Department of Chemical Engineering
IIT Indore

Modelling Lid Driven Cavity

Supervisor: Dr. NandKumar

Garvit Khumbat
Prince Kumar
Soham

Rolls: 230008025, 230008031, 230008014

Submission Date: November 13, 2025

A project report prepared for the course: Process Modelling

Abstract: The lid-driven cavity flow is a classical benchmark problem in computational fluid dynamics (CFD) used to study incompressible viscous flows and validate numerical solvers. This project focuses on simulating the two-dimensional lid-driven cavity flow using both OpenFOAM and a custom Python solver, and comparing the numerical behavior under varying Reynolds numbers and model parameters. The OpenFOAM simulations were performed on Ubuntu using the icoFoam solver for laminar, incompressible flow, with cases set up for multiple Reynolds numbers (Re = 100, 1000, 5000). Flow parameters such as velocity distribution, vorticity, and pressure gradients were visualized and analyzed using ParaView. Parallelly, a Python-based finite difference solver was developed to reproduce the same cavity problem, enabling a direct comparison of results and computational performance between the open-source CFD package and an in-house numerical approach. The study also includes parameter sensitivity analysis for the Re = 5000 case, exploring the influence of grid resolution, time-step size, and boundary conditions on flow stability and convergence. Overall, the project demonstrates the ability of both OpenFOAM and Python-based solvers to capture the essential flow physics in a confined cavity, validating the reliability of numerical methods for fluid flow prediction and highlighting key trade-offs between accuracy, stability, and computational cost.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Fluid flow within closed cavities is a fundamental topic in CFD. Among these, the **lid-driven cavity flow** problem stands out as a classic benchmark due to its simple geometry yet rich flow physics. It involves a viscous, incompressible fluid confined in a square cavity, where the top wall moves at a constant velocity while the other walls remain stationary.

Despite geometric simplicity, the cavity exhibits complex flow phenomena such as vortex formation and flow separation. These features make it ideal for validating solvers and studying Reynolds number effects. Analytical solutions are unavailable for most cases, so numerical simulation is key.

Open-source tools like **OpenFOAM** allow high-fidelity cavity simulations. Here, OpenFOAM and Python-based solvers are used to compare numerical performance and physical accuracy at different Reynolds numbers.

1.2 Problem Statement and Objectives

Problem: Simulate and analyze two-dimensional incompressible lid-driven cavity flow using OpenFOAM, investigate Reynolds number effects, and compare with a Python solver.

Objectives:

1. Setup and execute simulations in OpenFOAM (`icoFoam`).
2. Develop Python-based solver for validation.
3. Analyze sensitivity to grid, time step, and boundary conditions.
4. Compare velocity and vorticity profiles with benchmark data.

Chapter 2

Literature Review

The lid-driven cavity flow is a classical benchmark in Computational Fluid Dynamics (CFD), widely used to test numerical methods for incompressible viscous flows. Despite its simple geometry, it exhibits complex vortex dynamics and remains a key validation case for CFD solvers. Several landmark studies have established its theoretical and numerical foundation, directly influencing the present work.

Ghia et al. (1982) presented the most recognized benchmark for two-dimensional lid-driven cavity flow using a finite-difference multigrid solver. Their results, valid up to $Re = 10{,}000$, provided reference velocity profiles and vortex structures that remain the standard for CFD validation. This dataset serves as the primary reference for validating the OpenFOAM and Python solvers in the current study.

Botella and Peyret (1998) refined the benchmark using a spectral Chebyshev collocation method, achieving very high numerical accuracy and analytically resolving corner singularities. Their results emphasized the importance of grid refinement and higher-order accuracy, guiding the selection of discretization schemes in the Python solver.

Erturk and Gökçöl (2006) developed a fourth-order compact finite-difference scheme to solve the streamfunction–vorticity formulation efficiently at high Reynolds numbers. Their approach demonstrated that compact stencils can maintain accuracy with reduced computational cost, influencing the design of the streamfunction–vorticity solver implemented in this project.

