
# High Performance Computing with Python

## Final Report

Ayisha Ryhana Dawood

5653447

dawooda@informatik.uni-freiburg.de

August 15, 2023

# Contents

# 1

# Introduction

The Lattice Boltzmann method provides a simulation technique that uses a density distribution to model fluid flow including turbulence, multi-component and multi-phase flows as well as additional applications [1]. The report starts by discussing the Boltzmann Transport Equation and the different components of the method that are important from an implementation perspective.

Fluid mechanics deals with the movements and interactions of fluids and are of high importance in many fields of study and industry. Experimentation with fluids on a large scale can be challenging. Simulation of experiments provides a solution to this problem. As the scale of experiments increase, the computational technologies used in the simulations need to be upgraded to handle the complexities.

High Performance Computing (HPC) is the use of advanced computational techniques like parallel processing, multi-core processors, GPU accelerations, cluster computing, super-computing and others to achieve efficiency and accuracy in solving complex problems. C and FORTRAN are popular programming language used to achieve HPC due to the ability to write highly optimized code with control over memory management and hardware interactions. Python, well known for its simplicity and wide use in AI/ML, is also being used for HPC through libraries like NumPy. This allows for high-level programming while still being able to interface libraries implemented in C and FORTRAN.

In this report, HPC concepts are realized using Python. The Boltzmann Transport Equation is simulated without HPC for different initial configurations and boundary conditions to validate the implementation incrementally. Finally, the Sliding Lid induced cavity is simulated using HPC elements like spatial domain decomposition (splitting the lattice grid into sub-domains to be processed independently) and the MPI (`mpi4py` package) for message passing between the sub-domains. The performance across implementations with multiple processor counts are observed. The simulation is repeated for various initial configurations and the performance is noted.

The results in this report can be reproduced by the code base and instructions of the associated GitHub repository [1]

[1]The code is available at: https://github.com/AyishaR/HPC-fluid-mechanics-with-python

# 2

# Lattice Boltzmann Method

The Lattice Boltzmann method is a computational technique to approximate fluid behaviour by simulating the trajectory of discrete particles and their interactions on a lattice grid. The approximation is done by breaking down the fluid flow into discretized steps - streaming (movement of the particles along the axes of the lattice), collision (interaction of particles with one another) and boundary conditions (behaviour of the particles at the boundaries of the lattice grid).

## 2.1 The Boltzmann Transport Equation (BTE)

The Boltzmann equations are discretizations of the kinetic equations of Lattice Gas Cellular Automaton (LGCA) [2].

$$\frac{\partial f}{\partial x} + \mathbf{v}\nabla f = Q \tag{2.1}$$

Here, Q is the collision integral that models complex particle interactions and scattering processes. The BGK approximation is an alternate low-level discretization of the original Boltzmann equation that pushes the system towards an equilibrium.

$$f_i(\mathbf{r} + \mathbf{c}_i \cdot \Delta t, t + \Delta t) - f_i(\mathbf{r}, t) = -\frac{f_i(\mathbf{r}, t) - f_i^{\mathrm{eq}}(\mathbf{r}, t)}{\tau} \tag{2.2}$$

$$\underbrace{f_i(\mathbf{r} + \mathbf{c}_i \cdot \Delta t, t + \Delta t) - f_i(\mathbf{r}, t)}_{streaming} = \underbrace{-\omega \left(f_i(\mathbf{r}, t) - f_i^{\mathrm{eq}}(\mathbf{r}, t)\right)}_{collision} \tag{2.3}$$

The collision term is calculated using the equilibrium state and the relaxation time constant $\tau$. The system is driven into global equilibrium at the rate $\Omega$. This simplified model does not take into consideration the external forces on the system as the inter-atomic processes being modelled occur instantaneously compared to the changes induced by external forces. The

density and velocity of the fluid characterized by the particle probability density $f(\mathbf{r}, \mathbf{v}, t)$ is given as follows,

$$\rho(\mathbf{r}, t) = \int f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v} \tag{2.4}$$

$$\mathbf{u}(\mathbf{r}, t) = \frac{1}{\rho(\mathbf{r}, t)} \int \mathbf{v} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v} \tag{2.5}$$

The BTE can now be used to simulate fluid flow and calculate the state of the system at time step $t + \Delta t$ given the state at time $t$.

$$f_i(\mathbf{r} + \mathbf{c}_i \cdot \Delta t, t + \Delta t) = f_i(\mathbf{r}, t) - \omega \left(f_i(\mathbf{r}, t) - f_i^{\text{eq}}(\mathbf{r}, t)\right) \tag{2.6}$$

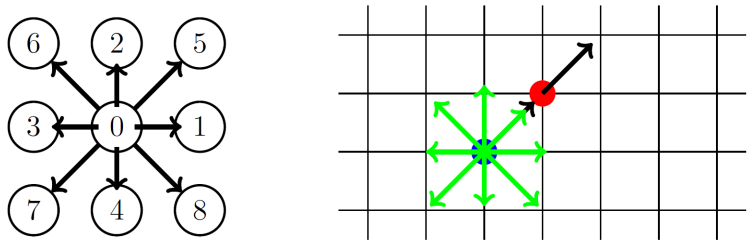
## 2.2 Lattice Grid

Lattice grids represent the discretized domain where the above equations are used to simulate fluid flow. The lattice structures are described by the notation

**DdQq**

where the character D indicates dimensionality and Q indicates velocity directions. d and q are integers that represent the number of dimensions and number of discrete velocity directions respectively.

Higher values of d and q lead to computationally expensive simulations while lower dimensions and fewer vector directions affect accuracy of the simulation. In this report, the D2Q9 lattice is used, i.e., 2D fluid flows are simulated with 9 discrete velocity vectors as a balance between accuracy and computational efficiency.




