THE LATTICE GAS MODEL AND LATTICE BOLTZMANN MODEL ON HEXAGONAL GRIDS

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

Kang Jin

Certificate of Approval:

Paul G. Schmidt
Associate Professor
Department of Mathematics

Amnon J. Meir, Chair
Professor
Department of Mathematics

Wenxian Shen
Professor
Department of Mathematics

Tin-Yau Tam
Professor
Department of Mathematics

Stephen L. McFarland
Dean
Graduate School

The Lattice Gas Model and Lattice Boltzmann Model On Hexagonal Grids

Kang Jin

A Thesis
Submitted to
the Graduate Faculty of
Auburn University
in Partial Fulfillment of the
Requirements for the
Degree of
Master of Science

Auburn, Alabama
August 8, 2005

THE LATTICE GAS MODEL AND LATTICE BOLTZMANN MODEL ON HEXAGONAL GRIDS

Kang Jin

Permission is granted to Auburn University to make copies of this thesis at its discretion, upon the request of individuals or institutions and at their expense. The author reserves all publication rights.

Signature of Author

Date

Copy sent to:

Name	Date

VITA

Kang Jin was born in Shanghai, China in 1979. He did all his undergraduate study in Shanghai. He graduated from East China Normal University in 2001 with a Bachelors Degree in Mathematics. He then went to the United States in 2002. He accepted an offer from Auburn University, and began his graduate study as well as being a Teaching Assistant.

THESIS ABSTRACT

THE LATTICE GAS MODEL AND LATTICE BOLTZMANN MODEL ON HEXAGONAL GRIDS

Kang Jin

Master of Science, August 8, 2005
(B.S., East China Normal University, 2001)

55 Typed Pages

Directed by Amnon J. Meir

We present an overview of the FHP model and the Lattice BGK model. Details regarding boundary conditions and initial conditions are discussed through implementations on driven cavity flow, Poiseuille flow, and flow past a cylinder. We describe our implementation of the method, justify the method and present some results, and analysis for different Reynolds numbers.

Acknowledgments

I thank Auburn University Mathematics and Statistics department for offering me the Graduate Teaching Assistant position. This is very important to me. Without this offer I could not be where I am today. I also thank the faculty in the Mathematics and Statistics department who give me lots of help. I thank my committee members Dr. Wenxian Shen, Dr. Paul Schmidt and Dr. T.Y. Tam. Special thank to my advisor, Dr. Amnon J. Meir. His support and advice are the key to my work.

Style manual or journal used Journal of Approximation Theory (together with the style known as "aums"). Bibliograpy follows van Leunen's *A Handbook for Scholars*.

Computer software used The document preparation package TEX (specifically LATEX) together with the departmental style-file `aums.sty`.

TABLE OF CONTENTS

LIST OF FIGURES ix

1 INTRODUCTION 1

2 THE FHP MODEL 4
- 2.1 Basic model 4
- 2.2 Macroscopic quantities 7
- 2.3 Implementation 8
  - 2.3.1 Data structure 8
  - 2.3.2 Program structure 9
  - 2.3.3 Results 10

3 THE LATTICE BGK MODEL 12
- 3.1 Basic model 12
- 3.2 The Navier-Stokes 13
  - 3.2.1 The conservation laws 13
  - 3.2.2 The Chapman-Enskog expansion 15
- 3.3 Boundary and initial conditions 19
- 3.4 Implementation 19
- 3.5 Results and data analysis 21
  - 3.5.1 Driven cavity 21
  - 3.5.2 Poiseuille flow 21
  - 3.5.3 Flow past a cylinder 27

4 CONCLUSIONS AND FUTURE WORKS 29

BIBLIOGRAPHY 30

APPENDICES 32
- .1 FHP collision look-up table 32
- .2 Partial Matlab code I: FHP streaming of a driven cavity 33
- .3 Partial code II: Lattice BGK initialization of a driven cavity 38
- .4 Partial code III: Lattice BGK collision of a driven cavity 41

LIST OF FIGURES

2.1 The hexagonal grid. One site can occupy a maximum of 7 particles. . . . 4

2.2 FHP collision rule 6

2.3 A 4 × 3 grid example. 8

2.4 A driven cavity example for the FHP model. The Reynolds number is 934. Left column: results of FHP. Middle column: results of FHP with an ensemble average on 10 experiments. Right column: results of Lattice BGK model for comparison. 11

3.1 Left boundary of a hexagonal grid. 20

3.2 Driven cavity at $Re = 10$. 22

3.3 Driven cavity at $Re = 100$. 22

3.4 Driven cavity at $Re = 200$. 23

3.5 Driven cavity at $Re = 400$. 23

3.6 Ghia-Ghia-Shin's [12] result. The plot of the velocity contour with a Reynolds number of 400. 24

3.7 Driven cavity at $Re = 800$. 24

3.8 Dimensionless x-velocity profile at the geometry center of the cavity for Reynolds number 10, 100, 200, 400, 800. Dashed line is the result from Ghia Ghia Shin [12] at Reynolds number 400. 25

3.9 Dimensionless y-velocity profile at the geometry center of the cavity for Reynolds number 10, 100, 200, 400, 800. Dashed line is the result from Ghia Ghia Shin [12] at Reynolds number 400. 26

3.10 An example of a Couette flow. Above: velocity contour. Below: x-velocity profile at $x = 800$. Solid line is the simulation, and dashed line is the actual solution. 27

3.11 An example of a uniform flow past a cylinder for $Re = 400$. Above: velocity contour. Below: vorticity contour. 28

Chapter 1

Introduction

The Lattice Gas model and Lattice Boltzmann model are methods for simulating fluids flows. The flow of incompressible fliuds can be described by the Navier-Stokes equation

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = -\nabla P + \nu \nabla^2 \vec{u} \tag{1.1}$$

and the continuity equation

$$\nabla \cdot \vec{u} = 0 \tag{1.2}$$

where $\vec{u}$ is the velocity, $P = p/\rho_0$ the kinematic pressure, p the pressure, ρ_0 the constant mass density and $\nu = \eta/\rho_0$ the kinematic shear viscosity, η the dynamic shear viscosity.

The Lattice Boltzmann model is derived from Lattice Gas model. These two models are different from models such as finite difference, finite volume, and finite element which are based on the discretization of partial differential equations (top-down models [1]). These two models are based on a discrete microscopic model which conserves desired quantities (such as mass and momentum); then the partial differential equations are derived by multi-scale analysis (bottom-up models).

First introduced in 1973, by Hardy, de Pazzis and Pomeau (HPP) [2], the HPP model is a Lattice Gas model. It simulates the microscopic behavior of the fluid utilizing a square grid. The basic idea is to create a simple Cellular Automaton obeying nothing but conservation laws at a microscopic level that was able to reproduce the complexity of real fluid flows. Fluid particles of identical mass are only allowed to travel on the lattice at unit speed. All lattice sites, which are the intersections of the lattice, are exclusive. This means that only one particle is allowed to travel at each of the four directions of one site. This gives a maximum of 4 particles at each site. Each site can take only a finite number of states, actually $2^4 = 16$ states. At each time step, collision occurs at each site, according to a collision rule which conserves the density and the momentum. Then particles travel along a straight line (free streaming) until they meet some other particle or the boundary.

This model is friendly to computer, since only a 4-bit variable is needed, and only logical operations are required. Also, only the information from the four neighbours are needed at each collision and streaming, this model is easy to parallelize. Although the calculations that the HPP requires are simple, it leads to a macroscopical anisotropical Navier-Stokes equation. This defect prevents the HPP from being applied to most fluid problems. In 1986 Frisch, Hasslacher and Pomeau (FHP) [3] introduced a lattice gas model based on a hexagonal grid. This grid change made the FHP model exhibit isotropy. Details of the FHP are discussed and several implementations are presented in the next chapter.

Similar to the HPP, logical operations made the FHP model easy to implement on computers. Now the biggest problem of the cellular automata is the noise, since it is based on a Fermi-Dirac distribution of the equilibrium population because of the exclusion principle. The Fermi-Dirac distruibution is a distribution that applies to particles called fermions. Fermions have half-integral values of the quantum mechanical property called spin and are “antisocial” in the sense that two fermions cannot exist in the same state. Protons, neutrons, electrons, and many other elementary particles are fermions. As we will see in the next chapter, the results of the FHP are very noisy. Ensemble average and space average should both be used. This may result in a grid size thousand times larger than the original problem. For example, if the final solution on a 100×100 grid is needed, averaging on 10×10 cells and ensemble averaging on 10 experiments leads to calculations 10 times the size of a 1000×1000 grid! The lack of Galilean invariance is another big problem of the FHP. The collision rules can be written in a look-up table. For the FHP model, this table should have a size of $2^7 \times 7$. For multi-dimensional simulations, the huge look-up table associated with the collision rules makes this almost impossible.

The Lattice Boltzmann model overcomes these defects very well. Instead of using boolean variables at each site, the Lattice Boltzmann model uses real numbers. The first model, proposed by McNamara and Zanetti [4], replaced the boolean variables with their ensemble average. The statistical noise is greatly reduced. After that the linear collision operator [5] came into being and then the enhanced collision rule [6]. The breakthrough is the single relaxation time approximation, known as the Lattice BGK model, named after Bhatnagar, Gross, and Krook [7]. This model dramatically reduces computation and good results are obtained in various applications. In this thesis, we will discuss in detail the Lattice BGK model. Simulations up to Reynolds number 1000 are presented.

Chapter 2

The FHP Model

2.1 Basic model

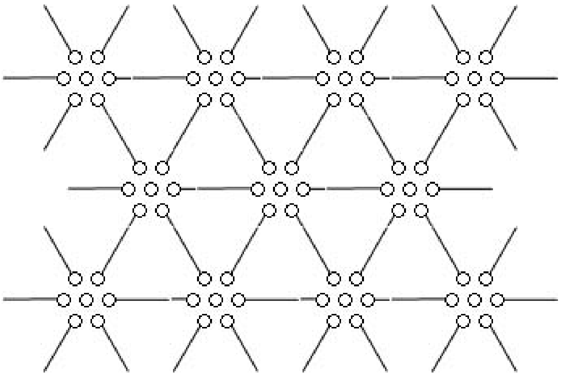

Figure 2.1: The hexagonal grid. One site can occupy a maximum of 7 particles.

