ISSN: 2754-6667

Journal of Critical Care & Emergency Medicine

SCIENTIFIC Research and Community

Review Article

Open Access

Approximating the Navier-Stokes Equations with the Explicit Finite Difference Method to Solve the Lid-driven Cavity Flow Problem

Elijah Kurien

Father Michael McGivney Catholic Academy, Ontario, Canada

ABSTRACT

The Navier-Stokes equations, one of the seven Millennium Problems, are a set of 2nd-order partial differential equations that are near impossible to solve. Instead of solving it, this paper uses the finite difference method to convert the differential terms in the Navier-Stokes Equations into algebraic equations to generate approximate solutions. The procedure to use the explicit finite difference method is outlined and proved in the context of solving the lid-driven cavity flow problem. The solution discussed is then applied in Mat lab and its accuracy and speed is evaluated. The finite difference method is then evaluated and its applications are outlined.

***Corresponding author**

Elijah Kurien, Father Michael McGivney Catholic Academy, Ontario, Canada.

Received: January 13, 2023; **Accepted:** January 23, 2023; **Published:** January 28, 2023

Keywords: Navier-Stokes, Finite Difference Method, Numerical Method, Lid-driven Cavity Flow, Fluid Dynamics

Introduction

Until the turn of the 19th century, fluid dynamics was an enigma to researchers. However, from the 1820s to the 1840s, Claude-Louis Navier and George Stokes independently worked on developing equations that could describe fluid flow at a molecular level. Their research bore a set of partial differential equations that describe fluid dynamics. These equations were named the Navier-Stokes equations (abbreviated as NS equations) [1]. From designing planes that generate the most lift to informing meteorologists about predicting the weather, the Navier-Stokes equations are essential to everyday life. Unfortunately, the NS equations are 2nd order partial differential equations. This quality makes them incredibly complex to solve. They are so challenging that the Clay Mathematics Institute is offering a $1 million prize for the first person to either prove or disprove the existence of globally defined smooth solutions to the equations ("Millennium Problems") [2]. Rather than finding exact solutions, this paper will explore how the explicit finite difference method (FDM) can approximate solutions to the NS equations. Like the Euler method (used for ordinary differential equations), FDM uses the Taylor Series and replaces differential terms with numerical approximations. These approximations make the problem much easier to solve [3]. This paper will answer the research question: *To what extent is the finite difference method feasible to quickly generate accurate approximations to the Navier-Stokes equations?* The finite difference methods will be tested on the lid-driven cavity flow problem - a universal benchmark for fluid dynamics.

Definitions

Derivatives are the rate of change or instantaneous slope at a particular point on a function (denoted by ***d*** or). They are computed on functions with only one independent variable. The derivative is algebraically defined using the following formula:

$$f'(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

Partial derivatives (denoted by ***d***) build upon the notion of a derivative. Like derivatives, partial derivatives are the instantaneous slope at a particular point on a function; however, they are computed on functions with two or more independent variables. Partial derivatives are calculated by differentiating a function with respect to any of its independent variables and treating any other variables as constants. For a function $f(x, y)$ its partial derivatives with respect to x and y can be calculated using the following formulas [4]:

$$\frac{\partial f}{\partial x} = \lim_{h\to 0}\frac{f(x+h,y)-f(x,y)}{h}$$
$$\frac{\partial f}{\partial y} = \lim_{h\to 0}\frac{f(x,y+h)-f(x,y)}{h}$$

The del operator (denoted by ∇) is a vector with the partial derivatives of a function. For a function with two independent variables, it can be calculated as follows:

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}$$

The gradient (denoted as ∇f) is the del operator applied to a multivariable function. The divergence measures how much a function diverges from its origin. It is the dot product of ∇ and a multivariable function written as ∇ · ***f*(*x*, *y*)**. The curl measures the tendency of a fluid to swirl around its origin and is the cross product of ∇ and a multivariable function written as **∇ x *f*(*x*, *y*)**. It can be calculated as follows:

$$\nabla \times f(x,y) = \frac{\partial f}{\partial y} - \frac{\partial f}{\partial x}$$

When considering the motion of a fluid at the boundary of a container, sometimes there may not be enough information to determine its properties. Thus, mathematicians use reasonable assumptions called boundary conditions to solve this problem. One type of boundary condition is the no-slip condition, which states that there is no change in the fluid's velocity parallel to the boundary relative to the wall's velocity. Another type of boundary condition is the impermeability boundary condition, which states that the fluid cannot permeate the boundary. This means there is no change in the fluid's velocity component that is perpendicular to the boundary relative to the wall's velocity [5].

Lid-Driven Cavity Flow Problem

The finite difference method's feasibility will be measured by how it solves the lid-driven cavity flow problem. The lid-driven cavity flow problem is a simple problem used to test solutions to the Navier-Stokes equation. It states that a square cavity with side lengths of 1 is closed with stationary walls on every side except the top. On the top side, however, the wall is moving at a speed of 1 unit/second [6].

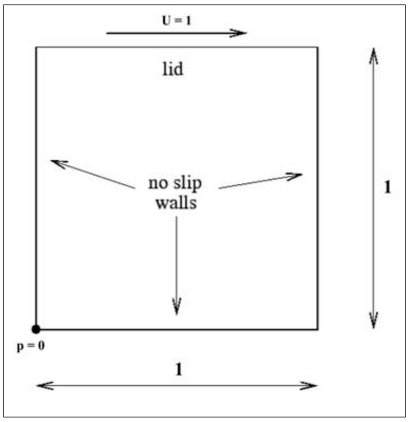

Figure 1: Visualization of the lid-driven cavity flow problem [6].