(a) (b)

Figure 2.1: (a) 9 discrete velocity directions. (b) D2Q9 lattice grid showing velocity vectors of one particle. [3]

## 2.3 Streaming

The discretization of velocity space results in velocity sets that define movements along the prescribed directions. Streaming is the shift of lattice velocity in these directions for $\Delta t$.

For a node $\mathbf{r}$ in a D2Q9 lattice, the velocity sets are described by a $9 \times 2$ $(q \times d)$ matrix as follows,

$$c = \begin{pmatrix} 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \end{pmatrix}^T \tag{2.7}$$

When $\Delta x$, $\Delta y$ and $\Delta t$ are unity, the discretized probability density function $f_i$ of node $\mathbf{r}$ will be shifted along the lattice with velocity $c_i$ to the immediate adjacent node in the specified directions.

## 2.4 Collision

Collision describes the interaction between particles on the lattice grid. It is discretized by a relaxation time approximation that implies that system locally relaxes towards an equilibrium distribution $f^{eq}$.

The equilibrium state depends on the local variables - the local density $\rho(\mathbf{r})$ and the local average velocity $\mathbf{u}(\mathbf{r})$.

$$\rho(\mathbf{r}) = \sum_i f_i \tag{2.8}$$

$$\mathbf{u}(\mathbf{r}) = \frac{1}{\rho(\mathbf{r})} \sum_i c_i f_i(\mathbf{r}) \tag{2.9}$$

$$f_i^{eq}(\rho(\mathbf{r}), \mathbf{u}(\mathbf{r})) = w_i \rho(\mathbf{r}) \left[ 1 + 3c_i \cdot \mathbf{u}(\mathbf{r}) + \frac{9}{2}(c_i \cdot \mathbf{u}(\mathbf{r}))^2 - \frac{3}{2}|\mathbf{u}(\mathbf{r})|^2 \right] \tag{2.10}$$

$w_i$ is a $1 \times 9$ $(1 \times q)$ matrix describing the weights of the particle interactions in the velocity directions.

$$w_i = (\frac{4}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{36}, \frac{1}{36}, \frac{1}{36}, \frac{1}{36}) \tag{2.11}$$

## 2.5 Boundary conditions

Boundary conditions define the behaviour of particles at the boundary nodes of the lattice grid. The sequence of execution with respect to streaming and collision operations is crucial. All boundary conditions discussed below, except pressure gradient are applied after the streaming step. These conditions

are applied to the particle probability density function $f_i$ and not the physical variables $\rho$ and $\mathbf{u}$.

The boundaries of the lattice grids can be implemented in two ways: [4]

1. Dry Node

   The boundaries are located on the link between 2 nodes, i.e., the node is part of the grid where there is no fluid pressure and act as an interface between surfaces depicted by nodes on either side.

2. Wet Node

   The boundaries are present on the lattice node, i.e., the node is located where the fluid is present.

In this implementation, dry nodes are used because of they are simpler to handle.

### 2.5.1 Periodic Boundary Conditions (PBC)

Periodic boundary conditions are used to implement symmetric conditions where the fluid that leaves the lattice grid through one boundary, re-enters through the opposite side during the streaming operation. In addition to the lattice shift during streaming, the following condition needs to be satisfied. [5]

$$f_i(\mathbf{x_0}, t) = f_i(\mathbf{x_{N-1}}, t) \tag{2.12}$$

where $x_0$ and $x_{N-1}$ are the first and last nodes in the physical domain of a lattice grid with N nodes. This is easily achieved using the `np.roll` function in the `numpy` package.

### 2.5.2 Rigid wall (Bounce back)

Bounce back conditions are implemented to simulate interaction of fluid particles with a rigid wall. The assumption is that the velocity vectors facing the wall perfectly bounce back ($180^0$) after streaming. For example, consider a wall on the left boundary of a lattice grid. The probability density values of channels facing the wall are $f_6, f_3, f_7$. After streaming, the channels $f_8, f_1, f_5$ are overwritten by the values in $f_6, f_3, f_7$.

For the generic implementation, $f^\star$ is the probability density before streaming, and $\bar{i}$ is the channel opposite to $i$. [5]

$$f_{\bar{i}}(\mathbf{x_b}, t + \Delta t) = f_i^\star(\mathbf{x_b}, t) \tag{2.13}$$

### 2.5.3 Moving wall

Moving walls along a fluid surface contribute to the momentum of the fluid. In addition to bounce back, the velocity of the wall $u_w$ is also incorporated to the equation. [5]

$$f_{\bar{i}}(\mathbf{x_b}, t + \Delta t) = f_i^{\star}(\mathbf{x_b}, t) - 2w_i\rho_w \frac{\mathbf{c_i} \cdot \mathbf{u_w}}{c_s^2} \tag{2.14}$$

Here, $c_s = \frac{1}{\sqrt{3}}$ is the speed of sound and $\rho_w$ is the density of the wall.

The Sliding Lid induced cavity simulation involves a moving wall on the top boundary of the lattice. To simplify calculation of channel populations $f_7, f_4, f_8$, the following implementation was used [6]. $\rho_N, u_N, v_N$ are the density, horizontal velocity and vertical velocity of the top boundary.

$$\begin{aligned}
\rho_N &= \frac{1}{1 + v_N}\left[f_0 + f_1 + f_3 + 2\left(f_2 + f_6 + f_5\right)\right] \\
f_4 &= f_2 - \frac{2}{3}\rho_N v_N \\
f_7 &= f_5 + \frac{1}{2}\left(f_1 - f_3\right) - \frac{1}{6}\rho_N v_N - \frac{1}{2}\rho_N u_N \\
f_8 &= f_6 + \frac{1}{2}\left(f_3 - f_1\right) - \frac{1}{6}\rho_N v_N + \frac{1}{2}\rho_N u_N
\end{aligned} \tag{2.15}$$

### 2.5.4 Pressure Gradient

A pressure variation of $\Delta p$ is applied between the inlet ($p_{in}$) and outlet ($p_{out}$). Pressure and density are related through the ideal gas equation of state.

$$p = c_s^2 \rho \tag{2.16}$$

with $c_s^2$=1/3 in lattice units.