The Lattice Gas Cellular Automata simulates molecular collision in a discretized fashion. Here let us consider a hexagonal grid shown in Figure 2.1. Each site is surrounded by 6 neighbours, connected by unit vectors

$$\vec{e}_i = (\sin(\frac{\pi}{3}(i-1)), \cos(\frac{\pi}{3}(i-1))), i = 1, ..., 6. \tag{2.1}$$

Exclusion principles allow a maximum of 7 particles at one site, one moving particle in each of the 6 directions, together with a rest particle in the middle. Here we use an occupation boolean variable $n_i(\vec{x}, t)$, $i = 0, 1, ..., 6$ (0 stands for the rest particle) to indicate particle presence or non-presence at the i^{th} direction of site $\vec{x}$ at time t. Thus a

7-bit variable is enough to carry the information at one site. All particles have the same mass and the same speed.

One time step consists of a collision and a streaming. Collision only occurs at the sites, while streaming takes place on the connection between each two sites. The collision rules conserve mass and momentum. Figure 2.2 shows the basic set of collision rules [8]. The left column are the in-states and the right are the out-states. In-state means that particles are moving towards the center of the site. After the collision follows the out-state, particles then move away from the center of the site and begin streaming. So a full time step is:

$$\text{in-state} \Longrightarrow \text{collision} \Longrightarrow \text{out-state} \Longrightarrow \text{streaming} \Longrightarrow \text{in-state}.$$

By rotating, flipping, and combining these 7 rules, one can get a full set of 128 collision rules. Notice that some in-states will lead to two equivalent out-states. It is not necessary to pick an out-state randomly at every site. Notice that picking a random number is very time consuming. Instead, one can pick a single random boolean variable for all the sites at one time step.

Compared to the original infinite number of traveling directions in the p.d e., the 6 directions model lacks degrees of freedom, yet it can display all the complexities of fliud phenomema [9]. This is the simplest isotropic model. By limiting the types of collision, the FHP can be divided into three types. The FHP-I only allows collisions of type (a) and (b). No rest particle is present. Types of collision other than (a) and (b) are replaced by simple streamig as if the particles don't collide at all. FHP-II adds the rest particle, together with collision type (c), (d), and (e). FHP-III includes all types of collisions.

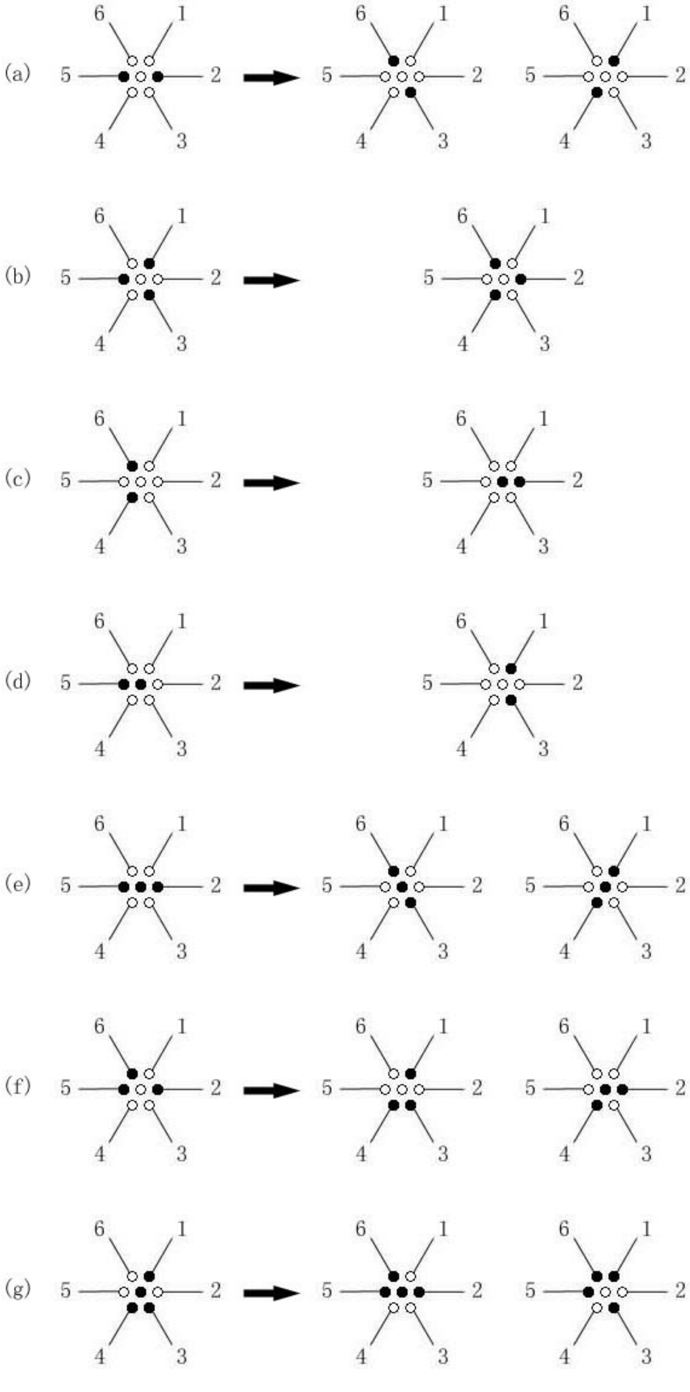

Figure 2.2: FHP collision rule

A no-slip boundary condition can be applied by a bounce back scheme, which means particles that hit the boundary at any angle should move back in the opposite direction. Reflection will lead to a slip boundary. The Dirichelet boundary condition can be set as a random variable on the boundary with a probability distuibution indicating the value at the boundary, then applying the collision followed by the bounce back scheme.

2.2 Macroscopic quantities

Noise is the biggest problem of the FHP model. Hence both space average and ensemble average should be used. The space average is achieved by averaging on small cells, for example, 16×16 cells. In Figure 2.4 we present an experiment of driven cavity flow using FHP-III. We used a 200×240 grid. Since the grid is hexagonal, we take space average on a 10×12 cell. This is close to a square, which then gives us a 20×20 square grid.

The density is given by $\rho = \sum_i n_i$. And ρ_0 is the mean density, which is the average on the whole grid.

Here are some model-dependent quantities derived by Frisch *et al.* [3] for the three types of FHP models.

	FHP-I	FHP-II	FHP-III
d	$\rho_0/6$	$\rho_0/7$	$\rho_0/7$
c_s	$\frac{1}{\sqrt{2}}$	$\sqrt{\frac{3}{7}}$	$\sqrt{\frac{3}{7}}$
ν	$\frac{1}{12}\frac{1}{d(1-d)^3} - \frac{1}{8}$	$\frac{1}{28}\frac{1}{d(1-d)^3}\frac{1}{1-4d/7} - \frac{1}{8}$	$\frac{1}{28}\frac{1}{d(1-d)}\frac{1}{1-8d(1-d)/7} - \frac{1}{8}$

where d is the mean density per link, c_s is the speed of sound, and ν is the kinematic viscosity.

2.3 Implementation

The information above is enough to write a code. Here we talk about the implementation of the FHP-III model. The model problem is a driven cavity flow. For simplicity, we use Matlab as programming language.

2.3.1 Data structure

We number the sites of the grid from left to right, and from top to bottom. Notice the odd rows have one more site than the even rows. A $m \times n$ grid is a grid that has $n+1$ sites on odd rows and n sites on even rows, and a total number of $m+1$ odd and even rows. Figure 2.3 shows a 4×3 grid.

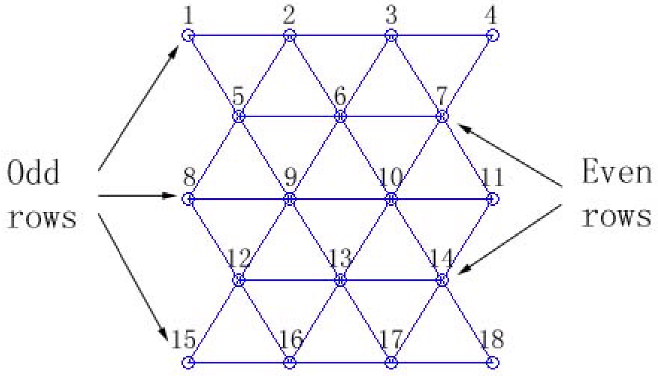

Figure 2.3: A 4×3 grid example.

As we discussed in the last section, a 7-bit variable is enough to carry all the information at one site, we need an array that has length of the total number of sites, and each entry is a 7-bit variable. Let $M(n)$ be this array and n be the site number corresponding to the position $\vec{x}$. In Matlab, you can use an integer from 0 to 127 as a 7-bit variable. (Do not use an array with 7 entries, it is too slow!) Here we set

$$\begin{aligned} n_i(\vec{x},t) &= sgn(M(n) \ \& \ 2^{i-1}), \ i = 1,..,6 \\ n_0(\vec{x},t) &= sgn(M(n) \ \& \ 2^6) \end{aligned} \tag{2.2}$$

where & is the logical operation "and", $sgn()$ is the sign function. This means the number $2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6$ indicate the presence of the $1^{st}, 2^{nd}, 3^{rd}, 4^{th}, 5^{th}, 6^{th}$ direction particle and the rest particle, respectively. Most of the times, $M(n)$ is a combination of the 7 numbers.

2.3.2 Program structure

The main function give all the information, including the size of the grid, and the characterisic speed for the driven cavity flow. Three subroutines would be created, the initialization, the collision and the streaming. The collision and the streaming should be in a loop controlled by a stop criterion specified. Here is the structure:

```
Initialization
for i = 1 to "ensemble average number"
    for j = 1 to "specified number of time steps"
        Create a random boolean variable
        Streaming
        Collision
    end
end
```

In the initialization, the collision look-up table and the array M are created. Then we initialize each site of the grid by creating one particle randomly at each of the 7 positions. In the streaming, most of the work is for the bounce back scheme at the boundary. For a driven cavity flow, the boundary condition at the top boundary can be set as a random variable on the boundary with a probability distuibution indicating the value. For example, suppose the characteristic speed is 0.8 (which is less than 1). Then $M(n)$ at the top boundary can be reset as $M(n) = X$, where X is a boolean random variable satisfying $\Pr(X = 1) = 0.8$. In the collision, just refer to the collision look-up table [Appendix A] created in the initialization, find out the out-state corresponding to the in-state for each site, use the one indicated by the random boolean variable created.