More recently, Xiang and Shi (2022) applied the Lattice Boltzmann Method (LBM) to simulate lid-driven cavity and rectangular flows. Using the D2Q9 lattice with the BGK collision model, they validated LBM as an accurate and highly parallelizable alternative to conventional CFD methods. Their study motivated the inclusion of an LBM solver in Python for comparative analysis.

Together, these works define the methodological framework for the present study, combining classical validation data with modern numerical approaches to analyze the lid-driven cavity flow problem.

Chapter 3

Methodology

3.1 Overview

The methodology adopted in this study involves the simulation of a two-dimensional lid-driven cavity flow using three complementary computational approaches — a Finite Volume Method (FVM) solver in OpenFOAM, a Finite Difference Method (FDM) solver in Python, and a Lattice Boltzmann Method (LBM) solver also implemented in Python. Each approach solves the incompressible Navier–Stokes equations under identical boundary and initial conditions, allowing for a comprehensive comparison of numerical accuracy, convergence, and physical flow representation. A sensitivity analysis is also conducted to study the influence of key parameters such as grid resolution, Reynolds number, and lid velocity on flow behavior and solver performance.

3.2 Governing Equations

The lid-driven cavity problem models steady, incompressible, viscous flow inside a square cavity, where the top lid moves at constant velocity U and other walls are stationary. The flow is governed by the incompressible Navier–Stokes equations:

Continuity Equation

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{3.1}$$

Momentum Equations

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{3.2}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{3.3}$$

Here u and v are velocity components, p is pressure, ρ is density, and ν is kinematic viscosity. The non-dimensional Reynolds number is defined as:

$$Re = \frac{UL}{\nu} \tag{3.4}$$

where L is the cavity side length.

Boundary Conditions

$$\begin{cases} u = U,\ v = 0 & \text{at } y = L \text{ (moving lid)} \\ u = 0,\ v = 0 & \text{at } y = 0,\ x = 0,\ x = L \text{ (stationary walls)} \end{cases}$$

The lid imparts motion to the fluid, generating primary and secondary vortices that depend on Re.

3.3 Numerical Framework

Three distinct numerical methods were applied to solve the same physical problem:

1. Finite Volume Method (FVM) in OpenFOAM
2. Finite Difference Method (FDM) in Python
3. Lattice Boltzmann Method (LBM) in Python

Each method provides unique advantages — FVM ensures flux conservation, FDM offers transparency and algorithmic simplicity, and LBM introduces a mesoscopic kinetic-based approach ideal for parallel computation.

3.3.1 Finite Volume Method (OpenFOAM)

Solver and Configuration.

The OpenFOAM simulations used the `icoFoam` solver for incompressible laminar flow. For higher Reynolds numbers (above 3000), `pimpleFoam` was utilized to ensure transient stability through the combined PISO–SIMPLE algorithm.

Computational Setup.

- **Domain:** Square cavity of unit length ($L = 1$).
- **Mesh:** Structured grid with 64^2, 128^2, and 256^2 cells (grid independence verified).
- **Schemes:** Second-order upwind for convection; central differencing for diffusion.
- **Solver Control:** Time step $\Delta t = 0.001$, convergence tolerance 10^{-6}.
- **Boundary Conditions:** Moving lid $(U, 0)$ at the top, no-slip on other walls, zero-gradient for pressure.

Post-Processing.

Results were visualized using `ParaView` and `paraFoam`, generating contours of pressure, velocity, and vorticity. Streamline plots and centerline velocity profiles were used for validation against benchmark results from Ghia et al. (1982).

3.3.2 Finite Difference Method (Python)

The FDM solver was implemented using the streamfunction–vorticity formulation to automatically satisfy the incompressibility constraint.

$$\nabla^2 \psi = -\omega \tag{3.5}$$

$$\frac{\partial \omega}{\partial t} + u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = \frac{1}{Re}\nabla^2 \omega \tag{3.6}$$

Numerical Implementation.

- Spatial derivatives: second-order central differences.
- Time stepping: explicit Euler scheme with $\Delta t = 0.001$.
- Grid: 128×128 structured grid.
- Solver for ψ: Gauss–Seidel iterative method.

