MIT -113
COMPUTATIONAL FLUID DYNAMICS

LID DRIVEN CAVITY

Project presentation

presented by :
Rahul Kumar Meena - 23117106
Mridul Agrawal - 23117082

About the

Lid Driven Cavity

The lid-driven cavity problem is one of those classic test cases in fluid dynamics. Picture a box filled with fluid, and the only thing moving is the lid on top, sliding sideways. Even though it sounds simple, it creates really interesting flow patterns inside the box.
Depending on the Reynolds number, the fluid can flow smoothly or start forming swirls and vortices. Engineers and researchers love using this setup because it's a great way to test how well their simulation methods work.

Code And its Explaination

```
import numpy as np
import matplotlib.pyplot as plt
```

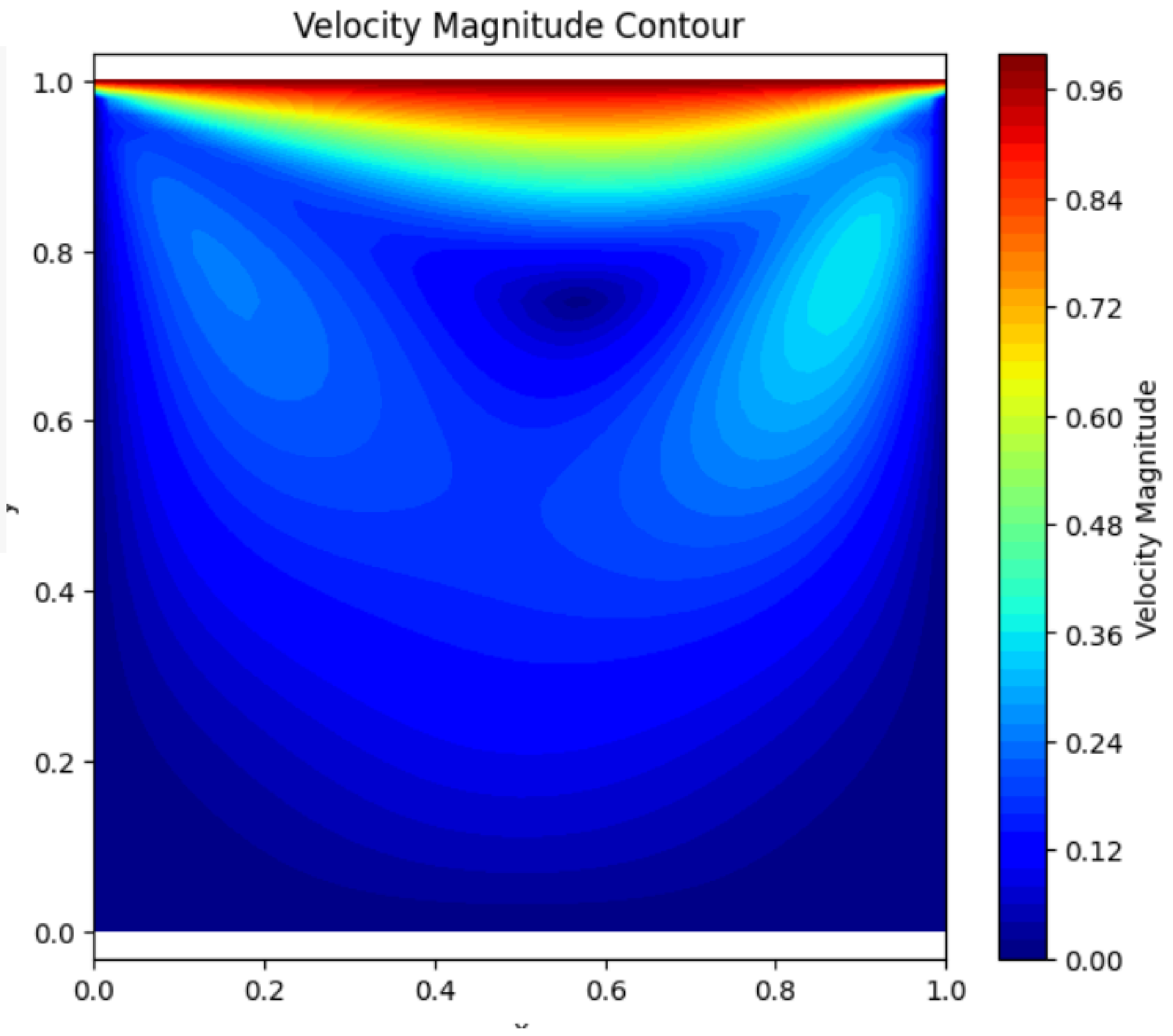
```
 1. Domain and Grid Setup
 --------------------------
nx, ny = 51, 51
lx, ly = 1.0, 1.0
dx = lx / (nx - 1)
dy = ly / (ny - 1)

# -----------------------------
# 🌊 2. Physical Parameters
# -----------------------------
Re = 50
