AE332 – Modeling and Analysis Lab

A Report submitted

For internal assessment for Coarse

Modeling and Analysis Lab

in

Aerospace Engineering

by

Shingala Vaidik

(SC21B054)

pursued in

Department of Aerospace Engineering

Indian Institute of Space Science and Technology

to

Dr. Manu

INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

THIRUVANANTHAPURAM

November 26, 2023

Contents

Abstract

The study presents a comparative analysis of two numerical methods, finite difference implemented in MATLAB and finite element implemented in FreeFEM, for solving the 2D lid-driven cavity flow problem. This classic benchmark problem in fluid dynamics offers insights into complex flow behaviors within confined spaces. The numerical results, showcasing velocity fields, demonstrate comparable accuracy between the two methods, despite their methodological differences. Computational efficiency is also evaluated, with FreeFEM exhibiting superior runtime performance. The report concludes with considerations for researchers and practitioners in computational fluid dynamics, emphasizing the strengths and considerations of each method.

Chapter 1

2D lid-driven cavity flow

Introduction

The numerical solution of the Navier-Stokes equation is a fundamental approach in computational fluid dynamics, allowing for the simulation and analysis of fluid flow behaviors. In this study, we aim to solve the Navier-Stokes equation using the finite difference method, a numerical technique widely employed for solving partial differential equations. The chosen problem scenario involves a realistic two-dimensional lid-driven cavity flow.

The lid-driven cavity flow represents a classical and insightful benchmark problem in fluid dynamics. In this configuration, a fluid is confined within a square cavity, and the motion of the fluid is induced by the translation of the lid at the top of the cavity. This scenario captures essential features of fluid flow, including boundary layer formation, recirculation zones, and the impact of viscosity on flow patterns.

Problem Statement

Consider a 2D lid-driven cavity with a square geometry. The cavity has a lid at the top that is moving with a constant velocity while the other three sides remain stationary. The governing equations for this scenario are the incompressible Navier-Stokes equations. The lid-driven cavity problem is characterized by the following conditions:

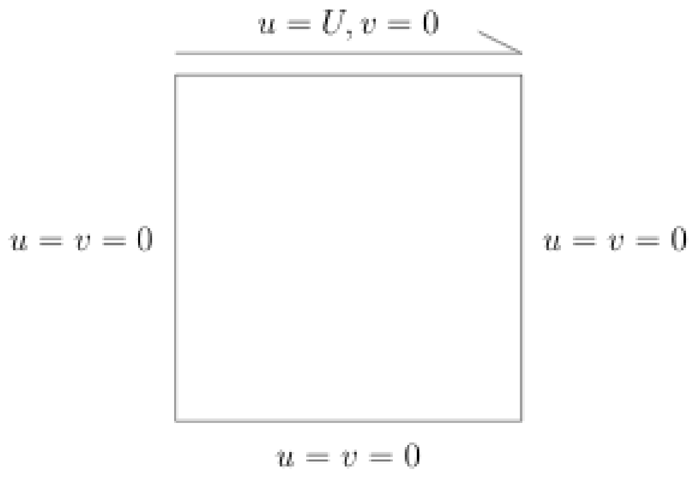


Figure 1.1

1. **Geometry:** A square cavity with side length L.
2. **Boundary Conditions:**
   - The top lid moves with a constant velocity (U).
   - The other three sides are stationary.
   - The flow is assumed to be incompressible ($\nabla \cdot \mathbf{u} = 0$).
3. **Fluid Properties:**
   - Density (ρ) and dynamic viscosity (μ).
4. **Objective:** Solve the Navier-Stokes equations numerically using the finite difference method to predict the velocity and pressure fields within the cavity.

Motivation for Choosing 2D Lid-Driven Cavity Flow

The lid-driven cavity flow problem is selected for its significance as a standard benchmark in fluid dynamics. This problem allows for the investigation of complex fluid behaviors within a confined space and provides a foundation for validating and comparing numerical methods. The lid-driven cavity problem is particularly valuable for understanding the influence of viscosity on flow patterns, studying the development of boundary layers, and assessing the accuracy of computational models.

