Rose-Hulman Undergraduate Mathematics Journal

Volume 23
Issue 1

Article 7

Implementation of A Least Squares Method To A Navier-Stokes Solver

Jada P. Lytch
Francis Marion University, jada.lytch@g.fmarion.edu

Taylor Boatwright
Francis Marion University, taylor.boatwright@g.fmarion.edu

Ja'Nya Breeden
Francis Marion University, janya.breeden@g.fmarion.edu

Implementation of A Least Squares Method To A Navier-Stokes Solver

Cover Page Footnote

This research was funded by the MADE in SC Established Program to Stimulate Competitive Research Grant through the National Science Foundation. This work was also supported by the Francis Marion University Ready to Experience Applied Learning Grant. We would also like to acknowledge Lukas Bystricky's implementation of Newton's Method for the Navier-Stokes equations.

Rose-Hulman Undergraduate Mathematics Journal
VOLUME 23, ISSUE 1, 2022

Implementation of a Least Squares Navier-Stokes Solver

By *Ja'Nya Breeden*, *Taylor Boatwright*, and *Jada Lytch*

Abstract. The Navier-Stokes equations are used to model fluid flow. Example applications include fluid structure interactions in the heart, climate and weather modeling, and flow simulations in computer gaming and entertainment. The equations date back to the 1800s, but research and development of numerical approximation algorithms continues to be an active area. To numerically solve the Navier-Stokes equations we implement a least squares finite element algorithm based on work by Roland Glowinski and his colleagues. The solver is coded using the C++ language and the deal.II finite element library. We investigate convergence rates, apply the least squares solver to the lid driven cavity problem, and discuss results.

1 Navier-Stokes Equations

1.1 Introduction

The Navier-Stokes equations are a set of partial differential equations first developed by French engineer Claude-Louis Navier and Irish physicist George Gabriel Stokes. Navier and Stokes used the work of Swiss Mathematician Leonhard Euler to derive the equations. The equations are used to describe the flow of incompressible, viscous fluids and consist of the momentum and continuity equations.

1.2 Continuity Equation

We derive the Navier-Stokes equations using a control element with the volume V = $\Delta x \Delta y \Delta z$. Figure 1 illustrates changes in mass flux across the element. The variable ρ represents the mass density and $\vec{u} = \langle u_1, u_2, u_3 \rangle$ represents the flow velocity in the x, y, and z directions.

Mathematics Subject Classification. 65M60, 76M10
Keywords. Navier-Stokes, fluid dynamics, finite element method

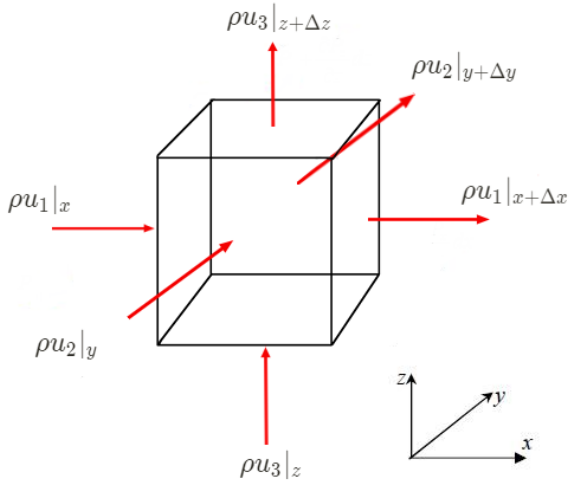


Figure 1: Control volume showing mass flux

Given a control element with volume $\Delta x\Delta y\Delta z$, we know, by conservation of mass, that the mass accumulation equals the mass inflow minus the mass outflow [2]. In three dimensions, the mass accumulation in the control volume is given by the equation:

$$\Delta x\Delta y\Delta z\frac{\partial \rho}{\partial t}\bigg|_{(x,y,z)} \approx \Delta y\Delta z\left[(\rho\vec{u}_1)|_x-(\rho\vec{u}_1)|_{x+\Delta x}\right]+\Delta x\Delta z\left[(\rho\vec{u}_2)|_y-(\rho\vec{u}_2)|_{y+\Delta y}\right] + \Delta x\Delta y\left[(\rho\vec{u}_3)|_z-(\rho\vec{u}_3)|_{z+\Delta z}\right], \tag{1}$$

where $\rho=\rho(x,y,z)$ is the fluid's density and $\vec{u}=\langle\vec{u}_1,\vec{u}_2,\vec{u}_3\rangle$ is the velocity vector field of the fluid with $\vec{u}_j=\vec{u}_j(x,y,z)$ for $1\le j\le 3$.

