A MAC GRID BASED MULTIGRID SOLVER FOR THE INCOMPRESSIBLE NAVIER STOKES EQUATIONS WITH IMMERSED BOUNDARIES

A THESIS

Presented to the Department of Mathematics and Statistics

California State University, Long Beach

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Mathematics

Option in Applied Mathematics

Committee Members:

Hui Sun, Ph.D. (Chair)
Bruce Chaderjian, Ph.D.
Chung-min Lee, Ph.D.

College Designee:

Will Murray, Ph.D.

By Matthew Dean Winger

B.S., 2018, California State University, Long Beach

December 2020

ABSTRACT

For this work, two-dimensional fluid dynamics within an enclosed square domain are simulated using the incompressible Navier-Stokes equations. The computational domain is discretized using the Marker-And-Cell (MAC) method which places velocity and pressure components in a mesh of staggered grids, offering improved solution stability as opposed to placing these components on the same grid. A modified form of the projection method is used to linearize the incompressible equations which produces a series of systems which are solved for velocity and pressure. With an immersed boundary method we can simulate the effect various surfaces with no-slip boundary conditions have on the fluid flow when placed inside our domain via a forcing term which is added to our systems. Solution of these linear systems is achieved by using the multigrid method, which uses restriction and interpolation operators that account for the staggered grids produced by the MAC method. With these tools we are able to describe a variety of types of fluid simulations, illustrated in this work with a series of numerical examples.

ACKNOWLEDGEMENTS

Many thanks go to my advisor, Paul Sun, for his unwavering support and mentorship, not only for this Thesis but for the entirety of my Master's degree and much of my undergraduate degree. You have opened innumerable doors for me by including me in your research, and prepared me well for my future in mathematics with your many opportunities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

Fluid dynamics are an important topic in mathematics that apply to a wide variety of disciplines such as engineering (Wackers et al. 2011), biology (Vargas and Mittal 2004; Rui, Guangbei, and Jihong 2008), epidemiology (Przekwas and Chen 2020), cardiology (Peskin 1977) and more. Accordingly, understanding and numerically approximating these fluid flows is an area of great importance to mathematics in the field of computational fluid dynamics. The incompressible Navier-Stokes equations describe the behavior of fluids at low mach numbers with uniform density. For this work these equations are used to simulate fluid flow in a two-dimensional square domain using a series of boundary conditions and initial values. This domain is discretized using a uniform mesh, and the fields for velocity and pressure are solved over a series of staggered grids via the Marker-and-Cell (MAC) method (Harlow and Welch 1965; Chen 2018).

Many schemes exist which can be used to approximate the solutions of the incompressible Navier-Stokes equations; commonly used is the finite element method (Tsega and Katiyar 2018), although spectral methods (Botella and Peyret 1998) and finite difference methods are also utilized, the latter of which is chosen for the present work. In particular we use a projection method similar to that developed by Bell and Colella (1991) offering improved accuracy in the divergence of the velocity as well as superior stability in the approximation of the nonlinear term, called Projection Method II (Brown, Cortez, and Minion 2001).

An important aspect of fluid dynamics is the interaction of a fluid flow with surfaces of various possible size, shape, and position. While an obvious approach to this type of simulation could be a direct modification to the mesh corresponding to the computational domain to include some surface, this may be difficult to implement particularly for surfaces which can move or transform with time. The Immersed Boundary Method first introduced by Peskin (1977) solves this problem by using a regular Eulerian mesh for fluid flow coupled with a Lagrangian mesh to describe the immersed boundary. This method is improved upon in Lima et al.'s (2003) work with

the Physical Virtual Model which is applied to the simulation in the present work.

Solving the linear systems produced by the projection method presents another choice of methods. Direct solution methods are not particularly useful to this application for a few reasons particularly due to the size of the calculations that will need to be performed and the resulting poor computational efficiency. Furthermore they can struggle to produce solutions for singular matrix systems, like those found when solving the incompressible Navier-Stokes equations. Because of this we choose an iterative solver in the form of the Multigrid method (Briggs, Henson, and McCormick 2000), which uses a series of grid transformations paired with a typical iterative method to produce an accurate solution with relatively few iterations of the method.

In the following sections these numerical methods and their application to the incompressible Navier-Stokes equations will be discussed in detail, in particular the Multigrid method. A series of numerical experiments are performed to verify and validate the methods used, and the computational efficiency and performance of the multigrid method is further investigated.

CHAPTER 2

INCOMPRESSIBLE NAVIER STOKES

For a bounded domain Ω in $\mathbb{R}^n$ we have the incompressible Navier-Stokes equations:

$$\mathbf{u}_t + \nabla p = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{F} \tag{2.1}$$

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

where **u** is a vector of n velocity components and p is pressure, which are variables we wish to solve for. Parameters are ν, the kinematic viscosity such that $\nu = \frac{1}{Re}$ where Re is the Reynolds number and finally **F** is a forcing term introduced by the immersed boundary method. The key to incompressibility is the divergence-free requirement for velocity as given by the continuity equation (2.2), enforcement of this condition is of particular importance when determining a solution method. Additional difficulty in solving these equations comes from both the pressure gradient as well as the nonlinear advective derivative, for which careful treatment is required to ensure convergence and stability.

CHAPTER 3

MAC GRID DISCRETIZATION

The domain to be used for this simulation is the unit square in $\mathbb{R}^2$ which will be labelled $\Omega := [0,1] \times [0,1]$. To discretize this domain we divide the x and y sides of Ω into $n = 2^k$ sub-intervals where $k \in \mathbb{N}$, uniformly dividing the domain into n^2 squares with sides of length $h = \frac{1}{n}$. We use the Marker-and-Cell (MAC) (Harlow and Welch 1965) grid scheme to define nodes that correspond to the x and y components of velocity and pressure on three separate grids, as opposed to a typical Cartesian scheme that uses a single grid for all three components, this arrangement of grids can be seen in Figure 3.1.

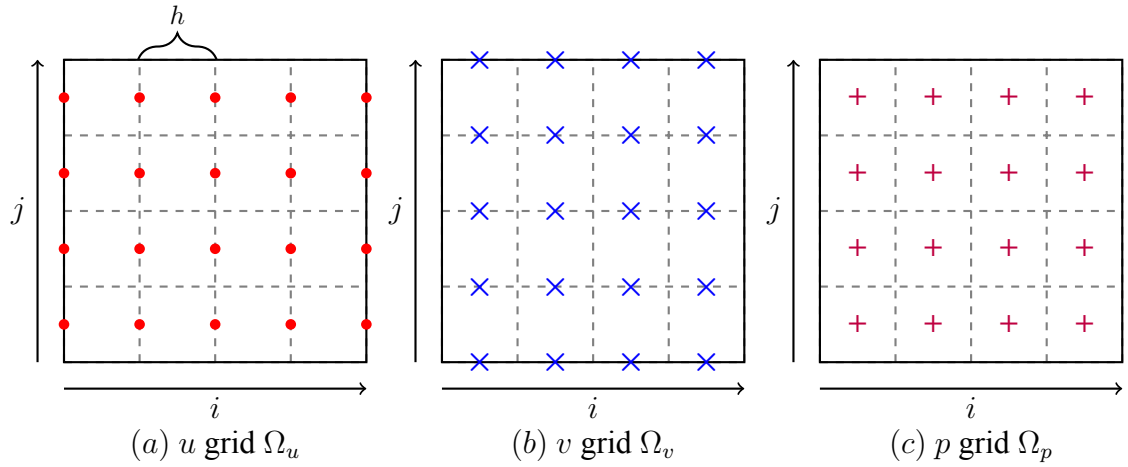

FIGURE 3.1. The three grids for solution nodes as defined by the MAC method.

The notation used for nodes on these staggered grids is similar to that of the Cartesian scheme, with a few minor changes as demonstrated in Figure 3.2.

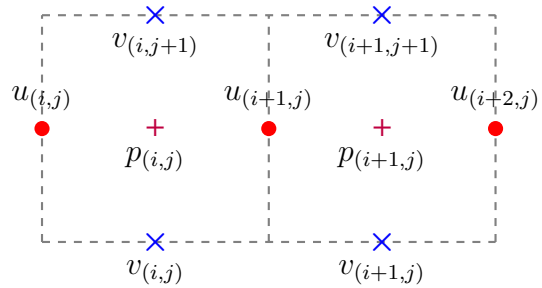

FIGURE 3.2. Indexing scheme for MAC method solution nodes.

With this discretization we have a vector for each grid, two velocity vectors $u, v \in \mathbb{R}^{n(n+1)}$, and a pressure vector $p \in \mathbb{R}^{n^2}$. Indexing of nodes follows from the Cartesian scheme with i, j subscript used to represent position on the grid. For the MAC scheme however, we introduce half-steps, and since each grid has a different number of nodes they will each have a different set of values for i, j.

$$\Omega_u \coloneqq \{u_{(i,j)} = u(ih, (j+0.5)h) : i \in [0, n], j \in [0, n-1]\}$$
$$\Omega_v \coloneqq \{v_{(i,j)} = v((i+0.5)h, jh) : i \in [0, n-1], j \in [0, n]\}$$
$$\Omega_p \coloneqq \{p_{(i,j)} = p((i+0.5)h, (j+0.5)h) : i \in [0, n-1], j \in [0, n-1]\}$$

Use of the MAC discretization has several benefits over the typical Cartesian scheme, enforcing incompressibility to machine round-off as well as providing improved solution stability. Particularly when solving for pressure, the MAC method overcomes a known "checkerboarding" issue when using the Cartesian scheme, observed in McDonough's (2007) work where the center-difference scheme for the gradient is zero for a non-constant grid. This phenomenon is demonstrated for the one dimensional case in Figure 3.3. For this example the center-difference gradient will produce the result $\nabla p = 0$ which implies p is constant, though by the figure this is clearly untrue.