2.3.3 Results

Here are some pictures of the driven cavity flow experiment. The Reynolds number turns out to be 934. The speed on the top is set to be 1. We compare the result with a Lattice BGK model with the same Reynolds number. Details of the LBM are presented in the next section. From the picture, the noise is clearly visible.

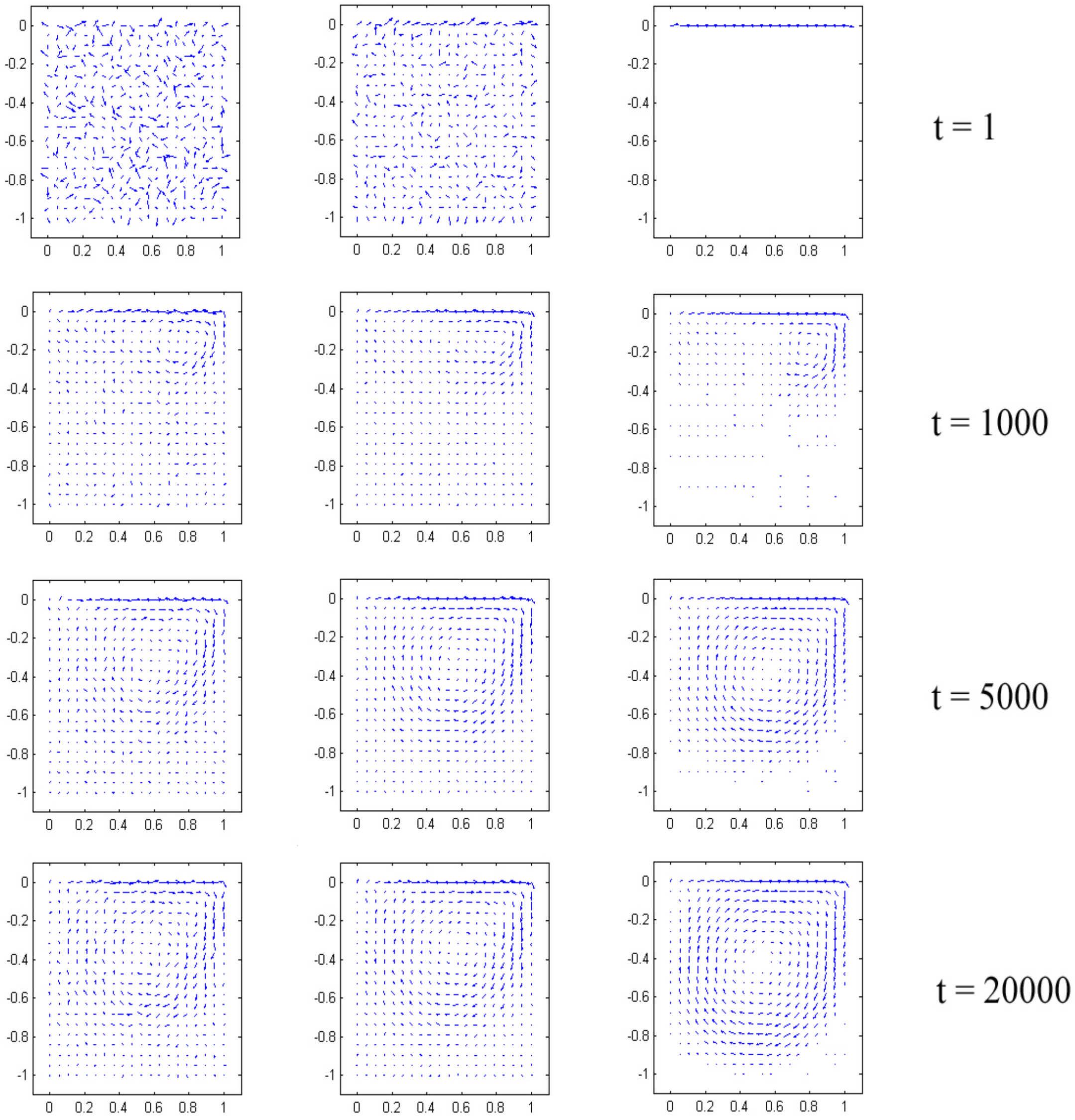

Figure 2.4: A driven cavity example for the FHP model. The Reynolds number is 934. Left column: results of FHP. Middle column: results of FHP with an ensemble average on 10 experiments. Right column: results of Lattice BGK model for comparison.

CHAPTER 3

THE LATTICE BGK MODEL

3.1 Basic model

Let us consider again the hexagonal grid. This time, the occupation number is replaced by its ensemble average value, or, the particle distribution function $f_i(\vec{x}, t)$. The meaning of the function $f_i(\vec{x}, t)$ is the probability of finding a particle moving in the i^{th} direction of the site $\vec{x}$ at time t. The collision rules in FHP are replaced with a collision operator Ω_i, and the particle distribution function should satisfy the Lattice Boltzmann equation

$$f_i(\vec{x} + e_i, t + 1) - f_i(\vec{x}, t) = \Omega_i. \tag{3.1}$$

This collision operator has lots of forms. Here we talk about the simplest one, the BGK single relaxation time model. Introduce the single relaxation parameter τ and the equilibrium distribution function $f_i^{eq}(\vec{x}, t)$. By assuming that the particle distribution function approaches the equilibrium state at a constant rate, we get

$$\Omega_i = -\frac{1}{\tau}(f_i - f_i^{eq}). \tag{3.2}$$

This gives us the equation

$$f_i(\vec{x} + e_i, t + 1) = (1 - w) f_i(\vec{x}, t) + w f_i^{eq}(\vec{x}, t) \tag{3.3}$$

where the weight $w = \frac{1}{\tau}$. The equilibrium distribution function has the form

$$f_i^{eq}(\vec{x},t) = \rho(\vec{x},t)\left(\frac{1-z}{6} + \frac{D}{6c^2}(\vec{e}_i \cdot \vec{u}) + \frac{D(D+2)}{12c^4}(\vec{e}_i \cdot \vec{u})^2 - \frac{D\vec{u}^2}{12c^2}\right), i = 1, ..., 6 \quad (3.4)$$

$$f_0^{eq}(\vec{x},t) = \rho(\vec{x},t)(z - \frac{u^2}{c^2}) \quad (3.5)$$

where $\rho(\vec{x},t) = \sum_i f_i$ is the density. Here z is a parameter, we choose $z = \frac{1}{2}$. Also D is the dimension, where $D = 2$. $c = |e_i|$, here $c = 1$. And the speed of sound c_s is controlled by the parameter z by

$$c_s = \sqrt{\frac{1-z}{2}}. \quad (3.6)$$

A given kinematic viscosity can be achieved by choosing a proper relaxation parameter τ from the relation

$$\nu = \frac{c^2}{D+2}\left(\tau - \frac{1}{2}\right). \quad (3.7)$$

3.2 The Navier-Stokes

3.2.1 The conservation laws

From the definition of the unit vectors e_i, one can get the following equations [11].

$$\begin{aligned}
\sum_i e_{i\alpha} &= 0 && (3.8)\\
\sum_i e_{i\alpha}e_{i\beta} &= \frac{c^2 b}{D}\delta_{\alpha\beta} && (3.9)\\
\sum_i e_{i\alpha}e_{i\beta}e_{i\gamma} &= 0 && (3.10)\\
\sum_i e_{i\alpha}e_{i\beta}e_{i\gamma}e_{i\delta} &= \frac{c^4 b}{D(D+2)}(\delta_{\alpha\beta}\delta_{\gamma\delta}+\delta_{\alpha\gamma}\delta_{\beta\delta}+\delta_{\alpha\delta}\delta_{\beta\gamma}) && (3.11)\\
\sum_i e_{i\alpha}e_{i\beta}e_{i\gamma}e_{i\delta}e_{i\epsilon} &= 0 && (3.12)
\end{aligned}$$

where $e_{i\alpha}$ stands for the α direction component (one of the i, j directions on the 2 dimentional plane) of the unit vector $\vec{e_i}$. Using the first two, one can obtain the moments of the equilibrium distribution function. First sum the equilibrium distribution function and get conservation of mass and momentum

$$\sum_i f_i^{eq} = \rho \tag{3.13}$$

$$\sum_i f_i^{eq} e_{i\alpha} = \rho u_\alpha. \tag{3.14}$$

Also, from the rest of the equations, one gets

$$\begin{aligned}
\sum_i f_i^{eq} e_{i\alpha}e_{i\beta} &= \frac{\rho(1-z)c^2}{D}\delta_{\alpha\beta} + \rho u_\alpha u_\beta && (3.15)\\
\sum_i f_i^{eq} e_{i\alpha}e_{i\beta}e_{i\gamma} &= \frac{\rho c^2}{D+2}(u_\alpha\delta_{\beta\gamma}+u_\beta\delta_{\alpha\gamma}+u_\gamma\delta_{\alpha\beta}). && (3.16)
\end{aligned}$$

3.2.2 The Chapman-Enskog expansion

The distribution funcition can be expanded as follows

$$f_i = f_i^{(0)} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)} + ... \tag{3.17}$$

where $|\epsilon| \ll 1$. Here we can use the Knudsen number Kn as ϵ. The Knudsen number is defined as

$$Kn = \frac{\lambda}{L}$$

where λ is the mean free path, and L is the characteristic length. One can think of this expansion of the distribution function (f) as an equilibrium distribution function ($f^{(0)}$) together with some pertubations ($f^{(1)}, f^{(2)}...$) of higher order in ϵ. We also expand $\vec{x}$ and t as

$$\vec{x} = \frac{\vec{x}_1}{\epsilon} + ..., \; t = \frac{t_1}{\epsilon} + \frac{t_2}{\epsilon^2} + ... \tag{3.18}$$

where $\vec{x}_1 = o(\epsilon)$, $t_1 = o(\epsilon)$, $t_2 = o(\epsilon^2)$. In this case, one get

$$\begin{aligned} \frac{\partial}{\partial x_\alpha} &= \epsilon \frac{\partial}{\partial x_{1\alpha}} + ..., & (3.19) \\ \frac{\partial}{\partial t} &= \epsilon \frac{\partial}{\partial t_1} + \epsilon^2 \frac{\partial}{\partial t_2} + ... & (3.20) \end{aligned}$$

Now we perform a Taylor expension on the Lattice Boltzmann equation 3.1 in both space and time, we obtain

$$\left[\left(\frac{\partial}{\partial t} + e_{i\alpha} \frac{\partial}{\partial x_\alpha} \right) + \frac{1}{2} \left(\frac{\partial}{\partial t} + e_{i\alpha} \frac{\partial}{\partial x_\alpha} \right)^2 \right] f_i(\vec{x}, t) = \Omega_i. \tag{3.21}$$

