MIT - 113: COMPUTATIONAL FLUID MECHANICS

Simulation of Lid-Driven Cavity Flow

- **Apoorv Bhardwaj - 23117026**
- **Atharv Priyadarshi - 23117034**

Abstract

This report details the development and implementation of a numerical solver for the two-dimensional lid-driven cavity flow problem, a classic benchmark in computational fluid dynamics (CFD). The solver employs a finite difference method, utilizing the vorticity-stream function formulation of the incompressible Navier-Stokes equations. We validate the solver against established benchmark data and analyze the flow characteristics at various Reynolds numbers, visualizing vorticity, velocity, streamlines, shear stress, and divergence. This project provides valuable insights into the dynamics of viscous, incompressible flows and strengthens understanding of numerical methods in CFD.

Introduction

The lid-driven cavity flow is a fundamental problem in fluid mechanics, characterized by a square cavity filled with fluid where the top lid moves at a constant velocity, driving the internal fluid motion. This seemingly simple setup exhibits complex flow patterns that vary significantly with the Reynolds number (Re), making it an ideal benchmark for validating numerical methods in CFD. This report presents a numerical solver developed using a finite difference approach and analyzes the simulated flow behavior at different Reynolds numbers.

Numerical Method

Our solver uses a second-order accurate finite difference method on a uniform staggered grid. The vorticity-stream function formulation is chosen for its computational efficiency and its inherent satisfaction of the incompressibility condition. This formulation eliminates the pressure term from the Navier-Stokes equations, simplifying the numerical solution.

The governing equations in this formulation are:

Vorticity Transport Equation: $\partial\omega/\partial t + u\partial\omega/\partial x + v\partial\omega/\partial y = v\nabla^2\omega$
Stream function Equation: $\nabla^2\psi = -\omega$
Velocity Components: $u = \partial\psi/\partial y$, $v = -\partial\psi/\partial x$

where:

- ω is vorticity

- ψ is the stream function
- u and v are the velocity components in the x and y directions, respectively
- v is the kinematic viscosity
- t is time
- x and y are the spatial coordinates

The numerical solution advances in time using the following steps:

1. **Boundary Conditions:** No-slip conditions (u = v = 0) are applied at all walls except for the top lid, where u = u_lid (lid velocity) and v = 0.
2. **Vorticity Solver:** The vorticity transport equation is discretized using a second-order central difference scheme in space and an explicit forward Euler scheme in time.
3. **Stream function Solver:** The Poisson equation for the stream function is solved using an iterative method (e.g., successive over-relaxation).
4. **Velocity Solver:** The velocity components are computed from the stream function using central differences.
5. **Convergence Check:** The solution is deemed converged when the maximum change in vorticity between successive time steps falls below a specified tolerance.

Code Implementation

The Python code is structured into several functions for modularity and readability (refer to the code documentation for details):

