A Study on Numerical Solution to the Incompressible Navier-Stokes Equation

Zipeng Zhao

May 2014

1 Introduction

1.1 Motivation

One of the most important applications of finite differences lies in the field of computational fluid dynamics (CFD). In particular, the solution to the Navier-Stokes equation grants us insight into the behavior of many physical systems. The 2D Incompressible Navier-Stokes equation has been studied extensively due to its analogous nature to many practical applications, and several numerical schemes have been developed to provide solutions dedicated to different environmental conditions (such as different Reynolds numbers). For instance, the ongoing work of MITs Aerospace Computational Design Laboratory leverages multidisciplinary aircraft models generated through numerical PDE solvers, a few of which may deal directly with the Navier-Stokes equation. Consequently, the primary intent of this project is to study an approach by Benjamin Seibold in demonstrating a numerical solver for the Navier-Stokes equation. The project focuses on many nuisances that this particular implementation brings, such as the complexity behind the very staggered grid and the manifestation of the non-linear parts of the equation onto the grid. We also cover the assignment of boundary conditions, starting with the simple case of a lid driven cavity problem. In addition, several parts of the equation are given implicitly, which requires efficient ways of solving large systems of equations.

1.2 Problem Definition

To start, we consider the 2D Incompressible Navier-Stokes equation:

$$u_t + p_x = -(u^2)_x - (uv)_y + \frac{1}{Re}(u_{xx} + u_{yy}) \tag{1}$$

$$v_t + p_y = -(uv)_x - (v^2)_y + \frac{1}{Re}(u_{xx} + u_{yy}) \tag{2}$$

$$u_x + v_y = 0 \tag{3}$$

on a rectangular domain $\Omega = [0, l_x] \times [0, l_y]$. (u, v) represents the velocity field, p represents the pressure, and Re represents the Reynolds number. A shortened derivation of this equation will be provided in Section 2. The four domain boundaries are denoted **N**orth, **S**outh, **W**est, and **E**ast. The domain stays fixed in time, and we consider no-slip boundary conditions on each wall, namely:

$$u(x, l_y) = u_N(x) \qquad v(x, l_y) = 0 \tag{4}$$
$$u(x, 0) = u_S(x) \qquad v(x, 0) = 0 \tag{5}$$
$$u(0, y) = 0 \qquad v(0, y) = v_W(y) \tag{6}$$
$$u(l_x, y) = 0 \qquad v(l_x, y) = v_E(y) \tag{7}$$

Our objective is to find an efficient way to discretize the PDEs and solve the system numerically using finite difference approximations.

2 Analysis

2.1 Shortened Derivation of Navier-Stokes Equation

The Navier-Stokes equation is a special case of the continuity equation, expressible through the Reynolds transport theorem:

$$\frac{d}{dt}\int_\Omega L dV = -\int_{\partial\Omega} L\mathbf{v}\cdot\mathbf{n}dA - \int_\Omega Q dV \tag{8}$$

where L represents some intensive property of the conservative substance, Ω represents the control volume, $\mathbf{v}$ the velocity of the substance, Q the sources and sinks in the control volume, dV the element volume, dA the element area, and $\mathbf{n}$ the normal vector to the element area. In essence, the theorem states that the sum of the changes of some intensive property defined over a control volume must be equal to the net gain through the boundaries of the volume plus what is created by the sources and what is consumed by the sinks inside that volume. In the equation above, $-\int_{\partial\Omega} L\mathbf{v}\cdot\mathbf{n}dA$ represents the net gain through the boundaries of Ω while $-\int_\Omega Q dV$ represents changes through sinks.