Navier-stokes Equations

The two-dimensional Navier-Stokes equations are a set of three equations that govern the motion of fluids. The first equation is as follows:

$$\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} = 0$$

This is known as the continuity or conservation of mass equation. The variables x and y are the x and y dimensions respectively and the variables U and V are the velocities in the x and y dimensions respectively. This equation can also be rewritten as **∇ · *f*(*x*, *y*)=0**; if the divergence is 0 then, on average, the fluid will not diverge towards or away resulting in a change in mass [7].

The second equation (denoted as ***B*(*x*, *y*, *t*)** and third equation ***C*(*x*, *y*, *t*)** (denoted as) are the Navier-Stokes momentum equations in the x and y dimensions respectively [8].

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + V\frac{\partial U}{\partial y} = -\frac{1}{\rho}\frac{\partial P}{\partial x} + \frac{\mu}{\rho}\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right)$$
$$\frac{\partial V}{\partial t} + U\frac{\partial V}{\partial x} + V\frac{\partial V}{\partial y} = -\frac{1}{\rho}\frac{\partial P}{\partial y} + \frac{\mu}{\rho}\left(\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2}\right)$$

In this equation, ρ is the density, P is the pressure, μ is the dynamic viscosity coefficient and t is time. If all the terms are moved to one side, the equations can now be denoted as ***B*(*x*,*y*,*t*)** and ***C*(*x*,*y*,*t*)** where

$$B(x,y,t) = \frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + V\frac{\partial U}{\partial y} + \frac{1}{\rho}\frac{\partial P}{\partial x} - \frac{\mu}{\rho}\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right) = 0$$
$$C(x,y,t) = \frac{\partial V}{\partial t} + U\frac{\partial V}{\partial x} + V\frac{\partial V}{\partial y} + \frac{1}{\rho}\frac{\partial P}{\partial y} - \frac{\mu}{\rho}\left(\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2}\right) = 0$$

Since ***B*(*x*,*y*,*t*)** calculates momentum in the x-dimension and ***C*(*x*,*y*,*t*)** calculates momentum in the y-dimension, the equations can be organized into a vector ***A*(*x*,*y*,*t*)**, which represents the 2nd Navier-Stokes equation in 2 dimensions shown below:

$$A(x,y,t) = \begin{bmatrix} B(x,y,t) \\ C(x,y,t) \end{bmatrix}$$

Deriving the Navier-Stokes Vorticity Transport Equation

The first step to solving the lid-driven cavity flow problem is to derive the Navier-Stokes Vorticity Equation by taking the curl of the 2nd Navier-Stokes equation, ***A*(*x*,*y*,*t*)**. The curl is defined as

∇ x *f*(*x*, *y*) which, in this case, is the cross product of $\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}$

and $A(x,y,t) = \begin{bmatrix} B(x,y,t) \\ C(x,y,t) \end{bmatrix}$. To take the curl of two 2x1

matrices, they must be arranged in a matrix that looks like this [9].

$$\begin{vmatrix} \hat{i} & \hat{j} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \\ B(x,y,t) & C(x,y,t) \end{vmatrix}$$

In this matrix, *i* and *j* represent unit vectors in each dimension. The curl is determinant of this 2x2 matrix, which is shown by the formula below

$$\det \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

This formula can be applied to the Navier Stokes equations to yield:

$$\det \begin{vmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \\ B(x,y,t) & C(x,y,t) \end{vmatrix} = \frac{\partial C}{\partial x} - \frac{\partial B}{\partial y}$$

Differentiating the ***C***(***x***,***y***,***t***) with respect to x yields:

$$\frac{\partial C}{\partial x} = \frac{\partial}{\partial t}\frac{\partial V}{\partial x} + U\frac{\partial^2 V}{\partial x^2} + \frac{\partial U}{\partial x}\frac{\partial V}{\partial x} + V\frac{\partial^2 V}{\partial x \partial y} + \frac{\partial V}{\partial x}\frac{\partial V}{\partial y} + \frac{1}{\rho}\frac{\partial^2 P}{\partial x \partial y} - \frac{\mu}{\rho}(\frac{\partial^2}{\partial x^2}\frac{\partial V}{\partial x} + \frac{\partial^2}{\partial y^2}\frac{\partial V}{\partial x})$$

Note that the product rule ***(μv)' = μ'v + μv'*** is used to differentiate the 2nd and 3rd terms. The same process can be done to differentiate ***B***(***x***,***y***, ***t***) with respect to y:

$$\frac{\partial B}{\partial x} = \frac{\partial}{\partial t}\frac{\partial U}{\partial y} + U\frac{\partial^2 U}{\partial x \partial y} + \frac{\partial U}{\partial x}\frac{\partial U}{\partial y} + V\frac{\partial^2 U}{\partial y^2} + \frac{\partial U}{\partial y}\frac{\partial V}{\partial y} + \frac{1}{\rho}\frac{\partial^2 P}{\partial x \partial y} - \frac{\mu}{\rho}(\frac{\partial^2}{\partial x^2}\frac{\partial U}{\partial y} + \frac{\partial^2}{\partial y^2}\frac{\partial U}{\partial y})$$

Subtracting the first equation from the second yields:

$$\frac{\partial}{\partial t}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) + U\frac{\partial}{\partial x}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) + \frac{\partial U}{\partial x}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) + V\frac{\partial}{\partial y}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) + \frac{\partial V}{\partial y}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) - \frac{\mu}{\rho}(\frac{\partial^2}{\partial x^2}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) + \frac{\partial^2}{\partial y^2}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}))$$

This can further simplify to:

$$\frac{\partial}{\partial t}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) + U\frac{\partial}{\partial x}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) + (\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y})(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) + V\frac{\partial}{\partial y}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) - \frac{\mu}{\rho}(\frac{\partial^2}{\partial x^2}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) + \frac{\partial^2}{\partial y^2}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}))$$

The term $(\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y})(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y})$ is equal to zero since, according to the first Navier-Stokes equation, $\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} = 0$.