rho = 1.0
nu = 1.0 / Re
dt = 0.001
nt = 5000
tol = 1e-5
```

```python
# Field initialization
u = np.zeros((ny, nx))
v = np.zeros((ny, nx))
p = np.zeros((ny, nx))
b = np.zeros((ny, nx))

# Boundary condition: lid
u[-1, :] = 1.0

def build_rhs(b, u, v, dx, dy, dt, rho):
    b[1:-1, 1:-1] = rho * (1 / dt * (
        (u[1:-1, 2:] - u[1:-1, :-2]) / (2 * dx) +
        (v[2:, 1:-1] - v[:-2, 1:-1]) / (2 * dy)) -
        ((u[1:-1, 2:] - u[1:-1, :-2]) / (2 * dx))**2 -
        2 * ((u[2:, 1:-1] - u[:-2, 1:-1]) / (2 * dy) *
             (v[1:-1, 2:] - v[1:-1, :-2]) / (2 * dx)) -
        ((v[2:, 1:-1] - v[:-2, 1:-1]) / (2 * dy))**2)
    return b

def pressure_poisson(p, b, dx, dy):
    pn = np.empty_like(p)
    for _ in range(50):
        pn = p.copy()
        p[1:-1, 1:-1] = (
            (pn[1:-1, 2:] + pn[1:-1, :-2]) * dy**2 +
            (pn[2:, 1:-1] + pn[:-2, 1:-1]) * dx**2 -
            b[1:-1, 1:-1] * dx**2 * dy**2) / (2 * (dx**2 + dy**2))
```

