Buoyancy-driven flow and fluid-structure interaction with moving boundaries

by

Songzhe Xu

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Program of Study Committee:
Baskar Ganapathysubramanian, Co-major Professor
Ming-Chen Hsu, Co-major Professor
Alberto Passalacqua
Wei Hong
Steven Hou

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2018

DEDICATION

To my lovely wife, Jiayu, and our dog, Elvis, and cat, Gouzi.

And to my Father and Mother.

For all of your support and company without which I couldn't finish this work.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to those who helped me in various aspects of conducting this research and the writing of this thesis. First and foremost, I would like to thank Dr. Baskar Ganapathysubramanian for his guidance, patience and support throughout this work. Besides the challenging and interesting projects, Dr. Ganapathysubramanian always shared his insights in the leading edge of related fields and guided me to more creative and thoughtful research ideas and topics in these fields. Secondly, I would like to thank Dr. Ming-Chen Hsu for his guidance in the first two chapters of this research. The deliberate discussions with Dr. Hsu and significant suggestions he offered clarified a lot of my doubts and made brilliant progress in these two chapters. I would also like to thank my committee members, Dr. Alberto Passalacqua, Dr. Wei Hong and Dr. Steven Hou, for their efforts and contributions to this work. I would like to additionally thank Dr. Hari Sundar for his guidance in the last chapter of this research. Dr. Sundar offered a lot of insightful ideas in computer science and valuable suggestions in the optimization of the framework we developed in this chapter. Finally, I would like to thank my colleagues, Alec Lofquist, Aditya Kommajosula and Fei Xu for their help to complete this research.

ABSTRACT

We deploy the residual-based variational multi-scale (VMS) method in the sense of large-eddy simulation (LES) in finite element method to buoyancy-driven flow in enclosures and consider an extensive range of Rayleigh number from laminar (10^3) to turbulent (10^{10}) in a 2D benchmark Rayleigh–Bénard problem. 3D simulations for a laminar and a turbulent case are performed and comparisons including mean profiles as well as fluctuation profiles with other numerical and experimental results are successfully carried out. A weakly imposed boundary conditions method is employed for both velocity and temperature, and it produces reasonable results with a much coarser mesh compared with the traditional imposition of boundary conditions. This suggests that the VMS framework with the weak imposition of boundary conditions is a computationally efficient approach to model buoyancy-driven flows in complex indoor environments.

In addition to the flow fields, we deploy the immersogeometric analysis (IMGA) method in the sense of the immersed boundary method (IBM) for objects moving in fluids onto an unstructured framework. The finite element formulation is stabilized by the VMS method in an unstructured background mesh. Weak imposition of boundary conditions is used to impose no-slip boundary condition on the immersed boundary. Adaptively refined quadrature rules are used to better capture the geometry of the immersed boundary and accurately integrate the background elements that intersect the immersed boundary. Treatment for the freshly-cleared nodes is considered. We assess the accuracy of the moving IMGA framework by analyzing object motion in a variety of flow structures, including freely dropping cylinder/sphere in viscous fluids and particle focusing in (un)obstructed channels. We show the quantities of interests are in good agreements with other analytical, numerical and experimental solutions. Advantages of this moving IMGA framework in computational cost and efficiency are indicated by the comparison with the body-fitted method using a commercial

computational fluid dynamic (CFD) software. The framework of moving IMGA is capable to be deployed in applications of particle control and manipulation in microfluidic channels.

The moving IMGA on the unstructured framework is further deployed to a scalable, adaptively refined, octree-based finite element approach for a better computational performance to track object motion. This enables using a parallel, hierarchically refined octree mesh as the background mesh, with a variationally consistent IMGA formulation on this background mesh. We integrate the unstructured framework of moving IMGA to the octree-based framework. We show good scaling results of the coupled framework on Stampede2, TACC. This illustrates the potential of the moving IMGA on the coupled framework to efficiently track complex particles in flows.

CHAPTER 1. INTRODUCTION

1.1 Buoyancy-Driven Flows Using Finite Element Method

The finite element method is being increasingly used for computational fluid dynamics (CFD). Early work has shown that the standard Galerkin approach is unable to solve the Navier–Stokes equations in convection dominated problems [1]. Subsequent analysis (and comparisons with central difference schemes) resulted in improvements in one-dimensional problems via upwinding, while still had issues in multi-dimensions [2, 3]. The work of Brooks and Hughes [1] showed that the Streamline-upwind Petrov Galerkin (SUPG) method overcomes these issues and yields a reasonable solution of the Navier–Stokes equation. Since velocity and pressure have differentials of unequal orders in the Navier–Stokes equation, it requires their basis functions also have different orders of differentiability. Pressure-stabilizing Petrov Galerkin (PSPG) overcomes this, and further allows the use of equal-order basis functions for velocity and pressure [4]. SUPG along with PSPG, as well as some other stabilizer methods, such as least-squares incompressibility constraint (LSIC) and Galerkin least-square (GLS), has shown a good success in a wide range of flow field until it becomes turbulent.

For turbulent flows, a more general formulation to stabilize the Navier–Stokes equations, residual-based variational multi-scale method (VMS), has been proposed and shown success to resolve turbulent flows [5]. The basic idea is to split the spatial domain (similar to the idea of large-eddy simulation (LES)) into a coarse scale represented by the finite element discretization, and a fine scale modeled by the residuals of the coarse scale. The terms incorporating fine scale variables serve as stabilizers that compensate the domain discretization, and can be interpreted as the combination of the classic SUPG/PSPG stabilizer and the turbulence modeling of flow fields.

An additional challenge with the solution of the Navier–Stokes equation is the imposition of the no-slip boundary conditions at walls. This strong imposition (i.e. Dirichlet conditions) of the no-slip

conditions, which explicitly specifies the Dirichlet values on the walls, results in the requirement of a highly refined mesh at the boundaries to resolve the steep velocity gradients. In this context, the weak imposition of boundary conditions in the sense of Nitsche's method [6] has been shown to produce reasonable results for isothermal flows [7–9]. Instead of the traditional way, the weak imposition of boundary conditions essentially applies the boundary traction (flux) to the fluid. As a result, it doesn't struggle to resolve the steep velocity gradient and allows relaxation of the mesh in the boundary layer, which is especially important for turbulent flows.

VMS with the weak imposition of boundary conditions gives the advantage to solve turbulent naturally occurring (buoyancy-driven) flows in indoor environments, which draws growing interests in high fidelity simulations. Naturally occurring flows could be utilized to reduce energy consumption in built environments. Accurately capturing the heat transfer produced by the buoyancy-driven flow is essential to calculate energy usage, where RANS models have been shown with problems [10–15], while LES methods are able to make good predictions [16–19]. Since VMS belongs to the LES family, and the weak imposition of boundary conditions can decrease the mesh density in the boundary layer, it is promising to deploy them to buoyancy-driven flows.

1.2 Particle Focusing in Fluids with Application to Microfluidics

In addition to flow fields, control and localization of particles (cells, precipitates) in aqueous flow are useful in biological processing, chemical reaction control, and for creating structured materials. The controlled motion and localization of cells and particles can automate cellular sample preparation and bio-sensing. Some examples include fast identification of *e. coli* in water, robust removal of circulating tumor cells from the blood plasma and fast separation of cells types for rapid flow cytometry and subsequent identification/tagging for genomic analysis. The precise, efficient and cheap localization of a heterogeneous collection of cells in a fluid medium is a foundational challenge in science and engineering. Most current approaches to particle localization in microfluidic devices is predominantly *active*, i.e., some external stimuli (electric field, permeate flow, stirrer) is used to create flow conditions that encourage particle separation and localization. However, ac-

tive control of particles in microfluidic devices results in device designs that are potentially more expensive, with multiple moving parts that can fail more frequently, and that require operation and transport in controlled environments. This translates to reduced applicability of state-of-art biomedical devices by reducing shelf-life, especially in remote areas and/or making robust devices too expensive for large scale use, especially in the global south. A general (computationally informed) strategy for passive control of particle localization in microfluidic channels will be transformative to this field.

In this context, the immersed boundary method (IBM) [20] – which embeds a solid object into a background Cartesian mesh of fluid without conforming them to each other – is a computationally convenient approach. The traditional body-fitted method has to deform the background mesh because of the object motion which may cause the mesh severely distorted, and frequent cumbersome remeshing is further required which is even more difficult for complex geometries. IBM doesn't suffer from these issues. One major challenge of IBM is to impose no-slip boundary condition on the immersed surface. Since the interface of the object and fluid does not align with the nodes in the background mesh, the traditional way to explicitly enforce Dirichlet boundary conditions on boundary nodes is non-trivial. There are typically two ways to impose the boundary conditions for an immersed boundary [21]. (1), continuous forcing approach: representing the effect of the immersed boundary as a forcing term added onto the Navier–Stokes equation and transmitting the boundary force to surrounding flow [22]. (2), discrete forcing approach: introducing ghost nodes (in the object domain) into the discretized system and interpolating the ghost nodes using the prescribed boundary conditions on the immersed boundary so that the immersed boundary conditions are implicitly incorporated [23]. While IBM shows great flexibility in solving complex boundary problems, it typically suffers from the reduced accuracy of the solution near the immersed boundary, which is related to the capture of the immersed geometry in the background mesh. To resolve these issues, an immersogeometric analysis (IMGA) method has been proposed and shown success for steady objects [24]. In this thesis, we extend this approach to account for moving objects.

The detailed simulations of object migration in micro-channels are still computational expensive within traditional unstructured framework for realistic 3D problems due to the necessity to solve the full fluid-structure coupling and associated small time steps and adaptive (re)meshing requirements. This calls for the development of an efficient, scalable, adaptively refined approach for particle tracking in fluids. We further deploy the moving IMGA to an octree-based framework in this work.

1.3 Dissertation structure

This Dissertation is organized as follows. In chapter 2, we deploy the VMS formulation to buoyancy-driven flow with a wide range of Rayleigh number. Weak imposition of boundary conditions is also deployed to the heat equation, and we highlight its capability to substantially decrease the mesh density in the boundary layer for a turbulent case. In Chapter 3, we develop and implement the IMGA for moving objects in a variety of flow structures. We illustrate its advantage regarding the computational cost and efficiency compared with the body-fitted method using the commercial CFD software Fluent. In Chapter 4, we integrate the unstructured moving IMGA framework to the octree-based adaptive meshing framework. We demonstrate the scaling studies of the coupled framework. Finally, we draw conclusions and motivate future work in Chapter 5.

1.4 References

[1] A. N. Brooks and T. J. R. Hughes. Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations. *Computer methods in applied mechanics and engineering*, 32(1-3):199–259, 1982.

[2] G. D. Raithby. A critical evaluation of upstream differencing applied to problems involving fluid flow. *Computer Methods in Applied Mechanics and Engineering*, 9(1):75–103, 1976.

[3] G. D. Raithby and K. E. Torrance. Upstream-weighted differencing schemes and their application to elliptic problems involving fluid flow. *Computers & Fluids*, 2(2):191–206, 1974.

[4] T. E. Tezduyar, S. Mittal, S. E. Ray, and R. Shih. Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements. *Computer Methods in Applied Mechanics and Engineering*, 95(2):221–242, 1992.

[5] Y. Bazilevs, V. M. Calo, J. A. Cottrell, T. J. R. Hughes, A. Reali, and G. Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 197(1):173–201, 2007.

[6] J. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 36:9–15, 1971.

[7] Y. Bazilevs and T. J. R. Hughes. Weak imposition of Dirichlet boundary conditions in fluid mechanics. *Computers & Fluids*, 36:12–26, 2007.

[8] Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes. *Computer Methods in Applied Mechanics and Engineering*, 199:780–790, 2010.

[9] Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Weak Dirichlet boundary conditions for wall-bounded turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 196(49):4853–4862, 2007.

[10] N. C. Markatos and K. A. Pericleous. Laminar and turbulent natural convection in an enclosed cavity. *International Journal of Heat and Mass Transfer*, 27(5):755–772, 1984.

[11] R. A. W. M. Henkes, F. F. Van Der Vlugt, and C. J. Hoogendoorn. Natural-convection flow in a square cavity calculated with low-reynolds-number turbulence models. *International Journal of Heat and Mass Transfer*, 34(2):377–388, 1991.

[12] A. M. Lankhorst and C. J. Hoogendoorn. Numerical computation of high rayleigh number natural convection and prediction of hot radiator induced room air motion. *Applied Scientific Research*, 47(4):301–322, 1990.

[13] G. Barakos, E. Mitsoulis, and D. Assimacopoulos. Natural convection flow in a square cavity revisited: laminar and turbulent models with wall functions. *International Journal for Numerical Methods in Fluids*, 18(7):695–719, 1994.

[14] K. J. Hsieh and F. S. Lien. Numerical modeling of buoyancy-driven turbulent flows in enclosures. *International Journal of Heat and Fluid Flow*, 25(4):659–670, 2004.

[15] S. Tieszen, A. Ooi, P. Durbin, and M. Behnia. Modeling of natural convection heat transfer. In *Proceedings of the Summer Program*, pages 287–302, 1998.

[16] J. Salat, S. Xin, P. Joubert, A. Sergent, F. Penot, and P. Le Quéré. Experimental and numerical investigation of turbulent natural convection in a large air-filled cavity. *International Journal of Heat and Fluid Flow*, 25(5):824–832, 2004.

[17] S.-H. Peng and L. Davidson. Large eddy simulation for turbulent buoyant flow in a confined cavity. *International Journal of Heat and Fluid Flow*, 22(3):323–331, 2001.

[18] A. Sergent, P. Joubert, and P. Le Quéré. Development of a local subgrid diffusivity model for large-eddy simulation of buoyancy-driven flows: application to a square differentially heated cavity. *Numerical Heat Transfer: Part A: Applications*, 44(8):789–810, 2003.

[19] C. Van Treeck, E. Rank, M. Krafczyk, J. Tölke, and B. Nachtwey. Extension of a hybrid thermal lbe scheme for large-eddy simulations of turbulent convective flows. *Computers & Fluids*, 35(8):863–871, 2006.

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

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

[22] C. S. Peskin. The fluid dynamics of heart valves: experimental, theoretical, and computational methods. *Annual review of fluid mechanics*, 14(1):235–259, 1982.

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

[24] F. Xu, D. Schillinger, D. Kamensky, V. Varduhn, C. Wang, and M.-C. Hsu. The tetrahedral finite cell method for fluids: Immersogeometric analysis of turbulent flow around complex geometries. *Computers & Fluids*, 141:135–154, 2016.

CHAPTER 2. A RESIDUAL-BASED VARIATIONAL MULTISCALE METHOD WITH WEAK IMPOSITION OF BOUNDARY CONDITIONS FOR INCOMPRESSIBLE BUOYANCY-DRIVEN CONVECTION PROBLEMS

A paper submitted to *Computer Methods in Applied Mechanics and Engineering*

Songzhe Xu, Ming-Chen Hsu and Baskar Ganapathysubramanian

Abstract

There is growing interest in high fidelity simulations of buoyancy driven flows in indoor environments. This is driven to a large extent by the push to utilize naturally occurring flows to reduce energy consumption in the built environment. Naturally occurring (buoyancy-driven) flows exhibit spatiotemporal thermal fluctuations, and accurately capturing these thermal variations is essential to calculate energy usage. In this work we deploy a residual-based variational multiscale (VMS) finite element LES model for accurately modeling buoyancy-driven flows in enclosed environments. We use the canonical example of the Rayleigh–Bénard convection problem in 2D and 3D to verify and validate the proposed VMS model. We show good comparison with benchmark numerical and experimental results across seven orders of magnitude variation in Rayleigh numbers ($Ra \sim 10^3$ to 10^{10}), covering both laminar, transition and turbulent regimes, without any extra treatments. We additionally employ weak enforcement of Dirichlet boundary conditions for both velocity and temperature, and show that comparable results can be produced with much coarser meshes. This suggests that the VMS framework with weak imposition of Dirichlet boundary conditions is a computationally efficient approach to model buoyancy-driven flow physics in complex indoor environments.

2.1 Introduction

Accurate simulation of coupled heat and momentum transport in incompressible fluids is essential in a variety of engineering applications. One critical application is in building simulations, which involves understanding and controlling the thermal and flow physics in complex, enclosed domains. There has been increasing interest in understanding and engineering the coupled thermal and momentum transport in inhabited domains so as to simultaneously increase the efficiency with which buildings use energy, while maintaining indoor comfort. Lower energy consumption reduces the impact of buildings on human health and the environment, reduces greenhouse gas emissions, enables the integration of onsite renewables, and lowers the economic hurdles to home ownership. According to the U.S. Energy Information Administration (EIA), even incremental improvements in energy efficiency have a significant impact on the U.S. energy budget, since buildings are responsible for approximately 40% of the total U.S. energy consumption [1].

One promising approach towards enhancing the energy sustainability of buildings has been to integrate passive natural ventilation (or buoyancy-driven flows) into the building environmental control system [2–7]. Such an approach requires a detailed understanding of how buoyancy-driven flows affect, and are affected by, different thermal boundary conditions. This is an area where robust and reliable computational tools have been scarce, especially for the building design community [8].

Standard computational fluid dynamics (CFD) approaches based on Reynolds-Averaged Navier–Stokes (RANS) models have been shown not to work well for this class of problems [9–14][1] and require site-specific and application-specific models [8]. In particular, most RANS-based models are unable to reliably predict boundary heat transfer coefficients (or Nusselt numbers) across the necessarily large range of Rayleigh numbers (Ra) that occur during building operation. In even comparatively simple geometries, the variation in thermal boundary conditions can result in Rayleigh

[1] This is often the case due to rapid variations in thermal boundary conditions caused by fluctuations in wind loads, as well as variations in incident solar radiation. Additionally, indoor spaces exhibit localized regions of laminar, transition and turbulent behavior. RANS based approaches have difficulty tracking this without *a priori* information, which is usually not available.

numbers spanning from 10^3 (laminar) to 10^{10} (turbulent), thus making it necessary for any method used to reliably and automatically predict Nusselt numbers across this range of flow conditions.

Recent results suggest that a more high-fidelity approach using Large-Eddy Simulation (LES) would enable accurately accounting for the effects of natural ventilation [15–17]. Large-eddy simulation based approaches has been applied for buoyancy-driven flow [18–21] with success. This is particularly promising with the increase in availability of (as well as ease of access, and use of) high-performance computing resources that make such simulations possible. Motivated by the need to reliably model thermal transport in complex enclosed domains, we propose and implement a finite element LES model based on variational multiscale (VMS) method [22–24] for buoyancy-driven flow. The VMS approach uses variational projections in place of the traditional filtered equations in LES and focuses on modeling the fine-scale equations. The method derives completely from the incompressible Navier–Stokes and heat equations and does not employ any eddy viscosities. Such VMS formulations have shown significant success in modeling turbulent flows [24,25] and continue to be successfully applied to a wide range of engineering applications [26–28].

In this work, our proposed VMS formulation is augmented with weakly enforced essential boundary conditions for buoyancy-driven flow. The weak enforcement of Dirichlet boundary conditions [29] improves the accuracy of simulations of flows with thin boundary layers, as in the case of flows in the built environment. Such a strategy releases the point-wise no-slip condition imposed at the boundary of the fluid domain and instead allows the flow to slip on the solid surface, thus minimizing the mesh resolution required to track the steep gradients close to the boundaries. Note that this effect reliably imitates the presence (and effect) of the thin boundary layer [30,31]. Enforcing Dirichlet boundary conditions weakly allows for an accurate overall flow solution even if the mesh size in the wall-normal direction is relatively large. This approach has substantially benefited efficient simulations of turbulent flow scenarios as demonstrated in [30, 32].

This paper is outlined as follows. In Section 2.2, we develop the formulations of VMS method and the weak imposition of essential boundary conditions for both velocity and temperature for the simulation of buoyancy-driven flows. Section 2.3 illustrates the numerical implementation of the

proposed method. Section 2.4 verifies and validates the VMS formulation and shows its ability to produce accurate results for an extensive range of Ra from 10^3 to 10^{10} for both 2D and 3D cases. The advantage of applying weakly enforce Dirichlet boundary conditions in buoyancy-driven flow simulations is also demonstrated in this section. Section 2.5 draws conclusions and outlines the potential of applying the proposed method in the future work of indoor simulations for complex buildings.

2.2 Variational multiscale formulation and discretization

2.2.1 Strong and weak formulations of the continuous problem

In what follows, $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, denotes the spatial domain of the problem with boundary Γ. The Navier–Stokes equations for incompressible flow may be written on Ω as

$$\frac{\partial \mathbf{u}}{\partial t} + \boldsymbol{\nabla} \cdot (\mathbf{u} \otimes \mathbf{u}) = -\frac{1}{\rho}\boldsymbol{\nabla} p + \nu \boldsymbol{\nabla}^2 \mathbf{u} + \mathbf{f}\,(\mathbf{u}, T)\,, \tag{2.1}$$

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0\,, \tag{2.2}$$

where $\mathbf{u}$ is the velocity, $\mathbf{f}$ is the forcing function (velocity-temperature coupling), p is the pressure, ρ is the fluid density, ν is the kinetic viscosity, T is the temperature, and $\boldsymbol{\nabla}$ is the spatial gradients. The energy equation may be written on Ω as

$$\frac{\partial T}{\partial t} + \boldsymbol{\nabla} \cdot (\mathbf{u} T) = \alpha \boldsymbol{\nabla}^2 T\,, \tag{2.3}$$

where α is the thermal diffusivity. Based on our application scenario, we assume the Boussinesq approximation for the velocity-temperature coupling, thus

$$\mathbf{f} = -\mathbf{a}\beta\,(T - T_r)\,, \tag{2.4}$$

where $\mathbf{a}$ is the gravitational acceleration, β is the thermal expansion, and T_r is the reference temperature. The problem (2.1)–(2.4) is subject to suitable boundary conditions, defined on the domain boundary, $\Gamma = \Gamma^{\mathrm{D}} \cup \Gamma^{\mathrm{N}}$ with $\Gamma^{\mathrm{D}} = \Gamma^{\mathrm{D}}_{\mathbf{u}} \cup \Gamma^{\mathrm{D}}_{T}$ and $\Gamma^{\mathrm{N}} = \Gamma^{\mathrm{N}}_{\mathbf{u}} \cup \Gamma^{\mathrm{N}}_{T}$:

$$\mathbf{u} = \mathbf{u}_{\mathbf{g}} \qquad \text{on } \Gamma^{\mathrm{D}}_{\mathbf{u}}\,, \tag{2.5}$$

$$T = T_{\mathrm{g}} \qquad \text{on } \Gamma_T^{\mathrm{D}}\,, \tag{2.6}$$

$$-\frac{p}{\rho}\,\mathbf{n} + \nu\,\boldsymbol{\nabla}\mathbf{u}\cdot\mathbf{n} = \mathbf{h}_{\mathbf{u}} \qquad \text{on } \Gamma_{\mathbf{u}}^{\mathrm{N}}\,, \tag{2.7}$$

$$\alpha\boldsymbol{\nabla}T\cdot\mathbf{n} = \mathrm{h}_T \qquad \text{on } \Gamma_T^{\mathrm{N}}\,, \tag{2.8}$$

where $\mathbf{u}_g$ and T_g are prescribed velocity and temperature at the Dirichlet boundaries $\Gamma_{\mathbf{u}}^{\mathrm{D}}$ and Γ_T^{D}, respectively, $\mathbf{h}_{\mathbf{u}}$ and h_T are given functions at the Neumann boundaries $\Gamma_{\mathbf{u}}^{\mathrm{N}}$ and Γ_T^{N}, respectivrly, and $\mathbf{n}$ is the unit normal vector.

Let $\mathcal{V}$ be the space of both trial solutions and test functions. The variational formulation is stated as: Find $\{\mathbf{u}, p, T\} \in \mathcal{V}$ such that $\forall\ \{\mathbf{w}, q, l\} \in \mathcal{V}$,

$$B\left(\{\mathbf{w}, q, l\}, \{\mathbf{u}, p, T\}\right) - F\left(\{\mathbf{w}, q, l\}, \{\mathbf{u}, p, T\}\right) = 0\,, \tag{2.9}$$

where

$$\begin{aligned} B\left(\{\mathbf{w}, q, l\}, \{\mathbf{u}, p, T\}\right) = &\int_\Omega \mathbf{w}\cdot\frac{\partial\mathbf{u}}{\partial t}\,\mathrm{d}\Omega - \int_\Omega \boldsymbol{\nabla}\mathbf{w} : (\mathbf{u}\otimes\mathbf{u})\,\mathrm{d}\Omega + \int_\Omega \boldsymbol{\nabla}\mathbf{w}\cdot\nu\boldsymbol{\nabla}\mathbf{u}\,\mathrm{d}\Omega \\ &- \int_\Omega \frac{p}{\rho}\boldsymbol{\nabla}\cdot\mathbf{w}\,\mathrm{d}\Omega + \int_\Omega q\boldsymbol{\nabla}\cdot\mathbf{u} \\ &+ \int_\Omega l\,\frac{\partial T}{\partial t}\,\mathrm{d}\Omega - \int_\Omega \boldsymbol{\nabla}l\cdot(\mathbf{u}T) + \int_\Omega \boldsymbol{\nabla}l\cdot\alpha\boldsymbol{\nabla}T\,\mathrm{d}\Omega\,, \end{aligned} \tag{2.10}$$

and

$$\begin{aligned} F\left(\{\mathbf{w}, q, l\}, \{\mathbf{u}, p, T\}\right) = &\int_\Omega \mathbf{w}\cdot\mathbf{f}\,\mathrm{d}\Omega \\ &- \int_{\Gamma_{\mathbf{u}}^N} (\mathbf{w}\cdot\mathbf{u})\,\mathbf{u}\cdot\mathbf{n}\,\mathrm{d}\Gamma + \int_{\Gamma_{\mathbf{u}}^N} \mathbf{w}\cdot\mathbf{h}_{\mathbf{u}}\,\mathrm{d}\Gamma - \int_{\Gamma_T^N} lT\mathbf{u}\cdot\mathbf{n}\,\mathrm{d}\Gamma + \int_{\Gamma_T^N} l\,\mathrm{h}_T\,\mathrm{d}\Gamma. \end{aligned} \tag{2.11}$$

Note that $\mathbf{f}$ is a function of T.

2.2.2 Semi-discrete variational multiscale formulation

Extending the variational multiscale theory proposed by [24] to the buoyancy-driven convection problem, the space of trial solution and weighting function is split into coarse and fine scales as $\mathcal{V} = \mathcal{V}^h \oplus \mathcal{V}'$, where the superscript h denotes resolved coarse scales represented by the finite

element discretization and the primed quantity corresponds to the unresolved scales that need to be modeled. The decomposition of the space leads to

$$\{\mathbf{u}, p, T\} = \{\mathbf{u}^h, p^h, T^h\} + \{\mathbf{u}', p', T'\}\,, \tag{2.12}$$

$$\{\mathbf{w}, q, l\} = \{\mathbf{w}^h, q^h, l^h\} + \{\mathbf{w}', q', l'\}\,. \tag{2.13}$$

Substituting Eq. (2.12) into Eq. (2.9) and choosing $\{\mathbf{w}, q, l\} = \{\mathbf{w}^h, q^h, l^h\}$ yields

$$B\left(\{\mathbf{w}^h, q^h, l^h\}, \{\mathbf{u}^h, p^h, T^h\} + \{\mathbf{u}', p', T'\}\right) - F\left(\{\mathbf{w}^h, q^h, l^h\}, \{\mathbf{u}^h, p^h, T^h\} + \{\mathbf{u}', p', T'\}\right) = 0\,. \tag{2.14}$$

Because $\{\mathbf{w}^h, q^h, l^h\}$ are in a finite-dimensional space, Eq. (2.14) leads to a finite-dimensional system of equations for which the coarse scale variables $\{\mathbf{u}^h, p^h, T^h\}$ are the unknowns. The variational statement (2.14) indicates that the coarse scale equations depend on the fine-scale fields. The fine scales $\{\mathbf{u}', p', T'\}$ are not given and their effect in the coarse-scale equations must, therefore, be modeled. To simplify Eq (2.14), three assumptions are typically employed: (1) the fine scales are orthogonal to the coarse scales with respect to the inner-product generated by the viscous term; (2) the fine scales are quasi-static; and (3) the fine scales variables are zero at the domain boundary.

