NUMERICAL ADVANCES FOR FLUID-STRUCTURE INTERACTION IN ENTANGLED POLYMER SOLUTIONS WITH APPLICATIONS TO ACTIVE MICROBEAD RHEOLOGY

Fuhui Fang

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mathematics in the College of Arts and Sciences.

Chapel Hill
2020

ABSTRACT

Fuhui Fang: Numerical Advances for Fluid-Structure Interaction in Entangled Polymer Solutions with Applications to Active Microbead Rheology
(Under the direction of M. Gregory Forest and Boyce E. Griffith)

Active microbead rheology is an important counterpart to passive microbead rheology. Both techniques have proven essential for exploring the viscoelastic properties of soft materials that yield at extremely low stress and strain thresholds. Many soft materials, especially arising in biology, are furthermore only available in small volumes that are not amenable to classical rheometers. In passive microbead rheology, thermal fluctuations of microbeads reveal the linear, equilibrium viscous and elastic moduli of the material across the frequency range that can be resolved by the microscope. In active microbead rheology, viscoelastic fluids can be driven out of equilibrium by controlled forces applied to magnetic microbeads, and the materials then exhibit a range of responses: the so-called linear response regime, where responses are proportional to the magnetic force at sufficiently low levels, and then a transition to a variety of nonlinear responses that are unique to different types of viscoelastic fluids. Understanding this rheological phenomenon is important in the study of dynamics of many biological systems involving flexible structures, such as ciliary transport of mucus in the human lung. Despite ongoing developments in modeling such systems, there is still a lack of accurate and efficient numerical methods and software packages that can describe such nonlinear phenomena quantitatively. Modeling viscoelastic fluids usually requires high spatial resolution and time-consuming simulations, and the interactions between fluids and flexible structures introduce additional numerical and computational challenges.

The main goal of this dissertation is to develop and analyze robust numerical methods for viscoelastic fluid-structure interaction (FSI) with applications to active microbead rheology, and in particular, the transition of a linear to nonlinear response exhibited by a specific class of viscoelastic fluids, entangled polymer solutions. We employ the immersed boundary (IB) method to model fluid-structure interaction and use the open-source software IBAMR to implement the simulations.

The simulations are guided and validated by experimental data. Motivated by numerical issues we encountered in the microbead simulations, we propose and implement a novel implicit solver for the constrained IB formulation provided by IBAMR, and we investigate its accuracy and efficiency with extensive numerical tests. Lastly, we develop adaptive mesh refinement (AMR) capabilities for solving the Stokes problem to enable more efficient simulations of high-resolution FSI problems at low Reynolds numbers.

To the loving memory of my grandmother, Fengxin Wang.

ACKNOWLEDGEMENTS

There are no proper words to convey my deepest appreciation to my thesis advisors, Professor Greg Forest and Professor Boyce Griffith, for their relentless guidance, support, and patience throughout my Ph.D. study and research. I am especially grateful for the freedom that they have allowed in my research, as well as their unique perspective on computational science and mathematical modeling, which will no doubt shape my own philosophy as I move forward.

I would like to thank the members of my dissertation committee: Professor Jingfang Huang, Professor Ehssan Nazockdast, and Professor Katherine Newhall for their encouragement and insightful comments to improve my work. I am extremely grateful to Professor Jingfang Huang for his guidance and support in my early career.

I am deeply indebted to my collaborators, fellow students and postdoctoral scholars for lending me their expertise and intuition to my scientific and technical problems. Special thanks go to Dr. David Wells, Dr. Amneet Bhalla, Dr. Ebrahim (Amin) Kolahdouz, and Dr. Aaron Barrett.

I also thank my friends Dangxing Chen, Yanni Lai, Yuan Gao, Jason Pearson, Yunyan He, and Xie He for the great moments we had here at UNC; Meng Chen and Qi Jiang for their encouragement and caring throughout the years despite the long distance between us.

Last but no least, I sincerely thank my parents Shuhong Wu and Ruyun Fang for their love and patience. They selflessly encouraged me to explore new directions in life and seek my own destiny. This journey would not have been possible without their encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
Introduction

1.1 Background and Motivation

Newtonian fluid models that assume that the fluid viscosity only depends on temperature and pressure are the simplest mathematical descriptions for the behavior of fluids. Typical Newtonian fluids include water, gasoline, and alcohol. However, non-Newtonian fluids are also ubiquitous in everyday life and biological applications. Many solutions of polymers, emulsions, and suspensions are non-Newtonian fluids, as are many commonly found substances such as blood and paint [2]. In contrast to Newtonian fluids, the viscosity of non-Newtonian fluids depends on the velocity gradient, e.g., shear rate in shear-dominated flows or elongation rate in extensional flows, or on the deformation history of the fluid [3]. Understanding the complex rheological behaviors exhibited by complex fluids is essential to the study of biological fluids and systems, and brings mathematical, modeling, and computational challenges [4], as the constitutive equations for the extra stress introduce additional nonlinear terms when coupled to the Navier-Stokes equations. In reality, biological fluids often are surrounded by and interact with dynamic flexible microstructures. For example, red blood cells are the predominant blood cells with flexibility attributable to the cell membrane. In human lungs, inhaled viruses, bacteria, and other particles can be trapped in the mucus layer and cleared by ciliary transport that propels the mucus layer toward the larynx to be swallowed [5, 6]. Mathematically, we could model the complex biological flows such as blood and mucus as viscoelastic fluids, and the micrometer-scale particles such as cells, bacteria, and cilia can be described by some flexible structures. Such biological problems are in essence the study of viscoelastic fluid-structure interaction (FSI).

Microbead rheology is the study of the rheological behavior of the fluid deformed by the micron-sized spheres immersed in it [7]. In contrast to passive microbead rheology, where the bead undergoes stochastic, and typically sub-diffusive, motion due to thermal fluctuations, in active microbead (AM) rheology, the motion of the microbeads is induced by external forces, which is usually created using an optical trap or an magnetic field [8]. Previous experiments and studies [8, 9] observe that driven magnetic microbeads immersed in entangled polymer solutions (EPS) transition from a linear Stokes regime to exhibit a nonlinear response above a certain threshold in the applied magnetic force, and the surrounding fluid is driven out of equilibrium. Numerically modeling the bead dynamics and this stress-induced nonlinear behavior is important in understanding the mechanics of biological systems at microscale and nanoscale, as flow transport of, and locomotion within, biological fluids by active structures (cells, bacteria, cilia) are governed by active microbead rheology.

Despite extensive ongoing work in the past decades to understand the dynamics of active structures in complex fluids, there remains a lack of accurate and efficient numerical methods and software packages for modeling such systems. Modeling complex fluids usually requires high spatial resolution and time-consuming simulations, which introduces additional numerical and computational difficulties. The IBAMR software provides a powerful modeling technology for fluid-structure interaction [10], and it is the main tool we use in this thesis. However, functionalities related to complex fluids are relatively new and still under development. Moreover, to date, there is no validated computational model of the fully coupled system capable of quantitatively capturing the nonlinear phenomenon in active microbead rheology. Thus, we are motivated to take this opportunity and develop accurate numerical methods for FSI problems, guided and validated by applications and experimental data from biology and engineering. In particular, we aim to develop robust numerical methods that can accurately resolve the flow as well as the viscous and polymeric contribution to the stress tensor, and capture the nonlinear responses in complex fluids induced by driven magnetic microbeads. The numerical advances will also be incorporated into IBAMR.

1.2 Outline of the Dissertation

The remainder of this dissertation is organized as follows.

In Chapter 2, we discuss the nonlinear response of entangled polymer solutions in active magnetic microbead rheology in detail. We begin with a complete description of the mathematical models of fluid mechanics and relevant derivations. In particular, we discuss the Rolie-Poly constitutive equations for viscoelastic fluids and the immersed boundary (IB) method for fluid-structure interaction, with verification tests for the numerical implementation. Next, we present the results for the active microbead simulations, and compare with the benchmark experimental data and computational results.

In Chapter 3, we discuss the preconditioning techniques for the constrained formulation of the IB method. Motivated by the numerical instability issues we encountered in the active microbead simulations, we introduce a new preconditioner based on the least squares commutator (LSC), and present results from numerical experiments.

Finally, in Chapter 4, we discuss the adaptive mesh refinement (AMR) strategy for the Stokes problem, which is currently unavailable in IBAMR. We introduce a scheme based on the lowest-order Raviart-Thomas space (RT_0), which we anticipate may potentially accelerate large-scale high-resolution viscoelastic FSI simulations. This is followed by validation against analytic solutions and error analysis with extensive numerical experiments.

CHAPTER 2

Active Microbead Rheology

As discussed in Chapter 1, entangled polymer solutions, such as λ-DNA solutions, can be driven substantially out of equilibrium by active magnetic beads and can exhibit a superlinear response characterized by a terminal velocity that scales nonlinearly with the applied force magnitude. Aspects of such systems have been studied previously using one-way coupled mathematical models that decouple the flow from the polymeric stress, but we are interested in numerical modeling of this nonlinear phenomenon quantitatively using a fully-coupled hydrodynamic approach. Biological fluids are complicated in nature, and interpreting rheological properties theoretically remains a hard question. It is also a challenging mathematical modeling problem, with additional difficulties brought by fluid-structure interaction. The generalized Stokes-Einstein relation (GSER) is the fundamental theory for interpreting many rheological properties of Newtonian-fluids in modern microrheology, and it could be generalized to non-Newtonian fluids [9]. However, in the regime of active microrheology, some key assumptions of the GSER could fail, which breaks the agreement between microscopic and macroscopic measurements. Consequently, there is a great need for novel microrheological techniques capable of capturing and explaining the nonlinear rheological properties, and this is a relatively recent area of study.

2.1 Mathematical Models of Fluid Mechanics

In this section, we introduce basic concepts and common constitutive equations for fluid dynamics and numerical models for fluid-structure interaction, from both macroscopic and microscopic perspectives. We focus on the continuum approach where the fluid is assumed to be a continuous material. We omit some detailed derivations, and we refer the interested reader to discussions in prior work [4, 8, 11, 12, 13].

2.1.1 Eulerian and Lagrangian Descriptions

We use Eulerian variables to describe the fluid motion and Lagrangian variables for the structure immersed in the fluid, as shown in Fig. 2.1. In the Eulerian frame, the fluid variables are defined on a fixed Cartesian coordinate system. For a fixed physical domain Ω, let $\boldsymbol{x} = (x, y, z)$ be the physical coordinates of a fixed spatial position, then the fluid velocity at position $\boldsymbol{x}$ at time t is $\boldsymbol{u}(\boldsymbol{x}, t) = (u(\boldsymbol{x}, t), v(\boldsymbol{x}, t), w(\boldsymbol{x}, t))$ and the pressure is $p(\boldsymbol{x}, t)$. On the other hand, we use the Lagrangian frame to describe the immersed structure in terms of material coordinates that move with the structure. Let $\boldsymbol{s} = (q, r, s) \in U$ label a material point, then $\boldsymbol{\chi}(\boldsymbol{s}, t) \in \Omega$ is the physical position of this point at time t.

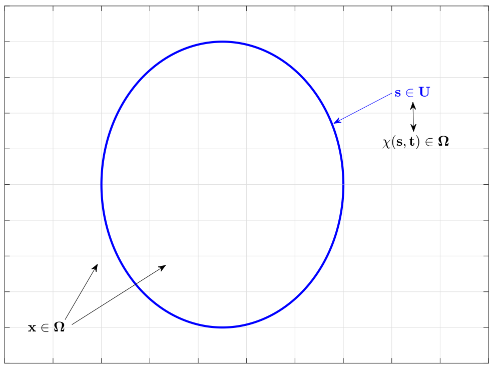

$\mathbf{s} \in \mathbf{U}$
$\chi(\mathbf{s}, \mathbf{t}) \in \mathbf{\Omega}$
$\mathbf{x} \in \mathbf{\Omega}$

Figure 2.1: Illustration of Eulerian and Lagrangian descriptions: on a Cartesian domain Ω with coordinates $\boldsymbol{x} = (x, y, z)$, $\boldsymbol{\chi}(\boldsymbol{s}, t)$ is the physical position of a Lagrangian point $\boldsymbol{s} = (q, r, s)$ on the immersed structure (blue ellipse) with domain U.

2.1.2 Conservation of Mass

The continuity equation reflects conservation of mass, which states a body of continuous matter has a constant mass m in a fixed volume V. Given the fluid density ρ, the mass can be written by a volume integral $m = \int_V \rho \, d\boldsymbol{x}$. Then we can develop the continuity equation by setting the net influx of mass in the control volume equal to the rate of change of mass. Given the fluid velocity $\boldsymbol{u}$ and pressure p, the equation can be written as

$$\frac{d}{dt} \int_V \rho \, d\boldsymbol{x} = - \int_{\partial V} (\rho \boldsymbol{u}) \cdot \boldsymbol{n} \, dA, \tag{2.1}$$

where $\boldsymbol{n}$ is the unit outward-pointing normal vector to the surface. We can convert this equation to a volume integral using the divergence theorem,

$$\int_V \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) \, d\boldsymbol{x} = 0. \tag{2.2}$$

In the regime of continuum mechanics, where the flow variables are assumed to be continuous in space and time (continuum assumption), we can derive the pointwise relation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \tag{2.3}$$

If the fluid is incompressible with a uniform mass density ρ_0, then Eq.(2.3) directly implies

$$\nabla \cdot \boldsymbol{u} = 0. \tag{2.4}$$

2.1.3 Conservation of Momentum

The principle of conservation of momentum requires that the total momentum of a closed system to be constant. The momentum is defined as the product of mass and velocity, and we can write the rate of change in momentum of a fluid particle with volume V as

$$\int_V \rho \frac{D\boldsymbol{u}}{Dt}\, d\boldsymbol{x}, \tag{2.5}$$

where $\frac{D\boldsymbol{u}}{Dt} = \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}$ is the material derivative, a frame invariant derivative that measures the change per unit time. For a quantity $f = f(\boldsymbol{x}, t) = f(\boldsymbol{\chi}(\boldsymbol{s}, t), t)$ for a particle P of which the position at time t is $\boldsymbol{\chi}(\boldsymbol{s}, t)$, the material derivative of f is

$$\frac{Df}{Dt} = \frac{\partial f(\boldsymbol{x}, t)}{\partial t} + \frac{\partial f(\boldsymbol{x}, t)}{\partial x_i} \frac{\partial x_i(\chi, t)}{\partial t} = \partial_t f + f_{,i} \partial_t x_i. \tag{2.6}$$

By Newton's second law of motion, the quantity in Eq. (2.5) is equal to the net force acting on that volume. There are two types of forces: (1) external forces, such as gravity, buoyancy, and electromagnetic forces, can be expressed as a volumetric force density $\boldsymbol{f}$; (2) internal forces are surface forces that measure the pressure and viscous drag that fluid particles exert on each other, denoted as traction (or Cauchy stress vector) $\boldsymbol{t}$. Then the balance principle for linear momentum is expressed as

$$\int_V \rho \frac{D\boldsymbol{u}}{Dt}\, d\boldsymbol{x} = \int_V \boldsymbol{f}\, d\boldsymbol{x} + \int_{\partial V} \boldsymbol{t}\, dA. \tag{2.7}$$

2.1.4 Stress and Strain in a Newtonian Fluid

By Cauchy's stress theorem, the traction vector $\boldsymbol{t}$ on a surface is determined by the Cauchy stress tensor $\boldsymbol{\sigma}$ and the outward-pointing unit normal vector $\boldsymbol{n}$ to the surface through

$$\boldsymbol{t} = \boldsymbol{\sigma} \cdot \boldsymbol{n}, \tag{2.8}$$

and we can apply the divergence theorem to Eq. (2.7) to obtain

$$\begin{aligned} \int_V \rho \frac{D\boldsymbol{u}}{Dt}\, d\boldsymbol{x} &= \int_V \boldsymbol{f}\, d\boldsymbol{x} + \int_{\partial V} \boldsymbol{\sigma} \cdot \boldsymbol{n}\, dA & (2.9) \\ &= \int_V \boldsymbol{f} + \nabla \cdot \boldsymbol{\sigma}\, d\boldsymbol{x}. & (2.10) \end{aligned}$$

The equivalent pointwise relation is

$$\rho \frac{D\boldsymbol{u}}{Dt} = \boldsymbol{f} + \nabla \cdot \boldsymbol{\sigma}. \tag{2.11}$$

Note that the components of the Cauchy stress tensor σ_{ij} are called the coordinate stresses, where the first index i represents the direction of the stress and the second index j represents the normal vector component n_j. The diagonal elements σ_{ii} are normal stresses, or hydrostatic stresses, which are related to volume change. The pressure, defined as the normal force per unit area, can be written in terms of the Cauchy stress tensor as $p = \frac{1}{3}(\sigma_{11} + \sigma_{22} + \sigma_{33})$. The pressure force is therefore $-p\mathbb{I}\boldsymbol{n}$, where $\mathbb{I}$ is the identity tensor. The tangential elements of the stress tensor $\sigma_{ij}, i \neq j$, are shear stresses or deviatoric stresses $\boldsymbol{\tau}$ which are related to shape change, with $\sigma_{ij} = \sigma_{ji}$. Given $\boldsymbol{\tau} = \boldsymbol{\sigma} + p\mathbb{I}$, we can re-write the stress tensor using the hydrostatic and deviatoric stress components:

$$\boldsymbol{\sigma} = -p\mathbb{I} + \boldsymbol{\tau}. \tag{2.12}$$

Note that internal forces generated from local deformation must depend on velocity differences, and we could use the rate-of-strain tensor $\boldsymbol{D}$ to characterize the deformation based on the symmetric part of the velocity gradient. $\boldsymbol{D}$ describes the rate of stretching and shearing,

$$\boldsymbol{D} = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T). \tag{2.13}$$

As an analogue to Hooke's law for solids, the linear stress-deformation relation defines the deviatoric stress to be linear in the rate of strain,

$$\boldsymbol{\tau} = 2\mu \boldsymbol{D} + (\xi - \frac{2\mu}{3})\mathbb{I}\nabla \cdot \boldsymbol{u}, \tag{2.14}$$

where μ is the dynamic viscosity of the fluid, and ξ is the second viscosity. A fluid satisfying the constitutive relation in Eq. (2.14) is called a Newtonian fluid.

The Navier-Stokes equations are summarized as

$$\begin{gathered} \rho_t + \nabla \cdot (\rho \boldsymbol{u}) = 0, \\ \rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}\right) + \nabla p - \nabla \cdot \left[2\mu \boldsymbol{D}(\boldsymbol{u}) + \left(\xi - \frac{2\mu}{3}\right)\mathbb{I}\nabla \cdot \boldsymbol{u}\right] = \boldsymbol{f}. \end{gathered} \tag{2.15}$$

If the fluid is incompressible and the dynamic viscosity μ remains constant, then Eq. (2.15) can be reduced to

$$\begin{gathered} \overbrace{\rho\Big(\underbrace{\frac{\partial \boldsymbol{u}}{\partial t}}_{\text{Variation}} + \underbrace{\boldsymbol{u} \cdot \nabla \boldsymbol{u}}_{\text{Convection}} \Big)}^{\text{Inertia}} + \underbrace{\nabla p}_{\text{Internal source}} - \underbrace{\mu \Delta \boldsymbol{u}}_{\text{Diffusion}} = \underbrace{\boldsymbol{f}}_{\text{External source}} \quad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} = 0, \end{gathered} \tag{2.16}$$

and we can use the kinematic viscosity $\nu = \frac{\mu}{\rho}$ to characterize the fluid.

2.1.5 Reynolds Number

In order to characterize the flow and predict flow patterns, we often use dimensionless quantities to describe the flow. Given the characteristic length scale L (e.g., the radius of the immersed sphere, the width of the channel), velocity scale U (e.g., the mean velocity of the flow), and the fluid density ρ, we can non-dimensionalize the flow variables via: $\boldsymbol{x}^* = \boldsymbol{x}/L,\ \boldsymbol{u}^* = \boldsymbol{u}/U,\ t^* = t/T,\ p^* = p/(\rho U^2),\ \boldsymbol{f}^* = \boldsymbol{f}L/(\mu U^2)$. We choose the characteristic time scale to be $T = \frac{L}{U}$, so that the differential operators are obtained from the chain rule as

$$\frac{\partial}{\partial t^*} = \frac{L}{U}\frac{\partial}{\partial t}, \nabla^* = L\nabla \tag{2.17}$$

Then the incompressible Navier-Stokes equations Eq. (2.16) in the rescaled variables become

$$\frac{\partial \boldsymbol{u}^*}{\partial t^*} + \boldsymbol{u}^* \cdot \nabla^* \boldsymbol{u}^* = -\nabla^* p^* + \frac{\mu}{\rho L U} \nabla^{*2} \boldsymbol{u}^* + \boldsymbol{f}^*, \tag{2.18}$$

$$\nabla^* \cdot \boldsymbol{u}^* = 0. \tag{2.19}$$

The Reynolds number is $\mathrm{Re} = \frac{\rho L U}{\mu}$. Dropping the asterisks to simplify the notation, the dimensionless incompressible Navier-Stokes equations are

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u} + \boldsymbol{f}, \tag{2.20}$$

$$\nabla \cdot \boldsymbol{u} = 0. \tag{2.21}$$

The Reynolds number can be interpreted as the ratio of inertial forces to viscous forces. At low Reynolds numbers, the flow is dominated by viscous forces and is in the laminar regime where fluid particles move slowly following smooth paths in parallel layers. For large Reynolds numbers, the viscous forces are negligible, and the flow show turbulent behavior with large fluctuations in the flow velocity and pressure.

In this dissertation, we focus on modeling flows at very small Reynolds numbers ($\mathrm{Re} \sim 10^{-6}$). Flows in this regime are called Stokes flows or creeping flows, and the idealized model with zero Reynolds number is the Stokes equations

$$-\nabla p + \mu \nabla^2 \boldsymbol{u} + \boldsymbol{f} = 0, \tag{2.22}$$

$$\nabla \cdot \boldsymbol{u} = 0. \tag{2.23}$$

2.1.6 Microscopic Polymer Models

Macroscopic physical properties of polymer solutions, such as viscosity and elasticity, are properties that can be seen with the naked eye, and they arise from microscopic entanglement of polymer chains [14]. In this section, we briefly discuss the microscopic scale models for describing the polymers and understanding the properties of viscoelastic fluids. We refer the interested readers to [15]. Dynamics of the polymer chains are usually described by tube theory [16], where a polymer chain is confined in a virtual tube to account for the uncrossability of surrounding chains. One

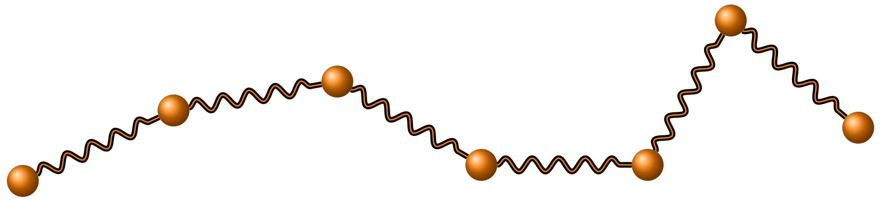


Figure 2.2: Illustration of the bead-spring polymer model with $N_e + 1$ beads connected by N_e springs in a polymer chain. When $N_e = 1$, it is the dumbbell model.

popular model to represent chainlike molecules is called the bead-spring model, where a polymer consists of $N_e + 1$ beads connected by N_e massless springs, as shown in Fig. 2.2. If the springs are Hookean springs, then the bead-spring chain is also called a Rouse chain. A major advantage of the bead-spring model is that it allows for orientability and stretchability. A simplified version of this model is the dumbbell model, where there are only two beads and one spring in a chain.

2.1.7 Non-Newtonian Fluids

A fluid that does not follow Eq. (2.13) is called a non-Newtonian fluid. We use the generalized Newtonian fluid constitutive equation to describe the fluid:

$$\boldsymbol{\sigma} = -p\mathbb{I} + 2\eta\boldsymbol{D}, \tag{2.24}$$

where the viscosity η is a function of the shear rate $\dot{\boldsymbol{\gamma}}$:

$$\eta = \eta(\dot{\boldsymbol{\gamma}}), \dot{\boldsymbol{\gamma}} = \sqrt{2(\boldsymbol{D} \colon \boldsymbol{D})} = \sqrt{2\sum_{i,j} D_{ij}D_{ji}}, \tag{2.25}$$

where : is the double dot product. In a simple shear flow, the shear rate is simplified as the absolute value of the rate of shear strain:

$$\dot{\gamma} = \left|\frac{\partial u}{\partial y}\right|. \tag{2.26}$$

In the case when $\eta \equiv \mu$ constant and shear-independent, Eq. (2.24) corresponds to the incompressible Newtonian fluid. When the viscosity decreases under shear strain, or $\partial\eta/\partial\dot{\gamma} < 0$, the fluid exhibits shear-thinning behavior with decreasing resistance to shear. Similarly, shear thickening occurs when $\partial\eta/\partial\dot{\gamma} > 0$.

