# EVALUATION OF TWO DIFFERENT TYPES D2QA MICROSCOPIC VELOCITY MODELS AT LOW RAYLEIGH NUMBER HEAT TRANSFER

PYLLISCIA SUMBOK AK ADAN

BACHELOR OF MECHANICAL ENGINEERING
UNIVERSITI MALAYSIA PAHANG

2009

## SUPERVISOR'S DECLARATION

I hereby declare that I have checked this project and in my opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Mechanical Engineering.

| | |
|---|---|
| Signature | |
| Name of Supervisor: | MUHAMAD ZUHAIRI BIN SULAIMAN |
| Position: | LECTURER |
| Date: | 21 NOVEMBER 2009 |

## STUDENT'S DECLARATION

I hereby declare that the work in this project is my own except for quotations and summaries which have been duly acknowledged. The project has not been accepted for any degree and is not concurently submitted for award of other degree.

| | |
|---|---|
| Signature: | |
| Name: | PYLLISCIA SUMBOK AK ADAN |
| ID Number: | MA06077 |
| Date: | 21 NOVEMBER 2009 |

## ACKNOWLEDGEMENTS

I am grateful for finally completed this thesis and would like to take this opportunity to express my sincere gratitude to many people that had supported and accompanied me throughout my thesis work. This thesis had become easier to completed as they had been together with me along my ups and down.

First, to my former supervisor En. Mohd Rosdzimin Abdul Rahman, for his endless support, ideas, continuous motivation and invaluable guidance, in making this research possible. He has a really thoughtful mind and his kindness for teaching and providing me with relevance information until I fully understand my thesis was very appreciated. I also appreciate him for believing me and achnowledging the importance of these thesis.

My sincere gratitude also goes to my second supervisor, En. Muhamad Zuhairi bin Sulaiman who had guide me on completing this thesis right in time. Thanks to his effort for sharing with me the incredibly time-consuming process for creating a thesis. Thanks again for giving me the space, time and emotional support I needed to follow my passion and complete what must have seemed like an endless task. I also sincerely thanks him for the time spent for reading and correcting my many mistakes.

My sincere thanks also go to all my friends for giving me extensive feedback and invalueable suggestion for improving the first presentation and draft of this thesis. I deeply appreciate their incrediable ability to take everything the roughest of me and make it all work. Thanks for the way you make a significant different.

I acknowledge my sincere indebtedness and gratitude to my parents for always being there on the phone for countless hours when I needed support and advice about everything. Your dedication and hard support had never cease to fill my heart with gratitude. Thanks again for being so lovely and caring.

Thank you all for your vision, caring, commitment and heartful action.

## ABSTRACT

The aim of this thesis is to study the methods of the lattice Boltzmann equations in order to be apply in two types of D2Q4 model in thermal fluid flow problems. LBM has been found to be useful in application involving interfacial dynamics and complex boundaries. These methods utilize the statistical mechanics of simple discrete models to simulate complex physical systems. The theory of lattice Boltzmann method in nine and four velocity model are reviewed. The isothermal and thermal equation have been derived from the Boltzmann equation by descretiziton on both time and phase space. In this isothermal problem, a few simple isotherml flow simulation will be done by using the nine velocity model. The concepts of distribution function are considered beside the theory of Boltzmann equations. Then the derivations of Navier-Stokes equation from the Boltzmann equations are also presented. Some simulation results are performed, to highlight the important features of the isothermal LB model. The application of lattice Boltzmann scheme in thermal fluid problem is investigated in chapter 3. By using the derivation of the discretised density distribution function, a 4-velocity model is applied to develop the internal energy distribution function. This model is validated to simulate the porous couette flow problem for thermal fluid flow problems. The performance for both types D2Q4 microscopic model is demonstrated in the simulations of porous thermal couette flow and natural convection flow in a square cavity. The simulation of thermal fluids flow is applied to two different types of four-velocity model that are Azwadi model and old model. The same simulation test is performed for both types and the accuracy and stability analysis of both models are stated. These models are compared and discussed to ensure its validity.

## ABSTRAK

Tujuan utama tesis ini adalah untuk mempelajari kaedah kekisi *Boltzmann* yang akan diaplikasikan pada dua jenis halaju model D2Q4 di dalam masalah terma. Kaedah kekisi *Boltzmann* telah pun diketahui berguna di dalam aplikasi yang membabitkan perhubungan di antara dinamik dan sempadan yang rumit. Algoritma ini adalah mudah dan dapat dilaksanakan pada intisari sesuatu model dengan mengunakan beberapa ratus garisan. Kaedah ini menggunakan mekanik statik pada modal berasingan yang mudah hingga kepada yang kompleks. Teori kekisi *Boltzmann* ini akan dikaji pada halaju model empat dan sembilan hala. Persamaan isoterma dan terma diperolehi daripada persamaan *Boltzmann* dengan mendekritasikan masa dan juga ruang fasa. Kemudian masalah isoterma akan menjadi fokus utama pada awal bab ketiga. Dalam masalah isoterma ini, beberapa simulasi mudah akan dijalankan dengan menggunakan halaju model sembilan. Konsep pengagihan fungsi akan di pertimbangkan disamping teori persamaan *Boltzmann*. Kemudian, terbitan persamaan *Navier Stokes* daripada persamaan *Boltzmann* akan turut diperkenalkan. Beberapa keputusan simulasi ditunjukkan bagi membuktikan kepentingan modal isoterma kaedah kekisi *Boltzmann*. Seterusnya aplikasi kekisi *Boltzmann* akan dikaji dalam bab ketiga. Dengan menggunakan menterbitan fungsi pengagihan tenaga, satu model empat halaju akan digunakan untuk menghasilkan fungsi pengagihan tenaga dalaman. Model ini sahih untuk simulasi masalah pengaliran *porous coutte* dalam masalah terma. Hasil prestasi kedua-dua model D2Q4 akan dilakukan pada simulasi pengaliran *porous coutte* dan pengaliran pemanasan semulajadi pada ruang segiempat. Simulasi ini akan dijalankan ada model lama dan juga model azwadi. Ujian simulasi yang sama akan dilakukan pada kedua-dua model dan analisis ketepatan serta kestabilan kedua-dua model akan dibincangkan. Model ini akan dinilai dan diuji untuk membezakan prestasi kedua-dua model.

# TABLE OF CONTENTS

## LIST OF TABLE

## LIST OF FIGURE

## *LIST OF SYMBOLS*

| | |
|---|---|
| f: | *D*ensity distribution function; |
| c | Microscopic velocity |
| $\Omega( f)$ | Collision integral. |
| $t$ | Time |
| $T$ | Temperature |
| $N$ | Kinematic viscosity |
| $A$ | Thermal diffusivity |
| $\beta$ | Thermal expansion coefficient |
| $T_C$ | Cold temperature |
| $T_H$ | Hot temperature |
| $u$ | Horizontal velocity |
| $\boldsymbol{u}$ | Velocity vector |
| $U$ | Horizontal velocity of top plate |
| $v$ | Vertical velocity |
| $V$ | Volume |
| $x$ | Space vector |
| $P$ | Pressure |
| $\tau_{f,g}$ | Time relaxation |
| $\upsilon$ | Shear viscosity |
| $\beta$ | Thermal expansion coefficient |
| $E$ | Internal energy |
| $\nu$ | kinematic viscosity |
| $\alpha$ | thermal diffusivity |
| $\mu$ | viscosity |
| $k$ | thermal conductivity |
| $\rho$ | Density |
| $T_\infty$ | Infinity temperature |
| $T_f$ | Film temperature |
| $T_s$ | Surface temperature |
| $A$ | Area of contact A |
| $\eta$ | Proportionally constant |
| $\chi$ | Thermal diffusivity |
| $\Omega$ | Collision operator |
| $T_m$ | Average temperature |
| $g$ | Acceleration due to gravity |
| $c$ | Microscopic velocity |
| $f^{eq}$ | Equilibrium distribution function |
| $f$ | Distribution function |
| $Pr$ | Prandtl number |
| $Ra$ | Rayleigh number |
| $Re$ | Reynolds number |

## LIST OF ABBREVIATIONS

| | |
|---|---|
| BGK | Bhatnagar Gross krook |
| CFD | Computational fluid dynamics |
| LBM | Lattice Boltzmann method |
| LGA | Lattice gas approach |
| D2Q4 | Two dimensional Four velocities model |
| D2Q9 | Two dimensional Nine velocities model |
| LGCA | Lattice gas cellular automata |
| PBC | Periodic Boundary condition |
| PDEs | Partial differential equations |

# CHAPTER 1

# INTRODUCTION

## 1.1 INTRODUCTION

### 1.1.1 Navier-Stokes equations

The Navier–Stokes equations describe the motion of fluid substances that is substances which can flow. These equations arise from applying Newton's second law to fluid motion, together with the assumption that the fluid stress is the sum of a diffusing viscous term (proportional to the gradient of velocity), plus a pressure term. The mathematical relationship governing equation fluid flow is the famous continuity equation and Navier Stokes equation given by. The Navier–Stokes equations dictate not position but rather velocity. A solution of the Navier–Stokes equations is called a velocity field or flow field, which is a description of the velocity of the fluid at a given point in space and time. Once the velocity field is solved for, other quantities of interest such as flow rate or drag force may be found (X.He and L.S.Luo). Some exact solutions to the Navier–Stokes equations exist. Examples of degenerate cases; with the non-linear terms in the Navier–Stokes equations equal to zero; are Poisuelle flow, Couette flow and the oscillatory Stokes boundary layer. But also more interesting examples, solutions to the full non-linear equations, exist; for example the Taylor–Green vortex. Note that the existence of these exact solutions does not imply they are stable: turbulence may develop at higher Reynolds numbers.

### 1.1.2. Lattice Boltzmann Method (LBM)

Lattice Boltzmann methods (LBM) is a class of computational fluid dynamics (CFD) methods for fluid simulation. Instead of solving the Navier–Stokes equations, the discrete Lattice Boltzmann (LB) equation is solved to simulate the flow of a Newtonian fluid with collision models such as Bhatnagar-Gross-Krook (BGK). LB scheme is a scheme evolved from the improvement of lattice gas automata and inherits some features from its precursor, the Lattice Gas Automata (LGA).

The main motivation for the transition from LGA to LBM was the desire to remove the statistical noise by replacing the Boolean particle number in a lattice direction with its ensemble average, the so-called density distribution function. Accompanying this replacement, the discrete collision rule is also replaced by a continuous function known as the collision operator. In the LBM development, an important simplification is to approximate the collision operator with the Bhatnagar-Gross-Krook (BGK) relaxation term. This lattice BGK (LBGK) model makes simulations more efficient and allows flexibility of the transport coefficients.

