Graduate J. Math. 3 (2018), 108 – 123

Numerical investigation on the projection method for the incompressible Navier-Stokes equations on MAC grid

Vorleak Yek

Abstract

The motion of a viscous fluid flow is described by the well-known Navier-Stokes equations. The Navier-Stokes equations contain the conservation laws of mass and momentum, and describe the spatial-temporal change of the fluid velocity field. This paper aims to investigate numerical solvers for the incompressible Navier-Stokes equations in two and three space dimensions. In particular, we focus on the second-order projection method introduced by Kim and Moin, which was extended from Chorin's first-order projection method. We impose periodic boundary conditions and apply Fourier-Spectral methods for the periodic boundary condition. Numerically, we discretize the system using centered differences scheme on Marker and Cell (MAC) grid spatially and the Crank-Nicolson scheme temporally. We then apply the fast Fourier transform to solve the resulting Poisson equations as sub-steps in the projection method. We will verify numerical accuracy and perform von Neumann stability analysis. In addition, we will simulate the particles' motion in the 2D and 3D fluid flow.

MSC 2010. Primary: 65M06; Secondary: 65F05.

1 Introduction

The Navier-Stokes equations, as the governing equations in fluid mechanics, describe the interaction and relation among quantities such as fluid velocity, pressure, and density. This set of equations can be applied everywhere in science and engineering fields, such as the study of swimming strategy of bacterial cells, the prediction of weather behavior, description of ocean currents, and the design of aerodynamically efficient aircraft wing [17, 2]. In addition, slight modifications of the equations can be used in other models such as electrodynamics, population dynamics, and aggregation swarming [13, 16, 8].

The incompressible Navier-Stokes equations, which describe the motion of an incompressible viscous fluid, can be written in a non-dimensional form as:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla P + \nu\Delta\mathbf{u} + \mathbf{f} \tag{1.1}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{1.2}$$

for $\mathbf{x} \in \Omega$ the fluid domain, where $\mathbf{u}(\mathbf{x},t)$ represents the velocity, $P(\mathbf{x},t)$ is the pressure, ν is a constant denoting the coefficient of kinematic viscosity, and $\mathbf{f}(\mathbf{x},t)$ is the external body force acting on the fluid. These equations were first introduced by Claude-Louis Navier and George Gabriel Stokes, with a given initial velocity field in Ω and certain boundary conditions on the fluid boundary $\partial\Omega$. Some examples of physically relevant boundary conditions are homogeneous or inhomogeneous Dirichlet, Neumann, and periodic boundary conditions [14]. In equation (1.1), the terms $\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}$ represent the material derivative, sometimes also called total derivative or convective derivative, which captures the change of a certain quantity, in our case velocity, for a fluid parcel moving with the fluid velocity. In other words, these terms describe the movement of the fluid flow field. The negative pressure gradient term $-\nabla P$ corresponds to fluid normal stress, and constrains the flow to prevent compressing. The dissipative viscous term $\nu\Delta\mathbf{u}$ corresponds to fluid shear stress. Thus, equation (1.1) of the Navier-Stokes equations states that the temporal change of the fluid flow field is determined by combined factors such as the normal stresses, shear stresses, and external forces.

Despite their wide range of practical uses, theoretical understanding of such equations remains incomplete. For example, the Clay Mathematics Institute has offered one million dollars prize to the significant breakthrough of the understanding of existence, uniqueness, and breakdown of solutions to the Navier-Stokes equations. Nevertheless, there are many methods being developed within the area of computational fluid dynamics in the last two decades to approximate the solution to the incompressible Navier-Stokes equations, under

finite difference, finite volume, or finite element discretization frameworks [14]. In many fluid-solid interaction applications, methods such as immersed boundary method, immersed interface method, ghost point and virtual node method have been developed. For the solution of the linear algebraic problem resulting from the discretization of the partial differential equations; there are techniques such as geometric and algebraic multigrid methods, Uzawa method, etc. [15, 9]. Many of these techniques are based on projection method.

This paper will focus on solving 2D and 3D incompressible Navier-Stokes equations using a second-order projection method introduced by Kim and Moin [12], which was an extension to Chorin's first-order projection method [5, 6]. We consider the case for periodic boundary condition for the purposes of simple simulation and use Fourier-spectral method. We will discretize the system using centered differences scheme on Marker and Cell (MAC) [1] grid spatially and the Crank-Nicolson scheme temporally. We then apply the fast Fourier transform to solve the Poisson equations that result from the projection method.

Here is the organization of this paper. In chapter 2, we will discuss the derivation of the Navier-Stokes equations using mass and momentum conservation laws. In chapter 3, we will talk about the first-order projection method introduced by Chorin and the second-order projection method introduced by Kim and Moin. In chapter 4, we will study the spatial discretization scheme using centered differences on 2D and 3D MAC grids. In two dimensions, we consider the solution $\mathbf{u} = (u, v)$ for which u is the velocity in the x-direction and v is the velocity in the y-direction. In three dimensions, we consider the solution $\mathbf{u} = (u, v, w)$ for which u is the velocity in the x-direction, v is the velocity in the y-direction, and w is the velocity in the z-direction. Then, in chapter 5, we will discuss how to numerically solve the Navier-Stokes equations using the second-order projection method that Kim and Moin proposed with fast Fourier transform. In chapter 6, we will numerically verify that the method gives second-order accuracy in both the velocity field and the pressure. Then, in chapter 7, we will show the stability analysis of a simplified version of Navier-Stokes equations without the nonlinear advection term using von Neumann stability analysis technique. In chapter 8, we will simulate the particles' motion with the fluid flow by interpolation. We will use MATLAB to implement the numerical method and solve for the approximation solution to the Navier-Stokes equations. Finally, in chapter 9, we will discuss the result and the conclusion.

2 Derivation

The Navier-Stokes equations are derived from physical laws such as conservation of mass and momentum. Equation (1.1) is called the Cauchy momentum equation, and it arises from Newton's second law. Equation (1.2) is the continuity equation in the case of incompressible fluid, and is derived from the conservation law of mass.

2.1 The Continuity Equation

Consider a computational region Ω filled with the incompressible viscous fluid. Let $\mathcal{R}$ be an arbitrarily small volume of fluid enclosed by the surface $\partial\mathcal{R}$. Let $\mathrm{d}\omega$ and $\mathrm{d}a$ denote the volume element of $\mathcal{R}$ and surface element on $\partial\mathcal{R}$, respectively. Since $\mathrm{d}a$ is small and the fluid is continuous, we can approximate the velocity $\mathbf{u}$ and the density ρ of the fluid in $\mathcal{R}$ as constant. Let $\mathbf{n}$ be the unit outer normal to $\partial\mathcal{R}$. Then, over a small time interval $\mathrm{d}t$, the volume $\mathrm{d}l$ of the fluid that flows through $\mathrm{d}a$ is:

$$\mathrm{d}l = \mathbf{u} \cdot \mathbf{n}\mathrm{d}a\mathrm{d}t.$$

We denote the mass increase in $\mathcal{R}$ due to the flux through $\mathrm{d}a$ within a small time interval $\mathrm{d}t$ as $\mathrm{d}m$, then we know that

$$\mathrm{d}m = -\rho\mathrm{d}l = -\rho\mathbf{u} \cdot \mathbf{n}\mathrm{d}a\mathrm{d}t.$$

Hence the mass inflow through $\partial\mathcal{R}$ is:

$$(\text{inflow of mass through } \partial\mathcal{R}) = -\iint_{\partial\mathcal{R}} \rho\mathbf{u} \cdot \mathbf{n}\mathrm{d}a\mathrm{d}t.$$

Now we denote the total mass within $\mathcal{R}$ to be M, and we assume that the mass is conserved. Then, the total increase of mass would be equal to the total inflow through $\partial\mathcal{R}$. This yields the equation:

$$\frac{\mathrm{d}M}{\mathrm{d}t} = -\iint_{\partial\mathcal{R}} \rho\mathbf{u} \cdot \mathbf{n}\mathrm{d}a.$$

By the divergence theorem,

$$\frac{\mathrm{d}M}{dt} = -\iint_{\partial\mathcal{R}} \rho\mathbf{u} \cdot \mathbf{n}\mathrm{d}a = -\iiint_{\mathcal{R}} (\nabla \cdot (\rho\mathbf{u}))\mathrm{d}\omega. \quad (2.1)$$

On the other hand, we know that

$$M = \iiint_{\mathcal{R}} \rho\mathrm{d}\omega.$$

By passing the time derivative into the integral and arranging the terms, we obtain:

$$\iiint_{\mathcal{R}} \left[\frac{\partial\rho}{\partial t} + \nabla \cdot (\rho\mathbf{u})\right] \mathrm{d}\omega = 0$$

The fact that $\mathcal{R}$ is arbitrary and the integrand is a continuous function gives us the continuity equation

$$\frac{\partial\rho}{\partial t} + \nabla \cdot (\rho\mathbf{u}) = 0.$$

By the assumption that the fluid is incompressible (i.e., the density is constant), the temporal and spatial variation of ρ vanish. Thus, we obtain the divergence free condition of the Navier-Stokes equations (1.2):

$$\nabla \cdot \mathbf{u} = 0.$$

2.2 The Cauchy Momentum Equation

Next, we derive the Cauchy momentum equation using the conservation law of momentum. Consider a domain Ω of incompressible viscous fluid, and let $\mathcal{R}$ be an arbitrarily small volume in Ω. Then, the total force $\mathbf{F}$ acted on $\mathcal{R}$ is composed of body force $\mathbf{f}$ exerted inside $\mathcal{R}$ and surface force, such as traction force, exerted on $\partial\mathcal{R}$. To compute the surface force, we define a second order tensor (or, $n \times n$ matrix) field $\boldsymbol{\sigma}$ called stress tensor at every spatial location $\mathbf{x} \in \Omega$. This tensor relates the surface force density $\boldsymbol{\psi}$ experienced at a material surface with its unit normal $\mathbf{n}$ via the relation $\boldsymbol{\psi} = \boldsymbol{\sigma} \cdot \mathbf{n}$. The Cauchy stress tensor is commonly accepted as an approximation to the stress tensor for linear responses of small material deformation. As we will see in this section, surface forces can be described as the spatial derivatives of the Cauchy stress tensor, which is composed of the normal stresses and the shear stresses.

For simplicity of calculation, we fix the domain $\Omega \subseteq \mathbb{R}^3$ as a rectangular region. Moreover, we assume $\mathcal{R} = \left[x_0 - \frac{dx}{2}, x_0 + \frac{dx}{2}\right] \times \left[y_0 - \frac{dy}{2}, y_0 + \frac{dy}{2}\right] \times \left[z_0 - \frac{dz}{2}, z_0 + \frac{dz}{2}\right]$ to be a small rectangular volume element of Ω. Then, dx, dy, and dz denote the size length of $\mathcal{R}$ in the x, y, and z directions, respectively. Let $\mathrm{d}v$ be the volume of $\mathcal{R}$, i.e., $\mathrm{d}v = dxdydz$. Note that $\partial\mathcal{R}$ is composed of six faces: front face $\left(x = x_0 + \frac{dx}{2}\right)$, back face $\left(x = x_0 - \frac{dx}{2}\right)$, left face $\left(y = y_0 - \frac{dy}{2}\right)$, right face$\left(y = y_0 + \frac{dy}{2}\right)$, bottom face $\left(z = z_0 - \frac{dz}{2}\right)$, and top face $\left(z = z_0 + \frac{dz}{2}\right)$. The surface forces acting on the front and back faces of $\partial\mathcal{R}$ are given by:

$$\begin{aligned}\boldsymbol{\psi}_{front} dydz &= \begin{bmatrix}\sigma_{xx} & \sigma_{xy} & \sigma_{xz}\\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz}\\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz}\end{bmatrix}\Bigg|_{(x_0+\frac{dx}{2},y,z)} \cdot \begin{bmatrix}1\\0\\0\end{bmatrix} dydz \qquad (2.2)\\ &= \begin{bmatrix}\sigma_{xx}(x_0+\frac{dx}{2},y,z)\\ \sigma_{yx}(x_0+\frac{dx}{2},y,z)\\ \sigma_{zx}(x_0+\frac{dx}{2},y,z)\end{bmatrix} dydz,\\ \boldsymbol{\psi}_{back} dydz &= \begin{bmatrix}\sigma_{xx} & \sigma_{xy} & \sigma_{xz}\\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz}\\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz}\end{bmatrix}\Bigg|_{(x_0-\frac{dx}{2},y,z)} \cdot \begin{bmatrix}-1\\0\\0\end{bmatrix} dydz \qquad (2.3)\\ &= -\begin{bmatrix}\sigma_{xx}(x_0-\frac{dx}{2},y,z)\\ \sigma_{yx}(x_0-\frac{dx}{2},y,z)\\ \sigma_{zx}(x_0-\frac{dx}{2},y,z)\end{bmatrix} dydz.\end{aligned}$$