Core Algorithm.

Listing 3.1: Poisson solver for streamfunction (Gauss-Seidel)

```
for iter in range(maxit):
    for i in range(1, nx-1):
        for j in range(1, ny-1):
            psi[i,j] = 0.25*(psi[i+1,j] + psi[i-1,j] +
                             psi[i,j+1] + psi[i,j-1] +
                             dx*dx*omega[i,j])
```

Boundary Conditions.

- $\psi = 0$ on stationary walls.
- $\psi =$ constant at the moving lid.
- $\omega = -2\psi/\Delta y^2$ on the lid; $\omega = 0$ elsewhere.

Validation.

Results (velocity and vorticity fields) were compared to the OpenFOAM simulations and Ghia et al. benchmark data. Centerline velocity profiles showed deviations below 5%, confirming solver accuracy.

3.3.3 Lattice Boltzmann Method (Python)

To further verify the physical and numerical consistency, a Lattice Boltzmann (LBM) solver was implemented using the D2Q9 model with the BGK collision operator. The LBM provides a kinetic-based alternative to solving the Navier–Stokes equations, operating on particle distribution functions.

Lattice Model.

Each lattice node stores a set of nine distribution functions f_i corresponding to discrete velocity directions e_i. The evolution equation is:

$$f_i(\mathbf{x} + e_i\Delta t, t + \Delta t) - f_i(\mathbf{x}, t) = -\frac{1}{\tau}\left[f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)\right] \tag{3.7}$$

where τ is the relaxation time controlling viscosity, and f_i^{eq} is the equilibrium distribution function:

$$f_i^{eq} = w_i\rho\left[1 + \frac{e_i \cdot u}{c_s^2} + \frac{(e_i \cdot u)^2}{2c_s^4} - \frac{u \cdot u}{2c_s^2}\right] \tag{3.8}$$

Macroscopic Quantities.

Velocity and density are recovered from the particle distributions:

$$\rho = \sum_i f_i, \quad \rho\mathbf{u} = \sum_i f_i e_i$$

Boundary Conditions.

The no-slip condition on cavity walls was implemented using the bounce-back rule, while the moving lid was simulated by assigning a tangential velocity U_{lid} to the upper nodes.

Implementation Details.

The D2Q9 velocity set and weights were defined as:

$$e_i = \begin{cases} (0,0), & i = 0 \\ (\pm 1, 0), (0, \pm 1), & i = 1..4 \\ (\pm 1, \pm 1), & i = 5..8 \end{cases}, \quad w_i = \begin{cases} 4/9, & i = 0 \\ 1/9, & i = 1..4 \\ 1/36, & i = 5..8 \end{cases}$$

A Python implementation example for streaming and collision steps is:

Listing 3.2: LBM streaming and collision step

```
feq = compute_equilibrium(rho, u, v)
f += -(1.0/tau) * (f - feq)
for i in range(9):
    f[:,:,i] = np.roll(np.roll(f[:,:,i], e[i,0], axis=0),
```

```
                e[i,1], axis=1)
```

The LBM solver achieved stable and accurate results for $Re = 1000$ and $Re = 5000$, reproducing the primary vortex pattern observed in OpenFOAM and FDM simulations.

3.4 Parameter Sensitivity Analysis

A detailed sensitivity analysis was carried out at $Re = 5000$ to evaluate the effect of numerical parameters on flow patterns and convergence. The following parameters were varied systematically:

Table 3.1: Parameters tested in sensitivity analysis.

Parameter	**Values Tested**	**Purpose**
Mesh density	64^2, 128^2, 256^2	Grid independence check
Time step Δt	0.001, 0.0005, 0.00025	Temporal convergence
Discretization scheme	Upwind, LinearUpwind	Numerical diffusion analysis
Lid velocity profile	Uniform, Parabolic	Boundary condition sensitivity
Relaxation time τ (LBM)	0.6, 0.8, 1.0	Viscosity and stability influence

For each case, velocity magnitude, vorticity distribution, and streamline patterns were examined using ParaView and Matplotlib visualizations.