Among different types of non-Newtonian fluids, we are mostly interested in viscoelastic fluids. As the name suggests, a viscoelastic fluid exhibits both elastic and viscous responses to deformations

— it may behave like an elastic solid under some conditions, but resemble a viscous liquid under other conditions. For a simple shear deformation characterized by its gradient $\boldsymbol{\gamma}$, the elastic response follows Hooke's law:

$$\boldsymbol{\sigma} = G\boldsymbol{\gamma}, \tag{2.27}$$

where G is the shear modulus that measures the elasticity of the material. The viscous response can be described in the same way as in the Newtonian fluid by the linear stress-deformation relation in Eqs. (2.13) and (2.14)

$$\boldsymbol{\sigma} = \eta\dot{\boldsymbol{\gamma}}. \tag{2.28}$$

One model that combines the viscoelastic elements is the linear Maxwell fluid model,

$$\boldsymbol{\sigma} + \lambda\frac{\partial \boldsymbol{\sigma}}{\partial t} = \eta\dot{\boldsymbol{\gamma}}, \tag{2.29}$$

where $\lambda = \frac{\eta}{G}$ is the Maxwell relaxation time. However, this equation is not frame-invariant, as $\frac{\partial \boldsymbol{\sigma}}{\partial t}$ is not frame-invariant. This can be shown by taking the time derivative of the stress tensor $\boldsymbol{\sigma}$ in a moving frame with a constant velocity $\boldsymbol{u}_0$,

$$\frac{\partial}{\partial t}\sigma_{ij}(\boldsymbol{x} + \boldsymbol{u}_0 t, t) = \frac{\partial \sigma_{ij}}{\partial t} + \boldsymbol{u}_0 \cdot \nabla\sigma_{ij}, \tag{2.30}$$

and this derivative is different from the time derivative in a stationary lab frame by a term $\boldsymbol{u}_0 \cdot \nabla\sigma_{ij}$. To fix this issue, one solution is to use the upper-convected or the lower-convected derivatives of a tensor, which are similar to the material derivative:

$$\text{Upper-convected derivative: } \overset{\triangledown}{\boldsymbol{\sigma}} = \frac{\partial \boldsymbol{\sigma}}{\partial t} + \boldsymbol{u} \cdot \nabla\boldsymbol{\sigma} - (\nabla\boldsymbol{u})^T \cdot \boldsymbol{\sigma} - \boldsymbol{\sigma} \cdot \nabla\boldsymbol{u}, \tag{2.31}$$

$$\text{Lower-convected derivative: } \overset{\triangle}{\boldsymbol{\sigma}} = \frac{\partial \boldsymbol{\sigma}}{\partial t} + \boldsymbol{u} \cdot \nabla\boldsymbol{\sigma} + \boldsymbol{\sigma} \cdot (\nabla\boldsymbol{u})^T + \nabla\boldsymbol{u} \cdot \boldsymbol{\sigma}. \tag{2.32}$$

As a generalization to the linear Maxwell model Eq. (2.29), two frame-invariant constitutive equations for the polymer stress $\boldsymbol{\tau}_p$ using the upper- and the lower-convected derivatives are

$$\text{Upper-convected Maxwell (UCM): } \boldsymbol{\tau}_p + \lambda \overset{\triangledown}{\boldsymbol{\tau}}_p = \eta_p \dot{\boldsymbol{\gamma}}, \tag{2.33}$$

$$\text{Lower-convected Maxwell (LCM): } \boldsymbol{\tau}_p + \lambda \overset{\vartriangle}{\boldsymbol{\tau}}_p = \eta_p \dot{\boldsymbol{\gamma}}, \tag{2.34}$$

where η_p is the polymer contribution to the total viscosity $\eta = \eta_s + \eta_p$. While the LCM model is seldomly used because of its inability to accurately describe the behavior of the continua, the UCM model is widely used and has extended to many more complete models of polymeric fluids.

Note that $\boldsymbol{\tau}_p$ represents the polymer contribution to the total stress, and often a Newtonian stress with a solvent viscosity η_s is added to the total stress so that the Cauchy stress for the viscoelastic fluid takes the form,

$$\boldsymbol{\sigma} = -p\mathbb{I} + \eta_s \dot{\boldsymbol{\gamma}} + \boldsymbol{\tau}_p, \tag{2.35}$$

where $\boldsymbol{\tau} = \eta_s \dot{\boldsymbol{\gamma}} + \boldsymbol{\tau}_p$ is called the total deviatoric stress. One popular constitutive model based on the UCM model is the Oldroyd-B moodel,

$$\boldsymbol{\tau} + \lambda \overset{\triangledown}{\boldsymbol{\tau}} = \eta(\dot{\boldsymbol{\gamma}} + \lambda_r \overset{\triangledown}{\dot{\boldsymbol{\gamma}}}), \tag{2.36}$$

where the retardation time $\lambda_r = \lambda(\eta_s/\eta)$ characterizes the retarded fluid motion. Note that as a general case of the Newontian and Maxwell fluid, it could be reduced to a Newtonian fluid when $\lambda = 0$ and $\lambda_r = 0$. When $\lambda_r = 0$, it reduces to a Maxwell fluid. In fact, the steady-state solution of the velocity field for an Oldroyd-B fluid is exactly the same as in the case of Newtonian fluids, and the non-Newtonian parameters only make an effect on the transient dynamics [17]. A drawback of the Oldroyd-B model is that allows a continuous stretching of polymers in the flow and an unbounded Hookean stress. An improved model is the Giesekus model, which is formed by adding a term proportional to $\boldsymbol{\tau}_p \cdot \boldsymbol{\tau}_p$ to the UCM model, so that the model is nonlinear in stresses,

$$\boldsymbol{\tau}_p + \lambda \overset{\triangledown}{\boldsymbol{\tau}}_p + \alpha \frac{\lambda}{\eta_p} \boldsymbol{\tau}_p \cdot \boldsymbol{\tau}_p = \eta_p \dot{\boldsymbol{\gamma}}, \tag{2.37}$$

where $\alpha < \frac{1}{2}$ is a dimensionless parameter. In our simulations, we mainly apply the Rolie-Poly model, which is a newer model. An advantage of the Rolie-Poly model is that it is based on a more comprehensive full chain model of polymer molecules including the processes of reptation, convective and reptation-driven constraint release, chain stretch, and contour length fluctuations [18, 19]. The Rolie-Poly model takes the form,

$$\boldsymbol{\tau}_p + \lambda \overset{\triangledown}{\boldsymbol{\tau}}_p = \eta_p \dot{\boldsymbol{\gamma}} - \frac{2}{3}\lambda(\boldsymbol{\tau}_p : \nabla \boldsymbol{u}) \left(\mathbb{I} + (1+\epsilon)\frac{\lambda}{\eta_p}\boldsymbol{\tau}_p \right), \tag{2.38}$$

where ϵ is a dimensionless parameter that controls the rate at which the polymeric viscosity η_p and the relaxation time λ change with the polymeric stress $\boldsymbol{\tau}_p$. In our numerical calculations, we use the time evolution of $\boldsymbol{\tau}_p$ [20],

$$\frac{d\boldsymbol{\tau}_p}{dt} = \underbrace{(\nabla \boldsymbol{u})^T \cdot \boldsymbol{\tau}_p + \boldsymbol{\tau}_p \cdot (\nabla \boldsymbol{u})}_{\text{Deformation due to flow}} - \underbrace{\frac{\boldsymbol{\tau}_p - \mathbb{I}}{\tau_d}}_{\text{Reptation}} - \underbrace{\frac{2}{\tau_R}\left(1 - \sqrt{\frac{3}{tr(\boldsymbol{\tau}_p)}}\right)\left[\boldsymbol{\tau}_p + \right.}_{\text{Chain retraction}} \underbrace{\left. \beta \left(\frac{tr(\boldsymbol{\tau}_p)}{3}\right)^{\delta} (\boldsymbol{\tau}_p - \mathbb{I}) \right]}_{\text{Convective constraint release}}, \tag{2.39}$$

where $\beta \in [0, 1]$ captures the effects of convective constraint release, and the exponent δ is a negative power determined by fitting to the full theory in [21]. Each chain has two characteristic relaxation times: τ_d, the linear relaxation time governed by reptation, measures the relaxation of orientation and corresponds to the relaxation time λ in the UCM, while τ_R is the tube Rouse time that measures relaxation of chain stretch. Both τ_d and τ_R depend on the Rouse relaxation time τ_e and the number of entanglement segments Z in a chain through

$$\tau_d = 3Z^3\tau_e, \tau_R = Z^2\tau_e, \tag{2.40}$$

with

$$\tau_e = \frac{N_e^2 \xi b^2}{3\pi^2 k_B T}, \tag{2.41}$$

where N_e is the number of monomers, and b is the segment length in an entanglement segment. ξ is the monomer friction coefficient (in units of mass/time) of the bead against the solvent. T is the temperature, and k_B is the Boltzman constant. Holding all other parameters constant, increasing τ_e by increasing b or N_e is equivalent to going from a chain composed by many springs and beads in the bead-spring model to just one dumbbell.

Eq. (2.39) is also called the single-mode Rolie-Poly equation. One may also extend this to the multi-mode equation, with each stress mode $\boldsymbol{\tau}_i$ satisfying Eq. (2.39) and contributing to the total polymer stress $\boldsymbol{\tau}_p$ in units of the shear modulus G_i:

$$\boldsymbol{\tau}_p = \sum_{i=1}^{n} G_i \boldsymbol{\tau}_i. \tag{2.42}$$

A multi-mode Rolie-Poly model can provide a more accurate description of a particular complex fluid because it describes different stretch of different parts in a polymer chain, but it requires additional parameter fitting. In this work, we adopt the single-mode model, which is usually an effective and sufficient alternative [18].

2.1.8 The Conformation Tensor

In our numerical calculations, we solve for the conformation tensor $\boldsymbol{C} = \langle \boldsymbol{RR} \rangle$ instead of the polymer stress tensor $\boldsymbol{\tau}_p$ because of its desirable linear algebra properties [22]. $\boldsymbol{R}$ is the end-to-end vector of a polymer chain in the dumbbell model, so this dyadic tensor is required to be positive-semidefinite to remain physical. The relationship between $\boldsymbol{C}$ and $\boldsymbol{\tau}_p$ is

$$\boldsymbol{\tau}_p = \frac{\eta_p}{\lambda}(\boldsymbol{C} - \mathbb{I}) = G(\boldsymbol{C} - \mathbb{I}). \tag{2.43}$$

Note that when the system is in equilibrium, $\boldsymbol{C} = \mathbb{I}$, which indicates no deformation occurs. The evolution of the conformation tensor can be written as [23]

$$\frac{D\boldsymbol{C}}{Dt} = (\nabla \boldsymbol{u})^T \cdot \boldsymbol{C} + \boldsymbol{C} \cdot \nabla \boldsymbol{u} + \boldsymbol{g}(\boldsymbol{C}), \tag{2.44}$$

where $\frac{D\boldsymbol{C}}{Dt} = \frac{\partial \boldsymbol{C}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{C}$ denotes the material derivative of $\boldsymbol{C}$, and $\boldsymbol{g}(\boldsymbol{C})$ is defined uniquely for each constitutive equation as follows

$$\boldsymbol{g}(\boldsymbol{C}) = \begin{cases} -\frac{1}{\lambda}(\boldsymbol{C} - \mathbb{I}), \text{ Oldroyd-B} \\ -\frac{1}{\lambda}(\boldsymbol{C} - \mathbb{I} + \alpha(\boldsymbol{C} - \mathbb{I})^2), \text{ Giesekus} \\ -\frac{1}{\tau_d}(\boldsymbol{C} - \mathbb{I}) - \frac{2(1-\sqrt{3/tr\boldsymbol{C}})}{\tau_R}(\boldsymbol{C} + \beta(\frac{tr\boldsymbol{C}}{3})^{\delta}(\boldsymbol{C} - \mathbb{I}), \text{ Rolie-Poly} \end{cases} . \tag{2.45}$$

2.1.9 Immersed Boundary Method

In previous sections, we discuss the governing equations for fluid dynamics, including the incompressible Navier-Stokes equations and viscoelastic constitutive models. For fluid-structure interaction, we use the well-known immersed boundary (IB) framework [24]. The IB method uses an Eulerian description of the fluid and a Lagrangian description of the immersed structure. The immersed structure is described by boundary points (IB markers), and we can use a "hollow shell" or a volumetric mesh representation. The Eulerian and Lagrangian variables are coupled via interaction equations involving the Dirac delta function. For a viscoelastic fluid, the governing equations for the IB approach include

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t}(\boldsymbol{x},t)+\boldsymbol{u}(\boldsymbol{x},t)\cdot\nabla\boldsymbol{u}(\boldsymbol{x},t)\right)=\nabla\cdot\boldsymbol{\sigma}+\boldsymbol{f}(\boldsymbol{x},t), \tag{2.46}$$

$$\nabla\cdot\boldsymbol{u}(\boldsymbol{x},t)=0, \tag{2.47}$$

$$\boldsymbol{f}(\boldsymbol{x},t)=\int_U \boldsymbol{F}(\boldsymbol{s},t)\delta(\boldsymbol{x}-\boldsymbol{\chi}(\boldsymbol{s},t))d\boldsymbol{s}, \tag{2.48}$$

$$\frac{\partial \boldsymbol{\chi}}{\partial t}(\boldsymbol{s},t)=\int_\Omega \boldsymbol{u}(\boldsymbol{x},t)\delta(\boldsymbol{x}-\boldsymbol{\chi}(\boldsymbol{s},t))d\boldsymbol{x}, \tag{2.49}$$

$$\boldsymbol{\sigma}=-p\mathbb{I}+\eta_s\left(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T\right)+\boldsymbol{\tau}_p, \tag{2.50}$$

where Eqs. (2.46) and (2.47) are the incompressible Navier-Stokes equations, which reduce to the standard Navier-Stokes equation in Eq. (2.16) if there is no polymer contribution $\boldsymbol{\tau}_p$ to the stress. $\boldsymbol{f}(\boldsymbol{x},t)$ is the volumetric force density applied by the structure to the fluid, and $\boldsymbol{F}(\boldsymbol{s},t)$ is the Lagrangian force density acting on the structure. Eq. (2.48) spreads the Lagrangian force to nearby fluid particles. Eq. (2.49) interpolates the fluid velocity to the structure and ensures that the immersed structure moves at the velocity of surrounding fluid particles. The Lagrangian force $\boldsymbol{F}$ can be determined to impose, either exactly or approximately, rigidity constraints.

In the two interaction equations Eq. (2.48) and (2.49), $\delta(\boldsymbol{x})$ denotes the three-dimensional delta function. In the numerical scheme, we use regularized delta functions $\delta_h(\boldsymbol{x})$ that are nonsingular for each h but approaches $\delta(\boldsymbol{x})$ as $h \to 0$, with the properties

$$\sum_{\boldsymbol{x}\in\Omega} \delta_h(\boldsymbol{x}-\boldsymbol{\chi})h^3 = 1 \text{ for all } \boldsymbol{\chi}, \tag{2.51}$$

$$\sum_{\boldsymbol{x}\in\Omega} (\boldsymbol{x}-\boldsymbol{\chi})\delta_h(\boldsymbol{x}-\boldsymbol{\chi})h^3 = 0 \text{ for all } \boldsymbol{\chi}, \tag{2.52}$$

where h denotes the Cartesian grid spacing. The three-dimensional regularized delta function is defined as the tensor product of one-dimensional regularized delta kernel $\phi(r)$ with support r,

$$\delta_h(\boldsymbol{x}) = \frac{1}{h^3}\phi\left(\frac{x}{h}\right)\phi\left(\frac{y}{h}\right)\phi\left(\frac{z}{h}\right). \tag{2.53}$$

While there are many choices for $\phi(r)$, here we present the widely used standard four-point kernel [24],

$$\phi(r) = \begin{cases} \frac{1}{8}(3-2|r|+\sqrt{1+4|r|-4r^2}), \text{ if } 0 \le r < 1, \\ \frac{1}{8}(5-2|r|+\sqrt{-7+12|r|-4r^2}), \text{ if } 1 \le r < 2, \\ 0, \text{ if } 2 \le r. \end{cases} \tag{2.54}$$

2.1.10 Implementation

The IBAMR software package is an adaptive and distributed-memory parallel implementation of the immersed boundary framework based on C++ [10]. IBAMR is built on SAMRAI [25] for adaptive mesh refinement infrastructure, PETSc [26] for linear solvers, libMesh [27] for finite element library, and *hypre* [28] for high performance preconditioners featuring parallel multigrid methods. In addition to the original IB method, IBAMR also supports solvers for IB based methods such as the immersed boundary-finite element method (IBFE) [29] and the immersed interface method (IIM) [30]. In this dissertation, we assume the structures are rigid bodies and use the CIB method (an IB method which implements the motion of rigid bodies using the constraint formulation) [31] to model fluid-structure interaction. Details of this method will be discussed in Sec. 3. The solvers for the viscoelastic fluids are integrated with IBAMR.

2.2 Verification of the Rolie-Poly Model

Before we perform numerical simulations for the microbead dynamics, we first use verification examples in two spatial dimensions with comparisons to benchmark computational studies to

$\rho(\mathrm{g/mm}^3)$	$\eta_s(\mathrm{Pa}\cdot\mathrm{s})$	$\tau_d(\mathrm{s})$	$\tau_R(\mathrm{s})$	$G(\mathrm{Pa})$	$\eta_p(\mathrm{Pa}\cdot\mathrm{s})$	β	δ
1.0×10^{-3}	41.348	0.05623	0.1	72800	4093.544	0	-0.5

Table 2.1: Model parameters for simulating DOW1568 flow through a contraction-expansion slit geometry and a cross-slot geometry.

investigate the accuracy of our implemented Rolie-Poly model. While there are not many existing numerical tests for the Rolie-Poly model, we consider two representative flow configurations in two dimensions: a contraction-expansion slit and a cross-slot. The geometries are modeled as immersed rigid structures, and they should not move or be deformed. [1] presents experimental data and numerical simulations for a monodisperse polystyrene DOW1568, and [32] also provides numerical results for both tests. Note that the authors of [1] fit a multi-mode Rolie-Poly model to the experimental data, while in contrast, [32] uses the dominant mode in a single-mode model. We set up our experiments for a single-mode Rolie-Poly model with parameters from [32]. The Reynolds number for both tests is on the order of 10^{-6}.

2.2.1 Flow through a Contraction-Expansion Slit

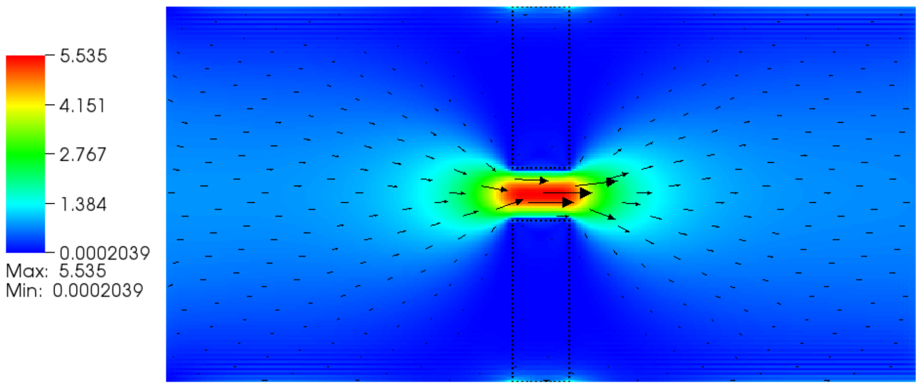
As shown in Fig. 2.3, we are interested in simulating flow through a pipe sudden contraction with a length of 20 mm and a width of $D = 10$ mm. The contraction is generated by two rectangular blocks adhering to the physical domain with a slit depth of 1.4 mm and a width of 1.5 mm. The blocks are modeled using one-dimensional Lagrangian elements, where the Lagrangian nodes are placed at a distance of twice the Cartesian grid spacing to avoid fluid leak. The contraction ratio is about 7.14 : 1. This flow geometry creates shear flows near the walls and extensional flows along the centerline from the inlet to the outlet [1].

We impose a parabolic normal velocity profile with maximum velocity 0.78 mm/s at the inlet (in the x-direction) and zero tangential velocity (in the y-direction),

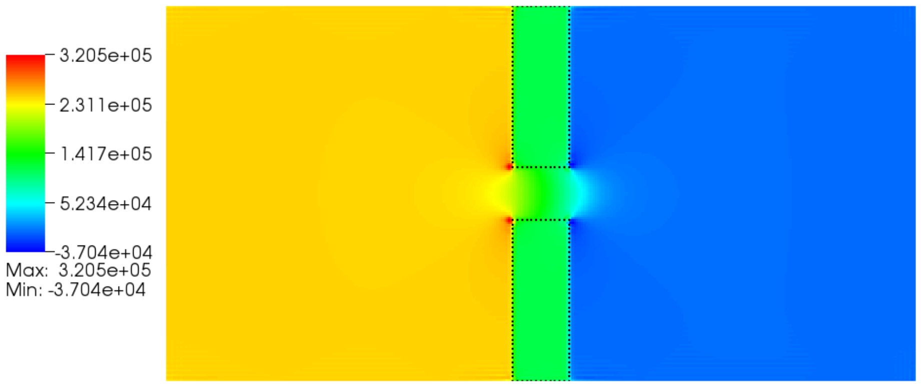
$$u = 0.78\left(1 - \frac{4(y-0.5D)^2}{D^2}\right), v = 0, \tag{2.55}$$

and we set homogeneous Neumann boundary conditions at the outlet. The conformation tensor is set to be the identity tensor at the inlet and the outlet. No-slip conditions are employed at the solid walls. The model parameters and corresponding units are summarized in Table. 2.1.

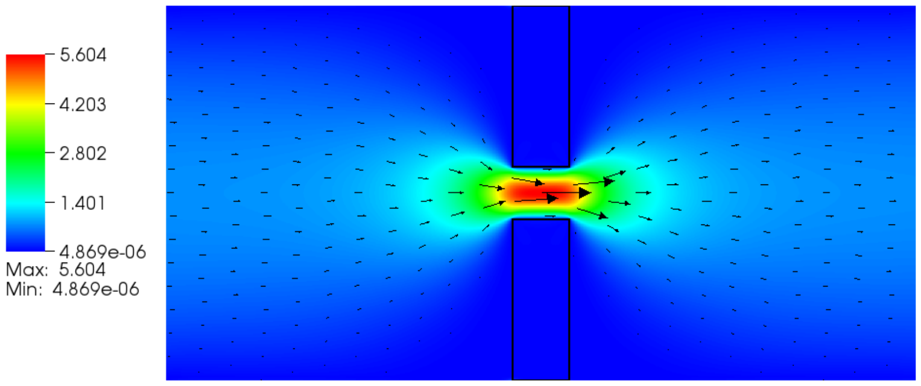
We measure the following four quantities along the centerline,



(a) Velocity magnitude and vector with Cartesian grid spacing $\Delta x = 0.0625$



(b) Pressure magnitude with Cartesian grid spacing $\Delta x = 0.0625$



(c) Velocity magnitude and vector with Cartesian grid spacing $\Delta x = 0.015625$

Figure 2.3: Two-dimensional flow through a contraction-expansion slit geometry with visualization of the steady-state velocity and pressure fields. The first two panels correspond to the coarsest mesh spacing, and the last panel corresponds to the finest mesh spacing.

1. Velocity in the x-direction,

2. First normal stress of the polymer stress τ_{xx},

3. First normal stress difference of the polymer stress $N_1 = \tau_{xx} - \tau_{yy}$,

4. Principal stress difference PSD $= \sqrt{(\tau_{xx} - \tau_{yy})^2 + 4\tau_{xy}^2}$. This is used to quantify the flow induced birefringe which has a linear relationship with PSD by the stress-optical law [33].

Note that to simplify the notations, we drop the subscript "p" in the polymer stress tensor $\boldsymbol{\tau}_p$. First, we conduct a grid convergence study to examine the spatial convergence of the simulations. With three different uniform mesh spacings $\Delta x = \{0.0625, 0.03125, 0.015625\}$ mm, we adjust the number of the CIB marker points to make sure the distance between two adjacent markers is twice the Cartesian grid spacing Δx, and perform the simulation with all other numerical parameters fixed. The results are presented in the left column of Fig. 2.4, with the x-axis showing the signed distance to the center of the geometry. As the grid is refined, the computed solutions are clearly converging. The panels in the right column show the comparison of our simulations (blue curves) with numerical results from [32] (red curves). In general, our numerical results are in good agreement with data published in the literature, as shown by the same data profile in all three panels. In particular, the velocity plot shows great consistency with the literature. We also visualize the steady-state velocity and pressure fields of the coarsest grid spacing (first two panels), along with the finest grid velocity field in the last panel of Fig. 2.3. In the first panel, we observe a very smooth velocity field. Our numerical method is capable of accurately capturing the large velocities near the slit without deforming the walls. Note that there is a small amount of fluid leak at the interface of the blocks and the physical boundary because the mesh is not fine enough, and the fluid leak is eliminated as we use finer meshes, as shown in the last panel in this figure. The second panel shows the pressure field with an overall pressure drop of $\Delta p = 3.2 \times 10^5$ Pa $= 3.2$ bar, which is consistent with the numerical results in [1]. In Fig. 2.5, we visualize the results for PSD. In addition to the two-dimensional simulation data, the PSD curves are also compared with three-dimensional simulated results (yellow curve) and experimental data (purple curve) from [1] in panel (b). In panel (c), the upper half shows the PSD contours from our simulation with contour intervals of 5 kPa, and the lower half is generated from the experimental data (figure reprinted with permission from [1]). Looking at panel (a), we notice that the first peak of our simulation almost overlaps with the three-dimensional simulation results. However, in general, both our method and the numerical method from the literature, tend to overpredict the stress near the slit. Possible causes for this discrepancy include: (1) our simulations are in two-dimensional while the experiments are in three-dimensional; (2) we use the single-mode Rolie-Poly model parameters proposed in [32], and the results should be more accurate if we use the full multi-mode model; (3) the experiment uses a geometry with rounded corners, while we use the sharp corners as in [32].

2.2.2 Flow through a Cross-Slot

The next example is the flow through a cross-slot geometry, which consists of four curved walls. Fig. 2.6 provides a schematic view of the cross-slot geometry with length $L = 10$ mm in an $L \times L$ computational domain. The curved walls are represented by black dots with radius 0.75 mm and the width of inlets/outlets is $D = 1.5$ mm. In this simulation, we use thick walls with two layers of marker points to ensure no fluid leak at the wall. It is possible to make the walls thicker by adding more layers towards the four corners while keeping the inner width of the channel fixed. However, preliminary simulations suggest that we do not benefit much from the additional layers.