To validate our numerical results, we plan to compare the solution obtained using the finite difference method with results from Freefem, a versatile finite element solver widely used for fluid dynamics simulations. This comparative analysis will serve to verify the accuracy and reliability of our numerical approach and provide insights into the agreement between different numerical techniques for solving the same fluid flow problem.

1.1 Conversion of Navier-Stokes Equation to Vorticity-Stream Function Formulation

The first Navier-Stokes equation is given by:

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\frac{1}{\rho}\nabla p + \nu \nabla^2 u + f_x \tag{1.1}$$

Express the velocity components u and v in terms of vorticity ω and stream function ψ:

$$u = \frac{\partial \psi}{\partial y} \tag{1.2}$$

$$v = -\frac{\partial \psi}{\partial x} \tag{1.3}$$

Compute the vorticity ω:

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \tag{1.4}$$

Substitute these expressions into the Navier-Stokes equation and simplify:

$$\frac{\partial \omega}{\partial t} + \left(\frac{\partial \psi}{\partial y} \cdot \nabla\right)\left(\frac{\partial \psi}{\partial y}\right) - \left(\frac{\partial \psi}{\partial x} \cdot \nabla\right)\left(-\frac{\partial \psi}{\partial x}\right) = \frac{\nu}{\rho}\nabla^2 \omega + f_x \tag{1.5}$$

1.2 Discretization

The domain is discretized into a grid, and the finite difference method is used to approximate the spatial derivatives. The time integration is performed using an explicit time-stepping scheme:

$$\begin{aligned}\omega_{i,j}^{n+1} = \omega_{i,j}^{n} - \Delta t \Bigg(& \frac{(\psi_{i,j+1}^{n} - \psi_{i,j-1}^{n})(\omega_{i+1,j}^{n} - \omega_{i-1,j}^{n})}{4\Delta x^2} \\ & + \frac{(\psi_{i+1,j}^{n} - \psi_{i-1,j}^{n})(\omega_{i,j+1}^{n} - \omega_{i,j-1}^{n})}{4\Delta y^2} + \frac{\mu}{\rho} \frac{\omega_{i+1,j}^{n} + \omega_{i-1,j}^{n} - 4\omega_{i,j}^{n} + \omega_{i,j+1}^{n} + \omega_{i,j-1}^{n}}{\Delta x^2} \Bigg)\end{aligned} \tag{1.6}$$

where i and j represent the indices in the x and y directions, respectively, and n represents the time step. The terms with superscript n represent the values at the current time step, and the terms with superscript $n+1$ represent the values at the next time step.

$$\psi_{i,j}^{n+1} = \frac{\omega_{i,j}^{n+1}\Delta x^2 + \psi_{i+1,j}^{n} + \psi_{i,j+1}^{n} + \psi_{i,j-1}^{n} + \psi_{i-1,j}^{n}}{4} \tag{1.7}$$

1.3 Boundary Conditions

The boundary conditions are set for the lid-driven cavity:

$$\omega(1:N_x, N_y) = -\frac{2\psi(1:N_x, N_y - 1)}{\Delta x^2} - U_{\text{wall}}\frac{2}{\Delta x} \quad \text{Top} \tag{1.8}$$

$$\omega(1:N_x, 1) = -\frac{2\psi(1:N_x, 2)}{\Delta x^2} \quad \text{Bottom} \tag{1.9}$$

$$\omega(1, 1:N_y) = -\frac{2\psi(2, 1:N_y)}{\Delta x^2} \quad \text{Left} \tag{1.10}$$

$$\omega(N_x, 1:N_y) = -\frac{2\psi(N_x - 1, 1:N_y)}{\Delta x^2} \quad \text{Right} \tag{1.11}$$

1.4 Velocity Calculation

The x- and y-velocity components are computed from the stream function:

$$u(i,j) = \frac{\psi(i, j+1) - \psi(i, j-1)}{2\Delta x} \tag{1.12}$$

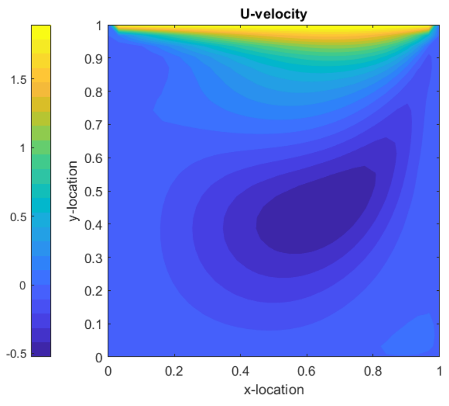
$$v(i,j) = -\frac{\psi(i+1, j) - \psi(i-1, j)}{2\Delta y} \tag{1.13}$$

