation.

Extension to 3D is straightforward. Following the same procedure, for the 3D Poisson equation:

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} = F(x, y, z), \tag{2.38}$$

we apply Eq. (2.31) to each derivatives

$$(1+\frac{\Delta x^2}{12}D_{xx})^{-1}D_{xx}p_{i,j,k} + (1+\frac{\Delta y^2}{12}D_{yy})^{-1}D_{yy}p_{i,j,k} + (1+\frac{\Delta z^2}{12}D_{zz})^{-1}D_{zz}p_{i,j,k} = F_{i,j,k}. \tag{2.39}$$

Multiply $(1+\frac{\Delta x^2}{12}D_{xx})(1+\frac{\Delta y^2}{12}D_{yy})(1+\frac{\Delta z^2}{12}D_{zz})$ on both sides, we have

$$(1+\frac{\Delta y^2}{12}D_{yy})(1+\frac{\Delta z^2}{12}D_{zz})D_{xx}p_{i,j,k}+(1+\frac{\Delta x^2}{12}D_{xx})(1+\frac{\Delta z^2}{12}D_{zz})D_{yy}p_{i,j,k}+ \tag{2.40}$$
$$(1+\frac{\Delta x^2}{12}D_{xx})(1+\frac{\Delta y^2}{12}D_{yy})D_{zz}p_{i,j,k}=(1+\frac{\Delta x^2}{12}D_{xx})(1+\frac{\Delta y^2}{12}D_{yy})(1+\frac{\Delta z^2}{12}D_{zz})F_{i,j,k}.$$

Expand the above equation, we can get

$$\begin{aligned}
&D_{xx}p_{i,j,k}+\frac{\Delta y^2D_{yy}+\Delta z^2D_{zz}}{12}D_{xx}p_{i,j,k}+\frac{\Delta y^2\Delta z^2}{144}D_{xx}D_{yy}D_{zz}p_{i,j,k}+ \\
&D_{yy}p_{i,j,k}+\frac{\Delta x^2D_{xx}+\Delta z^2D_{zz}}{12}D_{yy}p_{i,j,k}+\frac{\Delta x^2\Delta z^2}{144}D_{xx}D_{yy}D_{zz}p_{i,j,k}+ \\
&D_{zz}p_{i,j,k}+\frac{\Delta x^2D_{xx}+\Delta y^2D_{yy}}{12}D_{zz}p_{i,j,k}+\frac{\Delta x^2\Delta y^2}{144}D_{xx}D_{yy}D_{zz}p_{i,j,k}= \\
&\qquad(1+\frac{\Delta x^2}{12}D_{xx})(1+\frac{\Delta y^2}{12}D_{yy})(1+\frac{\Delta z^2}{12}D_{zz})F_{i,j,k}.
\end{aligned} \tag{2.41}$$

Move all the extra terms to the right-hand side, we have

$$\begin{aligned}
&D_{xx}p_{i,j,k}+D_{yy}p_{i,j,k}+D_{zz}p_{i,j,k}=RHS_{i,j,k}- \\
&\frac{\Delta x^2\Delta y^2+\Delta y^2\Delta z^2+\Delta x^2\Delta z^2}{144}D_{xx}D_{yy}D_{zz}p_{i,j,k}-\frac{\Delta y^2D_{yy}+\Delta z^2D_{zz}}{12}D_{xx}p_{i,j,k}- \\
&\qquad\frac{\Delta x^2D_{xx}+\Delta z^2D_{zz}}{12}D_{yy}p_{i,j,k}-\frac{\Delta x^2D_{xx}+\Delta y^2D_{yy}}{12}D_{zz}p_{i,j,k},
\end{aligned} \tag{2.42}$$

where

$$\begin{aligned}
RHS_{i,j,k}=&F_{i,j,k}+\frac{\Delta x^2}{12}D_{xx}F_{i,j,k}+\frac{\Delta y^2}{12}D_{yy}F_{i,j,k}+\frac{\Delta z^2}{12}D_{zz}F_{i,j,k}+ \\
&\frac{\Delta x^2\Delta y^2}{144}D_{xx}D_{yy}F_{i,j,k}+\frac{\Delta y^2\Delta z^2}{144}D_{yy}D_{zz}F_{i,j,k}+\frac{\Delta z^2\Delta x^2}{144}D_{zz}D_{xx}F_{i,j,k}+ \\
&+\frac{\Delta x^2\Delta y^2\Delta z^2}{1728}D_{xx}D_{yy}D_{zz}F_{i,j,k}.
\end{aligned}$$

The basic idea is still evaluating all the terms on the right-hand side of Eq. (2.42) explicitly. Similar to the 2D case, the last term of the $RHS_{i,j,k}$ can be ignored during application, because it is one-order smaller than other terms. Also, the expression of

operator $D_{xx}D_{yy}D_{zz}$ can be derived in the same way, except there will be twenty-seven points included in the stencil.

CHAPTER III

ONE-DIMENSIONAL TESTS

This chapter will focus on the one-dimensional tests. These tests are designed as preliminary studies of the feasibility of combining the compact scheme and the high-order immersed-boundary method. Both an advection-diffusion equation and a Poisson equation are solved using the aforementioned combination. Although 1D implementation of the method is straightforward and only elementary numerical skills are required to solve these equations, these tests are nevertheless valuable, because from them we will learn the rate of convergence as well the numerical stability of the new method.

3.1 One-dimensional advection-diffusion equation

The time-dependent advection-diffusion equation solved in this section is:

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} - \frac{1}{Re}\frac{\partial^2 u}{\partial x^2} = 0, \quad x \in [-1.0, -0.3] \cup [0.3, 1.0]. \tag{3.1}$$

$Re = 50.0$ is a constant in the above equation. Given the following boundary conditions:

$$u = 1.0 \quad at \quad x = -1.0$$

$$u = 0.0 \quad at \quad x = -0.3$$

$$u = 0.0 \quad at \quad x = 0.3$$

$$u = -1.0 \quad at \quad x = 1.0,$$

we can get the steady state solution of the problem

$$u(x) = \frac{e^{Rex} - e^{-0.3Re}}{e^{-Re} - e^{-0.3Re}}, \qquad -1.0 \le x \le -0.3,$$
$$u(x) = -\frac{e^{Rex} - e^{0.3Re}}{e^{Re} - e^{0.3Re}}, \qquad 0.3 \le x \le 1.0.$$

Fig. 3.1 shows the grid we use in this test. The computational domain is from -1 to 1, with $(N + 1)$ evenly distributed nodes. To simplify the problem, we treat $x_L = -0.3$ and $x_R = 0.3$ as immersed boundaries. x_L is used as the example to explain the extrapolation strategy. As is shown in the figure, node x_{i+1} is the GC in this case. The value at the GC is calculated by the extrapolation using the following m-th degree polynomial. If a general function ϕ is defined on this domain, this equation will be written as:

$$\phi = \sum_{i=0}^{m} c_i x^i. \tag{3.2}$$

The coefficients $c_i, i = 0, 1, \ldots, m$, are determined by nodal values at x_{i-m+1}, x_{i-m+2}, $\ldots$, x_i and x_L. If we want to apply the Dirichlet boundary condition at x_L, the following equation is used to solve c_i:

$$\begin{bmatrix} x_{i-m+1}^m & x_{i-m+1}^{m-1} & \ldots & x_{i-m+1} & 1 \\ x_{i-m+2}^m & x_{i-m+2}^{m-1} & \ldots & x_{i-m+2} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_i^m & x_i^{m-1} & \ldots & x_i & 1 \\ x_L^m & x_L^{m-1} & \ldots & x_L & 1 \end{bmatrix} \begin{pmatrix} c_m \\ c_{m-1} \\ \vdots \\ c_1 \\ c_0 \end{pmatrix} = \begin{pmatrix} \phi_{i-m+1} \\ \phi_{i-m+2} \\ \vdots \\ \phi_i \\ \phi|_{x=x_L} \end{pmatrix} \tag{3.3}$$

Otherwise, if the boundary condition at x_L is the Neumann boundary condition, the

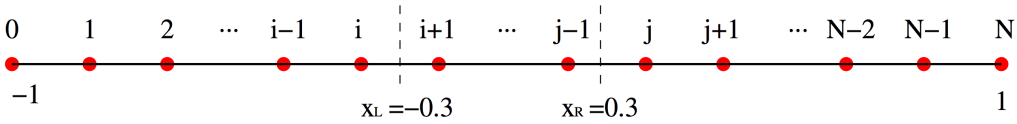


Figure 3.1: 1D schematic describing the extrapolation method.

equation below is solved:

$$\begin{bmatrix} x_{i-m+1}^{m} & x_{i-m+1}^{m-1} & \dots & x_{i-m+1} & 1 \\ x_{i-m+2}^{m} & x_{i-m+2}^{m-1} & \dots & x_{i-m+2} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i}^{m} & x_{i}^{m-1} & \dots & x_{i} & 1 \\ m x_{L}^{m-1} & (m-1) x_{L}^{m-2} & \dots & 1 & 0 \end{bmatrix} \begin{pmatrix} c_m \\ c_{m-1} \\ \vdots \\ c_1 \\ c_0 \end{pmatrix} = \begin{pmatrix} \phi_{i-m+1} \\ \phi_{i-m+2} \\ \vdots \\ \phi_i \\ \frac{\partial \phi}{\partial x}|_{x=x_L} \end{pmatrix} \tag{3.4}$$

Using different values of m will provide different level of accuracy for the extrapolation.

The immersed boundary at x_R is treated in a similar way as above. The only difference is that the nodal values at x_{j+m-1}, x_{j+m-2}, ..., x_j and x_R are used to evaluate the value at the GC, x_{j-1}.

If the immersed boundary coincide with the nodal points (i.e., body-fitted grid), say x_L and x_i, the immersed boundaries can be treated in the same way as the exterior boundaries at x_0 and x_N are treated, using one-sided schemes like (2.9) and (2.12).