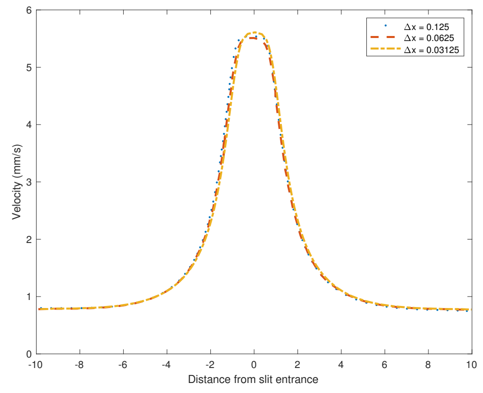
We define the center of the flow configuration as the stagnation point, and the line from the center of an inlet to the center of an outlet as the centerline. Parabolic flow profiles with maximum velocity magnitude $u = 3.135$ mm/s are imposed at the two inlets on the left and right boundary in opposite directions, with zero tangential velocities:

$$u_{\text{left}} = 3.135 \left(1 - \frac{4(y - 0.5L)^2}{D^2}\right), \tag{2.56}$$

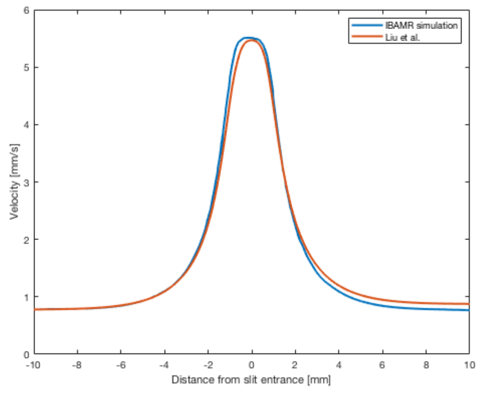
$$u_{\text{right}} = -3.135 \left(1 - \frac{4(y - 0.5L)^2}{D^2}\right), \tag{2.57}$$

$$v = 0. \tag{2.58}$$

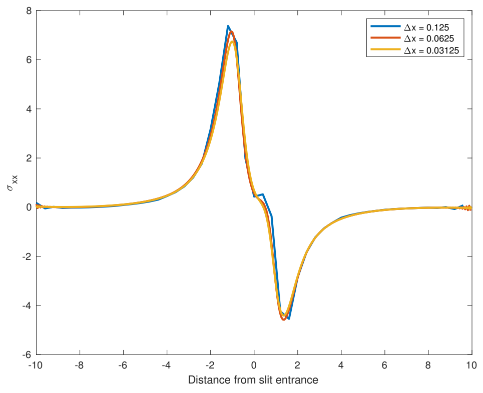
At the outlets, homogeneous traction-free boundary conditions are imposed in the y-direction and the tangential velocity is set to zero. No-slip conditions are imposed at (inner) solid walls. Neumann boundary conditions are imposed for the conformation tensor. The flow field is shown as the black arrows. The velocity magnitude is also plotted. With this cross-slot geometry, extensional flows are created in the central region near the stagnation point, and simple shear flows are near the



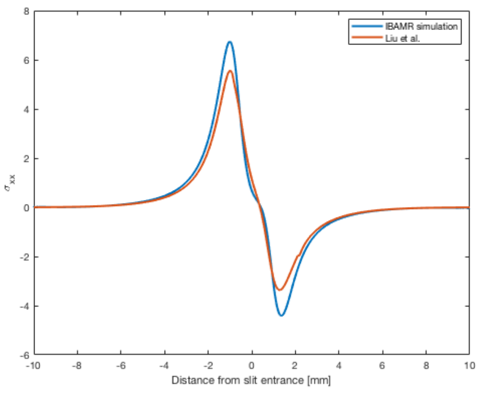
(a) Velocity convergence study



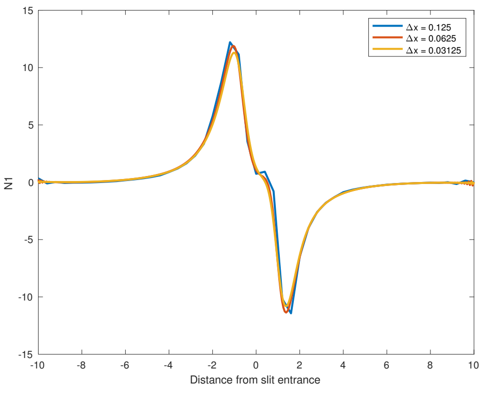
(d) Velocity along centerline



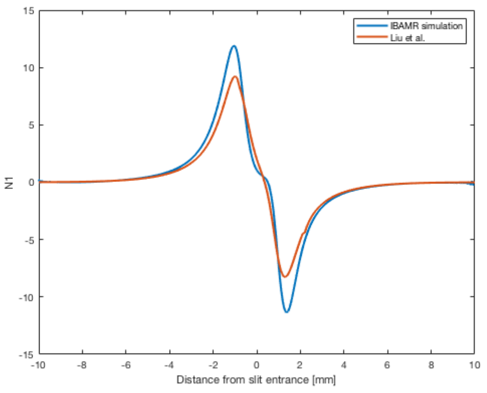
(b) τ_{xx} convergence study



(e) τ_{xx} along centerline

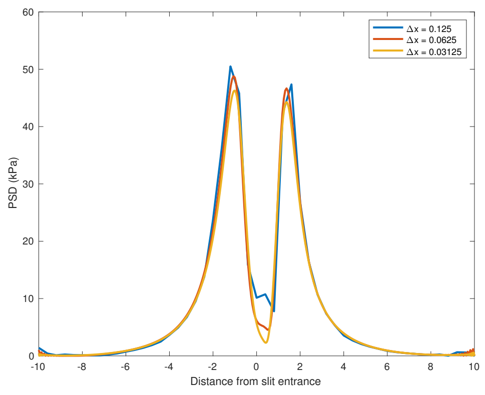


(c) N_1 convergence study

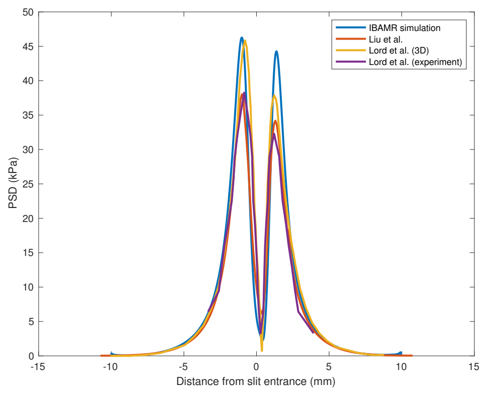


(f) N_1 along centerline

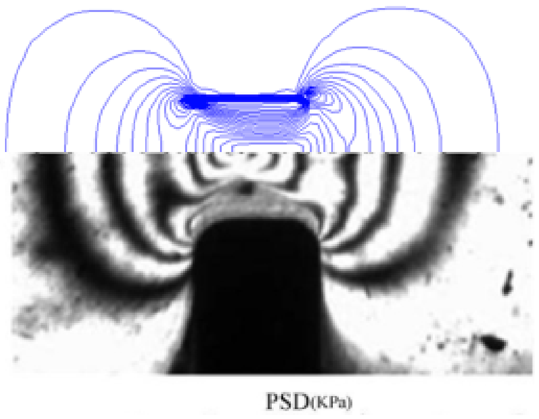
Figure 2.4: Results for the contraction-expansion slit geometry: velocity, first normal stress τ_{xx}, and first normal stress difference N_1 along the centerline. The grid spacing for the IBAMR simulation used in the right column is $\Delta x = 0.03125$.



(a) PSD convergence study



(b) PSD along centerline



(c) PSD contour plot

Figure 2.5: Principal stress difference (PSD) of the contraction-expansion slit geometry. The grid spacing for the IBAMR simulation used in panels (b) and (c) is $\Delta x = 0.03125$. Panel (c) compares slit flow PSD contours from our simulation (upper) and experimental results (lower, reprinted with permission from [1]). The contour interval for our simulation is 5 kPa.

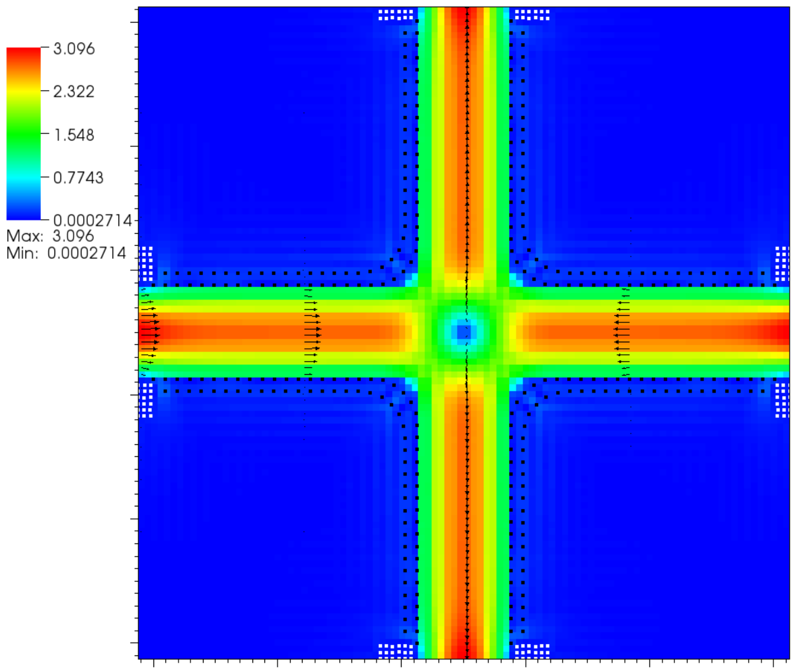


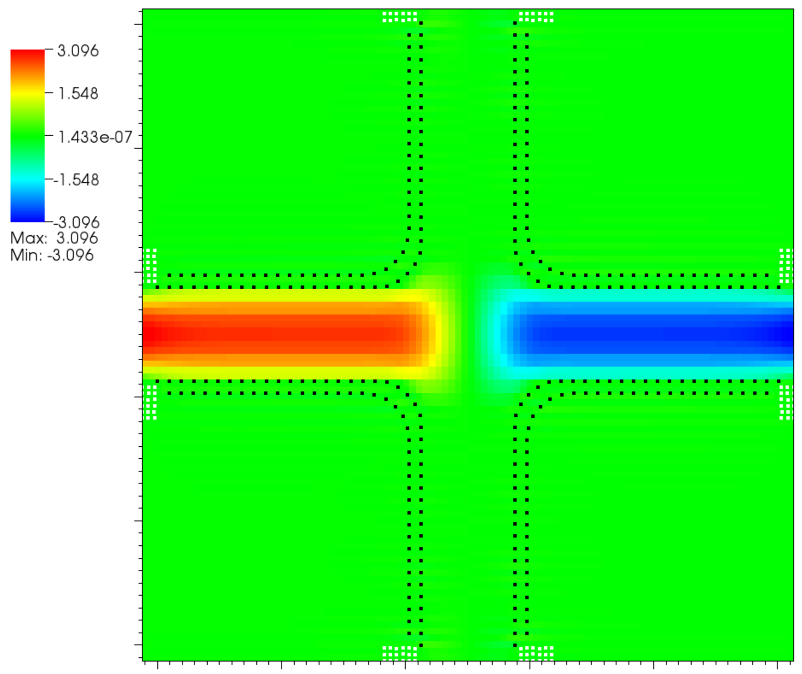
Figure 2.6: A schematic view of the cross-slot geometry with curved walls consisting of two layers of Lagrangian marker points (black dots). The Cartesian grid spacing is $\Delta x = 0.025$. With a parabolic velocity profile imposed at each inlet (left and right walls) in opposite directions, the steady-state flow field is indicated as black arrows with velocity magnitude shown in color. The computational domain and the geometry are extended by IB target points (white dots) for improved numerical stability.

walls. The velocity field is basically symmetric around the stagnation point. We use the same model parameters provided in Table. 2.1.

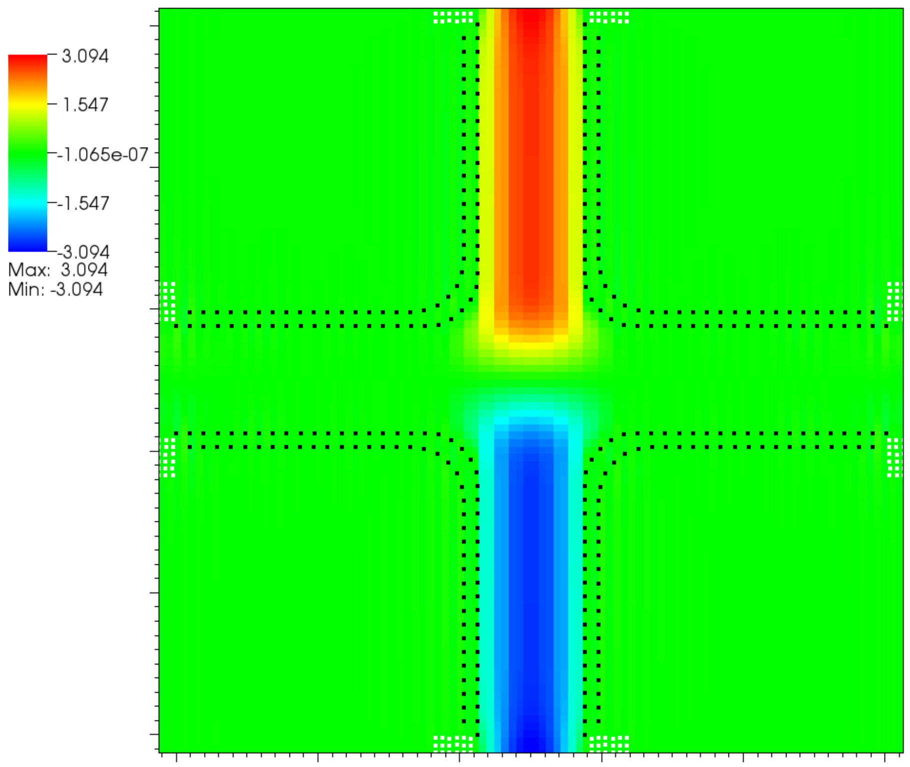
A key issue for this problem is that the current CIB solver does not converge if exactly constrained IB points extend to the outlet. We apply two techniques to fix this issue:

1. A combination of CIB and IB methods: we extend the physical domain by 2.5 grid spacings in each direction, and fill this small gap between the CIB walls and the new boundary with IB points. Those IB points, shown as white dots in Figs. 2.6 and 2.7, are tethered by very stiff linear springs (spring constant $\sim 5 \times 10^6$) to their initial positions in order to impose zero flow condition to stop the flow from leaking. Numerically, the extension part is solved using the original IB method instead of CIB.

2. Moving least squares (MLS): we replace the standard kernels for the regularized delta function with one-sided MLS reconstructions. With one-sided MLS, we spread and interpolate from



(a) Visualization of the x-component of the velocity field.



(b) Visualization of the y-component of the velocity field.

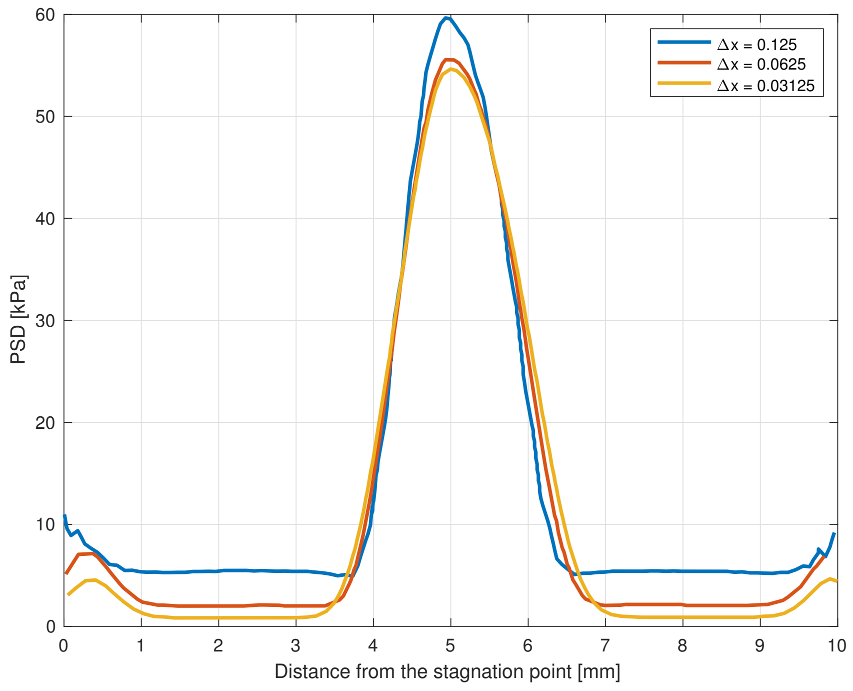
Figure 2.7: Visualization of the x- and y-components of the velocity field of flow through the cross-slot geometry with $\Delta x = 0.025$.

either outside or inside the geometry. Details of MLS and related verification tests are presented in the appendix.

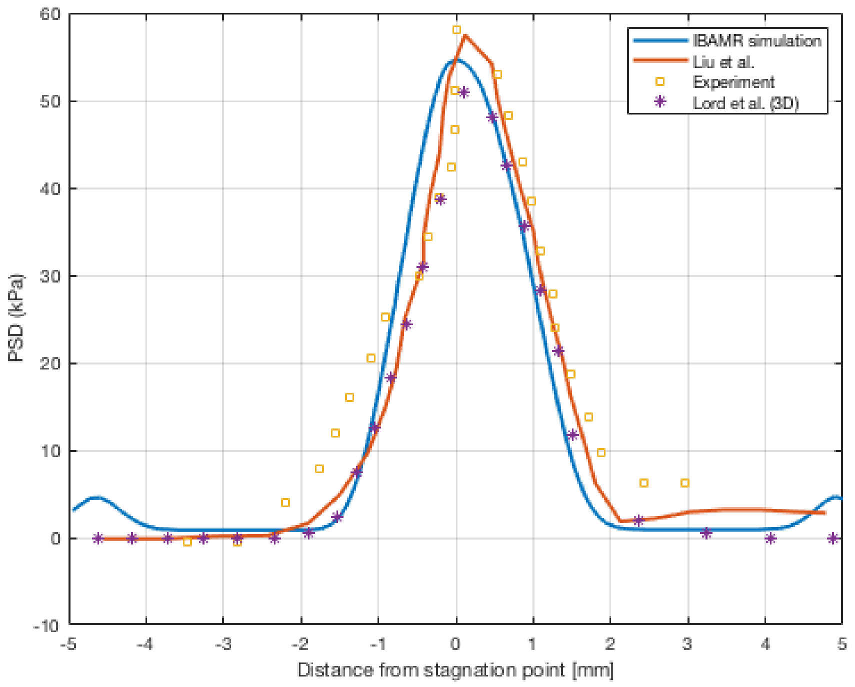
In Fig. 2.8, the first panel shows PSD along the centerline for simulations on three different uniform grid spacings. Note that we expect zero polymer stress near the inlet and the outlet. Despite this discrepancy, our computed solutions are converging. The second panel illustrates the comparison with experimental data (yellow squares), two-dimensional numerical simulations (red curve), and three-dimensional simulations (purple stars). Our simulated results are in good agreement with literature results. However, our method tends to under-predict the peak values of the stress near the stagnation point, in addition to the nonzero stress values at inlets and outlets. While the three-dimensional numerical test shows zero PSD at both inlet and outlet, the data for the experiment is missing, and the two-dimensional simulation tends to generate nonzero PSD at outlet as well. This suggests that the issue might be related to the spatial dimensions, and it might be fixed if we perform the simulation in three spatial dimensions. We are currently in the process of improving our numerical scheme in order to gain greater consistency.

2.3 Active Microbead Simulations

As we briefly discussed at the beginning of this chapter, a transition phenomenon from a linear to nonlinear response in entangled polymer solutions is shown by experiments and numerical models: with a constant external magnetic force F_M applied, the microbead initially experiences a high viscosity quasi-steady plateau in the linear Stokes regime; if the force is larger than a threshold, then the microbead starts accelerating with a decrease in apparent viscosity, and eventually it will converge to a terminal steady state with the equilibrium velocity larger than the Stokes response speed. This terminal velocity is found to scale nonlinearly with the force magnitude. We are interested in numerically modeling the transient dynamics and the quasi-steady behavior with IBAMR. Specifically, we consider a single magnetic bead immersed in a λ-DNA solution. The authors of [8] propose a simplified numerical scheme based on the Rolie-Poly model without considering hydrodynamic interactions; this study provides qualitative insights into the nonlinear response. However, to the best of our knowledge, no computational modeling for the fully coupled system has been developed. We use the UNC Linux-based computing cluster to perform all the simulations.



(a) Grid convergence study.



(b) Comparison with experimental data and numerical results from the literature.

Figure 2.8: PSD along the centerline for flow through the cross-slot geometry. The grid spacing used for the IBAMR simulation in the second panel is $\Delta x = 0.025$.

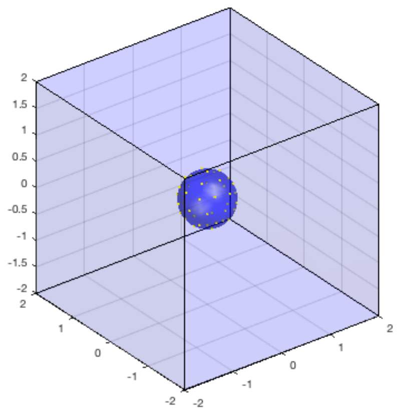


Figure 2.9: A schematic view of the microbead and the computational domain with periodic boundary conditions. A surface mesh with vertices shown as yellow dots is used to represent the microbead. The side length of the cubic domain is 8 times the microbead radius.

R (μm)	F_M (pN)	ρ (g/μm^3)	η_s (Pa·s)	τ_e (s)	Z	G (Pa)	β	δ
0.5	$1 \sim 12$	1.3×10^{-12}	0.005	0.00013	20	0.86	1	-0.5

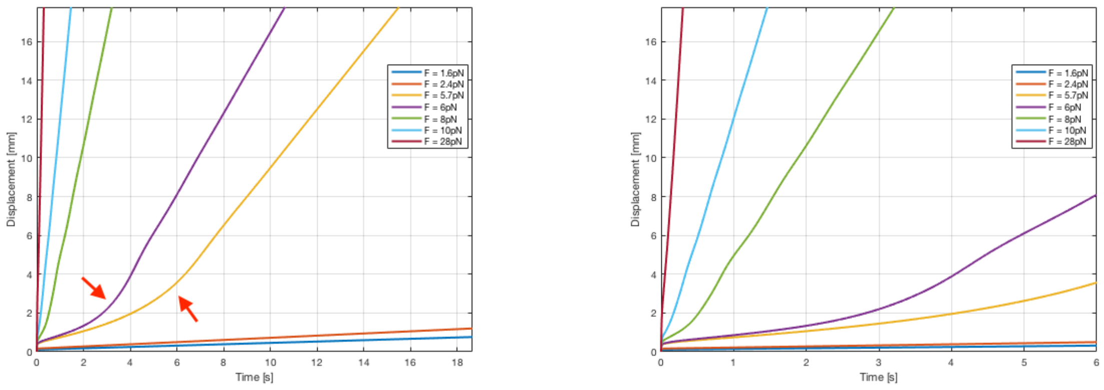
Table 2.2: Model parameters for simulating microbead rheology in a λ-DNA solution.

2.3.1 Problem Setup

As shown in Fig. 2.9, we use a triangular surface mesh to describe the microbead. The yellow dots represent the vertices of the triangular mesh, which correspond to Lagrangian markers. The current adaptive mesh refinement (AMR) implementation in IBAMR does not work for the Stokes problem, and due to the limits of computation, we need to restrict the computational domain to be a cube with a side length of 8 times the microbead radius and discretize the domain with a uniform mesh. However, our numerical scheme is able to model the immersed structure passing across the boundary, so we could use periodic boundary conditions to minimize the wall effects on the microbead dynamics and mimic an infinite domain. We are currently in the process of developing a feasible AMR implementation for Stokes and will discuss this further in Chapter 4. The parameters of the Rolie-Poly model are set to describe λ-DNA solution. Table. 2.2 summarizes the parameters we use for most of our simulations unless otherwise stated. The zero-shear rate viscosity of the polymeric fluid η_p is 2.68 Pa·s, approximately 536 times the solvent viscosity η_s.

2.3.2 Numerical Results

Effects of the force F_M First, we track the bead displacements over time with different external force magnitudes and visualize the results in Fig. 2.10. As discussed in [8], at early stages, the bead dynamics quickly converge to an equilibrium stage where the velocity U_0 follows Stokes law,



(a) Bead displacements over time (b) Zoom-in of early responses for $t < 6s$

Figure 2.10: Bead trajectories over time. The applied force magnitudes are $F_M = \{1.6, 2.4, 5.7, 6, 8, 10, 28\}$ pN. At early time points, the bead dynamics are in a Stokesian regime where the quasi-steady state velocity U_0 scales linearly with F_M. If F_M is below the force threshold F_T (the blue and red curves), this is the asymptotic steady state. If $F_M > F_T$ (the remaining five curves), the bead suddenly starts accelerating and eventually reaches a steady state with a terminal velocity U_∞ that is nonlinear in F_M.

$$F_D = 6\pi\eta_{\text{eff}} R U_0, \tag{2.59}$$

where F_D is the Stokes drag, and $F_D = F_M$ in magnitude at equilibrium. Note that in this simulation, we focus on the velocity in the normal direction, as the tangential velocities are zero. The polymer stress contributes to the effective viscosity η_{eff} through

$$\eta_{\text{eff}} = \eta_s + \frac{2}{3}\eta_p. \tag{2.60}$$

This suggests that, with the same model parameters, U_0 scales linearly with the magnetic force magnitude F_M, which is the canonical linear Stokes response. Prior experiments show that there exists a force threshold F_T such that:

- If $F_M < F_T$, U_0 is the asymptotic steady state. For example, the blue curve and the red curve in Fig. 2.10, which correspond to $F_M = 1.6$ pN and $F_M = 2.4$ pN respectively, have approximately constant slopes after the acceleration from the initial stationary state.