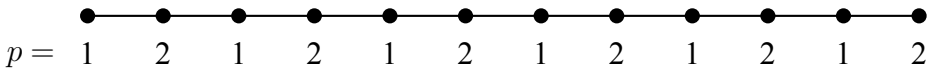

FIGURE 3.3. Pressure 'checkerboarding' effect in one dimension.

CHAPTER 4

PROJECTION METHOD II

Solution of the incompressible Navier-Stokes equation is achieved using a projection method similar that developed by Howell, Bell, and Colella's (1991) with improved accuracy and stability, referred to as Projection Method II (PMII) from Brown, Cortez, and Minion's (2001) work, with a key difference being the treatment of the advective derivatives which are approximated with the second-order Adams-Bashforth method. Based on the Helmholtz-Hodge decomposition, PMII decouples the solutions for velocity and pressure, with a "lagged" pressure term present in the calculations for velocity. This is accomplished by first solving for an intermediate velocity $\mathbf{u}^*$ calculated from the previous time step $\mathbf{u}^n$ without regard for the incompressibility condition such that $\mathbf{u}^{n+1}$, the solution in the next time step is the divergence-free part of $\mathbf{u}^*$; according to the Helmholtz-Hodge decomposition, this can be formulated as $\mathbf{u}^* = \mathbf{u}^{n+1} + \Delta t \nabla \phi^{n+1}$. Next we solve for the correction term ϕ^{n+1} that, when applied to the intermediate velocity, makes it divergence free; finally this correction term is used to update the lagged pressure term. This method provides a second-order accurate solution for both velocity and pressure.

The first step of this method is to find a solution for the intermediate velocity, $\mathbf{u}^* = [u^*, v^*]^T$ (containing x and y components respectively) with the equation:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + \nabla p^{n-1/2} = -[(\mathbf{u} \cdot \nabla)\mathbf{u}]^{n+1/2} + \frac{\nu}{2}\nabla^2(\mathbf{u}^n + \mathbf{u}^*) + \mathbf{F}^n \quad \text{on } \Omega \tag{4.1}$$

$$\mathbf{u}^* = \mathbf{u}_b \quad \text{on } \partial\Omega$$

where $\mathbf{u}_b$ is the boundary condition for velocity, which for this simulation will be pure Dirichlet on the boundaries of Ω. The second order Adams-Bashforth formula is used to approximate the advective derivatives which is written:

$$[(\mathbf{u} \cdot \nabla)\mathbf{u}]^{n+1/2} = \frac{3}{2}[(\mathbf{u} \cdot \nabla)\mathbf{u}]^n - \frac{1}{2}[(\mathbf{u} \cdot \nabla)\mathbf{u}]^{n-1}$$

Crank-Nicolson is used for the Laplace term for the purpose of stability and second-order accuracy. By taking the divergence of the Helmholtz-Hodge formulation $\mathbf{u}^* = \mathbf{u}^{n+1} + \Delta t \nabla \phi^{n+1}$ we obtain the equation to solve for the correction term ϕ^{n+1} through the following equation:

$$\begin{aligned} \Delta t \nabla^2 \phi^{n+1} &= \nabla \cdot \mathbf{u}^* \quad \text{on } \Omega \\ \hat{\mathbf{n}} \cdot \nabla \phi^{n+1} &= 0 \quad \text{on } \partial\Omega \end{aligned} \tag{4.2}$$

It is notable to mention that for the Navier-Stokes equation it is typical for the pressure to use Neumann boundary conditions where we have Dirichlet boundary conditions for velocity, enforced here via the ϕ^{n+1} term.

Finally we update both our intermediate velocity solution as well as the "lagged" pressure term for the next time step with our update term ϕ^{n+1} via:

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t \nabla \phi^{n+1} \tag{4.3}$$

$$\nabla p^{n+1/2} = \nabla p^{n-1/2} + \nabla \phi^{n+1} - \frac{\nu \Delta t}{2} \nabla \Delta \phi^{n+1} \tag{4.4}$$

While solving the next time step for velocity requires the gradient of pressure, the pressure term itself can be recovered with the following since ∇ is a linear operator:

$$p^{n+1/2} = p^{n-1/2} + \phi^{n+1} - \frac{\nu \Delta t}{2} \Delta \phi^{n+1}$$

The solution procedure and discretization of the operators used for PMII are discussed in detail in the following sections.

Solving for Intermediate Velocity $\mathbf{u}^*, \mathbf{v}^*$

The first step of PMII involves the vector $\mathbf{u}^* = [u^*, v^*]^T$ and so ultimately we have a pair of systems to solve corresponding to the x and y component of velocity. Here we will discuss only the process for solving u, but the methods used here are nearly identical and are easily transferable to the solution of v.

With this we can write out the first component of (4.1) which is to solve for u^*:

$$\frac{u^* - u^n}{\Delta t} + \nabla^{\Omega_u} p^{n-1/2} = -\frac{3}{2} AD_x(\mathbf{u}^n) + \frac{1}{2} AD_x(\mathbf{u}^{n-1}) + \frac{\nu}{2} \nabla_x^2 (u^n + u^*) + F_x^n \tag{4.5}$$

$$AD_x(\mathbf{u}^n) = u^n \odot \nabla_x u^n + T^{\Omega_u} v^n \odot \nabla_y u^n$$

where AD_x is the x component of the advective derivative and $\odot$ represents element-wise multiplication of vectors.

From (4.5) we see there is data from all three domains that contribute to solving for $\mathbf{u}^*$ on Ω_u. We will introduce the superscript Ω_u to indicate an operator which transfers data between grids; in this case the operator would transfer from Ω_p or Ω_v to Ω_u. Also we will introduce the interpolation operator T which represents interpolating data between Ω_u and Ω_v.

We can rewrite (4.5) as a linear system of equations of the form $Ax = b$ which will be solved for u^*:

$$(I - \frac{\nu \Delta t}{2} \nabla^2) u^* = \Delta t \left[-\frac{3}{2} AD_x(\mathbf{u}^n) - \frac{1}{2} AD_x(\mathbf{u}^{n-1}) + \frac{\nu}{2} \nabla^2 u^n + F_x^n - \nabla_x^{\Omega_u} p^{n-1/2} \right] + u^n \tag{4.6}$$

To discretize ∇^2, ∇_x, and ∇_y for $\mathbf{u} \in \Omega_u$ we use typical second-order finite difference methods:

$$\nabla^2 u^n_{(i,j)} = \frac{u^n_{(i-1,j)} + u^n_{(i+1,j)} + u^n_{(i,j-1)} + u^n_{(i,j+1)} - 4u^n_{(i,j)}}{h^2} \tag{4.7}$$

$$\nabla_x u^n_{(i,j)} = \frac{u^n_{(i+1,j)} - u^n_{(i-1,j)}}{2h} \tag{4.8}$$

$$\nabla_y u^n_{(i,j)} = \frac{u^n_{(i,j+1)} - u^n_{(i,j-1)}}{2h} \tag{4.9}$$

For $\nabla_x^{\Omega_u}$ operator we use the following scheme to discretize (noting in this case we are transferring data from the Ω_p grid to the Ω_u grid):

$$\nabla_x^{\Omega_u} p^n_{(i,j)} = \frac{p^n_{(i,j)} - p^n_{(i-1,j)}}{h} \tag{4.10}$$

Finally we have the term $T^{\Omega_u} v^n$, which is the transfer of data from our Ω_v grid to our Ω_u grid. This is achieved by using a simple linear interpolation:

$$T^{\Omega_u} v^n_{(i,j)} = \frac{v_{(i,j+1)} + v_{(i+1,j)} + v_{(i,j-1)} + v_{(i+1,j-1)}}{4} \tag{4.11}$$

Velocity Boundary Treatment

We have two cases to apply the Dirichlet boundary conditions on the Ω_u grid (this applies to the Ω_v grid as well). The left and right side of the domain corresponding to $x = 0$ and $x = 1$ respectively has nodes that lie directly on the boundary. Because of this we can directly impose our Dirichlet boundary conditions on these nodes in our matrix equation $Ax = b$. This is accomplished by modifying the matrix A by setting the diagonal component in the row corresponding to a boundary node of A equal to 1 and all others equal to 0, and setting the same row of b equal to the corresponding Dirichlet boundary value.

On the top and bottom of the Ω_u domain we have nodes that lie a half step ($\frac{h}{2}$) away from the $y = 0$ and $y = 1$ boundaries. For these points we introduce a "ghost node," for example $u_{(i,-1)}$, that lies a half step *outside* the boundary. To account for this we use a quadratic extrapolation to apply the Dirichlet boundary value to this ghost node, which we can then apply to our operators. Solving for the ghost node results in $u_{(i,-1)} = -2u_{(i,0)} + \frac{1}{3}u_{(i,1)} + \frac{8}{3}u_b$ where the left hand side is the ghost node and the right hand side is made up of data where u_b is the appropriate boundary value; this can then be applied to the finite difference operators.

