Development of A Direct-Forcing Immersed-Boundary Method on Unstructured Meshes for Multi-Body Interactions in Air-Water Two-Phase Flows

by

Haixuan Ye

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Naval Architecture and Marine Engineering)
in the University of Michigan
2020

Doctoral Committee:

Associate Professor Kevin J. Maki, Chair
Associate Professor Krzysztof J. Fidkowski
Professor Armin W. Troesch
Professor Yin Lu Young

Haixuan Ye
hxye@umich.edu
ORCID id: 0000-0003-1555-2687

Dedicated to my family

ACKNOWLEDGEMENTS

I cannot begin to express my sincere gratefulness to my advisor Prof. Kevin Maki for his relentless support of my Ph.D study and research, for his extensive knowledge, encouragement and patience. The compilation of this work and my thesis would not have been possible without his invaluable guidance and profound belief in my abilities. I would also like to extend my appreciation to the other committee members, Prof. Armin Troesch, Prof. Krzysztof Fidkowski and Prof. Yin Lu Young for their precious and vital comments that helped to significantly improve the quality of this work.

I must thank Dr. Yang Chen and Dr. Grzegorz Filip for their ingenious suggestions through the experience of using my work. Special thanks should also go to all my colleagues of the Computational Ship Hydrodynamics Lab for their invaluable friendships, countless supports and inspiring discussions. I very much appreciate Dr. Ping He for his great help and guidance on PETSc. I am extremely grateful to Dr. Kevin Ellwood and Dr. Wanjiao Liu, who have been providing great insight to make this work practical for industrial applications, and constructive advice on the development path of this work.

To my parents, thank you for your unconditional love and support throughout my life. Thank you both for giving me strength to chase my dreams. I would also like to sincerely thank my wife Yutong for your love and unwavering support, for brightening up my day with your smile, and for the countless sacrifices you have made to help me get to this point. Thank you for being the best part of my life.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

ABSTRACT

A direct-forcing immersed boundary method (IBM) is developed in the framework of a finite-volume incompressible solver for high-Reynolds-number flows. The method solves governing equations on a background mesh whose grid lines do not conform to the concerned surface geometry, whereby the difficulty of generating high-quality body-fitted meshes is significantly reduced. The boundary conditions on the surface of the geometry are enforced through interpolation. A unique aspect of the proposed IBM is that the method is compatible with unstructured meshes, and as such can be combined with body-fitted meshes, so that some geometries can be represented by body-fitted meshes, and other geometries are represented by the IBM. The method provides an accurate solution for the cases of moving objects in both single-phase and air-water two-phase flows. The method can also be applied to both steady and unsteady, laminar and turbulent flows. In the current work, the method is implemented for solving the Reynolds-Averaged Navier-Stokes equations, and for turbulent flows, the Spalart-Allmaras turbulence model is used.

A noticeable challenge of using IBMs is the difficulty in resolving boundary layers at high Reynolds numbers. In this thesis a universal wall function is implemented, which provides a smooth velocity profile from the outer edge of the logarithmic region down to the wall. The wall function improves accuracy when the mesh is not sufficiently fine to resolve the viscous sublayer. As a result, the stringent requirement of near-wall cell spacing for high-Reynolds-number flows is significantly alleviated.

The Volume-of-Fluid (VoF) method is used for air-water two-phase flows. A field extension method is used to enforce the boundary condition of the volume fraction

on the immersed surface.

Detailed verification and validation studies are performed to demonstrate that the current method is second-order accurate. A careful comparison is presented between the results of the IBM, the experimental data, and other numerical results. The comparison fully demonstrates the accuracy and feasibility of the method by examining the flow field and the force on the immersed surface. The validation case of a ship advancing with a rotating rudder is also performed. The results demonstrate the accuracy, flexibility and efficiency when the IBM is used combined with unstructured body-fitted meshes.

CHAPTER I

Introduction

1.1 Background and Motivation

There are many ship hydrodynamic problems that involve moving geometries, and many of which occur at high Reynolds numbers. For example, the relative motion between the propeller and rudder is what determines the performance of a ship when it is operated in the calm water or a seaway. In high-Reynolds-number ship applications, the near-wall mesh quality is critical to accurately predict flow features such as separation and reattachment. It is a non-trivial problem to generate high-quality wall-conforming meshes to handle large curvature on the ship hull and appendages. It becomes increasingly difficult to preserve the mesh quality if the ship motion and the relative motion between different parts of the ship are considered. Some of the commonly used techniques are deforming meshes, sliding meshes, overset meshes and IBMs.

The identifying feature of an IBM is the governing equations are solved on a background mesh whose grid lines do not conform to the surface geometry, which makes mesh generation trivial. However, unlike body-fitted meshes, where prism layers with large aspect ratios can be used to resolve the boundary layer, an IBM requires near-wall cells to be sufficiently fine by refinement in all three directions in 3D. This results in many more cells for an IBM than its body-fitted counterpart

to achieve a similar near-wall cell spacing in the normal direction to the wall. In addition, this problem is more notable for high-Reynolds-number flows due to the thin boundary layer.

IBMs enforce the wall boundary conditions on the IBs through various interpolation schemes. Similar to overset meshes, IBMs can also handle various types of motions without having to deform the mesh. However, IBMs differ from overset meshes since overset meshes still use body-conforming meshes, and the cell sizes on different blocks of meshes need to be similar to each other in the overlap region. Whereas IBMs use non-conforming background meshes and no overlap region is required, which simplifies the process of mesh generation. Although overset meshes can solve the governing equations on the meshes that conform to the wall boundaries, IBMs can use meshes with higher quality in terms of non-orthogonality and skewness. The shape of near-wall cells significantly affects the accuracy of the underlying numerical predictions and the convergence of solution. It also greatly influences the process of mesh generation.

In the past several decades, IBMs have become increasingly popular because of the ease of implementation and the compatibility with other numerical methods. The IBM is first proposed by *Peskin* (1972) to simulate blood flows in the heart. Since then, there have been years of development of IBMs. Most of the research is focused on solving the N-S equations directly for low-Reynolds-number flows, single-phase flows and on pure structured meshes. Upon reviewing the development path of IBMs, this thesis proposes an IBM that is suitable for (1) high-Reynolds-number flows, (2) working on both structured and unstructured meshes, together with traditional body-fitted boundaries, and (3) both single-phase and two-phase flows.

1.2 Literature Review

In recent decades, engineering analysis has benefited greatly from the development of computational methods. For example, a broad family of techniques that falls under the umbrella of CFD makes it possible to solve partial differential equations that govern the flows on complicated domains around the geometry of a ship, ground vehicle, aircraft, etc.

Due to the development of computer hardware (CPU, storage, memory, etc.) and numeric algorithms, CFD methods are becoming increasingly important in studying fluid dynamics. CFD methods have their own irreplaceable advantages compared with analytical and experimental methods:

Analytical Methods Are usually based on a deep understanding of the fundamental physics and can provide a good (sometimes very accurate) estimation using approximately simplified mathematical models. On the one hand, the simplified models can be easy to use and very efficient, but on the other hand, they may be not accurate for new problems or when the underlying assumptions are violated.

Experimental Methods Are widely used and a reliable way to investigate many physical problems and to validate numerical methods, but their drawbacks are also obvious. Experimental equipment is usually expensive and harder to obtain with respect to computational resources. Additionally, experimental methods require significant effort to reconfigure the facilities to setup for each test case when the system parameters change.

CFD Methods Have a wide range of choices, like Reynolds-Averaged Navier-Stokes (RANS, *Ferziger and Peric* (2012)), Large Eddy Simulation (LES, *Sagaut* (2006)) and Direct Numerical Simulation (DNS, *Moin and Mahesh* (1998)), to model complex physical phenomena. Detailed information of the flow fields can be

gathered once the simulation is completed without using special instrumentation as is required for experiments. While there have been advances on many fronts, such as turbulence modeling, higher-order discretization schemes, etc., it is still challenging for numerical methods to obtain accurate predictions for many problems. To be more specific, one of the greatest challenges is to generate high-quality body-fitted meshes, and to maintain the quality when the object is moving or its shape is complex. Many approaches are proposed to alleviate the requirements on the burden of meshing and handling boundaries that are in motion, for example, overset mesh (*Carrica et al.* (2007); *Henshaw and Schwendeman* (2008)), sliding mesh (*Beaudoin and Jasak* (2008)), deforming mesh, mesh local reconstruction, and IBMs. A brief comparison is made below to justify why an IBM is proposed for this work.

Overset Meshes The main idea is to use several blocks of meshes (usually structured) with overlap regions to represent the computational domain. Communication between different blocks relies on the interpolation in the overlap regions. This strategy can reduce the difficulty of generating high quality body-fitted meshes, because an entire domain can be divided into regions, and the mesh for each region can be generated piece by piece. In addition, the quality of meshes can be preserved during relative body motions, since the motions are represented by the relative motions between different blocks of meshes without any deformation of each block. Ideally, overset meshes can support arbitrarily large relative motions. However, as the interpolation is two-way, similar cell spacing for different blocks of meshes are required in the overlap regions. A good balance should be made among the number of blocks used to represent a complex geometry, the computational efficiency, and the error introduced by the interpolation.

Sliding Meshes Is developed particularly for the circumstances when two or

more parts of the mesh must only slide relative to one another, either through translation or rotation. Grid lines do not need to match at the moving interface. The communication across the interface of different sets of meshes is achieved via interpolation. A good application of sliding mesh is to model a rotating propeller behind the ship hull.

Deforming Meshes Is a commonly used technique for the simulations involving body motions. During the simulation, the mesh deforms based on the motion of boundaries. The drawback is that the quality of the mesh is hard to control during a simulation, which may lead to inaccurate predictions or divergence of the simulation.

Local Reconstruction Automatically regenerates the mesh in large deformed regions during the simulation. For cases with millions of cells or more, the procedure can be time consuming, and the quality of the mesh is hard to control.

IBMs As indicated by the name, this category of methods embeds geometries on background meshes, and the geometries are represented as immersed boundaries (IBs). Generally, the influence of IBs on the flow field is represented in the form of a body forcing term in the governing equations. Unlike any of the methods above, it avoids the complexity to generate body-fitted meshes and preserves the mesh quality during mesh motion. The governing equations are completely solved on the high quality background meshes. Moreover, it has no limitation on the type of motion, such as with sliding meshes.

1.2.1 Overview of Immersed Boundary Methods

The IBM is first proposed by *Peskin* (1972) to simulate blood flows in the heart. Different from a body-fitted approach, the IBM is used on a pure Cartesian grid. The

grid lines do not conform to the complex geometry of the heart and the effect of the heart boundary is represented by introducing a body forcing term into the governing equations. Since then, modifications and improvements have been made to better reproduce the effect of the embedded boundary. Generally, the manner in which an IBM imposes the effect of embedded boundary on the governing equations is what distinguishes one method from another.

The simulation of an incompressible single-phase flow is described by the Navier-Stokes (NS) equations as below:

$$\nabla \cdot \mathbf{u} = 0 \tag{1.1}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla p + \mu \nabla^2 \mathbf{u} \tag{1.2}$$

where $\mathbf{u}$ and p are the fluid velocity and pressure, respectively. ρ is the fluid density, and μ is the dynamic viscosity.

In an IBM, the computational model is setup as in Fig. 1.1. The solid boundary S is embedded on a background mesh. The governing equations are solved on the

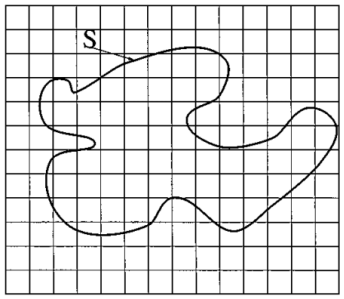


Figure 1.1: Sketch of an immersed boundary (*Fadlun et al.* (2000))

entire background mesh and the effect of the IB on the flow field is accounted for by introducing a forcing term $\mathbf{f}$ into the momentum equation:

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f} \tag{1.3}$$

Whether to introduce the effect of the immersed boundaries before or after the dis-

cretization of the governing equations classifies IB methods into two main categories: the continuous forcing approach or the direct forcing approach.

1.2.1.1 Continuous Forcing Approaches

Peskin (1972) first introduces an IBM using the continuous forcing approach to treat elastic boundaries. The blood flow in the heart is studied using a mixed Euler-Lagrangian finite difference solver. The IB is represented by a set of massless elastic fibers, which move with local fluid velocity. The locations of the elastic fibers are tracked in a Lagrangian fashion:

$$\frac{\partial \mathbf{X}_k}{\partial t} = \mathbf{U}(\mathbf{X}_k) \tag{1.4}$$

in which, $\mathbf{X}_k$ is the position of the kth Lagrangian point on the IB, and $\mathbf{U}$ is the local fluid velocity.

The effect of the IB is calculated from the fiber stress $\mathbf{F}$, which is related to the deformation of the fibers through a generalized Hooke's law. As the fiber stress $\mathbf{F}$ is calculated on the Lagrangian points which generally do not coincide with mesh points, a δ-function is used to spread the fiber stress to the neighboring mesh points as sketched in Fig. 1.2 and shown in Eqn. 1.5.

$$\mathbf{f}(\mathbf{x}, t) = \sum_k \mathbf{F}_k(\mathbf{X}_k)\delta(|\mathbf{x} - \mathbf{X}_k|) \tag{1.5}$$

To make the formulation suitable on a discrete mesh, the sharp δ-function is replaced by some smooth distribution function $d(\mathbf{x})$ as shown in Eqn. 1.6.

$$\mathbf{f}(\mathbf{x}, t) = \sum_k \mathbf{F}_k(\mathbf{X}_k)d(|\mathbf{x} - \mathbf{X}_k|) \tag{1.6}$$

The key factor in this approach is how to choose $d(\mathbf{x})$ since it determines how the

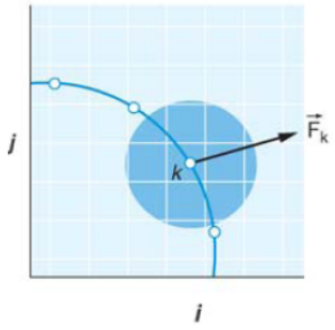


Figure 1.2: Spread of the stress force $\mathbf{F}_k$ at the kth Lagrangian point to the mesh points nearby (*Mittal and Iaccarino* (2005))

IB is represented. Different distribution functions as shown in Fig. 1.3 are employed in various studies by *Peskin* (1972); *Saiki and Biringen* (1996); *Beyer and LeVeque* (1992) and *Lai and Peskin* (2000). These methods are successfully applied for solving various problems involving flexible boundaries. However, regardless of the choice of the distribution function, the effect of the IB, which is represented by $\mathbf{f}(\mathbf{x}, t)$, is spread over several layers of mesh points as shown in Fig. 1.3. This effect smears the exact presence of an IB.

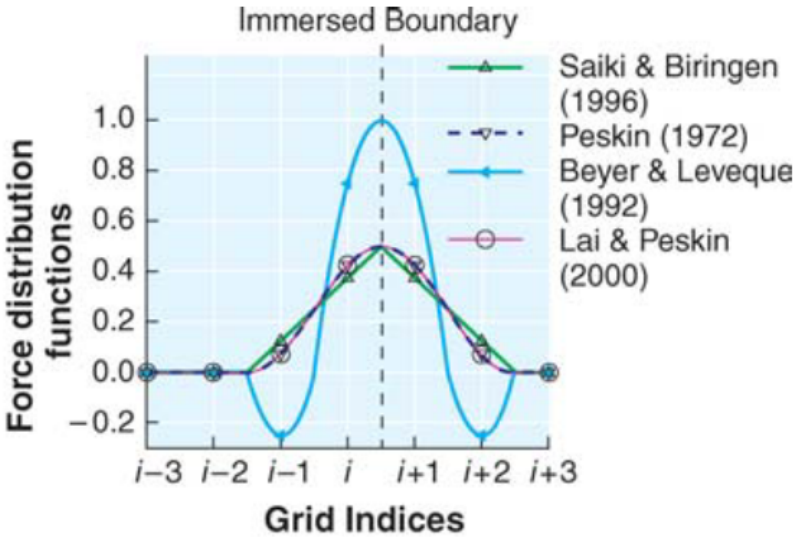


Figure 1.3: Distribution functions used in different studies (*Mittal and Iaccarino* (2005))

Goldstein et al. (1993) develops an IBM for problems with rigid boundaries. To circumvent the ill-posed situation of the generalized Hooke's law in the rigid limit, they propose to treat the body forcing $\mathbf{f}$ as a feedback mechanism in order to force the boundary to rest at its equilibrium position as shown in Eqn. 1.7. In this formulation, large negative values of α and β enforce the correct fluid velocity along the IB.

$$\mathbf{f} = \alpha \int_0^t \mathbf{U} d\tau + \beta \mathbf{U} \tag{1.7}$$

To represent a smooth geometry of the body surface, a narrow-band Gaussian distribution is used to spread the body force from the IB to the neighboring mesh points. The simulation is focused on low-Reynolds-number flows. Generally, larger Reynolds numbers require larger negative values of α and β, which can negatively impact the numerical stability. Moreover, there is no universal guidance about how to choose the values of these two variables, so a process of trial and error would be required for different applications to decide their values.

Saiki and Biringen (1996) extend the feedback forcing approach by including the velocity $\mathbf{v}$ of the Lagrange points on the IB in Eqn. 1.8 for the case of a moving boundary. They also use a fourth-order central difference approximation to mitigate the spurious flow oscillations on the IB.

$$\mathbf{f} = \alpha \int_0^t [\mathbf{U} - \mathbf{v}] d\tau + \beta [\mathbf{U} - \mathbf{v}] \tag{1.8}$$

Another approach, called the penalty method, is implemented by *Khadra et al.* (2000). They treat the IB as a porous medium and solve the flow field with the Navier-Stokes-Brinckman equation, which adds an additional term of volume drag, called Darcy drag, to the NS equation. Now the body force $\mathbf{f}$ in Eqn. 1.3 is the Darcy

drag as

$$\mathbf{f} = \frac{\mu}{K}\mathbf{U} \tag{1.9}$$

in which, K is the permeability of the IB. Theoretically, a zero value of K, corresponding to infinite large $\mathbf{f}$, represents impermeable solid boundary. Indeed, by taking $\alpha = 0$ and $\beta = \frac{\mu}{K}$ in Eqn. 1.7, the penalty method can be viewed as a specific formulation of the feedback forcing approach.

Dommermuth et al. (2007) develops an approach combining a penalty method and a VoF (*Hirt and Nichols* (1981); *Hibiki and Ishii* (2003)) method to simulate breaking waves around ships and the resulting hydrodynamic forces. However, they impose a free-slip boundary condition on the hull surface, which leads to solutions with smaller flow gradient near the slip boundary. Although this boundary condition is acceptable for many marine flows, it is not appropriate for cases where the no-slip hull boundary condition is important.

Abgrall et al. (2014) employs a penalty method on unstructured meshes for solving compressible flows. The IB is described as a zero level-set function. By using unstructured meshes, the mesh points are easy to cluster near the IB to ensure that the boundary layer has sufficient resolution. However, only stationary bodies like a NACA 0012 airfoil and laminar flows are considered.

All approaches that fall into the category of continuous forcing approaches introduce the forcing term into the momentum equation before its discretization. Thus, the implementation of a continuous forcing approach is independent from the underlying choice of discretization schemes, which makes this approach convenient for implementation into an existing solver.

However, due to the nature of these approaches, they are largely used for elastic boundaries. As introduced before, when applied to rigid boundaries, these methods suffer from the stability issue to varying degrees. In addition, as the IB generally does not coincide with mesh points, approaches in this category use a distribution

function to spread the body force from the exact position of the IB to the neighboring mesh points where the governing equations are actually solved. This feature leads to a diffuse representation of an IB over several cells. Since high-Reynolds-number flows exhibit thin boundary layers near the wall, the continuous forcing approaches become less favorable.

1.2.1.2 Direct Forcing Approaches

Researchers have also attempted to include the effect of IBs directly into the discretized governing equations. Consider the general form of advancing the momentum equation in one time step as

$$\frac{\mathbf{U}^{n+1} - \mathbf{U}^{n}}{\Delta t} = RHS + \mathbf{f} \tag{1.10}$$

in which, *RHS* includes the convection, diffusion and pressure gradient terms, and $\mathbf{f}$ is the body forcing term representing the effect of an IB.

First, consider the simplest situation in which the IB coincides with mesh points and all terms in *RHS* are discretized explicitly. Now the problem can be stated as: given the flow information at time step n, what value of $\mathbf{f}$ can yield $\mathbf{U}^{n+1}$ on the IB equal to $\mathbf{V}^{n+1}$, the velocity of the IB. In this case $\mathbf{f}$ is known as

$$\mathbf{f} = \frac{\mathbf{V}^{n+1} - \mathbf{U}^{n}}{\Delta t} - RHS \tag{1.11}$$

There are several aspects making the problem more complex, and different approaches aiming to overcome these difficulties distinguish themselves from one another.

First of all, IB generally do not coincide with mesh points. Thus the forcing term can be calculated either at the exact position of the IB or directly on the mesh points in the vicinity of the IB. In the first case, a distribution function is needed

to transfer the forcing term from the IB to the mesh points nearby. Similar to the continuous forcing approaches, the spreading process diffuses the IB to some degree. In the second case, the velocities in the vicinity of the IB need to be interpolated such that the desired velocity distribution on the IB is achieved. Thus, the accuracy of the method largely depends on how to formulate the interpolation.

Secondly, to enhance the numerical stability, especially at high Reynolds numbers, terms in the *RHS* are usually discretized implicitly, especially the diffusion term. The implicit discretization makes $\mathbf{f}$ depend on $\mathbf{U}^{n+1}$, thus solving $\mathbf{f}$ requires the inversion of a sparse matrix.

In addition, when body motions are considered, cells inside the IB at the previous time step can enter the fluid, which are called "freshly-cleared" cells (*Mittal and Iaccarino* (2005)). Special consideration of the time history of the primary variables (velocity, pressure or turbulent quantities) in these cells is needed.

Mohd-Yusof (1997) implements an IBM in a spectral method code. He suggests to calculate the forcing term directly from the difference between the solution of the governing equations and the desired velocity of the IB. Therefore, the desired velocity boundary condition on the IB can be imposed without any dynamic feedback process. A one-dimensional B-spline interpolation is used to obtain the velocity in the vicinity of the IB. This approach is used to accurately predict the laminar flow in a ribbed channel.

Based on Mohd-Yusof's work, *Shen and Chan* (2008a) employ a direct-forcing IBM with a VoF method to simulate wave interaction with submerged bodies. The free surface is treated as a boundary condition rather than fluid-fluid interface. Only the simulation of fully submerged bodies is carried out, and the interaction between the free surface and the IB is not considered.

Fadlun et al. (2000) extends Mohd-Yusof's work to three dimensions and solves the turbulent flow inside a motored IC piston/cylinder assembly. The velocity at the

first grid point external to the IB is linearly interpolated using the desired velocity on the IB and the velocity at the second grid point, which is obtained from solving the governing equations. The interpolation direction in Fadlun's approach is always along one grid line direction, but the choice is arbitrary, which can be problematic with complex three-dimensional boundaries as shown in Fig. 1.4(a).

To eliminate this ambiguity, *Balaras* (2004) further improves this approach by using the well-defined normal direction of the boundary and a linear interpolation. A comparison of the two interpolation methods is shown in Fig. 1.4. In his approach, the forcing term is applied to grid nodes external to the IB. By coupling this direct forcing approach to a LES solver, the flow around cylinder and the turbulent flow in a channel with a wavy wall are simulated. Very good agreement with analytical and numerical data is reported. *Balaras et al.* (2015) also extends the previous work by using a quadratic interpolation to reconstruct the velocities at the forcing points, and successfully simulates a rotating propeller behind a ship hull at Reynolds number $Re = 3.37 \times 10^5$.

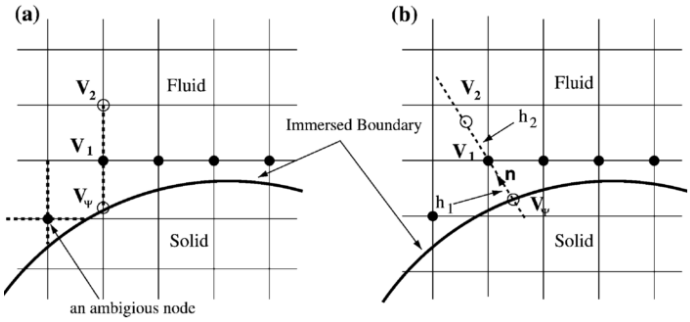


Figure 1.4: Different approaches of interpolation for the forcing points; forcing terms are applied to filled circles: (a) *Fadlun et al.* (2000) and (b) *Balaras* (2004)

Kalitzin and Iaccarino (2002) also improve Fadlun's work by applying the direct forcing method to the RANS equations. They incorporate a wall function to alleviate the restriction on the requirement of y^+ and accurately predicted the surface pressure and skin friction inside a turbine blade passage.

Based on the work of *Balaras* (2004), *Gilmanov et al.* (2003) and *Gilmanov and Sotiropoulos* (2005) develop a direct forcing approach suitable for complex 3D immersed boundaries on a Cartesian hybrid staggered/non-staggered grid layout. A Neumann boundary condition for pressure is explicitly imposed on the IB. They demonstrate that their method is second-order accurate by reconstructing the solution near the IB via a quadratic interpolation along the local normal direction.

Borazjani et al. (2008) applies a similar IBM on the curvilinear mesh to investigate fluid structure interaction. *Calderer et al.* (2014a) extends Borazjani's work by employing the level-set method to simulate the interaction between the free surface and floating structures. *Angelidis et al.* (2016) further extends their IBM to work on locally refined unstructured Cartesian meshes. The local refinement increases the local accuracy of interpolation near the IB. The turbulent flow around a hydro-kinetic turbine is simulated and they report very good agreement with published data.

Choi et al. (2007) proposes to reconstruct the normal and tangential components of velocity separately near the IB. They use a linear interpolation for the normal component, and a power-law function for the tangential component. The turbulent flows around simple geometries are simulated, and good agreement with other computational and experimental results is reported. However, the choice of the constants in their power-law function appears to be case specific.

Another way to enforce IB boundary conditions is the ghost-cell method. Ghost cells are defined as cells with centers inside the IB but have at least one neighboring cell in the fluid. By prescribing the velocity values in the ghost cells, IB boundary conditions are implicitly coupled into the momentum equation.

Majumdar et al. (2001) compares different interpolation methods for prescribing the velocity values in the ghost cells. However no significant advantage of higher-order interpolation is observed over the tri-linear interpolation (bi-linear in two dimensions). The concept of image points is adopted in order to ensure positive stencil weights in

the interpolation. A schematic of the concept of image points is shown in Fig. 1.5, in which G and I are the ghost point and image point, respectively. B is the foot of the perpendicular from G on the IB.

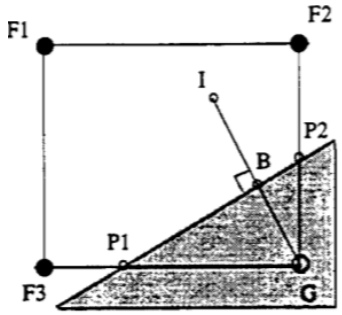


Figure 1.5: Schematic of the concept of image points (*Majumdar et al.* (2001))

Thus, the no-slip velocity boundary condition at B is enforced through the ghost point G as:

$$u_G = 2u_B - u_I \tag{1.12}$$

in which, u_I is interpolated from the solution of the governing equations at nearby mesh points.

Tseng and Ferziger (2003) develop a ghost-cell IBM suitable for both staggered and non-staggered Cartesian grids. A higher-order extrapolation for the values in the ghost cells is used to achieve higher-order accuracy.

Mittal et al. (2008) and *Ghias et al.* (2007) use a ghost cell approach and the concept of image point together with tri-linear interpolation (bi-linear interpolation for two dimensions) for the simulations of stationary and moving bodies at a wide range of Reynolds numbers.

Yang and Stern (2009) develop a ghost-cell method for wave-body interactions by using a finite-difference Cartesian-grid solver. The free surface is modeled with a level-set based ghost-fluid method.

Zhang et al. (2010) develops a level-set IBM for the interaction between free-

surface flows and structures. The problems of water entry and exit at a prescribed velocity, and a free-falling wedge are investigated. The predicted slamming coefficient is shown in good agreement with the experimental data.

Zhang et al. (2014) employs an IBM similar to *Yang and Stern* (2009) and a VoF method to simulate wave-body interactions. Unlike the work of *Shen and Chan* (2008a), their solver is able to simulate both fully submerged and surface-breaking bodies. The potential of the IBM together with a VoF method for simulations involving complex wave-structure interactions is demonstrated. However, only the prescribed motion and two-dimensional cases are considered.

For most of the works mentioned above, fractional step methods are used for solving the pressure-velocity coupling raised from the incompressible N-S equation. *Constant et al.* (2017) implements the IBM proposed by *Pinelli et al.* (2010) into the Pressure-Implicit Split-Operator (PISO) solver provided by OpenFOAM. A spreading function is used to impose the effect of the IB to the flow field. The solver is then carefully validated through test cases involving fixed and moving cylinders and spheres in a range of Reynolds number $Re = 30 \sim 3900$.

In general, due to the nature of direct forcing approaches, the IB is described as a sharp interface, which is very favorable for high-Reynolds-number flows. These approaches share the feature that they directly modify the discretized form of the momentum equation. In addition, these approaches largely depend on the specific spatial discretization schemes, thus increasing the difficulty of implementation compared with continuous forcing approaches.

1.2.1.3 Cut-cell Approaches

Cut-cell approaches are also based on Cartesian grids. *Ye et al.* (1999) proposes a cut-cell approach for simulating convection-dominated flows. Instead of a local reconstruction of velocities or distribution of the body force to represent the IB, cells

cut by the IB are reshaped by discarding the portions in the solid phase. They implement the approach in the finite-volume framework so that the solution satisfies the conservation laws. However, since the resulting cells need to be treated respectively due to their unique shapes, the method is difficult to extend from 2D simulations to 3D. In addition, very small cells emerging due to the complex shapes or motions of an IB could downgrade numerical stability.

1.3 Research Gap

From the overview of the history of IBMs, it is possible to conclude that:

- Most IBMs are based on structured meshes. Implementation on unstructured meshes is sparse. In addition, for simulating high-Reynolds-number flows over smooth surfaces, body-fitted meshes are still favorable for easily controlling the near-wall cell spacing with relatively fewer cells. However, few studies have been conducted on the combination of body-fitted meshes and IBMs. The hybrid approach would allow the use of body-fitted meshes for turbulent boundary layers and having IB representation in the regions where there is relative motion.

- Many works are focused on laminar flows, or LES simulations of turbulent flows. Few works have been conducted for implementing IBMs for RANS simulations, especially with wall functions.

- Study is sparse for the applications in high-Reynolds-number two-phase flows, especially involving the interactions between the free surface and the immersed surface.

- Compared with body-fitted meshes, all IBMs have the inherent disadvantage that more cells are required to achieving the same near-wall resolution.

1.4 Research Objectives

The goal of the proposed immersed boundary method is to provide a robust and accurate numerical tool for the simulations of ship flows in which there are multiple moving boundaries. The main objectives of the proposed work are summarized as:

- To implement a second-order direct-forcing immersed boundary method that is suitable for unstructured meshes.
- To combine body-fitted meshes and the IBM for simulations involving relative motions between different surfaces.
- To implement for RANS simulations based on the Spalart-Allmaras turbulence model. In addition, a universal wall function is implemented to alleviate the requirement of near-wall cell spacing for high-Reynolds-number applications.
- To implement for the problems with multiple moving objects in high-Reynolds-number air-water flows.
- To conduct thorough verification and validation with increasing complexity to demonstrate the robustness and efficiency of the proposed numerical tool.

The thesis is organized as following: Chapter II discusses the numerical details of the proposed IBM for single-phase laminar flows, including the representation of the IB, formulation of the forcing term in the momentum equation, and interpolation of physical variables in the vicinity of IBs. In addition, the governing equations, numerical discretization schemes, and the pressure-velocity decoupling method are described. Results of various test cases are represented to verify the code and demonstrate the order-of-accuracy of this method. The cases include pure convection and diffusion problem on both structured and unstructured meshes, manufactured solution of 2D Laplacian equation, flow around a 2D circular cylinder inside a cavity, and an oscillating cylinder in a cavity.

Chapter III describes the implementation for turbulent flows, including the underlying turbulence model, the implementation of the wall function and the enforcement of the turbulent variables near IBs. Subsequently, numerical simulations including the turbulent flow over a flat plate, the turbulent flow inside a 2D diffuser, the attached turbulent flow around an oscillating airfoil section, and the 3D turbulent flow around a model-scale KVLCC2 are presented to validate the accuracy of the solver.

Chapter IV discusses the proposed method for air-water two-phase flows. The VoF method and the enforcement of the corresponding boundary conditions are described in detail. Numerical results of the dam-break problems and the water exit of a cylinder are presented to validate the code.

In Chapter V, the turbulent air-water two-phase flow around a ship model with a rotating rudder is presented. The ship model is described through a body-fitted mesh while the rotating rudder blade is modeled using the IBM. Hydrodynamic forces are carefully compared with the experimental data and other numerical results. The results demonstrate the accuracy, flexibility and robustness of the combined usage of body-fitted meshes and the IBM.

Finally, Chapter VI summarizes the contributions of this work and provides suggestions for further developments in the future.

CHAPTER II

Numerical Methods

In this chapter the underlying governing equations, numerical discretization schemes, and the velocity-pressure decoupling method for laminar flow applications are first described. Subsequently, the implementation of the IBM is discussed in detail. Finally, numerical results are presented to demonstrate the order-of-accuracy of the proposed method.

2.1 Governing Equations

In this chapter, incompressible single-phase viscous laminar flows are considered. In addition, density is assumed to be uniform. Such flows can be described by the Navier-Stokes (N-S) equations, which are written as:

$$\nabla \cdot \mathbf{u} = 0 \tag{2.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = \nabla \cdot \left[\nu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)\right] - \nabla p \tag{2.2}$$

in which, $\mathbf{u}$ is the velocity vector; ν is the kinematic viscosity, and p is the pressure divided by the density. Boundary conditions are applied accordingly for $\mathbf{u}$ and p. Typically, on the wall boundary, the Dirichlet boundary condition $\mathbf{u} = \mathbf{u}_{\text{wall}}$ is used, where $\mathbf{u}_{\text{wall}}$ is the velocity of the solid boundary. The homogeneous Neumann

boundary condition is used for p.

2.2 Numerical Discretization and Solution Procedure

The governing equations are discretized by the Finite Volume Method (FVM). By applying the Gauss theorem, the volume integration of each term in the governing equations is expressed as the summation of the face fluxes. Subsequently, variables on face centers are interpolated from the neibouring cell values. For the convection term, a second-order upwind scheme is used to interpolate variables from cell centers to face centers. Linear interpolation is used for surface-normal gradients appearing from discretizing of the Laplacian term. In addition, explicit correction is applied to account for the non-orthogonality of the meshes (*Demirdžić* (2015)). Correction for the skewness of the meshes is not considered. Both implicit Euler and backward schemes are considered for the time derivative terms in unsteady simulations.

The PISO algorithm is used to solve the governing equations. The semi-discretized form of the momentum equation Eqn. 2.2 can be expressed as:

$$a_p \mathbf{u}_p = \mathbf{H}(\mathbf{u}) - \nabla p \tag{2.3}$$

in which, $\mathbf{u}_p$ is the unknown velocity at cell centers. a_p is the diagonal coefficient resulting from the discretization. $\mathbf{H}(\mathbf{u})$ is composed of all the off-diagonal terms, including the convection, diffusion, the explicit part of the time derivative term, and all source terms. Assuming the right hand side of Eqn. 2.3 is known, the velocity at cell centers can be expressed as:

$$\mathbf{u}_p = \frac{1}{a_p}\left(\mathbf{H}(\mathbf{u}) - \nabla p\right) \tag{2.4}$$

Continuity is satisfied by substituting Eqn. 2.4 into Eqn. 2.1 and rearranging the

terms, so that a Poisson equation is derived to solve the pressure as follows:

$$\nabla \cdot \left(\frac{1}{a_p} \nabla p_{\text{new}} \right) = \nabla \cdot \left(\frac{1}{a_p} \mathbf{H}(\mathbf{u}) \right) \tag{2.5}$$

After the pressure is solved, the velocity at cell centers is corrected by:

$$\mathbf{u}_p = \frac{1}{a_p} \left(\mathbf{H}(\mathbf{u}) - \nabla p_{\text{new}} \right) \tag{2.6}$$

By applying the Gauss's theorem to Eqn. 2.5, the velocity flux on cell faces is also updated:

$$\phi = \mathbf{S} \cdot \left(\frac{1}{a_p} \mathbf{H}(\mathbf{u}) \right)_f - \mathbf{S} \cdot \left(\frac{1}{a_p} \nabla p_{\text{new}} \right)_f \tag{2.7}$$

in which, $\mathbf{S}$ is the area vector of each cell face, and its direction is normal to the face pointing out of each cell.

The complete solution process of the PISO algorithm is summarized as:

1. The momentum equation Eqn. 2.2 is first solved using the velocity and pressure fields from the previous time step or previous iteration, and a predicted velocity field is obtained. The diagonal coefficient a_p and the operator $\mathbf{H}(\mathbf{u})$ are constructed based on the predicted velocity field.

2. The pressure Poisson equation Eqn. 2.5 is constructed to update the pressure.

3. Velocity at cell centers and face flux are updated using the new pressure field by Eqn. 2.6 and Eqn. 2.7. Subsequently the simulation is advanced to the next time step.

In practice, step 2 and 3 are repeated several times to solve the solution implicitly at the new time level.

2.3 Immersed Boundary Method

The core idea of the IBM is to solve the governing equations on background meshes. The governing equations are modified properly to represent the influence of solid boundaries. There are two essential steps for using the present IBM to solve single-phase laminar flows. In the first step, the cells on the background mesh are categorized into three types of cells:

- **Fluid cells** Are the cells in the realistic fluid domain where the governing equations are normally solved.
- **Forcing cells** Are the cells in the vicinity of the immersed surface. The governing equations are modified to take into account the presence of the immersed surface.
- **Solid cells** Are the rest of the cells on the background mesh. The numerical solution on these cells does not have physical meaning because they are outside the realistic fluid domain.

In the second step, a forcing term is calculated in each forcing cell through interpolation. The forcing term is then added to the governing equation such that the boundary conditions on the immersed surface are correctly reflected in the numerical solution. In the following sections, these two steps are introduced in detail.

2.3.1 Cell Categorization

The first step of using the proposed IBM is the cell categorization, and it is achieved in two steps. Fig. 2.1 illustrates the typical mesh setup for the present IBM to simulate the flow around a circular cylinder. The circular cylinder is represented using an IB surface, denoted as $\mathcal{S}$. It should be noted that the dots in Fig. 2.1 is only for the purpose of visualization of the cylinder cylinder, and it does not mean the cylinder is represented as a set a discrete points.

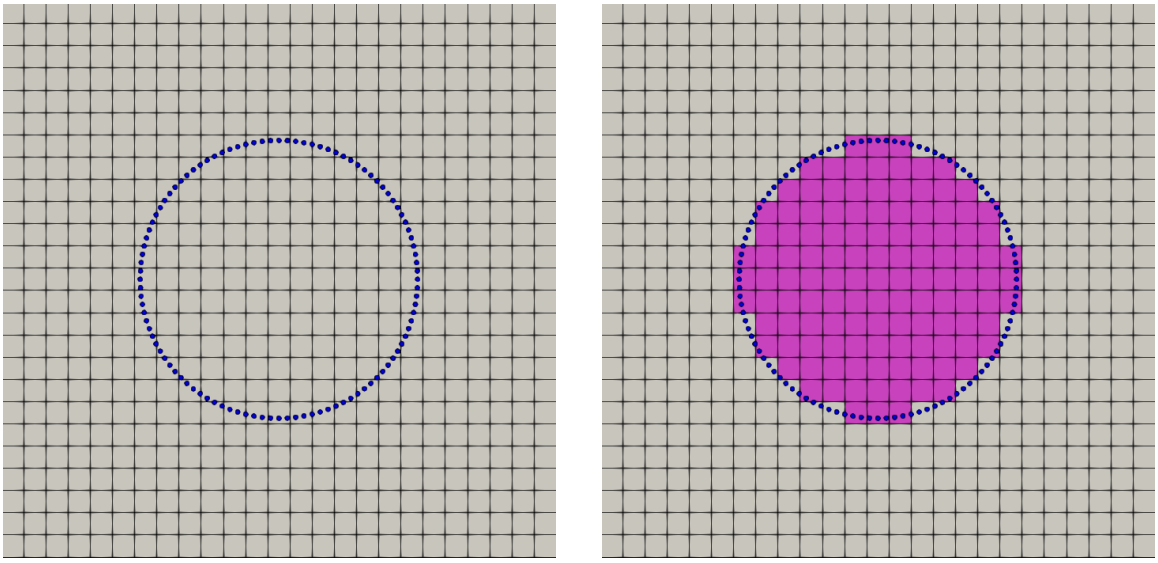


Figure 2.1: First step of the cell categorization. Left: before; right: after

The first step of the cell categorization is to classify the background cells into two groups, depending on whether the cell centers are inside or outside **S**. The term "outside **S**" refers to the flow region of interest (for example, the flow between the cylinder and the boundaries of the background mesh in Fig. 2.1).