- If $F_M > F_T$, the system is driven out of equilibrium, whereby the bead suddenly starts accelerating and eventually converges to a new steady state. For example, the two red arrows point to the acceleration of bead dynamics when $F_M = 5.7$ pN (yellow curve) and $F_M = 6$ pN (purple curve) which illustrate the bead takeoff events. The velocity then quickly converges to U_∞, as shown by the constant slope of these two curves after the acceleration point. For larger forces, it is harder to identify the acceleration stage as the early stages are much shorter in time, but in panel (b), we focus on the short times and observe the change in the slope of the $F_M = 8$ pN (green) and $F_M = 10$ pN (blue) curves. The transient dynamics indicate the nonlinear response of the λ-DNA solution.

To further study the relationship of the terminal velocity U_∞ and the force F_M and to quantify the force threshold F_T, we perform additional experiments with different values of F_M and summarize the results in Table 2.3. If $F_M < 5.5$ pN, U_∞ is considered to scale linearly with F_M. Note that although the ratio of U_∞ to F_M has doubled for $F_M = 5$ pN compared to $F_M = 1.6$ pN, yet if we look at $F_M = 5$ pN and $F_M = 5.5$ pN, the ratio increases by a factor of 4.3 when F_M is increased by only 0.5 pN. Therefore F_T is found to be approximately 5.5 pN. This is also visualized in Fig. 2.11 where we plot U_∞ against F_M for F_M up to 10 pN. If $F_M < 5.5$ pN, U_∞ is very small, and those dots could be connected by a straight line approximately. Beyond 5.5 pN, U_∞ is much larger and the corresponding dots cannot be connected by a straight line, indicating the nonlinear relationship between U_∞ and F_M.

Note that our results are qualitatively consistent with the experiments in [8], while the experimental data shows a different threshold $F_T \sim 2$ pN. There are several potential explanations for the differences: (1) some parameters, such as the solvent viscosity η_s, are not explicitly specified for the experiments, and those parameters could be different from the ones we use in our numerical tests; (2) a periodic computational domain is similar to a large nonperiodic domain but not exactly the same.

In Fig. 2.12, we visualize the apparent viscosity η_{app} over time, where η_{app} is inferred from the

F_M (pN)	U_∞ (μm/s)	U_∞/F_M
1.6	0.0351	0.0219
2.4	0.0557	0.0232
3	0.0743	0.0247
3.5	0.0933	0.0267
4	0.1180	0.0295
4.5	0.1532	0.034
5	0.2229	0.0446
5.5	1.0631	0.1933
6	3.4039	0.5673
8	5.8173	0.7272
10	12.4644	1.2464
28	62.7501	2.2411

Table 2.3: Summary of the magnetic force F_M and the steady-state velocity U_∞. For $F_M < 5.5$ pN, U_∞ scales linearly with F_M. When F_M is large, U_∞ is not proportional to F_M, which indicates a superlinear response of the fluid material.

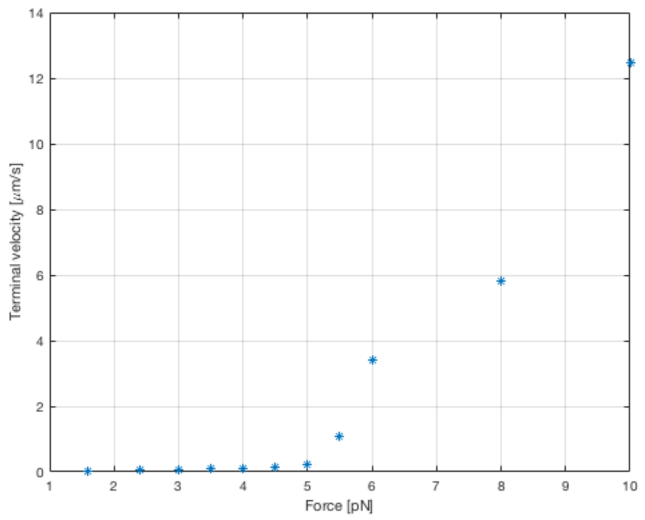


Figure 2.11: Sampling of terminal velocity U_∞ against the applied force F_M. There is approximately a linear relationship between U_∞ and F_M for small F_M. When $F_M \geq 5.5$ pN, bead takeoff phenomena occur and U_∞ does not scale linearly with F_M.

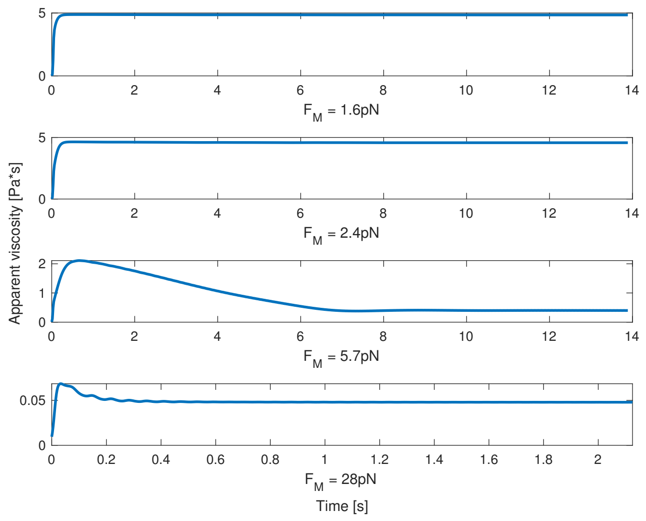


Figure 2.12: Apparent viscosity over time for different applied forces.

Stokes drag law,

$$\eta_{\text{app}} = \frac{F_M}{6\pi RU}. \tag{2.61}$$

Note that the apparent viscosity is constant and equal to η_s for Newtonian fluids. In the first two panels, with $F_M = 1.6$ pN and 2.4 pN, the apparent viscosity quickly transitions from 0 and converges to a constant value near 5 Pa·s. This suggests that the bead dynamics are in the linear Stokes regime with small force values applied. The third panel shows the apparent viscosity for $F_M = 5.7$ pN, where the nonlinear response is observed. Beyond the initial regime, the bead acceleration phase corresponds to the relatively slow drop in the apparent viscosity. When the bead dynamics converge to the terminal steady state, the apparent viscosity becomes constant. As $F_M = 5.7$ pN is very close to F_T, this is the "Goldilocks" phenomenon, where we observe the interesting transient behavior. The last panel shows results for $F_M = 28$ pN, which is much larger than the threshold F_T, so the initial Stokes regime and the acceleration phase are shorter than the $F_M = 5.7$ pN case, and the bead quickly moves through the viscosity overshoot phase and converges to the terminal steady state. Also, notice that the steady-state apparent viscosity decreases with larger forces, because the apparent viscosity for viscoelastic fluids depends on the shear rate. The changes in the apparent viscosity curve reflect a shear-thinning response.

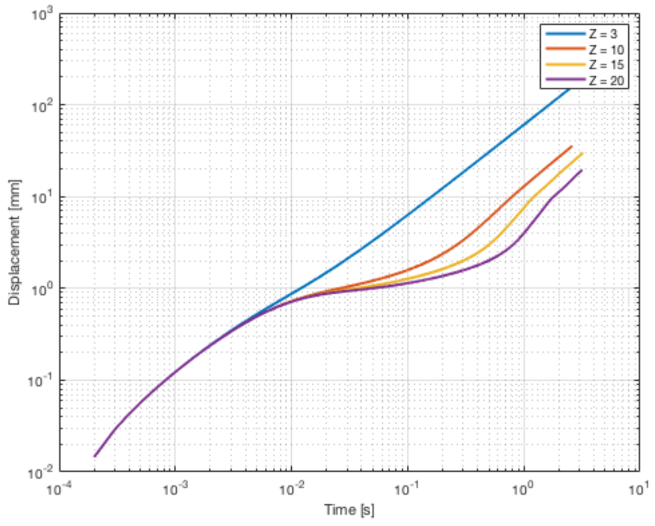


Figure 2.13: Log-log plot of displacements versus time as a function of the number of entanglements Z for $F_M = 6$ pN with other parameters fixed. With different values of Z, the trajectories overlap in short times. When $Z = 3$, no bead takeoff event is observed. As we increase Z, the bead dynamics exhibit a nonlinear response.

Effects of the number of entanglements Z Next, we study the effect of the number of entanglement segments in a chain Z on the bead dynamics. We perform experiments with different values of Z and apply a constant force of $F_M = 6$ pN. Other numerical parameters are fixed for each run. In Fig. 2.13, we visualize the bead displacements against time for each Z using logarithmic scales for both axes. Notably, all of the four curves overlap at short times, which indicates that the bead dynamics experience the same initial Stokes regime even with different Z. The $Z = 3$ curve (blue) is approximately linear, suggesting that the bead takeoff event is absent at smaller values of Z. At larger values of Z, we observe nonlinear dynamics beyond $t = 10$ ms. Looking at the terminal steady state for different values of Z, we find that the curves have approximately the same slope but different intercepts, which indicates that the terminal velocity also depends on Z.

Effects of the Rouse relaxation time τ_e We are also interested in the steady-state behavior of chain stretching as a function of the force F_M, the number of entanglements Z, and the Rouse relaxation time τ_e. Chain stretching is computed from the trace of the conformation tensor $tr(\boldsymbol{C})/3$, and it should be 1 for chain configurations close to equilibrium. We study chain stretching for a combination of parameters $F_M = \{1, 2, 5, 10, 30\}$ pN, $Z = \{3, 24\}$, and $\tau_e = \{1, 2, 3, 4\}$ ms. In

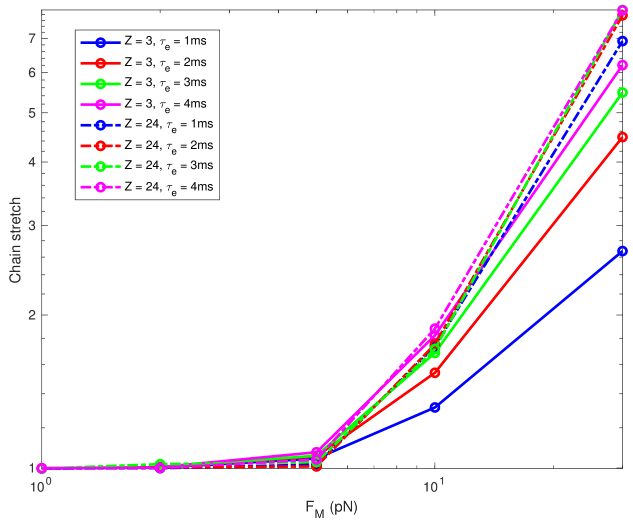


Figure 2.14: Loglog plot of chain stretch $tr(\boldsymbol{C})/3$ as a function of the force F_M, the number of entanglements Z, and the Rouse relaxation time τ_e.

Fig. 2.14, we visualize $tr(\boldsymbol{C})/3$ curves against the force F_M on logarithmic scales, and curves with the same color correspond to the same τ_e. Solid lines represent $Z = 3$ while dashed lines represent $Z = 24$. All curves are monotonically increasing, which indicates that chain stretching increases with increasing force. Looking at pairs in the same color, we observe that with a constant force, increasing Z results in an increase in chain stretching dynamics. With Z fixed, chain stretching also increases with τ_e, and therefore increasing τ_e could trigger a greater nonlinear response. Moreover, we notice that when $Z = 3$, chain stretching increases more significantly with τ_e compared to the increase at $Z = 24$. For example, the gap between the blue solid line and the red solid line is much larger than the gap between the blue and the red dashed lines. The red, green, and pink dashed lines are almost overlapping for $Z = 24$, which indicates small effects of τ_e on chain stretching behavior. As explained in [8], at smaller values of Z, chains cannot relax into their equilibrium configuration within time scales compared to those of the applied deformation, so the relaxation of individual entanglement segments dominates the relaxation dynamics. On the other hand, at large values of Z, the entangled network of chains dominates the relaxation dynamics.

2.3.3 One-way Coupling

The authors of [8] proposed a numerical model using the Rolie-Poly constitutive equation to simulate active microbead dynamics with a flow decoupling approximation. Specifically, hydrodynamic interactions are not considered in this scheme. Numerical results show that this method is effective in qualitatively explaining the nonlinear phenomenon. Here, we modify our numerical scheme to compute solutions with this one-way coupling method and compare with the results in [8]. We assume that the bead translational velocity generates an instantaneous quasi-steady Stokes velocity field, and the spherical velocity components can be written in a closed-form as

$$\begin{bmatrix} u_r \\ u_\theta \\ u_\phi \end{bmatrix} = \begin{bmatrix} \frac{U(t)}{2}\frac{R}{r}\cos(\theta)(3-(\frac{R}{r})^2) \\ -\frac{U(t)}{2}\frac{R}{r}\frac{\sin(\theta)}{2}(3+(\frac{R}{r})^2) \\ 0 \end{bmatrix}, \tag{2.62}$$

where $U(t)$ is the x-component of the bead translational velocity at time t, and the coordinates (r, θ, ϕ) are defined as

$$r = \sqrt{x^2+y^2+z^2},\ \theta = \arccos\left(\frac{x}{r}\right),\ \phi = \arccos\left(\frac{y}{\sqrt{y^2+z^2}}\right).$$

Note that we assume the flow to be axisymmetric, so $u_\phi = 0$. The corresponding velocity field in Cartesian coordinates can be expressed as

$$\boldsymbol{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \cos(\phi)\sin(\theta) & \cos(\phi)\cos(\theta) \\ \sin(\phi)\sin(\theta) & \sin(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} u_r \\ u_\theta \end{bmatrix}. \tag{2.63}$$

At each time step, instead of solving the Stokes equation, we analytically prescribe the velocity field and impose Dirichlet boundary conditions using Eq. (2.63). As a result, this is equivalent to performing the simulations on an infinite domain, and therefore we do not need to worry about wall effects and boundary conditions. Currently, we use the explicit time-stepping scheme for rigid body dynamics and update the bead translational velocity from time $t^k = k\Delta t$ to $t^{k+1} = (k+1)\Delta t$ following the steps below:

1. Prescribe the Stokes velocity field $\boldsymbol{u}^k$ using Eq. (2.63) given the x-component of the bead velocity U^k.

2. Compute drag force on the bead $F_D^k = 6\pi R\eta_s U^n + \int_{\partial V} \sum_{i=1}^{3} \tau_{i1}^k \, dA$, where the first component

F_M(pN)	U_{IBAMR} (μm/s)	$U_{\text{benchmark}}$ (μm/s)	Relative error
2	41.65	42.31	1.55%
6	125.85	124.86	0.79%
10	211.13	211.49	0.17%

Table 2.4: Comparisons of transient velocities with the code in literature for the reduced-order model without hydrodynamic interactions when the bead is subject to a magnetic force F_M.

is the Stokes drag, and the second component is the polymeric contribution computed from the polymer stress $\boldsymbol{\tau}_p^k$ on the bead surface with the subscript "p" dropped in the formula.

3. Compute the bead acceleration $a^k = \frac{F_M - F_D^k}{m}$, where the bead mass $m = \frac{4\pi}{3}\rho R^3$ in a buoyant case.

4. Update the bead velocity $U^{k+1} = U^k + \Delta t \cdot a^k$.

Note that the bead mass is small as the bead size is on the micrometer scale, and therefore the bead acceleration is very large compared to velocity. As a result, we have to use a very small time step size, and it will require a very long simulation to reach the steady state. This is the major drawback of the explicit time-stepping method, and currently we do not have a good solution to fix it. However, this motivates us to develop an effective implicit solver where a larger time step size could be used, and details will be discussed in Chapter 3. On the other hand, it is still useful to compare the velocities in the transient dynamics. In Table 2.4, we compare our simulated velocities after running the experiments for thousands of time steps with numerical results generated from the code in [8] at the same simulation time, with the magnetic force $F_M = \{2, 6, 10\}$ pN. The relative error $\frac{U_{\text{IBAMR}} - U_{\text{benchmark}}}{U_{\text{benchmark}}}$ is less than 2%, which serves as an additional verification check of our numerical method.

CHAPTER 3

Implicit Solver for the Constrained IB Formulation

The CIB method [31], a constraint-based IB method, has been used to model the interactions between fluids and moving or stationary rigid bodies. To impose constraints on the motion of the immersed body, traditional IB methods usually use a penalty formulation where each of the IB markers is tethered to its target position through a stiff spring [34, 35, 36], whereby an "approximate" Lagrange multiplier force is used to *approximately* impose the prescribed motion. Consequently, the penalty method relaxes the constraint and, in addition, may impose a time step size restriction. In contrast, the CIB method uses an *exact* Lagrange multiplier force to ensure that the markers move according to a prescribed velocity field, and it applies a Schur complement based preconditioner for solving the corresponding saddle point system. This method is demonstrated to be very efficient in solving the system while maintaining the rigid structure, and it is suitable to our problems at low Reynolds numbers, so we choose this method as our solver for the microbead simulations in Sec. 2.3. However, one of the major drawbacks is that the solver performance relies on some problem-specific parameters. As a result, we are motivated to develop a new preconditioner for the system to avoid the parameter re-tuning and dense linear algebra. The solver itself is an implicit method and is therefore numerically stable. In this chapter, we will discuss the mathematical formulations for the CIB method and the current version of the preconditioner. Then we will introduce our novel preconditioner and present test results from numerical experiments.

3.1 Linear System and Current Preconditioner

To simplify the notations, we first introduce the discrete formulation of the immersed boundary method described in Sec. 2.1.9. For a $d-$dimensional flow, we need to solve a linear system in the following form at each time step

$$\begin{bmatrix} \mathcal{A} & \mathcal{G} & \mathcal{S} \\ -\mathcal{D} & \mathbf{0} & \mathbf{0} \\ \mathcal{J} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \\ \boldsymbol{F} \end{bmatrix} = \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{h} = \mathbf{0} \\ \boldsymbol{W} \end{bmatrix}, \tag{3.1}$$

where $\mathcal{A} = \frac{\rho}{\Delta t}\mathcal{I} - \frac{\mu}{h^d}\mathcal{L}_h$ with $\mathcal{L}_h$ as the discrete vector-Laplacian operator. The first row of the matrix, $\begin{bmatrix} \mathcal{A} & \mathcal{G} & \mathcal{S} \end{bmatrix}$, corresponds to the momentum equation in Eq. (2.46), where $\boldsymbol{g}$ denotes all the additional forcing terms and $\mathcal{S}$ is the force spreading operator in Eq. (2.48). $\mathcal{G}$ and $\mathcal{D}$ are the discrete gradient operator and the discrete divergence operator, respectively; note that $\mathcal{G} = -\mathcal{D}^T$. $-\mathcal{D}\boldsymbol{u} = \mathbf{0}$ corresponds to the incompressibility equation in Eq. (2.47). $\mathcal{J}$ is the velocity interpolation operator, and $\mathcal{J}\boldsymbol{u} = \boldsymbol{W}$ enforces the condition that the structure moves at the prescribed velocity $\boldsymbol{W}$. Note that $\mathcal{S}$ and $\mathcal{J}$ are adjoint with respect to the weighted inner product $\mathcal{J} = \mathcal{S}^* = h^d\mathcal{S}$. We could adjust the matrix to be symmetric, which would require rescaling the right-hand side vector $\boldsymbol{W}$.

3.1.1 Current Preconditioner

Assume a d-dimensional problem is defined on a domain that is discretized using a N^d Cartesian grid, and we have a single structure described by N_b IB markers, then the matrix in the linear system in Eq. (3.1) has a size of $((d+1)N^2 + dN_b) \times ((d+1)N^2 + dN_b)$. Clearly, it is not practical to solve large linear systems like this directly, and we need an efficient and robust preconditioner. With the current scheme, the following steps are taken to construct the preconditioner.

1. The unconstrained fluid sub-problem:

   First, we use GMRES with a projection based preconditioner described in [37, 38] to solve the unconstrained fluid equation for pressure and velocity,

$$\begin{bmatrix} \mathcal{A} & \mathcal{G} \\ -\mathcal{D} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{u}} \\ \tilde{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{h} \end{bmatrix}. \tag{3.2}$$

2. Imposing the no-slip condition:

   The temporary velocity $\tilde{\boldsymbol{u}}$ and pressure $\tilde{p}$ solved in Eq. (3.2) do not satisfy the rigidity constraint. The slip velocity is computed as

$$\Delta \boldsymbol{V} = \boldsymbol{W} - \mathcal{J}\tilde{\boldsymbol{u}}, \tag{3.3}$$

and we solve the Schur complement system to enforce the no-slip condition at IB markers,

$$\mathcal{M}\boldsymbol{F} = \Delta \boldsymbol{V}, \tag{3.4}$$

where the Schur complement mobility matrix $\mathcal{M} = \mathcal{J}\mathcal{L}^{-1}\mathcal{S}$ is of size $dN_b \times dN_b$ and it maps the Lagrangian force to the interpolated velocity at IB markers. It is numerically impractical to compute the inverse of the Stokes operator for divergence-free flow $\mathcal{L}^{-1} = \mathcal{A}^{-1} - \mathcal{A}^{-1}\mathcal{G}(\mathcal{D}\mathcal{A}^{-1}\mathcal{G})^{-1}\mathcal{D}\mathcal{A}^{-1}$, and the existence of $\mathcal{L}^{-1}$ in between $\mathcal{J}$ and $\mathcal{S}$ makes $\mathcal{M}$ a dense matrix. As a result, the mobility matrix $\mathcal{M}$ is usually approximated using the Rotene-Prager-Yamakawa (RPY) tensor or fitted empirically. Detailed formulations are omitted in this dissertation, and we refer interested readers to [31].

The conditioning of the mobility matrix plays an important role in the conditioning of the overall system in Eq. (3.1), and it depends on the relative ratio between the fluid grid spacing Δx and the Lagrangian marker spacing Δs. If the Lagrangian markers are too far from each other compared to Δx, there will be fluid leak; however, if the markers are too close, it will lead to a rank-deficient $\mathcal{M}$. In addition, some other factors will also make an impact on the exact spacing, including the choice of the kernel function, dimensionality, whether we use a surface mesh or a filled geometry for the immersed structure, etc. Therefore, tuning the exact spacing is problem-specific.

3. The corrected fluid sub-problem:

For the optional last step, we correct the right-hand side vector, and solve the modified system:

$$\begin{bmatrix} \mathcal{A} & \mathcal{G} \\ -\mathcal{D} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \begin{bmatrix} \boldsymbol{g} + \mathcal{S}\boldsymbol{F} - \alpha \mathrm{vol}^{-1}\mathbf{1}^T\boldsymbol{F} \\ \boldsymbol{h} \end{bmatrix}, \tag{3.5}$$

where $\alpha = 1$ if $\mathcal{A}$ has a null-space, otherwise $\alpha = 0$. $\mathbf{1}$ is a vector of ones, and vol is the volume of the domain. The main goal for this step is to enforce momentum conservation for the steady Stokes flow on a fully periodic domain.

3.1.2 Linear Solver

We use Krylov subspace methods to solve the linear system Eq. (3.1) as well as the subsystems required for the preconditioner. The GMRES algorithm [39] is a robust solver for these problems. It uses the Arnoldi process to generate L_2 orthogonal basis vectors and solves a least squares problem to modify the approximations [40]. We also use the FGMRES method which allows for changes in the preconditioning at each iteration [41] to enable more efficient preconditioners and enhance robustness.

3.2 LSC Based Preconditioner

As discussed previously, the conditioning of the mobility matrix relies on the physical parameters specific to each problem, and bad choices will lead to the ill-conditioning of the linear system and poor performance of the solver. Motivated by this, we propose a new preconditioner for the linear system that is more general with comparable performance. In this section, we describe the derivation of a novel block preconditioner based on the Least Squares Commutator (LSC) and present numerical results.

3.2.1 Block Preconditioner

For a saddle point system in the form

$$\mathcal{C}x = \begin{bmatrix} C & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{3.6}$$

with a nonsingular (0,0) block C and a full rank (rectangular) matrix B, and the corresponding block preconditioner based on the Schur complement is defined as

$$P = \begin{bmatrix} C & 0 \\ 0 & S_c \end{bmatrix}, \; S_c = BC^{-1}B^T. \tag{3.7}$$

Then the preconditioned system $P^{-1}\mathcal{C}$ has only three distinct eigenvalues, and the Krylov solver (MINRES for symmetric C or GMRES for nonsymmetric C) will converge within three iterations [42]. Now the two key building blocks for the preconditioner are how to compute C^{-1} and S_c^{-1} efficiently. We already have some techniques to treat C^{-1} block of our problem, and will discuss later. The difficulty is how to approximate the inverse of the Schur complement S_c efficiently, as the existence of the inverse operator C^{-1} between two rectangular operators B and B^T makes S_c a dense operator [43].