Dividing both sides of (1) by $\Delta x\Delta y\Delta z$ and letting $\Delta x,\Delta y,\Delta z\to 0$, we use the limit definition of the partial derivatives to obtain the continuity equation

$$\frac{\partial \rho}{\partial t}+\frac{\partial(\rho\vec{u}_1)}{\partial x}+\frac{\partial(\rho\vec{u}_2)}{\partial y}+\frac{\partial(\rho\vec{u}_3)}{\partial z}=0. \tag{2}$$

For steady-state (time-independent), incompressible flow problems having relatively constant density, the equation simplifies to

$$\frac{\partial\vec{u}_1}{\partial x}+\frac{\partial\vec{u}_2}{\partial y}+\frac{\partial\vec{u}_3}{\partial z}=0. \tag{3}$$

1.3 Momentum Equation

The momentum equation (or equation of motion) is based on Newton's second law, where the sum of forces $\vec{F}^{(i)}$ acting on a body equals mass times acceleration,

$$\sum_i \vec{F}^{(i)} = m\vec{a} \quad \text{or} \quad \sum_i \begin{bmatrix} \vec{F}_1^i \\ \vec{F}_2^i \\ \vec{F}_3^i \end{bmatrix} = m \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vec{a}_3 \end{bmatrix}.$$

We consider a fixed control volume element (see Figure 2) as in the continuity equation. We note that the acceleration $\vec{a}$ is with respect to an object's (fluid volume's) moving reference frame (moving with the body). To change to a fixed control volume (fixed reference frame), the total time derivative (or substantial derivative) is used, viewed as a chain rule [2]. Considering the x-component (the first component in Newton's law having $m\vec{a}_1$), the right-hand side of Newton's law has the approximation (the y- and z-component derivations are similar)

$$\begin{aligned} m\vec{a}_1 \approx \quad & \Delta x\Delta y\Delta z[\frac{\partial(\rho\vec{u}_1)}{\partial t}] + \left[\Delta y\Delta z(\rho\vec{u}_1\vec{u}_1)|_{x+\Delta x} - \Delta y\Delta z(\rho\vec{u}_1\vec{u}_1)|_x\right] \\ & + \left[\Delta x\Delta z(\rho\vec{u}_2\vec{u}_1)|_{y+\Delta y} - \Delta x\Delta z(\rho\vec{u}_2\vec{u}_1)|_y\right] \\ & + \left[\Delta x\Delta y(\rho\vec{u}_3\vec{u}_1)|_{z+\Delta z} - \Delta x\Delta y(\rho\vec{u}_3\vec{u}_1)|_z\right]. \end{aligned}$$

The forces $\vec{F}^{(i)}$ acting on the fluid element can be surface forces (such as shear forces) and body forces (like gravitational or magnetic forces). The gravitational body force $\vec{g} = \langle \vec{g}_1, \vec{g}_2, \vec{g}_3 \rangle$ is used below as an example. The left-hand side of Newton's Second Law then takes the form

$$\begin{aligned} \sum_i \vec{F}_1^i \approx \ & \Delta y\Delta z(p+\tau_{xx})|_{x+\Delta x} - \Delta y\Delta z(p+\tau_{xx})|_x \\ & + \left[\Delta x\Delta z(\tau_{yx})|_{y+\Delta y} - \Delta x\Delta z(\tau_{yx})|_y\right] \\ & + \left[\Delta x\Delta y(\tau_{zx})|_{z+\Delta z} - \Delta x\Delta y(\tau_{zx})|_z\right] + \Delta x\Delta y\Delta z\rho\vec{g}_1, \end{aligned}$$

where p is the pressure and τ_{xx}, τ_{yx}, and τ_{zx} are the tensile stress and shear stresses, respectively, acting on the surface of the volume element in the x-direction (figure 2). Similar to the continuity equation, we divide both sides by $\Delta x\Delta y\Delta z$, let Δx, Δy, $\Delta z \to 0$, and use the limit definition for the partial derivatives to obtain

$$\frac{\partial(\rho\vec{u}_1)}{\partial t} + \frac{\partial(\rho(\vec{u}_1)^2)}{\partial x} + \frac{(\rho\vec{u}_2\vec{u}_1)}{\partial y} + \frac{(\rho\vec{u}_3\vec{u}_1)}{\partial z} \quad = -\frac{\partial p}{\partial x} + \frac{\partial\tau_{xx}}{\partial x} + \frac{\partial\tau_{xy}}{\partial y} + \frac{\partial\tau_{xz}}{\partial z} + \rho\vec{g}_1.$$

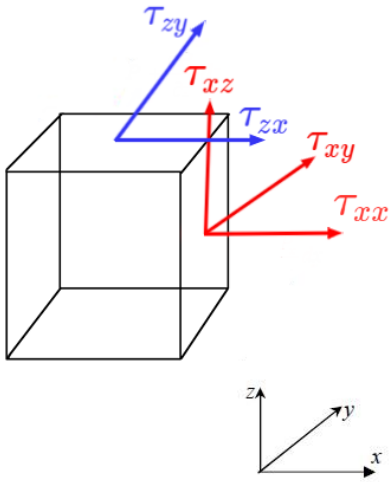


Figure 2: Control volume showing stresses [8]

For a fluid whose behavior is not changing over time (in a steady state), the first term on the left-hand side is zero. We assume the fluid to be incompressible (ρ is constant) and the fluid motion to be independent of time. Further, we assume the fluid to be Newtonian, where stresses are proportional to the symmetric part of the velocity gradient with viscosity μ being the proportionality constant [4]:

$$\rho\left(\vec{u}_1\frac{\partial\vec{u}_1}{\partial x}+\vec{u}_2\frac{\partial\vec{u}_1}{\partial y}+\vec{u}_3\frac{\partial\vec{u}_1}{\partial z}\right) = -\frac{\partial p}{\partial x}+\mu\left(\frac{\partial^2\vec{u}_1}{\partial x^2}+\frac{\partial^2\vec{u}_1}{\partial y^2}+\frac{\partial^2\vec{u}_1}{\partial z^2}\right)+\vec{f}_1.$$