The following are the modified finite difference formulas for operators on the bottom boundary $j = 0$:

$$\nabla^2 u^n_{(i,0)} = \frac{u^n_{(i-1,0)} + u^n_{(i+1,0)} + \frac{4}{3}u^n_{(i,1)} - 6u^n_{(i,0)}}{h^2} + \frac{8}{3}\frac{u_b}{h^2}$$
$$\nabla_y u^n_{(i,0)} = \frac{\frac{2}{3}u^n_{(i,1)} + 2u^n_{(i,0)}}{2h} - \frac{4}{3}\frac{u_b}{h}$$

Solving for the Pressure Update ϕ^{n+1}

The next step of our projection method involves a linear system whose solution ϕ^{n+1}, defined on Ω^p, is the correction term that will enforce the incompressibility constraint on the intermediate velocity solutions:

$$\Delta t \nabla^2 \phi^{n+1} = \nabla^{\Omega_p} \cdot \mathbf{u}^* \tag{4.12}$$

$$\hat{\mathbf{n}} \cdot \nabla \phi^{n+1} = 0 \tag{4.13}$$

Where we discretize the laplacian using a similar second order 5-stencil operator similar to (4.7), and the following operators to compute the gradient of the intermediate velocities on the Ω_p grid:

$$\nabla_x^{\Omega_p} u^*{}_{(i,j)} = \frac{u^*_{(i+1,j)} - u^*_{(i,j)}}{h} \tag{4.14}$$

$$\nabla_y^{\Omega_p} v^*{}_{(i,j)} = \frac{v^*_{(i,j+1)} - v^*_{(i,j)}}{h} \tag{4.15}$$

We can then rewrite (4.12) as a linear system of equations $Ax = b$ with (4.14) and (4.15) as:

$$\nabla^2 \phi^{n+1} = \frac{(\nabla_x^{\Omega_p} u^* + \nabla_y^{\Omega_p} v^*)}{\Delta t}$$

The linear system produced by this method when coupled with a zero-flux Neumann boundary conditions has a symmetric singular matrix A, and so to ensure we have a unique solution the RHS of the equation must satisfy the compatibility condition:

$$\int_{\Omega_p} \nabla^{\Omega_p} \cdot \mathbf{u}^* \, \mathrm{dA} = 0 \tag{4.16}$$

Using the divergence theorem, (4.16) can be re-written as an integral of the normal velocity along the boundary of the computational domain, and the compatibility condition is then equivalent to zero net fluid flux across the boundary. Since there is no fluid inflow for this simulation, the RHS of (4.12) satisfies this condition and the PDE is well-posed with the inclusion of the zero-mean

condition:

$$\int_{\Omega_p} \phi^{n+1} \, \mathrm{dA} = 0 \tag{4.17}$$

Because A is symmetric, A and A^T their null space and so solvability and uniqueness of the solution can be handled simultaneously. In this case the 1-dimensional null space is simply the constant vector **1**; with this we can apply Gram-Schmidt to satisfy (4.17):

$$\phi^{n+1} = \phi^{n+1} - \frac{\left(\sum_{\Omega_p} \phi^{n+1}\right)}{n^2}$$

Pressure Boundary Treatment

For the pressure update term we have the no-flux Neumann boundary conditions $\hat{n} \cdot \nabla \phi^{n+1} = 0$ on all sides of the domain. To impose this condition on our linear system we will introduce ghost nodes outside the domain opposite to all boundary nodes in Ω_p, then we use the center difference to determine the value of the ghost node.

For example at the bottom of the domain $j = 0$ we can use the ghost node $\phi^{n+1}_{(i,-1)}$ and approximate the boundary condition with $\nabla_y \phi^{n+1} = \frac{\phi^{n+1}_{(i,0)} - \phi^{n+1}_{(i,-1)}}{h} = 0$. With this we can make the following modification to the ∇^2 finite difference operator to accommodate the boundary condition:

$$\nabla^2 \phi^n_{(i,0)} = \frac{\phi^n_{(i-1,0)} + \phi^n_{(i+1,0)} + \phi^n_{(i,1)} - 3\phi^n_{(i,0)}}{h^2}$$

The corners of the computational domain differ slightly in that they require an additional application of the boundary data.

Velocity Update

Once we have solved for the pressure update term ϕ^{n+1} we can use it to update the intermediate velocity components $\mathbf{u}^*$ to the next time step $\mathbf{u}^{n+1}$. This is accomplished by taking

the $\nabla_x^{\Omega_u}$ and $\nabla_y^{\Omega_v}$ gradients on ϕ^{n+1} to compute u^{n+1} and v^{n+1} respectively via:

$$u^{n+1} = u^* - \Delta t \nabla_x^{\Omega_u} \phi^{n+1}$$
$$v^{n+1} = v^* - \Delta t \nabla_y^{\Omega_v} \phi^{n+1}$$

Boundary conditions for u^{n+1} and v^{n+1} are enforced in the solution of the intermediate step and so the solution is not updated on these boundaries (this also follows from the Neumann boundary condition imposed in the system (4.12)). The key of the projection method is that this step provides the correction to $\mathbf{u}^*$ to make $\mathbf{u}^{n+1}$ divergence free, satisfying (2.2).

Pressure Gradient Update

The final step is to update the "lagged" pressure terms $\nabla_x p^{n+1/2}$ and $\nabla_y p^{n+1/2}$ defined on the Ω_u and Ω_v domains respectively, which are necessary to compute the next step of velocity. To find these we use the equations:

$$\nabla_x p^{n+1/2} = \nabla_x p^{n-1/2} + \nabla_x^{\Omega_u} \phi^{n+1} - \frac{\nu \Delta t}{2} \nabla_x^{\Omega_u} \nabla^2 \phi^{n+1} \tag{4.18}$$
$$\nabla_y p^{n+1/2} = \nabla_y p^{n-1/2} + \nabla_y^{\Omega_v} \phi^{n+1} - \frac{\nu \Delta t}{2} \nabla_y^{\Omega_v} \nabla^2 \phi^{n+1} \tag{4.19}$$

where we use (4.2) to re-write the last terms of (4.18) and (4.19) to avoid introducing error from the solution of ϕ^{n+1} as:

$$\frac{\nu \Delta t}{2} \nabla_x^{\Omega_u} \nabla^2 \phi^{n+1} = \frac{\nu}{2} \nabla_x^{\Omega_u} \left(\nabla_x^{\Omega_p} u^* + \nabla_y^{\Omega_p} v^* \right) \tag{4.20}$$
$$\frac{\nu \Delta t}{2} \nabla_y^{\Omega_v} \nabla^2 \phi^{n+1} = \frac{\nu}{2} \nabla_y^{\Omega_v} \left(\nabla_x^{\Omega_p} u^* + \nabla_y^{\Omega_p} v^* \right) \tag{4.21}$$

Finally, using (4.20) and (4.21) and taking advantage of the linearity of the gradient operator we can recover the solution for pressure with:

$$p^{n+1/2} = p^{n-1/2} + \phi^{n+1} - \frac{\nu \Delta t}{2} \left(\nabla_x^{\Omega_p} u^* + \nabla_y^{\Omega_p} v^* \right) \tag{4.22}$$

CHAPTER 5

IMMERSED BOUNDARY METHOD

An often important aspect of fluid dynamics is the interaction of fluid flows with a structure and its subsequent influence on the dynamics of flow in the domain. The immersed boundary method (IBM) is used to simulate these fluid-structure interactions via a forcing term added to the incompressible Navier-Stokes equation. This method is advantageous due to its ease of implementation; rather than adapting the mesh to reflect the presence of a structure we instead define fluid flow governed by the incompressible Navier-Stokes equation on a Eulerian domain, in our case this is the unit square Ω discretized using the MAC grid method; the surface the fluid will interact with is then separately defined on a Lagrangian domain which is labelled Γ via nodes on the surface boundary, these respective coordinate systems can be seen in Figure 5.1. For this work we implement the Physical Virtual Model from Silva et al.'s (2003) work; this selection is made due to the model's simplicity comared to other IBM methods, since it is based only on the laws of conservation.

We define the Lagrangian domain by a set of k uniformly distributed nodes on the boundary of the desired surface. The distance between two Lagrangian nodes is Δs and we use the ratio $\frac{\Delta s}{h} \leq 0.9$ to determine the number of nodes α required based on the mesh size of the Eulerian domain.

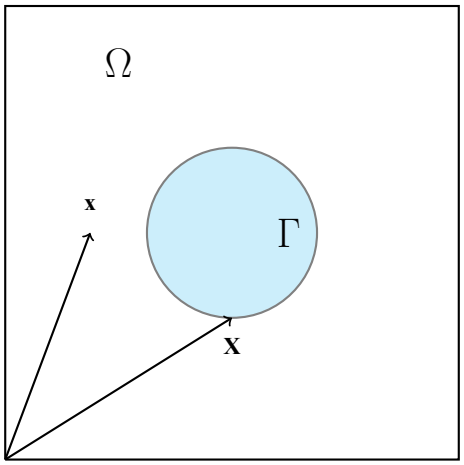

FIGURE 5.1. Immersed boundary method coordinate system, where x is in the Eulerian domain and X is in the Lagrangian domain.

Interactions of the surface with the fluid flow are modelled by the forcing term **F**=(F_x, F_y) which is added to the velocity equations for incompressible Navier-Stokes (2.1) in the domain Ω. The definition for the forcing term is given by:

$$\mathbf{F}(\mathbf{x}, t) = \int_{\Gamma} \mathbf{f}(\mathbf{X}, t)\delta(\mathbf{x} - \mathbf{X})\text{ds} \tag{5.1}$$

where $\mathbf{x} = (x_i, y_j)$ represents the coordinates of the Eulerian domains Ω_u and Ω_v, $\mathbf{X} = (X, Y)$ are the coordinates of nodes in the Lagrangian domain Γ with indices $k = 0, 1, 2...\alpha$. Finally $\mathbf{f} = (f_x, f_y)$ is the force field over the Lagrangian domain and δ is the Dirac delta function. The Dirac delta function is discretized using the distribution function D_{ij} defined in Juric's (1996) work with:

$$D_{ij} = \frac{f\left(\frac{X-x_i}{h}\right) f\left(\frac{Y-y_j}{h}\right)}{h^2} \tag{5.2}$$

$$f(r) = \begin{cases} h(r) & |r| < 1 \\ \frac{1}{2} - h(2 - |r|) & 1 < |r| < 2 \\ 0 & |r| > 2 \end{cases}$$

$$h(r) = \frac{3 - 2|r| + \sqrt{1 + 4|r| - 4|r|^2}}{8}$$

Because we need to find the forcing term on both domains Ω_u and Ω_v we will find D_{ij} for the coordinates on each domain; to achieve this we create a pair of matrices for the distribution function on each domain which will be represented as D^{Ω_u} and D^{Ω_v}, where each matrix is in $\mathbb{R}^{\alpha \times n(n+1)}$. Since fluid flow is defined on the Eulerian grid, the process of finding **f** on the Lagrangian domain first requires us to find the force field on the Eulerian domain, and then perform an interpolation to the Lagrangian domain Γ. The first step in this process begins by

solving for $\mathbf{F}_\Omega = (F_{\Omega_u}, F_{\Omega_v})$:

$$\mathbf{F}_\Omega = \mathbf{u}_t + \nabla p + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu\nabla^2\mathbf{u}$$