Lattice Boltzmann models can be operated on a number of different lattices, both cubic and triangular, and with or without rest particles in the discrete distribution function. A popular way of classifying the different methods by lattice is the DnQm scheme. Here "Dn" stands for "n dimensions" while "Qm" stands for "m speeds". For example, D3Q15 is a three-dimensional Lattice Boltzmann model on a cubic grid, with rest particles present.

Although LBM approach treats gases and liquids as systems consisting of individual particles, the primary goal of this approach is to build a bridge between the microscopic and macroscopic dynamics. It is by deriving macroscopic equations from microscopic dynamics by means of statistic, rather than to solve macroscopic equations.

## 1.2 PROBLEM BACKGROUND

The lattice Boltzmann method is an alternative approach to the finite difference, finite element, and finite volume techniques for solving Navier-Stokes equations. LB scheme is a scheme evolved from the improvement of lattice gas automata and inherits some features from its precursor, the LGA. The implementation of the Bhatnagar-Gross-Krook (BGK) approximation has been made for LB method to improve its computational efficiency. The algorithm of LBM is simple, and easily modified to allow for the application of other.

LBM originated from lattice gas automata (LGA) which is based on concepts from the kinetic theory of gases. Two dimensional, four velocity model is one of the most widely used today to modeling macroscopic flow phenomena. LGA views fluids as arrays of discrete particles living on a discrete lattice, evolving some interactive such as propagation and collision rules. Collision between the particles in D2Q4 model will occur, and the change in velocity of each particle including its performance will be found out.

### 1.2.1 Project Objective

To find out the performances for both types D2Q4 microscopic velocity models.

### 1.2.2 Project Scopes

The scopes of this project are limited to D1Q4 microscopic velocity model, at low Rayleigh number using heat transfer mechanism.

## 1.3 THESIS OUTLINE

The aim of this thesis is to study the methods of the lattice Boltzmann equations in order to apply in two types of D2Q4 model in thermal fluid flow problems. These methods utilize the statistical mechanics of simple discrete models to simulate complex physical systems. The theory of lattice Boltzmann method in nine and four velocity model are reviewed. Then the new concern here is in chapter 4. Two types of four velocity model are studied and will be evaluated at low Rayleigh number heat transfer. The performance for both types D2Q4 microscopic model is demonstrated in the simulations of porous thermal couette flow and natural convection flow in a square cavity. Comparison of both models will be analyzed as the final result.

In chapter 2, the isothermal fluid flows problem will be the main subject. The concepts of distribution function are considered beside the theory of Boltzmann equations. Then the derivations of Navier-Stokes equation from the Boltzmann equations are also presented. Some simulation results are performed, to highlight the important features of the isothermal LB model.

Then in chapter 3, the application of lattice Boltzmann scheme in thermal fluid problem is investigated. By using the derivation of the discretised density distribution function, a 4-velocity model is applied to develop the internal energy distribution function. This model is validating to simulate the porous couette flow problem for thermal fluid flow problems. The accuracy and stability analysis of the model are discussed.

In chapter 4, the simulation of thermal fluids flow is applied to two different types of four-velocity model. The same simulation test is performed for both types and the accuracy and stability analysis of both models are also discussed. These models are compared and tested to ensure its validity.

Finally in chapter 5, conclusions and discussion on future studies are presented.

# CHAPTER 2

# LITERATURE STUDY

## 2.1 LATTICE BOLTZMANN METHOD

The lattice Boltzmann method is a powerful technique for the computational modeling of a wide variety of complex fluid flow problems including single and multiphase flow in complex geometries. It is a discrete computational method based upon the Boltzmann equation. It considers a typical volume element of fluid to be composed of a collection of particles that are represented by a particle velocity distribution function for each fluid component at each grid point (X.HE, 1997). The time is counted in discrete time steps and the fluid particles can collide with each other as they move, possibly under applied forces. The rules governing the collisions are designed such that the time-average motion of the particles is consistent with the Navier-Stokes equation.

This method naturally accommodates a variety of boundary conditions such as the pressure drop across the interface between two fluids and wetting effects at a fluid-solid interface. It is an approach that bridges microscopic phenomena with the continuum macroscopic equations. Further, it can model the time evolution of systems. Lattice Boltzmann Method can be reviewed as a numerical method to solve the Boltzmann equation. In LB method, the phase space is discretized. In a LB model, the velocity of a particle can only be chosen from a velocity set, which has only a finite number of velocities.

The Lattice Boltzmann Equation (LBE) method is described for simulating micro- and meso-scale phenomena. The method is employed to study multiphase and multicomponent flows in microchannels.

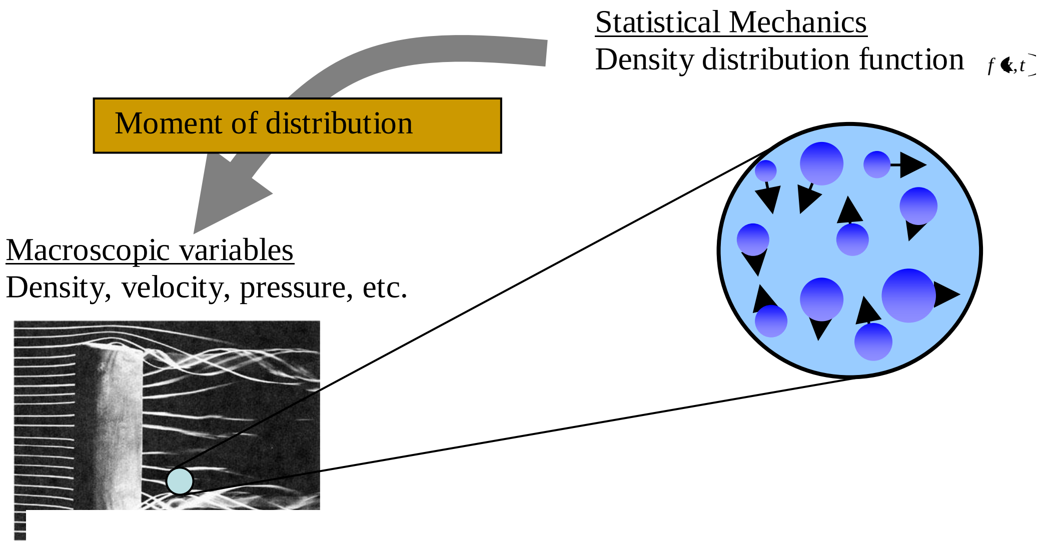

Statistical Mechanics
Density distribution function $f(x,t)$
Moment of distribution
Macroscopic variables
Density, velocity, pressure, etc.

**Figure 2.1:** Lattice Boltzmann Theory (Rosdzimin, 2008)

The primary goal of LBM is to build a bridge between the microscopic and macroscopic dynamics rather than to dealt with macroscopic dynamics directly. In other words, the goal is to derive macroscopic equations from microscopic dynamics by means of statistics rather than to solve macroscopic equation.

The Boltzmann equation for any lattice model is an equation for the time evolution of *fi* (**x**,*t*), the single-particle distribution at lattice site **x**:

$$f(x + c\Delta t, t + \Delta t) - f(x,t) = -\frac{f - f^{eq}}{\tau} \qquad (2.1)$$

## 2.2 COLLISION INTERGRAL, Ω( f )

Basic principle of LBM is including streaming step and collision step. The particles move to another place in the variable direction with their velocities (streaming step) and after they meet to each other, the collision happens (collision step) and the particles will separate again. (streaming step) ( Xiaoyi et al ,1996). We also can take an example from the ‘’snooker’. From this situation, means when a ball hit to another ball, its can firstly streaming and then its will collision and it become streaming again. Collisions between particles change their velocities, and make them move in and out of the domain. A collision term describes the net increase of the density of the number of particles in the domain due to the collision. One of the simplest collision models is the Bhatnagar, Gross and Krook (BGK) simplified collision model.

$$\Omega\ f\ = \frac{1}{\tau}\ f^{eq} - f \tag{2.2}$$

Boltzmann came out with the H-theorem where the value of distribution function will always tend to the equilibrium distribution function, $f^{eq}$ during collision process. The distribution function f can be relate to $f^{eq}$ .

## 2.3 BGK (BHATNAGAR, GROSS, AND KROOK)

n BGK model, the nonlinear collision term of the Boltzmann equation is replaced by a simpler term and the model makes the derivation of the transport equations for macroscopic variables much easier. A problem, which is easily solved by the BGK model, is that of relaxation of a state of a fluid to equilibrium.

$$f\ \ x + c\Delta t, t + \Delta t\ = f\ \ x,t\ - \frac{1}{\tau}\left[ f\ \ x,t\ - f^{eq}\ \ x,t\ \right] \tag{2.3}$$

## 2.4 BOUNDARY CONDITION

The set of conditions specified for the behavior of the solution to a set of differential equations at the boundary of its domain. Boundary conditions are important in determining the mathematical solutions to many physical problems. In a numerical simulation, it is impossible and unnecessary to simulate the whole universe. Generally we choose a region of interest in which we conduct a simulation. The interesting region has a certain boundary with the surrounding environment. Numerical simulations also have to consider the physical processes in the boundary region. In most cases, the boundary conditions are very important for the simulation region's physical processes. Different boundary conditions may cause quite different simulation results. Improper sets of boundary conditions may introduce nonphysical influences on the simulation system, while a proper set of boundary conditions can avoid that. So arranging the boundary conditions for different problems becomes very important. While at the same time, different variables in the environment may have different boundary conditions according to certain physical problems. Commonly there are several different types of boundary conditions.

### 2.4.1 Periodic boundary condition

Periodic boundary conditions (PBC) are a set of boundary conditions that are often used to simulate a large system by modeling a small part that is far from its edge. Periodic boundary conditions are particularly useful for simulating a part of a bulk system with no surfaces present. Moreover, in simulations of planar surfaces, it is very often useful to simulate two dimensions (e.g. x and y) with periodic boundaries, while leaving the third (z) direction with different boundary conditions, such as remaining vacuum to infinity. This setup is known as *slab boundary conditions*.

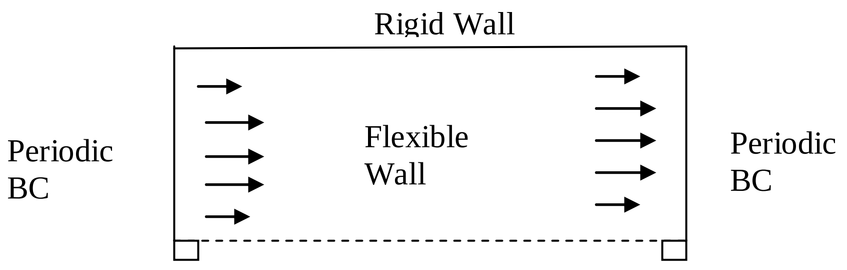


**Figure 2.2:** Periodic Boundary Condition

### 2.4.2 Bounce Back Boundary condition

The bounce-back boundary condition for lattice Boltzmann simulations is evaluated for flow about an infinite periodic array of cylinders. The bounce-back boundary condition is used to simulate boundaries of cylinders with both circular and octagonal cross-sections. The convergences of the velocity and total drag associated with this method are slightly sublinear with grid spacing. Error is also a function of relaxation time, increasing exponentially for large relaxation times. However, the accuracy does not exhibit a trend with Reynolds number between 0.1 and 100. The bounce-back boundary condition is shown to yield accurate lattice Boltzmann simulations with reduced computational requirements for computational grids of 170×170 or finer, a relaxation time less than 1.0 and any Reynolds number from 0.1 to 100. For this range of parameters the root mean square error in velocity and the relative error in drag coefficient are less than 1 % for the octagonal cylinder and 2 % for the circular cylinder.

