Worcester Polytechnic Institute

Department of Mechanical and Materials Engineering

C term March 7th, 2025

ME 5108 Introduction to Computational Fluid Dynamics

Final Project

Solution of 2D Incompressible Navier-Stokes Equation using the Vorticity-Stream function Formulation

Student	David Strom dstrom@wpi.edu
Course Professor	Prof. Aswin Gnanaskandan agnanaskandan@wpi.edu

Table of Contents

List of Figures

1.0 Introduction

Lid-driven cavity flow is a classic fluid dynamics problem in which a square domain contains a viscous fluid. A moving boundary (the 'lid') generates flow within the fluid as it moves across the top of the boundary. This flow, despite its simple setup, can have a very complex behavior. Lid-driven cavity flow is commonly used to validate numerical methods and Computational Fluid Dynamics (CFD) simulations as its solution is very well-studied, even over a broad range of reynold numbers (Shankar & Deshpande, 2000).

Other than using this problem as a test for CFD, lid-driven cavity flow has practical usage in mechanical processes that involve recirculating flows or moving boundaries. Some examples include short-dwell coaters within paper manufacturing as well as in some stages of polymer processing. In these processes, the internal motion of the fluid and the structure of the vortex can affect the coating of the paper or mixture of the fluid. That is why studying the change of these problems within a controlled space, like that of a simple square, helps in understanding similar problems in industrial applications.

Governing Equations

This project focuses on a 2D, incompressible formulation, although the problem can be extended to three dimensions. In 2D, the Navier-Stokes equations can be written in vorticity–streamfunction form. Let ψ denote the streamfunction and ω the vorticity:

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}, u = \frac{\partial \psi}{\partial y}, v = \frac{-\partial \psi}{\partial x}$$

The corresponding **vorticity transport equation** is

$$\frac{\partial \omega}{\partial t} + u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = v(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2})$$

and the **stream function Poisson equation** is

$$\nabla^2 \psi =- \omega$$

Here, u and v are the velocity components in the x and y directions, respectively; v is the kinematic viscosity. For a unit-square domain $(x, y) \in [0, 1] \times [0, 1]$, the boundary conditions impose no-slip on all walls, with the top wall (the "lid") moving at a constant velocity U_{wall} in the x-direction.

Consequently,

1. ψ=0 on the boundaries (stream function is zero at each wall)

2. ω on each boundary is derived via $\omega \approx -\frac{2\psi}{h^2}$ plus extra terms for the moving top edge.

Physical Boundary Conditions

- **No-slip** on the bottom, left, and right walls:
  $u(0,y) = 0,\ v(0,y) = 0,\ u(1,y) = 0, v(1,y) = 0,\ u(x,0) = 0,\ v(x,0) = 0$
- **Moving Lid** on the top: $u(x,1) = U_{Wall}, v(x,1) = 0$

These boundary conditions lead to a flow driven from above, with a dominant primary vortex in the cavity interior and smaller secondary vortices near the corners.

As pressure is not in the vorticity–streamfunction formulation, the solution proceeds by updating vorticity explicitly in time and then solving a Poisson equation for ψ. This method is a popular educational and research model for 2D incompressible flow (Shankar & Deshpande, 2000). In practice, though, higher Reynolds numbers will require either finer grids or more advanced schemes to capture thin boundary layers and complex secondary vortices correctly. Though, this overall framework is still a powerful tool for understanding and simulating internal viscous recirculating flows in CFD.

2.0 Numerical Methods

We discretize the governing equations on a uniform Cartesian grid (i, j) of N×N points, with grid spacing h=L/(N−1). Time-stepping is done explicitly for the vorticity transport equation; the streamfunction equation is solved iteratively each time step.

FTCS Explicit Scheme for Vorticity

We use a forward time–central space (FTCS) discretization:

$$\begin{aligned}\omega_{i,j}^{n+1} = \omega_{i,j}^{n} + \Delta t\Bigg[&-\left(\frac{\psi_{i,j+1}^{n} - \psi_{i,j-1}^{n}}{2h}\right)\left(\frac{\omega_{i+1,j}^{n} - \omega_{i-1,j}^{n}}{2h}\right)\\ &+\left(\frac{\psi_{i+1,j}^{n} - \psi_{i-1,j}^{n}}{2h}\right)\left(\frac{\omega_{i,j+1}^{n} - \omega_{i,j-1}^{n}}{2h}\right)\\ &+\nu\left(\frac{\omega_{i+1,j}^{n} + \omega_{i-1,j}^{n} + \omega_{i,j+1}^{n} + \omega_{i,j-1}^{n} - 4\omega_{i,j}^{n}}{h^2}\right)\Bigg]\end{aligned}$$

Gauss-Seidel for Streamfunction

After updating ω, the streamfunction ψ must satisfy

$$\psi_{i,j} = 0.25\left(\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1} + h^2\omega_{i,j}\right)$$

which is solved iteratively (Gauss–Seidel sweeps) within each time step until convergence.

Boundary Conditions

- **Streamfunction** $\psi = 0$ on all walls.
- **Vorticity** ω on each boundary is derived from $\omega \approx -\frac{2}{h^2}\psi$ plus additional terms for the moving top lid $- 2U_{wall}/h$

Advantages and Disadvantages

- **Advantages**:
    - The vorticity-streamfunction formulation eliminates the pressure term, simplifying boundary conditions.
    - The explicit time-marching for vorticity is straightforward to implement.
- **Disadvantages**:
    - FTCS explicit scheme can impose stringent time-step restrictions (stability constraints).
    - The streamfunction Poisson equation requires iterative solvers, adding computational cost.