Discretizing in time we have the following vectors in $\mathbb{R}^{n(n+1)}$ for the x and y components of the forcing term on the Eulerian grid:

$$F^n_{\Omega_u} = \frac{u^* - u^n}{\Delta t} + \nabla_x p^{n-1/2} + \frac{3}{2}AD_x(\mathbf{u}^n) - \frac{1}{2}AD_x(\mathbf{u}^{n-1}) - \frac{\nu}{2}\nabla^2(u^n + u^*) \tag{5.3}$$

$$F^n_{\Omega_v} = \frac{v^* - v^n}{\Delta t} + \nabla_y p^{n-1/2} + \frac{3}{2}AD_y(\mathbf{u}^n) - \frac{1}{2}AD_y(\mathbf{u}^{n-1}) - \frac{\nu}{2}\nabla^2(v^n + v^*) \tag{5.4}$$

The values of $\mathbf{u}^n$ and $\mathbf{u}^{n-1}$ are velocity values from previous time steps, and u^* and v^* represent the Dirichlet boundary data on the immersed boundary, in the case of the no-slip boundary condition we use $u^* = v^* = 0$. More details on deriving these forcing terms can be found in the work by Silva et al. (2003). $\mathbf{F}_\Omega$ is then interpolated to the Lagrangian domain Γ using the discrete Dirac delta and h via:

$$f^n_x = D^{\Omega_u} F^n_{\Omega_u} h^2 \qquad\qquad f^n_y = D^{\Omega_v} F^n_{\Omega_v} h^2 \tag{5.5}$$

where we solve for the time-discretized force field $\mathbf{f}$ on Γ. Finally we can then perform the interpolation given by (5.1) to apply the force field on Γ to the Eulerian domain, again using the discrete Dirac delta functions D^{Ω_u} and D^{Ω_v} although this time we instead use the transpose to convert from Γ to Ω.

$$F^n_x = (D^{\Omega_u})^T f^n_x \Delta s^2 \qquad\qquad F^n_y = (D^{\Omega_v})^T f^n_y \Delta s^2 \tag{5.6}$$

Now the completed forcing operator $\mathbf{F}^n$ on the Eulerian domain Ω is used in (4.6) to find the solution for velocity and pressure for the next time step.

CHAPTER 6

MULTIGRID METHOD

To compute solutions for our linear systems (4.6) and (4.12) generated by the projection method we use the Multigrid method (MG). This algorithm is based on typical relaxation methods - in the case of this paper the Gauss-Seidel method is used, but MG provides the advantage of higher accuracy with fewer iterations of the relaxation scheme, offering reduced computational cost for solution. The primary operators of MG are restriction and prolongation of the computational domain; restriction is where interval length is doubled, moving to a 'coarse' grid, and prolongation halves the interval length, going to a 'fine' grid. Our discretization of the computational domain into $n = 2^k$ sub-intervals with $k \in \mathbb{N}$ allows us to easily restrict and prolongate the MAC grids to coarser or finer levels. In addition to the restriction and relaxation operators, the MG method has two main solution components that are combined to create the Full Multigrid algorithm (FMG), these are the correction scheme and nested iteration.

The notation for MG indicates the grid a solution field is defined on. For example u^h is on the grid Ω^h with intervals of size h. The restriction operator R_h^{2h} transfers data from Ω^h to the coarser grid Ω^{2h} and the prolongation operator R_{2h}^h transfers from Ω^{2h} to finer grid Ω^h. To represent these grid transfers we can write $R_h^{2h}u^h = u^{2h}$ which is the restriction of the vector u, and $R_{2h}^h u^{2h} = u^h$ for prolongation; the effect of these operators is illustrated in Figure (6.1).

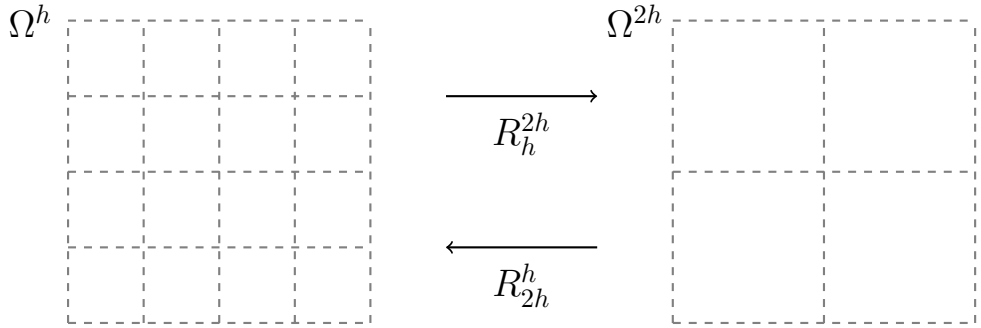

FIGURE 6.1. Restriction and Prolongation between the domains Ω^h and Ω^{2h}.

vCycle

The core tool of the multigrid method is the vCycle, a process of restriction and prolongation of the computational domain that first obtains an initial guess through a small number of relaxation iterations, and then solves for the error of this initial guess that is then used to correct the guess.

For the code used in our simulation, the vCycle is programmed recursively and a single iteration of the vCycle is designated $VC\left\langle v^h, b^h\right\rangle = u^h$ where v^h is the initial guess, b^h is the RHS of the linear system to be solved and u^h is the computed solution. The recursive definition of the VCycle is as follows:

Algorithm 6.1: VCycle Algorithm $VC\left\langle v^h, b^h\right\rangle$

Input: v^h initial guess, b^h system RHS

Result: u^h solution vector

if Ω^h *is the coarsest grid* **then**

Relax the system $A^h v^h = b^h$ 10 times

Return the solution u^h

else

Relax the linear system $A^h v^h = b^h$ 5 times with the initial guess v^h

Compute the residue $r^h = b^h - A^h v^h$ and restrict to the Ω^{2h} grid to obtain $b^{2h} = R_h^{2h} r^h$

Calculate $v^{2h} = VC\left\langle v^{2h}, b^{2h}\right\rangle$ on the Ω^{2h} grid with initial guess $v^{2h} = 0$

Interpolate v^2h to find the correction $v^h = v^h + R_{2h}^h v^{2h}$

Relax the linear system $A^h v^h = b^h$ 5 times on the Ω^h grid

end

Full Multigrid

While using the vCycle method offers a powerful correction scheme to improve the initial guess obtained from the first step of relaxation, a more accurate solution can be obtained by generating a *better* initial guess to relax on by performing a vCycle on a coarser grid. It is this

idea of nested iteration that leads to the Full Multigrid (FMG) method, which first produces an initial guess on the coarsest grid, then builds this guess up to the finest grid Ω^h to produce a more accurate solution than running a single vCycle. Similar to the VCycle we define FMG recursively, here the process is outlined:

Algorithm 6.2: Full Multigrid Algorithm $FMG\left\langle b^h \right\rangle$

Input: b^h system RHS

Result: u^h solution vector

if Ω^h *is the coarsest grid* **then**

$u^h \leftarrow VC\left\langle v^h, b^h \right\rangle$ with initial guess $v^h = 0$

else

Restrict $b^{2h} = R_h^{2h} b^h$ and solve $u^{2h} = FMG\left\langle b^{2h} \right\rangle$ with initial guess $v^{2h} = 0$ and return the solution

Prolongate $u^{2h} \rightarrow v^h = R_{2h}^h u^{2h}$ and perform a VCycle to obtain the solution $u^h = VC\left\langle v^h, b^h \right\rangle$

end

Grid Transferring

The primary operators used in the multigrid method are the Restriction and Prolongation operators, which, respectively, bring the computational grid to a coarser or finer level. For the purposes of this project, a simple linear restriction/prolongation scheme is used, although more complicated quadratic or cubic schemes are possible. The MAC grid arrangement of the grids used in our Navier-Stokes simulation require a more careful implementation of these operators, but they are not dissimilar to those used for a typical Cartesian grid scheme. Operators used in this section are obtained from Chen's (2018) work or are directly based on bi-linear interpolation.

Restriction

The restriction operator coarsens the computational domain, providing a transformation from the domain $\mathbb{R}^{n(n+1)}$ to $\mathbb{R}^{\frac{n}{2}(\frac{n}{2}+1)}$ in the case of the velocity components on Ω_u and Ω_v, and is

labelled R_h^{2h} where $h = \frac{1}{n}$ is the length of the interval. For the pressure grid Ω_p the domain is instead restricted from $\mathbb{R}^{n^2}$ to $\mathbb{R}^{\frac{n}{2}^2}$ due to the size of MAC method's staggered grids.

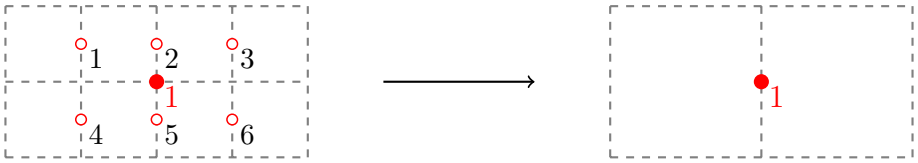

FIGURE 6.2. Restriction stencil for Ω_u.