This leaves the remaining terms to be as follows [9]:

$$\frac{\partial}{\partial t}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) + U\frac{\partial}{\partial x}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) + V\frac{\partial}{\partial y}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) - \frac{\mu}{\rho}(\frac{\partial^2}{\partial x^2}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}) + \frac{\partial^2}{\partial y^2}(\frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}))$$

Integrating the Stream Function and Vorticity Variables

According to the continuity function, $\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} = 0$. it can be proved that U and V are $\frac{\partial \psi}{\partial y}$ and $-\frac{\partial \psi}{\partial x}$

respectively, where ***ψ*** is the streamfunction. This a useful substitution because it implies that for any function, ***ψ***, the continuity equation will be satisfied. This can be proved as follows. If

$U = \frac{\partial \psi}{\partial y}$ and $V = -\frac{\partial \psi}{\partial x}$, then substituting the variables into the continuity equation yields:

$$\frac{\partial^2 \psi}{\partial x \partial y} - \frac{\partial^2 \psi}{\partial x \partial y} = 0$$

Thus, for any function, ***ψ*** , the continuity equation is satisfied

Vorticity (ϖ) can be defined as $\omega = \frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}$ to substitute the repeated binomial in the Navier Stokes equation with a single variable.

This greatly simplifies the equation into the following:

$$\frac{\partial \omega}{\partial t} + \frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y} - \frac{\mu}{\rho}(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}) = 0$$

This can be rearranged to solve for $\frac{\partial \omega}{\partial t}$ as shown below [9]:

$$\frac{\partial \omega}{\partial t} = \frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y} - \frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} + \frac{\mu}{\rho}(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2})$$

Deriving Poisson's Equation

Poisson's Equation is a powerful equation that relates streamfunction to vorticity. It is derived by substituting U and V with $\frac{\partial \psi}{\partial y}$ and $-\frac{\partial \psi}{\partial x}$ (Im) as shown below:

$$\omega = \frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}$$

$$\omega = -\frac{\partial^2 V}{\partial x^2} - \frac{\partial^2 U}{\partial y^2}$$

$$-\omega = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}$$

Deriving Taylor Series Approximations

To approximate derivatives in equations, the Taylor Series must be used. The Taylor series is an infinite sum composed of a function's derivatives about an arbitrary point. This polynomial is equivalent to the original function. The Taylor Series formula is:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)(x - x_0)^2}{2!} + \frac{f'''(x_0)(x - x_0)^3}{3!} + \dots$$

Derivatives are approximated by calculating the rate of change between two infinitely close points. Thus, the term, $x - x_0$, must be small so that the series can converge. This term can be substituted with another variable h where $h = x - x_0$. This also means that $x = x_0 + h$. The resulting equation is shown below:

$$f(x_0 + h) = f(x) + hf'(x_0) + \frac{h^2 f''(x_0)}{2!} + \frac{h^3 f'''(x_0)}{3!} + \dots$$

However, these sums are infinite so when using them in calculations, they must be truncated. When the sum is truncated after a certain amount of terms, it becomes an approximation of the original function. Since this is an approximation - not an exact value - there is an inevitable error present. To express the error, a truncated Taylor Series is said to be nth-order accurate when terms of degree n and above are removed from the infinite sum. Error terms are represented as $O(h^n)$ where n is the order accuracy. Order accuracy is used to quantify error because, even though the exact error is not known, the order of magnitude of the error is known [10].

Consider this 2nd-order accurate Taylor Series expansion:

$$f(x_0 + h) = f(x_0) + hf'(x_0) + O(h^2)$$

This is known as the forward difference approximation and can be rearranged to approximate the first derivative.

$$f(x_0 + h) - f(x_0) - O(h^2) = hf'(x_0)$$

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - O(h)$$

As h approaches zero, the error term, $O(h)$, approaches 0 as well. Thus, a perfect approximation of the first derivative would be the limit as h approaches zero of the formula above.

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

Interestingly enough, this formula is also the formula to calculate the derivative from first principles which highlights the relationship between this Taylor Series expansion and the derivative!

Suppose instead of adding h, $-h$ was added. The Taylor Series would be shown as follows

$$f(x_0 - h) = f(x_0) - hf'(x_0) + \frac{h^2 f''(x_0)}{2!} - \frac{h^3 f'''(x_0)}{3!} + \dots$$

This equation can also be rearranged create the backwards difference formula — used to approximate the first derivative.

$$f(x_0 - h) - f(x_0) - O(h^2) = -hf'(x_0)$$

$$f'(x_0) = \frac{f(x_0) - f(x_0 - h)}{h} + O(h)$$

The forwards difference formula overshoots the true derivative, which is proved since, $O(h)$ must be subtracted from the approximation. Furthermore, the backwards difference formula undershoots the true derivative since $O(h)$ is added to the approximation. Thus, the Taylor Series expansions of $f(x_0+h)$ and $f(x_0-h)$ can be subtracted from each other to create the central difference formula.

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2 f''(x_0}{2!} + \frac{h^3 f'''(x_0)}{3!} + \dots$$

$$f(x_0 - h) = f(x_0) - hf'(x_0) + \frac{h^2 f''(x_0)}{2!} - \frac{h^3 f'''(x_0)}{3!} + \dots$$

$$f(x_0 + h) - f(x_0 - h) = 2hf'(x_0) + O(h^3)$$

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} - O(h^2)$$

This same process can be applied to find the central difference approximation for the second derivative as shown below.