To achieve this goal, the solver first reads in the immersed surface $\mathcal{S}$ via a STL file, which represents $\mathcal{S}$ as a triangulated surface. Next, the axis-aligned bounding boxes tree data structure (AABB tree) is used to detect the relative positions between cell centers and $\mathcal{S}$. The AABB tree is provided in the Computational Geometry Algorithms Library (CGAL), which is an open source C++ library of computational geometry algorithms. The AABB tree component offers a static data structure and algorithms to perform efficient intersection and distance queries against sets of finite 3D geometric objects (*Alliez et al.* (2019)). After the immersed geometry is read into the program, the corresponding AABB tree is generated. For every cell center of the background mesh, a ray through the cell center with arbitrary direction is generated. The number of intersections between the ray and $\mathcal{S}$ is then calculated efficiently via the AABB tree. If the number of intersections is even, the cell center is outside $\mathcal{S}$; otherwise, the cell center is inside $\mathcal{S}$. Fig. 2.1 shows the result of the

cell categorization, where the cells with centers inside $\mathcal{S}$ are colored in pink. There is a very rare situation that the ray happens to be tangent to $\mathcal{S}$, which will result in a fault identification. However, it has never became a problem from all the simulations discussed in this thesis.

The second step of the cell categorization is to further classify the two groups of cells into fluid, forcing and solid cells. Two algorithms to identify these cells are introduced, and the relative positions between these cells and $\mathcal{S}$ depend on the choice of the algorithm. The algorithms also have an impact on the implementation and the performance of the IBM, which are discussed later in this chapter. The two algorithms are summarized as:

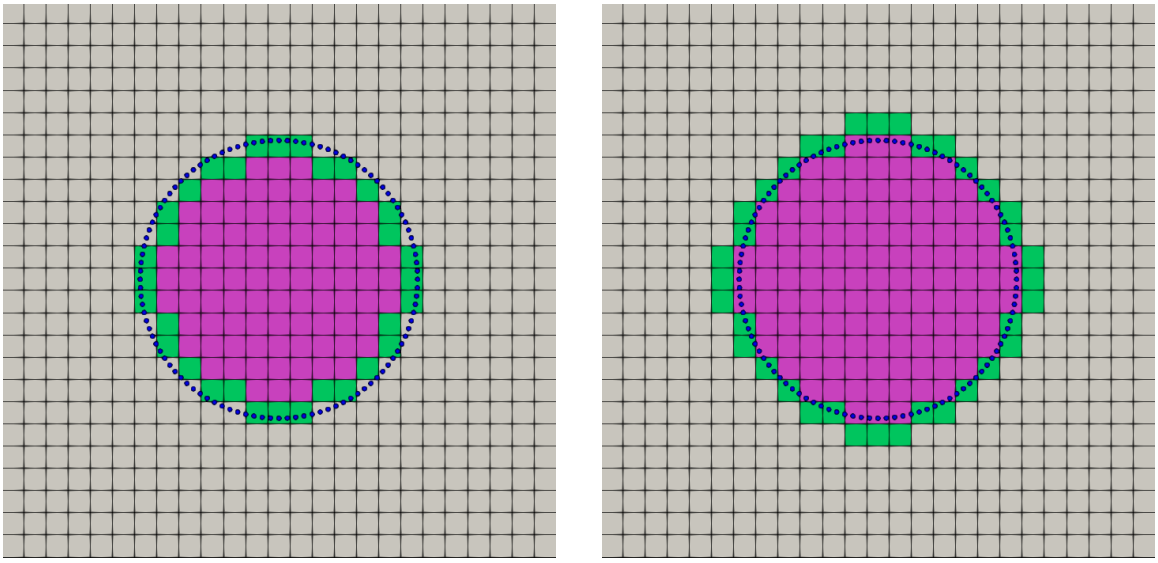
- **Algorithm 1**. For cells with centers inside $\mathcal{S}$, if it has at least one neibouring cell with its center outside $\mathcal{S}$, it is a forcing cell. Otherwise it is a solid cell. In addition, all the cells with centers outside $\mathcal{S}$ are fluid cells.

- **Algorithm 2**. For cells with centers outside $\mathcal{S}$, if it has at least one neibouring cell with its center inside $\mathcal{S}$, is is a forcing cells. Otherwise it is a fluid cell. In addition, all the cells with centers inside $\mathcal{S}$ are solid cells.

The results of the two algorithms are shown in Fig. 2.2. In summary, the forcing cells are inside $\mathcal{S}$ in the first algorithm, and outside $\mathcal{S}$ in the second algorithm.

At this stage, the different types of cells are identified. To distinguish these cells, a label σ is used in the solver, and the values in different cells are shown in Eqn. 2.8.

$$\sigma = \begin{cases} 1, & \text{Fluid cells} \\ 0, & \text{Forcing cells} \\ -100, & \text{Solid cells} \end{cases} \tag{2.8}$$



(a) **Algorithm 1** (b) **Algorithm 2**

Figure 2.2: The second step of the cell categorization. White: fluid cells; green: forcing cells; pink: solid cells

2.3.2 Interpolation of Velocity

After the process of the cell categorization, a forcing term is introduced into the momentum equation so that the solution satisfies the boundary conditions on the immersed surface as:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = \nabla \cdot \left[\nu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)\right] - \nabla p + \mathbf{f} \tag{2.9}$$

Here, $\mathbf{f}$ is the body force calculated at the centers of forcing and solid cells in Fig. 2.2. $\mathbf{f}$ can be calculated by re-arranging Eqn. 2.9 as:

$$\mathbf{f} = \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) - \nabla \cdot \left[\nu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)\right] + \nabla p \tag{2.10}$$

Now assume the simplest case where $\mathcal{S}$ is fixed in space, and the centers of the forcing cells are exactly on $\mathcal{S}$. Each term on the right hand side of Eqn. 2.10 is known via the boundary conditions on $\mathcal{S}$ in this case. Therefore, $\mathbf{f}$ can be directly determined from Eqn. 2.10 in this simplest case. The governing equations are subsequently solved

using the known value of $\mathbf{f}$.

However, the cell centers are not always on the surface $\mathcal{S}$ for general cases. In such cases, $\mathbf{f}$ is calculated using the interpolation of $\mathbf{u}$ to take into account the boundary conditions on $\mathcal{S}$ as:

$$\mathbf{f} = \frac{\partial \mathbf{u}^*}{\partial t} + \nabla \cdot (\mathbf{u}^* \mathbf{u}^*) - \nabla \cdot \left[\nu \left(\nabla \mathbf{u}^* + \nabla (\mathbf{u}^*)^T \right) \right] + \nabla p \tag{2.11}$$

where,

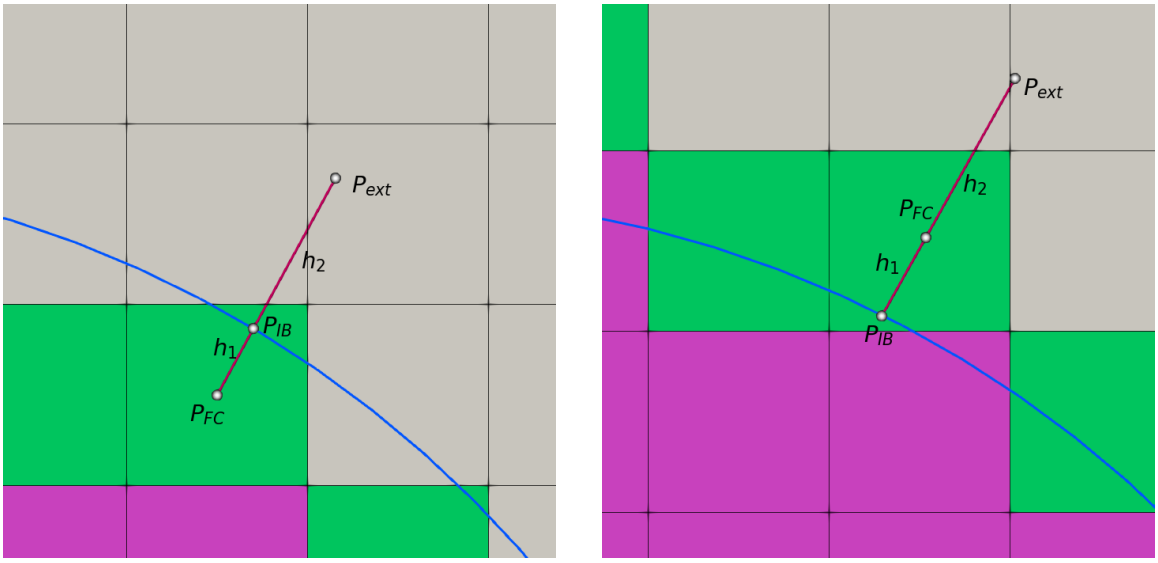
$$\mathbf{u}^* = \begin{cases} \mathcal{L}(\mathbf{u}), & \text{Forcing cells} \\ \mathbf{u}_{\text{body}}, & \text{Solid cells} \end{cases} \tag{2.12}$$

in which, the operator $\mathcal{L}(*)$ denotes a chosen interpolation scheme. In the forcing cells, $\mathbf{u}^*$ is interpolated from the fluid cells such that a no-slip velocity boundary condition is imposed exactly on $\mathcal{S}$; in the solid domain, $\mathbf{u}^*$ is simply set to be the velocity of the immersed body.

The operator $\mathcal{L}(*)$ depends on whether the forcing cells are inside or outside the immersed surface $\mathcal{S}$. Interpolation of the velocity by Eqn. 2.12 is composed of four steps:

- The closest point on $\mathcal{S}$ to the center of a forcing cell P_{IB} is identified via CGAL.
- An extended point P_{ext} inside the fluid domain is determined based on the selected algorithm.
- Velocity at the extended point $\mathbf{u}_{\text{ext}}$ is interpolated from nearby fluid cells.
- Velocity at the cell centers of the forcing cells is interpolated using $\mathbf{u}_{\text{ext}}$ and the velocity boundary condition on the IB.

Fig. 2.3 illustrates how P_{ext} is determined for both algorithms. h_1 is the distance between P_{IB} and P_{FC}, and h_2 is the distance between P_{IB} and P_{ext}.



(a) **Algorithm 1** (b) **Algorithm 2**

Figure 2.3: Identification of P_{ext} for the two algorithms

Subsequently, $\mathbf{u}^*$ is calculated as:

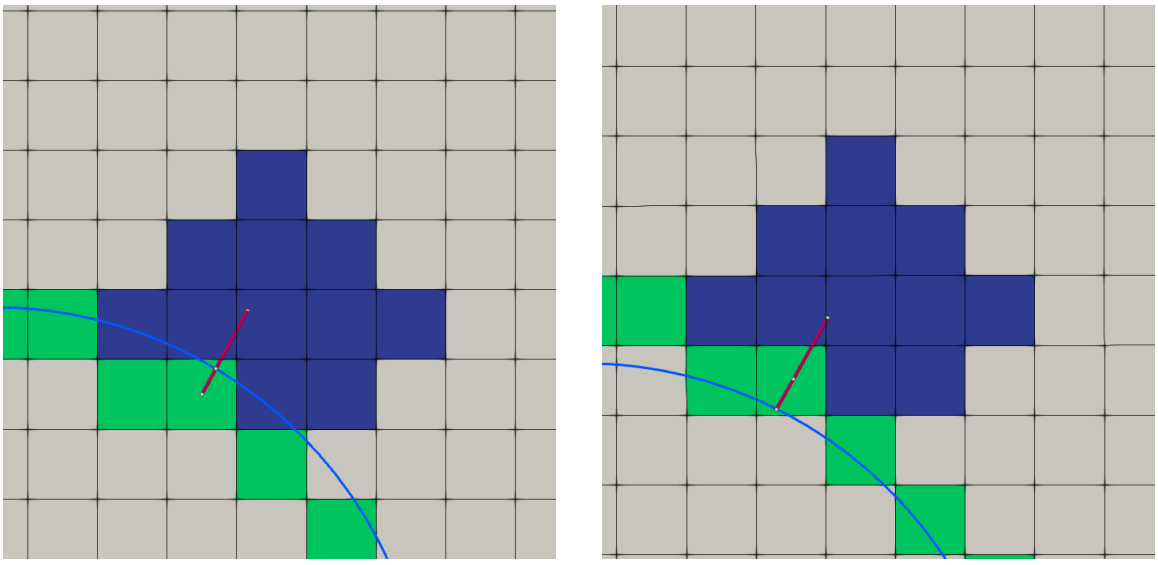
$$\mathbf{u}^* = \begin{cases} \frac{h_1+h_2}{h_2}\mathbf{u}_{\text{IB}} - \frac{h_1}{h_2}\mathbf{u}_{\text{ext}}, & \textbf{Algorithm 1} \\ \frac{h_2-h_1}{h_2}\mathbf{u}_{\text{IB}} + \frac{h_1}{h_2}\mathbf{u}_{\text{ext}}, & \textbf{Algorithm 2} \end{cases} \tag{2.13}$$

in which, $\mathbf{u}_{\text{IB}}$ is the exact velocity on the immersed surface $\mathcal{S}$.

At this stage, $\mathbf{u}_{\text{ext}}$ is unknown. Two steps are carried out to calculate $\mathbf{u}_{\text{ext}}$. First, the fluid cells which are used as stencils to interpolate $\mathbf{u}_{\text{ext}}$ need to be identified. Denote the set of stencil cells as $\mathcal{H}$. The procedure is composed of two steps, which are:

- Identify the cell that contains P_{ext}, and add it to $\mathcal{H}$. It should be noted that the distance h_2 is chosen such that the selected cell is a fluid cell.
- Check the cells that share faces with cells in set $\mathcal{H}$, and add each of these to $\mathcal{H}$ if it is a fluid cell. This step is repeated twice in total. Therefore, two layers of cells are searched, so that the interpolation stencil has enough cells and the interpolation is well-posed.

Fig. 2.4 illustrates the stencil for both algorithms.



(a) **Algorithm 1** (b) **Algorithm 2**

Figure 2.4: Stencil for the interpolation of $\mathbf{u}_{\text{ext}}$ with the two algorithms

It should be also noted there are circumstances which may affect the accuracy and the stability of present IBM. For example, when two IB surfaces are close to each other, or when an IB surface is close to the boundary of the computational domain, there may not be a suitable set of cells for the interpolation stencil of $\mathbf{u}_{\text{ext}}$. The stencil could have too few cells, or the arrangement of the cells can cause the interpolation to be ill-posed. Mesh refinement is one remedy in such cases, since to resolve the flow features in such regions, sufficient mesh refinement is indeed as a requirement for the flow solver anyway. With mesh refinement, it is more likely to find enough cells to interpolate $\mathbf{u}_{\text{ext}}$ with more cells clustered locally. However, there still remain cases in which too few cells can be found for the interpolation stencil regardless of mesh refinement (for example, when an IB surface intersects with the boundary of the computational domain). In this case, the interpolation is simplified by directly approximating $\mathbf{u}^*$ to be the rigid body velocity to stabilize the simulation.

The selection of the interpolation stencil is time consuming. For the cases when the immersed surface $\mathcal{S}$ is fixed, the stencils do not change during the simulations,

and the search is done once at the beginning of the simulation. The cost to generate the stencil is small compared to that of the entire simulation. A problem arises when $\mathcal{S}$ is moving. Since the set of forcing cells changes in time, the search needs to be performed at every time step, and the cost is a concern in this scenario.

To avoid unnecessary operations in the search process, a pre-processing step is executed to take advantage of the fact that the information of cell connections does not change, regardless of the motion of $\mathcal{S}$. If the cell connections are stored at the beginning of the simulation, the stencil selection can be done efficiently. The data structure of the sparse matrix of PETSc, which is an open-source suite of data structures and routines for parallel solution of scientific applications, is adopted. A sparse matrix is initialized at the beginning of each simulation to store the information of cell connections. Each cell corresponds to one row of the matrix. For each row, the indices of the non-zero columns correspond to the candidates in the stencil set $\mathcal{H}$. With this matrix generated once at the beginning of a simulation, the procedure of finding the stencils for interpolation is simplified to:

- Identify the cell that contains P_{ext}, and add it to $\mathcal{H}$. This step is exactly the same as before.

- Look up the row corresponding to the cell found in the previous step. Check through the indices of the non-zero columns. If $\sigma = 1$, the index of the non-zero column is added to $\mathcal{H}$.

Once the cell set $\mathcal{H}$ is determined, the velocity at the extended point $\mathbf{u}_{\text{ext}}$ is calculated by:

$$\mathbf{u}_{\text{ext}} = \sum w_i \mathbf{u}_i \tag{2.14}$$

in which, $\mathbf{u}_i$ is the velocity at the centers of the stencil cells, and w_i is the corresponding interpolation weight. In the current work, the second-order Laplacian weight method (*Frink* (1994)) is adopted for the interpolation. Compared with the

commonly used linear and quadratic interpolations, the Laplacian weight method is suitable on both structured and unstructured meshes. As for linear and quadratic interpolations, the number of cells in the stencil needs to be the same as the number of coefficients of the selected interpolant. Otherwise, a least-squares problem should be solved which involves inverse of a matrix. In comparison, the computational cost of the Laplacian weight method is less. In the Laplacian weight method, the interpolation coefficients w_i in Eqn. 2.14 are derived based on the following Laplacians:

$$\begin{aligned} L(x_{\text{ext}}) &= \sum w_i(x_i - x_{\text{ext}}) = 0 \\ L(y_{\text{ext}}) &= \sum w_i(y_i - z_{\text{ext}}) = 0 \\ L(z_{\text{ext}}) &= \sum w_i(z_i - z_{\text{ext}}) = 0 \end{aligned} \tag{2.15}$$

in which, $(x_{\text{ext}}, y_{\text{ext}}, z_{\text{ext}})$ is the Cartesian coordinate of the point P_{ext}; (x_i, y_i, z_i) is the coordinate of each cell center in the interpolation stencil.

The weights w_i are defined as

$$w_i = 1 + \Delta w_i \tag{2.16}$$

Subsequently, a cost function can be defined as:

$$C = \sum (\Delta w_i)^2 \tag{2.17}$$

where, Δw_i is calculated by minimizing the cost function via the method of Lagrange multipliers. Hence, Δw_i can be written as:

$$\Delta w_i = \lambda_x(x_i - x_{\text{ext}}) + \lambda_y(y_i - y_{\text{ext}}) + \lambda_z(z_i - z_{\text{ext}}) \tag{2.18}$$

in which, the Lagrange multipliers λ are calculated by:

$$\begin{aligned}
\lambda_x &= \left[-R_x(I_{yy}I_{zz} - I_{yz}^2) + R_y(I_{xy}I_{zz} - I_{xz}I_{yz}) - R_z(I_{xy}I_{yz} - I_{yy}I_{xz})\right]/D \\
\lambda_y &= \left[R_x(I_{xy}I_{zz} - I_{xz}I_{yz}) - R_y(I_{xx}I_{zz} - I_{xz}^2) + R_z(I_{xx}I_{yz} - I_{xy}I_{xz})\right]/D \\
\lambda_z &= \left[-R_x(I_{xy}I_{yz} - I_{yy}I_{xz}) + R_y(I_{xx}I_{yz} - I_{xy}I_{xz}) - R_z(I_{xx}I_{yy} - I_{xy}^2)\right]/D
\end{aligned} \tag{2.19}$$

where, D is the determinant of the symmetric tensor I.

$$D = \det(I) = \begin{vmatrix} \sum(x_i - x_{\text{ext}})^2 & \sum(x_i - x_{\text{ext}})(y_i - y_{\text{ext}}) & \sum(x_i - x_{\text{ext}})(z_i - z_{\text{ext}}) \\ \sum(x_i - x_{\text{ext}})(y_i - y_{\text{ext}}) & \sum(y_i - y_{\text{ext}})^2 & \sum(y_i - y_{\text{ext}})(z_i - z_{\text{ext}}) \\ \sum(x_i - x_{\text{ext}})(z_i - z_{\text{ext}}) & \sum(y_i - y_{\text{ext}})(z_i - z_{\text{ext}}) & \sum(z_i - z_{\text{ext}})^2 \end{vmatrix} \tag{2.20}$$

$$R_x = \sum(x_i - x_{\text{ext}}) \qquad R_y = \sum(y_i - y_{\text{ext}}) \qquad R_z = \sum(z_i - z_{\text{ext}}) \tag{2.21}$$

After substituting Eqn. 2.18 to 2.21 into Eqn. 2.16, w_i is finally calculated by non-dimensionalization as:

$$w_i = \frac{w_i}{\sum w_i} \tag{2.22}$$

At this point, velocity in the forcing cells is be interpolated using the boundary conditions on $\mathcal{S}$ and the velocity in the fluid cells, which are calculated by solving the governing equations.

2.3.3 Solution of the Modified Governing Equations

The solution procedure introduced in Section. 2.2 is modified by taking into account the forcing term $\mathbf{f}$ as summarized below:

- At the beginning of each time step t, the velocity $\mathbf{u}_1$ and pressure p_1 are known from the solution of the previous time step t^0 or from the previous PISO iteration. They satisfy all boundary conditions, including those on $\mathcal{S}$.

  Given the velocity boundary condition of $\mathcal{S}$, the velocity in the forcing cells and

solid cells can be calculated by Eqn. 2.12. The forcing term $\mathbf{f}$ is subsequently calculated by Eqn. 2.11.

- The modified momentum equation Eqn. 2.9 is solved and the predicted velocity field $\mathbf{u}_2$ is calculated.

- The modified pressure Poisson equation Eqn. 2.23 is solved.

$$\nabla \cdot \left(\frac{1}{a_p} \nabla p_{\text{new}} \right) = \nabla \cdot \left(\frac{1}{a_p} \mathbf{H}(\mathbf{u}_2) + \mathbf{f} \right) \tag{2.23}$$

- The velocity and face flux are updated using Eqn. 2.24 and Eqn. 2.25, respectively.

$$\mathbf{u}_p = \frac{1}{a_p} \left(\mathbf{H}(\mathbf{u}_2) - \nabla p_{\text{new}} + \mathbf{f} \right) \tag{2.24}$$

$$\phi = \mathbf{S} \cdot \left(\frac{1}{a_p} \mathbf{H}(\mathbf{u}_2) \right)_f - \mathbf{S} \cdot \left(\frac{1}{a_p} \nabla p_{\text{new}} \right)_f + \mathbf{S} \cdot \mathbf{f}_f \tag{2.25}$$

  in which, $\mathbf{f}_f$ denotes a central difference of $\mathbf{f}$ from cell centers to face centers.

- Return to Step 1 if velocity and pressure do not satisfy criteria of convergence. Otherwise, proceed to the next time step.

2.4 Calculation of the Force on the Immersed Surface

In many applications, the force exerted on the immersed boundaries is of practical interest. Since there are no explicit boundary patches used by the IBM, the traditional way of integrating the wall shear stress and pressure cannot be applied to the IBM in a straightforward manner. Therefore, it is important to discuss how the force is calculated in the current work following the idea of *Lee et al.* (2011).

Fig. 2.5 illustrates the domain for the calculation of the force on the immersed surface, where CV_1 and CV_2 denote the control volumes for the immersed surface and fluid, respectively. The control volume CV_1 is composed of all forcing and solid

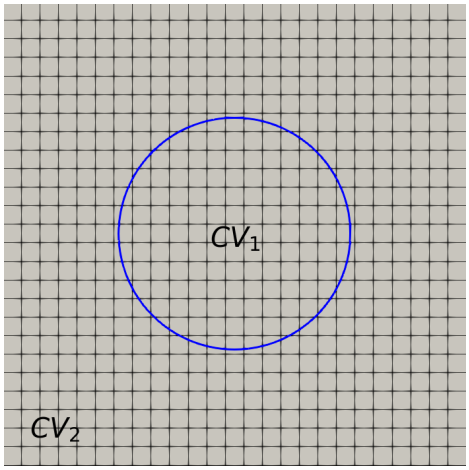


Figure 2.5: Diagram for the calculation of force on an immersed surface

cells and CV_2 is composed of all fluid cells. CS_1 is the control surface of CV_1.

The force $\mathbf{F}$ on CS_1 can be expressed as:

$$\mathbf{F} = -\int_{\mathrm{CS}_1} \left[p\mathbf{n} - \rho\nu\mathbf{n} \cdot (\nabla\mathbf{u} + \nabla\mathbf{u}^T) \right] dS \tag{2.26}$$

in which $\mathbf{n}$ is the surface normal direction of CS_1 pointing into the fluid domain.

Using the Gauss theorem, Eqn. 2.26 can be expressed as volume integration over CV_1 as:

$$\mathbf{F} = -\int_{\mathrm{CV}_1} \left[\nabla p - \nabla \cdot (\rho\nu \cdot (\nabla\mathbf{u} + \nabla\mathbf{u}^T)) \right] dV \tag{2.27}$$

Substitution into the momentum equation Eqn. 2.9 yields:

$$\mathbf{F} = \int_{\mathrm{CV}_1} \left[\frac{\partial \rho\mathbf{u}}{\partial t} + \nabla \cdot (\rho\mathbf{u}\mathbf{u}) - \rho\mathbf{f} \right] dV \tag{2.28}$$

With Eqn. 2.28, the calculation of the force $\mathbf{F}$ is converted into the volume integration of the forcing and solid domains near and inside the immersed surface $\mathcal{S}$. All terms in Eqn. 2.28 are known at the end of every time step, so the calculation is straightforward.

2.5 Body Motion

The proposed solver can handle both prescribed and predicted motions. For prescribed motion, the velocity and position of $\mathcal{S}$ at time t is calculated with the given function of motion. The velocity $\mathbf{u}_{\text{body}}$ in Eqn. 2.12 and $\mathbf{u}_{\text{IB}}$ in Eqn. 2.13 is calculated by:

$$\mathbf{u}_{\text{body}} = g(\mathbf{x}(t)) \qquad \mathbf{u}_{\text{IB}} = g(P_{\text{IB}}(t)) \tag{2.29}$$

where, $\mathbf{x}(t)$ is the coordinates of the cell center of a solid cell at time t; $g(*)$ denotes the motion function.

For predicted motion, the 6 Degrees-of-Motions (DoF) library (*Piro* (2013)) developed by the Computational Ship Hydrodynamics Laboratory (CSHL *Laboratory* (2020)) is used. After the force on the object is calculated at the end of each time step or PISO iteration, the acceleration of the body is calculated by solving the equations of motion. By integrating the acceleration twice in time, the velocity and position of the body are obtained. Subsequently the velocity boundary condition $\mathbf{u}_{\text{IB}}$ and velocity in the solid cells $\mathbf{u}_{\text{body}}$ are calculated in the same way as for the prescribed motion.

With body motion involved, the solution procedure described in the previous section needs to be modified. At the beginning of each time step, the position of the immersed object is updated based on the calculated motion at the end of the previous time step. The fluid, forcing and solid cells need to be re-categorized based on the new position of the IB. The interpolation stencils and weights are also recalculated. In addition, the velocity boundary condition is updated.

2.6 Solver Verification

In this section, the order of accuracy of the solver is examined for both fixed and moving immersed objects. It should be noted that the apparent order of accuracy

is not only affected by the proposed immersed boundary method, but also limited by the underlying numerical discretization schemes. Therefore, it is worth examining the order of accuracy of OpenFOAM's default solver in the first place to demonstrate that the order of accuracy of the immersed boundary solver is not downgraded by the implementation of the present IBM. To achieve this goal, the problems of linear convection and diffusion on both structured and unstructured meshes are carefully investigated.

2.6.1 Convection

The 2-D convection problem of a scalar T in a square domain is investigated in this section. The governing equation is written as:

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = 0 \tag{2.30}$$

in which, the scalar T is transported by the uniform velocity field $\mathbf{u} = (1/\sqrt{2}, 1/\sqrt{2}, 0)$. The value of T at any time instant can be calculated analytically and used as the exact solution for the calculation of the error.

Mesh convergence studies for both structured and triangular unstructured meshes are carried out to demonstrate the order of accuracy of the discretization of the convection term. Four successively finer meshes are used for the error analysis with the number of nodes on each side of the square domain is 20, 40, 80 and 160, respectively. The nodes are evenly distributed along each side. At beginning of the simulation, the profile of T is prescribed on the left side of the domain as:

$$T = \begin{cases} \sin^2(3\pi(y/H - 1/6)), & 1/6 < y/H < 1/2 \\ 0, & \text{otherwise} \end{cases} \tag{2.31}$$

where, H is the length of the side of the domain.

Fig. 2.6 presents an example of the simulation on an unstructured mesh with 40 nodes on each side of the domain.

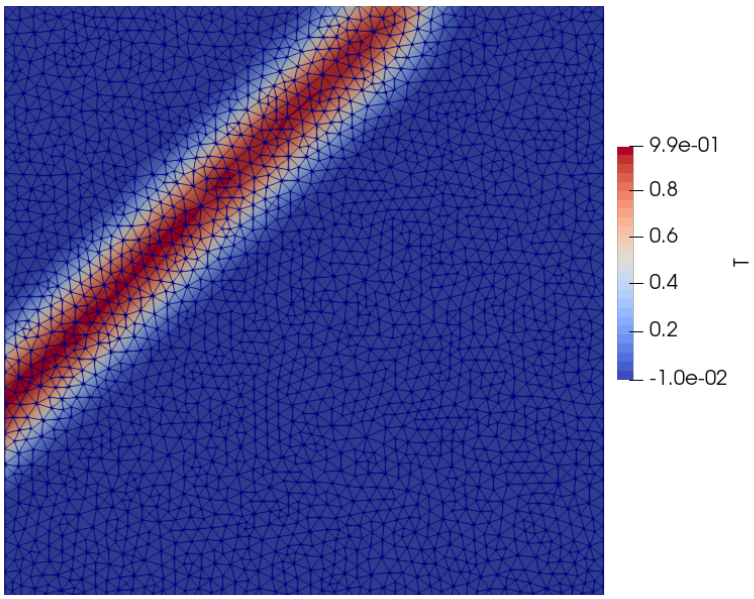


Figure 2.6: 2-D simulation of the convection of a scalar T on an unstructured mesh

The L_1, L_2 and L_∞ error norms are defined as:

$$L_1 = \frac{\sum_n V_n |T_n^g - T^e|}{\sum_n V_n} \qquad L_2 = \sqrt{\frac{\sum_n (V_n |T_n^g - T^e|)^2}{\sum_n V_n}} \qquad L_\infty = \max(|T_n^g - T^e|) \quad (2.32)$$

in which, n is the number of cell elements in a square control volume for the calculation of the error, shown in Fig. 2.7; V_n is the area of the cell element; T_n^g denotes the numerical solution of T on one mesh level and T^e is the analytical solution.

Fig. 2.8 and Fig. 2.9 show the variation of different error norms with the increaes of Δx in a log-log scale, in which Δx is the node spacing along each side of the domain. The results confirm that the discretization of the convection is second-order on both structured and unstructured meshes.

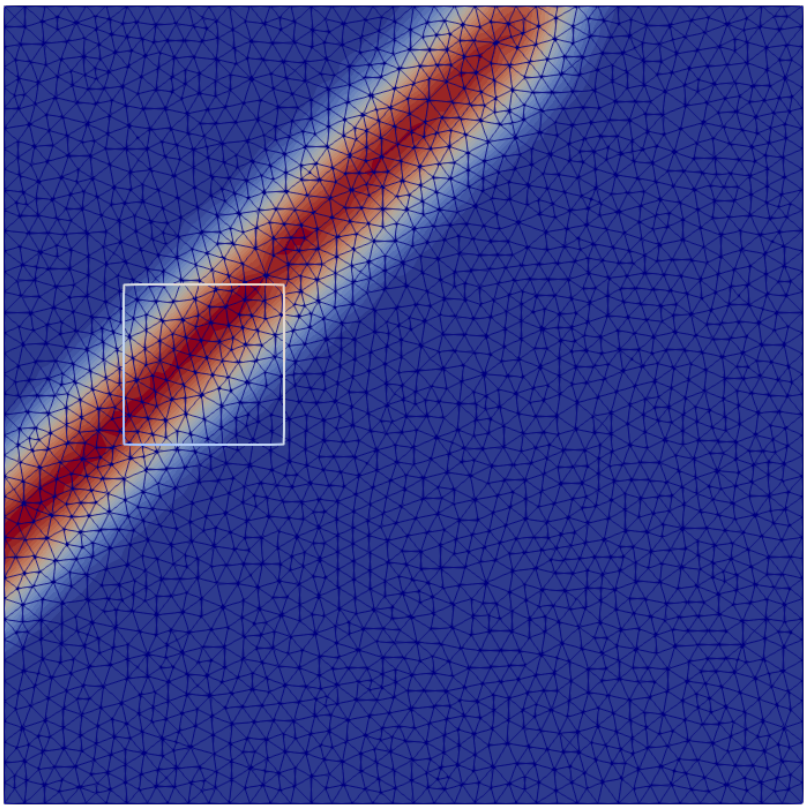


Figure 2.7: Control volume for the calculation of the error norms

2.6.2 Diffusion

The 2-D Laplacian equation is investigated in this section:

$$\nabla^2 T = 0 \tag{2.33}$$

The same square domain and meshes for studying the discretization of the convection term are used. For unstructured meshes, additional discretization error comes from the non-orthogonality and skewness of the meshes. The error resulting from mesh non-orthogonality is because the vector between the adjacent cell centers is non-parallel to normal vector of the face that the two cells share. The error resulting from mesh skewness is because the vector connecting the adjacent cell centers does not intersect with the shared face at the face center. Apparently, the unstructured

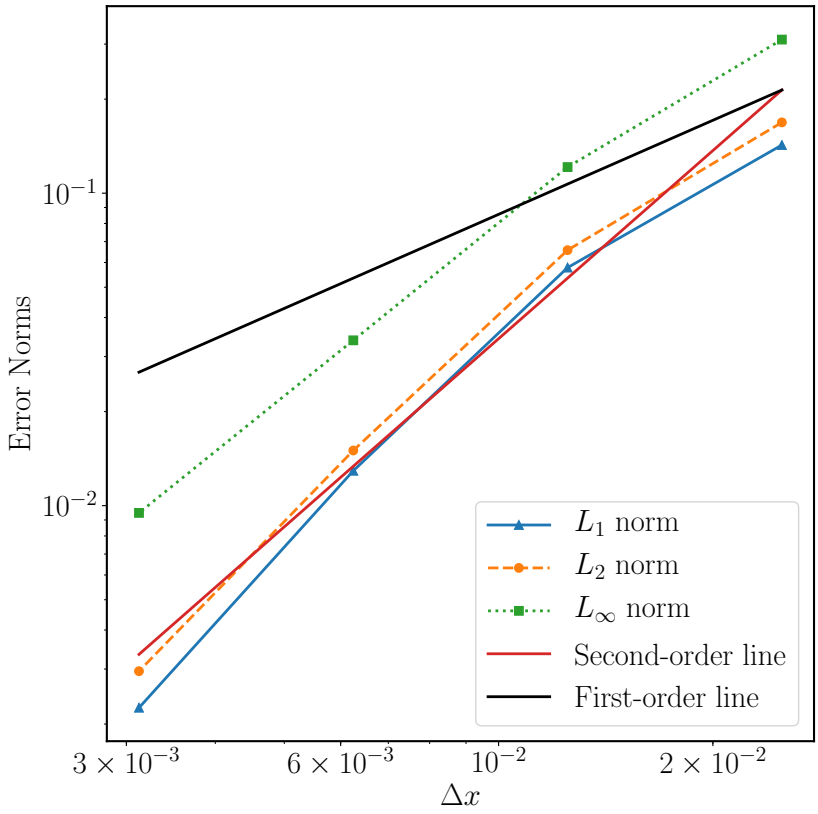


Figure 2.8: Mesh convergence study for the 2-D convection problem of a scalar on structured meshes

meshes used in the previous section include both sources of errors. However, OpenFOAM only has corrections for the mesh non-orthogonality. To fully understand the influence of OpenFOAM's correction due to mesh quality on the order of accuracy, simulation on a set of four meshes with pure non-orthogonality is also carried out. Fig. 2.10 shows an example of the mesh with only non-orthogonality but zero skewness.

A Dirichlet boundary condition is enforced for T on all boundaries as shown in Eqn. 2.34, and Fig. 2.11 shows a sample result on a structured mesh.

$$T = \begin{cases} \sin(\pi x/H), & \text{bottom} \\ 0, & \text{otherwise} \end{cases} \tag{2.34}$$

The results of the mesh convergence study are summarized in Fig. 2.12–2.14, which

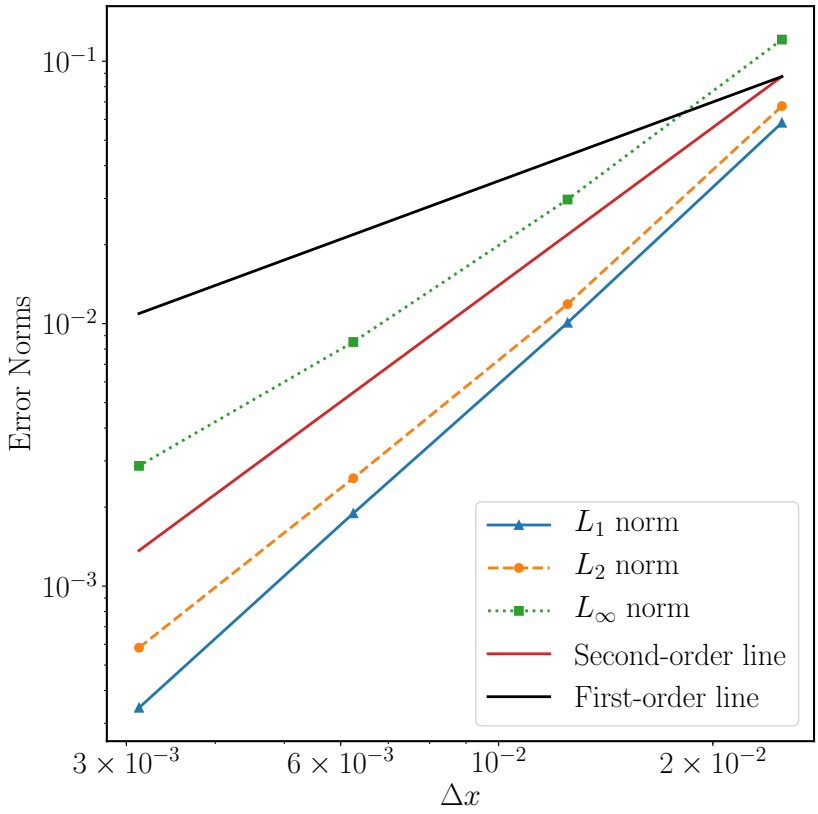


Figure 2.9: Mesh convergence study for 2-D convection problem of a scalar on unstructured meshes

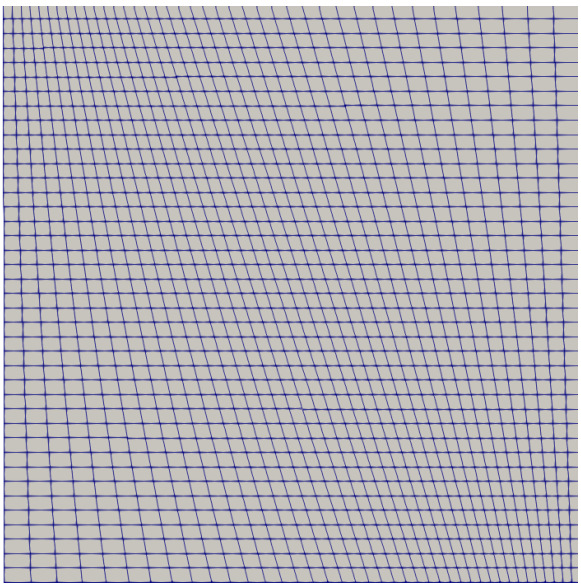


Figure 2.10: Mesh with pure non-orthogonality used for the 2-D Laplacian problem

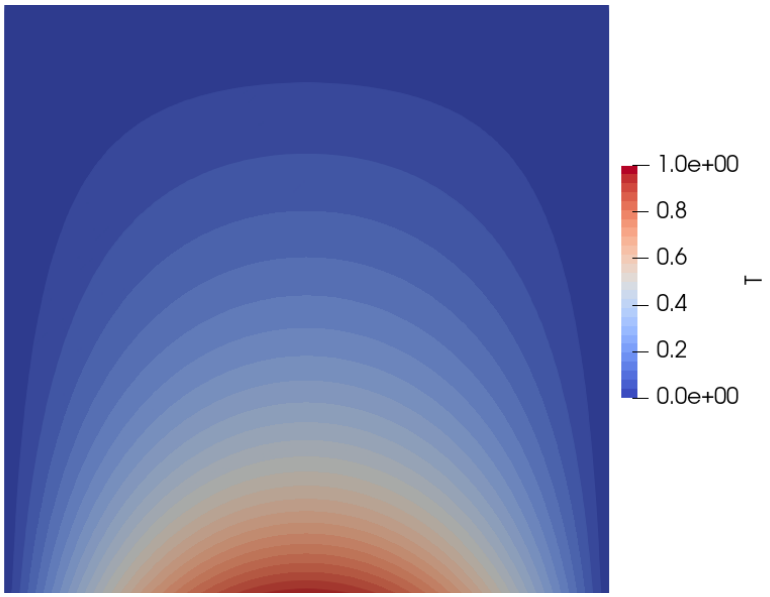


Figure 2.11: Simulation of a 2-D Laplacian problem on the structured mesh with 40×40 cells

show the variation of different norms with respect to the increase of the cell spacing. It demonstrates that on all three types of meshes, different norms have the same trend, indicating the discretization error for the diffusion term behaves the same locally and globally. It is evident from Fig. 2.12 that the discretization of the diffusion term is second-order accurate both globally and locally on the orthogonal structured meshes. In comparison, Figs. 2.13 and Fig. 2.14 show that the discretization is only first-order accurate even when the correction for mesh non-orthogonality is applied. In general cases when a solid wall has curvature, mesh non-orthogonality and skewness are almost inevitable. The results above demonstrate that OpenFOAM is expected to be first-order accurate for discretizing the Laplacian term in such cases.

