Title
Immersed Boundary Projection Method for Internal Flows

Author
Thoy, Yvonne

Publication Date
2021

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Immersed Boundary Projection Method for Internal Flows

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Aerospace Engineering

by

Yvonne Thoy

2021

ABSTRACT OF THE THESIS

Immersed Boundary Projection Method for Internal Flows

by

Yvonne Thoy
Master of Science in Aerospace Engineering
University of California, Los Angeles, 2021
Professor Jeffrey D. Eldredge, Chair

An extended work on the immersed boundary projection method derived by Taira and Colonius [TK08] and modified by Eldredge [Eld21]. Derivations in this paper closely follow the steps taken in the paper by Eldredge, and the method is applied to internal flows simulations with different boundary conditions: velocity and traction. Masking functions are incorporated into the governing incompressible Navier-Stokes equation by introducing the Heaviside function to allow for the prescription of boundary values at either side of the immersed interface. The discrete system of equations was reformulated into saddle-point form and solved using LU decomposition. Lastly, a prescribed traction term was isolated from the Lagrange multiplier term in the vorticity form of Navier-Stokes so that its value can be assigned for problems with traction (pressure) type boundary conditions. The lid-driven cavity problem and Poiseuille flow were simulated to show application using velocity and pressure conditions respectively. It is shown that the computed results agree well with other studies and exact solutions.

The thesis of Yvonne Thoy is approved.

Kunihiko Taira

Raymond M. Spearrin

Jeffrey D. Eldredge, Committee Chair

University of California, Los Angeles

2021

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest appreciation to my research advisor, Prof. Jeff Eldredge for his patience and continuous support throughout the course of my Master's degree and undergraduate year. His insightful feedback has helped me a lot during the research and the writing of this paper. I would also like to thank my father, who without this would have not been possible. My appreciation extends to the rest of my family and friends for their understanding and encouragement.

CHAPTER 1

Introduction

The immersed boundary (IB) method generally refers to the computational method that solves partial differential equations with grids that do not conform to body surfaces or boundaries. Instead, the immersed surfaces are treated as momentum forcing to satisfy boundary conditions or constraints by utilizing a set of Lagrangian points [MI05, CS02]. This allows flow with moving or deforming complex bodies to be solved effectively since the numerical solution no longer relies on the expensive body-fitted mesh generation. Due to the advantages of easy grid generation and potentially memory/CPU saving [KDH01], the immersed boundary method has become one of the popular methods to solve problems in different flow regimes or physics (e.g. hypersonic [DWZ10], heat transfer [KC04], turbulence [KIH09], etc.). A more comprehensive historical background and development of immersed boundary method can be found in an Annual Review of Fluid Mechanic by Mittal and Iaccarina [MI05].

The original immersed boundary method by Peskin, which we will refer to as IBM, was initially designed to solve hemodynamics problems in cardiac mechanics. The problem of interest was simulated on a Cartesian grid and forcing at the elastic boundary was generated based on Hooke's law at the immersed surface represented by a collection of Lagrangian points [Pes72, Pes77, PM89]. In this method, the continuous singular forcing integral along the boundary is approximated using discrete Dirac Delta Function (DDF) that regularizes the forces of the immersed points to the neighbouring grid points [Pes72]. This has since been extended for application on rigid body flow simulation by using a large spring constant

[BL92]. Other research groups like Goldstein et al. and Saiki and Biringen have also attempted to solve rigid body flows through feedback control [GHS93, SB96]. However, these methods were challenged with numerical instability that imposed a strict limit on the time steps, usually in the order of 10^{-4} due to stiff equations that results from large gain (stiffness) [GHS93].

Although Peskin's IBM has worked well for flexible elastic structures, it is not easy to directly impose boundary conditions or prescribe surface motions. The extended work by Taira and Colonius allows for prescribed motion through the use of Lagrange multiplier that enforces the no-slip condition on the surface which results in a saddle point matrix system that is solved through block LU decomposition and iterated using conjugate gradient method [TK08, TK07]. By treating pressure and boundary forces as Lagrange multipliers to the divergence-free constraints and no-slip condition respectively, they were able to reformulate the incompressible Navier-Stokes equations into block LU matrix form by following Perot's approach of fractional step method [BP93]. This extended method is known as the Immersed Boundary Projection Method (IBPM). Following Taira and Colonius' work, Liska and Colonius have devised a computational efficient IB method for external flows using Lattice Green's Function (LGF) and coupled viscous integration factor with Half Explicit Runge Kutta (IF-HERK) on an unbounded staggered Cartesian grid to solve the discrete momentum ordinary differential equation (ODE) [LC16, LC17]. Detailed formulation of the original IBM and IBPM can be found in the book written by Kajishima and Taira on Computational Fluid Dynamics Incompressible Turbulent Flows [KT17].

So far, it is clear that the primary advantage of IBM is due to the fact that this method only requires minimal geometric information about the structures which greatly simplifies the grid generations process. Since the structure interfaces are represented with immersed points, information about the body's orientation is not needed, and this allows for the use of non-conformal grid structures. Only the locations of the immersed points are required, and the spacing between each sample point has to be approximately equal to the Cartesian grid

spacing of the computational domain in order to prevent penetration of streamlines [TK07]. Aside from application on external or internal flow simulation, the easy implementation and reduction in the computational cost of IBM also serve as attractive features to simulate moving bodies [MI05]. However, as noted in the paper by Eldredge, this simple construction can result in two consequences: the same motion constraint is applied on both sides of the interface and the forces on the interface is the negative sum of the traction forces applied by the fluid [Eld21]. The general construction of IBM or IBPM does not physically differentiate the body interior and exterior region. Instead, the set of immersed points that represents the location of the interface acts as a division between the intentional (flow) and superfluous (usually the body interior) regions. This in particular is not suitable for rigid body rotation or think deformable bodies because the constraint force which is obtained as part of the solution is a mix of traction forces applied by the fluid on both sides of the interface [Eld21]. With the IBM and IBPM, these constraint forces cannot be easily separated for the intentional superfluous region. For deforming bodies, the local traction for the interior elastic region cannot easily be obtained. Similarly, the interior superfluous motion is also found to be inconsistent with rigid body rotation for transient flow over the rigidly rotating body [Eld21]. This issue has since been addressed by different means. To remove the unintended flow solution, Lacis et al. [LTB16] integrated the superfluous region by subtracting the interior dynamic; Wang and Eldredge [WE15] removed the treatment of non-rigid motion and Mittal et al. [MDB08] explicitly prescribed zero motion for the body interior.

In order to establish a general framework for isolating the one-sided force and to avoid nontrivial implementation of surface interpolations as in the methods mentioned above, Eldredge [Eld21] introduced the Heaviside function which is a natural companion to the Dirac delta function into the IBPM to define a discrete masked field. Applying finite difference operators onto this field for prototypical PDE (for example, the Poisson, convection-diffusion, and Navier-Stokes equation) leads to immersed layers term that contains the description of surface jumps. This in turn makes setting boundary conditions on each side of the interface

an easy process and also prevented the formation of superfluous flow inside the body. The specification of the interior region from exterior flow will serve as a base reference for this paper. In this paper, we will adopt this method to simulate internal flows and introduce modifications to prescribe traction-type boundary conditions.

The derivations and discretization of the governing equations are shown in Chapter 2. Starting with a general overview of the method in section 2.1, necessary elaborations grid elements, operators and the underlying masking functions are described prior to their usage in the discretization process in section 2.3. The identities shown in the section 2.1.3 are especially important as they are pertinent to the discussion on immersed single and double layers that resulted from masked field discretization. In which, allows the direct prescription of boundary values at both sides of the interface ($f_b^{\pm}$ for scalar value and $\mathbf{u}_b^{\pm}$ for vectors). Those identities are then incorporated into the governing equations in section 2.2 to introduce the masked version of incompressible Navier-Stokes and continuity. Lastly, a brief section on LU decomposition is included as a solution to the saddle-point problem. Since this paper also intends to include traction (pressure) type boundary conditions, the extended derivations are built directly from the discretized vorticity form of Navier-Stokes. The extended version will work similar to the original IBPM, except with a different boundary condition.

Chapter 3 will show the results for internal flow simulations. Two different problems will be used to verify our method. For internal flow with velocity boundary condition, we simulated the lid driven cavity problem and compared its result with Ghia et al [GGS82]. In this method, the surface normal vector is assigned to every immersed point. Through the comparison with Ghia's result, we noticed the consequence of sharp corners due to ambiguous direction. When immersed points are specified at corners, majority of the flow remains similar to Ghia's except the corner recirculation zone is absent. We address this by avoiding assignment of immersed point at sharp corners. Note that this can work because the spacing between two immersed points are required to be approximately equal to the grid size ($\Delta s = \Delta x$). As for traction (pressure) type boundary condition, we simulated

Poiseuille flow and compared the numerical result with exact solution. Note that, this is not a complete internal flow because it is modeled using two parallel plates in the IBPM domain. It is, however, interesting to show that by forcing pressure on end of the channel, the IBPM generated a flow that circulate around the plate.

The result sections will not cover extensive error analysis as it has already been established in the reference paper by Eldredge [Eld21] that the method is first order accurate at the boundary which is the same order of accuracy obtained in the original IBM and IBPM. However, we can show that the numerical result obtained in both cases matches closely with the result predicted using other method (coupled strongly implicit and multigrid methods (CSI-MG) by Ghia [GGS82]) and the exact solution.

CHAPTER 2

Method

2.1 Model

In this chapter, we will walk through the important identities required to discretize the incompressible Navier-Stokes equation by providing a layout on the staggered grid set up and also the definition of each operators. In this paper, the discretized system is built on a staggered Cartersian grid with uniform spacing (i.e. $\Delta x = \Delta y$ for 2-dimensional grid and $\Delta x = \Delta y = \Delta z$ for 3-dimensional). In actual simulation, the grid domain will always be finite. However, since this method will not require boundary conditions defined at grid boundary, the operators can be treated as notionally infinite [Eld21]. Second order finite difference scheme will be used to approximate the operators such as the discrete Heaviside function and interpolation on the staggered grid points. Finally, once all the discrete operators are defined, it is then applied onto the vorticity form of the incompressible Navier-Stokes equations to get the discrete expression. The final form of the system in combination with its constraints and boundary conditions can be formulated into a saddle point matrix which is solved using block LU decomposition in this paper. Lastly, for traction type boundary condition, the surface traction term which initially includes the total computed traction is dissected to include a prescribed source at the boundary.