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2 f''(x_0)}{2!} + \frac{h^3 f'''(x_0)}{3!} + \dots$$

$$f(x_0 - h) = f(x_0) - hf'(x_0) + \frac{h^2 f''(x_0)}{2!} - \frac{h^3 f'''(x_0)}{3!} + \dots$$

$$f(x_0 + h) - f(x_0 - h) = 2f(x_0) + 2\frac{h^2 f''(x_0)}{2!} + O(h^4)$$

$$f(x_0 + h) - 2f(x_0) + f(x_0 - h) = h^2 f''(x_0) + O(h^4)$$

$$f''(x_0) = \frac{f(x_0 + h) - 2f(x) + f(x_0 - h)}{h^2} - O(h^2)$$

In fluid dynamics, it is convention to set $0 < h < 1$ since the value must be very small. Thus, as the degree of h increases, the error term $O(h^n)$ becomes smaller. Therefore, it can be concluded that the central difference formula is the most accurate since the error term will be the smallest [10].

Discretization and Substitution

The 2D grid can be discretized into a square grid within the cavity as follows where i and j refer to x and y dimensions respectively Δt refers to the change in time and Δt refers to the step size in the x and y spatial dimensions. The terms Δt, Δh are all analogous to h in the Taylor Series Expansions but they represent different physical qualities of the problem at hand. The variables to follow are denoted in the form $\omega_{i,j}^{t}$ where i, j are the x and y dimensions and r represents the time step.

In the Navier-Stokes Vorticity Equation, the term $\frac{\partial \omega}{\partial t}$ can be approximated using the forwards difference approximation while the rest of the terms can be approximated using the more accurate central difference approximation. The reason for this is because the first derivative central difference approximation uses points

$f(t+1)$ and $f(t-1)$ but, at $t = \cap$, there is no $f(t-1)$ time step. Approximating the Navier-Stokes Vorticity Equation yields the following:

$$\frac{\omega_{i,j}^{t+1} - \omega_{i,j}^{t}}{\Delta t} = -\left(\frac{\psi_{i,j+1}^{t} - \psi_{i,j-1}^{t}}{2\Delta h}\right)\left(\frac{\omega_{i+1,j}^{t} - \omega_{i-1,j}^{t}}{2\Delta h}\right) + \left(\frac{\psi_{i+1,j}^{n} - \psi_{i-1,j}^{t}}{2\Delta h}\right)\left(\frac{\omega_{i,j+1}^{t} - \omega_{i,j-1}^{t}}{2\Delta h}\right) + \frac{\mu}{\rho}\left(\frac{\omega_{i+1,j}^{t} + \omega_{i,j+1}^{t} - 4\omega_{i,j}^{t} + \omega_{i-1,j}^{t} + \omega_{i,j-1}^{t}}{\Delta h^2}\right)$$

This can be rearranged to solve for vorticity at the next time step:

$$\omega_{i,j}^{t+1} = \omega_{i,j}^{t} + \Delta t\left[\left(\frac{\psi_{i+1,j}^{t} - \psi_{i-1,j}^{t}}{2\Delta h}\right)\left(\frac{\omega_{i,j+1}^{t} - \omega_{i,j-1}^{t}}{2\Delta h}\right) - \left(\frac{\psi_{i,j+1}^{t} - \psi_{i,j-1}^{t}}{2\Delta h}\right)\left(\frac{\omega_{i+1,j}^{t} - \omega_{i-1,j}^{t}}{2\Delta h}\right) + \frac{\mu}{\rho}\left(\frac{\omega_{i+1,j}^{t} + \omega_{i,j+1}^{t} - 4\omega_{i,j}^{t} + \omega_{i-1,j}^{t} + \omega_{i,j-1}^{t}}{\Delta h^2}\right)\right]$$

Poisson's equation, $-\omega = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}$ can also be approximated using central difference approximations and rearranged for $\psi_{i,j}^{t}$ as shown below [9]:

$$-\omega_{i,j}^{t} = \frac{\psi_{i+1,j}^{t} + \psi_{i-1,j}^{t} + \psi_{i,j+1}^{t} + \psi_{i,j-1}^{t} - 4\psi_{i,j}^{t}}{\Delta h^2}$$

$$\psi_{i,j}^{t} = \frac{\Delta h\,\omega_{i,j}^{t} + \psi_{i+1,j}^{t} + \psi_{i-1,j}^{t} + \psi_{i,j+1}^{t} + \psi_{i,j-1}^{t}}{4}$$

Boundary Conditions

Except for the top boundary which is moving horizontally at a speed of 1, the horizontal and vertical velocities are all 0 at the boundaries thanks to the impermeability and no-slip boundary conditions. Using Poisson Equation, $-\omega = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}$, it can be concluded that for the vertical boundaries on the left and right, $\frac{\partial^2 U}{\partial y^2}$ will be 0. This is because if a point on a vertical boundary is moved up or down, the horizontal velocity will always remain a constant zero thanks to the impermeability boundary condition. Likewise, for the top and bottom boundary, it can be concluded that $\frac{\partial^2 V}{\partial x^2}$ will always be zero because if a point is moved left or right, the vertical velocity will always remain constant. Thus, Poisson's equation can be simplified at each boundary to be:

$$-\omega_{top} = \frac{\partial^2 U}{\partial y^2}$$

$$-\omega_{bottom} = \frac{\partial^2 U}{\partial y^2}$$

$$-\omega_{left} = \frac{\partial^2 V}{\partial x^2}$$

$$-\omega_{right} = \frac{\partial^2 V}{\partial x^2}$$

Additionally, Taylor Series expansions for the stream function at points adjacent to the boundary can be generated. If a cavity is discretized into a grid with nodes from 1 to n in the x and y dimensions where 1 and n are both points at the boundaries, the Taylor Series expansions for points adjacent to boundaries are:

