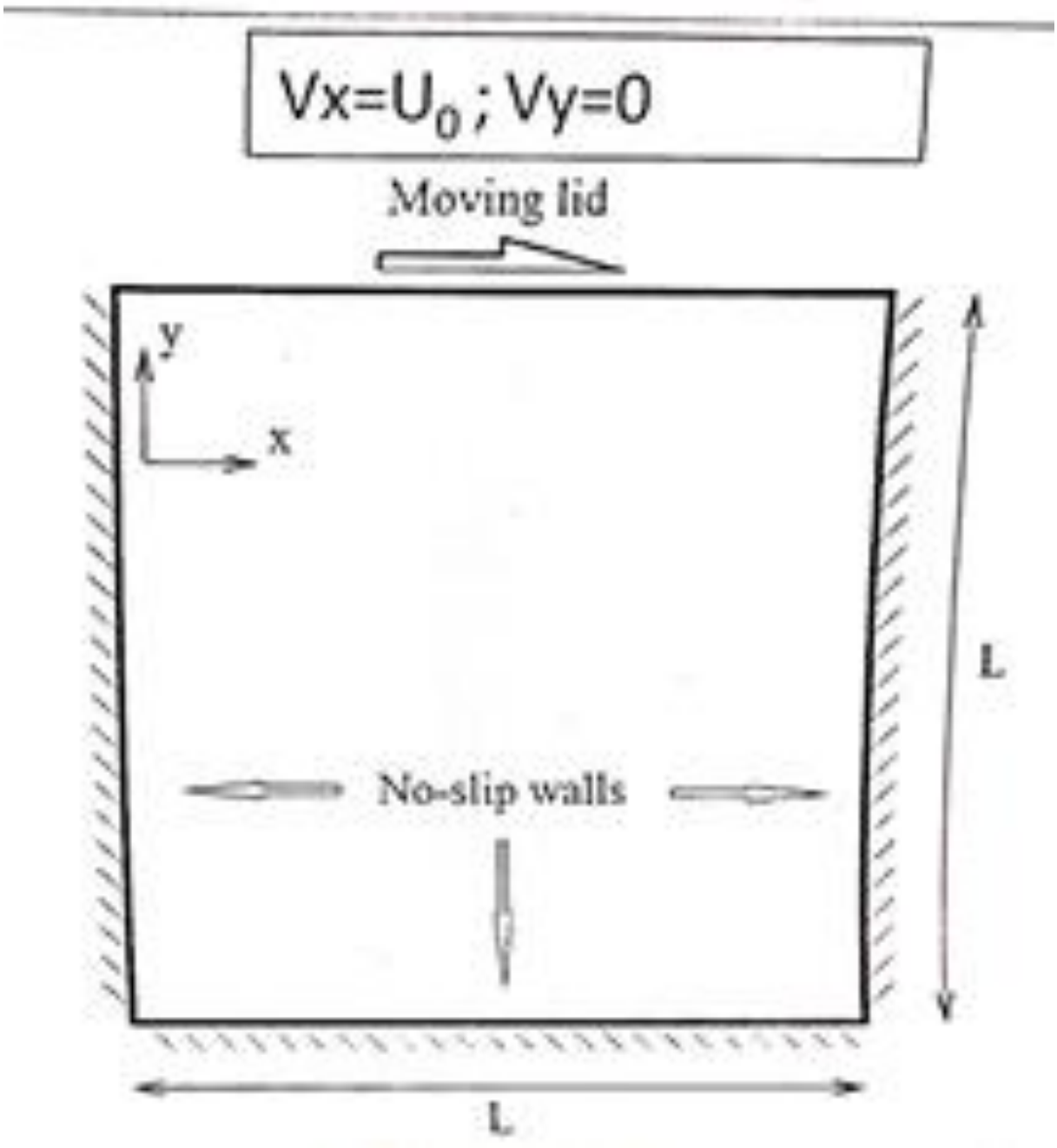
PROBLEM STATEMENT

Develop a numerical code to develop flow field in two-dimensional domain shown below using vorticity-stream function formulation.

FLOW INSIDE A LID DRIVEN CAVITY



- Assumptions:

We consider the following assumptions:

1. The flow of lid is in x direction only.
2. The plate moves with a constant velocity U in the x-direction, starting at t=0.
3. The fluid is viscous and incompressible.

- Model Equation and Initial and Boundary condition:

The vorticity transport equation in two dimensions is generally expressed as:

$$\frac{\partial \omega}{\partial t} + V_x \frac{\partial \omega}{\partial x} + V_y \frac{\partial \omega}{\partial y} = \nu \left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2} \right) \tag{1}$$

- Boundary Conditions:

- **Left Wall:**

$$V_x = 0$$
$$V_y = 0$$
$$\psi = 0$$
$$\omega_{\text{left}} = -\frac{2V_{y,i,j}}{\Delta x}$$

- **Right Wall:**

$$V_x = 0$$
$$V_y = 0$$
$$\psi = 0$$
$$\omega_{\text{right}} = -\frac{2V_{y,m,i}}{\Delta x}$$

- **Top Wall:**

$$
\begin{aligned}
V_x &= U_0 \\
V_y &= 0 \\
\psi &= 0 \\
\omega_{\text{top}} &= -2\left(\frac{U_0 - V_{x,i,N}}{\Delta y}\right)
\end{aligned}
$$

- **Bottom Wall:**

$$
\begin{aligned}
V_x &= 0 \\
V_y &= 0 \\
\psi &= 0 \\
\omega_{\text{bottom}} &= -\frac{2V_{x,i,j}}{\Delta y}
\end{aligned}
$$

- Discretization of vorticity transport equation using explicit time marching:

We start with the integral form of the vorticity transport equation:

$$\int_V \frac{\partial \omega}{\partial t}\, dV + \int_S \vec{u} \cdot \nabla \omega \, dA = \nu \int_S \nabla^2 \omega \, dA \tag{1}$$

where: - ω is the vorticity, - $\vec{u} = (u, v)$ is the velocity vector, - ν is the kinematic viscosity, - V represents the control volume, - S is the surface area of the control volume.

Using explicit time marching, the time derivative term is approximated as:

$$\int_V \frac{\partial \omega}{\partial t}\, dV \approx \Delta x \Delta y \left(\frac{\omega_{i,j}^{n+1} - \omega_{i,j}^{n}}{\Delta t} \right) \tag{2}$$

where: - $\omega_{i,j}^{n+1}$ and $\omega_{i,j}^{n}$ are the vorticity values at the current and previous time steps, respectively, - Δt is the time step, - Δx and Δy are the grid spacings in the x and y directions.