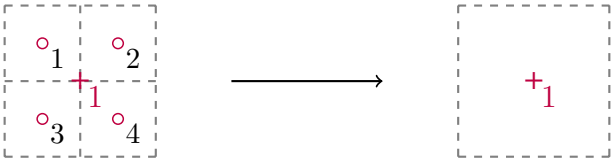

FIGURE 6.3. Restriction stencil for Ω_p.

While each grid type requires a unique restriction operator, those for Ω_u and Ω_v are nearly identical and the difference between the two can be observed using stencil notation (where restriction of the Ω_v grid can be represented by rotating Figure (6.2) by $90°$):

$$R^{\Omega_u} = \frac{1}{8}\begin{pmatrix} 1 & 2 & 1 \\ & * & \\ 1 & 2 & 1 \end{pmatrix} \quad R^{\Omega_v} = \frac{1}{8}\begin{pmatrix} 1 & & 1 \\ 2 & * & 2 \\ 1 & & 1 \end{pmatrix} \quad R^{\Omega_p} = \frac{1}{4}\begin{pmatrix} 1 & & 1 \\ & * & \\ 1 & & 1 \end{pmatrix}$$

We can use these operators to move to coarser grid levels where a coarser level corresponds to dividing the number of intervals n by 2. To implement these operators we can write them as equations, the restriction operator for the Ω_u can be written:

$$u_1^{2h} = \frac{1}{8}(u_1^h + u_3^h + u_4^h + u_6^h) + \frac{2}{8}(u_2^h + u_5^h)$$

and similarly for the Ω_p grid:

$$p_1^{2h} = \frac{1}{4}(p_1^h + p_2^h + p_3^h + p_4^h)$$

where the indices correspond to those seen in Figures 6.2 and 6.3 respectively.

Prolongation

Prolongation of the domains follows from the restriction operators, also using a bi-linear interpolation to refine the grids from $\mathbb{R}^{n(n+1)}$ to $\mathbb{R}^{2n(2n+1)}$ for the velocity components on Ω_u and Ω_v, and from $\mathbb{R}^{\frac{n}{2}^2}$ to $\mathbb{R}^{n^2}$ for the pressure on Ω_p; similar to restriction, the prolongation operators are labelled R^h_{2h}.

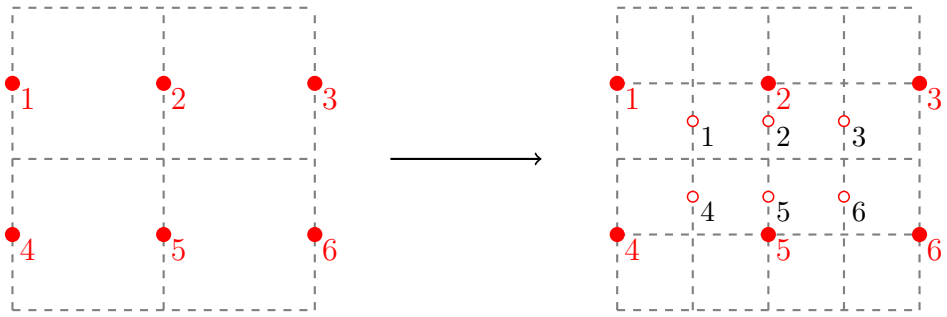

FIGURE 6.4. Prolongation stencil for Ω_u.

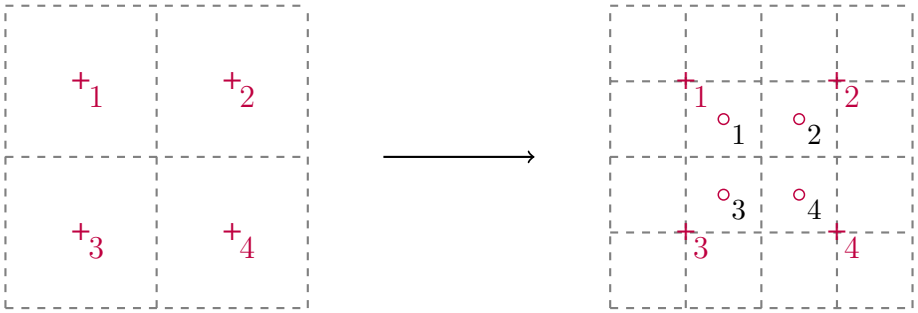

FIGURE 6.5. Prolongation stencil for Ω_p.

Similarly to the restriction operators, the prolongation operators for the Ω_u and Ω_v grids are nearly the same. In particular for the Ω_u grid we will have the prolongation operators:

$$u^h_1 = \frac{3}{8}\left[u^{2h}_1 + u^{2h}_2\right] + \frac{1}{8}\left[u^{2h}_4 + u^{2h}_5\right] \qquad u^h_3 = \frac{3}{8}\left[u^{2h}_2 + u^{2h}_3\right] + \frac{1}{8}\left[u^{2h}_5 + u^{2h}_6\right]$$

$$u^h_2 = \frac{3}{4}u^{2h}_2 + \frac{1}{4}u^{2h}_5 \qquad u^h_5 = \frac{3}{4}u^{2h}_5 + \frac{1}{4}u^{2h}_2$$

$$u^h_4 = \frac{3}{8}\left[u^{2h}_4 + u^{2h}_5\right] + \frac{1}{8}\left[u^{2h}_1 + u^{2h}_2\right] \qquad u^h_6 = \frac{3}{8}\left[u^{2h}_5 + u^{2h}_6\right] + \frac{1}{8}\left[u^{2h}_2 + u^{2h}_3\right]$$

For the Ω_p domain our prolongation operators will look like:

$$p_1^h = \frac{9}{16}p_1^{2h} + \frac{3}{16}\left[p_2^{2h} + p_3^{2h}\right] + \frac{1}{16}p_4^{2h} \qquad p_2^h = \frac{9}{16}p_2^{2h} + \frac{3}{16}\left[p_1^{2h} + p_4^{2h}\right] + \frac{1}{16}p_3^{2h}$$
$$p_3^h = \frac{9}{16}p_3^{2h} + \frac{3}{16}\left[p_1^{2h} + p_4^{2h}\right] + \frac{1}{16}p_2^{2h} \qquad p_4^h = \frac{9}{16}p_4^{2h} + \frac{3}{16}\left[p_2^{2h} + p_3^{2h}\right] + \frac{1}{16}p_1^{2h}$$

Again the indices for these equations follow from Figures 6.4 and 6.5 respectively. When performing prolongations on nodes that lie near the boundaries of the domain, we assume a zero value for nodes that lie outside the domain.

CHAPTER 7

NUMERICAL RESULTS

In this section we perform two numerical experiments to illustrate the convergence and stability of PMII. The first experiment verifies the rate of convergence for PMII using the 2D Taylor-Green vortex, an analytical solution for the incompressible Navier-Stokes equations; we also use this experiment to analyze the efficiency and speed of the MG method. The second experiment is a simulation of the lid-driven cavity (LDC) which consists of dynamic fluid motion as well as known difficulties in the numerical approximation of this simulation. We find that the method is second-order accurate in the L_2 and L_∞ norms in agreement with Brown, Cortez and Minion's (2001) work. Finally a validation experiment is performed with the LDC simulations where IBM is used to place a cylinder within the computational domain, where the resulting fluid flow is studied. The simulation is written in C++, and data analysis is performed using MATLAB.

2D Taylor-Green Vortex

To verify our computational results we use the 2-dimensional Taylor-Green vortex, an analytical solution for the incompressible Navier-Stokes equations. For this numerical experiment we use the domain $\Omega = [0, 2\pi] \times [0, 2\pi]$ and parameters $Re = 1$ and CFL $= 1$ corresponding to $\Delta t = h$ where the simulation is run to the final time step $T = 5$. The solution takes the form:

$$u(x, y, t) = cos(x)sin(y)F(t) \qquad v(x, y, t) = -sin(x)cos(y)F(t)$$
$$p(x, y, t) = -\frac{1}{4}(cos(2x) + cos(2y))F(t)^2 \qquad F(t) = e^{-2\nu t}$$

With boundary conditions

$$u(0, y, t) = u(2\pi, y, t) = sin(y)F(t) \qquad u(x, 0, t) = u(x, 2\pi, t) = 0$$
$$v(0, y, t) = v(2\pi, y, t) = 0 \qquad v(x, 0, t) = v(x, 2\pi, t) = -sin(x)F(t)$$

The simulation is run for the intervals $n = 32, 64, 128$ and 256 and the error for the fields

TABLE 7.1. Errors In Velocity and Pressure at $T = 5$ for Different Mesh Sizes In Both the L_2 and L_∞ Norms

		$n = 32$	$n = 64$	$n = 128$	$n = 256$	Rate
u	L_2	1.27e-05	3.08e-06	7.62e-07	1.90e-07	2.01
	L_∞	3.23e-05	6.79e-06	1.70e-06	4.26e-07	2.01
v	L_2	1.27e-05	3.08e-06	7.62e-07	1.90e-07	2.01
	L_∞	3.29e-05	6.79e-06	1.70e-06	4.26e-07	2.01
p	L_2	1.61e-03	4.02e-04	1.00e-04	2.51e-05	2.00
	L_∞	3.17e-03	8.00e-04	2.01e-04	5.02e-05	2.00

u, v and p is recorded at $T = 5$. From Table 7.1 we see the L_2 and L_∞ norms show second-order rate of convergence for PMII, agreeing with the results from Brown, Cortez and Minion's (2001) work. Rate of convergence is measured using the errors e_1 and e_2 of a solution field at two different interval lengths h_1 and h_2 with rate q:

$$q = \frac{log(e_1/e_2)}{log(h_1/h_2)}$$

Multigrid Method Analysis

With the Taylor-Green vortex example we analyze the efficiency and performance of the MG method. Using the same parameters as above the simulation is run for a single iteration with varying intervals $n = 32, 64, 128, 256, 512$ and 1024. The average speed of one iteration of FMG is recorded, as well as the number of iterations and residues for the solution of the velocity and pressure fields; the solver iterates FMG until the residue meets a specified tolerance, in this case 1e-6.