Combined together, the traction force exerted on the front and back faces of $\partial\mathcal{R}$ is given by

$$\left(\begin{bmatrix}\sigma_{xx}(x_0+\frac{dx}{2},y,z)\\ \sigma_{yx}(x_0+\frac{dx}{2},y,z)\\ \sigma_{zx}(x_0+\frac{dx}{2},y,z)\end{bmatrix} dydz - \begin{bmatrix}\sigma_{xx}(x_0-\frac{dx}{2},y,z)\\ \sigma_{yx}(x_0-\frac{dx}{2},y,z)\\ \sigma_{zx}(x_0-\frac{dx}{2},y,z)\end{bmatrix} dydz\right)\frac{dx}{dx} \qquad (2.4)$$

$$= \frac{\partial}{\partial x}\begin{bmatrix}\sigma_{xx}\\ \sigma_{yx}\\ \sigma_{zx}\end{bmatrix}\mathrm{d}v.$$

Similarly, the traction force exerted on the right and left faces of $\partial\mathcal{R}$ is:

$$\left(\begin{bmatrix}\sigma_{xy}(x,y_0+\frac{dy}{2},z)\\ \sigma_{yy}(x,y_0+\frac{dy}{2},z)\\ \sigma_{zy}(x,y_0+\frac{dy}{2},z)\end{bmatrix} dxdz - \begin{bmatrix}\sigma_{xy}(x,y_0-\frac{dy}{2},z)\\ \sigma_{yy}(x,y_0-\frac{dy}{2},z)\\ \sigma_{zy}(x,y_0-\frac{dy}{2},z)\end{bmatrix} dxdz\right)\frac{dy}{dy} \qquad (2.5)$$

$$= \frac{\partial}{\partial y}\begin{bmatrix}\sigma_{xy}\\ \sigma_{yy}\\ \sigma_{zy}\end{bmatrix}\mathrm{d}v.$$

Moreover, the traction force exerted on the top and bottom faces of $\partial\mathcal{R}$ is:

$$\left(\begin{bmatrix}\sigma_{xz}(x,y,z_0+\frac{dz}{2})\\ \sigma_{yz}(x,y,z_0+\frac{dz}{2})\\ \sigma_{zz}(x,y,z_0+\frac{dz}{2})\end{bmatrix} dxdy - \begin{bmatrix}\sigma_{xz}(x,y,z_0-\frac{dz}{2})\\ \sigma_{yz}(x,y,z_0-\frac{dz}{2})\\ \sigma_{zz}(x,y,z_0-\frac{dz}{2})\end{bmatrix} dxdy\right)\frac{dz}{dz} \qquad (2.6)$$

$$= \frac{\partial}{\partial z}\begin{bmatrix}\sigma_{xz}\\ \sigma_{yz}\\ \sigma_{zz}\end{bmatrix}\mathrm{d}v.$$

Hence, the total force acting on $\mathcal{R}$ is:

$$\begin{aligned}\mathbf{F} &= \begin{bmatrix}F_x\\F_y\\F_z\end{bmatrix} \qquad (2.7)\\ &= \begin{bmatrix}f_x\\f_y\\f_z\end{bmatrix}\mathrm{d}v + \frac{\partial}{\partial x}\begin{bmatrix}\sigma_{xx}\\ \sigma_{yx}\\ \sigma_{zx}\end{bmatrix}\mathrm{d}v + \frac{\partial}{\partial y}\begin{bmatrix}\sigma_{xy}\\ \sigma_{yy}\\ \sigma_{zy}\end{bmatrix}\mathrm{d}v + \frac{\partial}{\partial z}\begin{bmatrix}\sigma_{xz}\\ \sigma_{yz}\\ \sigma_{zz}\end{bmatrix}\mathrm{d}v.\end{aligned}$$

For a general dimension n, each component F_i of $\mathbf{F}$ can be written as:

$$F_i = f_i \mathrm{d}v + \sum_{j=1}^{n} \frac{\partial \sigma_{ij}}{\partial x_j}\mathrm{d}v. \qquad (2.8)$$

By Newton's second law, we have:

$$F_i = ma_i = m\frac{\mathrm{d}u_i(\mathbf{x}(t),t)}{\mathrm{d}t} = \rho\frac{\mathrm{d}u_i(\mathbf{x}(t),t)}{\mathrm{d}t}\mathrm{d}v.$$

Applying the chain rule, we get

$$\begin{aligned}F_i &= \rho\left[\sum_{j=1}^{n}\frac{\partial u_i}{\partial x_j}\frac{dx_j}{dt} + \frac{\partial u_i}{\partial t}\right]\mathrm{d}v\\ &= \rho\left[\sum_{j=1}^{n}\frac{\partial u_i}{\partial x_j}u_j + \frac{\partial u_i}{\partial t}\right]\mathrm{d}v. \qquad (2.9)\end{aligned}$$

By equation (2.8) and equation (2.9) we arrive at

$$f_i + \sum_{j=1}^{n} \frac{\partial \sigma_{ij}}{\partial x_j} = \rho \left[\sum_{j=1}^{n} \frac{\partial u_i}{\partial x_j} u_j + \frac{\partial u_i}{\partial t} \right]. \tag{2.10}$$

The stress tensor can be decomposed into the *normal stresses* (- PI) and the *deviatoric stresses* (τ) through the componentwise relation $\sigma_{ij} = \tau_{ij} - P\delta_{ij}$. The pressure P is the average of all the unit normal stresses, which can be defined by $P = -\frac{1}{n}\sum_{i=1}^{n} \sigma_{ii}$. The deviatoric stresses τ_{ij} is the deviation of the normal stresses from the stress tensor. By the Cauchy's infinitesimal strain tensor and the assumption that the fluid is Newtonian, we define the deviatoric stress as

$$\tau_{ij} = \nu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right). \tag{2.11}$$

Now, substituting σ_{ij} and $\rho = 1$ into equation (2.10), we have:

$$\begin{aligned}
&\frac{\partial u_i}{\partial t} + \sum_{j=1}^{n} \frac{\partial u_i}{\partial x_j} u_j \\
&= f_i + \sum_{j=1}^{n} \frac{\partial}{\partial x_j} (\tau_{ij} - P\delta_{ij}) \\
&= f_i + \sum_{j=1}^{n} \frac{\partial \tau_{ij}}{\partial x_j} - \sum_{j=1}^{n} \frac{\partial (P\delta_{ij})}{\partial x_j} \\
&= f_i - \frac{\partial P}{\partial x_i} + \sum_{j=1}^{n} \frac{\partial \tau_{ij}}{\partial x_j} \\
&= f_i - \frac{\partial P}{\partial x_i} + \nu \sum_{j=1}^{n} \frac{\partial}{\partial x_j} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \\
&= f_i - \frac{\partial P}{\partial x_i} + \nu \sum_{j=1}^{n} \frac{\partial^2 u_i}{\partial x_j^2} + \nu \sum_{j=1}^{n} \left(\frac{\partial}{\partial x_j} \frac{\partial u_j}{\partial x_i} \right).
\end{aligned}$$

Rearrange the orders of derivatives and summation on the last term of the last line, and by the incompressibility condition, we have:

$$\frac{\partial u_i}{\partial t} + \sum_{j=1}^{n} \frac{\partial u_i}{\partial x_j} u_j = f_i - \frac{\partial P}{\partial x_i} + \nu \sum_{j=1}^{n} \frac{\partial^2 u_i}{\partial x_j^2}.$$

In vector form, the Cauchy momentum equation of the Navier-Stokes equations can be written as:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{f} - \nabla P + \nu \Delta \mathbf{u}.$$

3 Projection Method

The fractional-step projection method first introduced by Chorin is the most frequently used method for solving the *primitive variable* incompressible Navier-Stokes equations [3, 14]. It was the first numerical scheme enabling a cost-effective solution of time-dependent problems in three-dimensional space [14]. The *primitive variable* method is based on formulating the Navier-Stokes equations in terms of velocity and pressure. Thus, the projection method aims to solve the incompressible Navier-stokes equations by dealing with the velocity and the pressure terms as a two-stage fractional step scheme. The first-half of the algorithm involves solving for the intermediate velocity while the second-half of algorithm is solving for the pressure [11]. The motivation behind the projection method is the Helmholtz-Hodge decomposition theorem, as stated below. This theorem and its proof can be found in many places such as [11].

Theorem 3.1. Helmholtz-Hodge Decomposition *Let Ω be a smooth, simply connected bounded domain with a smooth boundary $\partial\Omega$. Then, any smooth vector field $\boldsymbol{\mu}^*$ defined in Ω can be decomposed into a unique orthogonal decomposition*

$$\boldsymbol{\mu}^* = \boldsymbol{\mu} + \nabla \xi,$$

where $\boldsymbol{\mu}$ is a divergence-free vector field, and ξ is a scalar potential function.

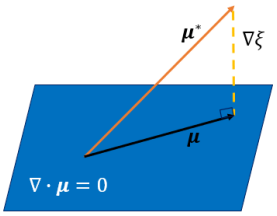

Fig. 1: A schematic view of Helmholtz-Hodge decomposition, the projection of a vector into a divergence-free manifold.

The projection method was first introduced by Alexandre Chorin [5, 6] as a first-order accurate numerical method for solving the Navier-Stokes equations. It soon became popular and widely applied in computational fluid dynamics, and was improved and generalized for higher order accuracy and various boundary conditions. Now, there are many versions of projection method. The one we use in this paper is the second-order accurate version developed by Kim and Moin [12].

The projection method breaks down the update in each numerical time-step to several sub-steps. The idea behind it is that one can first solve the momentum equation (1.1) for an intermediate velocity or a half-time step solution $\mathbf{u}^{n+\frac{1}{2}}$, not requiring divergence-free velocity. Then, apply the Helmholtz-Hodge decomposition theorem, so that the intermediate velocity $\mathbf{u}^*$ can be projected onto a solenoidal field to yield a discretely divergence-free velocity and a gradient field. Thereby, we may enforce the divergence-free constraint and update the pressure from solving a Poisson equation [3,

11]. In the whole process, the main computation steps hinge on developing a fast Poisson solver. Moreover, there are multiple ways to deal with the nonlinear convection term. Chorin used explicit forward temporal discretization. Here, we follow Kim and Moin to use Crank-Nicolson with the second-order Adams-Bashforth extrapolation for the nonlinear term.

3.1 Chorin's First Order Method

Based on Chorin's projection method, the numerical solution to the incompressible Navier-Stokes equations can be computed by the following steps:

Step 1: Compute an intermediate velocity $\mathbf{u}^*$ by omitting the pressure gradient

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\mathrm{d}t} = -(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \nu\Delta\mathbf{u}^n + \mathbf{f}^n.$$

Step 2: Solve for the pressure term P^{n+1}

$$\Delta P^{n+1} = \frac{\nabla \cdot \mathbf{u}^*}{\mathrm{d}t}.$$

Step 3: Compute the divergence-free velocity $\mathbf{u}^{n+1}$

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \mathrm{d}t\nabla P^{n+1}.$$

Chorin's projection algorithm only gives first-order accuracy in the velocity field and the pressure [3, 5, 11]. Kim and Moin extended it to a second-order projection algorithm by applying the Crank-Nicolson scheme to the momentum equation, which obtains the following:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\mathrm{d}t} + [(\mathbf{u} \cdot \nabla)\mathbf{u}]^{n+\frac{1}{2}} = -\nabla\mathbf{P}^{n+\frac{1}{2}} + \frac{\nu}{2}(\Delta\mathbf{u}^{n+1} + \Delta\mathbf{u}^n) + \mathbf{f}^{n+\frac{1}{2}}. \quad (3.1)$$

Now, the nonlinear term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ at a half-time step is computed explicitly using the second-order Adams-Bashforth extrapolation [3, 5, 11]. The next step is to solve for an intermediate velocity $\mathbf{u}^*$ in equation (3.1) by omitting the pressure gradient. By applying the Helmholtz-Hodge decomposition theorem, which states that any smooth velocity field $\mathbf{u}^*$ can be decomposed into a unique orthogonal decomposition of gradient- and divergence-free fields, the intermediate velocity $\mathbf{u}^*$ can be expressed as:

$$\mathbf{u}^* = \mathbf{u}^{n+1} + \mathrm{d}t\nabla\varphi,$$

where φ is a scalar potential function. Note that $\mathrm{d}t$ is not required, but it helps clarifying the discussion of the pressure. Taking the divergence on both sides of the equation, we obtain $\mathrm{d}t\Delta\varphi = \nabla \cdot \mathbf{u}^*$ since $\nabla \cdot \mathbf{u}^{n+1} = 0$ by the *incompressibility* condition. After solving for φ, we can compute the divergence-free velocity $\mathbf{u}^{n+1}$ using the intermediate $\mathbf{u}^*$ and the gradient of φ.

