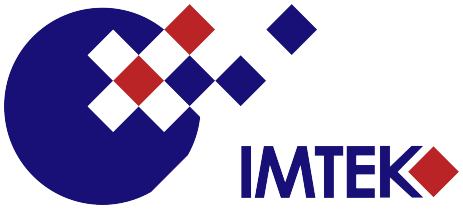
Department of Microsystems Engineering - IMTEK

High-Performance Computing with Python - SS2019

Implementation of the Lattice Boltzmann Method

Nima Riahi Dehkordi
-riahiden@tf.uni-freiburg.de-



Contents

Listings

1 Introduction

Boltzmann Transport Equation (BTE) which was developed by Ludwig Eduard Boltzmann (1844– 1906), statistically describes and predicts how the microscopic properties of particles define the macroscopic properties of matter such as the viscosity, thermal conductivity, etc. This is done by the means of streaming and scattering processes. [1, 6]
Lattice Boltzmann method (LBM) was introduced by McNamara and Zanetti in 1998 is a method based on a discretization of BTE which divides the space into equidistant grid cells for simulating fluid dynamics problems. Traditional approaches, utilize Navier–Stokes equations (NS) to solve mass, momentum, and energy conservation equations on discrete nodes, elements, or volumes to convert the nonlinear partial differential equations into a set of nonlinear algebraic equations, which are solved iteratively. Unlike traditional approaches, in LBM, the fluid is replaced by fractious particles. These particles stream along with given directions and collide at the lattice sites. Since the collision and streaming processes are local, LBM can be parallelized very well. This method is considered as one of the simplest microscopic approaches to model macroscopic dynamics. [2, 4, 6]

2 Methods and Theory

2.1 Boltzmann Transport Equation

A system can be described statistically by probability density function $f(r, c, t)$. it represents the number of molecules which are positioned between r and $r + dr$ with velocities between c and $c + dc$ at time t. An external force F can change the velocity of a gas molecule of unit mass from c to $c + Fdt$ and its position from r ro $r + cdt$. If there is no collision between the molecules it can be stated that the number of molecules before application of external force is equal to the number of molecules after application of external force. consequently

$$f(r + cdt, c + Fdt, t + dt)drdc - f(r, c, t)drdc = 0 \tag{2.1}$$

nevertheless, if collisions take place between the molecules there will be a difference between the numbers of molecules in the interval $drdc$. Hence

$$f(r + cdt, c + Fdt, t + dt)drdc - f(r, c, t)drdc = \Omega(f)drdcdt \tag{2.2}$$

This rate of change between final and initial status of the distribution function in equation 2.2 is called the collision operator Ω. Considering the moments of this probability density

$$\rho(r, t) = m \int f(r, c, t)dc \tag{2.3}$$

$$\rho(r, t)u(r, t) = m \int f(r, c, t)cdc \tag{2.4}$$

$$\rho(r,t)e(r,t) = \frac{1}{2}m\int u_a^2 f(r,c,t)dc \tag{2.5}$$

where m is molecular mass and u_a the particle velocity relative to fluid velocity $u_a = c - u$. Equations 2.3, 2.4 and 2.5 are conservation of mass, momentum and energy respectively. Since the collision term is complicated, it is possible to approximate the collision operator with a simple operator without introducing significant error to the outcome of the solution. The collision term is replaced by

$$\Omega = \omega(f^{eq} - f) = \frac{1}{\tau}(f^{eq} - f) \tag{2.6}$$

This replacement was introduced by Bhatnagar, Gross and Krook (BGK) in 1954. In equation 2.6 which is called BGK-Boltzmann equation, ω is called the collision frequency and τ is called relaxation factor. It is Assumed that f locally relaxes toward f^{eq} with a relaxation factor τ. [5, 6]
Given this approximation the BTE can be approximated as

$$\frac{\partial f}{\partial t} + c \bigtriangledown f = \frac{1}{\tau}(f^{eq} - f) \tag{2.7}$$

2.2 Lattice Boltzman Method

LBM is the discretization of BTE with respect to the dimensions of the problem and velocity directions. The common terminology is DnQm with n representing the dimensions of the problem and m the velocity directions. In this project, the D2Q9 discretization model is deployed and illustrated in Figure 2.

