Simulation of Lid-Driven Cavity using Lattice Boltzmann Method and Parallel Computing

Duration of the Project: 4 Months (Dec 2022 to Mar 2023)
Project Mentors: Nihal L Bendre, Prasad Savanur
Team Members: Anandita Tiwari, Prashant Kumar Choudary, Santosh C

Introduction:

Abstract:

Lid-Driven Cavity is a problem which deals with a square cavity consisting of three rigid walls with no-slip conditions and a lid moving with a tangential unit velocity. The Lattice Boltzmann Method is a mesoscopic simulation method that considers the behaviour of a collection of particles as a unit, and a distribution function is used to represent the collection of particles. The LBM code was originally developed in Python and subsequently optimized for faster execution by converting it to Cython. The code was further accelerated by leveraging the power of OpenMP parallelization.

Inspiration:

The motivation for exploring LBM could stem from the need to find alternative computational methods for fluid dynamics simulations that can handle more complex scenarios and provide more accurate results. For example, traditional methods like FVM may struggle with complex geometries and non-Newtonian fluids, which are prevalent in many real-world scenarios. Thus, exploring the potential of LBM could lead to more accurate and efficient simulations in such scenarios.

In the project, the team implemented LBM using Python and Cython and used OpenMP for parallelizing the code. Cython is a programming language that allows for faster execution of Python code by converting it to C code. OpenMP is a widely used API for parallel programming in shared-memory architectures. By using these tools, the team was able to take advantage of the strengths of each language and library to create an efficient and accurate implementation of LBM.

Methodology:

The motive of this project is to compare the expected results of Lid-driven cavity simulated in ANSYS Fluent with the Lattice Boltzmann Method (LBM) coded in Python, later to further optimise the code with Cython, and to implement parallel computing using OpenMP. The process is carried out in the following stages:

1. ANSYS Fluent Simulation:

Theory:

The simulation is done by solving the 2D Navier-Stokes equation for constant density flow. The Continuity and Momentum equations for a steady state 2D constant density flow are given as follows:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0$$

$$\rho\left[\frac{\partial u}{\partial t}+\frac{\partial u}{\partial x}u+\frac{\partial u}{\partial y}v+\frac{\partial u}{\partial z}w\right]=-\frac{\partial p}{\partial x}+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)+\rho g_x$$

$$\rho\left[\frac{\partial v}{\partial t}+\frac{\partial v}{\partial x}u+\frac{\partial v}{\partial y}v+\frac{\partial v}{\partial z}w\right]=-\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right)+\rho g_y$$

$$\rho\left[\frac{\partial w}{\partial t}+\frac{\partial w}{\partial x}u+\frac{\partial w}{\partial y}v+\frac{\partial w}{\partial z}w\right]=-\frac{\partial p}{\partial z}+\mu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right)+\rho g_z$$

The general and boundary conditions taken are as follows:

1. 2D Surface (Square of side 1m)
2. 3 stationary walls and 1 moving wall.
3. Viscous incompressible fluid is filled inside.
4. Velocity of moving wall (Lid): 0.1 m/s (Tangential)
5. Velocity of other walls: 0 m/s (No Slip)
6. Reynolds number: 100
7. Residuals: 0.001

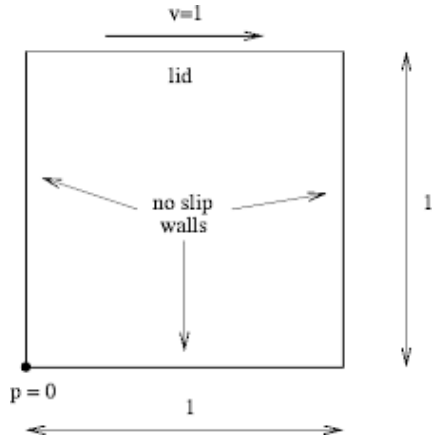

Fig. 1: Representation of Lid-Driven Cavity in consideration

Numerical Modelling:

Geometry: The geometry is developed based on the cavity size. The simulation of the lid-driven cavity flow has been used as a benchmark for developing the LBM because of its simple shape as well as its extensive application for analyzing complicated flows inside a cavity. This study is aimed at understanding the flow inside a cavity such that it would become important to understand how to implement boundary conditions inside

the cavity. No-slip boundary conditions were applied on both the sidewalls and the bottom of the cavity, as well as the obstacle surface, whereas a uniform horizontal velocity (i.e., U = 0.1) was applied at the lid of the cavity, as shown in Figure 1. In Figure 1, the size of each wall is set to the length L (i.e., L = 1). In order to understand the main circulation of the flow inside the cavity, a large circle is used to indicate the primary flow pattern in the cavity.

Meshing: As the Geometry is quite simple so we applied Face Meshing with edge divisions in 64 elements, which gave us pretty good statistics of mesh metrics.

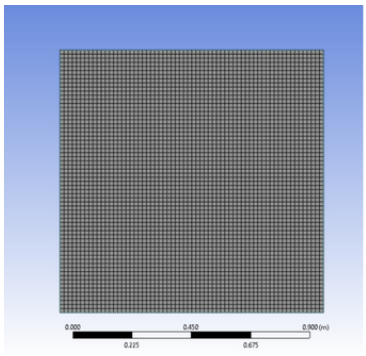

Fig. 2: Geometry with Mesh

2. LBM Code using Python:

Governing Equation:

In this study, the shape of a cavity considered is a square of size 1×1 (aspect ratio K=1), which includes three non-moving no-slip walls and a moving wall (U=0.1) at the top lid having a uniform horizontal velocity. In order to observe the flow instability in the calculation, which was our main concern in this study, Re values ranging from 100 to 5000 were considered. In order to calculate the cavity flow, the Lattice Boltzmann Equation (LBE) was adopted in this study. The LBE with the BGK collision operator is given as follows:

$$f_i\left(\mathbf{x}+\mathbf{c}_i\Delta t, t+\Delta t\right)-f_i\left(\mathbf{x},t\right)=-\frac{1}{\tau}\left(f_i\left(\mathbf{x},t\right)-f_i^{eq}\left(\mathbf{x},t\right)\right)$$

feqi(x,t) is the discrete distribution function and its equilibrium state, respectively, which form the nine-point integer D2Q9 lattice shown in Fig. 3. In addition, x is the spatial vector, ci is the discrete velocity in the i th direction, i is the moving index of a particular particle, and τ is the SRT that manages the rate of distribution function approaching the equilibrium state.

The **Equilibrium distribution function** can be presented as follows:

$$f_i^{eq}=\rho\mathbf{w_i}\left[1+3\mathbf{c_i}\cdot\mathbf{u}+\frac{9}{2}\left(\mathbf{c_i}\cdot\mathbf{u}\right)^2-\mathbf{u}\cdot\mathbf{u}\right]$$

Collision Function:

$$\tilde{f}_i\left(\mathbf{x},t\right)=f_i\left(\mathbf{x},t\right)-\frac{1}{\tau}\left(f_i\left(\mathbf{x},t\right)-f_i^{eq}\left(\mathbf{x},t\right)\right)$$

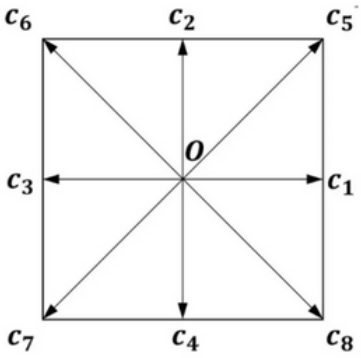

Fig. 3: D2Q9 Model

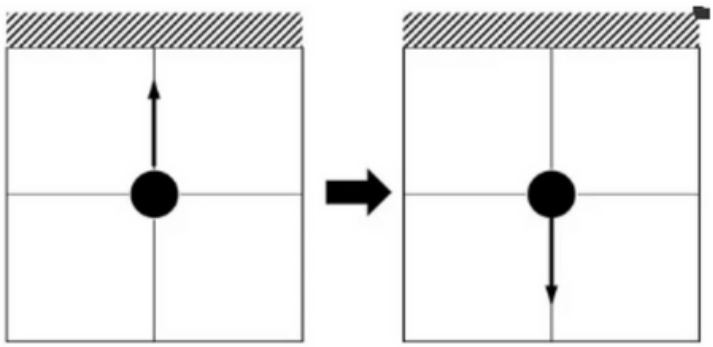
Fig. 4: Schematic diagram illustrating Bounceback condition