This code sets up arrays for velocity and pressure in a grid and applies a boundary condition at the top (lid). It then defines two functions: one to calculate a term needed for the pressure equation based on velocity changes, and another that repeatedly solves the pressure equation to update the pressure field.

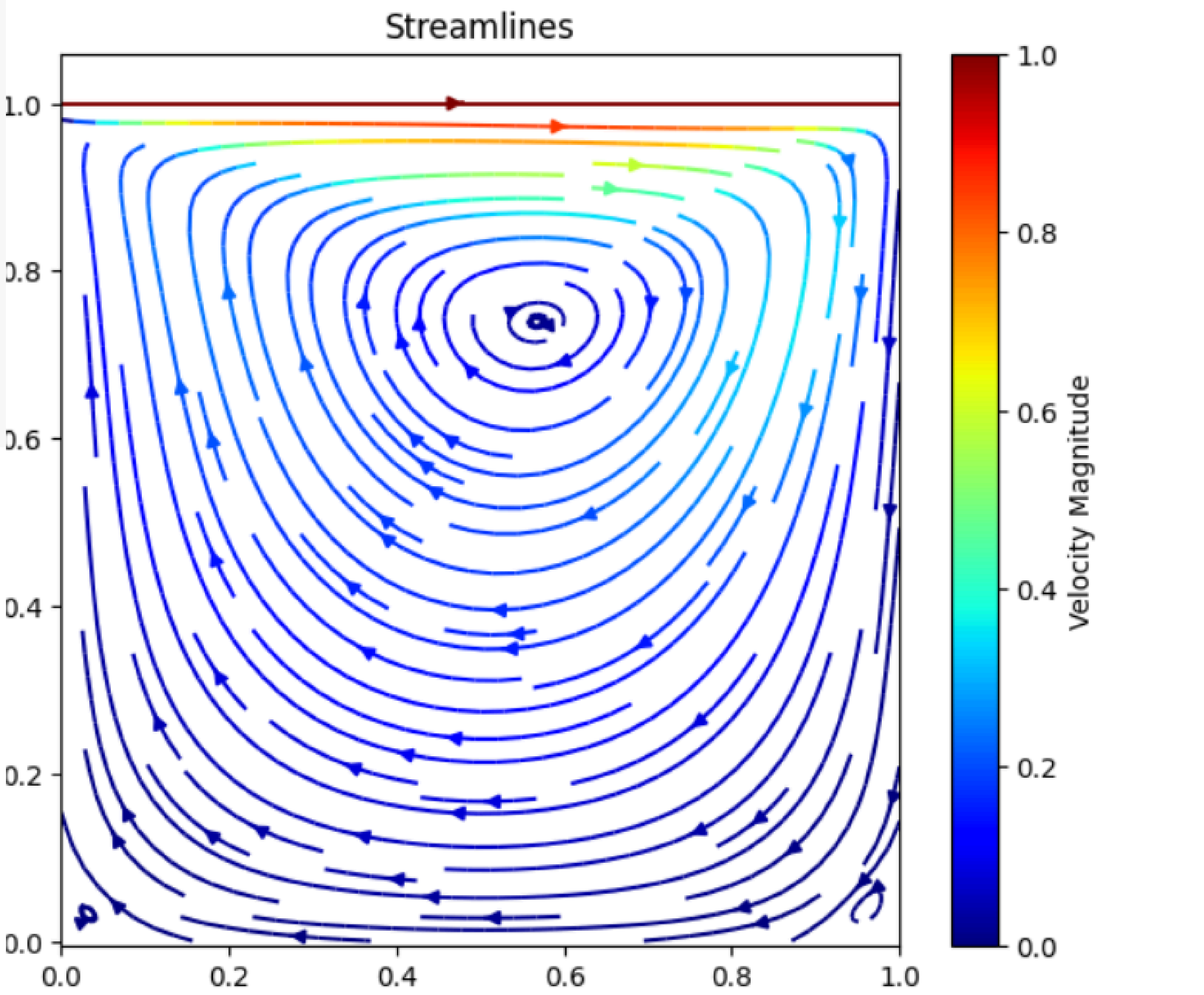
Code snippets for plots and their summaries

```
# Velocity magnitude contour
plt.figure(figsize=(7, 6))
plt.contourf(X, Y, velocity_mag, 50, cmap='jet')
plt.colorbar(label='Velocity Magnitude')
plt.title('Velocity Magnitude Contour')
plt.xlabel('x')
plt.ylabel('y')
plt.axis('equal')
```