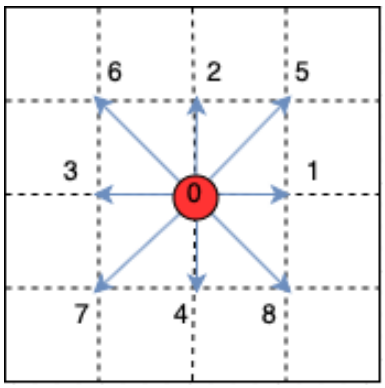


Figure 1: Lattice arrangements for D2Q9 discretization

Therefore, with i valid for specific velocity direction in Equation 2.8 the discretized version of

approximated BTE is

$$\frac{\partial f_i}{\partial t} + c_i \bigtriangledown f_i = \frac{1}{\tau}(f_i^{eq} - f_i) \tag{2.8}$$

Equation 2.8 can be descritized as

$$f_i(r + c_i\Delta t, t + \Delta t) = f_i(r,t) + \frac{\Delta t}{\tau}[f_i^{eq}(r,t) - f_i(r,t)] \tag{2.9}$$

Equation 2.9 can be applied to many physical phenomena by simply specifying a different equilibrium distribution function and source term (external force). [6]
In this project the expression for the equilibrium distribution function in the nine velocity directions is [5]

$$f_i^{eq}(r,t) = w_i\rho(r,t)\{1 + 3c_i.u(r,t) + \frac{9}{2}[c_i.u(r,t)]^2 - \frac{3}{2}u^2(r,t)\}; w_i = \begin{cases} \frac{4}{9} & i = 0 \\ \frac{1}{9} & i = 1,2,3,4 \\ \frac{1}{32} & i = 5,6,7,8 \end{cases} \tag{2.10}$$

2.3 Reynolds Number

Reynolds number (Re) is a dimensionless number which plays an important role in predicting the patterns in a fluid's behavior. Re, is used to determine whether the fluid flow is laminar or turbulent. The flow is considered to be in a laminar regime for Re up to $Re = 2300$. [3]
To calculate the Reynolds Number (Re) from a given kinematic viscosity

$$Re = \frac{u_0.L}{\nu} \tag{2.11}$$

where:

- L is the characteristic linear dimension
- u_0 is the velocity of fluid w.r.t. the wall
- ν is the kinematic viscosity of fluid

In this project the kinematic viscosity (ν), can be calculated analytically for a given collision frequency (ω) using equation 2.12

$$\nu = \frac{1}{3}(\frac{1}{\omega} - \frac{1}{2}) \tag{2.12}$$

3 Implementation

To implement LBM a discretization of probability density function $f(r, v, t)$ and density$\rho(r)$ is required. Since in this project the D2Q9 discretization model is used r can be interpreted as (x, y) or (length , width). For probability density function velocity sets also add another dimension to the matrix hence the f matrix is a 3-dimentional matrix. An illustration of $f(r, v, t)$ and $\rho(r)$ can be seen in Figure 2.

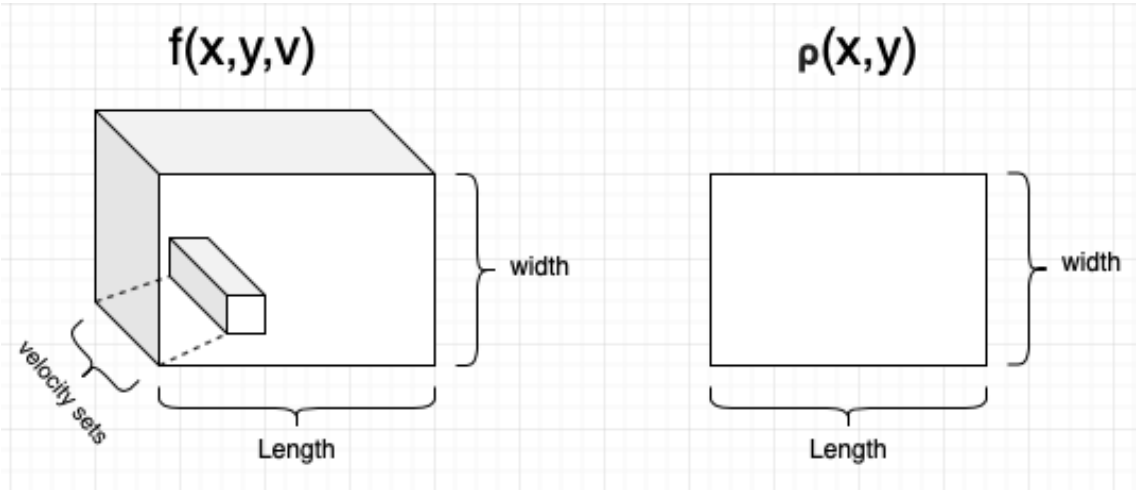


Figure 2: Visualisation of f and rho

In Equation 2.9 the l.h.s and r.h.s of the equation are called streaming and scattering term respectively.
The code is implemented in Python using *Numpy* [1] library.

3.1 Streaming term

The state of the simulation at each point of time is stored in probability density function f. vector c is a decomposition of velocities in the 2D cartesian grid. Values of c can be observed in Code Listing 1 with first value being x direction and second value the y direction.