3.2 Kim and Moin Second-Order Implicit

Based on Kim and Moin, we can solve the incompressible Navier-Stokes equations using the projection method by breaking it into the following three steps:

Step 1: Solve for the intermediate velocity $\mathbf{u}^*$ (with pressure gradient omitted)

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\mathrm{d}t} + \mathbf{g}^{n+\frac{1}{2}} = \frac{\nu}{2}(\Delta\mathbf{u}^n + \Delta\mathbf{u}^*) + \mathbf{f}^{n+\frac{1}{2}}, \quad (3.2)$$

where $\mathbf{g}^{n+\frac{1}{2}}$ represents some approximation to the nonlinear term, $\mathbf{u} \cdot \nabla\mathbf{u}$, at the half time level. We assume the case of periodic boundary condition.

Step 2: Perform the projection and solve for the scalar potential function φ^{n+1}

$$\mathrm{d}t\Delta\varphi^{n+1} = \nabla \cdot \mathbf{u}^*. \quad (3.3)$$

Note that this is equivalent to solving the divergence of the equation below

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\mathrm{d}t} = -\nabla\varphi^{n+1}.$$

Step 3: Compute the divergence-free velocity $\mathbf{u}^{n+1}$

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \mathrm{d}t\nabla\varphi^{n+1}.$$

This projection algorithm will give a second-order accuracy in both the velocity field and the pressure [3, 11, 12]. In addition, we can find the relation between the pressure P and φ by substituting

$$\mathbf{u}^* = \mathbf{u}^{n+1} + \mathrm{d}t\nabla\varphi^{n+1}$$

into equation (3.2). Hence, we have the following:

$$\begin{aligned}
&\frac{\mathbf{u}^{n+1} + \mathrm{d}t\nabla\varphi^{n+1} - \mathbf{u}^n}{\mathrm{d}t} + \mathbf{g}^{n+\frac{1}{2}} \\
&\qquad = \frac{\nu}{2}\left(\Delta\mathbf{u}^n + \Delta\left(\mathbf{u}^{n+1} + \mathrm{d}t\nabla\varphi^{n+1}\right)\right) + \mathbf{f}^{n+\frac{1}{2}} \\
&\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{dt} + \mathbf{g}^{n+\frac{1}{2}} \\
&\qquad = -\nabla\varphi^{n+1} + \frac{\nu}{2}\left(\Delta\mathbf{u}^n + \Delta\left(\mathbf{u}^{n+1} + \mathrm{d}t\nabla\varphi^{n+1}\right)\right) + \mathbf{f}^{n+\frac{1}{2}} \\
&\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{dt} + \mathbf{g}^{n+\frac{1}{2}} \\
&\qquad = -\nabla\left(\varphi^{n+1} - \frac{\nu}{2}\mathrm{d}t\Delta\varphi^{n+1}\right) + \frac{\nu}{2}\left(\Delta\mathbf{u}^n + \Delta\mathbf{u}^{n+1}\right) + \mathbf{f}^{n+\frac{1}{2}}
\end{aligned}$$

Therefore, φ is related to pressure by $P^{n+\frac{1}{2}} = \varphi^{n+1} - \frac{\mathrm{d}t\nu}{2}\Delta\varphi^{n+1}$. This pressure relation is due to the Crank-Nicolson scheme applied to the momentum equation.

4 Spatial Discretization on Marker and Cell Grid

Regular grids store all the unknown variables at the same locations, which leads to an instability called the *odd-even decoupling* or spurious oscillations in the pressure field when solving the incompressible Navier-Stokes equations [1]. This instability is due to the relationship between the pressure and velocity. Notice that in the time-discretized equations, the pressure depends on the first derivatives of the velocity, and the velocity depends on the first derivatives of the pressure. Consequently, using centered differences for spatial discretization and labeling the nodes odd or even, we see that the pressure at even nodes depends on the velocity at odd nodes. The pressure at odd nodes depends on the velocity at even nodes. Similarly, this applies the same way for the velocity. This instability can be avoided by discretizing spatial variables on a Marker and Cell (MAC) grid.

4.1 MAC Grid in 2D

Staggered-grid scheme or MAC grid automatically couples the grid scale pressure, which avoids odd-even decoupling solutions and prevents the pressure oscillations [1]. The MAC grid also allows us to obtain the exact divergence-free solution, and it can be viewed this way. Instead of using the grid points at the corners of each cell like in a standard grid, we define u on the vertical edge midpoints along the x-direction, v on the horizontal edge midpoints along the y-direction, and P on the center of each cell as shown in Fig. 2. In 2D, we assume the computational domain to be rectangular $\Omega = [0, L_1] \times [0, L_2]$, and there are n_1, n_2 discretization points in each of the x and y spatial direction where $\mathrm{d}x = \frac{L_1}{n_1}$ and $\mathrm{d}y = \frac{L_2}{n_2}$. Furthermore, we denote

$$\mathbf{x}_{i,j} = \left(\left(i + \tfrac{1}{2}\right)\tfrac{L_1}{n_1}, \left(j + \tfrac{1}{2}\right)\tfrac{L_2}{n_2}\right),$$

and we define $u_{i,j}, v_{i,j}$, and $\varphi_{i,j}$ as the following:

$$u_{i,j} = u\left(\mathbf{x}_{i-\frac{1}{2},j}\right),\ v_{i,j} = v\left(\mathbf{x}_{i,j-\frac{1}{2}}\right),\ \text{and}\ \varphi_{i,j} = \varphi\left(\mathbf{x}_{i,j}\right).$$

Similarly, $u^*_{i,j}$ is defined at the same grid points as $u_{i,j}$, and $v^*_{i,j}$ is defined at the same grid points as $v_{i,j}$.

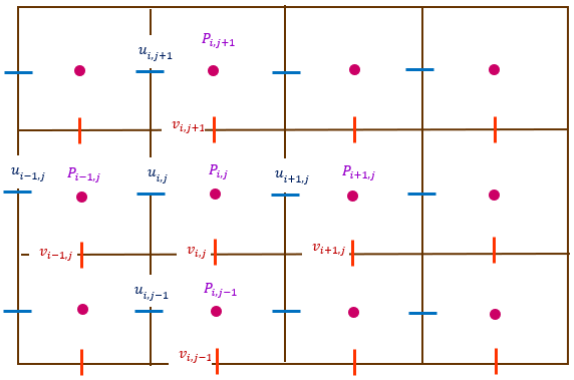

Fig. 2: A schematic view of the MAC grid in 2D, with locations and indices of the variables u, v, and P.

4.2 MAC Grid in 3D

In three-dimension, the u velocity grid points are located at the center of the back and front faces of the cube along the x-direction, v velocity grid points are located at the center of the left and right faces of the cube along the y-direction, and w velocity grid points are located at the center of the bottom and top faces of the cube along the z-direction as shown in Fig. 3. The pressure is located at the center of each cube. In 3D, we assume the computational domain to be rectangular $\Omega = [0, L_1] \times [0, L_2] \times [0, L_3]$, and there are n_1, n_2, n_3 discretization points in each of the x, y, and z spatial direction where $\mathrm{d}x = \frac{L_1}{n_1}$, $\mathrm{d}y = \frac{L_2}{n_2}$, and $\mathrm{d}z = \frac{L_3}{n_3}$. Furthermore, we denote spatial directions

$$\mathbf{x}_{i,j,k} = \left(\left(i + \tfrac{1}{2}\right)\tfrac{L_1}{n_1}, \left(j + \tfrac{1}{2}\right)\tfrac{L_2}{n_2}, \left(k + \tfrac{1}{2}\right)\tfrac{L_3}{n_3}\right),$$

and we define $u_{i,j,k}, v_{i,j,k}, w_{i,j,k}$, and $\varphi_{i,j,k}$ as below:

$$u_{i,j,k} = u\left(\mathbf{x}_{i-\frac{1}{2},j,k}\right),\ v_{i,j,k} = v\left(\mathbf{x}_{i,j-\frac{1}{2},k}\right),$$
$$w_{i,j,k} = w\left(\mathbf{x}_{i,j,k-\frac{1}{2}}\right),\ \text{and}\ \varphi_{i,j,k} = \varphi\left(\mathbf{x}_{i,j,k}\right).$$

Similarly, $u^*_{i,j}$, $v^*_{i,j}$ and $w^*_{i,j}$ are defined at the same grid points as $u_{i,j}$, $v_{i,j}$ and $w_{i,j}$, respectively.

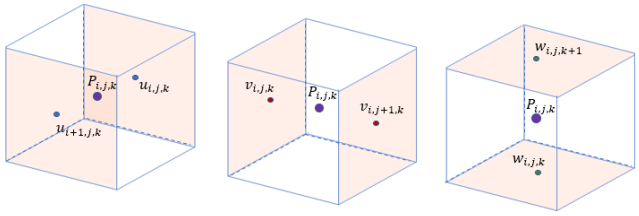

Fig. 3: A schematic view of the MAC grid in 3D, with locations and indices of the variables u, v, w, and P.

4.3 Discrete Gradient

We will use centered differences to compute the first derivatives of the velocity components u and v in the second-order Adams-Bashforth extrapolation and φ in the third step of the projection algorithm. In 2D, the first derivatives of u with respect to x and y are approximated by:

$$(u_x)_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2\mathrm{d}x},$$
$$(u_y)_{i,j} = \frac{u_{i,j+1} - u_{i,j-1}}{2\mathrm{d}y}.$$

The first derivatives of v with respect to x and y are computed similar to the first derivatives of u with respect to x and y. In 3D, the first derivatives of u with respect to x, y, and z are approximated by:

$$(u_x)_{i,j,k} = \frac{u_{i+1,j,k} - u_{i-1,j,k}}{2\mathrm{d}x},$$
$$(u_y)_{i,j,k} = \frac{u_{i,j+1,k} - u_{i,j-1,k}}{2\mathrm{d}y},$$
$$(u_z)_{i,j,k} = \frac{u_{i,j,k+1} - u_{i,j,k-1}}{2\mathrm{d}z}.$$

The first derivatives of v and w with respect to x, y, and z can be computed similar to the derivatives of u with respect to x, y, and z.

Now, let $\varphi_{i,j}$ be the scalar quantities located at the center of a cell. Then, in 2D, the first derivatives of φ with respect to x and y approximated at $u_{i,j}$ and $v_{i,j}$ are given by:

$$(\varphi_x)_{i-\frac{1}{2},j} = \frac{\varphi_{i,j} - \varphi_{i-1,j}}{\mathrm{d}x},$$
$$(\varphi_y)_{i,j-\frac{1}{2}} = \frac{\varphi_{i,j} - \varphi_{i,j-1}}{\mathrm{d}y}.$$

In 3D, the approximation to the first derivatives of φ with respect to x, y, and z are given by:

$$(\varphi_x)_{i-\frac{1}{2},j,k} = \frac{\varphi_{i,j,k} - \varphi_{i-1,j,k}}{\mathrm{d}x},$$
$$(\varphi_y)_{i,j-\frac{1}{2},k} = \frac{\varphi_{i,j,k} - \varphi_{i,j-1,k}}{\mathrm{d}y},$$
$$(\varphi_z)_{i,j,k-\frac{1}{2}} = \frac{\varphi_{i,j,k} - \varphi_{i,j,k-1}}{\mathrm{d}z}.$$

Note that φ_x is approximated at $\mathbf{x}_{i-\frac{1}{2},j,k}$, which is the same grid point as where $u_{i,j,k}$ is being approximated. Similarly, φ_y is approximated at $\mathbf{x}_{i,j-\frac{1}{2},k}$, which is the same grid point as where $v_{i,j,k}$ is being approximated. Lastly, φ_z is approximated at $\mathbf{x}_{i,j,k-\frac{1}{2}}$, which is the same grid point as where $w_{i,j,k}$ is being approximated.