In Figure 7.1 we observe the average time in seconds for an iteration of FMG. It is observed that the time for an iteration grows at a rate of $\mathcal{O}(n^2)$, indicating a similar order for the number of operations which is expected for our $\mathbb{R}^2$ discretization. Figure 7.2 shows the residue of the solution at successive iterations of FMG, it is observed that while smaller mesh sizes require more iterations to meet the tolerance, FMG is able to reduce the residue by a constant rate regardless of the mesh size.

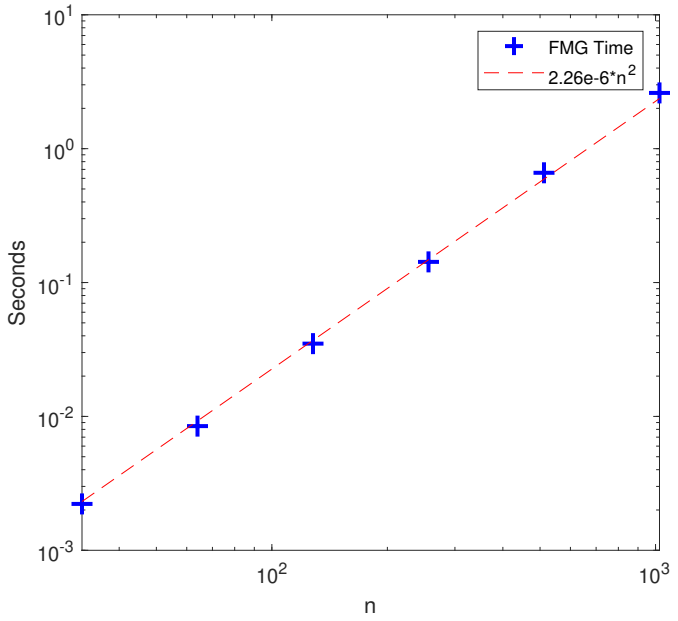

FIGURE 7.1. Average time for a single iteration of FMG with n intervals compared to an $\mathcal{O}(n^2)$ fitted line.

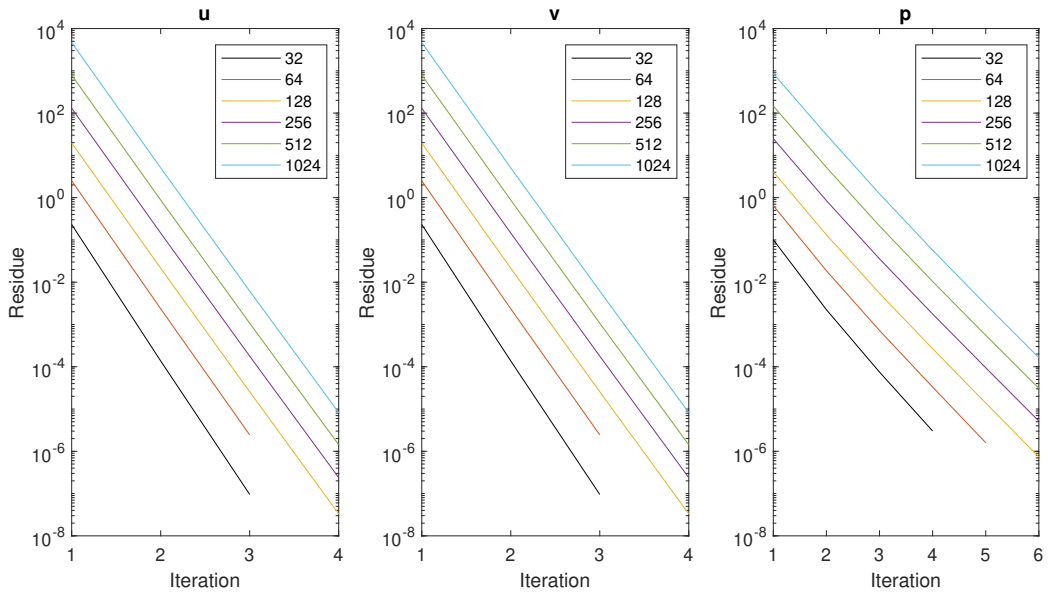

FIGURE 7.2. Residue of velocity and pressure solutions after successive iterations of FMG for different interval sizes.

Lid Driven Cavity

The lid-driven cavity (LDC) problem is a commonly used benchmark for the incompressible Navier-Stokes equations. Here we solve LDC for several values of Re and observe the behavior of the solutions including the formation of secondary vortices; we also analyze the singularities caused by discontinuities in the boundary conditions on the upper two corners of the domain. Finally IBM is used to simulate the effect the addition of one or more cylinders has on the fluid flow.

The LDC problem is performed on the domain $\Omega := [0,1] \times [0,1]$ with the following boundary conditions:

$$u(0,y,t) = u(1,y,t) = u(x,0,t) = 0 \qquad u(x,1,t) = 1$$

$$v(0,y,t) = v(1,y,t) = v(x,0,t) = 0 \qquad v(x,1,t) = 0$$

We use zero-velocity initial conditions and find solutions for the LDC problem at the values $Re = 100, 400, 1000$, and 3200 are plotted in Figure 7.3 using the mesh size $n = 64$ with $CFL = 0.8$ on the domain $[0,1] \times [0,1]$ until reaching steady-state. Here we can see the formation of secondary counter-rotating vortices in the bottom corners of the domain for the smaller Reynolds's numbers, and the formation of an additional counter-rotating vortex in the top left corner for the highest value of $Re = 3200$. These results are in good agreement to results from the work with Ghia, Ghia, and Shin (1982) using the same values for Re.

Corner Singularities

A known challenge in the solution of the Lid-Driven cavity is the presence of a pair of singularities in the upper corners of the domain. These corner singularities arise as a result of the discontinuity in boundary conditions of the top and sides of the cavity. Several investigations have been made into the nature and solution of these singularities (Burda, Novotnỳ, and Šístek 2012; Luchini 1991), which have found an asymptotic analytical solution for pressure which exists at the corners of the computational domain. The physical effect is a growth of pressure in

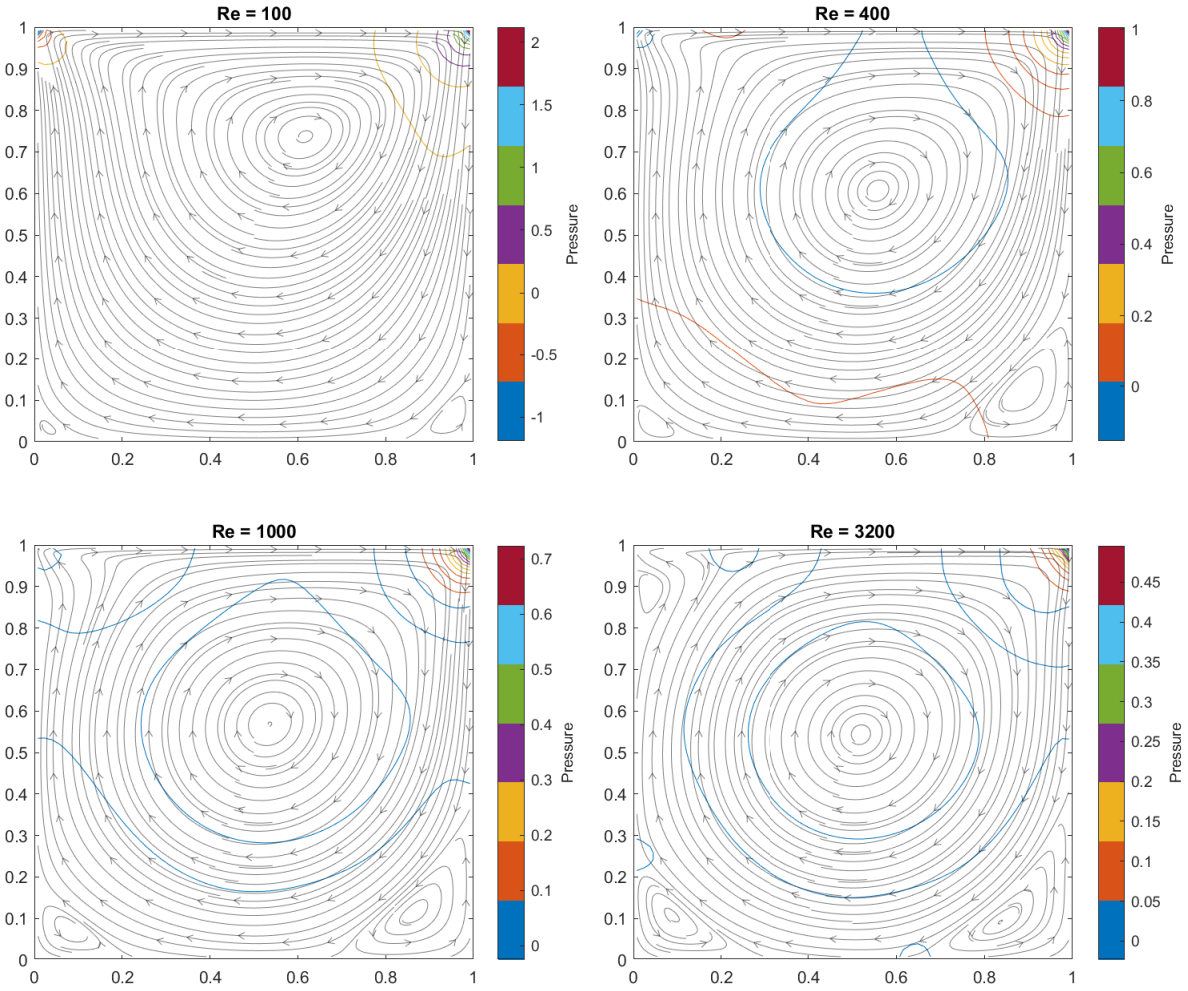

FIGURE 7.3. Solution for velocity and pressure of LDC for several values of Reynolds' number.

the respective corners, with the solution for pressure approaching $\pm\infty$ as the singularities are approached. Consequently the solution for pressure fails to satisfy the zero-flux Neumann boundary conditions given by (4.13) at these positions.