## 2.4 DISCRETIZATION OF MICROSCOPIC VELOCITY

For the discreatization of microscopic velocity, from the Gauss-Hermitte integration, we can integrate from the continuous velocity to the 9-discrete velocity and also 4-discrete velocity. We focused on the two type of discrete velocity [S. Harris]. For the (poiseulle and couette flow) isothermal fluid flow, we are focusing on 9-discrete velocity model and for the (porous couette flow) thermal fluid flow, we are using 4-discrete velocity model.

### 2.5.1. Isothermal Fluid Flow

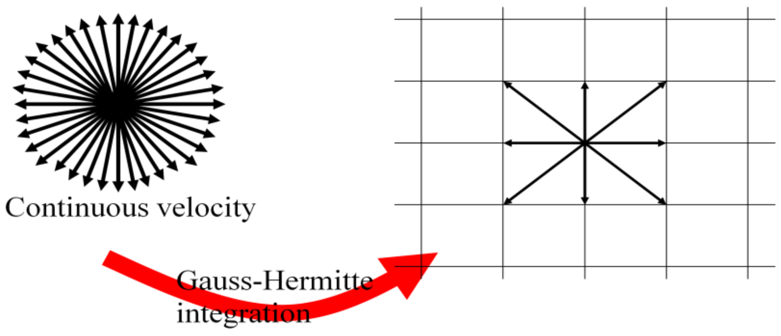


**Figure 2.3:** 9-Discrete Velocity Model

Figure 2.2 showed the 9-discrete velocity model that is going to be used in simulation of isothermal fluid flow (poiseulle flow and coutte flow).

#### 2.5.1.1 The Macroscopic Equation for Isothermal

By using chapmann-enskog expansion procedure, we can have the navier-stroke equations accurate in continuity equations and in momentum equation:

$$\nabla \bullet \mu = 0 \tag{2.4}$$

$$\frac{\partial \mu}{\partial t} + \mu \nabla \bullet \mu = -\nabla P + \left( \frac{2\tau - 1}{6} \right) \nabla^2 \mu \tag{2.5}$$

The relation between the time relaxation τ, in microscopic level and viscosity of fluid ν, in macroscopic level is;

$$\upsilon = \frac{2\tau - 1}{6} \tag{2.6}$$

### 2.5.2 Thermal Fluid Flow

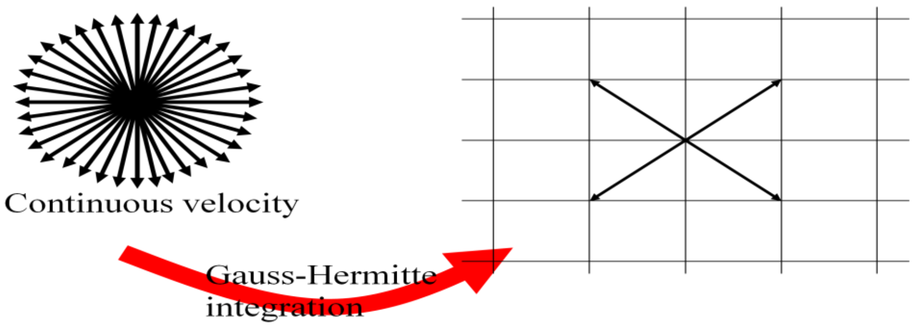


**Figure 2.4:** 4-Discrete Velocity Model

Figure 2.3 showed the 4 discrete velocities model for that are going to be used in simulation of thermal fluid flow (porous couette flow).

#### 2.5.2.1. The Macroscopic Equation for Thermal

By using the chapmann-enskog expansion procedure, we can get the derivation equation for the energy equation.

$$\frac{\partial T}{\partial t} + \nabla \cdot uT = \left(\tau_g - \frac{1}{2}\right)\nabla^2 T \tag{2.7}$$

Where:

$$\tau_g = \chi + \frac{1}{2} \tag{2.8}$$

## 2. 6 THEORY OF REYNOLD NUMBER

Reynolds number can be defined for a number of different situations where a fluid is in relative motion to a surface. These definitions generally include the fluid properties of density and viscosity, plus a velocity and a characteristic length or characteristic dimension. This dimension is a matter of convention - for example a radius or diameter is equally valid for spheres or circles, but one is chosen by convention. For flow in a pipe or a sphere moving in a fluid the diameter is generally used today. Other shapes (such as rectangular pipes or non-spherical objects) have an equivalent diameter defined. For fluids of variable density (e.g. compressible gases) or variable viscosity (non-Newtonian fluids) special rules apply. The velocity may also be a matter of convention in some circumstances, notably stirred vessels.

The Reynolds number can be obtained when one uses the dimensional form of the incompressible Navier-Stokes equations:

$$\rho\left(\frac{\partial \upsilon}{\partial t}+\upsilon\cdot\nabla^2 v\right)=-\nabla\rho+\mu\nabla^2 v+f \tag{2.9}$$

Each term in the above equation has the units of a volume force or, equivalently, an acceleration times a density. Each term is thus dependant on the exact measurements of a flow. When one renders the equation a dimensional, that is that we multiply it by a factor with inverse units of the base equation, we obtain a form which does not depend directly on the physical sizes. One possible way to obtain an a dimensional equation is to multiply the whole equation by the following factor:

$$\frac{D}{\rho v^2} \tag{2.10}$$

we can rewrite the Navier-Stokes equation without dimensions:

$$\frac{\partial v'}{\partial t'}+v'\cdot\nabla' v'=-\nabla' p'+\frac{\mu}{\rho D V}\nabla'^2 v'+f' \tag{2.11}$$

Where the term: $$\frac{\mu}{\rho D V}=\frac{1}{\mathrm{Re}} \tag{2.12}$$

Finally, dropping the primes for ease of reading:

$$\frac{\partial v}{\partial t}+v.\nabla v=-p+\frac{1}{\mathrm{Re}}\nabla^2 v+f \tag{2.13}$$

This is why mathematically all flows with the same Reynolds number are comparable.

## 2.7 RAYLEIGH NUMBER

In fluid mechanics, the Rayleigh number for a fluid is a dimensionless number associated with buoyancy driven flow (also known as free convection or natural convection). When the Rayleigh number is below the critical value for that fluid, heat transfer is primarily in the form of conduction; when it exceeds the critical value, heat transfer is primarily in the form of convection.

The Rayleigh number is named after Lord Rayleigh and is defined as the product of the Grashof number, which describes the relationship between buoyancy and viscosity within a fluid, and the Prandtl number, which describes the relationship between momentum diffusivity and thermal diffusivity. Hence the Rayleigh number itself may also be viewed as the ratio of buoyancy forces and (the product of) thermal and momentum diffusivities.

For free convection near a vertical wall, this number is

$$Ra_x = Gr_x \Pr = \frac{g\beta}{v\alpha} \; T_s - T_\infty \;\; x^3 \tag{2.14}$$

In the above, the fluid properties Pr, $v$, $\alpha$ and $\beta$ are evaluated at the *film temperature*, which is defined as

$$T_f = \frac{T_s + T\infty}{2} \tag{2.15}$$

For most engineering purposes, the Rayleigh number is large, somewhere around $10^6$ and $10^8$.

## 2.8 PRANDTL NUMBER

The Prandtl number Pr is a dimensionless number approximating the ratio of momentum diffusivity (kinematic viscosity) and thermal diffusivity. It is named after the German physicist Ludwig Prandtl.

It is defined as:

$$\Pr = \frac{v}{\alpha} = \frac{C_p u}{k} \tag{2.16}$$

Typical values for Pr are:

I. around 0.7-0.8 for air and many other gases,
II. around 0.16-0.7 for mixtures of noble gases or noble gases with hydrogen
III. around 7 for water
IV. around $10\times10^{24}$ for Earth's mantle
V. between 100 and 40,000 for engine oil,
VI. between 4 and 5 for R-12 refrigerant
VII. around 0.015 for mercury

In heat transfer problems, the Prandtl number controls the relative thickness of the momentum and thermal boundary layers. When *Pr* is small, it means that the heat diffuses very quickly compared to the velocity (momentum). This means that for liquid metals the thickness of the thermal boundary layer is much bigger than the velocity boundary layer.

The mass transfer analog of the Prandtl number is the Schmidt number.