The y and z components are

$$\rho\left(\vec{u}_1\frac{\partial\vec{u}_2}{\partial x}+\vec{u}_2\frac{\partial\vec{u}_2}{\partial y}+\vec{u}_3\frac{\partial\vec{u}_2}{\partial z}\right) = -\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2\vec{u}_2}{\partial x^2}+\frac{\partial^2\vec{u}_2}{\partial y^2}+\frac{\partial^2\vec{u}_2}{\partial z^2}\right)+\vec{f}_2$$

and

$$\rho\left(\vec{u}_1\frac{\partial\vec{u}_3}{\partial x}+\vec{u}_2\frac{\partial\vec{u}_3}{\partial y}+\vec{u}_3\frac{\partial\vec{u}_3}{\partial z}\right) = -\frac{\partial p}{\partial z}+\mu\left(\frac{\partial^2\vec{u}_3}{\partial x^2}+\frac{\partial^2\vec{u}_3}{\partial y^2}+\frac{\partial^2\vec{u}_3}{\partial z^2}\right)+\vec{f}_3$$

respectively, where we represent the sum of all body forces (including the gravitational force) with $\vec{f}=\langle\vec{f}_1,\vec{f}_2,\vec{f}_3\rangle$.
In vector form, the Navier-Stokes equations are

$$\begin{aligned} -\mu\Delta\vec{u} + \rho\vec{u}\cdot\nabla\vec{u} + \nabla p &= \vec{f} \\ \nabla\cdot\vec{u} &= 0. \end{aligned}$$

The organization of the rest of the paper is as follows. In Section 2 we introduce the Stokes equations, review the weak formulation, introduce the finite-dimensional discretization, and construct the system matrix. We are particularly interested in the Stokes system matrix, since it is the main component of Glowinski's least squares algorithm. In Section 3, we review system matrices that result from Newton and Picard iteration methods applied to the Navier-Stokes equations, for comparison with the Stokes system matrix. In Section 4, we outline the least squares algorithm with an emphasis that all five different finite element matrix solves are for Stokes matrix systems.

2 Stokes Equations

2.1 Stokes Equation

The Stokes equations

$$\begin{aligned} -\Delta\vec{u} + \nabla p &= \vec{f} \text{ in } \Omega \\ \nabla\cdot\vec{u} &= 0 \text{ in } \Omega \\ \vec{u} &= \vec{g} \text{ on } \Gamma, \end{aligned}$$

result from dropping the advection term $\vec{u}\cdot\nabla\vec{u}$ of the Navier-Stokes equations. Stokes equations can be used to model slow moving or creeping fluid flow. Note that Ω is the domain of the fluid and Γ is the boundary of the fluid domain. The third equation indicates that the velocity is controlled and equal to $\vec{g}$ at the boundary Γ of the fluid domain. Note, the boundary condition $\vec{g}$ is not to be confused with the gravitational force.

2.2 Weak Formulation

We use the Galerkin finite element method (FEM) to obtain the system of equations that we solve. We step through the mechanics of FEM for the Stokes equations below, since this relates directly to our least squares algorithm code. Our first step is to obtain the weak formulation. We multiply the Stokes equations by smooth functions $\vec{v}$ and q, called test functions. We integrate over the domain and apply integration by parts on different terms, reducing differentiability requirements on the unknowns (the velocity and pressure).

$$\int_\Omega \nabla \vec{v} \cdot \nabla \vec{u} \;\; d\Omega - \int_\Omega p \nabla \cdot \vec{v} \;\; d\Omega = \int_\Omega \vec{f} \cdot \vec{v} \;\; d\Omega.$$
$$\int_\Omega q \nabla \cdot \vec{u} \;\; d\Omega = 0$$

.

The velocity $\vec{u}$ components belong to the space

$$\mathrm{H}^1(\Omega) := \left\{ f : \int_\Omega f^2 d\Omega < \infty \text{ and } \int_\Omega \nabla f \cdot \nabla f \, d\Omega < \infty \right\}$$

and the pressure p belongs to the space [7]

$$\mathrm{L}^2(\Omega) := \left\{ f : \int_\Omega f^2 d\Omega = 0 \right\}$$

.

The weak formulation can be rewritten as

$$\begin{aligned} a(\vec{u}, \vec{v}) + b(\vec{v}, p) &= (\vec{f}, \vec{v}) && \forall \vec{v} \in \mathrm{H}^1(\Omega) \\ b(\vec{u}, q) &= 0 && \forall q \in \mathrm{L}^2(\Omega) \end{aligned}$$

where

$$\begin{aligned} a(\vec{u}, \vec{v}) &= \nu \int \nabla \vec{u} : \nabla \vec{v} \;\; d\Omega && \forall \vec{u}, \vec{v} \in \vec{\mathrm{H}}^1(\Omega) \\ b(v, q) &= -\int q \nabla \cdot \vec{v} \;\; d\Omega && \forall \vec{u} \in \vec{\mathrm{H}}^1(\Omega) \text{ and } q \in \mathrm{L}^2(\Omega). \end{aligned}$$