$$\text{Top: } \psi_{i,n-1} = \psi_{i,n} - \frac{\partial \psi_{i,n}}{\partial y}\Delta h + \frac{\partial^2 \psi_{i,n}}{\partial x^2}\frac{\Delta h^2}{2!} + \ldots$$

$$\text{Bottom: } \psi_{i,2} = \psi_{i,1} + \frac{\partial \psi_{i,1}}{\partial y}\Delta h + \frac{\partial^2 \psi_{i,1}}{\partial x^2}\frac{\Delta h^2}{2!} + \ldots$$

$$\text{Left: } \psi_{2,j} = \psi_{1,j} + \frac{\partial \psi_{1,j}}{\partial y}\Delta h + \frac{\partial^2 \psi_{1,j}}{\partial y^2}\frac{\Delta h^2}{2!} + \ldots$$

$$\text{Right: } \psi_{n-1,j} = \psi_{n,j} + \frac{\partial \psi_{n,j}}{\partial y}\Delta h + \frac{\partial^2 \psi_{n,j}}{\partial y^2}\frac{\Delta h^2}{2!} + ..$$

Using the equations $U = \frac{\partial \psi}{\partial y}$, $V = -\frac{\partial \psi}{\partial x}$, and the vorticity at the boundaries, these expressions can be simplified to be:

$$\text{Top: } \psi_{i,n-1} = \psi_{i,n} - U_{i,n}\Delta h - \omega_{i,n}\frac{\Delta h^2}{2!} + \ldots$$

$$\text{Bottom: } \psi_{i,2} = \psi_{i,1} + U_{i,1}\partial y\Delta h - \omega_{i,1}\frac{\Delta h^2}{2!} + \ldots$$

$$\text{Left: } \psi_{2,j} = \psi_{1,j} - V_{1,j}\partial y\Delta h - \omega_{1,j}\frac{\Delta h^2}{2!} + \ldots$$

$$\text{Right: } \psi_{n-1,j} = \psi_{n,j} + V_{n,j}\partial y\Delta h - \omega_{n,j}\frac{\Delta h^2}{2!} + \ldots$$

If $U = \frac{\partial \psi}{\partial y}$ and $V = -\frac{\partial \psi}{\partial x}$ both need to be 0 at every point on the boundary, ψ must be a constant since the derivative of a constant is 0. Thus, ψ can be any constant however, for the sake of this paper, $\psi = 0$ at the boundary. This is done so that the first terms in the boundary condition Taylor Series' above can be simplified. The resulting equations are:

$$\text{Top: } \psi_{i,n-1} = -U_{i,n}\Delta h - \omega_{i,n}\frac{\Delta h^2}{2!} + \ldots$$

$$\text{Bottom: } \psi_{i,2} = U_{i,1}\partial y \Delta h - \omega_{i,1}\frac{\Delta h^2}{2!} + \ldots$$

$$\text{Left: } \psi_{2,j} = -V_{1,j}\partial y \Delta h - \omega_{1,j}\frac{\Delta h^2}{2!} + \ldots$$

$$\text{Right: } \psi_{n-1,j} = V_{n,j}\partial y \Delta h - \omega_{n,j}\frac{\Delta h^2}{2!} + \ldots$$

Each series can be truncated to become 3rd-order accurate. The velocity terms, U and V, for the bottom, left and right boundary are all equal to 0 since the walls are stationary. Finally, **ω** can be isolated for [9]:

$$\text{Top: } \omega_{i,n} = -\frac{2\psi_{i,n-1}}{\Delta h^2} - \frac{2U_{i,n}}{\Delta h}$$

$$\text{Bottom: } \omega_{i,1} = -\frac{2\psi_{i,2}}{\Delta h^2}$$

$$\text{Left: } \omega_{1,j} = -\frac{2\psi_{2,j}}{\Delta h^2}$$

$$\text{Right: } \omega_{n,j} = -\frac{2\psi_{n-1,j}}{\Delta h^2}$$

Using all the discretized equations a computational solution can be determined.

Computational Solution

From here on out, the problem becomes too complex to solve on paper thus, Matlab must be used to carry out the remaining calculations. An iterative method is needed to approximate the function. There are 3 main types: Jacobi, Gauss-Seidel and Successive-over-relaxation (SOR). The method this paper will use is SOR because it requires the least amount of iterations. To comprehend how this method works, one must first understand the Gauss-Seidel Method. When given a set of equations, the Gauss-Seidel method start with the first equation and set every variable to be zero except for one which will be known as a. The method will then solve for a assuming that every variable is zero. Then, all the variables in the second equation are set to zero except the variable to be solved, b, for and any variables already computed. The variable, b, is updated as the most recent solution. This process loops over the set of equations until all the variable values begin to converge. Once all the variables change by a small specified error each iteration (usually a very small number like 10^{-7}), the method is completed as the variables values have converged. The SOR method uses the same steps except it takes a weighted average using the previous values of a variable when calculating the updated value of the variable to converge much faster [11].

The code used to generate the solution is adapted from Joe Molvar's '2D Lid Driven Cavity Flow Solver' on MathWorks file exchange shown in Appendix A [12, 13].

Experiment

Different real-life applications require different levels of accuracy when approximating the Navier-Stokes equations. For example, an aerospace engineer requires much more accuracy when compared to a video game designer. Thus, to put the efficiency of this method to the test, the step size used in the finite difference approximation will be modified and the time taken to run the algorithm will be measured. The algorithm was run with 6 different conditions: 4, 8, 16, 32, 64 and 128 nodes along each side of the cavity (node count). Since the side length of the square is 1, the step size is calculated by using the formula: $\frac{1}{\text{\# of nodes on a side}}$.

U, V and stream function were plotted at the next time step.