# CHAPTER 3

# METHADOLOGY

## 3.1 FLOW CHART

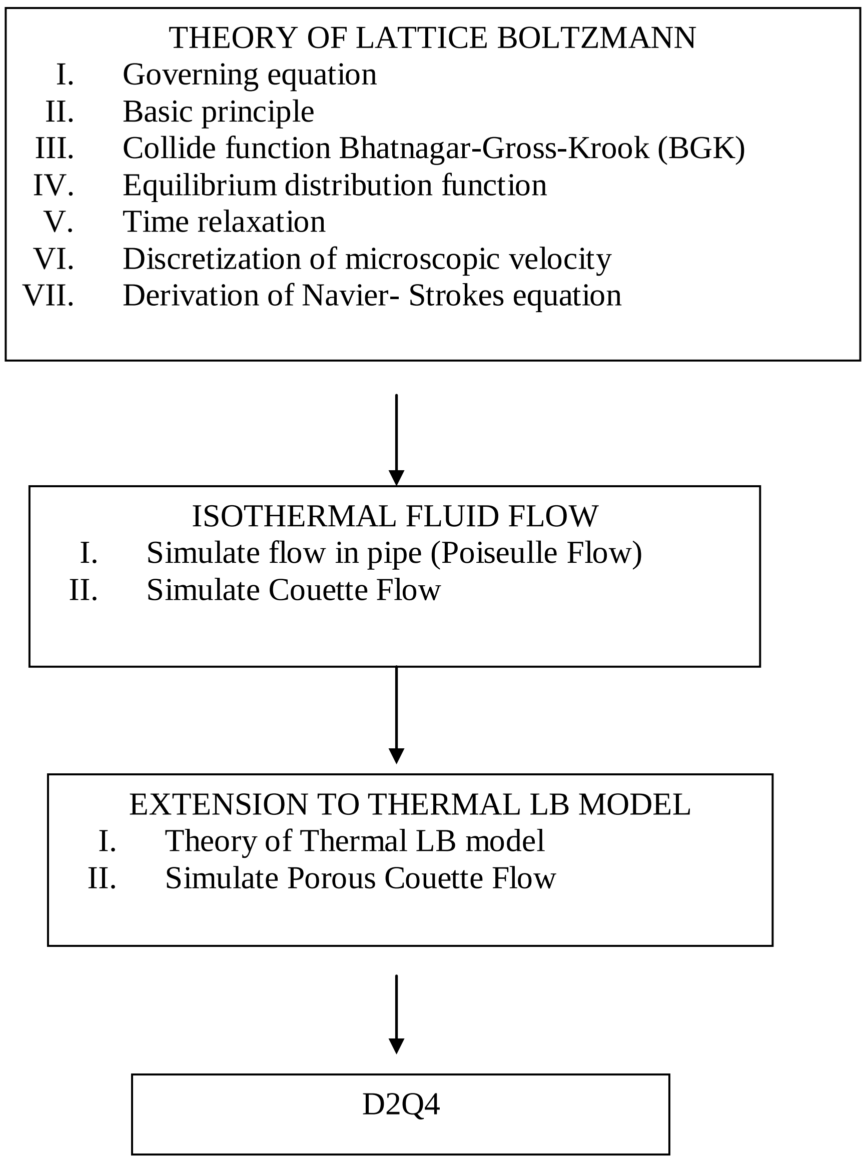

THEORY OF LATTICE BOLTZMANN
I. Governing equation
II. Basic principle
III. Collide function Bhatnagar-Gross-Krook (BGK)
IV. Equilibrium distribution function
V. Time relaxation
VI. Discretization of microscopic velocity
VII. Derivation of Navier- Strokes equation
ISOTHERMAL FLUID FLOW
I. Simulate flow in pipe (Poiseulle Flow)
II. Simulate Couette Flow
EXTENSION TO THERMAL LB MODEL
I. Theory of Thermal LB model
II. Simulate Porous Couette Flow
D2Q4

**Figure 3.1:** Flowchart for methodology

## 3.2 ALGORITHM

The algorithm flowchart for LBM is shown in Figure 3.2. It consists of two processes; advection and collision process. The initial values of density distribution $f$ are specified at each grid point. Then, the system evolves in the following steps.

- The advection term is solved by applying the streaming process of the density distribution function.
- Then the collision process is solved by BGK collision model.
- Next step is to define the boundary conditions based on the bounce back boundary conditions.

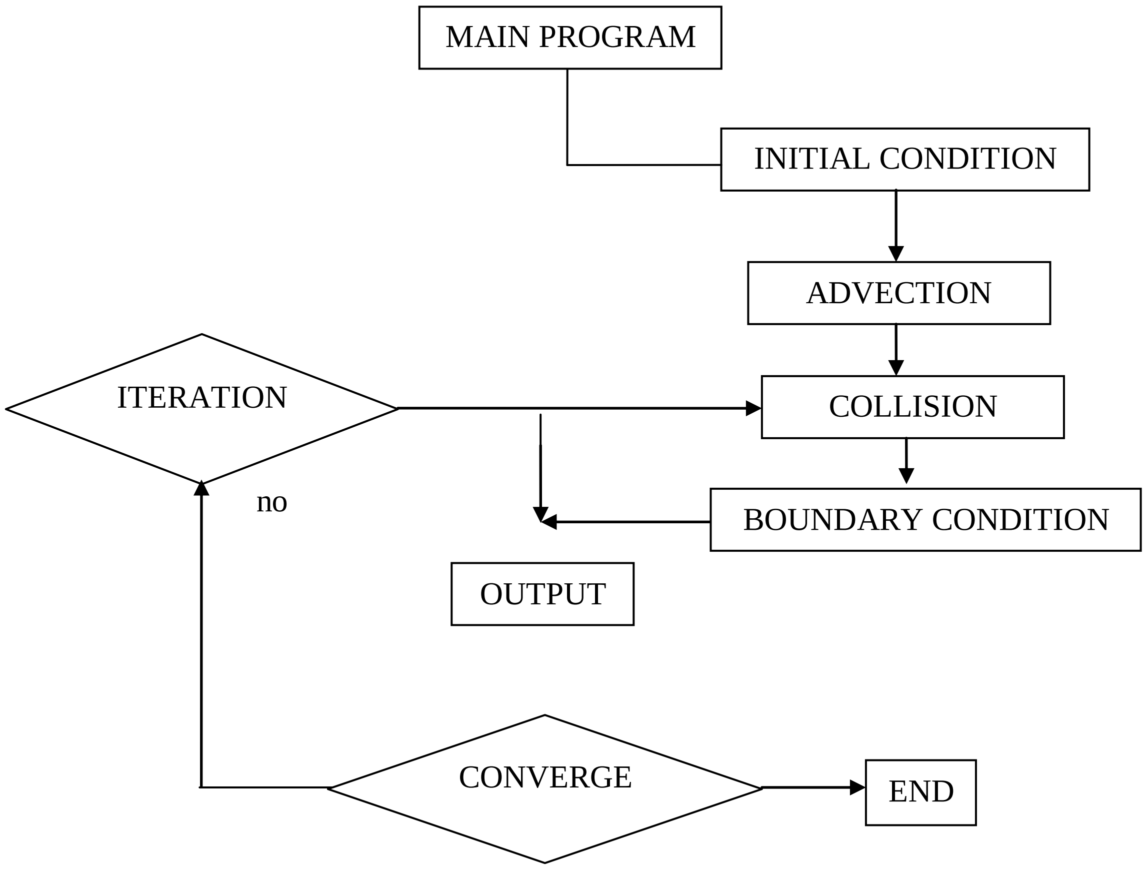


**Figure 3.2:** Original LBM Algorithm Flowchart.

## 3.3 SIMULATION RESULTS FOR ISOTHERMAL

To illustrate some of the important features of the lattice Boltzmann isothermal model, a number of simulations were performed.

### 3.3.1 Poiseulle flow

Numerical simulation for the Poiseulle flow driven by a pressure gradient was carried out to test the validity of the isothermal lattice Boltzmann model. The pressure gradient is set between the inlet and the outlet end of the channel. The densities are set at different values between the two ends. The bounce back boundary condition is applied at the top and bottom walls.

It is observed that it reaches a steady state corresponding to the parabolic solution of the channel flow (I.J. Sobej, 1985). The criterion of steady state is set by

$$\frac{\sum i \sum x \left[ f_i(x,t+1) - f_i(x,t) \right]^2}{\sum_i i \times M \times N} \tag{3.1}$$

where M and N are the mesh numbers in x and y direction. Two types of measurements were taken of the viscosity and the boundary conditions. Measurement of velocity u and the pressure are taken. The figure corresponds to a simulation using a lattice size of 4 x 33.

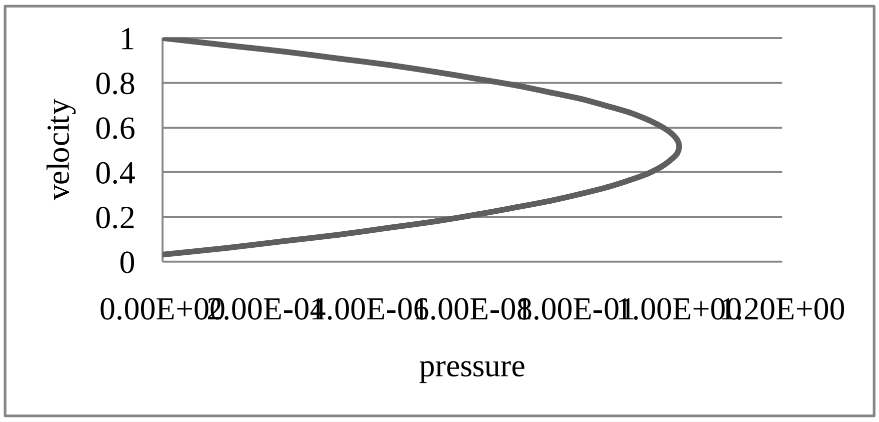


**Figure 3.3:** Graph of Poiseulle Flow

### 3.3.2 Couette flow

A numerical experiment involving the time evolution of the Couette flow is presented, in which the top plate moves with constant velocity, while the bottom plate is held fixed. The initial conditions correspond to a null velocity everywhere except on the top boundary, where the velocity is u = (1,0). The x-component of the velocity on the top plate is maintained at U = 1.00 (top plate boundary condition in LBM units), whereas the bottom one is at rest. No pressure gradient is included for this case.

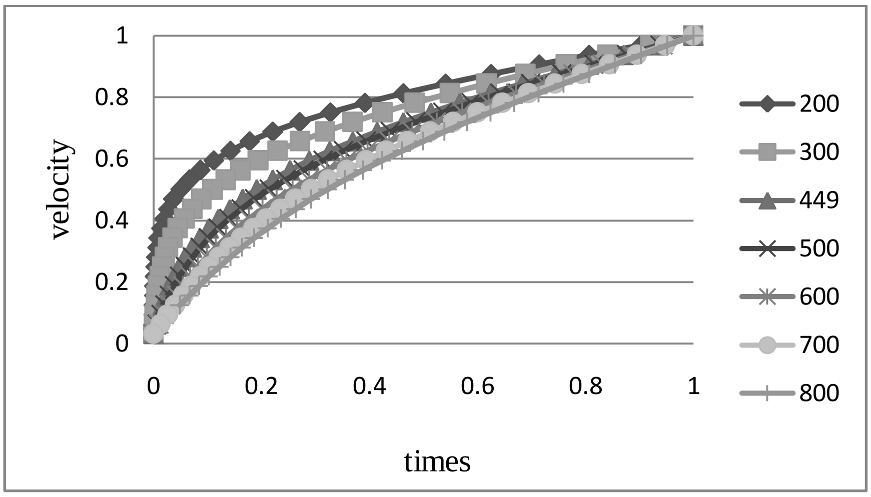


**Figure 3.4:** Graph of Couette Flow

Figure above shows a sequence of normalized velocity profile for different times. The lattice size for this experiment is 4 x 32 and the relaxation time is τ = 1.0 Do. The velocity profiles are drawn at times t = 200, 300… 800 in LBM units. Periodic boundary conditions are implemented in the x-direction. The solution for the steady state case is well known, and corresponds to the velocity increasing linearly from zero at the bottom to U at the top plate (C.S. Nor Azwadi and T.Tanahashi, 2006).

## 3.4 SIMULATION RESULT FOR THERMAL FLOW

In this section, we should apply a newly developed model to simulate heat transfer porous plate Couette flow problem (L.S. Suo,1999).

### 3.4.1 Thermal Porous Couette Flow

Periodic boundary conditions are used at the entrance and exit of the channel, and the non-equilibrium bounce back boundary conditions for velocity. For temperature boundary condition, the non-equilibrium bounce back boundary condition is used.