Different combinations of methods are used to get the numerical solutions of the problem. On one hand, if the interior boundaries don't lie on the nodal points (Fig. 3.1), the extrapolation method introduced above will be used to treat the interior boundaries, namely the immersed boundaries. Meanwhile, the interior nodes will be solve by either high-order schemes or second-order central-difference schemes. On the other hand, by manipulating the grid, we can also make the interior boundaries and grid nodes overlap with each other. Under these circumstances, there is no immersed boundary and the interior boundaries will be solved by one-sided schemes (2.9) and (2.12), which are third-order accurate. The high-order schemes will be adopted to solve the interior

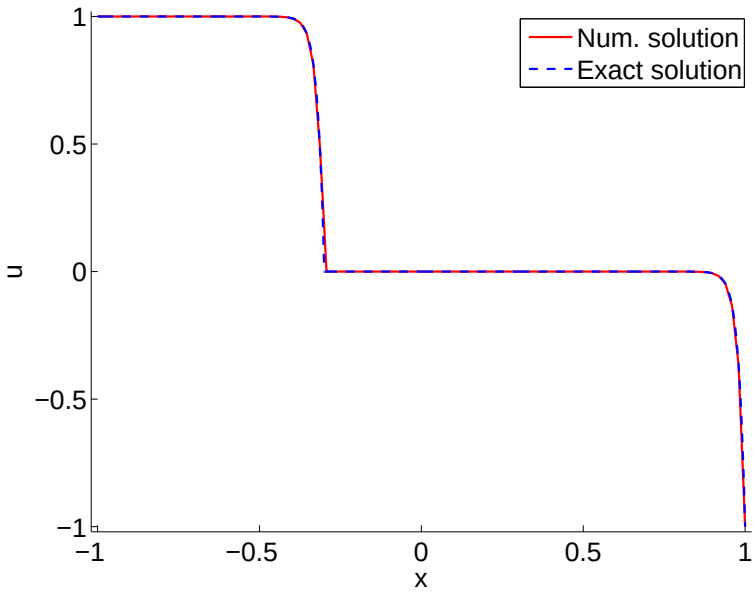


Figure 3.2: Steady state solution of the 1D advection-diffusion equation with N=99, and compact finite-difference scheme along with quadratic polynomial extrapolation.

nodes. Then, the results from different combinations will be compared with each other to assess the performance of both the compact scheme and the extrapolation method.

Fig. 3.2 shows the numerical solution with $N = 99$ and the exact solution. Fig. 3.3 to 3.6 are the log-log plots of the L_2 error-norm versus the grid spacing. The slope of these plots is the corresponding rate of convergence. We first verify that the method is second-order if a second-order central difference scheme is used for the interior discretization. The result is shown in Fig. 3.3, which shows that the combinations the second-order central difference scheme with a quadratic polynomial will give second-order rate of convergence. In Fig. 3.4, the interior boundaries overlap with the grid, so that the one-sided compact scheme will be used to discretize the internal boundary points while the symmetric compact scheme is used to solve the interior nodes. In Fig. 3.5, the grid no longer coincide with the internal boundaries. A quadratic polynomial will be used to extrapolate the value at the GCs. Comparing Fig. 3.4 and Fig. 3.5, we can find that both of them have nearly third-order rate of convergence, indicating that using quadratic polynomial to perform the extrapolation is just as efficient as us-

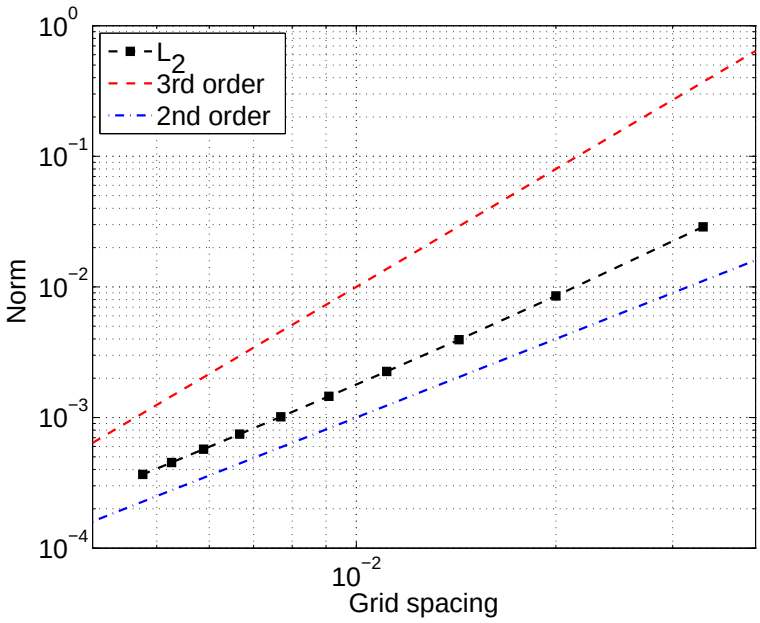


Figure 3.3: Convergence rate of the 1D advection-diffusion equation test. Interior nodes: second-order central-difference scheme. Immersed boundary: quadratic polynomial extrapolation.

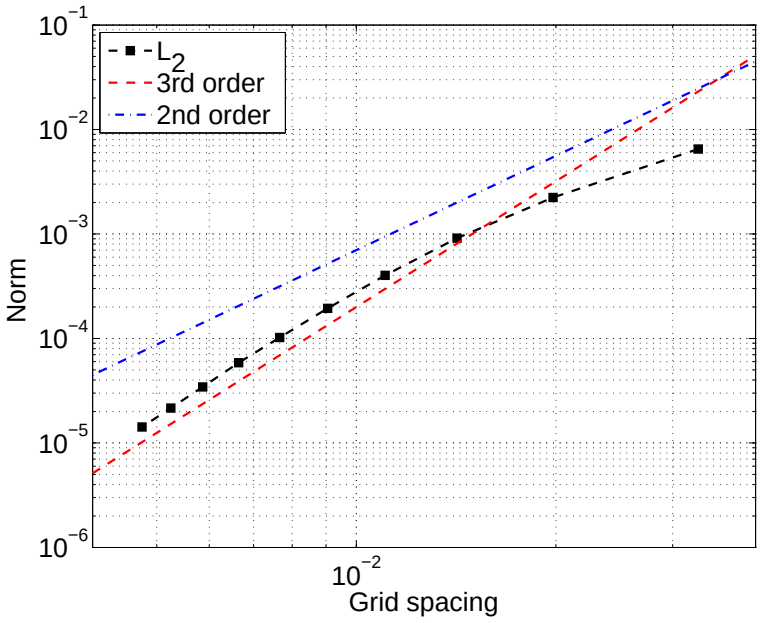


Figure 3.4: Convergence rate of the 1D advection-diffusion equation test. Interior nodes: compact finite-difference scheme. Immersed boundary: N/A (body-fitted grid and the one-sided scheme are used).

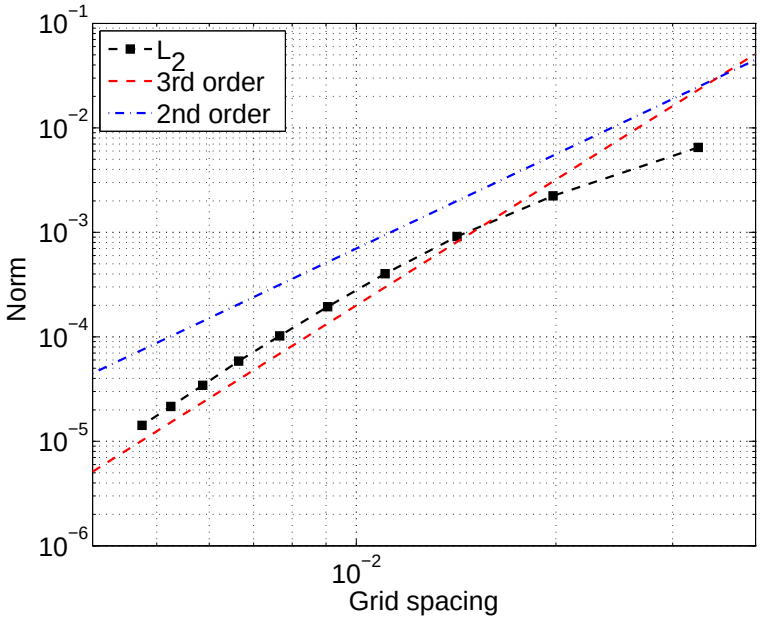


Figure 3.5: Convergence rate of the 1D advection-diffusion equation test. Interior nodes: compact finite-difference scheme. Immersed boundary: quadratic polynomial extrapolation, $m = 2$.

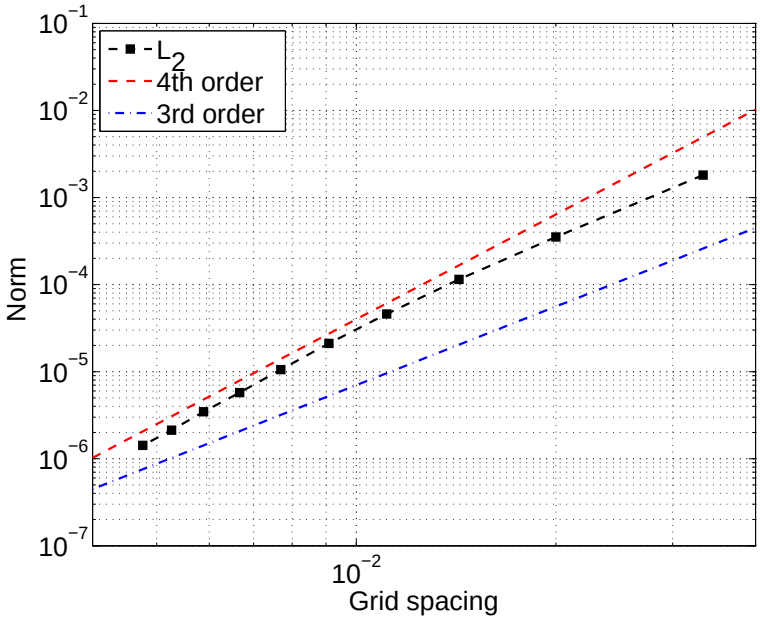


Figure 3.6: Convergence rate of the 1D advection-diffusion equation test. Interior nodes: compact finite-difference scheme. Immersed boundary: cubic polynomial extrapolation, $m = 3$.

ing the one-sided schemes, which requires that the grid has to conform to the internal boundary. More importantly, when it comes to two-dimensional and three-dimensional problems, the immersed boundaries rarely coincide with the grid nodes, and extrapolation/interpolation will be a more convenient choice.

Increasing the degree of the polynomial will help to increase the accuracy of the extrapolation. In Fig. 3.6, the cubic polynomial is used to replace the quadratic polynomial, and an overall fourth-order rate of convergence is achieved. From these results, we can tell the advantage of the current high-order scheme over the second-order scheme is obvious, since Fig. 3.3 shows that the latter scheme has only second-order rate of convergence.