Streaming Function:

$$f_i\left(\mathbf{x}+\mathbf{c}_i\Delta t, t+\Delta t\right)=\bar{f}_i\left(\mathbf{x}, t\right)$$

$\overline{f_i}$ indicates the post-collision distribution function. In LBM, the relaxation time should be related to the viscosity of the fluid through the Chapman–Enskog expansion in such a manner that the microscopic variables computed from the LBM can satisfy the Navier–Stokes equation with second-order accuracy (Erturk et al. [8]). The relaxation time τ depends on several variables—the fluid kinematic viscosity ν, the lattice spacing Δx, the time step Δt, and the density of the fluid ρ, as follows:

$$\tau=\frac{1}{2}+\frac{\nu}{\rho\Delta t c_s^2}$$

where the lattice spacing Δx and time step Δt are set to a unit, in the LBM, there are several lattice models that can be used for the simulation. In the case of the two-dimension domain, the possible directions (i.e., ci) in which the particles move would comprise nine different nodes, including the original node (i.e., 0) (D2Q9 model). When a particle moves and contacts the no-slip wall, it then bounces back in the oncoming direction while retaining its own speed.

In addition, **wi** and **ci** in the D2Q9 lattice are represented as follows:

$$\mathbf{w_i}=\begin{cases}4/9 & i=0,\\ 1/9 & i=1,2,3,4,\\ 1/36 & i=5,6,7,8\end{cases}$$

$$\mathbf{c_i}=\begin{cases}0 & i=0,\\ c\left(\cos\left(\frac{(i-1)\pi}{4}\right),\sin\left(\frac{(i-1)\pi}{4}\right)\right. & i=1,2,3,4,\\ \sqrt{2}c\left(\cos\left(\frac{(i-1)\pi}{4}\right),\sin\left(\frac{(i-1)\pi}{4}\right)\right. & i=5,6,7,8\end{cases}$$

3. Implementation of Cython and OpenMP:

As Python is an Interpreted Language, its execution time is much more than languages alike C and JAVA. But as writing syntax in Python is much simpler, we preferred Python and now for increasing the efficiency of our code and for decreasing Runtime, we will be using Cython Language and for further reduction of runtime, we will incorporate OpenMP to parallelize LBM code for Parallel Computing.

Cython is an extension of Python that adds static typing to variables, functions, and classes. It combines the simplicity of Python and the efficiency of C. The Cython compiler takes the commands in a .pyx file and compiles it to generate C or C++ source code and this can be run using a Python script.

OpenMP (Open Multi-Processing) is an application programming interface (API) that supports multi-platform shared-memory multiprocessing programming in C, C++, and Fortran, on many platforms, instruction-set architectures, and operating systems, including Solaris, AIX, FreeBSD, HP-UX, Linux, macOS, and Windows.

Results:

FVM:

Using Finite Volume Method (FVM) in ANSYS Fluent, we can observe that there will be the formation of a vortex in the middle of the cavity and relatively smaller eddies at the corners. Here, the size of the vortex depends on Reynolds Number. As the value of the Reynolds number increases, the size of the vortex increases with it.