2.6.3 Manufactured Solution of 2-D Steady Heat Conduction

As discussed in the previous section, for general flows over a curved surface, such as flows around a ship or a car, mesh non-orthogonality and skewness are almost inevitable even when structured meshes are used. When convection and diffusion are

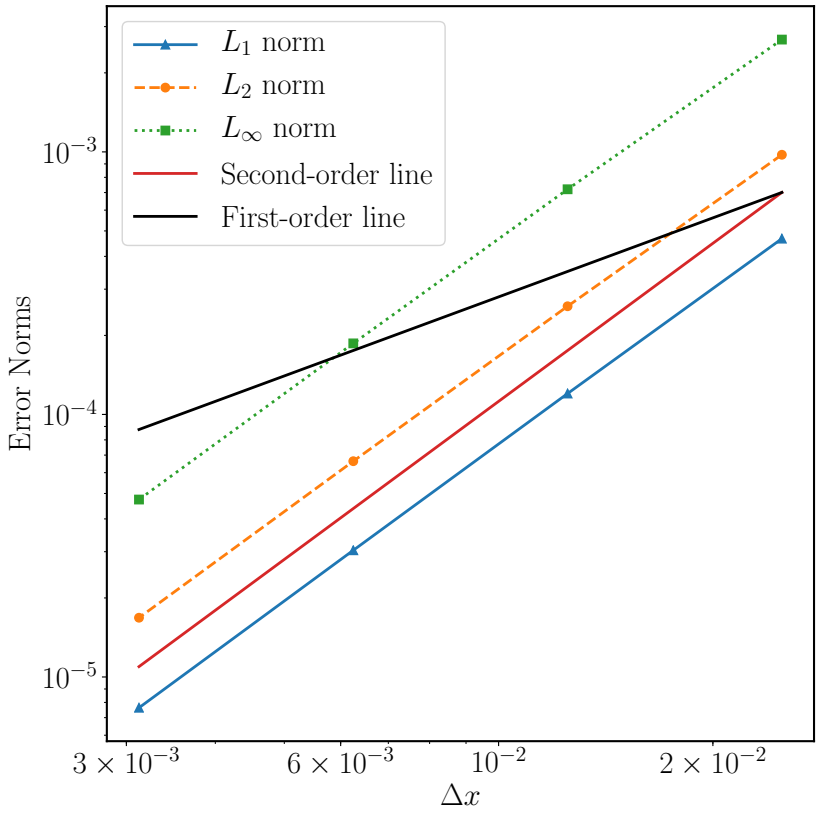


Figure 2.12: Mesh convergence study for the 2-D Laplacian problem on orthogonal structured meshes

both considered in the governing equations, the overall order of accuracy is very likely to be limited to first order by the discretization of the diffusion term.

By contrast, the current IBM can solve the governing equations for curved geometries on uniform background meshes. The quality of the mesh is maintained regardless of the shape of the immersed object even when body motion is considered. Since the discretization of the diffusion term is more sensitive to the mesh quality in OpenFOAM, a more rigorous verification test is performed using the method of manufactured solutions (MMS).

The basic idea of the MMS is to manually construct a source term for the governing equations such that a predefined exact solution can satisfy the governing equations and all boundary conditions. In this section, the manufactured solution for a 2-D

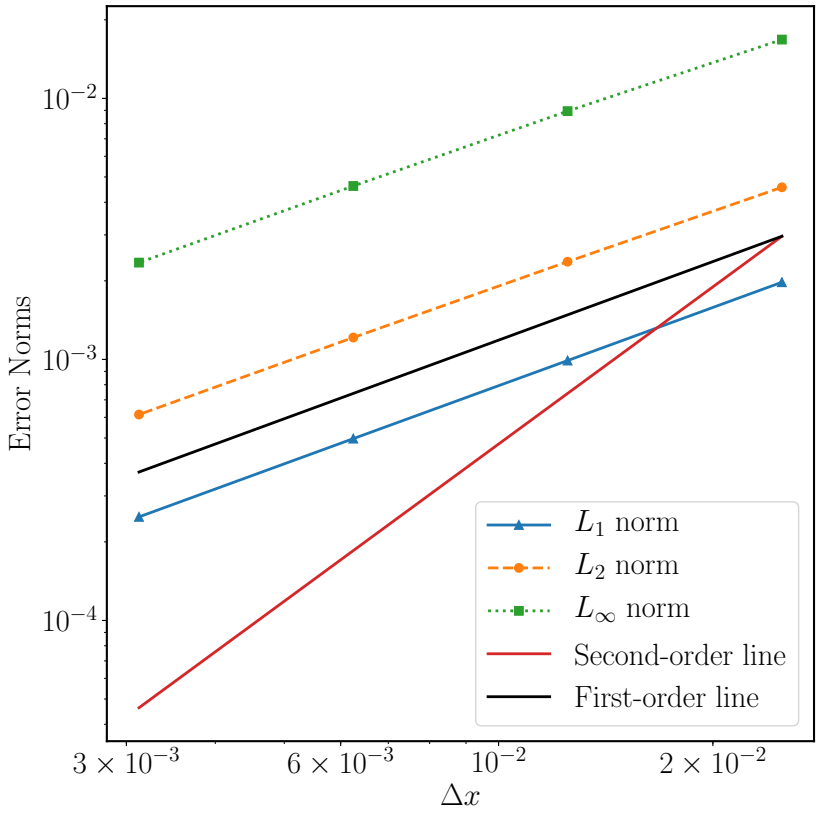


Figure 2.13: Mesh convergence study for the 2-D Laplacian problem on structured meshes with only non-orthogonality

steady-state solution for a scalar T is considered, which is governed by:

$$\nabla^2 T = \text{source} \tag{2.35}$$

in which, the source term on the right hand side is introduced by the MMS to guarantee the predefined solution satisfies this governing equation. In addition, the method of *Bond et al.* (2007) is adopted to construct the manufactured solution to ensure that boundary conditions are satisfied on all boundaries.

Fig. 2.15 shows the computational domain and the exact solution of T of the MMS problem. A 2-D unit square domain with a curved bottom is considered, and

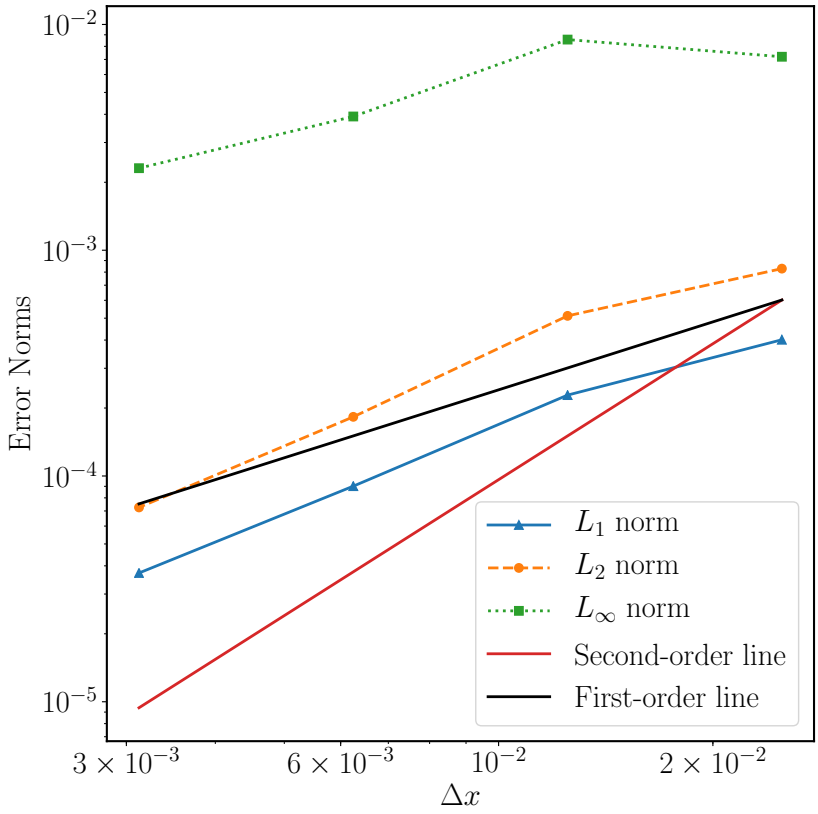


Figure 2.14: Mesh convergence study for the 2-D Laplacian problem on unstructured meshes

the curved bottom is described by the function $F(x, y)$ as:

$$F(x, y) = \frac{1}{2}\cos\left(\frac{\pi x}{2}\right) - y = 0 \tag{2.36}$$

The manufactured solution of T is constructed as:

$$T(x, y) = 300 + \left[25\cos\left(\frac{7\pi x}{4}\right) + 40\sin\left(\frac{4\pi y}{3}\right)\right]\left[-\frac{1}{2}\cos\left(\frac{\pi x}{2}\right) + y\right] \tag{2.37}$$

According to Eqn. 2.37, the curved boundary will satisfy the Dirichlet condition of $T = 300$. The Dirichlet boundary conditions are also imposed on other boundaries based on the solution of Eqn. 2.37. The source term is derived by substitution of Eqn. 2.37 into Eqn. 2.35 using the MATLAB symbolic math toolbox. The analytical source term in Eqn. 2.35 is given in Appendix. A.

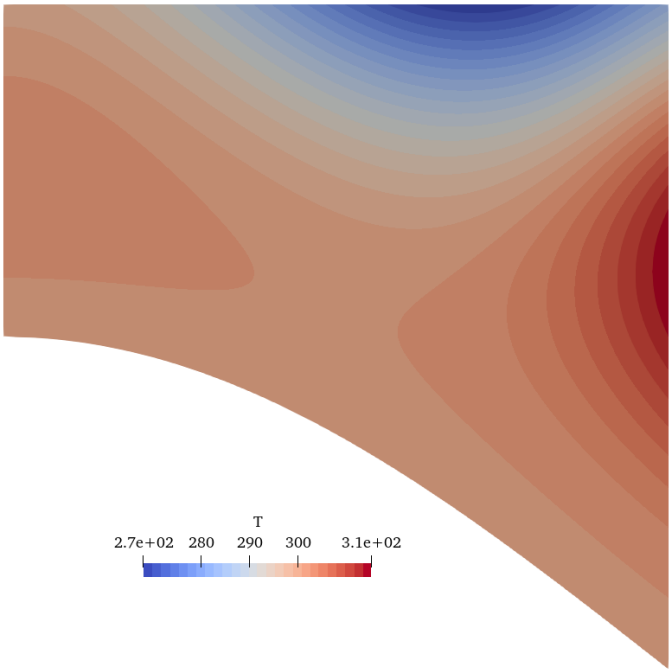


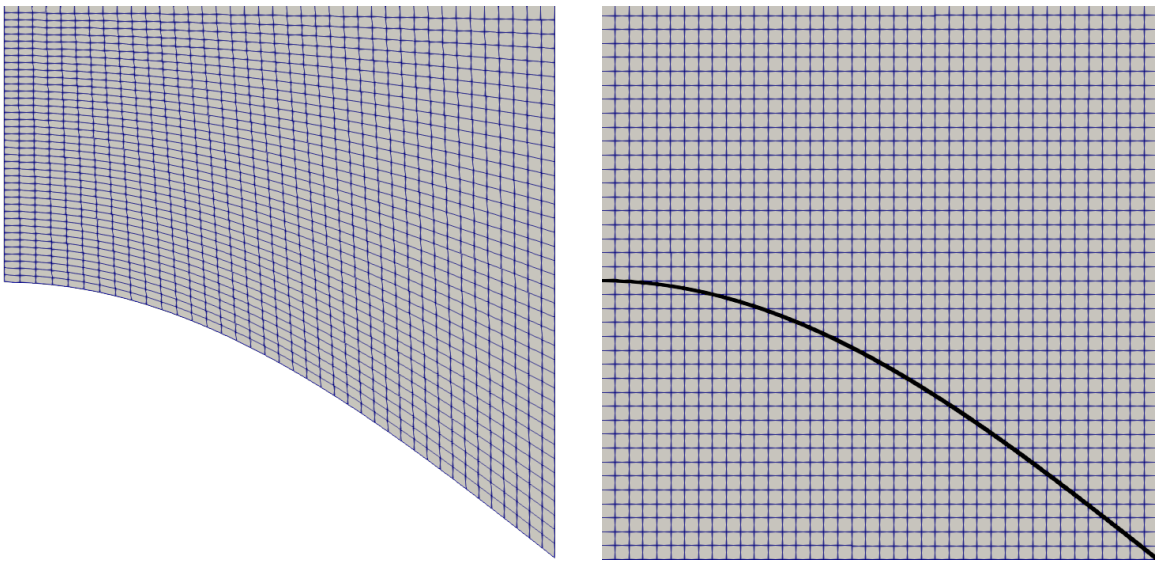
Figure 2.15: Computational domain and the exact solution of T of the MMS problem

Simulations are carried out using both body-fitted meshes and immersed boundary meshes, with five successively finer meshes. The number of nodes on each side of the domain is 20, 40, 80, 160 and 320. Fig. 2.16 shows examples of both types of meshes. Although a structured mesh is used, it has non-orthogonality and skewness. Hence, the result can also characterize the accuracy on unstructured meshes.

The order of accuracy for the discretization on the body-fitted mesh, and for IB interpolation with the forcing cells inside and outside the immersed surface are plotted in Fig. 2.17.

Fig. 2.17(a) confirms again that OpenFOAM is only first-order accurate when discretizing the Laplacian term on meshes with non-orthogonality and skewness.

Fig. 2.17(b) and (c) show that the IB interpolation is second-order accurate globally and locally. In addition, the results demonstrate that placing forcing cells inside or outside the immersed surface does not have influence on the order of accuracy either globally or locally.



(a) Body-fitted Mesh (b) Immersed boundary Mesh

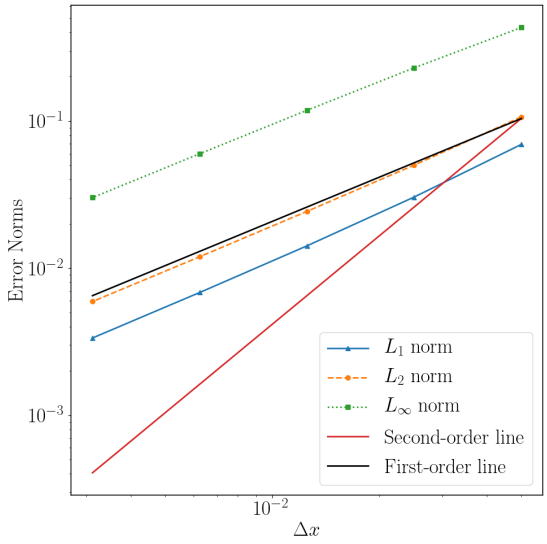
Figure 2.16: Examples of meshes with 40×40 cells for the MMS problem

In summary, for general cases, since the IBM solves the governing equations on the background mesh with less non-orthogonality and skewness, the numerical tests prove that the proposed IBM can achieve second-order accuracy. Especially for the mesh cells near a surface with large curvature, the accuracy of discretization is highly affected by the mesh quality. In such cases, the proposed IBM can be more accurate than the traditional body-fitted meshes by maintaining the mesh quality in that region.

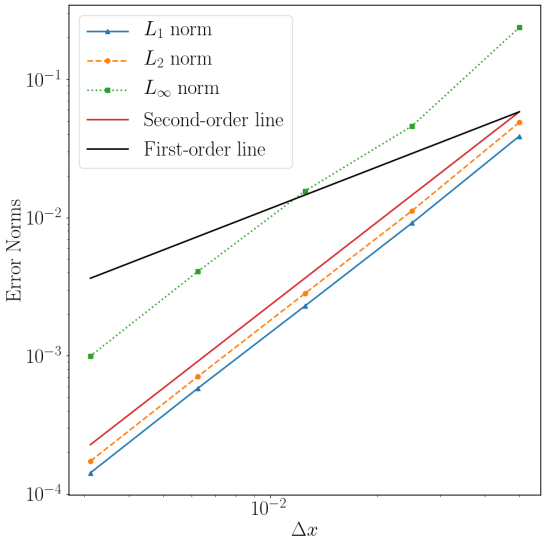
2.6.4 Flow Around a Cylinder Inside a Cavity

In the previous sections, the order of accuracy of the discretization of convection and diffusion terms is presented on different types of meshes. The proposed IBM can be second-order accurate in cases when the mesh non-orthogonality and skewness are unavoidable for body-fitted meshes. In comparison, OpenFOAM is only first-order accurate for the discretization of the diffusion term if the correction for the mesh non-orthogonality is included.

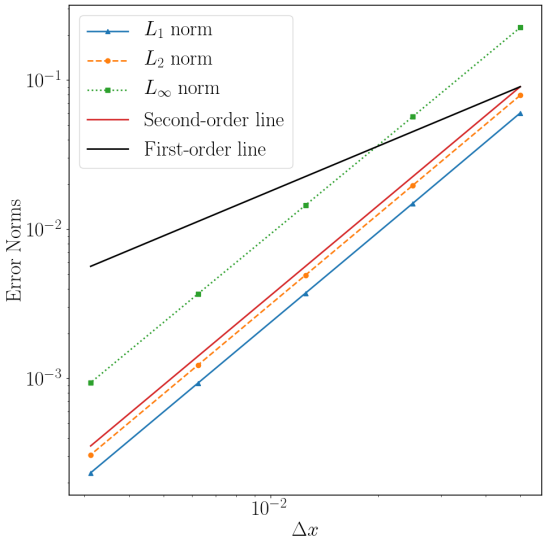
In previous sections, the order of accuracy of the discretization of separate terms



(a) Body-fitted mesh



(b) Forcing cells inside the IB



(c) Forcing cells outside the IB

Figure 2.17: Order of accuracy for the MMS problem using different types of meshes

is studied. In this section, the order of accuracy of the present IBM by solving the N-S equations for the 2-D flow around a fixed cylinder inside a lid-driven cavity is further investigated.

The computational domain and its size is depicted in Fig. 2.18. The outer square (illustrated as solid lines) is the boundary of the domain and is represented using traditional body-fitted meshes and corresponding boundary conditions. The inner

dashed circle is represented using the present IBM. A no-slip boundary condition is used for velocity. To eliminate the influence from the outer body-fitted boundaries, a square region is defined in which the norms of error are calculated. The region is represented as the dashed rectangle in Fig. 2.18.

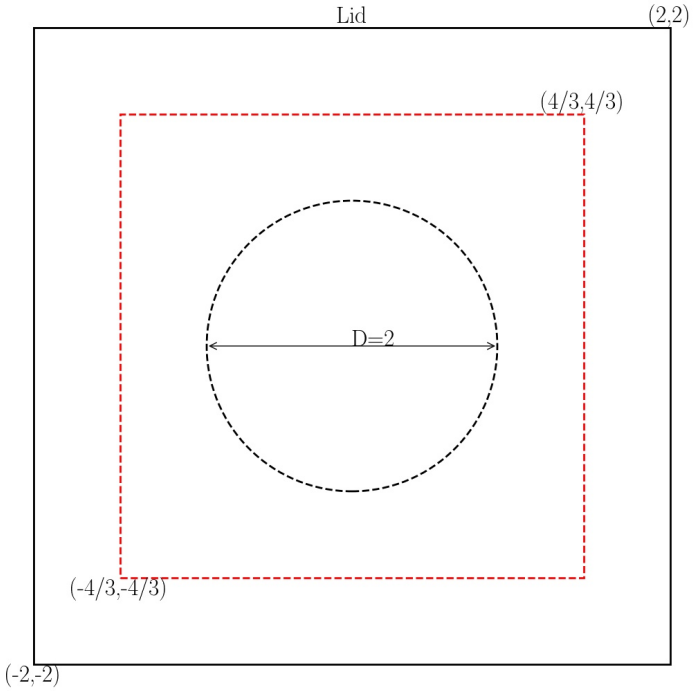


Figure 2.18: Computational domain of flow around a cylinder inside a cavity

Velocity is zero on all boundaries except on the top lid, where the vertical component of velocity $v(x)$ is zero and the horizontal component varies in the x direction as:

$$u(x) = \begin{cases} 1, & -1 \leq x \leq 1 \\ \frac{1}{2}(1 - cos(\pi(x+2))), & \text{otherwise} \end{cases} \tag{2.38}$$

The velocity decreases to zero at the two top corners to avoid singularity in the solution.

The Reynolds number based on the dimension of the side of the square cavity and the maximum velocity along the lid is 20. A set of five meshes is used for calculating the order of accuracy. The number of nodes on each side of the cavity are 20, 40, 80,

160 and 320, respectively. Due to the lack of analytical solution for this problem, the result by using finest mesh is used as the "true" answer.

The velocity field obtained on the finest mesh is plotted in Fig. 2.19.

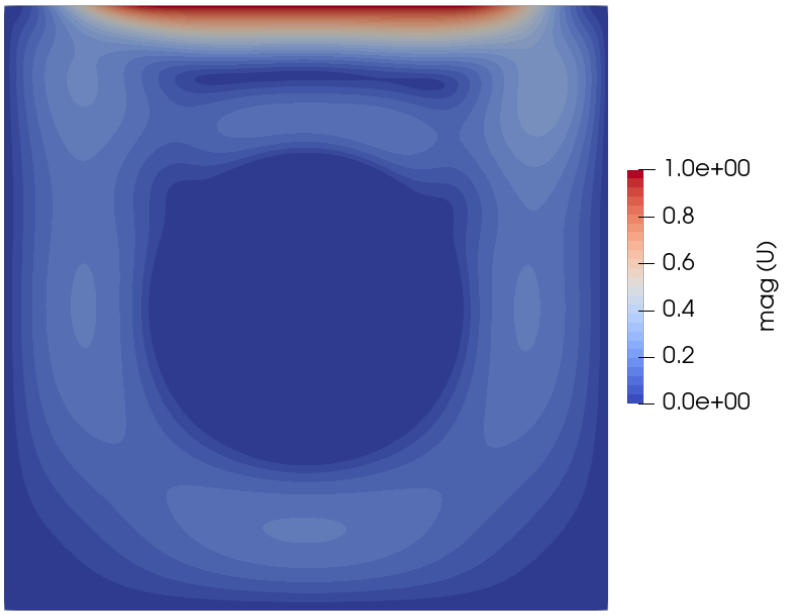


Figure 2.19: Velocity field around a cylinder inside a cavity by using a 320×320 mesh

Fig. 2.20 and Fig. 2.21 show the error norms of the horizontal and vertical components of velocity by using different layouts of forcing cells. For both layouts, the L_1 and L_2 norms of both velocity components exhibit second-order accuracy while the order of accuracy of the L_∞ norm is between first and second. The results demonstrate that for solving the N-S equations the proposed IBM is overall second-order accurate whereas first-order accurate locally. The difference of the layouts of the forcing cells has no apparent influence on the order of accuracy.

2.6.5 Oscillating Circular Cylinder in a Cavity

When body motions are considered, the roles of the cells can change. Specifically, fluid cells and solid cells can change to forcing cells, and vice versa, as shown in Fig.

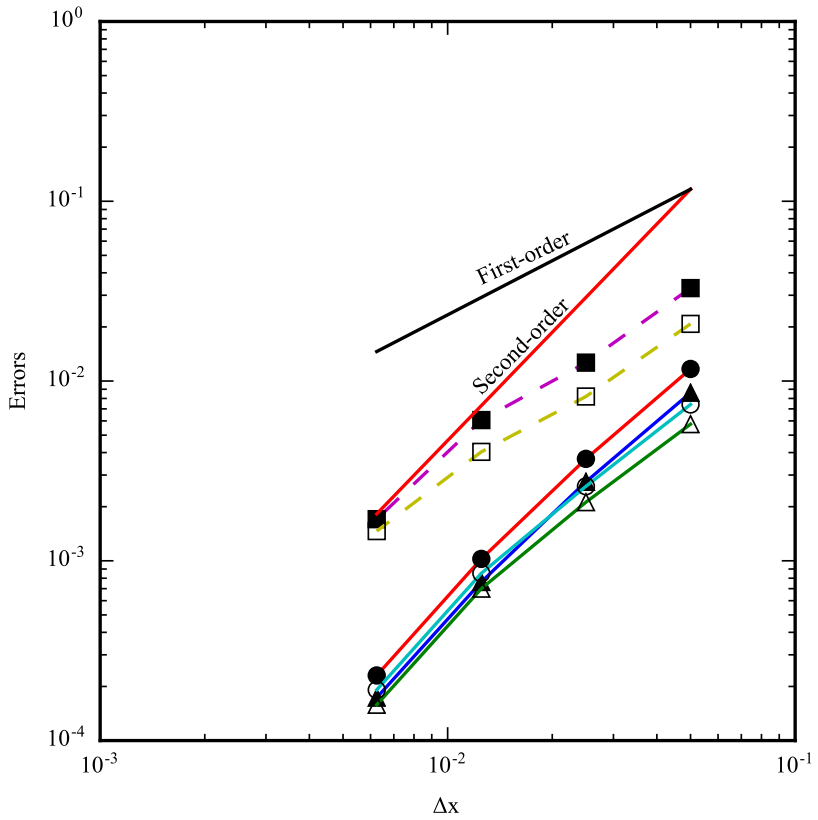


Figure 2.20: Error norms of a fixed circular cylinder in the cavity when the forcing cells are inside the cylinder. (△) L_1 norm; (◯) L_2 norm; (□) L_∞ norm; filled symbols are for u and open for v

2.22. Thus, it is important to examine the performance of the solver for solving the flow around moving geometries.

In this section, the case of a 2-D circular cylinder oscillating inside a cavity is considered.

In this case, the diameter of the cylinder is D and the dimension of the side of the cavity is $2D$. At the beginning, the cylinder rests at the center of the cavity. The cavity is stationary and the cylinder has horizontal harmonic motion described as:

$$x(t) = A\sin(\omega t) \tag{2.39}$$

in which the amplitude of the motion A is $0.1D$, $\omega = \pi$ s^{-1} corresponding to a period $T = 2$ s, and the Reynolds number (based on the diameter of the cylinder and the

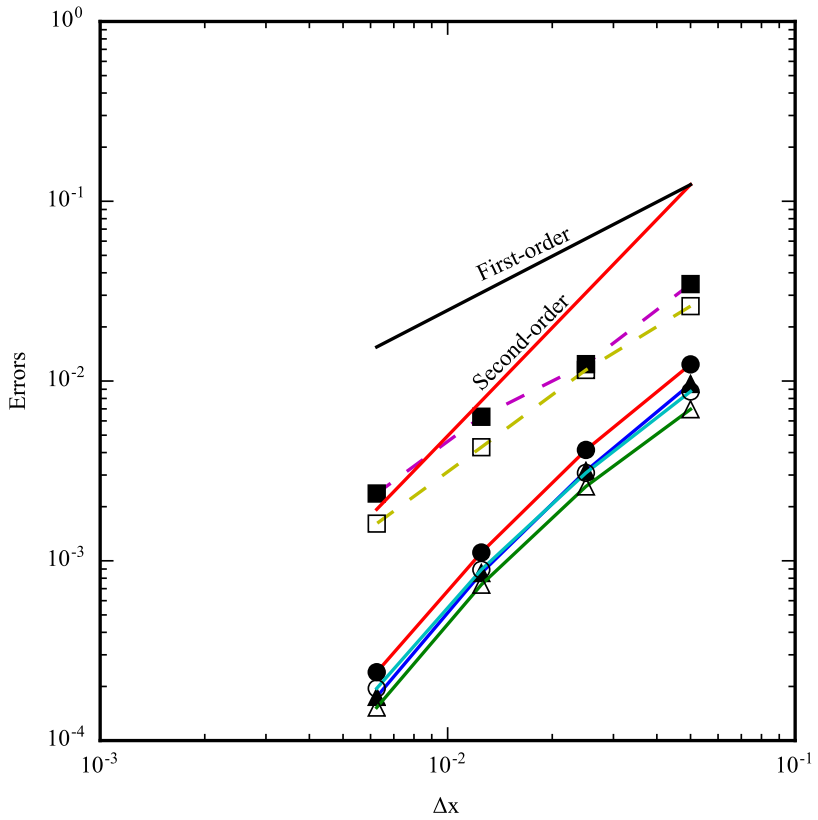


Figure 2.21: Error norms of a fixed circular cylinder in the cavity when the forcing cells are outside the cylinder. (△) L_1 norm; (◯) L_2 norm; (□) L_∞ norm; filled symbols are for u and open for v

maximum transverse velocity) is 12.56. Simulations are carried out on the same set of meshes used in the previous section. The result on the finest mesh with 320×320 nodes is regarded as the reference to calculate the error norms.

The implicit Euler scheme is used for the temporal discretization. The time step size for each mesh is set such that the Courant number is 0.2. In addition, The situations are avoided in which fluid cells change directly to solid cells, or vice versa. Body-fitted boundary conditions are applied on the boundaries of the cavity, and the circular cylinder is represented using the IBM. Specifically, Dirichlet and Neumann boundary conditions are imposed for velocity and pressure respectively on the sides of the cavity. A no-slip boundary condition for velocity on the cylinder is imposed through the IBM. The velocity boundary condition at every time step is calculated

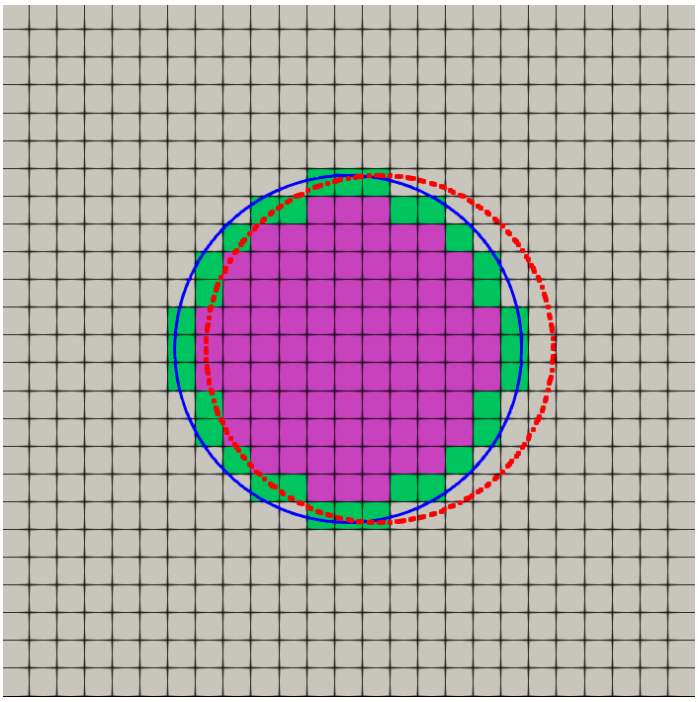


Figure 2.22: Types of cells are changing when the IB surface is moving. Solid circle is at the current time step, and the dashed circle is at the next time step

via Eqn. 2.39.

Fig. 2.23 and Fig. 2.24 show the error norms at $t = 0.25T$ corresponding to the time instant when the cylinder is located at its extreme right position. The results exhibit the same trend as in the previous sections. The layout of the forcing cells does not affect the order of accuracy. When the mesh is relatively coarse, the order of accuracy is between first and second order. After refining the mesh, the results show globally second order. Whereas the local order of accuracy is between first and second order.

Considering the situation in which the forcing cells change to fluid cells at the following time step, the velocity interpolated at the current time step is used as the interpolation stencil at the following time step. When using the PISO algorithm, more than one PISO iteration is typically used. The velocity in such cells is corrected by solving the N-S equations at the new time step. Therefore, no special treatment is

needed for the velocity in the forcing cells which will change to solid cells or fluid cells at the following time step. The results shown in Fig. 2.23 and Fig. 2.24 demonstrate that the order of accuracy of the solver is not affected when the types of cells change due to body motion.

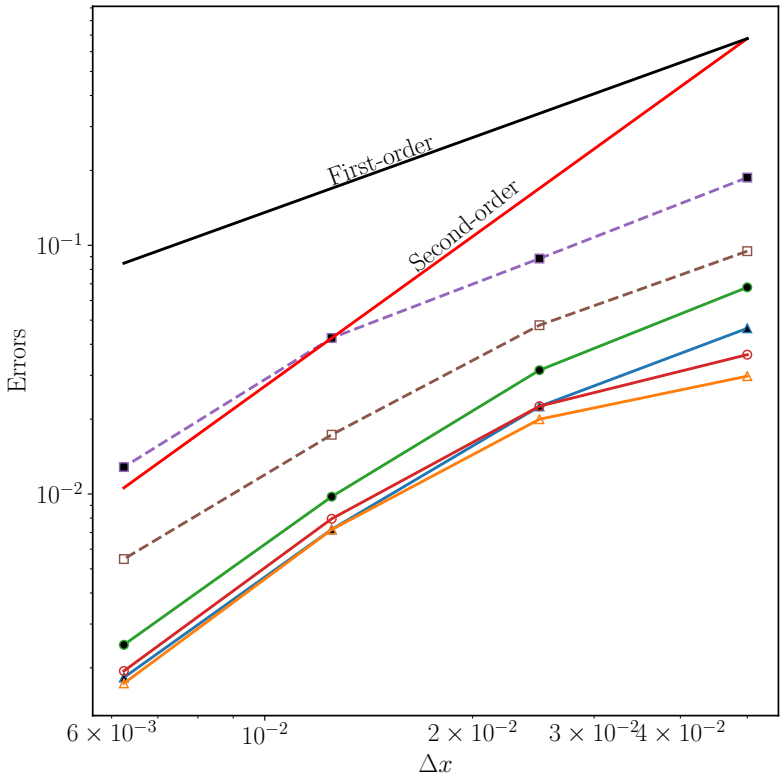


Figure 2.23: Error norms of an oscillating cylinder in cavity at $t = 0.25T$ when the forcing cells are inside the cylinder

It is also critical to examine the force on the body. Fig. 2.25 shows the horizontal force on the cylinder in one period calculated on the mesh with 160 cells. As a comparison, the result generated by using the original *pimpleFoam* solver using deforming body-fitted mesh with similar cell spacing is also shown. The time history of force when the forcing cells are outside the cylinder matches very well with the body-fitted result with respect to both amplitude and phase. When the forcing cells are inside the cylinder, it correctly predicts the phase, while the amplitude of the force is slightly overpredicted. In addition, there are more noises around the extreme values of the

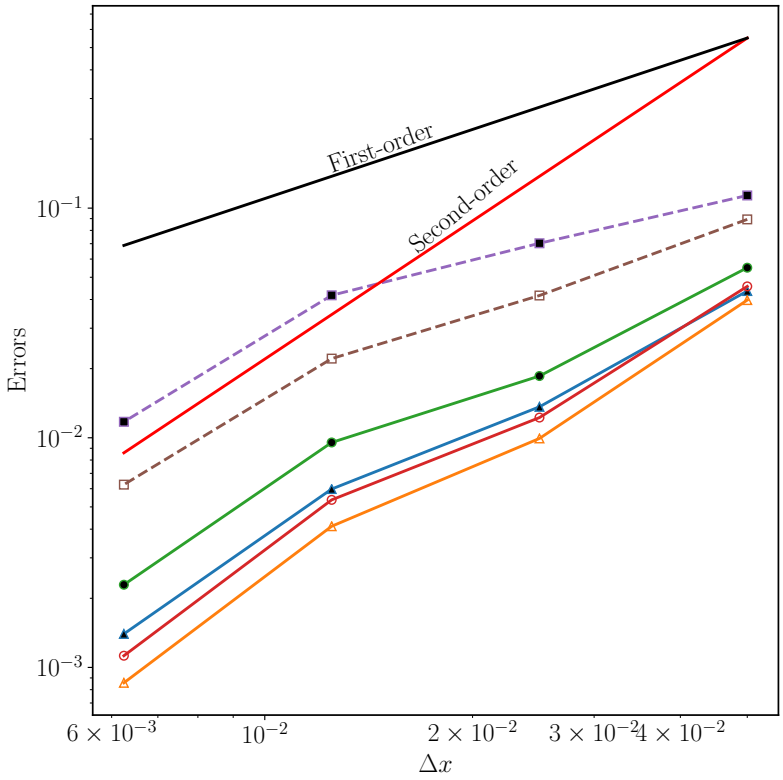


Figure 2.24: Error norms of an oscillating cylinder in cavity at $t = 0.25T$ when the forcing cells are outside the cylinder

force compared with the results by placing the forcing cells outside the cylinder.

A comparison of the IB mesh and the body-fitted mesh is shown in Fig. 2.26. The quality of the IB mesh is better than that of the body-fitted mesh. Especially when the body starts to move, the body-fitted mesh is distorted, which affects the numerical accuracy by increasing the non-orthogonality and skewness of the mesh. Moreover, for more complex geometries or large motions, the distorted mesh can also result in slow convergence or even divergence of the simulation. By contrast, the mesh quality is preserved when the IBM is adopted, which improves the order of accuracy and also makes the simulation more stable.

In summary, the current IBM can preserve the order of accuracy of the underlying discretization schemes regardless of the layout of the forcing cells. The IBM reaches the same order of accuracy and gets similar results using either of the layouts. The

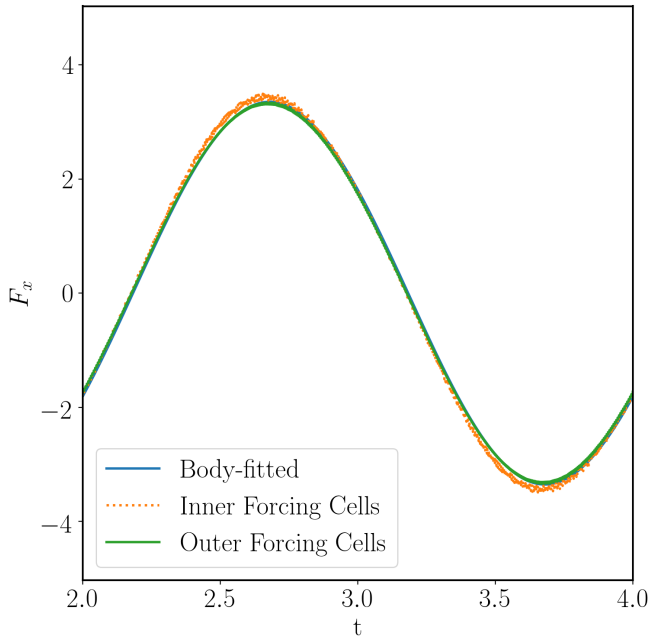
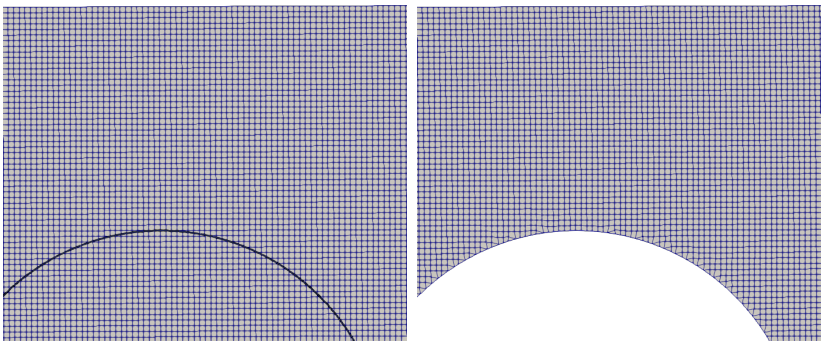


Figure 2.25: Comparison of the time history of the horizontal force on the cylinder

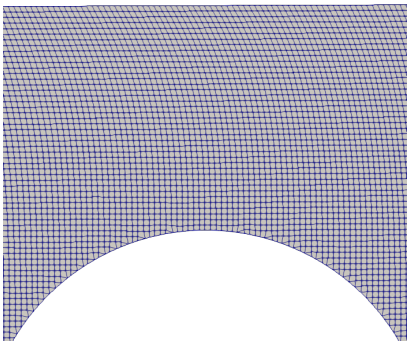
biggest difference appears when calculating the force on a moving body. Placing the forcing cells outside the body not only largely reduces the oscillation in the force, but also improves the overall accuracy. In addition, there is no extra numerical cost and extra complexity between the two layouts. Therefore, placing the forcing cells outside the IB surface is considered to be more suitable for the current implementation. Advantages of this layout will be further discussed when turbulence modeling is involved. In the reminder of this thesis, simulations will exclusively use forcing cells placed outside IB surfaces.

2.6.6 Flow around a Stationary Circular Cylinder at $Re = 200$

Despite that tests of order of accuracy is a good practice to show how fast the accuracy improves as the mesh is refined, it is not clear how accurate the results are in an absolute sense. Therefore, it is equally important to examine the accuracy of the solver by simulating benchmark cases and comparing with numerical and ex-



(a) IB mesh (b) Initial body-fitted mesh



(c) Body-fitted mesh deforms due to the body motion

Figure 2.26: A comparison of the IB mesh and the body-fitted mesh over the top of the oscillating circle

perimental data. In this section, the accuracy is demonstrated by investigating the well-established case of the laminar flow around a fixed circular cylinder at $Re = 200$, where the Reynolds number is defined as:

$$Re = \frac{DU}{\nu} \tag{2.40}$$

in which D is the diameter of the cylinder, U is the free-stream velocity, and ν is the viscosity.