2.1.1 Staggered Grids and Operators

Uniform staggered Cartesian grids can be visualized as having a set of grids stacked on top of another shifted grids. The advantage of using such grid is that it introduces certain operator properties in the IBPM to solve the entire discretized governing equations efficiently. In particular, the equality of the discrete divergence operator to the transpose of the negative discrete gradient operator ($D = -G^T$) and the symmetry of the Laplace operator ($L = L^T$). Moreover, discretizing the Pressure Poisson Equation (PPE) on staggered grid can help to prevent checkerboard instability which happens when all flow variables are defined on the same grid points. If regular Cartesian grid was used, the odd and even numbers of pressure values can be decoupled and cause spatial oscillations [KT17].

Figure 2.1 shows the illustration for the staggered grids. Each square cell is bordered by red lines with length $\Delta x = \Delta y$, and its cell elements are all colored circles. The green circles are cell centers, denoted with c. Blue circles are cell edges, ε, and yellow circles are cell faces f. The subscript x, y at the cell faces denotes the positive direction for vector quantities. Yellow lines are located at half cell $\frac{\Delta x}{2}$ and $\frac{\Delta y}{2}$. Based on Figure 2.1, the discrete divergence D, gradient G, Laplace L and curl C operators are each approximated with second order central difference scheme. The divergence and gradient operators are used to transform data between cell centers and faces, while the curl operator transform data at cell edges to cell faces.

$$D : f \rightarrow c$$

$$G : c \rightarrow f$$

$$C : \varepsilon \rightarrow f$$

Aside from f, c, ε that represents the grid spaces, there also a tensor space d that is made up of cell centers and edges.

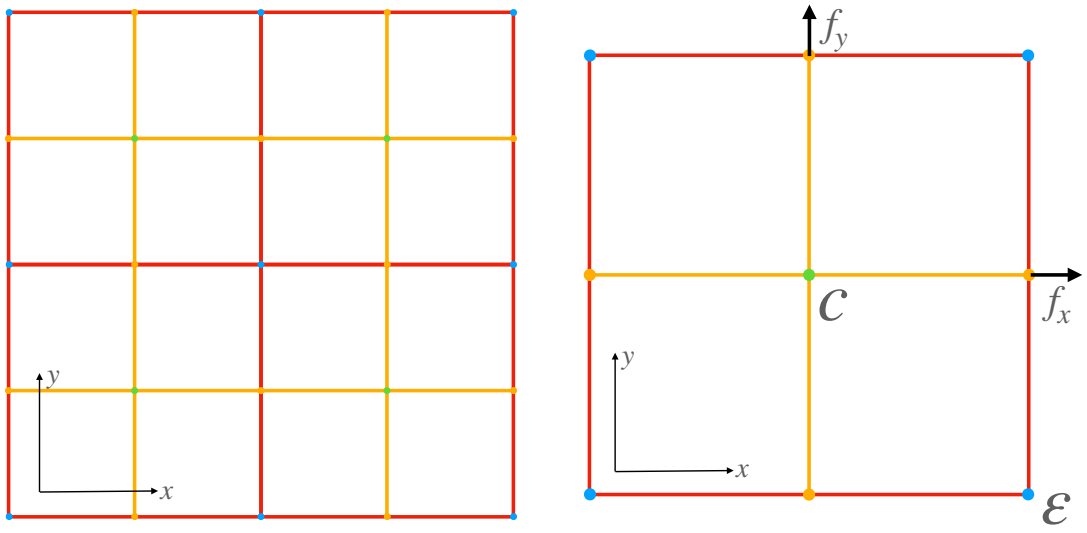

Figure 2.1: Staggered grid cells and the labeling of cell elements.

$$d = \begin{bmatrix} c & \varepsilon_z \\ \varepsilon_z & c \end{bmatrix} \tag{2.1}$$

where ε_z is a vector pointing towards a vertex in the z-direction in 3-dimensional visualization of the grid cell.

2.1.2 Interpolation and Regularization operators

Since the IBM is a method that simulates flow over a fixed Eulerian grid and represents immersed surfaces with Lagrangian grids which intends to separate the flow field solution into intentional and superfluous regions, there has to be a way to communicate flow information between these two grids. This is done through the interpolation and regularization operators. The former is introduced to interpolate velocity field values from the Eulerian grid (staggered grid) onto the Lagrangian grid (immersed points), while the latter is used to pass the boundary force effect from the Lagrangian grid to the neighbouring Eulerian grid

points.

Both interpolation I and regularization R operators can be derived by taking the convolution of the variables with the delta function

$$\begin{aligned} u(\xi) &= \int_x u(x)\delta(x - \xi(s))dx \\ f(x) &= \int_s F(s)\delta(\xi(s) - x)ds \end{aligned} \tag{2.2}$$

where x is an element of the Eulerian grid and s is an element of the Lagrangian grid. The integral can then be discretized to show $Iu(x) = u(\xi(s))$ and $f(x) = RF(\xi(s))$. The equality $R = I^T$ can be shown by absorbing the offset of the scaling ratio into unknown boundary force [TK07]. The detailed derivation and elaboration on these two operators can be found in various papers. Taira and colonius described the derivations of the operators [TK07], and Eldredge cataloged the detailed distinction of the operators identities/characteristics when applied on different grid spaces in the paper's Appendix [Eld21].

2.1.3 Heaviside Function and Masking Function

As mentioned in previous chapter, the work in this paper is based on the extension made my Eldredge [Eld21] on the IBPM. In the previous work, the Heaviside function serves an important features that introduces the masking function which allow for boundary conditions to be prescribed on both side of the immersed surface.

The Heaviside function H is defined as

$$H(y) = \begin{cases} 1, & \text{if } y > 0 \\ 1/2, & \text{if } y = 0 \\ 0, & \text{if } y < 0 \end{cases} \tag{2.3}$$

which means $H(-y) = 1 - H(-y)$. Applying this function onto an indicator function χ

that defines a space where the $\chi < 0$ at the interior, $\chi > 0$ at the exterior and $\chi = 0$ on the surface, it is clear that $H(\chi)$ or $H(-\chi)$ represents the mask function that has value of 1 on side of the surface and 0 on the other. The masked field expression can be shown as

$$\bar{f} \equiv f^{+}H(\chi) + f^{-}H(-\chi) \tag{2.4}$$

where f^{+} and f^{-} are field quantity on either side of the interface. Taking the temporal or spatial derivatives of Eq 2.4 will result in singular integral terms that describe jumps at the interface. This allow the formulation of the governing partial differential equation to be expressed in a single equation that contains jump terms. In discrete form, Eq 2.4 can be expressed as $\bar{f} \equiv f^{+}H_c^{+} + f^{-}H_c^{-}$ where the superscript $\pm$ on H denotes the interior/exterior region and the subscript c means value at cell centers. A full derivation of Heaviside function identities and its relation to the Dirac Delta function in the paper by Eldredge [Eld21]. In this section, we will focus on a few important identities.

To develop masking operators from the masked field equation (Eq 2.4) above, we note that $H'(y) = \delta(y)$. Taking the gradient of $H(\chi)$ results in

$$\nabla H(\pm\chi(x)) = \pm H'(\chi(x))\nabla\chi = \pm\delta(\chi(x))\mathbf{n}(x) \tag{2.5}$$

where $\mathbf{n}$ is the normal vector. For the interior region, discretizing Eq 2.5, we arrive with

$$GH_c^{-} = -R_f\mathbf{n} \tag{2.6}$$

R_f is the regularization operator that regularizes the surface normal vectors $\mathbf{n}$ onto the cell faces. Note that $R_f n = \delta(\chi)\mathbf{n}$. To solve for H_c^{-}, we can take the divergence of Eq 2.6 and invert the Laplacian operator through Lattice Green's Function.

$$
\begin{aligned}
DGH_c^- &= L_c H_c^- = -DR_f\mathbf{n} \\
H_c^- &= -L_c^{-1} DR_f\mathbf{n}
\end{aligned}
\tag{2.7}
$$

The exterior masking can then be obtained ($H_c^+ = 1 - H_c^-$). It can also be shown that $H_c^- = -DL_f^{-1}R_f\mathbf{n}$ due to the commutative properties between divergence and Laplacian operators for staggered grid. The additional negative sign roots from the negative of the lattice Green's function ($\nabla^2 G(x) = -\delta(x)$). To get vector data from the masked field, the cell center data can be interpolated to obtain data at the cell faces.

$$
H_f^{\pm} = I_c^f H_c^{\pm} \tag{2.8}
$$

where I_c^f means interpolation from cell center to cell face. The superscript f on the interpolation or regularization should not be confused with the masked scalar field value f as the mask field variable are always marked with either $+, -$ or a vinculum. Finally, we will show the three important identities of the masked field: the gradient, divergence and Laplacian of the masked field.

$$
\begin{aligned}
G\bar{f} &= \bar{G}f + I_c^f(f^+ - f^-)\cdot R_f\mathbf{n} \\
\bar{G}f &\equiv (Gf^+)\cdot H_f^+ + (Gf^-)\cdot H_f^-
\end{aligned}
\tag{2.9}
$$

$$
\begin{aligned}
D\bar{u} &= \bar{D}u + I_f^c((u^+ - u^-)\cdot R_f\mathbf{n}) \\
\bar{D}u &\equiv (Du^+)\cdot H_c^+ + (Du^-)\cdot H_c^-
\end{aligned}
\tag{2.10}
$$

where $\bar{u}$ is the masked vector data field. I_f^c is the interpolation from cell face to cell center which differs from I_c^f. Eq 2.9 and Eq 2.10 are both derived using a series of discrete product rule. For interest in detailed derivations of the product rule, please refer to Appendix B of [Eld21]. Lastly, combining the two identities, we get the Laplacian of the masked vector $\bar{f}$.

$$L_c\bar{f} = \bar{L_c}f + I_f^c(G(f^+ - f^-)\cdot R_f n) + D(I_c^f(f^+ - f^-)\cdot R_f \mathbf{n})$$
$$\bar{L_c}f \equiv (L_c f^+)\cdot H_c^+ + (L_c f^-)\cdot H_c^- \tag{2.11}$$

2.1.4 Lattice Green's Function

Another useful identity that will be utilized throughout this paper comes from the lattice Green's Function, it is used to solve the general Poisson equation.