4.4 Discrete Divergence

We will also use centered differences to compute the divergence of $\mathbf{u}^*$ in the second step of the projection algorithm. The approximation of the gradient of $\mathbf{u}^*$ is given by:

$$(u^*_x)_{i+\frac{1}{2},j} = \frac{u^*_{i+1,j} - u^*_{i,j}}{\mathrm{d}x},$$
$$(v^*_y)_{i,j+\frac{1}{2}} = \frac{v^*_{i,j+1} - v^*_{i,j}}{\mathrm{d}y}.$$

Note that u^*_x is approximated at $\mathbf{x}_{i+\frac{1}{2},j}$ and v^*_y is approximated at $\mathbf{x}_{i,j+\frac{1}{2}}$, which are the same grid points as where $\varphi_{i,j}$ is being approximated. Thus, in 2D, the discrete divergence of $\mathbf{u}^*$ approximates at the center of a cell is:

$$\begin{aligned}(\nabla \cdot \mathbf{u}^*)_{i,j} &= (u^*_x)_{i+\frac{1}{2},j} + (v^*_y)_{i,j+\frac{1}{2}} \\ &= \frac{u^*_{i+1,j} - u^*_{i,j}}{\mathrm{d}x} + \frac{v^*_{i,j+1} - v^*_{i,j}}{\mathrm{d}y}.\end{aligned}$$

Similarly, the discrete divergence of the velocity $\mathbf{u}^*$ in 3D is given by:

$$\begin{aligned}&(\nabla \cdot \mathbf{u}^*)_{i,j,k} = (u^*_x)_{i+\frac{1}{2},j,k} + (v^*_y)_{i,j+\frac{1}{2},k} + (w^*_z)_{i,j,k+\frac{1}{2}} \\ &= \frac{u^*_{i+1,j,k} - u^*_{i,j,k}}{\mathrm{d}x} + \frac{v^*_{i,j+1,k} - v^*_{i,j,k}}{\mathrm{d}y} + \frac{w^*_{i,j,k+1} - w^*_{i,j,k}}{\mathrm{d}z}.\end{aligned}$$

4.5 Discrete Laplacian

The Laplacian of φ in this pressure equation $P^{n+\frac{1}{2}} = \varphi^{n+1} - \frac{\mathrm{d}t\nu}{2}\Delta\varphi^{n+1}$ is computed by using the 5-stencil centered differences. The second derivatives of φ with respect to x and y are given by:

$$(\varphi_{xx})_{i,j} = \frac{\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}}{\mathrm{d}x^2},$$
$$(\varphi_{yy})_{i,j} = \frac{\varphi_{i,j+1} - 2\varphi_{i,j} + \varphi_{i,j-1}}{\mathrm{d}y^2}.$$

Thus, the discrete Laplacian of φ in 2D is given by:

$$\begin{aligned}&(\Delta\varphi)_{i,j} \\ &= (\varphi_{xx})_{i,j} + (\varphi_{yy})_{i,j} \\ &= \frac{\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}}{\mathrm{d}x^2} + \frac{\varphi_{i,j+1} - 2\varphi_{i,j} + \varphi_{i,j-1}}{\mathrm{d}y^2}.\end{aligned}$$

Note that in the case of $\mathrm{d}x = \mathrm{d}y$, we have:

$$\begin{aligned}(\Delta\varphi)_{i,j} &= (\varphi_{xx})_{i,j} + (\varphi_{yy})_{i,j} \\ &= \frac{\varphi_{i+1,j} + \varphi_{i,j+1} - 4\varphi_{i,j} + \varphi_{i-1,j} + \varphi_{i,j-1}}{\mathrm{d}x^2}.\end{aligned}$$

In 3D, the Laplacian of φ is computed using the 7-stencil centered differences, which is given by:

$$\begin{aligned}(\Delta\varphi)_{i,j,k} &= (\varphi_{xx})_{i,j,k} + (\varphi_{yy})_{i,j,k} + (\varphi_{zz})_{i,j,k} \\ &= \frac{\varphi_{i+1,j,k} - 2\varphi_{i,j,k} + \varphi_{i-1,j,k}}{\mathrm{d}x^2} \\ &+ \frac{\varphi_{i,j+1,k} - 2\varphi_{i,j,k} + \varphi_{i,j-1,k}}{\mathrm{d}y^2} \\ &+ \frac{\varphi_{i,j,k+1} - 2\varphi_{i,j,k} + \varphi_{i,j,k-1}}{\mathrm{d}z^2}.\end{aligned}$$

5 Numerical Method

There are many ways to discretize the Poisson equation. For examples, one can use finite difference, finite element, or (pseudo-) spectral methods [10]. Here, we consider the (pseudo-) spectral method where the solution ϕ and the source term f are evaluated on MAC grid points. We will use fast Fourier transform to solve equation (3.2) in step 1 and equation (3.3) in step 2 of the projection method introduced by Kim and Moin. From this chapter on, we will change the superscript time-step n to ι in order to avoid the confusion with the Fourier mode n. In addition, we will change $\mathbf{x}_{i,j}$ to $\mathbf{x}_{j,k} = \left(\left(j+\frac{1}{2}\right)\frac{L_1}{n_1}, \left(k+\frac{1}{2}\right)\frac{L_2}{n_2}\right)$ in order to avoid the confusion with the complex number i.

5.1 Fourier Transform

The *Poisson* equation is an elliptic partial differential equation, which can be written in the form:

$$\Delta\phi = f,$$

where the solution ϕ and f are smooth real functions on Ω. Consider the 2D Poisson equation above with periodic boundary conditions and smooth functions ϕ and f. Then, on domain $[0, L_1] \times [0, L_2]$, the function f can be written as:

$$f(x,y) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} \widehat{f}_{m,n} e^{\frac{i2\pi mx}{L_1}} e^{\frac{i2\pi ny}{L_2}}. \tag{5.1}$$

Similarly, the function ϕ can be expressed as:

$$\phi(x,y) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} \widehat{\phi}_{m,n} e^{\frac{i2\pi mx}{L_1}} e^{\frac{i2\pi ny}{L_2}}.$$

Here $\widehat{f}_{m,n}$ and $\widehat{\phi}_{m,n}$ are the Fourier coefficients corresponding to the (m,n)-th mode, and they can be calculated by:

$$\widehat{f}_{m,n} = \frac{1}{L_1L_2}\int_0^{L_1}\int_0^{L_2} f(x,y) e^{-\frac{i2\pi mx}{L_1}} e^{-\frac{i2\pi ny}{L_2}} \mathrm{d}x\mathrm{d}y,$$
$$\widehat{\phi}_{m,n} = \frac{1}{L_1L_2}\int_0^{L_1}\int_0^{L_2} \phi(x,y) e^{-\frac{i2\pi mx}{L_1}} e^{-\frac{i2\pi ny}{L_2}} \mathrm{d}x\mathrm{d}y.$$

Now, taking second derivatives of ϕ with respect to x and y, we get:

$$\phi_{xx}(x,y) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} \frac{-4\pi^2m^2}{L_1^2}\widehat{\phi}_{m,n} e^{\frac{i2\pi mx}{L_1}} e^{\frac{i2\pi ny}{L_2}},$$
$$\phi_{yy}(x,y) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} \frac{-4\pi^2n^2}{L_2^2}\widehat{\phi}_{m,n} e^{\frac{i2\pi mx}{L_1}} e^{\frac{i2\pi ny}{L_2}}.$$

Thus, Laplacian of ϕ is given by:

$$\begin{aligned}&\Delta\phi(x,y) \\ &= \phi_{xx}(x,y) + \phi_{yy}(x,y) \\ &= \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} -4\pi^2\left(\frac{m^2}{L_1^2}+\frac{n^2}{L_2^2}\right)\widehat{\phi}_{m,n} e^{\frac{i2\pi mx}{L_1}} e^{\frac{i2\pi ny}{L_2}}.\end{aligned} \tag{5.2}$$

Since $\Delta\phi = f$, by equation (5.1) and equation (5.2) we obtain:

$$\begin{aligned}&\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} -4\pi^2\left(\frac{m^2}{L_1^2}+\frac{n^2}{L_2^2}\right)\widehat{\phi}_{m,n} e^{\frac{i2\pi mx}{L_1}} e^{\frac{i2\pi ny}{L_2}} \\ &= \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} \widehat{f}_{m,n} e^{\frac{i2\pi mx}{L_1}} e^{\frac{i2\pi ny}{L_2}}.\end{aligned}$$

So, for each (m,n)-th mode, we have the following:

$$\widehat{\Delta\phi}_{m,n} = \widehat{f}_{m,n}$$
$$-4\pi^2\left(\frac{m^2}{L_1^2}+\frac{n^2}{L_2^2}\right)\widehat{\phi}_{m,n} = \widehat{f}_{m,n} \tag{5.3}$$
$$\widehat{\phi}_{m,n} = -\frac{1}{4\pi^2\left(\frac{m^2}{L_1^2}+\frac{n^2}{L_2^2}\right)}\widehat{f}_{m,n}. \tag{5.4}$$

5.2 Fast Fourier Transform

At each grid point, the point value $f_{j,k} = f(\mathbf{x}_{j,k})$ can be transformed using the discrete Fourier transform:

$$f_{j,k} = \sum_{m=1}^{n_1}\sum_{n=1}^{n_2} \widehat{f}_{m,n} e^{\frac{i2\pi mj}{n_1}} e^{\frac{i2\pi nk}{n_2}} \tag{5.5}$$
$$\widehat{f}_{m,n} = \frac{1}{n_1n_2}\sum_{j=1}^{n_1}\sum_{k=1}^{n_2} f_{j,k} e^{-\frac{i2\pi mj}{n_1}} e^{-\frac{i2\pi nk}{n_2}}. \tag{5.6}$$

Equation (5.6) is called the discrete Fourier transform (DFT). It is used for calculating the Fourier coefficient $\widehat{f}_{m,n}$ in equation (5.5), which is known as the inverse discrete Fourier transform (IDFT). The efficiency of the pseudo-spectral algorithm for solving the Poisson equation comes from the use of fast Fourier transform (FFT) method to compute DFT [4, 10]. Fast Fourier transform involves the Cooley-Tukey algorithm, which is an efficient way of calculating the constants in the interpolating trigonometric polynomial [4]. It takes order of $\mathcal{O}(n^2)$ floating point operations to compute DFT directly, but only takes order $\mathcal{O}(n\log n)$ operations to compute DFT using FFT given that the number of data points being used can be factored into powers of two. Thus, the computational complexity of computing the DFT using the FFT will reduce from $\mathcal{O}(n^2)$ to $\mathcal{O}(n\log n)$ [4]. More details about FFT and its algorithm can be found in the *Numerical Analysis* textbook written by Burden and Faires (2011).