The : symbol is a double dot product of the gradient vectors (we think of these as Jacobians) defined by

$$\nabla \vec{u} : \nabla \vec{v} = \sum_{i=1}^{3} \nabla \vec{u}_i \cdot \nabla \vec{v}_i = \sum_{i=1}^{3} \frac{\partial \vec{u}_i}{\partial x} \frac{\partial \vec{v}_i}{\partial x} + \frac{\partial \vec{u}_i}{\partial y} \frac{\partial \vec{v}_i}{\partial y} + \frac{\partial \vec{u}_i}{\partial z} \frac{\partial \vec{v}_i}{\partial z}$$

2.3 Discretization

We discretize the equations and approximate the infinite dimensional unknowns $\vec{u}$ and p with basis functions from the finite-dimensional subspaces $\mathrm{V}_g^h \subset \mathrm{H}^1(\Omega)$ and $\mathrm{S}^h \subset \mathrm{L}^2(\Omega)$ where h represents the size of the grid discretizing Ω. We note that the boundary condition $\vec{u} = \vec{g}$ on Γ is indicated by the subscript in V_g^h. We use Taylor-Hood finite elements in the discretization, since they are numerically stable [7]. One advantage

of the finite element method is its use of basis functions that have compact support and are non-zero on only a small portion of the domain, which lead to sparse matrices (and potentially less arithmetic operations in the solving process) that are non-zero in only a small portion of entries. To indicate the finite-dimensional approximation with finite elements, the h subscript is used:

$$\begin{aligned} a(\vec{u}^h, \vec{v}^h) + b(\vec{v}^h, p^h) &= (\vec{f}, \vec{v}^h) \qquad && \forall \vec{v}^h \in \mathrm{V}_g^h \\ b(\vec{u}^h, q^h) &= 0 && \forall q^h \in \mathrm{S}^h. \end{aligned} \tag{4}$$

Substituting in the basis function linear combination

$$\vec{u}^h(x) = \sum_{k=1}^{\mathrm{K}} \alpha_k \vec{v}_k(x) \qquad p^h(x) = \sum_{j=1}^{\mathrm{J}} \beta_k q_k(x),$$

we have

$$\begin{aligned} \sum_{k=1}^{\mathrm{K}} \alpha_k a(\vec{v}_k, \vec{v}_l) + \sum_{j=1}^{\mathrm{J}} \beta_j b(\vec{v}_l, q_j) &= (\vec{f}, \vec{v}_l) && \text{for all } 1 \le l \le \mathrm{K} \\ \sum_{k=1}^{\mathrm{K}} \alpha_k b(\vec{v}_k, q_l) &= 0 && \text{for all } 1 \le l \le \mathrm{J}, \end{aligned}$$

2.4 Matrix Formation

For each test (basis) function the corresponding equation can be written as a vector-vector dot product. For the momentum equation we have

$$\begin{bmatrix} a(v_1, v_1) & a(v_2, v_1) & \dots & a(v_{\mathrm{K}}, v_1) \end{bmatrix} * \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ . \\ . \\ \alpha_{\mathrm{K}} \end{bmatrix}.$$

For the continuity equation we have

$$\begin{bmatrix} b(v_1, q_1) & b(v_1, q_2) & \dots & b(v_1, q_{\mathrm{J}}) \end{bmatrix} * \begin{bmatrix} \beta_1 \\ \beta_2 \\ . \\ . \\ \beta_{\mathrm{K}} \end{bmatrix}.$$

Grouping these equations, we obtain the matrix system for Stokes:

$$\begin{bmatrix} \mathrm{A} & \mathrm{B}^{\mathrm{T}} \\ \mathrm{B} & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \mathrm{F} \\ 0 \end{bmatrix}, \tag{5}$$

where the block matrices A and B and block vector F have the form

$$\begin{aligned} \mathrm{A} &= [a_{ij}] = [a(v_j, v_i)] && 1 < i, j < \mathrm{K} \\ \mathrm{B} &= [b_{ij}] = [b(v_j, q_i)] && 1 < i < \mathrm{J} \text{ and } 1 < j < \mathrm{K} \\ \mathrm{F} &= \left[f_i\right] = (f, v_i). \end{aligned}$$

3 Newton and Picard Iterations

Two common methods used to solve the Navier-Stokes equations are the Newton iteration and Picard iteration. The goal of this section is to compare the Stokes matrix system to the Navier-Stokes matrix systems that result from the Newton and Picard iterations. The Newton and Picard iterations are obtained by linearizing the (nonlinear) Navier-Stokes equations. From this point onward, we work in two dimensions, corresponding to our code.