For the masked field defined in Eq 2.4, taking the divergence of the gradient of such field results in a general Poisson equation shown below.

$$\nabla^2\bar{f} = \bar{\nabla}f + (\nabla f^+ - \nabla f^-)\cdot\delta(\chi)\mathbf{n} + \nabla\cdot((f^+f^-)\delta(\chi)\mathbf{n})$$
$$\bar{\nabla}f = H(\chi)\nabla^2 f^+ + H(-\chi)\nabla^2 f^- = \bar{q} \tag{2.12}$$

where each $f^{\pm}$ satisfy the a Poisson equation, and $\bar{q}$ is an integrable function. Eq 2.12 is derived using the continuous form of the identities shown in Eq 2.9 and Eq 2.10. Note that the discrete form can be seen in Eq 2.11. In fact, the second and third terms on the right hand side represent the single and double layer potentials with strength defines by the difference in f^+ and f^- that accounts for the effects of the jump at the interface. The strength of the potential layer is usually a known value. The solution to this equation can be found by using the properties of lattice Green's function with $g(x)$ being the solution to Eq 2.13.

$$\nabla^2 g(x) = -\delta(x) \tag{2.13}$$

$$\bar{f}(y) = \int_R g(x-y)\bar{q}(x)\,dx - \int_R g(x-y)\mathbf{n}\cdot(\nabla f^+ - \nabla f^-)\delta(\chi)\,dx$$
$$- \nabla_y\cdot\int_R g(x-y)(f^+ - f^-)\delta(\chi)\mathbf{n}\,dx \tag{2.14}$$

The detailed derivation and properties of Eq 2.14 can be found in the Appendix of the paper by Eldredge [Eld21].

It can be seen that the discrete equations are grid based definition, which means the immersed points (interface) values cannot be easily prescribed since the immersed location does not usually coincide with the Cartesian grid points. In order to conveniently set values for f^+ and f^- at the interface, we utilize the interpolation and regularization functions to devise equations that describe the strength of the immersive single and double layer [Eld21]. Let $\mathbf{s}$ and $\mathbf{d}$ be the strength of the single and double layer respectively.

$$\begin{aligned} R_c\mathbf{s} &= I_f^c((\mathbf{u}^+ - \mathbf{u}^-) \cdot R_f\mathbf{n}) \\ R_f(d \cdot \mathbf{n}) &= I_c^f((f^+ - f^-) \cdot R_f\mathbf{n}) \end{aligned} \tag{2.15}$$

At the boundary, $\mathbf{u}^\pm = \mathbf{u}_b^\pm$ and $f^\pm = f_b^\pm$, where $\mathbf{u}^\pm$ is a vector field and $f^\pm$ is a scalar field. Hence,

$$\begin{aligned} \mathbf{s} &= (\mathbf{u}_b^+ - \mathbf{u}_b^-) \cdot \mathbf{n} \\ d &= f_b^+ - f_b^- \end{aligned} \tag{2.16}$$

2.2 Governing Equations

In this paper, we intend to solve the incompressible Navier-Stokes equation for internal flows. The familiar equation is given as

$$\begin{aligned} \rho\frac{\partial \mathbf{v}}{\partial t} + \rho\mathbf{v} \cdot \nabla\mathbf{v} &= \nabla \cdot \mathbf{S} \\ \mathbf{S} &\equiv -p\mathbf{I} + \mu(\nabla\mathbf{v} + \nabla^T\mathbf{v}) \\ \nabla \cdot \mathbf{v} &= 0 \end{aligned} \tag{2.17}$$

where $\mathbf{I}$ is the identity tensor. To clearly derive the masked version of Eq 2.17, we first look at the treatment of the convective-diffusive equation with masked field operators. Let

us rewrite the conventional convective-diffusive equation with the property of the indicator function $\nabla\chi = n$ for $|\nabla\chi| = 1$ on the surface $\chi = 0$.

$$\begin{aligned} \frac{\partial\bar{\varphi}}{\partial t} + \bar{\mathbf{v}}\cdot\nabla\bar{\varphi} &= -\nabla\cdot\bar{\mathbf{Q}} + \bar{q} + \sigma\delta(\chi) \\ \sigma &= (\mathbf{Q}^+ - \mathbf{Q}^-)\cdot\mathbf{n} + (\varphi^+ - \varphi^-)(\bar{\mathbf{v}} - \dot{X})\cdot\mathbf{n} \end{aligned} \tag{2.18}$$

where $\mathbf{v}$ is the fluid velocity, q is the volumetric forcing, $\mathbf{Q}$ is the flux vector, $\dot{X}$ is the surface velocity and σ is the combination of the convective and diffusive flux. $\mathbf{Q}$ has a generic expression of $-k\nabla\varphi$, and utilizing the properties Eq 2.4 by taking the gradient to get the jump terms;

$$\nabla\bar{f} = H(\chi)\nabla f^+ + H(-\chi)\nabla f^- + (f^+ - f^-)\delta(\chi)\mathbf{n} \tag{2.19}$$

the masked flux vector $\bar{\mathbf{Q}}$ can be expressed as

$$\bar{\mathbf{Q}} = -k\nabla\varphi + k(\varphi^+ - \varphi^-)\delta(\chi)\mathbf{n} \tag{2.20}$$

Combining Eq 2.18 and Eq 2.20, the expression for masked convective-diffusion then becomes

$$\frac{\partial\bar{\varphi}}{\partial t} + \bar{\mathbf{v}}\cdot\nabla\bar{\varphi} = k\nabla^2\varphi + \bar{q} + \sigma\delta(\chi) - \nabla\cdot((\varphi^+ - \varphi^-)\delta(\chi)\mathbf{n}) \tag{2.21}$$

Applying Eq 2.21 to the Navier-Stokes equation, we can get the masked version of Eq 2.17.

$$\begin{aligned} \rho(\frac{\partial\bar{\mathbf{v}}}{\partial t} + \bar{\mathbf{v}}\cdot\nabla\mathbf{v}) &= -\nabla\bar{p} + \mu\nabla^2\bar{\mathbf{v}} - \sigma\delta(\chi) - \nabla\cdot(\mathbf{\Sigma}\delta(\chi)) \\ \nabla\cdot\bar{\mathbf{v}} &= (\bar{\mathbf{v}}^+ - \bar{\mathbf{v}}^-)\cdot\delta(\chi)\mathbf{n} \end{aligned} \tag{2.22}$$

σ is a combination of surface force on both sides of the interface and jump in momentum fluxes. $\mathbf{\Sigma}$ is the viscous surface tensor.

$$\begin{aligned}\sigma &= (\mathbf{S}^+ - \mathbf{S}^-)\cdot\mathbf{n} - \rho(v^+ - v^-)(\bar{v} - \dot{X})\cdot\mathbf{n} \\ \mathbf{\Sigma} &= \mu((\mathbf{v}^+ - \mathbf{v}^-)\mathbf{n} + \mathbf{n}(\mathbf{v}^+ - \mathbf{v}^-))\end{aligned} \tag{2.23}$$

From the expression above, it is clear that by applying masking operators onto the Navier-Stokes equation introduces source terms to enforce continuity. In both convective-diffusive and Navier-Stokes equations, if φ^+, φ^- or $\mathbf{v}^+, \mathbf{v}^-$ is known at the interface (i.e. Dirichlet boundary conditions), σ becomes the Lagrange multiplier for the no-slip constraint $\delta^T(\chi)\bar{\varphi} = \frac{1}{2}(\varphi_b^+ + \varphi_b^-)$. Subscript b denotes boundary, and φ is replaced by $\mathbf{v}$ for the incompressible Navier-Stokes system.

For simulating internal flows, $\mathbf{v}_b+$ is set to 0 and $\mathbf{v}_b^-$ is set to $\mathbf{v}_b$. This will ensure zero motion in the superfluous region at all times. The constraint has now reduces to $\delta^T(\chi)\bar{\varphi} = \frac{1}{2}(\varphi_b)$, and the surface traction can be found using Eq 2.23.

$$\mathbf{S}^-\cdot\mathbf{n} = \sigma - \frac{1}{2}\rho\mathbf{v}_b\mathbf{v}_b\cdot\mathbf{n} \tag{2.24}$$

Eq 2.24 can be easily solve since the 2nd term (momentum flux) is a known quantity and σ is part of the overall solution.

As mentioned, we intend to use the vorticity form of the Navier-Stokes equation. We start by taking the curl of Eq 2.22.

$$\begin{aligned}\rho(\frac{\partial\bar{\omega}_t}{\partial t} - \nabla\times(\bar{v}\times\bar{\omega}_t)) &= \mu\nabla^2\bar{\omega}_t - \nabla\times(\sigma\delta(\chi)) - \nabla\times(\nabla\cdot(\mathbf{\Sigma}\delta(\chi))) \\ \bar{\omega}_t \equiv \nabla\times\bar{\mathbf{v}} &= \bar{\omega} + \delta(\chi)\mathbf{n}\times(v^+ - v^-)\end{aligned} \tag{2.25}$$

$\bar{\omega}_t$ is the augmented vorticity field that includes the masked vorticity field and the strength of the vortex sheet. Similar with the previous equations, the jump terms is evident in the vorticity field, and σ remains as the Lagrange multiplier to enforce the no-slip condition.

Helmholtz decomposition is used get velocity values from the vorticity solution because an additional scalar potential field $\bar{\varphi}$ is required as the curl of streamfunction $\bar{\psi}$ no longer

gives the complete velocity description due to the jump in normal velocities.

$$\bar{\mathbf{v}} = \nabla \times \bar{\psi} + \nabla\bar{\varphi} + \mathbf{U}_\infty \tag{2.26}$$

$\mathbf{U}_\infty$ is the uniform velocity. $\bar{\varphi}$ and $\bar{\psi}$ can each be isolated from Eq 2.26 by taking the divergence and curl to get Poisson equation. It should be noted that $\bar{\varphi}$ and $\bar{\psi}$ are required to match at the interface to ensure non-singular velocity field, and they should also goes to zero at infinity. The solution to these equation can be solve using the lattice green's function for poisson equation described in Eq 2.13 and Eq 2.14.

$$\begin{aligned} \nabla^2\bar{\varphi} &= \nabla \cdot \bar{\mathbf{v}} = (\mathbf{v}^+ - \mathbf{v}^-) \cdot \delta(\chi)\mathbf{n} \\ \nabla^2\bar{\psi} &= -\bar{\omega}_t = -\bar{\omega} + (\mathbf{v}^+ - \mathbf{v}^-) \times \delta(\chi)\mathbf{n} \end{aligned} \tag{2.27}$$

The discretization and solution to Eqs 2.17 will be shown in the next section. Similar process will be apply to the vorticity form to formulate the system of equations that is implemented in the code. These equations will serve as a base reference for the lid driven cavity flow results shown in Chapter 3. Finally, modification is made to Eq 2.17 in Chapter 2.1 section 2.4 to allow traction type boundary condition.