We decompose the domain Ω into a collection of N_{el} disjoint elements each denoted by Ω^e, $\Omega = \bigcup_{e=1}^{N_{\text{el}}} \Omega^e$. We follow the developments in [24] and arrive at the semi-discrete variational multi-scale formulation for the buoyancy-driven convection problem as: Find $\{\mathbf{u}^h, p^h, T^h\} \in \mathcal{V}^h$ such that $\forall \{\mathbf{w}^h, q^h, l^h\} \in \mathcal{V}^h$,

$$B^{\text{VMS}}\left(\{\mathbf{w}^h, q^h, l^h\}, \{\mathbf{u}^h, p^h, T^h\}\right) - F^{\text{VMS}}\left(\{\mathbf{w}^h, q^h, l^h\}, \{\mathbf{u}^h, p^h, T^h\}\right) = 0\,, \tag{2.15}$$

where

$$\begin{aligned}
&B^{\text{VMS}}\left(\{\mathbf{w}^h, q^h, l^h\}, \{\mathbf{u}^h, p^h, T^h\}\right) \\
&= \int_\Omega \mathbf{w}^h \cdot \frac{\partial \mathbf{u}^h}{\partial t}\,\mathrm{d}\Omega + \int_\Omega \mathbf{w}^h \cdot \left(\mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{u}^h\right)\mathrm{d}\Omega + \int_\Omega \boldsymbol{\nabla}\mathbf{w}^h \cdot \nu \boldsymbol{\nabla}\mathbf{u}^h\,\mathrm{d}\Omega \\
&- \int_\Omega \frac{p^h}{\rho}\boldsymbol{\nabla}\cdot\mathbf{w}^h\,\mathrm{d}\Omega + \int_\Omega q^h \boldsymbol{\nabla}\cdot\mathbf{u}^h \\
&+ \int_\Omega l^h \frac{\partial T^h}{\partial t}\,\mathrm{d}\Omega + \int_\Omega l^h \left(\mathbf{u}^h \cdot \boldsymbol{\nabla} T^h\right)\mathrm{d}\Omega + \int_\Omega \boldsymbol{\nabla} l^h \cdot \alpha \boldsymbol{\nabla} T^h\,\mathrm{d}\Omega
\end{aligned}$$

$$
\begin{aligned}
&- \sum_{e=1}^{N_{\mathrm{el}}} \int_{\Omega^e} \left(\mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{w}^h + \boldsymbol{\nabla} q^h \right) \cdot \mathbf{u}' \, \mathrm{d}\Omega - \sum_{e=1}^{N_{\mathrm{el}}} \int_{\Omega^e} \left(\mathbf{u}^h \cdot \boldsymbol{\nabla} l^h \right) T' \, \mathrm{d}\Omega - \sum_{e=1}^{N_{\mathrm{el}}} \int_{\Omega^e} \frac{p'}{\rho} \boldsymbol{\nabla} \cdot \mathbf{w}^h \, \mathrm{d}\Omega \\
&- \sum_{e=1}^{N_{\mathrm{el}}} \int_{\Omega^e} \mathbf{w}^h \cdot \left(\mathbf{u}' \cdot \boldsymbol{\nabla} \mathbf{u}^h \right) \mathrm{d}\Omega - \sum_{e=1}^{N_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\nabla} \mathbf{w}^h : \left(\mathbf{u}' \otimes \mathbf{u}' \right) \mathrm{d}\Omega \\
&- \sum_{e=1}^{N_{\mathrm{el}}} \int_{\Omega^e} l^h \left(\mathbf{u}' \cdot \boldsymbol{\nabla} T^h \right) \mathrm{d}\Omega - \sum_{e=1}^{N_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\nabla} l^h \cdot \left(\mathbf{u}' T' \right) \mathrm{d}\Omega \,, \qquad (2.16)
\end{aligned}
$$

and

$$
\begin{aligned}
&F^{\mathrm{VMS}} \left(\{ \mathbf{w}^h, q^h, l^h \}, \{ \mathbf{u}^h, p^h, T^h \} \right) \\
&= \int_{\Omega} \mathbf{w}^h \cdot \mathbf{f}^h \, \mathrm{d}\Omega + \sum_{e=1}^{N_{\mathrm{el}}} \int_{\Omega^e} \mathbf{w}^h \cdot \mathbf{f}' \, \mathrm{d}\Omega \,. + \int_{\Gamma_{\mathbf{u}}^N} \mathbf{w}^h \cdot \mathbf{h}_{\mathbf{u}} \, \mathrm{d}\Gamma + \int_{\Gamma_T^N} l^h \, \mathrm{h}_T \, \mathrm{d}\Gamma. \qquad (2.17)
\end{aligned}
$$

In the above, $\mathbf{f}' = -\mathbf{a}\beta T'$. Bazilevs et al. [24] proposed solutions of the fine scale variables as a linear approximation based on the residuals of the coarse-scale equations for isothermal flows. Extending the same approximation to the energy equation, we have

$$
\mathbf{u}' = -\tau_{\mathrm{M}} \boldsymbol{r}_{\mathrm{M}} \left(\{ \mathbf{u}^h, p^h, T^h \} \right) , \qquad (2.18)
$$

$$
p' = -\tau_{\mathrm{C}} r_{\mathrm{C}} \left(\mathbf{u}^h \right) , \qquad (2.19)
$$

$$
T' = -\tau_{\mathrm{E}} r_{\mathrm{E}} \left(\{ \mathbf{u}^h, T^h \} \right) , \qquad (2.20)
$$

where

$$
\boldsymbol{r}_{\mathrm{M}} = \frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h + \frac{1}{\rho} \boldsymbol{\nabla} p^h - \nu \boldsymbol{\nabla}^2 \mathbf{u}^h - \mathbf{f}^h, \qquad (2.21)
$$

$$
r_{\mathrm{C}} = \boldsymbol{\nabla} \cdot \mathbf{u}^h, \qquad (2.22)
$$

$$
r_{\mathrm{E}} = \frac{\partial T^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} T^h - \alpha \boldsymbol{\nabla}^2 T^h, \qquad (2.23)
$$

$$
\tau_{\mathrm{M}} = \left(\frac{4}{\Delta t^2} + \mathbf{u}^h \cdot \mathbf{G} \mathbf{u}^h + C_M \nu^2 \mathbf{G} : \mathbf{G} \right)^{-1/2} , \qquad (2.24)
$$

$$
\tau_{\mathrm{C}} = \left(\tau_{\mathrm{M}} \mathbf{g} \cdot \mathbf{g} \right)^{-1} , \qquad (2.25)
$$

$$
\tau_{\mathrm{E}} = \left(\frac{4}{\Delta t^2} + \mathbf{u}^h \cdot \mathbf{G} \mathbf{u}^h + C_E \alpha^2 \mathbf{G} : \mathbf{G} \right)^{-1/2} , \qquad (2.26)
$$

and

$$
G_{ij} = \frac{\partial \xi_k}{\partial x_i} \frac{\partial \xi_k}{\partial x_j}, \qquad (2.27)
$$

$$\mathbf{G} : \mathbf{G} = G_{ij} G_{ij}, \tag{2.28}$$

$$g_i = \sum_{j=1}^{3} \frac{\partial \xi_j}{\partial x_i}, \tag{2.29}$$

$$\mathbf{g} \cdot \mathbf{g} = g_i g_i. \tag{2.30}$$

In the above, C_M and C_E are positive constants derived from an appropriate element-wise inverse estimate [33, 34], and $\mathbf{G}$ and $\mathbf{g}$ are mesh-dependent quantities related to the mapping from physical elements to the iso-parametric element.

The terms in the second, third and fourth line in Eq. 2.16 make up the standard Galerkin form, $B\left(\{\mathbf{w}^h, q^h, l^h\}, \{\mathbf{u}^h, p^h, T^h\}\right)$, of the Navier–Stokes equation and energy equation. Note, however, that we expand the coarse-scale convection terms in these equations into their convective form [2]. The fifth line incorporates the classic stabilization terms, such as streamline-upwind/Petrov–Galerkin (SUPG) and pressure-stabilizing/Petrov–Galerkin (PSPG). We also have the SUPG stabilization term in the energy equation. The last two lines incorporate additional terms produced by the VMS formulation [24]. The terms incorporating the fine scale variables added onto the standard Galerkin terms can be interpreted as the combination of the classic stabilization coupled with VMS turbulence modeling for the buoyancy-driven convection problem.

2.2.3 Weakly imposed boundary condition for both velocity and temperature

Weakly imposed boundary conditions in the sense of Nitsche's method [35] has been very successfully applied for isothermal flows [29, 30, 32]. Building upon the detailed interpretation of the formulation of the weakly imposed boundary conditions in [29], we extend the weak imposition to the two equation system here. Decomposing the domain boundary Γ into N_{eb} surface elements each denoted by Γ^{b}, we propose the semi-discrete formulation with the weak boundary conditions for the

[2] This is done by integrating by parts again for the second term in the second line (coarse-scale convection term in momentum equation) and the first term in the sixth line in Eq 2.16 as well as for the second term in the fourth line (coarse-scale convection term in energy equation) and the first term in the last line in Eq 2.16, and considering the full-scale continuity constrain $\boldsymbol{\nabla} \cdot \left(\mathbf{u}^h + \mathbf{u}'\right) = 0$. As a result, both coarse-scale and fine-scale boundary terms generated by the integral by part of convection terms in Eq. 2.11 vanish in Eq. 2.17 with the assumption (3)

buoyancy-driven convection problem as

$$
\begin{aligned}
&B^{\mathrm{VMS}}\left(\{\mathbf{w}^h, q^h, l^h\}, \{\mathbf{u}^h, p^h, T^h\}\right) - F^{\mathrm{VMS}}\left(\{\mathbf{w}^h, q^h, l^h\}\right) \\
&- \sum_{b=1}^{N_{\mathrm{eb}}} \int_{\Gamma^{\mathrm{b}} \bigcap \Gamma^{\mathrm{D}}_{\mathbf{u}}} \mathbf{w}^h \cdot \left(-\frac{p^h}{\rho}\,\mathbf{n} + \nu\,\boldsymbol{\nabla}\mathbf{u}^h \cdot \mathbf{n}\right) \mathrm{d}\Gamma - \sum_{b=1}^{N_{\mathrm{eb}}} \int_{\Gamma^{\mathrm{b}} \bigcap \Gamma^{\mathrm{D}}_{T}} l^h \alpha \boldsymbol{\nabla} T^h \cdot \mathbf{n}\, \mathrm{d}\Gamma \\
&- \sum_{b=1}^{N_{\mathrm{eb}}} \int_{\Gamma^{\mathrm{b}} \bigcap \Gamma^{\mathrm{D}}_{\mathbf{u}}} \left(\nu\,\boldsymbol{\nabla}\mathbf{w}^h \cdot \mathbf{n} + \frac{q^h}{\rho}\,\mathbf{n}\right) \cdot \left(\mathbf{u}^h - \mathbf{u}_g\right) \mathrm{d}\Gamma - \sum_{b=1}^{N_{\mathrm{eb}}} \int_{\Gamma^{\mathrm{b}} \bigcap \Gamma^{\mathrm{D}}_{T}} \alpha \boldsymbol{\nabla} l^h \cdot \mathbf{n} \left(T^h - T_g\right) \mathrm{d}\Gamma \\
&+ \sum_{b=1}^{N_{\mathrm{eb}}} \int_{\Gamma^{\mathrm{b}} \bigcap \Gamma^{\mathrm{D}}_{\mathbf{u}}} \tau^B_M \mathbf{w}^h \cdot \left(\mathbf{u}^h - \mathbf{u}_g\right) \mathrm{d}\Gamma + \sum_{b=1}^{N_{\mathrm{eb}}} \int_{\Gamma^{\mathrm{b}} \bigcap \Gamma^{\mathrm{D}}_{T}} \tau^B_E l^h \cdot \left(T^h - T_g\right) \mathrm{d}\Gamma \;=\; 0. \qquad (2.31)
\end{aligned}
$$

Note since we use the convective form for the coarse-scale convection terms, there are no boundary terms generated from them. The second line in Eq. (2.31) incorporates the consistency terms arising from the integral by parts of pressure and viscous terms. The third line incorporates the so-called adjoint terms to optimize the convergence. The fourth line incorporates the penalty-like terms to help satisfy the prescribed Dirichlet boundary conditions. It has been noted in previous work that the penalty-like stabilization parameters τ^B_M and τ^B_E cannot be too large, because in that case the formulation would behave simply like the traditional strong boundary conditions, thus losing the advantage of the implementation. Interestingly, the terms should also not be too small due to stability considerations. Earlier work [29] proposed τ^B_M to have the following form $\tau^B_M = \mathrm{C}^B_M \nu / h$, where h is the wall-normal size of the boundary elements, and C^B_M is a positive constant. Similarly, we propose $\tau^B_E = \mathrm{C}^B_E \alpha / h$, with C^B_E having the same definition as C^B_M.

2.3 Numerical method and implementation

2.3.1 Time discretization and averaging over time

We employ a finite difference based fully implicit backward Euler scheme for the discretization of the equations in time. The time step Δt is selected using the CFL condition. We define the set of variables as $\mathbf{U} = \{\mathbf{u}^h, p^h, T^h\}$. The time-averaged statistics of $\mathbf{U}_a$ in the quasi-steady state can be expressed as $\mathbf{U}_a = \frac{\int_{t_i}^{t} \mathbf{U}\,\mathrm{d}t}{t - t_i}$, where t_i is the time at which we start collecting data to compute the averages, and $t - t_i$ is a sufficiently long time interval. The fluctuations of the variables $\mathbf{U}_f$ is defined

as $\mathbf{U}_f = \mathbf{U} - \mathbf{U}_a$. We also average the product $\mathbf{U}_f\mathbf{U}_f$ in the same time interval to obtain the time-averaged statistics of the fluctuation quantities $\{\mathbf{U}_f\mathbf{U}_f\}_a$. We report $\mathbf{U}_a$ (mean profiles) and $\{\mathbf{U}_f\mathbf{U}_f\}_a$ (fluctuation profiles) to verify and validate our framework in Section 2.4.

2.3.2 Block iteration method

We employ the Newton–Raphson method, $\mathbf{J} \cdot \delta\mathbf{U} = -\mathbf{E}$, to solve the set of non-linear equations. $\mathbf{J}$ is the Jacobian matrix, $\mathbf{E}$ is the vector of function values, and $\delta\mathbf{U}$ is the vector of the increments of variables. We employ a block-iterative strategy to solve this set of equations (due to the very stiff nature of the fully coupled system). In block iteration, at each iteration, the temperature field in the Navier–Stokes equation and the velocity field in the energy equation are considered known fields. Thus, the coupling between the two equations is weakened. Note that with this decoupling, the energy equation becomes linear because the velocity field is known, and reduces to the following

$$\int_\Omega l^h \frac{\partial T^h}{\partial t}\,\mathrm{d}\Omega + \int_\Omega l^h \left(\mathbf{u}^h \cdot \boldsymbol{\nabla} T^h\right)\mathrm{d}\Omega + \int_\Omega \boldsymbol{\nabla} l^h \cdot \alpha \boldsymbol{\nabla} T^h\,\mathrm{d}\Omega - \sum_{e=1}^{N_{\mathrm{el}}} \int_{\Omega^e} \left(\mathbf{u}^h \cdot \boldsymbol{\nabla} l^h\right) T'\,\mathrm{d}\Omega = 0. \quad (2.32)$$

Thus, in the block-iteration strategy, we form two individual systems and solve them self-consistently at every time-step.

At each time-step we first solve the energy equation for the temperature field with velocity and pressure fields known from the last iteration (or time-step). Then we pass the temperature field to the Navier–Stokes equation to solve for the velocity and pressure field. We update every variable after this block iteration, and compute the L_2 error of the fields. We continue the self-consistent block iteration until the relative L_2 norm error for every variable field is less than 0.1%. The workflow chart is as Fig. 2.1

2.3.3 Non-dimensional form

We solve the equations in their non-dimensional form. This is done by scaling the variables as follows:

$$\mathbf{x}^* = \frac{\mathbf{x}}{L_0}, \qquad \mathbf{u}^* = \frac{\mathbf{u}}{u_0}, \qquad t^* = \frac{t}{t_0}, \qquad p^* = \frac{p}{\rho u_0^2}, \qquad \theta = \frac{T - T_r}{\Delta T}. \quad (2.33)$$

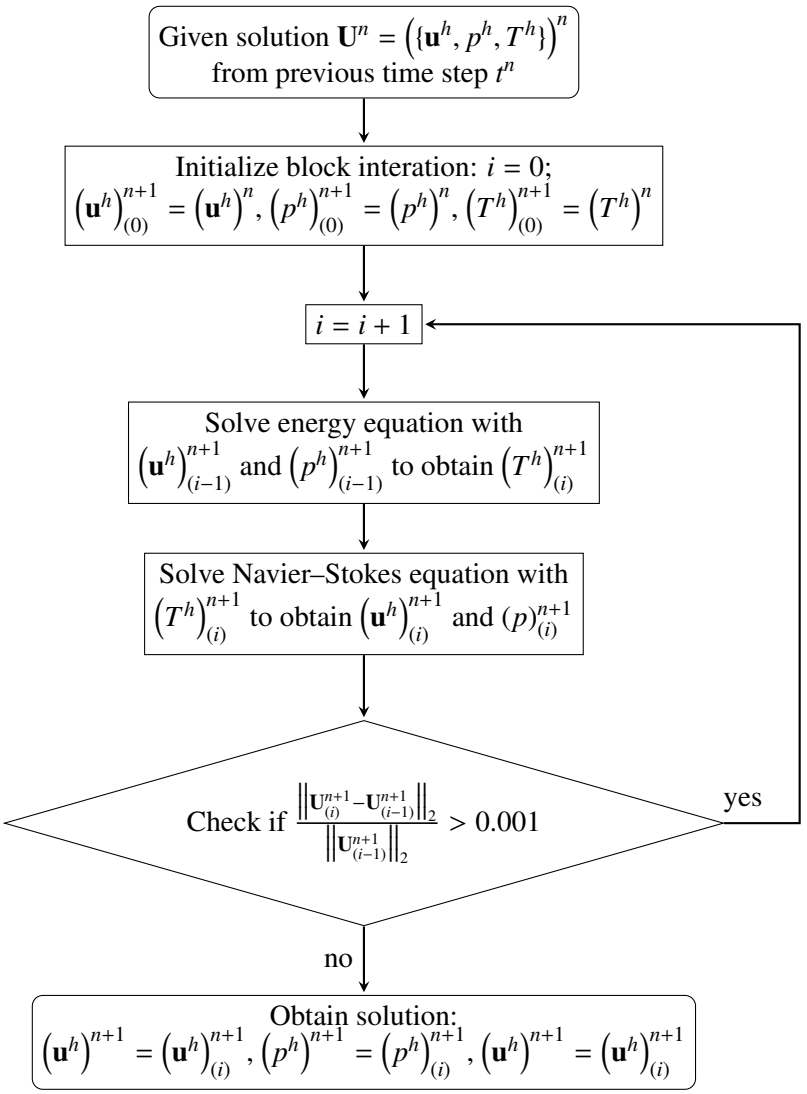


Figure 2.1: Flow chart of the block iteration for the Navier–Stokes and energy equation.

Here, L_0 is the characteristic length of the problem, u_0 is the characteristic velocity, $u_0 = (g\beta\Delta T L)^{1/2}$ for buoyancy-driven flow. t_0 is the characteristic time, $t_0 = L_0/u_0$. Pressure is scaled by ρu_0^2. ΔT is the temperature differential of the hottest temperature T_h and the coldest temperature T_c, $\Delta T = T_h - T_c$. θ is the dimensionless temperature. We choose the reference temperature T_r to be the coldest temperature T_c. This results in the following set of equations:

$$\frac{\partial \mathbf{u}^*}{\partial t^*} + \boldsymbol{\nabla}^* \cdot (\mathbf{u}^* \otimes \mathbf{u}^*) + \boldsymbol{\nabla}^* p^* - \sqrt{\frac{Pr}{Ra}} \boldsymbol{\nabla}^{*2} \mathbf{u}^* - \mathbf{f}^* = 0 \tag{2.34}$$

$$\boldsymbol{\nabla}^* \cdot \mathbf{u}^* = 0 \tag{2.35}$$

$$\frac{\partial \theta}{\partial t^*} + \boldsymbol{\nabla}^* \cdot (\mathbf{u}^* \theta) - \sqrt{\frac{1}{PrRa}} \boldsymbol{\nabla}^{*2} \theta = 0, \tag{2.36}$$

where Ra is the Rayleigh number, $Ra = g\beta\Delta T L^3/\nu\alpha$, and Pr is the Prandtl number, $Pr = \nu/\alpha$. $(\cdot)^*$ represents dimensionless quantities. $\boldsymbol{\nabla}^*$ is the dimensionless spacial gradient operator, $\boldsymbol{\nabla}^* = \frac{\partial}{\partial \mathbf{x}^*}$, and $\mathbf{f}^* = \mathbf{e}_g \theta$, where $\mathbf{e}_g$ is the unit vector pointing in the direction of gravity.

2.3.4 Computational method

We implement the variational multiscale method and the weakly imposed boundary condition method for buoyancy-driven flow within our in-house parallel finite element framework. We couple our framework with an open source package PETSC [36], to employ its scalable non-linear equation solvers (SNES) as well as its linear Krylov subspace solvers (KSP). Domain decomposition is performed with the parallel graphics partitioning package Parmetis [37].

2.4 Rayleigh–Bénard convection problem

The Rayleigh–Bénard convection phenomena serves as the canonical problem that we choose to model in this work. This serves as a very well studied problem, with extensive experimental and computational results available for comparison, as well as a typical representative of buoyancy driven flow in enclosed geometries that is our application thrust. We consider a (non-dimensional) Rayleigh–Bénard convection problem with temperature differential applied on the vertical walls. We

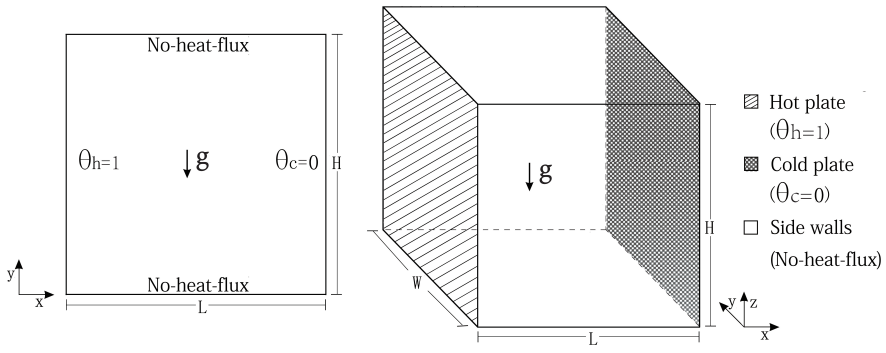


Figure 2.2: Geometry and boundary condition.

consider simulations in both 2D and 3D cases as shown in Fig. 2.2. No-slip boundary conditions are applied on all walls.

We first show 2D results with strongly imposed boundary condition. Mesh convergence studies are performed for an extensive range of Ra numbers from 10^3 to 10^{10} at $Pr = 0.7$. We choose the surface averaged Nusselt number $\overline{Nu}$ as the parameter of interest for the mesh convergence study. This is motivated from the building science application where the net heat transfer from surfaces (given by the Nusselt number) is a key quantity of interest. The local Nu and surface averaged $\overline{Nu}$ on the (hot) wall Γ_H with area A_{Γ_H} are defined as

$$Nu = \boldsymbol{\nabla}^* \theta \cdot \mathbf{n} \tag{2.37}$$

$$\overline{Nu} = \frac{\int_{\Gamma_H} \boldsymbol{\nabla}^* \theta \cdot \mathbf{n} d\Gamma}{A_{\Gamma_H}}. \tag{2.38}$$

Then we show 3D results with strongly imposed boundary conditions. Mesh convergence studies for a laminar case with $Ra = 1.89 \times 10^5$, and a turbulent case with $Ra = 1.5 \times 10^9$ are performed. The results of $\overline{Nu}$, mean profiles, and fluctuation profiles are compared with reported experimental results [18].

Finally, the weak imposition of Dirichlet boundary condition for both velocity and temperature is applied for the 2D cases. Mesh convergence studies and comprehensive comparisons with strongly imposed boundary condition are performed to illustrate the computational advantages of the weakly imposed boundary conditions.

2.4.1 2D case

2.4.1.1 Mesh convergence studies

We employ a unit square with a uniform mesh to report consistent results across a range of *Ra*. The results for 2D cases is shown in Table 2.1. We can see that for the laminar cases $\overline{Nu}$ rapidly converges. Even a coarse mesh is enough to resolve the thermal boundary layer for these cases. For higher *Ra*, a denser mesh is required. The convergence of $\overline{Nu}$ value is clear across the wide range of *Ra* from 10^3 to 10^{10}. We identify the converged value with an underline at each *Ra* when the error of $\overline{Nu}$ for two successive mesh densities is less than 2%.

Table 2.1: 2D mesh convergence results for $\overline{Nu}$.

	Ra (Laminar)			*Ra* (Transition)		*Ra* (Turbulent)		
Uniform mesh	10^3	10^4	10^5	10^6	10^7	10^8	10^9	10^{10}
50×50	1.117	2.240	4.488	8.585	14.99	22.78	28.90	33.15
100×100	1.118	2.244	4.511	8.772	16.15	27.91	43.23	57.66
200×200	1.118	2.245	4.516	8.810	16.44	29.67	51.18	81.22
400×400	-	-	-	-	-	30.1	53.81	93.40
600×600	-	-	-	-	-	-	54.29	96.63
800×800	-	-	-	-	-	-	-	97.29

2.4.1.2 Comparisons of averaged Nusselt numbers and maximum velocities

To validate the results of our solution, we compare the average Nusselt number $\overline{Nu}$, maximum horizontal velocity along the vertical median line, and maximum vertical velocity along the horizontal median line with previously reported computational results. We first compare our results with Refs [12,38] for the laminar region ($Ra \leq 10^6$). These comparisons are shown in Table 2.2 and illustrate excellent match with previous work.

We next compare our results with reported data for the turbulent cases with $Ra \geq 10^7$. Here, we compare our results with reported solutions that were computed using both RANS models [9, 12] and LES models [39,40] to indicate the difference between them, as shown in Table 2.3. We see that once *Ra* goes beyond 10^8, the RANS models start to produce increasingly diverging results from the more accurate LES models. Our results of $\overline{Nu}$ compare well with the LES models across the

Table 2.2: Comparisons of $\overline{Nu}$ and maximum velocities and locations along median lines from $Ra = 10^3$ to $Ra = 10^6$.

Ra		[38]	[12]	This work
10^3	$U_{max}(y)$	0.136(0.813)	0.153(0.806)	0.138(0.815)
	$V_{max}(x)$	0.138(0.178)	0.155(0.181)	0.139(0.180)
	$\overline{Nu}$	1.118	1.114	1.118
10^4	$U_{max}(y)$	0.192(0.823)	0.193(0.818)	0.194(0.825)
	$V_{max}(x)$	0.234(0.119)	0.234(0.119)	0.235(0.12)
	$\overline{Nu}$	2.243	2.245	2.245
10^5	$U_{max}(y)$	0.153(0.855)	0.132(0.859)	0.132(0.855)
	$V_{max}(x)$	0.261(0.066)	0.258(0.066)	0.259(0.065)
	$\overline{Nu}$	4.519	4.51	4.516
10^6	$U_{max}(y)$	0.079(0.850)	0.077(0.859)	0.078(0.850)
	$V_{max}(x)$	0.262(0.038)	0.262(0.039)	0.263(0.040)
	$\overline{Nu}$	8.799	8.806	8.810

complete range of Ra, as expected. We also compare the maximum velocities and corresponding locations in Table 2.4. Our results compare well with other reported LES results for $Ra = 10^7$ and $Ra = 10^8$. For $Ra = 10^9$, our results generally agree with [41] along the horizontal median line, but exhibit a discrepancy for the maximum horizontal velocity. We speculate that this could be due to an under-resolved boundary layer, which may result in different point-wise estimates even though the $\overline{Nu}$ shows converged behavior. To test this speculation, we use a clustered mesh that concentrates elements near the boundaries so that the boundary layer is well resolved (while using the same number of degree of freedoms). We employ a hyperbolic tangent stretching function to cluster the uniform mesh. The hyperbolic function is $\boldsymbol{x}_{new} = \frac{1}{2}\left(1 + \frac{\tanh\left(a(\frac{\mathbf{x}}{\boldsymbol{L}} - \frac{1}{2})\right)}{\tanh\left(\frac{a}{2}\right)}\right)$, where $\boldsymbol{x}$ is the original coordinate in the uniform mesh, $\boldsymbol{x}_{new}$ is the new coordinate in the stretching direction, $\boldsymbol{L}$ is the domain size in the corresponding stretching direction, and a (= 5) is the stretching factor. The stretched mesh is show in Figure 2.3. The comparison with the stretched mesh on the vertical median line is shown in Table 2.5. We can see that the stretched mesh produces results that are closer to the previously reported results.

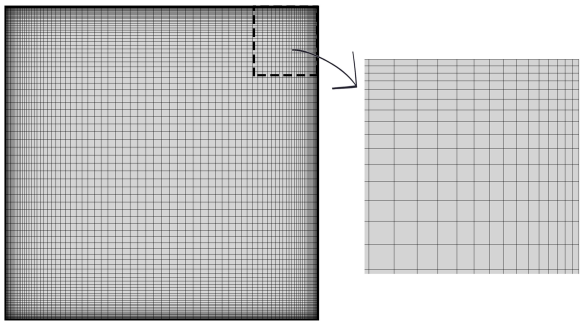


Figure 2.3: Stretched mesh.

Table 2.3: 2D comparisons of $\overline{Nu}$ from $Ra = 10^7$ to $Ra = 10^{10}$.

Ra	RANS [9]	RANS [12]	LES [39]	LES [40]	This work
10^7	16.47	-	17.2	16.76	16.49
10^8	32.05	32.3	31.2	30.43	30.1
10^9	74.96	60.1	58.1	51.25	54.29
10^{10}	156.85	134.6	-	99.96	97.29

Table 2.4: 2D comparisons of the maximum velocities and locations along median lines from $Ra = 10^7$ to $Ra = 10^{10}$.