3.2.2 Least Squares Commutator

The Least Squares Commutator preconditioner (LSC), proposed in [43], is a preconditioning strategy for solving saddle point systems. Using the notation of the saddle point system in Eq. (3.6), we can replace the inverse operator C^{-1} by an approximation operator C_p^{-1} at the end of the expression, and S_c is approximated by $\hat{S}_c$,

$$S_c \approxeq \hat{S}_c = BB^T C_p^{-1}. \tag{3.8}$$

In the case of equality $S = \hat{S}_c$, we are looking for a C_p such that

$$B^T C_p = CB^T. \tag{3.9}$$

This relation is derived by manipulating $BC^{-1}B^T = BB^T C_p^{-1}$ with linear algebra techniques:

$$\begin{gathered} BC^{-1}B^T = BB^T C_p^{-1} \Rightarrow B(C^{-1}B^T = B^T C_p^{-1}) \\ \Rightarrow C^{-1}B^T = B^T C_p^{-1} \Rightarrow B^T = CB^T C_p^{-1} \Rightarrow B^T C_p = CB^T. \end{gathered} \tag{3.10}$$

However, since Eq.(3.9) is an overdetermined system, we are unable to solve this algebraic problem exactly. Instead, we approximate C_p^{-1} by minimizing the Frobenius norm of $S_c - \hat{S}_c$ with the least squares method, i.e. $\min \|B^T C_p - CB^T\|_F$, and the solution is given by

$$C_p = (BB^T)^{-1} BCB^T, \tag{3.11}$$

and the corresponding Schur complement preconditioner can be written as

$$\hat{S}_c^{-1} = (BB^T)^{-1} BCB^T (BB^T)^{-1}. \tag{3.12}$$

In addition to a nonsingular C and a full rank B, we require C to be symmetric positive-definite (SPD) in order to apply this LSC based preconditioner. Now the only part that requires special

Matrix block	size
$\mathcal{C}_{00}$	$dN^2 \times dN^2$
$\mathcal{C}_{01}$	$dN^2 \times (N^2 + dN_b)$
$\mathcal{C}_{10}$	$(N^2 + dN_b) \times dN^2$
$\mathcal{C}_{11}$	$(N^2 + dN_b) \times (N^2 + dN_b)$

Table 3.1: The size of each block in the reformulated linear system in Eq. (3.14) for the immersed boundary formulation.

techniques is the inverse of the product of the two adjoint operators (BB^T). This is a challenging problem in general, and it often relies on the special structures of the matrix B. For example, for the incompressible Navier-Stokes problem, B is the divergence operator, and BB^T is simply the discrete Laplace operator. As a result, $(BB^T)x = y$ is just the Poisson problem which could be solved efficiently using our existing numerical methods. We will discuss options to handle this issue for our immersed boundary formulation in the next section.

3.2.3 Applying to Immersed Boundary Formulation

In this section, we discuss how to apply the preconditioning techniques described previously to our linear system in Eq. (3.1). As stated in Sec. (3.2.2), we require the (0,0) block to be SPD, and the (1,0) and (0,1) blocks are the transpose of each other and of full rank. Intuitively, we want to make the matrix block for the fluid sub-problem (the matrix in Eq. (3.2)) as our (0,0) block. However, this formulation does not satisfy the requirements as the matrix is indefinite. So we reformulate the blocks as

$$\mathcal{C} = \left[\begin{array}{c:cc} \mathcal{A} & \mathcal{G} & \mathcal{S} \\ \hdashline -\mathcal{D} & \mathbf{0} & \mathbf{0} \\ \mathcal{J} & \mathbf{0} & \mathbf{0} \end{array}\right] \tag{3.13}$$

$$= \begin{bmatrix} \mathcal{C}_{00} & \mathcal{C}_{01} \\ \mathcal{C}_{10} & \mathcal{C}_{11} \end{bmatrix}, \tag{3.14}$$

and it can be verified that the requirements on the blocks are met. For a problem in $d-$dimensional, given the number of fluid grid points in each direction N and the number of IB markers N_b, the size of each block is summarized in Table 3.1. Fast algorithms for $\mathcal{A}^{-1}$ are already available in IBAMR, such as the multigrid method, and therefore in this dissertation, we focus on the the Schur complement preconditioner

$$\hat{\mathcal{S}}_c^{-1} = \mathcal{M}^{-1}\mathcal{C}_{10}\mathcal{C}_{00}\mathcal{C}_{01}\mathcal{M}^{-1} \tag{3.15}$$

$$= \begin{bmatrix} \mathcal{D}\mathcal{G} & \mathcal{D}\mathcal{S} \\ \mathcal{J}\mathcal{G} & \mathcal{J}\mathcal{S} \end{bmatrix}^{-1} \begin{bmatrix} -\mathcal{D} \\ \mathcal{J} \end{bmatrix} \mathcal{A} \begin{bmatrix} \mathcal{G} & \mathcal{S} \end{bmatrix} \begin{bmatrix} \mathcal{D}\mathcal{G} & \mathcal{D}\mathcal{S} \\ \mathcal{J}\mathcal{G} & \mathcal{J}\mathcal{S} \end{bmatrix}^{-1}, \tag{3.16}$$

$$\text{with } \mathcal{M} = \mathcal{C}_{10}\mathcal{C}_{01}. \tag{3.17}$$

Note that $\mathcal{M}$ must be distinguished from the Schur complement mobility matrix discussed in Sec 3.1.1. However, the new $\mathcal{M}$ matrix plays a similar role by relating degrees of freedom on the Lagrangian marker points to each other. With simplified notations, the preconditioner can be written as

$$\mathcal{P}^{-1} = \begin{bmatrix} \mathcal{C}_{00}^{-1} & 0 \\ 0 & \mathcal{M}^{-1}\mathcal{C}_{10}\mathcal{C}_{00}\mathcal{C}_{01}\mathcal{M}^{-1} \end{bmatrix}. \tag{3.18}$$

We explore different methods for computing $\mathcal{M}^{-1}b$, including incomplete LU decomposition and block preconditioner $\hat{\mathcal{M}}^{-1}$ that solves some blocks of $\hat{\mathcal{M}}^{-1}\mathcal{M} = \mathbb{I}$ exactly. Currently, the best solution is to use a Krylov solver with a multigrid preconditioner, and we are continuously working on improving the preconditioner of $\mathcal{M}$ by utilizing its special structure. The numerical scheme for applying the preconditioner of the linear system Eq. (3.1) to the right-hand side vector $\begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{h} \\ \boldsymbol{W} \end{bmatrix}$ is as follows:

1. Solve $\mathcal{C}_{00}\boldsymbol{u} = \boldsymbol{g}$ with one multigrid V-cycle.
2. (a) Construct a preconditioner for $\mathcal{M}$ with one multigrid V-cycle and solve $\mathcal{M}\begin{bmatrix} \tilde{p} \\ \tilde{\boldsymbol{F}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{h} \\ \boldsymbol{W} \end{bmatrix}$ with GMRES (loose tolerance of 10^{-3}).
   (b) Apply the operators $\mathcal{C}_{01}$, $\mathcal{C}_{00}$, and $\mathcal{C}_{10}$ to $\begin{bmatrix} \tilde{p} \\ \tilde{\boldsymbol{F}} \end{bmatrix}$ in order.
   (c) Repeat the process in Step. (2a) and apply the operator to the resulting vector from Step. (2b) to get $\begin{bmatrix} p \\ \boldsymbol{F} \end{bmatrix}$.

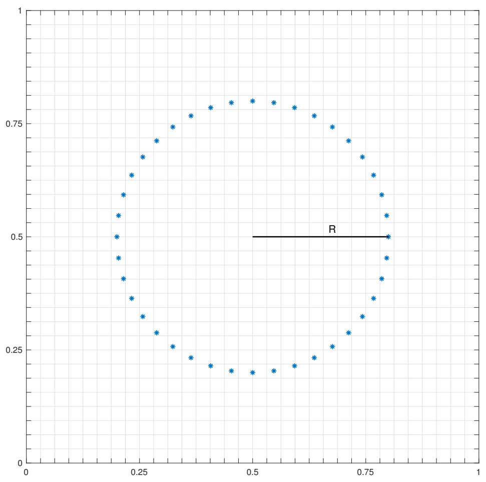


Figure 3.1: The computational domain and the immersed structure for testing the LSC based preconditioner in two-dimensional. The structure is a disk with radius R and it is described by IB markers (blue dots) put at a distance $\Delta s = 1.5\Delta x$.

3.2.4 MATLAB Prototype

We have implemented the routine as a MATLAB prototype with full details to test the effectiveness of the new preconditioner. As a test problem, we model an elastic structure as a disk with radius $R = 0.3$ located at the center of a unit square. We place the IB markers at a distance of 1.5 Cartesian grid spacings, as shown in Fig. 3.1. The Lagrangian force is described as a stretching force $\boldsymbol{F} = \kappa \frac{\partial \boldsymbol{X}^2}{\partial \boldsymbol{s}^2}$, where the constant κ characterizes the stiffness of the elastic material. The standard four-point kernel in Eq.(2.54) is used to construct the force-spreading operator and the velocity-interpolation operator.

Exact solve First, we verify the effectiveness of the block preconditioner discussed theoretically in Sec. 3.2.1. With three different Eulerian mesh spacings ($N = 32, 48, 64$), we put IB markers such that the relative ratio of IB marker spacings to fluid grid spacings is fixed at 1.5. We use GMRES to solve the linear system in Eq. (3.1) and summarize the number of iterations required to reduce the residual by a factor of 10^{-10} in Table. 3.2. As a baseline test (shown as Test 0), we solve the system without any preconditioner, and the required number of iterations grows in $\mathcal{O}(N)$. Clearly, the GMRES solver without a preconditioner is converging very slowly. Then for Test 1, we construct the Schur complement based block preconditioner in Eq. (3.7) with exact solve for the two diagonal blocks. The solver converges in three iterations which is independent of the grid spacing, in accordance with the theorem from [42]. Next, for Test 2, we replace $\mathcal{S}_c^{-1}$ by $\hat{\mathcal{S}}_c^{-1}$ defined in Eq. (3.16) and compute inverse of $\mathcal{M}$ directly. With this preconditioner, the required number of iterations is approximately $\mathcal{O}(\log N)$. The amount of work required at each iteration is $\mathcal{O}(N)$, so the time complexity for this solver is $\mathcal{O}(N \log N)$. Although in practice we need to avoid direct matrix inversion, this test demonstrates the strong performance of the LSC based preconditioner given an effective solver for $\mathcal{M}^{-1}$.

Number of Eulerian mesh points (N)	32	48	64
Number of IB markers (N_b)	40	60	80
Test 0: no preconditioner	496	724	983
Test 1: exact solve for $\mathcal{P}^{-1}$	3	3	3
Test 2: approximated $\mathcal{P}^{-1}$	9	11	11

Table 3.2: Number of iterations by GMRES for solving the linear system in Eq. (3.1) under grid refinement with no preconditioner, exact solve for $\mathcal{P}^{-1}$, and approximated $\mathcal{P}^{-1}$ with exact solve for $\mathcal{M}^{-1}$.

Approximated solve for $\mathcal{M}^{-1}$ We adopt the routine described in Sec. 3.2.3 for solving $\mathcal{M}x = b$ approximately, and apply it to the LSC preconditioner in Eq. (3.16). We refer to the solver for the linear system in Eq. (3.1) as the outer GMRES solver, while the solver for $\mathcal{M}x = b$ is the inner solver. The numbers of iterations required by the inner GMRES solver and the outer GMRES solver are summarized in Table 3.3. Note that the performance of the outer solver is comparable to the solver performance with exact solve for $\mathcal{M}^{-1}$ shown as Test 2 in Table 3.2. Test 2 benchmarks the optimal performance we could possibly achieve, therefore indicating a loose approximation of $\mathcal{M}^{-1}$ will suffice. Although the number of iterations is not exactly $\mathcal{O}(\log N)$, the outer solver is still considered to be scalable because the required number of iterations increases by approximately a constant number as we refine the mesh. Similarly, the inner solver is also scalable. Since in practice we solve the system with a matrix-free method and we need to dynamically construct the preconditioner instead of storing it, the goal is to reduce the total number of multigrid V-cycles performed at each time step. We also experiment with different numbers of V-cycles for both $\mathcal{C}_{00}$ block and $\mathcal{M}$, but we observe no improvement to the number of iterations required by the outer GMRES solver, compared with using just one V-cycle for both parts.

N	16	32	64
Outer solver	7	10	15
Inner solver	12	15	20

Table 3.3: Number of iterations required by the outer GMRES solver for the linear system in Eq. (3.1) and the inner GMRES solver for $\mathcal{M}x = b$.

Time-independent Stokes A more important question is whether we could apply this new preconditioner to time-independent Stokes flow, as for now we do not have an efficient implicit solver for it. The linear system now becomes

$$\mathcal{C}_{\text{independent}} = \left[\begin{array}{c:cc} -\frac{\mu}{h^d}\mathcal{L}_h & \mathcal{G} & \mathcal{S} \\ \hdashline -\mathcal{D} & \mathbf{0} & \mathbf{0} \\ \mathcal{J} & \mathbf{0} & \mathbf{0} \end{array}\right] \tag{3.19}$$

$$= \begin{bmatrix} \mathcal{C}_{00} & \mathcal{C}_{01} \\ \mathcal{C}_{10} & \mathcal{C}_{11} \end{bmatrix}. \tag{3.20}$$

The discrete Laplacian matrix $\mathcal{L}_h$ is indefinite, so we need to modify the (0,0) block in order to apply the block preconditioner. A good feature of the new $\mathcal{C}_{00}$ is that it is symmetric positive-definite on the null space of $\mathcal{C}_{10}$, thus we can use the augmented Lagrangian formulation [44] that is equivalent to Eq. (3.19):

$$\begin{bmatrix} \mathcal{C}_{00} + \gamma\mathcal{C}_{01}\mathcal{C}_{10} & \mathcal{C}_{01} \\ -\mathcal{C}_{10} & \mathcal{C}_{11} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} \\ \boldsymbol{F} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{h} \end{bmatrix} + \gamma\mathcal{C}_{01}\boldsymbol{W} \\ -\boldsymbol{W} \end{bmatrix}, \tag{3.21}$$

where γ is a positive scalar. The routine for solving $\mathcal{M}x = b$ remains the same as in the time-dependent case. The key question is how to compute the inverse of $\mathcal{C}_{00\text{aug}} = \mathcal{C}_{00} + \gamma\mathcal{C}_{01}\mathcal{C}_{10}$ efficiently, as $\mathcal{C}_{00\text{aug}}$ is a dense matrix without the special structure like the Laplacian. We test different γ values coupled with different numbers of multigrid V-cycles to compute $\mathcal{C}_{00\text{aug}}^{-1}$. Table 3.4 shows the number of iterations required by the outer GMRES solver to solve Eq. (3.21) with exact solve for $\mathcal{M}^{-1}$ on a grid with $N = 32$. We find $\gamma = 0.005$ to be optimal. Also, we notice that there is no significant improvement on the number of iterations beyond four multigrid V-cycles. We further demonstrate this by applying different numbers of V-cycles under grid refinement, as shown in Table 3.5.

Number of V-cycles/γ	1	0.1	0.01	0.005	0.001
1	191	65	43	42	44
2	116	41	34	33	36
3	86	34	30	31	33
4	72	31	29	30	32
5	62	27	28	30	32

Table 3.4: Number of iterations by the outer GMRES solver for solving the linear system in Eq. (3.21) with different numbers of V-cycles and γ values on a 32×32 mesh.

N/number of V-cycles	1	2	3	4	5
16	33	27	24	23	23
32	42	33	31	30	30
64	63	49	44	41	41

Table 3.5: Number of iterations by the outer GMRES solver for solving the linear system in Eq. (3.21) with different numbers of V-cycles under grid refinement, with $\gamma = 0.005$.

We also try replacing $C_{00\text{aug}}^{-1}$ by C_{00}^{-1} in the preconditioner to avoid the issue addressed previously, and C_{00}^{-1} can be handled easily with existing tools in the IBAMR software. Table 3.6 summarizes the number of iterations required by the outer solver. Compared to the performance in Table 3.5, this scheme requires many more iterations. However, the solver still appears to be scalable, and we have a simpler (0,0) block in the preconditioner to solve. This trade-off between the outer solver performance and the inner solver performance needs to be evaluated in terms of the total number of V-cycles.

Filled geometry Instead of using a spherical shell, we can also use a filled volumetric geometry. In the two-dimensional case, we could use a triangular mesh over the disk uniformly to describe the structure, as shown in Fig. 3.2. Note that in general it is not possible to set the precise marker distances, but we need to control the minimum distance between two markers and make sure the edge lengths are approximately equal. The $\mathcal{C}_{00}$ block and the corresponding block in the preconditioner remains unchanged, but the sizes of $\mathcal{C}_{01}$ and $\mathcal{C}_{10}$ are larger in this case. We perform numerical experiments with different grid spacing ratios and summarize the required numbers of iterations

N	16	32	64
Outer solver	50	64	84

Table 3.6: Number of iterations required by the outer GMRES solver for the augmented linear system (Eq. 3.21) using C_{00}^{-1} as the (0,0) block in the preconditioner .

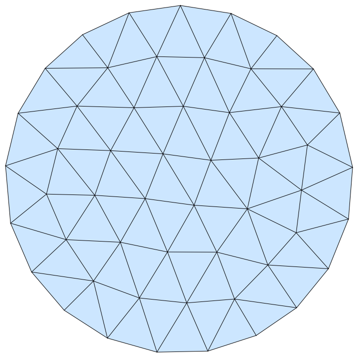


Figure 3.2: A two-dimensional disk filled by triangular mesh with IB markers placed at vertices.

	Solver	N = 16	N = 32	N = 64
$\Delta s = 1.5\Delta x$	Outer	9	15	
	Inner	60	310	1000+
$\Delta s = 2\Delta x$	Outer	10	12	15
	Inner	14	63	174
$\Delta s = 2.5\Delta x$	Outer	10	12	17
	Inner	9	17	32

Table 3.7: Number of iterations required by the outer GMRES solver for the linear system in Eq. (3.1) and the inner GMRES solver for $\mathcal{M}x = b$ using a filled geometry.

by the outer solver and the inner solver in Table 3.7 under grid refinement. Test results show that the method still works and scales if Lagrangian points are separated far enough, at a distance of at least $2.5\Delta x$.

3.2.5 Penalty Formulation and PETSc Implementation

We are working on an efficient parallel implementation of the algorithm and integrating it with the IBAMR software package, so that we could utilize the high performance solvers and tools built in IBAMR and its dependent libraries. We start with the penalty formulation of the immersed boundary method to simplify the linear algebra. Note that the major drawback of the penalty method is the time step size restriction for explicit time-stepping scheme, but the penalty method itself has no accuracy issues. Here, we employ the implicit time-stepping scheme, so using the penalty formulation could be a good place to start without loss of generality. For the IB method, the implicit time-stepping scheme computes the fluid variables and updates the force and position of the Lagrangian markers at time level $n+1$ as follows,

$$\frac{\rho}{\Delta t}(\boldsymbol{u}^{n+1}-\boldsymbol{u}^{n}) = -\nabla_h p^{n+1} + \mu\Delta_h \boldsymbol{u}^{n+1} + \mathcal{S}^n \boldsymbol{F}^{n+1}, \tag{3.22}$$

$$\nabla_h \cdot \boldsymbol{u}^{n+1} = 0, \tag{3.23}$$

$$\boldsymbol{F}^{n+1} = \kappa(\boldsymbol{X}^0 - \boldsymbol{X}^{n+1}), \tag{3.24}$$

$$\boldsymbol{X}^{n+1} = \boldsymbol{X}^n + \Delta t(\mathcal{S}^n)^* \boldsymbol{u}^{n+1}, \tag{3.25}$$

where we use the spring model to construct the body force $\boldsymbol{F}$ with spring constant κ and the equilibrium position $\boldsymbol{X}^0$. We can use Eqs. (3.24) and (3.25) to eliminate $\boldsymbol{F}^{n+1}$ in Eq. (3.22) and then formulate a system with Eulerian unknowns only:

$$\frac{\rho}{dt}(\boldsymbol{u}^{n+1}-\boldsymbol{u}^{n}) = -\nabla_h p^{n+1} + \mu\Delta_h \boldsymbol{u}^{n+1} + \kappa\mathcal{S}^n(\boldsymbol{X}^0 - \boldsymbol{X}^n - \Delta t(\mathcal{S}^n)^*\boldsymbol{u}^{n+1}), \tag{3.26}$$

$$\nabla \cdot \boldsymbol{u}^{n+1} = 0, \tag{3.27}$$

or in matrix form,

$$\begin{bmatrix} \frac{\rho}{\Delta t} - \mu\mathcal{L}_h + \kappa\Delta t\mathcal{S}\mathcal{S}^* & \mathcal{G} \\ -\mathcal{D} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{g}} \\ \boldsymbol{0} \end{bmatrix}. \tag{3.28}$$

We drop the superscripts in Eq. (3.28) to simplify the notations, and the right-hand side vector contains the terms from the time level n: $\tilde{\boldsymbol{g}} = (\frac{\rho}{\Delta t} + \mu\Delta_h)\boldsymbol{u}^n + \kappa\mathcal{S}^n(\boldsymbol{X}^0 - \boldsymbol{X}^n)$. The structure of the linear system in Eq. (3.28) is algebraically equivalent to the block structure in Eq. (3.14) with a symmetric positive-definite (0,0) block along with full rank (0,1) and (1,0) blocks which are transpose of each other. Recall that we define the "$\mathcal{M}$" matrix required for the LSC based preconditioner as the product of the $\mathcal{C}_{10}$ block and the $\mathcal{C}_{01}$ block, and in this formulation, $\mathcal{M} = -\mathcal{D}\mathcal{G}$, which is simply the Poisson problem that we can handle easily with fast algorithms available in the IBAMR package such as the matrix-free multigrid method. The hardest part for this formulation is to find a scalable preconditioner for $\mathcal{C}_{00} = \frac{\rho}{\Delta t} - \mu\mathcal{L}_h + \kappa\Delta t\mathcal{S}\mathcal{S}^*$ that is spectrally equivalent to $\mathcal{C}_{00}^{-1}$ with robust performance, especially for large κ or small ρ.

We have built an initial version of the solver based on the IBAMR framework, and we utilize the data structures from the SAMRAI library as well as the linear solvers built in PETSc. Here we present results from applications of this code. Similar to the MATLAB test problems, we test

κ / N	64	128	256	512
10	2	2	2	3
100	3	3	5	6
1000	6	7	10	14
10000	14	22	50	117

Table 3.8: Number of iterations required by the outer FGMRES solver to solve Eq. (3.28) with exact solvers for $\mathcal{C}_{00}^{-1}$ and $\mathcal{M}^{-1}$.

the code with an ellipse with width 0.25 and height 0.3 on a unit square with periodic boundary conditions discretized using a uniform $N \times N$ Cartesian grid. The fluid density $\rho = 1.0$ and viscosity $\mu = 0.01$. The time step size is fixed at 10^{-4}.

Exact solvers for $\mathcal{C}_{00}^{-1}$ and $\mathcal{M}^{-1}$ First, we solve $\mathcal{C}_{00}^{-1}$ and $\mathcal{M}^{-1}$ in the preconditioner Eq. (3.18) exactly by using a direct solver as the preconditioner and then solve the linear system in Eq. (3.28) with FGMRES. Table 3.8 summarizes the number of iterations required by the outer solver to reach a relative tolerance of 10^{-6}. The solver converges in a fixed number of iterations for $\kappa \leq 10$; for κ up to 1000, the number of iterations increases by approximately a constant number under grid refinement, so we still consider the solver to be scalable.

Ideally, we want to make $\kappa \sim \frac{1}{\Delta t}$ so that the structure is stiff enough to make IB points move less than $0.1\Delta x$ in relative positions to avoid deformations. In our case, when $\kappa = 10^4$, the number of iterations required is doubled as we refine the mesh by a factor of 2, so the solver does not scale any more. An area for improvement is to make the solver work for $\kappa \geq \frac{1}{\Delta}$.

Inexact solvers for $\mathcal{C}_{00}^{-1}$ and $\mathcal{M}^{-1}$ Next, we approximate $\mathcal{C}_{00}^{-1}$ and $\mathcal{M}^{-1}$ with the multigrid method and relax the relative tolerance for the inner solvers to 10^{-2}. The Poisson solver for $\mathcal{C}_{00}^{-1}$ usually converges 3 iterations. Prior publications [45] show that multgrid is spectrally equivalent to $\mathcal{C}_{00}^{-1}$ for this problem. On the other hand, we solve for $\mathcal{M}^{-1}$ with FGMRES and the multigrid preconditioner in the PETSc library. In Table. 3.9, we show the number of iterations required by the inner FGMRES solver for solving $\mathcal{C}_{00}^{-1}$ and the outer solver for the linear system Eq. (3.28). The inner solver converges in 1 to 2 iterations under grid refinement for different κ values, suggesting that the multigrid solver is scalable for this sub-problem. Also, note that the performance of the outer FGMRES solver is comparable to the solver with exact solvers for $\mathcal{C}_{00}^{-1}$ and $\mathcal{M}^{-1}$ shown in Table 3.8.

κ	solver	$N = 64$	$N = 128$	$N = 256$	$N = 512$
10	outer	3	3	3	3
	inner	1	1	1	1
100	outer	3	3	6	7
	inner	$1 \sim 2$	$1 \sim 2$	1	1
1000	outer	6	7	11	16
	inner	$1 \sim 2$	$1 \sim 2$	1	1
10000	outer	15	23	50	119
	inner	$1 \sim 2$	$1 \sim 2$	1	1

Table 3.9: Number of iterations required by the FGMRES solvers with inexact solvers for $\mathcal{C}_{00}^{-1}$ and $\mathcal{M}^{-1}$ with relative tolerance 10^{-2}.

Solver	N = 64	N = 128	N = 256
Outer	3	3	3
Inner	1	1	1

Table 3.10: Number of iterations required by the FGMRES solvers for the microbead model with inexact solvers for $\mathcal{C}_{00}^{-1}$ and $\mathcal{M}^{-1}$ with relative tolerance 10^{-2}.