The lattice grid is created with a buffer layer of one node. $x_0$ and $x_{N+1}$ are the buffer nodes for lattice with size N between inlet and outlet. The boundary conditions are applied before the streaming step. [5]

$$\begin{aligned}
f_i^{\star}(x_0, y, t) &= f_i^{eq}(\rho_{in}, \mathbf{u_N}) + \left(f_i^{\star}(x_N, y, t) - f_i^{eq}(x_N, y, t)\right) \\
f_i^{\star}(x_{N+1}, y, t) &= f_i^{eq}(\rho_{out}, \mathbf{u_N}) + \left(f_i^{\star}(x_1, y, t) - f_i^{eq}(x_1, y, t)\right)
\end{aligned} \tag{2.17}$$

# 3

# Implementation and Code Snippets

This section discusses the implementation details and few code snippets to explain them better.

## 3.1 Lattice grid visualization

The Lattice Bolztmann Method models particles on the lattice grid by discretization. Visualizing the results is an easy way to verify that the simulations are working as intended. This includes density plots, streamlines indicating the velocity, decay plots and others. The density plots in this report do not show the buffer layers. The x-axis is horizontal and the y-axis is vertical. The velocity vector directions are same as in Figure 2.1.

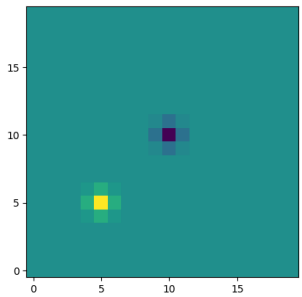



Figure 3.1: Density plot of a sample D2Q9 lattice grid of grid size 20x20 with one time step propagation and relaxation.

## 3.2 Code Elements

The `numpy` package is extensively used for all array data structures. The numpy array `f` with shape `(9,nx,ny)` stores the probability density across all channels, `rho` with shape `(nx,ny)` stores the density values and `u` with shape `(2,nx,ny)` stores the velocity values $u_x$ and $u_y$ for each node on the lattice grid.

## 3.3 Algorithmic Workflow

Every simulation follows a workflow that comprises a consistent set of components. It begins with parsing the command-line arguments to configure the parameters of the simulation. Each parameter has a default value configured that can be over-written by passing custom values through the command line. Some parameters that are common across all simulations include sizeof the grid (`-nx, -ny`), omega (`-o`), time steps (`-nt`) and the interval to log values for visualization or post-processing (`-nt_log`). Few simulation specific parameters can also be configured like inlet and outlet density (`-rho_in and -rho_out`) for Poiseuille Flow simulation. The arguments can also be used to trigger specific types of execution like comparison of fluid flow with different initial velocities or different omega values.

The next step is the simulation. It begins with streaming (propagation), followed by applying boundary conditions specific to the scenario. In case of pressure gradients, the boundary condition is applied before streaming. This is followed by collision (relaxation), which models the interaction between particles as the system locally relaxes towards equilibrium. [7]

The simulation is followed by plotting relevant graphs. The density of the lattice grid is most common. Flow specific plots like velocity profile or density variations are plotted and compared to analytical values where relevant.

## 3.4 Parallel implementation

The parallellization of LBM is done using spatial domain decomposition. The lattice grid is divided into $n$ non-overlapping sub-domains for independent processing in a computing unit or processor. The sub-domains are adjacent parts of the lattice grid and need to communicate the channel populations periodically. The final populations of the channels in each sub-domain also needs to be consolidated at the end of the simulation from the different processing units to be presented as a full lattice grid. This is achieved by the message passing interface (MPI).

The number of computing units `size` and the dimensions of the lattice grid $N_x \times N_y$ are used to calculate the number of sub-domains along each

**Algorithm 1** Lattice Boltzmann Method

f: $(9, N_x, N_y)$
$\rho$: $(N_x, N_y)$ Initial density $\rho_0$
u: $(2, N_x, N_y)$ Initial velocity $u_0$
c: $(9, 2)$
w: $(1, 9)$
nt: Number of timesteps

1: $\rho\,(0) = \rho_0$; $u\,(0) = u_0$
2: **for** $i = 1$ to $nt$ **do**
3: $f'(t+1) = streaming(f(t))$ Section 2.3
4: $f'(t+1) = boundary_handling(f'(t+1))$ Section 2.5
5: $f^{eq}(t) = equilibrium(\rho'(t+1), u'(t+1))$
6: $f(t+1) = f'(t+1) + \omega(f^{eq}(t+1) - f'(t+1))$ Section 2.4
7: **end for**

axis `sects_X` and `sects_Y`. These, in addition to the `rank` of the computing unit are used to calculate the coordinates of the sub-domain of lattice grid processed by it.

Each sub-domain has a buffer layer of size 1 that is communicated with sub-domains on all sides. This communication is done through the `Sendrecv` function in `mpi4py`. In one communication, data from the top non-buffer layer is sent to the last buffer layer of the sub-domain on top and similar data vector is received from the sub-domain below into the last buffer layer of the current sub-domain (Communication from Rank 0 to rank 1 in Figure 3.2). This way the data vectors are exchanged in pairs along each of the 4 directions - up, down, left and right.