Notice that Einstein summation is used. So $e_{i\alpha}\dfrac{\partial}{\partial x_\alpha} = \sum\limits_{\alpha=1,2} e_{i\alpha}\dfrac{\partial}{\partial x_\alpha}$. Using the expansions of $f, \dfrac{\partial}{\partial x_\alpha}, \dfrac{\partial}{\partial t}$, together with equation 3.2, we get

$$\begin{aligned}\left[\left(\epsilon\frac{\partial}{\partial t_1} + \epsilon^2\frac{\partial}{\partial t_2} + \epsilon e_{i\alpha}\frac{\partial}{\partial x_{1\alpha}}\right) + \frac{1}{2}\left(\epsilon\frac{\partial}{\partial t_1} + \epsilon^2\frac{\partial}{\partial t_2} + \epsilon e_{i\alpha}\frac{\partial}{\partial x_{1\alpha}}\right)^2\right] \\ \times\left(f_i^{(0)} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)}\right) = -\frac{1}{\tau}(f_i^{(0)} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)} - f_i^{eq}).\end{aligned} \tag{3.22}$$

Set the 0^{th} order approximation $f_i^{(0)}$ to be the equilibrium distribution function f_i^{eq}. The conservation of mass and momentum requires that $\sum\limits_i f_i^{(k)} = 0$ and $\sum\limits_i f_i^{(k)} e_{i\alpha} = 0$, $k = 1, 2$. From these equations to first-order in ϵ we get

$$\frac{\partial}{\partial t_1} f_i^{(0)} + e_{i\alpha}\frac{\partial}{\partial x_{1\alpha}} f_i^{(0)} = -\frac{f_i^{(1)}}{\tau}. \tag{3.23}$$

Summing over i and from equation 3.13 and 3.14 we get

$$\frac{\partial\rho}{\partial t_1} + \frac{\partial}{\partial x_{1\alpha}}\rho u_\alpha = 0. \tag{3.24}$$

Now multiply equation 3.23 by the unit vectors $e_{i\beta}$ and again sum over i, using equation 3.15 gives

$$\frac{\partial}{\partial t_1}\rho u_\beta + \frac{\partial}{\partial x_{1\alpha}}\rho u_\alpha u_\beta - \frac{\partial}{\partial x_{1\alpha}}\frac{\rho(1-z)c^2}{D}\delta_{\alpha\beta} = 0. \tag{3.25}$$

From equation 3.22 to second-order in ϵ and by equation 3.23 we get

$$\begin{aligned}\left[\frac{\partial}{\partial t_2} + \frac{1}{2}\frac{\partial}{\partial t_1}\left(\frac{\partial}{\partial t_1} + e_{i\alpha}\frac{\partial}{\partial x_{1\alpha}}\right) + \frac{1}{2}e_{i\alpha}\frac{\partial}{\partial x_{1\alpha}}\left(\frac{\partial}{\partial t_1} + e_{i\beta}\frac{\partial}{\partial x_{1\beta}}\right)\right] f_i^{(0)} \\ + \left(\frac{\partial}{\partial t_1} + e_{i\alpha}\frac{\partial}{\partial x_{1\alpha}}\right) f_i^{(1)} = -\frac{1}{\tau} f_i^{(2)}.\end{aligned} \tag{3.26}$$

Summing over i and using equation 3.24, all $\frac{\partial}{\partial t_1} + e_{i\alpha}\frac{\partial}{\partial x_{1\alpha}}$ vanished, and one gets

$$\frac{\partial}{\partial t_2}\rho = 0.$$

Again multiplying the equation by a unit vector $e_{i\gamma}$ gives the following

$$\left[\frac{\partial}{\partial t_2}e_{i\gamma} + \frac{1}{2}\frac{\partial}{\partial t_1}\left(\frac{\partial}{\partial t_1}e_{i\gamma} + e_{i\alpha}e_{i\gamma}\frac{\partial}{\partial x_{1\alpha}}\right) + \frac{1}{2}e_{i\alpha}e_{i\gamma}\frac{\partial}{\partial x_{1\alpha}}\left(\frac{\partial}{\partial t_1} + e_{i\alpha}\frac{\partial}{\partial x_{1\alpha}}\right)\right] f_i^{(0)} + \left(\frac{\partial}{\partial t_1}e_{i\gamma} + e_{i\alpha}e_{i\gamma}\frac{\partial}{\partial x_{1\alpha}}\right) f_i^{(1)} = -\frac{1}{\tau}f_i^{(2)}. \tag{3.27}$$

By multiplying equation 3.23 by $e_{i\alpha}e_{i\gamma}\frac{\partial}{\partial x_{1\alpha}}$, one can rewrite the second term of $f_i^{(1)}$ as

$$\frac{\partial}{\partial x_{1\alpha}}e_{i\alpha}e_{i\gamma}f_i^{(1)} = -\tau\left(\frac{\partial}{\partial t_1}\frac{\partial}{\partial x_{1\alpha}}e_{i\alpha}e_{i\gamma}f_i^{(0)} + \frac{\partial}{\partial x_{1\beta}}\frac{\partial}{\partial x_{1\alpha}}e_{i\alpha}e_{i\beta}e_{i\gamma}f_i^{(0)}\right). \tag{3.28}$$

Combining this term with the third term of $f_i^{(0)}$, one gets

$$\begin{aligned}&\left[\frac{\partial}{\partial t_2}e_{i\gamma} + \frac{1}{2}\frac{\partial}{\partial t_1}\frac{\partial}{\partial t_1}e_{i\gamma} + e_{i\alpha}e_{i\gamma}\frac{\partial}{\partial x_{1\alpha}} - \right.\\ &\quad\left.\left(\tau - \frac{1}{2}\right)\left(\frac{\partial}{\partial t_1}\frac{\partial}{\partial x_{1\alpha}}e_{i\alpha}e_{i\gamma} + \frac{\partial}{\partial x_{1\beta}}\frac{\partial}{\partial x_{1\alpha}}e_{i\alpha}e_{i\beta}e_{i\gamma}\right)\right] f_i^{(0)} + \frac{\partial}{\partial t_1}e_{i\gamma}f_i^{(1)}\\ &\quad = -\frac{1}{\tau}f_i^{(2)}.\end{aligned} \tag{3.29}$$

Summing over i, the right-hand side is 0. The second term of $f_i^{(0)}$ is 0 by equation 3.25. The term of $f_i^{(1)}$ is 0 by the conservation of momentum requirement. The third term of $f_i^{(0)}$ can be obtained by equation 3.15 to the order $O(u)$ and then converting time derivatives into spatial derivatives using equation 3.24, we get

$$\begin{aligned}\frac{\partial}{\partial t_2}\rho u_\gamma &= \left(\tau-\frac{1}{2}\right)\left[\frac{\partial}{\partial t_1}\frac{\partial}{\partial x_{1\alpha}}\frac{\rho(1-z)c^2}{D}\delta_{\alpha\gamma}\right.\\ &\quad \left.+\frac{\partial}{\partial x_{1\alpha}}\frac{\partial}{\partial x_{1\beta}}\frac{\rho c^2}{D+2}\left(u_\alpha\delta_{\beta\gamma}+u_\beta\delta_{\alpha\gamma}+u_\gamma\delta_{\alpha\beta}\right)\right]\\ &= \left(\tau-\frac{1}{2}\right)\left[\frac{\partial}{\partial x_{1\alpha}}\frac{\partial}{\partial x_{1\alpha}}\frac{\rho c^2}{D+2}u_\gamma\right.\\ &\quad \left.+\frac{\partial}{\partial x_{1\gamma}}\left(\left(\frac{2c^2}{D+2}-\frac{(1-z)c^2}{D}\right)\frac{\partial}{\partial x_{1\alpha}}\rho u_\alpha\right)\right].\end{aligned} \tag{3.30}$$

By setting the kinematic shear viscosity $\nu = \left(\tau-\frac{1}{2}\right)\frac{c^2}{D+2}$ and the kinematic bulk viscosity $\varsigma = \left(\tau-\frac{1}{2}\right)\left(\frac{2c^2}{D+2}-\frac{(1-z)c^2}{D}\right)$, the above equation becomes

$$\frac{\partial}{\partial t_2}\rho u_\gamma = \nu\frac{\partial^2}{\partial x_{1\alpha}^2}\rho u_\gamma + \frac{\partial}{\partial x_{1\gamma}}\left(\varsigma\frac{\partial}{\partial x_{1\alpha}}\rho u_\alpha\right). \tag{3.31}$$

Using all these equations (provided above), one can show that the Navier-Stokes equation

$$\frac{\partial\rho}{\partial t}u_\alpha + \frac{\partial}{\partial x_\beta}\rho u_\beta u_\alpha = -\frac{\partial}{\partial x_\beta}\left[\frac{\rho(1-z)c^2}{D}\delta_{\alpha\beta}\right] + \nu\frac{\partial^2}{\partial x_\beta^2}\rho u_\alpha + \frac{\partial}{\partial x_\alpha}\left(\varsigma\frac{\partial}{\partial x_\beta}\rho u_\beta\right) \tag{3.32}$$

and the continuity equation

$$\frac{\partial\rho}{\partial t} + \frac{\partial}{\partial x_\alpha}\rho u_\alpha = 0 \tag{3.33}$$

are satisfied. For an incompressible flow, these two equations reduce to equation (1.1) and (1.2).

3.3 Boundary and initial conditions

The bounce back scheme is still good for the no-slip boundary condition. Bounce back boundary conditions only give first-order accuracy. A Dirichlet boundary condition can be achieved by solving the system of equations at the boundary sites

$$\begin{aligned} \frac{1}{2}f_1 + f_2 + \frac{1}{2}f_3 - \frac{1}{2}f_4 - f_5 - \frac{1}{2}f_6 &= \rho u_x \\ \frac{\sqrt{3}}{2}(f_1 - f_3 - f_4 + f_6) &= \rho u_y \end{aligned} \tag{3.34}$$

where $\vec{u} = (u_x, u_y)$ is the velocity vector. For example, in the simulation of a driven cavity flow, assume the top boundary has the speed $\vec{u} = (speed, 0)$. The f_1, f_6 are from the inside sites, and we can keep f_2, f_5 as constants. Only f_3, f_4 are unknowns. This is a linear system of equations involving two unknowns. Notice that when $\vec{u} = (0, 0)$, it is a bounce back scheme.