Applying the divergence theorem to $-\int_{\partial\Omega} L\mathbf{v}\cdot\mathbf{n}dA$, equation (8) changes to:

$$\frac{d}{dt}\int_\Omega L dV = -\int_\Omega \nabla\cdot(L\mathbf{v})dV - \int_\Omega Q dV \tag{9}$$

Assuming L and $\frac{\partial L}{\partial t}$ are continuous, we apply Leibniz's integral rule and obtain:

$$\int_\Omega \frac{\partial L}{\partial t}dV = -\int_\Omega \nabla\cdot(L\mathbf{v})dV - \int_\Omega Q dV$$
$$\Rightarrow \int_\Omega \frac{\partial L}{\partial t} + \nabla\cdot(L\mathbf{v}) + Q dV = 0 \tag{10}$$

Since this must be true for any control volume, the integrand itself must equal 0:

$$\Rightarrow \frac{\partial L}{\partial t} + \nabla \cdot (L\mathbf{v}) + Q = 0 \tag{11}$$

To produce the Navier-Stokes equation, we begin by making the following assumptions:

- The fluid of interest behaves as a continuum.
- All relevant variables involved are differentiable
  - i.e. Fields such as pressure, velocity, density, temperature, etc.

Furthermore, we adopt a Lagrangian view of the fluid. That is, when measuring a given property, we follow a parcel of fluid along its streamline. The corresponding derivative is hence:

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla \tag{12}$$

where $\mathbf{v}$ represents the velocity of the fluid.

Consider a finite control volume Ω and its bounding surface $\partial\Omega$. We let L from the Reynolds transport theorem equal $\rho\mathbf{v}$, where ρ represents the density of the fluid:

$$\frac{\partial(\rho\mathbf{v})}{\partial t} + \nabla \cdot (\rho\mathbf{v}\mathbf{v}) + Q = 0 \tag{13}$$

where Q represents sources and sinks of the fluid in Ω.

Expanding above, we obtain:

$$\mathbf{v}\frac{\partial \rho}{\partial t} + \rho\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\mathbf{v} \cdot \nabla\rho + \rho\mathbf{v} \cdot \nabla\mathbf{v} + \rho\mathbf{v}\nabla \cdot \mathbf{v} = \mathbf{b} \tag{14}$$

$$\Rightarrow \mathbf{v}(\frac{\partial \rho}{\partial t} + \mathbf{v} \cdot \nabla\rho + \rho\nabla \cdot \mathbf{v}) + \rho(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla\mathbf{v}) = \mathbf{b} \tag{15}$$

$$\Rightarrow \mathbf{v}(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho\mathbf{v})) + \rho(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla\mathbf{v}) = \mathbf{b} \tag{16}$$

Applying Reynolds transport theorem separately to ρ, we obtain the conservation of mass:

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho\mathbf{v}) = 0 \tag{17}$$

in the absence of sources or sinks of mass. Notice that if ρ is constant, the above reduces to $\nabla \cdot \mathbf{v} = 0$. We will refer to this simple equation as the mass conservation constraint for incompressible fluids. Plugging (17) into (16), we get:

$$\rho(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla\mathbf{v}) = \mathbf{b} \tag{18}$$

which is very much analogous to $m\mathbf{a} = \mathbf{F}$.

Now consider the generic body force $\mathbf{b}$ from before:

$$\mathbf{b} = \nabla \cdot \sigma + \mathbf{f} \tag{19}$$

where σ is the Cauchy stress tensor and $\mathbf{f}$ is the net of all other body forces such as gravity. Specifically (in 3D):

$$\sigma_{ij} = \begin{pmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{pmatrix} \tag{20}$$

$$= -\begin{pmatrix} \pi & 0 & 0 \\ 0 & \pi & 0 \\ 0 & 0 & \pi \end{pmatrix} + \begin{pmatrix} \sigma_{xx} + \pi & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} + \pi & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} + \pi \end{pmatrix} \tag{21}$$

$$= -\pi I + T \tag{22}$$

where $\pi = -\frac{1}{3}(\sigma_{xx} + \sigma_{yy} + \sigma_{zz})$ (mean normal stress) and $tr(T) = 0$.