Listing 1: Velocity direction decomposition

```
#velocity vector directions based on D2Q9
c = np.array([[ 0,  0],[ 1,  0],[ 0,  1],[-1,  0],[ 0, -1],[ 1,  1],[-1,  1],[-1, -1],[ 1,
    -1]])
```

The streaming term of the Equation 2.9 is implemented in Code Listing 2 using Numpy library, roll function which takes care of the periodic boundry conditions.

[1] https://www.numpy.org

Listing 2: Streaming Term

```
def streaming(f):
    #stream
    for i in range(q):
        f[:,:,i] = np.roll(np.roll(f[:,:,i], c[i,0], axis=0), c[i,1], axis=1)
    return f
```

3.2 Scattering term

Computation of scattering term in Python can be seen in Code Listing 4 which has 2 parts. First part computes the equilibrium distribution function with the function of same name. And second part computes the collision where omega or ω is the collision frequency.

Listing 3: Weight Vector

```
# weight values for each direction
w_i = np.array([4/9, 1/9, 1/9, 1/9, 1/9, 1/9, 1/36, 1/36, 1/36, 1/36])
```

Listing 4: Scattering Term

```
def equilibrium(rho, u):
    cu = np.dot(u,c.T)
    cu2 = cu ** 2
    u2 = u[:,:,0] ** 2 + u[:,:,1] ** 2
    return ((1 + 3*(cu.T) + 9/2*(cu2.T) - 3/2*(u2.T)) * rho.T ).T * w_i

def collision(f, omega):
    rho = f_rho(f)
    u = f_vel(f)
    #to compute the new f, first we need to compute feq
    feq = equilibrium(rho, u)
    #relaxation
    f = f + omega * (feq - f)
    return f, rho, u
```

3.3 Parallelization

The strategy for parallelization in this project is spatial domain decomposition of the probability density function. The collision part of Equation 2.9 can be highly parallelized. This

parallelization is acheived by means of Message Passing Interface (MPI)[2]. This scheme of parallelization requires a communication during streaming step. Figure 3 depicts the concept of spatial domain decomposition.

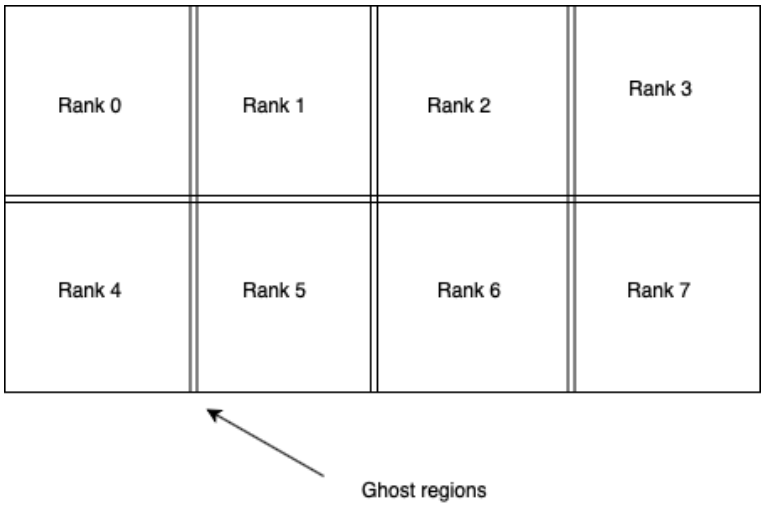


Figure 3: Parallelization strategy

As can be seen in Figure 3, the communication between different domains is done using ghost regions. Hence it is required to have four steps of communication. These four steps are communication to the right, left, top and bottom. Moreover, ghost regions are not required at the borders since there exist no adjacent domain for them. [5]

4 Results

4.1 Shear Wave Decay

Objective of this section is to analyse the behavior of a wave like density with time. The chosen lattice is 50*50.

4.1.1 Density Variation

In this section the initial density distribution is chosen as $\rho(r,0) = \rho_0 + \epsilon sin(\frac{2\pi x}{L_x})$ and to fulfill the stokes flow condition, ρ_0, ϵ, and ω have the following values respectively: 0.2, 0.01 and 0.4. Initial density profile is depicted in Figure 4.

[2] https://www.mpi-forum.org/mpi-40/

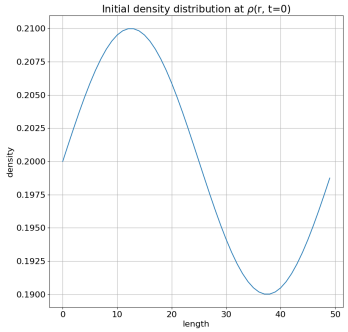


Figure 4: Initial Density Distribution

- **Density Evolution:** As it can be observed in Figure 5 from the evolution of density profile, the distribution tends to become uniform across the lattice. Hence $\rho(r, t \to \infty) = \rho_0$ holds.