As reported by experiments and other numerical results, asymmetric vortex shedding is observed at this Reynolds number due to instability. Despite the relative low

Reynolds number, the case is challenging because the accurate prediction of the force on the cylinder and the vortex shedding largely depends on how well the flow around the cylinder is resolved.

In this section, the drag coefficient C_D, the lift coefficient C_L, and the Strouhal number St are quantitatively investigated and compared with other numerical and experimental results. The force coefficients and the Strouhal number are defined as:

$$C_D = \frac{F_D}{0.5\rho U^2 A} \qquad C_L = \frac{F_L}{0.5\rho U^2 A} \qquad St = \frac{f_s D}{U} \tag{2.41}$$

in which, F_D and F_L are the drag and lift forces, respectively, ρ is the density, U is the free-stream velocity, A is the projected area of the cylinder equal to the diameter of the cylinder, and f_s is the frequency of vortex shedding.

The size of the computational domain is $50D \times 50D$. The cylinder is located at the center of the domain. Three non-uniform meshes with increasing numbers of cells and local refinement around the cylinder are considered. The number of cells of each mesh is 249^2, 498^2 and 996^2, respectively. The local refinement around the cylinder is shown in Fig. 2.27.

Fig. 2.28 shows the time histories of lift and drag coefficients on the three levels of meshes. The results of $C_{D,\text{ave}}$, $C_{D,\text{rms}}$, $C_{L,\text{rms}}$, and St are listed in Table 2.1. Table 2.1 shows that all variables converge with refining the mesh and all relative errors between the medium and finest levels are less than 1.0%. Considering the compromise between accuracy and computational cost, only the medium mesh is used for the cases of the transversely oscillating cylinder in the next section.

Table 2.1: $C_{D,\text{ave}}$, $C_{D,\text{rms}}$, $C_{L,\text{rms}}$ and St on the three levels of meshes

Mesh	$C_{D,\text{ave}}$	$C_{D,\text{rms}}$	$C_{L,\text{rms}}$	St
249^2	1.379	0.0456	0.671	0.192
498^2	1.357	0.0446	0.677	0.195
996^2	1.344	0.0442	0.676	0.195

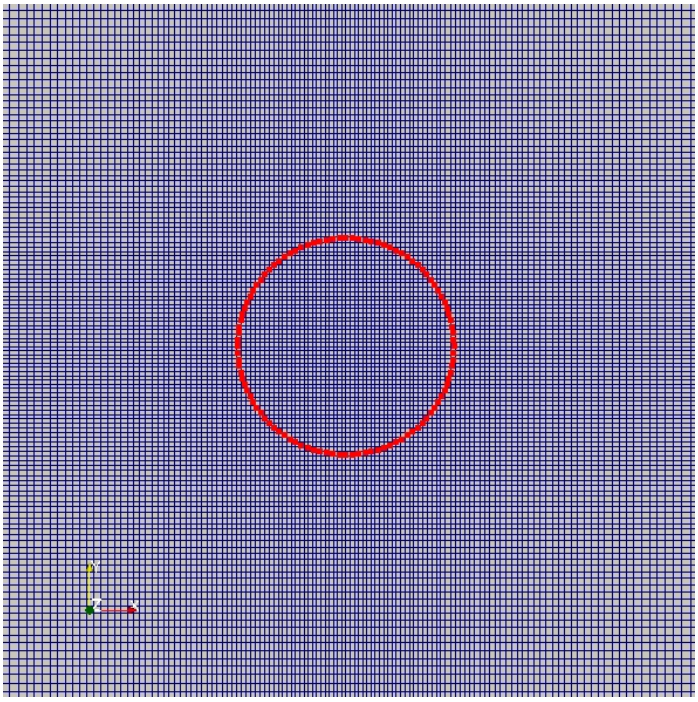


Figure 2.27: Local refinement around the cylinder on the mesh with 498^2 cells

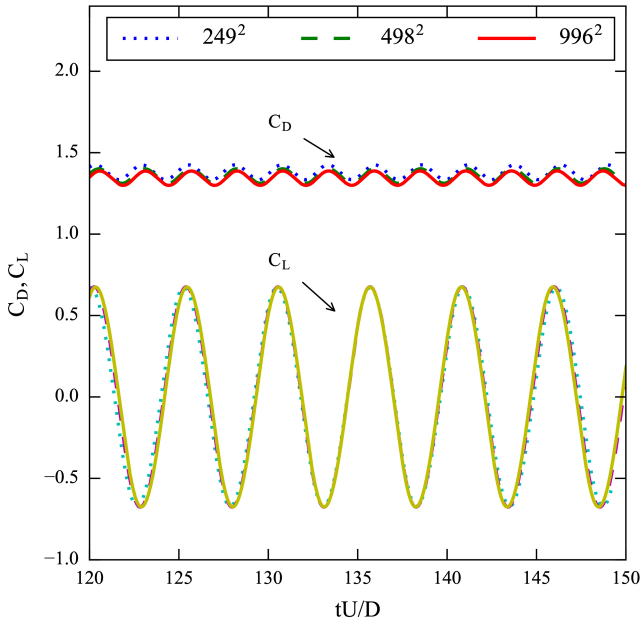


Figure 2.28: Time histories of lift and drag coefficients on the three levels of meshes

Comparison is made with several numerical and experimental results shown in Table 2.2. The predicted lift and drag forces are within the spread of other published

results. In addition, the correct prediction of the St number indicates the vortex shedding is captured correctly.

Table 2.2: C_D, C_L and St for flow over a circular cylinder at $Re = 200$

Source	$C_{D,\text{ave}}$	$C_{D,\text{rms}}$	$C_{L,\text{rms}}$	St
Present (498^2)	1.357	0.045	0.677	0.195
Calhoun (2002)	1.17	0.058	0.67	0.202
Russell and Wang (2003)	1.29	0.022	0.50	0.195
Rosenfeld et al. (1991)	1.31	0.04	0.65	0.20
Wright and Smith (2001)	1.33	0.04	0.68	0.196
Braza et al. (1986)	1.40	0.05	0.75	-
Liu et al. (1998)	1.31	0.049	0.69	0.192
Choi et al. (2007)	1.36	0.048	0.64	0.191

2.6.7 Transversely Oscillating Cylinder in a Free-stream

In this section, the numerical test of a transversely oscillating cylinder in a free-stream is conducted. This test case uses the same computational domain and geometry as in the previous section. The free-stream velocity is set to be 0.925 m/s corresponding to $Re = 185$. The motion of the cylinder in the direction perpendicular to the direction of the free-stream is prescribed as:

$$y(t) = A\sin(2\pi f_e t) \tag{2.42}$$

in which, $A = 0.2D$ is the amplitude of the motion, and f_e is the excitation frequency. Two excitation frequencies are considered: $f_e/f_0 = 0.8$ and 1.2, where f_0 is the natural shedding frequency of the stationary cylinder. The results are compared with numerical results in *Yang and Balaras* (2006) and *Yang and Stern* (2012).

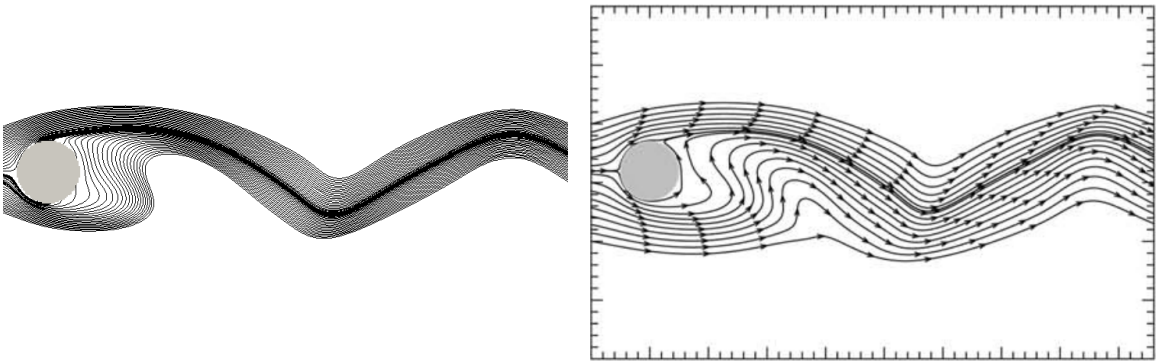
Fig. 2.29 shows the comparison of the instantaneous streamlines when the cylinder is at the extreme upper position. The present numerical results match well with the reference data at both oscillating frequencies. Compared with $f_e/f_0 = 0.8$, at $f_e/f_0 = 1.2$ the flow separates from the upper side of the cylinder, and has stronger

interaction with the vortex formed from the lower side. Therefore, a stronger recirculation appears behind the cylinder. Fig. 2.30 shows the instantaneous vorticity contours at the same time instant. The same conclusion can be drawn from the comparison of the streamlines. The present numerical results show good agreement with other numerical results with respect to the vortex structure.

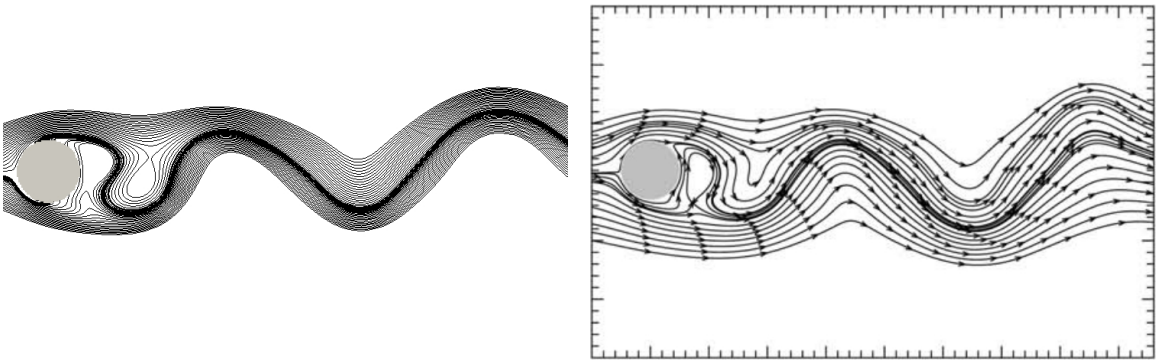
Fig. 2.31 and Fig. 2.32 show the time histories of the lift and drag coefficients. At $f_e/f_0 = 0.8$, the time histories of both drag and lift coefficients are dominated by one single frequency. The present results of drag and lift coefficients agree well with the other numerical results in terms of both amplitude and phase. At $f_e/f_0 = 1.2$, the time histories of both life and drag coefficients become irregular due to the strong recirculation behind the cylinder. Discrepancy between the present results and the other numerical results is more notable at this frequency of oscillation, especially between $150 < tU/D < 170$. However, the phenomenon that both lift and drag forces are influenced by a higher harmonic is overall well captured. The accuracy of the force prediction is satisfying especially when $tU/D > 170$.

2.7 Summary

In this chapter, the governing equations, the solution procedure and the numerical details of the proposed IBM are described. The order of accuracy of the underlying discretization schemes and the IBM are carefully investigated through a series of test cases with increasing complexity, including the order of accuracy of the discretization of the convection and diffusion terms, solution of the N-S equations, and simulations for both stationary and moving bodies. As for the convection term, OpenFOAM can achieve second order of accuracy on both structured and unstructured meshes. It can also achieve second order of accuracy for the discretization of the diffusion term on structured meshes. However, on unstructured meshes, OpenFOAM exhibits first-order accuracy even when the correction for non-orthogonality is taken into account.



(a) Present work (b) *Yang and Balaras* (2006)

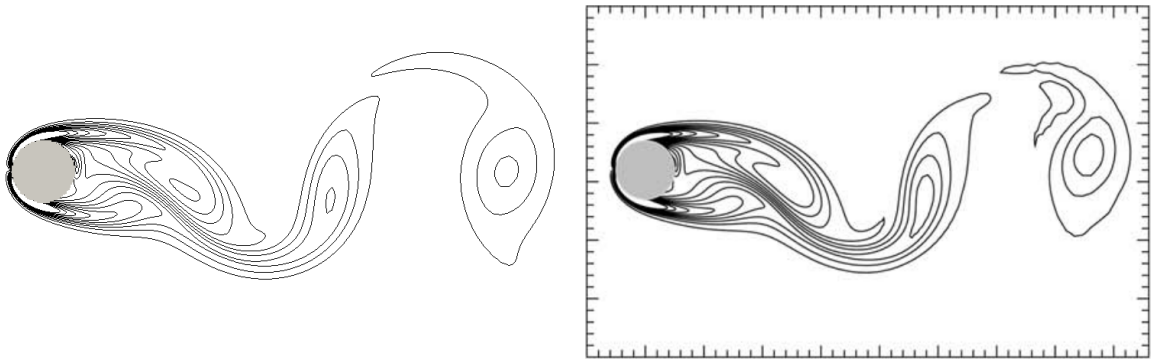


(c) Present work (d) *Yang and Balaras* (2006)

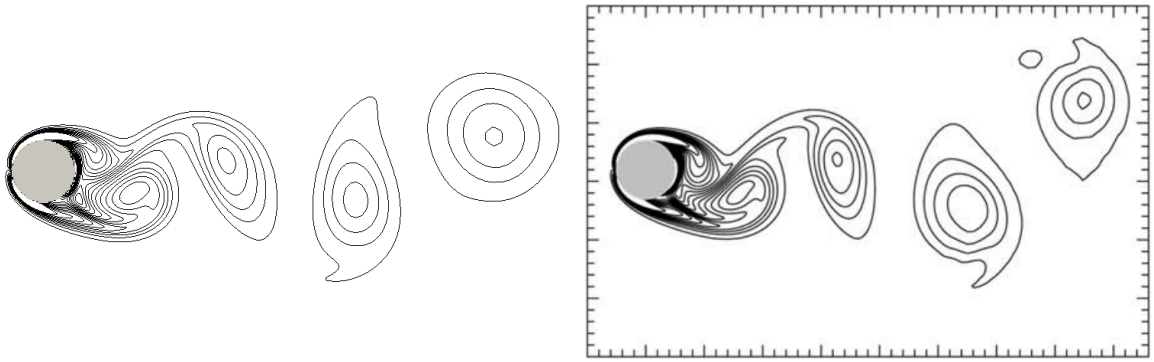
Figure 2.29: Cylinder oscillating transversely in a free-stream with different frequencies: instantaneous streamlines when the cylinder is located at its extreme upper position. Top row: $f_e/f_0 = 0.8$, bottom row: $f_e/f_0 = 1.2$

Unfortunately, non-orthogonality and skewness are almost unavoidable in practice with curved surface or when the surface is moving. Since for the present IBM, the grid lines do not conform the immersed surface, the mesh quality is preserved regardless of the shape of the surface and its motion. Therefore, the global second order of accuracy on orthogonal structured meshes is preserved with the IBM.

Next, different layouts of the forcing cells are tested. The results show no notable difference as for the order of accuracy. Discrepancy appears when calculating the force on the immersed surface. Although the phase of the time history of force is well predicted by both layouts, there is less oscillation in force by placing the forcing cells outside the immersed surface. In addition, it better predicts the amplitude of the force. Finally, the accuracy of the proposed IBM is investigated through the



(a) Present work (b) *Yang and Balaras* (2006)



(c) Present work (d) *Yang and Balaras* (2006)

Figure 2.30: Cylinder oscillating transversely in a free-stream with different frequencies: instantaneous vorticity contours when the cylinder is located at its extreme upper position. Range of vorticity: $-6 < \omega < 6$. Top row: $f_e/f_0 = 0.8$, bottom row: $f_e/f_0 = 1.2$

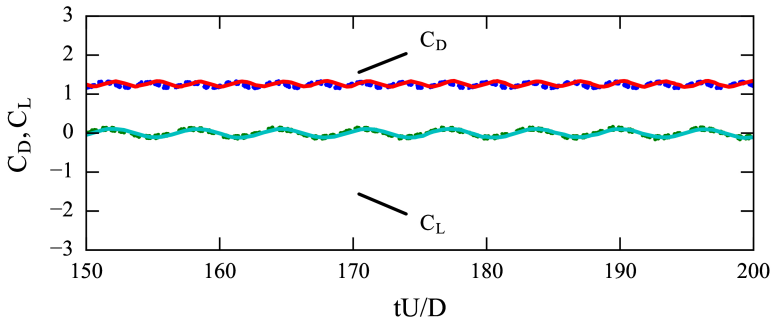


Figure 2.31: Cylinder oscillating transversely in free-stream: time histories of the lift and drag coefficients at $f_e/f_0 = 0.8$: (——) *Yang and Stern* (2012); (- - - -) the present work.

simulations of the flows around fixed and oscillating cylinders. The lift and drag forces, and main flow structures are compared with other numerical resources, and

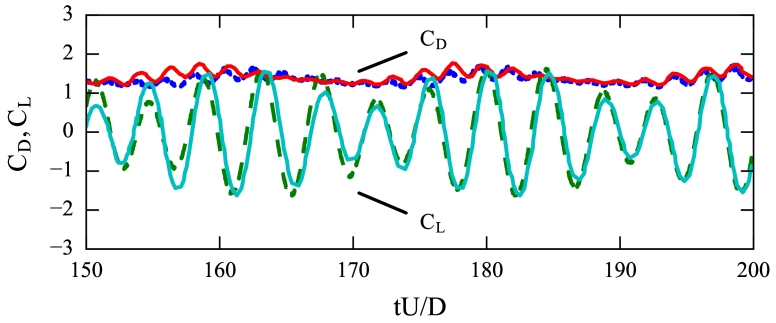


Figure 2.32: Cylinder oscillating transversely in free-stream: time histories of the lift and drag coefficients at $f_e/f_0 = 1.2$: (——) *Yang and Stern* (2012); (- - - -) the present work.

good agreement is achieved.

CHAPTER III

Development for Single-Phase Turbulent Flows

This chapter discusses the coupling between the IBM and turbulence modeling in RANS simulations. Specifically, the Spalart-Allmaras turbulence model developed by *Spalart and Allmaras* (1992) is considered. In this chapter, the governing equations for turbulent flows are presented. Integration of the turbulence model into the proposed IBM framework is then described. In this thesis, only the layout of placing the forcing cells outside IB surfaces is considered for the RANS simulations. The main reasons are discussed later in this chapter. In addition, since for a RANS simulation it is reasonable to expect the mesh is not fine enough to fully resolve boundary layers, a universal wall function is used which gives a continuous velocity profile from the outer edge of the logarithmic layer down to the wall. In the second part of this chapter, the proposed method is fully validated on cases involving different geometries including a flat plate, an asymmetric diffuser, a pitching airfoil, and a ship model.

3.1 Governing Equations

In the framework of the current IBM, incompressible turbulent flows can be described by the RANS equations written as:

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

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{uu}) = \nabla \cdot \left[\nu_{\text{eff}} \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)\right] - \nabla p + \mathbf{f} \tag{3.2}$$

Comparing between Eqn. 3.2 and Eqn. 2.9, the difference is ν is replaced by effective viscosity $\nu_{\text{eff}} = \nu + \nu_t$, where ν_t is the turbulent viscosity. In addition, $\mathbf{u}$ and p represent the Reynolds-averaged flow quantities.

To close the governing equations, the Spalart-Allmaras turbulence model is used to calculate ν_t. One additional transport equation is solved for the modified turbulent viscosity $\tilde{\nu}$:

$$\frac{D}{Dt}\tilde{\nu} = \nabla \cdot (D_{\tilde{\nu}_{\text{eff}}} \nabla \tilde{\nu}) + \frac{C_{b2}}{\sigma_{\nu_t}} |\nabla \tilde{\nu}|^2 + C_{b1} \tilde{S} \tilde{\nu} - C_{w1} f_w \frac{\tilde{\nu}^2}{y^2} \tag{3.3}$$

in which, y is the distance to the nearest wall and other coefficients are listed in Appendix. B. The wall boundary condition of $\tilde{\nu}$ is $\tilde{\nu}_{\text{wall}} = 0$.

The turbulent viscosity is subsequently obtained using:

$$\nu_t = \tilde{\nu} f_{v1} \tag{3.4}$$

where the function f_{v1} is given by

$$f_{v1} = \frac{\chi^3}{\chi^3 + C_{v1}^3} \tag{3.5}$$

and

$$\chi = \frac{\tilde{\nu}}{\nu} \qquad C_{v1} = 7.1 \tag{3.6}$$

3.2 Wall Modeling

In practice, it is reasonable to expect the mesh is not fine enough to fully resolve the turbulent boundary layers at high Reynolds numbers. In this case, a wall function is favorable to help predict the wall shear stress. In the present IB framework, the

Spalding's velocity profile (*Spalding* (1961)) is used to predict the friction velocity u_τ on $\mathcal{S}$:

$$y^+ = u^+ + \frac{1}{E}\left[e^{\kappa u^+} - 1 - \kappa u^+ - \frac{1}{2}(\kappa u^+)^2 - \frac{1}{6}(\kappa u^+)^3\right] \tag{3.7}$$

in which, $\kappa = 0.41$ is the Von Karman constant, and $E = 9.8$. The non-dimensional near-wall distance and velocity are defined as:

$$y^+ = \frac{y u_\tau}{\nu} \qquad u^+ = \frac{u_\parallel}{u_\tau} \tag{3.8}$$

where y is the distance to the wall, and $u_\parallel$ is the magnitude of the velocity component tangential to the IB. Eqn. 3.8 provides a universal velocity profile from the outer edge of the logarithmic layer down to the wall as plotted in Fig. 3.1.

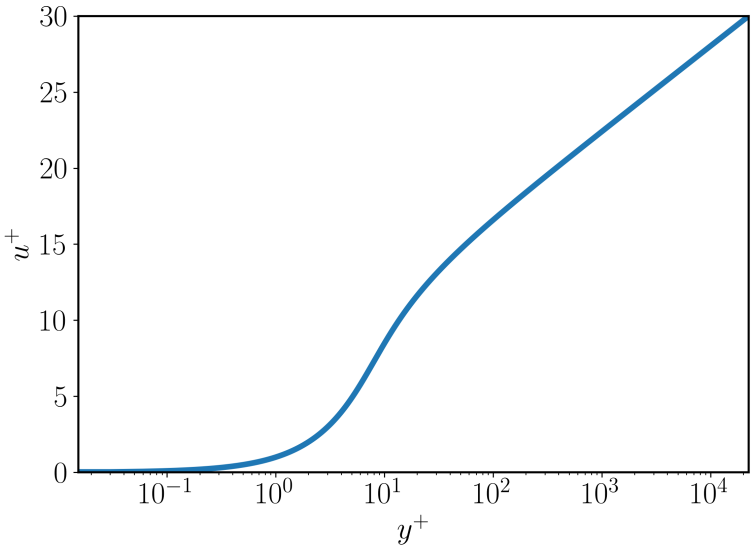


Figure 3.1: The Spalding's velocity profile: u^+ vs. y^+ in the near wall region

3.3 Implementation of the IBM

The Spalart-Allmaras turbulence model needs to be modified to couple with the present IBM. First of all, the wall distance y of every cell center is corrected by taking

into account the presence of the immersed surface $\mathcal{S}$. If the surface is stationary, y needs to be calculated only once at the beginning of the simulation. If the surface is moving, y needs to be recalculated at the beginning of every time step. Since the turbulence model is solved after the RANS equations, the velocity in Eqn. 3.3 already satisfies the velocity boundary condition. Subsequently, only $\tilde{\nu}$ needs to be modified to enforce its boundary condition on the exact immersed surface. The modification is based on the near-wall behavior of $\tilde{\nu}$. $\tilde{\nu}$ can be non-dimensionalized by:

$$\tilde{\nu}^+ = \frac{\tilde{\nu}}{\nu} \tag{3.9}$$

As pointed out in *Kalitzin et al.* (2005), the non-dimensionalized modified eddy-viscosity $\tilde{\nu}^+$ has a linear profile in the near-wall region, and can be described as:

$$\tilde{\nu}^+ = \kappa y^+ \tag{3.10}$$

where, $\kappa = 0.41$ is the von Karman constant.

The linear profile satisfies both the transport equation of $\tilde{\nu}$, and the boundary condition of $\tilde{\nu} = 0$ on the wall. Therefore $\tilde{\nu}$ in the forcing cells can be calculated from Eqn. 3.10 as:

$$\tilde{\nu} = \kappa y u_\tau \tag{3.11}$$

In the solid cells $\tilde{\nu}$ is simply set to be zero.

It is worth comparing the influence of different layouts of the forcing cells. The turbulence model requires that $\tilde{\nu}$ is non-negative in the entire computational domain. Negative values usually result from inappropriate boundary conditions of $\tilde{\nu}$, use of unbounded discretization, low quality of meshes and many other factors. They often lead to the divergence of the simulation. Normally after solving the transport equation Eqn. 3.3, $\tilde{\nu}$ is bounded to ensure its value is non-negative. Consider placing the forcing

cells inside the immersed surface and $\tilde{\nu} = 0$ on the immersed surface $\mathcal{S}$. $\tilde{\nu}^*$ in the forcing cells can be interpolated according to Eqn. 2.13 as:

$$\tilde{\nu} = -\frac{h_1}{h_2}\tilde{\nu}_{\text{ext}} = -\frac{h_1}{h_2}\kappa y u_\tau \leq 0 \tag{3.12}$$

which always introduces negative $\tilde{\nu}$ in the forcing cells.

In comparison, for the layout of placing forcing cells outside the immersed surface $\mathcal{S}$, $\tilde{\nu}^*$ in the forcing cells is interpolated as:

$$\tilde{\nu} = \frac{h_1}{h_2}\tilde{\nu}_{\text{ext}} = \frac{h_1}{h_2}\kappa y u_\tau \geq 0 \tag{3.13}$$

$\tilde{\nu}$ remains non-negative automatically without additional bounding operation after the interpolation. Moreover, flow variables in the forcing cells have more physical meanings when they are outside the immersed surface in the flow domain. Therefore, only the layout of placing the forcing cells outside is considered for the purpose of coupling with the turbulence model.

Apparently, u_τ should be calculated in advance to evaluate $\tilde{\nu}$ in the forcing cells. u_τ is calculated by using Eqn. 3.7 as follows: At the extended point P_{ext} in Fig. 2.3, the magnitude of the velocity tangential to the wall is calculated as Eqn. 3.14 after solving the momentum equations.

$$u_{\parallel} = \parallel \mathbf{u}_{\text{ext}} - (\mathbf{u}_{\text{ext}} \cdot \mathbf{n})\mathbf{n} \parallel \tag{3.14}$$

in which, $\mathbf{n}$ is the unit wall-normal vector passing through P_{ext}.

Given $u_{\parallel}$ and y (which is the distance from P_{ext} to the wall), u_τ is calculated from Eqn. 3.7 using the Newton-Raphson method. A initial guess of u_τ is required to start the iteration. At the first time step, the initial guess of u_τ is determined with $y^+ = 30$. Thereafter, u_τ in the previous time step or previous PISO iteration is used

as the initial guess.

3.4 IB Wall Function

As shown in Fig. 3.1, if the cell centers of forcing cells are in the logarithmic layer, the linear interpolation in Eqn. 2.13 actually introduces large error. In this circumstance, a wall function which can properly adjust the velocity in the forcing cells is required. After the velocity in the forcing cells is interpolated by Eqn. 2.13, it is decomposed into the wall-normal and wall-tangential components as:

$$\mathbf{u}_{\perp} = (\mathbf{u} \cdot \mathbf{n})\mathbf{n} \qquad \mathbf{u}_{\parallel} = \mathbf{u} - \mathbf{u}_{\perp} \tag{3.15}$$

in which $\mathbf{n}$ is the same as in Eqn. 3.14, since P_{ext}, P_{IB} and P_{FC} are collinear.

Subsequently, the wall-tangential component is corrected using Eqn. 3.7 and u_τ calculated from the interpolated velocity at P_{ext}. Therefore the velocity in the forcing cells has a wall-tangential component that satisfies the Spalding's velocity profile, and a wall-normal component that satisfies the linear interpolation between the velocity on the wall and the velocity in the flow field.

The complete solution procedure for turbulent flows is summarized as the following steps:

1. Start a time step. Categorize the background cells into fluid cells, forcing cells and solid cells.

2. Interpolate the velocity in the forcing cells using the Laplacian weight interpolation and Eqn. 2.13. In addition, set the velocity in the solid cells to be the rigid body velocity.

3. Calculate the wall shear velocity via Eqn. 3.7 based on the velocity and wall distance at P_{ext}.

4. Correct the velocity component tangential to the immersed surface at the cell center of forcing cells via Eqn. 3.7.

5. Calculate the source term $\mathbf{f}$ in the momentum equation by Eqn. 2.10.

6. Solve the modified RANS equations using PISO algorithm.

7. Enforce $\tilde{\nu}$ in the forcing cells and solid cells by Eqn. 3.11.

8. Solve the transport equation for $\tilde{\nu}$ and update ν_t in the whole domain.

9. March to the next PISO iteration or the next time step.

3.5 Numerical Results

The turbulent flow solver is validated through numerical cases involving 2-D and 3-D, fixed and moving immersed surfaces. Different aspects of the flow are examined including forces on the immersed surface, near-wall profiles of flow variables and wall shear stress. The accuracy of the solver is carefully investigated through the test cases.

3.5.1 Turbulent Flow Over a 2-D Flat Plate

The turbulent flat plate case is carried out at Reynolds number of $Re = 2.5 \times 10^6$ based on the length of the flat plate. It is a simplified version of the NASA test case (*NASA* (2018b)), in which there is a ramp before the plate. The length of the plate is 2 m. The velocity and turbulent viscosity ν_t are examined along the vertical line at $x = 0.97$ m. In addition, the surface skin friction coefficient C_f along the plate is investigated.

Since the case is steady-state, the time derivative term is neglected. Numerical tests are carried out by both the body-fitted (BF) mesh solver (the default *simple-*

Foam) and the present IB solver. Details of the case setup are summarized in Table 3.1.

Table 3.1: Description of the flat plate case setup

Name	Solver	Number of cells (wall tangent× normal direction)	Near-wall y^+	Wall function
BF_ref	BF	50× 50	1	N
BF_WF	BF	50× 50	52	Y
IBM_ref	IB	50× 60	1	N
IBM_WF	IB	50× 60	52	Y

It should be noted the IB meshes have additional cells in the wall normal direction because it has cells in the solid domain. The immersed surface, which is the flat plate in this case, coincides with the grid line, so the cell spacing is exactly the same for both IB and BF meshes.

Fig. 3.2 shows the numerical results of the velocity profile at $x = 0.97$ m together with the Spalding's velocity profile. The result "IBM_ref" demonstrates that given a sufficiently fine mesh, the present IB implementation can accurately resolve the boundary layer. When the wall function is used, the result "IBM_WF" can also predict the velocity profile in the logarithmic region correctly.

Fig. 3.3 shows the ν_t profile at $x = 0.97$ m, in which "Fun3D" represents the numerical result from the NASA database, *NASA* (2018b). The IB results, either with or without the wall function, accurately predict the ν_t profile. In contrast, the BF mesh solver with wall function overpredicts the ν_t value. It should be highlighted the difference of how the wall function is integrated between OpenFOAM body-fitted solver and the present IB solver. In OpenFOAM, the velocity in the cell next to the wall is directly calculated from the governing equations. The frictional velocity is then calculated through wall function. To supply a correct wall shear stress to the calculation of the velocity in the near-wall cells, the ν_t value on the wall is corrected. Since the mesh is not sufficiently fine, the near-wall velocity gradient is normally

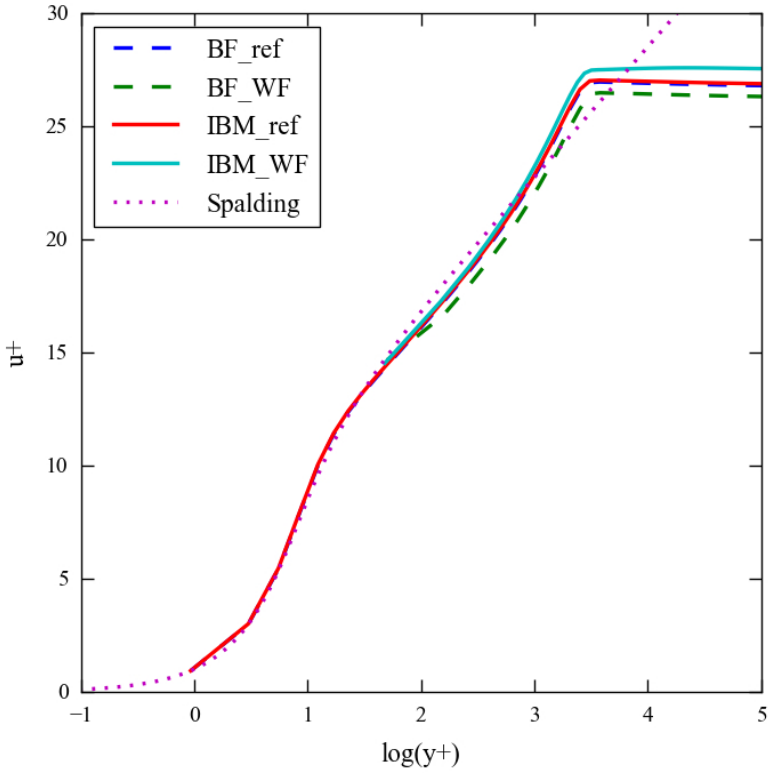


Figure 3.2: Velocity profile at $x = 0.97$ m

underpredicted, and it requires an additional amount of ν_t at the wall to obtain a correct wall shear stress in the momentum equation. Subsequently, the eddy viscosity is overpredicted. On the other hand, in the present IB implementation, the near-wall velocity and eddy viscosity are directly modified based on the wall function and the prescribed near-wall profile. Therefore, a more accurate profile of ν_t is obtained.

The surface skin friction coefficient along the plate is presented in Fig. 3.4. It can be seen that "IBM_ref" matches well with the "BF_ref" results whereas both of them slightly overpredict the surface skin friction compared with "Fun3D". This can be partially explained by the difference between the setup of the case as there is no ramp in the present simulations and difference in the number of cells. In addition, the results of NASA are obtained by a compressible solver whereas our solver is incompressible. As for using the wall function, the BF solver overpredicts the surface skin friction all along the plate. Whereas the IB solver overpredicts the surface skin

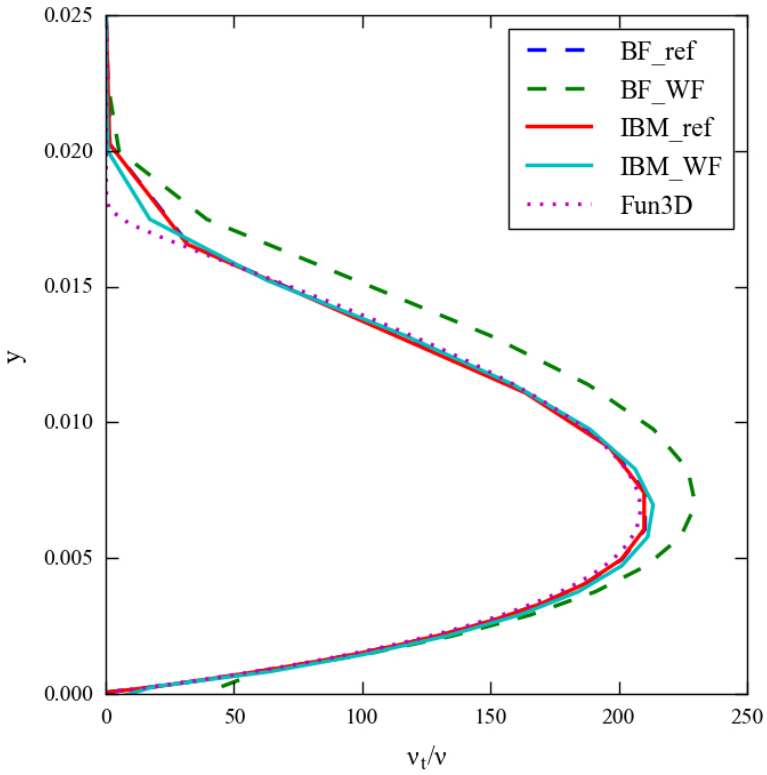


Figure 3.3: ν_t profile at $x = 0.97$ m

friction at $x < 0.5$ and underpredicts it at $x > 0.5$.

3.5.2 Turbulent Flow in an Asymmetric Diffuser

This validation study examines the separated flow through a 2D asymmetric diffuser. A detailed description of the model can be found on *NASA* (2018a). The simulation is carried out at Reynolds number of $Re = 20{,}000$ based on the centerline velocity at the inlet channel and the channel height. To generate a fully turbulent flow at the inlet similar to the experiment, the one-seventh power-law velocity profile is used to prescribe the inlet velocity. Uniform profiles of ν_t and $\tilde{\nu}$ are used which can introduce errors compared with the experiment as pointed out in *Crawford and Birk* (2015). The simulation setup is summarized in Table 3.2 and the mesh used by the IB solver is shown in Fig. 3.5. It should be noted that to obtain similar wall-normal cell spacing for "BF_WF" and "IBM_WF", more cells are used for "IBM_WF" than

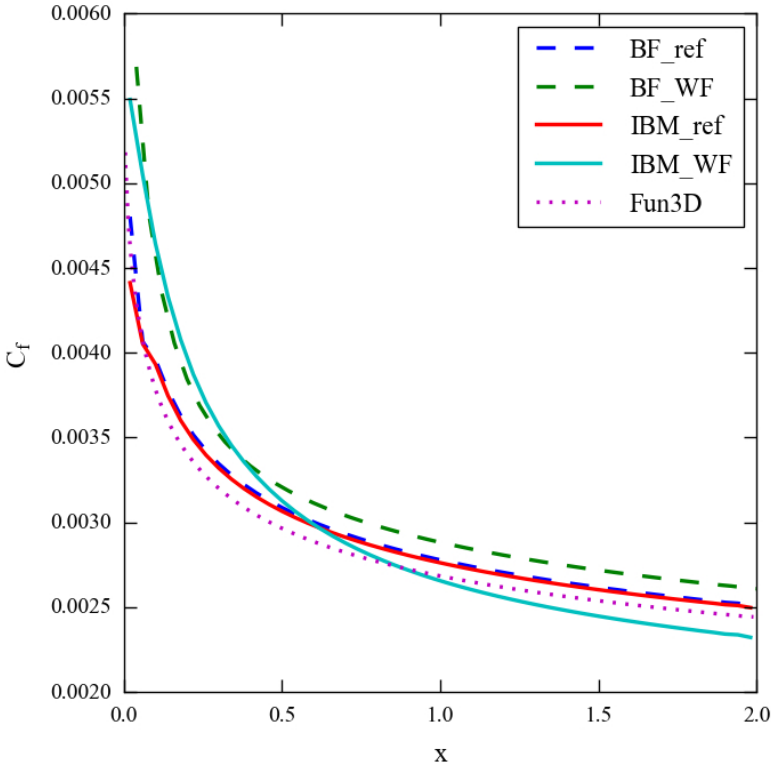


Figure 3.4: Surface skin friction coefficient along the plate

Table 3.2: Description of the diffuser case setup

Name	number of cells (fluid domain)	Near-wall y^+
BF_ref	12000	1
BF_WF	4800	15 (average) 33 (max)
IBM_WF	34491	15 (average) 33 (max)

"BF_WF".

The horizontal velocity profiles together with the experimental data (*Obi et al.* (1993)) at different locations along the diffuser are presented in Fig. 3.6. It shows that the IB solver can well predict the separation and reattachment at the bottom of the diffuser. The discrepancy between the IB result and the experimental data mainly appears at the top of the ramp. However, similar magnitude of disagreement with the experimental data also exists in the body-fitted results. Therefore, the discrepancy is more likely because the solution is very sensitive to the inlet boundary conditions

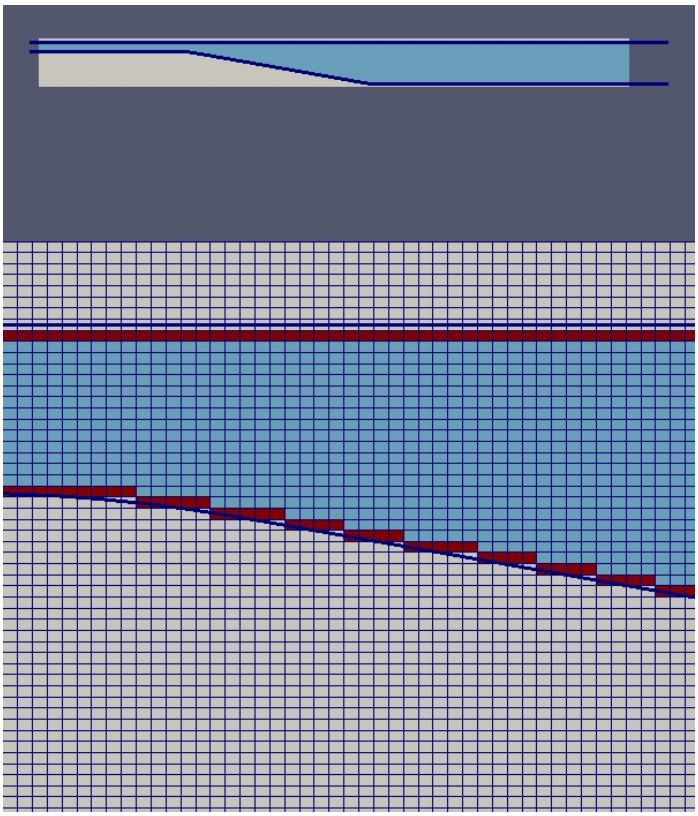


Figure 3.5: Computational mesh for the IB solver. Top and bottom are the global and local views, respectively

and the choice of turbulence model.