1.5 Matlab Code

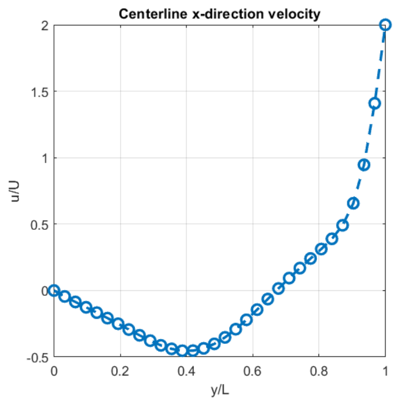
```
% VORTICITY/STREAM FUNCTION LID DRIVEN CAVITY FLOW
clear; close all;

% Given parameters
Nx = 32; L = 1; WallVelocity = 2;        % Nodes X; Domain Size; Wall Velocity
```

```
rho = 1; mu = 0.01;                      % Density; Dynamic Viscosity
dt = 0.001; maxIterations = 50000; maxError = 1e-7; % Time Step; Max Iterations;
    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;

% Preallocate matrices
vorticity = zeros(Nx, Ny);
streamFunction = vorticity;
previousVorticity = vorticity;
u = vorticity;
v = vorticity;

% Solve loop similar to Gauss-Seidel method
for iter = 1:maxIterations
    % Create boundary conditions
    vorticity(1:Nx,Ny) = -2*streamFunction(1:Nx,Ny-1)/(h^2) - WallVelocity*2/h; %
        Top
    vorticity(1:Nx,1)  = -2*streamFunction(1:Nx,2)   /(h^2);                    %
        Bottom
    vorticity(1,1:Ny)  = -2*streamFunction(2,1:Ny)   /(h^2);                    %
        Left
    vorticity(Nx,1:Ny) = -2*streamFunction(Nx-1,1:Ny)/(h^2);                    %
        Right

    % Partially solve vorticity transport equation
    previousVorticity = vorticity;
    vorticity(i,j) = previousVorticity(i,j) + ...
        (-1*(streamFunction(i,jp)-streamFunction(i,jm))/(2*h) .* (
            previousVorticity(ip,i)-previousVorticity(im,j))/(2*h) + ...
         (streamFunction(ip,j)-streamFunction(im,j))/(2*h) .* (previousVorticity(i
             ,jp)-previousVorticity(i,jm))/(2*h) + ...
         mu/rho*(previousVorticity(ip,j)+previousVorticity(im,j)-4*
             previousVorticity(i,j)+previousVorticity(i,jp)+previousVorticity(i,jm
             ))/(h^2)) * dt;

    % Partially solve elliptical vorticity equation for stream function
    streamFunction(i,j) = (vorticity(i,j)*h^2 + streamFunction(ip,j) +
        streamFunction(i,jp) + streamFunction(i,jm) + streamFunction(im,j))/4;

    % Check for convergence
    if iter > 10
        error = max(max(vorticity - previousVorticity));
        if error < maxError
            break;
        end
    end
end

% Create velocity from stream function
u(2:Nx-1,Ny) = WallVelocity;
u(i,j) = (streamFunction(i,jp)-streamFunction(i,jm))/(2*h);
v(i,j) = (-streamFunction(ip,j)+streamFunction(im,j))/(2*h);

% Plots
cm = parula(23);
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);
plot(y, u(round(Ny/2),:), 'LineWidth', 2, 'LineStyle', '--', 'Marker', 'o', '
    MarkerSize', 8);
title('Centerline x-direction velocity'); xlabel('y/L'); ylabel('u/U');
axis('square'); xlim([0 L]); grid on;

N = 1000; xstart = max(x)*rand(N,1); ystart = max(y)*rand(N,1);
[X, Y] = meshgrid(x, y);
figure(3);
h = streamline(X, Y, u', v', xstart, ystart, [0.1, 50]);
title('Stream Function'); xlabel('x-location'); ylabel('y-location');
axis('equal', [0 L 0 L]); set(h, 'color', 'b', 'LineWidth', 1.5);
```