3.0 Results & Discussion

3.1 Flow Inside a Square Cavity at Re=100 — Baseline

Firstly, consider N=32 grid points, L=1m, ν=0.01 m2/s, nu=0.01m2/s, Uwall=1m/s. This yields $Re = 100$. We use $\Delta t = 0.001s$ with a tolerance of 10^{-7} for the vorticity RMS error.

3.1.1 Steady-State Solution and Velocity Contours

After the code reaches steady state, a plot of the 2D contours of u and v are generated and displayed along with the ψ contour as seen in figures 1, 2, and 3. These show the expected primary recirculating vortex in the cavity, with u positive near the top wall and v showing downward flow along the right side.

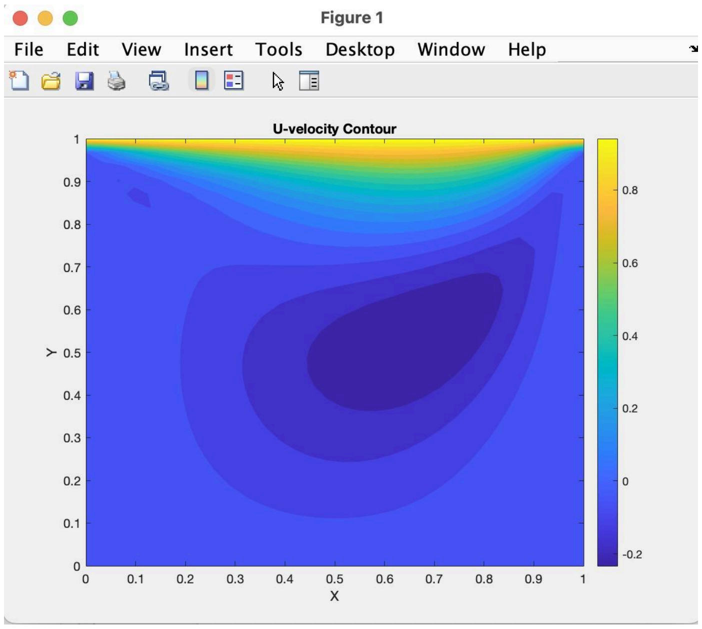

Figure 1: U-velocity Contour for N=32 & Re = 100

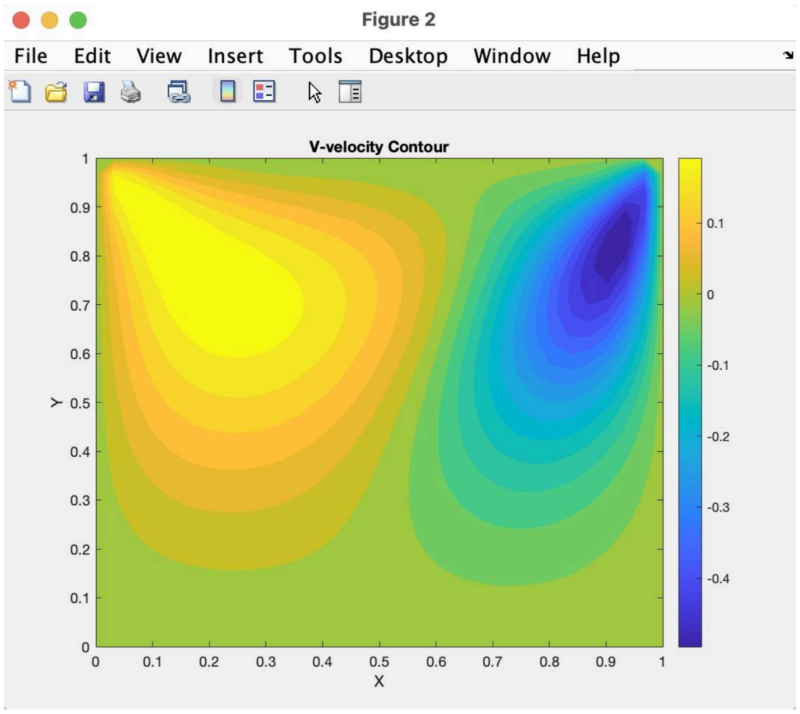

Figure 2: V-velocity Contour for N=32 & Re = 100

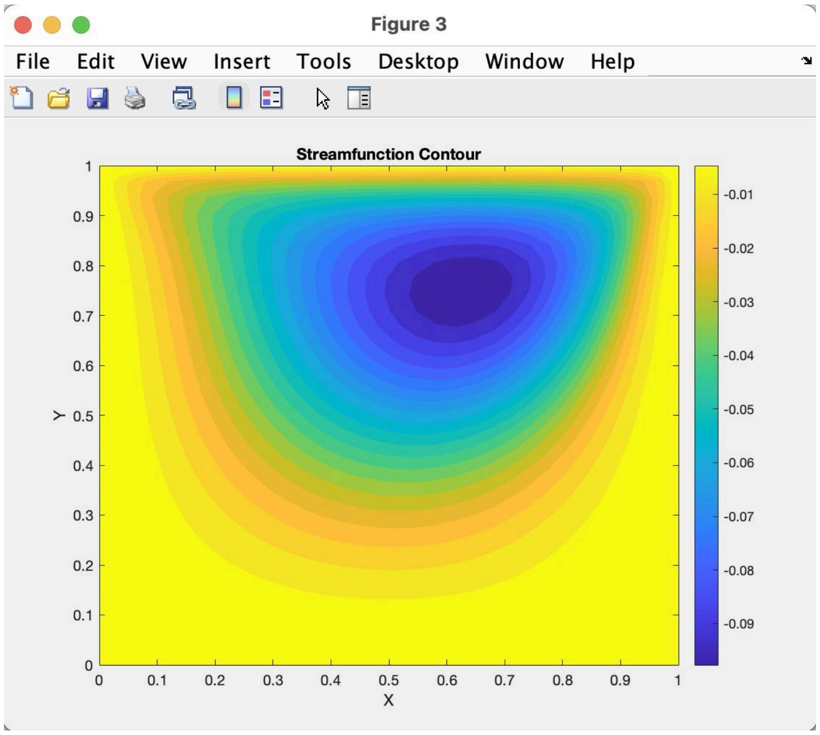

Figure 3: Streamfunction Contour for N=32 & Re = 100

3.1.2 Streamlines and Number of Vortices

Streamlines are plotted using the computed velocity field. At Re=100, there is one central vortex. Usually, two small vortices may appear in the bottom corners, but they are often very weak, especially at this Re and grid size. By examining the streamline patterns in figure 4, it can be seen that there is:

- **1 main vortex** occupying the majority of the cavity.
- **Possible very small corner vortices** near the bottom corners (Not shown at this grid resolution).