Ra		[9]	[41]	[42]	[43]	This work
10^7	$U_{max}(Y)$	-	0.0621(0.851)	0.0562(0.879)	0.0548(0.92)	0.0564(0.878)
	$V_{max}(X)$	-	0.265(0.020)	0.264(0.021)	0.270(0.021)	0.264(0.023)
10^8	$U_{max}(Y)$	0.0615(0.941)	0.0466(0.937)	0.0385(0.928)	0.0353(0.94)	0.0399(0.930)
	$V_{max}(X)$	0.217(0.0135)	0.268(0.0112)	0.266(0.012)	0.274(0.013)	0.266(0.0125)
10^9	$U_{max}(Y)$	-	0.0190(0.966)	-	-	0.0270(0.937)
	$V_{max}(X)$	-	0.258(0.0064)	-	-	0.267(0.00667)
10^{10}	$U_{max}(Y)$	0.0278(0.9625)	0.0278(0.94)	-	-	0.0143(0.956)
	$V_{max}(X)$	0.202(0.0055)	0.257(0.49)	-	-	0.270(0.00375)

Table 2.5: 2D comparison of the maximum horizontal velocity and the location along the vertical median line for $Ra = 10^9$.

Ra		[41]	This work (uniform)	This work (stretching)
10^9	$U_{max}(Y)$	0.0190(0.966)	0.0270(0.937)	0.0207(0.943)

2.4.1.3 Comparisons of mean profiles

We perform comparison with ref [12] of the mean profiles for a laminar case with $Ra = 10^5$, and a turbulent case with $Ra = 10^9$. We plot the mean temperature and vertical velocity along the horizontal median line and the mean temperature and horizontal velocity along the vertical median line in Figure 2.4, and 2.5. We see that the overall comparisons for the laminar case are excellent (indicating that the VMS treatment essentially vanishes in the laminar case, when the full physics is accurately captured by the coarse scale). For the turbulent case, shown in Figure 2.5, the comparisons are good [3], except for the horizontal velocity along the vertical median line. We also plot the result from the stretched mesh as seen in Figure 2.5 (d). As anticipated, the stretched mesh improves the result. In the latter part of the results section, we show that using weak boundary conditions produces accurate results even when using a uniform mesh. We conclude this sub-section by plotting 2D contours of temperature and velocities of the laminar case and turbulent case in Figure 2.6.

2.4.2 3D case

2.4.2.1 Mesh convergence studies

We start this sub-section by performing mesh convergence studies for two cases: a laminar case with $Ra = 1.89 \times 10^5$, and a turbulent case with $Ra = 1.5 \times 10^9$. To compare with available results, the laminar case is simulated in a unit cubic cavity, while the turbulent case is simulated in a cuboid with an aspect ratio of $1 \times 1 \times 0.32$, which is identical to the experimental geometry reported in ref [18]. For the turbulent case, we employ the clustered mesh discussed earlier (with $a = 5$). We report convergence results of $\overline{Nu}$ in Table 2.6. We notice that, for both cases, the convergence of the surface averaged Nusselt number is rapid.

[3] Note that we compare our results at $Ra = 10^9$ with 2D LES results in ref [18] which is computed at $Ra = 1.5 \times 10^9$.

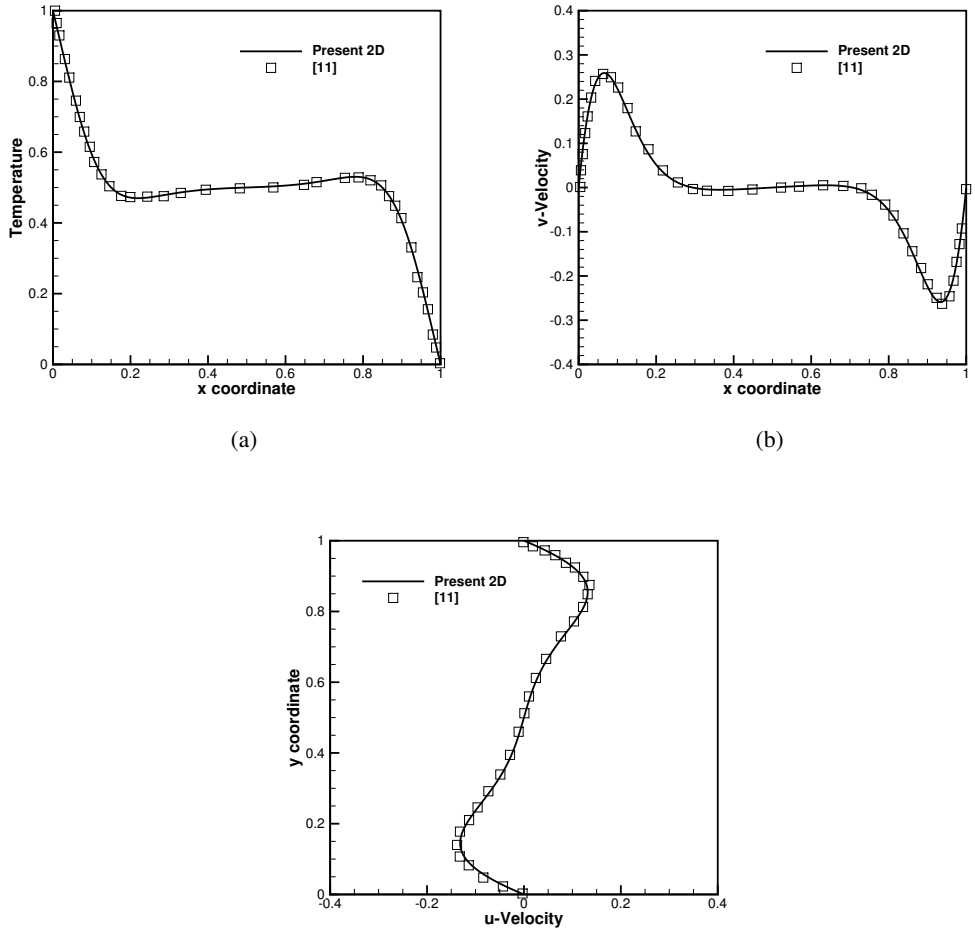


Figure 2.4: Mean profile comparisons of present 2D and [12] at $Ra = 10^5$. (a) Temperature along the horizontal median line. (b) Vertical velocity along the horizontal median line. (c) Horizontal velocity along the vertical median line.

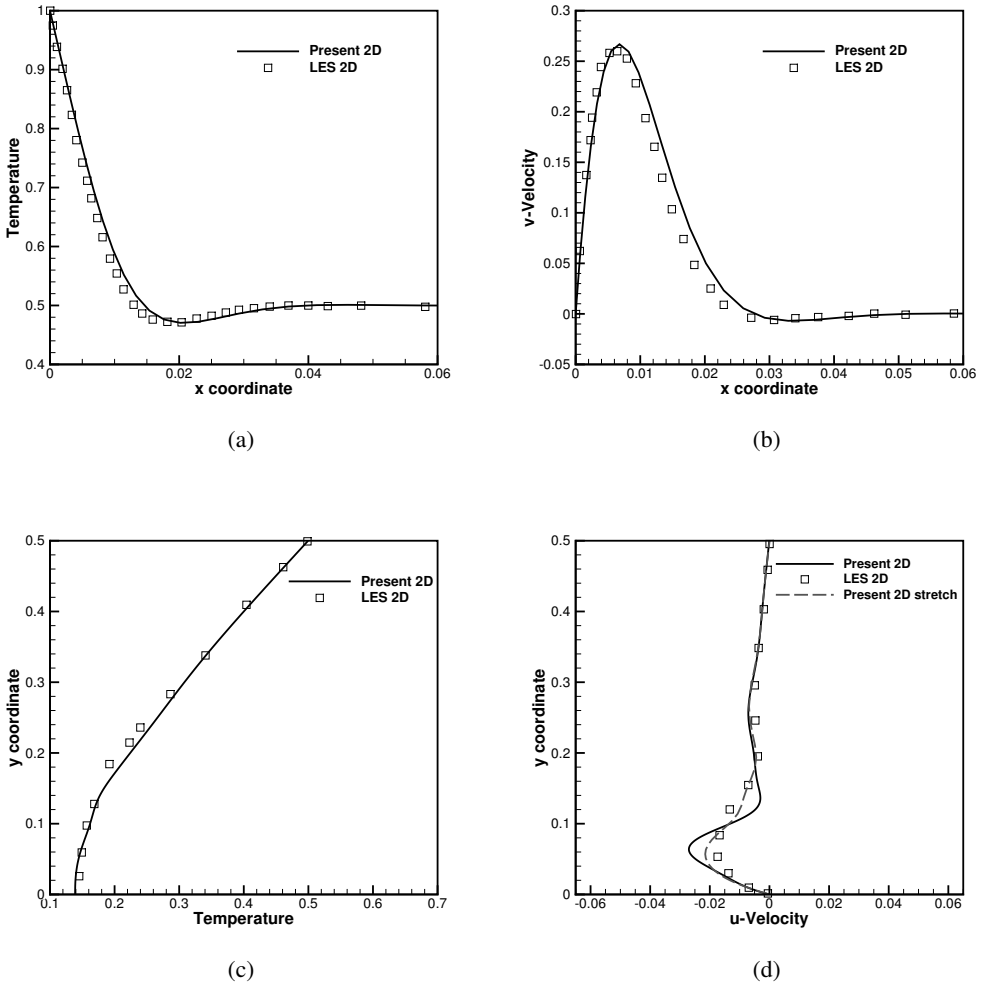
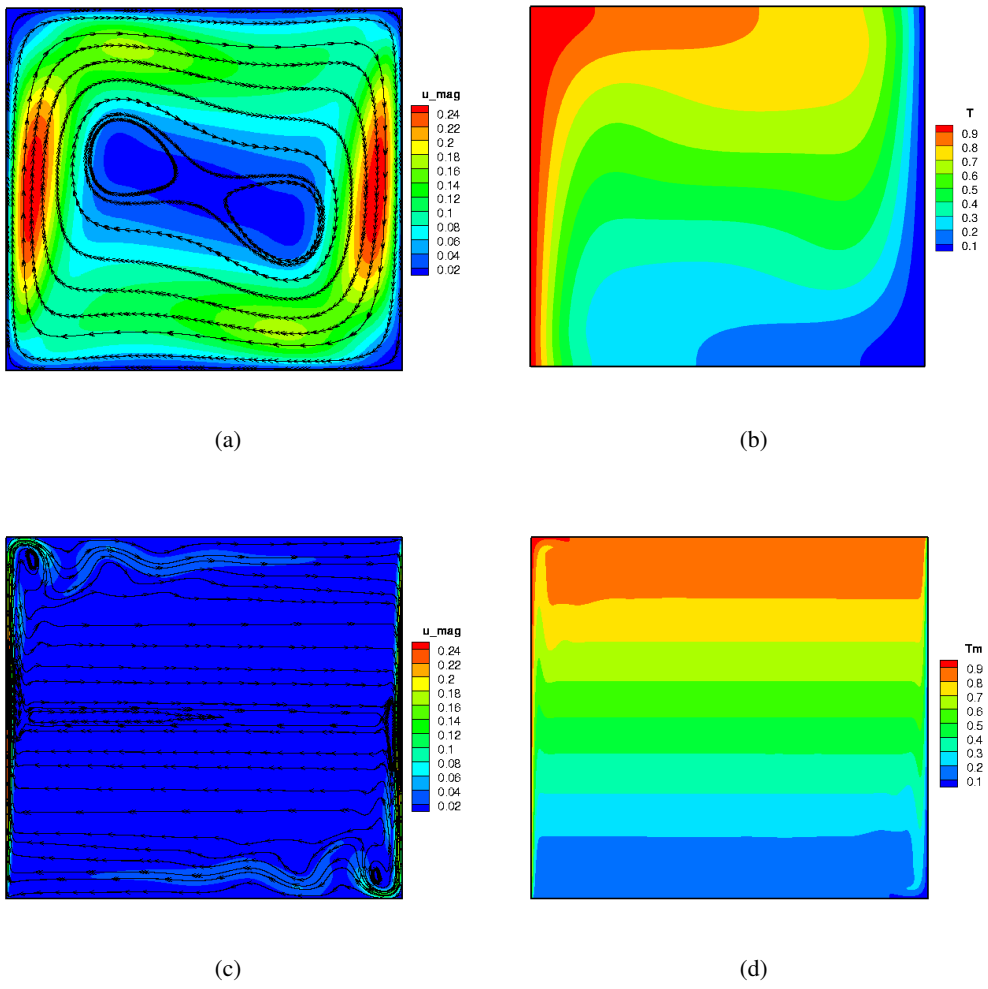


Figure 2.5: Mean profile comparisons for present 2D at $Ra = 10^9$, and 2D LES at $Ra = 1.5 \times 10^9$ in [18]. (a) Temperature along the horizontal median line. (b) Vertical velocity along the horizontal median line. (c) Temperature along the vertical median line. (d) Horizontal velocity along the vertical median line.



(a) (b)

(c) (d)

Figure 2.6: Visualizations of present 2D cases. (a) Streamline and velocity magnitude contour at $Ra = 10^5$ (b) Temperature contour at $Ra = 10^5$. (c) Streamline and velocity magnitude contour at $Ra = 10^9$ (d) Temperature contour at $Ra = 10^9$.

Table 2.6: 3D mesh convergence results for $\overline{Nu}$.

Mesh	$Ra = 1.89 \times 10^5$	$Ra = 1.5 \times 10^9$
$50 \times 50 \times 50$	5.221	60.2
$70 \times 70 \times 70$	5.252	60.7
$100 \times 100 \times 100$	5.265	60.8

Table 2.7: 3D comparisons of $\overline{Nu}$ for $Ra = 1.89 \times 10^5$.

	This work		[44]		[45]	
Ra	Overall	Mid-plane	Overall	Mid-plane	Overall	Mid-plane
1.89×10^5	5.265	5.535	5.252	5.537	5.31	5.71

2.4.2.2 Comparisons of averaged Nusselt numbers

We compare the converged value of $\overline{Nu}$ for the laminar case with available experimental and LES results in Table 2.7. We get good agreement with the numerical correlation result in [44], as well as the experimental results reported in [45]. For the turbulent case, we compare the $\overline{Nu}$ with the 3D DNS result reported in [18]. This comparison is shown in Table 2.8 and exhibit good match.

2.4.2.3 Comparisons of mean profiles

We perform mean profile comparisons for $Ra = 1.89 \times 10^5$ with numerical results in [44] and experimental results in [46]. We plot the mean temperature and vertical velocity along the horizontal median line, and mean temperature and horizontal velocity along the vertical median line, as shown in Figure 2.7. The overall comparisons are good as seen in Figure 2.7 (a), (b). Along the vertical direction, bigger discrepancies are found (between the numerical solutions and experimental result) for the temperature near the top and bottom walls as seen in Figure 2.7 (c). Note that both our numerical solution as well as the numerical solutions from [44] match but differ from the experimental results. This is because it is very difficult to physically maintain a perfect no-

Table 2.8: 3D comparison of $\overline{Nu}$ for $Ra = 1.5 \times 10^9$ on the median plane at the hot wall.

Ra	This work	DNS in [18]
1.5×10^9	60.8	60.1

heat-flux boundary condition in experiments, which can be clearly seen by the non-perpendicular slope to those walls in the experimental result [4]. For horizontal velocity along the vertical direction, the comparison is again good as seen in Figure 2.7 (d).

For the turbulent case of $Ra = 1.5 \times 10^9$, we impose the experimental values of temperature from [18] at top and bottom walls as Dirichlet boundary condition, and compare our results with the experimental results, as well as 3D DNS and 3D LES results with the same boundary condition in [18] as shown in Figure 2.8. We can see that in the horizontal direction, all the numerical results are nearly identical, but slightly underestimate the temperature compared to the experimental results. For vertical velocity, our 3D result matches the 3D LES result, with minor variations with the 3D DNS results and experimental results. In the vertical direction, all the numerical results are close, but differ from the experimental result for temperature. We speculate that this could again be due to the difficulty in maintaining a constant temperature at these large Ra numbers in the experiment. The horizontal velocity compares very well, with our 3D result matching both the experiment as well as the 3D DNS results.

2.4.2.4 Comparisons of fluctuation profiles

The comparisons of the fluctuation quantities with [18] are shown in Figure 2.9. Along the horizontal direction, our fluctuation results best match the experimental results for temperature as compared with other numerical results 2.9(a). The velocity fluctuations are reasonably captured in 2.9(b). Note, however, that the scale of these fluctuations is two orders of magnitude smaller than the corresponding fluctuations of the horizontal velocity component in the vertical direction in 2.9(d), which is well captured by our framework. Finally, the temperature fluctuations in the vertical direction are shown in 2.9(c), with all numerical results overestimating the magnitudes. In this case too, the magnitude of the temperature fluctuations is one order magnitude lower than that in the horizontal direction in 2.9(a). Overall, these results suggest that the fluctuations in both temperature and velocity are reasonably well captured by the VMS based approach.

[4] One way to fix this discrepancy is to specify the experimental values of temperature on top and bottom walls as Dirichlet boundary conditions

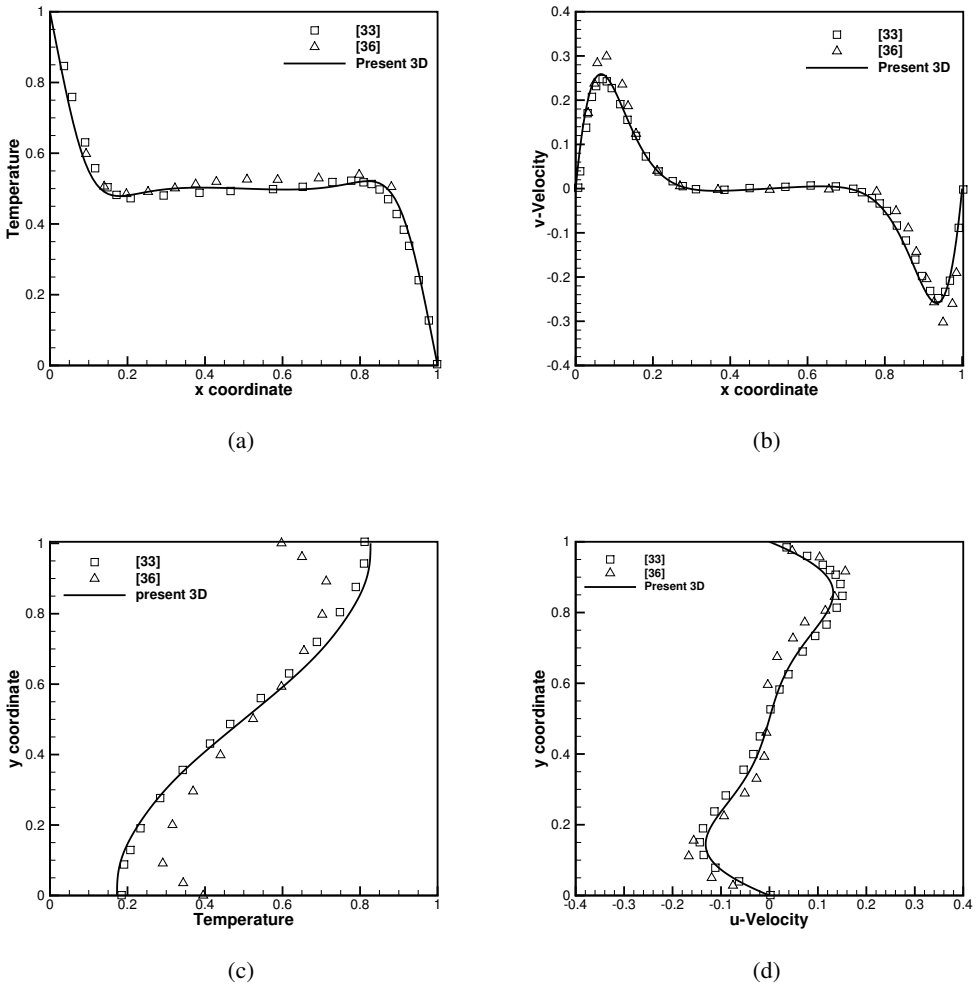


Figure 2.7: Mean profile comparisons for present 3D at $Ra = 1.89 \times 10^5$, numerical result at $Ra = 10^5$ in [44], and experimental result at $Ra = 1.89 \times 10^5$ in [46]. (a) Temperature along the horizontal median line. (b) Vertical velocity along the horizontal median line. (c) Temperature along the vertical median line. (d) Horizontal velocity along the vertical median line.

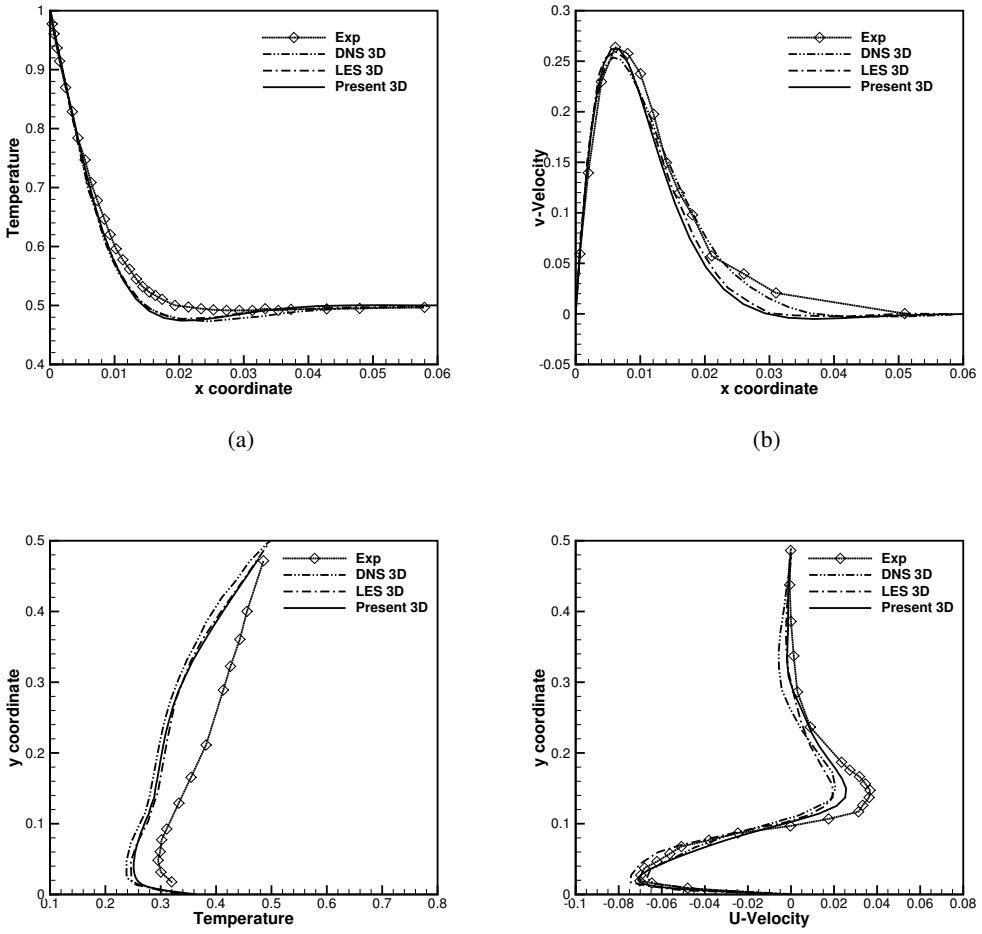


Figure 2.8: Mean profile comparisons for present 3D and [18] at $Ra = 1.5 \times 10^9$. (a) Temperature along the horizontal median line. (b) Vertical velocity along the horizontal median line. (c) Temperature along the vertical median line. (d) Horizontal velocity along the vertical median line.

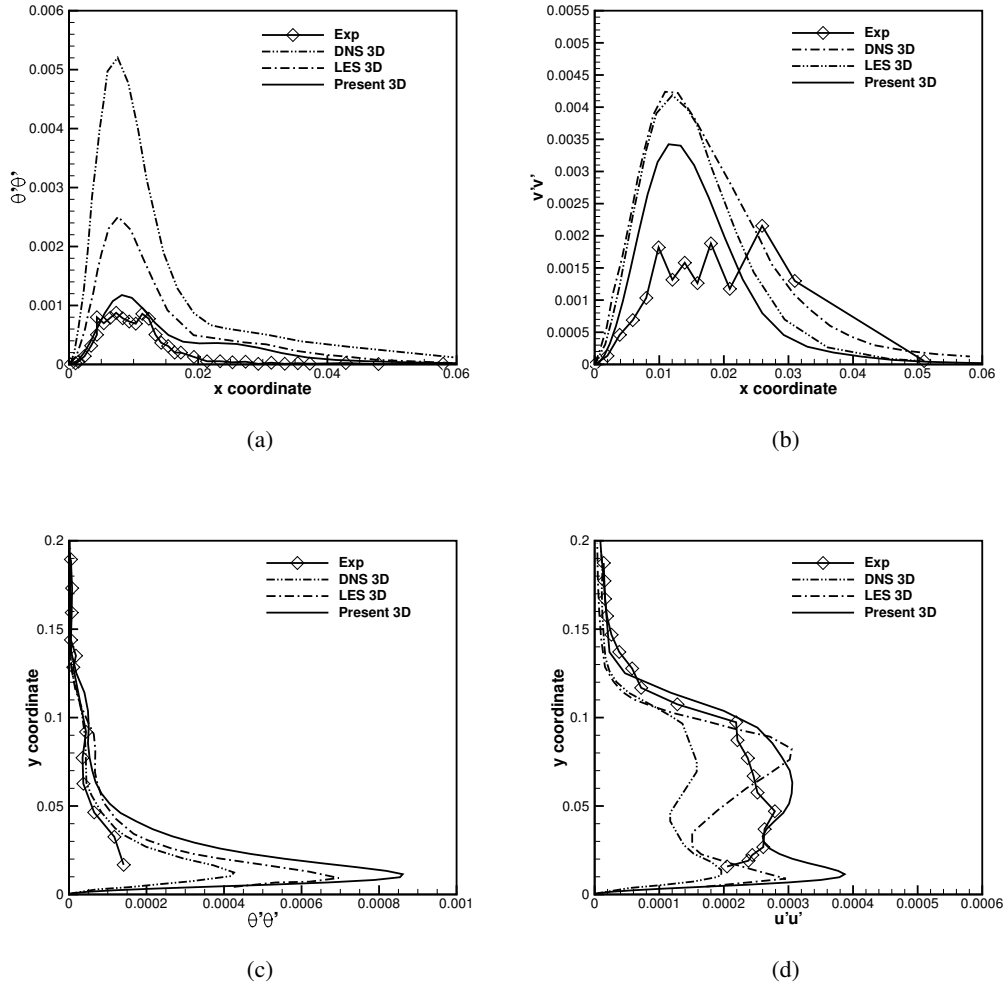


Figure 2.9: Fluctuation distribution comparison for present 3D and [18]. (a) Temperature along horizontal direction. (b) Vertical velocity along horizontal direction. (c) Temperature along vertical direction. (d) Horizontal velocity along vertical direction.

Table 2.9: 2D mesh convergence results for $\overline{Nu}$ with weak BC.

	Ra (Laminar)			*Ra* (Transition)		*Ra* (Turbulent)		
Uniform mesh	10^3	10^4	10^5	10^6	10^7	10^8	10^9	10^{10}
50×50	1.117	2.241	4.503	8.765	16.35	29.63	61.94	157.1
100×100	1.118	2.244	4.514	8.801	16.44	29.96	53.54	104.3
200×200	1.118	2.245	4.515	8.815	16.48	30.10	54.17	96.08
400×400	-	-	-	-	-	-	54.41	97.46

2.4.3 2D case with weakly imposed boundary condition

2.4.3.1 Mesh convergence studies

In this sub-section, we weakly impose Dirichlet boundary conditions (weak BC), and re-simulate the 2D cases for *Ra* from 10^3 to 10^{10} with a uniform mesh. Since the Prandtl number *Pr* is close to unity, the thickness of the thermal boundary layer will be comparable with that of the fluid boundary layer. As a result, we apply weak BC for both velocity and temperature. We compute $\overline{Nu}$ on the hot wall where weak BC is imposed as Eq. (2.39), following the same theory of traction computation for weak BC of velocity in [47].

$$\overline{Nu} = \frac{\int_{\Gamma_H} \boldsymbol{\nabla}^* \theta \cdot \mathbf{n} \, d\Gamma - \int_{\Gamma_H} \tau_E^{B^*} \cdot \left(\theta - \theta_g\right) \, d\Gamma}{A_{\Gamma_H}}, \tag{2.39}$$

where $\tau_E^{B^*}$ is now non-dimensionalized as $\tau_E^{B^*} = C_E^B / h^*$, and h^* and θ_g are the non-dimensional wall-normal element size and prescribed Dirichlet temperature at the hot wall respectively. The mesh convergence results for $\overline{Nu}$ are shown in Tab. 2.9. As expected, for the laminar cases, the weak BC performs similar to the strong BC (since the strong BC already performed well well at coarse meshes). However, for high *Ra*, weak imposition of Dirichlet BC shows significant advantages in accurately capturing the thermal boundary effects, even with coarse meshes. For example, the strong BC implementations required a mesh of 400×400 and 600×600 to produce reasonable $\overline{Nu}$ for $Ra = 10^9$ and $Ra = 10^{10}$ respectively, while weak BC only needs a mesh of 100×100 and 200×200 for these cases. These results strongly point to the computational advantages of weak imposition of Dirichlet boundary conditions.