The normalized temperature profile for Pr = 0.71, Ra= 100 and Re = 5, 10, 20, 30, and 40 are shown in figure 3.4.

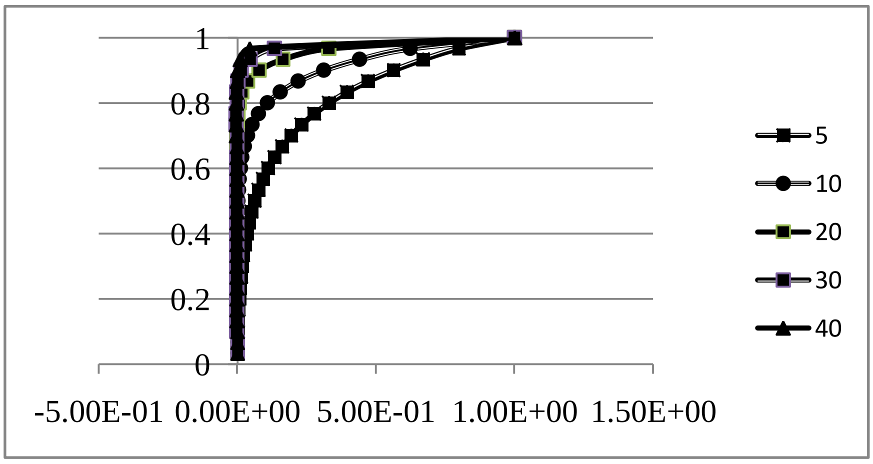


**Figure 3.5:** Temperature Profile at Pr= 0.71 and Ra= 100

Figure 3.6 shows the normalized temperature profile for Ra= 100 and Re = 10 with Pr = 0.2, 0.8 and 1.5.

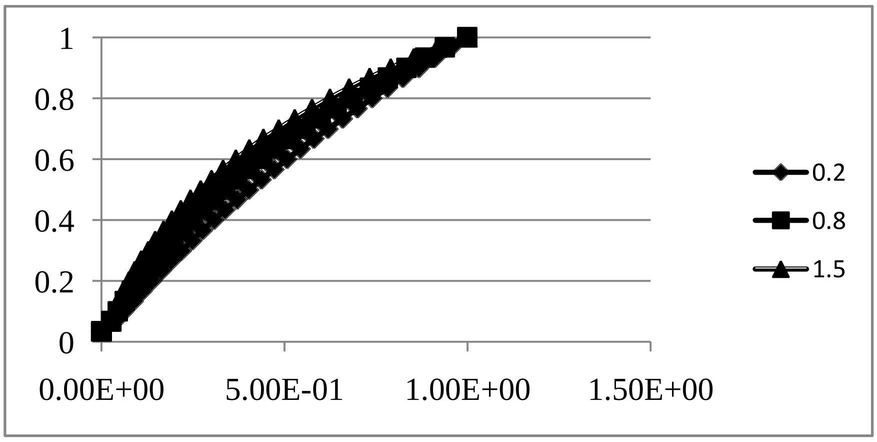


**Figure 3.6:** Temperature Profile at Ra=100 and Re=10.

## 3.5 OLD VELOCITY MODEL VERSUS AZWADI'S VELOCITY MODEL

### 3.5.1 Old Velocity Model

The original velocity model was proposed by He et al. He et al in their model introduces the internal energy density distribution function which can be derived from the Boltzmann equation. This model is shown to be a suitable model for simulating real thermal problems.

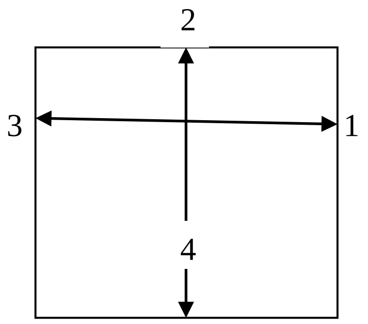


**Figure 3.7:** 2 D Lattice structure of old velocity model

### 3.5.2 Azwadi's Velocity Model

The new type of lattice model which only use four-velocity was developed for internal energy density distribution function in incompressible limit. Both the evolution equations have been directly derived from the continuous Boltzmann equation and Maxwell-Boltzmann equilibrium distribution function.

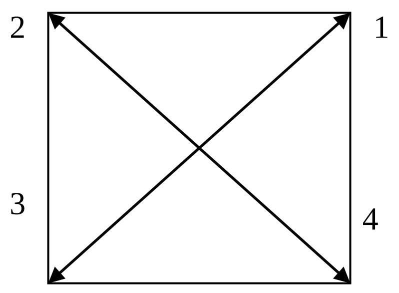


**Figure 3.8:** 2 D Lattice Structure of Azwadi's velocity model

## 3.6 SUMMARY

The main contributions in this thesis, the evaluation of two different types of four-velocity model are developed. The simulation of thermal fluids flow is applied to two different types of four-velocity model. The same simulation test is performed for both types and the accuracy and stability analysis of both models are also discussed. These models are compared and tested to ensure its validity.

# CHAPTER 4

# RESULTS AND DISCUSSION

## 4.1 TEST CASES: NATURAL CONVECTION IN SQUARE CAVITY

The natural convection in a square cavity plays a very interesting role in a lot of engineering applications, such as the solar energy system, the cooling of the electronic circuits, the conditioning of the air and many others.

In this paper, the cavity testing is a square cavity where we analyzed the influence of the position of a heat stripe, placed on the left side of the cavity on natural convection heat transfer. The cavity is cooled on the right side of the wall at the same time. The top and the bottom of the cavity are adiabatic.

The temperature different between the left and right walls introduces a temperature gradient in a fluid, and the consequent density different induces a fluid motion, that is convection.

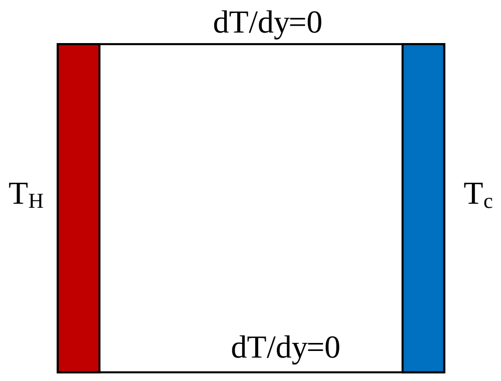


**Figure 4.1:** Schematic Diagram for natural convection in a square cavity

## 4.2 GRID DEPENDENCE TEST

Grid dependence test was carried out to find the most suitable grid size for every Rayleigh number simulation. In this study, the number of grid points is taken the same in both x and y directions. That is the grid is taken as N x N, where N is the grid number in each spatial directions. Figure 4.2 and Figure 4.3 show the grid dependency test's result for 2D natural convection in a square cavity at Ra = $10^3$ and Pr = 0.71.

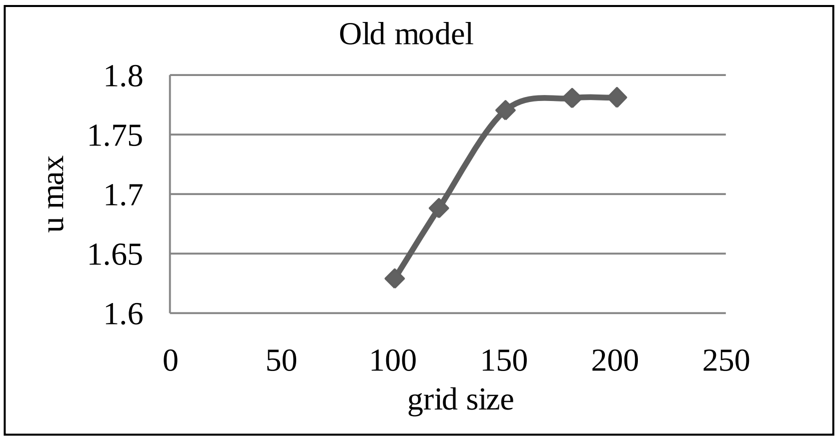


**Figure 4.2:** Grid dependence test for Old model at Ra = $10^4$ and Pr = 0.71

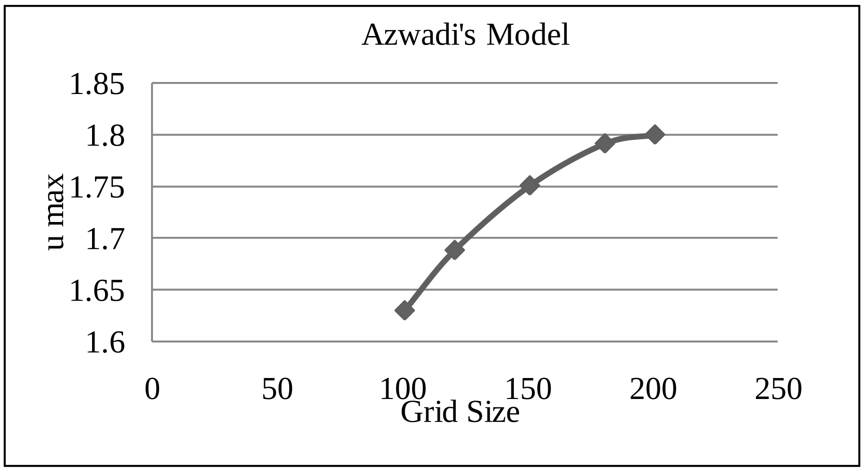


**Figure 4.3:** Grid dependence test for Azwadi's model at Ra = $10^4$ and Pr = 0.71

From the graph, it can be clearly seen that when N increase, the $U_{max}$ value will also increased. The grid sizes of 151 x 151 is sufficient for the simulation at Rayleigh number of Ra = $10^3$ for old model and 181 x 181 for Azwadi's model.

## 4.3 NUSSELT NUMBER COMPARISON

The Nusselt number is a dimensionless number, which describes the relationship between convective heat transfer and conductive heat transfer. The Nusselt number is defined as the ratio of convection heat transfer to conduction heat transfer, where the heat conduction is under the same conditions as the heat convection except with a stagnant (motionless) fluid.

A Nusselt number of ~1 would indicate "slug flow" or laminar flow with convection heating having a magnitude similar to conduction heating. A large Nusselt number ~ 100 to 1000 means very active convection, a characteristic of turbulent flow. The convection and conduction heat flows are parallel to each other and to the surface normal of the boundary surface, and are all perpendicular to the mean fluid flow in the simple case.

Typically the average Nusselt number is expressed as a function of the Rayleigh number and the Prandtl number, written as: Nu = *f* (Ra, Pr).

The calculated average Nusselt numbers are taken from the simulation result of both models. The averaged Nusselt numbers are changing with the increasing number of Rayleigh numbers.