For this example we set $Re = 1$ and $CFL = 1$ corresponding to $\Delta t = h$ and take the solution at $T = 0.25$ for the intervals $n = 64, 128, 256, 512$, and 1024. From the work by Burda, Novotnỳ, and Šístek (2012) we have the following asymptotic analytical solution for the corner singularity at $(0, 1)$ with polar coordinates:

$$p(r,\theta)_{(0,1)} = \frac{\nu}{r}\frac{1}{\frac{\pi^2}{4}-1}(\pi \sin\theta - 2\cos\theta) \qquad \theta \in \left(-\frac{\pi}{2}, 0\right) \tag{7.1}$$

Using $\theta = -\frac{\pi}{4}$ we find the solution on the diagonal coming from the (0,1) singularity. Because of our choice of $Re = 1$, the (0,1) and (1,1) singularities are symmetric and so we can find the analytical solution of the (1,1) singularity via $p(r, -\frac{3\pi}{4})_{(1,1)} = -p(r, -\frac{\pi}{4})_{(0,1)}$. We compare our results with the analytical solutions in Figure 7.5; while the maximum value of the approximated solutions has some error (on the nodes closest to the boundary), the remainder of the solution is well approximated. An adapted mesh can be used in the region about these singularities to improve solution quality as seen with Burda, Novotnỳ, and Šístek's (2012) work - in our case a uniform mesh is used and may explain some of the error present in the approximations at the node nearest to the boundary. Figure 7.4 shows the maximum value of the approximations for the singularities at the various intervals - we can see the magnitude of the solution grows with rate $\mathcal{O}(h^{-1}) \propto \mathcal{O}(r^{-1})$, consistent with the analytical solution (7.1).

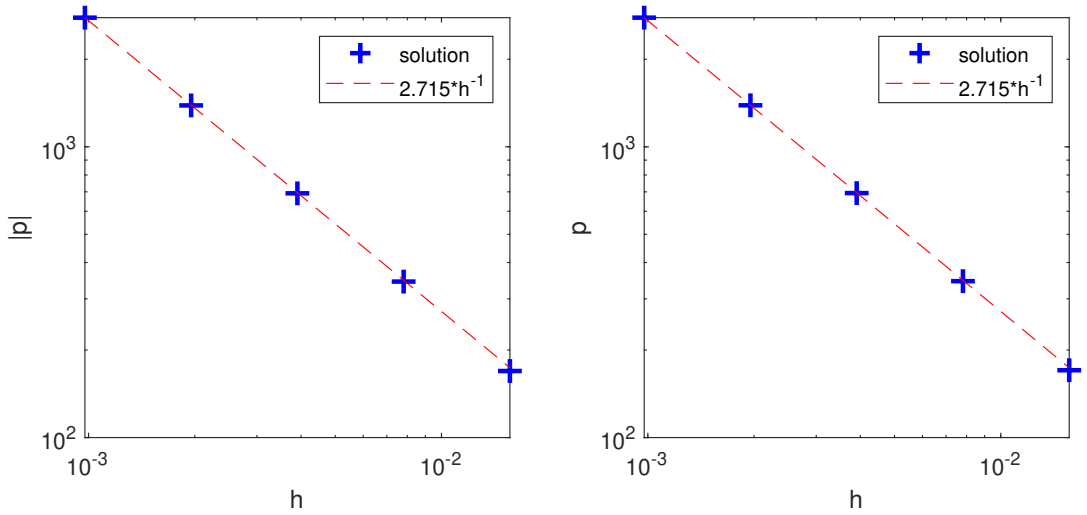

FIGURE 7.4. Absolute value of pressure at the $(0,1)$ singularity (left) and the value of pressure at the $(1,1)$ singularity (right).

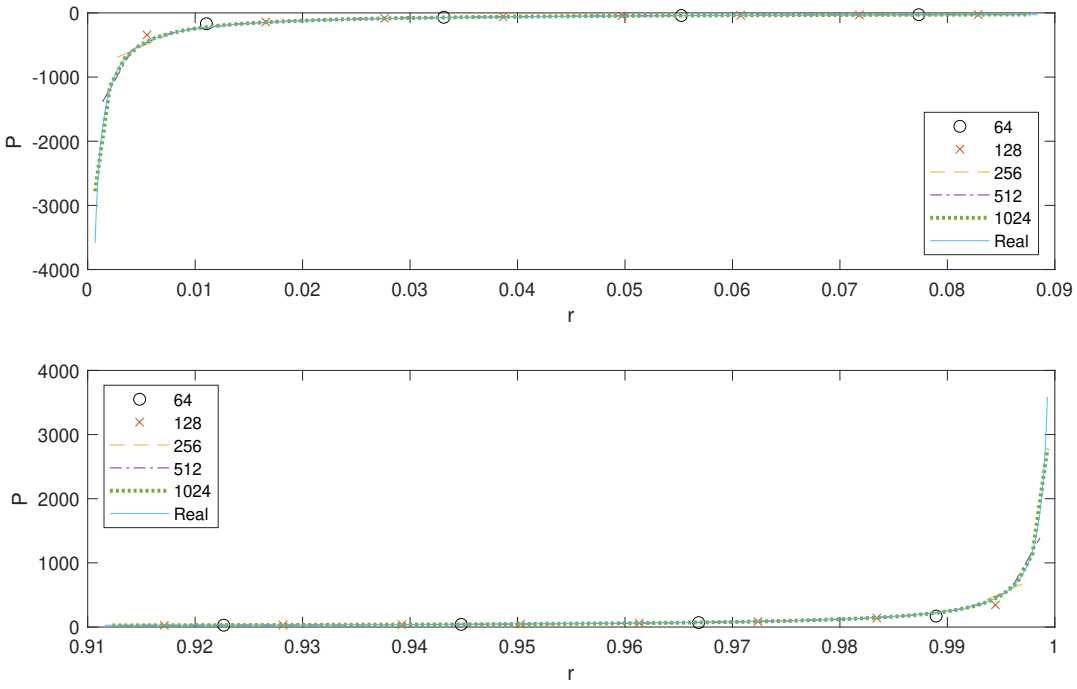

FIGURE 7.5. Comparison of pressure approximation of various mesh sizes with analytical solution of left singularity (top) and right singularity (bottom).

Immersed Boundary

This final experiment will perform several simulations of LDC using several values of Reynolds' number to validate the use of IBM by placing a cylinder within the domain and observing its effect on fluid flow. For this numerical experiment we use the same domain and boundary conditions as LDC, the domain is divided into 64^2 uniform squares and a cylinder is placed at $(x, y) = (0.7, 0.5)$ with radius $r = 0.15$. The simulation is run to $T = 50$ with $\Delta t = 0.8/n$ for the values $Re = 100$, 400, 1000, and 3200. These results can be seen at $T = 50$ in Figure 7.7.

Comparing these results to those for LDC in Figure 7.3 we can see cylinder displaces the central vortex and in case of the higher Reynolds' numbers this vortex is distorted and split. Similar to LDC counter-rotating vortices in the corners of the domain are observed, though their size is diminished compared to those in the LDC simulation. Finally we can observe fluid flow within the immersed boundary, these are a result of the application of IBM (the forcing term **F** is shown in Figure 7.6) and their direction of flow is counter to the exterior fluid flow.

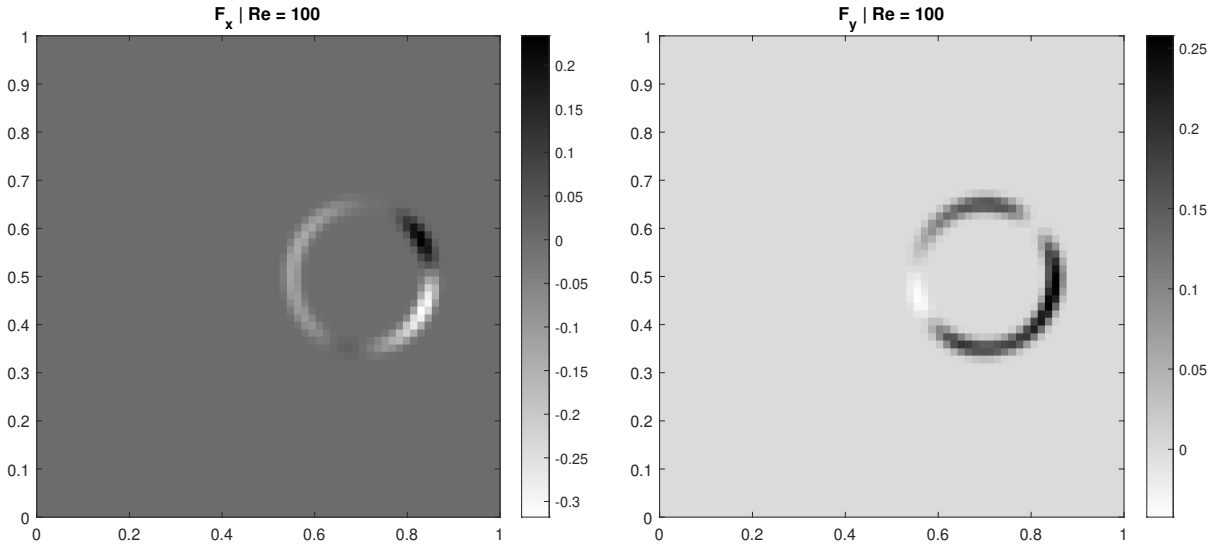

FIGURE 7.6. The x and y components of the forcing term F (5.1) in the Eulerian domain Ω for $Re = 100$.