Table 2.10: 2D comparisons of maximum velocities and locations along median lines for strong BC and weak BC at $Ra = 10^9$.

Ra		Strong BC	Weak BC
10^9	$U_{max}(Y)$	0.0270(0.937)	0.0191(0.93)
	$V_{max}(X)$	0.267(0.00667)	0.273(0.01)

2.4.3.2 Comparisons with strong BC for the turbulent case

We select a turbulent 2D case with $Ra = 10^9$, and perform detailed comparisons of weak BC with a uniform mesh density of 100×100 and strong BC with a uniform mesh density of 600×600. The maximum velocities and corresponding locations on median lines for strong BC and weak BC are shown in Table 2.10. We can see in the vertical direction, weak BC with the mesh density of 100×100 is able to fix the mismatch of the maximum horizontal velocity in strong BC, which is caused by the insufficient mesh density in the boundary layer even with the mesh density of 600×600. Similar results are observed in the horizontal direction.

Comparisons of mean profiles along median lines are shown in Figure 2.10. In the horizontal direction, since we only plot results within a thin layer 0.06 from the hot wall, the weak BC case only has a few points because of the mesh density. However, we can still observe a good agreement between strong BC and weak BC. In the vertical direction, both cases produce close results for temperature as seen in Figure 2.10(c). For the horizontal velocity, we remind the reader that we had to resort to a clustered mesh for the strong BC, but are able to get identical results with a uniform mesh for the weak BC case, as seen in Figure 2.10(d).

2.5 Conclusions

We have extended the variational multiscale method to buoyancy-driven flow, and verified and validated the framework with a Rayleigh–Bénard convection problem for both 2D and 3D cases. We show excellent comparisons for 2D across a wide range of Rayleigh numbers, without any special treatments. We also successfully compared simulation results with 3D experimental results as well as other LES results for both laminar and turbulent conditions. We also extended the weak

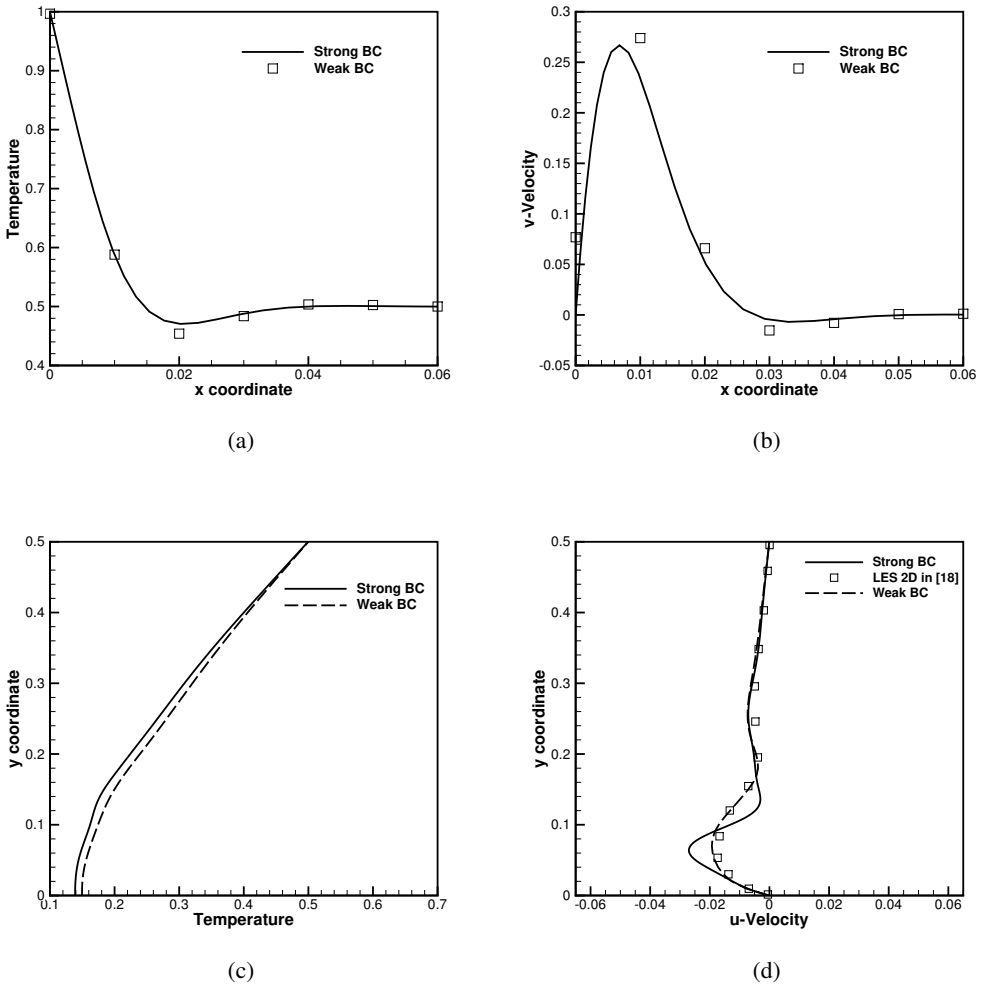


Figure 2.10: Mean profile comparisons of uniform mesh for 2D strong BC and weak BC at $Ra = 10^9$. Strong BC mesh: 600×600; Weak BC mesh: 100×100. (a) Temperature along the horizontal median line. (b) Vertical velocity along the horizontal median line. (c) Temperature along the vertical median line. (d) Horizontal velocity along the vertical median line.

imposition of boundary condition idea to the buoyancy-driven flow case. We are able to show the significant computational advantage of weak imposition of boundary conditions. This suggests that the VMS framework with weak imposition of Dirichlet boundary conditions is a computationally efficient approach to model buoyancy-driven flow physics in complex indoor environments. Our future work involves deploying this framework to complex indoor environments to study energy characteristics as well as contaminant transport in the built environment [48, 49].

2.6 References

[1] J. D. Kelso. Buildings energy data book. *US Dept. of Energy*, 2011.

[2] F. Oldewurtel, A. Parisio, C. N. Jones, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and M. Morari. Use of model predictive control and weather forecasts for energy efficient building climate control. *Energy and Buildings*, 45:15–27, 2012.

[3] S. Petersen and S. Svendsen. Method for simulating predictive control of building systems operation in the early stages of building design. *Applied energy*, 88(12):4597–4606, 2011.

[4] S. Privara, J. Cigler, Z. Váňa, F. Oldewurtel, C. Sagerschnig, and E. Žáčeková. Building modeling as a crucial part for building predictive control. *Energy and Buildings*, 56:8–22, 2013.

[5] S. Privara, J. Širokỳ, L. Ferkl, and J. Cigler. Model predictive control of a building heating system: The first experience. *Energy and Buildings*, 43(2):564–572, 2011.

[6] C. V. Rao and J. B. Rawlings. Linear programming and model predictive control. *Journal of Process Control*, 10(2):283–289, 2000.

[7] T. Salsbury, P. Mhaskar, and S. J. Qin. Predictive control methods to improve energy efficiency and reduce demand in buildings. *Computers & Chemical Engineering*, 51:77–85, 2013.

[8] M. Santamouris and F. Allard. *Natural ventilation in buildings: a design handbook*. Earthscan, 1998.

[9] N. C. Markatos and K. A. Pericleous. Laminar and turbulent natural convection in an enclosed cavity. *International Journal of Heat and Mass Transfer*, 27(5):755–772, 1984.

[10] R. A. W. M. Henkes, F. F. Van Der Vlugt, and C. J. Hoogendoorn. Natural-convection flow in a square cavity calculated with low-reynolds-number turbulence models. *International Journal of Heat and Mass Transfer*, 34(2):377–388, 1991.

[11] A. M. Lankhorst and C. J. Hoogendoorn. Numerical computation of high rayleigh number natural convection and prediction of hot radiator induced room air motion. *Applied Scientific Research*, 47(4):301–322, 1990.

[12] G. Barakos, E. Mitsoulis, and D. Assimacopoulos. Natural convection flow in a square cavity revisited: laminar and turbulent models with wall functions. *International Journal for Numerical Methods in Fluids*, 18(7):695–719, 1994.

[13] K. J. Hsieh and F. S. Lien. Numerical modeling of buoyancy-driven turbulent flows in enclosures. *International Journal of Heat and Fluid Flow*, 25(4):659–670, 2004.

[14] S. Tieszen, A. Ooi, P. Durbin, and M. Behnia. Modeling of natural convection heat transfer. In *Proceedings of the Summer Program*, pages 287–302, 1998.

[15] Z. Zhai. Application of computational fluid dynamics in building design: aspects and trends. *Indoor and built environment*, 15(4):305–313, 2006.

[16] Y. Li and P. V. Nielsen. Cfd and ventilation research. *Indoor Air*, 21(6):442–453, 2011.

[17] D. Etheridge. A perspective on fifty years of natural ventilation research. *Building and Environment*, 91(Supplement C):51 – 60, 2015. Fifty Year Anniversary for Building and Environment.

[18] J. Salat, S. Xin, P. Joubert, A. Sergent, F. Penot, and P. Le Quéré. Experimental and numerical investigation of turbulent natural convection in a large air-filled cavity. *International Journal of Heat and Fluid Flow*, 25(5):824–832, 2004.

[19] S.-H. Peng and L. Davidson. Large eddy simulation for turbulent buoyant flow in a confined cavity. *International Journal of Heat and Fluid Flow*, 22(3):323–331, 2001.

[20] A. Sergent, P. Joubert, and P. Le Quéré. Development of a local subgrid diffusivity model for large-eddy simulation of buoyancy-driven flows: application to a square differentially heated cavity. *Numerical Heat Transfer: Part A: Applications*, 44(8):789–810, 2003.

[21] C. Van Treeck, E. Rank, M. Krafczyk, J. Tölke, and B. Nachtwey. Extension of a hybrid thermal lbe scheme for large-eddy simulations of turbulent convective flows. *Computers & Fluids*, 35(8):863–871, 2006.

[22] T. J. R. Hughes, L. Mazzei, and K. E. Jansen. Large eddy simulation and the variational multiscale method. *Computing and Visualization in Science*, 3:47–59, 2000.

[23] T. J. R. Hughes, L. Mazzei, A. A. Oberai, and A. Wray. The multiscale formulation of large eddy simulation: Decay of homogeneous isotropic turbulence. *Physics of Fluids*, 13:505–512, 2001.

[24] Y. Bazilevs, V. M. Calo, J. A. Cottrell, T. J. R. Hughes, A. Reali, and G. Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 197(1):173–201, 2007.

[25] Y. Bazilevs and I. Akkerman. Large eddy simulation of turbulent Taylor–Couette flow using isogeometric analysis and the residual-based variational multiscale method. *Journal of Computational Physics*, 229:3402–3414, 2010.

[26] Y. Bazilevs, K. Takizawa, T. E. Tezduyar, M.-C. Hsu, N. Kostov, and S. McIntyre. Aerodynamic and FSI analysis of wind turbines with the ALE–VMS and ST–VMS methods. *Archives of Computational Methods in Engineering*, 21:359–398, 2014.

[27] K. Takizawa, Y. Bazilevs, T. E. Tezduyar, M.-C. Hsu, O. Øiseth, K. M. Mathisen, N. Kostov, and S. McIntyre. Engineering analysis and design with ALE–VMS and Space–Time methods. *Archives of Computational Methods in Engineering*, 21:481–508, 2014.

[28] K. Takizawa, Y. Bazilevs, T. E. Tezduyar, C. C. Long, A. L. Marsden, and K. Schjodt. ST and ALE-VMS methods for patient-specific cardiovascular fluid mechanics modeling. *Mathematical Models and Methods in Applied Sciences*, 24:2437–2486, 2014.

[29] Y. Bazilevs and T. J. R. Hughes. Weak imposition of Dirichlet boundary conditions in fluid mechanics. *Computers & Fluids*, 36:12–26, 2007.

[30] Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes. *Computer Methods in Applied Mechanics and Engineering*, 199:780–790, 2010.

[31] M.-C. Hsu, I. Akkerman, and Y. Bazilevs. Wind turbine aerodynamics using ale–vms: Validation and the role of weakly enforced boundary conditions. *Computational Mechanics*, 50(4):499–511, 2012.

[32] Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Weak Dirichlet boundary conditions for wall-bounded turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 196(49):4853–4862, 2007.

[33] C. Johnson. *Numerical solution of partial differential equations by the finite element method*. Cambridge University Press, Sweden, 1987.

[34] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods, 2nd ed.* Springer, 2002.

[35] J. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 36:9–15, 1971.

[36] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. Gropp, D. Kaushik, et al. Petsc users manual revision 3.8. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States), 2017.

[37] G. Karypis, K. Schloegel, and V. Kumar. Parmetis: Parallel graph partitioning and sparse matrix ordering library. *Version 1.0, Dept. of Computer Science, University of Minnesota*, page 22, 1997.

[38] G. de Vahl Davis. Natural convection of air in a square cavity: a bench mark numerical solution. *International Journal for numerical methods in fluids*, 3(3):249–264, 1983.

[39] H. Sajjadi, M. Gorji, S. F. Hosseinizadeh, G. R. Kefayati, and D. D. Ganji. Numerical analysis of turbulent natural convection in square cavity using large-eddy simulation in lattice boltzmann method. *Iranian Journal of Science and Technology. Transactions of Mechanical Engineering*, 35(M2):133, 2011.

[40] S. Chen, H. Liu, and C. Zheng. Numerical study of turbulent double-diffusive natural convection in a square cavity by les-based lattice boltzmann model. *International Journal of Heat and Mass Transfer*, 55(17):4862–4870, 2012.

[41] H. N. Dixit and V. Babu. Simulation of high rayleigh number natural convection in a square cavity using the lattice boltzmann method. *International journal of heat and mass transfer*, 49(3):727–739, 2006.

[42] P. Le Quéré. Accurate solutions to the square thermally driven cavity at high rayleigh number. *Computers & Fluids*, 20(1):29–41, 1991.

[43] D. C. Wan, B. S. V. Patnaik, and G. W. Wei. A new benchmark quality solution for the buoyancy-driven cavity by discrete singular convolution. *Numerical Heat Transfer: Part B: Fundamentals*, 40(3):199–228, 2001.

[44] T. Fusegi, J. M. Hyun, K. Kuwahara, and B. Farouk. A numerical study of three-dimensional natural convection in a differentially heated cubical enclosure. *International Journal of Heat and Mass Transfer*, 34(6):1543–1557, 1991.

[45] G. D. Mallinson and G. De Vahl Davis. Three-dimensional natural convection in a box: a numerical study. *Journal of Fluid Mechanics*, 83(1):1–31, 1977.

[46] R. J. Krane and J. Jessee. Some detailed field measurements for a natural convection flow in a vertical square enclosure. In *Proceedings of the First ASME-JSME Thermal Engineering Joint Conference, 1983*, volume 1, pages 323–329, 1983.

[47] Y. Bazilevs and I. Akkerman. Large eddy simulation of turbulent taylor–couette flow using isogeometric analysis and the residual-based variational multiscale method. *Journal of Computational Physics*, 229(9):3402–3414, 2010.

[48] A. Fontanini, M. G. Olsen, and B. Ganapathysubramanian. Thermal comparison between ceiling diffusers and fabric ductwork diffusers for green buildings. *Energy and Buildings*, 43(11):2973–2987, 2011.

[49] A. Fontanini, U. Vaidya, and B. Ganapathysubramanian. A methodology for optimal placement of sensors in enclosed environments: A dynamical systems approach. *Building and Environment*, 100:145–161, 2016.

CHAPTER 3. IMMERSOGEOMETRIC ANALYSIS OF MOVING OBJECTS IN INCOMPRESSIBLE FLOWS

This chapter includes a manuscript titled "Immersogeometric analysis of moving objects in incompressible flows", in preparation for submission to *Computers & Fluids* journal, authored by Songzhe Xu, Fei Xu, Ming-Chen Hsu and Baskar Ganapathysubramanian

Abstract

We deploy the immersogeometric method for moving objects. The method immerses objects into non-boundary-fitted meshes and weakly enforces Dirichlet boundary conditions on the object boundaries. The object evolution is driven by integrated surface force and external force (gravity). The finite element formulation is stabilized by the variational multiscale method. Adaptively refined quadrature rules are used to better capture the geometry of the immersed boundary and accurately integrate the background elements that intersect the immersed boundary. Treatment for the freshly-cleared nodes is considered. We assess the accuracy of the method by analyzing object motion in different flow structures including freely dropping objects in viscous fluids and particle focusing in unobstructed and obstructed micro-channels. We show that quantities of interest are in very good agreements with analytical, numerical and experimental solutions. We also show a much better computational efficiency than Fluent with the body-fitted method. The framework of moving immersogeometric method is capable to be deployed in further applications of particle inertia migration in microfluidic channels.

3.1 Introduction

Control and localization of finite-size particles (e.g., cells and precipitates) in aqueous flow is useful in biological processing, chemical reaction control, and for creating structured materials.

Some examples include fast identification of *E. coli* in water, robust removal of circulating tumor cells from the blood plasma, and fast separation of cells types for rapid flow cytometry. The precise, efficient and cheap localization of a heterogeneous collection of cells in a fluid medium is an important challenge with multiple engineering and health applications. Most current approaches to particle localization in microfluidic devices are predominantly *active*, i.e., some external stimuli, such as electric field, permeate flow, and stirrer, are used to create flow conditions that encourage particle separation and localization. However, active control of particles in microfluidic devices results in device designs that are potentially more expensive, with multiple moving parts that can fail more frequently, and that require operation and transport in controlled environments. A general strategy for passive control of particle localization and focusing in microfluidic channels will be transformative to this field.

Studies suggest that such a passive flow control paradigm is possible by utilizing the notion of *inertial migration* and focusing of particles in microfluidic channels [1]. Segre and Silberberg [2, 3] first experimentally observed the phenomena of focusing of particles in a straight channel (or tube) flow. Under conditions where inertial effects are non-negligible (i.e. channel Reynolds number ≥ 5), particles undergo a lateral motion across the flow streamlines until they reach a stable equilibrium located between the channel centerline and the confining walls. Subsequent theoretical studies provided a general understanding of the lift forces and how the structure of lift forces depends on the particle size, channel dimensions and flow rate (or Reynolds number). Recent studies have further shown that placing bluff obstacles, such as micropillarsin the microchannel produces a particle size/inertia dependent effect, thus providing additional knobs for passive control. While the localization (or focusing) of particles in unobstructed microfluid channels is well known, the behavior (and control) of localization of particles in *obstructed microfluidic channels* is a very novel problem with a rich physical underpinning. The ability to exploit these phenomena (for separation, concentration, and sorting of cells and biomolecules with high specificity) requires highly accurate force and trajectory calculations. This serves as the primary motivation for the current work.

Tracking finite-size particle motion in inertial flows (especially in obstructed geometries) is a computationally challenging problem. Approaches to simulate such a system include boundary-fitted and non-boundary-fitted methods. In the boundary-fitted approach, the fluid problem is solved on a mesh that conforms to the fluid–object (particle) interface and deforms around it. The fluid problem on the deforming domain is written in an arbitrary Lagrangian–Eulerian (ALE) [4–6] or a space–time (ST) [7,8] coordinate system. Boundary-fitted methods have the advantage of satisfying kinematic constraints such as no-slip boundary conditions by construction. However, for situations that involve large translational and/or rotational interface motions, the boundary-fitted mesh can become severely distorted if it is continuously deformed from a single reference configuration, harming both the conditioning of the discrete problem and the accuracy of its solution. Applying boundary-fitted methods to complex moving-interface problems may therefore require specialized solution strategies to maintain fluid mesh quality. One approach is remeshing, in which all or part of the fluid domain is re-discretized when mesh distortion becomes too extreme [9–11]. Mesh management is complicated further if the object moves into and out of contact with other objects, changing the topology of the fluid domain [12, 13].

For these reasons, non-boundary-fitted approaches have become a popular alternative for the simulation of moving-interface problems. Non-boundary-fitted methods approximate the solution of boundary value problems on analysis meshes that do not necessarily conform to the boundary of the domain. The analysis object is arbitrarily superimposed onto (or *immersed* into) a background fluid mesh. Such methods have greater geometric flexibility than their boundary-fitted counterparts. However, kinematic constraints such as Dirichlet boundary conditions at the immersed interface can no longer be imposed strongly on the discrete solution space. To apply interface conditions, one must devise a suitable method for weak enforcement. The first non-boundary-fitted approach that became widely known for computational fluid dynamics (CFD) was the immersed boundary method [14, 15]. Since then, immersed methods have been applied to a variety of flow problems [16–19]. In the context of finite elements, several adaptations of immersed methods were explored in the 2000s for the simulation of fluid interacting with moving objects. Glowinski and

coworkers [20–22] simulated viscous flow interacting with rigid particles by enforcing the rigid-body-motion constraint on the overlapping fluid mesh through a distributed Lagrange multiplier field. Zhang, Liu and cowokers [23–26] developed the immersed finite element method (IFEM) to use a flexible Lagrangian solid mesh that moves on top of a background Eulerian fluid mesh. Casquero et al. [27] later enhanced IFEM by introducing non-uniform rational B-splines (NURBS) as the basis functions to improve the robustness and accuracy of the immersed method. In addition, Rüberg and Cirak [28] and Kadapa et al. [29] applied Nitsche's method [30] at the immersed interface with background B-spline finite elements for the simulation of moving-boundary problems.

While immersed methods show great flexibility in solving complex moving-boundary problems, they typically suffer from reduced accuracy of the solution near the immersed boundary. Kamensky et al. [31] and Xu et al. [32] found that the reduced accuracy is partially related to the representation of the geometry in the immersed domain. The immersogeometric analysis (IMGA) was proposed to alleviate this issue, to faithfully capture the immersed geometry in the intersected elements. The method also alleviates the difficulties associated with CFD mesh generation around complex design geometries. The immersogeometric method is comprised of the following main components. A variational multiscale (VMS) formulation of incompressible flow [33–36] is used, which provides accuracy and robustness in both laminar and turbulent flow simulations. The Dirichlet boundary conditions on the surface of the immersed geometry are enforced weakly using an extension of Nitsche's method [37, 38]. This weak boundary condition formulation can be integrated over the immersed object surface directly using its computer-aided design (CAD) boundary representation (B-rep) [39, 40]. An exact representation of the design geometry is therefore used in the simulation, sharing the same philosophy with isogeometric analysis [41]. Adaptively refined quadrature rules are used to accurately integrate the background elements cut by the immersed boundary. It was found in Xu et al. [32] that improved quadrature in intersected elements is critical for obtaining accurate flow solutions when using immersed methods. In this work, we apply the immersogeometric method to the analysis of moving-particle problems.

Applying the IMGA to problems of moving objects raises some challenges, in which the most significant one is the treatment of freshly-cleared nodes – the nodes that are previously inside the object at one time step, but are outside (i.e. in the fluid domain) at the next time step due to object motion. These nodes lack a time history of the fluid field. This may result in discontinuities and divergences due to bad initial guesses. Udaykumar et al. [42] approximated the velocity and pressure on those freshly-cleared nodes via interpolation of neighboring cells. We adopt this technique and extend it into the context of finite element framework.

The paper is organized as follows. In Section 3.2, we summarize the numerical formulation of the Navier–Stokes equations posed on a non-boundary-fitted discretization. Section 3.3 describes the details of implementation of the framework. In Section 3.4, we verify and validate the developed methods by some cases, including freely falling cylinder/ball in viscous fluids, and particle focusing and migration in (un)obstructed channels. We demonstrate the capability of the developed framework on such applications of fluid–structure interaction. Finally, we draw conclusions and motivate future research in Section 3.5.

3.2 Immersogeometric methodology

In this section, we summarize the variational formulation of the Navier–Stokes equations of incompressible flow and its spatial and temporal discretizations. We also emphasize the weak enforcement of boundary conditions, which is an essential component of the IMGA framework.

3.2.1 Governing equations of incompressible flow

The Navier–Stokes equations of the incompressible flow are written as

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\boldsymbol{\nabla}\mathbf{u} - \mathbf{f}\right) - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma} = \mathbf{0}\,, \tag{3.1}$$

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

where ρ, $\mathbf{u}$, and $\mathbf{f}$ are the fluid density, the flow velocity and the external force per unit mass, respectively. The stress and strain-rate tensors are defined respectively as

$$\boldsymbol{\sigma}(\mathbf{u}, p) = -p\,\mathbf{I} + 2\mu\,\boldsymbol{\varepsilon}(\mathbf{u})\,, \tag{3.3}$$

$$\boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}\left(\boldsymbol{\nabla}\mathbf{u} + \boldsymbol{\nabla}\mathbf{u}^{\mathrm{T}}\right)\,, \tag{3.4}$$

where p is the pressure, $\mathbf{I}$ is an identity tensor and μ is the dynamic viscosity. The problem (3.1)–(3.4) is accompanied by suitable boundary conditions, defined on the boundary of the fluid domain, $\Gamma = \Gamma_t^{\mathrm{D}} \cup \Gamma^{\mathrm{N}}$:

$$\mathbf{u} = \mathbf{u}_{\mathbf{g}} \qquad \text{on } \Gamma_t^{\mathrm{D}}\,, \tag{3.5}$$

$$-p\,\mathbf{n} + 2\mu\,\boldsymbol{\varepsilon}(\mathbf{u})\,\mathbf{n} = \mathbf{h} \qquad \text{on } \Gamma^{\mathrm{N}}\,, \tag{3.6}$$

where $\mathbf{u}_{\mathbf{g}}$ denotes the prescribed velocity at the Dirichlet boundary Γ_t^{D}, $\mathbf{h}$ is the traction vector at the Neumann boundary Γ^{N}, and $\mathbf{n}$ is the unit normal vector pointing in the wall-outward direction.

3.2.2 Semi-discrete variational multiscale formulation

Consider a collection of disjoint elements $\{\Omega^e\}$, $\cup_e \Omega^e \subset \mathbb{R}^d$, with closure covering the fluid domain: $\Omega \subset \cup_e \overline{\Omega^e}$. Note that Ω^e is not necessarily a subset of Ω because of the non-conforming fluid–structure interface. Let $\mathcal{V}_u^h$ and $\mathcal{V}_p^h$ be the discrete velocity and pressure spaces of functions supported on these elements. The strong problem (3.1)–(3.6) may be recast in a weak form and posed over these discrete spaces to produce the following semi-discrete problem: Find $\mathbf{u}^h \in \mathcal{V}_u^h$ and $p^h \in \mathcal{V}_p^h$ such that for all $\mathbf{w}^h \in \mathcal{V}_u^h$ and $q^h \in \mathcal{V}_p^h$:

$$B^{\mathrm{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) - F\left(\{\mathbf{w}^h, q^h\}\right) = 0\,. \tag{3.7}$$

The bilinear form B^{VMS} and the load vector F^{VMS} are given as

$$\begin{aligned} B^{\mathrm{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) = &\int_\Omega \mathbf{w}^h \cdot \rho\left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{u}^h\right) d\Omega + \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}\left(\mathbf{u}^h, p^h\right)\, d\Omega \\ &+ \int_\Omega q^h\, \boldsymbol{\nabla}\cdot\mathbf{u}^h\, d\Omega \end{aligned} \tag{3.8}$$

$$
\begin{aligned}
&- \sum_e \int_{\Omega^e \cap \Omega} \left(\rho \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{w}^h + \boldsymbol{\nabla} q^h \right) \cdot \mathbf{u}' \, d\Omega \\
&- \sum_e \int_{\Omega^e \cap \Omega} p' \, \boldsymbol{\nabla} \cdot \mathbf{w}^h \, d\Omega + \sum_e \int_{\Omega^e \cap \Omega} \rho \mathbf{w}^h \cdot (\mathbf{u}' \cdot \boldsymbol{\nabla} \mathbf{u}^h) \, d\Omega \\
&- \sum_e \int_{\Omega^e \cap \Omega} \rho \boldsymbol{\nabla} \mathbf{w}^h : (\mathbf{u}' \otimes \mathbf{u}') \, d\Omega,
\end{aligned} \tag{3.9}
$$

and

$$
F\left(\{ \mathbf{w}^h, q^h \} \right) = \int_\Omega \mathbf{w}^h \cdot \rho \mathbf{f} \, d\Omega + \int_{\Gamma^N} \mathbf{w}^h \cdot \mathbf{h} \, d\Gamma \,, \tag{3.10}
$$

where $\mathbf{u}'$ is defined as

$$
\mathbf{u}' = -\tau_{\mathrm{M}} \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f} - \frac{1}{\rho} \boldsymbol{\nabla} \cdot \boldsymbol{\sigma} \left(\mathbf{u}^h, p^h \right) \right) \,, \tag{3.11}
$$

and p' is given by

$$
p' = -\rho \tau_{\mathrm{C}} \, \boldsymbol{\nabla} \cdot \mathbf{u}^h \,. \tag{3.12}
$$

Eq. (3.8)–(3.12) feature the residual-based VMS formulation of Navier–Stokes equations of incompressible flows [36]. The additional terms added onto the standard weak Galerkin form can be interpreted as a combination of streamline/upwind Petrov Galerkin (SUPG) stabilization and VMS large-eddy simulation of turbulence modeling [34–36, 43–46]. The stabilization parameters are designed as

$$
\tau_{\mathrm{M}} = \left(\frac{C_t}{\Delta t^2} + \mathbf{u} \cdot \mathbf{G} \, \mathbf{u} + C_I \, \nu^2 \, \mathbf{G} : \mathbf{G} \right)^{-1/2} \,, \tag{3.13}
$$

$$
\tau_{\mathrm{C}} = \left(\tau_{\mathrm{M}} \, \mathrm{tr} \, \mathbf{G} \right)^{-1} \,, \tag{3.14}
$$

where Δt is the time-step size, C_I is a positive constant [47–49], $\nu = \mu / \rho$ is the fluid kinematic viscosity, $\mathbf{G}$ is the element metric tensor calculated by the mapping from the isoparametric element to its physical counterpart $\mathbf{x}(\xi)$:

$$
G_{ij} = \sum_{k=1}^{d} \frac{\partial \xi_k}{\partial x_i} \frac{\partial \xi_k}{\partial x_j} \,, \tag{3.15}
$$

$\mathrm{tr} \, \mathbf{G}$ is the trace of $\mathbf{G}$, and the parameter C_t is typically set to 4 [36, 45].