This suggests that approximations to $\mathcal{C}_{00}^{-1}$ and $\mathcal{M}^{-1}$ with a loose tolerance will suffice. Again, the solver is scalable with κ up to 1000. The time complexity is $\mathcal{O}(N \log N)$ approximately, which is consistent with the results from the MATLAB prototype for the full formulation in Eq. (3.13).

Applications to the microbead simulation We also test the solver with parameters from the microbead model: fluid density $\rho = 1.3 \times 10^3$ kg/m^3 and viscosity $\mu_s = 5.0 \times 10^{-3}$ Pa·s. The exact solution for the fluid velocity is on the scale of 1.0×10^{-8} m/s. The stiffness constant $\kappa = \frac{1}{\Delta t} = 1.0 \times 10^4$. The outer FGMRES solver converges within 3 iterations, suggesting that we could potentially use this solver instead of CIB for the microbead simulations. See Table 3.10 for details.

Limitations and future work The implicit solver is a powerful solver for many applications where an explicit solver does not work. For example, the microbead simulation requires a very small time step size, and the numerical experiments generally cannot be run long enough to reach the steady state for reasons of running time and memory. For the implicit solver, we usually dynamically choose a time step size Δt that satisfies the convective CFL condition and is typically larger than a fixed time step size for the explicit solver,

$$\|\boldsymbol{u}\|_\infty \Delta t \leq C \Delta x, \tag{3.29}$$

where C is the convective CFL number [46]. However, one issue with the implicit solver is that for problems that can be solved by an explicit solver (e.g., flow-past-a-cylinder), the implicit solver is usually slower in terms of the elapsed time, despite the fact that the implicit solver can take fewer iterations with a larger time step size. Currently, we are actively exploring solutions to reduce the running time of the implicit solver. In addition, the current version of our solver works with $\kappa \leq \frac{0.1}{\Delta t}$, and we aim to improve the scheme so that it can solve problems with a larger $\kappa \sim \frac{1}{\Delta t}$.

CHAPTER 4

Stokes AMR

Adaptive mesh refinement (AMR) is a method for dynamically placing high spatial resolution only near key features where such precision is required [47]. Using a uniform grid over the entire domain generates a solution with a uniform accuracy that is usually not needed for practical applications, and the computational cost and required storage can exhibit quadratic growth for two-dimensional problems and cubic growth in three dimensions. We wish to use IBAMR's locally refined staggered-grid (marker-and-cell or MAC) fluid solver, but the discretization implemented in the software breaks down for the time-independent Stokes equations. However, the MAC scheme is equivalent to a discontinuous Galerkin scheme with a particular choice of numerical flux [48], and the finite element strategy for AMR (i.e., hanging nodes) works for this problem already. Hence, one way to solve it for finite differences is to describe coarsening and refinement in the same way we handle hanging nodes with quad and hex meshes with finite elements.

4.1 Existing Methods

The MAC scheme, first proposed in [49], is a discretization technique for modeling incompressible fluid flow using a so-called staggered grid. In particular, the computational domain is discretized into cells with velocities defined on cell sides and pressure defined at cell centers. The main advantage of this method is that it prevents the "checkerboard instability" in the velocity field and the pressure field that can occur with cell-centered discretizations. Another advantage is that we can achieve second-order accuracy using a stencil composed of just four neighbors of a cell in two spatial dimensions.

[51] proposes a locally-refined scheme based on quadtree. It uses a virtual stencil consisting of "real" nodes with true degrees of freedom (DOFs) or boundary values and ghost cells to define control volumes at the coarse-fine interface and boundaries. Local mass and momentum conservation is ensured by matching the fluxes at the sides of the coarse control volumes to the sum of the fluxes at the sides of the two fine control volumes.

[52] introduces a mimetic reformulation scheme of finite element methods using the natural divergence operator with pressure approximations by the piecewise constant space (Q_0) and velocity approximations by the the lowest-order Raviart-Thomas space (RT_0). The natural (primary) divergence operator is the coordinate-invariant definition of the analytic divergence, and it can be extended to RT_0. For a finite element partition $\mathcal{T}_h$, the natural divergence operator of $\boldsymbol{u}^h \in RT_0$ is defined as

$$\mathrm{DIV}(\boldsymbol{u}^h)|_{\kappa} = \frac{1}{\mu(\kappa)} \sum_{\boldsymbol{f} \in \mathcal{F}_h(\kappa)} \sigma_{\boldsymbol{f}} F_{\boldsymbol{f}}, \forall \kappa \in \mathcal{T}_h, \tag{4.1}$$

where $\sigma_{\boldsymbol{f}}$ is the unit normal vector and $\mathcal{F}_h(\kappa)$ is the set of oriented faces of κ with

$$\boldsymbol{u}^h = \sum_{\boldsymbol{f} \in \mathcal{F}_h} F_{\boldsymbol{f}} \boldsymbol{u}_{\boldsymbol{f}}, \forall \boldsymbol{u}^h \in RT_0. \tag{4.2}$$

The authors demonstrated first-order accuracy using RT_0 elements in the reformulated finite element methods on non-affine quadrilateral grids.

Similar to Eq. (4.1), one can construct the primary operators for the gradient operator and the curl operator, and the derived operators from the duality relation from the primary operators. [53] discusses the mimetic finite difference method based on the derived operators which can preserve or mimic the mathematical properties and physical laws of the underlying PDE problem. In future work, we plan to consider incorporating the mimetic scheme into our method.

[48] demonstrates that the MAC scheme is algebraically equivalent to the discontinuous Galerkin scheme (DG) using the lowest order Raviart-Thomas space with slight modifications based on the Legendre polynomials. As a result, the analysis on the DG scheme can be applied to the MAC scheme, and therefore it can generalize the MAC scheme to higher order and irregular meshes. Our numerical scheme is inspired by this equivalence.

[54] introduces an adaptive finite difference scheme where the hanging nodes on the coarse-fine interface are used to ensure mass and momentum conservation. In particular, volume flux is conserved through making the velocity on a coarse cell face equivalent to the average of the velocities of the two fine cells, for a refinement ratio of 2. Also, this method employs stress-based interpolation, in which the product of the viscosity and the velocity gradient is conserved at the coarse-fine interface.

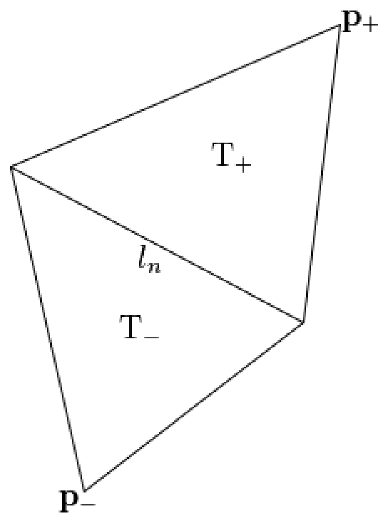


Figure 4.1: Definition of the lowest order Raviart Thomas element (RT_0), figure from: https://en.wikipedia.org/wiki/File:Raviart_thomas_labelled.png.

4.2 RT_0 Based Finite Difference Scheme

Specifically, we are solving some equation on a coarse grid and finer grids that are overlaid on these regions of the coarse grid tagged for refinement. We need to set up ghost nodes on the fine level, where we do not solve the equation at these nodes but we need these DOFs in our numerical stencil to solve at "real" nodes. We use a two-way coupling method by defining the source DOFs only on the fine level and then prolonging it to the coarse level, and by generating ghost values for the unknowns on the fine level from the variables on the coarse level. Hence the key question is how to define the ghost values for prolongation and restriction. The method we adopt here is to use the lowest order Raviart Thomas space (RT_0). As shown in Fig. 4.1, RT_0 elements are defined as:

$$\boldsymbol{f}_n(\boldsymbol{r}) = \begin{cases} \frac{ln}{2A_n^+}(\boldsymbol{r}-\boldsymbol{p}_+), & \text{if } \boldsymbol{r}\in\boldsymbol{T}_+ \\ -\frac{ln}{2A_n^-}(\boldsymbol{r}-\boldsymbol{p}_-), & \text{if } \boldsymbol{r}\in\boldsymbol{T}_- \\ \boldsymbol{0}, & \text{otherwise} \end{cases} . \tag{4.3}$$

For two-dimensional problems, staggered DOFs are defined to be piecewise linear in one direction, and piecewise constant in the other direction. Details will be discussed in the next two sections.

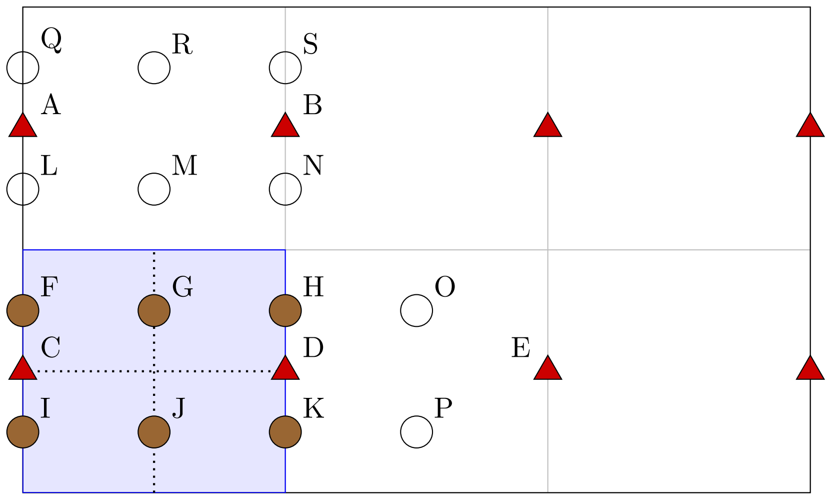


Figure 4.2: A schematic view of solving Poisson's equation with DOFs defined on x-edges. Here a corner refinement region (shaded in blue) is considered. DOFs on the coarse level are labeled as red triangles. DOFs that we are solving for on the fine level are marked as closed circles, while ghost nodes are represented by open circles.

4.3 Poisson's Equation

To illustrate the idea, we solve the two-dimensional Poisson equation with DOFs defined on x-edge centers and we use two levels of overlapping grids:

$$\Delta u = f. \tag{4.4}$$

The physical domain is taken to be a unit square and is discretized using a $N \times N$ Cartesian grid with mesh widths $\Delta x = \Delta y = h = \frac{1}{N}$ (assuming N is even). Dirichlet boundary conditions are imposed in the x-direction, and periodic boundary conditions in the y-direction. Using the standard five-point finite difference scheme, Eq. (4.4) is discretized as

$$\frac{-4u_{i,j} + u_{i+1,j} + u_{i-1,j} + u_{i,j-1} + u_{i,j+1}}{h^2} = f_{i,j}, \tag{4.5}$$

where $u_{i,j}$ is the unknown at $(ih, (j+\frac{1}{2})h)$. We consider four refinement cases that should cover most of the refinement scenarios:

- Refining the left (or right) half of the domain.
- Refining the top (or bottom) half of the domain.

- Refining a rectangular region of the domain at the boundaries (e.g., lower left corner covering $\frac{1}{4}$ of the domain).
- Refining an L-shaped region which is the opposite of the corner case.

Take the corner case as an example. As shown in Fig. 4.2, we refine the region in the lower left corner which is shaded in blue. u and f are defined at the centers of the x-edges (edges perpendicular to the x-axis), therefore they are piecewise linear in the x-direction and piecewise constant in the y-direction. The DOFs on the coarse level are labeled as triangles, while the DOFs on the fine level are shown as circles. The open circles represent the ghost nodes. To simplify the notation, we use letters A ... S to denote the locations, and u^i and f^i represent the unknown and the right hand side variable at location i respectively. Based on the definition of RT_0, we compute the ghost values as

$$u^L = u^A, u^M = \frac{1}{2}(u^A + u^B), u^N = u^B, \tag{4.6}$$

$$u^O = u^P = \frac{1}{2}(u^D + u^E). \tag{4.7}$$

The way we assign u^M, u^O, and u^P corresponds to the rule that the variable is piecewise linear in the x-direction, while for u^L and u^N we follow that the variable is constant on the same x-edge inside the same cell (constant in the y-direction). Note that we need to adjust for the boundary condition. For example, in the edge case where Fig. 4.2 is the entire domain, the stencil for point I, J, K will need u^Q, u^R, and u^S, respectively. An alternate version of Fig. 4.2 with only true DOFs is shown in Fig. 4.3 by removing ghost nodes for a non-periodic domain. On the other hand, we transfer information from the fine level to the coarse level by defining f^C and f^D as

$$f^C = \frac{1}{2}(f^F + f^I), f^D = \frac{1}{2}(f^H + f^K). \tag{4.8}$$

With the definitions above, we are able to set up a linear system in the form of $Au = f$, where A is a block matrix taking the form

$$\begin{bmatrix} A_{\text{coarse}} & 0 \\ B & A_{\text{fine}} \end{bmatrix} \begin{bmatrix} u_{\text{coarse}} \\ u_{\text{fine}} \end{bmatrix} = \begin{bmatrix} f_{\text{coarse}} \\ f_{\text{fine}} \end{bmatrix}. \tag{4.9}$$

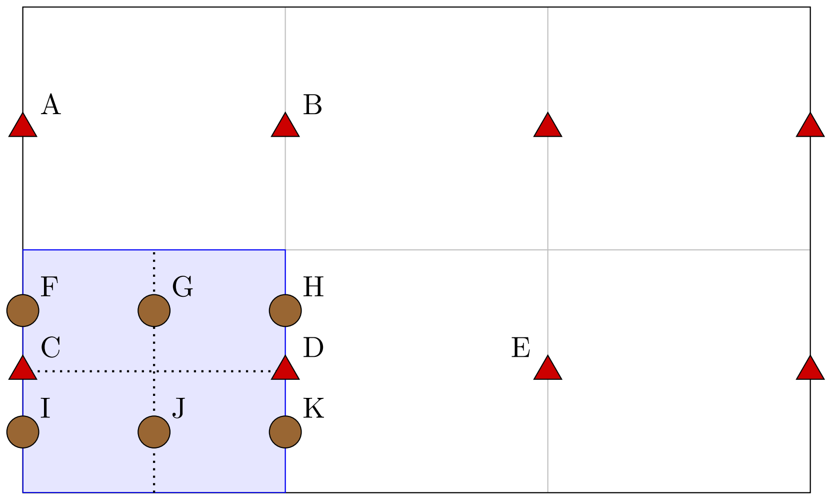


Figure 4.3: An alternate view of Fig. 4.2 (solving Poisson's equation), with true DOFs only.

The matrix B is a sparse matrix representing the coarse-fine coupling such as Eqs. (4.6) and (4.7). Note that we could form the matrix A explicitly and solve directly if the matrix size is small. Alternatively, and more efficiently, we could use the matrix-free implementation and solve the system in a more efficient way using iterative solvers. After solving the linear system, we prolong the DOFs to the coarse level from the fine level, in the overlapping region. For example, in Fig. 4.2,

$$u^C = \frac{1}{2}(u^F + u^I), u^D = \frac{1}{2}(u^H + u^K), \tag{4.10}$$

while the values u^A and u^B are computed from the coarse level only. For the first refinement case, the matrix A can be reformulated as a symmetric positive-definite matrix by incorporating these constraints and statically eliminating the "redundant" DOFs. However, for the other refinement cases, we lose symmetry in the situation when for example, in Fig. 4.2, the node B depends on K, but K is independent of B for a non-periodic domain. Obtaining symmetric positive-definite discretizations is a future goal, as it is important to have a SPD viscous operator for time-independent Stokes.

To validate our numerical scheme and investigate the order of convergence, we use the following exact solution to test our method:

$$u_{\text{exact}} = \sin(\pi x)\cos(2\pi x), \tag{4.11}$$

$$f_{\text{exact}} = -5\pi^2 \sin(\pi x)\cos(2\pi y). \tag{4.12}$$

We summarize the error in L_1, L_2, and L_∞ norms for each refinement case in Table. 4.1, where $\|e_{m\cdot h}\|_p = \|u_{m\cdot h} - u_{\text{exact}}\|_p$ represents the corresponding norm with grid spacing $m \cdot h$ on the coarse level, with $h = \frac{1}{128}$. Note that $u_{m\cdot h}$ is only evaluated on the coarse grid — it represents the solution on the coarse level if the region is not refined, and it is prolonged from the fine level using Eq. (4.10) in the overlapping region. The rate of convergence is also shown in the table, and we observe second-order accuracy for all cases in all three error norms. Fig. 4.4 visualizes the pointwise error for $h = \frac{1}{32}$. Note that as we expect, the largest errors occur at the coarse-fine interface. For the L-shaped refinement case, we also observe large errors near the periodic boundary where we obtain ghost values from the the other side of the boundary on the coarse level. This example generates promising results, and we could generalize it by incorporating DOFs defined on the y-edges and solving problems in higher dimensions with more refinement levels.

4.4 Stokes Equation

Given that our method works well for the Poisson problem, we then consider the two-dimensional Stokes equation:

$$\begin{cases} \mu\nabla^2\boldsymbol{u} - \nabla p &= \boldsymbol{f} \\ -\nabla\cdot\boldsymbol{u} &= 0 \end{cases}, \tag{4.13}$$

with $\mu = 1$ for simplicity. We use a staggered-grid finite difference scheme to discretize the equations. The physical domain is taken to be the periodic unit square and is discretized using a $N \times N$ Cartesian grid with mesh widths $\Delta x = \Delta y = h = \frac{1}{N}$ (assuming N is even for simplicity). The centers of the Cartesian grid cells are located at $\boldsymbol{x}^{i,j} = ((i-\frac{1}{2})h, (j-\frac{1}{2})h)$, where $i, j = 1, \ldots, N$. The pressure $p^{i,j} = p(\boldsymbol{x}^{i,j})$ is defined at cell centers. The x-component of the fluid velocity is defined at the centers of the x-edges of the grid cells, $u^{i+\frac{1}{2},j} = u(\boldsymbol{x}^{i+\frac{1}{2},j})$. Similarly, the y-component of the fluid velocity is defined at the centers of the y-edges of the grid cells, $v^{i,j+\frac{1}{2}} = v(\boldsymbol{x}^{i,j+\frac{1}{2}})$. The force components $\boldsymbol{f} = (f_1, f_2)$ are likewise defined at the centers of the x- and y-edges of the grid

Refinement case	Error term	L_1	L_2	L_∞
	$\|\|e_{4h}\|\|$	5.79×10^{-4}	7.21×10^{-3}	1.66×10^{-3}
Left	$\|\|e_{2h}\|\|$	1.42×10^{-4}	1.81×10^{-4}	4.64×10^{-4}
	$\|\|e_{h}\|\|$	3.53×10^{-5}	4.52×10^{-5}	1.22×10^{-4}
	$\log_2(\|\|e_{4h}\|\|/\|\|e_{2h}\|\|)$	2.02	1.99	1.84
	$\log_2(\|\|e_{2h}\|\|/\|\|e_{h}\|\|)$	2.01	2.00	1.92
	$\|\|e_{4h}\|\|$	9.64×10^{-4}	1.24×10^{-3}	3.47×10^{-3}
Top	$\|\|e_{2h}\|\|$	2.36×10^{-4}	3.11×10^{-4}	9.12×10^{-4}
	$\|\|e_{h}\|\|$	5.80×10^{-5}	7.80×10^{-5}	2.34×10^{-4}
	$\log_2(\|\|e_{4h}\|\|/\|\|e_{2h}\|\|)$	2.03	2	1.93
	$\log_2(\|\|e_{2h}\|\|/\|\|e_{h}\|\|)$	2.02	2.00	1.96
	$\|\|e_{4h}\|\|$	7.95×10^{-4}	1.01×10^{-3}	3.26×10^{-3}
Corner	$\|\|e_{2h}\|\|$	1.93×10^{-4}	2.50×10^{-4}	8.73×10^{-4}
	$\|\|e_{h}\|\|$	4.77×10^{-5}	6.22×10^{-5}	2.27×10^{-4}
	$\log_2(\|\|e_{4h}\|\|/\|\|e_{2h}\|\|)$	2.04	2.00	1.90
	$\log_2(\|\|e_{2h}\|\|/\|\|e_{h}\|\|)$	2.02	2.01	1.94
	$\|\|e_{4h}\|\|$	7.28×10^{-4}	9.71×10^{-4}	3.25×10^{-3}
L-shape	$\|\|e_{2h}\|\|$	1.80×10^{-4}	2.45×10^{-4}	8.76×10^{-4}
	$\|\|e_{h}\|\|$	4.47×10^{-5}	6.51×10^{-5}	2.28×10^{-4}
	$\log_2(\|\|e_{4h}\|\|/\|\|e_{2h}\|\|)$	2.02	1.99	1.89
	$\log_2(\|\|e_{2h}\|\|/\|\|e_{h}\|\|)$	2.00	1.99	1.94

Table 4.1: Error analysis for solving Poisson's equation on overlapping grids with four refinement cases.

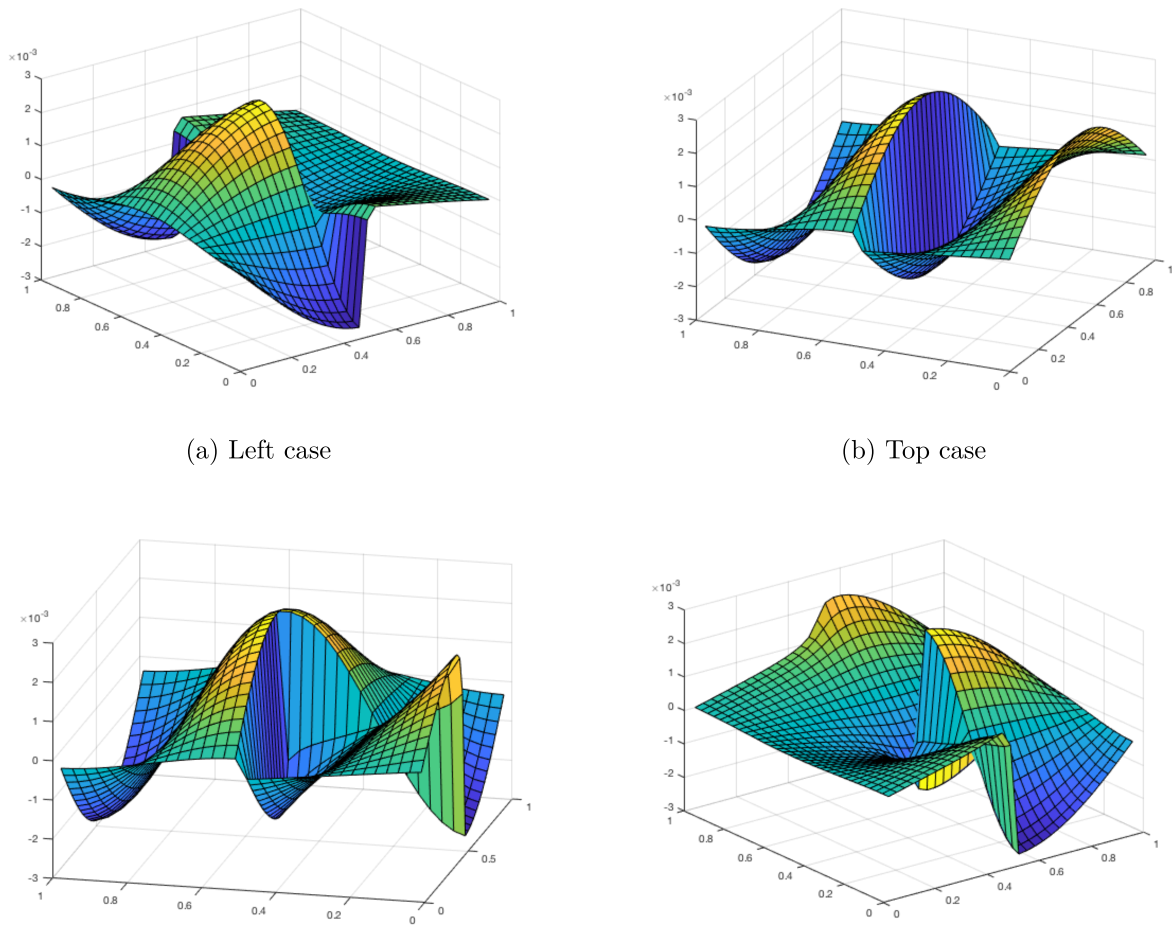
cells, respectively. As an example, Fig. 4.5 illustrates the staggered grid layout. The velocity DOFs, defined at side centers, are marked as circles with u in red and v in black. The pressure DOFs are shown as blue squares at cell centers.

Using the standard five-point finite difference stencil, we can write the discretized form of the equations as

$$\begin{aligned}
\frac{4u^{i+\frac{1}{2},j} - u^{i-\frac{1}{2},j} - u^{i+\frac{3}{2},j} - u^{i+\frac{1}{2},j-1} - u^{i+\frac{1}{2},j+1}}{h^2} + \frac{p^{i+1,j} - p^{i,j}}{h} &= f_1^{i+\frac{1}{2},j} \\
\frac{4v^{i,j+\frac{1}{2}} - v^{i-1,j+\frac{1}{2}} - v^{i+1,j+\frac{1}{2}} - v^{i,j-\frac{1}{2}} - v^{i,j+\frac{3}{2}}}{h^2} + \frac{p^{i,j+1} - p^{i,j}}{h} &= f_2^{i,j+\frac{1}{2}} \\
\frac{u^{i+\frac{1}{2},j} - u^{i-\frac{1}{2},j}}{h} + \frac{v^{i,j+\frac{1}{2}} - v^{i,j-\frac{1}{2}}}{h} &= 0
\end{aligned} \tag{4.14}$$

Note that stencils involving boundary nodes need to be adjusted accordingly.