The results obtained from the simulation show excellent agreement with the existing theoretical and experimental literature which validates our correct approach towards FVM. The same has been shown below:

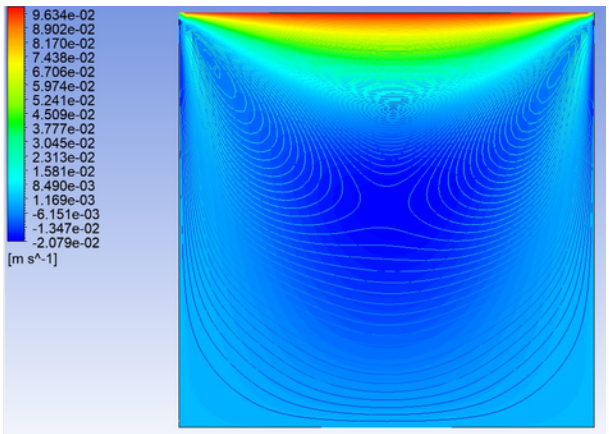

Fig. 5: FVM simulation from ANSYS

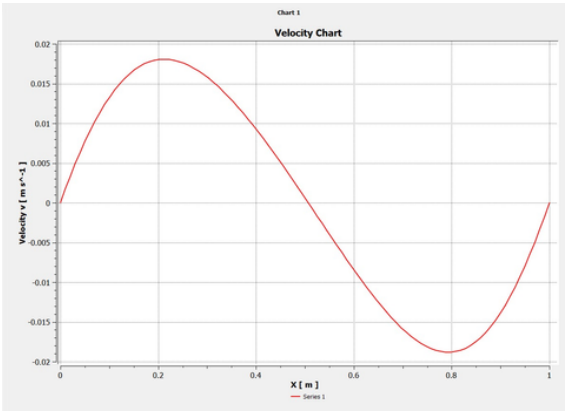

Fig. 6: Velocity distribution in the cavity

LBM:

The results obtained from Lattice Boltzmann Method (LBM) in Python resonate with those from the FVM method and existing literature. As discussed earlier, the code for LBM has been optimised using Cython and the results are extracted and illustrated using MATLAB to directly compare with the results from FVM approach. Fig. 7 and Fig. 8 show the simulation result and velocity distribution obtained from the Cython code. We can observe that the results are quite similar to that of FVM shown in Fig. 5 and Fig. 6, which indicates that the LBM is an efficient and accurate approach for simulating fluid flows, especially for complex geometries like lid-driven cavities.

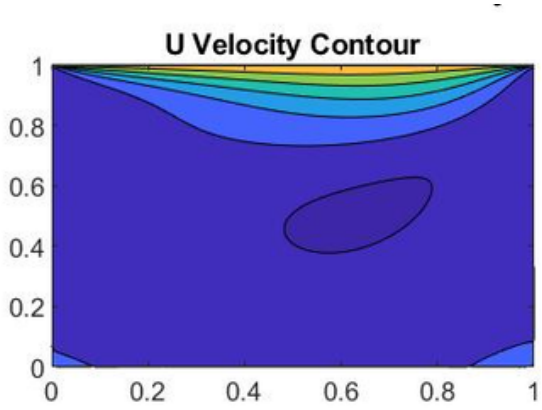

Fig. 7: LBM simulation from Cython

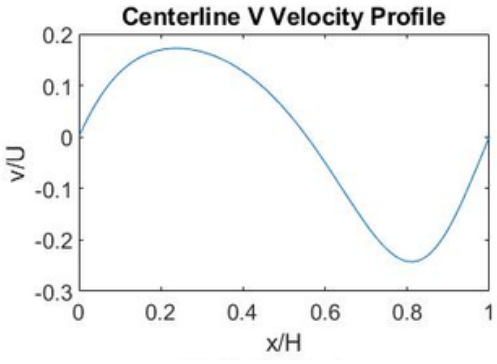

Fig. 8: Velocity distribution in the cavity

Also, the goal of attaining computational time speedup from Python to Cython was successful. There was a huge improvement in execution time for LBM code when the Python code was optimised using Cython as shown in Fig. 8. Even lower execution time was attained when OpenMP parallelisation was introduced for the Cython code. For a 30x30 Nodes, 2000 iterations configuration, there was an immense 76x speedup in execution time from Cython code as compared to Python, and a 86x speedup from Cython+OpenMP as compared to Python.