Node Count	U-Velocity	V-Velocity	Streamfunction

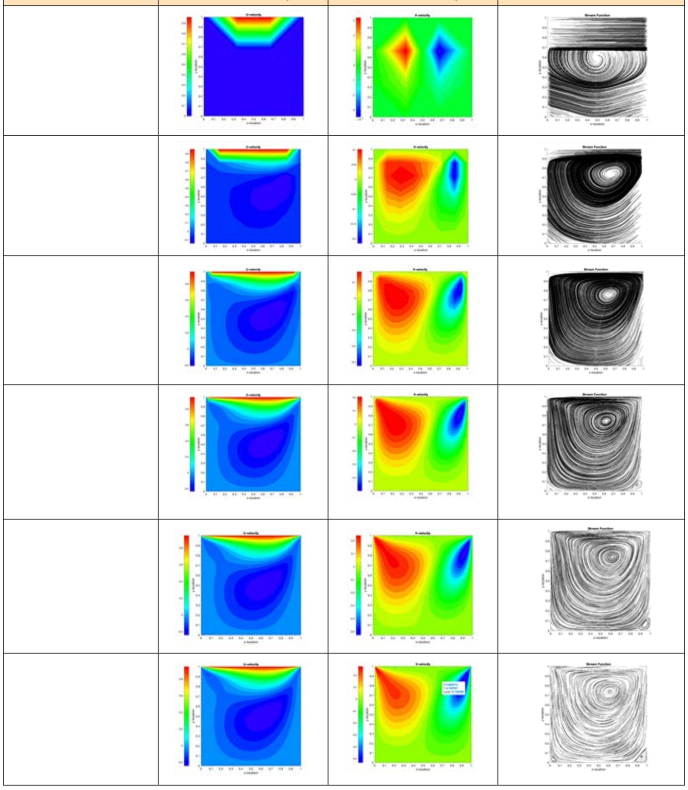

Figure 2: Average Runtime of Finite Difference Method Algorithm Plotted With Exponential Line of Best Fit

Table 1: Lid-Driven Cavity Flow Problem Solved Using the Finite Difference Method at Different Node Counts

Node Count per side	Time (s)					Average (s)
4	3,433	3,616	3,560	3,373	3,424	3,4812
8	4,145	3,997	4,020	4,373	4,061	4,1192
16	4,247	4,121	4,190	4,271	4,215	4,2088
32	4,856	4,900	4,847	4,946	4,857	4,8812
64	7,273	7,191	6,869	7,299	6,999	7,1262
128	25,853	25,801	25,071	25,992	25,272	25,5978

Evaluation

It is important to consider that the finite difference method is not perfect. Each Taylor Series approximation used to replace a differential term caused errors. While the exact error cannot be determined, using the error terms from the centered, forward, and backwards difference formulas, the order of magnitude of the error is known. In the Navier-Stokes Vorticity Equation, all the differential terms that were partial derivatives with respect to x or y were approximated with central difference approximations except for the partial derivative term with respect to t which was approximated with a central difference approximation. The central difference method has an error of $\boldsymbol{O(h2)}$ meaning the error of nodes at the same timestep is $\boldsymbol{O(h2)}$. The term $\frac{\partial\omega}{\partial t}$ was approximated using a forwards difference approximation with an error of $\boldsymbol{O(h)}$. Since $\boldsymbol{O<h<1}$, the greatest error caused by using Taylor Series is $\boldsymbol{O(h)}$. Thus, the error of the Navier-Stokes equations using the finite difference method is $\boldsymbol{O(h)}$ (Yew) [10].

One way to decrease this error is by using a forward difference approximation to approximate $\frac{\partial\omega}{\partial t}$ at the first time step and use a central difference approximation to approximate every successive timestep. It was previously mentioned that the central difference method could not be used to approximate $\frac{\partial\omega}{\partial t}$ because there was not enough information; however after information for 2 timesteps is created, there will be enough meaning the finite difference method can operate at a higher accuracy. Furthermore, higher order accurate Taylor Series approximations can be generated by increasing the number of terms to increase the accuracy. For example, with the following Taylor Series' a centered difference approximation can be generated:

$$f(x_0+h) = f(x) + hf'(x_0) + \frac{h^2 f''(x_0)}{2!} + \frac{h^3 f'''(x_0)}{3!} + \frac{h^4 f^{4)}(x_0)}{4!} + O(h^5)$$

$$f(x_0-h) = f(x) - hf'(x_0) + \frac{h^2 f''(x_0)}{2!} - \frac{h^3 f'''(x_0)}{3!} + \frac{h^4 f^{4)}(x_0)}{4!} - O(h^5)$$

$$f(x_0+2h) = f(x) + 2hf'(x_0) + \frac{4h^2 f''(x_0)}{2!} + \frac{8h^3 f'''(x_0)}{3!} + \frac{16h^4 f^{4)}(x_0)}{4!} + O(h^5)$$

$$f(x_0+2h) = f(x) - 2hf'(x_0) + \frac{4h^2 f''(x_0)}{2!} - \frac{8h^3 f'''(x_0)}{3!} + \frac{16h^4 f^{4)}(x_0)}{4!} - O(h^5)$$

The difference between the first and second equation and the difference between the third and fourth equation can be taken to yield 2 equations

$$f(x_0+h) - f(x_0-h) = 2hf'(x_0) + \frac{2h^3 f'''(x_0)}{3!} + O(h^5)$$

$$f(x_0+2h) - f(x_0-2h) = 4hf'(x_0) + \frac{16h^3 f'''(x_0)}{3!} + O(h^5)$$

The first equation can be multiplied by 8 and the difference of the first and second equation can be calculated. Finally, the equation can be rearranged to approximate the first derivative with an error of $\boldsymbol{O(h^4)}$

$$f'(x_0) = \frac{8f(x_0+h) - 8f(x_0-h) - f(x_0+2h) + f(x_0-2h)}{12h} - O(h^4)$$

Thus, through increasing the order accuracy of Taylor Series approximations, the accuracy of the finite difference method can be increased! Since computers can perform simple arithmetic such as this at a very quick rate, the use of more accurate approximations will have a minuscule effect on the computational time required. However, one flaw is that more points are required for more accuracy. In the example above, 4 points are required compared to the 2 points previously used in centered difference approximations. To conclude, the finite difference method is very flexible because provides variable levels level of accuracy needed to solve different real-world problems — the approximations can be tweaked to favor accuracy or quick computational runtime.