For the advection term $\int_S \vec{u} \cdot \nabla \omega \, dA$, we apply central differencing in space:

$$\int_S \vec{u} \cdot \nabla \omega \, dA \approx -u_{i,j} \frac{\omega_{i+1,j} - \omega_{i-1,j}}{2\Delta x} - v_{i,j} \frac{\omega_{i,j+1} - \omega_{i,j-1}}{2\Delta y} \tag{3}$$

where $u_{i,j}$ and $v_{i,j}$ are the velocity components at point (i, j).

The diffusion term $\nu \int_S \nabla^2 \omega \, dA$ is discretized using central differencing for the Laplacian:

$$\nu \int_S \nabla^2 \omega \, dA \approx \nu \left(\frac{\omega_{i+1,j} - 2\omega_{i,j} + \omega_{i-1,j}}{\Delta x^2} + \frac{\omega_{i,j+1} - 2\omega_{i,j} + \omega_{i,j-1}}{\Delta y^2} \right) \tag{4}$$

Now, substituting the discrete forms of each term into the integral equation, we obtain:

$$\frac{\omega_{i,j}^{n+1}-\omega_{i,j}^{n}}{\Delta t}=-u_{i,j}^{n}\frac{\omega_{i+1,j}^{n}-\omega_{i-1,j}^{n}}{2\Delta x}-v_{i,j}^{n}\frac{\omega_{i,j+1}^{n}-\omega_{i,j-1}^{n}}{2\Delta y}+\nu\left(\frac{\omega_{i+1,j}^{n}-2\omega_{i,j}^{n}+\omega_{i-1,j}^{n}}{\Delta x^{2}}+\frac{\omega_{i,j+1}^{n}-2\omega_{i,j}^{n}+\omega_{i,j-1}^{n}}{\Delta y^{2}}\right) \tag{5}$$

$$\omega_{i,j}^{n+1}=\omega_{i,j}^{n}+\Delta t\left(-u_{i,j}^{n}\frac{\omega_{i+1,j}^{n}-\omega_{i-1,j}^{n}}{2\Delta x}-v_{i,j}^{n}\frac{\omega_{i,j+1}^{n}-\omega_{i,j-1}^{n}}{2\Delta y}+\nu\left(\frac{\omega_{i+1,j}^{n}-2\omega_{i,j}^{n}+\omega_{i-1,j}^{n}}{\Delta x^{2}}+\frac{\omega_{i,j+1}^{n}-2\omega_{i,j}^{n}+\omega_{i,j-1}^{n}}{\Delta y^{2}}\right)\right) \tag{6}$$

Expressing this equation in terms of coefficients a_p, a_w, a_e, a_n, and a_s:

$$a_p\omega_{i,j}^{n+1}=a_w\omega_{i-1,j}^{n}+a_e\omega_{i+1,j}^{n}+a_n\omega_{i,j+1}^{n}+a_s\omega_{i,j-1}^{n}+a_p\omega_{i,j}^{n}$$

where the coefficients are:

$$a_p=1+\Delta t\left[\nu\left(\frac{2}{\Delta x^{2}}+\frac{2}{\Delta y^{2}}\right)+u_{i,j}^{n}\frac{1}{\Delta x}+v_{i,j}^{n}\frac{1}{\Delta y}\right]$$

$$a_w=\Delta t\left[\frac{\nu}{\Delta x^{2}}-\frac{u_{i,j}^{n}}{2\Delta x}\right]$$

$$a_e=\Delta t\left[\frac{\nu}{\Delta x^{2}}+\frac{u_{i,j}^{n}}{2\Delta x}\right]$$

$$a_n=\Delta t\left[\frac{\nu}{\Delta y^{2}}-\frac{v_{i,j}^{n}}{2\Delta y}\right]$$

$$a_s=\Delta t\left[\frac{\nu}{\Delta y^{2}}+\frac{v_{i,j}^{n}}{2\Delta y}\right]$$

- Discretization of stream function equation using implicit time marching:

The stream function equation for vorticity, ω, in terms of the stream function, ψ, is given by:

$$\iint \omega \, dA = -\iint \nabla^2 \psi \, dA$$

$$\nabla^2 \psi_{i,j}^{n+1} \approx \frac{\psi_{i+1,j}^{n+1} - 2\psi_{i,j}^{n+1} + \psi_{i-1,j}^{n+1}}{\Delta x^2} + \frac{\psi_{i,j+1}^{n+1} - 2\psi_{i,j}^{n+1} + \psi_{i,j-1}^{n+1}}{\Delta y^2}$$

$$\omega_{i,j}^{n+1} = -\left(\frac{\psi_{i+1,j}^{n+1} - 2\psi_{i,j}^{n+1} + \psi_{i-1,j}^{n+1}}{\Delta x^2} + \frac{\psi_{i,j+1}^{n+1} - 2\psi_{i,j}^{n+1} + \psi_{i,j-1}^{n+1}}{\Delta y^2} \right)$$

Thus, the discretized equation for ψ at the time level $n+1$ is:

$$\left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} \right) \psi_{i,j}^{n+1} = \frac{1}{\Delta x^2} \psi_{i+1,j}^{n+1} + \frac{1}{\Delta x^2} \psi_{i-1,j}^{n+1} + \frac{1}{\Delta y^2} \psi_{i,j+1}^{n+1} + \frac{1}{\Delta y^2} \psi_{i,j-1}^{n+1} - \omega_{i,j}^{n+1}$$

Rearranging, we obtain a linear equation of the form:

$$a_P \psi_{i,j}^{n+1} = a_E \psi_{i+1,j}^{n+1} + a_W \psi_{i-1,j}^{n+1} + a_N \psi_{i,j+1}^{n+1} + a_S \psi_{i,j-1}^{n+1} + b$$

$$a_P = \frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}$$
$$a_E = \frac{1}{\Delta x^2}$$
$$a_W = \frac{1}{\Delta x^2}$$
$$a_N = \frac{1}{\Delta y^2}$$
$$a_S = \frac{1}{\Delta y^2}$$
$$b = -\omega_{i,j}^{n+1}$$

Python Code:

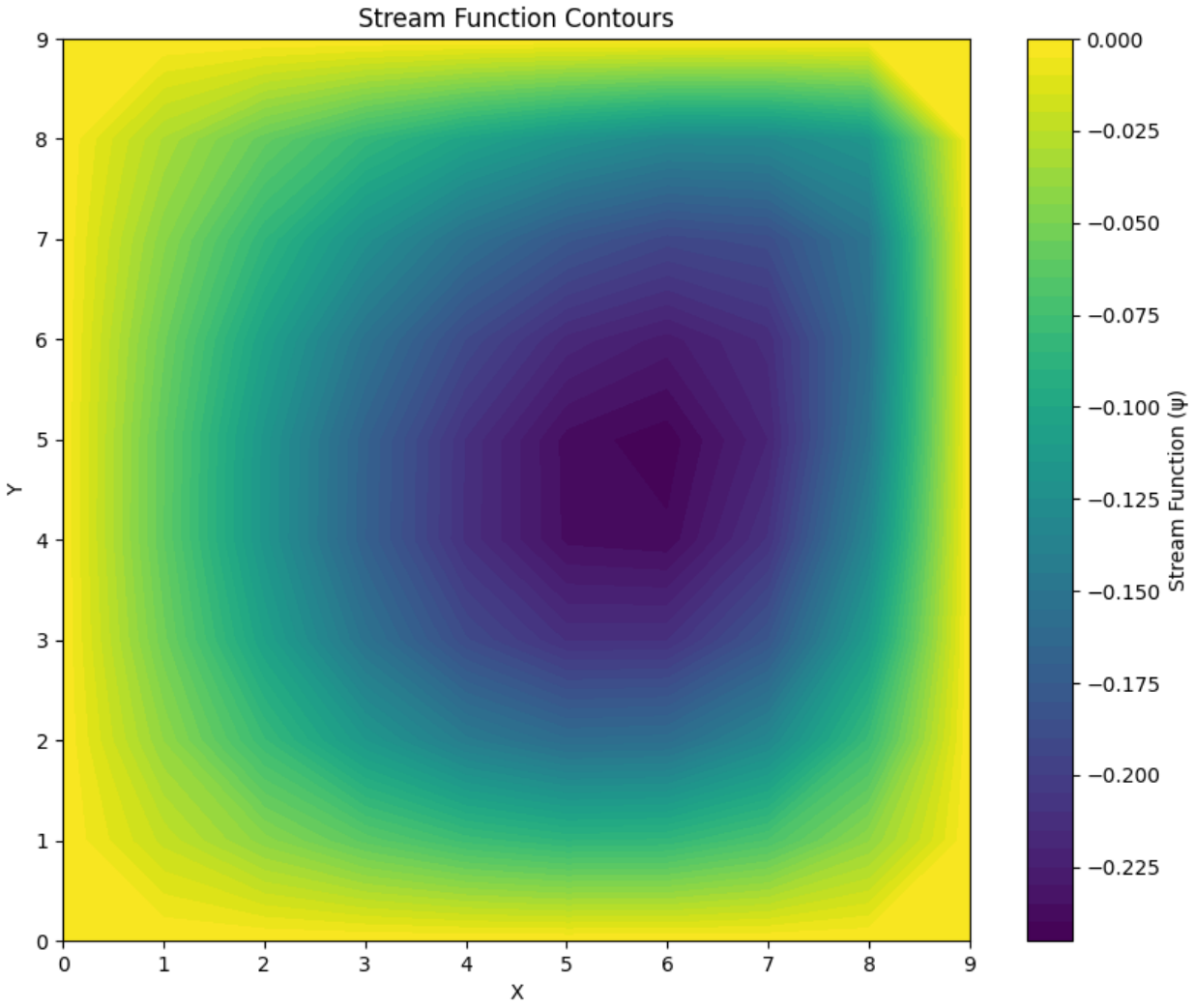
```
import numpy as np
import matplotlib.pyplot as plt

# Prompt for inputs
Lx, Ly = 1.0, 1.0  # Domain size remains constant in this example
Nx = int(input("Enter grid size in x-direction: "))
Ny = int(input("Enter grid size in y-direction: "))
Re = float(input("Enter Reynolds number: "))
U_top = float(input("Enter the velocity of the moving lid (U_top):
"))

# Derived parameters
nu = 1.0 / Re         # Viscosity (nu = 1/Re)
dx, dy = Lx / (Nx - 1), Ly / (Ny - 1)  # Grid spacing in x and y
dt = 0.001            # Time step, can be adjusted for stability
n_steps = 5000        # Number of time steps
tol = 1e-6            # Convergence tolerance for stream function

# Initialize variables
omega = np.zeros((Nx, Ny))  # Vorticity
psi = np.zeros((Nx, Ny))    # Stream function
Vx = np.zeros((Nx, Ny))     # Velocity in x direction
Vy = np.zeros((Nx, Ny))     # Velocity in y direction

# Function to apply boundary conditions
def apply_boundary_conditions(omega, psi):
    # Top wall (moving lid)
    psi[:, -1] = 0
    omega[:, -1] = -2 * U_top / dy

    # Bottom wall
    psi[:, 0] = 0
    omega[:, 0] = 0
```