Again, we consider the four refinement cases listed in Sec. 4.3 with two overlapping grids. Here, we just discuss the case with an L-shaped refinement region which should cover all scenarios for ghost values computations. On the finer level where an L-shaped domain is refined, we divide the domain



(a) Left case (b) Top case

(c) Corner case (d) L-shaped case

Figure 4.4: Error plots for solving Poisson's equation on overlapping grids covering a unit square with grid spacing $h = \frac{1}{32}$. Four refinement cases are considered.

into two parts: a square on the top ($[0, 0] \times [0.5, 0.5]$) and a rectangular domain on the bottom ($[0, 0.5] \times [1, 1]$), as shown in Fig. 4.6. The top domain is discretized using a $N \times N$ Cartesian grid with grid spacing $\frac{h}{2}$, and the bottom domain is discretized using $2N \times N$ grid. We use the standard five-point stencil in Eq. (4.14) for interior nodes, and adjust the stencil for periodicity and the top-bottom interface. The representation of u is defined to be piecewise linear in the x-direction and piecewise constant in the y-direction; the representation of v is piecewise constant in the x-direction and piecewise linear in the y-direction; the representation of p is cell-wise constant. Specifically, we handle the ghost values for u in the same as we handle the DOFs for the two-dimensional Poisson problem in Fig. 4.2 using Eqs. (4.6) and (4.7). We treat the ghost values for v likewise. For example, assume Fig. 4.7 shows a small part of the domain far from the boundary, the triangles represent

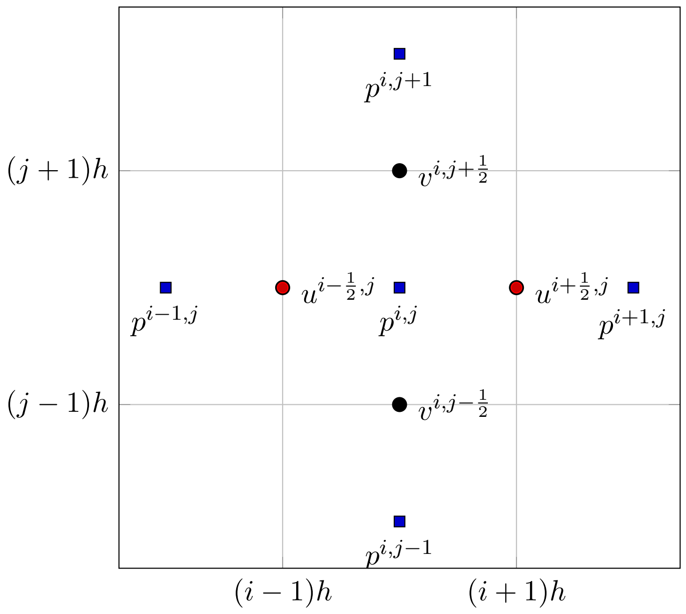


Figure 4.5: A schematic view of the staggered grid layout for solving the two-dimensional Stokes problem, with velocity DOFs defined on edges and pressure DOFs defined at cell centers.

the DOFs on the coarse level, while closed circles and open circles correspond to the DOFs that we solve for and ghost nodes on the fine level respectively. We can define the ghost values as

$$v^L = v^M = \frac{1}{2}(v^A + v^B), \tag{4.15}$$

$$v^N = v^C, v^O = \frac{1}{2}(v^C + v^E), v^P = v^E, \tag{4.16}$$

while the right hand side vector f_2 at locations B and C is prolonged from the fine level,

$$f_2^B = \frac{1}{2}(f_2^F + f_2^G), f_2^D = \frac{1}{2}(f_2^J + f_2^K). \tag{4.17}$$

Since the representation of the pressure is taken to be constant over the entire cell, the ghost node value is the same as the pressure in the corresponding coarse cell. Take Fig. 4.8 as an example, where we use the same legends as before. We need the pressure at ghost nodes C, D, E, and F,

$$p^C = p^D = p^A, \tag{4.18}$$

$$p^E = p^F = p^B. \tag{4.19}$$

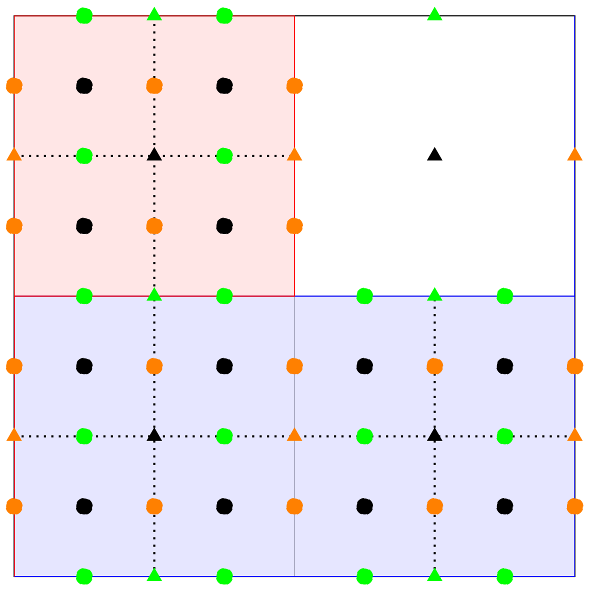


Figure 4.6: Illustration of the L-shaped refinement region, where we divide the fine level into a square (shaded in red) and a rectangular region (shaded in blue). DOFs on the coarse level are shown as triangles, and DOFs on the fine level are shown as circles. u, v, and p are labeled in orange, green, and black, respectively. Here we show the true DOFs only.

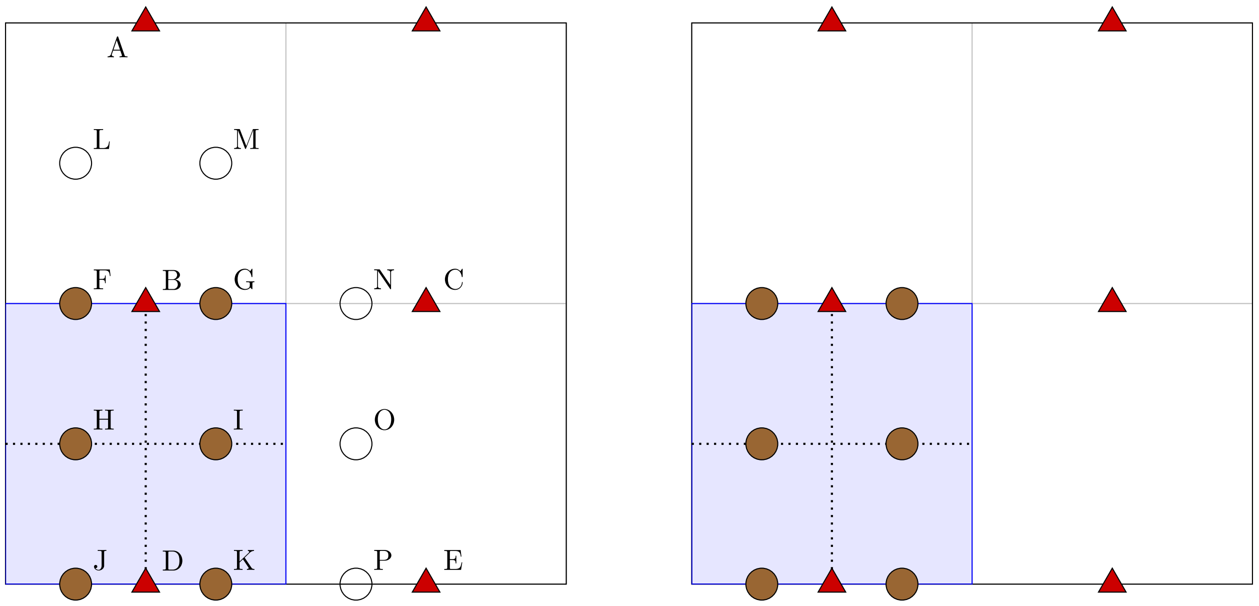


Figure 4.7: Illustration of the v-component of the velocity field for the Stokes problem. In the left panel, DOFs on the coarse level are labeled as triangles. For a corner refinement region (shaded in blue), true DOFs on the fine level are marked as closed circles, while ghost nodes are represented by open circles. The right panel shows the true DOFs only.

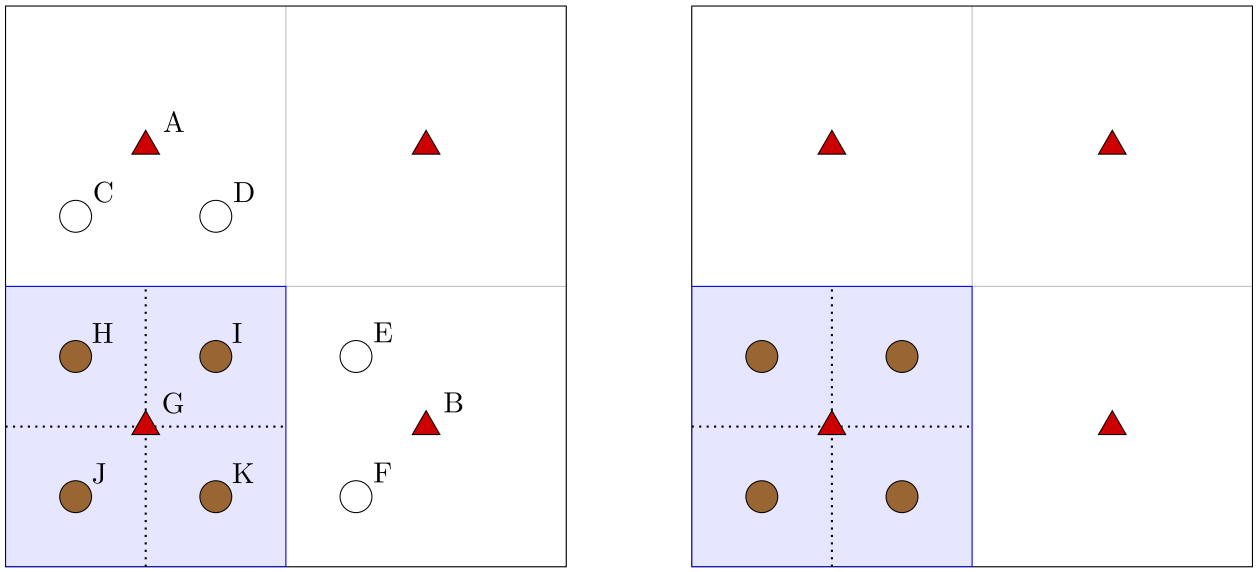


Figure 4.8: Illustration of the pressure DOFs for the Stokes problem. DOFs on the coarse level are labeled as triangles. For a corner refinement region (shaded in blue), true DOFs on the fine level are marked as closed circles, while ghost nodes are represented by open circles. The right panel shows true DOFs.

And the coarse-grid values on the coarse-fine interface, such as p^G, are post-processed as

$$p^G = \frac{1}{4}(p^H + p^I + p^J + p^K). \tag{4.20}$$

Note that with this scheme, we are not refining the pressure DOFs. However, accurate locally conservative velocity approximations are favored over pressure field computations, as the velocity field is usually more important than the pressure [52]. In a future study, we plan to focus on the refinement scheme for the pressure field.

We need to handle the stencil carefully for the nodes near the coarse-fine interface or the periodic boundaries. For example, in the L-shaped refinement region shown in Fig. 4.6, v (green dots) on the red horizontal line $y = 0.5$ is only solved once; in other words, those DOFs either belongs to the top mesh or the bottom mesh and will be used as ghost values for stencils of nodes on the other mesh. The discrete problem can be written in the matrix form,

$$\begin{bmatrix} \begin{bmatrix} A_{\text{coarse}} & B^T_{\text{coarse}} \\ B_{\text{coarse}} & \tilde{0} \end{bmatrix} & 0 & 0 \\ 0 & \begin{bmatrix} A_{\text{top}} & B^T_{\text{top}} \\ B_{\text{top}} & 0 \end{bmatrix} & 0 \\ 0 & 0 & \begin{bmatrix} A_{\text{bottom}} & B^T_{\text{bottom}} \\ B_{\text{bottom}} & 0 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} u_{\text{coarse}} \\ v_{\text{coarse}} \\ p_{\text{coarse}} \end{bmatrix} \\ \begin{bmatrix} u_{\text{top}} \\ v_{\text{top}} \\ p_{\text{top}} \end{bmatrix} \\ \begin{bmatrix} u_{\text{bottom}} \\ v_{\text{bottom}} \\ p_{\text{bottom}} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} f_{\text{coarse},1} \\ f_{\text{coarse},2} \\ \tilde{0} \end{bmatrix} \\ \begin{bmatrix} f_{\text{top},1} \\ f_{\text{top},2} \\ 0 \end{bmatrix} \\ \begin{bmatrix} f_{\text{bottom},1} \\ f_{\text{bottom},2} \\ 0 \end{bmatrix} \end{bmatrix}. \quad (4.21)$$

Note that we remove the null space for pressure by adding $\frac{1}{h}$ to the last element to the (1,1) block for the coarse level (resulting in $\tilde{0}$ instead of a zero matrix), and add $\frac{1}{h}p^{N,N}$ to the corresponding right hand side vector element. Recall that after solving the linear system, we also need to post-process the solutions u, v, and p on the coarse level in the overlapping regions as the average of the corresponding find-grid solutions.

4.4.1 Implementation

Currently, we are using a MATLAB prototype to perform computational experiments and validate our numerical scheme. An improved implementation based on the library PETSc library is under development. We will incorporate the solver into the IBAMR software package and provide more test results in the future.

4.4.2 Numerical Results

An analytical solution in 2D is given by:

$$[u, v] = [\sin(4\pi x)\cos(4\pi y), -\cos(4\pi x)\sin(4\pi y)], \quad (4.22)$$

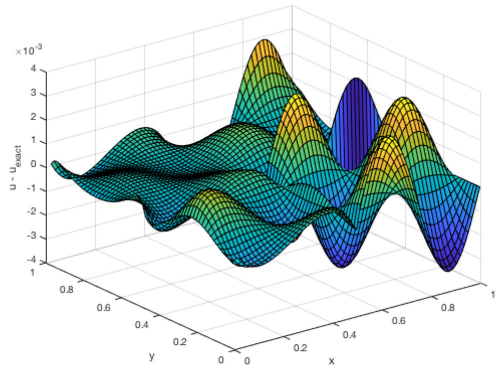
$$p = \pi\cos(4\pi x)\cos(4\pi y), \quad (4.23)$$

$$\boldsymbol{f} = [28\pi^2\sin(4\pi x)\cos(4\pi y), -36\pi^2\cos(4\pi x)\sin(4\pi y)]. \quad (4.24)$$

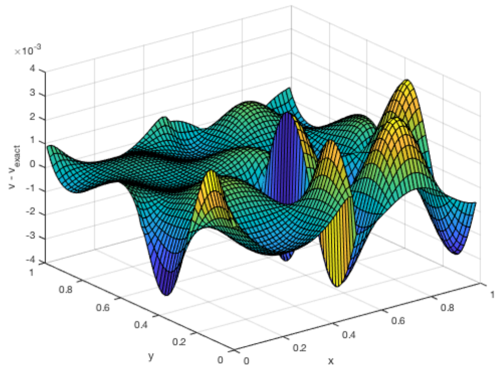
		L_1	L_2	L_∞
	$\lVert e_{4h}\rVert$	2.52×10^{-3}	3.82×10^{-3}	1.27×10^{-2}
	$\lVert e_{2h}\rVert$	6.58×10^{-4}	9.74×10^{-4}	3.22×10^{-3}
u	$\lVert e_{h}\rVert$	1.70×10^{-4}	2.48×10^{-4}	8.79×10^{-4}
	$\log_2(\lVert e_{4h}\rVert/\lVert e_{2h}\rVert)$	1.94	1.97	1.99
	$\log_2(\lVert e_{2h}\rVert/\lVert e_{h}\rVert)$	1.95	1.97	1.87
	$\lVert e_{4h}\rVert$	2.53×10^{-3}	3.83×10^{-3}	1.28×10^{-2}
	$\lVert e_{2h}\rVert$	6.57×10^{-4}	9.74×10^{-4}	3.22×10^{-3}
v	$\lVert e_{h}\rVert$	1.69×10^{-4}	2.48×10^{-4}	8.80E-04
	$\log_2(\lVert e_{4h}\rVert/\lVert e_{2h}\rVert)$	1.95	1.97	1.99
	$\log_2(\lVert e_{2h}\rVert/\lVert e_{h}\rVert)$	1.96	1.97	1.87
	$\lVert e_{4h}\rVert$	4.43×10^{-2}	6.39×10^{-2}	2.82×10^{-1}
	$\lVert e_{2h}\rVert$	1.23×10^{-2}	1.83×10^{-2}	8.58×10^{-2}
p	$\lVert e_{h}\rVert$	3.25×10^{-3}	4.96×10^{-3}	2.36×10^{-2}
	$\log_2(\lVert e_{4h}\rVert/\lVert e_{2h}\rVert)$	1.85	1.80	1.72
	$\log_2(\lVert e_{2h}\rVert/\lVert e_{h}\rVert)$	1.92	1.89	1.86

Table 4.2: Error analysis for the incompressible Stokes problem with an L-shaped region refinement.

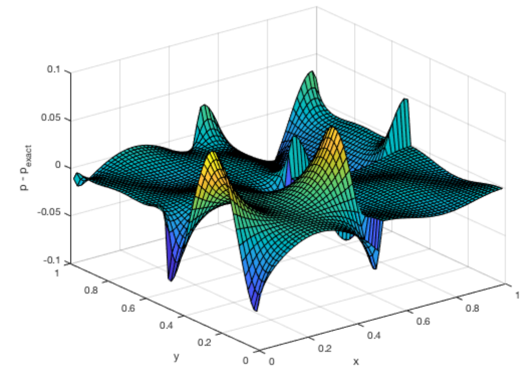
We perform a grid convergence study and solve the Stokes problem on three uniform grids with grid spacings $\frac{1}{32}, \frac{1}{64}, \frac{1}{128}$ on the coarse level respectively. The error norms and the rates of convergence are summarized in Table. 4.2 with $h = \frac{1}{128}$. We observe approximately second-order convergence for both velocity and pressure. Note that similar to the Poisson's equation discussed previously, the order of accuracy is only being measured from the post-processed solution on the coarse grid. The pointwise errors on the coarse grid are plotted in Fig. 4.9. As we expect, the error is the largest at the coarse-fine interface, and it decays rapidly away from the interface. Since we are more interested in the velocity field, we compare our results to the error norms for solving the Stokes equation on a single level uniform grid with mesh width h, as shown in Fig. 4.10. Note that the solution is converging in second-order on the single level grid, which is the best rate of convergence we could possibly achieve by using an L-shaped refinement region. In addition, with the same mesh width on the coarse level, the error norms of our refinement method are about half of the error norms of solving on just one level.



(a) u-component of the velocity field

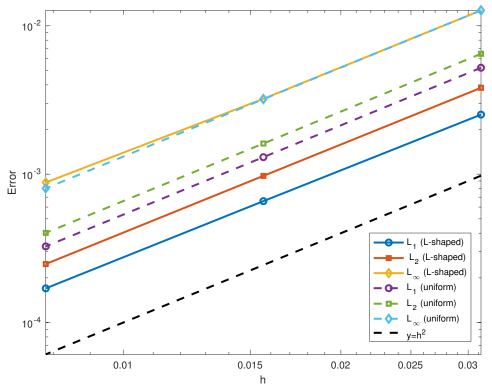


(b) v-component of the velocity field

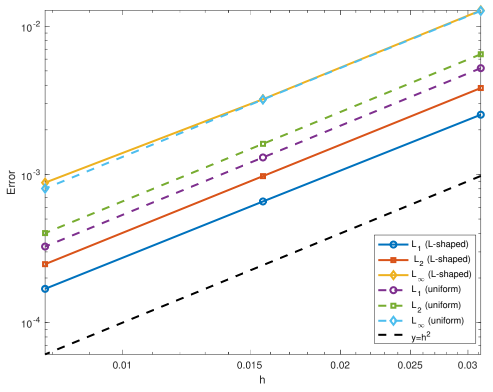


(c) Pressure field p

Figure 4.9: Error plots of the velocity field and the pressure field for solving the incompressible Stokes problem with an L-shaped refinement region.



(a) u-component of the velocity field



(b) v-component of the velocity field

Figure 4.10: Error norms of the velocity field for solving the incompressible Stokes problem with a single level uniform grid versus two levels with an L-shaped refinement region.

APPENDIX A

Moving Least Squares

As an alternative to the standard delta function kernels, the moving least squares (MLS) interpolation method can be employed when we spread the force from Lagrangian markers on the immersed structure to Eulerian nodes of the Cartesian grid or interpolate the fluid velocity to the Lagrangian mesh. MLS, first proposed in [55], reconstructs a continuous function by weighted least squares approximations given function values at sample points around the target point within the support of the kernel function.

A.1 One-sided MLS Method

For a material point at position $\boldsymbol{\chi}$ on the background Cartesian grid, we first find the corresponding Cartesian grid cell and determine the interpolation stencil based on the chosen kernel function. For a stencil with m interpolation points, we use the standard kernel function to compute the regular IB weights for each dimension and store in a $d \times m$ matrix $\boldsymbol{D}$ where $D_{k,i}$ corresponds to the weight for the ith interpolation point in dimension k. For example, Eq. (2.54) gives the weights for the standard four-point kernel. Next, we compute the tensor product of the one-dimensional kernel functions as $\boldsymbol{T}$ where

$$T_{i_1,i_2} = D_{1,i_1} D_{2,i_2}, i_1, i_2 = 1 \dots m, \text{ for } d = 2, \tag{A.1}$$

$$T_{i_1,i_2,i_3} = D_{1,i_1} D_{2,i_2} D_{3,i_3}, i_1, i_2, i_3 = 1 \dots m, \text{ for } d = 3. \tag{A.2}$$

Here we only discuss the two-dimensional case to simplify the notations, and it could be generalized to the three-dimensional case easily. To set up the least squares problem, we define $\boldsymbol{p}$ as the vector of basis functions $\boldsymbol{p} = [1; \boldsymbol{\chi}]$ and $\boldsymbol{G}$ as the Gram matrix where

$$G_{j,k} = \sum_{i_1,i_2=1}^{m} T_{i_1,i_2} x_{i_1,j} x_{i_2,k}, \text{ with } j, k = 1, 2, \tag{A.3}$$

where $x_{i_1,j}$ is the j-th element of the vector $\boldsymbol{x}_{i_1}$ consisting of 1 and the position of i_1-th interpolation point in the stencil, similar to $\boldsymbol{p}$. $x_{i_2,k}$ is defined likewise. Then we solve for the Lagrange multiplier $\boldsymbol{L}$ in the equation

$$\boldsymbol{GL} = \boldsymbol{p}. \tag{A.4}$$

The modified MLS weights $\boldsymbol{\psi}$ are therefore defined as

$$\psi_{i_1,i_2} = T_{i_1,i_2}(L_1 + L_2 x_{i_1,2} + L_3 x_{i_2,3}), \tag{A.5}$$

which replace the standard kernel function in Eq. (2.53) for spreading and interpolation.

Note that by one-sided MLS, we set up a mask function taking on values in $[0, 1]$ to determine where to interpolate, and compute the function value at all Cartesian nodes. When we compute the regular IB weights in Eq. (A.1), we also multiply the tensor product by the corresponding mask function value. For example, a simple discrete mask function has the value 1 at locations where we want perform MLS and the value 0 everywhere else. When the mask data is set to be 1 for all Cartesian nodes, it is equivalent to the regular IB kernel without MLS.

A.2 Numerical Tests

We use numerical examples in two spatial dimensions proposed in [30] to verify and investigate the accuracy of the MLS method. In particular, we are interested in the pressure-driven flow inside a channel.

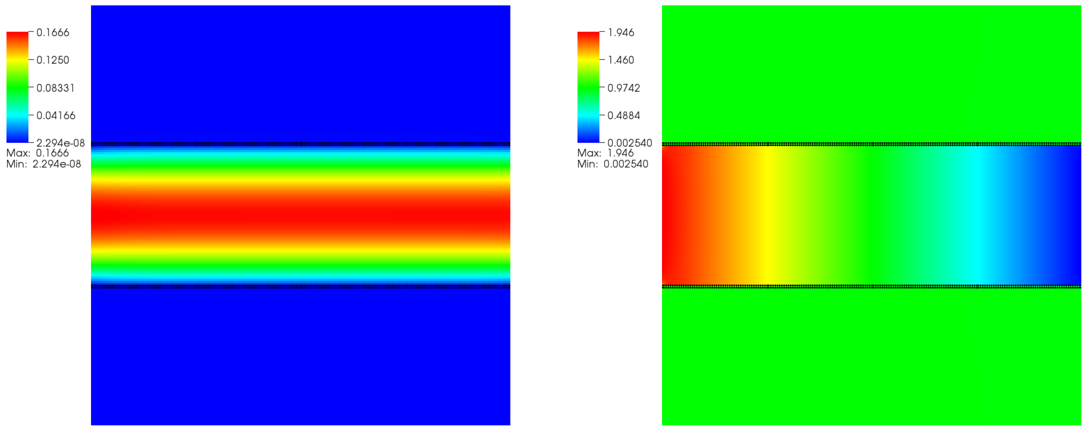
A.2.1 Grid-aligned Channel

The first example considers a grid-aligned horizontal channel, as shown in Fig. A.1. On a square domain with side length L, a horizontal channel of length L and width H is set up in the middle of the domain, so that the lower wall is at $y_0 = \frac{1}{2}(L - H)$. The Lagrangian nodes for the channel walls, shown as black dots, are placed at a distance of twice the background grid spacing. Consider a flow subject to constant pressure at the inlet and outlet, we can use the plane Poiseuille equation to model the flow,

$$\begin{aligned} u(y) &= \frac{p_0}{2\mu}(y - y_0)\left(1 - \frac{y - y_0}{H}\right), \\ v &= 0, \\ \frac{dp}{dx} &= -p_0, \end{aligned} \tag{A.6}$$

with a constant pressure gradient p_0 in the x-direction throughout the channel. The parameters we use are summarized in Table A.1.