The accuracy of the IBM can be also demonstrated by a comparison of the streamlines as presented in Fig. 3.7. The locations of separation and reattachment are well captured by the IB simulation.

3.5.3 2D Oscillating Airfoil in Turbulent Flow

In this section, the capability of the solver is investigated by the simulation of a 2D airfoil pitching in the turbulent flow. The Reynolds number based on the chord length and free-stream flow speed is $Re = 1.95 \times 10^6$. The airfoil oscillates sinusoidally as:

$$\alpha(t) = \alpha_0 + \alpha_a \sin\left(\frac{2U_\infty \kappa}{C} t\right), \tag{3.16}$$

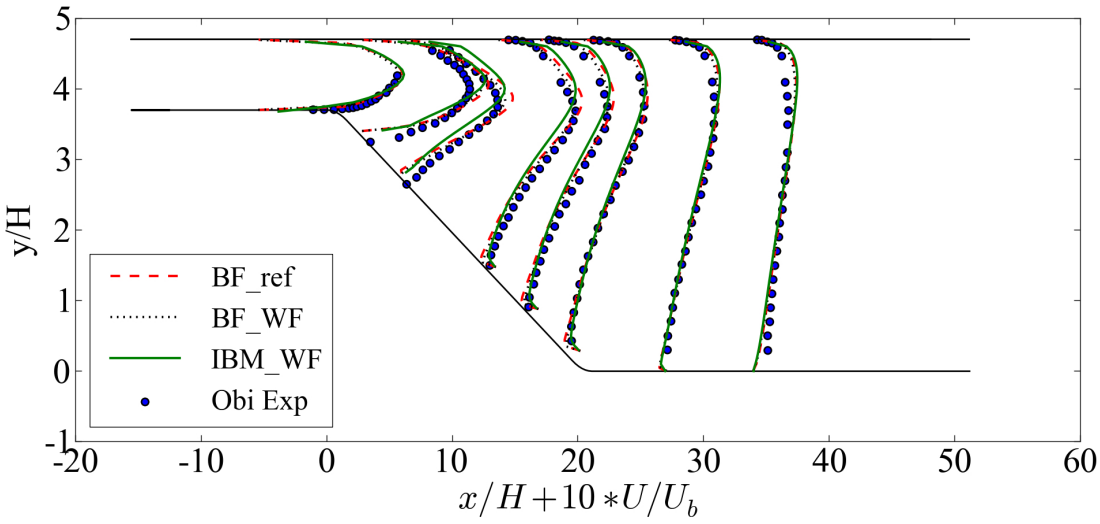


Figure 3.6: Horizontal velocity profiles at different locations

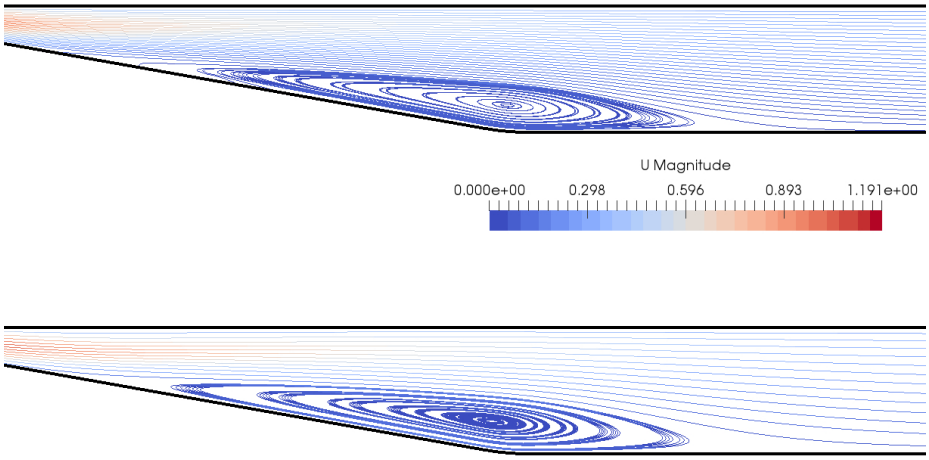


Figure 3.7: Streamlines at the bottom of the diffuser. Top: IB result; bottom: body-fitted result

in which α_0 is the mean value of the angle of attack, α_a is the amplitude of the motion, $\kappa = 0.1$ is the reduced frequency, and C is the chord length.

To compare with the experiment (*Piziali* (1994)), the pitching parameters $\alpha_0 = 4°$ and $\alpha_a = 4.2°$ are used, resulting in an attached turbulent flow. Although the flow

is fully attached, the combination of the high Reynolds number, the body motion and the use of wall function makes this case challenging enough to demonstrate the capability of the solver. The simulations are also conducted by using the IB method without using the wall function, and body-fitted mesh. The body-fitted mesh is generated using the same background mesh as the IBM.

The domain size is $15C \times 10C$ with $5C$ upstream of the airfoil, $10C$ downstream of the airfoil and $5C$ at the two sides. The mesh is built up from a very coarse background mesh and then refined near the airfoil using the OpenFOAM utility *snappyHexMesh*, resulting in a total number of cells about one hundred thousand and average $y^+ \approx 500$. The global and local layout of the mesh is shown in Fig. 3.8.

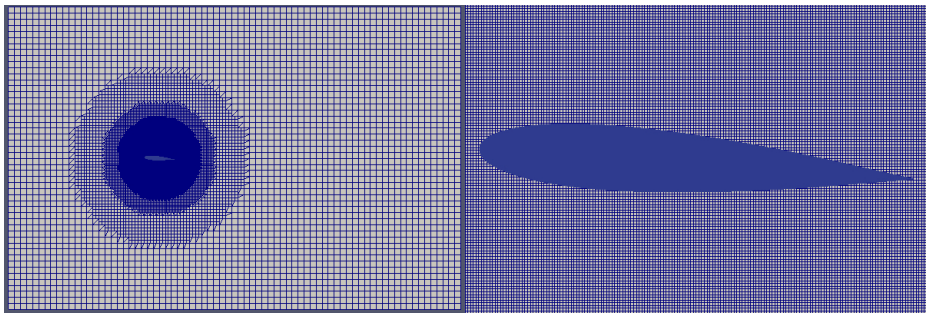
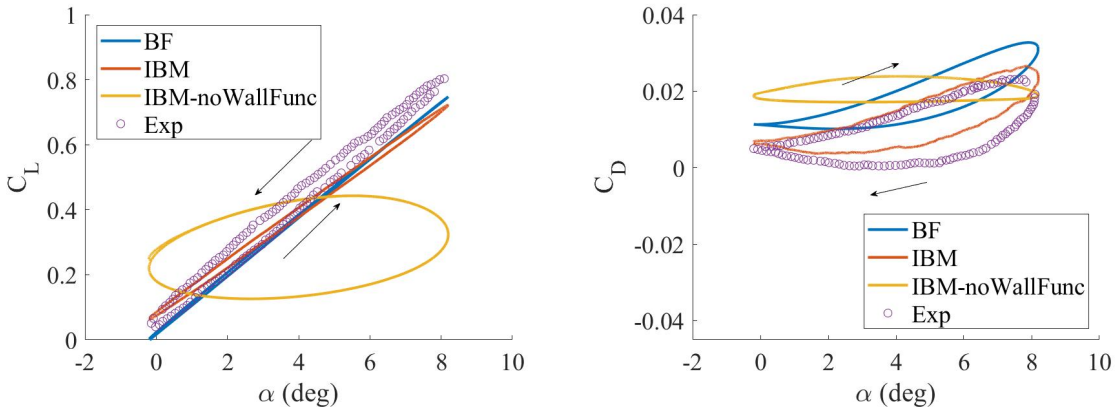


Figure 3.8: Computational mesh for the oscillating airfoil at $Re = 1.95 \times 10^6$

Comparison of the computed hysteresis loops of lift and drag coefficients is shown in Fig. 3.9. It shows that the result of C_L by using the body-fitted mesh is underpredicted, and C_D is overpredicted. However, in general the body-fitted results capture the behavior of the lift and drag forces at different pitching angles. In comparison, C_L shows the present IBM underpredicts at the top of the upstroke, and C_D shows underprediction especially during the downstroke of the circle. Moreover, C_L and C_D predicted by IBM match with the experimental data around $\alpha = 0^\circ$ better than the body-fitted results. It is because the IB background mesh is orthogonal around the airfoil, while there are mesh non-orthogonality and skewness for the body-fitted mesh. In addition, when the airfoil is pitching, the IB background mesh remains orthogonal,

while the cells of the body-fitted mesh near the airfoil are distorted, which affects the accuracy of the solution.

As shown in Fig. 3.9, the results of the IBM without the wall function fail to qualitatively predict both C_L and C_D. The comparison between the IB results with and without the wall function demonstrates that the usage of the IB wall function significantly improve the accuracy of the present IBM when the near-wall mesh is not sufficiently fine to resolve the boundary layer.



(a) Lift coefficient C_L (b) Drag coefficient C_D

Figure 3.9: Comparison of the hysteresis loop of lift and drag coefficients

The computational time is summarized in Table 3.3. The initialization of the IBM is done only once at the beginning of the simulation, so it does not contribute much to the overall computational cost. For this specific case, the computational time related to the IBM per time step is close to the time for solving the governing equations. The most time-consuming step of the IBM is to update the near-wall distance by recalculating the distance between every cell center and the IB surface. It should be noted that the breakdown of the computational time in this specific case does not necessarily hold true for all cases, but serves as an indication of the relative costs of the IBM and flow solver. One reason that the cost varies case by case is due to the calculation of the minimum distance to solid walls both IB and body-fitted. For laminar flows, this variable is only needed in the forcing cells. On the other hand,

for turbulent flows, this distance is needed for the destruction term of the turbulence equation, and as such must be calculated in every cell, including fluid, solid and forcing cells. This also has implications on the cost of the IBM for moving body simulations. In addition, the time cost for updating the near-wall distance is mainly depends on the number of cells. In comparison, the time cost for solving the flow depends not only on the number of cells, but also on the particular fluid dynamics and the quality of the mesh. Flows around complex geometries with features, such as flow separation and vertex shedding, tend to be simulated using more iterations to solve the linear systems of equations, which may significantly increase the time cost in each time step.

Table 3.3: Summary of the computational time in the simulation of the 2D oscillating airfoil

Operation	time cost (s)
Initialization of the IBM	8.64
Update the IB interpolation stencils (per time step)	0.11
Update the near-wall distance (per time step)	2.37
Apply IB boundary conditions (per time step)	0.01
Solve the flow by the governing equations (per time step)	4.88 ± 0.14

3.5.4 Resistance and Flow Pattern of a Double-body KVLCC2 Tanker

In the past several decades, CFD tools are increasingly used in the early stage to evaluate ship-hull designs. However, such marine applications by using IBMs are limited (*Yang and Stern* (2009)). For flows around a ship hull, to correctly resolve the boundary layer is critical to accurately predict the viscous shear stress on the hull. Body-fitted meshes have the advantage to cluster mesh cells in the wall-normal direction to form the prism layers along the hull surface. In comparison, to achieve the similar near-wall cell size, IBMs require far more cells since the background mesh needs refinement in all three directions. This issue is far less severe for a 2D simulation where the number of cells grows as $\mathcal{O}(n^2)$ (n is the number of cell in one direction),

Table 3.4: Principal particulars of KVLCC2

Particulars	Value
Scale ratio, λ	116
Length, L (m)	2.7586
Breadth, B (m)	0.5
Depth, D (m)	0.2586
Draft, T (m)	0.1793
Block coefficient C_B	0.8098
Reynolds number Re	4.6×10^6

or when the ratio between the near-wall cell size and size of the geometry is relatively small.

In this section, a 3D double-body simulation of a model-scale KVLCC2 hull is conducted using the proposed IBM coupling with the IB wall function. The resistance and the wake flow are compared with the experimental data of *Lee et al.* (2003) and *Kim et al.* (2001). The KVLLC2 ship model is selected because it represents the general hull form of modern tankers. Moreover, this model is widely used for CFD benchmarks of ship resistance, maneuvering and propulsion (*Larsson et al.* (2013); *Shen and Korpus* (2015), and *Stern et al.* (2011)), and thus sufficient experimental data is accessible to the public.

The principal particulars of the KVLCC2 in model scale are listed in Table 3.4. In the present simulation, the model is fixed in zero trim and sinkage condition. The simulation is so-called double-body, because a symmetric boundary condition is used on the flat free-surface plane. As a result, free-surface effect is not considered at the current stage. The free-stream velocity is set such that the Reynolds number is exactly the same as in the experiment (*Kim et al.* (2001)).

A set of three meshes are generated using *snappyHexMesh* with a refinement ratio of $\sqrt[3]{2}$ in each direction between two consecutive levels. Only the cells around the ship hull and in the wake region are refined, and the solid cells inside the ship hull are removed to speed up the simulation. The total number of cells of each mesh are

Table 3.5: Total number of cells of the meshes for double-body KVLCC2 simulation

Mesh Name	Number of Cells (Millions)
Coarse	8.09
Medium	15.89
Fine	28.56

summarized in Table 3.5. Fig. 3.10 illustrates the computational domain and the locally refined mesh around the hull.

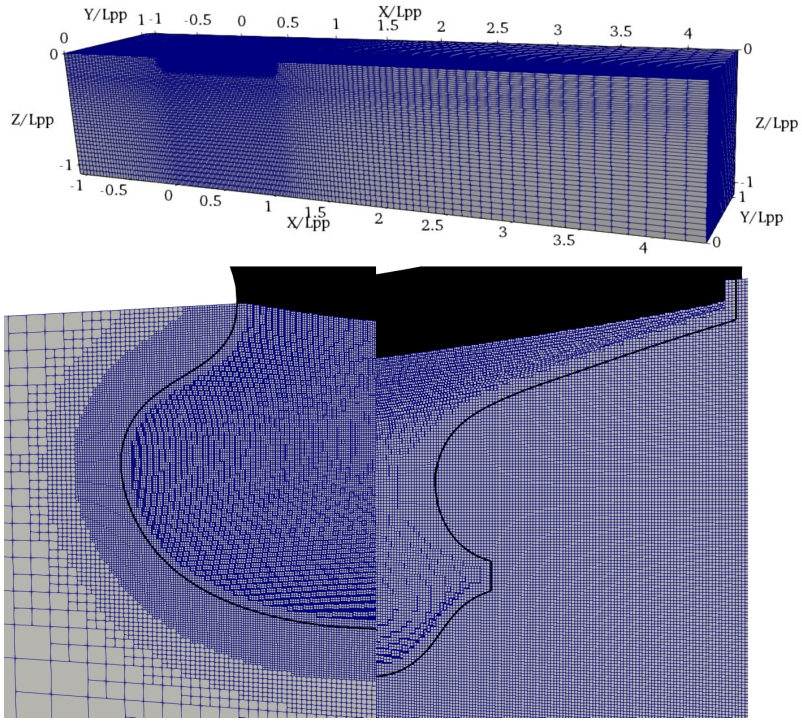


Figure 3.10: The computational domain and the local mesh for the double-body simulation of a KVLCC2 tanker

It should be noted that the free-surface effect is considered in the experiment. However, since the experiment was carried out at $Fr = 0.142$, the total resistance is dominated by the frictional resistance. Thus, it is reasonable to make the comparison between the present numerical results and the experiment. The numerical results and the relative errors with respect to the experimental data are listed in Table. 3.6. The total resistance is well predicted by the medium and fine meshes, while the relative

Table 3.6: Numerical results of C_T and the relative errors with respect to the experimental results

Mesh	Experiment ($\times 10^3$)	IBM ($\times 10^3$)	relative errors (%)
Coarse	4.110	3.897	-5.19
Medium	4.110	4.015	-2.31
Fine	4.110	4.108	-0.04

error on the coarse mesh is slightly larger than 5%. Fig. 3.11 shows the total resistance coefficient C_T predicted by different meshes comparing with the experimental data. A monotonic convergence can be observed with refining the mesh.

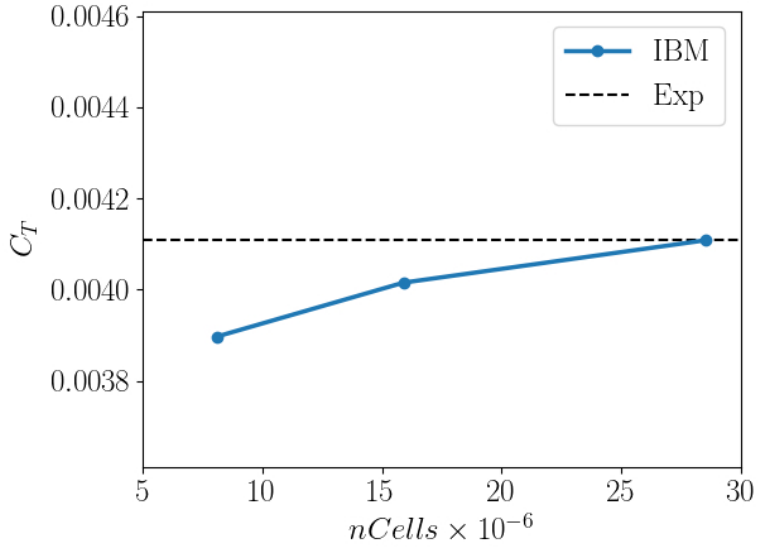
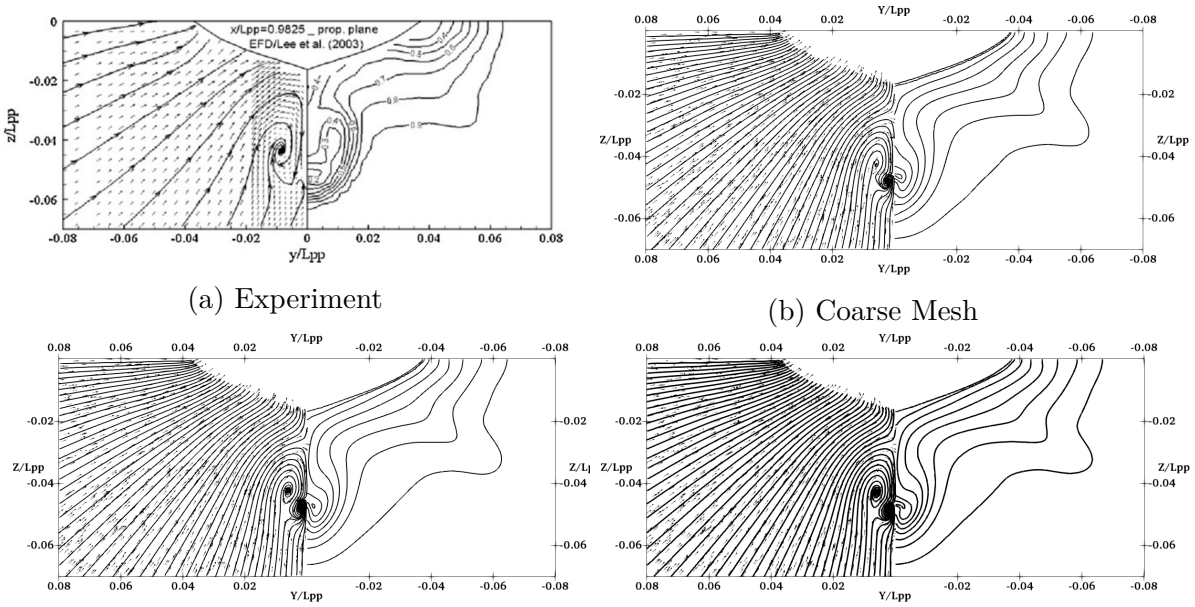


Figure 3.11: total resistance coefficient C_T as a function of number of cells

Finally, the numerically predicted wake flow on the propeller plane is shown in Fig. 3.12, together with the experimental data. The main difference is in the region $0 \leq Y/L_{pp} \leq 0.01$ and $-0.05 \leq Z/L_{pp} \leq -0.03$. In this region, the numerically predicted axial velocity component attaches the hull while flow separation is observed in the experiment. This is likely due to the tendency of the Spalart-Allmaras model to underpredict flow separation. Further study on implementing more advanced turbulence models into the present IB solver is required to better predict such flow features.

In addition, once the flow starts to massively separate due to the large curvature near the stern, the underlying assumption of the IBM that the flow is attached does not hold true, which causes the discrepancy compared with the experimental data. In general, the present IBM can not only predict the integrated quantities such as the total resistance with good accuracy, but also capture the major local flow patterns, which is of great importance for propeller design.



(a) Experiment

(b) Coarse Mesh

(c) Medium Mesh

(d) Fine Mesh

Figure 3.12: Comparison of the wake field predicted by different meshes on the propeller plane $Y/L_{pp} = 0.9825$

3.6 Summary

In this chapter, the coupling between the present IBM, the RANS solver and the Spalart-Allmaras turbulence model is thoroughly discussed. One major disadvantage of IBMs is the difficulty to control the mesh refinement near IB surfaces. This disadvantage is amplified when solving turbulent flows at high Reynolds numbers. To alleviate the requirement on mesh refinement, a universal wall function is implemented to couple with the current IBM. The wall function provides a near-wall profile

of velocity from the outer edge of the logarithmic region down to the IB surface. The present IBM solver is capable of solving high-Reynolds-number flows for both stationary and moving IB surfaces. The accuracy of the prediction of the force on the IB surfaces, the flow field including flow separation and reattachment, profiles of flow variables are carefully examined through various test cases. The tests include both canonical cases such as the flow over a flat plate and airfoil, inside an asymmetric diffuser, and around a 3D realistic geometry with high curvature such as the ship hull. The results demonstrate that capability of the present IB solver for simulating high-Reynolds-number flows involving both stationary and moving IB surfaces. The numerical details and validation of the proposed IBM for the simulations of both laminar and turbulent flows have also submitted to Part M: Journal of Engineering for the Maritime Environment (manuscript ID: JEME-20-0054, in review).

CHAPTER IV

Development for Air-Water Two-Phase Flows

In this chapter, the implementation of the IBM for air-water two-phase flows is developed. Specifically, the implementation considers the scenarios including when an IB penetrates the air-water interface and when an IB moves relative to a body-fitted boundary in the presence of the air-water interface. The VoF method is used to model the air-water interface. The IB boundary condition of velocity and the IB wall function for air-water two-phase flows is exactly the same as for single-phase flows. A Neumann boundary condition is applied for the dynamic pressure p_{rgh} in the same manner as for the total pressure p in single-phase flows. In addition to the IB boundary conditions of the velocity and the dynamic pressure, IB boundary condition of the fluid volume fraction introduced by the VoF method is applied to impose correct density gradient on IBs. To validate the capability of the solver, various numerical tests including waves in tanks with different shapes, the dam-break problem, and water exit of a circular cylinder are carried out.

In recent years, an increasing number of researches is conducted on the implementation of IBMs for two-phase flows.

Sanders et al. (2011) presents a level-set two-phase flow solver employing a finite difference IBM. The solver is validated by solving the decay of the heave motion of a buoyant cylinder and the roll motion of a box in 2D. It seems that the method has

not been extended to 3D or complex geometries.

Shen and Chan (2008b) propose a methodology that combines a discrete-forcing IBM and the VoF method to simulate flow interactions between free-surface waves and submerged solid bodies in 2D. Good agreement of the free-surface profile is presented. However, no additional boundary condition of the volume fraction on the IB needs to be considered since the solid bodies are fully submerged.

Yang and Stern (2009) present a combined sharp interface IB/level-set Cartesian grid method for the LES simulations of 3D free-surface flows. The contact angle boundary condition on the IB is implemented to enforce the boundary condition of the level-set function ϕ, which requires the linear interpolation of ϕ in the vicinity of the IB. A series of simulations, including flows around simple geometries and ship hulls, are performed. The capability of the solver is demonstrated through the comparison of the flow field and the free-surface profile.

Sun and Sakai (2016) present a numerical model that combines an IBM and the VoF method for simulating the two-phase flow in a twin screw kneader, which has two counter-rotating screw elements. The free-surface boundary condition of the volume fraction α near the IB is enforced by either a simple dilation method or by solving an additional "extension" equation introduced by (*Sussman* (2001)).

Calderer et al. (2014b) proposes a new level-set IBM for solving 3D fluid-structure interaction problems. The spatially-filtered N-S equations are used as governing equations, and they are solved using the fractional step method on curvilinear grids. The boundary conditions on the IBs are enforced through interpolation and enforcement of the velocity in the cells in the vicinity of the IBs. A Neumann boundary condition is applied to the level set function ϕ at the IB. A two-step approach is proposed to compute the force and moment on the IB by projecting the pressure and the viscous stress to a set of grid faces that encloses the IB. The accuracy of the solver is demonstrated by a series of test cases including water entry and exit of a horizontal

circular cylinder, free roll decay of different floating geometries, and wedge impact on the free-surface.

To date the combined usage of IBM and multi-phase flow solvers especially for ship hydrodynamic flows is not explored as much as the application of IBMs for single-phase flows. Most of the methods are focused on using Cartesian grids, and require the interpolation for either the level-set function or the volume fraction in a similar way for the enforcement of the velocity boundary condition on the IBs. In addition, the benefit of the combined usage of unstructured body-fitted meshes and IBMs has not been explored.

In this chapter, the implementation of the IBM in an air-water two-phase flow solver is introduced. The VoF method is used to track the air-water interface. The dilation method in *Sun and Sakai* (2016) is adopted to deal with the intersection between the air-water interface and IBs. The goal of the implementation is to develop a solver that is suitable and efficient for ship hydrodynamic applications with minimal modification from the IB single-phase solver. Various numerical results are presented to demonstrate the capability of the new method to accurately predict the force, flow field and the interaction between the IBs and air-water interface.

4.1 Governing Equations

Two-phase incompressible turbulent flows can be described by the RANS equations:

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

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = \nabla \cdot \left[\rho \nu_{\text{eff}} \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T \right) \right] - \nabla p + \rho \mathbf{g} \tag{4.2}$$

which are similar to the governing equations in Chap. 2 with the additional source term $\rho \mathbf{g}$ in the momentum equation due to gravity. The local density ρ and effective

viscosity ν_{eff} depend on the local distribution of the fluid phases. In addition, the surface tension force is neglected.

To reduce the complexity of enforcing the pressure boundary conditions, the dynamic pressure p_{rgh} is used instead of the total pressure p. The dynamic pressure is expressed as:

$$p_{\text{rgh}} = p - \rho \mathbf{g} \cdot \mathbf{x} \tag{4.3}$$

Substitution of Eqn. 4.3 into Eqn. 4.2 yields the final form of the momentum equation:

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = \nabla \cdot \left[\rho \nu_{\text{eff}} \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)\right] - \nabla p_{\text{rgh}} - \mathbf{g} \cdot \mathbf{x} \nabla \rho \tag{4.4}$$

The VoF method is applied to solve air-water two-phase flows by introducing a transport equation for the fluid volume fraction α as shown in Eqn. 4.5. The method numerically represents the interface as a thin layer instead of a sharp boundary.

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\mathbf{u}\alpha) + \nabla \cdot [\mathbf{u}_r (1 - \alpha)\alpha] = 0 \tag{4.5}$$

where, the last term is an artificial compression term that is introduced to confine the smearing of α. Different fluid phases are indicated by different values of α as shown in Eqn. 4.6.

$$\begin{cases} \alpha = 0, & \text{air} \\ \alpha = 1, & \text{water} \\ 0 < \alpha < 1, & \text{near the interface} \end{cases} \tag{4.6}$$

To compute the compression term in Eqn. 4.5, $\mathbf{u}_r$ at face centers $\mathbf{u}_{\text{r,f}}$ are calculated by:

$$\mathbf{u}_{\text{r,f}} = \mathbf{n}_f \min \left\{ C_\alpha \frac{|\phi|}{|\mathbf{S}_f|}, \max \left(\frac{|\phi|}{|\mathbf{S}_f|}, 1 \right) \right\} \tag{4.7}$$

in which $\mathbf{S}_f$ is the face area vector and its direction is the face normal direction. ϕ is the velocity flux, C_α is the compression coefficient, and $\mathbf{n}_f$ is the interface normal vector.

Larger values of C_α permit greater compression of the smeared layer at the interface, but it can result in the decreasing stability of calculating the surface curvature at the same time. For all the simulations in this thesis, C_α is set to be 1.5 to balance between the stability and accuracy.

The interface normal vector $\mathbf{n}_f$ in Eqn. 4.7 is calculated by:

$$\mathbf{n}_f = \frac{(\nabla\alpha)_f}{|(\nabla\alpha)_f + \delta|} \tag{4.8}$$

where δ is a stabilizing factor computed as:

$$\delta = \frac{1 \times 10^{-8}}{\left(\frac{\sum V_i}{N}\right)^{1/3}} \tag{4.9}$$

in which, V_i is the cell volume and N is the total number of computational cells.

The boundary condition of α at the intersection of the interface and the solid wall is provided through the contact-angle boundary condition:

$$\mathbf{n}_f \cdot \mathbf{n}_B = \cos\theta \tag{4.10}$$

in which $\mathbf{n}_B$ is the unit normal vector pointing from the fluid phase to solid phase. θ is the contact angle.

For all the simulations in this thesis, a neutral contact angle $\theta = \pi/2$ is assumed. Substitution of Eqn. 4.8 into Eqn. 4.10 yields a simplified wall boundary condition of α:

$$\mathbf{n}_f \cdot \mathbf{n}_B = \frac{(\nabla\alpha)_f}{|(\nabla\alpha)_f + \delta|}\mathbf{n}_B = 0 \qquad \nabla_n\alpha = 0 \tag{4.11}$$

which is a Neumann boundary condition of α on the solid wall.

Subsequently, the local density and viscosity in the momentum equation Eqn. 4.4 is calculated as:

$$\rho = \alpha\rho_w + (1-\alpha)\rho_a \quad \text{and} \quad \nu = \alpha\nu_w + (1-\alpha)\nu_a \tag{4.12}$$

where the subscript w and a represent water and air, respectively.

The entire solution process of the air-water flow solver is summarized as follows:

1. At the beginning of each time step or the beginning of each PISO iteration, the VoF equation Eqn. 4.5 is solved in the first place.

2. Density and viscosity are updated based on the local volume fraction α.

3. Velocity and pressure are solved using the PISO algorithm introduced in the previous chapter.

4. Start the next PISO iteration or proceed to the next time step.

4.2 IB Treatment within the Air-Water Flow Solver

The IB version of the air-water two-phase flow solver is modified based on the OpenFOAM two-phase flow solver interFoam. On IBs, the boundary condition of velocity and the wall function for turbulent flows are implemented in the same way as for single-phase flows. The Neumann boundary condition of p_{rgh} is imposed in the same way as for the total pressure p in single-phase flows. It should be noted that the dynamic pressure p_{rgh} has a large gradient across the air-water interface, which brings more challenge to the simulations of air-water flows. In addition, the Neumann boundary condition of the volume fraction α is imposed on IBs.

In the VoF equation Eqn. 4.5, the velocity field satisfies the impermeable and no-slip boundary conditions on solid walls. Since there is no flux across the solid wall,

the convection term in Eqn. 4.5 is zero regardless of the wall boundary condition of α. It may look unnecessary to enforce the Neumann boundary condition of α on the IB. However, to impose the Neumann boundary condition is indeed important if examining the gradient of density term in Eqn. 4.4. The term arises because the dynamic pressure p_{rgh} is used instead of the total pressure p. For body-fitted meshes, the Neumann boundary condition of α results in the zero-gradient boundary condition of density on the solid wall. However, the behavior of α inside the IBs is undefined. Considering the large density ratio between water and air, the modification of the α field in the vicinity of the IBs has a great influence on the evaluation of the density gradient, subsequently the overall accuracy.

In this thesis, the dilation method proposed in *Sun and Sakai* (2016) is adopted. The method is described as follows:

1. Mark all solid cells and store in a list $\mathcal{L}$.

2. Loop through all marked cells in $\mathcal{L}$.

   (a) Check all the cells that share nodes with the marked cell i, and store the indices of all unmarked cells. The volume fraction α of these cells is used to interpolate the α in cell i.

   (b) After storing all the unmarked cells, use the inverse square distance method to interpolate α_i as

   $$\alpha_i = \frac{\sum_j w_j \alpha_j}{\sum_j w_j} \tag{4.13}$$

   where

   $$w_j = \frac{1}{d^2} \tag{4.14}$$

   where d is the distance between the cell centers of i and j. j is the cell index of each unmarked cell.

   (c) Store the cell index i into a temporary list $\mathcal{S}$.

(d) Go to next marked cell $i+1$.

3. Remove all cells in $\mathcal{S}$ from $\mathcal{L}$.

4. Go back to the beginning of Step. 2.

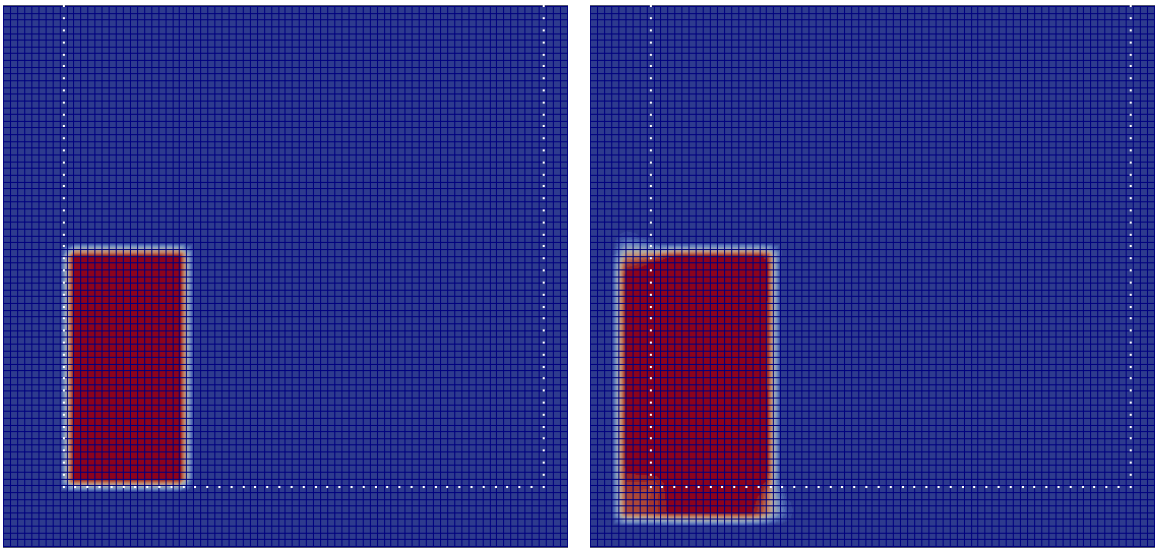
The number of iterations of Step. 2 determines how many layers of solid cells are extended into the IB. After the extension, the volume fraction is extend naturally into the solid domain, which provides a good approximation of the Neumann boundary condition.

It should be noted that this process of extension changes the fluid volume in the solid domain solely to enforce the Neumann boundary condition of α on the IB. Therefore, it changes the total fluid volume in the whole domain. However, since the velocity boundary condition is imposed on the IB. Change of the fluid volume in the solid domain does not affect the total volume in the fluid domain, where the flow is of interest. In all the simulations in the current work, no notable issue is observed.

To demonstrate the importance of imposing the Neumann boundary condition of α on IBs, the dam-break problem is examined. In this problem, a volume of water rests at the corner of a tank. After it is released, the dam breaks and the water starts to surge along the bottom floor of the tank. The left figure in Fig. 4.1 shows the initial setup of the dam-break problem, and the water is colored in red. The white dots represent the walls of the tank which are modeled by an IB. An orthogonal structured mesh is used as the background mesh.

The right figure in Fig. 4.1 shows the effect of the dilation method with four iterations. Therefore the α field extends to the first four layers of cells in the solid region.

Fig. 4.2 shows the comparison of the shape of the air-water interface between the present IBM and the results on the body-fitted mesh with the same mesh resolution after the dam is released. The shape of the interface and the total volume of water,



(a) Original α field (b) α field after the extension

Figure 4.1: Extension of α into the solid region

is not correctly predicted without enforcing the boundary condition of α. The result demonstrates the importance of enforcing the α boundary condition in addition to the enforcement of the velocity boundary condition.

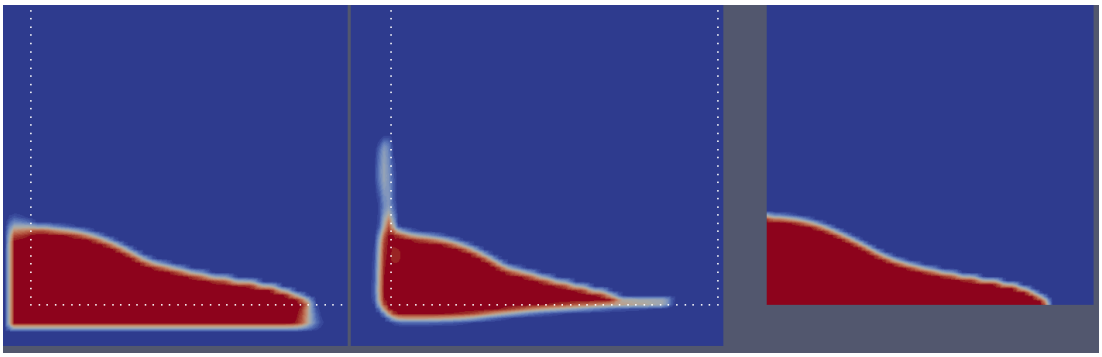
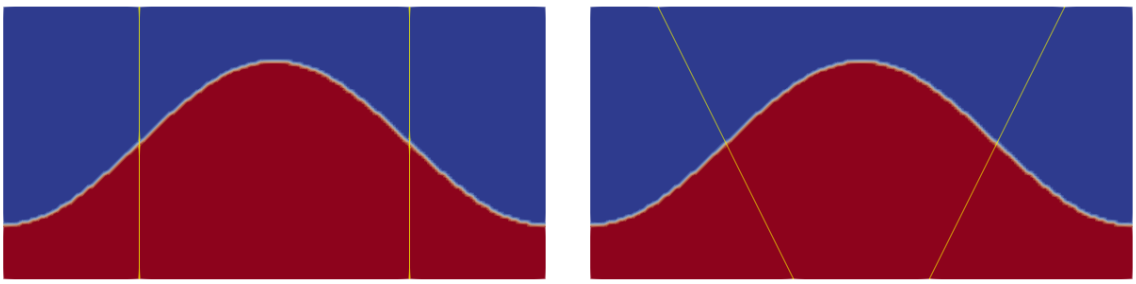


Figure 4.2: Comparison of the shape of the free surface. Left: Enforcement of the α BC; middle: no enforcement of the α BC; right: body-fitted mesh

In the following sections, the accuracy of the present IBM is examined through various test cases in both 2D and 3D, for both stationary and moving IBs.

4.3 Waves in Tanks with Different Shapes

In this section, the wave motion in tanks with two different shapes is considered. The volume of water is first initialized based on a sine function. Waves start to reflect between the side walls of the tank after the water is released. Further due to the reflection at the two side tank walls, the air-water interface oscillates similar to a standing wave. The IB is used to represent the side walls of the tank, and the no-slip boundary condition is applied. Fig. 4.3 shows the entire computational domain and the initial profile of the water. The IBs, which represent the tank walls, are shown as yellow solid lines.