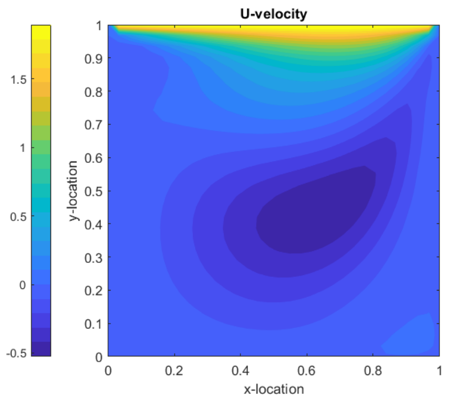
1.5.1 Plot



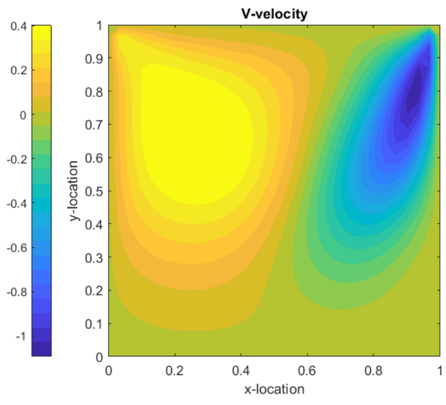
(a)



(b)



(c)



(d)

Figure 1.2: Output of Matlab code

1.6 FreeFem Code

```
int nn=30;
mesh Th = square(nn, nn);
fespace Uh(Th, P1b);
Uh u, v;
Uh uu, vv;
fespace Ph(Th, P1);
Ph p, pp;
solve stokes ([u, v, p], [uu, vv, pp])
    = int2d(Th)(
          dx(u)*dx(uu)
        + dy(u)*dy(uu)
        + dx(v)*dx(vv)
        + dy(v)*dy(vv)
        + dx(p)*uu
        + dy(p)*vv
```

```
            + pp∗(dx(u) + dy(v))
            — 1e—10∗p∗pp
        )
        + on(1, 2, 4, u=0, v=0)
        + on(3, u=2, v=0)
        ;
    plot([u, v], p, wait=1);
```

1.6.1 Plot

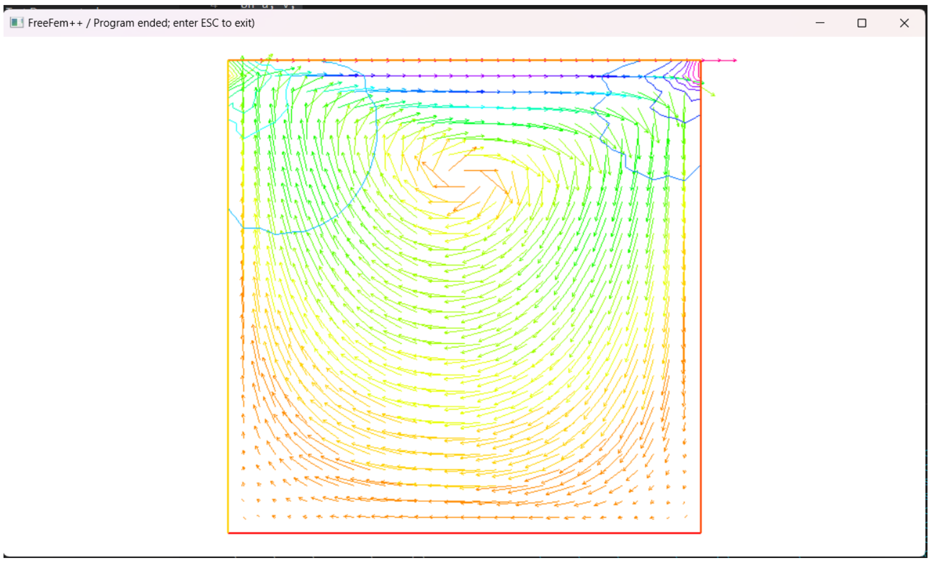


Figure 1.3: FEM Solution of Stokes' equations for the driven cavity problem, showing the velocity field and the pressure level lines