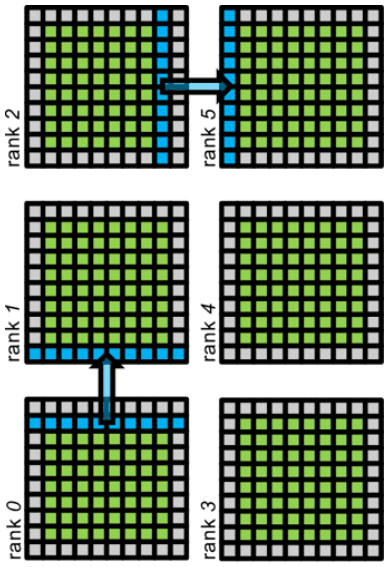



Figure 3.2: Spatial domain decomposition and visualization of the Message Passing Interface (MPI). The green cells are the physical domain and the grey cells are the buffer layer. [3]

# 4

# Simulation results

## 4.1 Validating LBM components

The realization of Sliding Lid induced cavity by parallelization starts with implementing and understanding the different elements of the Lattice Boltzmann method incrementally. This makes it easier to comprehend the concept and avoid errors in developing the simulation code.

### 4.1.1 Shear Wave Decay

Shear Wave Decay refers to the attenuation of the shear waves propagated through a material. The fluid system is initialized with a sinusoidal velocity or sinusoidal density and the behaviour of the system is monitored across time steps. The lattice grid has Periodic Boundary Conditions on all four boundaries.

Sinusoidal velocity is simulated with the following initial conditions for density and velocity.

$$\begin{aligned} \rho\left(\mathbf{r},0\right) &= 1 \\ \mathbf{u}\left(\mathbf{r},0\right) &= \begin{pmatrix} \varepsilon sin\left(\frac{2\pi y}{L_y}\right) \\ 0 \end{pmatrix} \end{aligned} \tag{4.1}$$

As $t \to \infty$, the velocity decays to 0 as the system reaches equilibrium.

Sinusoidal density is simulated with the following initial conditions for density and velocity.

$$\begin{aligned} \rho\left(\mathbf{r},0\right) &= \rho_0 + \varepsilon sin\left(\frac{2\pi x}{L_x}\right) \\ \mathbf{u}\left(\mathbf{r},0\right) &= 0 \end{aligned} \tag{4.2}$$

As $t \to \infty$, the density decays to 0 as the system reaches equilibrium. But unlike velocity, the density oscillates before eventually reaching 0.

The same is repeated for values of omega ranging from 0.1 to 1.5. In each simulation, the viscosity is measured in two ways. One is the simulated

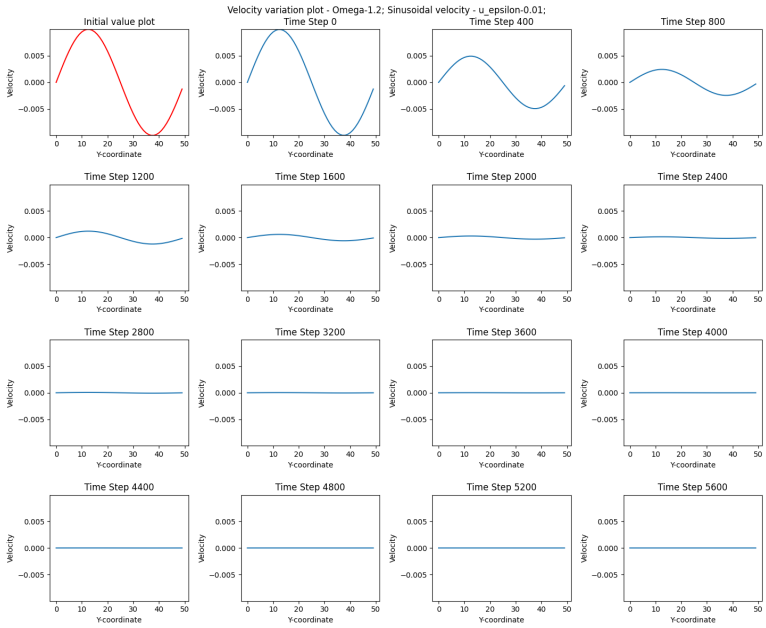
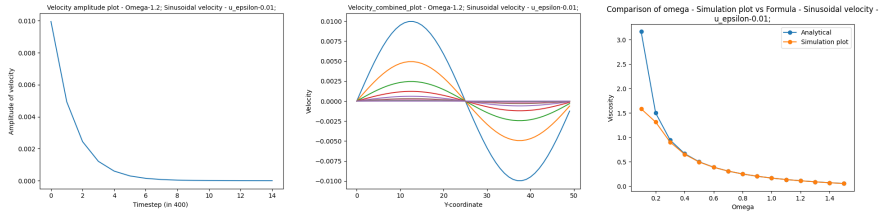



Figure 4.1: Time evolution of velocity at x=25 in lattice grid size 50x50 with configuration $\omega = 1.2, \varepsilon = 0.01$ and Periodic Boundary Conditions




(a) Amplitude of velocity along the Y-axis

(b) Time evolution of velocity

(c) Analytical vs simulated viscosity

Figure 4.2: Values for x=50 in lattice grid size 100x100 with configuration $\omega = 1.2$, Sinusoidal velocity with $\varepsilon = 0.01$ and Periodic Boundary Conditions

viscosity calculated from the velocity values in different time steps. Other is the measured viscosity calculated by assuming that the initial configuration fulfills the Stokes flow condition in Equation 4.3. This gives Equation 4.4 for analytical kinematic viscosity [7]. The plots in Figure 4.2c and Figure 4.4c compare the viscosity calculated by the two methods.