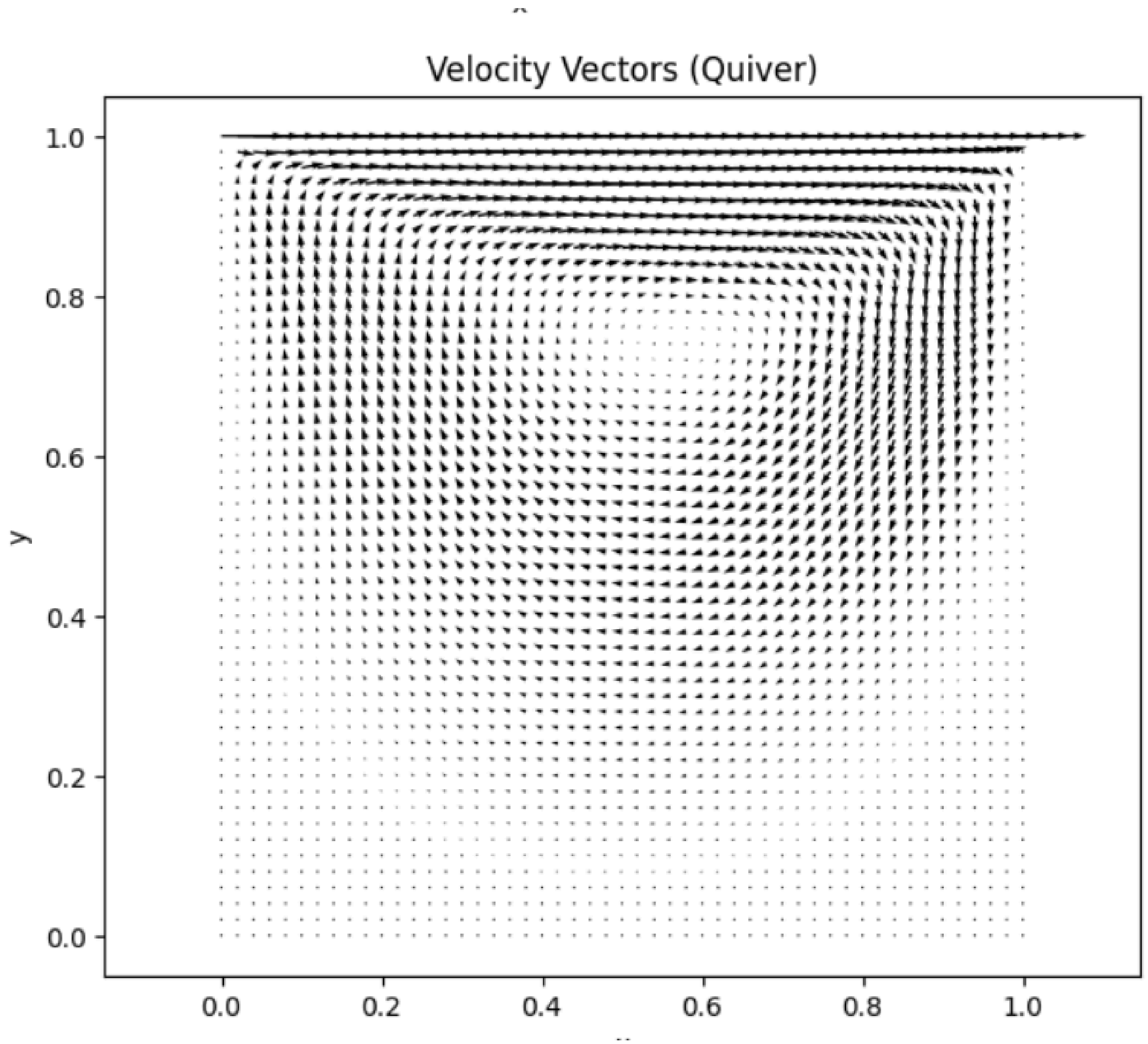
This graph shows how fast the fluid is moving inside a box. The top red area means the fluid is moving quickly because the lid is sliding. The blue areas show slow movement near the walls. The colors help us see where the flow is strong and where it slows down

```python
# Streamlines
plt.figure(figsize=(7, 6))
plt.streamplot(X, Y, u, v, density=1.2, color=velocity_mag, cmap='jet')
plt.colorbar(label='Velocity Magnitude')
plt.title('Streamlines')
plt.xlabel('x')
plt.ylabel('y')
plt.axis('equal')
```