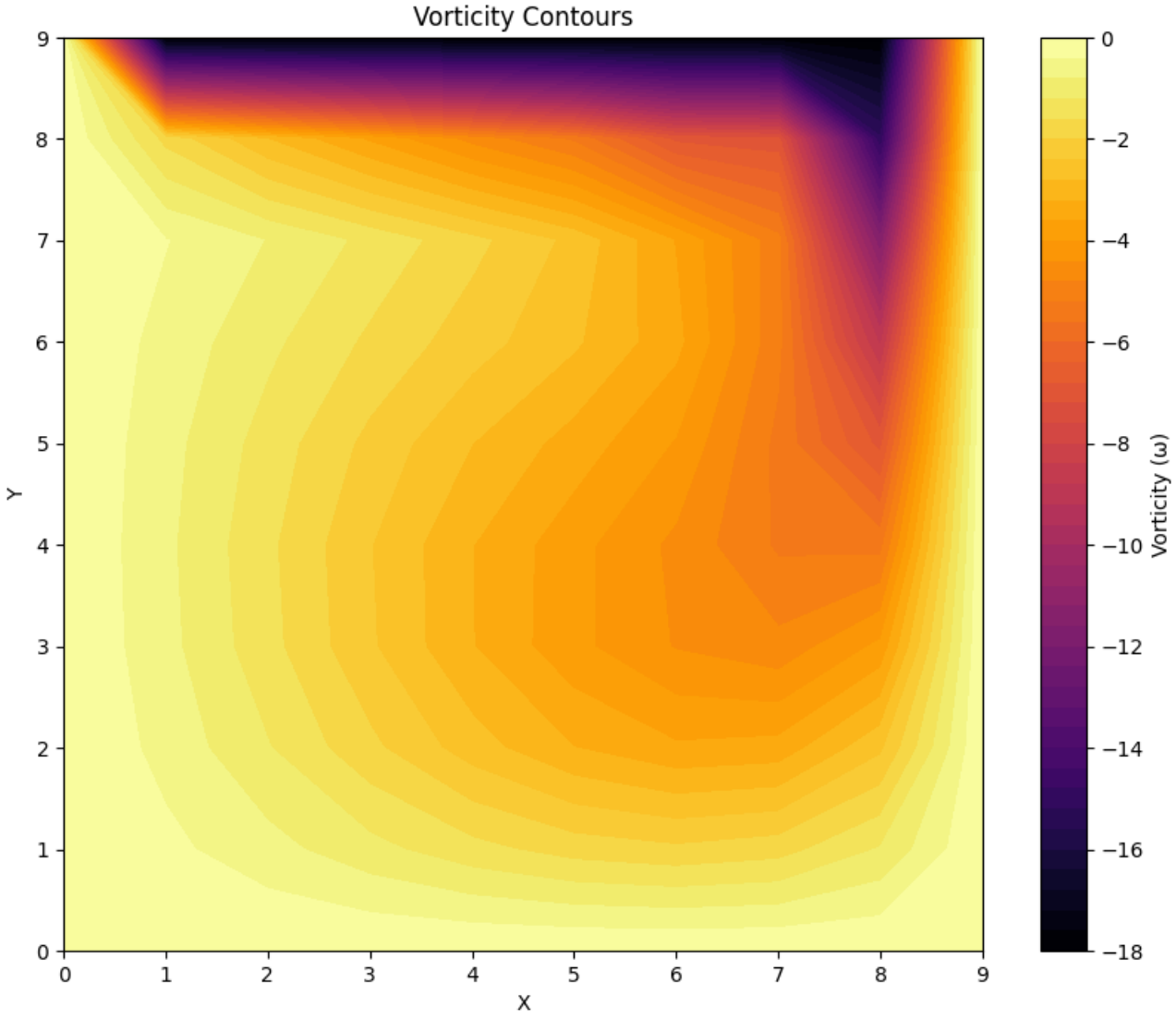
```
    # Left and right walls
    psi[0, :] = 0
    psi[-1, :] = 0
    omega[0, :] = 0
    omega[-1, :] = 0

# Time-stepping loop
for step in range(n_steps):
    # Compute velocity components from stream function
    Vx[1:-1, 1:-1] = (psi[1:-1, 2:] - psi[1:-1, :-2]) / (2 * dy)
    Vy[1:-1, 1:-1] = -(psi[2:, 1:-1] - psi[:-2, 1:-1]) / (2 * dx)

    # Discretize the vorticity transport equation
    omega_new = np.copy(omega)
    omega_new[1:-1, 1:-1] = (
        omega[1:-1, 1:-1]
        - dt * (Vx[1:-1, 1:-1] * (omega[2:, 1:-1] - omega[:-2, 1:-1]) / (2 *
dx)
                + Vy[1:-1, 1:-1] * (omega[1:-1, 2:] - omega[1:-1, :-2]) / (2 *
dy))
        + dt * nu * ((omega[2:, 1:-1] - 2 * omega[1:-1, 1:-1] + omega[:-
2, 1:-1]) / dx**2
                    + (omega[1:-1, 2:] - 2 * omega[1:-1, 1:-1] + omega[1:-1,
:-2]) / dy**2)
    )

    # Update omega with new values and apply boundary conditions
    omega = np.copy(omega_new)
    apply_boundary_conditions(omega, psi)

    # Solve Poisson equation for stream function using Gauss-Seidel
iteration
    for _ in range(100):  # Inner iterations for Poisson equation
        psi_old = np.copy(psi)
```