$$\frac{\partial \mathbf{u}}{\partial t} = \nu \Delta \mathbf{u} \tag{4.3}$$

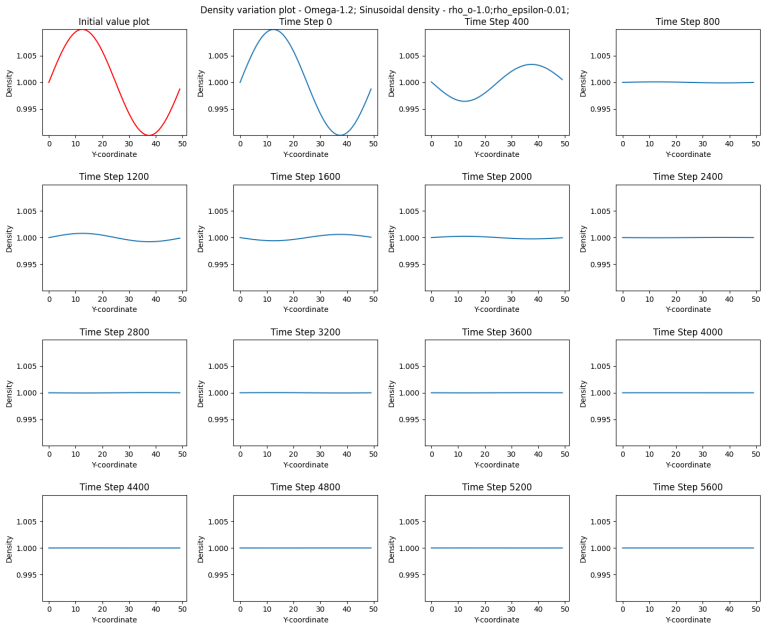
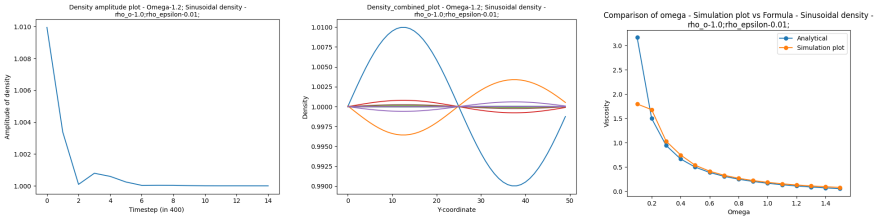



Figure 4.3: Time evolution of density at x=25 in lattice grid size 50x50 with configuration $\omega = 1.2, \rho_0 = 1$ and $\varepsilon = 0.01$ and Periodic Boundary Conditions

$$\nu = c_s^2 \left( \frac{1}{\omega} - \frac{1}{2} \right) \tag{4.4}$$

The two values of kinematic viscosity - analytical and calculated by the plot, are comparable for higher values of $\omega$. But for lower values, there is significant deviation between the two values implying that the simulation results may not be accurate. Thus, simulations use omega values where the calculated kinematic viscosity are almost equal for accurate representations.



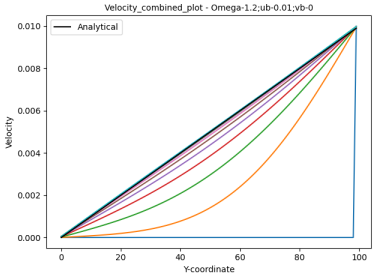

(a) Amplitude of density along the Y-axis (b) Time evolution of density (c) Analytical vs simulated viscosity

Figure 4.4: Values for x=50 in lattice grid size 100x100 with configuration $\omega = 1.2$, Sinusoidal density with $\varepsilon = 0.01$ and Periodic Boundary Conditions

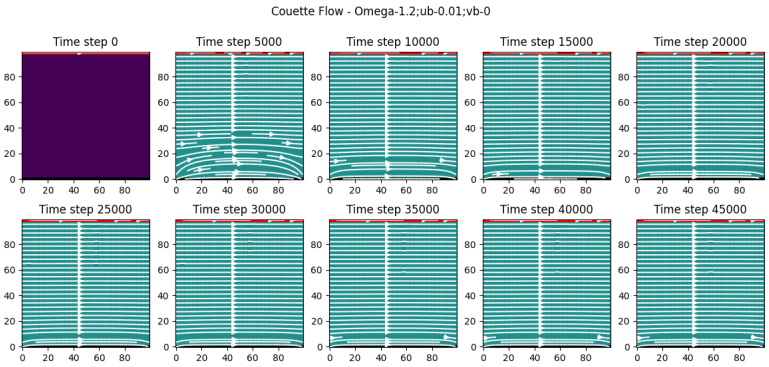
### 4.1.2 Couette Flow

Couette flow describes fluid flow between a moving wall and a rigid wall. This section emphasizes the understanding of rigid and moving walls in the Lattice Boltzmann method. The top boundary is modelled as a moving wall with velocity $u_x$ and the bottom boundary is a rigid wall. Positive values of $u_x$ indicate that the wall is moving to the right. Periodic Boundary Conditions are applied at the left and right boundaries. The initial configuration $(t = 0)$ is as follows,

$$\begin{aligned} \rho\,(0) &= 1.0 \\ \mathbf{u}\,(0) &= 0 \end{aligned} \tag{4.5}$$




(a) Time evolution of velocity along the Y-axis.




(b) Time evolution of density plot with velocity streamlines.

Figure 4.5: Couette Flow in lattice grid size 100x100 with configuration $\omega = 1.2$, Moving wall on top boundary (Colour red) with $u_x$=0.01, Rigid wall on bottom boundary (Colour black) and Periodic Boundary Conditions on the left and right boundaries.