3.2 One-dimensional Poisson equation

When using the projection method to solve the Navier-Stokes equations, finding the proper way to handle the Poisson equation is always a major problem, as this step is usually the most costly procedure. Thus, it is very critical to evaluate the performance of the current immersed-boundary algorithm on solving the Poisson equation. The test problem solved in this part can be described as

$$\frac{d^2p}{dx^2} = (\cos^2 x - \sin x)e^{\sin x}, \quad x \in [-1.0, -0.3] \cup [0.3, 1.0], \tag{3.5}$$

with following boundary conditions:

$$\begin{aligned} p &= 1.0 \quad at \quad x = -1.0 \\ \frac{dp}{dx} &= 0.0 \quad at \quad x = -0.3 \\ \frac{dp}{dx} &= 0.0 \quad at \quad x = 0.3 \\ p &= -1.0 \quad at \quad x = 1.0. \end{aligned}$$

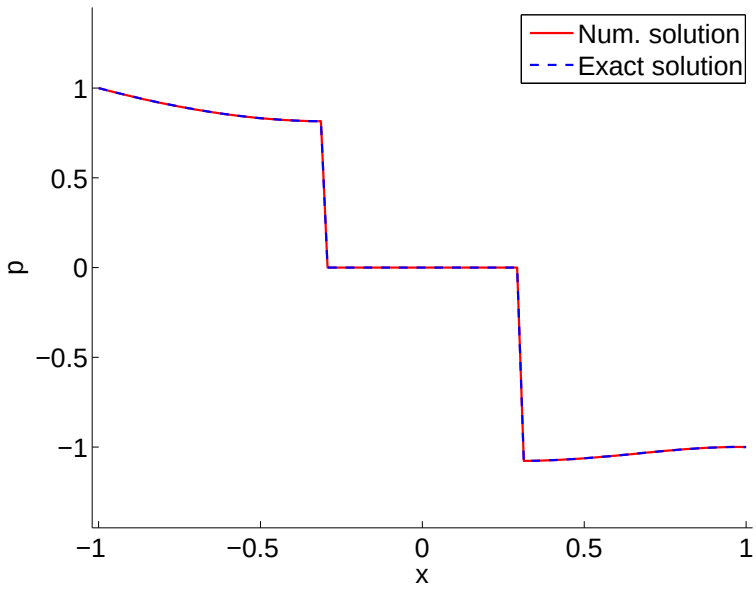


Figure 3.7: Solution of the 1D Poisson equation with N=99, compact finite-difference scheme and cubic polynomial extrapolation.

The exact solution is given by:

$$p(x) = e^{\sin x} - (1+x)e^{-\sin 0.3}\cos 0.3 + 1 - e^{-\sin 1}, \quad -1.0 \le x \le -0.3,$$

$$p(x) = e^{\sin x} + (1-x)e^{\sin 0.3}\cos 0.3 - 1 - e^{\sin 1}, \qquad 0.3 \le x \le 1.0.$$

The problem has mixed Dirichlet and Neumann boundary conditions. The compact scheme discretization of the second derivative can be written as

$$\mathbf{A}p'' = \mathbf{B}p. \tag{3.6}$$

Matrices **A** and **B** are banded matrices formed by the coefficients of the right-hand side and left-hand side of Eq. (2.6), respectively. Since we already know the exact value of p'', which is $(\cos^2 x - \sin x)e^{\sin x}$, we can substitute it into Eq. (3.6) and solve for p. The most critical problem is how to add the GCs into the matrix system.

Fig. 3.7 shows the shape of the solution of the Poisson equation. The boundary-conformal case, in which the interior boundaries are overlapped with the grid nodes

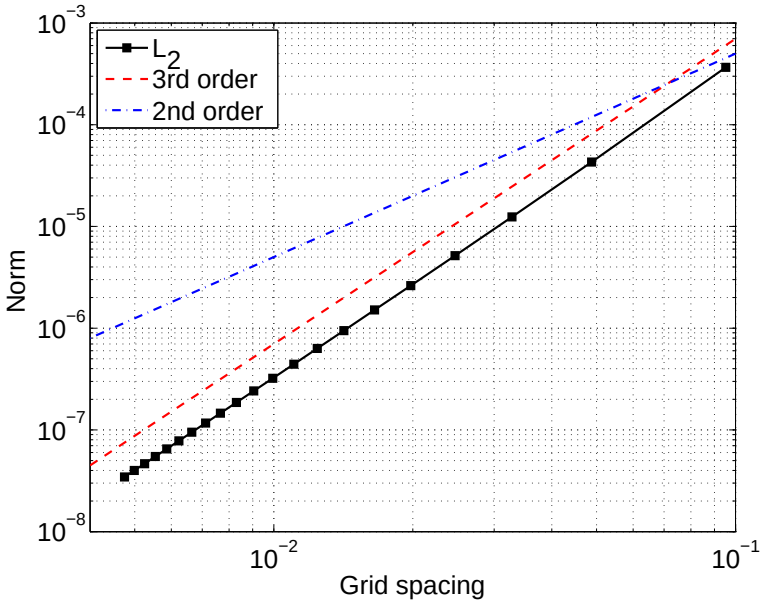


Figure 3.8: The reference case, of which the rate of convergence is shown. Interior nodes: compact finite-difference scheme. Immersed boundary: N/A (body-fitted grid).

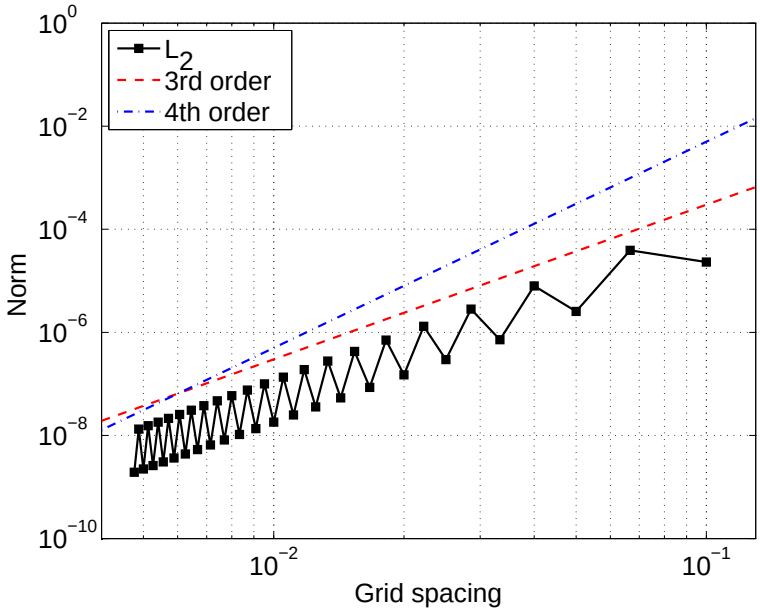


Figure 3.9: Convergence rate the 1D Poisson equation test. Interior nodes: compact finite-difference scheme. Immersed boundary: cubic polynomial extrapolation, $m = 3$.

(i.e., the body-fitted grid), will be solved as a reference. To incorporate the Neumann boundary condition at the interior boundary in the body-fitted grid, the following one-sided third-order schemes are used:

$$\left.\frac{df}{dx}\right|_{i=0} = \frac{1}{6h}[18(f_1 - f_0) - 9(f_2 - f_0) + 2(f_3 - f_0)],$$
$$\left.\frac{df}{dx}\right|_{i=N} = \frac{1}{6h}[18(f_N - f_{N-1}) - 9(f_N - f_{N-2}) + 2(f_N - f_{N-3})].$$

As we can see in Fig. 3.8, even though a fourth-order compact scheme is used to discretize the interior nodes, the overall rate of convergence is limited to third-order. This is expected since the third-order one-sided schemes and the interior boundaries have dominated the error.

When it comes to cases with immersed boundaries, special treatments are also required to take care of the GCs. In these situations, the following extrapolation scheme is used to represent the nodal value at the GC, x_{i+1}, instead of Eq. (3.2):

$$p_{GC} = \sum_{0}^{m-1} c_i' p_{i-m+1} + c_m' \left.\frac{dp}{dx}\right|_{x=x_L}. \tag{3.7}$$

In this equation, p_i is the nodal values at i-th node. With the help of this equation, we can include the GC into the matrix from Eq. (3.6). Fig. 3.9 is the L_2 error-norm of the solution using a cubic polynomial, $m = 3$. We can tell that the cubic polynomial in general gives us a third-order rate of convergence, which is comparable to the performance of the body-fitted grid discussed in Fig. 3.8. There are some oscillations in the convergence history, which are expected because as the grid is refined, the distance of the GC to the physical boundary is varying randomly and the accuracy of the extrapolation thus varies.

One advantage of using polynomial to do the extrapolation is that we can extend it to higher-order easily. Fig. 3.10 shows the convergence performance if we replace the cubic polynomial with a fifth degree polynomial ($m = 5$). The result shows that the rate

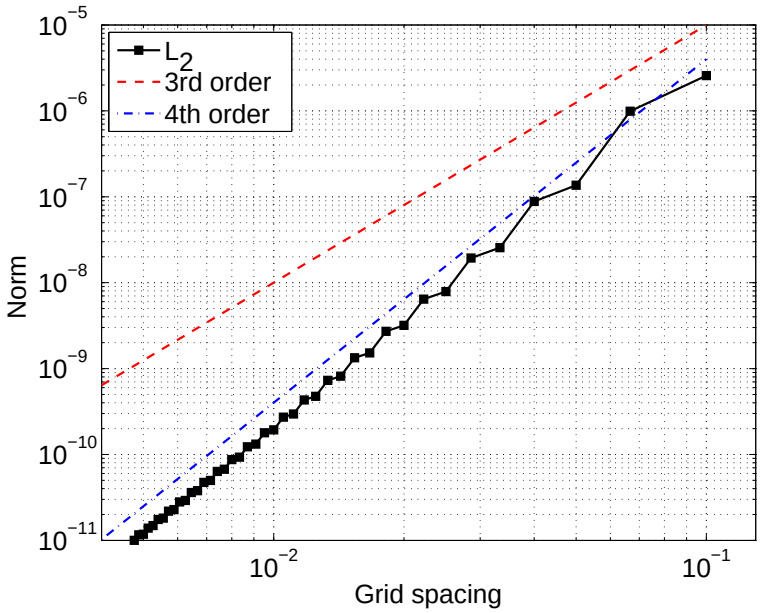


Figure 3.10: Convergence rate of the 1D Poisson equation test. Interior nodes: compact finite-difference scheme. Immersed boundary: extrapolation with a polynomial of fifth degree, $m = 5$.

of convergence is increased to fourth-order accordingly.