The Navier-Stokes equation assumes incompressible isotropic Newtonian fluids, that is:

$$\tau = \mu \frac{\partial u}{\partial y} \tag{23}$$

where τ represents the shear stress, μ represents the shear viscosity of the fluid, and y represents the displacement in the direction perpendicular to the velocity u. Equivalently:

$$\tau_{ij} = \mu(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) \tag{24}$$

Provided that $\nabla \cdot \mathbf{v} = 0$, we can easily show by plugging (24) into (22), and the result into (19) that:

$$\mathbf{b} = -\nabla\pi + \mu\nabla^2\mathbf{v} \tag{25}$$

notice that π represents the mechanical pressure. Plugging (25) into (18), we obtain the Navier-Stokes equation for incompressible isotropic fluids:

$$\rho(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla\mathbf{v}) = -\nabla\pi + \mu\nabla^2\mathbf{v} \tag{26}$$

In the notation of Seibold's documentation [1], this can be expanded and rewritten as:

$$u_t + p_x = -(u^2)_x - (uv)_y + \frac{1}{Re}(u_{xx} + u_{yy}) \tag{27}$$

$$v_t + p_y = -(uv)_x - (v^2)_y + \frac{1}{Re}(u_{xx} + u_{yy}) \tag{28}$$

$$u_x + v_y = 0 \tag{29}$$

with (29) being the mass conservation constraint for incompressible fluids.

2.2 Finite Difference Approach

The solution of the Navier-Stokes equation in time is given by u, v, p, and q, where the new variable q is the stream function (a function whose orthogonal gradient is the velocity field). The approach that Seibold takes to propagate the solution can be intuitively understood as a sequence of three different refinements per time step to include additional terms and correct the velocity fields. Certain refinements are given implicitly, which we will later show to be solved exactly by Cholesky decomposition and Gaussian elimination for efficiency.

Let U^n and V^n be the velocity field at the n^{th} time step (time t), and the mass conservation constraint for incompressible fluids is satisfied. We find the solution at the $(n+1)^{st}$ step (time $t + \triangle t$) by:

1. Adding nonlinear terms explicitly to find an intermediate solution:
   (A time-step-limiting CFL condition is introduced here)

$$\frac{U^* - U^n}{\triangle t} = -((U^n)^2)_x - (U^n V^n)_y \tag{30}$$

$$\frac{V^* - V^n}{\triangle t} = -(U^n V^n)_x - ((V^n)^2)_y \tag{31}$$

2. Adding viscosity terms ($\mu \nabla^2 \mathbf{v}$) implicitly to find another intermediate solution:

$$\frac{U^{**} - U^*}{\triangle t} = \frac{1}{Re}(U^{**}_{xx} + U^{**}_{yy}) \tag{32}$$

$$\frac{V^{**} - V^*}{\triangle t} = \frac{1}{Re}(U^{**}_{xx} + U^{**}_{yy}) \tag{33}$$

3. Apply Chorin's projection method to obtain p^{n+1}. Then update the intermediate solution with p^{n+1} to leave a divergence-free U^{n+1}:
   For any vector field $\mathbf{\Phi^{**}}$, we can apply the Helmholtz decomposition to obtain:

$$\mathbf{\Phi^{**}} = \mathbf{\Phi_{sol}} + \mathbf{\Phi_{irrot}} \tag{34}$$

   where $\mathbf{\Phi_{sol}}$ is divergence-free and $\mathbf{\Phi_{irrot}}$ is irrotational ($\nabla \times \mathbf{\Phi_{irrot}} = 0$). In our case, we know that $\mathbf{U^{n+1}}$ is divergence-free (mass conservation constraint for incompressible fluids) and pressure is irrotational, so starting from:

$$\frac{\mathbf{U^{n+1}} - \mathbf{U^{**}}}{\triangle t} = -\frac{1}{\rho}\nabla p^{n+1} \tag{35}$$

   we have:

$$\mathbf{U^{**}} = \mathbf{U^{n+1}} + \mathbf{U_{irrot}} \tag{36}$$

   where

$$\mathbf{U_{irrot}} = \frac{\triangle t}{\rho}\nabla p^{n+1} \tag{37}$$

Taking the divergence of both sides of (36), we obtain:

$$\nabla^2 p^{n+1} = \frac{\rho}{\triangle t} \nabla \cdot \mathbf{U}^{**} \tag{38}$$

Solve for p^{n+1}, then find ∇p^{n+1}. Use the result to update the velocity field:

$$\mathbf{U^{n+1}} = \mathbf{U}^{**} - \frac{\triangle t}{\rho} \nabla p^{n+1} \tag{39}$$

Notice that both 2 and 3 require taking the inverse of large matrices, which can be done efficiently using Cholesky decomposition and Gaussian elimination. In addition, for visualization, we compute $F^n = (V^n)_x - (U^n)_y$ and solve the Poisson equation $-\triangle Q^n = -F^n$.