2.3 Discretization

In this section, the governing equations will be discretized using identities shown above. Starting with the general Navier-Stokes equation, the semi-discrete form is

$$\begin{aligned} \frac{d\bar{v}}{dt} + N(\bar{\mathbf{v}}) &= -G\bar{p} + \nu L_f\bar{\mathbf{v}} - R_f\tau - D_dR_d\mathbf{S} \\ D\bar{\mathbf{v}} &= (R_c)^T(\mathbf{d}\cdot\mathbf{n}) \\ (R_f)^T\bar{\mathbf{v}} &= \bar{\mathbf{v}}_b \end{aligned} \tag{2.28}$$

where

$$\begin{aligned} \mathbf{d} &= \mathbf{v}_b^+ + \mathbf{v}_b^- \\ \bar{\mathbf{v}}_b &= \frac{1}{2}(\mathbf{v}_b^+ + \mathbf{v}_b^-) \\ \mathbf{S} &= \nu(\mathbf{d} \otimes \mathbf{n} + \mathbf{n} \otimes \mathbf{d}) \end{aligned} \tag{2.29}$$

τ is the surface traction vector, $\mathbf{S}$ is the discrete viscous surface tensor from Eq 2.23, $\otimes$ is the tensor product, $N(\bar{\mathbf{v}})$ is the convective acceleration, and $\bar{p}$ is the grid pressure which is treated as a Lagrange multiplier similar to τ to enforce continuity. By defining a linear diffusion operator $L_f^\nu \equiv \frac{d}{dt} - \nu L_f$ and reshuffle the right hand side terms of Eq 2.28, the system of equations can be formulated into the matrix shown below.

$$\begin{bmatrix} L_f^\nu & -G & -R_f \\ D & 0 & 0 \\ R_f^T & 0 & 0 \end{bmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{p} \\ \tau \end{pmatrix} = \begin{bmatrix} -N(\bar{\mathbf{v}} - D_d R_d \mathbf{S} \\ (R_c)^T(\mathbf{d} \cdot \mathbf{n}) \\ \bar{\mathbf{v}}_b \end{bmatrix} \tag{2.30}$$

Eq 2.30 is in fact a saddle-point problem. There are various ways to approach such problem, and we will solve it with block LU decomposition in the next section (2.5).

Applying similar treatment to the vorticity form of Navier-Stokes equation shown in Eq 2.25, we get the discrete system.

$$\begin{aligned} \frac{d\bar{\omega}_t}{dt} + C^T R_f \tau &= -C^T N(\bar{\mathbf{v}}) + \nu L_\varepsilon \bar{\omega}_t - C^T D_d R_d \mathbf{S} \\ (R_f)^T \bar{\mathbf{v}} &= \bar{\mathbf{v}}_b \end{aligned} \tag{2.31}$$

where

$$\bar{\omega}_t = C^T \bar{\mathbf{v}} = \bar{\omega} + R_\varepsilon(\mathbf{n} \times \mathbf{d}) \tag{2.32}$$

As derived above (Eq 2.26),from the result of the vorticity field, the discrete Helmholtz decomposition can be used to extract velocity field data.

$$\begin{aligned} \bar{\mathbf{v}} &= C\bar{s} + G\bar{f} + \mathbf{U}_\infty \\ L_\varepsilon \bar{s} &= -\bar{\omega}_t \\ L_c \bar{f} &= R_c(\mathbf{d}\cdot\mathbf{n}) \end{aligned} \tag{2.33}$$

By substituting Eq 2.33 into Eq 2.31, the system of equation for the voriticity expression can then be written into saddle-point form using similar identity of the linear diffusion operator $L_\varepsilon^\nu \equiv \frac{d}{dt} - \nu L_\varepsilon$.

$$\begin{bmatrix} L_\varepsilon^\nu & C^T R_f \\ -R_f^T C L_\varepsilon^{-1} & 0 \end{bmatrix} \begin{pmatrix} \bar{\omega}_t \\ \tau \end{pmatrix} = \begin{bmatrix} -C^T N(\bar{\mathbf{v}}) - C^T D_d R_d \mathbf{S} \\ \bar{\mathbf{v}}_b - R_f^T G L_c^{-1} R_c(\mathbf{d}\cdot\mathbf{n}) - R_f^T \mathbf{U}_\infty \end{bmatrix} \tag{2.34}$$

2.4 Traction Boundary Modification

In this section, we will expand on the discrete vorticity form of Navier-Stokes equation. Let's revisit Eq 2.31. The original surface traction consist of the jumps in surface traction force and momentum flux across the interface. In order to prescribe a traction boundary condition, we will add an "additional" $C^T R\tau_p$ term on the left hand side of the expression. The expression is still the same except we have isolated the prescribed traction value from the solution. The subscript p stands for prescribe. This is especially useful when pressure type boundary condition is required as the pressure data is contained within the traction solution together with viscous shear stress.

$$\frac{d\bar{\omega}_t}{dt} + C^T R_f \tau + C^T R\tau_p = -C^T N(\bar{\mathbf{v}}) + \nu L_\varepsilon \bar{\omega}_t - C^T D_d R_d \mathbf{S} \tag{2.35}$$

Since $C^T R\tau_p$ is a known value, the left hand side of the saddle-point system shown in Eq 2.34 will remains the same, and the"additional" term will be moved to the other side. In the code implementation, the prescribed traction is treated as a vector line source.

$$\begin{bmatrix} L_\varepsilon^\nu & C^T R_f \\ -R_f^T C L_\varepsilon^{-1} & 0 \end{bmatrix} \begin{pmatrix} \bar{\omega_t} \\ \tau \end{pmatrix} = \begin{bmatrix} -C^T N(\bar{\mathbf{v}}) - C^T D_d R_d \mathbf{S} - C^T R \tau_p \\ \bar{\mathbf{v}}_b - R_f^T G L_c^{-1} R_c (\mathbf{d} \cdot \mathbf{n}) - R_f^T \mathbf{U}_\infty \end{bmatrix} \tag{2.36}$$

As mentioned, the semi-discrete equation in the saddle-point system will be solved using LU decomposition, which also allow us to follow the time integration method devised by Liska and Colonius [LC16] that uses second order integration factor with half explicit Runge-Kutta (IF-HERK) scheme.

2.5 Saddle Point Matrix Solver

After discretizing the governing equations, we arrive with a saddle-point system (also known as the Karush-Kuhn-Tucker "KKT" system). This familiar expression is often a consequence of the discretization of partial differential equation with the enforcement of constraints for fluid problems, generally exist to minimize the energy of the system [BGL05]. Due to the vast application of saddle-point system, there have been extensive studies on methods to solve the mathematical problem. In particular, Benzi, Golub and Liesen has provided a very detailed review on numerical solutions for solving saddle-point problems [BGL05]. In this paper, the system is solved with block LU decomposition, elaborated in 2.5.1.

2.5.1 Block LU Decomposition

This section is meant to show the solution to the general saddle-point problem so it can be applied to any system derived in this paper, either Eq 2.30, Eq 2.34 or Eq 2.36. Note that the derivations in this section closely follows the Appendix in the paper by Beckers and Eldredge [BE21].

Let's consider a general system with semi-definite matrix A (as in L_ε^ν or L_f^ν mentioned in previous sections) that can be factorised into lower and upper matrices.

$$\begin{bmatrix} A & B_1^T \\ B_2^T & -C \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{r_1} \\ \mathbf{r_2} \end{pmatrix} \tag{2.37}$$

$$\begin{bmatrix} A & B_1^T \\ B_2^T & -C \end{bmatrix} = \begin{bmatrix} A & 0 \\ B_2 & S \end{bmatrix} \begin{bmatrix} I & A^{-1}B_1^T \\ 0 & I \end{bmatrix} \tag{2.38}$$

where I is the identity matrix, and S is the Schur Compliment ($S \equiv -C - B_2A^{-1}B_1^T$). Using this properties, we can easily solve for $\mathbf{x}, \mathbf{y}$ by introducing an intermediate steps. We start by substituting Eq 2.38 in Eq 2.37.

$$\begin{bmatrix} A & 0 \\ B_2 & S \end{bmatrix} \begin{bmatrix} I & A^{-1}B_1^T \\ 0 & I \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{r_1} \\ \mathbf{r_2} \end{pmatrix} \tag{2.39}$$

It is clear that Eq 2.39 can be split into two equations using intermediate terms $\mathbf{x}^*$ and $\mathbf{y}^*$.

$$\begin{bmatrix} I & A^{-1}B_1^T \\ 0 & I \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{x}^* \\ \mathbf{y}^* \end{pmatrix} \tag{2.40}$$

$$\begin{bmatrix} A & 0 \\ B_2 & S \end{bmatrix} \begin{pmatrix} \mathbf{x}^* \\ \mathbf{y}^* \end{pmatrix} = \begin{pmatrix} \mathbf{r_1} \\ \mathbf{r_2} \end{pmatrix} \tag{2.41}$$

Based on Eq 2.40 and Eq 2.41, a series of algorithm can be perform to get the inverse of the original matrix and the solution $\mathbf{x}, \mathbf{y}$.

$$\begin{aligned} \mathbf{x} &= \mathbf{x}^* - A^{-1}B_1^T\mathbf{y} \\ \mathbf{y} &= \mathbf{y}^* \\ \mathbf{x}^* &= A^{-1}\mathbf{r_1} \\ \mathbf{y}^* &= S^{-1}[\mathbf{r_2} - B_2\mathbf{x}^*] \end{aligned} \tag{2.42}$$

Resulting in

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{bmatrix} A^{-1} + A^{-1}B_1^T S^{-1} B_2 A^{-1} & -A^{-1}B_1^T S^{-}1 \\ -S^{-1}B_2 A^{-1} & S^{-}1 \end{bmatrix} \begin{pmatrix} \mathbf{r_1} \\ \mathbf{r_2} \end{pmatrix} \tag{2.43}$$

If we treat $\mathbf{x}$ as the solution vector, and $\mathbf{y}$ as the constraint force, Eq 2.43 is the same form as Eq 2.30, Eq 2.34 and Eq 2.36.