3.1 Newton Iteration

To obtain the Newton iteration, we apply Newton's method

$$\mathrm{H}'(\mathrm{X}_k)(\delta\mathrm{X}) = -\mathrm{H}(\mathrm{X}_k).$$

to the Navier-Stokes equations, where δX represents the difference between the unknown variables for two consecutive iterations in the Newton Method. We iterate to find the zero of the function

$$\vec{\mathrm{H}}(\mathrm{X}) = \begin{bmatrix} \rho(\vec{u}_1 * \vec{u}_{1x} + \vec{u}_2 * \vec{u}_{1y}) + p_x - \mu(\vec{u}_{1xx} + \vec{u}_{1yy} - \vec{f}_1) \\ \rho(\vec{u}_1 * \vec{u}_{2x} + \vec{u}_2 * \vec{u}_{2y}) + p_y - \mu(\vec{u}_{2xx} + \vec{u}_{2yy} - \vec{f}_2) \\ \vec{u}_{\mathrm{X}} + \vec{u}_y \end{bmatrix}.$$

To obtain $\mathrm{H}'(\mathrm{X}_{k+1})(\delta\mathrm{X})$ we use the Gateaux derivative

$$\mathrm{H}'_{\mathrm{G}}(\mathrm{X}_{k+1})(\delta\mathrm{X}) = \lim_{\epsilon \to 0} \frac{\mathrm{H}(\mathrm{X}_{k+1} + \epsilon\delta\mathrm{X}) - \mathrm{H}(\mathrm{X}_{k+1})}{\epsilon}$$

where X is

$$\mathrm{X} = \begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \\ p \end{bmatrix}$$

and

$$X_{k+1} + \epsilon \delta X = \begin{bmatrix} \vec{u}_1^{k+1} \\ \vec{u}_2^{k+1} \\ p^{k+1} \end{bmatrix} + \epsilon \begin{bmatrix} \delta \vec{u}_1 \\ \delta \vec{u}_2 \\ \delta p \end{bmatrix},$$

where

$$\delta X = \begin{bmatrix} \delta \vec{u}_1 \\ \delta \vec{u}_2 \\ \delta p \end{bmatrix} = \begin{bmatrix} \vec{u}_1^{k+1} - \vec{u}_1^{k} \\ \vec{u}_2^{k+1} - \vec{u}_2^{k} \\ p^{k+1} - p^{k} \end{bmatrix}.$$

Substituting the results into Newton's method and simplifying, we obtain

$$-\Delta \vec{u}_{k+1} + \vec{u}_k \cdot \nabla \vec{u}_{k+1} + \vec{u}_{k+1} \cdot \nabla \vec{u}_k + \nabla p_{k+1} = \vec{f} + \vec{u}_k \cdot \nabla \vec{u}_k.$$

3.2 Picard Iteration

The Picard iteration can be obtained by dropping the two terms $\vec{u}_{k+1} \nabla \vec{u}_k$ and $\vec{u}_k \nabla \vec{u}_k$ of the Newton iteration, resulting in

$$-\Delta \vec{u}_{k+1} + \vec{u}_k \cdot \nabla \vec{u}_{k+1} + \nabla p_{k+1} = \vec{f}.$$

3.3 Matrix Systems

Similar to the Stokes equations, we can obtain the weak formulation for both Newton and Picard iterations, discretize the weak forms, and formulate the matrix equations. We state the corresponding system matrices for both the Newton iteration and Picard iteration below.

The matrix equation for the Newton iteration is

$$\begin{bmatrix} A + C_1 + C_2 & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} c_{\vec{u}} \\ c_p \end{bmatrix} = \begin{bmatrix} F \\ g \end{bmatrix}.$$

The matrix equation for the Picard iteration is

$$\begin{bmatrix} A + C_2 & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} c_{\vec{u}} \\ c_p \end{bmatrix} = \begin{bmatrix} F \\ g \end{bmatrix},$$

where $c_{\vec{u}}$ and c_p are the unknown values of the discretized velocity and pressure,

$$c(\vec{w}, \vec{u}, \vec{v}) = \int (\vec{w} \cdot \nabla \vec{u}) \cdot \vec{v} \quad d\Omega \qquad \forall \vec{u}, \vec{v}, \vec{w} \in \vec{H}^1(\Omega)$$

and

$$C_1 = [c_{ij}^1] = c(\vec{u}_k, \vec{u}_{k+1}, \vec{v})$$
$$C_2 = [c_{ij}^2] = c(\vec{u}_{k+1}, \vec{u}_k, \vec{v}).$$

Comparing the matrix systems that result from the Newton and Picard iterations, we see that the Picard iteration has a simpler system matrix (with one less term in the upper left block). However, simpler still is the Stokes matrix, with only one term in the upper left block. For large problems (having many unknowns corresponding to highly refined meshes) matrices are inverted approximately (in iterations) and the Stokes system is easier and costs less to invert using iterative methods in comparison to the Newton and Picard iterations [5]. This fact motivates the development of our code and the implementation of the least squares solver proposed by Glowinski and colleagues [6].

4 Code

4.1 System Initialization

Our least squares solver minimizes the least squares function [7]

$$J(\vec{u}^h) = \frac{1}{2} a(\vec{y}^h(\vec{u}^h), \vec{y}^h(\vec{u}^h)), \tag{6}$$

where $\vec{y}^h$ in V_0^h is obtained from $\vec{u}^h$ by solving the Stokes system

$$\begin{aligned} a(\vec{y}^h, \vec{v}^h) + b(\vec{v}^h, \sigma^h) &= a(\vec{u}^h, \vec{v}^h) + c(\vec{u}^h, \vec{u}^h, \vec{v}^h) - (\vec{f}, \vec{v}^h) \qquad \forall \vec{v}^h \in V_0^h \\ b(\vec{y}^h, q^h) &= 0 \qquad \forall q^h \in S^h. \end{aligned}$$

We note the boundary condition $\vec{y}^h = 0$ on Γ for the system and that it is indicated by the subscript in V_0^h (similar to section 2.3. Prior to solving the system above, a Stokes solve (see equations (4) and (5)) is performed (with boundary conditions $\vec{u}^h = \vec{g}$) to obtain an initial guess $\vec{u}_0^h$. The equations above are then solved (let $\vec{y}_0^h$ correspond to $\vec{u}_0^h$).