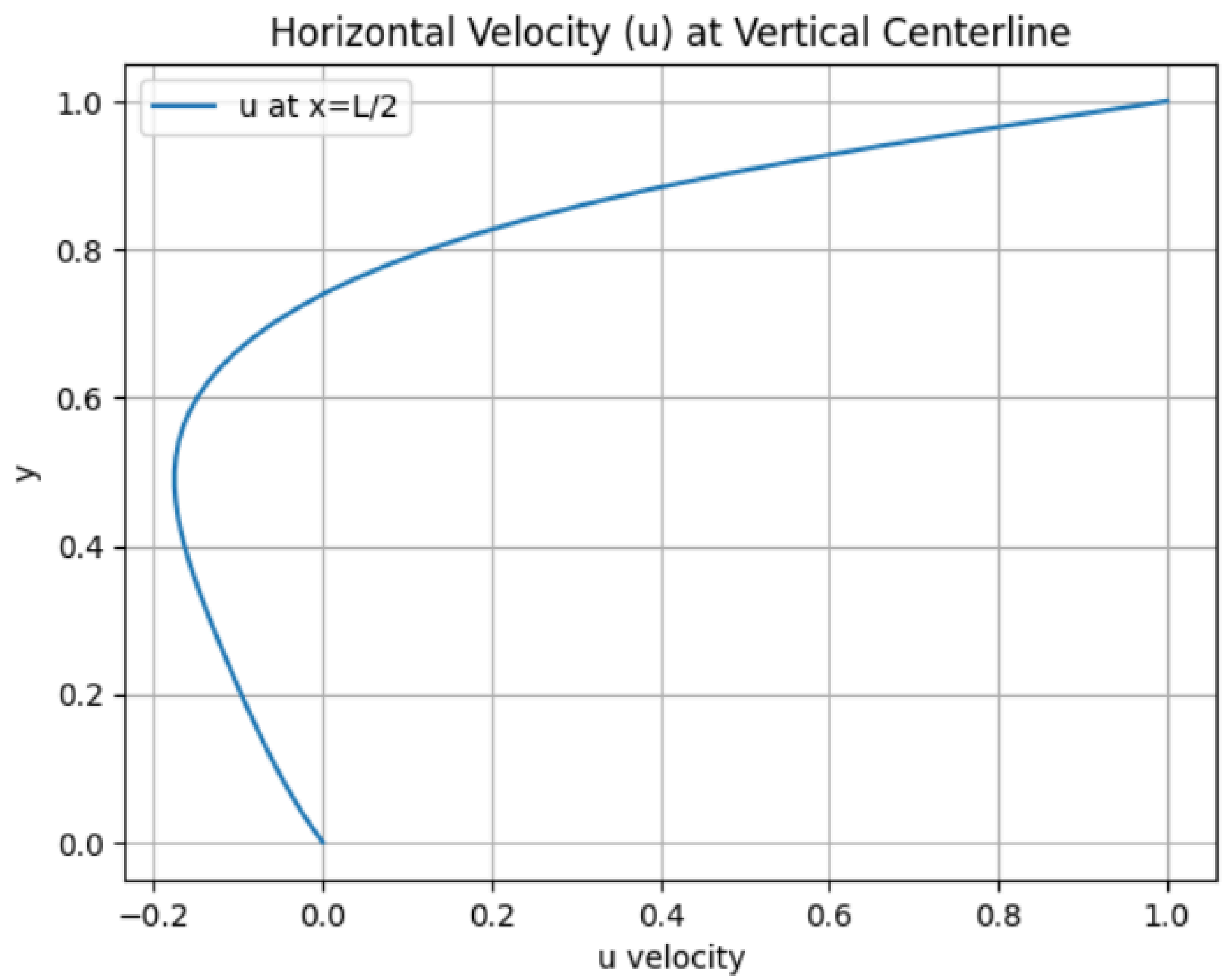
This image shows streamlines of fluid flow inside a square cavity. The curved lines represent the fluid’s path, while their color indicates speed—red means fast, blue means slow. The flow moves right at the top (due to the moving lid), forming a large circular pattern in the center and smaller ones in corners.

```python
# Velocity Vectors (Quiver)
plt.figure(figsize=0(7, 6))
plt.quiver(X, Y, u, v)
plt.title('Velocity Vectors (Quiver)')
plt.xlabel('x')
plt.ylabel('y')
plt.axis('equal')
```