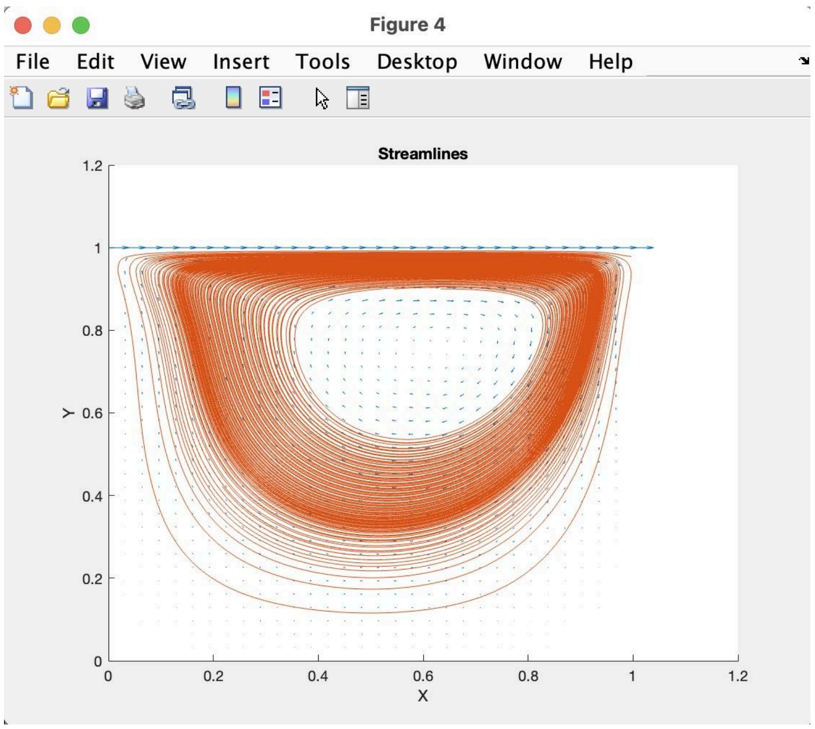

Figure 4: Streamline plot for N=32 & Re = 100

3.1.3 Comparison with Benchmark Data

In this section, there is a comparison of the results from the simulation at Re=100 against the reference paper "High-Re Solutions for Incompressible Flow using the Navier-Stokes Equations and a Multigrid Method." This is conducted by extracting the velocity profiles along the vertical and horizontal centerlines of the structure. Then, a plot of the numerical values against the benchmark values is plotted.

U-Velocity Along Vertical Centerline

The u-velocity component is extracted along the vertical centerline at x=0.5, where velocity values vary with y. Figure 5 below presents the comparison between the computed u-velocity and the benchmark data.

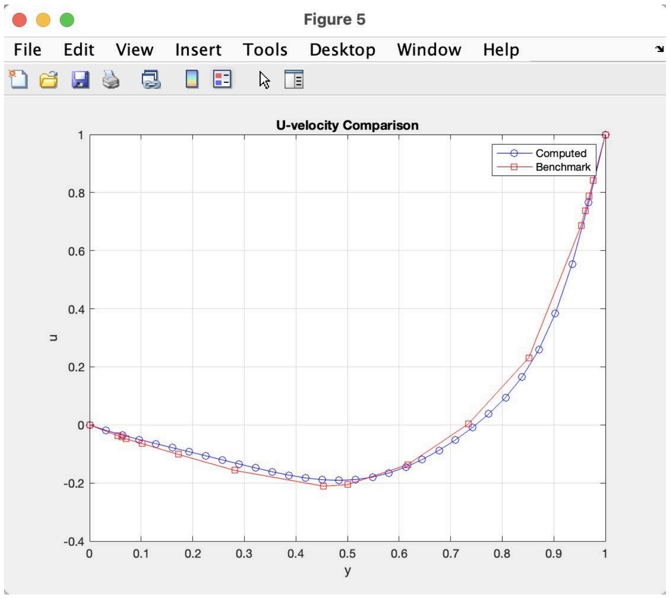

Figure 5: U-velocity Centerline Comparison with Benchmark Values for N=32 & Re = 100

Observations:

- The computed u-velocity shows a smoother, though less accurate profile compared to the benchmark solution
- The general trend of the velocity distribution is spot on, though there is small differences in the exact values

The differences in numerical values can be attributed to numerical diffusion, grid resolution, the discretization scheme, or insufficient iterations for convergence.

V-Velocity Along Horizontal Centerline

The v-velocity component is extracted along the horizontal centerline at y=0.5, where velocity values vary with x. Figure 6 below presents the comparison between the computed v-velocity and the benchmark data.