**Table 4.1:** Table of Nusselt Number Comparison

| Rayleigh Number | Benchmark | Azwadi's Model | Old Model |
|---|---|---|---|
| 1000 | 1.116 | 1.116 | 1.117 |
| 10 000 | 2.201 | 2.108 | 2.203 |
| 100 000 | 4.549 | 4.542 | 5.539 |

## 4.4 OLD MODEL RESULT VERSUS AZWADI'S MODEL RESULT

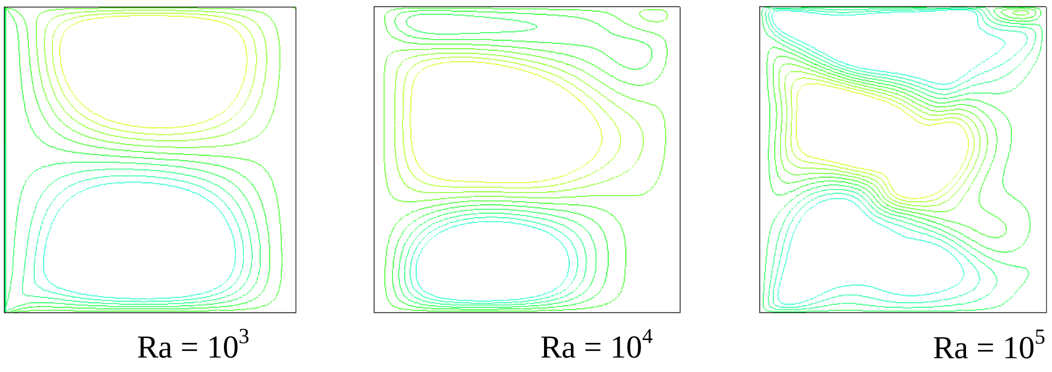


**Figure 4.4:** Equivalent state for Azwadi's model

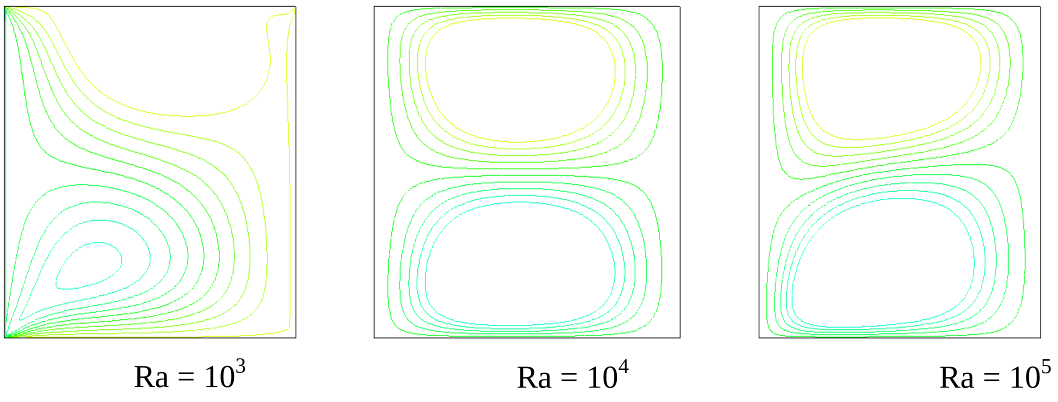
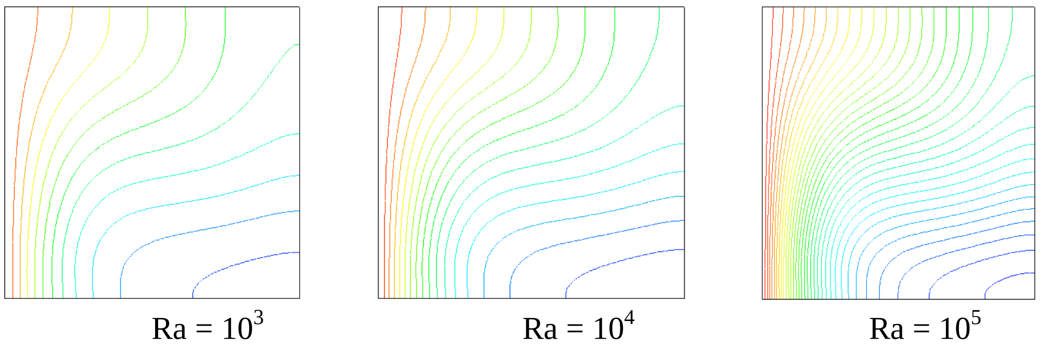


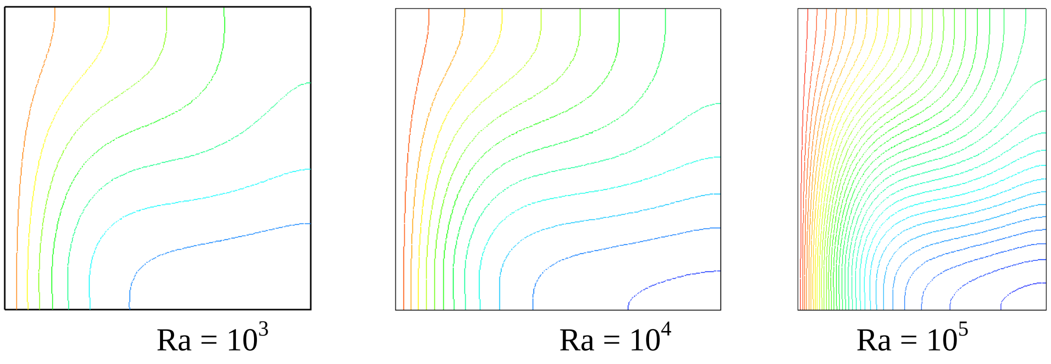
**Figure 4.5:** Equivalent state for Old model

Temperature contours at equilibrium state for flows at Ra = $10^3 \sim 10^5$ are shown in figure. By increasing in Rayleigh number, a high degree of convection is observed such that distinct thermal boundary layers start appearing near the isothermal walls. The thickness of the boundary layer decrease as Rayleigh number increase.



Ra = $10^3$ Ra = $10^4$ Ra = $10^5$

**Figure 4.6:** Isotherms state for Azwadi's Model



Ra = $10^3$ Ra = $10^4$ Ra = $10^5$

**Figure 4.7:** Isotherms state for Old Model

At lower Rayleigh number heat transfer was dominated by the conduction mode. This can be seen from the straight and equally spaced isotherm line. At Ra = $10^3$, the isotherms are almost parallel to the wall indicating that conduction is the dominant heat transfer mechanism. At Ra = $10^4$, isotherms start to be horizontally parallel to the wall at the cavity center. Finally at Ra = $10^5$, all isotherms are almost horizontally parallel to the wall indicating that the convection is the main heat transfer mechanism. The isotherms lines were distorted because of the buoyancy induced convection becomes more predominant than conduction. Hot fluid moves from the source until reaches the opposite of the wall and moves outwards along the cold wall under the effect of cooling. Increasing the Rayleigh number, isotherms are distorted more due to the stronger convection effect, leading to the stable stratification of the isotherms.

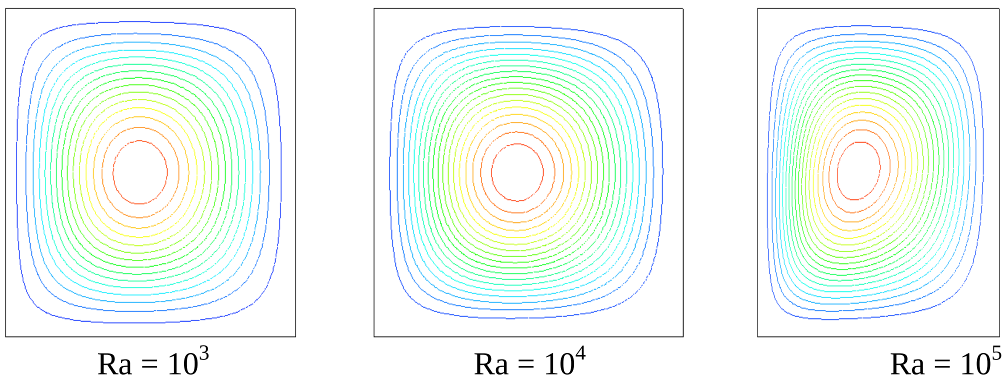


**Figure 4.8:** Streamline for Azwadi's Model

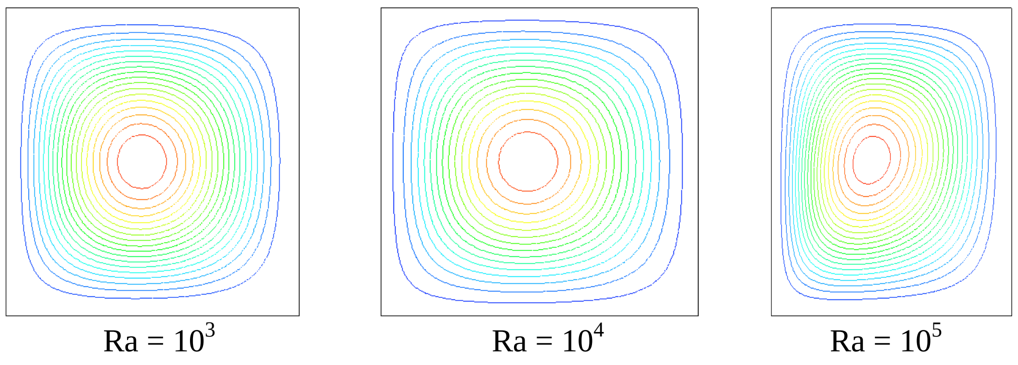


**Figure 4.9:** Streamline for Old's Model

Vortex appears at the center of the cavity with circular shape for both models. Circular vortex at the center of cavity was distorted when using higher Rayleigh number and shape become oval due to the convection effect. We can say that for Ra = $10^3$ and Ra = $10^4$, the grid size of 151 x 151 can give very accurate results for Old model and 181 x 181 for Azwadi model. However for Ra = $10^5$, both models requires high spatial resolution and it is found to be stable at grid size 201 x 201.

# CHAPTER 5

# CONCLUSIONS AND RECOMMENDATION

## 5.1 CONCLUSIONS

Chapter one introduced the LBM and it advantages. The model of LBM developed from continuous Boltzmann equation, has evolved into a powerful tool for modeling complex flow. The LBM has a number of advantages over other conventional CFD methods. The algorithm can also be easily modified to allow for the application of other, more complex components. Thus the LBM is an ideal tool in fluid simulations. The project objective, project problems and backgrounds are also discussed in chapter one.

In chapter two, the literature reviews of the LBM and the boundary conditions involved are mention clearly. The lattice Boltzmann theory and the discretization of microscopic velocity for isothermal and thermal fluid flow are shown here. Several types of boundary conditions that used in the LBM simulation and also in the isothermal model and thermal model have been discussed here. Finally, the theory of Reynold number, Rayleigh number and also Prandtl number is discussed.