3.3 Conclusion

In this chapter we have presented a one-dimensional formulation of the high-order immersed-boundary method, and we have applied it to the one-dimensional advection-diffusion problem and one-dimensional Poisson equation to test the method's convergence and stability. The results indicate that the high-order immersed-boundary treatment does not affect the stability of the compact scheme. Moreover, proper choice of the extrapolation or forcing scheme can be used to achieve the desired convergence rate. These results have shown the promise of the current immersed-boundary method.

CHAPTER IV

TWO-DIMENSIONAL NUMERICAL TESTS

In the previous chapter, one-dimensional tests have shown promising results. However, we cannot jump to the conclusion that the new methodology will still work perfectly for the Navier-Stokes equations, as the equations are nonlinear and stiff in general. In this chapter, we first test the performance of the two-dimensional least squares approximation described in Chapter II. Then, the efficiency and accuracy of the high-order Poisson solver will also be evaluated. After that, two more numerical experiments are conducted to assess the performance of the complete program. The first study is designed to evaluate the performance of the compact scheme when there is no immersed boundary in the computational domain. The second test is used to test the combination of the compact scheme with the high-order immersed-boundary treatment. Results from both the second-order immersed-boundary method and the high-order one are presented.

4.1 Least squares treatment

Generally speaking, using cubic polynomials in the least squares method can give us at least fourth-order rate of convergence. However, if there is a immersed boundary in the domain, the data points will gather on one side of the interface. How this will affect the convergence rate is unclear. Moreover, when applying this least squares forcing treatment into fluid problems, we need to handle both the Dirichlet boundary condition from the velocity components and the Neumann boundary condition from the pressure term. Hence, it is also a problem to determine whether different boundary conditions will influence the performance of the lease squares method. In this section, a test is

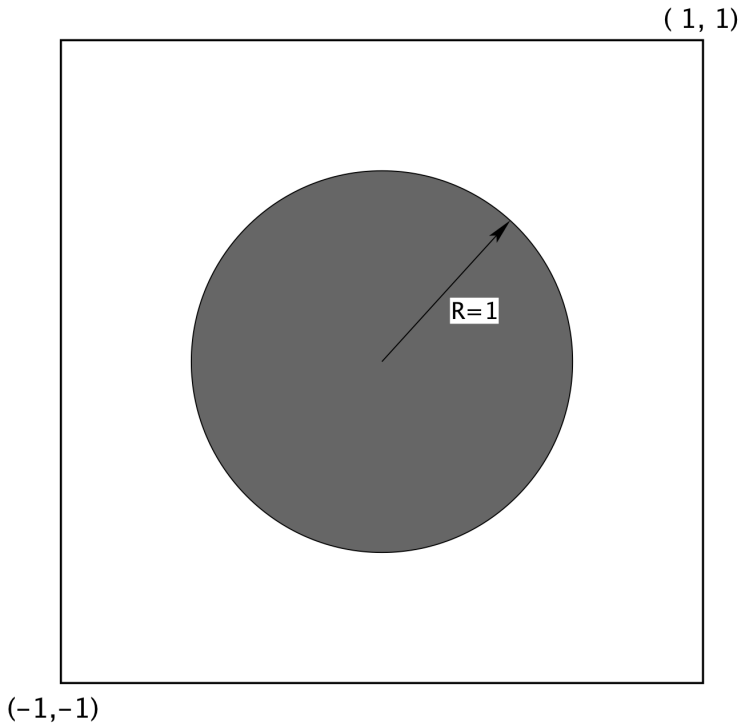


Figure 4.1: The computational domain used to test the least squares immersed-boundary treatment; the circle surrounding the grey area defines the immersed boundary.

designed to answer both questions.

A function and its derivatives,

$$\phi = e^{x+xy+y} \tag{4.1}$$

$$\frac{\partial \phi}{\partial x} = (1+y)e^{x+xy+y} \tag{4.2}$$

$$\frac{\partial \phi}{\partial y} = (1+x)e^{x+xy+y} \tag{4.3}$$

are defined in the blank region of a 2×2 domain with a circular immersed boundary at the center (Fig. 4.1). Since we already know the exact function, we can get both the nodal values and the derivatives at the interior boundary points and the grid points. We can get the nodal values at the hybrid cells through three different ways. First, we can calculate them through the function directly. Second, we can use the interpolation based on Eq. (2.21), which assumes a Dirichlet boundary condition at the boundary points.

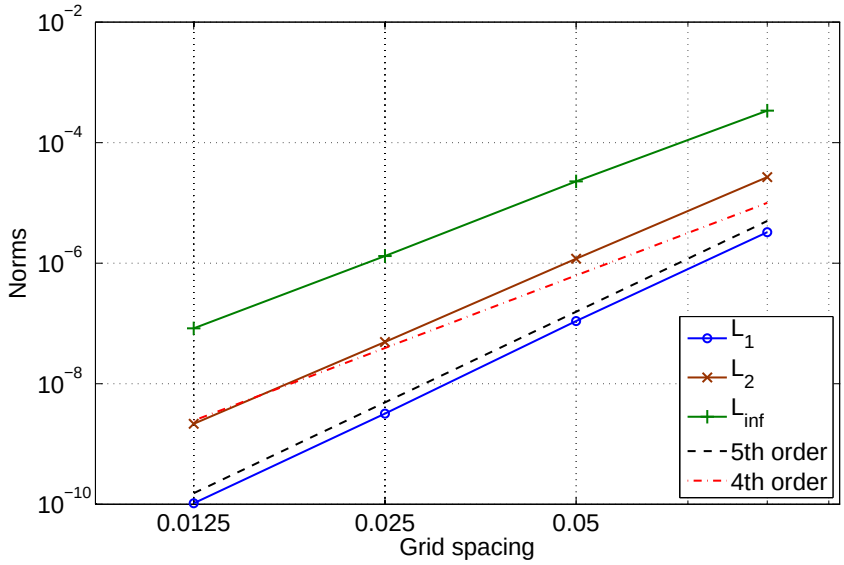


Figure 4.2: Rate of convergence of the least squares interpolation of function (4.1) when the Dirichlet boundary condition is applied at the interior boundary points.

Finally, we can also obtain those values through Eq. (2.22), which assumes a Neumann boundary condition at the boundary points. The interpolations will be conducted on a series of grids ($N \times N = 21 \times 21$, 41×41, 61×61 and 81×81). Results from the second and third method will be compared with the exact values, giving us the two convergence plots shown in Fig. 4.2 and Fig. 4.3. In these tests, the polynomial has degree of three, and the interpolate radius is large enough to make sure there are at least ten data points selected for each interpolation.

From both Fig. 4.2 and 4.3, we can see that the high-order feature is still remained when there is immersed boundary present. When the Dirichlet condition is applied (Fig. 4.2), this method shows a nearly fifth-order rate of convergence. Even though this high-order accuracy is not maintained that well when the Neumann boundary condition at the boundary points is used, a consistent fourth-order convergence rate is still observed in Fig. 4.3.

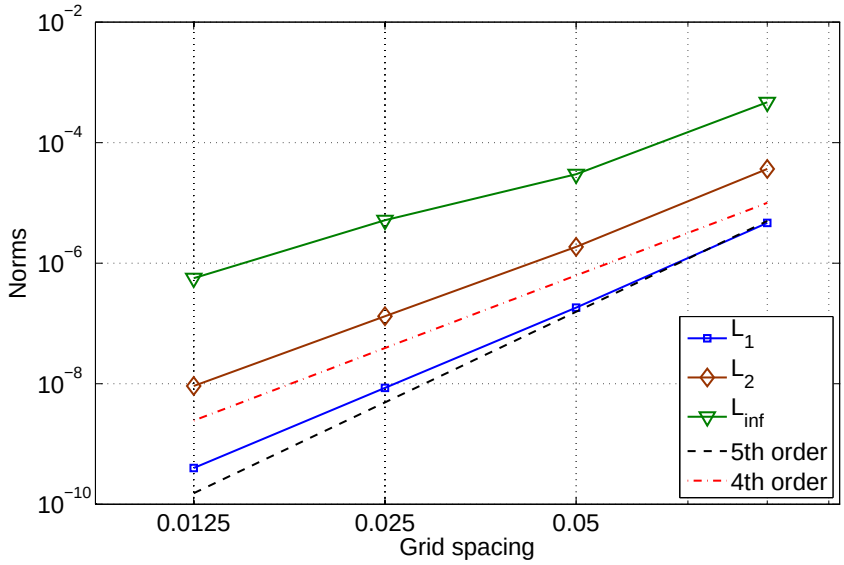


Figure 4.3: Rate of convergence of the least squares interpolation of function (4.1) when the Neumann boundary condition is applied at the interior boundary points.

4.2 A high-order Poisson solver for the interior points

When solving the multi-dimensional Poisson equation, we borrow the theory from Singer *et al.* [26] and use it in an innovative way to avoid the significant computational overhead. We manage to keep the structure of left-hand side of the equation the same as the standard central-difference schemes, so that all the efficient solvers developed to solve such kind of matrix systems can be applied easily in the current high-order approach. In this section, the new Poisson solver is combined with an ADI solver to solve the following problem.

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = -5 \sin x \sin 2y, \tag{4.4}$$
$$p(0, y) = p(\pi, y) = p(x, 0) = p(x, \pi) = 0.$$

The exact solution is:

$$p = \sin x \sin 2y.$$

More details about the method used to solve the equation is described in Section 2.2.5.

There will be no immersed boundary inside the domain in this test. Results from different grids, $N \times N = 21 \times 21$, 41×41, 61×61 and 81×81, will be compared with the exact solution to compute the error norms. Fig. 4.4 shows that a fourth-order rate of convergence is achieved by the new solver as desired. More importantly, since the current method only requires calculations of a few extra explicit terms, no significant overhead is observed when compared with the second-order solver.