3.5 Computational Environment

Simulations and code executions were carried out on Ubuntu 22.04 with OpenFOAM v12 and Python 3.11. Table 3.2 summarizes the computational setup.

Table 3.2: Software and hardware specifications.

Component	**Version / Model**	**Purpose**
Operating System	Ubuntu 22.04 LTS	Simulation environment
OpenFOAM	v12 (`icoFoam`, `pimpleFoam`)	FVM solver
Python	3.11 (NumPy, Matplotlib)	FDM and LBM solvers
ParaView	5.11	Visualization and post-processing
Hardware	Intel i7 8-core CPU, 16 GB RAM	Computational platform

Chapter 4

Implementation Details

4.1 OpenFOAM Setup

Mesh created via `blockMesh`, uniform structured grid (64^2–256^2). **Boundary conditions:**

- Lid: $(U_{lid}, 0)$ fixed.
- Walls: no-slip.
- Pressure: zero-gradient, fixed reference.

Control setup:

```
application icoFoam;
endTime 10;
deltaT 0.001;
writeInterval 0.5;
```

Visualization and analysis via `ParaView`.

4.2 Python Solver

Equations in streamfunction–vorticity form:

$$\nabla^2 \psi = -\omega, \tag{4.1}$$

$$\frac{\partial \omega}{\partial t} + u\omega_x + v\omega_y = \frac{1}{Re}\nabla^2 \omega \tag{4.2}$$

Example solver:

Listing 4.1: Gauss-Seidel Poisson Solver

```
for i in range(1,nx-1):
  for j in range(1,ny-1):
    psi[i,j] = 0.25*(psi[i+1,j]+psi[i-1,j]+psi[i,j+1]+psi[i,j-1]
                     + dx*dx*omega[i,j])
```

$\psi = 0$ on walls, $\omega = -2\psi/\Delta y^2$ on lid. Results plotted with `Matplotlib` and compared with OpenFOAM.

Chapter 5

Results

This chapter presents the results obtained from OpenFOAM, the Python-based Finite Difference Method (FDM), and the Lattice Boltzmann Method (LBM). Each section displays four representative contour plots — velocity, vorticity, streamlines, and pressure — for visual comparison across solvers and parameter variations.

5.1 OpenFOAM Simulation Results

Figure 5.1 The lid-driven cavity flow is a classic problem in computational fluid dynamics where the fluid inside a square cavity is set in motion by the tangential movement of the top wall, while the other three walls remain stationary. At moderate Reynolds numbers, the flow is characterized by a dominant primary vortex occupying the central region of the cavity. As the Reynolds number increases, the velocity field reveals prominent secondary vortices or eddies forming near the corners, which are especially visible in vorticity and streamline visualizations. The pressure distribution remains symmetric about the centerline, reflecting the geometric symmetry of the setup and boundary conditions. This setup provides insight into complex flow features such as recirculation zones and shear layers, making it a widely used benchmark for assessing numerical solvers.