The methodology and the algorithm that been used for the simulation was explained in chapter three. Results of the numerical simulations for the Poiseulle flow, Couette flow, and porous Couette flow using the isothermal and thermal lattice Boltzmann model have been performed. Results for all the above fluid flow problems show that LBM is a reliable CFD technique and agreed with the analytical solution and conventional approach. The differences between both models are also shown at the end of chapter three.

Objective of the project was achieved and discussed at the end of chapter four. The grid dependency test was done to find out the most suitable grid size to be used in the simulations of natural convections in a square cavity. It was found out that the grid size 151 x 151 is suitable for old model and 181 x 181 for Azwadi's model. Then both models was simulated at Ra = $10^3$, Ra = $10^4$ and Ra = $10^5$. The Nusselt numbers for each simulation was compared to the previous study and the result obtained agreed well with the benchmark.

These old and Azwadi's model were shown to be a suitable and suitable for the computational of low and moderate Rayleigh number. Results obtained give good agreement with the benchmark solution. For the simulation at high Rayleigh number, we were forced to apply small value of time relaxation and this will increase the simulation time. The results obtained prove that both models are efficient to study flow and heat transfer in the future. The performance for both model are almost the same except that Azwadi's model give slightly higher accuracy. It is because the Azwadi's model gives more direction of velocity particle compared to the old model but this direction can only be seen clearly in 3D model. Time consuming for Azwadi's model is also higher than the old model. The time taken for simulation is effect by the direction of particle velocity. The higher the value of the direction leads to the increasing of time consuming for simulation.

## 5.2 RECOMMENDATION

The performance for both models can be seen clearly by doing the simulation for 3-D model instead of 2-D model. However, higher time consuming are required to run the 3-D simulation successfully. Therefore, some solutions to overcome the time consuming should be studied in details. In that way, the simulations can be done in shorter time.

## REFERENCES

C.Cercignani, “Theory and Application of Boltzmann Equation”, Scottish Academic Press, 1975.

C. Cercignani, “The Boltzmann equation and its application, in Applied mathematical Sciences”, Springer Verlag, New York, 1988

C. S. Nor Azwadi and T. Tanahashi, “Simplified thermal lattice Boltzmann in incompressible limit”, Int. J. Mod. Phys. B, 20 (17), pp2437-2449 (2006)

I. J. Sobey, “Observation of waves during oscillatory channel flow”, J. Fluid Mesh., 151, pp395-426 (1985)

J.C. Tannehill, D.A.Anderson and R.H.Pletcher, “Computational Fluid Mechanics and Heat Transfer”, Taylor and Francis, 1997.

L. S. Luo, “Lattice-gas automata and lattice Boltzmann equations for two dimensional hydrodynamics”, Ph.D Thesis, Georgia Institute of Technology, 1993.

N. S. Martys and H. Chen, “Simulation of multicomponent fluids in complex three-dimensional geometries by the lattice Boltzmann method”, Phys, Rev. E, 53, pp743-750 (1996)

S. Harris, “An introduction to the theory of the Boltzmann equation”, Holt, Rinehart and Winston, New York, 1971.

X. He and L. S. Luo , “A priori derivation of the lattice Boltzmann equation”, Phys. Rev. E, 55, pp6333-6336 (1997)

X. He and L. S. Luo, “Lattice Boltzmann Model for the Incompressible Navier-Stokes equation”, J. Comp. Phys., 171, pp336-356 (2001)

Xiaoyi He and Li-Shi Luo, “*A priori* derivation of the lattice Boltzmann equation”, phys. Fluid, 55(6), 1996

## APPENDIX A1

**Final Year Project 1**

| **PROJECT ACTIVITIES** | **W 2** | **W 3** | **W 4** | **W 5** | **W 6** | **W 7** | **W 8** | **W 9** | **W 10** | **W 11** | **W 12** | **W 13** | **W 14** | **W 15** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Literature Study | 🟥 | 🟥 | 🟥 | 🟥 | | | | | | | | | | |
| Lattice Boltzmann | | | 🟥 | 🟥 | 🟥 | 🟥 | | | | | | | | |
| Isothermal Flow | | | | | | | | | | | | | | |
| 1. Poiseulle Flow | | | | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | | | | | | |
| 2. Couette Flow | | | | | | | 🟥 | 🟥 | 🟥 | 🟥 | | | | |
| Thermal Fluid Flow | | | | | | | | | | | | | | |
| Porous Couette Flow | | | | | | | | | 🟥 | 🟥 | 🟥 | | | |
| Submit Report | | | | | | | | | | | | 🟥 | 🟥 | |
| Presentation | | | | | | | | | | | | | 🟥 | |

**A2**

**Final Year Project 2**

| **PROJECT ACTIVITIES** | | **W 2** | **W 3** | **W 4** | **W 5** | **W 6** | **W 7** | **W 8** | **W 9** | **W 10** | **W 11** | **W 12** | **W 13** | **W 14** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Solve Equation** | | 🟥 | | | | | | | | | | | | |
| **Grid Dependency Test** | | | | | | | | | | | | | | |
| Old Model | 101 x 101 | 🟥 | 🟥 | | | | | | | | | | | |
| | 121 x 121 | | 🟥 | | | | | | | | | | | |
| | 151 x 151 | | 🟥 | | | | | | | | | | | |
| | 181 x 181 | | | 🟥 | | | | | | | | | | |
| | 201 x 201 | | | 🟥 | | | | | | | | | | |
| Azwadi model | 101 x 101 | | | | 🟥 | | | | | | | | | |
| | 121 x 121 | | | | 🟥 | | | | | | | | | |
| | 151 x 151 | | | | | 🟥 | | | | | | | | |
| | 181 x 181 | | | | | 🟥 | | | | | | | | |
| | 201 x 201 | | | | | | 🟥 | | | | | | | |
| **Simulate at Diff Ra** | | | | | | | | | | | | | | |
| Old Model | Ra = $10^3$ | | | | | | | 🟥 | | | | | | |
| | Ra = $10^4$ | | | | | | | 🟥 | | | | | | |
| | Ra = $10^5$ | | | | | | | | 🟥 | | | | | |
| Azwadi Model | Ra = $10^3$ | | | | | | | | 🟥 | | | | | |
| | Ra = $10^4$ | | | | | | | | | 🟥 | | | | |
| | Ra = $10^5$ | | | | | | | | | 🟥 | | | | |
| Check Nu | | | | | | | | | | | 🟥 | | | |
| Plot Using AVS | | | | | | | | | | | | 🟥 | 🟥 | |
| Presentation | | | | | | | | | | | | | | 🟥 |
| Submit Report | | | | | | | | | | | | | | |

## B

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Lattice Boltzmann Method     !
!   based on Square Lattice    !
!     AZWADI MODEL               !
!   Natural convection     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      program natural_convection

      parameter(cd = 9, xd = 181, yd = 181, dd = 4) !UBAH XD, YD
      =121,151,181,201
      real*8  cx(1:cd), cy(1:cd)
      real*8  dx(1:dd), dy(1:dd)
      real*8  f(1:cd,1:xd,1:yd), f0(1:cd,1:xd,1:yd),
ff(1:cd,1:xd,1:yd)
      real*8  g(1:dd,1:xd,1:yd), g0(1:dd,1:xd,1:yd),
gg(1:dd,1:xd,1:yd)
      real*8  ux(1:xd,1:yd), uy(1:xd,1:yd), rho(1:xd,1:yd)
      real*8  temp(1:xd,1:yd),sumf,sumg
      real*8  vel(1:xd,1:yd)
      real*8  u0, rho0, pi,tauv,th,tc,pt,ra,nyu,ok,di,tauc,ri,re

      u0 = 0.0                ! wall velocity
      rho0 = 1.0              ! constant density
      pt = 0.71               ! Prandtl number
      ra = 10000              ! rayleigh number
      th = 1.0                ! left hot wall
      tc = 0.0                ! right cool/initial wall temperature

      write(*,*) 'calculation start'
      write(*,*) 'nstep?'                                     !
      iteration step for display
      read(*,*) nstep
      write(*,*) 'nstep =',nstep

      gra = (0.0577**2)/(yd)

      nyu = (gra*((yd-1)**3)*(th-tc)*pt/ra)**0.5
      tauv = 3*nyu + 0.5
      di    = nyu/pt    !!!!!!!!!!!!!!!!!NEED MODIFICATION
      tauc = di + 0.5

      write(*,*) 'nyu  =',nyu
      write(*,*) 'di   =',di
      write(*,*) 'Tauv =',tauv
      write(*,*) 'Tauc =',tauc
      write(*,*) 'gra =',gra
      write(*,*) 'everything is ok?,press 0 if ok'
      read (*,*) ok

      call initial1
      (cx,cy,f,f0,g,g0,ux,uy,rho,temp,u0,rho0,th,tc,cd,xd,yd,dx,dy,dd,
x1,   x2,y1,y2)

      do iter = 1, 1000000
```

```
            call collide1 (f,f0,tauv,cd,xd,yd,g,g0,tauc,dd)

            call force (cx,cy,ux,uy,temp,f,f0,gra,cd,xd,yd)

            call stream1 (f,cd,xd,yd,g,dd)

            call boundary1
      (f,f0,rho,u0,cd,xd,yd,g,g0,temp,dd,x1,x2,y1,y2)

            call calc1
      (cx,cy,f,ux,uy,rho,u0,rho0,cd,xd,yd,temp,g,tc,th,dd,dx,dy,x1,x2,
y1,   y2)

            if (mod(iter,nstep) .eq.0) then
                  write(*,*) 'time step=', iter
                  write(*,*) 'velocity=',ux(xd/4,yd/4)
                  write(*,*) 'temperature= ',temp(xd/4,yd/4)
            end if

            call equilibrium1
(cx,cy,f0,ux,uy,rho,cd,xd,yd,dx,dy,g0,temp,dd)
!!!!!!!!!!!!!!    check for convergence
      if (mod(iter,nstep) .eq.0) then
            write (*,*)'sum_f = ',sumf
                  do k = 1,cd
                        do i = 1,xd
                              do j = 1,yd
                                    ff(k,i,j) = f(k,i,j)
                           end do
                        end do
                  end do

                  write (*,*)'sum_g = ',sumg
                  do k = 1,4
                        do i = 1,xd
                              do j = 1,yd
                                    gg(k,i,j) = g(k,i,j)
                           end do
                        end do
                  end do

    endif

      if (mod(iter,nstep) .eq.1) then
                  sumf = 0.
                  do k = 1,cd
                        do i = 1,xd
                              do j = 1,yd
                                    sumf = sumf +(f(k,i,j)-
ff(k,i,j))**2
                              end do
                        end do
                  end do

                  sumf = (sumf/9.0*(yd)*(xd))**0.5

                  !***** if converge*****::!
                  if (sumf .le. 1.0d-3) then
                  write(*,*)'solution_f converge'
!                       go to 100
                  end if