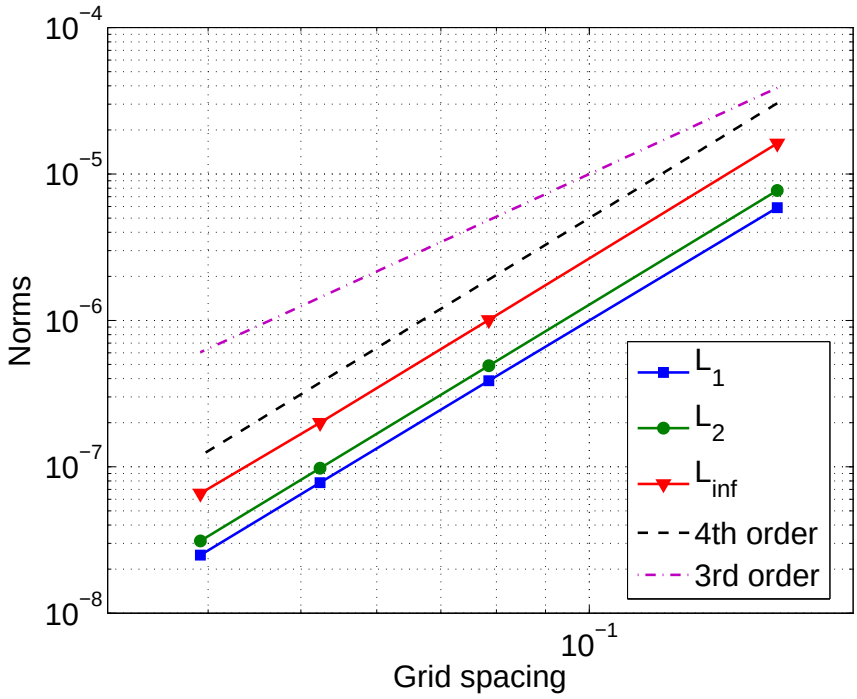


Figure 4.4: Rate of convergence of the new Poisson solver with correction terms for solving Eq. (4.4).

4.3 Kovasznay flow

Kovasznay flow [27] represents the flow behind a two-dimensional grid. Since its exact solution has already been found, it is often used to test the performance of CFD programs. In our study, exact solutions are applied at the exterior boundaries as the Dirichlet boundary condition. The domain is $-0.5 \leq x \leq 1.0$ and $-0.5 \leq y \leq 0.5$. The

streamlines of this flow are plotted in Fig. 4.5(a). There is no immersed boundary in this test. To get the convergence rate of the program, the same problem is solved using a hierarchy of grids of 30×20, 60×40, 120×80, and 240×160 points. The results are compared with the exact solution:

$$1 + u = 1 - e^{\lambda x} \cos 2\pi y, \tag{4.5}$$

$$v = \frac{\lambda}{2\pi} e^{\lambda x} \sin 2\pi y, \tag{4.6}$$

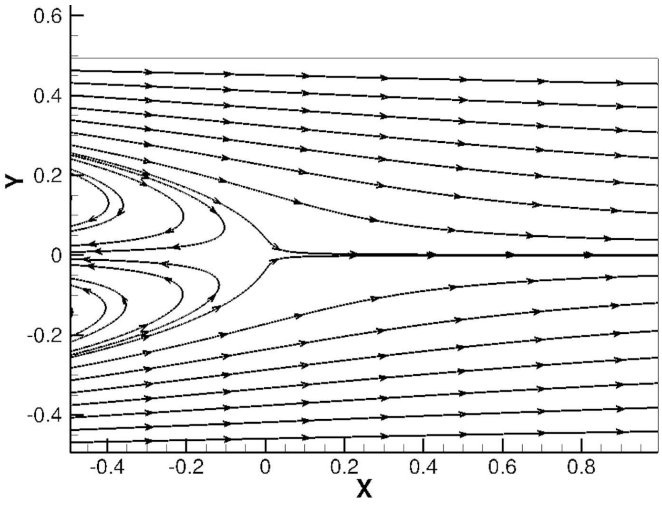
$$\lambda = \frac{\mathrm{Re}}{2} - \sqrt{\frac{\mathrm{Re}^2}{4} + 4\pi^2}, \tag{4.7}$$

where u and v are the velocity components, and Re is the Reynolds number, chosen to be 40 here. Fig. 4.5(b) shows the general trend of L_1, L_2 and L_∞ error norms of the solution on different grids. Two reference lines, denoting second-order and third-order convergence rate respectively, are also included in the figure. The results indicate that an overall fourth-order convergence rate can be achieved by this program, which is exactly what we expected.

4.4 Flow past a circular cylinder

The second test is about flow past a circular cylinder. The fourth-order compact scheme for the interior flow is combined with both the previous second-order immersed-boundary treatment and the current higher-order treatment to solve the entire domain. We want to use this study to assess the overall influence of the immersed-boundary method on the compact scheme. We set $Re = U_\infty d/\nu = 100$ in all the tests, where U_∞ is the free stream velocity, d is the diameter of the cylinder, and ν is the kinematic viscosity. Again, uniform Cartesian grids are employed in the study. The same problem is solved on a series of grids with 40×40, 80×80, 160×160, 320×320, and 640×640 points. Since the exact solution of this problem does not exist, we use the result from the 640×640 grid as the reference. Results from other coarser grids will be compared

(a)



(b)

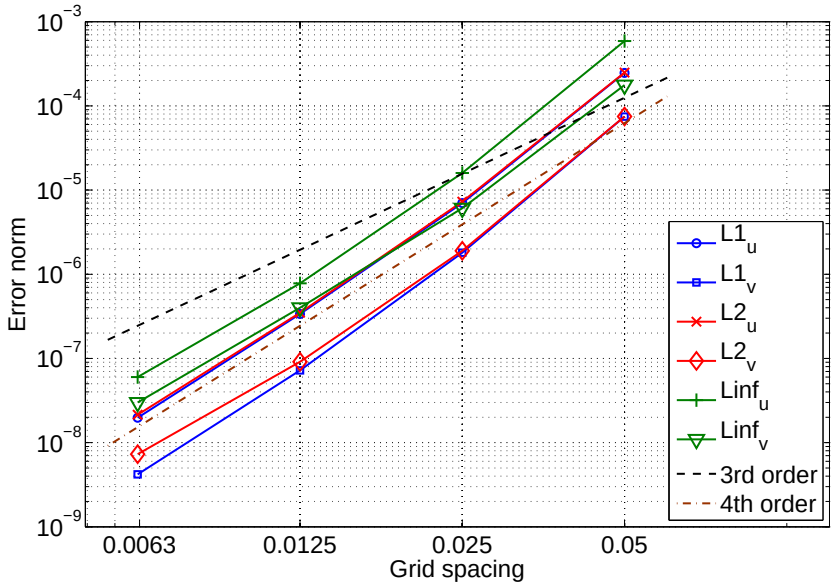


Figure 4.5: (a) Flow pattern of Kovasznay flow at Re=40. (b) L_1, L_2 and L_∞ norms of the error for the streamwise velocity u and transverse velocity v versus the grid spacing.

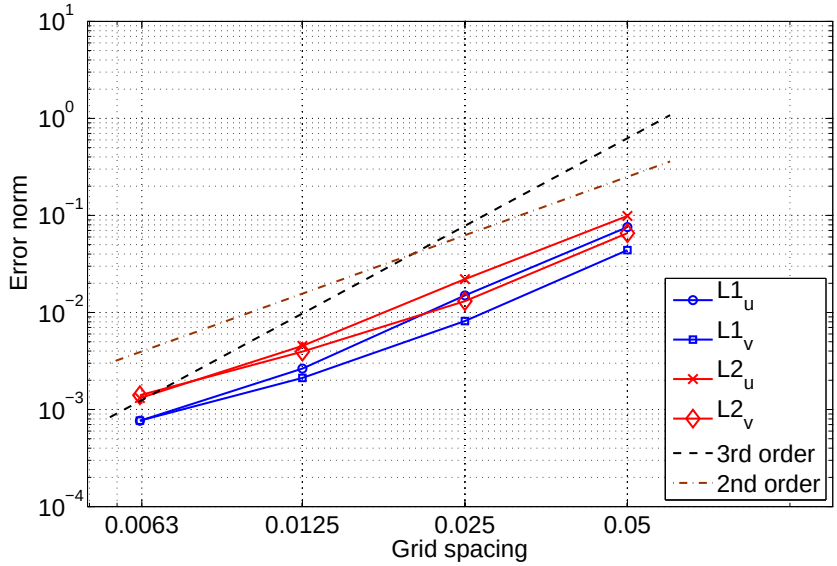


Figure 4.6: Convergence of the compact scheme combined with a previous second-order immersed-boundary method where a linear extrapolation is used to treat the immersed boundary. Plotted are the L_1 and L_2 norms of the error for the streamwise velocity u and transverse velocity v versus the grid spacing.

with this reference to compute errors. The time step is $\Delta t = 0.0001d/U_\infty$, and the computational domain is $-d \leq x \leq d$ and $-d \leq y \leq d$. Results at 2000th time step will be analyzed in the same way we showed in Section 4.3.

Fig. 4.6 shows the log-log plots of error norms versus grid spacing for the second-order immersed-boundary method. The overall convergence rate of the program is nearly second order. An important conclusion from this result is that the immerse-boundary treatment does not cause any particular problem to the compact scheme used for the interior. The result also means the immersed-boundary treatment plays a dominant role in determining the total convergence rate. Fig. 4.7 shows the error distribution in the domain. As is shown in the figure, errors are concentrated around the immersed boundary for both velocity components. This is another evidence that indicates how the immersed boundary is treated is very crucial. So our next step is to use the least squares method introduced earlier to improve the overall accuracy.