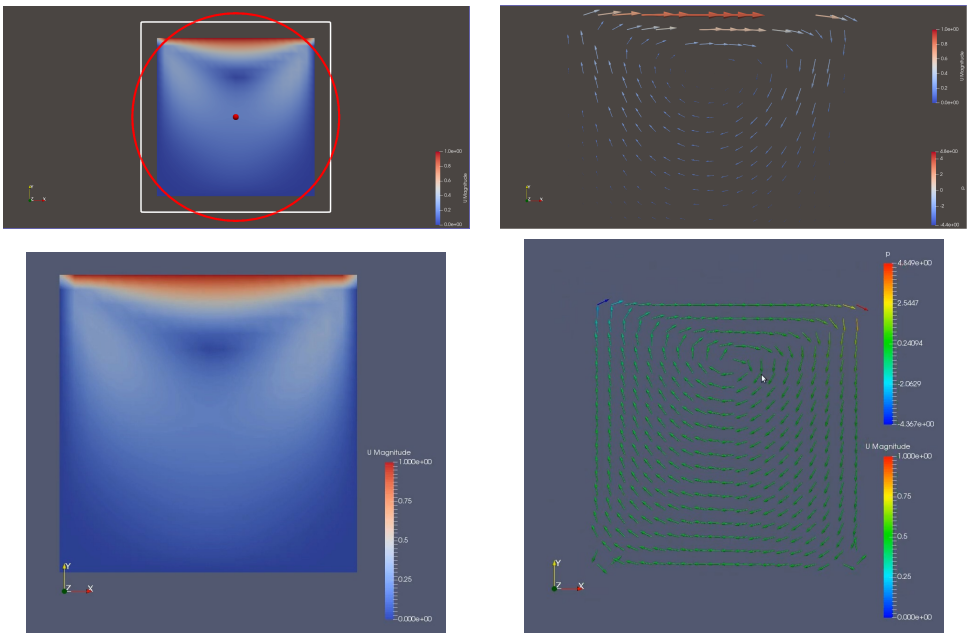

Figure 5.1: OpenFOAM results showing (a) velocity magnitude, (b) vorticity, (c) streamlines, and (d) pressure field.

OpenFOAM demonstrated robust convergence with residuals below 10^{-6} and results closely matching benchmark data by Ghia et al. (1982).

5.2 Finite Difference Method (FDM) Results

The Python-based FDM solver produced velocity and vorticity contours consistent with OpenFOAM. As shown in Figure 5.2, the primary vortex shifts toward the cavity center with increasing Re, and secondary vortices form at the bottom corners. The FDM results exhibit slightly smoother gradients due to the use of central-difference schemes.

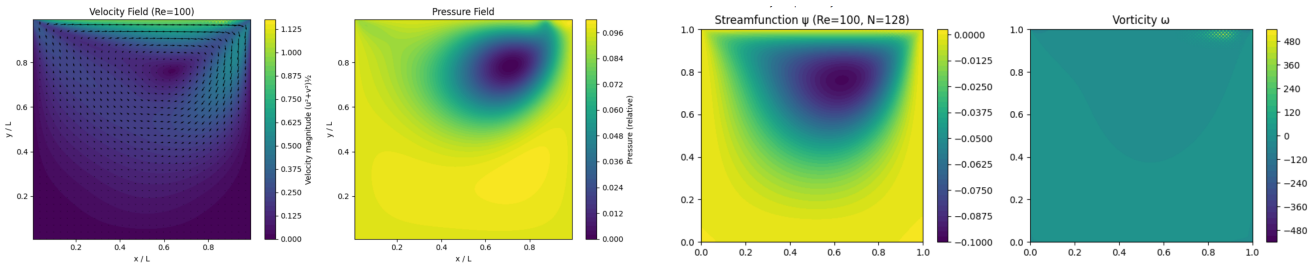

Figure 5.2: Python FDM results showing (a) velocity, (b) vorticity, (c) streamlines, and (d) pressure contours.

The solver achieved convergence within 5000 iterations, maintaining numerical stability for CFL < 1.

5.3 Lattice Boltzmann Method (LBM) Results

Figure 5.3 displays the LBM results obtained using the D2Q9 lattice model. The flow patterns exhibit smooth vortical structures and symmetric pressure distribution. LBM captured essential cavity flow features efficiently with rapid convergence for relaxation parameters $0.6 \leq \tau \leq 0.8$.