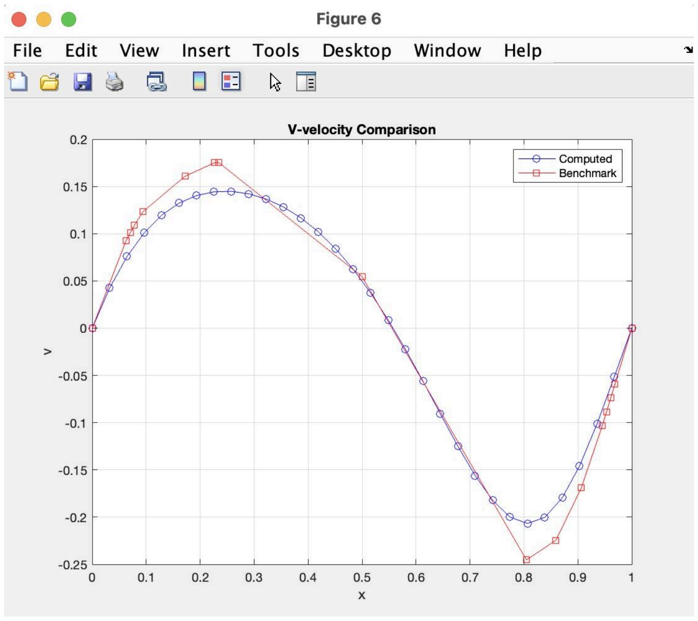

Figure 6: V-velocity Centerline Comparison with Benchmark Values for N=32 & Re = 100

Observations:

- The computed V-velocity shows a smoother profile compared to the benchmark solution
- The benchmark data suggests a more pronounced peak and sharper transitions, whereas the computed values exhibit a smoother curve
- The general shape and trend matches, though the exact values can slightly differ at times, especially at the peaks

The differences in numerical values could be attributed to numerical diffusion, grid resolution, the discretization scheme, or insufficient iterations for convergence.

3.2 Grid Convergence Study

Below, please find figures 7-10 that show the centerline comparison against the benchmark values for v and u at N = 64 and N = 128.

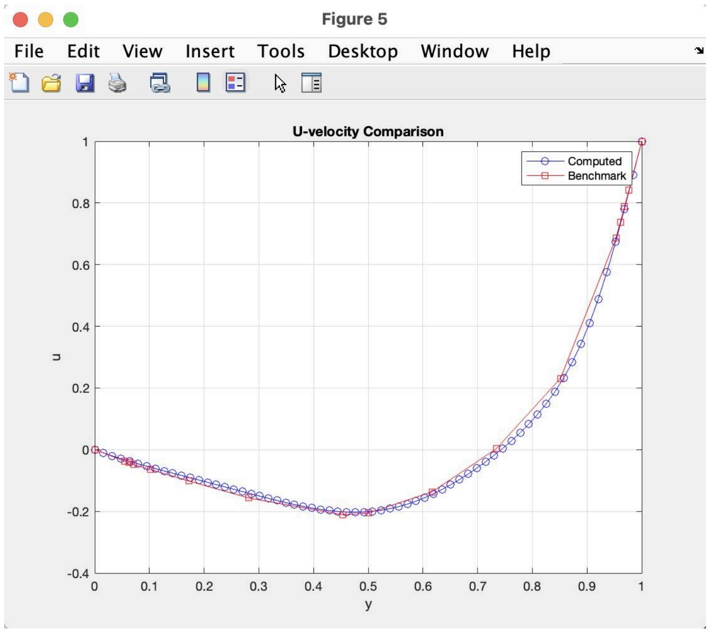

Figure 7: U-velocity Centerline Comparison with Benchmark Values for N=64 & Re = 100

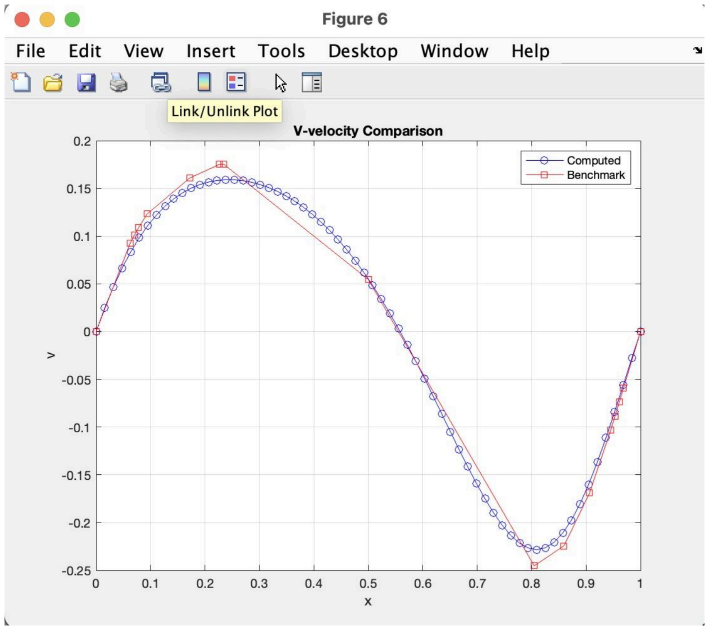

Figure 8: V-velocity Centerline Comparison with Benchmark Values for N=64 & Re = 100