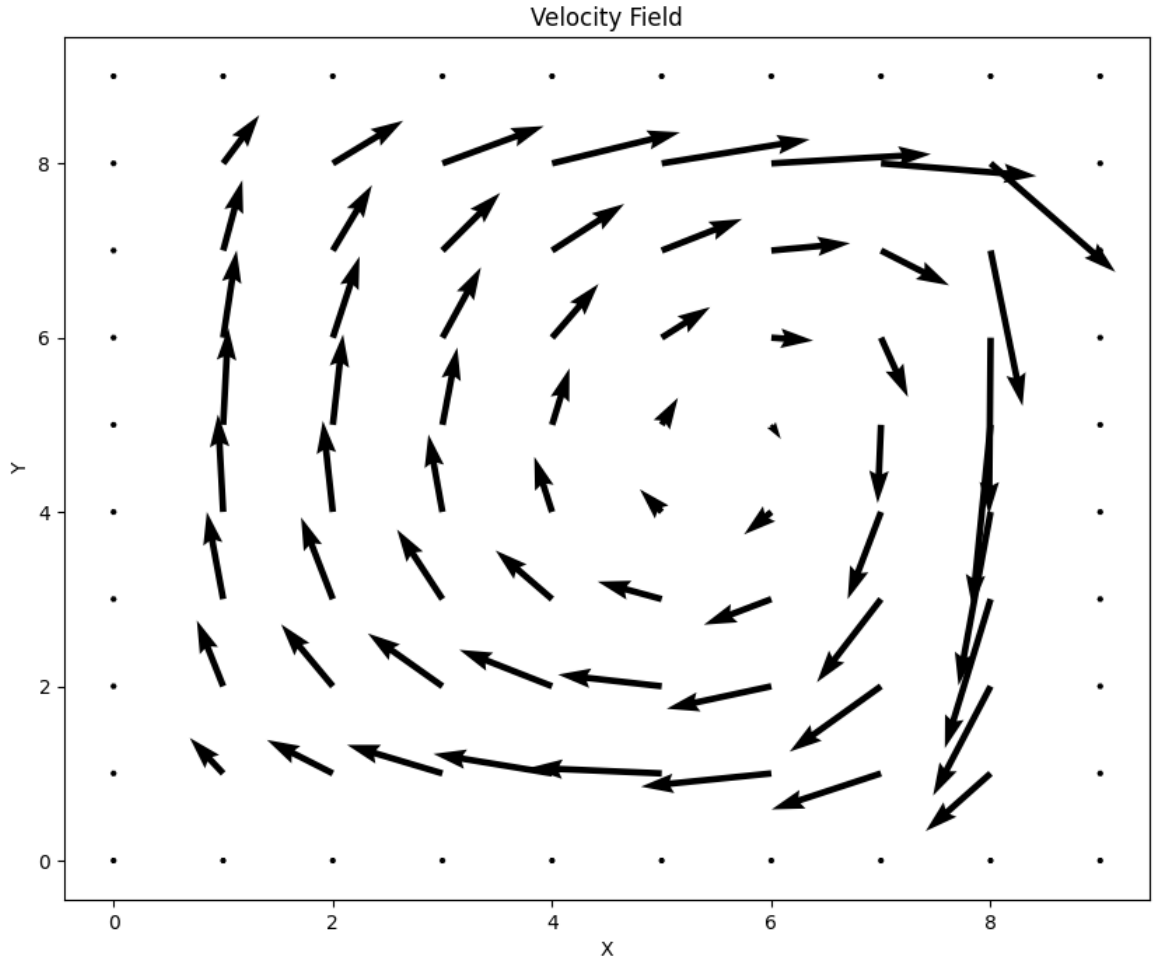
```
        psi[1:-1, 1:-1] = 0.25 * (psi[2:, 1:-1] + psi[:-2, 1:-1] + psi[1:-1, 2:]
+ psi[1:-1, :-2]
                                  + omega[1:-1, 1:-1] * dx**2)
        # Check for convergence
        if np.max(np.abs(psi - psi_old)) < tol:
            break

# Plotting results

# Stream Function Contour
plt.figure(figsize=(10, 8))
plt.contourf(psi.T, levels=50, cmap="viridis")
plt.colorbar(label="Stream Function (ψ)")