Using the same set-up as described earlier, we change the second-order extrapola-

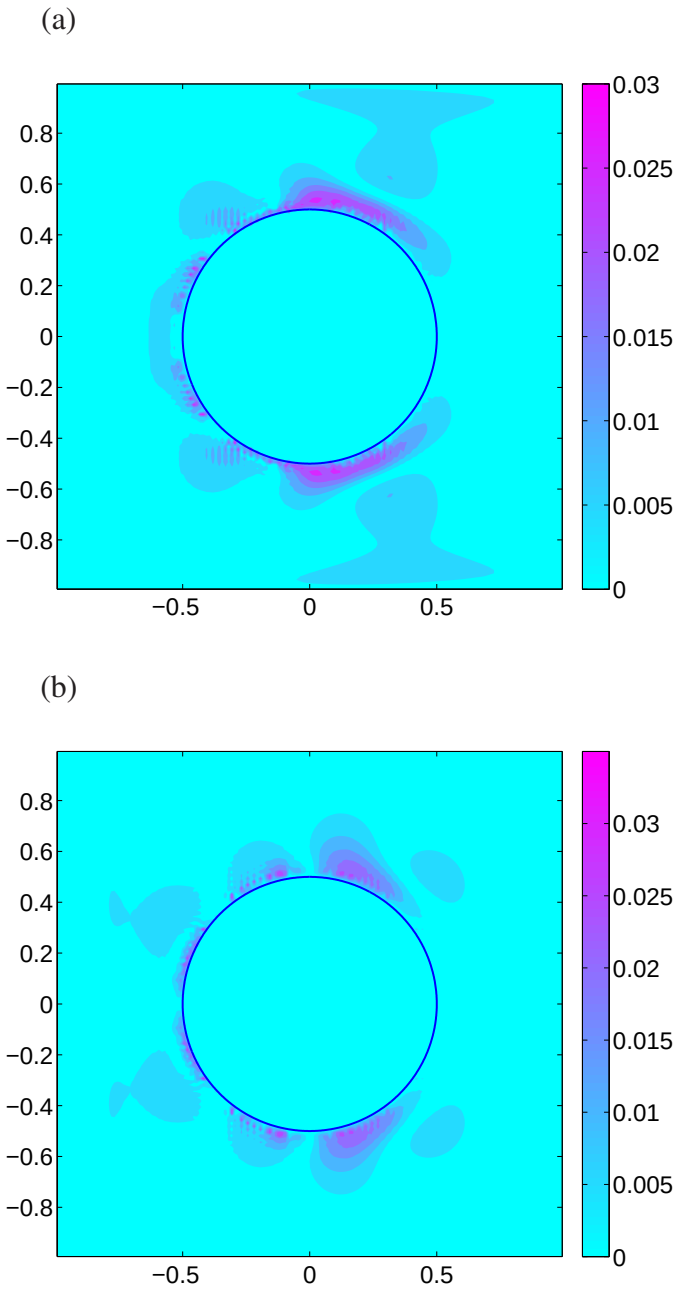


Figure 4.7: Error distribution of different velocity components on the 160×160 grid. (a) u-velocity; (b) v-velocity. The results from the 640×640 grid are chosen as references.

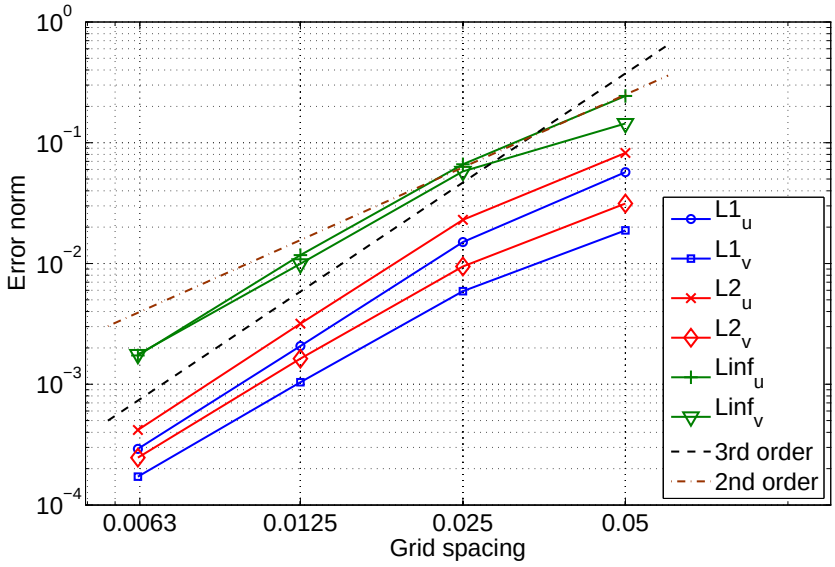


Figure 4.8: Convergence of the high-order immersed-boundary method, where the least squares method is applied at the immersed boundary. Plotted are the L_1, L_2 and L_∞ norms of the error for the streamwise velocity u and transverse velocity v versus the grid spacing.

tion into high-order least squares method described in Chapter II. Fig. 4.8 shows the new convergence plot after increasing the accuracy of the immersed-boundary treatment. The result from the finest grid (640×640) is still chosen as the baseline. As is shown in this figure, an overall third-order rate of convergence is achieved. The accuracy is reduced from the fourth-order likely because of the low-order treatment at the exterior boundaries.

In terms of the efficiency, the program has better performance than the original all second-order program. A lot of optimizations, such as using faster matrix system solver, finding the best sequence of loops, have been applied into the final code. The same flow past a circular cylinder case is solved using both the high-order program and the second-order program on the same platform (Ubuntu 12.04, Intel FORTRAN compiler, Intel Core i5-2400 CPU @ 3.10GHz). The computational time of 2000 time steps is recorded to evaluate the performance of these two programs. When the problem size is small, the high-order program spends most of the time doing the optimizations,

so that the second-order program still runs much faster. For example, for a 40×40 grid, it takes the high-order program 1393s to reach 2000 time steps, whereas it only takes the second-order one 724s. However, if the size of the problem increases, the time spent on solving the algebraic equations, instead of the optimizations, will dominate the computational time, leading to a better performance of the high-order program. For instance, when a 160×160 grid is adopted, to reach the same time step, the high-order one uses only 5416s, and the second-order one takes 8527s. For the fluid-structure interaction problems we are interested in, a relatively fine mesh is usually required to resolve the complex geometries; thus, the high-order program will still be very efficient.

4.5 Conclusion

In this chapter we have presented the method combining the compact finite-difference scheme and two immersed-boundary treatments. Kovasznay flow and flow past a circular cylinder are used to evaluate the performance of the method. In the cylinder test, even though the second-order immersed-boundary method limits the rate of convergence to second-order, the results are promising since the immersed-boundary treatment does not appear to affect the stability or accuracy of the compact scheme. Then after we apply the high-order least squares method, the rate of convergence of the program is raised to third-order. Furthermore, in terms of computational costs, we did not notice any obvious overhead caused by these high-order schemes.

CHAPTER V

CONCLUSIONS

5.1 Summary of present work

In this work, we aim to develop a high-order program to solve incompressible flow with interior boundaries by combining the compact scheme and a high-order immersed-boundary method. The body of this thesis can be divided into three parts. In Chapter II, the theory behind this work, including the compact scheme, the least squares treatment, as well as the high-order Poisson solver are detailed. Chapter III focuses on the one-dimensional tests of both the advection-diffusion equation and the Poisson equation. The results indicate that the high-order immersed-boundary method and the compact scheme are compatible with each other to provide a high-order rate of convergence. At the beginning of Chapter IV, two separate tests are designed to test if the proposed high-order immersed-boundary method and the high-order Poisson solver can achieve desired accuracy. Then our complete 2D code is utilized to conduct more benchmark tests. The Kovasznay flow test proves that the program is indeed fourth-order accurate if there is no immersed boundary present. The simulations of flow past a circular cylinder show the advantage of the high-order immersed-boundary treatment over the original second-order one.

5.2 Contributions of present work

The objective of this work was to develop an overall high-order immersed-boundary method code. In achieving this goal, we have made the following contributions.

- We have developed the 2D and 3D formulations for the fourth-order Poisson solver. These formulations can be used to develop the program to solve the

viscous incompressible Navier-Stokes equations without introducing significant computational overhead.

- We have developed a nominally fourth-order immersed-boundary method by combining the compact scheme and the least-squares based immersed-boundary method.
- We have performed both 1D and 2D tests to study the convergence of the high-order immersed-boundary method.

5.3 Directions for future work

5.3.1 Improve some details of the program

The program shows a lot of promise in dealing with problems with interior boundaries. Yet, there are several details in the program that can be improved. The first one is the exterior boundary treatment. According to our study, the program itself is limited to third-order convergence rate by the exterior boundary treatment, despite the fourth-order compact scheme and the least squares treatment. To achieve a truly fourth-order performance, high-order one-sided schemes for both Dirichlet boundary condition and Neumann boundary condition at the exterior boundaries need to be incorporated into the program.

Apart from the exterior boundary issue, when it comes to fluid-structure interaction and moving-boundary problems, it has been known that sharp-interface based immersed-boundary method may introduce artificial oscillations in the pressure [28, 29, 30, 31]. The issue will become more serious in our problem, since the low-dissipation, low-dispersion properties come with the high-order schemes. Fortunately, it has been addressed in several previous publications where different approaches were implemented to suppress the oscillation[17, 32, 33]. In our future work, some of these approaches may be incorporated in the current method.

5.3.2 Utilize the program to solve flapping wing problems

Recent interest in biomimetic micro air vehicles has motivated the study of the aerodynamics of flapping wings in nature. However, such wings typically involve complex kinematics, and the wing shape is time-dependent and consists of important three-dimensional features due to active and/or passive flexibility of the wing structure. Even though the Reynolds number of these biological wings is several orders of magnitude lower than typical aircraft, the numerical modeling often requires direct numerical simulations and is thus highly demanding. One of the major applications of the complete program is to solve such problems. However, the current program has some convergence issues when dealing with thin structures. Because of the special geometry of the thin structures, the data points for the interpolation will be biased towards certain directions, causing troubles in the least squares treatment. We find a way to circumvent this problem. Instead of using the complete form of the two-dimensional cubic polynomial, we keep the one that only contains 1, x, y, xy, x^2y, and xy^2 terms. As are shown in Fig. 5.1 and 5.2, however, if we redo the tests done in Section 2.1, we can find that this incomplete polynomial lower the rate of convergence of the least squares treatment by at least one. The impact of this incomplete polynomial on the program will be studied in the future, but we still manage to do several preliminary tests with it.