(a) Rectangle (b) Inverted trapezoid

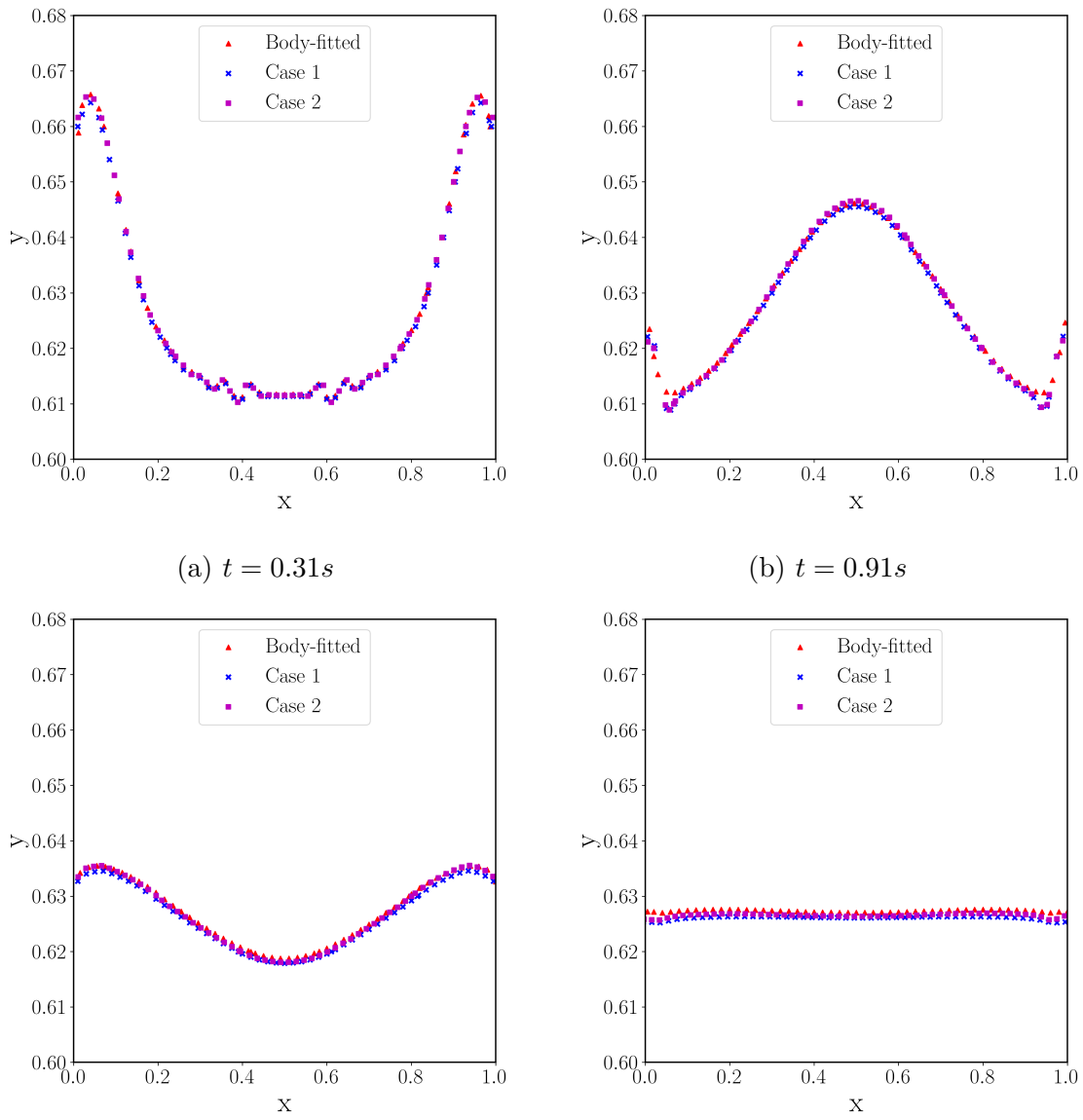
Figure 4.3: Setup of the waves in the tanks with different shapes

A simulation using structured body-fitted meshes with similar cell spacing is also carried out as a reference. As for the rectangular tank, two different layouts of the IBs are considered as shown in Table 4.1.

Table 4.1: Different layouts of the IB for representing a rectangular tank

Case	Grid lines coincide with the IB
Case 1	Y
Case 2	N

In Case 1, the velocity boundary condition is applied exactly on face centers through the IBM. In this sense, the IBM solver is equivalent to interFoam which uses body-fitted meshes. The only difference is the fluid volume fraction α is extended into



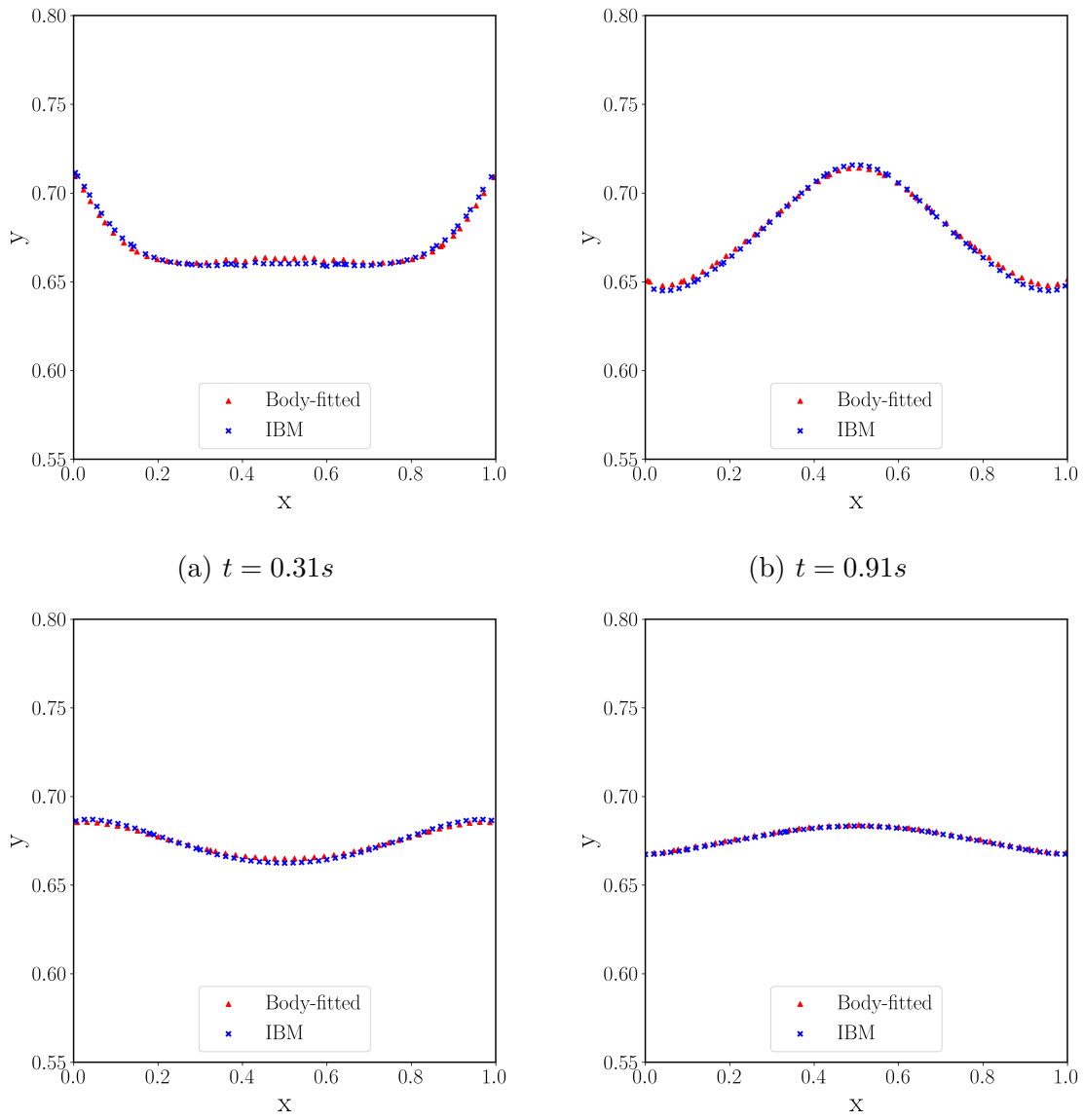
(a) $t = 0.31s$ (b) $t = 0.91s$

(c) $t = 1.31s$ (d) $t = 1.91s$

Figure 4.4: Air-water interface profiles in the rectangular tank at different time instants

the solid phase to accurately evaluate the density gradient on the IB. In comparison, Case 2 is more representative because in general cases IBs do not coincide with grid lines, especially when IBs have curvature. Fig. 4.4 shows the profile of the air-water interface at different time instants in the rectangular tank.

Fig. 4.4 shows that the profile of the air-water interface predicted by the IBM



(a) $t = 0.31s$ (b) $t = 0.91s$

(c) $t = 1.31s$ (d) $t = 1.91s$

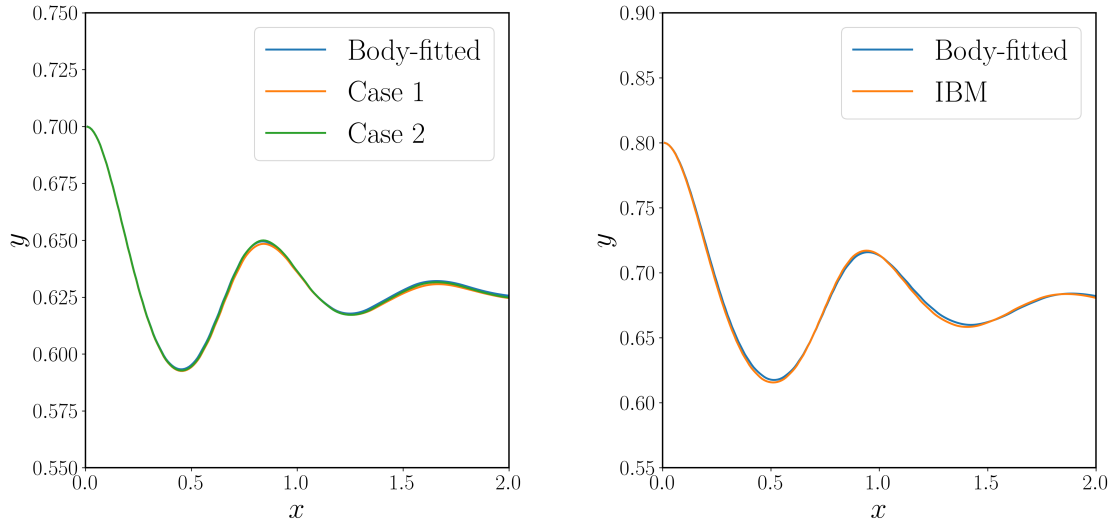
Figure 4.5: Air-water interface profiles in the inverted trapezoid tank at different time instants

matches very well with the body-fitted results regardless of whether the IBs coincide with the grid lines. The results demonstrate that the velocity boundary condition on the wall is well imposed through interpolation, and the density gradient is correctly calculated through the extension of the α field.

Fig. 4.5 shows the profile of the air-water interface at different time instants in

the inverted trapezoid tank. In this case, the position of the IBs relative to the grid lines represents the more general case where velocity needs to be interpolated in the vicinity of the IBs, and the α field needs to be extended to satisfy the Neumann boundary condition on the IBs. It can be seen that the air-water interface predicted by the IBM agrees well with the body-fitted results.

The time histories of the air-water interface at the centerline $x = 0.5$ of both tanks are shown in Fig. 4.6. The location of interface oscillates harmonically with the amplitude decreasing in time. The numerical results show good agreement with the reference results using the body-fitted meshes for both shapes of the tanks.



(a) Rectangular tank (b) Trapezoidal tank

Figure 4.6: Time histories of the air-water interface at the centerline $x = 0.5$

As a summary, the comparison of the profiles of the air-water interface in tanks with different shapes of side walls is carried out. The results demonstrate that the present IBM can interpolate the velocity to satisfy its boundary condition on the solid wall. The extension method of the α field can correctly enforce the Neumann boundary condition on the IBs. It results in a correct prediction of the density gradient near the IBs, which is crucial for the overall accuracy of the two-phase flow simulations.

4.4 3D Dam-Break with an Obstacle

In this section, the classic dam-break problems with an obstacle are investigated. Two different setups from different sources of experiments are considered such that the capability of the solver can be examined from various aspects, including the flow velocity, the impact force on the obstacle, and the local pressure.

4.4.1 Dam-Break No.1

The first test case discusses the interaction between a vertical square cylinder and a single large wave caused by the dam break in a rectangular tank. The experimental data is found in *Raad and Bidoae* (2005) provided by Profs. Catherine Petroff and Harry Yeh. Numerical simulations are also carried out by *Raad and Bidoae* (2005) using their three-dimensional Eulerian-Lagrangian marker and micro-cell method.

Fig. 4.7 illustrates the setup of the numerical simulation. The dimensions of the tank are 1.6 m×0.61 m×0.75 m. A volume of water with the size 0.4 m×0.61 m×0.3 m is initially placed at the left end of the tank. The dimensions of the vertical obstacle are 0.12 m × 0.12 m × 0.75 m. It is placed downstream of the volume of water with center of the bottom of the obstacle at $(0.96, 0, 0)$. As reported in *Raad and Bidoae* (2005), the bottom of the tank was not completely drained in the physical experiment. Therefore, a thin layer of water with 0.01 m in depth is setup in the present simulation as shown in Fig. 4.7.

It should be noted the way that the water is released in the present simulation is different from how the experiment was conducted. In the physical experiment, the water is blocked by a gate. The gate is removed vertically with finite speed at the beginning of the test. In comparison, the water is released by the gate with infinite opening speed in the present simulation. *Lin and Chen* (2013) discuss the influence of the opening speed of the gate on the time history of the impact force on the obstacle. Their results show that the peak of the impact force is delayed as the finite opening

speed of the gate decreases.

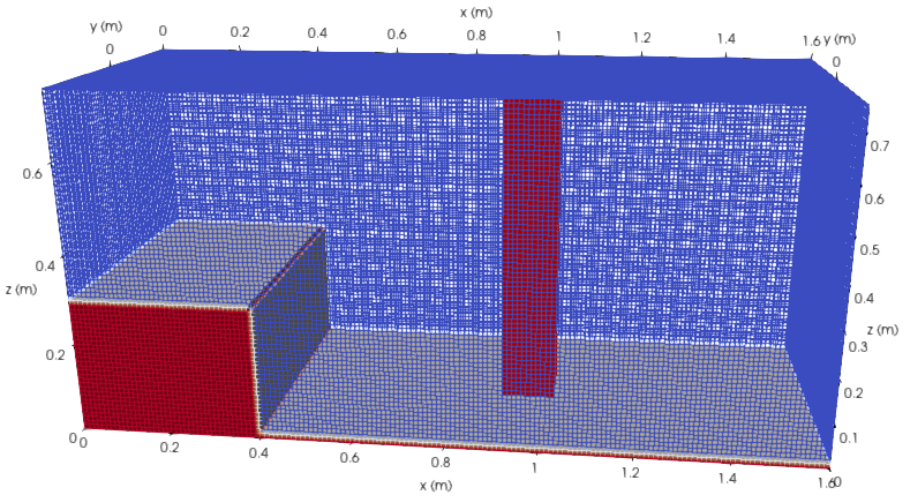
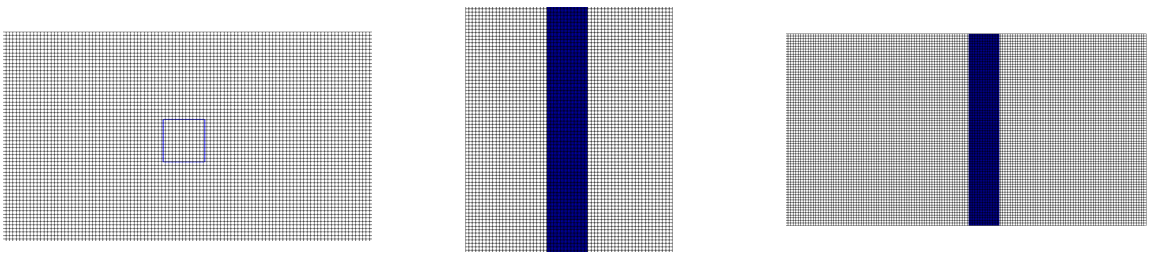


Figure 4.7: Case setup of Dam-Break No.1

In the present simulations, the walls of the tank are represented by the body-fitted wall boundaries as shown in Fig. 4.7. The solid walls of the vertical obstacle are represented by the IB. A set of three background meshes with a refinement ratio of $\sqrt{2}$ in each direction is used to validate the solver. The total numbers of cells of the background meshes are $114 \times 53 \times 43$, $161 \times 75 \times 61$ and $228 \times 106 \times 86$, respectively. The relative position between the IB and the medium background mesh is shown in Fig. 4.8.



(a) top view (b) front view (c) side view

Figure 4.8: Relative position between the IB and the finest background mesh

The density and viscosity are $\rho_w = 1000$ kg/m^3, $\nu_w = 1 \times 10^{-6}$ m^2/s for the water, and $\rho_a = 1$ kg/m^3, $\nu_a = 1 \times 10^{-5}$ m^2/s for the air. The gravitational acceleration

is $g = 9.81$ m/s^2. In the simulations, the time step size is adjusted automatically to keep the global Courant number less than 1.0 and the Courant number near the air-water interface less than 0.3.

A probe is used to record the flow velocity before the water reaches the obstacle. The probe is located in front of the obstacle at $(0.754, 0, 0.026)$. In addition, the impact force on the obstacle is calculated.

Fig 4.9 shows the wave propagation and its interaction with the obstacle at different time instants. It provides a general idea of what the critical phases of the dam-break problem look like. At $t = 0$ s, the water is released. After the water hits the front side of the obstacle, it runs up along the front wall and causes a large impact force. Afterwards, the water that travels around the obstacle joins together behind the obstacle, travels to the end of the tank, and hits the back side of the obstacle after being reflected by the end wall of the tank. It causes a second impact in the opposite direction compared to the first peak of impact. In addition, the second impact force is expected to be weaker than the first one because the velocity of the front of the wave is decreasing in general.

To further evaluate the accuracy of the solver, Fig. 4.10 shows the x-component of the velocity at the velocity probe using all three background meshes. The data is shifted in time such that $t = 0$ s in the figure is the moment when the water first reaches the probe. Specifically, $t = 0$ s in the figure corresponds to 0.238 s after the water is released. The experimental data is plotted together for the purpose of comparison. The gaps in the experimental data at $0.6 < t < 0.85$ s and $t > 1.5$ s are due to the presence of bubbles in the water as explained in *Raad and Bidoae* (2005). The velocity at $t = 0$ s is slightly overpredicted by the medium and fine meshes, which means the water moves faster in present simulations than in the experiment. It should be noted that in the simulations, the floor is set to be covered by a thin layer of water of thickness 0.01 m. The layer of water is used to mimic the wet floor in the

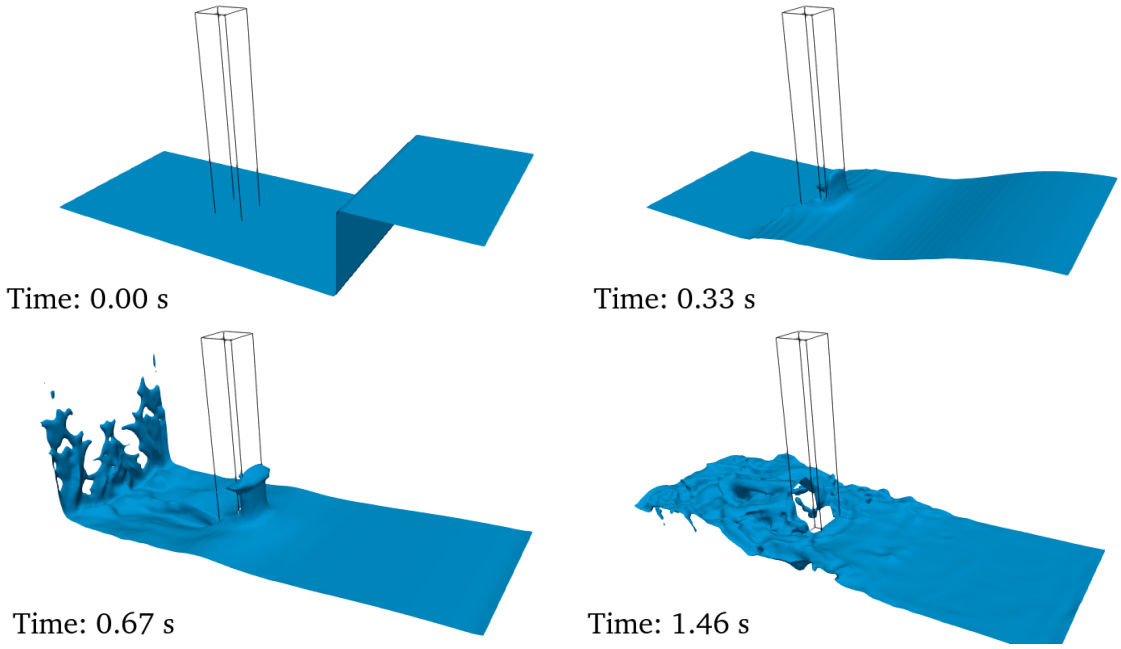


Figure 4.9: Simulation of the 3D dam-break problem with a vertical square obstacle

experiment. However it cannot perfectly reproduce the experimental environment. Another reason is that the gate in front of the water is released with finite speed in the experiment, which reduces the velocity at the water front near the floor. A similar conclusion is drawn in *Lin and Chen* (2013) by investigating the influence of the releasing speed of the gate. The medium and fine meshes well predict the decrease in the velocity of the water (e.g. $0 < t < 1.5$ s) due to the blockage of the obstacle. After $t = 1.5$ s, the wave is reflected from the tank wall at $x = 1.6$ m, and it is further blocked by the back side of the obstacle. The water near the bottom floor in front of the obstacle is almost stationary. This can be confirmed from Fig. 4.10 that the velocity at the probe drops to almost zero after $t = 1.5$ s.

Fig. 4.11 shows the impact force on the obstacle. It is worth pointing out that $t = 0$ s in this figure corresponding to the time when the simulation starts, which explains the zero impact force at about $t < 0.25$ s. The different background meshes predict consistent starting time of the impact. Compared with the experimental data, it can be seen that the first impact happens early than in the experiment which is

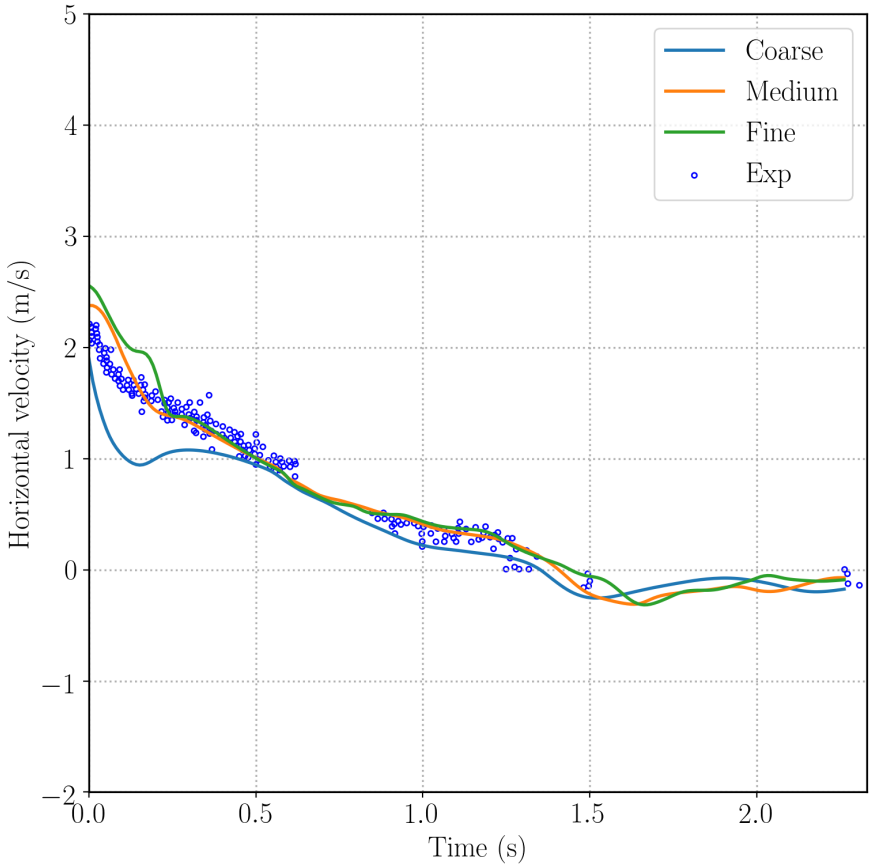


Figure 4.10: Time history of the horizontal velocity in front of the obstacle at $(0.754, 0, 0.026)$ using different background meshes

consistent to the behavior of the horizontal velocity at the velocity probe discussed before. Fig. 4.11 shows that the numerical results slightly underpredict the positive peak value at around $t = 0.4$ s. Afterwards, the impact force decreases gradually to zero around $0.4 < t < 1.5$ s, which is consistent with experimental data. At $t \approx 1.5$ s, the wave reflected from the end the tank arrives and impacts on the back side of the obstacle causing a negative peak of the impact force. During the last phase of the dam break, the impact force decreases to zero again.

In summary, the accuracy of the solver is well demonstrated via the comparisons between the numerical solutions and the experimental data for the impact force and the horizontal velocity in the front of the obstacle.

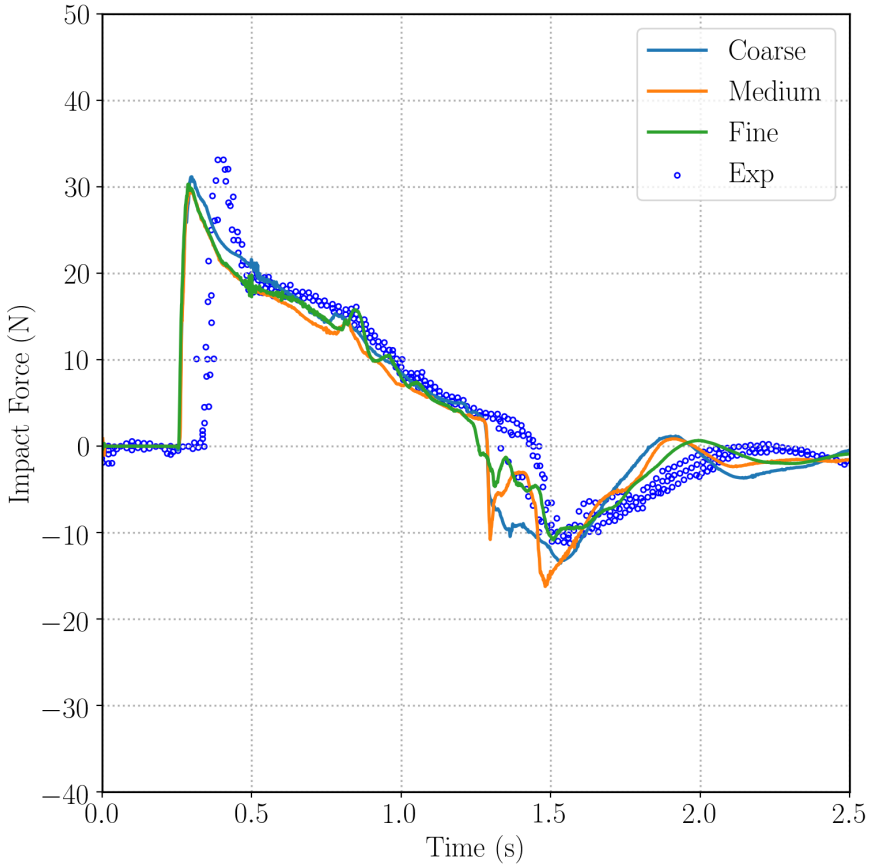


Figure 4.11: Time history of the impact force on the obstacle using different background meshes

4.4.2 Dam-Break No.2

In the previous section, the discussion is focused on the velocity of the water and the total impact force, which is an integrated variable. It is equally important to investigate the local pressure during the impact, as well as the water elevation at different places. To fulfill this goal, a different setup of the 3D dam-break problem with an obstacle is used in this section. The height of the obstacle is much smaller than in the previous case, which means the water also flows over the obstacle from its top. The physical experiment was carried out by the Maritime Research Institute Netherlands (MARIN, *Kleefsman* (2005)) to investigate the phenomenon of green water on the deck of a ship. Local pressure at different positions of the obstacle, and

the water elevation at different locations of the tank were recorded in the experiment. The results of numerical simulations are also provided in *Kleefsman* (2005).

Fig. 4.13 shows the computational domain and the numerical setup. The dimensions of the tank are 3.22 m $\times$ 1 m $\times$ 1 m. At the beginning of the simulation, the volume of water with the size 1.22 m $\times$ 1 m $\times$ 0.55 m is located at the end of the tank from $x = 2$ m. A rectangular obstacle is placed in front of the dam to represent a container on the deck of the ship. The dimensions of the obstacle are 0.16 m $\times$ 0.4 m $\times$ 0.16 m with its front side (the side which faces the dam) positioned at $x = 0.83$ m. The water elevation is monitored at two places along the vertical plane $y = 0$ m, which are $x = 1.0$ m and 2.66 m. Eight sensors are used to record the local pressure on the top and front sides of the obstacle, and the locations are listed in Table 4.2. The numerical results of the time histories at the pressure sensors P1, P3, P5 and P7 are compared with the experimental data. The locations of these four sensors are illustrated in Fig. 4.12. The sensors P1 and P3 are on the side facing to the initial volume of water. The blue solid line represents the plane $y = 0$ m as a reference.

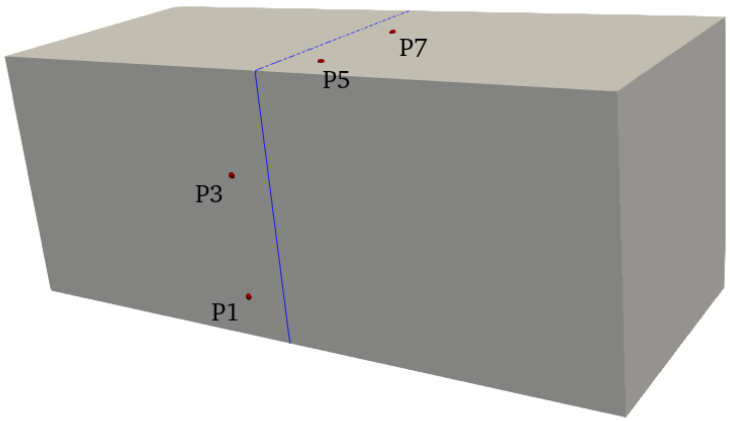


Figure 4.12: Locations of the pressure sensors P1, P3, P5 and P7

The density and viscosity of the water are $\rho_w = 998.2$ kg/m^3, $\nu_w = 1 \times 10^{-6}$ m^2/s, and $\rho_a = 1$ kg/m^3, $\nu_a = 1.48 \times 10^{-5}$ m^2/s for the air. The gravitational acceleration is $g = 9.81$ m/s^2. Similar to the previous section, a set of three background meshes with a refinement ratio of $\sqrt{2}$ in each direction is used. The number of cells of the background meshes are $64 \times 32 \times 32$, $90 \times 46 \times 46$ and $128 \times 64 \times 64$, respectively. All the sides of the tank are represented by the body-fitted boundary conditions as no-slip walls, and the obstacle is modeled with an IB. The time step size is adjusted automatically to keep the global Courant number less than 0.75 and the Courant number near the free surface less than 0.3.

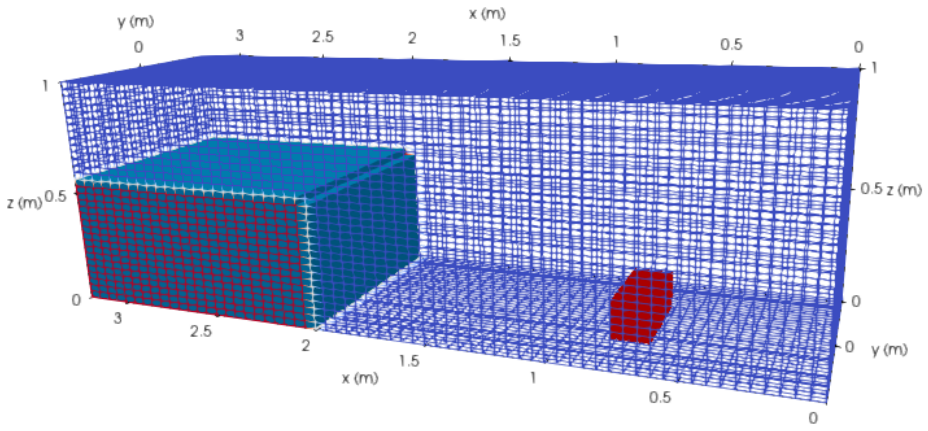
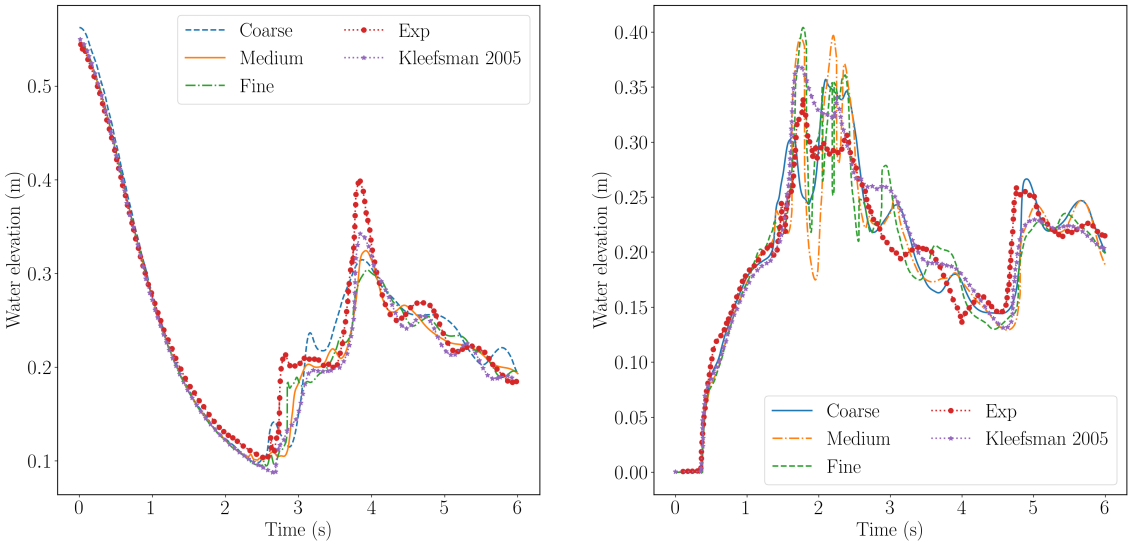


Figure 4.13: Case setup of the 3D dam-break problem with an obstacle: Problem No.2

Table 4.2: Locations of the pressure sensors in the 3D dam-break problem No.2

Sensor	Location
P1	$(0.83, -0.026, 0.025)$
P2	$(0.83, -0.026, 0.063)$
P3	$(0.83, -0.026, 0.099)$
P4	$(0.83, -0.026, 0.136)$
P5	$(0.806, 0.026, 0.16)$
P6	$(0.769, 0.026, 0.16)$
P7	$(0.733, 0.026, 0.16)$
P8	$(0.696, 0.026, 0.16)$

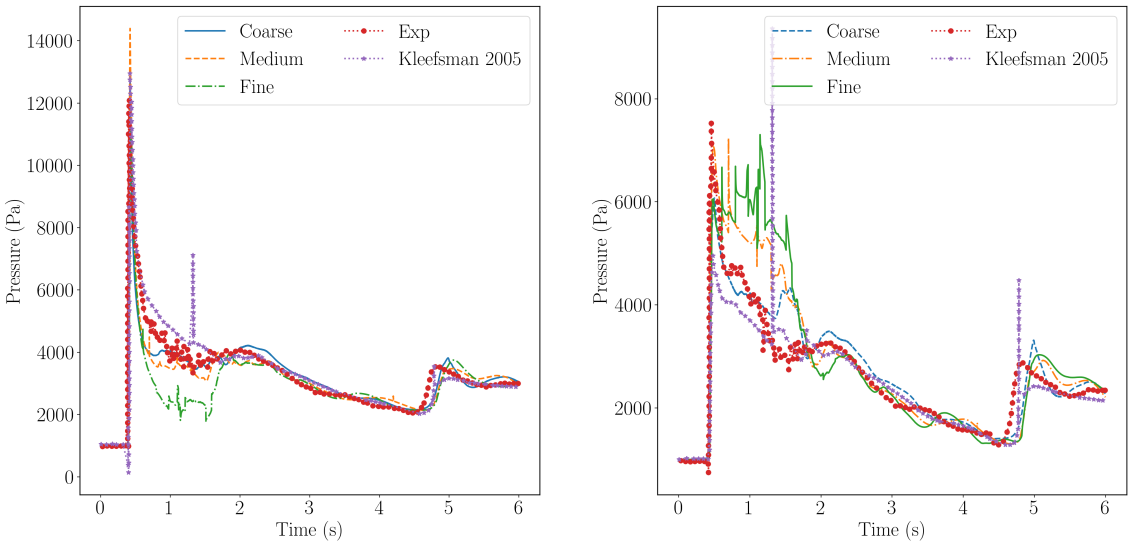
Fig. 4.14 shows the time histories of the water elevation at the locations $x = 2.66$ m and 1.0 m, which are inside the initial position of the dam and in front of the obstacle, respectively. At about $t = 2.5$ s and 1.8 s, the water reflected from the wall of the tank at $x = 0$ m arrives at the two probes, and it leads to high frequency signals in the experimental data. For the time histories beyond these points, there is a large difference between the numerical results and the experimental data. However, before the water travels back from the wall of the tank, the numerical results agree well with the experiment data. At $x = 1$ m, the front of the water arrives at around $t = 0.5$ s, and the numerical results accurately capture the instant when the air-water interface starts to rise.



(a) $x = 2.66$ m (b) $x = 1$ m

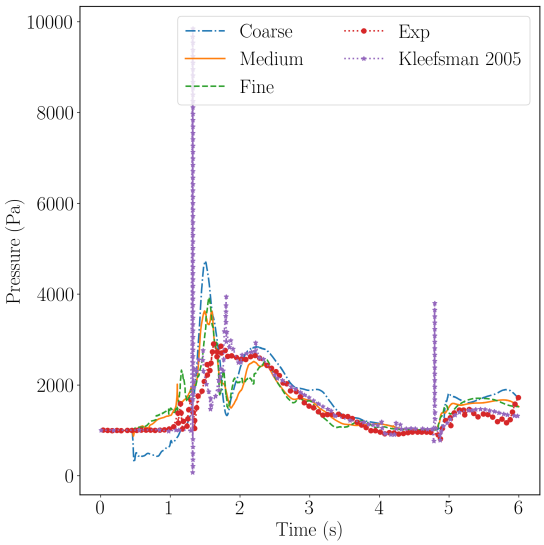
Figure 4.14: Time histories of the water elevation at different locations

As shown in Fig. 4.15, the time histories of the local pressure at four different sensors are selected to compare with the experimental data. The present numerical simulation well captures the instants when the water starts to impact on the sensors, especially for the ones on the front side of obstacle. The peak pressure at P1 matches with the experimental data very well, while the peak pressure at P3 is underpredicted by the fine mesh. At around $2 < t < 4$ s, the pressure at both sensors drops gradually.

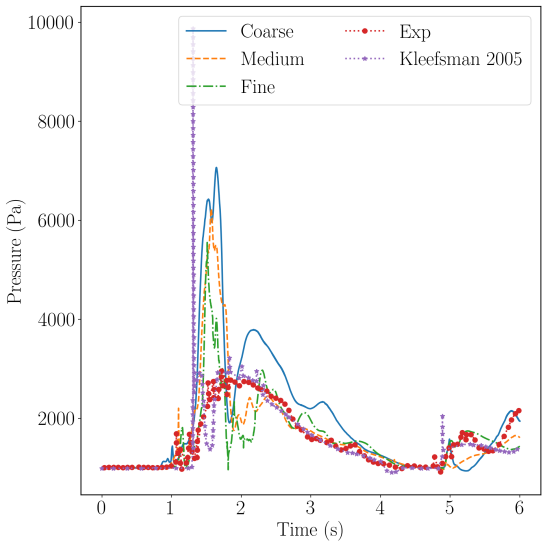


(a) P1

(b) P3



(c) P5



(d) P7

Figure 4.15: Time histories of the pressure at different locations

For the numerical results at P5 and P7, which are on the top surface of the obstacle, the numerical results show a large oscillation. It is because when the water flow over the obstacle, the large vertical velocity of the water causes the water to detach from the surface of the obstacle. Subsequently, air is entrapped when the water starts to impact on the top the obstacle. However, the effect of air compressibility

is not considered in the current solver. Fig. 4.16 shows the profile of the air-water interface around the pressure sensor P7 at around $t = 1.1$ s. It can be clearly seen that the air is entrapped and a bubble is formed around P7. As a result, the large oscillation appears in the numerical results at P5 and P7 in the limitations of the current numerical framework.

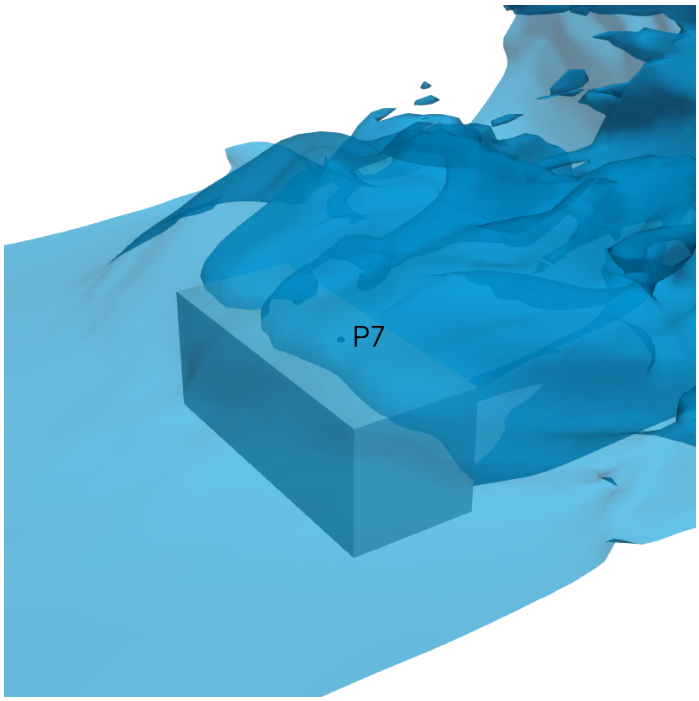


Figure 4.16: Air entrapped on the top of the obstacle in the dam-break problem No.2

Overall, the results demonstrate that the present IBM solver can well handle the problems of wave interaction with solid walls with respect to the evolution of the air-water interface, velocity of the water, force on the obstacle and the local pressure due to the impact.