3.2.3 Variationally consistent weak boundary conditions

The standard way of strongly imposing Dirichlet boundary conditions in Eq. (3.7) is not feasible in IMGA. We thus employ the weak enforcement in the sense of Nitsche's method [30] proposed by Bazilevs et al. [38, 50, 51]. Assuming the fluid domain boundary Γ is decomposed into N_{eb} surface elements each denoted by Γ^{b}, the semi-discrete problem becomes

$$
\begin{aligned}
& B^{\mathrm{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) - F^{\mathrm{VMS}}\left(\{\mathbf{w}^h, q^h\}\right) \\
& - \int_{\Gamma_t^{\mathrm{D}}} \mathbf{w}^h \cdot \left(-p^h\,\mathbf{n} + 2\mu\,\boldsymbol{\varepsilon}(\mathbf{u}^h)\,\mathbf{n}\right)\, d\Gamma \\
& - \int_{\Gamma_t^{\mathrm{D}}} \left(2\mu\,\boldsymbol{\varepsilon}(\mathbf{w}^h)\,\mathbf{n} + q^h\,\mathbf{n}\right) \cdot \left(\mathbf{u}^h - \mathbf{u}_{\mathbf{g}}\right)\, d\Gamma \\
& + \int_{\Gamma_t^{\mathrm{D}}} \tau^B \mathbf{w}^h \cdot \left(\mathbf{u}^h - \mathbf{u}_{\mathbf{g}}\right)\, d\Gamma \;=\; 0\,.
\end{aligned}
\tag{3.16}
$$

The detailed interpretation of different terms can be found in [38]. The only parameter, τ^B, is a penalty-like stabilization parameter that helps to satisfy the no-slip condition on the boundary. Bazilevs et al. [38, 50, 51] indicate that the stabilization parameter should be chosen as a compromise of the following conditions. If τ^B is too large, the penalty term dominates the formulation, overshadowing the variational consistency that is responsible for the good performance of the method, and can result in an ill-conditioned stiffness matrix. If τ^B is too small, on the other hand, the solution is not stable, and the solver may confront convergence issues as well. In this paper, we use the definition proposed in [52], which scales the stabilization parameter as $\tau^{\mathrm{B}} = C^{\mathrm{B}}\rho h/\Delta t$, where C^{B} is a positive constant. For simplicity, we choose h to be the wall normal size of the whole cut element. This choice leads to a slight over-estimation of the penalty parameter, due to the fact that only a part of an intersected element is inside the fluid domain. It was shown that excessive over-penalization in Nitsche type formulation on non-conforming meshes tends to produce oscillatory coupling forces [53–57]. Local surface traction calculations would likely suffer from the over-estimation of τ^{B}. In the present work, however, we are only interested in the overall fluid force acting on the surfaces of immersed objects, where the force integral over the closed surfaces may counteract the local over-estimation in opposite directions, which results in a reasonable net overall

force and smooths out the effect of local over-estimation. Our numerical experiments reveal that the proposed formulation of penalty preserves the accuracy very well.

3.2.4 Time discretization and iterative method

We employ the backward Euler finite difference scheme to complete the discretization in time

$$\frac{\partial \boldsymbol{u}}{\partial t} = \frac{\boldsymbol{u}^n - \boldsymbol{u}^{n-1}}{\Delta t} = \mathcal{L}(\boldsymbol{u}^n, p^n), \tag{3.17}$$

where the operator $\mathcal{L}(\boldsymbol{u}^n, p^n)$ represents all the other terms except the time-dependent term Eq. (3.1) evaluated at the current time step. Δt is selected to follow CFL condition. We implement the parallelized moving immersogeometric method within our in-house parallel finite element framework. The domain decomposition is achieved via Parmetis [58]. The (non)linear solution procedure is taken care of by PETSC [59]. We utilize the SNES construct (line search quasi-Newton), which uses the KSP construct, specifically the BCGS solver, for the linearized system. For some time steps, the SNES solver may diverge from prescribed relative tolerance of residual norm. We then apply sub-preconditioner LU to the sub-blocks of the matrix and resolve those time steps. We ensure for each time step, the residual norm is decreased below prescribed criterions (absolute and relative tolerances of SNES iterations, etc).

3.3 Implementation of moving B-rep

3.3.1 Modeling the rigid body motion

The objects are modeled as rigid bodies. We denote the velocities of the objects as $\mathbf{v}_i$, with the subscript i indicating the i^{th} immersed object, the motion of the objects can be described in the Lagrangian reference frame by

$$\frac{d\mathbf{x}_i^{\mathrm{c}}}{dt} = \mathbf{v}_i^{\mathrm{c}}, \qquad \frac{d\mathbf{v}_i^{\mathrm{c}}}{dt} = \frac{\mathbf{F}_i}{m_i}, \tag{3.18}$$

$$\frac{d\boldsymbol{\theta}_i^{\mathrm{c}}}{dt} = \boldsymbol{\omega}_i^{\mathrm{c}}, \qquad \frac{d\boldsymbol{\omega}_i^{\mathrm{c}}}{dt} = \frac{\mathbf{T}_i}{J_i}, \tag{3.19}$$

where $\mathbf{x}_i^{\mathrm{c}}$ and $\boldsymbol{\theta}_i^{\mathrm{c}}$ are the linear and angular locations of the centroid of i^{th} object, $\mathbf{v}_i^{\mathrm{c}}$ and $\boldsymbol{\omega}_i^{\mathrm{c}}$ are linear and angular velocities of the centroid of i^{th} object, $\mathbf{F}_i$ and $\mathbf{T}_i$ are the overall force and torque integrated over the surface of i^{th} object, and m_i and J_i are the mass and moment of inertia of the i^{th} object. $\mathbf{F}_i$ and $\mathbf{T}_i$ are computed from the solution of the fluid field, and defined as follows

$$\mathbf{F}_i = \oint_{\Gamma_i} \boldsymbol{\sigma}_i(\mathbf{u}, p) \cdot \mathbf{n}_i d\Gamma, \qquad \mathbf{T}_i = \oint_{\Gamma_i} \mathbf{r}_i \times (\boldsymbol{\sigma}_i(\mathbf{u}, p) \cdot \mathbf{n}_i)\, d\Gamma, \tag{3.20}$$

Where Γ_i is the boundary of the i^{th} object, $\boldsymbol{\sigma}_i(\mathbf{u}, p)$ is the stress tensor acting on the surface of i^{th} object, $\mathbf{r}_i$ is the distance vector from the centroid of i^{th} object to the evaluated point on its surface, and the coordinates $\mathbf{x}_i$ and velocities $\mathbf{v}_i$ of the evaluated point on the surface of i^{th} object are computed as

$$\mathbf{x}_i = \mathbf{x}_i^{\mathrm{c}} + \mathbf{r}_i, \qquad \mathbf{v}_i = \mathbf{v}_i^{\mathrm{c}} + \boldsymbol{\omega}_i^{\mathrm{c}} \times \mathbf{r}_i. \tag{3.21}$$

where $\mathbf{n}_i$ is the unit normal vector at the evaluated point that points outward from the i^{th} object. In the discrete form, assuming the overall force and torque acting on the i^{th} object surface are constant during one time step, we have

$$\frac{(\mathbf{x}_i^{\mathrm{c}})^{n+1} - (\mathbf{x}_i^{\mathrm{c}})^n}{\Delta t} = \frac{(\mathbf{v}_i^{\mathrm{c}})^{n+1} + (\mathbf{v}_i^{\mathrm{c}})^n}{2}, \qquad \frac{(\mathbf{v}_i^{\mathrm{c}})^{n+1} - (\mathbf{v}_i^{\mathrm{c}})^n}{\Delta t} = \frac{(\mathbf{F}_i)^n}{m_i}, \tag{3.22}$$

$$\frac{(\boldsymbol{\theta}_i^{\mathrm{c}})^{n+1} - (\boldsymbol{\theta}_i^{\mathrm{c}})^n}{\Delta t} = \frac{(\boldsymbol{\omega}_i^{\mathrm{c}})^{n+1} + (\boldsymbol{\omega}_i^{\mathrm{c}})^n}{2}, \qquad \frac{(\boldsymbol{\omega}_i^{\mathrm{c}})^{n+1} - (\boldsymbol{\omega}_i^{\mathrm{c}})^n}{\Delta t} = \frac{(\mathbf{T}_i)^n}{m_i}. \tag{3.23}$$

$(\mathbf{F}_i)^n$ and $(\mathbf{T}_i)^n$ are discretized in space and computed with weakly imposed boundary condition as follows

$$(\mathbf{F}_i)^n = \sum_{b=1}^{N_{eb}} \int_{\Gamma^{\mathrm{b}} \bigcap \Gamma_i} \boldsymbol{\sigma}_i(\mathbf{u}^n, p^n) \cdot \mathbf{n}_i d\Gamma - \sum_{b=1}^{N_{eb}} \int_{\Gamma^{\mathrm{b}} \bigcap \Gamma_i} \tau^B(\mathbf{u}^{\mathbf{n}} - \mathbf{v}_i^n) d\Gamma, \tag{3.24}$$

$$(\mathbf{T}_i)^n = \sum_{b=1}^{N_{eb}} \int_{\Gamma^{\mathrm{b}} \bigcap \Gamma_i} \mathbf{r}_i \times (\boldsymbol{\sigma}_i(\mathbf{u}^n, p^n) \cdot \mathbf{n}_i)\, d\Gamma - \sum_{b=1}^{N_{eb}} \int_{\Gamma^{\mathrm{b}} \bigcap \Gamma_i} \mathbf{r}_i \times \tau^B(\mathbf{u}^{\mathbf{n}} - \mathbf{v}_i^n) d\Gamma \tag{3.25}$$

Each object velocity is evaluated using an explicit forward Euler scheme, which requires small Δt to ensure accuracy and stability. Each object location is updated by the average velocities, which is essentially Crank–Nicolson scheme, and therefore more stable and accurate. [1]

[1] Some higher order schemes in explicit BDF family as well as implicit RK2 family are also tried. However, we found they don't improve the solution with a significantly bigger Δt.

3.3.2 In-out test

The in-out test essentially returns a boolean value depending on if a certain point is inside-or-outside the object. There are a variety of approaches for performing an in-out test, including the ray-tracing algorithm. In our simulations, since we only consider analytical shapes for our objects, we can determine if a query point is inside-or-outside an object using the geometry equation of the object. The equation of an object geometry will change every time step because of the linear and angular motion of the object. We transform the query point into the Lagrangian reference frame of the object in the in-out test so that we can retain the consistency of the geometry equation to avoid its computation in Euler reference frame every time step during the movement of the object.

3.3.3 Treatment of freshly-cleared nodes

Consider the nodes in the background mesh that are covered by the object at this time step. At the next time step, some of these nodes may no longer be covered by the object due to the motion of the object. We call these nodes 'freshly-cleared'. Note that we have no fluid history of the flow field recorded on those nodes. [2] To obtain a reasonable solution on those nodes and complete the fluid field solution, we interpolate using the solution of their neighbor nodes in the fluid domain and boundary values of nearby points on the object surface. This provides a good guess to start the solution process. We denote the union of the variables in fluid field as $\mathbf{U} = \{\mathbf{u}, p\}$, and the union of the variables on the i^{th} object surface as $\mathbf{V} = \{\mathbf{v}_i, p_i\}$. The union of interpolated variables on the freshly-cleared nodes corresponding to the i^{th} object is denoted as $\mathbf{U}_i^{\text{freshly-cleared}}$.

$$\mathbf{U}_i^{\text{freshly-cleared}} = \frac{\sum_{k=1}^{N_\text{f}} \frac{1}{d_\text{f}^k}\mathbf{U}^k + \sum_{k=1}^{N_\text{s}} \frac{1}{d_\text{s}^k}\mathbf{V}_i^k}{\sum_{k=1}^{N_\text{f}} \frac{1}{d_\text{f}^k} + \sum_{k=1}^{N_\text{s}} \frac{1}{d_\text{s}^k}} \tag{3.26}$$

[2] In fact, most of the freshly-cleared nodes should be on the intersected elements because Δt is small, and therefore the distance that an object can move in one time step would be small as well. We do solve for those nodes even if they are inside the object, but the values on those nodes are adjusted to balance the corresponding intersected elements and ensure the weakly imposed no-slip boundary condition on the object surface is satisfied, and therefore achieve the accuracy of the solution on their neighbor nodes in the fluid field. As a result, the values on the freshly-cleared nodes are not the correct solution of the fluid field.

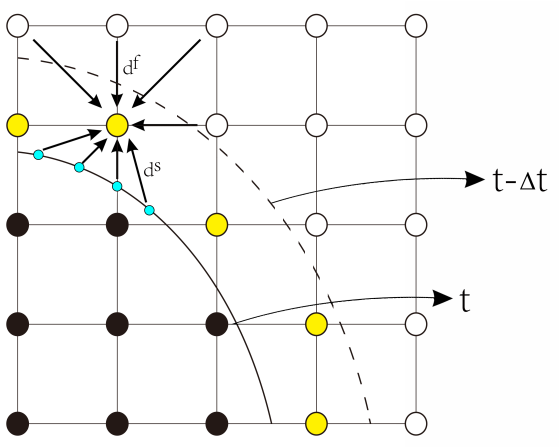


Figure 3.1: Schematic of the interpolation of the freshly-cleared nodes.

where d_{f} and d_{s} are the distances from neighbor nodes in fluid domain and nearby points on the object surface to the freshly-cleared node, respectively, and N_{f} and N_{s} are their respective numbers. All the nodes in fluid domain and points on object surface that are used to interpolate at one particular freshly-cleared node are set to be in the region centered at the freshly-cleared node with a radius of $O(h)$. [3] A schematic of this is shown in Fig. 3.1.

3.3.4 Work flow of the framework

We first solve for the fluid field solution $\mathbf{U}^n$ at current time step t^n, provided the updated coordinates of surface points of each object $\mathbf{x}_i^n$ and the updated boundary velocities on surface points of each object $\mathbf{v}_i^n$, and fluid field solution $\mathbf{U}^{n-1}$ from last time step t^{n-1}. Then we evaluate the overall force and torque over each object surface, $\mathbf{F}_i^n$ and $\mathbf{T}_i^n$, based on $\mathbf{U}^n$ and $\mathbf{v}_i^n$. Then we update the coordinates of surface points of each object $\mathbf{x}_i^{n+1}$ at next time step t^{n+1}. Then we check if there are any freshly-cleared nodes in the fluid field, and interpolate for those nodes by $\mathbf{U}^n$ at neighbor nodes in fluid field, and $\mathbf{V}_i^n$ on nearby points on corresponding object surface. We need to compute the boundary values of pressure on surface points of each object p_i^n before the interpolation of pressure. Finally, we update the velocities on surface points of each object $\mathbf{v}_i^{n+1}$ as boundary conditions for $\mathbf{U}^{n+1}$ at t^{n+1}, and pass the fluid field $\mathbf{U}^n$ to solve for $\mathbf{U}^{n+1}$. This procedure is shown in Fig. 3.2.

[3] We also ensure that N_{f} and N_{s} are comparable.

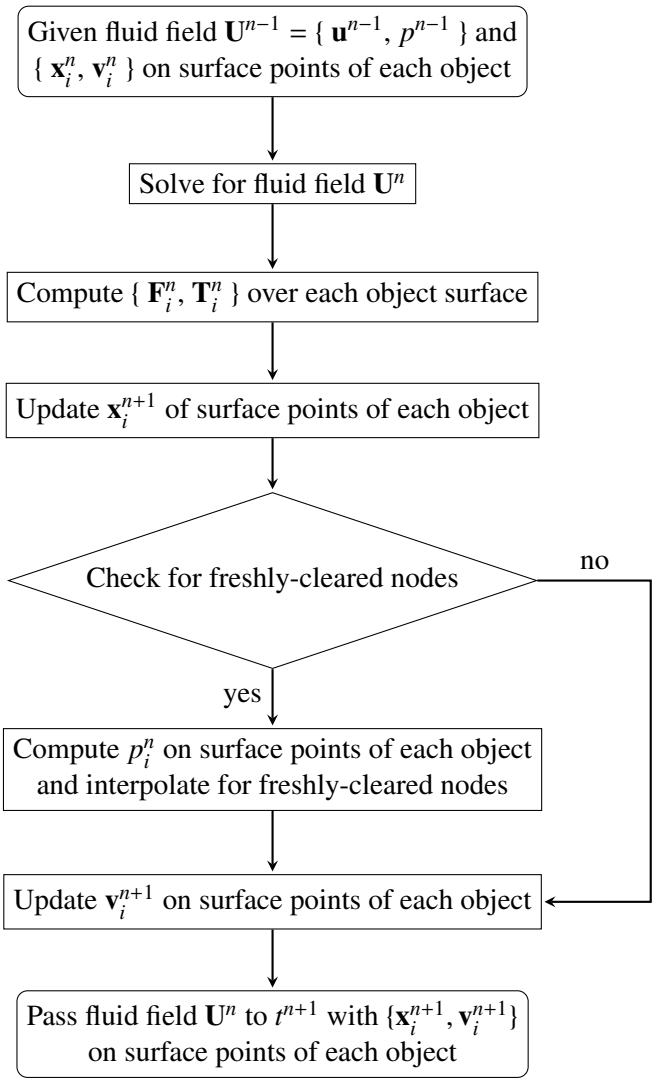


Figure 3.2: Flow chart of the process of moving IMGA.

3.3.5 Non-dimensionalization

We non-dimensionalize the variables and parameters as follows

$$
\begin{aligned}
&x^* = \frac{x}{L_0} \qquad u^* = \frac{u}{u_0} \qquad t^* = \frac{t}{L_0/u_0} \qquad \rho^* = \frac{\rho}{\rho^f} \\
&p^* = \frac{p}{u_0\mu/L_0} \qquad g^* = \frac{g}{u_0\mu/(\rho_f L_0^2)} \quad \text{(viscous scaling)} \\
&p^* = \frac{p}{\rho_f u_0^2} \qquad g^* = \frac{g}{u_0^2/L_0} \quad \text{(inertial scaling)}
\end{aligned} \tag{3.27}
$$

where L_0 is the characteristic length and u_0 is the characteristic velocity depending on the problem of interests, ρ_f is the fluid density, and g is the gravity. Reynolds number $Re = \frac{\rho_f u_0 L_0}{\mu}$. We employ viscous scaling when $Re \ll 1$, and inertial scaling when Re is moderate.

3.4 Verification and validation

We illustrate the moving IMGA framework using several benchmark results.

3.4.1 Free falling cylinder with low Re (2D)

A freely dropping cylinder with diameter D and density ρ_s will reach an equilibrium state with a constant terminal velocity V_T in a sufficiently high channel filled with a viscous fluid. Drag force F_{D}, buoyancy F_{b}, and gravitational force F_{g} will eventually reach an equilibrium so that the net force is zero, and the cylinder moves with the constant terminal velocity. For creeping flow with low Re number ($Re \ll 1$), we have an analytical solution for the terminal velocity of a cylinder with an infinite length [60].

$$
V_T = \frac{\left(\rho_s - \rho_f\right) g D^2}{16\mu}\left(\ln\left(\frac{W}{D}\right) - 0.9157 + 1.7244\left(\frac{D}{W}\right)^2 - 1.7302\left(\frac{D}{W}\right)^4\right). \tag{3.28}
$$

Where W is the channel width, and g is the gravitational acceleration. The infinite cylinder length allows us to perform 2D simulations to verify our moving immersogeometric framework. A mesh convergence study is performed to show that the simulated terminal velocity converges to the analytical solution with increasing mesh density. We choose $\rho_f = 10^3$ kg/m^3, $\rho_s = 1.25 \times 10^3$

kg/m^3, gravity acceleration $g = 9.81$m/s^2, dynamic viscosity $\mu = 0.5$ kg/(m $\cdot$ s), cylinder diameter $D = 5 \times 10^{-3}$ m, channel width $W = 0.04$ m and channel height $H = 0.06$ m. With all these parameters, we have the terminal velocity $V_{\mathrm{T}} = 9.12 \times 10^{-3}$ m/s, and $Re = 0.0912$ with V_T to be the characteristic velocity. Since $Re \ll 1$, we non-dimensionalize this problem with the viscous scaling. Initial conditions are set as zero velocity and pressure in the whole fluid domain. No slip boundary condition is imposed on the lateral and bottom walls, and no velocity gradient and zero pressure boundary conditions are imposed on the top wall. The problem setup is shown in Fig. 3.3 (a).

The intersection region of the fluid domain and the object surface needs more refinement to capture the immersed boundary and resolve the no-slip boundary condition. In this case, since we know the cylinder trajectory is a straight line, we can fix the mesh by refining it all the way along the trajectory to avoid any remeshing and interpolation between two meshes. Hsu et al. [39] suggested that the ratio between the element size of the immersed surface and the element size of the background mesh that intersects the immersed surface needs to be at most $1/2$. We choose the ratio to be $1/2$ for all the simulations in this paper. In addition, we also keep the ratio between the sizes of the largest and smallest background elements to be around 4 as a reasonable maximum aspect ratio of any background elements in the fluid domain. The fixed stretched mesh for this case is shown in Fig. 3.3 (b). The dimensionless element size across the fluid–structure interface is set to be 0.1, 0.05, and 0.025, which results in a total element number of 35×120, 70×240, and 140×480. The dimensionless time step Δt is set to be 2×10^{-3} to retain the stability of the framework.

The result of the mesh convergence study with the analytical terminal velocity is presented in Fig. 3.4. We can clearly see our numerical terminal velocity is converging to the analytical result as increasing mesh density. The relative error compared with the analytical terminal velocity is 3.84%, 1.26%, and 0.1% for the mesh of 35×120, 70×240, and 140×480, respectively. The mesh density of 70×240 (dimensionless element size across the interface to be 0.05) has already produced a very accurate result. We also present the velocity magnitude contour along with the streamlines at time $t = 1$ after the cylinder reaches the terminal velocity in Fig. 3.5. We can see two

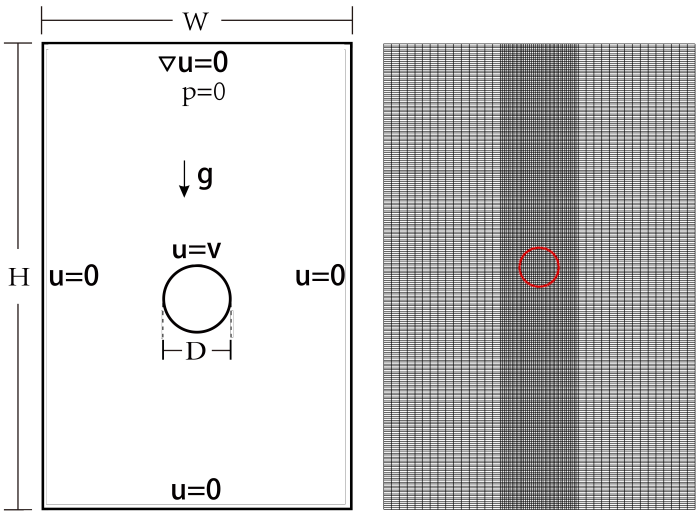


Figure 3.3: Free falling cylinder with low *Re* (2D). (a) Problem setup. (b) Cluster mesh.

large vortices caused by the downward motion of the cylinder as physically expected in this case. Similar velocity magnitude contour and streamlines are also observed in [61].

3.4.2 Free falling sphere with moderate *Re*

In this case, we compare the moving immersogeometric framework with experimental benchmark results. We employ a similar case as the last section, but replace the 2D cylinder to a 3D sphere, and simulate a freely dropping sphere to compare with the experimental results in [62]. The size of the domain is 0.1 m $\times$ 0.16 m $\times$ 0.1 m. The diameter of the sphere is $D = 0.015$ m, and it is released at rest at the height where its bottom apex is 0.12 m away from the bottom wall. The working fluid has a density $\rho_f = 960$ kg/m^3, and a dynamic viscosity $\mu = 0.058$ kg/(m $\cdot$ s). The density of the sphere is $\rho_s = 1120$ kg/m^3. This results in a Reynolds number $Re = 31.9$ with the characteristic velocity to be $V_0 = 0.128$ m/s. Since Re is moderate in this case, we non-dimensionalize the equation with inertial scaling. The dimensionless element size across the interface is set to be 0.05, and the dimensionless Δt is set to be 0.01. The same initial conditions and boundary conditions as in the last section are imposed.

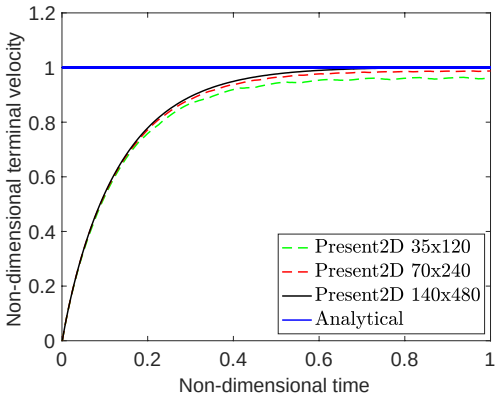


Figure 3.4: Mesh convergence results compared with the analytical solution.

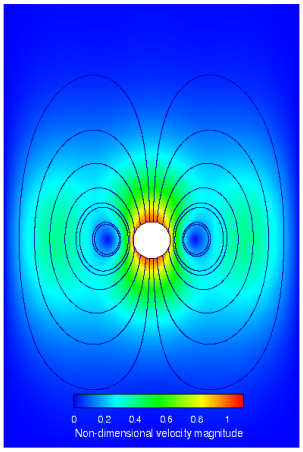
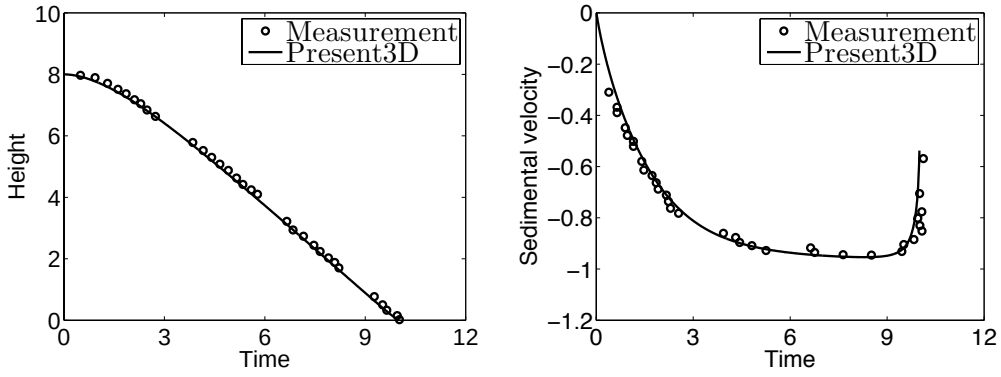


Figure 3.5: Velocity magnitude contour and streamlines at $t = 1$.

We show the comparison of the dimensionless height of the sphere bottom apex away from the bottom wall, as well as the dimensionless sedimental velocity of the sphere in Fig. 3.6. We can see a very good agreement in both the trajectory as well as the sedimental velocity between our numerical results and the experimental results. In addition, we can observe that sedimental velocity magnitude drastically decreases as the sphere approaches the bottom wall. This is because when the sphere moves closer towards the bottom boundary, the boundary will generate a so-called wall-effect force to prevent the sphere moving closer, and this force turns out to be much larger than the gravity in this case. Wall-effect force is also a dominating force in particle focusing problems. We



(a) (b)

Figure 3.6: Comparison of the sphere trajectory and velocity. (a) Height of the sphere bottom apex. (b) Sedimental velocity.

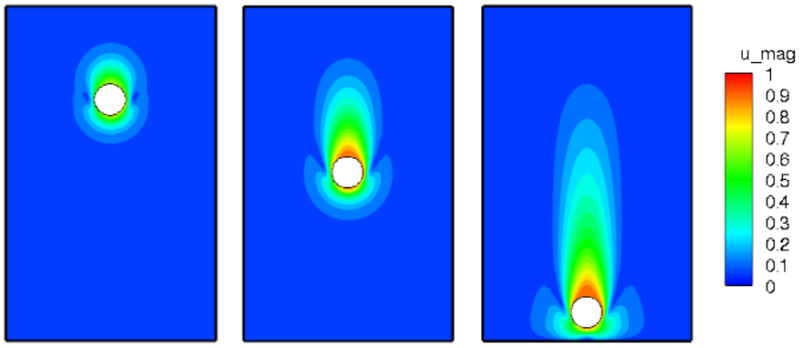


Figure 3.7: Velocity magnitude contour at different heights.