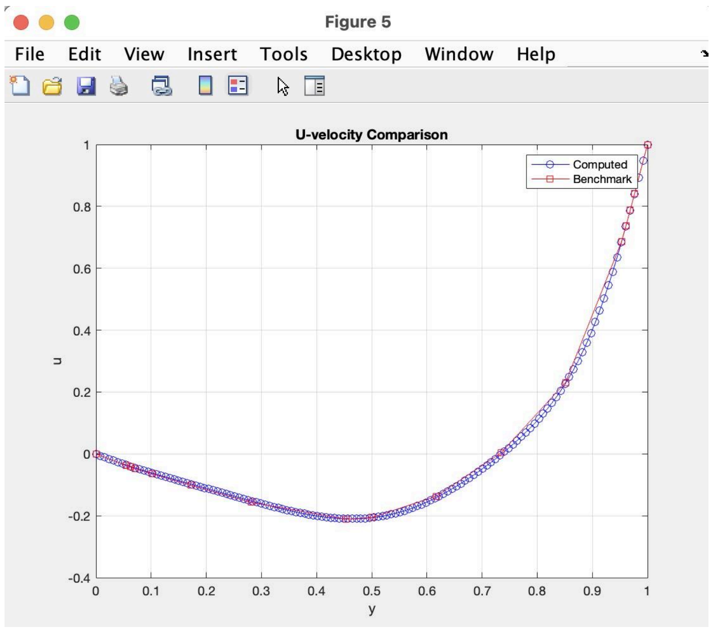

Figure 9: U-velocity Centerline Comparison with Benchmark Values for N=128 & Re = 100

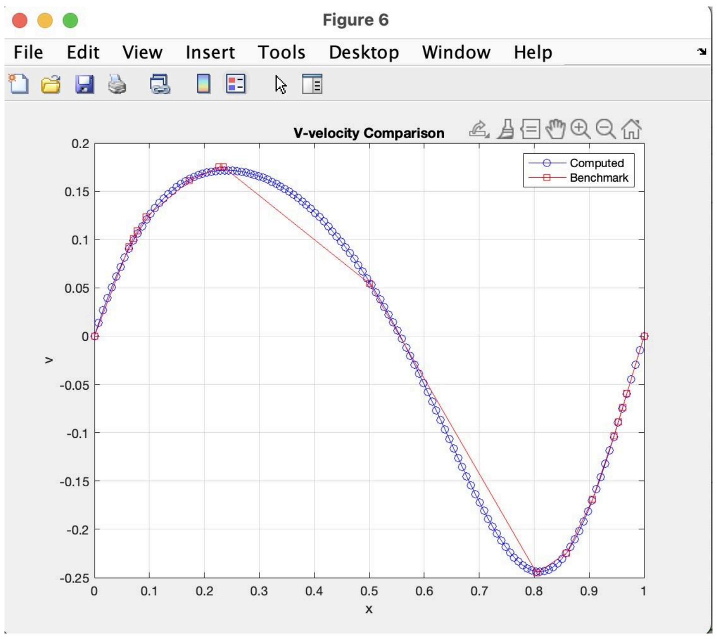

Figure 10: V-velocity Centerline Comparison with Benchmark Values for N=128 & Re = 100

The smallest total error (sum of u and v errors) occurs at N = 128, where nearly each benchmark data point is nearly on the graph, making it the most suitable grid for this Reynolds number. The less dense grids kept the general trend but have increased error. This shows the relationship between using finer grids to achieve increasing accuracy. However, this also increases the computational cost, as the N=128 simulation took much longer than the N=64 and N=32 simulation to fully complete.

3.3: Flow inside a square cavity at Reynolds number = 1000

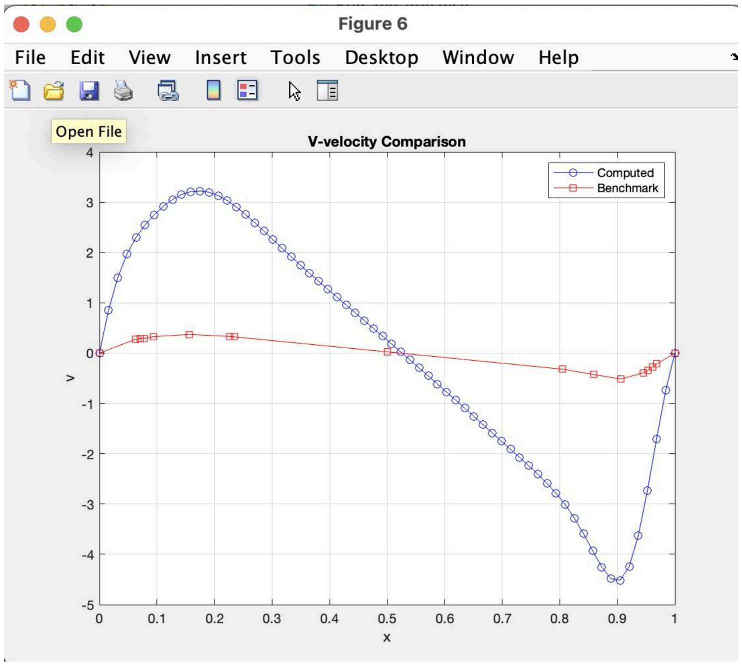

Figure 11: V-velocity Centerline Comparison with Benchmark Values for N=64 & Re = 1000

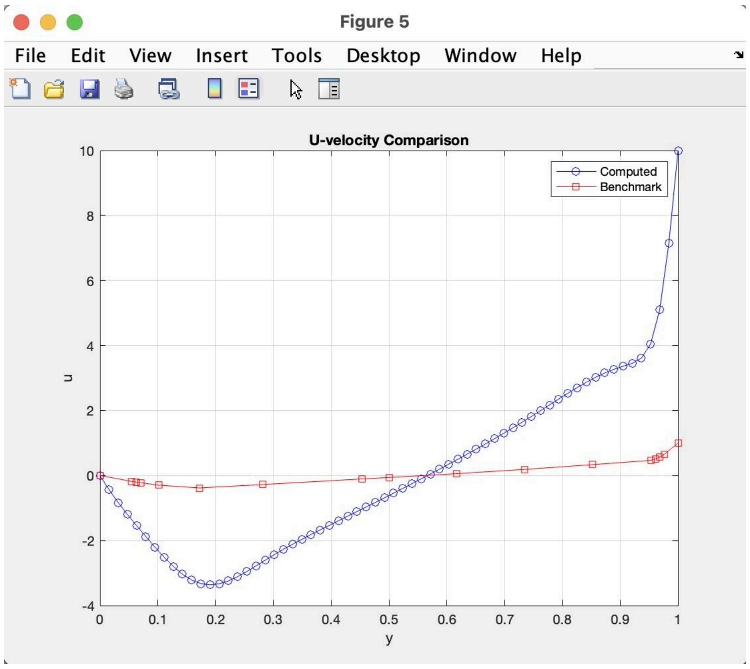

Figure 12: U-velocity Centerline Comparison with Benchmark Values for N=64 & Re = 1000