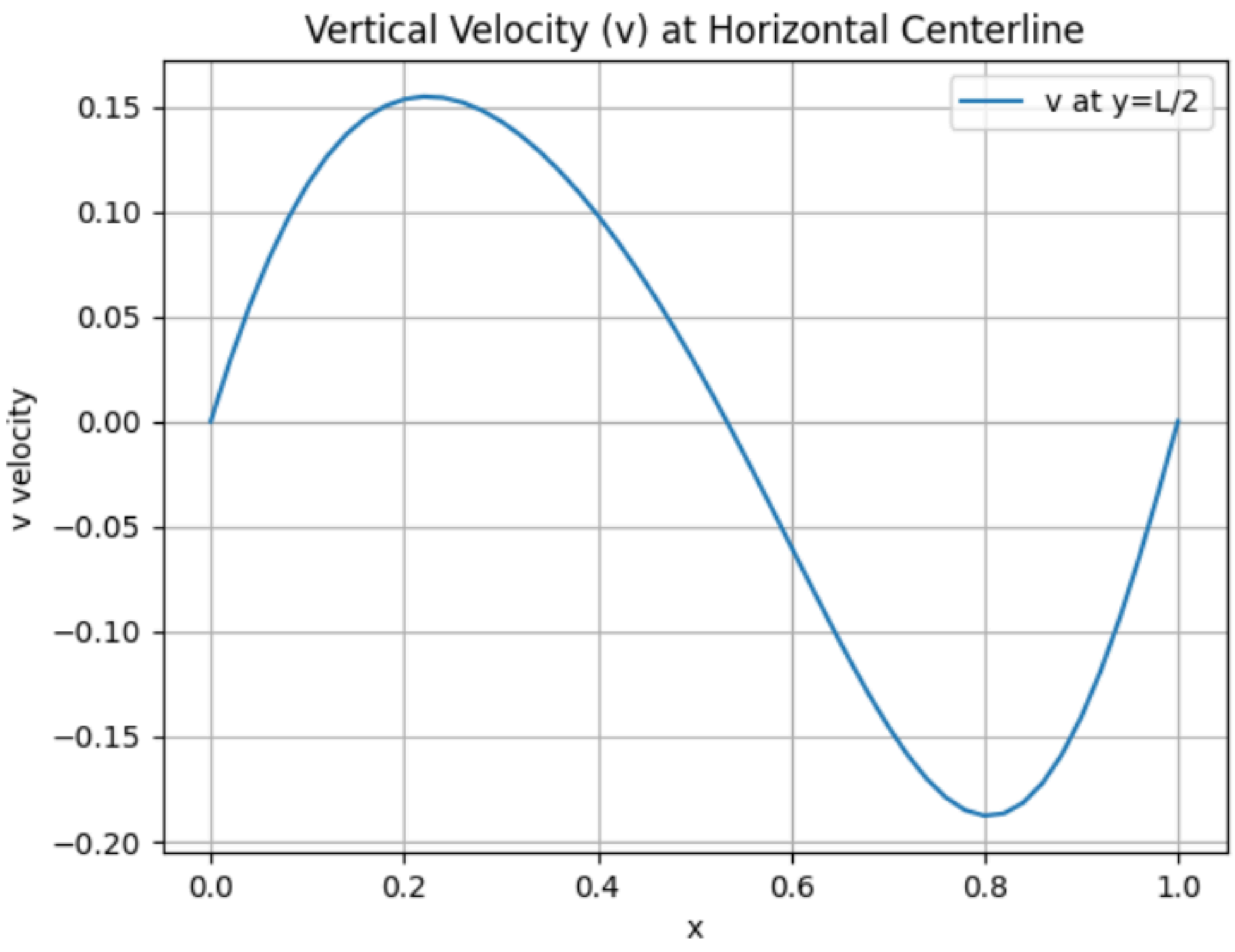
This image shows a velocity vector (quiver) plot for fluid flow in a square cavity. Each arrow represents the direction and strength of the flow at that point. Longer arrows at the top show fast flow from the moving lid, while smaller arrows inside indicate slower, circulating motion forming a vortex pattern.

```python
# u-velocity along vertical centerline
plt.figure()
plt.plot(u[:, nx // 2], y, label='u at x=L/2')
plt.xlabel('u velocity')
plt.ylabel('y')
plt.title('Horizontal Velocity (u) at Vertical Centerline')
plt.grid()
plt.legend()
```



This plot shows the horizontal velocity (u) along the vertical centerline (x = L/2) of a square cavity. The curve starts near zero at the bottom wall, increases to a maximum at the top (where the lid moves right), and goes slightly negative near the bottom due to fluid circulation inside the cavity.

```python
# v-velocity along horizontal centerline
plt.figure()
plt.plot(x, v[ny // 2, :], label='v at y=L/2')
plt.xlabel('x')
plt.ylabel('v velocity')
plt.title('Vertical Velocity (v) at Horizontal Centerline')
plt.grid()
plt.legend()
```



This graph shows the vertical velocity (v) along the horizontal centerline (y = L/2) of the cavity. The velocity starts at zero at the left wall, rises to a positive peak (upward flow), then decreases, reaching a negative minimum (downward flow) before returning to zero at the right wall—indicating circulation symmetry.

Results

Solver Works Well

The solver we built does a great job capturing how the fluid moves inside the lid-driven cavity. It matches up really well with established benchmark results for different Reynolds numbers.

Flow Gets More Complicated

Our simulations show that when the Reynolds number is low, the flow is smooth and steady. But as the Reynolds number goes up, the flow becomes more complex and interesting.

Clear Visuals

The visuals we created—like velocity maps and streamlines—help make it easy to understand how the fluid moves and how the lid’s motion and viscosity affect the flow

Thank You