4.5 Water Exit of a Circular Cylinder

In this section, the IBM is used to simulate a moving surface in the air-water two-phase flow. The test case is the water exit of a 2D circular cylinder with constant vertical velocity. The physical experiments were conducted by *Miao* (1989).

The numerical results of *Zhu et al.* (2007), which are obtained by the Constrained Interpolation Profile (CIP) method, are also presented as a reference.

The computational domain and the initial position of the cylinder with the radius of $R = 0.0625$ m are shown in Fig. 4.17. The domain is 1 m wide and the water depth is $h = 0.5$ m. At the beginning of the simulation, the cylinder accelerates upwards from rest and gradually reaches the final constant velocity v. Eventually, the cylinder exits the water until the water fully dropped from the surface the cylinder. In the present simulations, the cylinder is represented using the IB, and a uniform structured mesh is used as the background mesh with the cell size $\Delta x = \Delta y = 0.04R$.

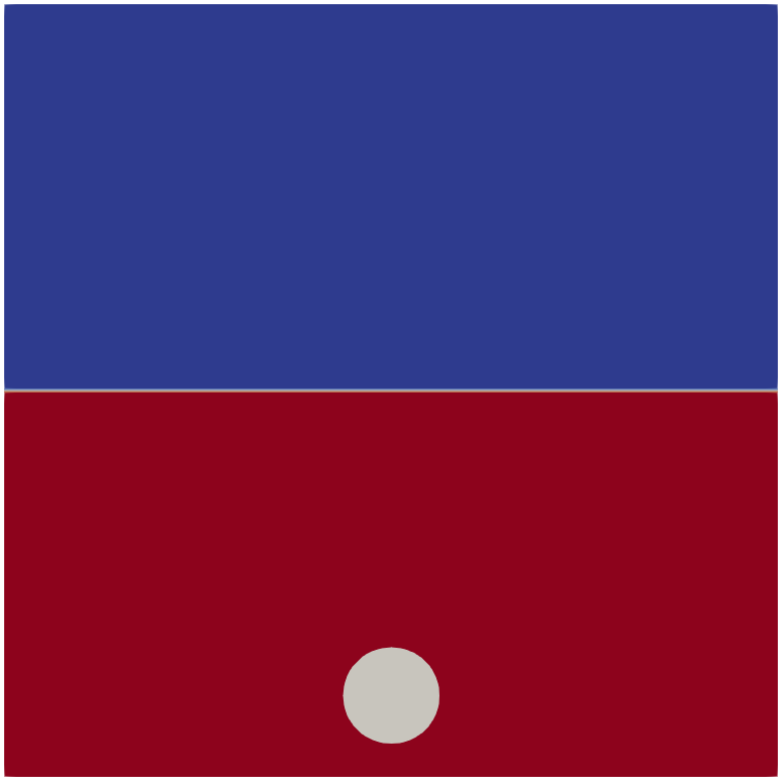


Figure 4.17: Computational domain and the initial condition of the water exit of a circular cylinder

In the experiment, the dimensionless time $vt/R = 0$ corresponds to the time instant when the top of the cylinder touches the air-water interface. $vt/R = -5.5$ and -5 correspond to the time instants where the cylinder starts to move and reaches

the final velocity v, respectively. The prescribed motion of the cylinder is described by the position of the center of the cylinder with respect to time:

$$y(t) = \begin{cases} \frac{1}{2}v(t - \frac{t_0}{\pi}\sin(\frac{\pi t}{t_0})), & t < t_0 \\ v(t - \frac{1}{2}t_0), & t \geq t_0 \end{cases} \tag{4.15}$$

where t_0 is the time of ramping up the velocity. It should be noted that in the simulations, $t = 0$ s corresponds to the time instant when the cylinder starts to move.

Two numerical tests are carried out with different final velocity v as listed in Table 4.3.

Table 4.3: Parameters for the water exit tests

Test	Fr	v(m/s)	t_0(s)
No.1	0.4627	0.5124	0.0610
No.2	0.6903	0.7644	0.0409

The time step size is automatically adjusted to keep the global Courant number less than 1.0 and the Courant number near the free surface less than 0.3.

Fig. 4.18 and Fig. 4.19 show the time histories of the exit coefficient C_e for the two test cases, respectively. The time $t = 0$ of the present simulations is shifted to compare with the experimental data. The exit coefficient C_e is defined as:

$$C_e = \frac{F}{\rho v^2 R} \tag{4.16}$$

where ρ is the density of water and F is the vertical hydrodynamic force on the cylinder.

For both test cases, the present numerical results show good agreement with the experimental data. In general, C_e shows similar trend for both exit speeds. Before the dimensionless time $vt/R < -5$, the cylinder starts to accelerate from its initial position. It results in a large negative peak in force, which is captured by all the

three results for both cases. For the time range $-5 < vt/R < 0$, the cylinder moves upward with a constant speed. As the cylinder approaches the surface of the water, C_e gradually decreases with a nearly constant rate of change. In addition, the vertical speed of the cylinder does not have much influence on the rate of decrease of C_e. When the cylinder starts to exit from the water at $vt/R = 0$, the hydrodynamic force drops significantly, and eventually becomes zero when the cylinder is drained after it completely leaves the water. In summary, for both vertical speeds of the cylinder, the present IBM quantitatively captures the unsteady behavior of the hydrodynamic force at different phases of the two water-exit tests.

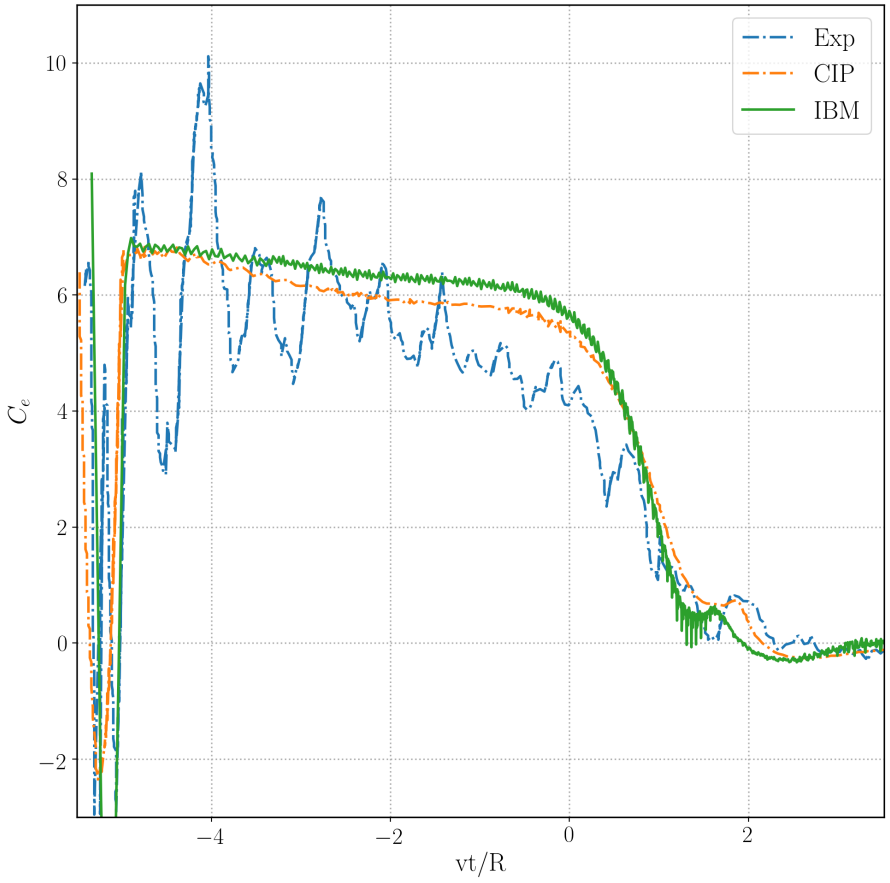


Figure 4.18: Time history of the exit coefficient C_e for the water exit test No.1

Fig. 4.20 shows the comparison of the profile of the air-water interface with the numerical solution of *Zhu et al.* (2007) at different time instants. The present results

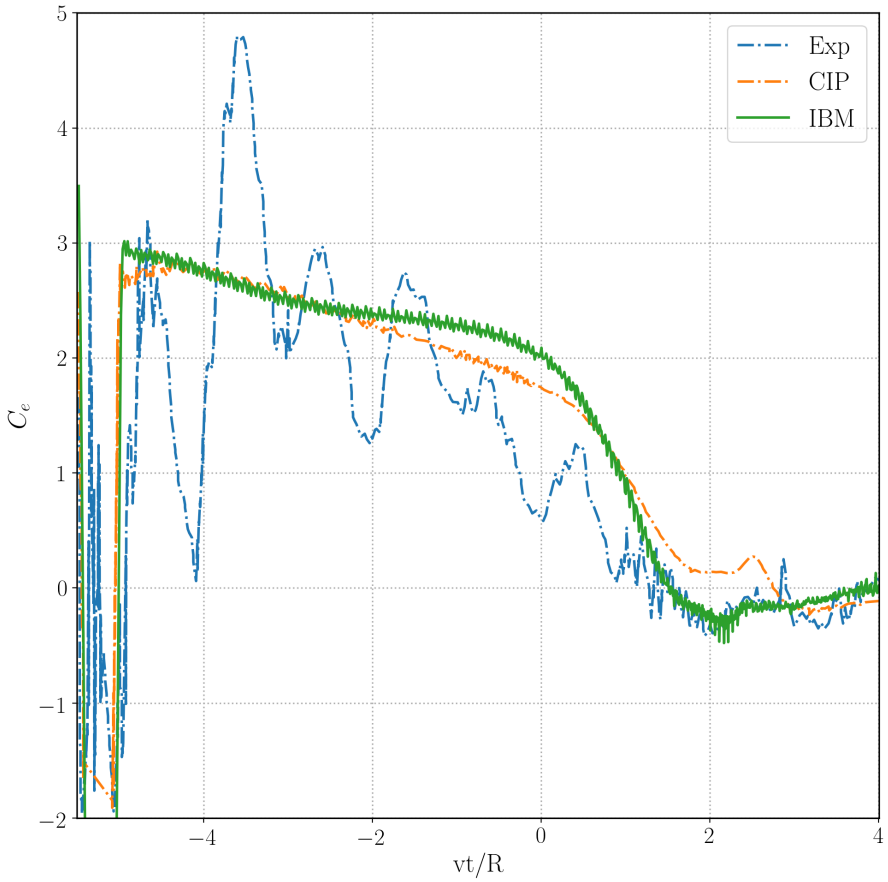


Figure 4.19: Time history of the exit coefficient C_e for the water exit test No.2

show overall good agreement with the other numerical results with respect to the hydrodynamic force on the cylinder with different exit velocity. When the cylinder exits the water after $vt/R = 0$, the present results capture the thin layer of water on top of the cylinder. The present results also well predict the deformation of the air-water interface when the layer of water starts to fall down along the surface of the cylinder at $vt/R > 1$.

4.6 Summary

In this chapter, the implementation of coupling the IBM and the incompressible air-water two-phase solver interFoam is described in detail. The VoF method is

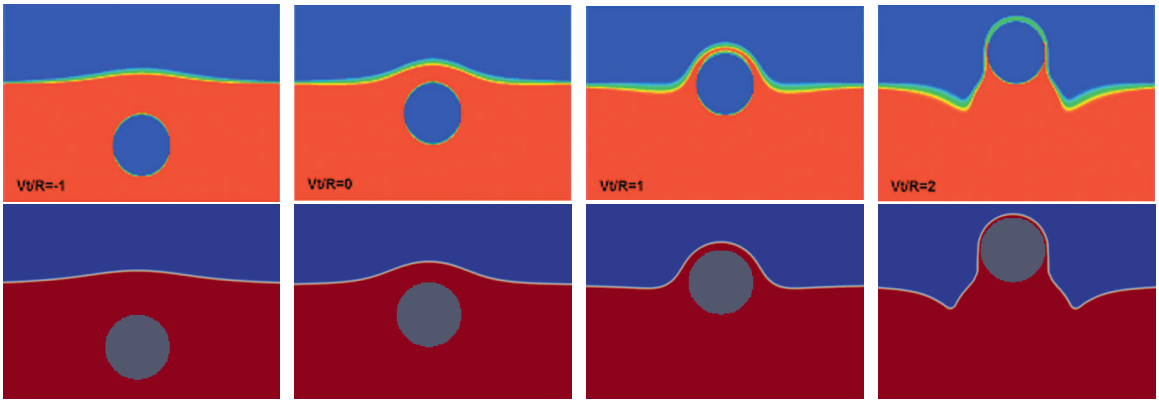


Figure 4.20: Comparison of the profiles of the air-water interface at different time instants in Case No.1. Top row: *Zhu et al.* (2007); bottom row: IBM

used to capture the air-water interface. The VoF method introduces a transport equation for the fluid volume fraction α. A Neumann boundary condition of α is required on the solid wall boundary, which is equivalent to the application of a neutral contact angle $\theta = 90°$ at the intersection between the air-water interface and the wall. To simplify the pressure boundary condition on the solid wall, the total pressure in the momentum equation is replaced by the dynamic pressure, which introduces the density gradient term to the momentum equation. Because the velocity already satisfies the no-penetration boundary condition on the solid boundary via the IBM, it seems that α does not need any special treatment in the vicinity of the IB to have zero flux of α. However, the density field depends on the distribution of α. The calculation of the density gradient is problematic if α is undefined inside the IB, which makes the enforcement of the Neumann boundary condition of α on the IB very important. In addition, due to the large density ratio between air and water, the error in the calculation of the density gradient is amplified by the error of the enforcement of the Neumann boundary condition. The present IBM adopts a dilation method *Sun and Sakai* (2016) which naturally extends α into the IB to approximate the Neumann boundary condition of α. In the second part of this chapter, the accuracy of the solver is validated through different test cases involving both stationary and moving IBs.

Various aspects including the local pressure, forces on the IBs and the flow field are carefully investigated and compared with the experimental data and other numerical sources. The numerical results demonstrate that the IBM can effectively handle the interaction between the air-water interface flows and the IB.

CHAPTER V

RANS Simulations of a Ship Advancing with a Rotating Rudder

The goal of the present IBM is to provide a robust and efficient numerical tool for ship hydrodynamic applications involving moving solid surfaces. From the discussions in the previous chapters, it can be seen that the body-fitted mesh has the advantage of easily controlling the near-wall cell spacing, yet the IBM is more flexible to deal with moving objects without deforming the background mesh. In this chapter, the new solver is tested on a problem with a combined usage of the body-fitted unstructured mesh and the IBM, which is referred as the hybrid method. The problem under consideration is a ship model advancing with a semi-balanced rudder at different deflection angles. The comparison with the experimental data and numerical results on pure body-fitted meshes is made to demonstrate the benefit of the hybrid method on the simulation involving a realistic geometry in the field of ship hydrodynamics. This test case is used as a benchmark case in the SIMMAN 2014 workshop, whose purpose is to evaluate the capability of different ship maneuvering simulation methods. The physical experiments were carried out in the Seakeeping and Maneuvering Tank, Japan Marine United (JMU) Corp. (*Y. Yoshimura and Yano* (2013)) in 2012. The force on the ship was recorded for the purpose of numerical validation. If one were to use conventional body-fitted meshes, multiple simulations at each deflection angle of

the rudder would be required. Inspired by the single-run procedure used on the ship self-propulsion problems described in *Xing et al.* (2008) and *Shen et al.* (2015), it is of interest to study if the process of running multiple simulations can be replaced by a single run with the rudder rotating gradually. It is very challenging to use traditional body-fitted mesh for the moving rudder, because the large deformation of the mesh around the rudder can significantly affect the mesh quality. Subsequently, it may lead to the divergence of the simulation. In addition, the sliding mesh technique (for example the arbitrary mesh interface (AMI) method in OpenFOAM) is also unsuitable in this case due to the arrangement of the rudder, which will be shown in the following section. In comparison, the present IBM provides a feasible way to conduct such a simulation without the need to generate a moving body-fitted mesh. It is also worth mentioning that the overset mesh technique is an alternative way to model the moving rudder. However it still requires effort to generate body-fitted mesh around the rudder, and to suitably distribute the mesh between the rudder horn and the moving part of the rudder for proper interpolation (*Shen et al.* (2015)).

5.1 KCS Ship Model

The ship considered in this chapter is the model-scale KRISO container ship (KCS), which is widely used in the experiments and numerical validations of ship hydrodynamics. The ship model and the rudder are shown in Fig. 5.1. The propeller was not considered in the experiment.

The main particulars of the model-scaled KCS and the rudder are listed in Table 5.1 and Table 5.2, respectively.

A right-handed earth-fixed coordinate system is used in the simulations. The origin is located at the forward perpendicular, with the x-axis pointing to the stern, and z-axis pointing upwards. In the simulations, the ship model is fixed in the even-keel condition without considering the vertical motions. Whereas the experimental

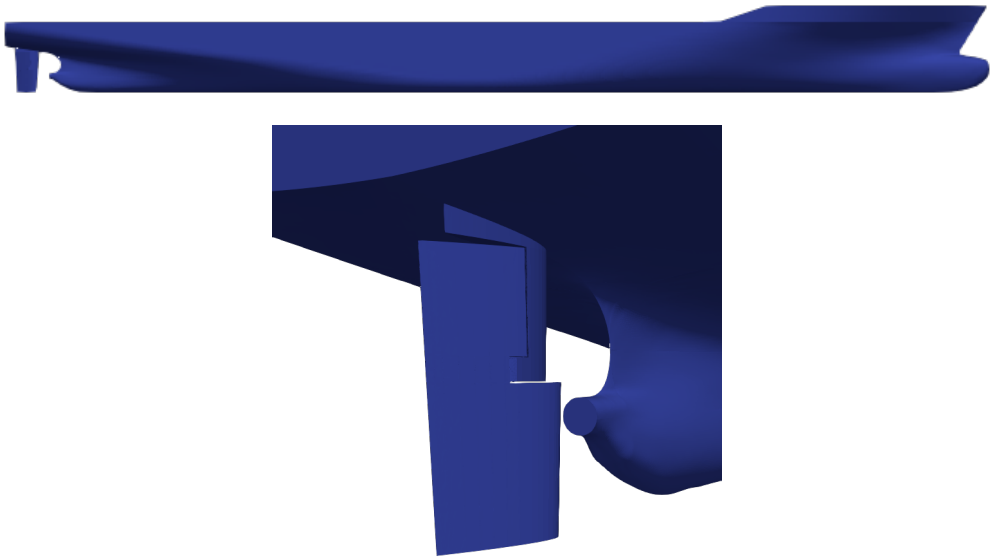


Figure 5.1: Geometry of the KCS ship model and the rudder

Table 5.1: Main Particulars of the model-scaled KCS

Main particulars	Symbol	Value
Scale	λ	105
Length between perpendiculars	$L_{\text{pp}}(m)$	2.19
Length of waterline	$L_{\text{wl}}(m)$	2.2143
Width	$B_{\text{wl}}(m)$	0.3067
Depth	$D(m)$	0.1810
Draft	$T(m)$	0.1029
Displacement	$\Delta(m^3)$	0.0449
Wetted area without rudder	$S(m^2)$	0.8644
Longitudinal center of buoyancy (fwd+)	LCB ($\%L_{\text{pp}}$)	-1.48
Vertical center of gravity	KG (m)	0.118
Moment of inertia	$K_{\text{yy}}/L_{\text{pp}}$	0.25

Table 5.2: Main Particulars of the model-scaled rudder

Main particulars	Value
Scale	105
Type	Semi-balanced horn rudder
Area of rudder (m^2)	0.0104
Lateral Area of rudder (m^2)	0.0049

ship model is free to sink and trim in the experiments. The forward speed of the ship model is $V_{\text{model}} = 1$ m/s, corresponding to $Fr = 0.216$ and 19.9 knots in full

scale. The Reynolds number is 1.72×10^6. Four rudder angles are tested, which are $\delta = -5°, -10°, -15.1°$ and $-20.1°$, respectively.

Since the ship hull and the rudder horn are fixed, it is reasonable to use body-fitted mesh to represent these parts to efficiently resolve the boundary layer. The moving part of the rudder is modeled using an IB to take full advantage of the IBM for modeling moving geometries. The numerical solutions are compared with the experimental data to demonstrate the capability of the present IBM. Since the accuracy is limited by not only the IBM, but also the underlying solver and discretization schemes, the RANS simulations with the rudder at fixed deflection angles are also carried out using body-fitted meshes as a comparison. Three sets of simulations are carried out to systematically show the capability of the solver as listed in Table 5.3. In the first set, a mesh convergence study is conducted by using body-fitted meshes to find an appropriate resolution of the background mesh to balance the accuracy and computational cost. In the second set, the simulations of the rudder at all fixed deflection angles are carried out on body-fitted meshes. The body-fitted results represent the baseline of the accuracy of the solver that the present IBM combined with. The simulation with the hybrid method at the maximum deflection angle $\delta = -20.1°$ is also carried out to validate the IBM. In addition, a simulation without the IB wall function is carried out using the hybrid method at the maximum deflection angle to examine the effect of the IB wall function. In the third set, the simulation of a rotating rudder is conducted with the hybrid method. The results demonstrate that, with the help of the present IBM, the single-run procedure can be applied to this problem to predict the hydrodynamic forces accurately and efficiently.

5.2 Mesh Convergence Study

In this section, the simulation with the rudder at $\delta = 0°$ is carried out. A set of three body-fitted meshes with a constant refinement ratio is used to test the spatial

Table 5.3: Summary of the simulations of a ship model advancing with a deflected rudder

Case No.	Mesh Refinement	Mesh Type	Symmetric	$\delta(^\circ)$	WF	RR
1	C	BF	Y	0	Y	N
2	M	BF	Y	0	Y	N
3	F	BF	Y	0	Y	N
4	M	BF	N	-5	Y	N
5	M	BF	N	-10	Y	N
6	M	BF	N	-15.1	Y	N
7	M	BF	N	-20.1	Y	N
8	M	Hybrid	N	-20.1	Y	N
9	M	Hybrid	N	-20.1	N	N
10	M	Hybrid	N	—	Y	Y

C: coarse, M: medium, F: fine. BF: body-fitted. Symmetric: symmetric boundary condition on the centerline. WF: wall function. RR: rotating rudder.

convergence and to find an appropriate background mesh for the following simulations by the IBM. The meshes are generated by snappyHexMesh, an automatic mesh generation utility provided by OpenFOAM. The meshes are systematically refined by refining the background meshes with a refinement ratio of $\sqrt{2}$ in each direction. In addition, the mesh is further refined near the ship hull, the rudder and the air-water interface. Only half of the ship is used in the simulations with a symmetric boundary condition applied on the centerline to accelerate the simulation. The total number of cells of each mesh is listed in Table 5.4.

Table 5.4: Number of cells for the mesh convergence study

Mesh name	Coarse	Medium	Fine
Number of cells	518,640	1,220,948	2,382,616

Fig. 5.2 shows the computational domain and the local mesh refinement around the ship. The dimensions of the computational domain are $4L_{\text{pp}} \times L_{\text{pp}} \times 1.5L_{\text{pp}}$. The boundary conditions are summarized in Table 5.5, where the bottom and the far field of the domain are both included in the inlet boundary.

The viscous pressure drag coefficient C_p, the frictional drag coefficient C_v, and the

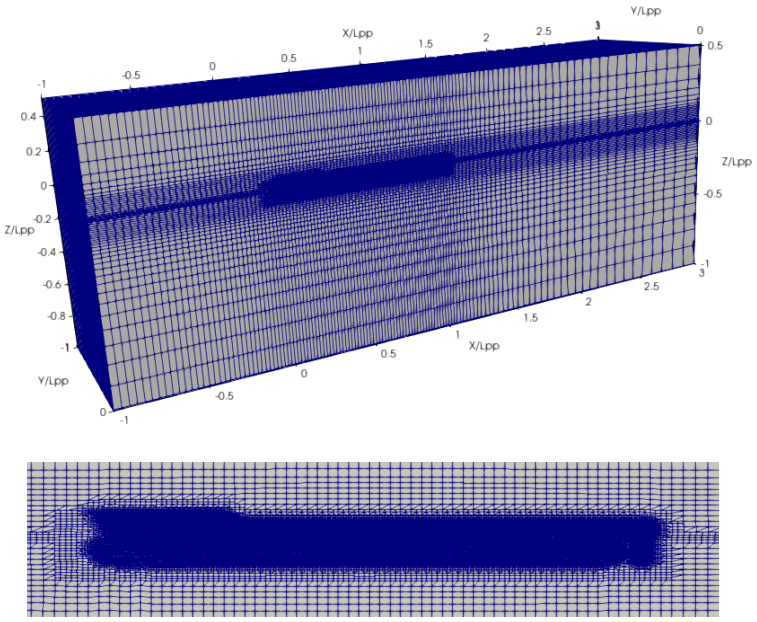


Figure 5.2: Computational domain and the local mesh refinement

Table 5.5: Summary of the boundary conditions for the mesh convergence study

Boundary Names	α	U	p_{rgh}
Inlet	waveAlpha	waveVelocity	zeroGradient
Outlet	zeroGradient	zeroGradient	fixedValue
Top	inletOutlet	pressureInletOutletVelocity	totalPressure
Centerline	symmetryPlane	symmetryPlane	symmetryPlane
Hull and rudder	zeroGradient	movingWallVelocity	fixedFluxPressure
Boundary Names	$\tilde{\nu}$	ν_t	
Inlet	fixedValue 5.871e-6	fixedValue 1.27e-6	
Outlet	zeroGradient	zeroGradient	
Top	zeroGradient	calculated	
Centerline	symmetryPlane	symmetryPlane	
Hull and rudder	fixedValue 0	nutUSpaldingWallFuntion	

total drag coefficient C_T are defined as:

$$C_p = \frac{F_p}{\frac{1}{2}\rho U^2 LT} \quad C_v = \frac{F_v}{\frac{1}{2}\rho U^2 LT} \quad C_T = \frac{F_T}{\frac{1}{2}\rho U^2 LT} \tag{5.1}$$

where F_p, F_v and F_T are the viscous pressure drag, frictional drag and the total drag on the ship hull and the rudder. Table 5.6 shows the mesh convergence results of the

drag coefficients on different meshes. The total drag coefficient C_T shows that the numerical solutions converge with satisfying agreement.

Table 5.6: Results of the drag coefficients in the mesh convergence study

Mesh name	C_p	C_v	C_T	Exp	Error
Coarse	0.00272	0.01682	0.01954	0.0178	9.8%
Medium	0.00237	0.01686	0.01924	0.0178	8.1%
Fine	0.002244	0.01696	0.01921	0.0178	7.9%

Fig. 5.3 shows the profile of the air-water interface on the ship hull extracted with the volume fraction $\alpha = 0.5$. It can be seen that the medium and fine meshes agree well with each other, while the coarse mesh fails to predict water elevation near the bow and the stern. Fig. 5.4 shows the water elevation predicted on the medium mesh. The result shows that the waves crests generated at the bow, the shoulder and the stern are well captured. In the following sections, the medium mesh is used for all the simulations considering a balance of accuracy and efficiency.

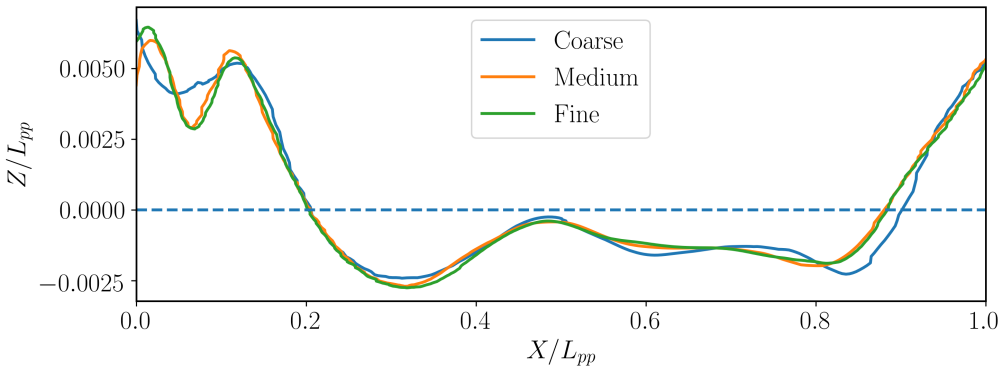


Figure 5.3: Free-surface profile on the ship hull with $\alpha = 0.5$

5.3 Simulations with the Rudder at Fixed Deflection Angles

The accuracy of the IBM solver relies on not only the additional operations introduced by the IBM, but also the underlying RANS solver and numerical discretization

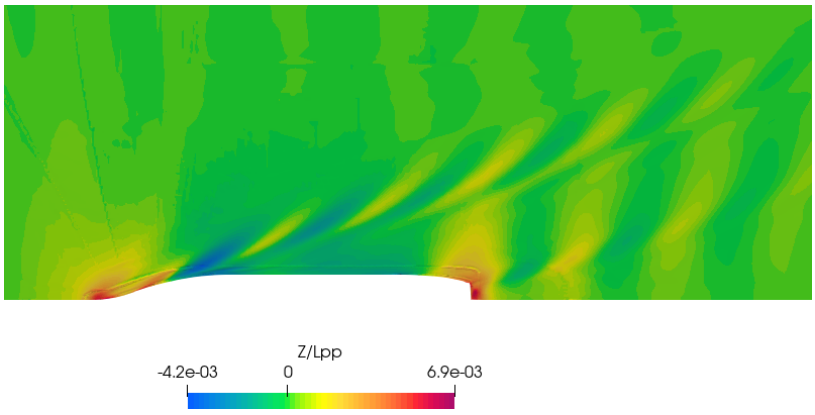


Figure 5.4: Free-surface elevation on the medium mesh

schemes as discussed in the previous chapters. In this section, the simulations with the rudder at fixed deflection angles using pure body-fitted meshes are carried out. Therefore, the accuracy of the underlying RANS solver and numerical schemes can be evaluated. As a comparison, the case with the rudder at the maximum deflection angle $\delta = -20.1°$ is also simulated using the IBM with a fixed IB. Comparisons of the surge coefficient X', sway coefficient Y' and the normal force coefficient FN' of the rudder with the experimental data are made to validate the solver. The coefficients are defined in Eqn. 5.2 as:

$$X' = \frac{X}{\frac{1}{2}\rho U^2 LT} \quad Y' = \frac{F_v}{\frac{1}{2}\rho U^2 LT} \quad FN' = \frac{FN}{\frac{1}{2}\rho U^2 LT} \tag{5.2}$$

where X and Y are the surge and sway forces on the ship including the rudder, respectively. FN is the force on the moving part of the rudder in the direction normal to the rudder mid plane.

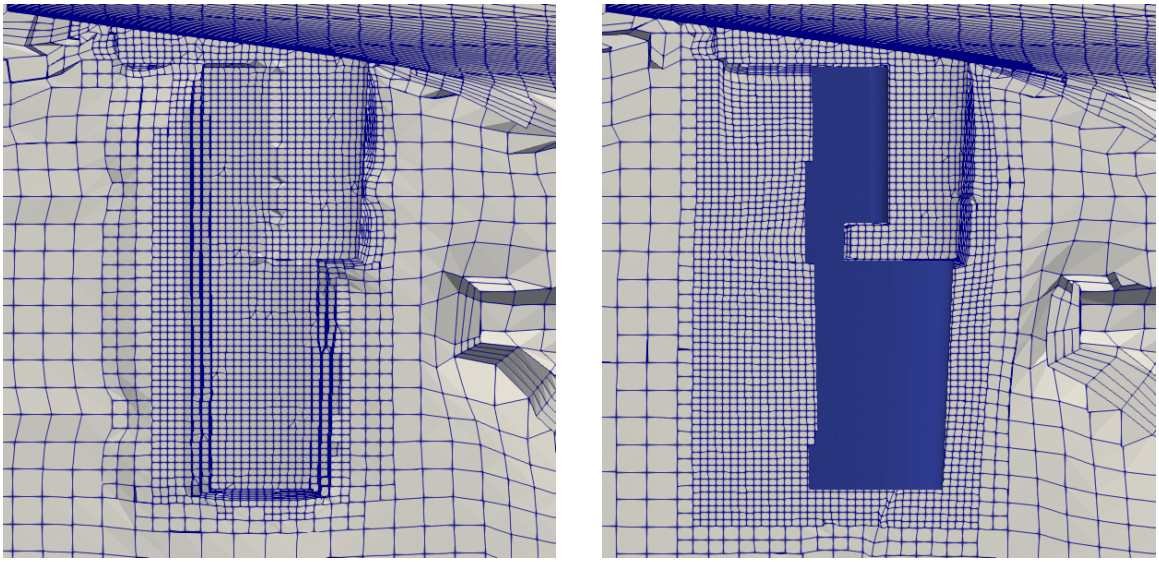
Since the ship hull with the deflected rudder is not symmetric, the medium mesh used in the previous section is mirrored with respect to its centerline. The total number of cells used by both the body-fitted mesh and the IBM is listed in Table 5.7.

Fig. 5.5 shows the mesh in the vicinity of the rudder for both body-fitted and

Table 5.7: Total number of cells for the simulations of the rudder at fixed deflection angles

Mesh name	Body-fitted	Hybrid
Number of cells	2,441,378	2,499,112

IB meshes at the maximum deflection angle. It can be seen that in the small gap between the rudder horn and the moving part, the number of cells are not enough for the interpolation of velocity by IBM. For cells in such region, the velocity in the forcing cells is simply set to be the rigid-body velocity. For the case of a stationary rudder, the velocity in these forcing cells is set to be zero. In addition, compared with the body-fitted mesh, the background mesh does not have prism boundary layers around the rudder. The near-wall cell spacing is larger for the IB background mesh than the body-fitted mesh. Therefore, the IB wall function introduced in Chapter 3 is also applied. To evaluate the effect of the IB wall function, a simulation using the IBM without the wall function is also conducted.



(a) Body-fitted mesh (b) IB mesh

Figure 5.5: Mesh in the vicinity of the rudder at $\delta = -20.1°$

Fig. 5.6, Fig. 5.7 and Fig. 5.8 show the results of the force coefficients X', Y' and FN', respectively. In the figures, "BF" represents the results of cases No. 4–

7 in Table 5.3. It should be noted that for each deflection angle, there are two data points for the experimental data. It is because in the experiments, tests were carried out at $\delta = \pm 5°, \pm 10°, \pm 15.1°$ and $\pm 20.1°$. The numerical results are reflected accordingly with respect to the y axis to compare with the experimental data. Fig. 5.6 shows that when $\delta > -15°$, the total drag is well predicted by OpenFOAM. However, when $\delta \leq -15°$ the numerical results overpredict the drag. It is because when the deflection angle is large, the flow massively separates from the surface of rudder. The Spalart-Allmaras turbulence model used in the current simulation is not fully capable of capturing the massive flow separation on the rudder. In addition, the wall function that is used in the IBM is based on the near-wall velocity profile in an attached turbulent flow, which does not hold true when the deflection angle is large. Compared between the results of Cases No.8 and No.9, it shows that the usage of the wall function does not have notable impact on the total drag. It may because that the rudder is in the region of the wake flow, where the boundary layer effect is not as significant as on the ship hull.

Fig. 5.7 shows that the numerical prediction of the sway force matches well with the experimental data at all deflection angles. The sway force increases consistently with the increase of the deflection angle. It makes sense since the dominant contribution of the sway force comes from the lift force on the rudder, and the lift force increases with the increase of the deflection angle. The prediction of the IBM at the maximum rudder angle is less than the results on the body-fitted mesh, while both results fall into the uncertainty region of the experimental data. In addition, the IB wall function used in the IBM has no notable influence on the sway force.

Fig. 5.8 shows the force on the moving part of the rudder in the direction normal to the rudder mid plane. Again, the numerical predictions match well at all deflection angles. The results at the maximum deflection angle obtained by the body-fitted mesh and the IBM agree well with each other, which again validates the accuracy of the

IBM. The result without using the IB wall function underpredicts the normal force on the rudder. In the circumstance of simulating turbulent flows at high Reynolds numbers, it is almost unavoidable for the IBM to have similar near-wall resolution as the body-fitted mesh if the total number of cells is similar. Although due to the fact that the rudder is in the wake flow, the boundary layer effect is not as significant as on the ship hull. Using the IB wall function still improves the overall accuracy, especially when it only slightly increases the computational time.

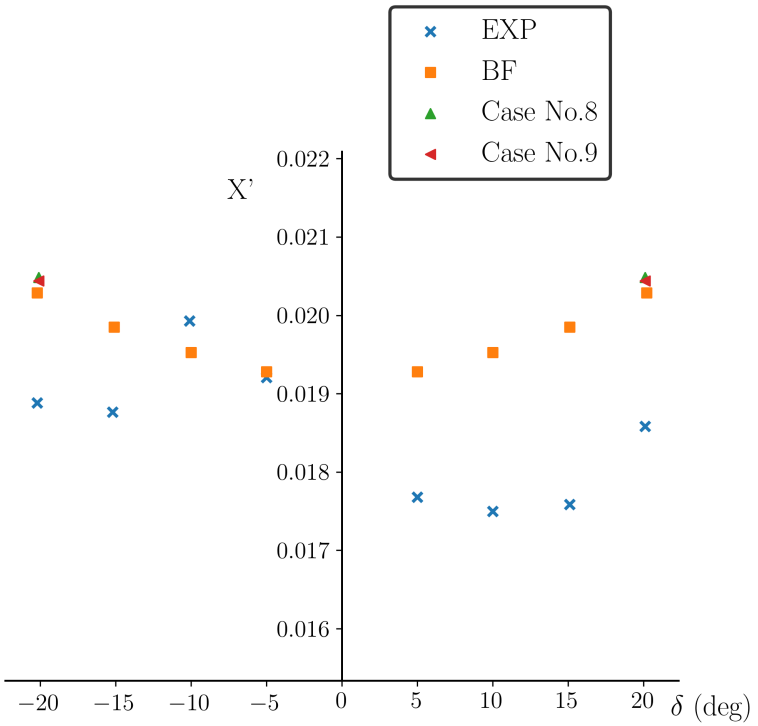


Figure 5.6: Surge force coefficient X' predicted at the fixed deflection angles

Fig. 5.9 and Fig. 5.10 show the velocity and dynamic pressure at $z = -0.065$ m when $\delta = -20.1°$, respectively. The velocity field shows the flow separation happens at this deflection angle, which confirms the discussion about the forces on the rudder in the previous paragraphs. For both the velocity and the pressure, the results of the IBM match well with the body-fitted results.

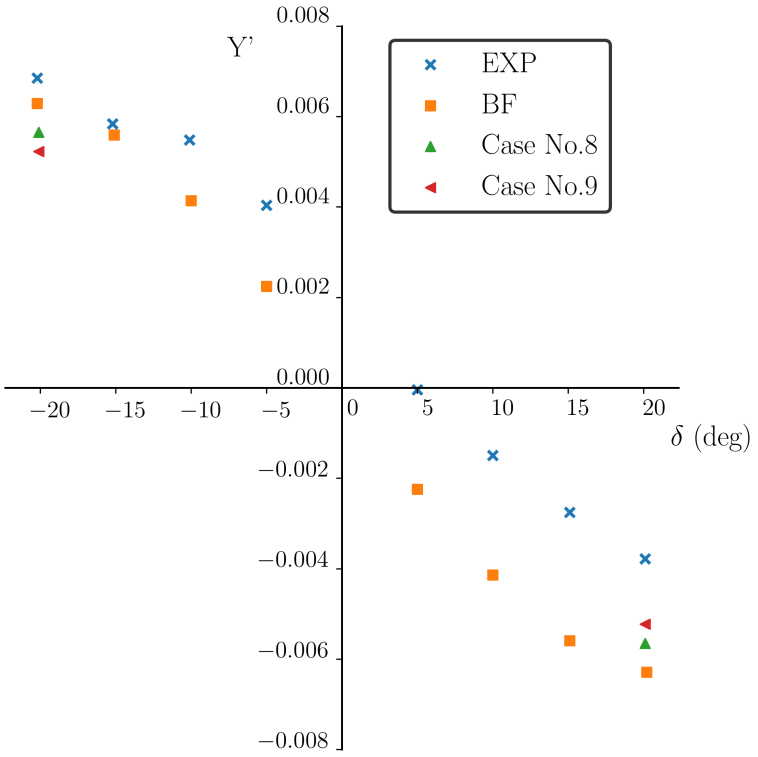


Figure 5.7: Sway force coefficient Y' predicted at the fixed deflection angles

5.4 Simulation with a Rotating Rudder

In this section, the simulation of the rotating rudder with the IBM is presented. Starting from the neutral position corresponding to $\delta = 0°$, the rudder starts to rotate with a half cosine ramp with a period of time t_r. When it reaches the target deflection angles, $-5°, -10°, -15.1°$ and $-20.1°$, the rudder stays fixed within a period of time t_{hold}. The rudder then starts to rotate again until it reaches the next target deflection angle. Fig. 5.11 shows the time history of the deflection angle of the rudder, in which $t_r = 2$ s and $t_{\text{hold}} = 5$ s.