While there is no limit to how large the step size can be, there is a limit to how small the step size can be. This is because the finite difference method is not unconditionally stable meaning under some conditions; the calculated values will begin to diverge. The limit to how small the step size can be is called the Courant's Number (Co) which is given by the formula $Co = \frac{U\Delta t}{\Delta h}$ where U is the velocity, t is the time step and h is the step size. For the finite difference method to be stable, the Courant number at all the nodes must be less than 1. Since velocity multiplied by time is distance, a Courant number greater than 1 means that the fluid has travelled a greater distance than the step size. This causes the fluid to "skip" cells, which results in the divergence. In this example, was 0.001, the greatest velocity was 0.9 and the smallest Δh was $\frac{1}{128}$ therefore, the largest Courant number in the experiment was 0.1152. By solving the inequality $1 > \frac{0.9(0.001)}{\Delta h}$, the greatest value possible was $\frac{1}{1111}$ before an unstable approximation would be reached.

Finally, the experiment showed that this finite difference method algorithm was linear in time complexity. This means that as nodes are added, the time taken for the algorithm to complete increases linearly. This can be shown by graphing the time taken for each condition. Note that instead of representing the nodes per side on the x-axis, the x-axis represents (nodes per side) since the number of nodes inside a square grid increases exponentially as the number of nodes on a side increases.

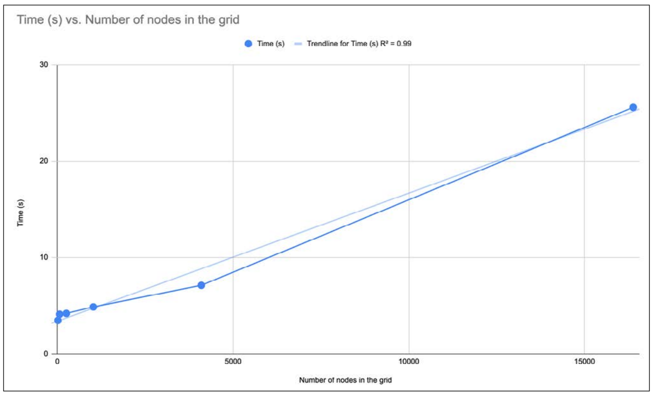

Table 2: Average of 5 trials measuring the runtime of each algorithm with varying numbers of nodes

This graph proves that the algorithm has a linear time complexity. A linear time complexity means that as the number of nodes increases, the time taken to run the algorithm increases at a linear rate. The line of best fit of the graph is $\boldsymbol{0.00133x + 3.39}$ with an $\boldsymbol{R^2} = 0.99$. The extremely high $\boldsymbol{R^2}$ shows that the line of best fit is very accurate in predicting the runtime of the finite difference method algorithm for different number of nodes. In real life, most problems require much more nodes and computations, so, mathematicians can use the line of best fit to predict how long the finite difference method will take to solve [14].

Conclusion

The finite difference method is feasible for simple problems that prioritize speed over accuracy. The finite difference method error is $O(h)$ which is quite high. Additionally, the step size, time step size and velocity are all limited due to Courant's number. The Courant's number limit also implies that using the finite difference method for fast-flowing fluid causes accuracy to be sacrificed to ensure the Courant's number stays below zero. However, the finite difference method is very quick and computationally simplistic. It requires no more than simple algebra and has a linear time complexity. The finite difference method offers itself to predicting ventilation inside homes or to biological uses such as modeling the flow of blood in the human body. Both these processes do not require perfect accuracy and through the finite difference method, they can be computed very quickly. In studying ventilation of homes, simulations can be run many times over to ensure the most optimal air quality for those living in the environment. By studying these simulations, more efficient and safe ventilation systems can be created that improve air quality. Studying the flow of blood with the finite difference method can help pharmaceutical companies understand the bio distribution of a drug they create and understand how to better target specific organs with a computationally "cheap" algorithm. Furthermore, the use of the Taylor Series offers the finite difference method great flexibility. The finite difference method can create approximations that are slightly more accurate by using Taylor Series' with more terms to reduce the error. Further research should be done to compare the finite difference method to the other main methods like the finite element method and finite volume method to create a more balanced review on ways to solve the Navier-Stokes Equations. To conclude, while the finite difference method may not be the most accurate method to solve the Navier-Stokes equations, very quick offers approximations are accurate enough for most general uses.