plt.title("Stream Function Contours")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()

# Vorticity Contour
plt.figure(figsize=(10, 8))
plt.contourf(omega.T, levels=50, cmap="inferno")
plt.colorbar(label="Vorticity (ω)")
plt.title("Vorticity Contours")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()

# Velocity Field
plt.figure(figsize=(10, 8))
plt.quiver(Vx.T, Vy.T)
plt.title("Velocity Field")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

Outputs:

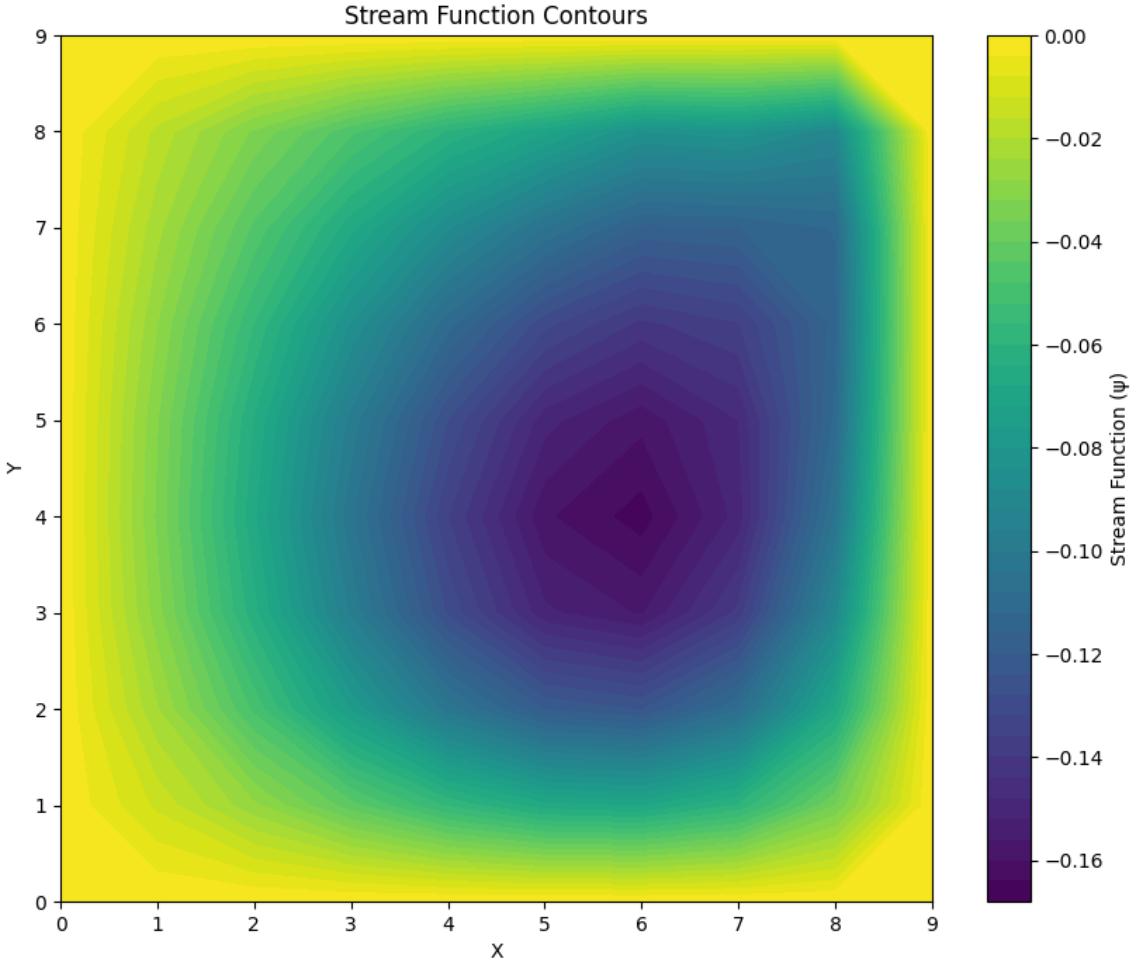
```
Enter grid size in x-direction: 10
Enter grid size in y-direction: 10
Enter Reynolds number: 50
Enter the velocity of the moving lid (U_top): 1
```

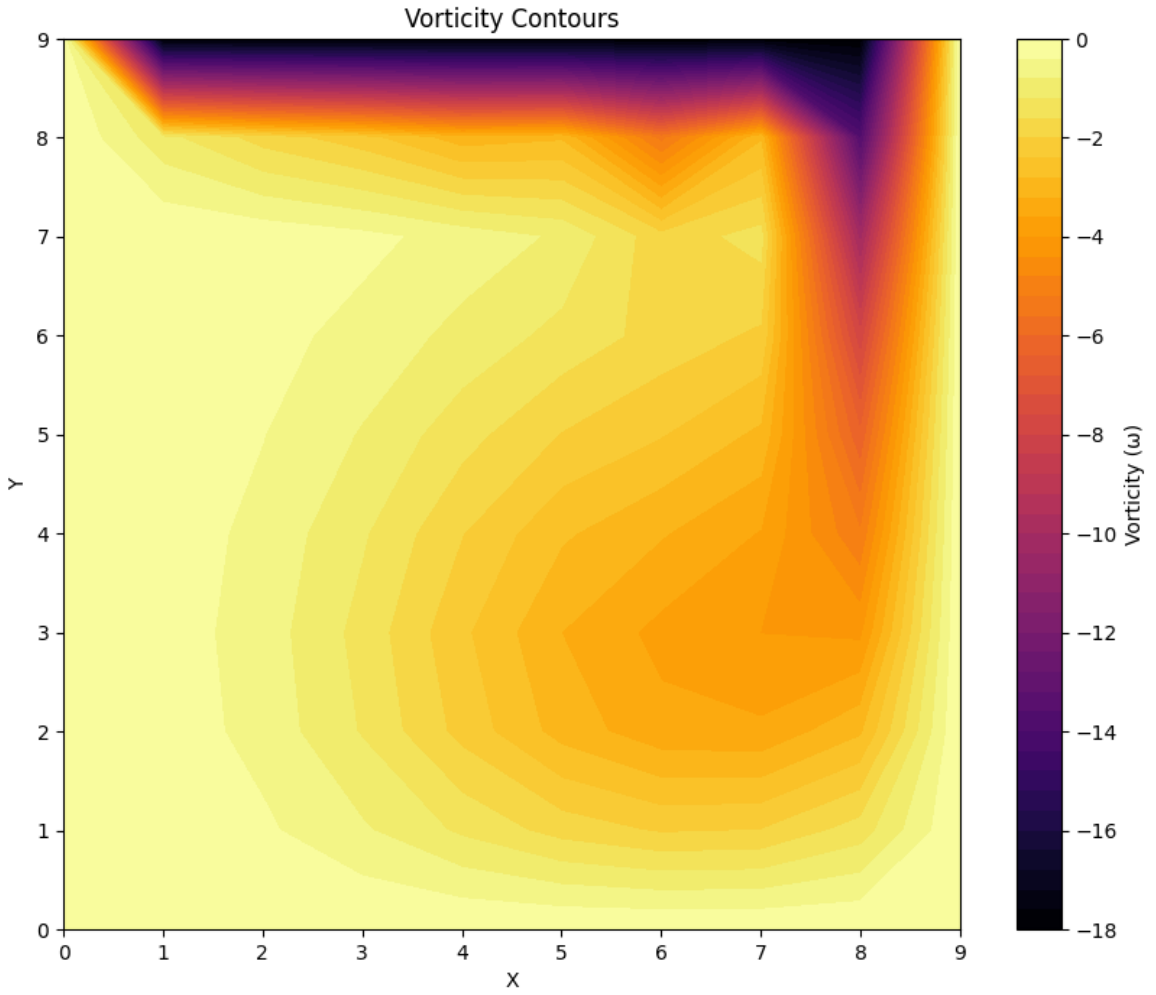


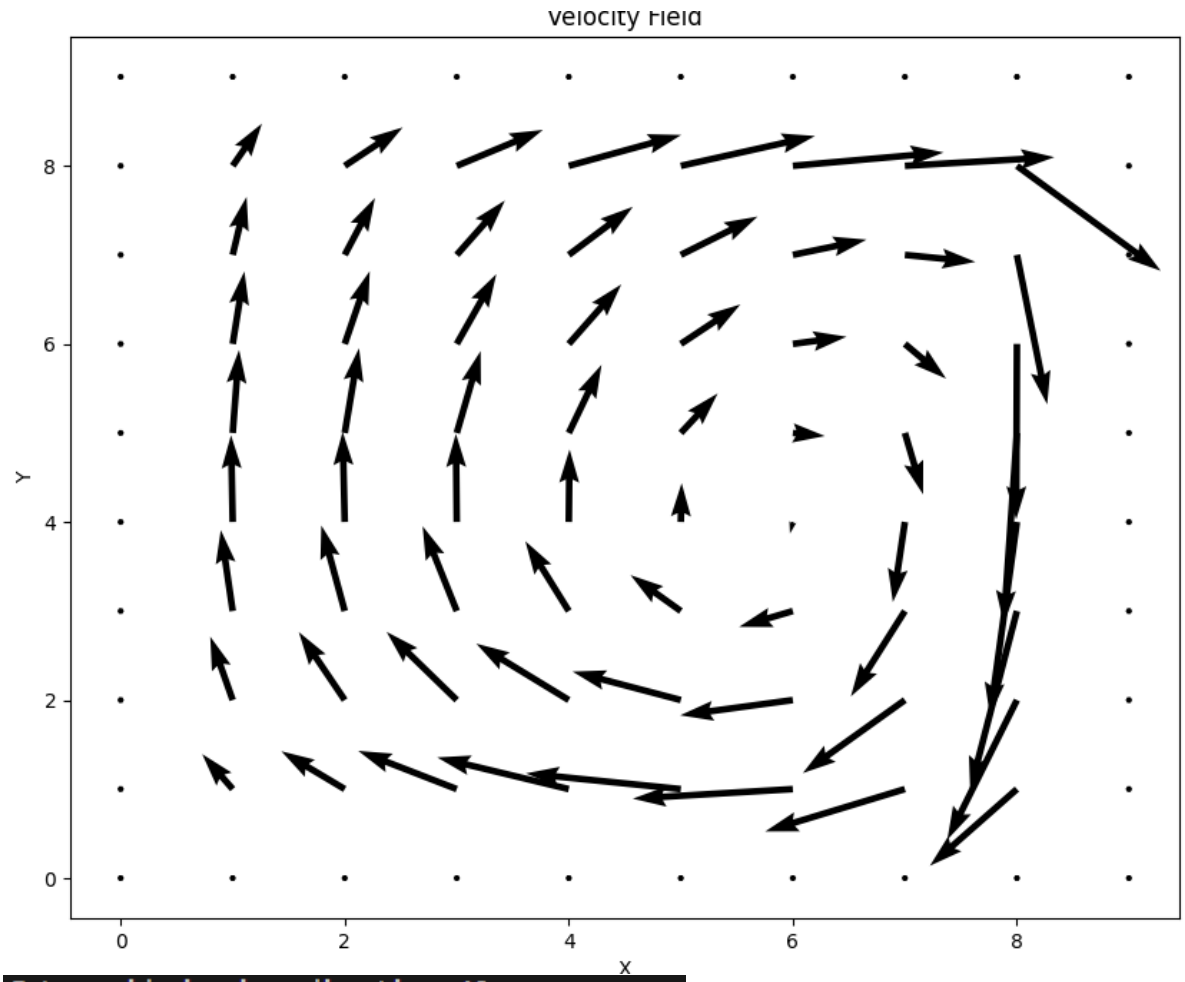

Vorticity Contours
Y
X
Vorticity (ω)
0
1
2
3
4
5
6
7
8
9
0
−2
−4
−6
−8
−10
−12
−14
−16
−18



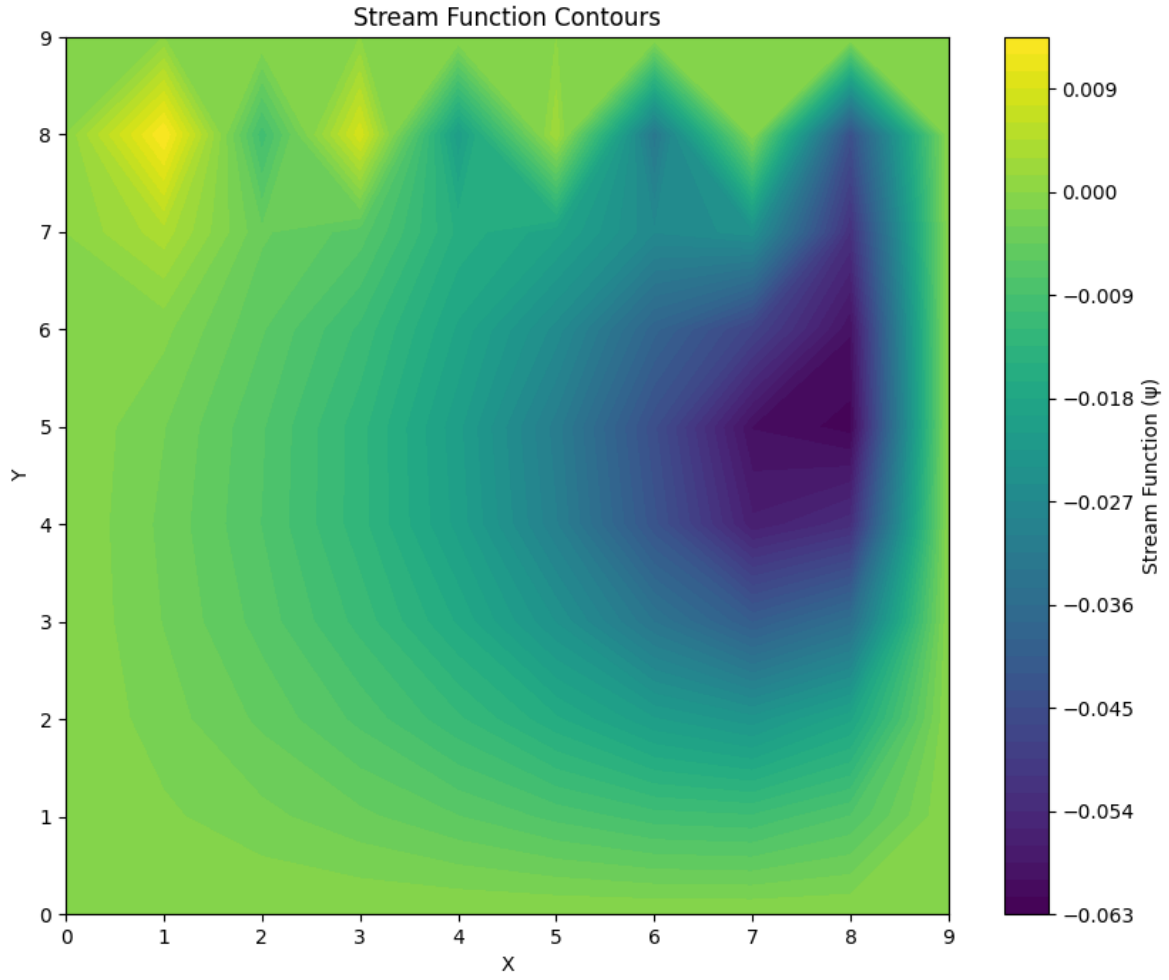