5.3 Solving the Momentum Equation in Step 1

Solving the projection method introduced by Kim and Moin, we first approximate the nonlinear term $\mathbf{u}\cdot\nabla\mathbf{u}$ at a half-time step using the second-order explicit Adams-Bashforth extrapolation. Then, we will use the fast Fourier transform to solve for $\mathbf{u}^*$. Here is the algorithm. Applying the Crank-Nicolson scheme to the momentum equation, omitting the pressure term, we obtain:

$$\frac{\mathbf{u}^* - \mathbf{u}^\iota}{\mathrm{d}t} + \mathbf{g}^{\iota+\frac{1}{2}} = \frac{\nu}{2}(\Delta\mathbf{u}^\iota + \Delta\mathbf{u}^*) + \mathbf{f}^{\iota+\frac{1}{2}}, \tag{5.7}$$

where $\mathbf{g}^{\iota+\frac{1}{2}}$ is the approximation of the nonlinear term computed by the Adams-Bashforth extrapolation. The second-order explicit Adams-Bashforth extrapolation is computed by the following scheme:

$$\mathbf{g}^{\iota+\frac{1}{2}} = \frac{3}{2}\left(\mathbf{u}^\iota\cdot\nabla\mathbf{u}^\iota\right) - \frac{1}{2}(\mathbf{u}^{\iota-1}\cdot\nabla\mathbf{u}^{\iota-1}) + O(\mathrm{d}t^2).$$

We will use centered differences to compute $\mathbf{g}^{\iota+\frac{1}{2}}$. Then, we will solve for the unknown $\mathbf{u}^*$ by applying FFT method. So, rearranging equation (5.7) for the unknown $\mathbf{u}^*$, we have:

$$\begin{aligned}(I - \frac{\nu\mathrm{d}t}{2}\Delta)\mathbf{u}^* &= \mathbf{u}^\iota + \frac{\nu\mathrm{d}t}{2}(\Delta\mathbf{u}^\iota) + \mathrm{d}t\mathbf{f}^{\iota+\frac{1}{2}} - \mathrm{d}t\mathbf{g}^{\iota+\frac{1}{2}}\\ &= (I + \frac{\nu\mathrm{d}t}{2}\Delta)\mathbf{u}^\iota + \mathrm{d}t\mathbf{f}^{\iota+\frac{1}{2}} - \mathrm{d}t\mathbf{g}^{\iota+\frac{1}{2}}.\end{aligned} \tag{5.8}$$

Applying the Fourier transform and use the result in equation (5.3), we see that for each (m,n)-th mode, the equation (5.8) can be written as:

$$\left(1 + \frac{\nu\mathrm{d}t}{2}\left(4\pi^2\left(\frac{m^2}{L_1^2} + \frac{n^2}{L_2^2}\right)\right)\right)\widehat{\mathbf{u}^*}_{m,n} = \left(1 - \frac{\nu\mathrm{d}t}{2}\left(4\pi^2\left(\frac{m^2}{L_1^2} + \frac{n^2}{L_2^2}\right)\right)\right)\widehat{\mathbf{u}^\iota}_{m,n} + \mathrm{d}t\widehat{\mathbf{f}}^{\iota+\frac{1}{2}}_{m,n} - \mathrm{d}t\widehat{\mathbf{g}}^{\iota+\frac{1}{2}}_{m,n}.$$

Let $\widehat{\mathbf{k}}_{m,n} = \left(1 - \frac{\nu\mathrm{d}t}{2}\left(4\pi^2\left(\frac{m^2}{L_1^2} + \frac{n^2}{L_2^2}\right)\right)\right)\widehat{\mathbf{u}^\iota}_{m,n} + \mathrm{d}t\widehat{\mathbf{f}}^{\iota+\frac{1}{2}}_{m,n} - \mathrm{d}t\widehat{\mathbf{g}}^{\iota+\frac{1}{2}}_{m,n}$. Then, in 2-Dimensional space, each (m,n)-th Fourier mode is given by:

$$\widehat{\mathbf{u}^*}_{m,n} = \frac{1}{1 + \frac{\nu\mathrm{d}t}{2}\left(4\pi^2\left(\frac{m^2}{L_1^2} + \frac{n^2}{L_2^2}\right)\right)}\widehat{\mathbf{k}}_{m,n}.$$

Similarly, in 3-Dimensional space, each (m,n,l)-th Fourier mode is given by:

$$\widehat{\mathbf{u}^*}_{m,n,l} = \frac{1}{1 + \frac{\nu\mathrm{d}t}{2}\left(4\pi^2\left(\frac{m^2}{L_1^2} + \frac{n^2}{L_2^2} + \frac{l^2}{L_3^2}\right)\right)}\widehat{\mathbf{k}}_{m,n,l},$$

where

$$\widehat{\mathbf{k}}_{m,n,l} = \left(1 - \frac{\nu\mathrm{d}t}{2}\left(4\pi^2\left(\frac{m^2}{L_1^2} + \frac{n^2}{L_2^2} + \frac{l^2}{L_3^2}\right)\right)\right)\widehat{\mathbf{u}^\iota}_{m,n,l} + \mathrm{d}t\widehat{\mathbf{f}}^{\iota+\frac{1}{2}}_{m,n,l} - \mathrm{d}t\widehat{\mathbf{g}}^{\iota+\frac{1}{2}}_{m,n,l}.$$

We will use the FFT to compute the Fourier coefficient $\widehat{\mathbf{u}^*}_{m,n}$ in 2D and $\widehat{\mathbf{u}^*}_{m,n,l}$ in 3D. Finally, to obtain the solution $\mathbf{u}^*$, we will take the inverse fast Fourier transform (IFFT).

5.4 Solving Poisson Equation in Step 2

Once $\mathbf{u}^*$ is known, we need to project it onto a solenoidal field by applying the Helmholtz-Hodge decomposition theorem. This left us with the Poisson equation to solve:

$$\mathrm{d}t\Delta\varphi^{\iota+1} = \nabla\cdot\mathbf{u}^*. \tag{5.9}$$

To solve equation (5.9), we first need to approximate the divergence of $\mathbf{u}^*$ using centered differences scheme. Then, we can apply the Fourier transform to the Poisson equation $\Delta\varphi = f$, where $f = \frac{1}{\mathrm{d}t}\nabla\cdot\mathbf{u}^*$. Using the result from equation (5.4), we obtain the following Fourier coefficient $\widehat{\varphi}_{m,n}$ at (m,n)-th mode:

$$\widehat{\varphi}_{m,n} = -\frac{1}{4\pi^2\left(\frac{m^2}{L_1^2} + \frac{n^2}{L_2^2}\right)}\widehat{f}_{m,n}.$$

Similarly, in a three dimensional space, we obtain the following Fourier coefficient $\widehat{\varphi}_{m,n,l}$ at each (m,n,l)-th mode:

$$\widehat{\varphi}_{m,n,l} = -\frac{1}{4\pi^2\left(\frac{m^2}{L_1^2} + \frac{n^2}{L_2^2} + \frac{l^2}{L_3^2}\right)}\widehat{f}_{m,n,l}.$$

Similar to step 1, we will use the FFT to compute the Fourier coefficient $\widehat{\varphi}^{\iota+1}$ at each (m,n)-th and (m,n,l)-th modes. Then, we will take the IFFT to obtain the solution $\varphi^{\iota+1}$.

5.5 Computing the Divergence-Free Velocity in Step 3

Once we obtain $\varphi^{\iota+1}$ in step 2, we can compute the divergence-free velocity at time $\iota+1$ by the following equation:

$$\mathbf{u}^{\iota+1} = \mathbf{u}^* - \mathrm{d}t\nabla\varphi^{\iota+1}.$$

We approximate the gradient of $\varphi^{\iota+1}$ using centered differences scheme. Note that the above equation results from the Helmholtz-Hodge decomposition theorem, and it ensures that $\mathbf{u}^{\iota+1}$ will be divergence-free.

6 Order of Accuracy

The authors in [3] used normal mode analysis to show that the projection method introduced by Kim and Moin is second-order accurate in both the velocity and the pressure. For this reason, we expect that the Crank-Nicolson projection method developed by Kim and Moin

using second-order explicit Adams-Bashforth for the nonlinear term approximation, and centered differences for spatial discretization will give us a second-order accurate approximation in both the velocity and the pressure. We will verify that the method is second-order accurate by the method of manufactured solutions, which consists in imposing an exact solution $\mathbf{u}(\mathbf{x}, t)$ for velocity and $P(\mathbf{x}, t)$ for pressure. Then, we substitute its gradient, Laplacian, and divergence into the momentum equation to solve for the natural force flow term $f(\mathbf{x}, t)$. Once the force, pressure, initial velocity, and the viscosity are determined, we can numerically solve the equation for an approximate solution at a final time-step T. We can verify that the method is second-order accurate by refining the number of grid points from N to $2N$ and reducing the time-step by a half (from $\mathrm{d}t$ to $\frac{\mathrm{d}t}{2}$) each time we compute the approximate solution. Then, the order of accuracy of the method is given by the rate of convergence of the numerical solution to the exact solution.

Consider the case of a 1D problem. Let N be the number of grid points, h be the step-size, and denote E to be the error of maximum norm of the differences between the exact solution and the numerical solution. Then, the rate of convergence is defined as:

$$E(h) = C \cdot h^{\alpha},$$
$$E\left(\frac{h}{2}\right) = C \cdot \left(\frac{h}{2}\right)^{\alpha},$$

where C is a constant. Now, dividing the errors above, we get:

$$\frac{E(h)}{E(\frac{h}{2})} = \frac{C \cdot h^{\alpha}}{C \cdot \left(\frac{h}{2}\right)^{\alpha}} = 2^{\alpha}.$$

Taking log on both sides of the equation above, we obtain:

$$\log\left(\frac{E(h)}{E(\frac{h}{2})}\right) = \alpha \log(2).$$

Therefore, the rate of convergence is as fast as the power α defines below:

$$\alpha = \frac{\log\left(\frac{E(h)}{E(\frac{h}{2})}\right)}{\log(2)}. \tag{6.1}$$

If we decrease h by a half (i.e., increase N by 2) and also decrease the time-size step $\mathrm{d}t$ by a half, then we can numerically verify that the method is second-order accurate by confirming the power α is approximately two.

6.1 Second-Order Accuracy of Velocity and Pressure in 2D

In two dimensions, consider the solution $\mathbf{u} = (u, v)$, where u is the velocity in the x-direction and v is the velocity in the y-direction. We defined the exact solution $\mathbf{u}$ as below:

$$u(x, y, t) = cos(2\pi(x - t))sin(4\pi y), \tag{6.2}$$
$$v(x, y, t) = -\frac{1}{2}sin(2\pi(x - t))cos(4\pi y). \tag{6.3}$$

Then, the divergence-free velocity condition is satisfied since $u_x + v_y = 0$, and we have periodic boundary conditions. Note that the velocity u has one period of cosine function in the x-direction and two periods of sine function in the y-direction. On the other hand, the velocity v has one period of sine function in the x-direction and two periods of cosine in the y-direction. For simplicity, we choose the time t to be related only in the x-direction. We let the pressure to be the following function:

$$P(x, y, t) = cos(2\pi(x - t))sin(4\pi y).$$

Now, we set the time $t = 0$ in equation (6.2) and equation (6.3) to get the initial velocity. We will compare the error between the numerical solution and the exact solution at the final time $T = 0.2$ using a time step of size $\mathrm{d}t = 0.01$ and $\nu = 0.001$ in the region $[0, 1] \times [0, 1]$. As we can see from Tab. 2, the power α of the velocities u, v, and pressure P is approximately two as we doubled the number of grid points in both x and y-direction and decreased $\mathrm{d}t$ by a half. In two dimensions, Tab. 2 verified that the method gives a second-order accuracy in both the velocity field and the pressure.

Tab. 1: Second-Order Accuracy of Velocity and Pressure in 2D

$N_1 \times N_2$	$\mathrm{d}t$	error u	error v	error P
64×64	0.01	2.56e-3	3.93e-3	1.79e-3
128×128	0.005	6.55e-4	9.13e-4	4.79e-4
256×256	0.0025	1.66e-4	2.19e-4	1.23e-4
512×512	0.00125	4.18e-5	5.36e-5	3.13e-5

Tab. 2: Second-Order Accuracy of Velocity and Pressure in 2D

$N_1 \times N_2$	$\mathrm{d}t$	rate u	rate v	rate P
64×64	0.01	-	-	-
128×128	0.005	1.96	2.11	1.91
256×256	0.0025	1.98	2.06	1.96
512×512	0.00125	1.99	2.03	1.98

6.2 Second-Order Accuracy of Velocity and Pressure in 3D

In three dimensions, consider the solution $\mathbf{u} = (u, v, w)$, where u is the velocity in the x-direction, v is the velocity in the y-direction, and w is the velocity in the

z-direction. We defined the exact solution $\mathbf{u}$ as:

$$u(x,y,z,t) = cos(2\pi x)sin(4\pi(y-t))cos(6\pi z), \quad (6.4)$$
$$v(x,y,z,t) = sin(2\pi x)cos(4\pi(y-t))cos(6\pi z), \quad (6.5)$$
$$w(x,y,z,t) = sin(2\pi x)sin(4\pi(y-t))sin(6\pi z). \quad (6.6)$$

The velocity field $\mathbf{u}$ satisfies the divergence-free condition, i.e., $u_x + v_y + w_z = 0$, and we have periodic boundary conditions. The pressure is defined by the function:

$$P(x,y,z,t) = cos(2\pi(x-t))sin(4\pi y)sin(6\pi z).$$

The initial velocity is defined by setting the time $t = 0$ in equations (6.4)-(6.6). We will compute the numerical solution at final time $T = 0.1$ using a time step of size $\mathrm{d}t = 0.02$ and $\nu = 0.001$ in the domain $[0,1] \times [0,1] \times [0,1]$. From Tab. 4 and Tab. 5 we see that the rates of convergence in both the velocity $\mathbf{u}$ and the pressure P are quadratic. In other words, α is approximately two when we double the number of grid points in x, y, and z-direction and decrease $\mathrm{d}t$ by a half. In three dimensions, this verified that the method is second-order accurate in both the velocity field and the pressure.