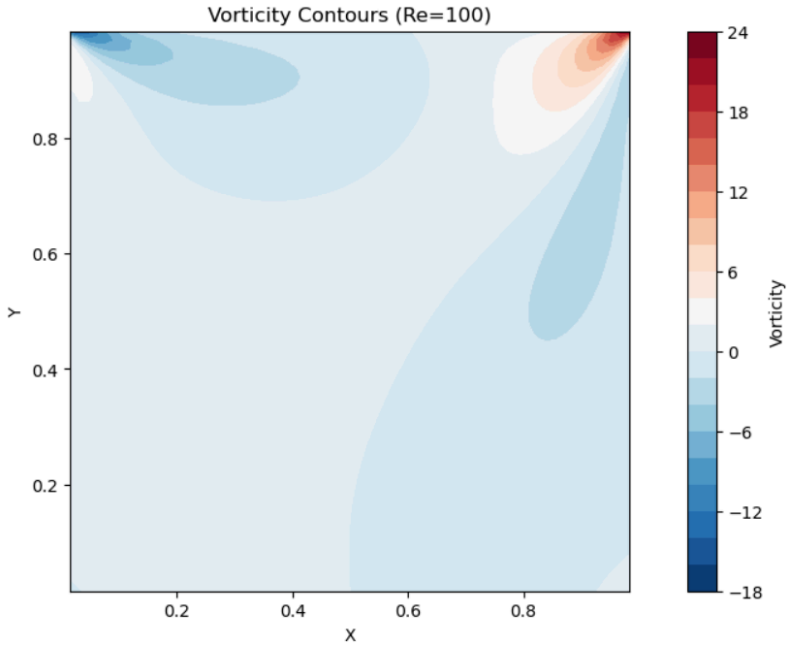
1. setup_domain(n_x, l): Creates the computational grid.
2. initialize_fields(n_x, n_y): Initializes the flow variables.
3. boundary_conditions(omega, psi, delta_x, u_lid): Applies the boundary conditions.
4. vorticity_solver(...): Implements the finite difference discretization and time-stepping for vorticity.
5. streamfunction_solver(...): Solves the Poisson equation for the streamfunction.
6. velocity_solver(...): Computes the velocity components.
7. calculate_divergence(...): Computes the divergence of the velocity field.
8. calculate_vorticity(...): Computes vorticity from velocities.
9. lid_driven_cavity(...): Combines all the solver steps within a loop.
10. interpolate_field(...): Performs bilinear interpolation for particle advection.
11. plot_results(...): Generates all the visualizations.