The left boundary shown in Figure 3.1 is not smooth in a microscopic view because of the hexagonal structure of the grid. But it is smooth enough in a macroscopic view. One can take the macroscopic boundary as the average of this boundary.

For initial conditions, one may use the equilibrium distribution from the given values of ρ and $\vec{u}$. Bad initial distribution, for example, far away from the equilibrium distribution, will make the model unstable, and eventually lead to blow up.

3.4 Implementation

Let us do a driven cavity flow again. This time, an array of 7 floating-point variable is needed for the information at one site. So we create a $T \times 7$ matrix M, where T is

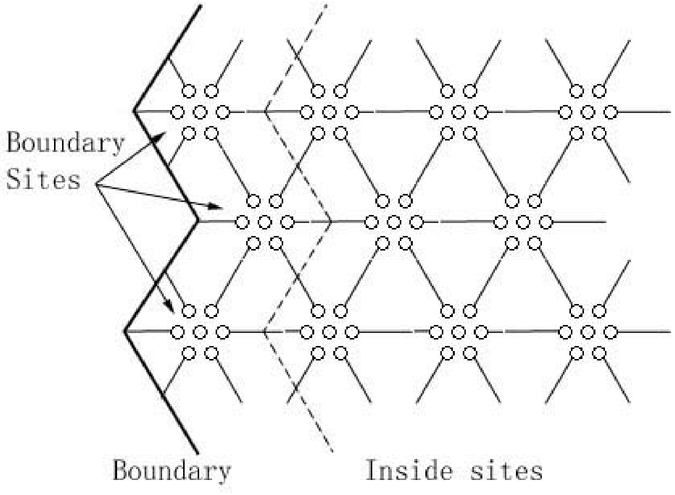

Figure 3.1: Left boundary of a hexagonal grid.

the total number of sites. We number the sites the same way as in FHP. The program structure is also pretty much the same as in FHP, except in the LBGK model, the collision and the streaming are combined together by the Lattice Boltzmann equation. The equilibrium distribution function is a mapping from the given ρ and $\vec{u}$ to the matrix M. So first we can use this to initialize M. In the collision, we calculate the ρ and $\vec{u}$ by

$$\begin{aligned} \rho(\vec{x},t) &= \sum_i M(n,i) \\ u_x(\vec{x},t) &= \vec{M}(n)\cdot\vec{e}_{ix} \\ u_y(\vec{x},t) &= \vec{M}(n)\cdot\vec{e}_{iy} \end{aligned} \tag{3.35}$$

where n is the number of the site corresponding to $\vec{x}$, and $\vec{e_i} = (\sin(\frac{\pi}{3}(i-1)), \cos(\frac{\pi}{3}(i-1)))$, which is the link to the 6 neighbour sites. Then apply the weighted equilibrium distribution function (equation 3.3) with a proper τ.

3.5 Results and data analysis

3.5.1 Driven cavity

Here we present a driven cavity example again. In figures 3.2, 3.3, 3.4, 3.5, and 3.7, we give the velocity vectors (left) and velocity contour (right) at the steady state for Reynolds number 10, 100, 200, 400 and 800. We give the result of Ghia, Ghia, and Shin [12] for Reynolds number at 400 for comparison (Their computations were performed using the time-marching capabilities of WIND to approach the *ste*ady-state flow starting from the freestream conditions). We also give the velocity profiles for u and v through the geometirc center of the cavity. For comparison, refer to Shuling Hou and Qisu Zou *et al* [13].

3.5.2 Poiseuille flow

Here we also consider a Poiseuille flow example. The analytic solution is given by

$$U_x(y) = \frac{G}{2\mu} y(d-y)$$

where the G is a constant pressure gradient which represents a uniform body force in the direction of the positve x-direction, d is the width of the channel. The grid size is 120×1000. A uniform flow comes from the left and goes out on the right. The top and

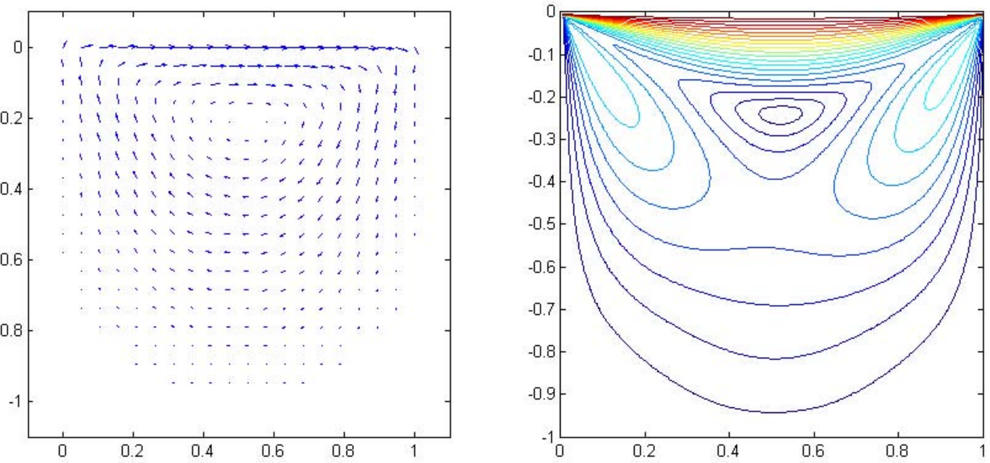

Figure 3.2: Driven cavity at $Re = 10$.

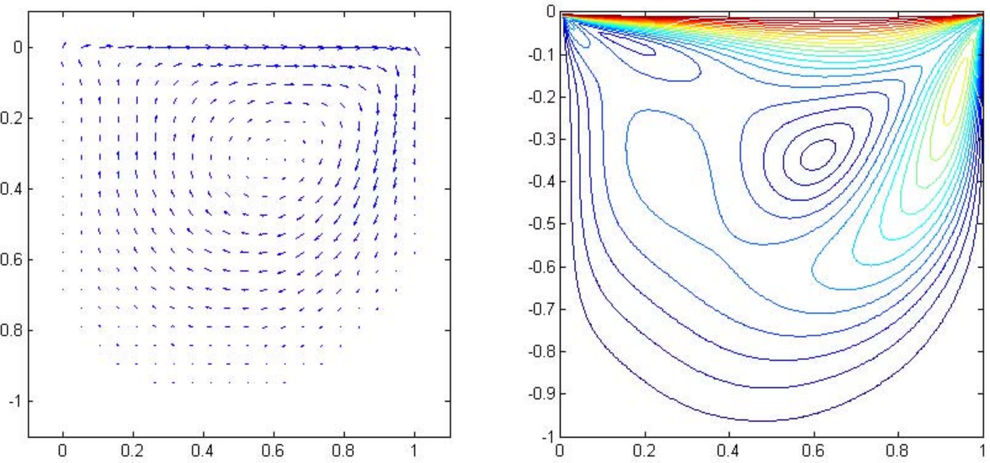

Figure 3.3: Driven cavity at $Re = 100$.

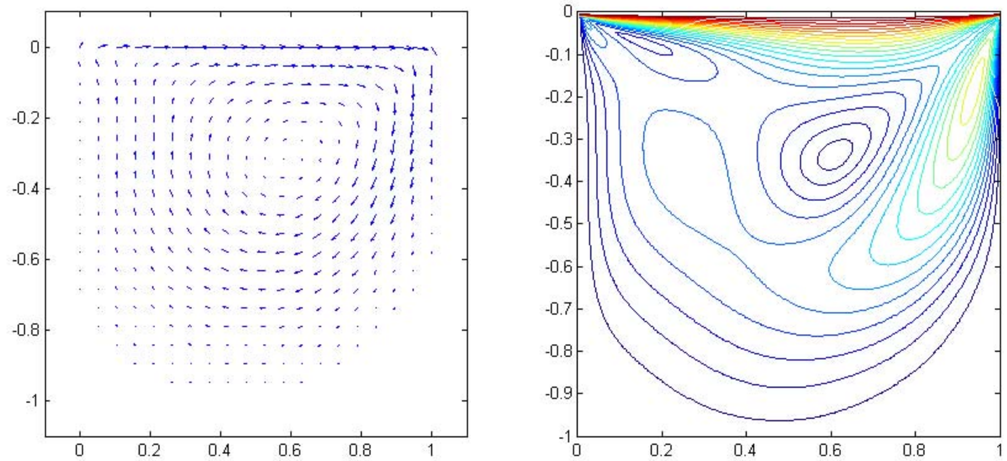

Figure 3.4: Driven cavity at $Re = 200$.

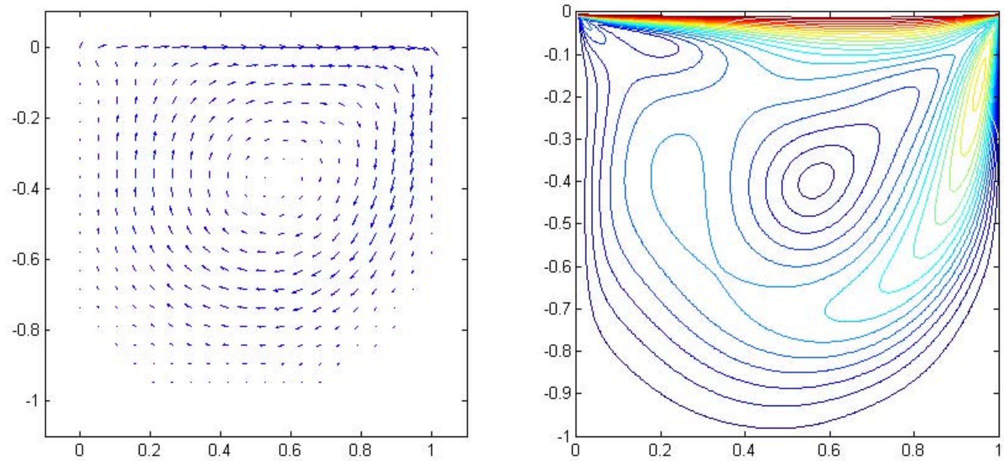

Figure 3.5: Driven cavity at $Re = 400$.