Our next step is to determine $\vec{g}^h$ (not to be confused with the boundary condition $\vec{g}$) by solving

$$\begin{aligned} a(\vec{g}^h, \vec{v}^h) + b(\vec{v}^h, \theta^h) &= a(\vec{y}^h, \vec{v}^h) + c(\vec{v}^h, \vec{u}^h, \vec{y}^h) + c(\vec{u}^h, \vec{v}^h, \vec{y}^h) \qquad \forall \vec{v}^h \in V_0^h \\ b(\vec{g}^h, q^h) &= 0 \qquad \forall q^h \in S^h. \end{aligned} \tag{7}$$

We note that this system is also a Stokes system and let the solution be $\vec{g}_0^h$. The next Stokes system solves take place in the main algorithm loop (step 4 in flowchart figure 3). The parameter ρ_n is determined (see figure 3 - step 4A) by searching for a minimum of the least squares function. Recall that the least squares function definition (6) has to do with $\vec{y}^h$. Newton's method is used to find the minimum ([6] explains this in detail). Overall, each pass through the search requires two Stokes system solves shown below. First we solve for $\vec{y}_1^h \in V_0^h$ (and θ_1^h) in

$$\begin{aligned} a(\vec{y}_1^h, \vec{v}^h) + b(\vec{v}^h, \theta_1^h) &= a(\vec{w}^h, \vec{v}^h) + c(\vec{u}^h, \vec{w}^h, \vec{v}^h) + c(\vec{w}^h, \vec{u}^h, \vec{v}^h) \qquad \forall \vec{v}^h \in V_0^h \\ b(\vec{y}_1^h, q^h) &= 0 \qquad \forall q^h \in S^h. \end{aligned}$$

We then solve for $\vec{y}_2^h$ in V_0^h (and θ_2^h) in

$$\begin{aligned} a(\vec{y}_2^h, \vec{v}^h) + b(\vec{v}^h, \theta_2^h) &= c(\vec{w}^h, \vec{w}^h, \vec{v}^h) \qquad \forall \vec{v}^h \in V_0^h \\ b(\vec{y}_2^h, q^h) &= 0 \qquad \forall q^h \in S^h. \end{aligned}$$

The relationship (see [6] for details)

$$\vec{y}^h = \vec{y}_0^h - \rho \vec{y}_1^h + \rho^2 \vec{y}_2^h \tag{8}$$

is used in the Newton's method minimization process to obtain the minimum ρ_n (substituting the relation for $\vec{y}^h$ into the definition of the least squares function). After the minimum is found, $\vec{u}^h$ and $\vec{y}^h$ are updated (step 4B in figure 3). The system (7) is then solved again in step 4C (its norm also gets smaller like the least squares function $J(u^{n+1})$ and it can be checked for falling below a certain tolerance - see [6]). If the relative tolerance on the least squares function is met, the code returns. If not, γ_n and w^{k+1} are determined (steps 4E and 4F) and the loop continues.

Overall, the algorithm implementation is a series of Stokes solves. The steps of this algorithm are a conjugate gradient method applied to the Navier-Stokes equations. Conjugate gradient algorithm properties and the convergence of the algorithm are discussed in [6].

4.2 Algorithm

The following algorithm utilizes the aforementioned Stokes solves to implement the minimization algorithm used in [6]. The variable k represents the iteration number for the Stokes solve.