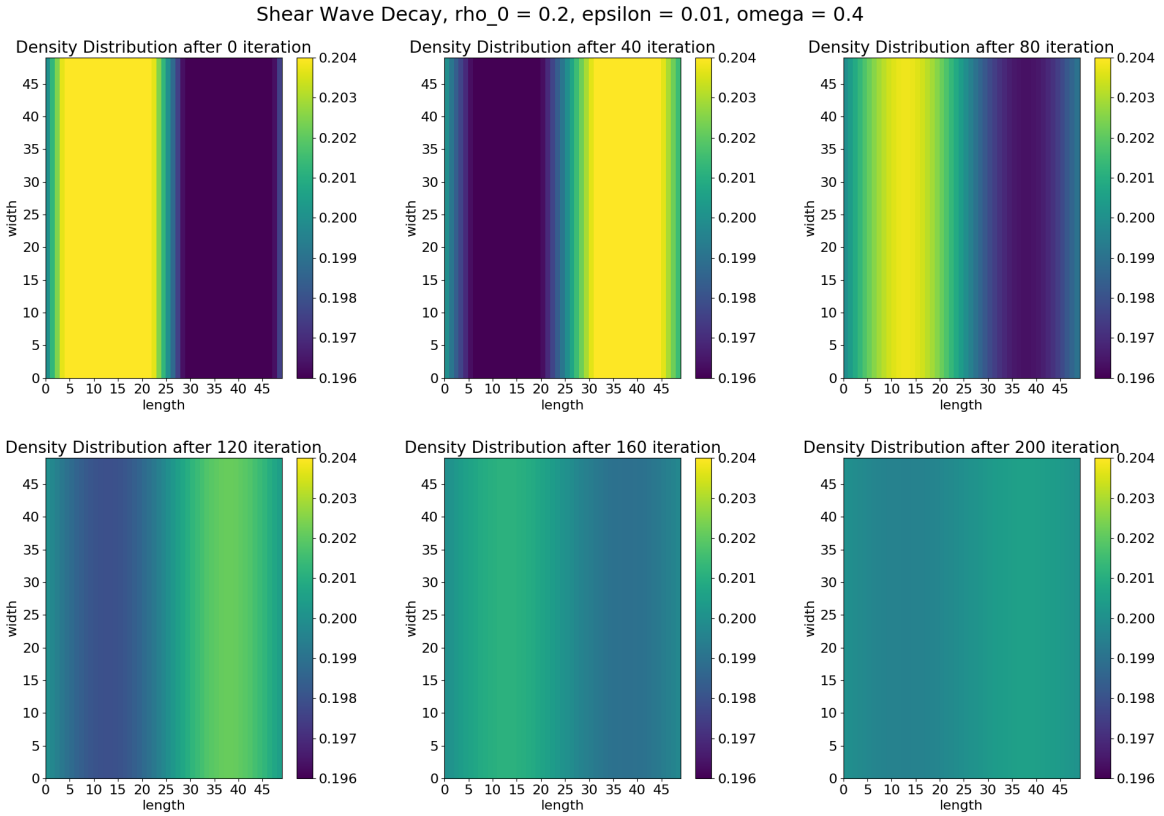


Figure 5: Evolution of density profile

- **Velocity Evolution:** As depicted in Figure 6, the velocity decays and tends toward zero as the simulation time goes further. $u_y(x, t \to \infty) = 0$

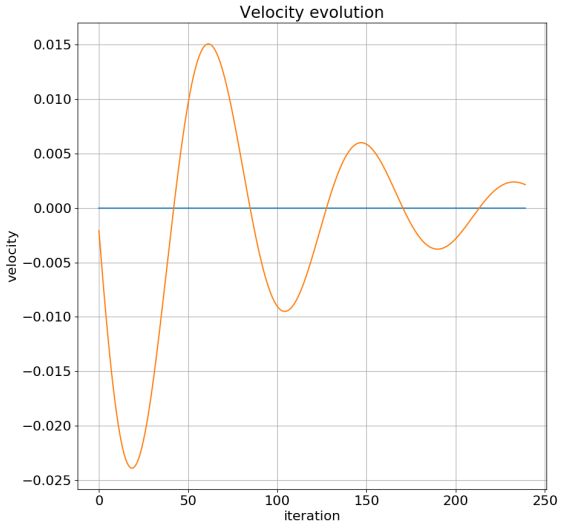


Figure 6: Evolution of velocity profile

4.1.2 Velocity Variation

Initial density is $\rho(r) = 1$ and the initial velocity in x direction is $u_x(r) = u_{x0} + \epsilon sin(\frac{2\pi y}{L_y})$ and $u_y(r) = 0$.
As depicted in Figure 7,it can be seen that the velocity converges to its initial value $u_x(r, t \to \infty) = u_0$.