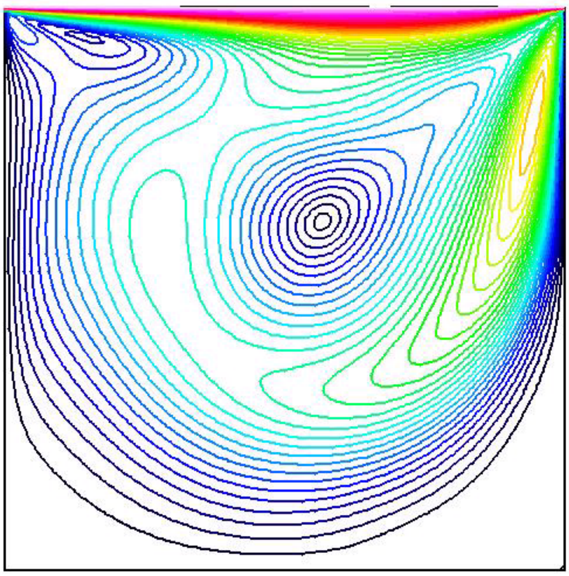

Figure 3.6: Ghia-Ghia-Shin's [12] result. The plot of the velocity contour with a Reynolds number of 400.

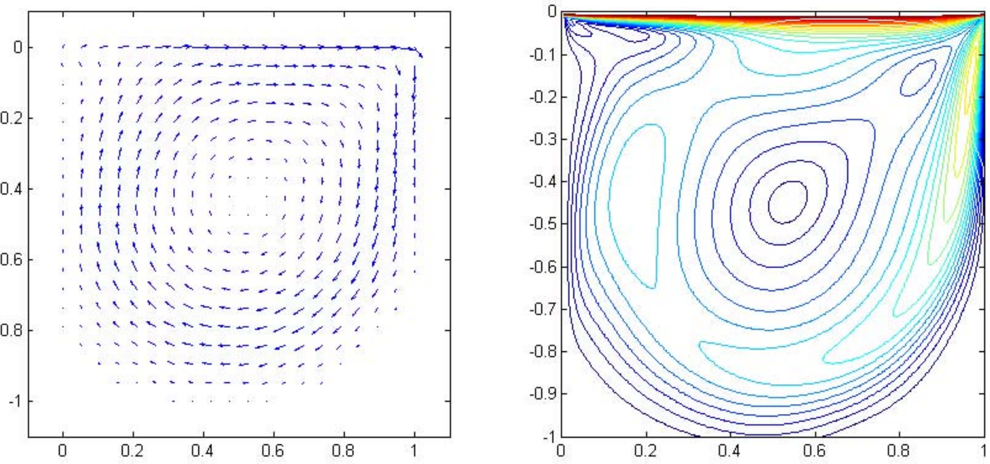

Figure 3.7: Driven cavity at $Re = 800$.

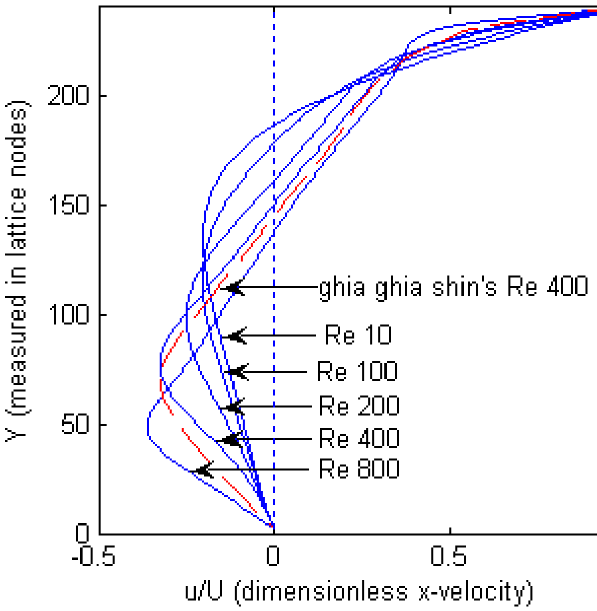

Figure 3.8: Dimensionless x-velocity profile at the geometry center of the cavity for Reynolds number 10, 100, 200, 400, 800. Dashed line is the result from Ghia Ghia Shin [12] at Reynolds number 400.

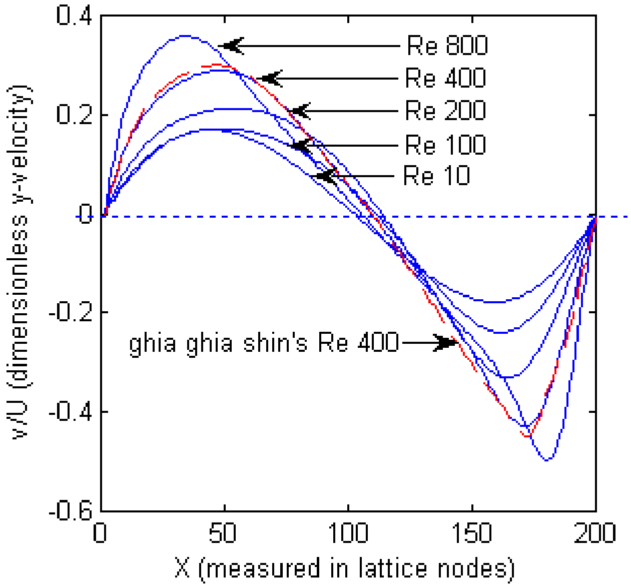

Figure 3.9: Dimensionless y-velocity profile at the geometry center of the cavity for Reynolds number 10, 100, 200, 400, 800. Dashed line is the result from Ghia Ghia Shin [12] at Reynolds number 400.

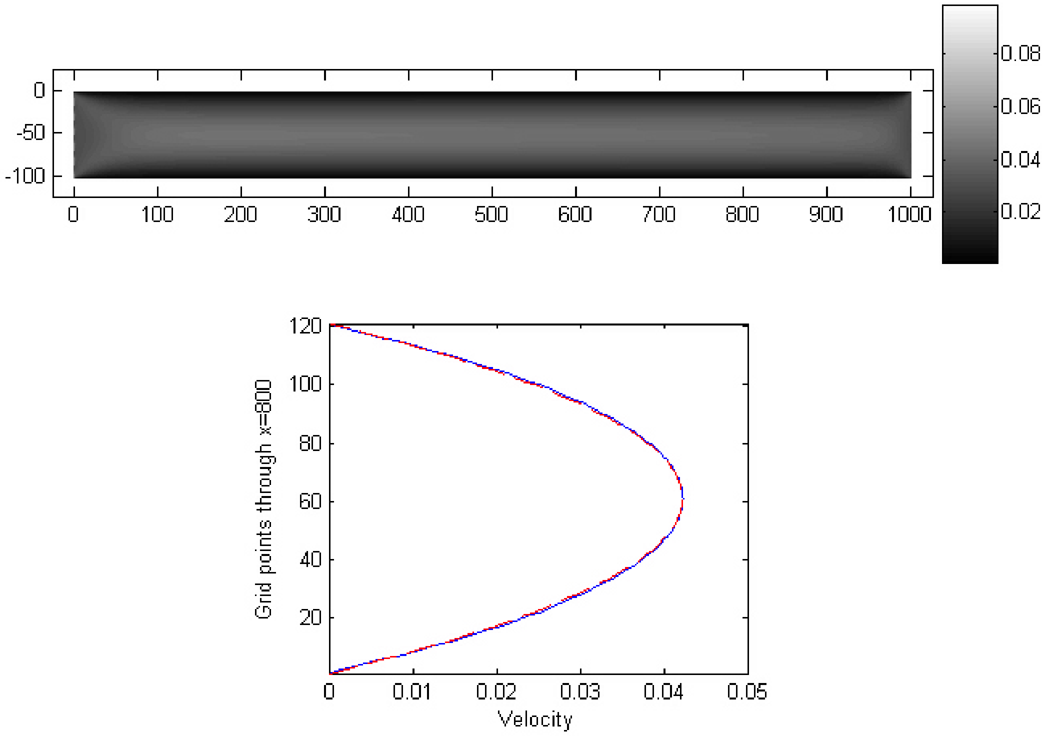

Figure 3.10: An example of a Couette flow. Above: velocity contour. Below: x-velocity profile at $x = 800$. Solid line is the simulation, and dashed line is the actual solution.

bottom are no-slip boundaries. We give the x-direction velocity profile at $x = 800$. From figure 3.10 one see that it is a parabolic profile.

3.5.3 Flow past a cylinder

This is an example of a uniform flow past a cylinder. This example is done on a 360×1000 grid. The speed of the uniform flow coming from the left is 0.5. The cylinder was placed in the center of the left with a diameter 120. The Reynolds number is 400. The top and bottom are no-slip boundaries. As is well known that with a Reynolds

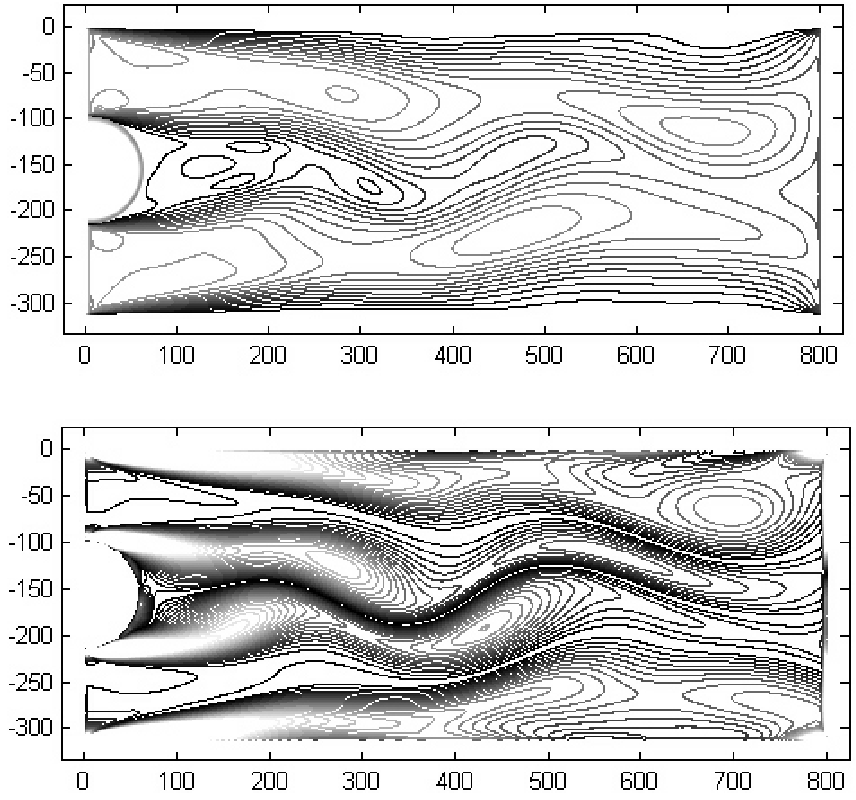

Figure 3.11: An example of a uniform flow past a cylinder for $Re = 400$. Above: velocity contour. Below: vorticity contour.

number greater than 100, the flow past a cylinder will give a Von Karman vortex street. Here we give the figures of both velocity contour and vorticity contour at time step 5000. Both figures show the back half of the cylinder.