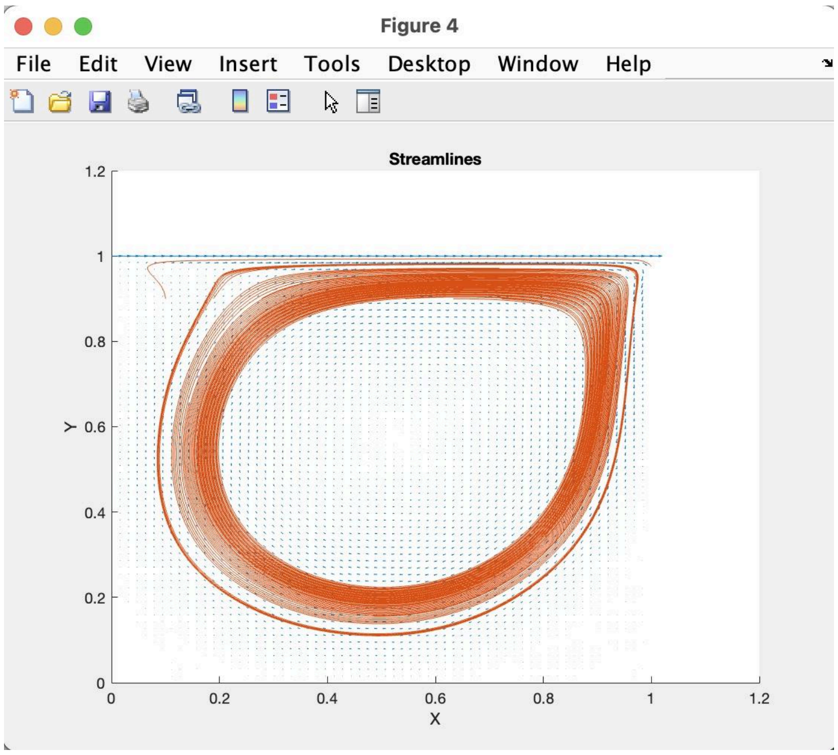

Figure 13: Streamline plot for N=64 & Re = 1000

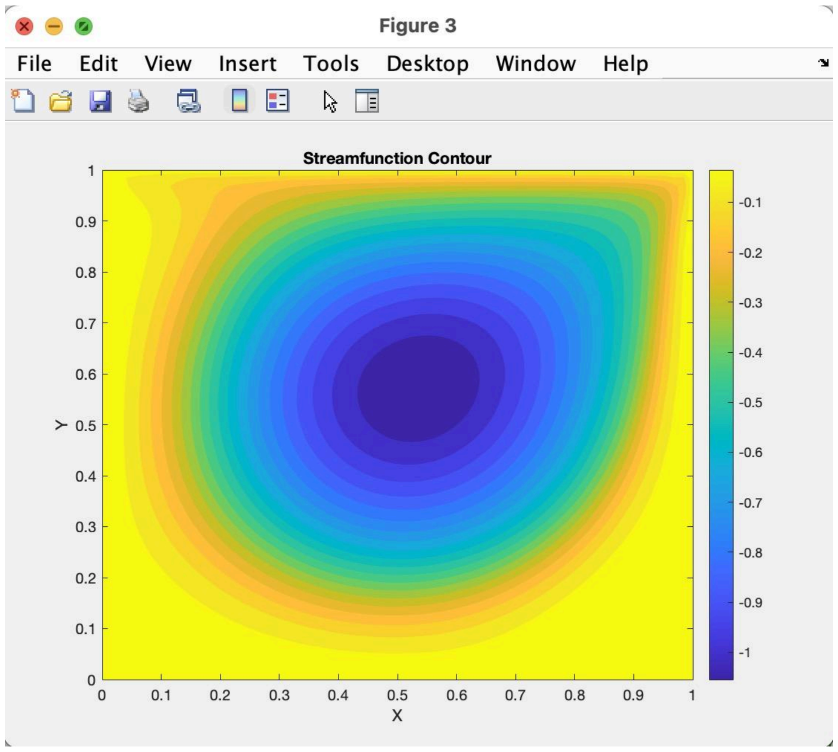

Figure 14: Streamfunction contour for N=64 & Re = 1000

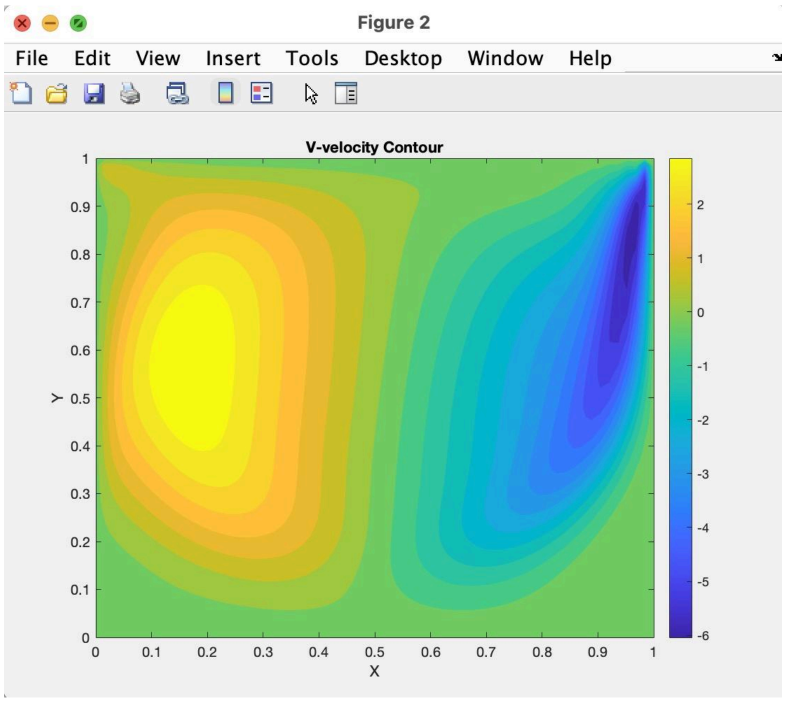

Figure 15: V-velocity Contour for N=64 & Re = 1000

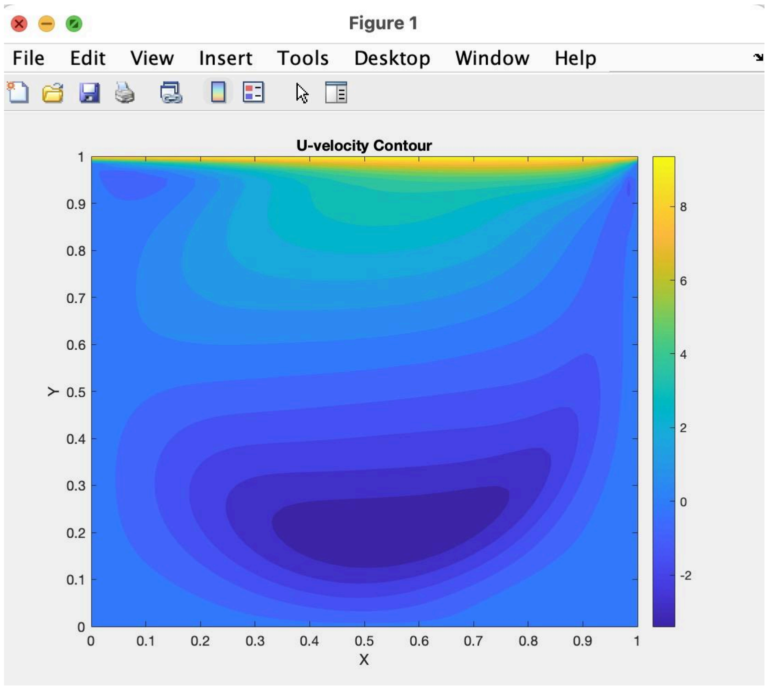

Figure 16: U-velocity Contour for N=64 & Re = 1000

These simulated v and u centerline charts in figures 11 and 12 have similar behaviors as the benchmark, but the actual values can sometimes be over an order of magnitude larger. With this, the benchmark plots look almost flat at times. This behavior can occur from a combination of factors, including numerical diffusion, grid resolution limitations, and truncation errors from the discretization scheme. At higher Reynolds numbers like this, often it is necessary to use a finer grid resolution in order to properly capture these values. Additionally, the explicit FTCS method, while simple to implement, can introduce instability that can cause numerical errors, particularly in regions of high vorticity like this example. This Reynolds number has 1 vortex as seen in figure 13.

4.0 Summary

In this project, a lid-driven cavity flow problem was simulated using a numerical scheme employing a forward-time central-space (FTCS) method for the vorticity transport equation and a Gauss-Seidel iterative solver for the streamfunction equation. The first case, at an Re =100, showed one vortex with v and u velocity profiles closely matching those from the benchmark study. Additionally, conducting a convergence study at Re = 100 showed that increasing grid resolution improved the accuracy of the scheme at the cost of computational power and time. At Re = 1000, the flow became more complex. The u, v, and streamfunction contours exhibited the same behaviour, though at a more exaggerated scale. Furthermore, some discrepancies between the numerical results and the benchmark values occurred, which were attributed to numerical diffusion, grid resolution effects, and limitations of the explicit time-stepping scheme. Overall, this study highlights the importance of numerical methods in CFD and emphasizes the trade-off between computational efficiency and solution accuracy.