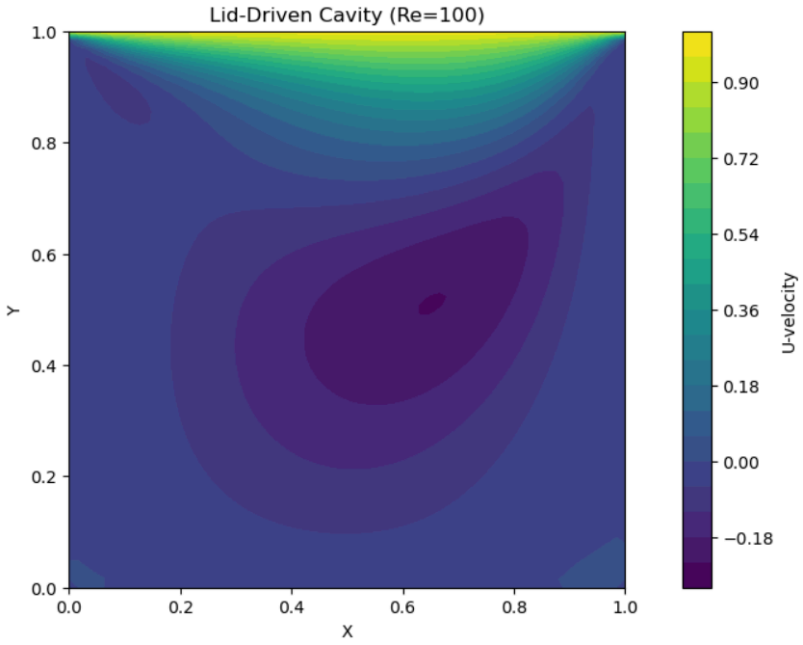
Validation

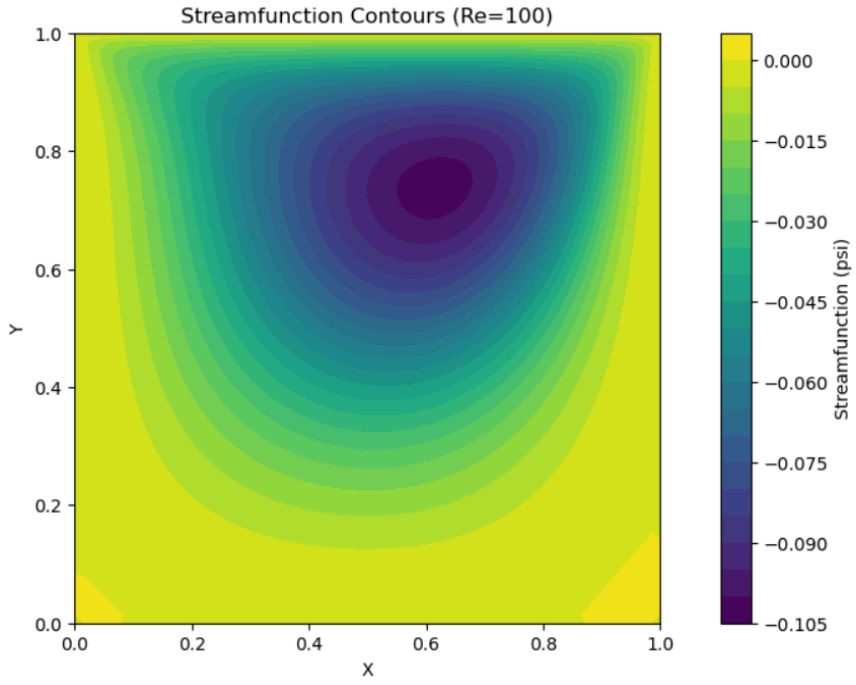
The code was validated against benchmark results from Ghia, Ghia, and Shin (1982) [1] for Reynolds numbers of 100, 400, and 1000.

Results and Discussion

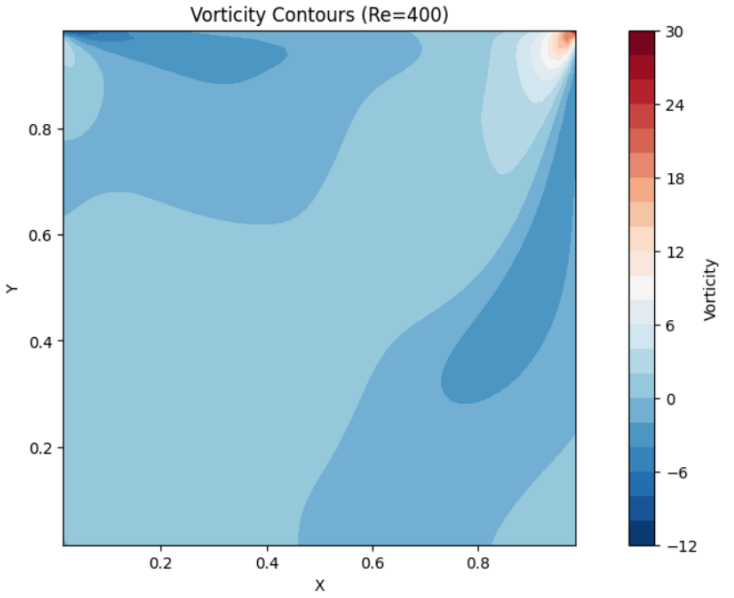
Case 1: Re = 100 (Nx = 64)