Chapter 4

Conclusions and future works

We can see from the implementations of the Lattice Gas model and Lattice Boltzmann model that both of these models can successfully simulate complex fluid flows. From the structure of the FHP model, we know that it is unconditionally stable, while it is very noisy. Large scale calculation is required in order to reduce the noise, which makes it slower than the Lattice BGK model. Other drawbacks are lack of Galilean invariance, and it is not good for multi-dimensional simulation. The Lattice Boltzmann model is a better model. It is fast and it gives good simulations. Also, thanks to the structure of this model, it is very easy to parallelize. One can divide the whole domain into smaller subdomains; the only information that goes in between those subdomains are the updating of information on the boundaries of those subdomains, which is a very small amount compared to the inside part.

So one of the future works is to do a 3-D model and parallelize it. Later I will try to do MagnetoHydroDynamic (MHD) flow which involves simulating the current using the Lattice Boltzmann model.

BIBLIOGRAPHY

[1] D. A. Wolf-Gladrow, *Lattice-Gas Cellular Automata and Lattice Boltzmann Models*, An Introduction, Springer-Verlag Berlin Heidelberg 2000. ISSN 0075-8434 ISBN 3-540-66973-6

[2] J. Hardy, Y. Pomeau & O. de Pazzis, *Time evolution of two-dimensional model system.* I. Invariant states and time correlation functions, J. Math. Phys. 14 (1973), pp1746-1759.

[3] U. Frisch, B. Hasslacher, and Y. Pomeau. *Lattice-gas automata for the Navier-Stokes equation.* Physical Review Letters, 56:pp1505-1508, 1986.

[4] G. R. McNamara and G. Zanetti. *Use of the Boltzmann equation to simulate lattice-gas automata.* Physical Review Letters, 61:pp2332-2335, 1988.

[5] F. J. Higuera and J. Liménez. *Boltzmann approach to lattice gas simulations.* Europhysics Letters, 9 (7):pp663-668, 1989.

[6] F. J. Higuera, S. Succi, and R. Benzi. *Lattice gas dynamics with enhanced collisions.* Europhysics Letters, 9 (4):pp345-349, 1989.

[7] P. L. Bhatnagar, E. P. Gross, and M Krook. *A model for collision processes in gases.* I. Small amplitude processes in charged and neutral one-component systems. Physical Review, 94 (3):pp511-525, 1954.

[8] J. Buick, W. Easson, and C. Greated. *Simulation of wave motion using a lattice gas model.* International Journal for Numberical Methods in Fluids, 22:pp313-321, 1996.

[9] Sauro Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*, Clarendon Press, Oxford, 2001. ISBN 0-19-850398-9

[10] D. R. Noble, S. Chen, J. G. Georgiadis, and R. O. Buckius. *A consistent hydrodynamic boundary condition for the lattice Boltzmann method.* Physics of Fluids, 7 (1):pp203-209, 1995.

[11] S. Wolfram. *Cellular automaton fluids* 1: Basic theory. Journal of Statistical Physics, 45(3/4): pp471-529, 1986.

[12] U. Ghia, K. N. Ghia, and CT Shin, *High Resolutions for incompressible flow using the navier-stokes equations and a multigrid method*, Journal of Computational Physics, 48:pp387-411, 1982.

[13] S. Hou and Q. Zou, S. Chen, G. Doolen, A. C. Cogley. *Simulation of Cavity Flow by the Lattice Boltzmann Method*. Journal of Computational Physics 118, pp329-347 1995.

Appendices

.1 FHP collision look-up table

This is the collision look-up table. It gives the two out-states corresponding to each of the 128 in-states (right in bracket). Refer to 2.1.1 Data structure to find the meaning of the number of the in-states and out-states.

0	[0 0]	16	[16 16]	32	[32 32]	48	[48 48]
1	[1 1]	17	[96 96]	33	[33 33]	49	[49 49]
2	[2 2]	18	[9 36]	34	[65 65]	50	[41 81]
3	[3 3]	19	[98 37]	35	[35 35]	51	[51 51]
4	[4 4]	20	[72 72]	36	[18 9]	52	[104 25]
5	[66 66]	21	[42 42]	37	[19 98]	53	[105 114]
6	[6 6]	22	[13 74]	38	[69 11]	54	[27 45]
7	[7 7]	23	[102 75]	39	[39 39]	55	[107 107]
8	[8 8]	24	[24 24]	40	[80 80]	56	[56 56]
9	[36 18]	25	[52 104]	41	[81 50]	57	[57 57]
10	[68 68]	26	[84 44]	42	[21 21]	58	[116 89]
11	[38 69]	27	[45 54]	43	[83 101]	59	[117 117]
12	[12 12]	28	[28 28]	44	[26 84]	60	[60 60]
13	[74 22]	29	[90 108]	45	[54 27]	61	[122 122]
14	[14 14]	30	[30 30]	46	[77 86]	62	[93 93]
15	[15 15]	31	[110 110]	47	[87 87]	63	[63 63]

64	[64 64]
65	[34 34]
66	[5 5]
67	[67 67]
68	[10 10]
69	[11 38]
70	[70 70]
71	[71 71]
72	[20 20]
73	[100 82]
74	[22 13]
75	[23 102]
76	[76 76]
77	[86 46]
78	[78 78]
79	[79 79]
80	[40 40]
81	[50 41]
82	[73 100]
83	[101 43]
84	[44 26]
85	[106 106]
86	[46 77]
87	[47 47]
88	[88 88]
89	[58 116]
90	[108 29]
91	[109 118]
92	[92 92]
93	[62 62]
94	[94 94]
95	[95 95]
96	[17 17]
97	[97 97]
98	[37 19]
99	[99 99]
100	[82 73]
101	[43 83]
102	[75 23]
103	[103 103]
104	[25 52]
105	[114 53]
106	[85 85]
107	[55 55]
108	[29 90]
109	[118 91]
110	[31 31]
111	[111 111]
112	[112 112]
113	[113 113]
114	[53 105]
115	[115 115]
116	[89 58]
117	[59 59]
118	[91 109]
119	[119 119]
120	[120 120]
121	[121 121]
122	[61 61]
123	[123 123]
124	[124 124]
125	[125 125]
126	[126 126]
127	[127 127]

.2 Partial Matlab code I: FHP streaming of a driven cavity

```
function StreamingDrivenCavity


global speed;
global M;

global T;

global n;

global m_odd;

global m_even;

global k;


% Bounce back

% Top

% 1 reflect to 3

temp = bitand(M(1:n+1),1);

M(1:n+1) = M(1:n+1) + temp*(4 - 1);

% 6 reflect to 4

temp = bitand(M(1:n+1),32)/32;

M(1:n+1) = M(1:n+1) + temp*(8 - 32);

% Top-right corner

temp = bitand(M(n+1), 4)/4;

M(n+1) = M(n+1) + temp*(8 - 4);

for i = 1:2*n+1:T

    % Left bounce back

    temp = bitand(M(i), 32)/32;

    M(i) = M(i) + temp*(4 - 32);

    temp = bitand(M(i), 16)/16;
    M(i) = M(i) + temp*(2 - 16);
    temp = bitand(M(i), 8)/8;
    M(i) = M(i) + temp*(1 - 8);
    % Right bounce back
    temp = bitand(M(i+n), 1);
    M(i+n) = M(i+n) + temp*(8 - 1);
    temp = bitand(M(i+n), 2)/2;
    M(i+n) = M(i+n) + temp*(16 - 2);
    temp = bitand(M(i+n), 4)/4;
    M(i+n) = M(i+n) + temp*(32 - 4);
end
% Bottom bounce back
temp = bitand(M(T-n:T), 8)/8;
M(T-n:T) = M(T-n:T) + temp*(1 - 8);
temp = bitand(M(T-n:T), 4)/4;
M(T-n:T) = M(T-n:T) + temp*(32 - 4);

% Reset boundary condition on top
%M(1:n+1) = M(1:n+1) - bitand(M(1:n+1),64) + 64;
M(1:n+1) = bitor(M(1:n+1),64);
M(1:n+1) = bitand(M(1:n+1),76);
M(1:speed:n+1) = M(1:speed:n+1) - bitand(M(1:speed:n+1), 64);
M(1:speed:n+1) = M(1:speed:n+1) - bitand(M(1:speed:n+1), 2) + 2;
```

```
% Streaming

% Streaming on direction 3 & 6

temp1 = bitand(M(1:T-n-1), 4)/4; % 3

temp2 = bitand(M(n+2:T), 32)/32; % 6

M(1:T-n-1) = M(1:T-n-1) - temp1*4;

M(n+2:T) = M(n+2:T) - temp2*32;

M(1:T-n-1) = M(1:T-n-1) + temp2*4;

M(n+2:T) = M(n+2:T) + temp1*32;

% Streaming on direction 1 & 4

temp1 = bitand(M(n+2:T-1), 1); % 1

temp2 = bitand(M(2:T-n-1), 8)/8; % 4

M(n+2:T-1) = M(n+2:T-1) - temp1;

M(2:T-n-1) = M(2:T-n-1) - temp2*8;

M(n+2:T-1) = M(n+2:T-1) + temp2;

M(2:T-n-1) = M(2:T-n-1) + temp1*8;

% Streaming on direction 2 & 5

temp1 = bitand(M(n+2:T-n-2), 2)/2; % 2

temp2 = bitand(M(n+3:T-n-1), 16)/16; % 5

M(n+2:T-n-2) = M(n+2:T-n-2) - temp1*2;

M(n+3:T-n-1) = M(n+3:T-n-1) - temp2*16;

M(n+2:T-n-2) = M(n+2:T-n-2) + temp2*2;

M(n+3:T-n-1) = M(n+3:T-n-1) + temp1*16;
```

```
for i = 1:m_even-1

    temp1 = bitand(M(i*(2*n+1)), 2)/2;

    temp2 = bitand(M(i*(2*n+1)+1), 16)/16;

    M(i*(2*n+1)) = M(i*(2*n+1)) - temp1*2;

    M(i*(2*n+1)) = M(i*(2*n+1)) + temp2*2;

    M(i*(2*n+1)+1) = M(i*(2*n+1)+1) - temp2*16;

    M(i*(2*n+1)+1) = M(i*(2*n+1)+1) + temp1*16;

    temp1 = bitand(M(i*(2*n+1)+n+1), 2)/2;

    temp2 = bitand(M(i*(2*n+1)+n+2), 16)/16;

    M(i*(2*n+1)+n+1) = M(i*(2*n+1)+n+1) - temp1*2;

    M(i*(2*n+1)+n+1) = M(i*(2*n+1)+n+1) + temp2*2;

    M(i*(2*n+1)+n+2) = M(i*(2*n+1)+n+2) - temp2*16;

    M(i*(2*n+1)+n+2) = M(i*(2*n+1)+n+2) + temp1*16;

end


% Interchange 14 25 36

temp1 = bitand(M(1:T), 1);

temp2 = bitand(M(1:T), 8)/8;

M(1:T) = M(1:T) - temp1 - temp2*8 + temp1*8 + temp2;

temp1 = bitand(M(n+2:T), 2)/2;

temp2 = bitand(M(n+2:T), 16)/16;

M(n+2:T) = M(n+2:T) - temp1*2 - temp2*16 + temp1*16 + temp2*2;

temp1 = bitand(M(1:T), 4)/4;
```

```
temp2 = bitand(M(1:T), 32)/32;

M(1:T) = M(1:T) - temp1*4 - temp2*32 + temp1*32 + temp2*4;

bigskip
```