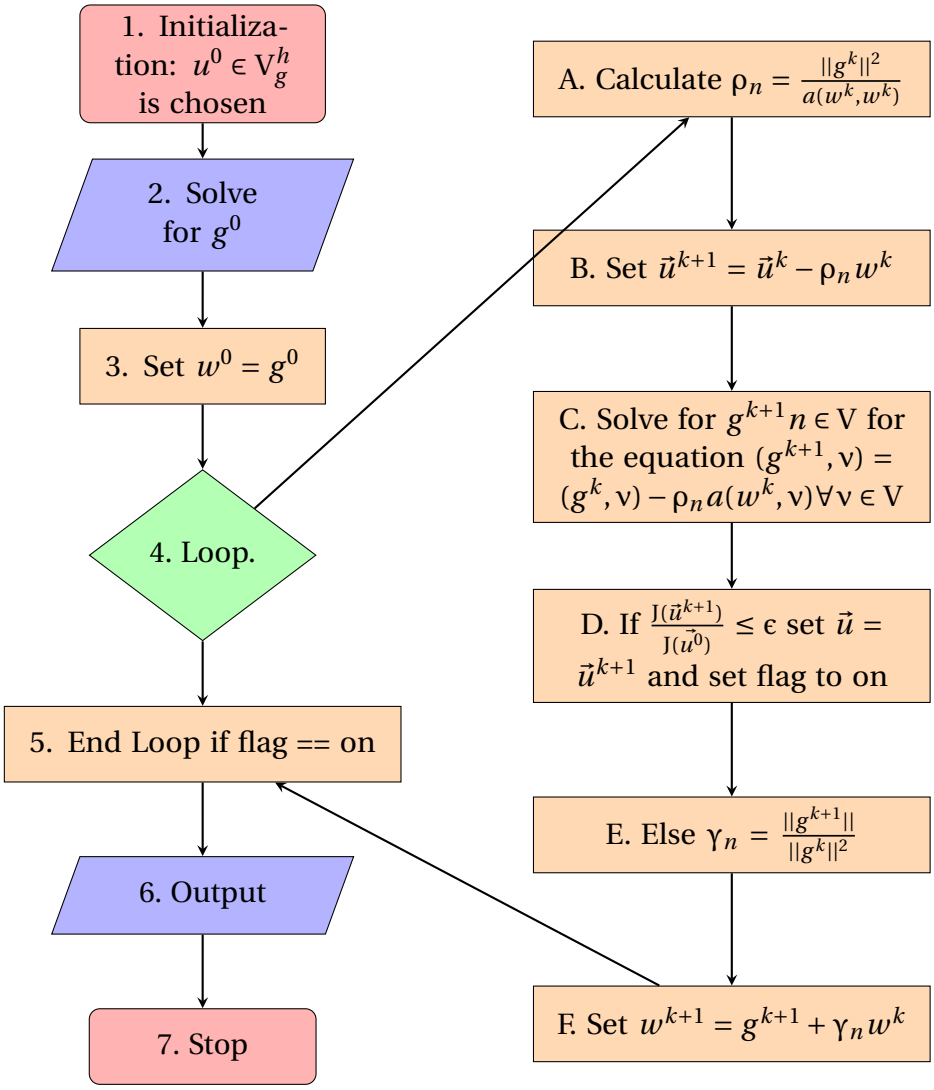


Figure 3: Flowchart for Navier-Stokes least squares solve.

The figure above visualizes the flow of the least squares code, based on the algorithm by Roland Glowinski and his colleagues [6]. We note the inner loop implementation to determine ρ_n using Newton's method with a relative error tolerance of 10^{-6} (as discussed above). The code's outer loop continues until the relative error of J_n is less than the stopping criterion 10^{-06}. The least squares code utilizes the academic library deal.II to assist in the implementation of the finite element method [1, 3].

5 Results and Discussion

5.1 Exact Solution Convergence Rates

The test functions, $\vec{u}$ and p

$$\vec{u} = \begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \end{bmatrix} = \begin{bmatrix} e^x \sin(y) \\ e^x \cos(y) \end{bmatrix}$$
$$p = -\frac{1}{2}e^{2x} + \frac{1}{2}$$

on the unit square domain were chosen so that the right hand side of the exact solution, F, will equal 0:

$$\vec{f} = \begin{bmatrix} -(e^x sin(y) - e^x sin(y)) + (e^x sin(y))(e^x sin(y)) + (e^x cos(y))(e^x cos(y)) - e^{2x} \\ -(e^x cos(y) - e^x cos(y)) + (e^x sin(y))(e^x cos(y)) + (e^x cos(y))(-e^x sin(y)) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Below are the convergence results in Table 1 for the exact solution. The convergence rates were calculated between each pair of consecutive runs using the formula

$$k = \frac{ln(\frac{E_1}{E_2})}{ln(\frac{h_1}{h_2})},$$

where the errors E_i have been measured using the H_1 or L_2 norms. We see that the rates of convergence are as expected since the velocity finite elements are quadratic (order 2) in each coordinate direction and the pressure finite elements are linear (order 1) in each coordinate direction.

Mesh	L_2	Convergence	H_1	Convergence
8x8	2.659e-05	-	1.611e-03	-
16x16	3.061e-06	3.1188	3.778e-04	2.0923
32x32	3.740e-07	3.0329	9.264e-05	2.0279
64x64	4.648e-08	3.0084	2.304e-05	2.0075

Table 1: Velocity Convergence Table

Mesh	L_2	Convergence	H_1	Convergence
8x8	5.122e-03	-	2.639e-01	-
16x16	1.264e-03	2.0187	1.321e-01	0.9984
32x32	3.141e-04	2.0087	6.604e-02	1.0002
64x64	7.828e-05	2.0045	3.302e-02	1.0000

Table 2: Pressure Convergence Tables

5.2 Lid-Driven Cavity Problem

Definition 3.1 The **Reynolds number (Re)** helps predict flow behavior in fluid flow problems. Defined by

$$\mathrm{R}e = \frac{u\mathrm{L}}{\nu} = \frac{\rho u\mathrm{L}}{\mu},$$

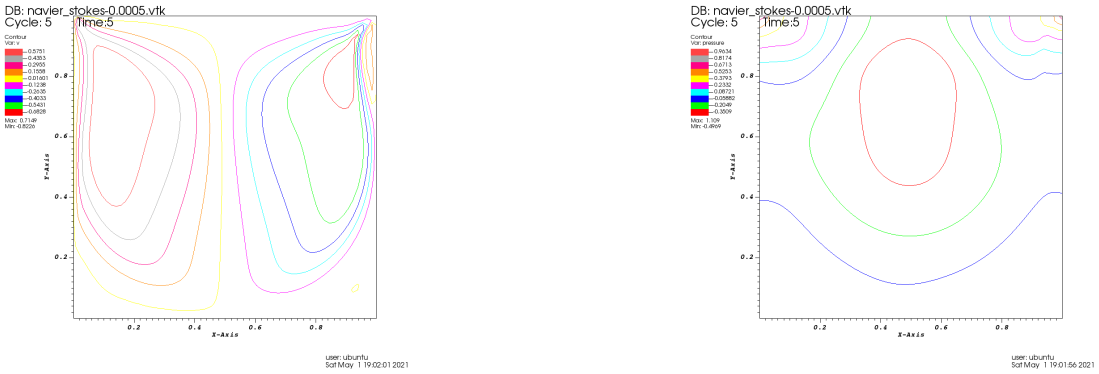
it can be used to determine if a flow is laminar or turbulent.