```
Enter grid size in x-direction: 10
Enter grid size in y-direction: 10
Enter Reynolds number: 100
Enter the velocity of the moving lid (U_top): 1
```

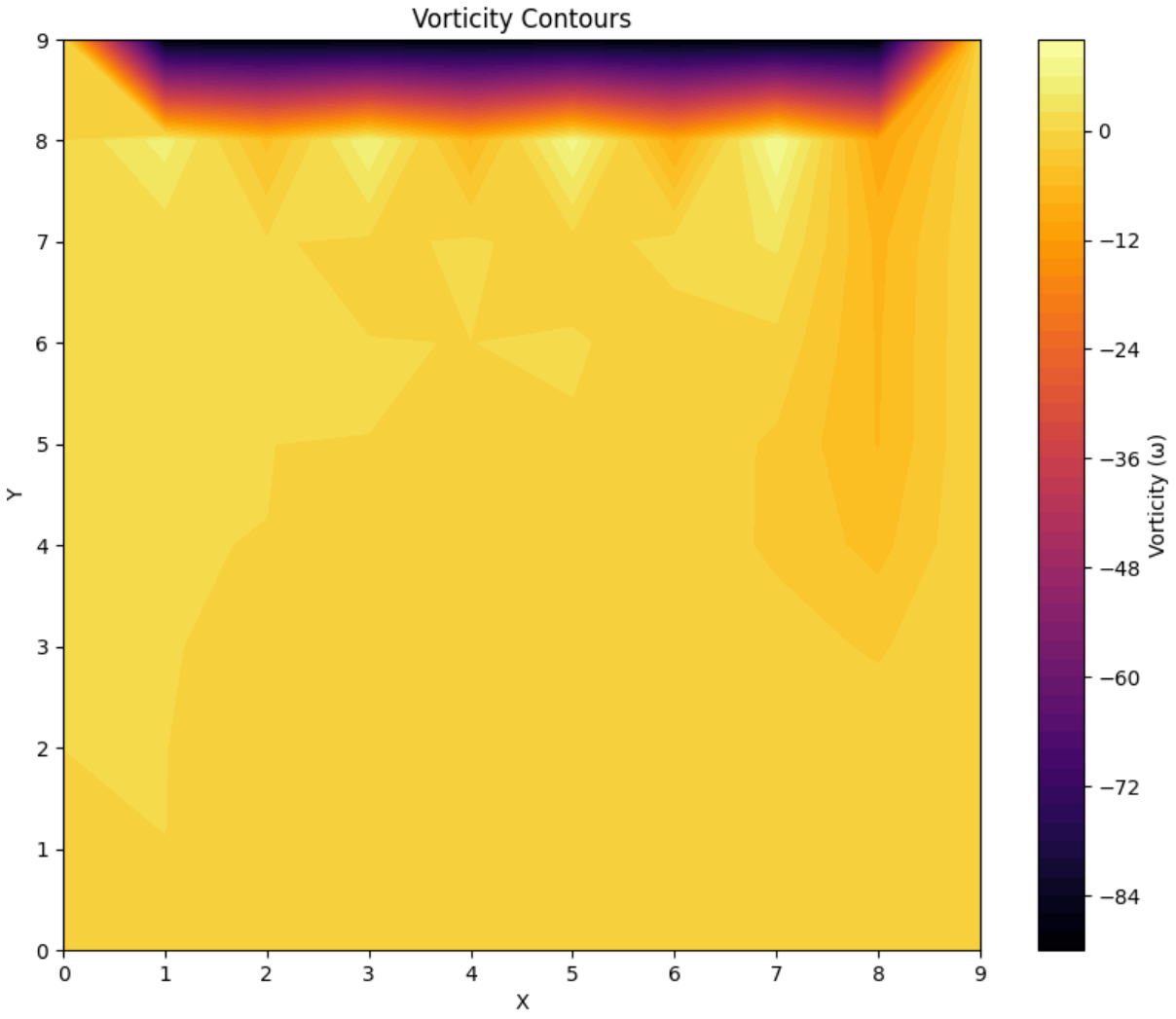

Stream Function Contours
Y
X
Stream Function (ψ)
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
−0.12
−0.14
−0.16

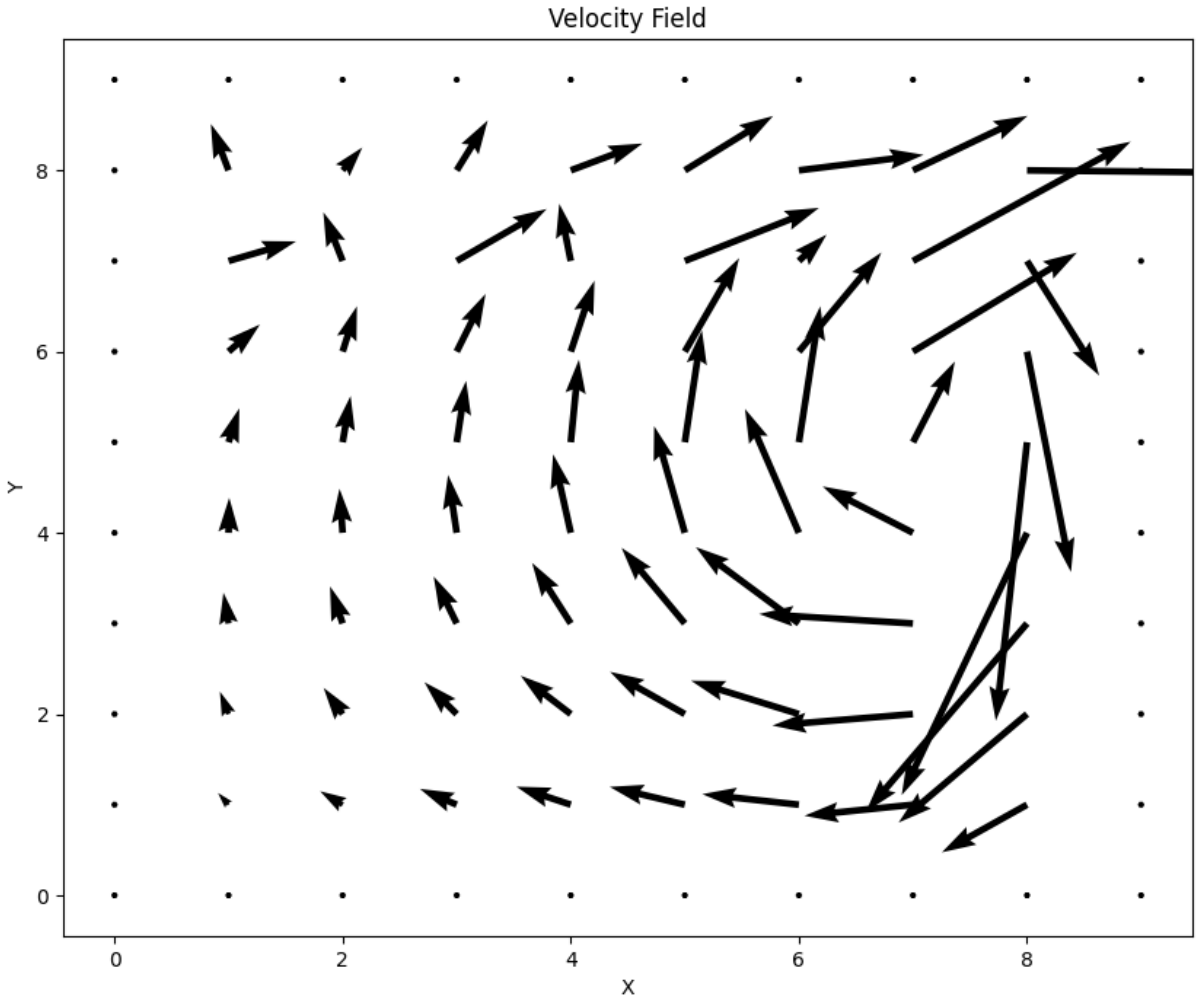

Vorticity Contours
Y
X
Vorticity (ω)
0
1
2
3
4
5
6
7
8
9
0
−2
−4
−6
−8
−10
−12
−14
−16
−18




```
Enter grid size in x-direction: 10
Enter grid size in y-direction: 10
Enter Reynolds number: 500
Enter the velocity of the moving lid (U_top): 5
```


Stream Function Contours
Y
X
Stream Function (ψ)
0
1
2
3
4
5
6
7
8
9
0.009
0.000
−0.009
−0.018
−0.027
−0.036
−0.045
−0.054
−0.063


Vorticity Contours
Y
X
Vorticity (ω)
0
1
2
3
4
5
6
7
8
9
0
−12
−24
−36
−48
−60
−72
−84


Velocity Field
Y
X
0
2
4
6
8
0
2
4
6
8