At $t \to \infty$, the velocity profile becomes stable, maintaining a consistent pattern. The fluid layers on top move with velocity close to that of the moving wall, while the fluid layers at the bottom are close to stagnant.

Figure 4.5a shows the velocity profile evolving from the initial configuration to the stable state.

The analytical solution for the velocity can be calculated as follow, [8]

$$u\left(y\right) = \frac{y}{L_y} u_x \tag{4.6}$$

where $L_y$ is the length of the grid along Y-axis. This velocity is plotted against the simulation values in Figure 4.5a.

### 4.1.3 Poiseuille Flow

Poiseuille Flow models the fluid flow in a pipe characterized by pressure difference between the inlet and outlet. The top and bottom boundary are rigid walls with bounce back boundary conditions and the left and right boundaries have Periodic Boundary Conditions. Pressure gradients are implemented as a difference in density according to Equation 2.16. The initial configuration is the same as Equation 4.5. In addition $\rho_{in}$ and $\rho_{out}$ are defined such that $\Delta\rho = \rho_{in} - \rho_{out}$ is of the order $10^{-3}$. Here, $\rho_{in} = 1.001$, and $\rho_{out} = 0.999$. This implementation is to experiment with pressure gradient based boundary conditions.

As $t \to \infty$, the velocity profile becomes stable, with the center of the pipe (Can be imagined as a line passing through the pipe equidistant from the rigid walls) having the maximum velocity and the fluid close to the rigid walls of the pipe having close to zero velocity.

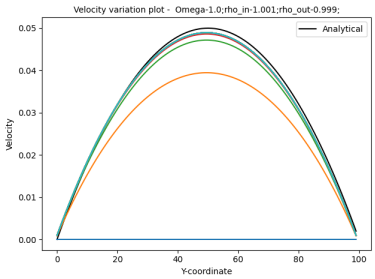
The velocity profile can also be calculated using the Navier-Stokes equation under the assumption of laminar flow. [9] Streaming velocity only has an X-component that is aligned with the axis of the pipe (Equation 4.7. Integrating along the wall normal direction and using boundary conditions $u(0) = 0$ and $u(h) = 0$, the velocity profile is given in Equation

$$\frac{\partial p\left(x\right)}{\partial t} = \frac{1}{\mu} \frac{\partial^2 u_x\left(y\right)}{\partial y^2} \tag{4.7}$$

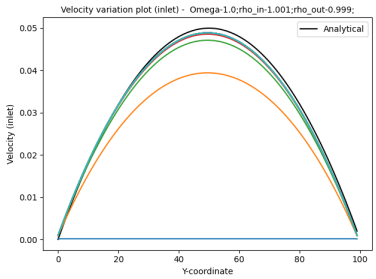
$$u\left(y\right) = -\frac{1}{2\mu} \frac{dp}{dx} y\left(h - y\right) \tag{4.8}$$

where $h$ is the diameter of the pipe (here, $ny$) and dynamic viscosity $\mu = \rho\nu$. The partial derivative is replaced by a total derivative as pressure only depends on the x-coordinate.

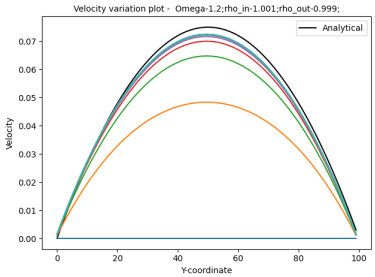
This analytical velocity is plotted against the time evolution of velocity through simulation. Figure 4.6a compares it to velocity profile generated at the midpoint of the length of the pipe ($X = \frac{nx}{2}$) and Figure 4.6b compares it to the velocity profile generated at the inlet ($X = 0$). The velocity profile at the inlet and midpoint of the pipe length are visually identical. This correlates the laminar flow assumption. The plots also show that the stable velocity profile is closer to the analytical value as $\omega$ decreases.



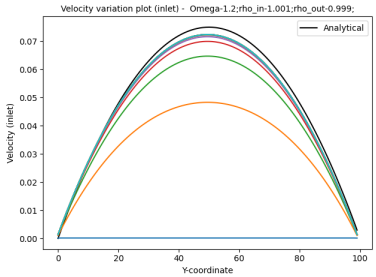

(a) Time evolution of velocity along the Y-axis at X=50 ($\omega = 1.0$)




(b) Time evolution of velocity along the Y-axis at X=0 (Inlet) ($\omega = 1.0$)




(c) Time evolution of velocity along the Y-axis at X=50 ($\omega = 1.2$)




(d) Time evolution of velocity along the Y-axis at X=0 (Inlet) ($\omega = 1.2$)

Figure 4.6: Poiseuille Flow in lattice grid size 100x100 with configuration Rigid wall on top and bottom boundary (Colour black) and Pressure gradient on the left and right boundaries with $\rho_{in} = 1.001$ at the inlet and $\rho_{out} = 0.999$ at the outlet.

## 4.2 Sliding Lid

Sliding Lid induced cavity is modelled by an enclosed fluid with a moving lid. The top boundary is a moving wall with velocity $u_x$ and the other three boundaries are rigid walls with bounce back conditions. The initial configuration is the same as Equation 4.5. In addition, the velocity of the moving wall is given by $u_x$ and positive values of $u_x$ indicate that the wall is moving to the right.

As $t \to \infty$, the vortexes formed stabilize and are different based on the initial configuration of the simulation.

## 4.3 Flow patterns analysis with different Reynolds number

The sliding lid on an enclosed fluid induces vortexes in the fluid that are characterized by a dimensionless entity called the **Reynolds Number (Re)**.