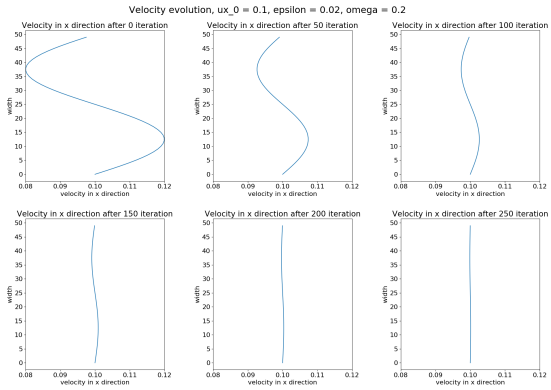


Figure 7: Evolution of velocity profile over time in x (length) direction

A plot of theoretical viscosity is illustrated in Figure 8

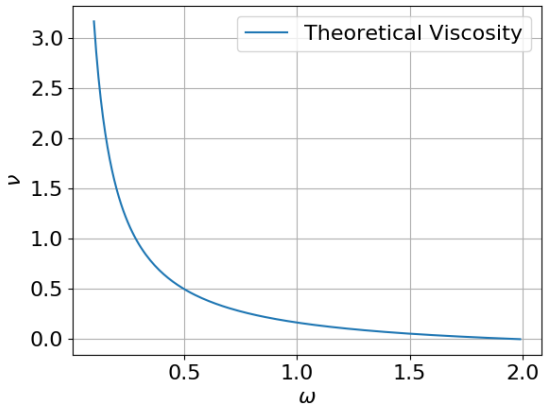


Figure 8: Theoretical Viscosity

4.2 Poiseuille flow

In this section a Poiseuille flow, which is movement of a fluid trapped between two walls, is simulated. The chosen lattice size is (50*50) where collision frequency $\omega = 1.3$ is selected. A constant uniform velocity of $u(x = 0, y) = 0.1$ prescribed at the inlet.

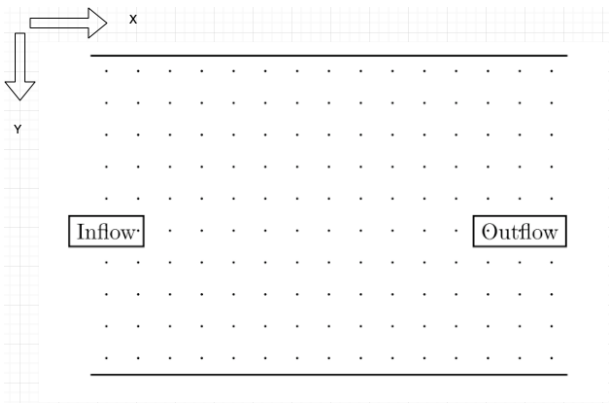


Figure 9: Poiseuille flow simulation domain with hard walls on top and bottom [5]

As can be seen from the developed velocity profile in Figure 10, the flow has a parabolic shape which shows that it has a laminar flow regime which is the case for $Re < 2300$.

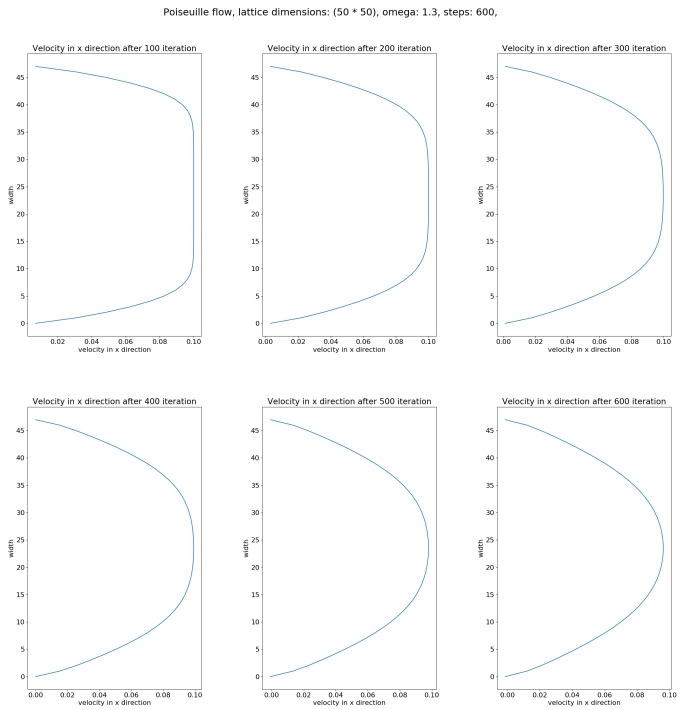


Figure 10: Evolution of velocity profile over time in x (length) direction

4.3 Couette flow

For simulation of Couette flow which is an inlet with a moving wall on top, periodic boundary conditions should be applied from east to west. Bounce-back should also be applied on solid non-moving walls which is in this simulation the bottom wall. An illustration of Couette flow simulation domain is depicted in Figure 11.