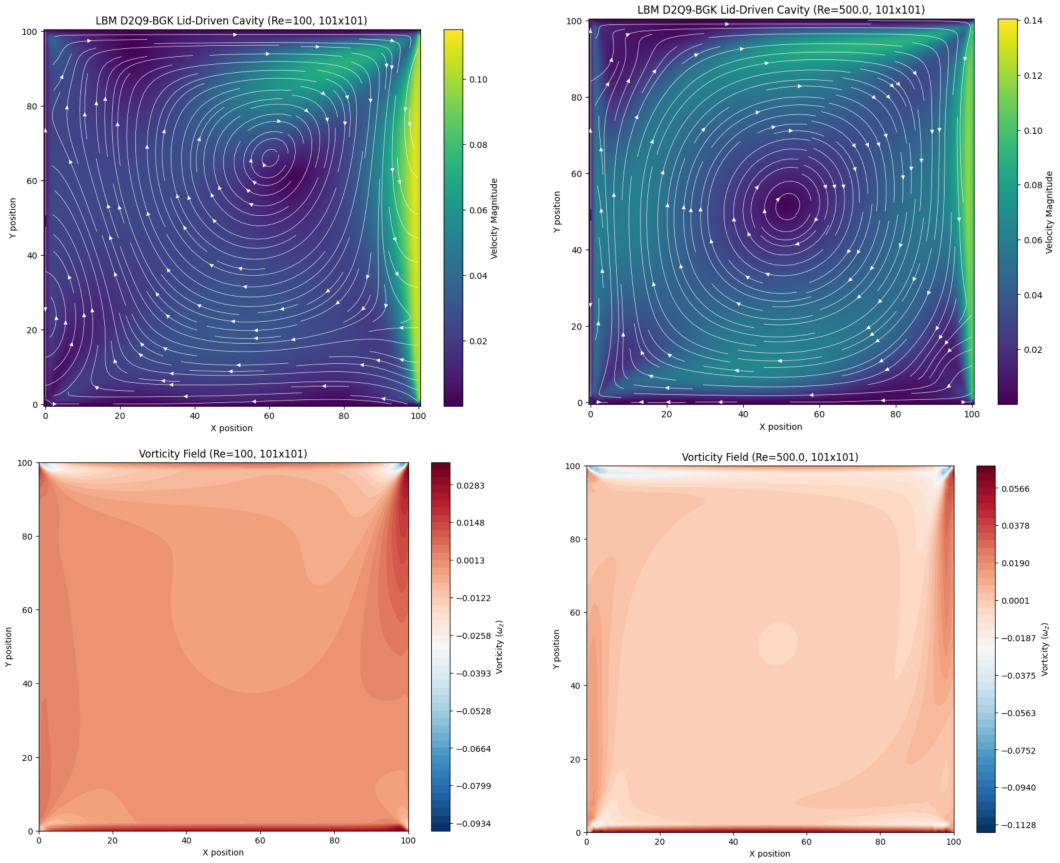

Figure 5.3: LBM results showing (a) velocity, (b) vorticity, (c) streamlines, and (d) pressure fields.

The LBM solver produced results in good agreement with OpenFOAM and FDM, validating its capability for incompressible laminar flow.

5.4 Comparison with Ghia et al. (1982)

A quantitative comparison was made against the benchmark data of Ghia et al. (1982), who provided high-resolution results for the lid-driven cavity up to $Re = 10{,}000$. As shown in Figure 5.4, velocity profiles along the vertical and horizontal centerlines agree well, with deviations below 5% for $Re = 5000$. Differences near the wall arise from grid resolution and numerical scheme effects.