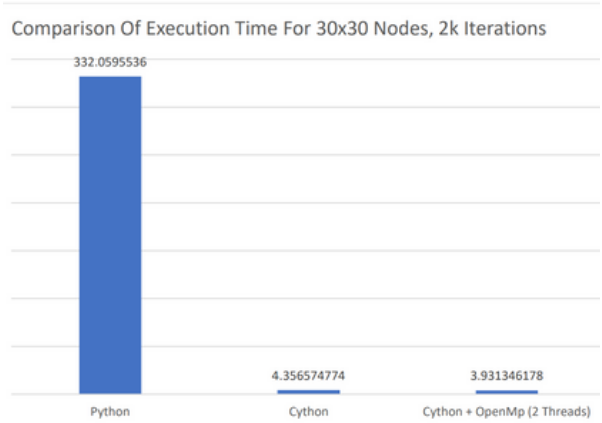

Fig. 9: Execution time comparison with different approaches

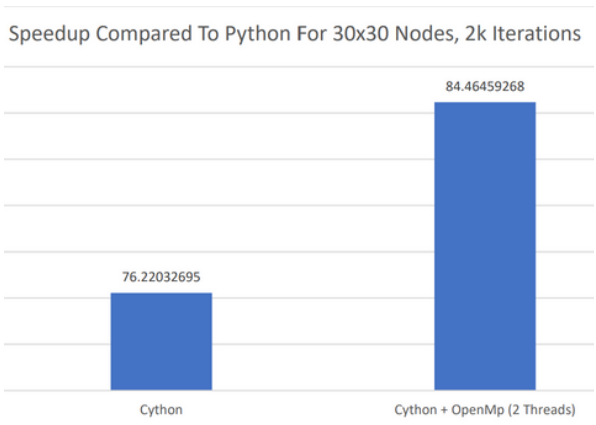

Fig. 10: Speedup compared to Python

Conclusion:

The implementation of the Lattice Boltzmann Method (LBM) using Python, Cython, and OpenMP for parallel computing has been successfully carried out in this project. We compared the results obtained from our LBM implementation with those from the FVM implementation and found that LBM performs comparably, with some advantages over FVM. Python and Cython have proven to be effective tools for implementing LBM due to their high-level syntax and ease of use. OpenMP has also been useful for parallelizing the LBM code, resulting in significant improvements in performance.

The project has also presented some challenges during implementation and optimization and provided insights into the performance characteristics of the LBM implementation. The optimization techniques used, such as loop unrolling and memory management, have been effective in improving the efficiency of the LBM implementation. Overall, this project provides valuable knowledge and skills in the field of computational fluid dynamics, programming, optimization, and problem-solving, which can be applied to various domains in science, engineering, and technology.

Future Work:

In the future, we plan to introduce obstacles of various shapes into the LBM simulations to simulate more complex fluid flow scenarios. The LBM implementation can also be extended to handle multiple fluids and multiphase flows, which are common in many engineering applications. Furthermore, the LBM can be combined with other simulation methods, such as molecular dynamics and continuum mechanics, to study phenomena at different scales.

We are also planning to explore the use of CUDA, a parallel computing platform developed by NVIDIA, as an alternative to OpenMP for further improving the performance of the LBM implementation. CUDA can significantly increase the speed of simulations by allowing the LBM code to run on graphics processing units (GPUs) instead of central processing units (CPUs). This approach can also handle much larger datasets, which is critical for simulating more complex geometries and scenarios. Additionally, we plan to optimize the LBM implementation for CUDA to achieve maximum performance gains

References:

- Lattice Boltzmann Method: Fundamentals and Engineering Applications with Computer Codes by A. A. Mohamad
- Application of Lattice Boltzmann Method for incompressible viscous flows by D. Arumuga Perumal, Anoop K. Dass
- ANSYS Simulations: A Hands-on Introduction to Engineering Simulations