.3 Partial code II: Lattice BGK initialization of a driven cavity

```
function LBM_InitializeDrivenCavity(m,n)


global speed;

global m_odd;

m_odd = fix(m/2) + 1

global m_even;

m_even = m + 1 - m_odd

global T;

T = m_odd*(n+1) + m_even*n;

global M;

M = zeros(T,7);

global E;

E = M;

global M_temp;

M_temp = M;

global lx;
global ly;

global e;

global c;

global i_top;

i_top = 2:n;

global i_inside; % The indices of the inside sites

i_inside = [];

for i = 1:m_even-1

i_inside = [i_inside i*(2*n+1)-n+1:i*(2*n+1)];

i_inside = [i_inside i*(2*n+1)+2:i*(2*n+1)+n];

end

i_inside = [i_inside T-2*n:T-n-1];

global i_left1; % The indices of the left boundary, except the two corners.

i_left1 = 2*n+2:2*n+1:T-2*n;

global i_left2;

i_left2 = n+2:2*n+1:T-2*n;

global i_right1; % The indices of the right boundary, except the two corners.

i_right1 = 2*n+2+n:2*n+1:T-n;

global i_right2;

i_right2 = 2*n+1:2*n+1:T-n-1;

global i_25; % The indices of the inside sites for 2 & 5 directions.

i_25 = [];

for i = 1:m_even-1
```

```
i_25 = [i_25 i*(2*n+1)-n+2:i*(2*n+1)-1];

i_25 = [i_25 i*(2*n+1)+2:i*(2*n+1)+n];

end

i_25 = [i_25 T-2*n+1:T-n-2];

global top_eq;


% Calculate equilibrium distribution as the initial condition given by p, u

p(1:T,:) = 1;

u = zeros(T,2);

u(1:n+1,1) = speed;

t = find(p);

u(t,:) = u(t,:)./[p(t) p(t)];

u_square = u(:,1).^2 + u(:,2).^2;

d0 = 1/2;

for i = 1:6

E(:,i) = p.*((1-d0)/6 + 1/3*(e(i,:)*u')'/c^2 + 2/3*(e(i,:)*u')'.%

^2/c^4 - 1/6*u_square/c^2);

end

E(:,7) = p.*(d0 - u_square/c^2);

%sum(sum(E)')/(T-n-1)

%LBM_Visualize(M)

% Do not calculate E on the bottom, left and right

E(T-n:T,:) = 0; % This result in pure streaming from the bottom boundary to
```

```
the inside.

E(i_left1,:) = 0;

E(i_right1,:) = 0;

M = E;

top_eq = M(1:n+1,:);
```

.4 Partial code III: Lattice BGK collision of a driven cavity

```
function LBM_CollisionDrivenCavity


global speed;

    speed = 0.1;

global n;

global m_odd;

    m_odd = fix(m/2) + 1;

global m_even;

    m_even = m + 1 - m_odd

global T;

    T = m_odd*(n+1) + m_even*n;

global M;

    M = zeros(T,7);

global E;
```

```
    E = M;
global M_temp;
    M_temp = M;
global ly;
    ly = sqrt(3)/2;
global D; % Dimension
    D = 2;
global b; % # of directions
    b = 6;
global d;
    d = 1/2;
global c; % Unit speed
global p; % Density
global u; % Speed
global e;
    e = [[0.5 ly];[1 0];[0.5 -ly];[-0.5 -ly];[-1 0];[-0.5 ly]];
    e = c*e;
global i_inside;
global i_top;
global i_left1;
global i_left2;
global i_right1;
global i_right2;
global i_25;

global top_eq;


% Calculate equilibrium distribution

p = (sum(M'))';

u = [0.5*M(:,1)+M(:,2)+0.5*M(:,3)-0.5*M(:,4)-M(:,5)-0.5*M(:,6),

ly*M(:,1)-ly*M(:,3)-ly*M(:,4)+ly*M(:,6)];

t = find(p);

u(t,:) = u(t,:)./[p(t) p(t)];

u = c*u;

u_square = u(:,1).^2 + u(:,2).^2;

d0 = 1/2;

for i = 1:6

E(:,i) = p.*((1-d0)/6 + 1/3*(e(i,:)*u')'/c^2 + 2/3*(e(i,:)*u')'.%

^2/c^4 - 1/6*u_square/c^2);

end

E(:,7) = p.*(d0 - u_square/c^2);

E(T-n:T,:) = M(T-n:T,:); % This result in pure streaming from the bottom

boundary to the inside.

E(i_left1,:) = M(i_left1,:);

E(i_right1,:) = M(i_right1,:);

E(1,:) = M(1,:);

E(n+1,:) = M(n+1,:);
```

```
E(1:n+1,:) = M(1:n+1,:);

% Collision operation
M_temp = M;
M(n+2:T,:) = 0;
tao = 1;
w = 1/tao;
% rest particle
M(:,7) = (1-w)*M_temp(:,7) + w*E(:,7);
% 3 direction
M(i_inside,3) = (1-w)*M_temp(i_inside-n-1,3) + w*E(i_inside-n-1,3);
M(i_right1,3) = (1-w)*M_temp(i_right1-n-1,3) + w*E(i_right1-n-1,3);
M(T-n+1:T,3) = (1-w)*M_temp(T-2*n:T-n-1,3) + w*E(T-2*n:T-n-1,3); % Bottom boundary
% 6 direction
M(i_inside,6) = (1-w)*M_temp(i_inside+n+1,6) + w*E(i_inside+n+1,6);
M(i_left1,6) = (1-w)*M_temp(i_left1+n+1,6) + w*E(i_left1+n+1,6);
M(1:n,6) = (1-w)*M_temp(n+2:2*n+1,6) + w*E(n+2:2*n+1,6); % Top boundary
% 1 direction
M(i_inside,1) = (1-w)*M_temp(i_inside+n,1) + w*E(i_inside+n,1);
M(i_right1,1) = (1-w)*M_temp(i_right1+n,1) + w*E(i_right1+n,1);
M(2:n+1,1) = (1-w)*M_temp(n+2:2*n+1,1) + w*E(n+2:2*n+1,1); % Top boundary
% 4 direction
```

```
M(i_inside,4) = (1-w)*M_temp(i_inside-n,4) + w*E(i_inside-n,4);
M(i_left1,4) = (1-w)*M_temp(i_left1-n,4) + w*E(i_left1-n,4);
M(T-n:T-1,4) = (1-w)*M_temp(T-2*n:T-n-1,4) + w*E(T-2*n:T-n-1,4); % Bottom
boundary
% 2 direction
M(i_25,2) = (1-w)*M_temp(i_25-1,2) + w*E(i_25-1,2);
M(i_right1,2) = (1-w)*M_temp(i_right1-1,2) + w*E(i_right1-1,2);
M(i_right2,2) = (1-w)*M_temp(i_right2-1,2) + w*E(i_right2-1,2);
M(i_left2,2) = (1-w)*M_temp(i_left2,5) + w*E(i_left2,5);
% 5 direction
M(i_25,5) = (1-w)*M_temp(i_25+1,5) + w*E(i_25+1,5);
M(i_left1,5) = (1-w)*M_temp(i_left1+1,5) + w*E(i_left1+1,5);
M(i_left2,5) = (1-w)*M_temp(i_left2+1,5) + w*E(i_left2+1,5);
M(i_right2,5) = (1-w)*M_temp(i_right2,2) + w*E(i_right2,2);
M(1:n+1,3) =
speed*(2*M(1:n+1,1)+M(1:n+1,2)+M(1:n+1,5)+2*M(1:n+1,6)+M(1:n+1,7)) -
M(1:n+1,2) + M(1:n+1,5) + M(1:n+1,6);
M(1:n+1,4) =
-speed*(2*M(1:n+1,1)+M(1:n+1,2)+M(1:n+1,5)+2*M(1:n+1,6)+M(1:n+1,7)) +
M(1:n+1,1) + M(1:n+1,2) - M(1:n+1,5);

% Bottom Boundary layer
M(T-2*n:T-n-1,1) = M(T-n:T-1,4);
```

```
M(T-n:T-1,4) = 0;

M(T-2*n:T-n-1,6) = M(T-n+1:T,3);

M(T-n+1:T,3) = 0;

% Left Boundary

M(i_left1-n,1) = M(i_left1,4);

M(i_left1+1,2) = M(i_left1,5);

M(i_left1+n+1,3) = M(i_left1,6);

M(i_left1,:) = 0;

% Right Boundary

M(i_right1+n,4) = M(i_right1,1);

M(i_right1-1,5) = M(i_right1,2);

M(i_right1-n-1,6) = M(i_right1,3);

M(i_right1,:) = 0;
```