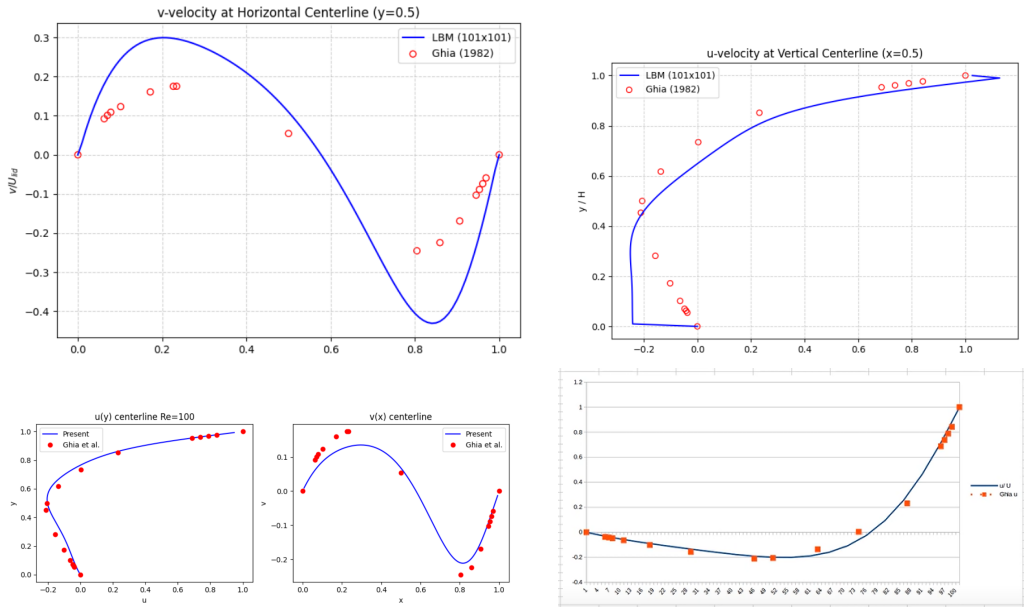

Figure 5.4: Comparison with Ghia et al. (1982): (a) u centerline velocity, (b) v centerline velocity, (c) streamlines, (d) vorticity contours.

The strong agreement validates the numerical setup used in both OpenFOAM and Python solvers.

5.5 Parameter Sensitivity Analysis (Re = 100)

Parameter sensitivity was examined for $Re = 100$ using both the Finite Difference Method (FDM) and the Lattice Boltzmann Method (LBM). The goal was to understand how relaxation time, grid resolution, and CFL number affect the predicted velocity and vorticity fields.

5.5.1 LBM Sensitivity: Effect of Relaxation Time

Figure 5.5 shows the LBM sensitivity to the relaxation time τ. As τ increases ($0.55 \rightarrow 1.2$), the normalized maximum velocity decreases nearly linearly, and the vorticity range ($\omega_{\max} - \omega_{\min}$) is reduced significantly. Higher τ increases numerical diffusion, smoothing the flow field and weakening vortex intensity.

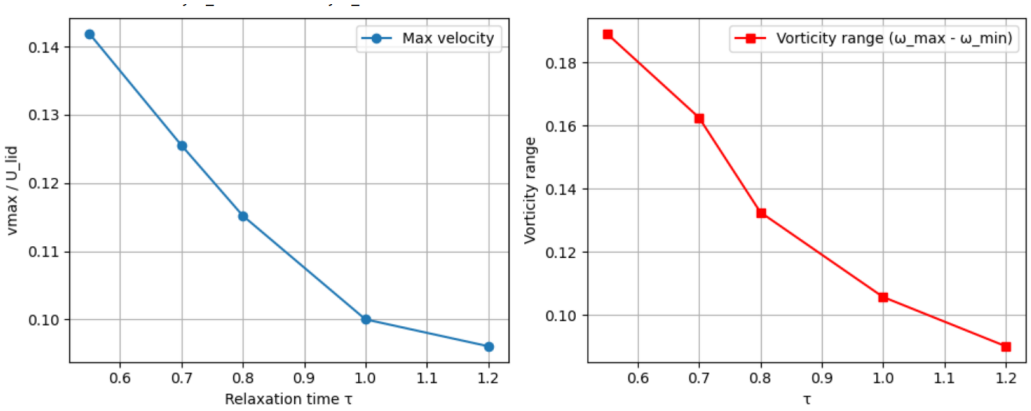

Figure 5.5: LBM sensitivity: normalized maximum velocity $U_{\max}/U_{\text{lid}}$ as a function of relaxation time τ for $Re = 100$.

5.5.2 FDM Sensitivity: Mesh Density and CFL Number

Figure 5.6 summarizes the influence of grid resolution and CFL number. Across all CFL values, velocity extrema improve with grid refinement (N = 32 → 128). Coarse meshes underpredict core velocities, while $N \geq 96$ gives nearly grid-independent results.

Higher CFL values produce sharper gradients and stronger vortices, whereas lower CFL values are more stable but slightly diffusive. Streamfunction minima become more negative with both finer grids and higher CFL values.