CHAPTER 3

Result

In this chapter, we will show the numerical results for lid driven cavity in section 3.1 and Poiseuille flow in section 3.2. For lid driven cavity, its computed result will be compared to Ghia's results that were generated using CSI-MG method. As for Poiseuille flow, the results are compared to exact solution. All results shown in this chapter are obtained through a Julia language code that can be found on GitHub **JuliaIBPM** page. Jupyter notebooks that shows sample outline for both simulations are available under **ViscousFlow.jl** repository [TE21].

3.1 Lid Driven Cavity

The lid driven cavity was simulated for Reynolds number RE of 100 and 400 in order to compare with Ghia's result. For both cases, the results were simulated using 208×208 grid cell with $\Delta x = \Delta y = 0.005$ for domain range $y = x = [-0.505 : 0.505]$. The square boundary is created with length of 1 from -0.5 to 0.5 with immersed point spacing of $\Delta s = 1.5\Delta x$. In order to avoid instability, the domain grid has to be at least 1 Δx larger than the intentional region because the system has to recognise the zero motion setting in the superfluous region, which is why the grid domain is set to -0.505 to 0.505. The time step $\Delta t = 0.00125$ is computed with the minimum between Courant–Friedrichs–Lewy (CFL) condition of 0.5 and Fourier number of 0.5 to ensure stability. All walls were set to zero motion ($\mathbf{v} = 0$) except for the top walls which is moving at velocity of 1.

The steady state solutions are shown in section 3.1.1. It can be seen from the streamlines and vorticity flow field visualizations in Figure 3.1 and 3.2, the flow is totally contained within boundary. No spurious flow were generated in the superfluous region. There are, of course, some leakage near the boundary in the superfluous region. However, this is due to the regularization operator that smear the effect of the constraint force. Moreover, the smoothing of the Heaviside function is also a contributor to the boundary error. For this reason, the increment of error due to relaxation of grid spacing for cell close to the immersed interface is also the greatest.

Side by side comparisons with Ghia's results show good agreement between both methods. Since Ghia et al. have also made comparisons with other studies, we can confidently verify our method using their numerical data. Even though Figure 3.3 and 3.6 shows streamlines that are almost identical, a problem did arise during the generation of corner circulation zone. Due to the ambiguity in the direction of normal vector at sharp corners, the counter rotating vortices did not show until the immersed points at the corners are removed while maintaining $\Delta s \approx \Delta x$. In Figure 3.5 and 3.8, we can clearly see the matching velocities value at geometric center except for one outlier data in v-velocity plot for RE 400. Since our domain goes from -0.5 to 0.5, when comparing to Ghia's data, we have to shift our domain by 0.5. Although we show results for grids that are almost twice as densed as Ghia's, the result doesn't deviate much when we use 130×130 grid cells. All visual comparisons are shown in section 3.1.2.

3.1.1 Streamlines patterns and Vorticity contour for $RE = 100$ and $RE = 400$

Below are the flow visualization for both cases. It can be seen that from the figures shown, as Reynolds number increases, the center of the vortex started to move towards the center of the cavity which can be observed more easily from the streamline patterns. Note that the vorticity contours are both plotted using Ghia's value in Table III [GGS82].

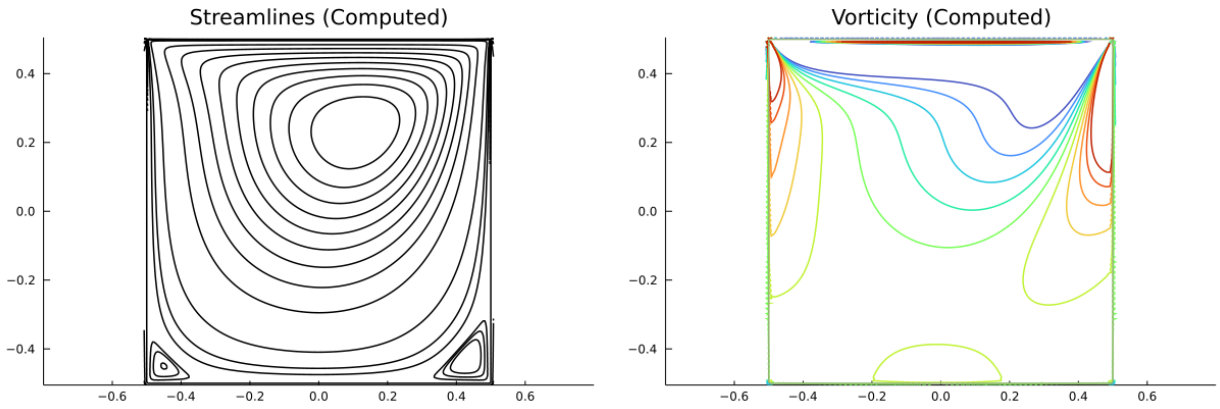

Figure 3.1: Streamline and vorticity contour for $RE = 100$ on $[208 \times 208]$ grid with $\Delta x = 0.005$.

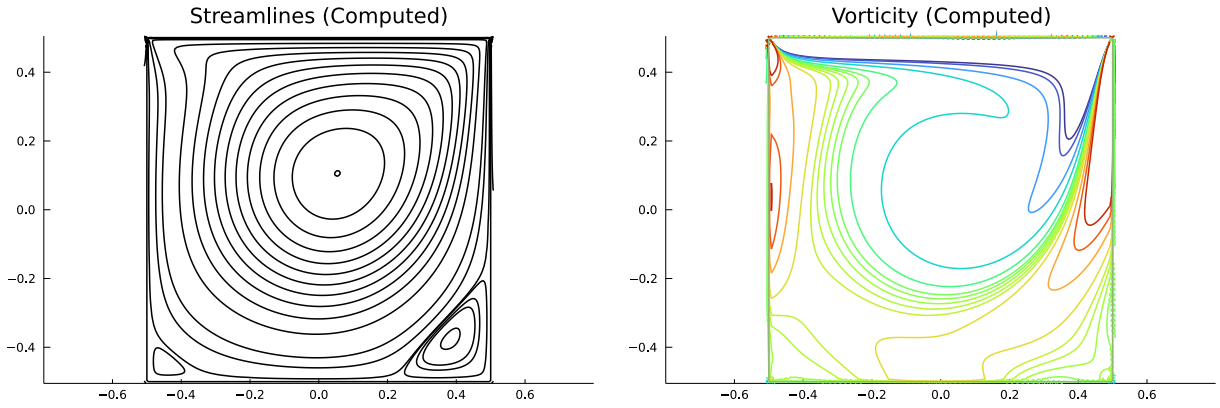

Figure 3.2: Streamline and vorticity contour for $RE = 400$ on $[208 \times 208]$ grid with $\Delta x = 0.005$.

3.1.2 Plot comparison with Ghia's result

3.1.2.1 RE = 100

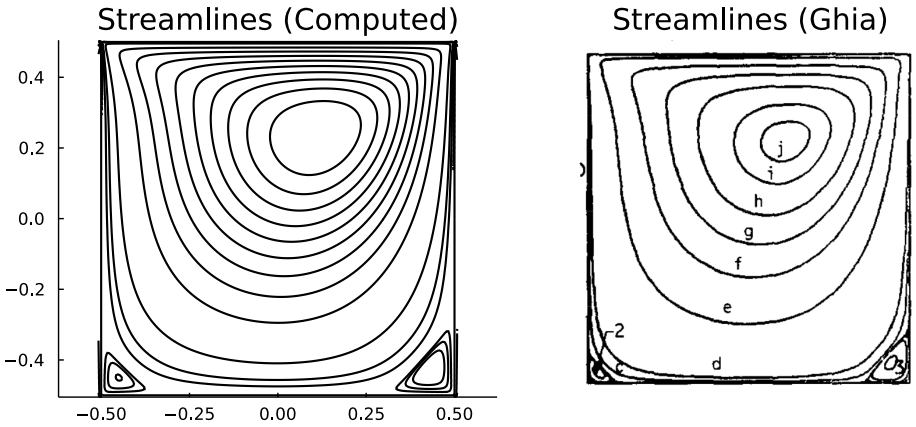

Figure 3.3: Streamlines computed for $RE = 100$ on $[208 \times 208]$ grid using IBPM compare with Ghia's result computed on $[129 \times 129]$ using CSI-MG method.

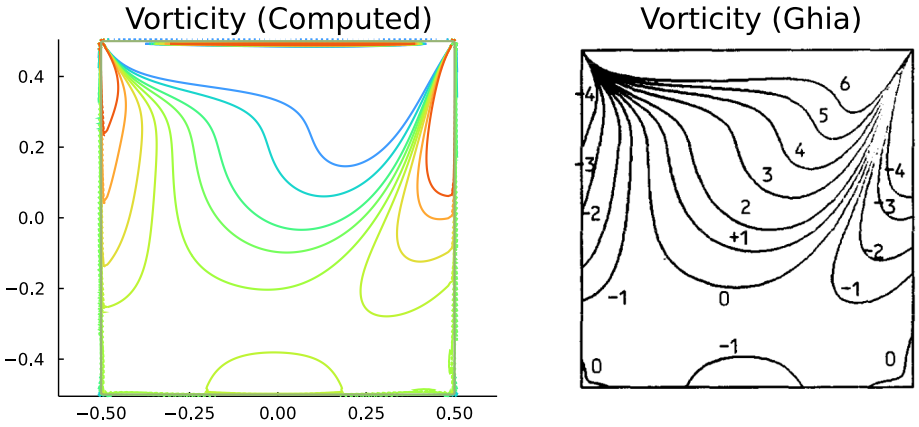

Figure 3.4: Vorticity contours computed for $RE = 100$ on $[208 \times 208]$ grid using IBPM compare with Ghia's result computed on $[129 \times 129]$ using CSI-MG method.