One of the tests is the flapping wing described in Yin *et al.* [34]. As is shown in Fig. 5.3, the wing movement is still controlled by the following prescribed translational and rotational motion at the leading edge, except that the wing is rigid in this test.

$$x_0(t) = \frac{A_0}{2}\cos(2\pi f t), \tag{5.1}$$

$$\alpha(t) = \alpha_0 + \beta\sin(2\pi f t). \tag{5.2}$$

$x_0(t)$ indicates the horizontal position of the leading edge, $\alpha(t)$ measures the angle between the wing and the x direction in the counterclockwise direction, A_0 is the stroke

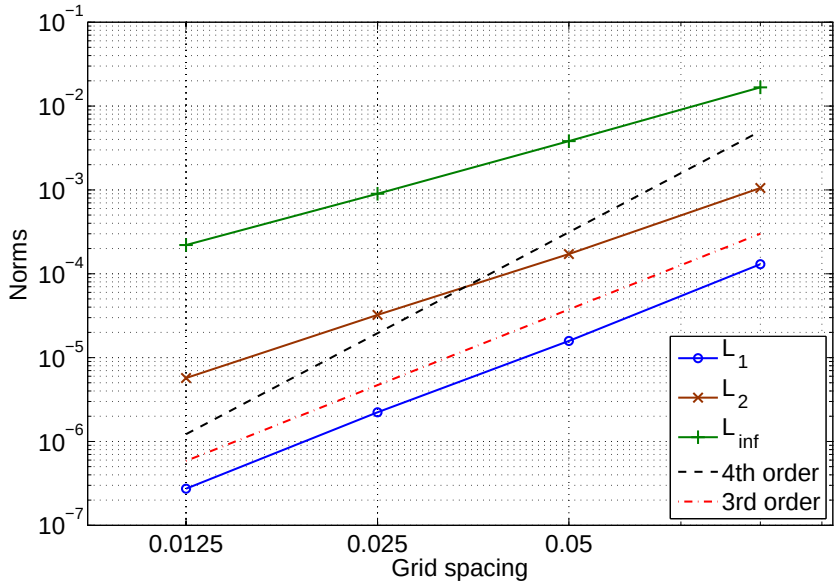


Figure 5.1: Rate of convergence of the incomplete least squares interpolation of function (4.1) when the Dirichlet boundary condition is applied at the interior boundary points.

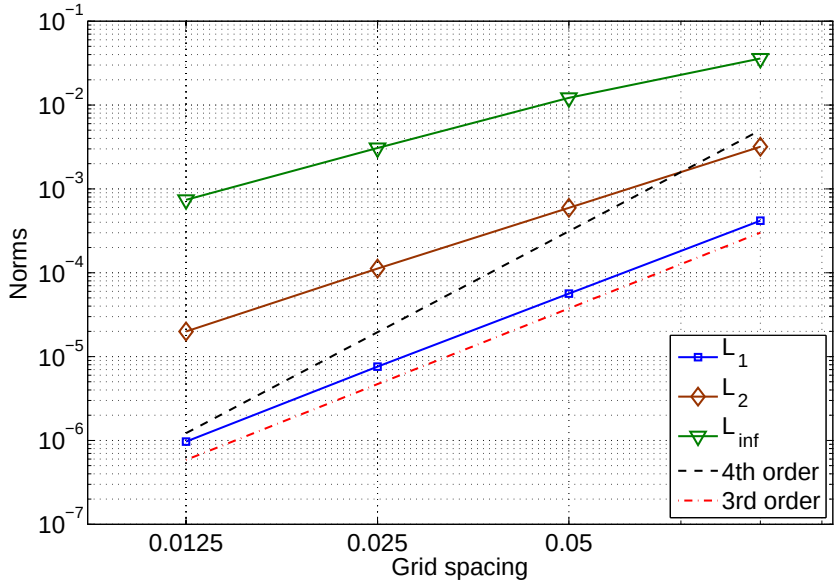


Figure 5.2: Rate of convergence of the incomplete least squares interpolation of function (4.1) when the Neumann boundary condition is applied at the interior boundary points.

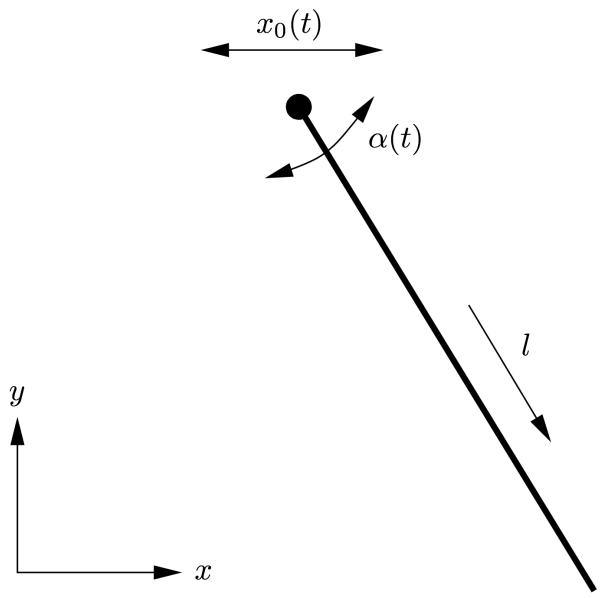
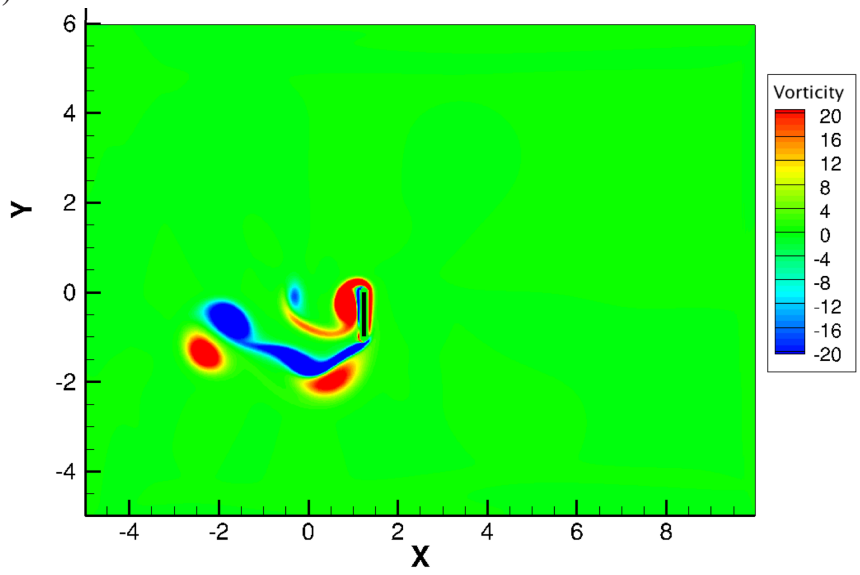


Figure 5.3: A schematic of the rigid wing during hovering flight.

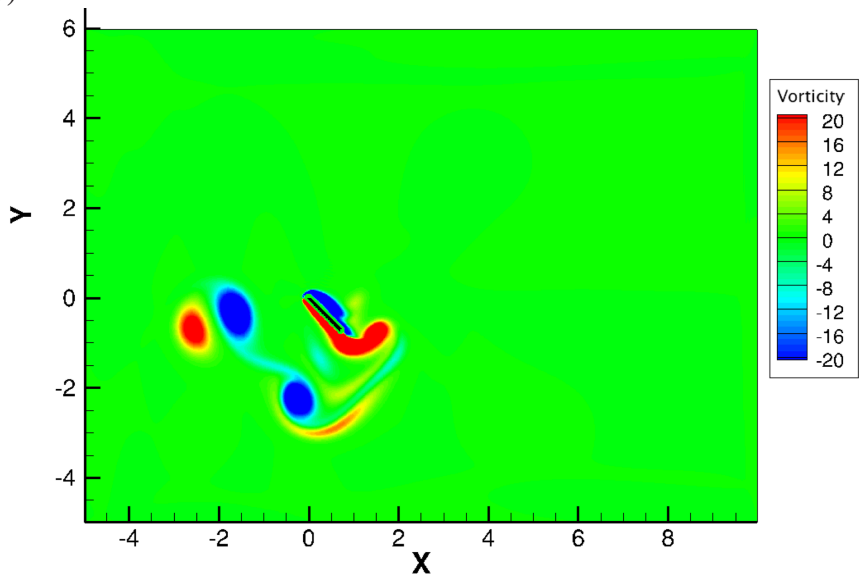
distance of the leading edge, α_0 determines the initial angle of the wing, β is the amplitude of the rotation angle and f is the flapping frequency. In the current test, we set $A_0/c = 2.5$, $\alpha_0 = -\pi/2$, $\beta = \pi/4$, and $Re = \pi A_0 f c/\nu_f = 30$, where c is the cord length and ν_f the fluid viscosity. Fig. 5.4 (a) to (e) show the vorticity field of the flapping wing in one cycle.

More carefully designed numerical experiments will be conducted in the future, and results will be compared with those from the original second-order program to evaluate the performance of the high-order program.

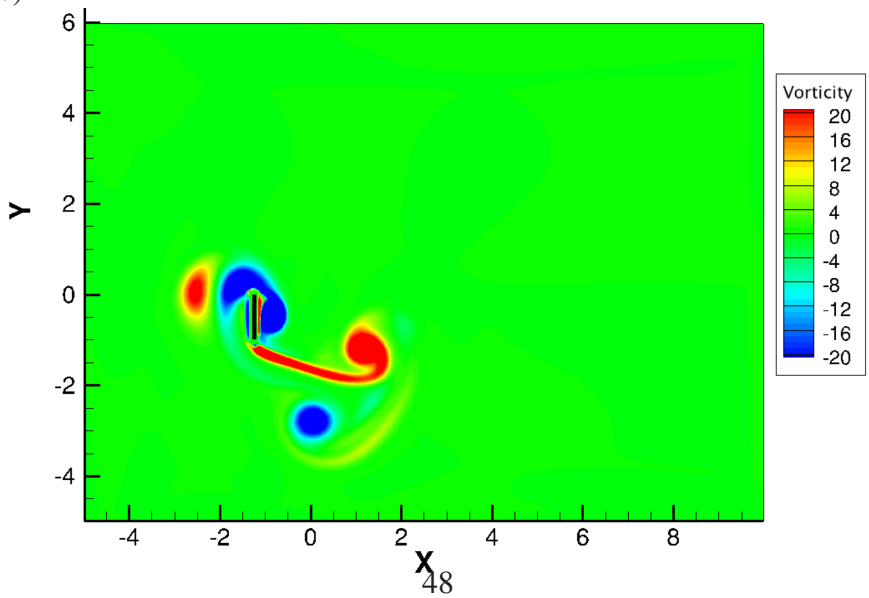
(a)