Fig. 5.12∼5.14 show the surge and sway forces on the ship hull including the rudder, and the normal force on the moving part of the rudder, respectively. At different deflection angles, the numerical results predicted by using the rotating rudder match well with body-fitted results. When $\delta \leq -15.1°$, the total force predicted by

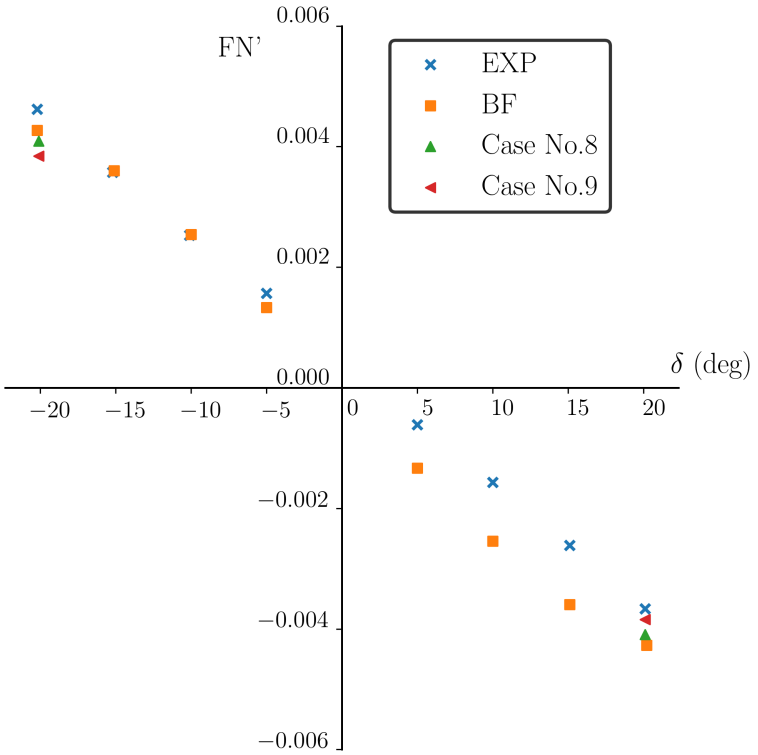
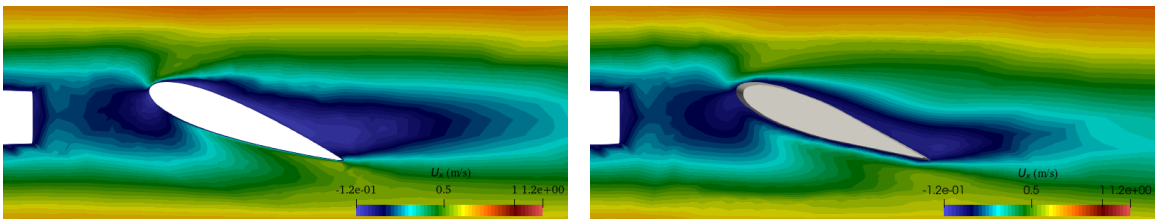
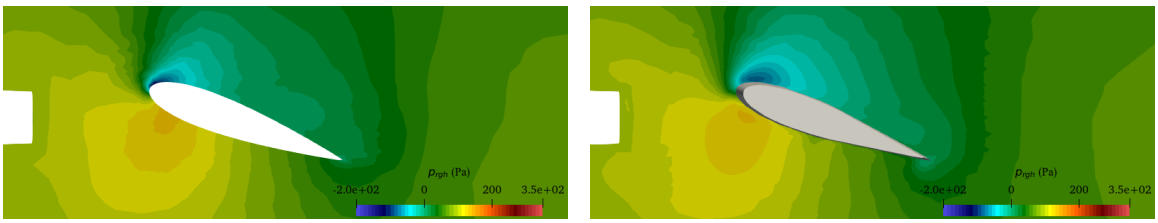


Figure 5.8: Normal force coefficient of the rudder FN' predicted at the fixed deflection angles



(a) Body-fitted mesh (b) IB mesh

Figure 5.9: Comparison of the velocity at $z = -0.065$ m when $\delta = -20.1°$



(a) Body-fitted mesh (b) IB mesh

Figure 5.10: Comparison of the dynamic pressure at $z = -0.065$ m for $\delta = -20.1°$

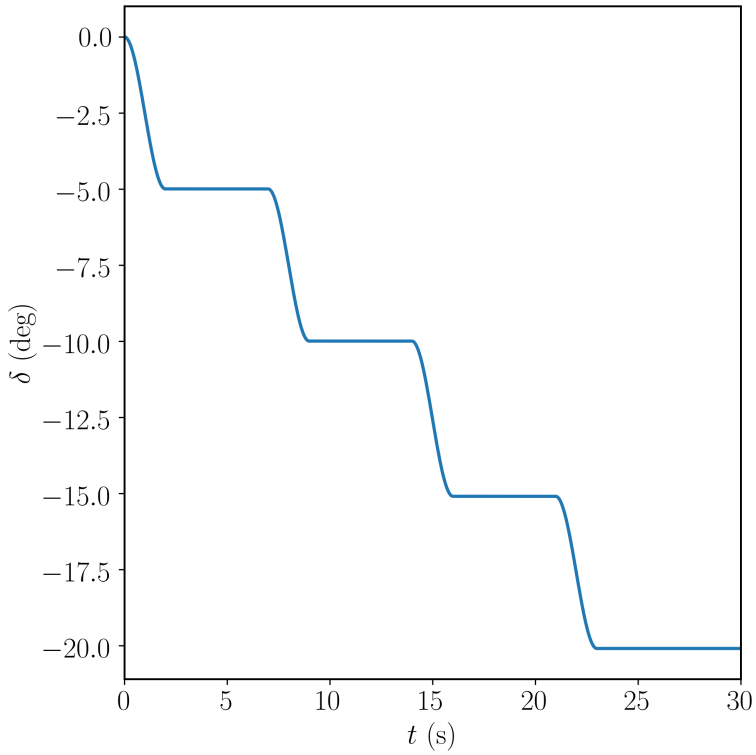


Figure 5.11: Time history of the deflection angle of the rudder

both the body-fitted mesh and the IBM has larger discrepancy compared with the experimental data than when the deflection angle is relatively small. However, the predicted force by using the IBM is still as good as using the body-fitted mesh. As discussed previously, the accuracy of the IBM solver is also limited by the accuracy of the underlying RANS solver and the numerical discretization schemes. It makes sense that the IBM does not obtain more accurate results if the error is dominated by the discretization and modeling errors.

In the previous discussion, it is shown that using the single-run procedure can well predict the force on the ship hull and the rudder. However, the IBM requires more computational time due to the additional operations of categorization of the cells, interpolation of velocity, and use of the IB wall function. It is worth studying if using the single-run procedure is less time-consuming than running with the body-fitted mesh for four times with the rudder at each deflection angle. Table 5.8 shows the

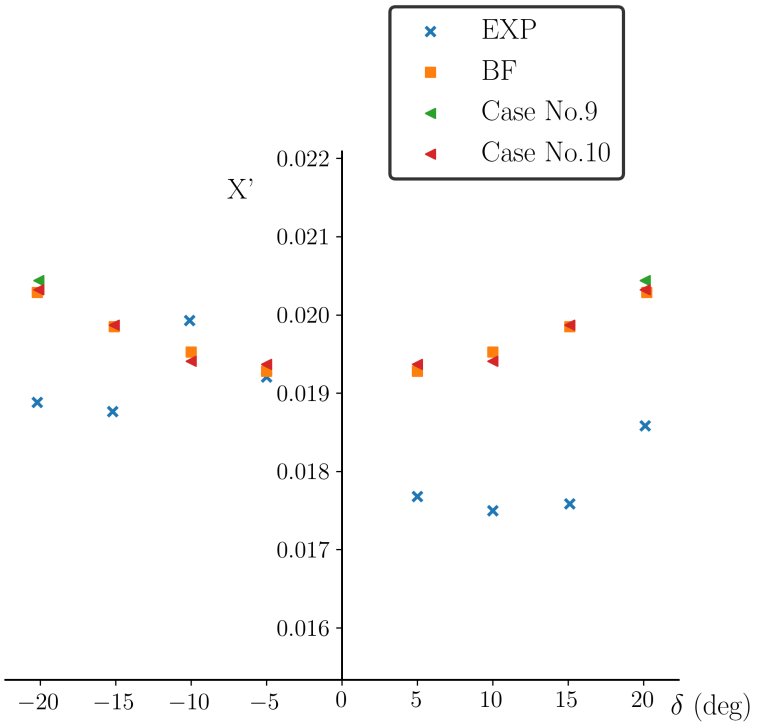


Figure 5.12: Surge force coefficient X' predicted with a slowly rotating rudder

comparison of the computational time between using the body-fitted mesh and the IBM. All cases run in parallel on two Intel Xeon Phi 7250 CPUs (136 processors in total). The comparison shows that even for simulating one static deflection angle, the IBM is about 5% faster than using the body-fitted mesh. This is because by using the body-fitted mesh around the rudder, there are small cells with bad quality especially near the trailing edge of the moving part of the rudder. Since the time step size is adjusted automatically according to the limitation of the maximum Courant number, the time step size is decreased due to the presence of the small cells. In contrast, the IB mesh in the vicinity of the rudder is much closer to orthogonal than the body-fitted mesh, and the small cells are avoided. Therefore, it runs faster with larger time step size when the maximum Courant number is set to be the same. It is also shown that for this specific case, the computational time saved by the larger time step is more than the time cost by the additional operations due to the IBM. It should be

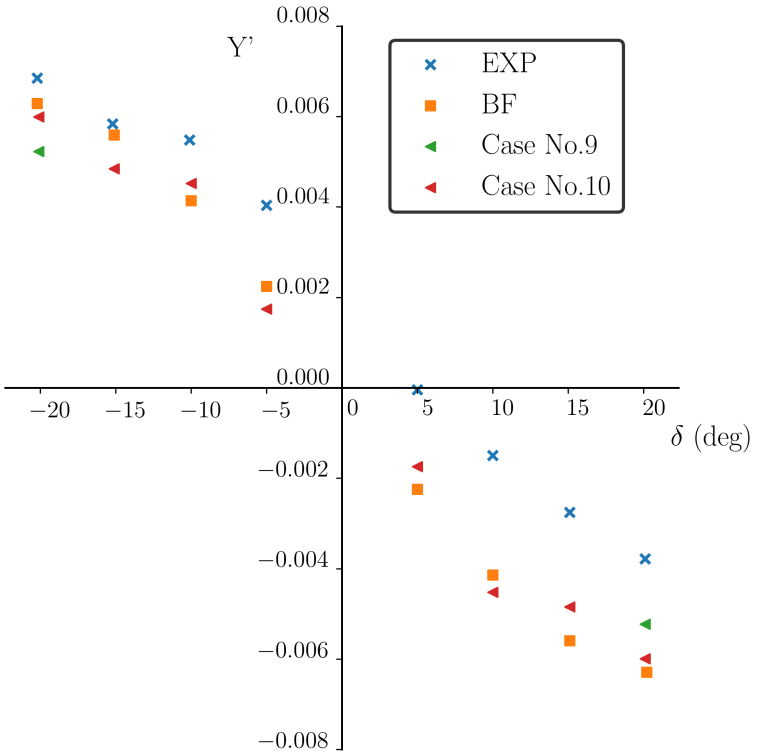


Figure 5.13: Sway force coefficient Y' predicted with a slowly rotating rudder

noted that the quality of the body-fitted mesh definitely has the potential to improve by manually creating the mesh instead of generating the mesh automatically like in the current research. However it requires much more effect to generate a body-fitted mesh with good quality than using the IBM on the background mesh.

The computational time of simulating all the deflection angles saved by using the IBM is even more notable, which is about 40%. In addition to the benefit for simulating one deflection angle as discussed above, the single-run procedure speeds up the whole simulation even further, because the flow field at every deflection angle is developed from the solution at the previous deflection angle instead of the initial condition. Therefore it is reasonable that the solution at each deflection angle converges faster.

The results discussed above demonstrate that the present IBM can simplify the simulation of a ship advancing at different deflection angles of the rudder with good

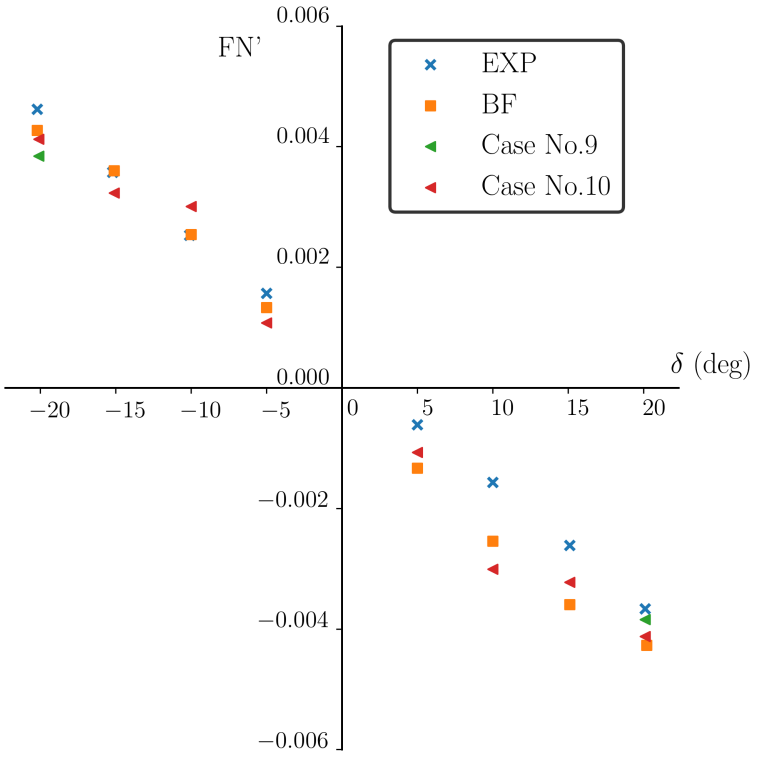


Figure 5.14: Normal force coefficient of the rudder FN' with a slowly rotating rudder

accuracy. It can significantly speed up the whole simulation process by saving the effort of generating mesh with good quality around the complex geometry. In addition, users do not need to worry about any degradation of the mesh quality due to the motion of the geometry.

Table 5.8: Comparison of the computational time between using the body-fitted mesh and the IBM

Case	Wall clock time (hrs)
Body-fitted (one deflection angle)	13.58
Body-fitted (four deflection angles)	54.32
Hybrid (one deflection angle)	12.87
Hybrid (rotating rudder)	32.72

5.5 Summary

In this chapter, the IBM is used to simulate a rotating rudder behind a ship hull. Only the moving part of the rudder is modeled with the IB, while the ship hull and the rudder horn is still modeled using the unstructured body-fitted mesh. The body-fitted mesh has the advantage of controlling the near-wall cell spacing with relatively fewer cells by using the prism boundary layers, while the IBM has the advantage of simplifying the simulation of the flow around moving complex geometries. The combined usage benefits from the advantages of both, and is validated from the test cases in this chapter. The numerical results of the force show that the present numerical solutions match well with the experimental data when the deflection angle of the rudder is small. While when $\delta < -15.1^{\circ}$, massive flow separation appears on the suction surface of the rudder. In this case, both the results of the body-fitted mesh and the IBM have a large discrepancy from the experimental data. It is because the Spalart-Allmaras turbulence model used by both techniques and the wall function are based on the assumption of attached turbulent flow.

The numerical results obtained by simulating a rotating rudder with the IBM demonstrate that the single-run procedure can effectively replace the multiple simulations required at each individual deflection angle of the rudder. The comparison of the computational time shows the benefit of using the IBM even for the case with the rudder at a fixed deflection angle. By representing the rudder with the IB, the mesh quality is largely preserved especially in the regions near the trailing edge. Therefore, the simulation using the IBM runs faster by using a relatively larger time step size under the condition of the same maximum Courant number. A 5% speed up is achieved for the simulation with a fixed deflected rudder. By using the single-run procedure, the overall computational time cost is further reduced because the flow field at each deflection angle does not need to develop from the initial condition, but from the developed flow field at the previous deflection angles. The single-run pro-

cedure runs 40% faster for four deflection angles than using the body-fitted mesh to run four times at each individual deflection angle.

In summary, this chapter demonstrates both the efficiency and accuracy of the combined usage of the IBM and body-fitted unstructured mesh, which is difficult to achieve by using solely body-fitted unstructured meshes. If the IBM is integrated with more advanced turbulence models and transition models, the IBM solver has the potential to predict more accurately the turbulent flow with massive separation.

CHAPTER VI

Conclusions and Future Work

6.1 Summary

A direct-forcing IBM is developed in the framework of a finite-volume incompressible solver for high-Reynolds-number flows. The IBM can be used on both structured and unstructured meshes for representing both fixed and moving complex geometries at various Reynolds numbers, ranging from laminar to turbulent. The IBM is successfully applied to simulate both single-phase and two-phase flows. With the capability of the combined usage of the IBM and the body-fitted strategy on an unstructured background mesh, it significantly simplifies the simulations when one boundary moves relative to others in a close proximity (e.g. a rudder that has a rotating part relative to the fixed rudder horn), without loss of accuracy.

The numerical details of the present IBM, including the process of cell categorization, layout of the forcing cells and the interpolation, are discussed in details in Chapter 2. Detailed verification through order-of-accuracy tests and the method of manufactured solutions demonstrates that the present IBM is of second order. Although the order of accuracy of the underlying numerical schemes will decrease to the first order in the presence of skewed and non-orthogonal cells, the IBM more accurately represents boundary conditions, and allows for orthogonal meshes to be used for complex geometries. The improved mesh quality can also lead to less com-

putational cost, which is demonstrated through the simulation of a ship advancing with a rotating rudder.

The calculation of force on the IB surfaces is done by the volume integration of the terms in the momentum equation. Compared with calculating the force by the surface integration on the IB surface in the same way as on the body-fitted mesh, the current method avoids the steps of projecting the pressure and strain-rate tensor from the background mesh, where the governing equations are solved, onto the IB surface. In addition, the method does not require any additional modification for moving IB surfaces, or for two-phase flows. The accuracy of the solver is validated through test cases including fixed and moving IB surfaces. To speed up the simulations involving moving IB surfaces, the information of cell connectivity at the beginning of the simulations is stored.

The coupling between the IBM and the Spalart-Allmaras turbulence model for RANS simulations is carefully discussed in Chapter 3. It is a common disadvantage for IBMs that the near-wall cell spacing is harder to control than body-fitted meshes. It means IBMs require much finer background meshes than body-fitted meshes to achieve a similar near-wall cell spacing. To utilize background meshes that are not fine enough to fully resolve boundary layers, an IB wall function is implemented in the present IBM. The wall function provides a smooth velocity profile from the outer edge of the logarithmic region down to the solid wall. The wall function makes the present IBM flexible and accurate to predict turbulent flows without the need to fully resolve the boundary layer. The IB solver is validated through benchmark cases including the turbulent flow over a flat plate, flow inside a diffuser, and flow around an oscillating airfoil. The challenging case of the turbulent flow over a KVLCC2 ship model is also carried out. Good agreement is achieved among the current numerical results, the experimental data, and other numerical results in terms of the force on the IB surface and the flow field.

The coupling between the IBM and the air-water two-phase flow solver is developed in Chapter 4. The VoF method is used to model the air-water interface by solving a transport equation for the fluid volume fraction. In the momentum equation, the total pressure is replaced by the dynamic pressure to simplify the pressure boundary conditions. By doing so, the density gradient is also introduced into the momentum equation, which is problematic for the interaction between the air-water interface and the IB surface. The dilation procedure from *Sun and Sakai* (2016) is adopted to extend the field of the volume fraction into the IB surface efficiently. By doing so, a homogeneous Neumann boundary condition is imposed on the IB surface. It is equivalent to a contact angle of 90 degrees. Numerical simulations of waves in tanks with two different shapes, the dam-break problems, and the water exit of a circular cylinder are carried out. The numerical results match well with the experimental data with respect to the force on the IB surface, the flow field and the profile of the air-water interface.

The simulation of the model-scale KCS advancing with a rotating semi-balanced rudder is conducted in Chapter 5. The aim is to use the single-run procedure to predict the force on the ship instead of multiple runs at several fixed deflection angles of the rudder. The accuracy of the single-run procedure is demonstrated by comparing between the experimental data, the results by using purely body-fitted mesh and the results by the combined usage of the IBM and body-fitted mesh. In addition, the single-run procedure offers a 40% speedup compared with multiple runs using purely body-fitted mesh for each deflection angle.

In summary, this thesis implements a second-order direct-forcing IBM that works on both structured and unstructured meshes in the framework of OpenFOAM. The IBM can be used alone to represent all wall boundaries in a simulation, as well as together with body-fitted boundary conditions to simulate the relative motion between different solid surfaces.

To be efficient for simulating turbulent flows, the wall function is implemented in the cases when the near-wall mesh is not fine enough to fully resolve the boundary layer. The interpolation of the velocity boundary condition results in a linear interpolation for the velocity component normal to the IB surface. For laminar flows, the tangential velocity component is also assumed to have a linear distribution between the IB surface and the solution of the governing equations. For turbulent flows, the tangential velocity component is corrected based on the Spalding's velocity profile, which also assumes the flows are fully attached. In addition, the Spalart-Allmaras turbulence model used in this research is favourable for attached turbulent flows or flows with mild separation. Due to these underlying assumptions and the limitations of the turbulence model, the proposed work is most suitable for laminar and attached turbulent flows. Whereas for flows with separation, the validation results in this work show that the accuracy is case specific. In general, the current method is less accurate when the flow separation is massive, especially for turbulent flows.

The present IBM is also implemented for air-water two-phase flows, where the boundary condition of the volume fraction at the intersection of the air-water interface and IB surface is properly handled by a dilation procedure. The performance of the present IBM is thoroughly verified and validated through various test cases.

The combined usage between the IBM and the unstructured body-fitted mesh is shown by simulating a ship model advancing with a rotating rudder. The ship hull and the fixed rudder horn is modeled by the body-fitted boundary conditions, while the rotating rudder blade is modeled by an IB. The results demonstrate both the accuracy, flexibility and robustness of the present IBM.

6.2 Key Contributions

1. Develop and validate a framework of a second-order direct-forcing IBM, that is suitable for both structured and unstructured meshes. The framework aims

to serve as a robust and efficient numerical tool to simulate single-phase and air-water two-phase flows accurately in a wide range of Reynolds numbers.

2. The framework is designed such that the IBM can be used together with body-fitted boundaries to simulate the relative motions between walls. This feature provides flexibility in the cases when a body-fitted boundary is suitable for the boundary layers. The simulation of a ship advancing with a rotating rudder is one of the examples where the ship hull and the rudder horn can be modeled using body-fitted boundaries whereas an IB surface is more suitable for the rotating part of the rudder.

3. Couple the present IBM with the Spalart-Allmaras turbulence model. Implement a universal wall function to efficiently alleviate the requirement on the near-wall mesh refinement in the high-Reynolds-number flows. The wall function significantly increases the range of the applications of the present IBM, because a wall-resolved mesh is not always available for RANS simulations of turbulent flows around complex geometries.

4. Couple the present IBM with the air-water two-phase flow solver. The dilation method proposed by *Sun and Sakai* (2016) is adopted. One of the key contributions of the current work is to extend this method to unstructured meshes, and to apply it in the ship hydrodynamic applications.

6.3 Future Work

1. The present IBM demonstrates its capability of handling the relative motion between an IB surface and a solid wall represented by body-fitted meshes. The relative motion between multiple IB surfaces is worth studying in the future. For example, the IBM can be used to investigate the impact problem between

two objects, where both objects are modeled as IB surfaces. The current implementation allows for arbitrary numbers of IB surfaces. The missing part is a proper model that can calculate the contact force when the IB surfaces move close to each other.

2. Numerical simulations of a ship advancing with a rotating rudder show that the Spalart-Allmaras turbulence model is inadequate to simulate turbulence flows with massive separation. To further improve the capability of the present IBM for simulating turbulence flows, two-equation turbulence models, such as the $k-\omega$ SST turbulence model or other advanced turbulence modeling (such as LES and Detached Eddy Simulations) can be considered. To capture the flow separation more accurately, transition models can also be considered to implement in the present IBM framework.

3. When simulating high-Reynolds-number flows, the present solver requires users to determine the distance from the IB surface where the shear velocity is calculated using the wall function. In addition, this distance is the same along an IB surface. A future study can be conducted to automatically decide the distance and have varying distance along the IB surface depending on the local boundary-layer thickness.

4. The velocity interpolation of the present IBM is implemented based on one major assumption that the flow is attached. In the future, the accuracy of the interpolation may be improved by considering the actual velocity profile in the region of separation. In this case, the location of separation should be predicted accurately. Otherwise, using an interpolation based on a profile of separated flow in a fully-attached region is equally erroneous.

5. All state-of-the-art IBMs interpolate the velocity using either linear interpolation, least square interpolation, inverse square distance weighting or the Lapla-

cian interpolation used in the current work. One interesting future study is the possibility to use machine learning to generate the interpolation stencils based on the analysis of realistic flows.

APPENDIX

APPENDIX A

Source Term for the Manufactured Solution of a 2-D Steady Heat Conduction Problem

The source term in Eqn. 2.35 is given for the 2-D steady heat convection equation. This source term is derived by substituting a manually constructured solution of the scalar into the governing equation. The MATLAB symbolic math toolbox is used.

$$\begin{aligned}\nabla^2 T = \{&25a^2\cos(ax)[y - \frac{1}{2}\cos(cx)] - 25ac\sin(ax)sin(cx) \\ &+ \frac{1}{2}c^2\cos(cx)[25\cos(ax) + 40\sin(by)] \\ &- 40b^2\sin(by)[y - \frac{1}{2}\cos(cx)] + 80b\cos(by)\}\end{aligned} \tag{A.1}$$

where,

$$a = \frac{7\pi}{4} \qquad b = \frac{4\pi}{3} \qquad c = \frac{\pi}{2} \tag{A.2}$$

APPENDIX B

Coefficients in the Spalart-Allmaras Turbulence Model

The coefficients in Eqn. 3.3 are listed below.

$$D_{\tilde{\nu}_{\text{eff}}} = \frac{\tilde{\nu} + \nu}{\sigma_{\nu_t}} \tag{B.1}$$

$$f_w = g\left[\frac{1 + c_{w_3}^6}{g^6 + c_{w_3}^6}\right]^{1/6} \tag{B.2}$$

$$g = r + c_{w_2}(r^6 - r) \tag{B.3}$$

$$r = \min\left[\frac{\tilde{\nu}}{\tilde{S}\kappa^2 y^2}, 10\right] \tag{B.4}$$

$$\tilde{S} = \max\left[\Omega + \frac{f_{v_2}\tilde{\nu}}{\kappa^2 y^2}, c_s\Omega\right] \tag{B.5}$$

where, $\Omega = \sqrt{2}|\mathbf{W}|$ is the magnitude of the vorticity and $\mathbf{W} = \frac{1}{2}\left[\nabla\mathbf{u} - (\nabla\mathbf{u})^T\right]$.

$$f_{v_2} = 1 - \frac{\chi}{1 + \chi f_{v_1}} \tag{B.6}$$

f_{v_1} is defined as Eqn. 3.5.

$$C_{w_1} = \frac{C_{b_1}}{\kappa^2} + \frac{1 + C_{b_2}}{\sigma_{\nu_t}} \tag{B.7}$$

Model coefficients are:

$$\begin{aligned} c_{b_1} = 0.1355 \quad c_{b_2} = 0.622 \qquad c_{w_2} = 0.3 \quad c_s = 0.3 \\ c_{w_3} = 2.0 \quad \sigma_{\nu_t} = \frac{2}{3} \qquad \kappa = 0.41 \end{aligned} \tag{B.8}$$

BIBLIOGRAPHY

BIBLIOGRAPHY

Abgrall, R., H. Beaugendre, and C. Dobrzynski (2014), An immersed boundary method using unstructured anisotropic mesh adaptation combined with level-sets and penalization techniques, *Journal of Computational Physics*, *257*, 83–101.

Alliez, P., S. Tayeb, and C. Wormser (2019), Cgal 5.0 - 3d fast intersection and distance computation, https://doc.cgal.org/latest/AABB_tree/index.html, accessed: 2019-12-23.

Angelidis, D., S. Chawdhary, and F. Sotiropoulos (2016), Unstructured cartesian refinement with sharp interface immersed boundary method for 3d unsteady incompressible flows, *Journal of Computational Physics*, *325*, 272–300.

Balaras, E. (2004), Modeling complex boundaries using an external force field on fixed cartesian grids in large-eddy simulations, *Computers & Fluids*, *33*(3), 375–404.

Balaras, E., S. Schroeder, and A. Posa (2015), Large-eddy simulations of submarine propellers, *Journal of Ship Research*, *59*(4), 227–237.

Beaudoin, M., and H. Jasak (2008), Development of a generalized grid interface for turbomachinery simulations with openfoam, in *Open source CFD International conference*, vol. 2.

Beyer, R. P., and R. J. LeVeque (1992), Analysis of a one-dimensional model for the immersed boundary method, *SIAM Journal on Numerical Analysis*, *29*(2), 332–364.

Bond, R. B., C. C. Ober, P. M. Knupp, and S. W. Bova (2007), Manufactured solution for computational fluid dynamics boundary condition verification, *AIAA journal*, *45*(9), 2224–2236.

Borazjani, I., L. Ge, and F. Sotiropoulos (2008), Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3d rigid bodies, *Journal of Computational physics*, *227*(16), 7587–7620.

Braza, M., P. Chassaing, and H. H. Minh (1986), Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder, *Journal of Fluid Mechanics*, *165*, 79–130.

Calderer, A., S. Kang, and F. Sotiropoulos (2014a), Level set immersed boundary method for coupled simulation of air/water interaction with complex floating structures, *Journal of Computational Physics*, *277*, 201–227.

Calderer, A., S. Kang, and F. Sotiropoulos (2014b), Level set immersed boundary method for coupled simulation of air/water interaction with complex floating structures, *Journal of Computational Physics*, *277*, 201–227.

Calhoun, D. (2002), A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions, *Journal of Computational Physics*, *176*(2), 231–275.

Carrica, P. M., R. V. Wilson, R. W. Noack, and F. Stern (2007), Ship motions using single-phase level set with dynamic overset grids, *Computers & fluids*, *36*(9), 1415–1433.

Choi, J.-I., R. C. Oberoi, J. R. Edwards, and J. A. Rosati (2007), An immersed boundary method for complex incompressible flows, *Journal of Computational Physics*, *224*(2), 757–784.

Constant, E., J. Favier, M. Meldi, P. Meliga, and E. Serre (2017), An immersed boundary method in OpenFOAM: verification and validation, *Computers & Fluids*, *157*, 55–72.

Crawford, J., and A. Birk (2015), Influence of inlet boundary conditions on simulations of an asymmetric diffuser with the v2f turbulence model, in *ASME Turbo Expo 2015: Turbine Technical Conference and Exposition*, pp. V02BT40A009–V02BT40A009, American Society of Mechanical Engineers.

Demirdžić, I. (2015), On the discretization of the diffusion term in finite-volume continuum mechanics, *Numerical Heat Transfer, Part B: Fundamentals*, *68*(1), 1–10.

Dommermuth, D. G., et al. (2007), An application of cartesian-grid and volume-of-fluid methods to numerical ship hydrodynamics, *Tech. rep.*, DTIC Document.

Fadlun, E., R. Verzicco, P. Orlandi, and J. Mohd-Yusof (2000), Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, *Journal of computational physics*, *161*(1), 35–60.

Ferziger, J. H., and M. Peric (2012), *Computational methods for fluid dynamics*, Springer Science & Business Media.

Frink, N. (1994), Recent progress toward a three-dimensional unstructured navier-stokes flow solver, in *32nd Aerospace Sciences Meeting and Exhibit*, p. 61.

Ghias, R., R. Mittal, and H. Dong (2007), A sharp interface immersed boundary method for compressible viscous flows, *Journal of Computational Physics*, *225*(1), 528–553.

Gilmanov, A., and F. Sotiropoulos (2005), A hybrid cartesian/immersed boundary method for simulating flows with 3d, geometrically complex, moving bodies, *Journal of Computational Physics*, *207*(2), 457–492.

Gilmanov, A., F. Sotiropoulos, and E. Balaras (2003), A general reconstruction algorithm for simulating flows with complex 3d immersed boundaries on cartesian grids, *Journal of Computational Physics*, *191*(2), 660–669.

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

Henshaw, W. D., and D. W. Schwendeman (2008), Parallel computation of three-dimensional flows using overlapping grids with adaptive mesh refinement, *Journal of Computational Physics*, *227*(16), 7469–7502.

Hibiki, T., and M. Ishii (2003), One-dimensional drift-flux model and constitutive equations for relative motion between phases in various two-phase flow regimes, *International Journal of Heat and Mass Transfer*, *46*(25), 4935–4948.

Hirt, C. W., and B. D. Nichols (1981), Volume of fluid (vof) method for the dynamics of free boundaries, *Journal of computational physics*, *39*(1), 201–225.

Kalitzin, G., and G. Iaccarino (2002), Turbulence modeling in an immersed-boundary rans method, *Annual Research Briefs*, pp. 415–426.

Kalitzin, G., G. Medic, G. Iaccarino, and P. Durbin (2005), Near-wall behavior of rans turbulence models and implications for wall functions, *Journal of Computational Physics*, *204*(1), 265–291.

Khadra, K., P. Angot, S. Parneix, and J.-P. Caltagirone (2000), Fictitious domain approach for numerical modelling of navier–stokes equations, *International journal for numerical methods in fluids*, *34*(8), 651–684.

Kim, W., S. Van, and D. Kim (2001), Measurement of flows around modern commercial ship models, *Experiments in fluids*, *31*(5), 567–578.

Kleefsman, T. (2005), Water impact loading on offshore structures, *A Numerical Study, EU Project No.: GRD1-2000-25656*.

Laboratory, C. S. H. (2020), https://cshl.engin.umich.edu/about.

Lai, M.-C., and C. S. Peskin (2000), An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, *Journal of Computational Physics*, *160*(2), 705–719.

Larsson, L., F. Stern, and M. Visonneau (2013), Cfd in ship hydrodynamicsâĂŤresults of the gothenburg 2010 workshop, in *MARINE 2011, IV International Conference on Computational Methods in Marine Engineering*, pp. 237–259, Springer.

Lee, J., J. Kim, H. Choi, and K.-S. Yang (2011), Sources of spurious force oscillations from an immersed boundary method for moving-body problems, *Journal of computational physics*, *230*(7), 2677–2695.

Lee, S.-J., H.-R. Kim, W.-J. Kim, and S.-H. Van (2003), Wind tunnel tests on flow characteristics of the kriso 3,600 teu containership and 300k vlcc double-deck ship models, *Journal of Ship Research*, *47*(1), 24–38.

Lin, S.-Y., and Y.-C. Chen (2013), A pressure correction-volume of fluid method for simulations of fluid–particle interaction and impact problems, *International journal of multiphase flow*, *49*, 31–48.

Liu, C., X. Zheng, and C. Sung (1998), Preconditioned multigrid methods for unsteady incompressible flows, *Journal of Computational Physics*, *139*(1), 35–57.

Majumdar, S., G. Iaccarino, and P. Durbin (2001), Rans solvers with adaptive structured boundary non-conforming grids, *Annual Research Briefs, Center for Turbulence Research, Stanford University*, pp. 353–466.

Miao, G. (1989), Hydrodynamic forces and dynamic response of circular cylinders in wave zones, *University of Trondheim, Norway, Doctors Thesis, 1989-3*.

Mittal, R., and G. Iaccarino (2005), Immersed boundary methods, *Annu. Rev. Fluid Mech.*, *37*, 239–261.

Mittal, R., H. Dong, M. Bozkurttas, F. Najjar, A. Vargas, and A. von Loebbecke (2008), A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, *Journal of computational physics*, *227*(10), 4825–4852.

Mohd-Yusof, J. (1997), Combined immersed-boundary/b-spline methods for simulations of ow in complex geometries, *Annual Research Briefs. NASA Ames Research Center= Stanford University Center of Turbulence Research: Stanford*, pp. 317–327.

Moin, P., and K. Mahesh (1998), Direct numerical simulation: a tool in turbulence research, *Annual review of fluid mechanics*, *30*(1), 539–578.

NASA (2018a), https://www.grc.nasa.gov/www/wind/valid/buice/buice01/buice01.html.

NASA (2018b), https://turbmodels.larc.nasa.gov/flatplate.html.

Obi, S., K. Aoki, and S. Masuda (1993), Experimental and computational study of turbulent separating flow in an asymmetric plane diffuser, in *Ninth Symposium on Turbulent Shear Flows, Hyoto, Japan*, pp. 305–1.

Peskin, C. S. (1972), Flow patterns around heart valves: a numerical method, *Journal of computational physics*, *10*(2), 252–271.

Pinelli, A., I. Naqavi, U. Piomelli, and J. Favier (2010), Immersed-boundary methods for general finite-difference and finite-volume Navier–Stokes solvers, *Journal of Computational Physics*, *229*(24), 9073–9091.

Piro, D. J. (2013), A hydroelastic method for the analysis of global ship response due to slamming events., Ph.D. thesis, Department of Naval Architecture and Marine Engineering, University of Michigan.

Piziali, R. (1994), An experimental investigation of 2D and 3D oscillating wing aerodynamics for a range of angle of attack including stall, *NASA Technical Memorandum*.

Raad, P. E., and R. Bidoae (2005), The three-dimensional eulerian–lagrangian marker and micro cell method for the simulation of free surface flows, *Journal of Computational Physics*, *203*(2), 668–699.

Rosenfeld, M., D. Kwak, and M. Vinokur (1991), A fractional step solution method for the unsteady incompressible Navier-Stokes equations in generalized coordinate systems, *Journal of Computational Physics*, *94*(1), 102–137.

Russell, D., and Z. J. Wang (2003), A Cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow, *Journal of Computational Physics*, *191*(1), 177–205.

Sagaut, P. (2006), *Large eddy simulation for incompressible flows: an introduction*, Springer Science & Business Media.

Saiki, E., and S. Biringen (1996), Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method, *Journal of Computational Physics*, *123*(2), 450–465.

Sanders, J., J. E. Dolbow, P. J. Mucha, and T. A. Laursen (2011), A new method for simulating rigid body motion in incompressible two-phase flow, *International journal for numerical methods in fluids*, *67*(6), 713–732.

Shen, L., and E.-S. Chan (2008a), Numerical simulation of fluid–structure interaction using a combined volume of fluid and immersed boundary method, *Ocean Engineering*, *35*(8), 939–952.

Shen, L., and E.-S. Chan (2008b), Numerical simulation of fluid–structure interaction using a combined volume of fluid and immersed boundary method, *Ocean Engineering*, *35*(8-9), 939–952.

Shen, Z., and R. Korpus (2015), Numerical simulations of ship self-propulsion and maneuvering using dynamic overset grids in openfoam, in *Tokyo 2015 A Workshop on CFD in Ship Hydrodynamics. Presented at the Tokyo 2015 A Workshop on CFD in Ship Hydrodynamics, Tokyo, Japan*.

Shen, Z., D. Wan, and P. M. Carrica (2015), Dynamic overset grids in openfoam with application to kcs self-propulsion and maneuvering, *Ocean Engineering*, *108*, 287–306.

Spalart, P., and S. Allmaras (1992), A one-equation turbulence model for aerodynamic flows, in *30th aerospace sciences meeting and exhibit*, p. 439.

Spalding, D. (1961), A single formula for the law of the wall, *Journal of Applied Mechanics*, *28*(3), 455–458.

Stern, F., et al. (2011), Experience from simman 2008âĂŤthe first workshop on verification and validation of ship maneuvering simulation methods, *Journal of Ship Research*, *55*(2), 135–147.

Sun, X., and M. Sakai (2016), Numerical simulation of two-phase flows in complex geometries by using the volume-of-fluid/immersed-boundary method, *Chemical Engineering Science*, *139*, 221–240.

Sussman, M. (2001), An adaptive mesh algorithm for free surface flows in general geometries, in *Adaptive method of lines*, pp. 227–252, Chapman and Hall/CRC.

Tseng, Y.-H., and J. H. Ferziger (2003), A ghost-cell immersed boundary method for flow in complex geometry, *Journal of computational physics*, *192*(2), 593–623.

Wright, J. A., and R. W. Smith (2001), An edge-based method for the incompressible Navier–Stokes equations on polygonal meshes, *Journal of Computational Physics*, *169*(1), 24–43.

Xing, T., P. Carrica, and F. Stern (2008), Computational towing tank procedures for single run curves of resistance and propulsion, *Journal of fluids engineering*, *130*(10).

Y. Yoshimura, H. Y., Y. Fukui, and H. Yano (2013), Mathematical model for maneuvring simulation including roll motion, in *Conference Proc. JASNAOE*, vol. 16, pp. 17–20.

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

Yang, J., and F. Stern (2009), Sharp interface immersed-boundary/level-set method for wave–body interactions, *Journal of Computational Physics*, *228*(17), 6590–6616.

Yang, J., and F. Stern (2012), A simple and efficient direct forcing immersed boundary framework for fluid–structure interactions, *Journal of Computational Physics*, *231*(15), 5029–5061.

Ye, T., R. Mittal, H. Udaykumar, and W. Shyy (1999), An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries, *Journal of computational physics*, *156*(2), 209–240.

Zhang, C., N. Lin, Y. Tang, and C. Zhao (2014), A sharp interface immersed boundary/vof model coupled with wave generating and absorbing options for wave-structure interaction, *Computers & Fluids*, *89*, 214–231.

Zhang, Y., Q. Zou, D. Greaves, D. Reeve, A. Hunt-Raby, D. Graham, P. James, and X. Lv (2010), A level set immersed boundary method for water entry and exit, *Communications in Computational Physics*, *8*(2), 265–288.

Zhu, X., et al. (2007), Water entry and exit of a horizontal circular cylinder, *J. Offshore Mech. Arct. Eng*, *129*(4), 253–264.