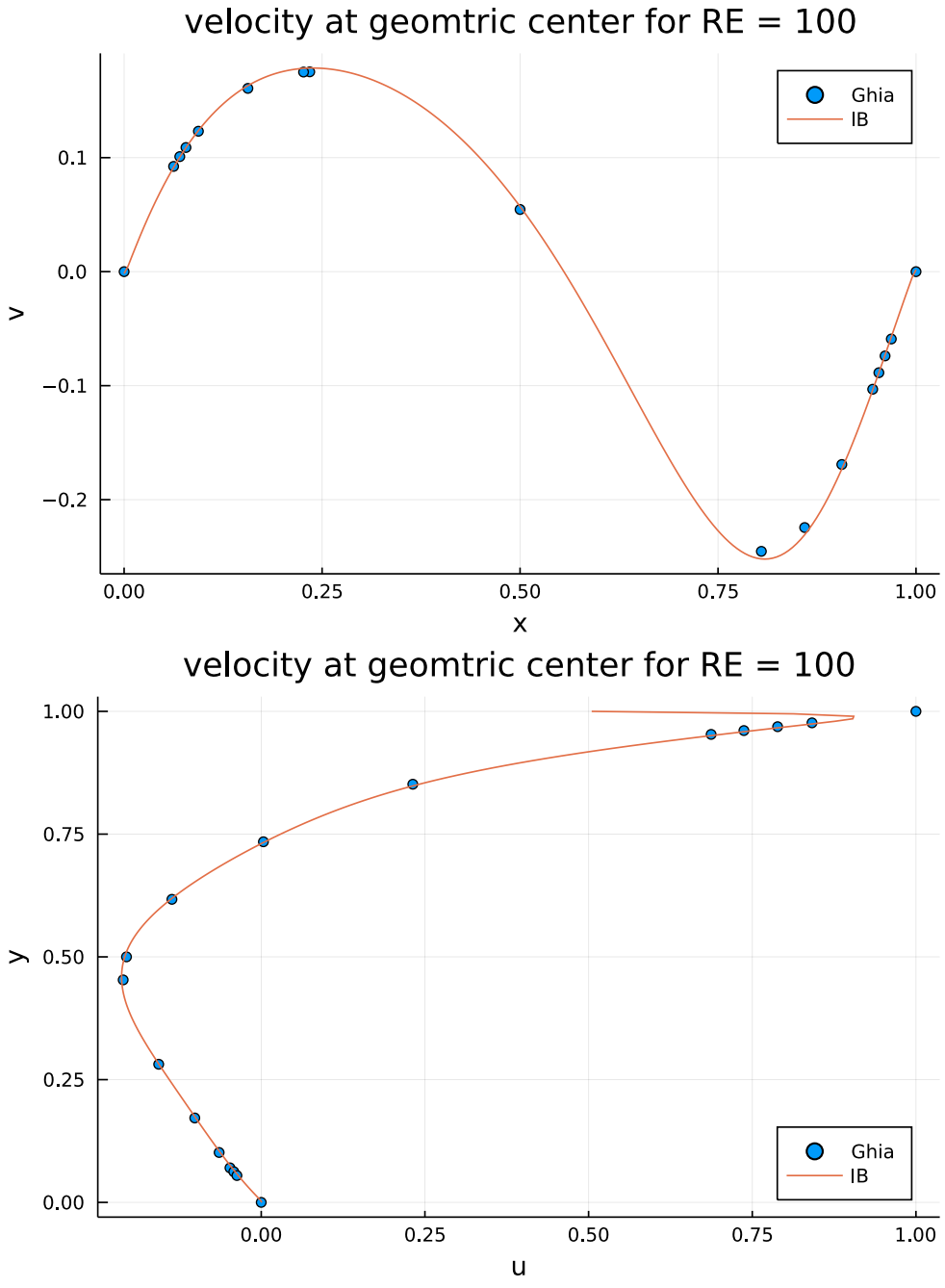

Figure 3.5: u and v velocity data comparison along the geometric center for $RE = 100$.

3.1.2.2 RE = 400

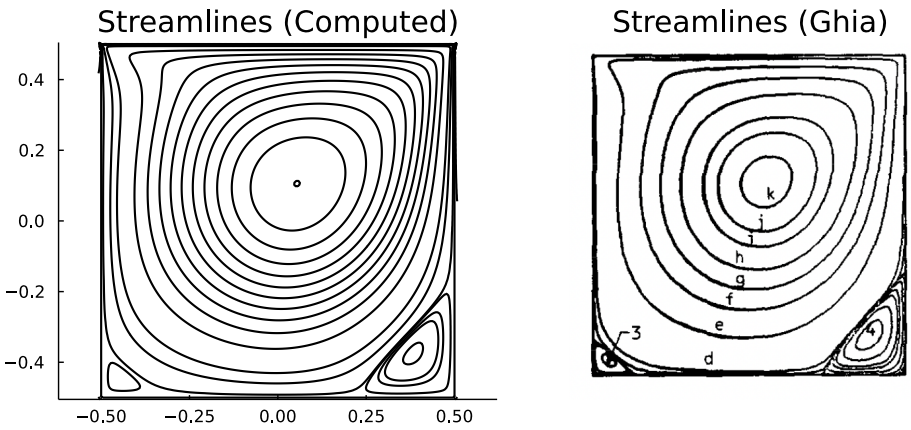

Figure 3.6: Streamlines computed for $RE = 400$ on $[208 \times 208]$ grid using IBPM compare with Ghia's result computed on $[129 \times 129]$ using CSI-MG method.

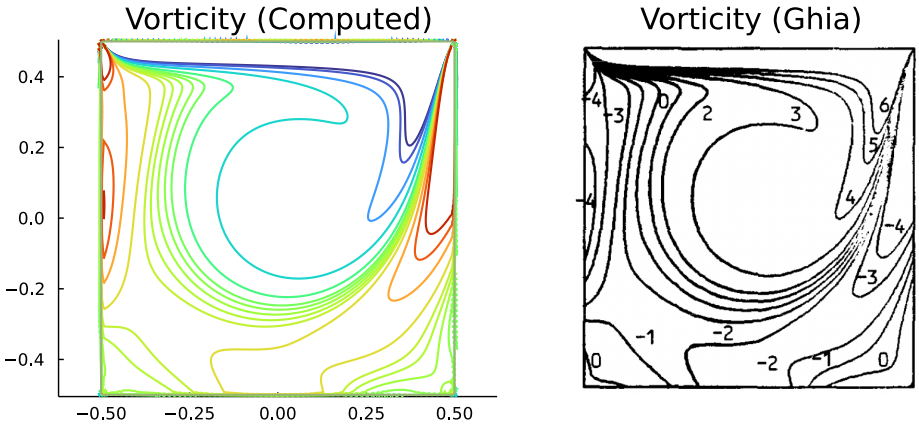

Figure 3.7: Vorticity contours computed for $RE = 100$ on $[208 \times 208]$ grid using IBPM compare with Ghia's result computed on $[129 \times 129]$ using CSI-MG method.

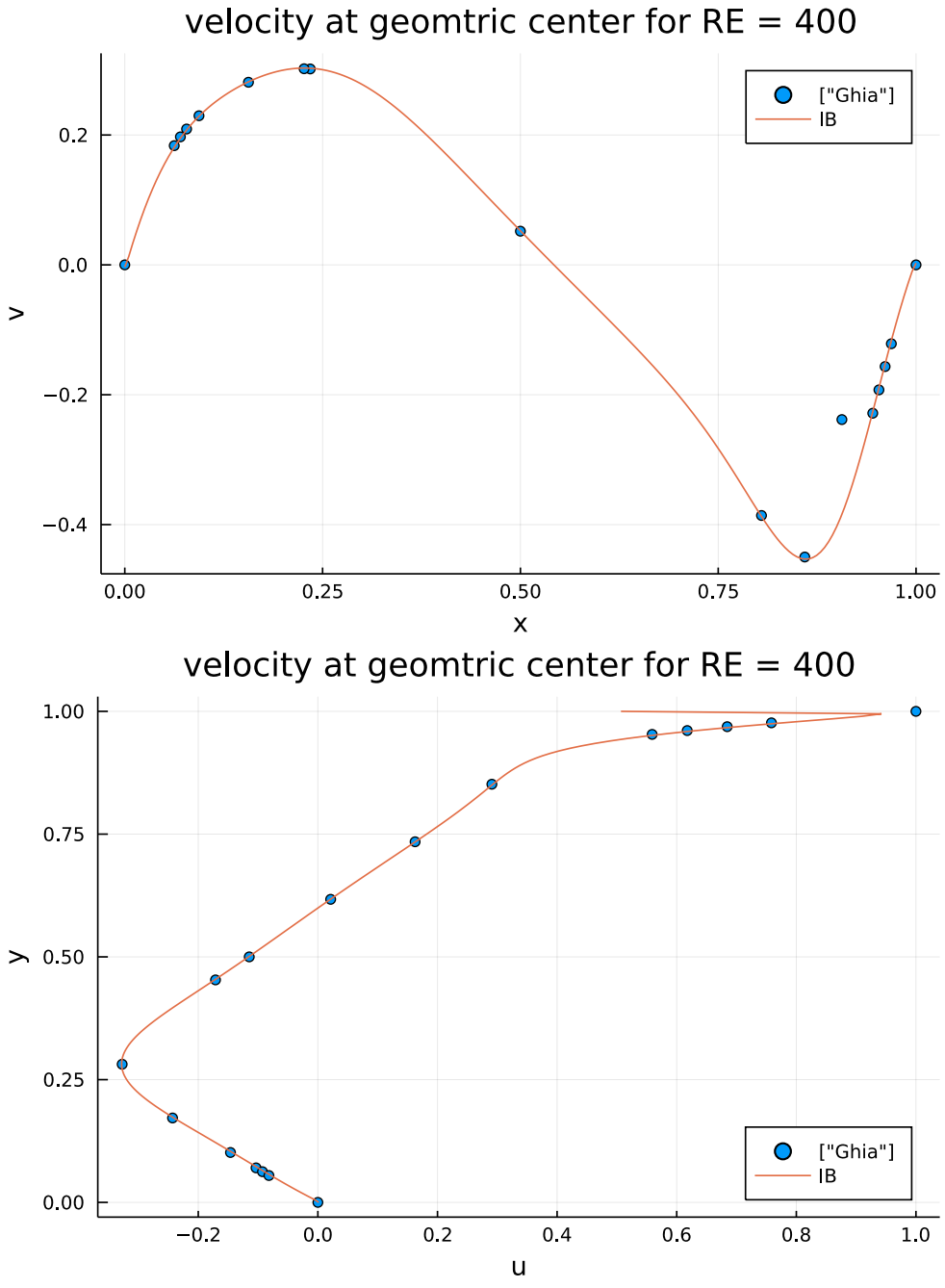

Figure 3.8: u and v velocity data comparison along the geometric center for $RE = 400$.