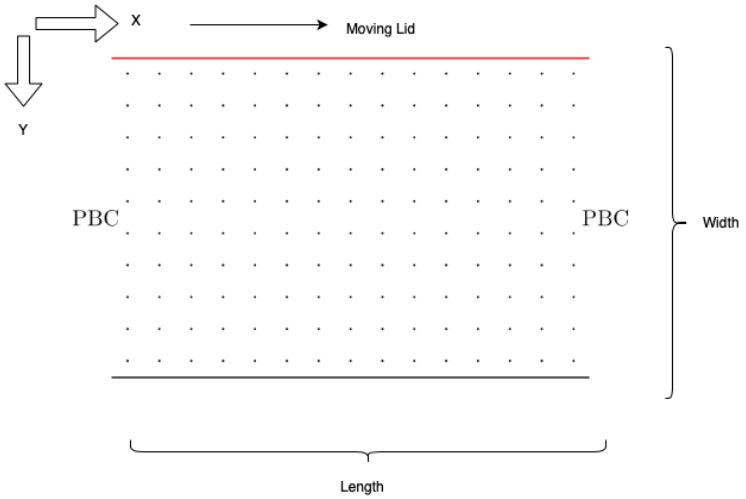


Figure 11: Couette flow simulation domain with hard wall in the bottom and moving lid on top

4.4 Lid-driven Cavity

In this section, the Lid-driven cavity problem is implemented for a lattice of size (300 * 300) with initial density $\rho(r, t = 0) = 1$ and velocity field of $u_x(r, t = 0) = 0, u_y(r, t = 0) = 0$.
A moving lid with a constant velocity of u_0 is moving on top of three solid walls as depicted in Figure 12.

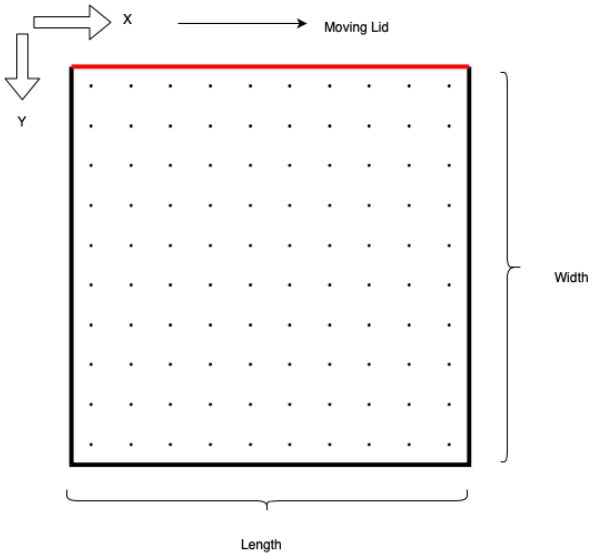
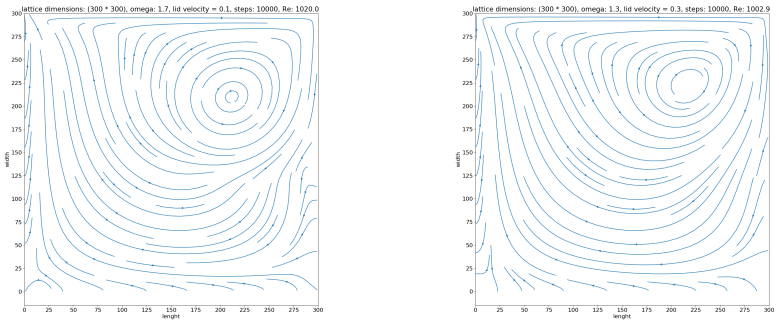


Figure 12: Lid-driven cavity simulation domain with hard walls on left, right and bottom and moving lid on top

For a given kinematic viscosity, the collision frequency can be calculated using Equation 2.12:

$$\omega = \frac{2}{6\nu + 1} \tag{4.1}$$

For a collision frequency $\omega = 1.7$, lid velocity $u_0 = 0.1$, and Length $L = 300$ using Equation 2.12 the reynolds number is $Re \approx 1000$.
For another set of parameters; $\omega = 1.3$, $u_0 = 0.3$ and the same length, the reynolds number is also $Re \approx 1000$.
Two different runs of Lid-driven Cavity with mentioned parameters in illustrated in Figure 13.