2.3 Spatial Discretization

Much of this implementation's nuisances lie in the discretization. In particular, we cannot discretize each variable at the same position on the grid. Doing so will prevent us from conducting the three aforementioned refinements in sequence and result in heavy loss of resolution and potential numerical instabilities. Thus, we adopt a very staggered grid approach. The figure below shows the grid with boundary cells:

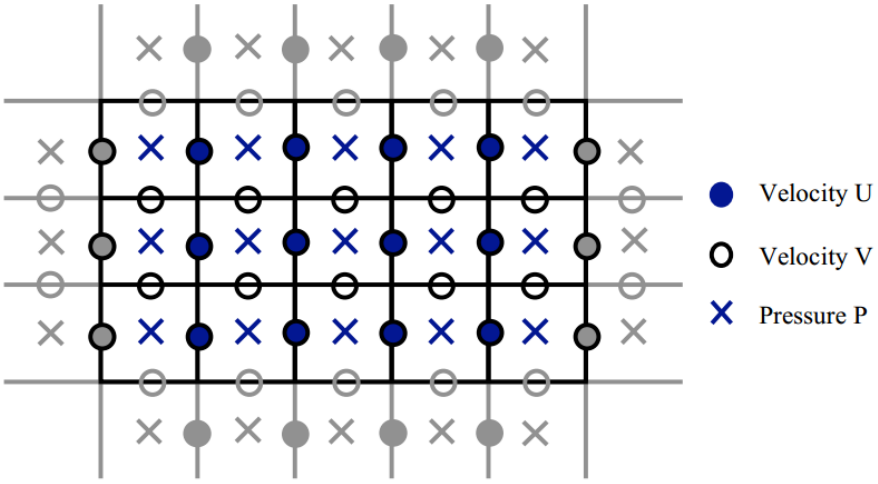

Figure 1: Very staggered grid with boundary cells[2]

At the first refinement step, we find (U^*, V^*) using (U^n, V^n). We cannot multiply U and V directly if they live in different positions. We must first

interpolate to get the terms to be at the same position. To do so, we use:

$$U_{i+\frac{1}{2},j} = \frac{U_{i+1,j} + U_{i,j}}{2} \tag{40}$$

$$U_{i,j+\frac{1}{2}} = \frac{U_{i,j+1} + U_{i,j}}{2} \tag{41}$$

$$V_{i+\frac{1}{2},j} = \frac{V_{i+1,j} + V_{i,j}}{2} \tag{42}$$

for updating U^* and analogously for V^*. This is a central differencing approach. Seibold also covers an upwinding approach in his documentation, which we will not cover here. Note that by taking the derivative $-((U^n)^2)_x$, we are finding a value at $U_{i,j}$'s position by differencing a value $\frac{1}{2}h_x$ to the right with a value $\frac{1}{2}h_x$ to the left. By taking the derivative $-(U^n V^n)_y$, we are finding a value at $U_{i,j}$'s position by differencing a value $\frac{1}{2}h_y$ to the top with a value $\frac{1}{2}h_y$ to the bottom (assuming $y_{i,j+1} > y_{i,j}$). This symmetry produces a value that shares the same position as the original $U_{i,j}$. Once again, this is analogous for V^*.

In the second refinement step, we find U^{**} using $U_{xx} + U_{yy}$. We need $U^{**}_{i,j}$ to be at the same place as $U_{i,j}$. The second difference for $U_{i,j}$ is symmetric in the interior points in both dimensions. That is, for U_{xx}, it uses a point h_x to the right, a point on top of itself, and a point h_x to the left; for U_{yy}, it uses a point h_y to the top, a point on top of itself, and a point h_y to the bottom. As a result, the second difference operator leaves a value that shares the same position as the original $U_{i,j}$, thereby satisfying the discretization requirement. This is analogously extended to V^{**}.