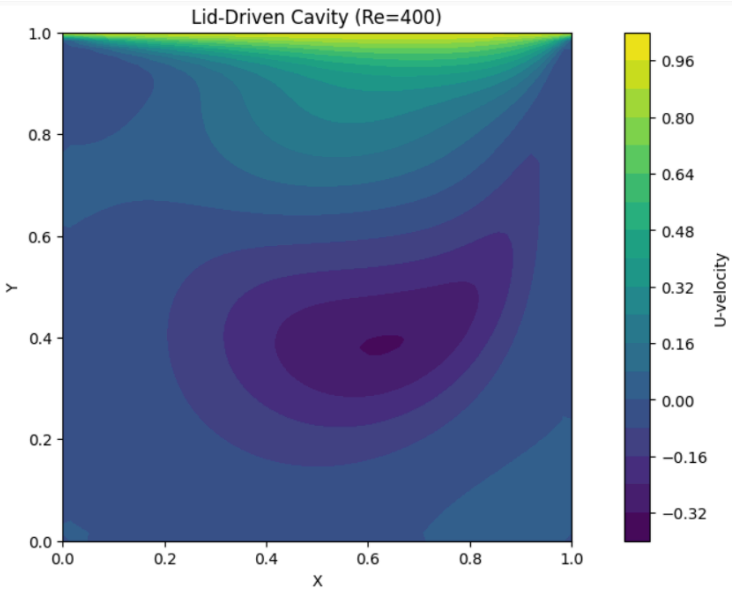


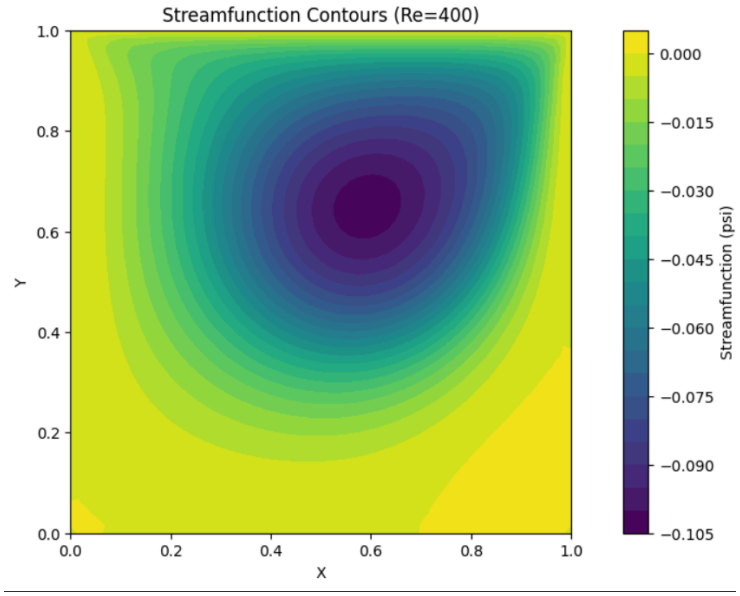




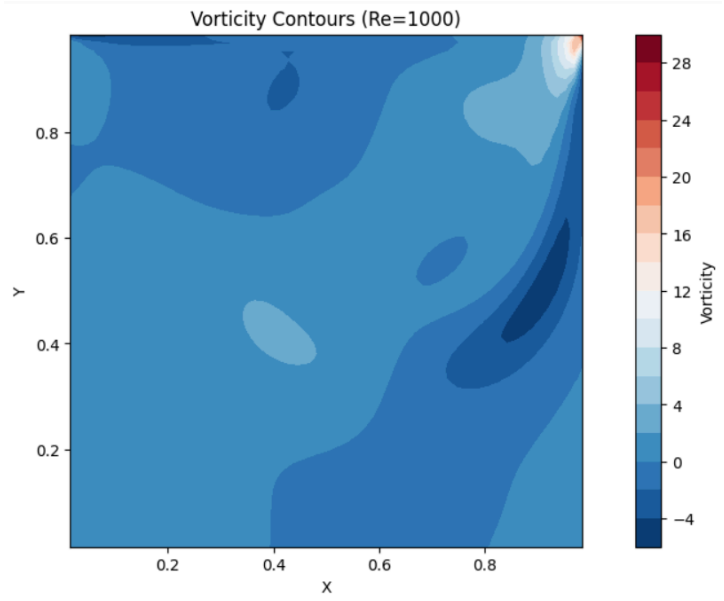
Case 2: Re = 400 (Nx = 128)