Tab. 3: Second-Order Accuracy of Velocity in 3D

$N_1 \times N_2 \times N_3$	$\mathrm{d}t$	error u	error v	error w
64×64×64	0.02	2.47e-2	1.67e-2	1.33e-2
128×128×128	0.01	6.07e-3	3.92e-3	3.19e-3
256×256×256	0.005	1.51e-3	9.53e-4	7.80e-4
512×512×512	0.0025	3.76e-4	2.38e-4	1.93e-4

Tab. 4: Second-Order Accuracy of Velocity in 3D

$N_1 \times N_2 \times N_3$	$\mathrm{d}t$	rate u	rate v	rate w
64×64×64	-	-	-	
128×128×128	0.01	2.02	2.09	2.06
256×256×256	0.005	2.01	2.04	2.03
512×512×512	0.0025	2.00	2.00	2.01

Tab. 5: Second-Order Accuracy of Pressure in 3D

$N_1 \times N_2 \times N_3$	$\mathrm{d}t$	error P	rate P
64×64×64	0.02	3.48e-2	-
128×128×128	0.01	8.95e-3	1.96
256×256×256	0.005	2.25e-3	1.99
512×512×512	0.0025	5.62e-4	2.00

In both 2D and 3D, we can see that the rates of convergence of the velocity and the pressure are quadratic. Therefore, we have numerically verified that the method is second-order accurate in both the velocity and the pressure.

7 Stability Analysis

We will analyze the stability of the Fourier-spectral method with the projection method using von Neumann stability analysis. Since our formulation approximates $(\mathbf{u} \cdot \nabla)\mathbf{u}$ at a half-time step using the second-order Adams-Bashforth extrapolation, this can reduce or eliminate the dependence of the stability of the method on the magnitude of viscosity [3]. We only consider the stability in the case when the magnitude of the velocity is very small so that the nonlinear term can be neglected. In other words, we are considering the stability of the time-dependent Stokes equations.

The Crank-Nicolson scheme with the projection method for the time-dependent Stokes flow that we are considering for stability analysis can be formulated:

$$\mathbf{u}^* - \mathbf{u}^\iota = \frac{\nu \mathrm{d}t}{2}\left(\Delta \mathbf{u}^* + \Delta \mathbf{u}^\iota\right) \quad (7.1)$$
$$\Delta \varphi^{\iota+1} = \nabla \cdot \mathbf{u}^* \quad (7.2)$$
$$\mathbf{u}^{\iota+1} = \mathbf{u}^* - \nabla \varphi^{\iota+1}. \quad (7.3)$$

Here, we consider the region of $[0,1] \times [0,1]$ with periodic boundary conditions, so that $L_1 = L_2 = 1$. The discrete values $u_{j,k}$, $v_{j,k}$, and $\varphi_{j,k}$ on the MAC grid can be expressed as:

$$\begin{aligned} u_{j,k} &= \sum_{m=1}^{n_1}\sum_{n=1}^{n_2} \widehat{u}_{m,n} e^{i2\pi m \frac{j}{n_1}} e^{i2\pi n \frac{k+\frac{1}{2}}{n_2}}, \\ v_{j,k} &= \sum_{m=1}^{n_1}\sum_{n=1}^{n_2} \widehat{v}_{m,n} e^{i2\pi m \frac{j+\frac{1}{2}}{n_1}} e^{i2\pi n \frac{k}{n_2}}, \\ \varphi_{j,k} &= \sum_{m=1}^{n_1}\sum_{n=1}^{n_2} \widehat{\varphi}_{m,n} e^{i2\pi m \frac{j+\frac{1}{2}}{n_1}} e^{i2\pi n \frac{k+\frac{1}{2}}{n_2}}. \end{aligned} \quad (7.4)$$

Theorem 7.1. ***Crank-Nicolson Stability***
The Crank-Nicolson scheme with the projection method formulated by equations (7.1)-(7.3) *for solving the time-dependent Stokes equations is unconditionally stable.*

We will prove the above theorem via the following four lemmas. Lemma 7.2 derives an expression for the Fourier modes of the auxiliary velocity in terms of those of the velocity in the previous step and an amplification factor. Lemma 7.3 gives an expression for the solution of equation (7.2) in Fourier space. These expressions are then combined in Lemma 7.4 to provide an expression for the time update of the velocity in Fourier space in terms of an update matrix. Finally, Lemma 7.5 shows that the spectral radius of this matrix is less than one and and therefore the time-stepping procedure is stable.

Lemma 7.2. *Let* $\lambda = \sqrt{(2\pi m)^2 + (2\pi n)^2}$ *and* $\rho_{m,n} = (1 - \frac{\nu \mathrm{d}t}{2}\lambda^2)/(1 + \frac{\nu \mathrm{d}t}{2}\lambda^2)$, *then each* (m,n)*-th Fourier*

mode of equation (7.1) *is given by the following:*

$$\begin{bmatrix}\widehat{u}^*_{m,n}\\ \widehat{v}^*_{m,n}\end{bmatrix} = \rho_{m,n}\begin{bmatrix}\widehat{u}^\iota_{m,n}\\ \widehat{v}^\iota_{m,n}\end{bmatrix}.$$

Proof. Rearranging equation (7.1) and solving for $\mathbf{u}^*$ leads us to

$$\mathbf{u}^* = \left(\frac{\left(I + \frac{\nu \mathrm{d}t\Delta}{2}\right)}{\left(I - \frac{\nu \mathrm{d}t\Delta}{2}\right)}\right)\mathbf{u}^\iota. \tag{7.5}$$

From equation (5.2), we have

$$\begin{aligned}&\Delta u(x,y)\\ &= \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} -4\pi^2\left(m^2+n^2\right)\widehat{u}_{m,n}e^{i2\pi mx}e^{i2\pi ny}.\end{aligned}$$

Thus, defining $\lambda = \sqrt{(2\pi m)^2 + (2\pi n)^2}$ and $\rho_{m,n} = (1 - \frac{\nu \mathrm{d}t}{2}\lambda^2)/(1 + \frac{\nu \mathrm{d}t}{2}\lambda^2)$, equation (7.5) can be transform using discrete Fourier transform:

$$\sum_{m=1}^{n_1}\sum_{n=1}^{n_2}\begin{bmatrix}\widehat{u}^*_{m,n}e^{i2\pi m\frac{j}{n_1}}e^{i2\pi n\frac{k+\frac{1}{2}}{n_2}}\\ \widehat{v}^*_{m,n}e^{i2\pi m\frac{j+\frac{1}{2}}{n_1}}e^{i2\pi n\frac{k}{n_2}}\end{bmatrix} \tag{7.6}$$

$$= \sum_{m=1}^{n_1}\sum_{n=1}^{n_2}\begin{bmatrix}\rho_{m,n}\widehat{u}^\iota_{m,n}e^{i2\pi m\frac{j}{n_1}}e^{i2\pi n\frac{k+\frac{1}{2}}{n_2}}\\ \rho_{m,n}\widehat{v}^\iota_{m,n}e^{i2\pi m\frac{j+\frac{1}{2}}{n_1}}e^{i2\pi n\frac{k}{n_2}}\end{bmatrix}. \tag{7.7}$$

So, for each (m,n)-th mode, we obtain:

$$\begin{bmatrix}\widehat{u}^*_{m,n}\\ \widehat{v}^*_{m,n}\end{bmatrix} = \rho_{m,n}\begin{bmatrix}\widehat{u}^\iota_{m,n}\\ \widehat{v}^\iota_{m,n}\end{bmatrix}.$$

■

Lemma 7.3. *The solution to equation* (7.2) *can be written in the following discrete Fourier transform:*

$$\varphi_{j,k} = \sum_{m=1}^{n_1}\sum_{n=1}^{n_2} -\frac{\rho_{m,n}}{\lambda^2}\Big(i2\pi m\widehat{u}^\iota_{m,n}e^{i2\pi m\frac{j}{n_1}}e^{i2\pi n\frac{k+\frac{1}{2}}{n_2}} + i2\pi n\widehat{v}^\iota_{m,n}e^{i2\pi m\frac{j+\frac{1}{2}}{n_1}}e^{i2\pi n\frac{k}{n_2}}\Big).$$

Proof. Applying the discrete Fourier transform to $\Delta\varphi$ and $\nabla\cdot\mathbf{u}^*$ we have:

$$\begin{aligned}&(\Delta\varphi)_{j,k}\\ &= \sum_{m=1}^{n_1}\sum_{n=1}^{n_2} -\left[\left(2\pi m\right)^2 + \left(2\pi n\right)^2\right]\widehat{\varphi}_{m,n}e^{i2\pi m\frac{j+\frac{1}{2}}{n_1}}e^{i2\pi n\frac{k+\frac{1}{2}}{n_2}}.\end{aligned}$$

$$\begin{aligned}&(\nabla\cdot\mathbf{u}^*)_{j,k}\\ &= \sum_{m=1}^{n_1}\sum_{n=1}^{n_2} i2m\pi\rho_{m,n}\widehat{u}^\iota_{m,n}e^{i2\pi m\frac{j}{n_1}}e^{i2\pi n\frac{k+\frac{1}{2}}{n_2}}\\ &\quad + \sum_{m=1}^{n_1}\sum_{n=1}^{n_2} i2n\pi\rho_{m,n}\widehat{v}^\iota_{m,n}e^{i2\pi m\frac{j+\frac{1}{2}}{n_1}}e^{i2\pi n\frac{k}{n_2}}\\ &= \sum_{m=1}^{n_1}\sum_{n=1}^{n_2}\Big[\rho_{m,n}\Big(i2m\pi\widehat{u}^\iota_{m,n}e^{-i2\pi m\frac{1}{2n_1}}\Big)\\ &\quad + \rho_{m,n}\Big(i2n\pi\widehat{v}^\iota_{m,n}e^{-i2\pi n\frac{1}{2n_2}}\Big)\Big]e^{i2\pi m\frac{j+\frac{1}{2}}{n_1}}e^{i2\pi n\frac{k+\frac{1}{2}}{n_2}}.\end{aligned}$$

Substitute into equation (7.2), for each (m,n)-th mode, we obtain:

$$\begin{aligned}&-\left[\left(2\pi m\right)^2 + \left(2\pi n\right)^2\right]\widehat{\varphi}_{m,n}\\ &= \rho_{m,n}\Big(i2m\pi\widehat{u}^\iota_{m,n}e^{-i2\pi m\frac{1}{2n_1}} + i2n\pi\widehat{v}^\iota_{m,n}e^{-i2\pi n\frac{1}{2n_2}}\Big).\end{aligned}$$

Since $\lambda = \sqrt{(2\pi m)^2 + (2\pi n)^2}$, we have:

$$\begin{aligned}&-\lambda^2\widehat{\varphi}_{m,n}\\ &= \rho_{m,n}\Big(i2m\pi\widehat{u}^\iota_{m,n}e^{-i2\pi m\frac{1}{2n_1}} + i2n\pi\widehat{v}^\iota_{m,n}e^{-i2\pi n\frac{1}{2n_2}}\Big).\end{aligned}$$