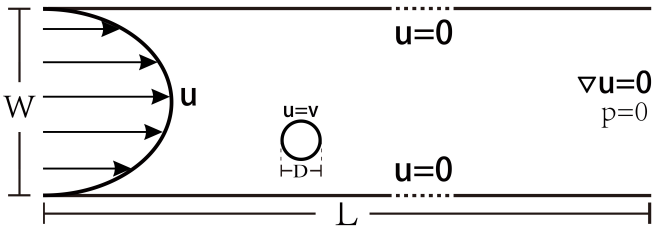
also show the contour of the velocity magnitude at different heights of the sphere bottom apex away from the bottom wall during the sedimentation as shown in 3.7. The heights are 7.15, 4.88 and 0.41 respectively. Similar contours can be found in experimental studies [62].

3.4.3 Neutral buoyant circular particle focusing in a straight channel

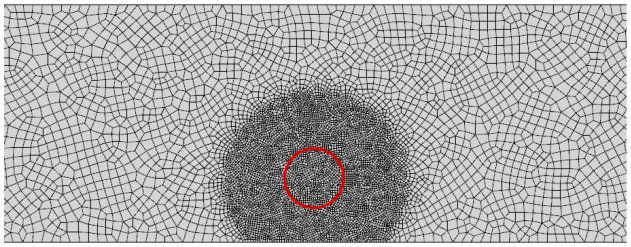
In the last two cases, the ambient fluid is quiescent, and the motion of the object is caused by gravity and further causes the motion of the fluid. In this case, we would like to deploy our framework to model more complex flow physics. A circular particle moving in a straight channel will focus at unique locations in the channel cross-section. In such a case, the motion of the particle

is driven by the fluid flow in a fully developed channel (parabolic fluid velocity profile). The fluid velocity profile is disturbed locally by the particle, which leads to a more complicated velocity gradient (i.e. more complicated viscous force) on the particle surface, and therefore results in a more intricate interaction between the particle and fluid. According to Amini et al. [63], there are two dominant forces in this case, shear-gradient force and wall-effect force, that eventually laterally push the object to a unique focusing position. From the numerical result of Feng et al. [64] at $Re = 40$, if we release a particle with a diameter of $D = 0.25$, at a position $w < 0.5$, it will focus at the position $w = 0.272$. To observe the focusing phenomena, a sufficiently long channel is needed. We choose a 2D channel with a dimensionless size of 40×1, and release the particle at the position $w = 0.225$, with an initial horizontal velocity $V_0 = 0.66$. The dimensionless element size across the interface is set to be 0.05, and the dimensionless Δt is set to be 5×10^{-4}. The initial conditions of the fluid are set as parabolic velocity and linear pressure distributions based on Re in a fully developed channel. The boundary conditions are the same parabolic velocity profile at the inlet, and zero velocity gradient and zero pressure at the outlet. No slip boundary condition is imposed on channel walls.

In this case, remeshing has to be performed during the particle motion, and we utilize the meshing code Gmsh. We refine a bounding circle that contains the particle. The radius of the bounding circle is set to be three times the radius of the particle. We can let the particle move for a while in the bounding circle to avoid remeshing and interpolation between two meshes every time step. We remesh and interpolate when the horizontal or lateral movement in the bounding sphere is more than a distance of particle radius, and reset the bounding sphere center to be the particle centroid after remeshing. Between the refined and unrefined region, Gmsh is able to relax the mesh to avoid a steep change of the mesh density. The problem setup and an unstructured mesh example are shown in Fig. 3.8. We present the particle trajectory in Fig. 3.9. The criterion we employ to verify the focusing is that the lateral position doesn't change across 5 dimensionless seconds (out of totally simulated 32.8 dimensionless seconds), which is sufficiently long. From the Fig. 3.9, we can see our framework is able to show the focusing phenomena, and it shows a similar trajectory compared



(a)



(b)

Figure 3.8: Neutral buoyant circular particle focusing in a straight channel. (a) Problem setup. (b) A mesh example.

with Feng et al.'s result. Our result shows a focusing position at the position $w = 0.280$, which is slightly higher than Feng et al.'s result, with a relative error of 0.8% scaled by the channel width.

3.4.4 Circular particle focusing in a straight channel with pillar

We simulate a 2D channel with a circular pillar as the obstruction in the channel. The origin is set to be the bottom left corner of the channel. Buoyancy is considered in this case. All of the following parameters are in dimensionless form. The size of the channel ($L \times H$) is 30×5, the diameter of the pillar is $D = 2.5$, and it is centered at $(5, 2.5)$. The diameter of the particle is $d = 1$, and it is released at $(0.6, 1.25)$ at rest. The density ratio between particle and fluid is 1.01. Reynolds number is $Re = 20$, scaled by the diameter of the particle and maximum inlet velocity. $\Delta t = 3 \times 10^{-3}$, and inertia scaling is used. The initial condition for velocity and pressure is parabolic and linear profile respectively, as a fully developed flow in a straight channel without obstruction. The boundary condition is parabolic velocity at the inlet, and zero velocity gradient and zero pressure at the outlet. In this case, we have a fixed pillar in the channel, and we can also immerse the pillar. Gmsh is able

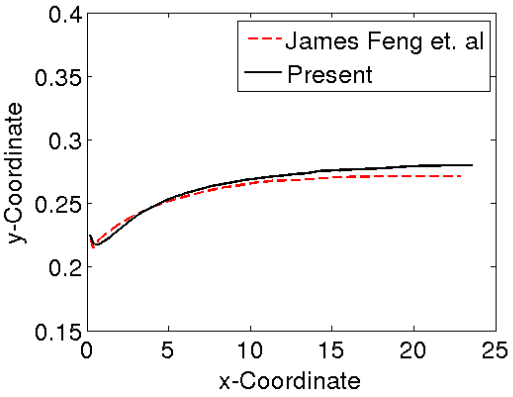
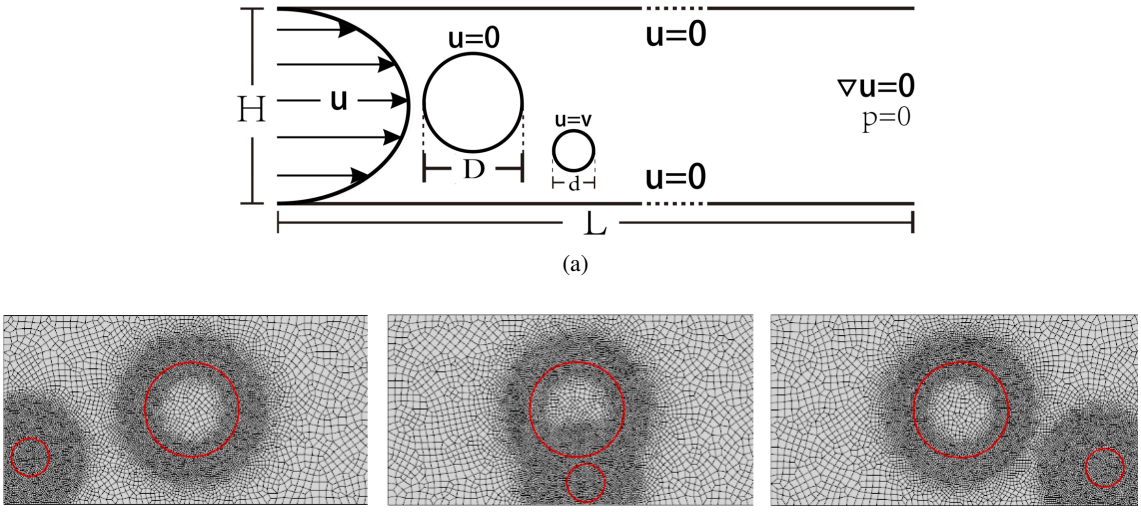


Figure 3.9: Comparison of the particle trajectory.

to refine a ring shaped region with a given center and band width for the fluid–structure interface. The case setup and unstructured moving meshes at different times are shown as Fig. 3.10.

For this case, since we lack other numerical results to verify our framework, we also simulate an equivalent dimensional case using Fluent with the body-fitted method. The detailed problem setup in Fluent is shown in Appendix. Mesh convergence study for the particle trajectory is performed for these two methods, and they both produce converged trajectory of the particle. The convergence result of moving IMGA is shown in Fig. 3.11. All the mesh densities produce similar results, which indicates the coarsest mesh would be able to reasonably predict the particle motion. However, we observed that using Fluent with the body-fitted method, the particle trajectory doesn't converge until the finest mesh. It demonstrates that our moving IMGA would spend a cheaper computational cost to predict the trajectory in this problem. Since the mesh will change during the particle motion, which is noticeable for the moving IMGA caused by the topology change of the refinement region, we report time-weighted average mesh densities for the moving IMGA. We choose the finest meshes in both moving IMGA framework and body-fitted Fluent to illustrate the computational efficiency. The comparison is shown in Table 3.1. Our moving IMGA is about 8 times faster than body-fitted Fluent at the same mesh density.

We present the dimensionless particle trajectory and velocity comparisons with the finest meshes in both simulations as shown in Fig. 3.12 and 3.13. We can see they match well. The final vertical



(a)

(b)

Figure 3.10: Circular particle focusing in a straight channel with pillar. (a) Problem setup. (b) Moving meshes at different times.

Table 3.1: Comparison of computational effort.

	Moving IMGA	Body-fitted Fluent
Average number of elements	28268	28182
Number of cores	16	16
Total simulation time (min)	199	1446

position from our framework and Fluent is 1.12 and 1.08 respectively. The relative error between them is 0.8% of the channel height. In addition, our framework shows the horizontal velocity eventually reaches a steady value of 0.646, which is 92.9 % of the undisturbed fluid velocity at the same height, and the vertical velocity eventually reaches about zero.

A more detailed comparison is shown in Fig. 3.14. We plot the velocity contour and streamlines at three locations $(2.59, 1.05)$, $(4.95, 0.566)$, and $(7.17, 0.776)$. We can see the particle starts to disturb the symmetry of the downstream wake region when it passes below the pillar, and completely distorts the wake region during its further movement. The lower circulation in the wake region almost disappears. The upper circulation also shrinks and reorients, and detaches from the pillar surface. Our future goal is to explore how the particle with different sizes and releasing positions

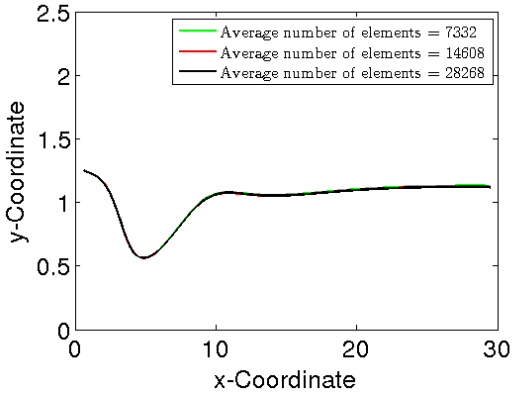


Figure 3.11: Mesh convergence results of the particle trajectory.

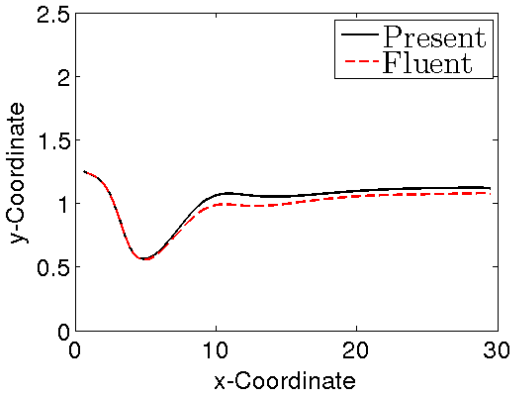
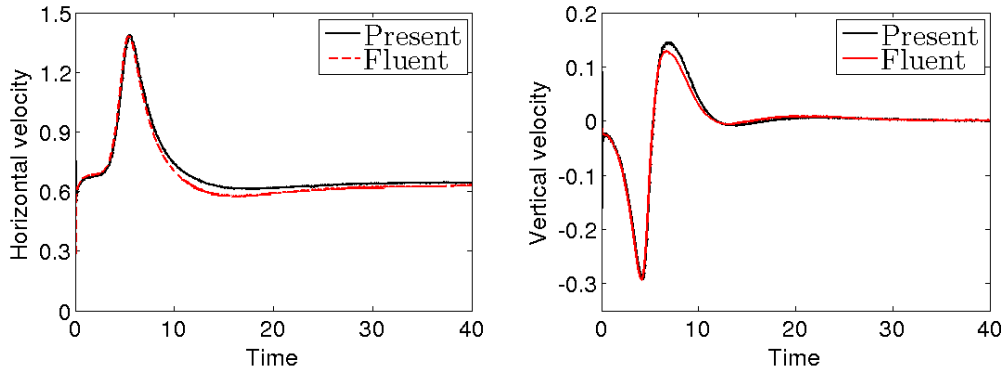


Figure 3.12: Comparison of particle trajectory.

would affect the flow structure, and how the flow structure would affect the trajectory of the particle. In addition, when the particle passes below the pillar, two boundaries, the pillar and the bottom wall, will play a role on the particle motion. They will both generate wall-effect forces, but with opposite direction to prevent the particle moving closer to them. The two wall-effect forces would compete along with the inertia from the fluid motion. It would also be with interests to investigate the consequent effect on the particle trajectory.



(a) (b)

Figure 3.13: Comparison of particle velocities. (a) Horizontal velocity. (b) Vertical velocity.

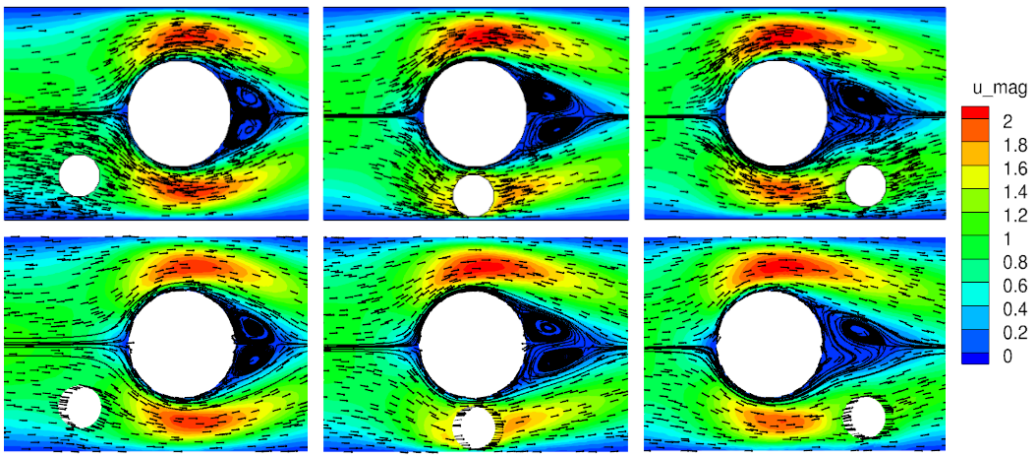


Figure 3.14: Comparison of contours of fluid velocity magnitude and streamlines. (upper) Moving IMGA. (lower) Body-fitted Fluent.

3.5 Conclusions and future work

Mesh convergence study is performed to verify the moving immersogeometric method with a 2D freely dropping cylinder. The terminal velocity of the cylinder is verified with the analytical solution as well. A 3D freely dropping sphere is further simulated. The sphere sedimental height and velocities are validated by experimental results. Reasonable contours of fluid velocity at different heights are observed as well. For the particle focusing problems, which have more complex interactions between fluid and particle, we are able to predict the particle trajectory in an unobstructed channel. For an obstructed channel decorate with a pillar, we also produce good agreements of particle trajectory and velocities as well as flow structures compared with body-fitted method simulated using Fluent, while we have advantages in computational cost and efficiency. This framework of moving immersogeometric method is concluded to be capable to predict the object evolution and resulting flow field in problems of fluid–structure interaction, and to be deployed to applications of particle inertia migration in microfluidic channels.

3.6 Appendix: Fluent simulation setup

The simulation of particle moving in an obstructed (pillar) channel with body-fitted method was done on ANSYS Fluent 16.1 using the built-in Dynamic Mesh (DM) module. The Navier–Stokes equations are solved by the default solver using the SIMPLE algorithm. The particle evolution is calculated in the *6 Degree-of-Freedom (6 DOF)* solver for rigid-bodies. An implicit mesh update feature is enabled – the dynamic mesh is updated during the course (convergence) of the solution of flow fields in one time-step [65], which results in a stable mesh update and better tracking of the particle positions. The remeshing is automatically determined by the solver based on the worst element skewness due to the particle motion and controlled by given remeshing parameters. The flow fields are projected from the previous mesh to the current mesh upon remeshing. The flow-fields are solved over a dimensional domain, 0.05 m $\times$0.3 m, where the fluid properties of density and dynamic viscosity measure, 1000 kg/m^3, and 0.1 kg/m^3, respectively. The particle is released at x = 0.006 m, and, y = 0.0125 m, with the origin set at bottom left corner. The particle mass, and

moment-of-inertia about the z-axis are externally supplied through a User-Defined Function (UDF) using the `DEFINE_SDOF_PROPERTIES` macro, and measure, 0.0793 kg, and 1×10^{-6} kg·m^2. The buoyant-force on the particle is imposed as an external force, `prop[SDOF_LOAD_F_Y]` = 0.770 N. The weight of the particle is computed by enabling the gravity option under the 6DOF solver with the value of 9.81 m/s^2. The initial conditions are taken as fully-developed profiles of velocity and pressure, in the absence of the particle, and pillar, whose maximum values measure, 0.2 m/s, and 19.2 Pa. The boundary conditions include parabolic-velocity inlet, and zero-pressure outlet. The remeshing parameters are guided by the default mesh-sizes from the initial undeformed domain. In addition, a constant-factor of 0.5 is chosen under the spring-based smoothing-parameters of the dynamic mesh.

3.7 References

[1] D. Di Carlo. Inertial microfluidics. *Lab on a Chip*, 9(21):3038–3046, 2009.

[2] G. Segré and A. Silberberg. Behaviour of macroscopic rigid spheres in poiseuille flow part 1. determination of local concentration by statistical analysis of particle passages through crossed light beams. *Journal of fluid mechanics*, 14(1):115–135, 1962.

[3] G. Segré and A. Silberberg. Behaviour of macroscopic rigid spheres in poiseuille flow part 2. experimental results and interpretation. *Journal of fluid mechanics*, 14(1):136–157, 1962.

[4] T. J. R. Hughes, W. K. Liu, and T. K. Zimmermann. Lagrangian–Eulerian finite element formulation for incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 29:329–349, 1981.

[5] J. Donea, S. Giuliani, and J. P. Halleux. An arbitrary Lagrangian–Eulerian finite element method for transient dynamic fluid–structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33(1-3):689–723, 1982.

[6] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodriguez-Ferran. Arbitrary Lagrangian–Eulerian methods. In *Encyclopedia of Computational Mechanics*, Volume 3: Fluids, chapter 14. John Wiley & Sons, 2004.

[7] T. E. Tezduyar, M. Behr, and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces – the deforming-spatial-domain/space–time procedure: I. The concept and the preliminary numerical tests. *Computer Methods in Applied Mechanics and Engineering*, 94(3):339–351, 1992.

[8] T. E. Tezduyar, M. Behr, S. Mittal, and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces – the deforming-spatial-domain/space–time procedure: II. Computation of free-surface flows, two-liquid flows, and flows with drifting cylinders. *Computer Methods in Applied Mechanics and Engineering*, 94(3):353–371, 1992.

[9] A. A. Johnson and T. E. Tezduyar. Parallel computation of incompressible flows with complex geometries. *International Journal for Numerical Methods in Fluids*, 24:1321–1340, 1997.

[10] A. A. Johnson and T. E. Tezduyar. 3D simulation of fluid-particle interactions with the number of particles reaching 100. *Computer Methods in Applied Mechanics and Engineering*, 145:301–321, 1997.

[11] A. A. Johnson and T. E. Tezduyar. Advanced mesh generation and update methods for 3D flow simulations. *Computational Mechanics*, 23:130–143, 1999.

[12] K. Takizawa, T. E. Tezduyar, A. Buscher, and S. Asada. Space–time interface-tracking with topology change (ST-TC). *Computational Mechanics*, 54:955–971, 2013.

[13] K. Takizawa, T. E. Tezduyar, A. Buscher, and S. Asada. Space–time fluid mechanics computation of heart valve models. *Computational Mechanics*, 54:973–986, 2014.

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

[15] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annual Review of Fluid Mechanics*, 37:239–261, 2005.

[16] D. M. McQueen and C. S. Peskin. Computer-assisted design of butterfly bileaflet valves for the mitral position. *Scandinavian journal of thoracic and cardiovascular surgery*, 19(2):139–148, 1985.

[17] R. P. Beyer. A computational model of the cochlea using the immersed boundary method. *Journal of Computational Physics*, 98(1):145–162, 1992.

[18] R. Dillon, L. Fauci, and D. Fogelson, A.and Gaver III. Modeling biofilm processes using the immersed boundary method. *Journal of Computational Physics*, 129(1):57–73, 1996.

[19] L. J. Fauci and C. S. Peskin. A computational model of aquatic animal locomotion. *Journal of Computational Physics*, 77(1):85–108, 1988.

[20] R. Glowinski, T.-W. Pan, T. I. Hesla, and D. D. Joseph. A distributed Lagrange multiplier/-fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 25(5):755–794, 1999.

[21] R. Glowinski, T. W. Pan, T. I. Hesla, D. D. Joseph, and J. Périaux. A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: Application to particulate flow. *Journal of Computational Physics*, 169(2):363–426, 2001.

[22] R. Glowinski and Y. Kuznetsov. Distributed Lagrange multipliers based on fictitious domain method for second order elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 196:1498–1506, 2007.

[23] L. Zhang, A. Gerstenberger, X. Wang, and W. K. Liu. Immersed finite element method. *Computer Methods in Applied Mechanics and Engineering*, 193:2051–2067, 2004.

[24] W. K. Liu, D. W. Kim, and S. Tang. Mathematical foundations of the immersed finite element method. *Computational Mechanics*, 39(3):211–222, 2007.

[25] X. S. Wang, L. T. Zhang, and W. K. Liu. On computational issues of immersed finite element methods. *Journal of Computational Physics*, 228(7):2535–2551, 2009.

[26] X. Wang and L. T. Zhang. Modified immersed finite element method for fully-coupled fluid–structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 267:150–169, 2013.

[27] H. Casquero, C. Bona-Casas, and H. Gomez. A NURBS-based immersed methodology for fluid–structure interaction. *Computer Methods in Applied Mechanics and Engineering*, 284:943–970, 2015.

[28] T. Rüberg and F. Cirak. Subdivision-stabilised immersed B-spline finite elements for moving boundary flows. *Computer Methods in Applied Mechanics and Engineering*, 209–212:266–283, 2012.

[29] C. Kadapa, W. G. Dettmer, and D. Perić. A stabilised immersed boundary method on hierarchical b-spline grids for fluid–rigid body interaction with solid–solid contact. *Computer Methods in Applied Mechanics and Engineering*, 318:242–269, 2017.

[30] J. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 36:9–15, 1971.

[31] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, and T. J. R. Hughes. An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves. *Computer Methods in Applied Mechanics and Engineering*, 284:1005–1053, 2015.

[32] F. Xu, D. Schillinger, D. Kamensky, V. Varduhn, C. Wang, and M.-C. Hsu. The tetrahedral finite cell method for fluids: Immersogeometric analysis of turbulent flow around complex geometries. *Computers & Fluids*, 141:135–154, 2016.

[33] T. J. R. Hughes, L. Mazzei, and K. E. Jansen. Large eddy simulation and the variational multiscale method. *Computing and Visualization in Science*, 3:47–59, 2000.

[34] T. J. R. Hughes, L. Mazzei, A. A. Oberai, and A. Wray. The multiscale formulation of large eddy simulation: Decay of homogeneous isotropic turbulence. *Physics of Fluids*, 13:505–512, 2001.

[35] T. J. R. Hughes, G. Scovazzi, and L. P. Franca. Multiscale and stabilized methods. In E. Stein, R. de Borst, and T. J. R. Hughes, editors, *Encyclopedia of Computational Mechanics*, Volume 3: Fluids, chapter 2. John Wiley & Sons, 2004.

[36] Y. Bazilevs, V. M. Calo, J. A. Cottrel, T. J. R. Hughes, A. Reali, and G. Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 197:173–201, 2007.

[37] J.A. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 36:9–15, 1970.

[38] Y. Bazilevs and T. J. R. Hughes. Weak imposition of Dirichlet boundary conditions in fluid mechanics. *Computers & Fluids*, 36:12–26, 2007.

[39] M.-C. Hsu, C. Wang, F. Xu, A. J. Herrema, and A. Krishnamurthy. Direct immersogeometric fluid flow analysis using B-rep CAD models. *Computer Aided Geometric Design*, 43:143–158, 2016.

[40] C. Wang, F. Xu, M.-C. Hsu, and A. Krishnamurthy. Rapid B-rep model preprocessing for immersogeometric analysis using analytic surfaces. *Computer Aided Geometric Design*, 52:190–204, 2017.

[41] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[42] H. S. Udaykumar, R. Mittal, and P. Rampunggoon. Interface tracking finite volume method for complex solid–fluid interactions on fixed meshes. *International Journal for Numerical Methods in Biomedical Engineering*, 18(2):89–97, 2002.

[43] A. N. Brooks and T. J. R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32:199–259, 1982.

[44] T. E. Tezduyar. Stabilized finite element formulations for incompressible flow computations. *Advances in Applied Mechanics*, 28:1–44, 1992.

[45] T. E. Tezduyar and Y. Osawa. Finite element stabilization parameters computed from element matrices and vectors. *Computer Methods in Applied Mechanics and Engineering*, 190:411–430, 2000.

[46] M.-C. Hsu, Y. Bazilevs, V. M. Calo, T. E. Tezduyar, and T. J. R. Hughes. Improving stability of stabilized and multiscale formulations in flow simulations at small time steps. *Computer Methods in Applied Mechanics and Engineering*, 199:828–840, 2010.

[47] C. Johnson. *Numerical solution of partial differential equations by the finite element method*. Cambridge University Press, Sweden, 1987.

[48] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods, 2nd ed.* Springer, 2002.

[49] A. Ern and J. L. Guermond. *Theory and Practice of Finite Elements*. Springer, Berlin, 2004.

[50] Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Weak Dirichlet boundary conditions for wall-bounded turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 196:4853–4862, 2007.

[51] Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes. *Computer Methods in Applied Mechanics and Engineering*, 199:780–790, 2010.

[52] M. C. H. Wu, D. Kamensky, C. Wang, A. J. Herrema, F. Xu, M. S. Pigazzini, A. Verma, A. L. Marsden, Y. Bazilevs, and M.-C. Hsu. Optimizing fluid–structure interaction systems with immersogeometric analysis and surrogate modeling: Application to a hydraulic arresting gear. *Computer Methods in Applied Mechanics and Engineering*, 316:668–693, 2017.

[53] N. Kikuchi. A smoothing technique for reduced integration penalty methods in contact problems. *International Journal for Numerical Methods in Engineering*, 18(3):343–350, 1982.

[54] J. D. Sanders, J. E. Dolbow, and T. A. Laursen. On methods for stabilizing constraints over enriched interfaces in elasticity. *International Journal for Numerical Methods in Engineering*, 78:1009–1036, 2009.

[55] F. Liu and R. I. Borja. Stabilized low-order finite elements for frictional contact with the extended finite element method. *Computer Methods in Applied Mechanics and Engineering*, 199:2456–2471, 2010.

[56] L. De Lorenzis, İ. Temizer, P. Wriggers, and G. Zavarise. A large deformation frictional contact formulation using NURBS-based isogeometric analysis. *International Journal for Numerical Methods in Engineering*, 87:1278–1300, 2011.

[57] R. A. Sauer and L. De Lorenzis. A computational contact formulation based on surface potentials. *Computer Methods in Applied Mechanics and Engineering*, 253(0):369–395, 2013.

[58] G. Karypis, K. Schloegel, and V. Kumar. Parmetis: Parallel graph partitioning and sparse matrix ordering library. *Version 1.0, Dept. of Computer Science, University of Minnesota*, page 22, 1997.

[59] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. Gropp, D. Kaushik, et al. Petsc users manual revision 3.8. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States), 2017.

[60] J. Happel and H. Brenner. *Low Reynolds number hydrodynamics: with special applications to particulate media*, volume 1. Springer Science & Business Media, 2012.

[61] H. Casquero, C. Bona-Casas, and H. Gomez. A nurbs-based immersed methodology for fluid–structure interaction. *Computer Methods in Applied Mechanics and Engineering*, 284:943–970, 2015.

[62] A. Ten Cate, C. H. Nieuwstad, J. J. Derksen, and H. E. A. Van den Akker. Particle imaging velocimetry experiments and lattice-boltzmann simulations on a single sphere settling under gravity. *Physics of Fluids*, 14(11):4012–4025, 2002.

[63] H. Amini, W. Lee, and D. Di Carlo. Inertial microfluidic physics. *Lab on a Chip*, 14(15):2739–2761, 2014.

[64] J. Feng, H. H. Hu, and D. D. Joseph. Direct simulation of initial value problems for the motion of solid bodies in a newtonian fluid. part 2. couette and poiseuille flows. *Journal of fluid mechanics*, 277:271–301, 1994.