3.2 Poiseuille flow with traction type boundary conditon

The Poiseuille flow simulation is used to test for the modification made on the IBPM that prescribe traction (pressure) boundary condition instead of velocity. In this simulation, two plates with length $L = 30$ are immersed into a domain of size $[x = [-16, 16], y = [-5, 5]]$. The gap between the plates were maintained at $h = 2$, and the results were simulated over a $[416x128]$ grid with 500 immersed point in total (250 on each plate). The computational Reynolds number is set to $RE_c = 50$ with CFL $= 0.5$ and grid size of $\Delta x = \Delta y = 0.08$. Resulting in $\Delta t = 0.04$. The immersed point spacing is the same as the lid driven cavity case $\Delta s = 1.5\Delta x$. At the inlet and outlet, traction is treated as a vectored line source. Two sets of points are first arranged in a straight line at the two boundaries, and then a traction strength is prescribed onto the defined location. This is known as the "line source strength". We prescribe -1 at the inlet as the pressure is in the opposite direction of unit normal, and 0 at the outlet. From Poiseuille flow analysis, a linear pressure gradient along channel is expected. The computed steady state velocity is compared with theoretical solution in Figure 3.9. Note that the velocity profile is sampled at $x = 0$ which is far downstream of the predicted entrance length of $L_en = 0.1$. The magnitude of the entrance length might seem insignificant, but it can actually influence the shape of the profile. The profile shape matches the exact profile more nicely when velocity value were sampled further from the entrance length. Figure 3.9 is a plot of the velocities normalized by its maximum. Meaning, they should peak at 1, and the plot is only comparing the profile shapes. The theoretical U/U_{max} is given in Eq 3.2.

$$U(y) = \frac{1}{2}\frac{h}{\mu}\frac{dp}{dx}(\frac{y}{h} - (\frac{y}{h})^2) \tag{3.1}$$

$$\frac{U(y)}{U_{max}} = 4(\frac{y}{h} - (\frac{y}{h})^2) \tag{3.2}$$

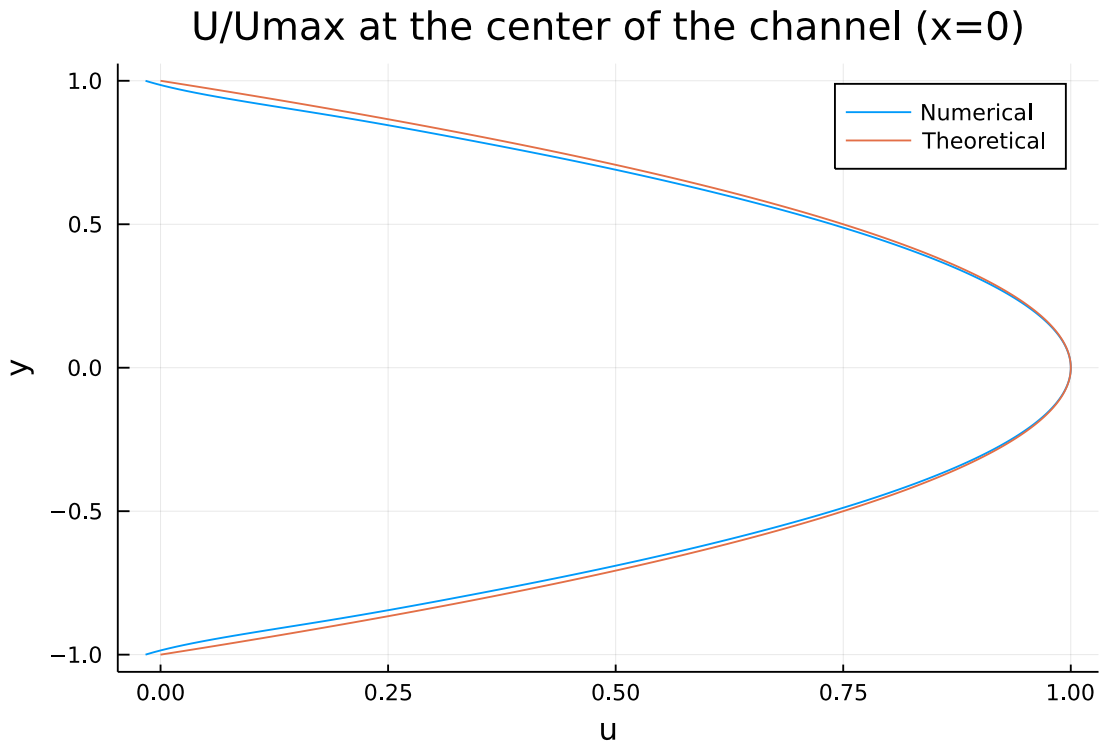

Figure 3.9: Comparison of velocity profiles normalized by its maximum value at the center line for $RE = 50$.

To compare the magnitude of velocity profiles, we normalize the profile with the average velocity U_{avg} instead of the maximum. The theoretical equation for U/U_{avg} is given in Eq 3.3. The results shown in Figure 3.10 shows the discrepancies between both profile at center line. The max error $U_{exact} - U_{numerical}$ which peak at $y = 0$ is -0.0374. On a scale that max at 1.5, 0.0374 is not significant, and the error is likely due to slope of the numerical dp/dx which is not 1/30. Instead, the computed dp/dx is approximately 0.65/30 based on the pressure profile shown in Figure 3.11. Nonetheless, the overall magnitude of the velocity profiles still agrees well with each other.

$$\frac{U(y)}{U_{max}} = 6(\frac{y}{h} - (\frac{y}{h})^2) \tag{3.3}$$

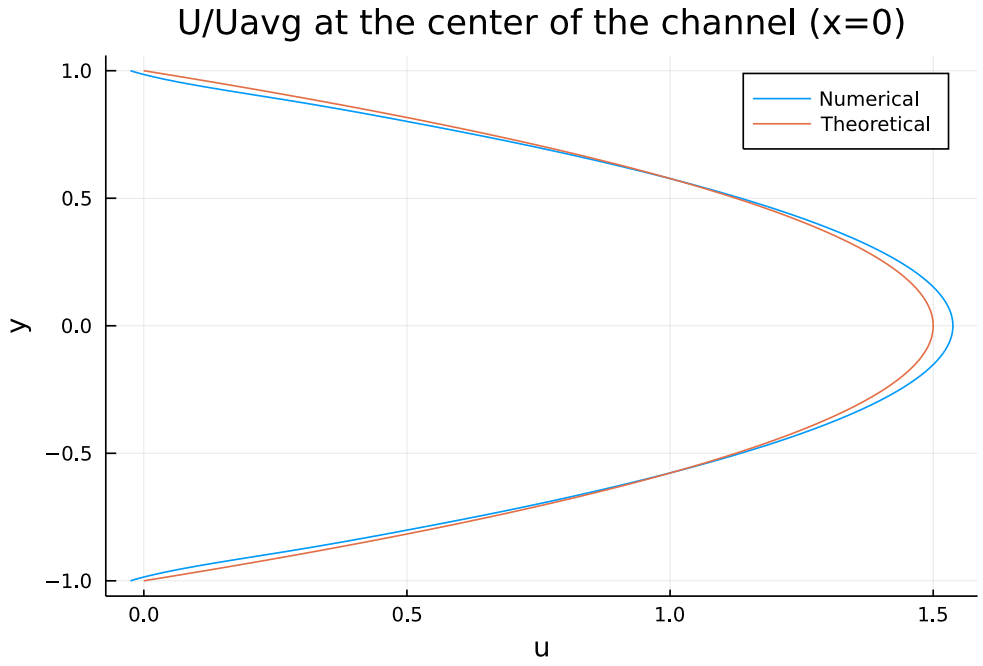

Figure 3.10: Comparison of velocity profiles normalized by its average value at the center line for $RE = 50$.

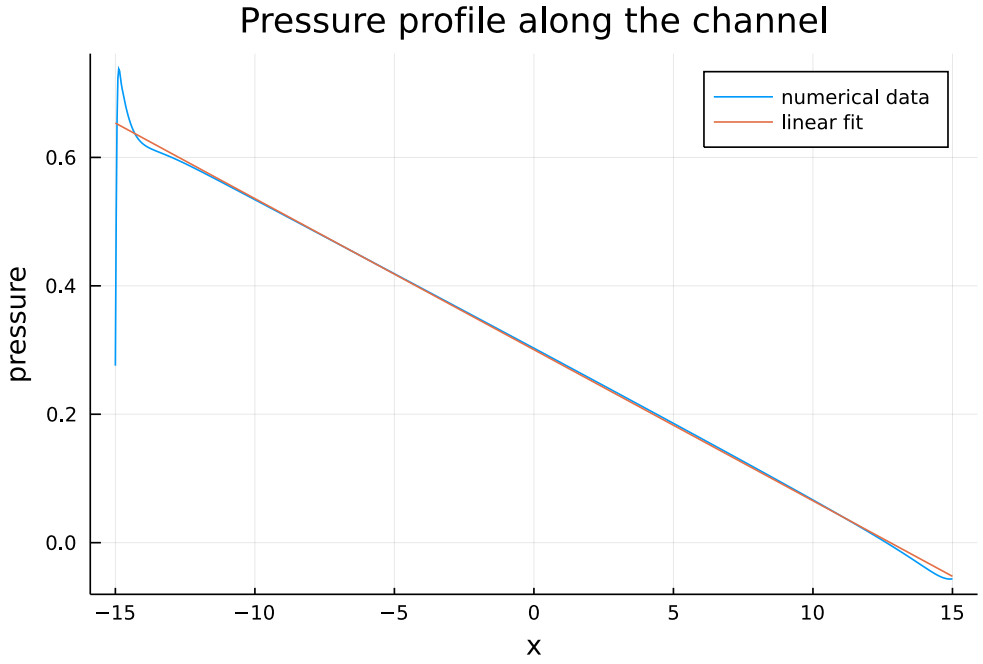

Figure 3.11: Pressure gradient along the channel, linear fitted to get slope value of 0.0235.

Although the numerical result matches with theoretical solution, Figure 3.12 and 3.13 clearly shows that the simulation is not totally internal. There are flow circulating back from the outlet into the inlet in superfluous region. If we limit our focus to region inside the channel, this is indeed a true Poiseuille flow. However, when we observe the entire computational domain, this is actually a circulating flow around the plates. Although it is not shown in this paper, one can apply similar treatment can be done to the traction term as we did to the velocity field by introducing $\tau_b^{\pm}$ to suppress spurious flow outside the channel.