Appendix A: Joe Molvar's Finite Difference Method Code

```
%%% GIVENS
Nx = 32; L = 1; Wall_Velocity = 1;
% Nodes X; Domain Size; Velocity

mu = 0.01;
Dynamic Viscosity;

dt = 0.001; maxIt = 50000; maxe = 1e-7;
% Time Step; Max iter; Max error

%%% SETUP 1D GRID
Ny = Nx;
h=L/(Nx-1);
x = 0:h:L;
y = 0:h:L;
im = 1:Nx-2; i = 2:Nx-1; ip = 3:Nx;
jm = 1:Ny-2; j = 2:Ny-1; jp = 3:Ny;

%%% PRE-LOCATE MATRIXES
Vo = zeros(Nx,Ny);
St = Vo;
Vop = Vo;
u = Vo;
v = Vo;

%%% SOLVE LOOP SIMILAR USING SUCCESSIVE-OVER-RELAXATION
for iter = 1:maxIt

 %%% CREATE BOUNDARY CONDITIONS
 % Top
 Vo(1:Nx,Ny) = -2*St(1:Nx,Ny-1)/(h^2) - Wall_Velocity*2/h;
 % Bottom
 Vo(1:Nx,1) = -2*St(1:Nx,2) /(h^2);
 % Left
 Vo(1,1:Ny) = -2*St(2,1:Ny) /(h^2);
 % Right
 Vo(Nx,1:Ny) = -2*St(Nx-1,1:Ny)/(h^2);

 %%% PARTIALLY SOLVE VORTICITY TRANSPORT EQUATION
 Vop = Vo;
 Vo(i,j) = Vop(i,j) + (-1*(St(i,jp)-St(i,jm))/(2*h) .*...
 (Vop(ip,i)-Vop(im,j))/(2*h)+(St(ip,j)-St(im,j))/(2*h) .*...
 (Vop(i,jp)-Vop(i,jm))/(2*h)+...
 mu*(Vop(ip,j)+Vop(im,j)-4*Vop(i,j)+Vop(i,jp)+...
 Vop(i,jm))/(h^2))*dt;

 %%% PARTIALLY SOLVE ELLIPTICAL VORTICITY EQUATION
 FOR STREAM FUNCTION
 St(i,j) = (Vo(i,j)*h^2 + St(ip,j) + St(i,jp) +...
 St(i,jm) + St(im,j))/4;

 %%% CHECK FOR CONVERGENCE
 if iter > 10
 error = max(max(Vo - Vop))
 if error < maxe
 break;
 end
 end
end

%%% CREATE VELOCITY FROM STREAM FUNCTION
u(2:Nx-1,Ny) = Wall_Velocity;
u(i,j) = (St(i,jp)-St(i,jm))/(2*h);
v(i,j) = (-St(ip,j)+St(im,j))/(2*h);

%%% PLOTS
cm = hsv(ceil(100/0.7)); cm = flipud(cm(1:100,:));
figure(1);
contourf(x,y,u',23,'LineColor','none');
title('U-velocity');
xlabel('x-location');
ylabel('y-location')
axis('equal',[0 L 0 L]);
colormap(cm); colorbar('westoutside');

figure(2); h=streamline(X,Y,u',v',xstart,ystart,[0.1, 200]);
title('Stream Function');
xlabel('x-location');
ylabel('y-location')
axis('equal',[0 L 0 L]);
set(h,'color','k')
```

References

1. William L Hosch (2022) Navier-Stokes Equation. Britannica. www.britannica.com/science/Navier-Stokes-equation
2. Millennium Problems (2022) Clay Mathematics Institute, www.claymath.org/millennium-problems
3. Frey P, De Buhan M (2008) Finite Difference Method. Laboratoire Jacques-Louis Lions 79-81. https://www.ljll.

math.upmc.fr/frey/cours/UdC/ma691/ma691_ch6.pdf

4. Joel Feldman, Andrew Rechnitzer, Elyse Yeager (2021) Vector Calculus. vol. LP4 Vector Calculus: Creative Commons 146-150. https://math.libretexts.org/Bookshelves/Calculus/CLP-4_Vector_Calculus_(Feldman_Rechnitzer_and_Yeager)/04%3A_Integral_Theorems
5. Nikolaos Katopodes (2019) Free-Surface Flow: Environmental Fluid Mechanics. Butterworth-Heinemann 652-708.
6. Dhondt G (2014) Lid-driven cavity. Web.mit.edu. Retrieved December 30, 2022. https://web.mit.edu/calculix_v2.7/CalculiX/ccx_2.7/doc/ccx/node14.html
7. Connor N (2019) what is Navier-Stokes Equation – Definition. Thermal Engineering. https://www.thermal-engineering.org/what-is-navier-stokes-equation-definition/
8. Al Makky A (2012) Incompressebile Form of the Navier-Stokes Equations in Cartesian Coordinates. Cfd2012. https://cfd2012.com/navier-stokes-equations.html
9. Hong G Im (2001) A Finite Difference Code for the Navier-Stokes Equations in Vorticity/Stream Function Formulation. University of Michigan, Worchester. Power Point Slides. http://www.fem.unicamp.br/~phoenics/SITE_PHOENICS/Apostilas/CFD-1_U%20Michigan_Hong/Lecture05.pdf
10. Yew A (2011) Numerical Differentiation: Finite Differences. APMA 0160 https://www.dam.brown.edu/people/alcyew/handouts/numdiff.pdf
11. Karunanithi S, Gajalakshmi N, Malarvizhi M, Saileshwari M (2018) A Study on Comparison of Jacobi, Gauss Seidel and SOR Methods for the Solution in System of Linear Equations. International Journal of Mathematics Trends and Technology 56: 214-222.
12. Joe Molvar (2020) 2D Lid Driven Cavity Flow Solver. Mathworks. www.mathworks.com/matlabcentral/fileexchange/77025-2d-lid-driven-cavity-flow-solver
13. Molvar Joe (2020) A Fast and Short Matlab Code to Solve the Lid Driven Cavity Flow Problem Using the Vorticity-stream Function Formulation. https://drive.google.com/file/d/1DSGt6Y8QkYMoTSE_3lZzcQYf2Kogx4Pr/view
14. Sjodin B (2016) what's The Difference Between FEM, FDM, and FVM? Machine Design. Retrieved December 29, 2022. https://www.machinedesign.com/3d-printing-cad/fea-and-simulation/article/21832072/whats-the-difference-between-fem-fdm-and-fvm.