Vorticity
20
16
12
8
4
0
-4
-8
-12
-16
-20
Y
X
6
4
2
0
-2
-4
-4
-2
0
2
4
6
8

(b)


Vorticity
20
16
12
8
4
0
-4
-8
-12
-16
-20
Y
X
6
4
2
0
-2
-4
-4
-2
0
2
4
6
8

(c)


Vorticity
20
16
12
8
4
0
-4
-8
-12
-16
-20
Y
X
6
4
2
0
-2
-4
-4
-2
0
2
4
6
8

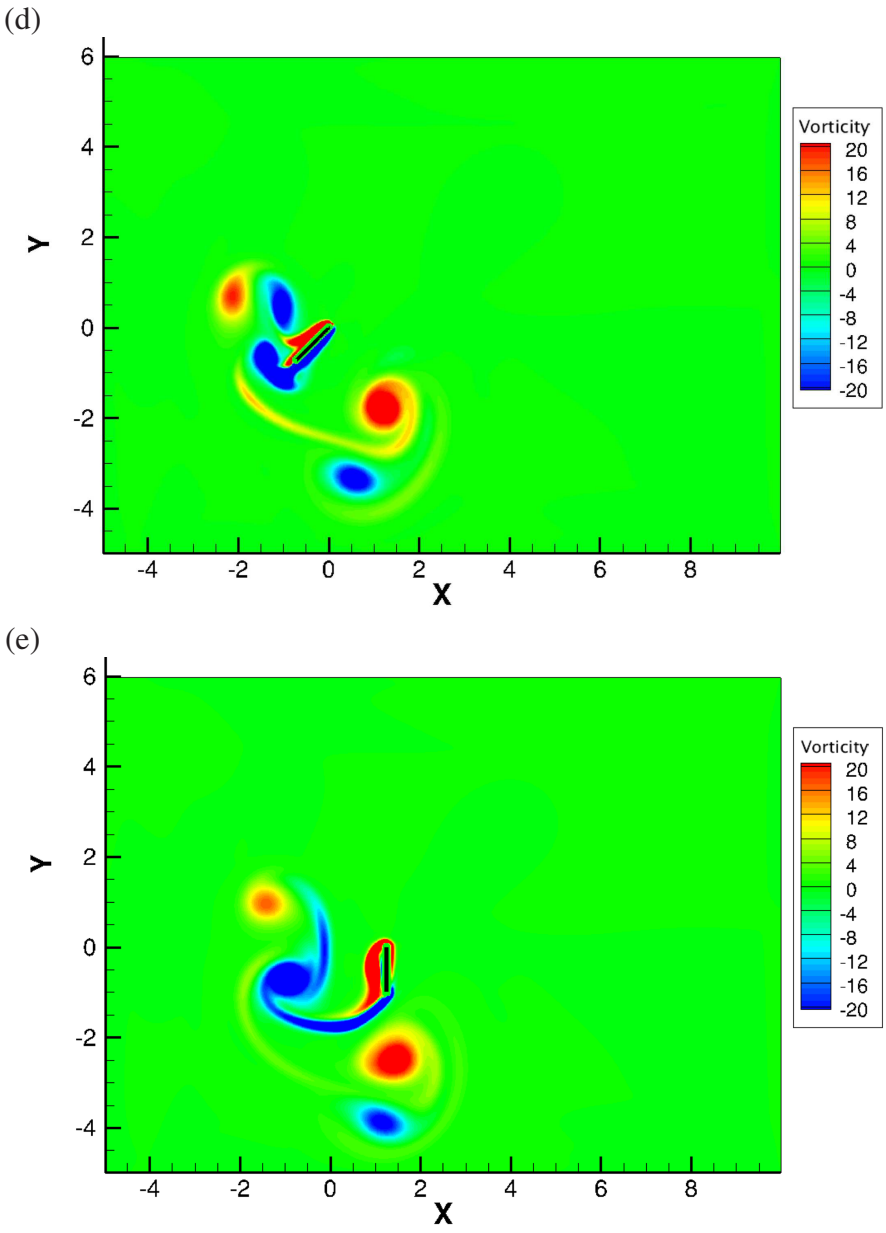


Figure 5.4: Vorticity field of the domain with a flapping wing in one cycle. (a) $t = T$; (b) $t = \frac{5T}{4}$; (c) $t = \frac{6T}{4}$; (d) $t = \frac{7T}{4}$; (e) $t = 2T$.

REFERENCES

[1] Joel H Ferziger and Milovan Perić. *Computational methods for fluid dynamics*, volume 3. Springer Berlin, 1996.

[2] C. S. Peskin. Flow patterns around heart valves: a numerical method. *J. Comput. Phys.*, 10:252–271, 1972.

[3] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[4] James Albert Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

[5] D Goldstein, R Handler, and L Sirovich. Modeling a no-slip flow boundary with an external force field. *Journal of Computational Physics*, 105(2):354–366, 1993.

[6] Tao Ye, Rajat Mittal, HS Udaykumar, and Wei Shyy. An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries. *Journal of Computational Physics*, 156(2):209–240, 1999.

[7] HS Udaykumar, R Mittal, P Rampunggoon, and A Khanna. A sharp interface cartesian grid method for simulating flows with complex moving boundaries. *Journal of Computational Physics*, 174(1):345–380, 2001.

[8] Yu-Heng Tseng and Joel H Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of computational physics*, 192(2):593–623, 2003.

[9] R. Mittal, H. Dong, M. Bozkurttas, F. M. Najjar, A. Vargas, and A. von Loebbeck. A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *J. Comput. Phys.*, 227(10), 2008. 4825-4852.

[10] Jianming Yang and Elias Balaras. An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. *Journal of Computational Physics*, 215(1):12–40, 2006.

[11] Jung Hee Seo and Rajat Mittal. A High-Order Immersed Boundary Method for Acoustic Wave Scattering and Low-Mach Number Flow-Induced Sound in Complex Geometries. *Journal of computational physics*, 230:1000–1019, 2011.

[12] Sylvain Laizet and Eric Lamballais. High-order compact schemes for incompressible flows: A simple and efficient method with quasi-spectral accuracy. *Journal of Computational Physics*, 228(16):5989–6015, September 2009.

[13] Xianyi Zeng and Charbel Farhat. A systematic approach for constructing higher-order immersed boundary and ghost fluid methods for fluid-structure interaction problems. *Journal of Computational Physics*, 231(7):2892–2923, April 2012.

[14] Y.C. Zhou, Shan Zhao, Michael Feig, and G.W. Wei. High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. *Journal of Computational Physics*, 213(1):1–30, March 2006.

[15] F Gibou and Ronald Fedkiw. A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem. *Journal of Computational Physics*, pages 1–40, 2005.

[16] Mark N. Linnick and Hermann F. Fasel. A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains. *Journal of Computational Physics*, 204(1):157–192, March 2005.

[17] Haoxiang Luo, Hu Dai, Paulo J.S.a. Ferreira de Sousa, and Bo Yin. On the numerical oscillation of the direct-forcing immersed-boundary method for moving boundaries. *Computers & Fluids*, 56:61–76, March 2012.

[18] Haoxiang Luo, Rajat Mittal, Xudong Zheng, Steven a Bielamowicz, Raymond J Walsh, and James K Hahn. An immersed-boundary method for flow-structure interaction in biological systems with application to phonation. *Journal of computational physics*, 227(22):9303–9332, November 2008.

[19] J.M.C. Pereira, M.H. Kobayashi, and J.C.F. Pereira. A Fourth-Order-Accurate Finite Volume Compact Method for the Incompressible Navier-Stokes Solutions. *Journal of Computational Physics*, 167(1):217–243, February 2001.

[20] Sanjiva K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, November 1992.

[21] Ratnesh K. Shukla, Mahidhar Tatineni, and Xiaolin Zhong. Very high-order compact finite difference schemes on non-uniform grids for incompressible Navier-Stokes equations. *Journal of Computational Physics*, 224(2):1064–1094, June 2007.

[22] Zhilin Li. A Fast Iterative Algorithm for Elliptic Interface Problems. *SIAM Journal on Numerical Analysis*, 35:230–254, 1998.

[23] Y Hoarau, D Faghani, and M Braza. Direct numerical simulation of the three-dimensional transition to turbulence in the incompressible flow around a wing. *Flow, turbulence and combustion*, pages 119–132, 2003.

[24] P Ferreira de Sousa and JCF Pereira. Fourth- and tenth-order compact fnite difference solutions of perturbed circular vortex fows. *International journal for numerical methods in fluids*, (December 2004):603–618, 2005.

[25] Raymond E Gordnier and Miguel R Visbal. Compact difference scheme applied to simulation of low-sweep delta wing flow. *AIAA journal*, 43(8):1744–1752, 2005.

[26] I. Singer and E. Turkel. High-order finite difference methods for the Helmholtz equation. *Computer Methods in Applied Mechanics and Engineering*, 163(1-4):343–358, September 1998.

[27] L. I. G. Kovasznay and Geoffrey Taylor. Laminar flow behind a two-dimensional grid. *Mathematical Proceedings of the Cambridge Philosophical Society*, 44(01):58, October 2008.

[28] Petter A Berthelsen and Odd M Faltinsen. A local directional ghost cell approach for incompressible viscous flow problems with irregular boundaries. *Journal of computational physics*, 227(9):4354–4397, 2008.

[29] Dartzi Pan and Tzung-Tza Shen. Computation of incompressible flows with immersed bodies by a simple ghost cell method. *International journal for numerical methods in fluids*, 60(12):1378–1401, 2009.

[30] Markus Uhlmann. An immersed boundary method with direct forcing for the simulation of particulate flows. *Journal of Computational Physics*, 209(2):448–476, 2005.

[31] Chuan-Chieh Liao, Yu-Wei Chang, Chao-An Lin, and JM McDonough. Simulating flows with moving rigid boundary using immersed-boundary method. *Computers & Fluids*, 39(1):152–167, 2010.

[32] Jung Hee Seo and Rajat Mittal. A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations. *Journal of computational physics*, 230(19):7347–7363, 2011.

[33] Jongho Lee, Jungwoo Kim, Haecheon Choi, and Kyung-Soo Yang. Sources of spurious force oscillations from an immersed boundary method for moving-body problems. *Journal of computational physics*, 230(7):2677–2695, 2011.

[34] Bo Yin and Haoxiang Luo. Effect of wing inertia on hovering performance of flexible flapping wings. *Physics of Fluids*, 22(11):111902, 2010.