The mask function value is set to be 1 outside the channel and 0 inside the channel. The normal and tangential velocity components are prescribed at the inlet and outlet, while no-slip conditions



(a) Velocity field (b) Pressure field

Figure A.1: Visualization of the steady-state flow in a horizontal channel with channel walls represented by black dots. We use one-sided MLS to spread and interpolate from outside the channel. The Cartesian grid spacing used in this simulation is $\Delta x = 2^{-7}$.

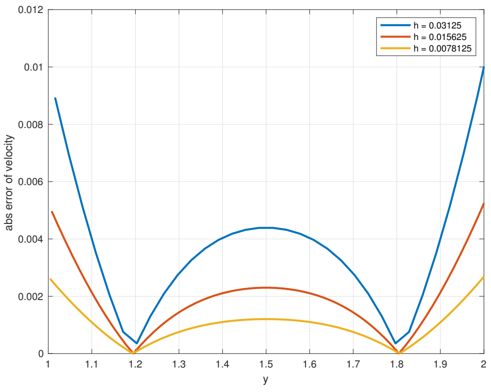
L	H	ρ	μ	p_0
3.0	1.0	1.0	0.5	$\frac{2}{3}$

Table A.1: Model parameters used for simulating plane Poiseuille flow in a horizontal channel.

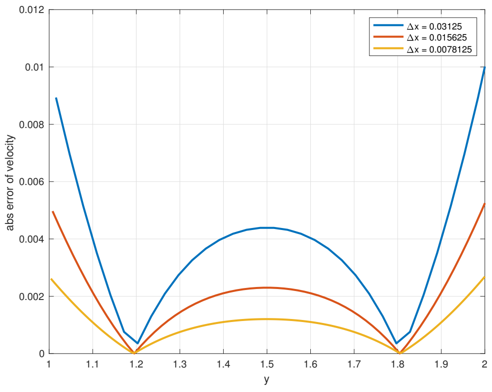
are imposed along the solid walls. The steady-state velocity field and pressure field are visualized in Fig. A.1, and both fields are smooth. Given the exact solution in Eq. (A.6), we are able to compute the error analytically. In particular, we are interested in the velocity profile perpendicular to the wall in the middle part of the channel, the velocity gradient along the lower wall, and the pressure along the centerline. Fig. A.2 visualizes the pointwise absolute error under grid refinement. Note that large errors occur near the inlet/outlet and at the walls. Fig. A.3 shows the errors of these three quantities in L_1, L_2, and L_∞ norms. First-order convergence of the velocity and the pressure is observed for all three error norms. The order of accuracy in the L_1 and L_2 norms of the velocity gradient is smaller than first-order, while it is not convergent in L_∞ norm.

In Table A.2, we show that the error norms of the pressure field solved using one-sided MLS are less than half of the error norms from the regular delta kernel functions. We are getting very close error values for the velocity and the velocity gradient with the two methods.

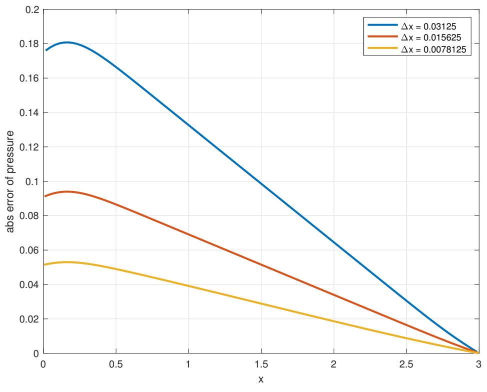
Next, we use the same channel configuration and simulate a viscoelastic fluid. The exact analytical solutions of the steady-state velocity and conformation tensor $\boldsymbol{C}$ solution of Poiseuille



(a) Velocity



(b) Velocity gradient

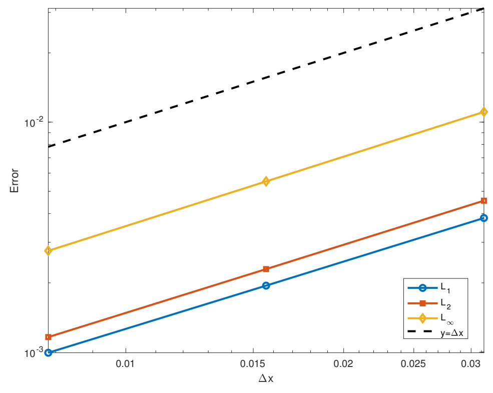


(c) Pressure

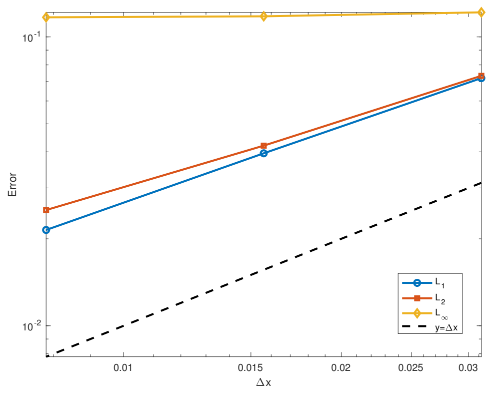
Figure A.2: Pointwise error of velocity, velocity gradient, and pressure for plane Poiseuille flow in a horizontal channel.

Quantity	Method	L_1	L_2	L_∞
Velocity	MLS	9.9292×10^{-4}	0.0113	0.0025
	regular	9.3369×10^{-4}	0.0106	0.0028
Velocity gradient	MLS	0.0215	0.0252	0.1169
	regular	0.0209	0.0260	0.1400
Pressure	MLS	0.0286	0.0332	0.0530
	regular	0.0615	0.0790	0.1282

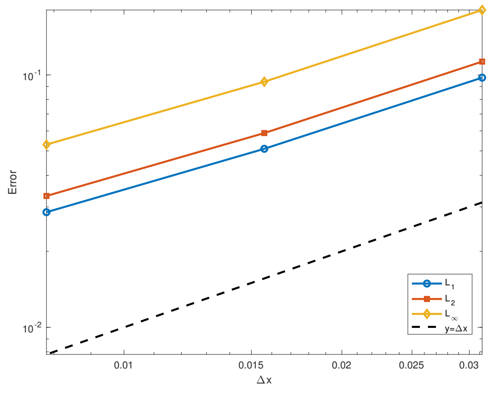
Table A.2: Comparisons of error norms for plane Poiseuille flow in a horizontal channel using regular IB kernel and one-sided MLS reconstruction when $\Delta x = 2^{-7}$.



(a) Velocity



(b) Velocity gradient



(c) Pressure

Figure A.3: Log-log plots of the error norms of velocity, velocity gradient, and pressure for plane Poiseuille flow in a horizontal channel, with $y = \Delta x$ reference line.

flow of the Oldroyd-B fluid are given by

$$
\begin{aligned}
u &= \frac{p_0}{2(\eta_s + \eta_p)}(y - y_0)\left(1 - \frac{(y - y_0)}{H}\right), \\
v &= 0, \\
C_{xx} &= 1 + 2\left(\lambda \frac{p_0}{2(\mu_s + \mu_p)}\left(1 - \frac{2(y - y_0)}{H}\right)\right)^2, \\
C_{yy} &= 1, \\
C_{xy} &= \lambda \frac{p_0}{2(\mu_s + \mu_p)}\left(1 - \frac{2(y - y_0)}{H}\right),
\end{aligned}
\tag{A.7}
$$

where the solvent viscosity $\eta_s = 0.5$, polymer viscosity $\eta_p = 0.5$, and the relaxation time $\lambda = 0.1$ s. Dirichlet boundary conditions are set up for the conformation tensor. In Fig. A.4, we visualize the L_1 and L_2 error norms of the velocity and the components of the conformation tensor $\boldsymbol{C}$ across the entire domain. We observe approximately first-order convergence.

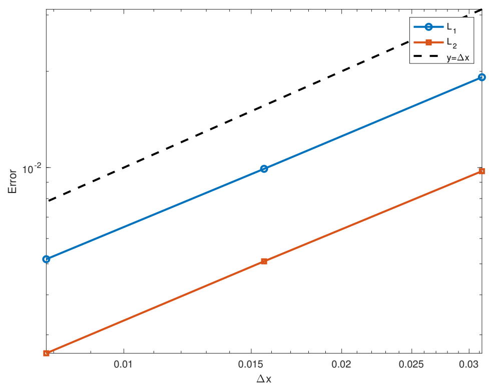
A.2.2 Inclined Channel

Next, we are interested in simulating the plane Poiseuille flow in a channel inclined at $\theta = 22.5$ degrees. We adjust the position of the IB markers so that the Euclidean distance between two nearby points are still twice the Cartesian grid width. This example is more challenging as the channel is not aligned with the background grids. Fig. A.5 provides a schematic view of the flow configuration, where the imposed velocity profile is also rotated so that the fluid is flowing along the channel with a constant pressure gradient. Assume the width of the inlet (vertical distance) is D, the velocity profile is given by

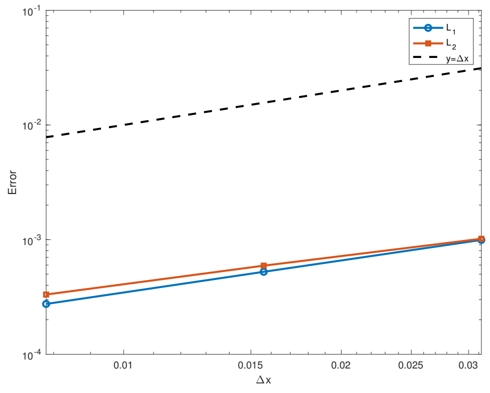
$$
u = \frac{\cos(\theta) p_0 D}{\mu(\frac{L}{\cos(\theta)} + D\tan(\theta))}(-x\sin(\theta) + (y - y_0)\cos(\theta))(1 - (-x\sin(\theta) + (y - y_0)\cos(\theta))), \tag{A.8}
$$

$$
v = \frac{\sin(\theta) p_0 D}{\mu(\frac{L}{\cos(\theta)} + D\tan(\theta))}(-x\sin(\theta) + (y - y_0)\cos(\theta))(1 - (-x\sin(\theta) + (y - y_0)\cos(\theta))). \tag{A.9}
$$

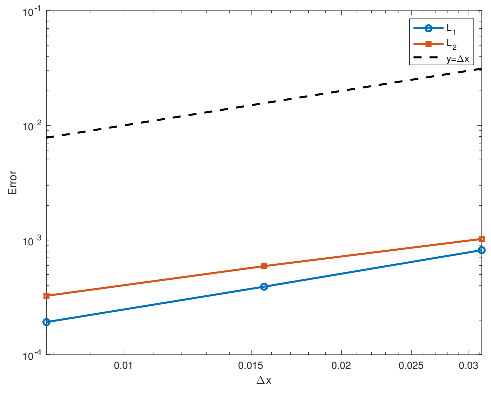
No-slip conditions are imposed at solid walls. The left panel in Fig. A.5 shows the velocity magnitude $\sqrt{u^2 + v^2}$, and the right panel shows the pressure distribution. Both the velocity field and the pressure field are smooth without fluid leak. In Fig. A.6, we show the grid convergence study for the velocity field. Same as the horizontal case, we observe first-order convergence.



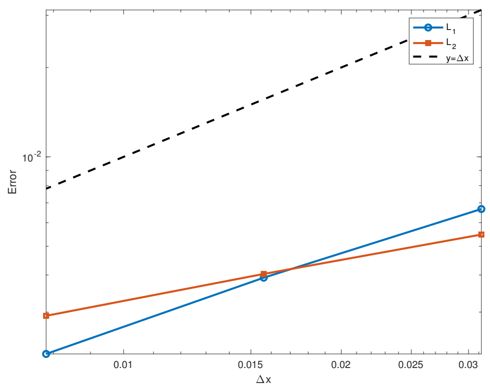
(a) Velocity



(b) C_{xx}

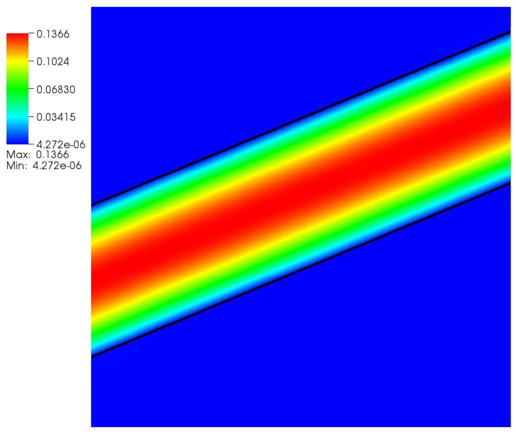


(c) C_{yy}

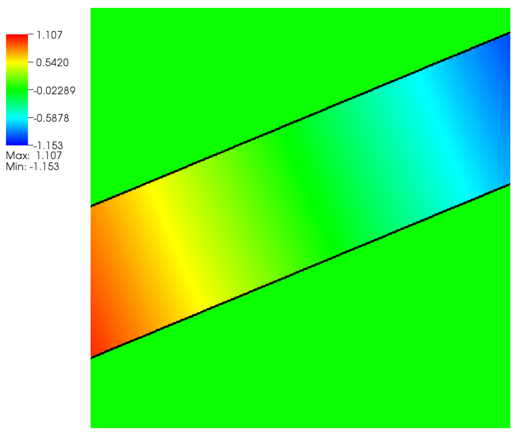


(d) C_{xy}

Figure A.4: Log-log plots of the error norms of velocity and conformation tensor components for Oldroyd-B flow in a horizontal channel, with $y = \Delta x$ reference line.



(a) Velocity magnitude $\sqrt{u^2 + v^2}$



(b) Pressure p

Figure A.5: Visualization of the steady-state flow in a channel inclined at 22.5 degrees with channel walls represented by black dots. We use one-sided MLS to spread and interpolate from outside the channel. The Cartesian grid spacing used in this simulation is $\Delta x = 2^{-7}$.

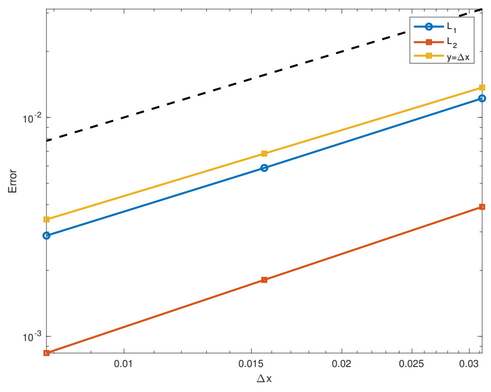


Figure A.6: Log-log plots of the error norms of the velocity field for viscous flow in an inclined channel, with $y = \Delta x$ reference line.

REFERENCES

[1] T. Lord, L. Scelsi, D. Hassell, M. Mackley, J. Embery, D. Auhl, O. Harlen, R. Tenchev, P. Jimack, and M. Walkley, "The matching of 3d rolie-poly viscoelastic numerical simulations with experimental polymer melt flow within a slit and a cross-slot geometry," *Journal of Rheology*, vol. 54, no. 2, pp. 355–373, 2010.

[2] A. Kargar and M. Akbarzade, "Analytic solution of natural convection flow of a non-newtonian fluid between two vertical flat plates using homotopy perturbation method (hpm)," *World Applied Sciences Journal*, vol. 20, no. 11, pp. 1459–1465, 2012.

[3] E. Becker, "Simple non-newtonian fluid flows," in *Advances in applied mechanics*, vol. 20, pp. 177–226, Elsevier, 1980.

[4] S. E. Spagnolie, "Complex fluids in biological systems," *Biological and Medical Physics, Biomedical Engineering*, 2015.

[5] M. R. Knowles and R. C. Boucher, "Mucus clearance as a primary innate defense mechanism for mammalian airways," *The Journal of Clinical Investigation*, vol. 109, no. 5, pp. 571–577, 2002.

[6] P. Zhang, W. R. Summer, G. J. Bagby, and S. Nelson, "Innate immunity and pulmonary host defense," *Immunological Reviews*, vol. 173, pp. 39–51, 2000.

[7] D. C. Venerus and H. C. Öttinger, *A modern course in transport phenomena.* Cambridge University Press, 2018.

[8] J. Cribb, P. Vasquez, P. Moore, S. Norris, S. Shah, M. Forest, and R. Superfine, "Nonlinear signatures in active microbead rheology of entangled polymer solutions," *Journal of Rheology*, vol. 57, no. 4, pp. 1247–1264, 2013.

[9] T. M. Squires and T. G. Mason, "Fluid mechanics of microrheology," *Annual Review of Fluid Mechanics*, vol. 42, pp. 413–438, 2010.

[10] IBAMR, *An adaptive and distributed-memory parallel implementation of the immersed boundary (IB) method.* `https://ibamr.github.io/`.

[11] W. Layton, *Introduction to the numerical analysis of incompressible viscous flows.* SIAM, 2008.

[12] F. Irgens, *Continuum mechanics.* Springer Science & Business Media, 2008.

[13] G. K. Batchelor, *An introduction to fluid dynamics.* Cambridge Mathematical Library, Cambridge University Press, 2000.

[14] M. Abadi, M. F. Serag, and S. Habuchi, "Entangled polymer dynamics beyond reptation," *Nature Communications*, vol. 9, no. 1, pp. 1–12, 2018.

[15] S. Thomas, S. Rafiei, S. Maghsoodlou, and A. Afzali, *Foundations of nanotechnology, volume two: Nanoelements formation and interaction.* CRC Press, 2014.

[16] P. G. de Gennes, "Reptation of a polymer chain in the presence of fixed obstacles," *The Journal of Chemical Physics*, vol. 55, no. 2, pp. 572–579, 1971.

[17] T. Hayat, M. Khan, and M. Ayub, “Exact solutions of flow problems of an oldroyd-b fluid,” *Applied Mathematics and Computation*, vol. 151, no. 1, pp. 105–119, 2004.

[18] A. E. Likhtman and R. S. Graham, “Simple constitutive equation for linear polymer melts derived from molecular theory: Rolie–poly equation,” *Journal of Non-Newtonian Fluid Mechanics*, vol. 114, no. 1, pp. 1–12, 2003.

[19] G. A. Holroyd, S. J. Martin, and R. S. Graham, “Analytic solutions of the rolie poly model in time-dependent shear,” *Journal of Rheology*, vol. 61, no. 5, pp. 859–870, 2017.

[20] Y. Renardy and M. Renardy, “A singular perturbation study of the rolie-poly model,” *Journal of Non-Newtonian Fluid Mechanics*, vol. 262, pp. 52–67, 2018.

[21] R. S. Graham, A. E. Likhtman, T. C. McLeish, and S. T. Milner, “Microscopic theory of linear, entangled polymer chains under rapid deformation including chain stretch and convective constraint release,” *Journal of Rheology*, vol. 47, no. 5, pp. 1171–1200, 2003.

[22] A. Barrett, *An Adaptive Viscoelastic Fluid Solver: Formulation, Verification, and Applications to Fluid-Structure Interaction.* PhD thesis, University of North Carolina at Chapel Hill, 2019.

[23] M. A. Hulsen, R. Fattal, and R. Kupferman, “Flow of viscoelastic fluids past a cylinder at high weissenberg number: stabilized simulations using matrix logarithms,” *Journal of Non-Newtonian Fluid Mechanics*, vol. 127, no. 1, pp. 27–39, 2005.

[24] C. S. Peskin, “The immersed boundary method,” *Acta Numerica*, vol. 11, pp. 479–517, 2002.

[25] SAMRAI, *Structured Adaptive Mesh Refinement Application Infrastructure.* `https://computing.llnl.gov/projects/samrai`.

[26] PETSc, *Portable, Extensible Toolkit for Scientific Computation.* `https://www.mcs.anl.gov/petsc/`.

[27] libMesh. `http://libmesh.github.io`.

[28] HYPRE, *Scalable Linear Solvers and Multigrid Methods.* `https://computing.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods`.

[29] B. E. Griffith and X. Luo, “Hybrid finite difference/finite element immersed boundary method,” *International Journal for Numerical Methods in Biomedical Engineering*, vol. 33, no. 12, p. e2888, 2017.

[30] E. M. Kolahdouz, A. P. S. Bhalla, B. A. Craven, and B. E. Griffith, “An immersed interface method for discrete surfaces,” *Journal of Computational Physics*, vol. 400, p. 108854, 2020.

[31] B. Kallemov, A. Bhalla, B. E. Griffith, and A. Donev, “An immersed boundary method for rigid bodies,” *Communications in Applied Mathematics and Computational Science*, vol. 11, no. 1, pp. 79–141, 2016.

[32] Q. Liu, Y. Liu, C. Jiang, and X. Wang, “Numerical simulation of viscoelastic flows during injection mold filling based on rolie-poly model,” *Journal of Non-Newtonian Fluid Mechanics*, vol. 263, pp. 140–153, 2019.

[33] T. Schneider, L. Goubergrits, U. Kertzscher, and O. Paschereit Christian, “Development of a new 3d shear stress measurement technique using the birefringence,” in *Proceedings of the 14th international symposium on applications of laser techniques to fluid mechanics*, 2008.

[34] M.-C. Lai and C. S. Peskin, “An immersed boundary method with formal second-order accuracy and reduced numerical viscosity,” *Journal of Computational Physics*, vol. 160, no. 2, pp. 705–719, 2000.

[35] Y. Kim and C. S. Peskin, “Penalty immersed boundary method for an elastic boundary with mass,” *Physics of Fluids*, vol. 19, no. 5, p. 053103, 2007.

[36] J. M. Teran and C. S. Peskin, “Tether force constraints in stokes flow by the immersed boundary method on a periodic domain,” *SIAM Journal on Scientific Computing*, vol. 31, no. 5, pp. 3404–3416, 2009.

[37] B. E. Griffith, “An accurate and efficient method for the incompressible navier–stokes equations using the projection method as a preconditioner,” *Journal of Computational Physics*, vol. 228, no. 20, pp. 7565–7595, 2009.

[38] M. Cai, A. Nonaka, J. B. Bell, B. E. Griffith, and A. Donev, “Efficient variable-coefficient finite-volume stokes solvers,” *Communications in Computational Physics*, vol. 16, no. 5, pp. 1263–1297, 2014.

[39] Y. Saad and M. H. Schultz, “Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems,” *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 3, pp. 856–869, 1986.

[40] M. Matinfar, H. Zareamoghaddam, M. Eslami, and M. Saeidy, “Gmres implementations and residual smoothing techniques for solving ill-posed linear systems,” *Computers & Mathematics with Applications*, vol. 63, no. 1, pp. 1–13, 2012.

[41] Y. Saad, “A flexible inner-outer preconditioned gmres algorithm,” *SIAM Journal on Scientific Computing*, vol. 14, no. 2, pp. 461–469, 1993.

[42] A. J. Wathen, “Preconditioning,” *Acta Numerica*, vol. 24, pp. 329–376, 2015.

[43] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, “Block preconditioners based on approximate commutators,” *SIAM Journal on Scientific Computing*, vol. 27, no. 5, pp. 1651–1668, 2006.

[44] M. Benzi and M. A. Olshanskii, “An augmented lagrangian-based approach to the oseen problem,” *SIAM Journal on Scientific Computing*, vol. 28, no. 6, pp. 2095–2113, 2006.

[45] R. D. Guy, B. Philip, and B. E. Griffith, “Geometric multigrid for an implicit-time immersed boundary method,” *Advances in Computational Mathematics*, vol. 41, no. 3, pp. 635–662, 2015.

[46] A. P. S. Bhalla, R. Bale, B. E. Griffith, and N. A. Patankar, “A unified mathematical framework and an adaptive numerical method for fluid–structure interaction with rigid, deforming, and elastic bodies,” *Journal of Computational Physics*, vol. 250, pp. 446–476, 2013.

[47] M. J. Berger and J. Oliger, “Adaptive mesh refinement for hyperbolic partial differential equations,” *Journal of Computational Physics*, vol. 53, no. 3, pp. 484 – 512, 1984.

[48] G. Kanschat, “Divergence-free discontinuous galerkin schemes for the stokes equations and the mac scheme,” *International Journal for Numerical Methods in Fluids*, vol. 56, no. 7, pp. 941–950, 2008.

[49] F. H. Harlow and J. E. Welch, “Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface,” *Physics of Fluids*, vol. 8, no. 12, pp. 2182–2189, 1965.

[50] P. K. Kundu, I. M. Cohen, and D. Dowling, *Fluid mechanics.* Academic Press, 2012.

[51] M. Lipp and R. Helmig, “A locally-refined locally-conservative quadtree finite-volume staggered-grid scheme,” in *Droplet Interactions and Spray Processes*, pp. 149–159, Springer, 2020.

[52] P. B. Bochev and D. Ridzal, “Rehabilitation of the lowest-order raviart–thomas element on quadrilateral grids,” *SIAM Journal on Numerical Analysis*, vol. 47, no. 1, pp. 487–507, 2009.

[53] K. Lipnikov, G. Manzini, and M. Shashkov, “Mimetic finite difference method,” *Journal of Computational Physics*, vol. 257, pp. 1163–1227, 2014.

[54] T. Gerya, D. May, and T. Duretz, “An adaptive staggered grid finite difference method for modeling geodynamic stokes flows with strongly variable viscosity,” *Geochemistry, Geophysics, Geosystems*, vol. 14, no. 4, pp. 1200–1225, 2013.

[55] P. Lancaster and K. Salkauskas, “Surfaces generated by moving least squares methods,” *Mathematics of Computation*, vol. 37, no. 155, pp. 141–158, 1981.