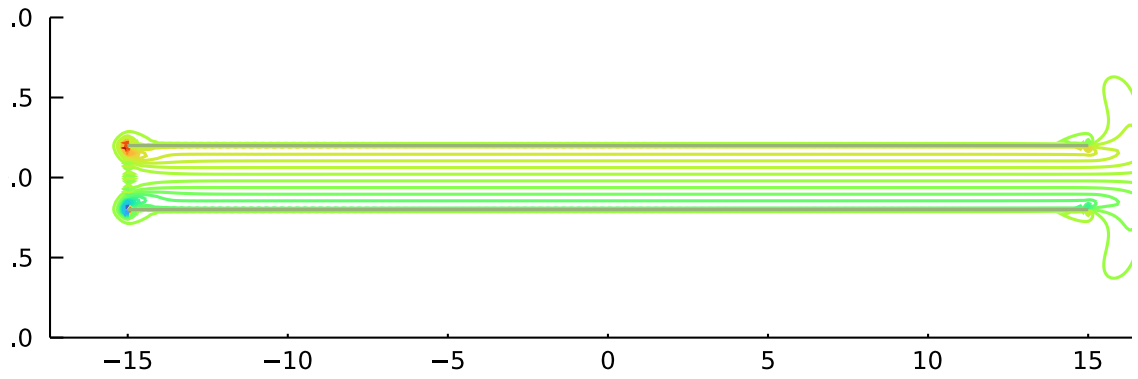

Figure 3.12: Vorticity contour of Poiseuille flow prescribed with traction boundary condition for $RE = 50$.

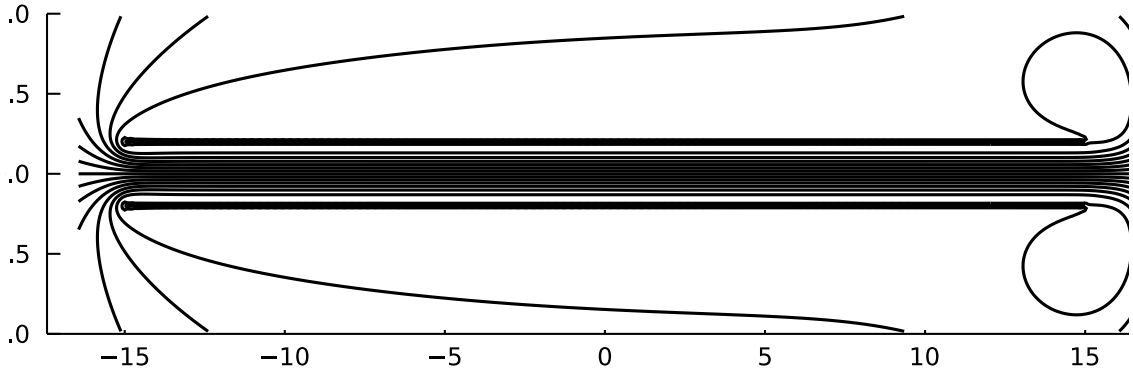

Figure 3.13: Streamline pattern of Poiseuille flow prescribed with traction boundary condition for $RE = 50$.

CHAPTER 4

Conclusion

In this paper, we have extended the work of Eldredge [Eld21] on IBPM to simulate internal flows by utilizing the masking function to ensure zero motion on the superfluous region. We were also able to modify the discrete governing equations to include traction type boundary conditions. The method starts by introducing a set of important identities derived from Heaviside function and incorporate them into the the continuous incompressible Navier-Stokes equation to include jump terms at the interface. The system of equations is then semi-discretized and reorganized into a saddle-point problem to be solved using block LU decomposition. IF-HERK scheme were used to iterate the solution through time. As mentioned, we intend to use the vorticity form of Navier-Stokes equation (as implement in the ViscousFlow package), so the same process is repeated for the curl of Navier-Stokes equation. From the vorticity field, velocity data were extracted using Helmholtz decomposition. This method was used to simulate the lid driven cavity flow, and the computed results were compared with the results by Ghia et al. [GGS82] for Reynolds number of 100 and 400. The comparison shows good agreement. No spurious flow were generated in the superfluous (external) region. A little smearing effect were observed at the boundary, and it is possibly due to the approximation by the regularization operator and the smoothing effect of the Heaviside function which limits the boudary error to first order accurate even though the operators were discretized using second order central differencing. For Poiseuille flow simulation, the discrete vorticity-Navier-Stokes equation were expanded to introduce the prescribe traction term. The comparison with theoretical solution generally shows good agreement with minor error caused by the inexact pressure gradient. It is also worth mentioning that

the streamline visualization shows that the flow field is not limited to the internal region within the channel. It is in fact a flow circulation around the plates. This is due to the prescribed traction is the total forcing from both sides of the interface. However, this can be solved following similar steps as we did for the velocity by allowing the prescription of $\tau_b^{\pm}$.

In this paper, we have explored the enforcement of different Dirichlet boundary conditions for internal flows which can certainly be expanded to accept other conditions like Neumann boundary condition or even time varying forcing. The hope of establish such framework is that it would serve as a convenient tool to solve fluid problems or potentially problems in other discipline like acoustic for example. It would be interesting to see more applications of this method on flow in complex geometries with moving parts (i.e. impeller, deforming body, etc).

REFERENCES

[BE21] Diederik Beckers and Jeff D. Eldredge. "Planar potential flow on Cartesian grids." *J. Fluid Mech.*, 2021. Unpublished.

[BGL05] M. Benzi, G. H. Golub, and J. Liesen. "Numerical solution of saddle point problems." *Acta Numerica*, p. 14:1–137., 2005.

[BL92] R.P. Beyer and R.J. LeVeque. "Analysis of a one-dimensional model for the immersed boundary method." *SIAM J. Numer. Anal.*, p. 29:332–364, 1992.

[BP93] B.Perot. "An analysis of the fractional step method." *J. Comput. Phys.*, pp. 108:51–58, 1993.

[CS02] C.S.Peskin. "The immersed boundary method." *Cambridge University Press*, p. 479–517, 2002.

[DWZ10] Le Duan, Xiaowen Wang, and Xiaolin Zhong. "A high-order cut-cell method for numerical simulation of hypersonic boundary-layer instability with surface roughness." *J. Comput. Phys.*, p. 229(19):7207–7237, 2010.

[Eld21] Jeff D. Eldredge. "A method of immersed layers on cartesian grids, with application to incompressible flows." *arXiv preprint arXiv:2103.04521*, 2021.

[GGS82] U. Ghia, K. N. Ghia, and C. T. Shin. "High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method." *J. Comput. Phys.*, pp. 48:387–411, 1982.

[GHS93] D. Goldstein, R. Handler, and L. Sirovich. "Modeling a no-slip flow boundary with an external force field." *J. Comput. Phys.*, p. 105:354–366, 1993.

[KC04] J.W. Kim and H.C. Choi. "An Immersed-Boundary Finite-Volume Method for Simulation ofHeat Transfer in Complex Geometries." *KSME Int'l Journal*, p. 18(6):1026 1035, 2004.

[KDH01] J.W. Kim, D.J.Kim, and H.C.Choi. "An Immersed-Boundary Finite-Volume Method for Simulations of Flow in Complex Geometries." *J. Comput. Phys.*, p. 171:132–150, 2001.

[KIH09] S.W. Kang, G. Iaccarino, F. Ham, and P. Moin. "Prediction of wall-pressure fluctuation in turbulent flows with an immersed boundary method." *J. Comput. Phys.*, p. 228:6753–6772, 2009.

[KT17] Takeo Kajishima and Kunihiko Taira. *Computational Fluid Dynamics Incompressible Turbulent Flows*. Springer, 2017.

[LC16] S. Liska and T. Colonius. "A fast lattice Green's function method for solving viscous incompressible flows on unbounded domains." *J. Comput. Phys.*, p. 316:360–384, 2016.

[LC17] S. Liska and T. Colonius. "A fast immersed boundary method for external incompressible viscous flows using lattice Green's functions." *J. Comput. Phys.*, p. 331:257–279, 2017.

[LTB16] Ugis Lacis, Kunihiko Taira, and Shervin Bagheri. "A stable fluid–structure-interaction solver for low-density rigid bodies using the immersed boundary projection method." *J. Comput. Phys.*, p. 305:300–318, 2016.

[MDB08] Rajat Mittal, Haibo Dong, Meliha Bozkurttas, FM Najjar, Abel Vargas, and Alfred Von Loebbecke. "A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries." *J. Comput. Phys.*, pp. 227(10):4825–4852,, 2008.

[MI05] R. Mittal and G. Iaccarino. "Immersed boundary methods." *Annu. Rev. Fluid Mech.*, p. 37:239–61, 2005.

[Pes72] C.S. Peskin. "Flow patterns around heart valves: a numerical method." *J. Comput. Phys.*, pp. 10:252–271, 1972.

[Pes77] C. S. Peskin. "Numerical analysis of blood flow in the heart." *J. Comput. Phys.*, p. 25:220–252, 1977.

[PM89] C. S. Peskin and D. M. McQueen. "A three-dimensional computational method for blood flow in the heart. I. Immersed elastic fibers in a viscous incompressible fluid." *J. Comput. Phys.*, p. 81:372–405, 1989.

[SB96] E. M. Saiki and S. Biringen. "Numerical simulation of a cylinder in uniform flow: Application of a virtual boundary method." *J. Comput. Phys.*, p. 123:450, 1996.

[TE21] Y. Thoy and J.D. Eldredge. "Github Code: JuliaIBPM/ViscousFlow.jl (v0.4.7.)." *https://github.com/JuliaIBPM/ViscousFlow.jl*, 2021.

[TK07] T.Colonius and K.Taira. "The immersed boundary method: a projection approach." *J. Comput.Phys.*, p. 225:2118–2137, 2007.

[TK08] T.Colonius and K.Taira. "A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions." *Comput. Methods Appl. Mech. Engrg.*, p. 197:2131– 2146, 2008.

[WE15] C. Wang and J. D. Eldredge. "Strongly coupled dynamics of fluids and rigid-body systems with the immersed boundary projection method." *J. Comput. Phys.*, p. 295:87–113, 2015.