The Reynolds number (Re) is a dimensionless quantity that characterizes

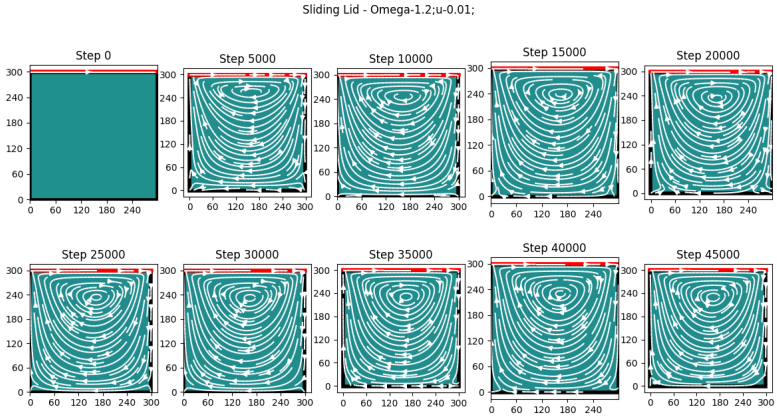



Figure 4.7: Sliding Lid - Time evolution of density at x=150 in lattice grid size 300x300 with configuration $\omega = 1.2$, Moving wall at the top boundary (Colour Red) with $u_x$=0.01 and Rigid walls with bounce back (Colour Black) on left, right and bottom boundaries. Re = 27

the flow behavior of a fluid. It depends on the viscosity, density, velocity, and characteristic length of the enclosure. A higher value of Re (> 2000) indicates fluid flow shifting towards turbulence. [10]

$$Re = \frac{Lu}{\nu} \tag{4.9}$$

$L$ is the length of the moving wall, $u$ is the velocity and $\nu$ is the kinematic viscosity that can be calculated from $\omega$ by Equation 4.4. $Lu$ represent the inertial terms and $\nu$ represents the viscous terms.

In this section, the sliding lid induced cavity is simulated for Re ranging from 100 to 1000 by varying different parameters of the initial configuration.

### 4.3.1 Varying velocity

The velocity of the moving wall on the top boundary is varied to achieve Re from 100 to 1000. The length and $\omega$ (and thus viscosity) are constant. The lattice grid of size $1000 \times 1000$ is initialized with $\omega = 1.2$ and $u_x$ varying from 0.01 to 0.11. The simulations are run for 50000 time steps.

### 4.3.2 Varying omega

The omega value determines the kinematic viscosity by Equation 4.4. The omega is varied between 0.5 and 1.54 to achieve Re between 100 and 1000. The lattice grid of size $1000 \times 1000$ is initialized with $u_x = 0.5$ and varying omega values. The simulations are run for 50000 time steps.

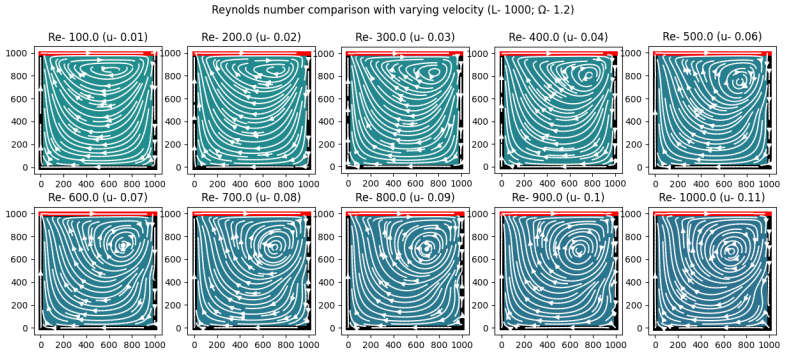



Figure 4.8: Sliding Lid - Snapshot of density of lattice grid size 1000x1000 with configuration $\omega = 1.2$, Moving wall at the top boundary (Colour Red) with $u_x$ varying from 0.01 to 0.11 (thus generating Re from 100 to 1000), and Rigid walls with bounce back (Colour Black) on left, right and bottom boundaries.

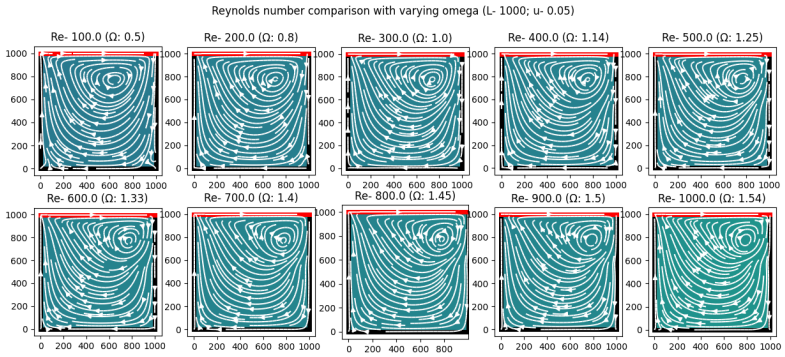



Figure 4.9: Sliding Lid - Snapshot of density of lattice grid size 1000x1000 with configuration $\omega$ varying from 0.5 to 1.54 (thus generating Re from 100 to 1000), Moving wall at the top boundary (Colour Red) with $u_x = 0.05$, and Rigid walls with bounce back (Colour Black) on left, right and bottom boundaries.