Thus, solve for $\widehat{\varphi}_{m,n}$ we obtain:

$$\widehat{\varphi}_{m,n} = -\frac{\rho_{m,n}}{\lambda^2}\Big(i2m\pi\widehat{u}^\iota_{m,n}e^{-i2\pi m\frac{1}{2n_1}} + i2n\pi\widehat{v}^\iota_{m,n}e^{-i2\pi n\frac{1}{2n_2}}\Big). \tag{7.8}$$

Finally, substitute $\widehat{\varphi}_{m,n}$ in equation (7.8) into equation (7.4), we obtain:

$$\begin{aligned}\varphi_{j,k} &= \sum_{m=1}^{n_1}\sum_{n=1}^{n_2}\Big[-\frac{\rho_{m,n}}{\lambda^2}\Big(i2m\pi\widehat{u}^\iota_{m,n}e^{-i2\pi m\frac{1}{2n_1}}\\ &\quad + i2n\pi\widehat{v}^\iota_{m,n}e^{-i2\pi n\frac{1}{2n_2}}\Big)\Big]e^{i2\pi m\frac{j+\frac{1}{2}}{n_1}}e^{i2\pi n\frac{k+\frac{1}{2}}{n_2}}\\ &= \sum_{m=1}^{n_1}\sum_{n=1}^{n_2} -\frac{\rho_{m,n}}{\lambda^2}\Big(i2\pi m\widehat{u}^\iota_{m,n}e^{i2\pi m\frac{j}{n_1}}e^{i2\pi n\frac{k+\frac{1}{2}}{n_2}}\\ &\quad + i2n\pi\widehat{v}^\iota_{m,n}e^{i2\pi m\frac{j+\frac{1}{2}}{n_1}}e^{i2\pi n\frac{k}{n_2}}\Big).\end{aligned}$$

■

Lemma 7.4. *Each (m,n)-th Fourier mode of equation* (7.3) *can be expressed as the following:*

$$\begin{bmatrix}\widehat{u}^{\iota+1}_{m,n}\\ \widehat{v}^{\iota+1}_{m,n}\end{bmatrix} = \rho_{m,n}\begin{bmatrix}\frac{n^2}{m^2+n^2} & -\frac{mn}{n^2+m^2}\\ -\frac{mn}{n^2+m^2} & \frac{m^2}{m^2+n^2}\end{bmatrix}\begin{bmatrix}\widehat{u}^\iota_{m,n}\\ \widehat{v}^\iota_{m,n}\end{bmatrix}.$$

Proof. The gradient of φ can be written in the following discrete Fourier transform,

$$(\nabla\varphi)_{j,k} = \begin{bmatrix}\sum_{m=1}^{n_1}\sum_{n=1}^{n_2} -\frac{\rho_{m,n}}{\lambda^2}Q_1\\ \sum_{m=1}^{n_1}\sum_{n=1}^{n_2} -\frac{\rho_{m,n}}{\lambda^2}Q_2\end{bmatrix},$$

where Q_1 represents

$$\Big(\Big(i2\pi m\Big)^2 \widehat{u}^{\iota}_{m,n} e^{i2\pi m\frac{j}{n_1}} e^{i2\pi n\frac{k+\frac{1}{2}}{n_2}} + \Big(i2\pi\Big)^2 mn\widehat{v}^{\iota}_{m,n} e^{i2\pi m\frac{j+\frac{1}{2}}{n_1}} e^{i2\pi n\frac{k}{n_2}}\Big)$$

and Q_2 represents

$$\Big(\Big(i2\pi\Big)^2 mn\widehat{u}^{\iota}_{m,n} e^{i2\pi m\frac{j}{n_1}} e^{i2\pi n\frac{k+\frac{1}{2}}{n_2}} + \Big(i2\pi n\Big)^2 \widehat{v}^{\iota}_{m,n} e^{i2\pi m\frac{j+\frac{1}{2}}{n_1}} e^{i2\pi n\frac{k}{n_2}}\Big).$$

Now, substitute the discrete Fourier transform of $\mathbf{u}^{\iota+1}$, $\mathbf{u}^*$, and $\nabla\varphi$ into equation (7.3), we see that at each (m,n)-th mode is given by:

$$\begin{bmatrix} \widehat{u}^{\iota+1}_{m,n} e^{i2\pi m\frac{j}{n_1}} e^{i2\pi n\frac{k+\frac{1}{2}}{n_2}} \\ \widehat{v}^{\iota+1}_{m,n} e^{i2\pi m\frac{j+\frac{1}{2}}{n_1}} e^{i2\pi n\frac{k}{n_2}} \end{bmatrix} = \rho_{m,n} \begin{bmatrix} 1-\frac{(2\pi m)^2}{\lambda^2} & -\frac{(2\pi)^2 mn}{\lambda^2} \\ -\frac{(2\pi)^2 mn}{\lambda^2} & 1-\frac{(2\pi n)^2}{\lambda^2} \end{bmatrix} \begin{bmatrix} \widehat{u}^{\iota}_{m,n} e^{i2\pi m\frac{j}{n_1}} e^{i2\pi n\frac{k+\frac{1}{2}}{n_2}} \\ \widehat{v}^{\iota}_{m,n} e^{i2\pi m\frac{j+\frac{1}{2}}{n_1}} e^{i2\pi n\frac{k}{n_2}} \end{bmatrix}.$$

Since $\lambda^2 = (2\pi m)^2 + (2\pi n)^2$, the equation above is simplified to:

$$\begin{bmatrix} \widehat{u}^{\iota+1}_{m,n} \\ \widehat{v}^{\iota+1}_{m,n} \end{bmatrix} = \rho_{m,n} \begin{bmatrix} \frac{n^2}{m^2+n^2} & -\frac{mn}{n^2+m^2} \\ -\frac{mn}{n^2+m^2} & \frac{m^2}{m^2+n^2} \end{bmatrix} \begin{bmatrix} \widehat{u}^{\iota}_{m,n} \\ \widehat{v}^{\iota}_{m,n} \end{bmatrix}.$$

■

Lemma 7.5. $\left\|\widehat{\boldsymbol{u}}^{\iota+1}_{m,n}\right\| \le \left\|\widehat{\boldsymbol{u}}^{0}_{m,n}\right\|$ *for all* ι*, and the method is unconditionally stable.*

Proof. The eigenvalues of the matrix is given by the characteristic equation of this 2×2 matrix,

$$\begin{vmatrix} \frac{n^2}{m^2+n^2}-\eta & -\frac{mn}{n^2+m^2} \\ -\frac{mn}{n^2+m^2} & \frac{m^2}{m^2+n^2}-\eta \end{vmatrix} = \left(\frac{n^2}{m^2+n^2}-\eta\right)\left(\frac{m^2}{m^2+n^2}-\eta\right) - \left(\frac{mn}{n^2+m^2}\right)^2.$$

Setting the determinant equals zero, we have

$$\begin{aligned} &\eta^2 - \left(\frac{n^2}{m^2+n^2}+\frac{m^2}{m^2+n^2}\right)\eta = 0 \\ &\eta^2 - \eta = 0 \\ &\eta(\eta-1) = 0. \end{aligned}$$

This implies that the eigenvalues are $\eta = 0$ or $\eta = 1$. Since $\rho_{m,n} = \frac{1-\frac{\nu dt}{2}\lambda^2}{1+\frac{\nu dt}{2}\lambda^2}$, it is true that $|\rho_{m,n}| \le 1$. So, for any vector norm $\|\cdot\|$ and any ι, we have

$$\begin{aligned} \left\|\widehat{\mathbf{u}}^{\iota+1}_{m,n}\right\| &\le |\rho_{m,n}|\cdot|\eta|\cdot\left\|\widehat{\mathbf{u}}^{\iota}_{m,n}\right\| \\ &\le (|\rho_{m,n}|\cdot|\eta|)^2\cdot\left\|\widehat{\mathbf{u}}^{\iota-1}_{m,n}\right\| \\ &\le (|\rho_{m,n}|\cdot|\eta|)^{\iota+1}\cdot\left\|\widehat{\mathbf{u}}^{0}_{m,n}\right\|. \end{aligned}$$

Since $\eta \le 1$ and $|\rho_{m,n}| \le 1$, we have $\left\|\widehat{\mathbf{u}}^{\iota+1}_{m,n}\right\| \le \left\|\widehat{\mathbf{u}}^{0}_{m,n}\right\|$ for all ι.

■

The last lemma concludes that the error $\mathbf{u}^{\iota+1}$ does not propagate in time. Therefore, the Crank-Nicolson scheme with the projection method formulated by equations (7.1)-(7.3) for solving the time-dependent Stokes equations is unconditionally stable. Note that our stability analysis is only on the time-dependent Stokes equations. To include the nonlinear effect from the Navier-Stokes equations, we may follow the same procedure. However, the Fourier transform of the nonlinear term involves convolution and needs more careful treatment. We leave the full stability analysis of Navier-Stokes equations to future work.

8 Particles Simulation

In this chapter, we will do a simulation of the particles flowing along the velocity fluid field in 2D and 3D spaces with initial velocity $\mathbf{u}(\mathbf{x},0) = \mathbf{0}$ and a specified body force $\mathbf{f}(\mathbf{x},t)$ on the fluid flow. We enforce the divergence-free condition in the equation, and assume the case of periodic boundary condition. For the purpose of simulation, we select the particles on the fluid field at various locations and design the force flow to create certain patterns.

8.1 Particles Simulation in 2D

We will show two examples for 2D velocity field. The first example simulates the particles on a fluid field with very small viscosity, which flows according to the Navier-Stokes equations. The second example simulates the particles on a fluid field with a higher viscosity, which mimics the behavior of Stokes flow. Stokes flow corresponds to the case where the inertial term is so small compared to the viscous term of the fluid equation that it can be neglected [13]. One property about Stokes flow is that it has memory and can be reversed if we reverse the force field. Navier Stokes equation, on the other hand, creates turbulence, and can not be reversed to the original state with an opposite force field. We define the force flow in 2D simulation to be:

$$\mathbf{f}(\mathbf{x},\mathbf{u},t) = a\sin(\omega t)\left[\frac{b(\mathbf{x}-\mathbf{x}^{\perp}) + (\mathbf{x}-\mathbf{r}_c)^{\perp}\cdot \mathbb{1}_{\|\mathbf{x}-\mathbf{r}_c\|\le r}}{\sqrt{\|\mathbf{x}-\mathbf{r}_c\|^2+0.1}}\right] + c\mathbf{u}(1-\|\mathbf{u}\|^2). \tag{8.1}$$

The symbol $\perp$ represents a perpendicular vector, i.e., $(x, y)^{\perp} = (-y, x)$. Since $\mathbf{x}^{\perp}$ is perpendicular to $\mathbf{x}$ we have $\mathbf{x}^{\perp} \cdot \mathbf{x} = 0$. In this case, $(-y, x) \cdot (x, y) = 0$.
The symbol $\mathbb{1}$ represents an indicator function. An indicator function is a function defined on a set of X that indicates membership of an element in a subset A of X, having the value 1 for all elements of A and the value 0 for all elements of X not in A. In this example, the set X is the domain of the fluid field $[0, 1] \times [0, 1]$. The subset A of X is a circle centered at $\mathbf{r_c}$ with radius r. So, all the (x, y) values within the region of the circle will have a force in both x-direction and y-direction. In particularly, the force will be uniform within the circle since we divided it by its norm.

This force will make the particles move along this fluid field. The length of the arrows represents the magnitude or velocity of the vector fields. The direction of the arrows tell us the path of how a particle would move along a certain location. Note that the force equation depends on the spatial grids, velocity fields, and is periodic in time with a period of $2\pi/\omega$. This means that the force field can be reversible. Moreover, there are three components in the force field. The first component corresponds to a drifting force, varying based on the location. The second component creates a rotational motion within a circular region with radius r centered at $\mathbf{r_c}$. The last component creates a frictional or damping force. This means that when the velocity is too strong, this friction term will force the velocity in a negative direction so that the velocity becomes weaker. The inclusion of the damping component is for the stability of the numerical solution.