5.0 Attached Code

```
clc; clear; close all;
%% Parameters
N = 32;                    % Number of mesh points in each direction; Used 32,
64, and 128
L = 1;                     % Domain length in meters
Uwall = 1;                % Wall velocity in m/s
rho = 1;                   % Density in kg/m^3
nu = 0.01;                 % Kinematic viscosity in m^2/s
dt = 0.001;                % Time step
tol = 1e-7;                % Error tolerance
max_iter = 5000;           % Maximum iterations
h = L / (N-1);             % Grid spacing
Re = Uwall * L / nu;       % Reynolds number
%% Initialize variables
omega = zeros(N, N);       % Vorticity matrix
psi = zeros(N, N);         % Stream function matrix
u = zeros(N, N);           % x-velocity component
v = zeros(N, N);           % y-velocity component
[X, Y] = meshgrid(linspace(0, L, N), linspace(0, L, N)); % Grid coordinates
%% Solver Loop
```

```
for iter = 1:max_iter
   omega_old = omega;

   % Boundary Conditions
   omega(:, N) = (-2 * psi(:, N-1) / h^2) - (2 * Uwall / h); % Top boundary
(moving lid)
   omega(:, 1) = -2 * psi(:, 2) / h^2; % Bottom boundary
   omega(1, :) = -2 * psi(2, :) / h^2; % Left boundary
   omega(N, :) = -2 * psi(N-1, :) / h^2; % Right boundary

   % Update vorticity using FTCS explicit method
   for i = 2:N-1
       for j = 2:N-1
           dPsidy = (psi(i, j+1) - psi(i, j-1)) / (2*h);
           dPsidx = (psi(i+1, j) - psi(i-1, j)) / (2*h);
           dOmegadx = (omega(i+1, j) - omega(i-1, j)) / (2*h);
           dOmegady = (omega(i, j+1) - omega(i, j-1)) / (2*h);
           d2Omegadx2 = (omega(i+1, j) - 2*omega(i, j) + omega(i-1, j)) / h^2;
           d2Omegady2 = (omega(i, j+1) - 2*omega(i, j) + omega(i, j-1)) / h^2;

           omega(i, j) = omega_old(i, j) + dt * (-dPsidy * dOmegadx + dPsidx *
dOmegady + nu * (d2Omegadx2 + d2Omegady2));
       end
   end

   % Solve for stream function using Gauss-Seidel iteration
   for iter_psi = 1:100
       for i = 2:N-1
           for j = 2:N-1
               psi(i, j) = 0.25 * (psi(i+1, j) + psi(i-1, j) + psi(i, j+1) +
psi(i, j-1) + h^2 * omega(i, j));
           end
       end
   end

   % Compute error
   error = sqrt(sum(sum((omega - omega_old).^2)) / sum(sum(omega_old.^2)));

   % Convergence check
   if error < tol
       fprintf('Converged at iteration %d with error %.8f\n', iter, error);
       break;
   end
end
%% Compute velocity field from stream function
for i = 2:N-1
   for j = 2:N-1
       u(i, j) = (psi(i, j+1) - psi(i, j-1)) / (2*h);
       v(i, j) = -(psi(i+1, j) - psi(i-1, j)) / (2*h);
   end
end
% Explicitly enforce the moving lid boundary condition
u(:, N) = Uwall;
v(:, N) = 0;
%% Extract midline velocity profiles
mid_x = floor(N/2);
mid_y = floor(N/2);
y_midline = linspace(0, L, N);
x_midline = linspace(0, L, N);
u_midline = u(mid_x, :);
v_midline = v(:, mid_y);
%% Benchmark Data (Re = 100)
bm_y = [1.00000, 0.9766, 0.9688, 0.9609, 0.9531, 0.8516, 0.7344, 0.6172,
0.5000, 0.4531, 0.2813, 0.1719, 0.1016, 0.0703, 0.0625, 0.0547, 0.0000];
bm_u = [1.00000, 0.84123, 0.78871, 0.73722, 0.68717, 0.23151, 0.00332,
-0.13641, -0.20581, -0.21090, -0.15662, -0.10150, -0.06434, -0.04775, -0.04192,
-0.03717, 0.00000];
bm_x = [1.0000, 0.9688, 0.9609, 0.9531, 0.9453, 0.9063, 0.8594, 0.8047, 0.5000,
0.2344, 0.2266, 0.1719, 0.0938, 0.0781, 0.0703, 0.0625, 0.0000];
bm_v = [0.00000, -0.05906, -0.07391, -0.08864, -0.10313, -0.16914, -0.22445,
-0.24533, 0.05454, 0.17527, 0.17507, 0.16077, 0.12317, 0.10890, 0.10091,
0.09233, 0.00000];
%% Benchmark Data (Re = 1000)
% bm_y = [1.00000, 0.9766, 0.9688, 0.9609, 0.9531, 0.8516, 0.7344, 0.6172,
0.5000, 0.4531, 0.2813, 0.1719, 0.1016, 0.0703, 0.0625, 0.0547, 0.0000];
% bm_u = [1.00000, 0.65928, 0.57492, 0.51117, 0.46604, 0.33304, 0.18719,
0.05702, -0.06080, -0.10648, -0.27805, -0.38289, -0.29730, -0.22220, -0.20196,
-0.18109, 0.00000];
%
% bm_x = [1.0000, 0.9688, 0.9609, 0.9531, 0.9453, 0.9063, 0.8594, 0.8047,
0.5000, 0.2344, 0.2266, 0.1563, 0.0938, 0.0781, 0.0703, 0.0625, 0.0000];
% bm_v = [0.00000, -0.21388, -0.27669, -0.33714, -0.39188, -0.51550, -0.42665,
-0.31966, 0.02526, 0.32235, 0.33075, 0.37095, 0.32627, 0.29012, 0.29012,
0.27485, 0.00000];
%% Contour Plots
figure; contourf(X, Y, u', 20, 'LineColor', 'none'); colorbar;
title('U-velocity Contour'); xlabel('X'); ylabel('Y');
figure; contourf(X, Y, v', 20, 'LineColor', 'none'); colorbar;
title('V-velocity Contour'); xlabel('X'); ylabel('Y');
figure; contourf(X, Y, psi', 20, 'LineColor', 'none'); colorbar;
title('Streamfunction Contour'); xlabel('X'); ylabel('Y');
% Streamlines
figure;
hold on;
quiver(X, Y, u', v');
startX = linspace(0.1, 0.9, 10);
startY = ones(size(startX)) * 0.9;
streamline(X, Y, u', v', startX, startY);
```

```
hold off;
title('Streamlines');
xlabel('X');
ylabel('Y');
%% Comparison Plots
figure; plot(y_midline, u_midline, 'b-o', bm_y, bm_u, 'r-s'); title('U-velocity
Comparison'); xlabel('y'); ylabel('u'); legend('Computed', 'Benchmark'); grid
on;
figure; plot(x_midline, v_midline, 'b-o', bm_x, bm_v, 'r-s'); title('V-velocity
Comparison'); xlabel('x'); ylabel('v'); legend('Computed', 'Benchmark'); grid
on;
fprintf('Simulation and comparison complete.\n');
```