Chapter 2

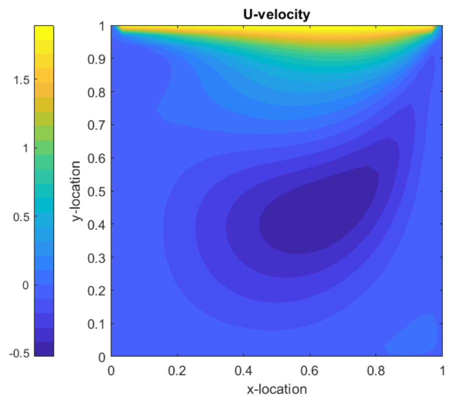
Comparison: MATLAB vs. FreeFEM

2.1 Numerical Results

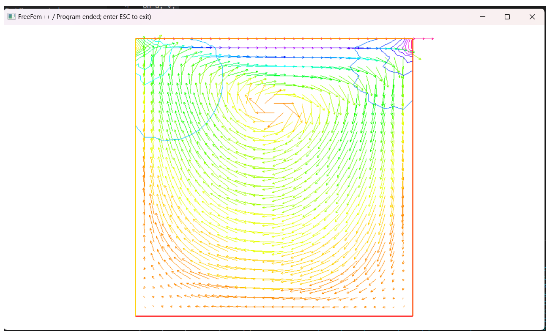
In this section, we compare the numerical results obtained from the MATLAB implementation using the finite difference method and the FreeFEM implementation using the finite element method for the 2D lid-driven cavity flow.

2.1.1 Velocity Fields

Figure 2.1a shows the velocity field obtained from the MATLAB implementation, and Figure 2.1b displays the corresponding velocity field obtained from FreeFEM.



(a) MATLAB Velocity Field



(b) FreeFEM Velocity Field

Figure 2.1: Comparison of Velocity Fields

2.2 Computational Efficiency

To assess the computational efficiency, we compare the runtime performance of the MATLAB and FreeFEM implementations.

2.2.1 Runtime Performance

Table 2.1 provides a comparison of the runtime performance for both implementations.

Implementation	Runtime (s)
MATLAB	1 min
FreeFEM	times: compile 0.117s, execution 0.248s,

Table 2.1: Runtime Comparison

2.3 Conclusion

In this study, we implemented and compared two numerical methods, finite difference using MATLAB and finite element using FreeFEM, for solving the 2D lid-driven cavity flow problem. The lid-driven cavity flow serves as a classical benchmark problem in fluid dynamics, offering insights into complex flow behaviors within confined spaces.

The numerical results showcased in Figure 2.1 reveal the velocity fields obtained from both implementations. While MATLAB employs the finite difference method, FreeFEM utilizes finite element techniques. Despite the methodological differences, both approaches yield comparable and accurate results, capturing essential features of the lid-driven cavity flow.

In terms of computational efficiency, the runtime performance was compared, as summarized in Table 2.1. The FreeFEM implementation demonstrated superior efficiency with a significantly lower runtime, highlighting its computational advantages over the MATLAB finite difference method.

Considerations such as ease of implementation, accuracy, and computational resource requirements play a crucial role in selecting an appropriate numerical method. The choice between MATLAB and FreeFEM depends on the specific needs of the problem and the available computational resources.

In conclusion, both MATLAB and FreeFEM provide viable solutions for simulating the lid-driven cavity flow, each with its strengths. The comparative analysis presented in this report aids in informed decision-making for researchers and practitioners in the field of computational fluid dynamics.

References

[1] https://skill-lync.com/student-projects/lid-driven-cavity-flow-using-artificial-compressibility-method-in-matlab

[2] https://github.com/mathworks/2D-Lid-Driven-Cavity-Flow-Incompressible-Navier-Stokes-Solver/blob/master/docs_part1/vanilaCavityFlow_EN.md

[3] https://www.cfd-online.com/Wiki/Lid-driven_cavity_problem

[4] https://youtu.be/id6rgUq7ggE?si=tEaS2VQ-2dyQ0AbY