[65] Ansys FLUENT. Theory guide release 16.1. *Ansys Inc*, 2015.

CHAPTER 4. TRACKING MOVING OBJECTS IN FLUIDS: A SCALABLE, IMMERSED BOUNDARY METHOD ON OCTREES

This chapter includes a manuscript titled "Tracking moving objects in fluids: a scalable, immersed boundary method on octrees", in preparation for submission to *Computers & Fluids* journal, authored by Songzhe Xu, Alec Lofquist, Milinda Fernando, Ming-Chen Hsu, Baskar Ganapathysubramanian and Hari Sundar

Abstract

Control and localization of particles (cells, precipitates) in microfluidic devices are useful in biological processing, chemical reaction control, and for creating structured materials. It is known that particles localize (or focus) to equilibrium positions in microfluidic channels in the inertial flow regime. Our specific application problem of interest involves tracking the lateral motion (i.e. inertial migration) of a single rigid particle (which is a good approximation of suspensions in the dilute limit) as it traverses a microchannel decorated with obstacles. Numerically characterizing the trajectory of a particle as a function of particle size, channel geometry and flow conditions will greatly enable design of improved biomedical devices. This is challenging due to the full fluid–structure coupling and associated small time steps and adaptive (re)meshing requirements. The full fluid–structure coupling is due to the finite Reynolds number ($10 \leq Re \leq 100$) which necessitates solving the full Navier-Stokes equations. We showcase a scalable, adaptively refined, octree based finite element approach to track inertial migration of particles. Arbitrary particles are tracked by formulating the fluid–structure interaction using the immersed boundary strategy. This enables using a parallel, hierarchically refined octree mesh as the background mesh, with a variationally consistent immersed boundary formulation tracking the particle motion on this background mesh. Our framework is based on the DENDRO framework, which has previously shown excellent scaling behavior.

We show good scaling of the framework on Stampede2, TACC. This illustrates the potential of a variationally consistent immersed boundary approach for tracking particles in flow – a problem with a huge variety of applications.

4.1 Introduction

Control and localization of particles (cells, precipitates) in aqueous flow are useful in biological processing, chemical reaction control, and for creating structured materials. The controlled motion and localization of cells and particles can automate cellular sample preparation and bio-sensing. Some examples include fast identification of *e. coli* in water, robust removal of circulating tumor cells from the blood plasma and fast separation of cells types for rapid flow cytometry and subsequent identification/tagging for genomic analysis. The precise, efficient and cheap localization of a heterogeneous collection of cells in a fluid medium is a foundational challenge in science and engineering. A general (computationally informed) strategy for passive control of particle localization in microfluidic channels will be transformative to this field.

Researchers have recently discovered [1] and demonstrated [2, 3] the ability to passively engineer the cross-sectional shape of a fluid *(without particles in it)* using the notion of inertial flow deformations induced by **sequences of pillars** that disrupt the flow. This is a *purely passive approach* for flow control that relies on flow physics around bluff bodies. Since these transformations provide a deterministic mapping of fluid elements from upstream to downstream of a pillar, one can sequentially arrange pillars to apply the associated nested maps and therefore program complex fluid structures. This idea has been rapidly picked up by the microfluidics and manufacturing community to make structured particles and fibers [4–7] and for reagent recovery in medical diagnostic devices [8].

Studies have shown that this passive flow control paradigm can be extended to passively localize ***particles*** in fluid flow using a sequence of obstacles that differentially act on various sized particles (based on size and location). While the localization (or 'focusing') of particles in unobstructed microfluid channels is well known, the behavior (and control) of localization of particles

in **obstructed microfluidic channels** is a very novel problem with a rich physical underpinning. Segre and Silberberg [9, 10] first experimentally observed the phenomena of focusing of particles in a straight channel (or tube) flow. Particles moving in such flows undergo a lateral motion across the flow streamlines until they reach a stable equilibrium located between the channel centerline and the confining walls. Subsequent theoretical studies provided a general understanding of the lift forces and how the structure of lift forces structure depends on the particle size, channel dimensions and flow rate (or Reynolds number). However, the precise calculation needed to design devices that can exploit the migration of particles in flow, such as the separation, concentration, and sorting of cells and biomolecules with high specificity requires highly accurate force calculations, which essentially becomes a computational exercise.

Our particular problem of interest is to track the lateral migration of a single, rigid particle as it traverses a microchannel that is decorated with a pillar obstacle (see Figure 4.1). Our intent is to understand how initial release location, as well as particle size and pillar geometry affect migration patterns.

Tracking particle motion in inertial flows (especially in obstructed geometries) is a computationally daunting proposition. This is further complicated by that fact that the construction of migration maps for particles (as a function of particle location, flow conditions, and particle size) requires **several thousands of simulations** tracking individual particles. This calls for the development of an efficient, scalable approach for single particle tracking in fluids. We bring together three distinct elements to accomplish this: (a) a parallel octree based adaptive mesh generation framework, (b) a variational multiscale (VMS) based treatment that enables flow condition agnostic simulations (laminar or turbulent) [11], and (c) a variationally consistent immersed boundary method (IBM) to efficiently track moving particles in a background octree mesh [12]. This project builds on our existing codes for adaptive meshing (DENDRO) and Finite Elements (TALYFEM). *These generic names are used for the double-blind review policies. The real names along will be replaced for the final version.* In the next section, we give a brief introduction to the formulation of our target problem followed by a brief introduction of the immersed boundary method. We then present our

adaptive meshing framework that is tailored for the immersed boundary method. We wrap up with experiments demonstrating the scalability of our code.

4.2 Target problem

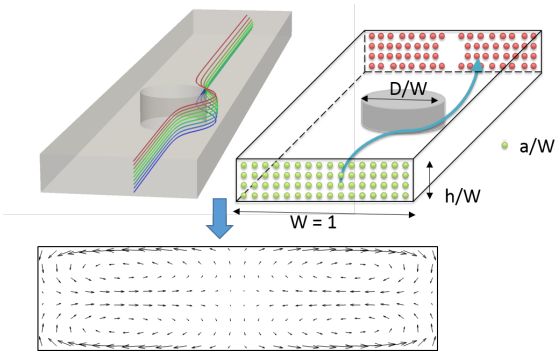


Figure 4.1: An illustration of the canonical target problem. Following standard practice in fluid dynamics, we normalize length scales by the channel width, W, and consider all physical variables in dimensionless quantities. This allows broad usability of the resulting computations, due to kinematic and dynamic similarity principles. The canonical problem is parametrized by 5 variables: (a) the size of the particle (a), (b) the location, δ and diameter, D of the pillar, (c) the flow speed, characterized in terms of the Reynolds number ($\mathfrak{R}$), and (d) the height of the microchannel, h.

We are interested in tracking the motion and lateral forces acting on a single particle—of size a—released at discrete points of the inlet (points shown in green in Figure 4.1). As an individual particle flows down the channel (of width W), it is affected by the spatially varying flow field caused by obstruction due to the pillar (of diameter D). Note that for typical particle sizes ($a/W \geq 0.1$), the moving particle itself causes changes to the flow field (so called *blockage effect*). We are particularly interested in reporting the net lateral displacement as a function of initial release location at the inlet. We consider a finite distance downstream (typically $6D$ downstream of the pillar, due to manufacturability constraints) across which we track the particle motion. The particle displacement is reported as a vector field (Figure 4.1). Additionally, the time history of the lateral forces acting on the particle will be reported.

For each set of parameters ($a, W, D, \mathfrak{R}, h$), we hope to simulate the time evolution of the particle-fluid interaction in the domain. Our preliminary results show that it is necessary to have a refined

mesh close to the pillar surface, the particle surface as well as the channel walls to fully resolve the fluid velocity features. Based on the DENDRO framework, we anticipate requiring `256x256x1024` $\sim 70 \times 10^6$ hexahedral elements to discretize the domain. Each time step requires solving the Navier-Stokes equations with no-slip boundary conditions on the particle and the channel walls. Once the velocity field (due to the interaction between the particle and pillar and fluid) is computed, the inertial forces on the particle are computed by performing a surface integration of the fluid stress on the particle surface. This is then used to update the location of the particle for the next time step.

The dimensionless Navier–Stokes equations for incompressible flow is written as

$$\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \boldsymbol{\nabla}\mathbf{u} - \mathbf{f}\right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma} = \mathbf{0} \,, \tag{4.1}$$

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0 \,, \tag{4.2}$$

where $\mathbf{u}$ and $\mathbf{f}$ are the flow velocity and the external force, respectively. The stress and strain-rate tensors are defined respectively as

$$\boldsymbol{\sigma}(\mathbf{u}, p) = -p\,\mathbf{I} + 2\frac{1}{\mathfrak{R}}\,\boldsymbol{\varepsilon}(\mathbf{u}) \,, \tag{4.3}$$

$$\boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}\left(\boldsymbol{\nabla}\mathbf{u} + \boldsymbol{\nabla}\mathbf{u}^{\mathrm{T}}\right) \,, \tag{4.4}$$

where p is the pressure, $\mathbf{I}$ is an identity tensor. The problem (4.1)–(4.4) is accompanied by suitable boundary conditions, defined on the boundary of the fluid domain, $\Gamma = \Gamma^{\mathrm{D}} \cup \Gamma^{\mathrm{N}}$:

$$\mathbf{u} = \mathbf{u_g} \qquad \text{on } \Gamma^{\mathrm{D}} \,, \tag{4.5}$$

$$-p\,\mathbf{n} + 2\frac{1}{\mathfrak{R}}\,\boldsymbol{\varepsilon}(\mathbf{u})\,\mathbf{n} = \mathbf{h} \qquad \text{on } \Gamma^{\mathrm{N}} \,, \tag{4.6}$$

where $\mathbf{u_g}$ denotes the prescribed velocity at the Dirichlet boundary Γ^{D}, $\mathbf{h}$ is the traction vector at the Neumann boundary Γ^{N}, and $\mathbf{n}$ is the unit normal vector pointing in the wall-outward direction.

Consider a collection of disjoint elements $\{\Omega^e\}$, $\cup_e \Omega^e \subset \mathbb{R}^d$. The fluid domain is covered by the closure of the collection: $\Omega \subset \cup_e \overline{\Omega^e}$. Note that Ω^e is not necessarily a subset of Ω with the immersed boundary method. Let $\mathcal{V}_u^h$ and $\mathcal{V}_p^h$ be the finite-dimensional spaces of discrete test functions and trial solutions for velocity and pressure, which are denoted as superscript h, and represent resolved

scales (coarse scale) produced by the finite element discretization. The strong problem (4.1)–(4.6) may be recast in a weak form and posed over these discrete spaces to produce the following semi-discrete problem (using the VMS modeling approach): Find $\mathbf{u}^h \in \mathcal{V}_u^h$ and $p^h \in \mathcal{V}_p^h$ such that for all $\mathbf{w}^h \in \mathcal{V}_u^h$ and $q^h \in \mathcal{V}_p^h$:

$$B^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) - F^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}\right) = 0\,. \tag{4.7}$$

The bilinear form B^{VMS} and the load vector F^{VMS} are given as

$$\begin{aligned}
&B^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) \\
&= \int_\Omega \mathbf{w}^h \cdot \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{u}^h\right) d\Omega \\
&+ \int_\Omega \boldsymbol{\nabla}\mathbf{w}^h : \boldsymbol{\sigma}\left(\mathbf{u}^h, p^h\right) d\Omega \\
&+ \int_\Omega q^h\, \boldsymbol{\nabla}\cdot\mathbf{u}^h\, d\Omega \\
&- \sum_e \int_{\Omega^e\cap\Omega} \left(\mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{w}^h + \boldsymbol{\nabla} q^h\right)\cdot \mathbf{u}'\, d\Omega \\
&- \sum_e \int_{\Omega^e\cap\Omega} p'\, \boldsymbol{\nabla}\cdot\mathbf{w}^h\, d\Omega \\
&+ \sum_e \int_{\Omega^e\cap\Omega} \mathbf{w}^h \cdot (\mathbf{u}'\cdot\boldsymbol{\nabla}\mathbf{u}^h)\, d\Omega \\
&- \sum_e \int_{\Omega^e\cap\Omega} \boldsymbol{\nabla}\mathbf{w}^h : (\mathbf{u}'\otimes\mathbf{u}')\, d\Omega,
\end{aligned} \tag{4.8}$$

and

$$F^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}\right) = \int_\Omega \mathbf{w}^h\cdot\mathbf{f}\, d\Omega + \int_{\Gamma^N} \mathbf{w}^h\cdot\mathbf{h}\, d\Gamma\,, \tag{4.9}$$

where primes denote the unsolved scales (fine scale) that need to be modeled, and their effect needs to be added onto the coarse scale. $\mathbf{u}'$ is defined as

$$\mathbf{u}' = -\tau_{\text{M}}\left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h\cdot\boldsymbol{\nabla}\mathbf{u}^h - \mathbf{f} - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}\left(\mathbf{u}^h, p^h\right)\right)\,, \tag{4.10}$$

and p' is given by

$$p' = -\tau_{\text{C}}\,\boldsymbol{\nabla}\cdot\mathbf{u}^h\,. \tag{4.11}$$

$\mathbf{u}'$ and p' are approximated by the residuals of momentum equation and continuity equation, respectively, and τ_M and τ_C are corresponding coefficients with the definitions in [11]. Equations (4.8)–(4.11) feature the VMS formulation of Navier-Stokes equations of incompressible flows [11]. The additional terms added onto the standard weak Galerkin form can be interpreted as a combination of streamline/upwind Petrov Galerkin (SUPG) stabilization and VMS large-eddy simulation of turbulence modeling.

The particle evolution is written as

$$\begin{aligned} m\frac{d\boldsymbol{V}}{dt} &= \boldsymbol{F} \\ J\frac{d\boldsymbol{\omega}}{dt} &= \boldsymbol{T} \end{aligned} \tag{4.12}$$

where $\boldsymbol{V} = [u_p, v_p, w_p]^T$ and $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ are the particle linear and angular velocities, and $\boldsymbol{F} = [F_x, F_y, F_z]^T$ and $\boldsymbol{T} = [\tau_x, \tau_y, \tau_z]^T$ are the force and torque acting on the particle. The force is computed as the surface integral of the fluid stress over the particle surface, and an explicit time-stepper is used to update the particle location and velocity.

4.3 Immersed boundary method

The immersed boundary method (IBM) was first introduced by Peskin in the context of fluid-structure interaction (FSI) for a heart simulation with associated blood flow to avoid remeshing when the solid body deformed [13, 14]. The IBM embeds the solid geometry into a background Cartesian mesh without conforming them to each other, and the effect of the immersed boundary on the fluid field has to be formulated by imposing the boundary conditions of the immersed geometry and distributed on the background Cartesian mesh. Since the IBM does not require a conforming mesh, it becomes computationally convenient to track the motion of particles of arbitrary shape while avoiding a cumbersome boundary fitted (re)meshing process.

The implementation of the IBM requires some refinement of the background mesh across the immersed surface to better capture the shape of the interface as well as to resolve the no-slip boundary condition. This is accomplished by using selective quadrature (i.e. only using those gauss

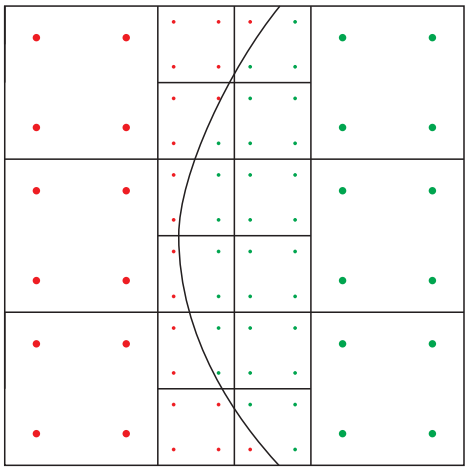


Figure 4.2: A schematic of the volume assembly in the IBM method. We loop over each element and each Gauss point within each element. An in-out test is performed to identify whether that Gauss point is lies inside the particle (red points) or inside the fluid (green points). Only the Gauss points in the fluid domain are used to assemble the elemental matrices.

points that lie in the fluid and not inside the immersed particle). This necessitates performing an in/out test to determine the Gauss points inside the fluid domain (red dots) on which we assemble, while discarding the Gauss points inside the object (green dots), as shown in Figure 4.2.

The no-slip boundary condition (which is a Dirichlet boundary condition) is converted into an equivalent Neumann condition (in the sense of the Nitsche method [15]). Thus, we perform a surface integral over the immersed boundary to weekly impose the Dirichlet boundary condition of the immersed boundary [16–18]. Assuming the immersed boundary Γ_{I} is decomposed into N_{eb} surface elements each denoted by $\Gamma_{\mathrm{I}}^{\mathrm{b}}$, the semi-discrete problem becomes

$$
\begin{aligned}
& B^{\mathrm{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) - F^{\mathrm{VMS}}\left(\{\mathbf{w}^h, q^h\}\right) \\
& - \sum_{b=1}^{N_{eb}} \int_{\Gamma_{\mathrm{I}}^{\mathrm{b}} \cap \Gamma_{\mathrm{D}}} \mathbf{w}^h \cdot \left(-p^h \mathbf{n} + 2\frac{1}{\mathfrak{R}} \boldsymbol{\varepsilon}(\mathbf{u}^h)\, \mathbf{n} \right) d\Gamma \\
& - \sum_{b=1}^{N_{eb}} \int_{\Gamma_{\mathrm{I}}^{\mathrm{b}} \cap \Gamma^{D}} \left(2\frac{1}{\mathfrak{R}} \boldsymbol{\varepsilon}(\mathbf{w}^h)\, \mathbf{n} + q^h \mathbf{n} \right) \cdot \left(\mathbf{u}^h - \mathbf{u}_{\mathbf{g}} \right) d\Gamma \\
& + \sum_{b=1}^{N_{eb}} \int_{\Gamma_{\mathrm{I}}^{\mathrm{b}} \cap \Gamma^{D}} \tau^B \mathbf{w}^h \cdot \left(\mathbf{u}^h - \mathbf{u}_{\mathbf{g}} \right) d\Gamma \;=\; 0\,. \qquad (4.13)
\end{aligned}
$$

The boundary terms added to the governing equation are the second, third and last line in Eq. 4.13, and a detailed interpretation of different terms can be found in [16]. Only the penalty-like stabiliza-

tion parameter, τ^B, is a heuristic that has to be appropriately chosen. We use the definition proposed in [19], which scales the stabilization parameter as $\tau^{\mathrm{B}} = \mathrm{C}h/\Delta t$, where C is a positive constant, h is the size of the cut element, and Δt is the time step. The boundary terms are imposed onto the surface Gauss points, which are then interpolated by their background Cartesian grids as shown in Figure 4.3. In this way we can apply the Dirichlet boundary condition on the immersed boundary of the object to the fluid field.

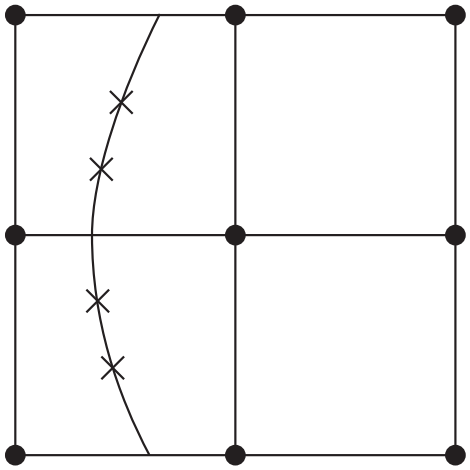


Figure 4.3: Schematic showing how the surface assembly of IBM is performed. The triangulated surface mesh is used to identify surface Gauss points (the 'X' locations). The surface integral terms (i.e. the last three terms in Eq. 3.16) are computed at these surface Gauss points, and then distributed to the nodal locations.

When we move an object, there would be some space that is previously occupied by the object becoming the space of the fluid domain due to the object motion. We call nodes in such space as freshly-cleared nodes. We have no fluid history on those nodes. As a result, we interpolate those nodes using the solution of their neighbor nodes in the fluid domain and boundary values of nearby points on the object surface to complete the solution of fluid field. We denote the union of the variables of fluid field as $\mathbf{U} = \{\mathbf{u}, p\}$, and the union of the variables of the moving object surface as $\mathbf{V} = \{\mathbf{v}, p\}$. The interpolation of the variables of freshly-cleared nodes is written as

$$\mathbf{U}^{\text{moved-out}} = \frac{\sum_{k=1}^{N^{\mathrm{f}}} \frac{1}{d_k^{\mathrm{f}}} \mathbf{U}_k + \sum_{k=1}^{N^{\mathrm{s}}} \frac{1}{d_k^{\mathrm{s}}} \mathbf{V}_k}{\sum_{k=1}^{N^{\mathrm{f}}} \frac{1}{d_k^{\mathrm{f}}} + \sum_{k=1}^{N^{\mathrm{s}}} \frac{1}{d_k^{\mathrm{s}}}}, \tag{4.14}$$

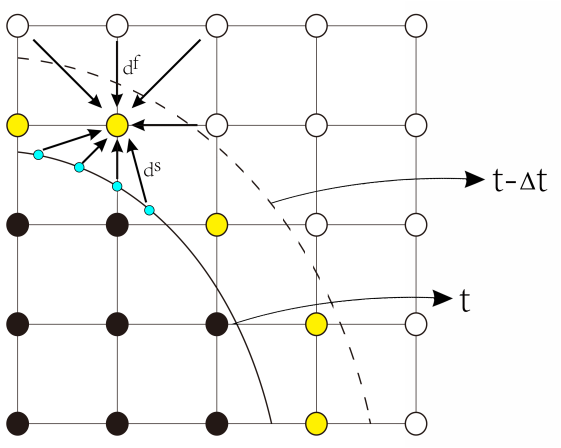


Figure 4.4: Interpolation of freshly-cleared node.

where d^{f} and d^{s} are the distances of corresponding nodes in fluid domain and points on the object surface to the freshly-cleared node, and N^{f} and N^{s} are their numbers.

4.4 Scalable IBM on octree meshes

While the concept of adaptive space partitions is well studied, developing such methods for the immersed boundary method on large distributed systems presents significant challenges. This work builds on our existing methods for performing large-scale finite element computations using octree-refined meshes. We have extended this work to support the particle localization simulations. We provide a brief description on building the octree mesh in parallel and performing FEM computations and refer to [20] for additional details.

While the elemental matrix assemblies are done using TalyFEM described in the next section, DENDRO provides the adaptive mesh refinement (AMR) and all parallel data-structures. For this project, DENDRO was extended to support meshes with *holes* in it. This is because of the presence of pillars in the channels where we do not need to solve. An example of such a mesh is shown in Figure 4.5.

The main steps in building and maintaining an adaptive mesh in DENDRO are:

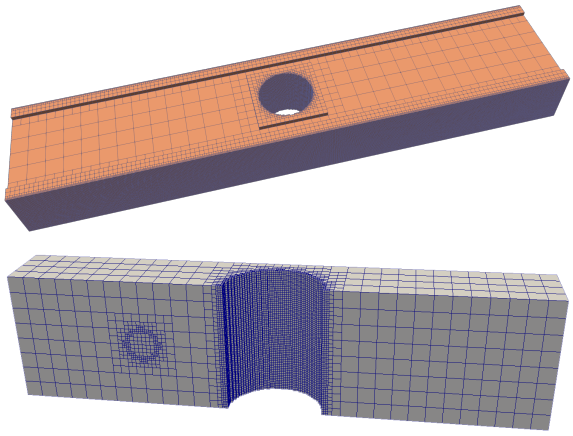


Figure 4.5: (**top**) An example of the adaptive mesh for the target geometry (§4.2) created by DENDRO. (**bottom**) A slice through our 3D mesh to illustrate the refinement around the pillar and particle.

Refinement: The sparse grid is constructed based on the geometry. Proceeding in a top-down fashion, a cell is refined if a surface (pillar/particle) passes through it. During the same step, we also determine if the cell is completely inside the pillar, and eliminate it from the mesh in that case. Since the refinement happens in an element-local fashion, this step is embarrassingly parallel. The user passes a function that given coordinates, x, y, z returns the distance from the pillar(s). The eight corners of an octant are tested using this function. If all 8 points have a positive distance (outside), then we retain this element, but do not refine further. If all 8 points have a negative distance (inside), then this element is removed from the mesh. If some of the corners of the octant are inside and others outside, then this octant is refined. This is repeated till we reach the desired level of refinement is achieved. In distributed memory, all processes start from the root and refine until at least p octants requiring further refinement are produced. Then using a weighted space-filling-curve based partitioning, we partition the octants. Note that we do not communicate the octants as every process has a copy of the octants, and all that needs to be done at each process is to retain a subset of the current octants and recurse. Since we use FEM, a 2:1 balancing is enforced following the refinement operation.

2:1 Balancing: We enforce a condition in our distributed octrees that no two neighboring octants differ in size by more than a factor of two. This makes subsequent operations simpler without affecting the adaptive properties. Our balancing algorithm is similar to existing approaches for balancing octrees [21–23] with the added aspect that it does not generate octants if the ancestor does not exist in the input. This is done to ensure that the *holes* are not filled in. The algorithm proposed by Bern [21] is easily extensible to support this case, as we simply need to skip adding balancing octants that violate the criteria.

Partition: Refinement and the subsequent 2:1-balancing of the octree can result in a non-uniform distribution of elements across the processes, leading to load imbalance. This is particularly challenging in the presence of holes, as this can make the mesh heavily load-imbalance. The Hilbert ordering enables us to equipartition the elements by performing a parallel scan on the number of elements on each process followed by point-to-point communication to redistribute the elements. The presence of holes in the domain, does not adversely affect this as the partition only tries to equally divide the elements across the processes. The immersed boundary can affect the performance, as it is likely localized on a small subset of processes. These processes will be the only ones that will be performing the surface assembly (§4.4.2). This can be accommodated by estimating the relative cost of volume *vs.* surface assembly and performing a weighted partition of the elements. This is not currently done for the results presented in the paper, and we hope to have this completed soon.

Meshing: By meshing we refer to the construction of the (numerical) data structures required for FEM computations from the (topological) octree data. DENDRO already has efficient implementations for building the required neighborhood information and for managing overlapping domains between processors (*ghost* or *halo* regions). The key difference with our previous applications is the requirement to handle meshes with holes, as all neighbors might not be present in the mesh. This also complicates the process of applying boundary conditions. We added support for defining *subdomains* within DENDRO. The subdomains are defined using a function that takes a coordinate (x, y, z) as input and returns `true` or `false` depending on whether that coordinate is part of the

subdomain or not. The subdomain leverages the core mesh data-structure and additionally defines a unique mapping for nodes that are part of the subdomain. It also keeps track of which nodes belong to subdomain boundaries. Therefore, subdomains have a small overhead and store significantly less data that the main mesh data-structure. For our target application, it is important to identify both the external (domain) boundary as well as the internal boundary (the pillar surface). The subdomain stores two `bits` to keep track of whether a node is non-boundary, external, or internal boundary.

4.4.1 Matvec - integration with TALYFEM

We previously developed code for calculating the elemental matrix and vector for solving Navier-Stokes with IBM in our in-house FEM framework, TALYFEM, which is designed for arbitrary unstructured meshes. We chose to integrate the core of our in-house framework with DENDRO to avoid re-implementing the NS+IBM kernel.

Node coordinates and elemental connectivity are implicit in the octree's structure, so DENDRO recalculates these values on the fly as the octree is traversed. In order to adapt the TALYFEMmesh data structure to DENDRO, we store these values explicitly. To do so in full requires extra memory and a synchronization step after the octree mesh is (re)generated. To avoid the memory overhead, we instead integrate these two meshes by considering a mesh containing a single hexahedral element. As we iterate through the octree mesh for assembly we re-position the nodes in the hexahedral element in TALYFEMto match the octree element from DENDRO, allowing us to use TALYFEM's routines to calculate the elemental matrix and vector. We also copy nodal data (velocity and pressure) from DENDRO's buffers to support the assembly code. This allowed us to reuse our existing assembly implementation in TALYFEMwith almost no changes.

By default, TALYFEM recalculates the isoparametric to physical mapping at each integration point, as these values change depending on the shape of the physical element. Since the octree mesh has only one possible element shape, we pre-calculate and cache these values. During initialization, we create a fake element at each level in the octree and evaluate the basis functions at the typical

Gaussian quadrature points. When the assembly code needs to access these values, we pull them from the corresponding level in the cache instead of recalculating them at each element.

4.4.2 Sampling the immersed boundary & adding corrections

The object boundary is generated as a triangulated mesh. Surface integration points are then calculated for each triangle element using standard Gaussian quadrature, as well as other necessary parameters such as the unit normal and boundary value of velocity at each Gauss point. The surface Gauss points are then sorted and distributed. This is done by associating each surface Gauss point with corresponding bottom left node of an octree element with the maximum refinement. This octree element can then be aligned on the space-filling curve, and the process that contains the real element can be easily found by the partitioning of the space-filling curve. To find the actual background element containing the Gauss point, we loop over all the elements in the process to check if the element is an ancestor of the associated element of the Gauss point or if they are exactly the same element. Boundary conditions weakly imposed on the Gauss points can be then interpolated by (distribute to) the nodes of the background element. Since the elements and the surface Gauss points are both sorted and aligned on the space-filling curve, we can loop over them simultaneously with an efficiency of $O(m+n)$ instead of a nested loop with an efficiency of $O(m\times n)$, where m is the number of elements in the background mesh and n is the number of surface Gauss points.