(a) $\omega = 1.7$, $u_0 = 0.1$ (b) $\omega = 1.3$, $u_0 = 0.3$

Figure 13: Lid-driven cavity velocity profile after 10000 steps

Evolution of Lid-driven cavity for two different above mentioned settings with approximately the same Re is depicted in Figure 14 and Figure 15.

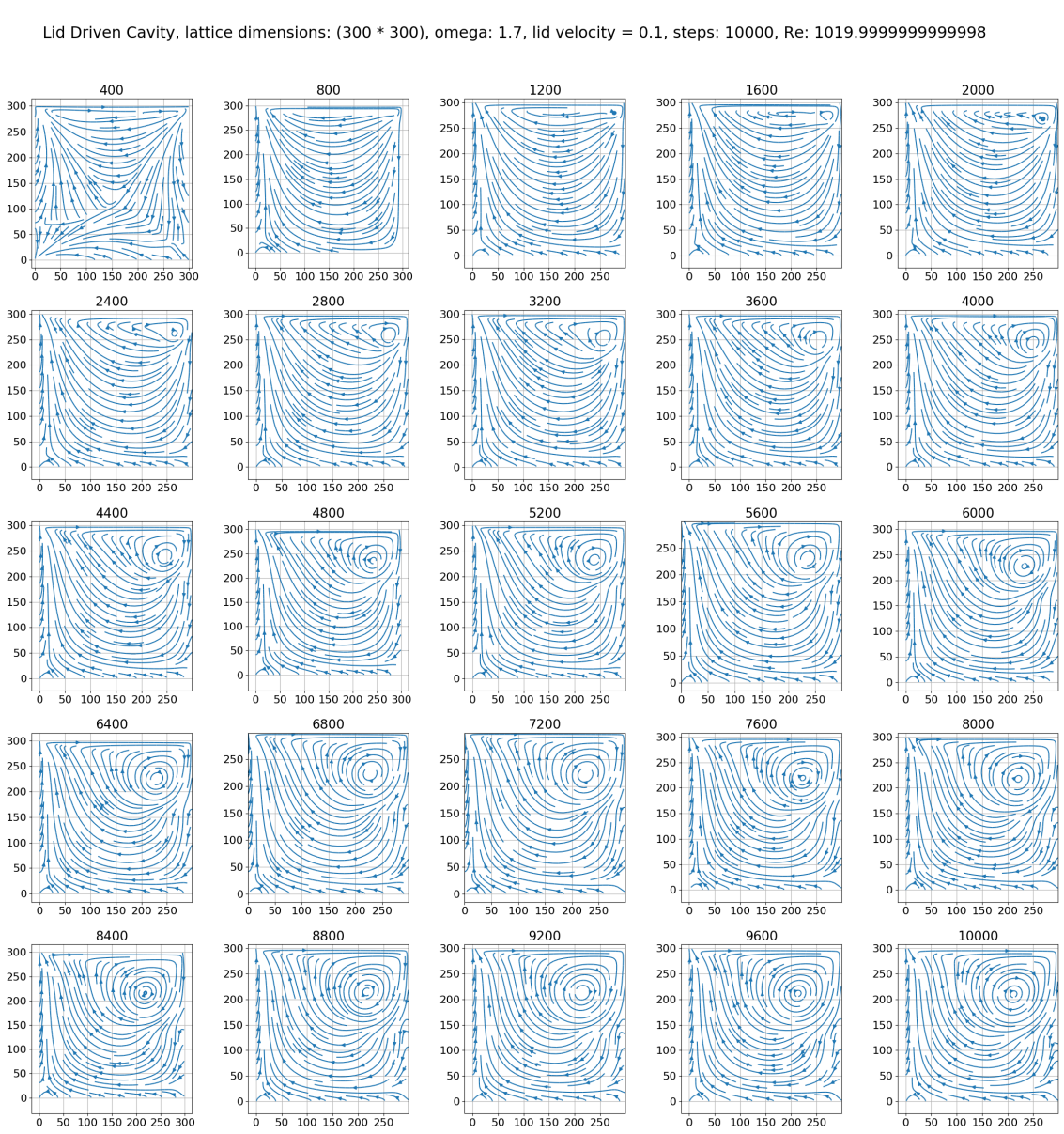


Figure 14: Evolution of Lid-driven cavity with $\omega = 1.7$, $u_0 = 0.1$

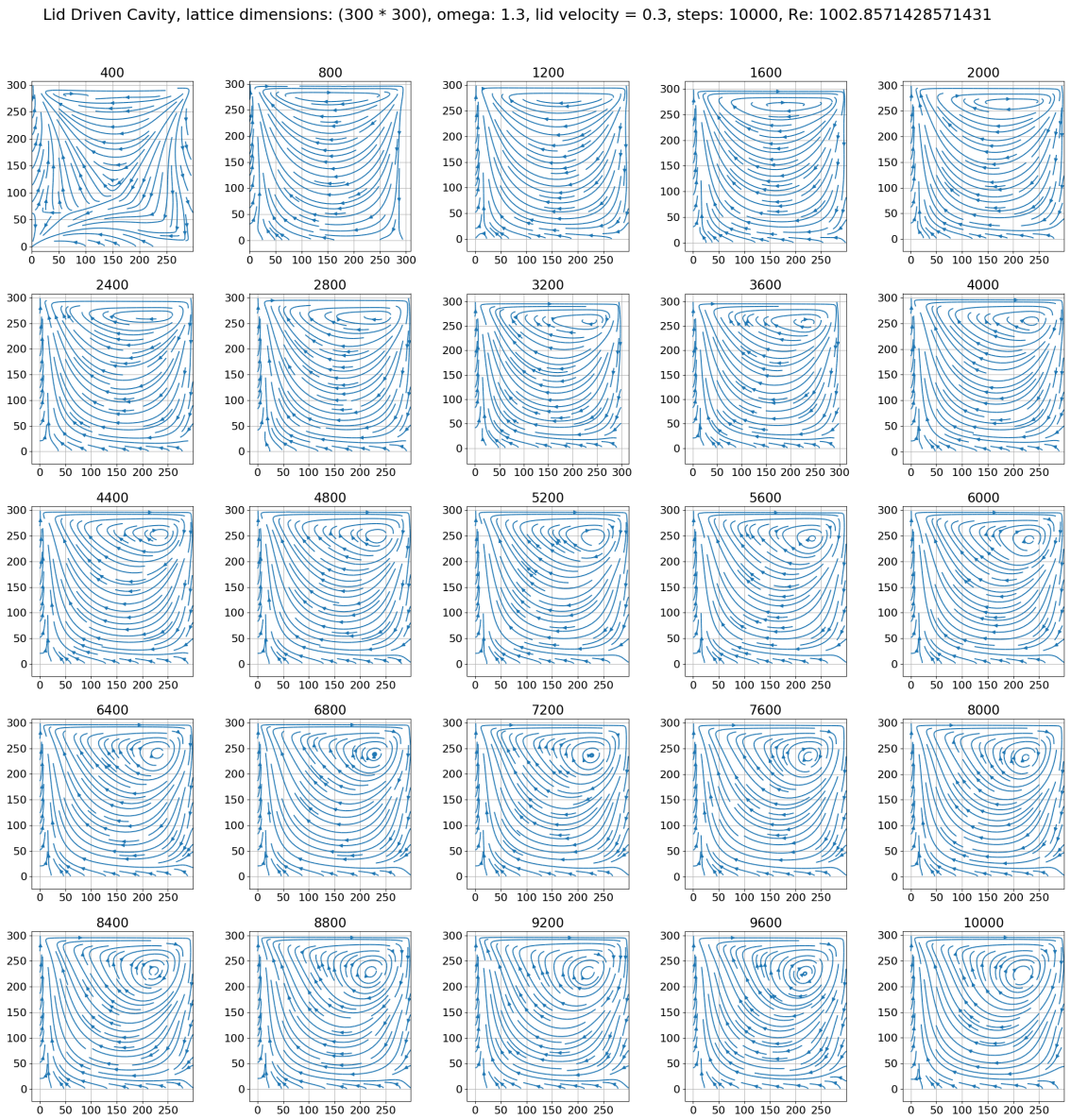


Figure 15: Evolution of Lid-driven cavity with $\omega = 1.3$, $u_0 = 0.3$

5 Conclusions

In the Lid-driven cavity problem it can be seen that if Re is equal, value of collision frequency (ω) and lid velocity (u_0) doesn't affect the final results and the flow profile would look similar.

Bibliography

[1] The boltzmann transport equation. http://www.iue.tuwien.ac.at/phd/bina/dissse2.html. Accessed: 2019-08-03.

[2] Lattice-boltzmann; a versatile tool for multiphase. https://fas.org/sgp/othergov/doe/lanl/pubs/00285550.pdf. Accessed: 2019-08-03.

[3] What is the reynolds number? https://www.simscale.com/docs/content/simwiki/numerics/what-is-the-reynolds-number.html. Accessed: 2019-08-16.

[4] S. Chen and G. D. Doolen. Lattice boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30(1):329–364, 1998.

[5] A. Greiner and L. Pastewka. *High-Performance Computing with Python - Lecture notes*. University of Freiburg, May 2019.

[6] A. A. Mohamad. *Lattice Boltzmann Method*. Springer, London, 2011.