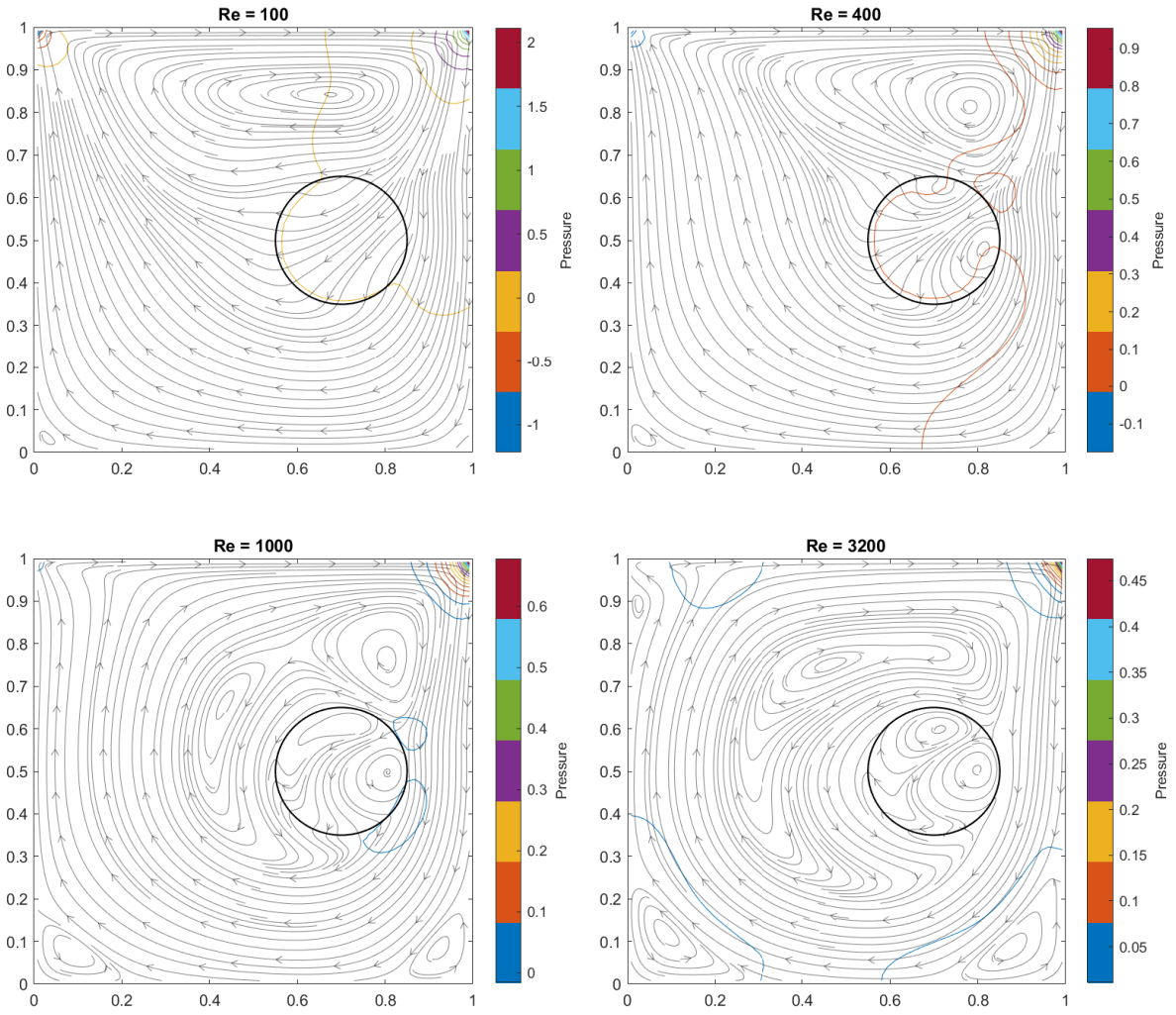

FIGURE 7.7. Solution for velocity and pressure of LDC with an immersed boundary using several values of Reynolds' number.

CHAPTER 8

CONCLUSION

In this work a method to solve the incompressible Navier-Stokes equations is introduced, enabling the simulation of complex and dynamic fluid flows. In particular we use a second-order projection method with the MAC grid to discretize the domain, and the multigrid method is employed to solve the resulting linear system. Additionally we have the ability to simulate the effect of various surfaces on the fluid flow via the implementation of the immersed boundary method. A series of numerical experiments are performed to verify the accuracy of the projection method, as well as validating the use of this simulation in conjunction with the immersed boundary method and without, for which we observe favorable results in all cases.

Continuing work on this project would involve the introduction of a series of coupled advection-diffusion-reaction equations to the incompressible Navier-Stokes system, as well as corresponding immersed boundaries. With these changes we would be able to observe the concentration of mixing and reacting chemical species in the domain.

APPENDIX

SIMULATION CODES

The codes used in this paper are available to download at:
https://github.com/Winguh/Navier-Stokes-Multigrid

REFERENCES

REFERENCES

Bell, John B., Phillip Colella, and Louis H. Howell. 1991. "An Efficient Second–Order Projection Method for Viscous Incompressible Flow." Paper presented at 10th Computational Fluid Dynamics Conference, Honolulu, HI, June 1991. https://doi.org/10.2514/6.1991-1560.

Botella, O., and R. Peyret. 1998. "Benchmark Spectral Results on the Lid-Driven Cavity Flow." *Computers & Fluids* 27, no. 4: 421–433. https://doi.org/10.1016/S0045-7930(98)00002-4.

Briggs, William L., Van Emden Henson, and Steve F. McCormick. 2000. *A Multigrid Tutorial*. SIAM. https://www.researchgate.net/publication/220690328_A_Multigrid_Tutorial_2nd_Edition.

Brown, David L., Ricardo Cortez, and Michael L. Minion. 2001. "Accurate Projection Methods for the Incompressible Navier–Stokes Equations." *Journal of Computational Physics* 168, no. 2: 464–499. https://doi.org/10.1006/jcph.2001.6715.

Burda, P., J. Novotnỳ, and J. Šístek. 2012. "Analytical Solution for Singularities in Stokes Flow and Applications to Finite Element Solution of Navier–Stokes Equations with High Precision." Paper presented at 7th International Conference on Computational Fluid Dynamics, Hawaii, HI, July 2012. https://www.iccfd.org/iccfd7/assets/pdf/papers/ICCFD7-3305_paper.pdf.

Chen, Long. 2018. "Programming of MAC Scheme for Stokes Equations." https://www.math.uci.edu/~chenlong/226/MACcode.pdf.

Ghia, U., Kirti N. Ghia, and C.T. Shin. 1982. "High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method." *Journal of Computational Physics* 48, no. 3: 387–411. https://doi.org/10.1016/0021-9991(82)90058-4.

Harlow, Francis H., and J. Eddie Welch. 1965. "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface." *The Physics of Fluids* 8, no. 12: 2182–2189. https://doi.org/10.1063/1.1761178.

Juric, Damir. 1996. "Computations of Phase Change." PhD diss., University of Michigan. ProQuest Dissertations Publishing. http://hdl.handle.net/2027.42/129865.

Luchini, Paolo. 1991. "Higher–Order Difference Approximations of the Navier-Stokes Equations." *International Journal for Numerical Methods in Fluids* 12, no. 5: 491–506. https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.1650120506.

McDonough, James M. 2007. "Lectures in Computational Fluid Dynamics of Incompressible Flow: Mathematics, Algorithms and Implementations." *Mechanical Engineering Textbook Gallery* 4. https://uknowledge.uky.edu/me_textbooks/4.

Peskin, Charles S. 1977. “Numerical Analysis of Blood Flow in the Heart.” *Journal of Computational Physics* 25, no. 3: 220–252. https://doi.org/10.1016/0021-9991(77)90100-0.

Przekwas, Andrzej, and Zhijian Chen. 2020. “Washing Hands and the Face May Reduce COVID-19 Infection.” *Medical Hypotheses* 144: 110261. https://doi.org/10.1016/j.mehy.2020.110261.

Rui, Zhang, Tu Guangbei, and Ling Jihong. 2008. “Study on Biological Contaminant Control Strategies Under Different Ventilation Models in Hospital Operating Room.” *Building and Environment* 43, no. 5: 793–803. https://doi.org/10.1016/j.buildenv.2007.01.018.

Silva, A.L.F., Lima E., A. Silveira-Neto, and J.J.R. Damasceno. 2003. “Numerical Simulation of Two-Dimensional Flows Over a Circular Cylinder Using the Immersed Boundary Method.” *Journal of Computational Physics* 189, no. 2: 351–370. https://doi.org/10.1016/S0021-9991(03)00214-6.

Tsega, Endalew Getnet, and V.K. Katiyar. 2018. “Finite Element Solution of the Two-dimensional Incompressible Navier-Stokes Equations Using MATLAB.” *Applications & Applied Mathematics* 13, no. 1: 535–565. https://www.pvamu.edu/aam/wp-content/uploads/sites/182/2018/06/35_R1065_Getnet_AAM_V13_1_pp_535_565_060118.pdf.

Vargas, Abel, and Rajat Mittal. 2004. “Aerodynamic Performance of Biological Airfoils.” Paper presented at 2nd AIAA Flow Control Conference, Portland, OR, June 2004. https://doi.org/10.2514/6.2004-2319.

Wackers, Jeroen, Barry Koren, Hoyte Christiaan Raven, A Van der Ploeg, A.R. Starke, G.B. Deng, Patrick Queutey, Michel Visonneau, Takanori Hino, and Kunihide Ohashi. 2011. “Free-Surface Viscous Flow Solution Methods for Ship Hydrodynamics.” *Archives of Computational Methods in Engineering* 18, no. 1: 1–41. https://doi.org/10.1007/s11831-011-9059-4.