Definition 3.2: The **Lid-Driven Cavity problem** is a test problem that calculates the velocity of a system where a shear force interacts with the surface of a unit square. We set the velocity to be 1 when $y = 1$ and 0 at all other points.

Table 3 shows results for the lid-driven cavity test problem for Reynolds numbers between 100 and 2,000. It shows the number of iterations needed to converge to an error tolerance of 10^{-6} are given at two different mesh sizes. We note that as the mesh is refined (and the number of mesh cells increases), the number of iterations to reach convergence decreases for each Reynolds number. For a 32x32 mesh at a Reynolds number of 2,000 the results of the lid-driven cavity are as expected and comparable to those of Glowinski [6].

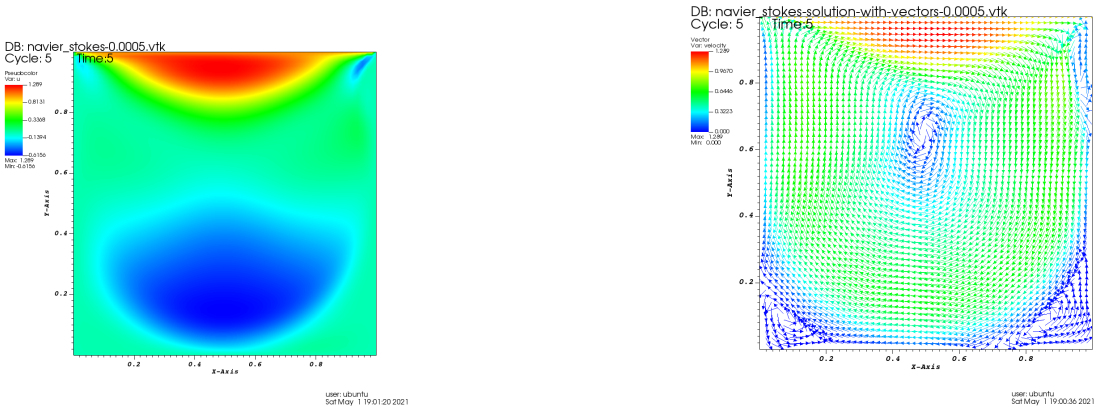
Re	32x32	64x64
100	9	9
200	50	49
500	335	313
800	539	408
1000	708	153
2000	4013	1886

Table 3: Iteration Count



(a) Velocity Contour (b) Pressure Contour

Figure 4: 32x32 Mesh, Reynolds Number 2000



(a) Pseudocolor of Velocity (b) Velocity Vectors

Figure 5: 32x32 Mesh, Reynolds Number 2000

6 Future Work

In the short term, we would like to test the code with higher Reynolds' numbers and finer mesh sizes (like 128x128 and 256x256) and compare them to Glowinski's time-dependent code [6]. If the results are promising, we would like to parallelize the code and utilize an indirect solver to achieve finer mesh sizes at higher Reynolds numbers. The indirect solver would require a preconditioner, and we could therefore exploit the simplicity of the Stokes system matrix. Successful results would lead to developing this code for real-world applications, such as seen in computer graphics gaming simulations or fluid structure interaction modeling. Finally, we are also interested in extending this implementation to the least squares algorithm having time dependence.

References

[1] Daniel Arndt, Wolfgang Bangerth, Denis Davydov, Timo Heister, Luca Heltai, Martin Kronbichler, Matthias Maier, Jean-Paul Pelteret, Bruno Turcksin, and David Wells. The deal.ii finite element library: design, features, and insights. *Computers and Mathematics with Applications*, 81:407–422, 2021.

[2] R. Byron Bird, Warren E. Stewart, and Edwin N. Lightfoot. *Transport Phenomena*. John Wiley and Sons, Inc., 2007.

[3] Lukas Bystricky. Finite elements, 2017.

[4] Neal Coleman. A derivation of the navier-stokes equations. 2010.

[5] Howard C Elman, David J Silvester, and Andrew J Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics and Scie, 2014.

[6] Roland Glowinski. *Multidisciplinary Methods for Analysis Optimization and Control of Complex Systems*. Springer-Verlag Berlin Heidelberg, Inc., 2005.

[7] Max Gunzburger. *Finite Element Methods for Viscous Incompressible Flows*. Academic Press, Inc., 1989.

[8] CFD Online. Navier-stokes equations, 2011.

Ja'Nya Breeden
Francis Marion University
janya.breeden@g.fmarion.edu

Taylor Boatwright
Francis Marion University
taylor.boatwright@g.fmarion.edu

Jada Lytch
Francis Marion University
jada.lytch@g.fmarion.edu