4.4.3 Timestepping and particle evolution

The time-dependent Navier–Stokes equation is solved with a fully implicit scheme (backward Euler time-stepping). The backward Euler time-stepper for the NS equation is given as

$$\frac{\partial \boldsymbol{u}}{\partial t} = \frac{\boldsymbol{u}^n - \boldsymbol{u}^{n-1}}{\Delta t} = \mathcal{L}(\boldsymbol{u}^n, p^n), \tag{4.15}$$

where the operator $\mathcal{L}(\boldsymbol{u}^n, p^n)$ represents all the other terms except the time-dependent term evaluated at the current time step in the Navier–Stokes Eq (3.1). Δt is selected to follow the CFL condition.The (non)linear solution procedure is taken care of by PETSC [24]. We utilize the SNES construct (line

search quasi-Newton), which uses the KSP construct , specifically the BCGS solver. An additive Schwarz preconditioner (asm) is used to enable parallel preconditioning and solving on decomposed sub-domains

Once the fluid field is solved, a surface integral over the immersed boundary is then performed to calculate the surface force and torque (**F** and **T**) that is exerted on the object by the fluid.

$$\begin{aligned}\mathbf{F} = &\sum_{b=1}^{N_{eb}} \int_{\Gamma_{\mathrm{I}}^{\mathrm{b}} \cap \Gamma^D} \boldsymbol{\sigma}(\mathbf{u}^h, p^h) \cdot \mathbf{n} d\Gamma \\ &- \sum_{b=1}^{N_{eb}} \int_{\Gamma_{\mathrm{I}}^{\mathrm{b}} \cap \Gamma^D} \tau^B (\mathbf{u}^h - \mathbf{u_g}) d\Gamma,\end{aligned} \tag{4.16}$$

$$\begin{aligned}\mathbf{T} = &\sum_{b=1}^{N_{eb}} \int_{\Gamma_{\mathrm{I}}^{\mathrm{b}} \cap \Gamma^D} \mathbf{r} \times \left(\boldsymbol{\sigma}(\mathbf{u}^h, p^h) \cdot \mathbf{n}\right) d\Gamma \\ &- \sum_{b=1}^{N_{eb}} \int_{\Gamma_{\mathrm{I}}^{\mathrm{b}} \cap \Gamma^D} \mathbf{r} \times \tau^B (\mathbf{u}^h - \mathbf{u_g}) d\Gamma,\end{aligned} \tag{4.17}$$

where **r** is the distance vector from the particle centroid to any point on its surface. The last terms in Eq. 4.16 and Eq. 4.17 are the penalty-like term that are added onto the surface force calculation due to the weak imposition of boundary conditions. The total force acting on the object is the summation of the surface force and any external body forces (gravity & buoyancy).

The particle is modeled as a rigid body, and the motion is described as

$$\frac{d\mathbf{x}^{\mathrm{c}}}{dt} = \mathbf{v}^{\mathrm{c}}, \qquad \frac{d\mathbf{v}^{\mathrm{c}}}{dt} = \frac{\mathbf{F}}{m}, \tag{4.18}$$

$$\frac{d\boldsymbol{\theta}^{\mathrm{c}}}{dt} = \boldsymbol{\omega}^{\mathrm{c}}, \qquad \frac{d\boldsymbol{\omega}^{\mathrm{c}}}{dt} = \frac{\mathbf{T}}{J}, \tag{4.19}$$

where $\mathbf{x}^{\mathrm{c}}$ and $\boldsymbol{\theta}^{\mathrm{c}}$ are the linear and angular locations of the centroid of patricle, $\mathbf{v}^{\mathrm{c}}$ and $\boldsymbol{\omega}^{\mathrm{c}}$ are linear and angular velocities of centroid of particle, and m and J are the particle mass and moment of inertia, respectively. The coordinates **x** and velocities **v** at any point on the particle surface is computed as

$$\mathbf{x} = \mathbf{x}^{\mathrm{c}} + \mathbf{r}, \qquad \mathbf{v} = \mathbf{v}^{\mathrm{c}} + \boldsymbol{\omega}^{\mathrm{c}} \times \mathbf{r}. \tag{4.20}$$

In the discrete form, assuming the integral of the force and torque over the particle surface are constant during one time step, we have

$$\frac{(\mathbf{x}^{\mathrm{c}})^{n+1} - (\mathbf{x}^{\mathrm{c}})^{n}}{\Delta t} = \frac{(\mathbf{v}^{\mathrm{c}})^{n+1} + (\mathbf{v}^{\mathrm{c}})^{n}}{2}, \tag{4.21}$$

$$\frac{(\mathbf{v}^{\mathrm{c}})^{n+1} - (\mathbf{v}^{\mathrm{c}})^{n}}{\Delta t} = \frac{(\mathbf{F})^{n}}{m_i}, \tag{4.22}$$

$$\frac{(\boldsymbol{\theta}^{\mathrm{c}})^{n+1} - (\boldsymbol{\theta}^{\mathrm{c}})^{n}}{\Delta t} = \frac{(\boldsymbol{\omega}^{\mathrm{c}})^{n+1} + (\boldsymbol{\omega}^{\mathrm{c}})^{n}}{2}, \tag{4.23}$$

$$\frac{(\boldsymbol{\omega}^{\mathrm{c}})^{n+1} - (\boldsymbol{\omega}^{\mathrm{c}})^{n}}{\Delta t} = \frac{(\mathbf{T})^{n}}{m_i}. \tag{4.24}$$

The particle velocity is evaluated by an explicit forward Euler scheme, which requires small Δt to ensure accuracy and stability. Each object location is updated by the average velocities, which is essentially Crank–Nicolson scheme, and therefore more stable and accurate.

4.4.4 Intergrid transfers

It is essential that we need to change the refinement of the mesh to capture the evolving solution. In particular, the mesh has to be coarsened and refined with the moving particle. In the distributed memory setting, this can also imply a need to repartition and restore load-balance. Every few time steps, we remesh, based on the velocity of the moving particle. This is similar to the initial mesh generation and refinement, except that it is based on the current position of the particle. This is followed by the 2:1 balance enforcement and meshing. Once the new mesh is generated, we transfer the evolution solution from old mesh to the newly generate mesh using interpolations as needed. Since the intergrid transfers happen only between parent and child (for coarsening and refinement) or are unchanged, this can be performed on the old mesh using standard linear interpolation, followed by a simple repartitioning based on the new mesh. An example of the mesh for our target problem is shown in Figure 4.5 and examples of the adaptive mesh refinement following the moving particle are shown in Figure 4.6.

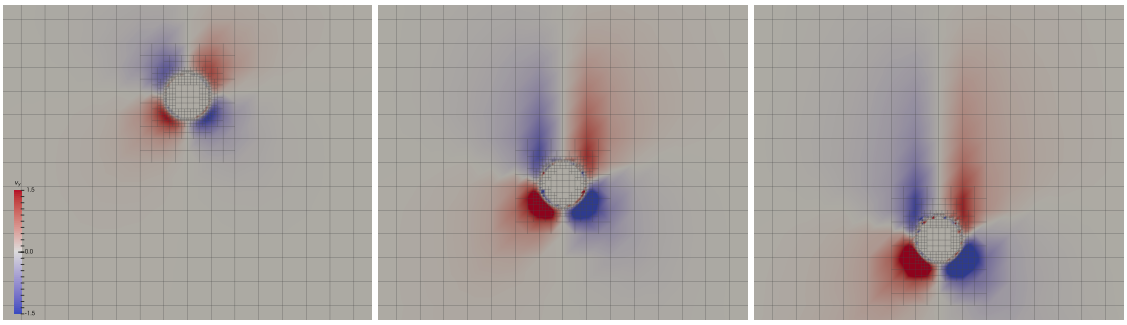


Figure 4.6: An example of the octree-grid along with the velocity along the y-axis being plotted at three different time-points. Notice the balanced 2:1 refinement as we move closer to the particle surface.

4.5 Experiments & Results

We perform tests on a canonical example of a sphere dropping in a domain of size $8 \times 8 \times 8$ with a constant downward force applied to it. For simplicity, we ignore the force of the fluid on the sphere, only allowing it to move by the constant external force. Some preliminary results can be found in Figure 4.6, and 4.7.

4.5.1 Implementation details

The DENDRO framework implemented in `C++` using `MPI` for distributed memory parallelism and `OpenMP` for shared memory parallelism. The TALYFEM framework is also implemented in `C++` with `MPI` and is used in this work for evaluating basis functions and interpolating nodal data to support assembly. Our code is tightly integrated with `PETSc v3.7` [24]'s distributed matrix and vector data-structures and utilizes its SNES and KSP solvers.

These tests were compiled and run on Stampede2's Skylake nodes. DENDRO, TALYFEM, and the main program were compiled with the Intel 17.0.4 compiler with the following flags: `-xCORE-AVX2 -axCORE-AVX512,MIC-AVX512 -O3 -DNDEBUG` (targetting the AVX2 instruction set, using static dispatch for AVX512 extension differences between Skylake vs KNL nodes, level 3 optimizations, and asserts disabled). We used the system PETSc 3.7 module, which was also compiled with the Intel 17 compiler with debugging disabled and the following optimization

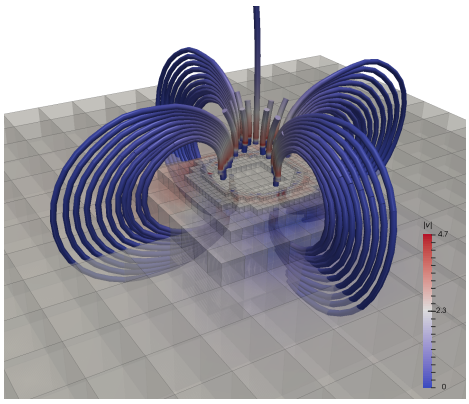


Figure 4.7: Representative streamlines around the moving sphere.

flags: `-xCORE-AVX2 -axMIC-AVX512,CORE-AVX512 -O2`. Timing information was reported using PETSc's logging framework.

4.5.2 Meshes/domains

We focus on showing scaling of the framework. We collect timing for the case of a dropping sphere (of size 1). The same domain size is employed. We run each case for 5 time steps, and use the same time step for all tests. We adaptively refine the mesh around the interface of the sphere three levels deeper than the rest of the background mesh, remeshing after each time step as the sphere moves. Note that such frequent remeshing is one of the challenges of our target application. The mesh is defined by a pair of minimum refinement l and maximum refinement h, where the background mesh element size ranges from $8/2^l$ to $8/2^h$ at the interface. We adjust the characteristic length of the surface mesh in sync with the refinement of the interface mesh, keeping a ratio of 1:2 for the surface triangle size to the interface element size. We run this experiment on five background/interface refinement levels: 4/7, 5/8, 6/9, 7/10, and 8/11. Each refinement level has roughly seven to eight times more degrees of freedom to solve for than the previous level, with 4/7 having 29,000 degrees of freedom and 8/11 reaching 70.2 million. We note that given specific l and h and the same sizes of domain and immersed object, the overall problem size is consistent

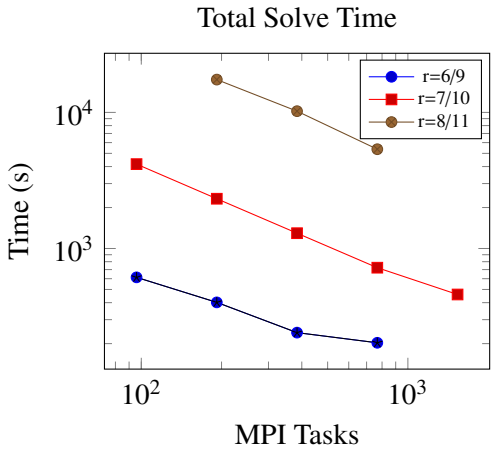


Figure 4.8: Total time (assembly + solve), summation of each time step for various mesh refinements.

independent of the number of processes being used for the simulation. To this effect, we believe presenting performance for different l/h combinations for different number of processes, in the style of a strong scaling is appropriate. Weak scalability can be observed in our strong scaling plots, across the different problem sizes. Unfortunately, we were unable to get additional sample points for the results to be convincing. Based off the two sample points, the weak scalability appears to be good, although additional sample points will allow us to make a stronger claim. Indeed, performing weak scaling for such complex real-world applications is harder because it is difficult to ensure that N/p (the grain size) stays relatively constant. Therefore, we ignore the analysis of weak scalability in this work.

4.5.3 Parallel Scalability

For our target application, the key goal is to be able to perform the simulations quickly, given the sheer number of simulations we need to perform. Given this, and the relatively moderate size of our problems, the focus is on strong scalability. We first present strong scalability results for the overall simulation, including the cost of remeshing in Figure 4.8 for three problem sizes. Overall our code scales well, with continued reductions in solve time. We report additional results to get a deeper

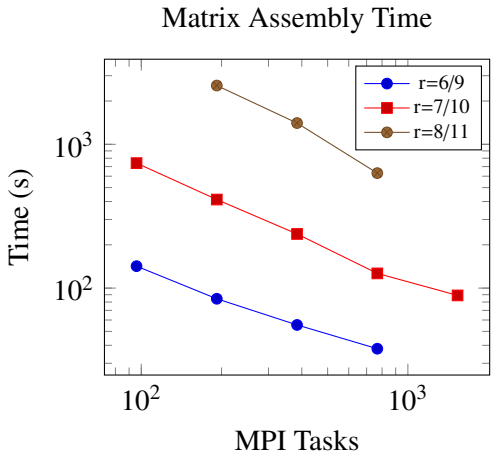


Figure 4.9: Matrix assembly time (volume + surface assembly) for various mesh refinements.

understanding of the performance and scalability of the different parts of our code. We present strong scalability results for Matrix assembly Figure 4.9 and Vector assembly Figure 4.10. Both methods scale reasonably well, but the overall time for matrix assembly is more expensive compared to the vector assembly. This is largely due to the complexity of the operator and we are working on switching to a matrix-free approach. We also report just the time spent in the remeshing stage. The remeshing stage refers to the combination of generating a new mesh, interpolating between two meshes and reinitializing the matrix, vector and solver. Effectively, this is the overhead paid for having good adaptivity. The scaling of remeshing, shown in Figure 4.13, is not as good as the other parts of the code, but the magnitude of time it takes is much smaller than solving the NS equations. Again, note that this is strong scaling, and the meshing code is sufficiently optimized (fast), making it much harder to demonstrate strong scalability across the full range. The actual solving of the linear system is implemented and optimized by PETSc depending on the KSP solver we choose, and we don't have much control of it. We observed it scales similarly as the total assembly.

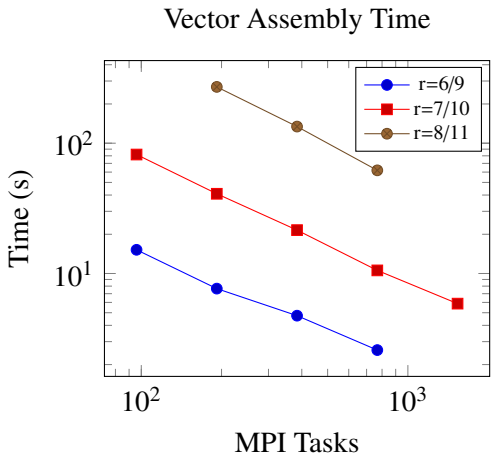


Figure 4.10: Vector assembly time (volume + surface assembly) for various mesh refinements.

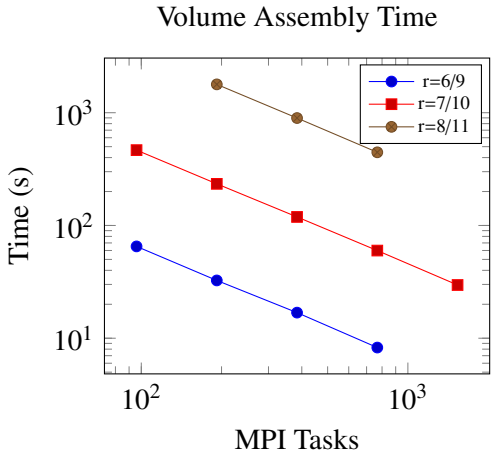


Figure 4.11: Volume assembly time (matrix + vector) for various mesh refinements.

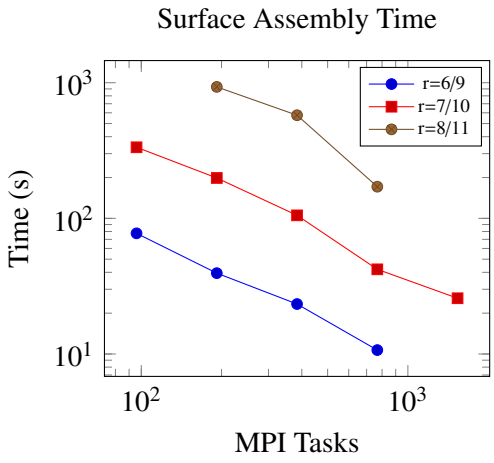


Figure 4.12: Surface assembly time (matrix + vector) for various mesh refinements.

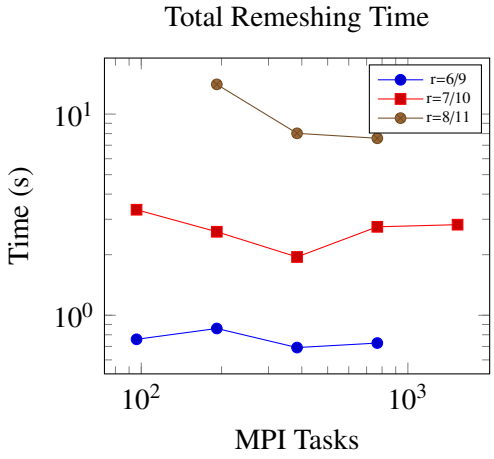


Figure 4.13: Total time for adaptive remeshing for various mesh refinements.

4.5.4 Overhead of immersed boundary corrections

As the geometry moves, we perform the in/out test in order to identify which background nodes are to be solved. Nodes which are fully inside the geometry (not on elements that are outside and intersect the object) are marked and set to have a Dirichlet boundary condition of zero. In addition, we must also redistribute the surface Gauss points to the appropriate processes as the mesh has been re-partitioned, which is difficult to achieve in an unstructured approach, as described in 4.4.2. In our experiments, we see this total bookkeeping time taking up to 10% of our total solve time and scaling poorly with the number of processes. This is due to a reliance on MPI all-gather routines in applying zero Dirichlet boundary condition on the fully inside nodes to simplify our initial implementation which we believe this can be improved, while the (re)distribution of surface Gauss points is fast in our observation. However, this does not appear to affect our overall scalability.

Applying the actual IBM corrections to the matrix and vector takes a significant amount of time - sometimes more than volume assembly, as shown in Figure 4.14 - but appears to scale better than volume assembly. This is likely because we weight non-interface background mesh elements the same as elements containing surface Gauss points when partitioning the mesh. This leads to a work imbalance where processes all perform roughly equal parts of volume assembly, but only some participate in surface assembly. We plan to introduce an elemental ”work factor” to the partitioning algorithm to address this imbalance in the future. This issue is also associated with the surface force integral over the immersed boundary, which we have ignored in these experiments by using a constant external force on the sphere. Surface assembly also does not use the cached basis function values (as volume assembly does), as each surface Gauss point may have a unique position relative to its background element.

4.6 Conclusions & Future directions

We showcased the performance of a scalable, IBM framework based on octree meshes for tracking particle localization in complex geometry microfluid channels. This framework allows us to efficiently construct the deformation maps for particles under a broad range of experimentally ac-

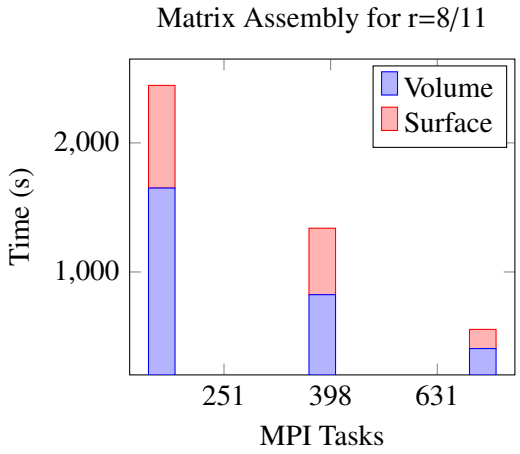


Figure 4.14: Total time spent in matrix assembly broken down by volume vs surface for refinement level 8/11.

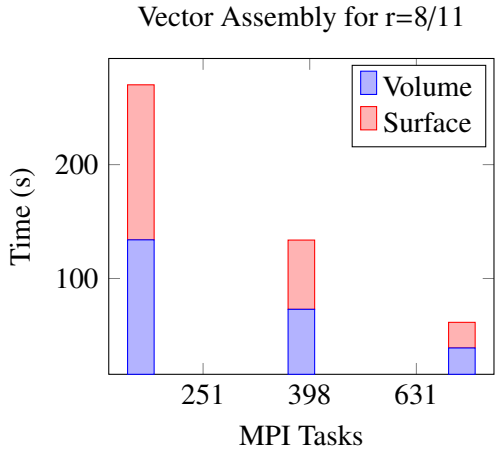


Figure 4.15: Total time spent in vector assembly broken down by volume vs surface for refinement level 8/11.

cessible parameters (as illustrated in Figure 4.1), which will result in a passive approach for particle localization. Our approach demonstrates excellent strong scalability for the overall solve time, even with frequent remeshing because our framework keeps the overhead of AMR relatively low, making the overall approach scalable. The surface assembly required in immersed boundary corrections is improved from the unstructured approach. Our immediate goals are to improve the performance for the matrix assembly and incorporate a dynamic load balancing that accounts for the additional work involved in the surface computations.

4.7 References

[1] H. Amini, E. Sollier, M. Masaeli, Y. Xie, B. Ganapathysubramanian, H. A. Stone, and D. Di Carlo. Engineering fluid flow using sequenced microstructures. *Nature communications*, 4:1826, 2013.

[2] D. Stoecklein, C.-Y. Wu, K. Owsley, Y. Xie, D. Di Carlo, and B. Ganapathysubramanian. Micropillar sequence designs for fundamental inertial flow transformations. *Lab on a Chip*, 14(21):4197–4204, 2014.

[3] D. Stoecklein, C.-Y. Wu, D. Kim, D. Di Carlo, and B. Ganapathysubramanian. Optimization of micropillar sequences for fluid flow sculpting. *Physics of Fluids*, 28(1):012003, 2016.

[4] J. K. Nunes, C.-Y. Wu, H. Amini, K. Owsley, D. Di Carlo, and H. A. Stone. Fabricating shaped microfibers with inertial microfluidics. *Advanced Materials*, 26(22):3712–3717, 2014.

[5] K. S. Paulsen, D. Di Carlo, and A. J. Chung. Optofluidic fabrication for 3d-shaped particles. *Nature communications*, 6, 2015.

[6] C.-Y. Wu, K. Owsley, and D. Di Carlo. Rapid software-based design and optical transient liquid molding of microparticles. *Advanced Materials*, 27(48):7970–7978, 2015.

[7] K. S. Paulsen and A. J. Chung. Non-spherical particle generation from 4d optofluidic fabrication. *Lab on a Chip*, 16(16):2987–2995, 2016.

[8] H. Amini, W. Lee, and D. Di Carlo. Inertial microfluidic physics. *Lab on a Chip*, 14(15):2739–2761, 2014.

[9] G. Segré and A. Silberberg. Behaviour of macroscopic rigid spheres in poiseuille flow part 1. determination of local concentration by statistical analysis of particle passages through crossed light beams. *Journal of fluid mechanics*, 14(1):115–135, 1962.

[10] G. Segré and A. Silberberg. Behaviour of macroscopic rigid spheres in poiseuille flow part 2. experimental results and interpretation. *Journal of fluid mechanics*, 14(1):136–157, 1962.

[11] Y. Bazilevs, V. M. Calo, J. A. Cottrel, T. J. R. Hughes, A. Reali, and G. Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 197:173–201, 2007.

[12] F. Xu, D. Schillinger, D. Kamensky, V. Varduhn, C. Wang, and M.-C. Hsu. The tetrahedral finite cell method for fluids: Immersogeometric analysis of turbulent flow around complex geometries. *Computers & Fluids*, 141:135–154, 2016.

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

[14] C. S. Peskin. Flow patterns around heart valves: a digital computer method for solving the equations of motion. *IEEE Transactions on Biomedical Engineering*, (4):316–317, 1973.

[15] J. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 36:9–15, 1971.

[16] Y. Bazilevs and T. J. R. Hughes. Weak imposition of Dirichlet boundary conditions in fluid mechanics. *Computers & Fluids*, 36:12–26, 2007.

[17] Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Weak Dirichlet boundary conditions for wall-bounded turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 196:4853–4862, 2007.

[18] Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes. *Computer Methods in Applied Mechanics and Engineering*, 199:780–790, 2010.

[19] M. C. H. Wu, D. Kamensky, C. Wang, A. J. Herrema, F. Xu, M. S. Pigazzini, A. Verma, A. L. Marsden, Y. Bazilevs, and M.-C. Hsu. Optimizing fluid–structure interaction systems with immersogeometric analysis and surrogate modeling: Application to a hydraulic arresting gear. *Computer Methods in Applied Mechanics and Engineering*, 316:668–693, 2017.

[20] H. Sundar, R. Sampath, and G. Biros. Bottom-up construction and 2:1 balance refinement of linear octrees in parallel. *SIAM Journal on Scientific Computing*, 30(5):2675–2708, 2008.

[21] M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quadtrees and quality triangulations. *International Journal of Computational Geometry & Applications*, 9(06):517–532, 1999.

[22] H. Sundar, R. S. Sampath, S. S. Adavani, C. Davatzikos, and G. Biros. Low-constant parallel algorithms for finite element simulations using linear octrees. In *SC'07: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. ACM/IEEE, 2007.

[23] C. Burstedde, L. C. Wilcox, and O. Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.

[24] E. Arge, A. M. Bruaset, and H. P. Langtangen, editors. *Efficient Management of Parallelism in Object Oriented Numerical Software Libraries*. Birkhäuser Press, 1997.

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

We have deployed the VMS to buoyancy-driven flow and verified and validated the framework with a Rayleigh–Bénard convection problem for both 2D and 3D cases in Chapter 2. Converged results in terms of heat transfer on hot walls are shown for each case. It indicates the ability to produce reasonable results with variation of seven orders of magnitude (10^3 to 10^{10}) in Rayleigh number covering laminar, transition and turbulent regime in 2D cases without any extra treatment. 3D comparisons for the turbulent case also show good agreement with some other LES and DNS results for both mean and fluctuation profiles, and our results match the experimental data best for the fluctuation profiles. Weak imposition of boundary condition for both velocity and temperature is also extended to buoyancy-driven flow, and it is able to substantially reduce the computational cost (36 times) while still produce reasonable results in a turbulent 2D case.

In addition to the flow fields, IMGA is deployed to moving objects in fluids in Chapter 3. The moving IMGA framework is verified by the convergence study of a freely dropping cylinder (2D) in a viscous fluid with low Reynolds number. We show a good agreement of the analytical terminal velocity of the cylinder as well in this case. Experimental validations of object trajectory and sedimental velocity are also carried out successfully by a freely dropping sphere with moderate Reynolds number. 2D simulations of particle focusing in an unobstructed channel and an obstructed channel decorated with pillar also reasonably predict the particle motion and resulting flow fields compared with some other numerical results. Comparison with a body-fitted method using Fluent indicates advantages of the moving IMGA regarding the computational cost and efficiency.

In Chapter 4, We further integrate the moving IMGA on the unstructured framework to the octree-based adaptive refinement framework. We showcase a good strong scalability for the overall solve time on the coupled framework using a dropping sphere case. The coupled framework also

improves the surface assembly. However, it indicates that the surface assembly still takes considerable time, which is caused by the imbalanced load in the traditional balancing process because the elements that cut the immersed boundary involve additional work in the surface computations. The overall solve time may be further substantially decreased by incorporating a dynamic load balancing that accounts for this issue of surface assembly.

5.2 Future work

With the framework developed in this work, we suggest to solve potential problems directly in following areas:

1. Naturally occurring coupled indoor (VMS for buoyancy-driven flow) and outdoor (static IMGA for buoyancy-driven flow) air ventilation and heat transfer in a housing community to design a more pleasant and healthy living environment.

2. Medical and biomechanical applications, such as designing obstructed micro-devices to control and manipulate particles in blood vessel, which can be used in embolism transport control, and cancer cell separation (moving IMGA).

3. A CFD tool to model and adjust Maxey–Riley equation [1] in varied scenarios of physics. Maxey–Riley equation is a one-way coupling method that assumes the flow field is not disrupted by object motion. Assumptions and modeling have to be reasonably proposed for the variations of Maxey–Riley equation based on its original form according to the physics. This work (moving IMGA on coupled framework) would be a good tool to fast supply accurate CFD data for the modeling of Maxey-Riley equation.

4. Particle motion in non-isothermal flows (VMS for buoyancy-driven flow and moving IMGA). This combines the two major topics in this work – buoyancy-driven flow and fluid-structure interaction with moving boundaries. Examples include seafloor rock movement caused by geothermal and rock transport and sedimentation in volcano eruption.

5.3 References

[1] M. R. Maxey and J. J. Riley. Equation of motion for a small rigid sphere in a nonuniform flow. *The Physics of Fluids*, 26(4):883–889, 1983.