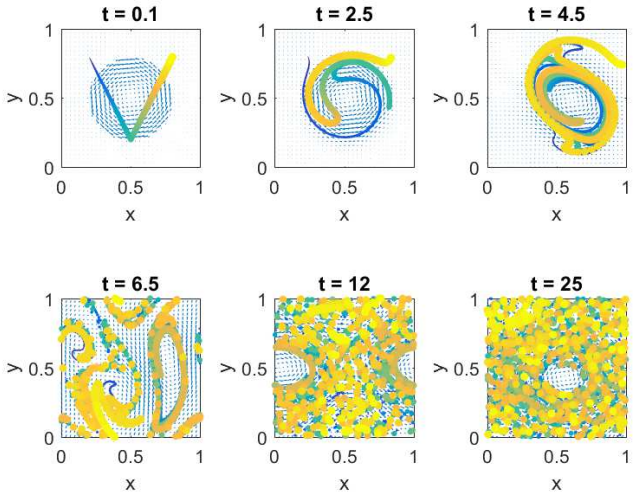

Fig. 4: A 2D simulation of particles' motion in a rotational fluid velocity field with low viscosity.

Fig. 4 shows how the particles with V-shape move in time based on the initial velocity and the force given above with a very low constant coefficient of kinematic viscosity $\nu = 0.001$, $\mathrm{d}t = 0.001$, $N_1 = N_2 = 128$, and $T = 25$. For the first example, we take $a = 1$, $\omega = 0.2\pi$, $b = 0.01$, $c = 0.1$, $\mathbf{r}_c = (0.5, 0.5)$, and $r = 0.3$ in the force flow equation (8.1). From Fig. 4, we can see that if we let it run for a longer time period the particles will mix together and will not return to the original shape. Note that the particle V-shape started to twirl toward the center first, and then formed an oval shape and started to drifted out of the circular motion. After a while, the particles mix together and can not be recovered to its original V-shape even with the reversed forcing.

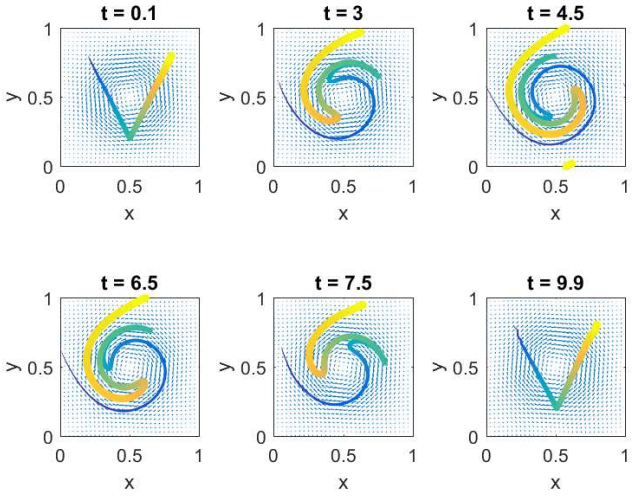

Fig. 5: A 2D simulation of particles' motion in a rotational fluid velocity field with high viscosity, showing reversibility property.

This is different from the Navier-Stokes equation with a high ν and appropriate force term, which can restore the particle's original shape within a given time period as shown in Fig. 5. We can obtain the motion of particle V-shape in Fig. 5 that behaves like Stokes flow by letting $\nu = 1$, $\mathrm{d}t = 0.001$, $N_1 = N_2 = 128$, and $T = 10$. We take $a = 50$, $\omega = 0.2\pi$, $b = 0$, $c = 0.1$, $\mathbf{r}_c = (0.5, 0.5)$, and $r = 0.3$ in the force flow equation (8.1). Notice that in the second example, there is no drifting force involved, so the particles will only rotate to the left hand side, and then will rotate it back to the right hand side to restore the V-shape. The V-shape particle was first started to twirl toward the center of the circle, and then at the half-time period, it started to unwind and return back to the original shape. We can refer this as a *time − reversibility* of Stokes flows. However, due to numerical error and the fact that the equation that we are solving is not exactly the real Stokes equation, we can see that there is a little changed in the V-shape. In other words, it did not restore back the exact same V-shape that we originally started from the beginning. Nevertheless, if we let ν to be higher

like $\nu = 5$ and let it run for one time period, we will see that the restoring V-shape looks closer to the original V-shape.

8.2 Particles Simulation in 3D

In the three dimensional simulation, we will consider the case of a larger viscosity since the particles will be spreading and mixing together in time with a very small viscosity. The domain of the fluid field is $[0,2] \times [0,2] \times [0,3]$, and we let $N_1 = N_2 = N_3 = 64$, $\mathrm{d}t = 0.01$, $T = 10$, and $\nu = 0.5$. In the 3D example, we define the force flow in x, y, and z-direction to be:

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} -a(y-c) \cdot \mathbb{1}_A(r - \gamma(x,y,c))(\gamma(x,y,c)) \\ a(x-c) \cdot \mathbb{1}_A(r - \gamma(x,y,c))(\gamma(x,y,c)) \\ b(cos(\frac{\pi(z-d)}{2} + 1)) \cdot \mathbb{1}_A \end{bmatrix}$$

where $a = 250$, $b = 0.3$, $c = 1$, $d = 2$, $r = 0.4$, $\gamma(x,y,c) = \sqrt{(x-c)^2 + (y-c)^2}$, and $A = (x-c)^2 + (y-c)^2 < r^2$. The factors $-(y-c)$ and $(x-c)$ cause the velocity to rotate counterclockwise in the xy-plane within the region of a circle with radius 0.4 centered at (1,1). The factor $(r - \gamma(x,y,c))(\gamma(x,y,c))$ is responsible for the rotational strength, which causes the velocity to have a weak rotational force near the center and a strong rotational force away from the center. The z-component of the force f_3 points upward, and varying along the z-direction. Overall, this force will rotate the particles in the xy-plane and push the particles to move upward in the z-direction as shown in Fig. 6.

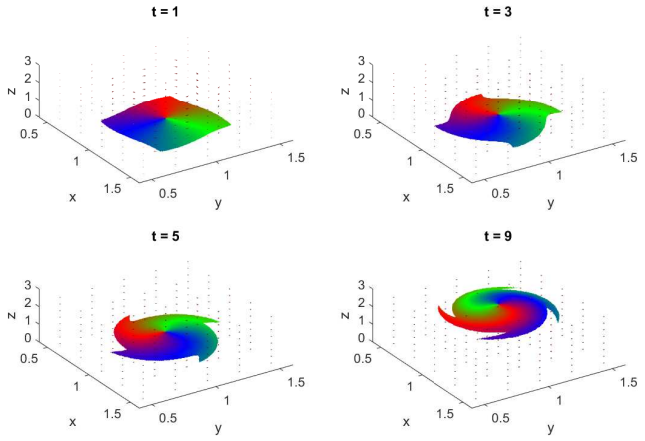

Fig. 6: A 3D simulation of particles' motion in a rotational and ascending fluid velocity field.

9 Conclusion

We have derived the Navier-Stokes equations using the conservation laws of mass and momentum. The first part of the equations is called the momentum equation because it arises from Newton's second law, $F = ma$. The second part of the equations is known as the continuity equation. There are many different techniques such as finite difference, finite volume, and spectral method being developed within the area of CFD to approximate the solution of the incompressible Navier-Stokes equations. We numerically solve the Navier-Stokes equations with periodic boundary condition using the second-order projection method introduced by Kim and Moin, which was extended from Chorin's first-order projection method. As shown by Brown, Cortez, and Minion in the article "Accurate Projection Methods for the Incompressible Navier-Stokes Equations" using normal mode analysis, the projection method suggested by Kim and Moin gives a second-order accuracy in both velocity field and pressure. To confirm their analytical result, we numerically verified that the method is second-order accurate by applying fast Fourier transform and finite differences scheme to solve the resulting Poisson equations as sub-steps in the projection method. The motivation behind the projection method is the Helmholtz-Hodge decomposition theorem, which states that any smooth velocity field $\mathbf{u}^*$ can be decomposed into a unique orthogonal decomposition of gradient- and divergence-free fields. Thus, the idea behind the projection method is to use the momentum equation to solve for an intermediate velocity, which is not required to be divergence-free. Then, applied the Helmholtz-Hodge decomposition theorem to project $\mathbf{u}^*$ onto a solenoidal field to yield a divergence-free velocity and a gradient field. Once $\mathbf{u}^*$ is known, we can use it to solve an elliptic equation that enforces the divergence constraint and update the pressure. By applying the Crank-Nicolson scheme for temporal discretization, centered differences for spatial discretization, and a Fourier-spectral discretization to solve a Poisson equation on MAC grids, we see that the rates of error decay in L_∞ norm for both the velocity and the pressure are quadratic. This verified that the method is second-order accurate in both the velocity and the pressure. We have also performed the stability analysis of the time-dependent Stokes equations, which is the case when the magnitude of the velocity is very small so that the nonlinear term can be neglected. Solving the Poisson equation using Fourier-spectral method with von Neumann stability analysis, we see that the error of the velocity does not propagate in time. Thus, we conclude that the Crank-Nicolson scheme with the projection method for solving the time-dependent Stokes equations is unconditionally stable. For the purpose of particles simulation on the fluid flow, we included two cases of fluid flow in 2D and one case of fluid flow in 3D. The first case in 2D simulated the particles on the fluid flows with very small viscosity, which act according to the Navier-Stokes equations. The second case simulated the particles on the fluid flows with a higher viscosity, which behave like Stokes flow, and the particles' movement are reversible. The particles in 3D case were also behave similar to Stokes flow.

References

[1] S. Armfield, "Finite difference solutions of the Navier-Stokes equations on staggered and non-staggered grids." *Computers and Fluids* 20 (1991): 1-17.

[2] H. Berg and R. Anderson. "Bacteria Swim by Rotating their Flagellar Filaments." *Nature* 245 (1973): 380-382.

[3] D. Brown, R. Cortez, and M. Minion, "Accurate Projection Methods for the Incompressible Navier-Stokes Equations." *Journal of Computational Physics* 168 (2001): 464-499.

[4] R. Burden and D. Faires, *Numerical Analysis*. 9th ed. Boston, MA: Brooks/Cole, Cengage Learning, 2011.

[5] A. Chorin, "Numerical solutions of the Navier-Stokes equations." *Mathematics of Computation* 22, (1968): 745-762.

[6] A. Chorin, "On the convergence of discrete approximations to the Navier-Stokes equations." *Mathematics of Computation* 23 (1969): 341-353.

[7] Clay Mathematics Institute. "The Millennium Prize Problems." http://www.claymath.org/millennium-problems/navier-stokes-equation (accessed February 23, 2018).

[8] L. Edelstein-Keshet, "Mathematical models of swarming and social aggregation." In *Proceedings of the 2001 International Symposium on Nonlinear Theory and Its Applications, Miyagi, Japan,* (2001): 1-7.

[9] M. Fortin and A. Fortin. "A generalization of Uzawa's algorithm for the solution of the Navier-Stokes equations." *International Journal for Numerical Methods in Biomedical Engineering* 1, no. 5 (1985): 205-208.

[10] V. Fuka, "PoisFFT-A Free Parallel Fast Poisson Solver." *Applied Mathematics and Computation* 267, (2015): 356-364.

[11] R. Guy and A. Fogelson, "Stability of approximate projection methods on cell-centered grids." *Journal of Computational Physics* 203 (2005): 517-538.

[12] J. Kim and P. Moin, "Application of a fractional-step method to incompressible Navier-Stokes equations." *Journal of Computational Physics* 59 (1985): 308-323.

[13] S. Kim and S. Karrila. *Microhydrodynamics: Principles and Selected Applications*. Dover, 2005.

[14] L. Quartapelle, *Numerical Solution of the Incompressible Navier-Stokes Equations*. Basel: Birkhäuser Verlag, 1993.

[15] S. Vanka, "Block-implicit multigrid solution of Navier-Stokes equations in primitive variables." *Journal of Computational Physics* 65, no. 1 (1986): 138-158.

[16] N. Vitanov and Z. Dimitrova. "Application of the method of simplest equation for obtaining exact traveling-wave solutions for two classes of model PDEs from ecology and population dynamics." *Communications in Nonlinear Science and Numerical Simulation* 15 (2010): 2836-2845.

[17] Younsi, R. *Navier-Stokes Equations: Properties, Description and applications*. New York: Nova Science, 2012.

Vorleak Yek

Department of Mathematics and Statistics,
California State University, Long Beach,
1250 Bellflower Blvd, Long Beach, CA 90840, USA.

E-mail addresses: vorleakyek.aleks@gmail.com