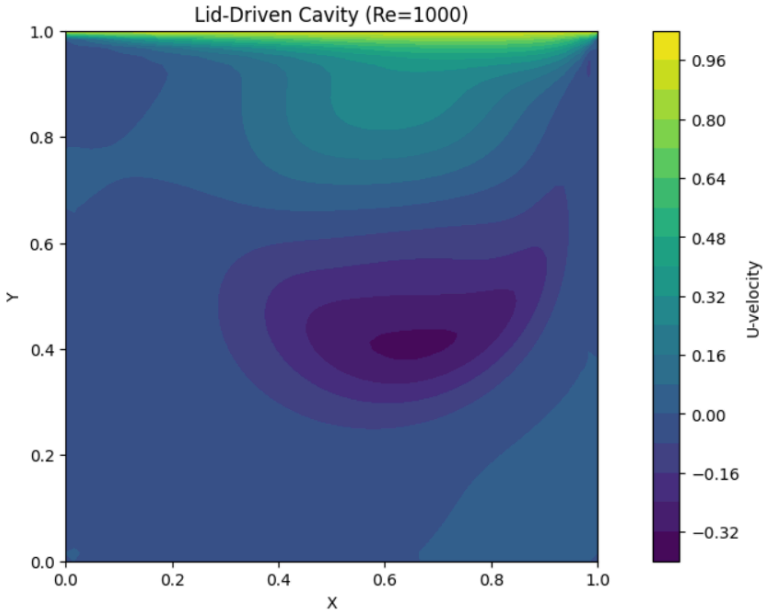

Vorticity Contours (Re=400)
Y
X
0.2
0.4
0.6
0.8
Vorticity
30
24
18
12
6
0
−6
−12

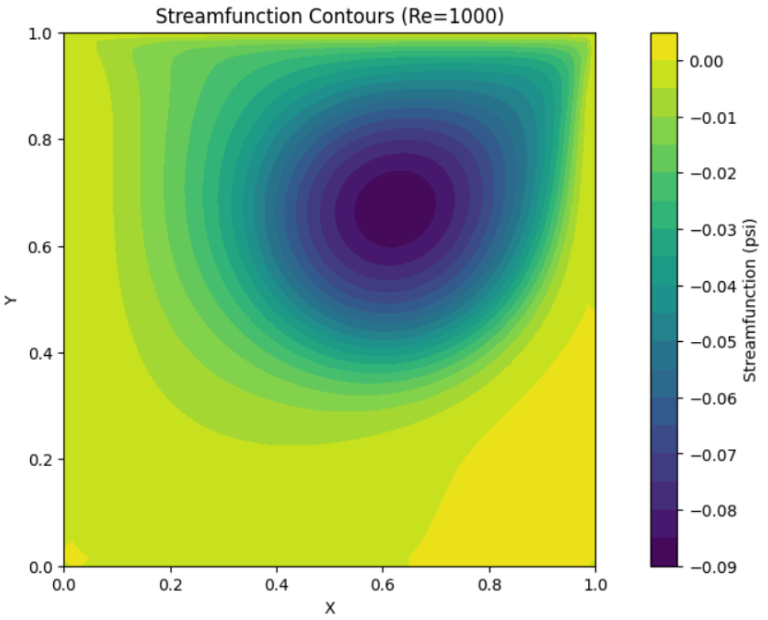

Lid-Driven Cavity (Re=400)
Y
X
0.0
0.2
0.4
0.6
0.8
1.0
U-velocity
0.96
0.80
0.64
0.48
0.32
0.16
0.00
−0.16
−0.32



Case 3: Re = 1000 (Nx = 256)







Learnings and Outcomes

Through this project, I gained a deeper understanding of:

Numerical Methods for CFD: Practical experience in applying finite difference methods to solve the Navier-Stokes equations.
Vorticity-Stream function Formulation: Understanding of this approach and its advantages for incompressible flows.
Flow Physics: Insights into the dynamics of lid-driven cavity flow and the effect of the Reynolds number on flow patterns.

Code Development for Scientific Computing: Improved skills in structuring, implementing, and documenting scientific code in Python.