## 4.4 Scaling with processors

In this section, the performance of the parallelized simulation is observed by running on different number of processors. For each execution, the time of execution is measured and MLUPS is calculated. MLUPS (Million Loops Per Second) is a measure of the computational efficiency in loop-intensive operations.

$$MLUPS = \frac{TotalLoops}{ExecutionTime * 10^6} \tag{4.10}$$

The sliding lid induced cavity is simulated with three grid sizes $100 \times 100$, $500 \times 500$, $1000 \times 1000$. They are initialized with $\omega = 1.2$ and moving wall on the top boundary with $u_x = 0.01$ while the three other boundaries are rigid walls with bounce back conditions. The number of processors range from 1 to 2500 in uneven intervals to accommodate splitting between the grid sizes.

All the scaling test experiments are performed on bwUniCluster [1].

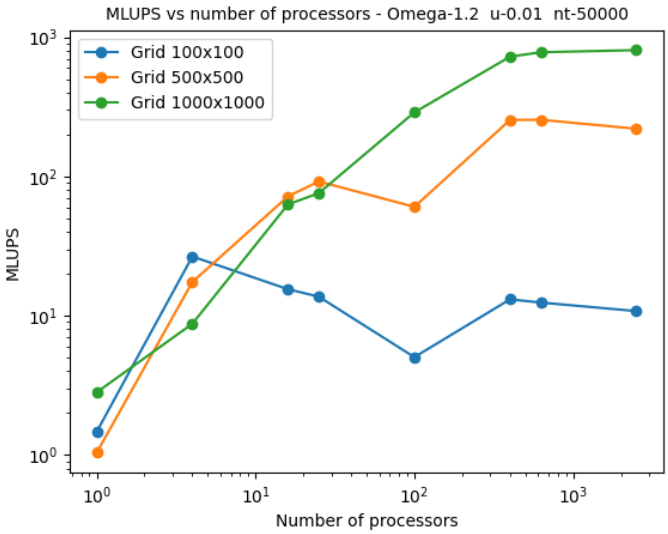



Figure 4.10: Sliding Lid induced cavity - Plot of MLUPS vs Number of processors for different grid sizes with configuration $\omega = 1.2$, Moving wall at the top boundary (Colour Red) with $u_x = 0.01$, and Rigid walls with bounce back (Colour Black) on left, right and bottom boundaries simulated for 50000 time steps.

The optimal number of processors for execution scales with the size of the lattice grid. Higher processor counts work well for larger grids. Performance degrades as the number of processors increases due to the latency of the communication and time lag in synchronization between the sub-domains. This is in line with Amdahl's Law.

[1]The authors acknowledge support by the state of Baden-Württemberg through bwHPC. `https://wiki.bwhpc.de/e/Category:BwUniCluster_2.0`

# 5

# Conclusion

The report starts by introducing the Lattice Boltzmann Method and motivating the need for High Performance Computing in the field of fluid dynamics. The idea of Python for HPC is also discussed.

Chapter 2 talks about the theoretical aspects of the Lattice Boltzmann Method and discretized equations for the same. The different components of the method including the mathematical elements required when simulating are explained. Multiple boundary condition options like Periodic Boundary Conditions, Rigid wall, Moving wall and Pressure gradients are introduced.

Chapter 3 discusses the implementation details including visualization techniques, packages and overall workflow of each of the simulations. The parallelization approach using spatial domain decomposition and MPI are explained.

Chapter 4 delves into the results for the simulation. The experiments are performed incrementally, starting from Shear Wave Decay with only PBC, followed by Couette flow with one moving and one rigid wall. Pressure gradient boundaries are also tested with Poiseuille Flow to deepen the understanding of the LBM. Finally, the Sliding Lid induced cavity is implemented in serial. With confidence in the accuracy of the simulation code, the experiment is parallelized and run on bwUniCluster [1] for different configurations to observe the flow pattern variations. Scaling tests were also done across a range of processor counts to assess performance. The optimal number of processors for execution scales with the size of the lattice grid but the performance degrades as the number of processors increases in line with Amdahl's Law due to communication overhead.

[1]The authors acknowledge support by the state of Baden-Württemberg through bwHPC. `https://wiki.bwhpc.de/e/Category:BwUniCluster_2.0`

# Bibliography


[1] Wagner Alexander J (NDSU Department of Physics). A practical introduction to the lattice boltzmann method. https://www.ndsu.edu/fileadmin/physics.ndsu.edu/Wagner/LBbook.pdf.

[2] Dieter A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models*. Springer Berlin, Heidelberg, 2004.

[3] Lars Pastewka and Andreas Greiner. Hpc with python: An mpi-parallel implementation of the lattice boltzmann method. 2019.

[4] H. Liu and J. G. Zhou. Lattice boltzmann approach to simulating a wetting–drying front in shallow flows. *Journal of Fluid Mechanics*, 743:32–59, 2014.

[5] Sauro Succi. *The Lattice Boltzmann Equation: For Complex States of Flowing Matter*. Oxford University Press, 04 2018.

[6] A. A. Mohamad. *Boundary Conditions*, pages 47–51. Springer London, London, 2019.

[7] Timm Krueger, Halim Kusumaatmaja, Alexander Kuzmin, Orest Shardt, Gonçalo Silva, and Erlend Magnus Viggen. The lattice boltzmann method. 2017.

[8] P. Nagy-György and Csaba Hős. A graphical technique for solving the couette-poiseuille problem for generalized newtonian fluids. *Periodica Polytechnica Chemical Engineering*, 63:200–209, 2018.

[9] A. A. Mendiburu, L. R. Carrocci, and João Andrade de Carvalho. Analytical solution for transient one- dimensional couette flow considering constant and time-dependent pressure gradients. 2009.

[10] T. P. Chiang, Wen-Hann Sheu, and Robert R. Hwang. Effect of reynolds number on the eddy structure in a lid-driven cavity. *International Journal for Numerical Methods in Fluids*, 26:557–579, 1998.