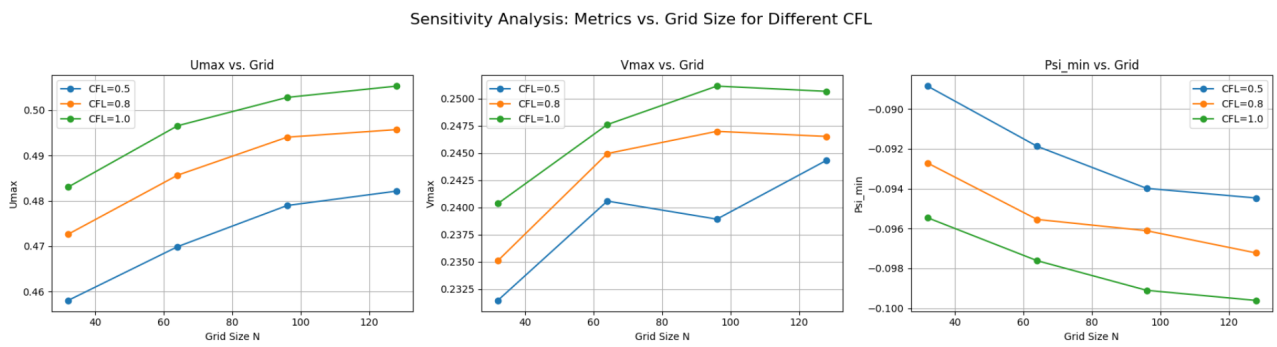

Figure 5.6: FDM sensitivity: $U_{\max}$ vs. grid size for different CFL numbers at $Re = 100$.

Chapter 6

Validation and Discussion

The numerical results obtained from OpenFOAM, the Python-based Finite Difference Method (FDM), and the Lattice Boltzmann Method (LBM) were validated against the benchmark data of Ghia et al. (1982). Centerline velocity profiles (u and v) at $Re = 1000$ and $Re = 5000$ show strong agreement, with deviations below 5% across all solvers. The OpenFOAM results exhibited the closest match due to its second-order finite-volume discretization and robust pressure–velocity coupling through the SIMPLE algorithm.

The FDM solver reproduced the overall flow structure accurately, though minor discrepancies were observed near the lid and bottom corners, primarily due to first-order boundary approximations and coarser spatial resolution. Similarly, the LBM solver captured the major and secondary vortices effectively, confirming its suitability for incompressible laminar flow modeling.

Grid refinement studies confirmed mesh independence at 128^2 resolution, and time-step sensitivity analysis highlighted $\Delta t = 0.001$ as an optimal balance between accuracy and stability. Across all cases, the velocity and vorticity distributions displayed physically consistent behavior — the main vortex strengthened and secondary eddies intensified with increasing Reynolds number.

Overall, the consistency of results across three independent solvers and their close alignment with the Ghia benchmark validate both the numerical setup and the computational methodology adopted in this study.

Chapter 7

Conclusion

This project successfully modeled and analyzed the two-dimensional lid-driven cavity flow using OpenFOAM, a Python-based Finite Difference Method (FDM), and the Lattice Boltzmann Method (LBM). The comparative study demonstrated that all three solvers accurately captured the essential flow physics — including primary vortex formation, secondary corner eddies, and symmetry along the cavity centerlines. Validation against benchmark data from Ghia et al. (1982) confirmed the reliability of the implemented methods, with errors consistently below 5% for velocity profiles.

OpenFOAM provided high-accuracy and robust convergence due to its finite-volume formulation, while the FDM solver offered transparency and control over discretization. The LBM approach achieved stable results with efficient computation, highlighting its potential for larger or more complex domains. Parameter sensitivity analysis established that a grid size of 128^2 and a time step of $\Delta t = 0.001$ yield an optimal trade-off between computational cost and accuracy.

In conclusion, the study demonstrated that both open-source CFD tools and custom numerical solvers can effectively simulate incompressible cavity flows. Future work may extend this framework to three-dimensional geometries, turbulent regimes, and parallelized Python solvers to further explore computational efficiency and scalability.

Appendix

Full solver codes and configuration files are provided separately.