In the third refinement step, we find p^{n+1} using $\nabla \cdot \mathbf{U}^{**}$. Hence, we need the first derivative of $\mathbf{U}^{**}$ at the position where p^{n+1} is. Thus, we adopt the following approximation:

$$(U_x)_{i+\frac{1}{2},j} \approx \frac{U_{i+1,j} - U_{i,j}}{h_x} \tag{43}$$

$$(U_y)_{i,j+\frac{1}{2}} \approx \frac{U_{i,j+1} - U_{i,j}}{h_y} \tag{44}$$

which puts the pair (U_x, U_y) at the position of $p_{i,j}$.

2.4 Boundary Conditions

The assignment of boundary conditions follows naturally from the discretization. At the points that lie on the boundary, the value is directly prescribed, such as U at the west and east boundary, and V at the north and south boundary. At points where a boundary exists in between, an averaging operation is conducted between the points inside the grid and the points outside; the result is prescribed as the fixed boundary value. For pressure, which adopts Neumann boundary conditions, we take a forward difference leaping across the boundary (since the

boundary is always normal to pressure in our lid-driven cavity problem) and assign the result to zero (free boundary for pressure). Since we have direct access to the solution at each time step, we do not need to change the operator matrices (i.e. the matrices that perform the three refinements) substantially.

2.5 Solving Linear Systems

The method we used have implicit steps. This requires us to take the inverse of potentially very large matrices. Seibold's implementation adopts Cholesky decomposition, which takes advantage of the fact that all of our operator matrices are positive definite or can be made positive definite without affecting the solution and factors the operator matrix into a lower triangular part multiplied by an upper triangular part. An introduction to the concept for Cholesky decomposition can be found in Strang's textbook[3]. Since finding the inverse of a triangular matrix is much easier than finding the inverse of some arbitrary square matrix, we save a significant amount of computation. Prior to the decomposition, Seibold also reorders the operator matrix in order to minimize the amount of fill-ins due to elimination steps (MATLAB's backslash command uses Gaussian elimination). As such, the new permutation must be applied to all relevant matrices and solution vectors.

2.6 Demonstration

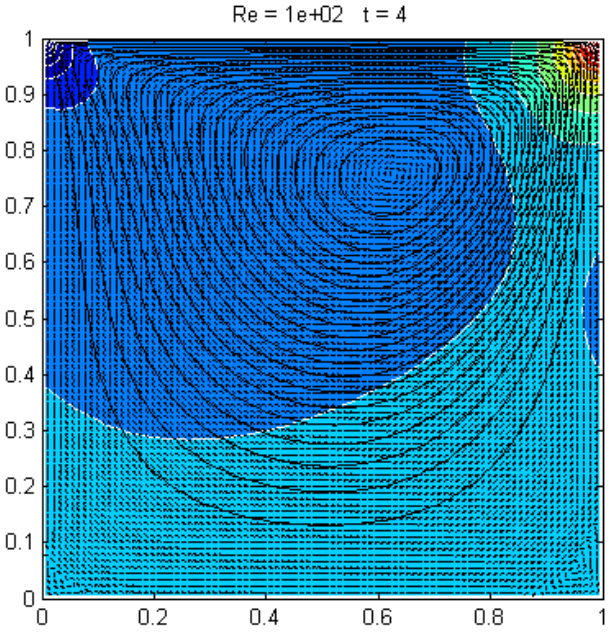

Figure 2: A demonstration run of the code

The above plot shows a run of the code in its default form from $t = 0$ to $t = 4$. Since our objective is not to study any particular physical application, we did not run the code extensively for simulation purposes. However, Kong demonstrates a very interesting use of the code in his project in 2008[2], which involves changing the boundary conditions to match that of a electro-chemical device.

References

[1] Benjamin Seibold, *A compact and fast Matlab code solving the incompressible Navier-Stokes equations on rectangular domains.* http://www-math.mit.edu/seibold (1994).

[2] Tian Fook Kong, *Finite Difference Modeling of Micro Optofluidic Switch by solving 2-D Navier-Stokes Equations.* http://math.mit.edu/classes/18.086/2008/ (2008).

[3] Gilbert Strang, *Computational Science and Engineering*, First Edition. Wellesley-Cambridge Press, 2008.