                  sumg = 0.
                  do k = 1,4
                        do i = 1,xd
                              do j = 1,yd
                                    sumg = sumg +(g(k,i,j)-
gg(k,i,j))**2
                              end do
                        end do
                  end do

                  sumg = (sumg/4.0*(yd)*(xd))**0.5

                  !***** if converge*****::!
                  if (sumg .le. 1.0d-3) then

                  write(*,*)'solution_g converge'
!                       go to 100
                  end if

            if (sumg .le. 1.0d-3 .and. sumf .le. 1.0d-3) then
            write(*,*)'both solution converge'

            go to 100
            end if

            end if

      end do
100         bnu = 0.0
      open (unit=30,file='u
vel1.dat',status='replace',action='write',iostat=ierror)
      write(30,*)'thermal diffusivity      ',di
      write(30,*)'Prandtl number           ',pt
      write(30,*)'hydro relax. time        ',tauv
      write(30,*)'termo relax. time        ',tauc
      write(30,*)'solution converge at     ',iter

      do j = 1,yd

      write(30,*) ux((xd+1)/2,j)*(yd-1)/di

      end do

      close(30)

      open (unit=31,file='v
vel1.dat',status='replace',action='write',iostat=ierror)
      do i = 1,xd

      write(31,*) uy(i,(yd+1)/2)*(yd-1)/di

      end do

      close(31)
      open
(unit=32,file='variables.dat',status='replace',action='write',iostat=i
error)
      write(32,*)'x-vel, y-vel, temp'
      do j = 1,yd
      do i = 1,xd

      write(32,*) ux(i,j)*(yd-1)/di,uy(i,j)*(yd-1)/di,temp(i,j)

      end do
      end do

      close(32)
      end

      subroutine initial1
(cx,cy,f,f0,g,g0,ux,uy,rho,temp,u0,rho0,th,tc,cd,xd,yd,dx,dy,dd,x1,x2,
y1,y2)
!----------------------------------------------------------------------
      real*8  cx(1:cd), cy(1:cd), w(1:cd)
      real*8  dx(1:dd), dy(1:dd)
      real*8  f(1:cd,1:xd,1:yd), f0(1:cd,1:xd,1:yd)
      real*8  g(1:dd,1:xd,1:yd), g0(1:dd,1:xd,1:yd)
      real*8  ux(1:xd,1:yd), uy(1:xd,1:yd), rho(1:xd,1:yd)
      real*8  temp(1:xd,1:yd)
      real*8  u0,  rho0, pi,tc,th

      pi = 4.0*atan(1.0)
    cx(1) =  0.0
    cy(1) =  0.0
    do k = 2,9
      w(k) = 1.
      if(mod(k, 2) .eq. 1.) w(k) = sqrt(2.)
            cx(k) = w(k)*cos((k-2)*pi/4.0)
        cy(k) = w(k)*sin((k-2)*pi/4.0)
    enddo

      dx(1) = 1.0             !!!!!!!!!!! NEED MODIFICATION
      dy(1) = 1.0             !!!!!!!!!!! NEED MODIFICATION
      dx(2) = -1.0            !!!!!!!!!!! NEED MODIFICATION
      dy(2) = 1.0             !!!!!!!!!!! NEED MODIFICATION
      dx(3) = -1.0            !!!!!!!!!!! NEED MODIFICATION
      dy(3) = -1.0            !!!!!!!!!!! NEED MODIFICATION
      dx(4) = 1.0             !!!!!!!!!!! NEED MODIFICATION
      dy(4) = -1.0            !!!!!!!!!!! NEED MODIFICATION

      do i = 1,xd
                  do j = 1,yd
                        rho(i,j) = 1.0

                        if (i.eq.1) then
                              ux(i,j) = 0.0
                              uy(i,j) = 0.0
                              temp(i,j) = th
                        else
                              ux(i,j) = 0.0
                              uy(i,j) = 0.0
                              temp(i,j) = tc
                        end if
                  end do
            end do
      call equilibrium1 (cx,cy,f0,ux,uy,rho,cd,xd,yd,dx,dy,g0,temp,dd)

      do k = 1,cd
            do i = 1,xd
                  do j = 1,yd
                        f(k,i,j) = f0(k,i,j)
                  end do
            end do
      end do

      do k = 1,dd
            do i = 1,xd
                  do j = 1,yd
                        g(k,i,j) = g0(k,i,j)
                  end do
            end do
      end do

      return
      end

    subroutine equilibrium1
(cx,cy,f0,ux,uy,rho,cd,xd,yd,dx,dy,g0,temp,dd)
!----------------------------------------------------------------------
      real*8  cx(1:cd), cy(1:cd), f0(1:cd,1:xd,1:yd)
      real*8  dx(1:dd), dy(1:dd)
      real*8  g0(1:dd,1:xd,1:yd)
    real*8  ux(1:xd,1:yd), uy(1:xd,1:yd), rho(1:xd,1:yd)
      real*8  temp(1:xd,1:yd)
      real*8  u2, tmp
      integer i, j, k, m

    do i = 1,xd
            do j = 1,yd
                        u2 = ux(i,j)**2 + uy(i,j)**2
                        f0(1,i,j) = rho(i,j)*(1. - 3./2.*u2)*4./9.
                        do k = 1,4
                        m = k*2    ; tmp = cx(m)*ux(i,j) +
cy(m)*uy(i,j)
                        f0(m,i,j) = rho(i,j)*(1. + 3.*tmp +
9./2.*tmp**2 - 3./2.*u2)/9.
                        m = k*2 + 1; tmp = cx(m)*ux(i,j) +
cy(m)*uy(i,j)
                        f0(m,i,j) = rho(i,j)*(1. + 3.*tmp +
9./2.*tmp**2 - 3./2.*u2)/36.
                        end do
            end do
      end do

      do i = 1,xd
            do j = 1,yd
                  do k = 1,dd
                        tmpg = dx(k)*ux(i,j) + dy(k)*uy(i,j)
                  !!!!!!!!!!! NEED MODIFICATION
                        g0(k,i,j) = rho(i,j)*temp(i,j)*(1 + tmpg )/4
            !!!!!!!!!!! NEED MODIFICATION
                  end do
       end do
      end do

      return
      end


   subroutine collide1 (f,f0,tauv,cd,xd,yd,g,g0,tauc,dd)
!-----------------------------------------------------------------------
      real*8  f(1:cd,1:xd,1:yd), f0(1:cd,1:xd,1:yd), tauv
      real*8  g(1:dd,1:xd,1:yd), g0(1:dd,1:xd,1:yd), tauc
      integer i, j, k

    do k = 1,cd
            do i = 1,xd
                  do j = 1,yd
                        f(k,i,j) = f(k,i,j) - (f(k,i,j) -
f0(k,i,j))/tauv
                        if (f(k,i,j) .le. -1) then
                              write(*,*)i,j,'error'
                        endif
                  end do
            end do
      end do

      do k = 1,dd
            do i = 1,xd
                  do j = 1,yd
                        g(k,i,j) = g(k,i,j) - (g(k,i,j) -
g0(k,i,j))/tauc
                        if (g(k,i,j) .le. -1) then
                              write(*,*)i,j,'error'
                        endif
                  end do
            end do
      end do

      return
      end


      subroutine force (cx,cy,ux,uy,temp,f,f0,gra,cd,xd,yd)
!-----------------------------------------------------------------------
      real*8 f(1:cd,1:xd,1:yd), temp(1:xd,1:yd),cy(1:cd),cx(1:cd)
      real*8 f0(1:cd,1:xd,1:yd)
      real*8 ux(1:xd,1:yd), uy(1:xd,1:yd)
      integer i, j, k

      tempor = 0.0
      do i = 1,xd
       do j = 1,yd
         tempor = tempor + temp(i,j)
         tempave = tempor/(yd*yd)
         end do
      end do

      do k = 1,cd
      do i = 1,xd
       do j = 1,yd
         f(k,i,j) = f(k,i,j) + 3*gra*(cy(k)-
uy(i,j))*f0(k,i,j)*(temp(i,j)-tempave)
```

```
         end do
        end do
       end do

       return
       end

       subroutine stream1 (f,cd,xd,yd,g,dd)
!-----------------------------------------------------------------------
       real*8  f(1:cd,1:xd,1:yd),g(1:dd,1:xd,1:yd)
     real*8  tmp(1:cd,1:xd,1:yd),tmpg(1:dd,1:xd,1:yd)
       integer i, j, k
!
       do k = 1,cd
        do i = 1,xd
         do j = 1,yd
          tmp(k,i,j) = f(k,i,j)
         end do
        end do
       end do

       do k = 1,dd
        do i = 1,xd
         do j = 1,yd
          tmpg(k,i,j) = g(k,i,j)
         end do
        end do
       end do

       do k = 1,cd
        if(k .eq. 1) then
         do i = 1,xd; do j = 1,yd
          ii = i    ; jj = j
          f(k,ii,jj) = tmp(k,i,j)
         end do; end do

        else if(k .eq. 2) then
         do i = 1,xd-1; do j = 1,yd
          ii = i + 1; jj = j
          f(k,ii,jj) = tmp(k,i,j)
             end do; end do

        else if(k .eq. 3) then
         do i = 1,xd-1; do j = 1,yd - 1
          ii = i + 1; jj = j + 1
          f(k,ii,jj) = tmp(k,i,j)
         end do; end do

        else if(k .eq. 4) then
         do i = 1,xd; do j = 1,yd - 1
          ii = i    ; jj = j + 1
          f(k,ii,jj) = tmp(k,i,j)
         end do; end do

        else if(k .eq. 5) then
         do i = 2,xd; do j = 1,yd - 1
          ii = i - 1; jj = j + 1
          f(k,ii,jj) = tmp(k,i,j)
         end do; end do
```

```
 else if(k .eq. 6) then
  do i = 2,xd; do j = 1,yd
   ii = i - 1; jj = j
   f(k,ii,jj) = tmp(k,i,j)
  end do; end do

 else if(k .eq. 7) then
  do i = 2,xd; do j = 2,yd
   ii = i - 1; jj = j - 1
   f(k,ii,jj) = tmp(k,i,j)
  end do; end do

 else if(k .eq. 8) then
  do i = 1,xd; do j = 2,yd
   ii = i    ; jj = j - 1
   f(k,ii,jj) = tmp(k,i,j)
  end do; end do

 else if(k .eq. 9) then
  do i = 1,xd-1; do j = 2,yd
   ii = i + 1; jj = j - 1
   f(k,ii,jj) = tmp(k,i,j)
  end do; end do
 end if
end do


do k = 1,dd
 if(k .eq. 1) then
  do i = 1,xd-1; do j = 1,yd - 1
   ii = i + 1; jj = j + 1
   g(k,ii,jj) = tmpg(k,i,j)
  end do; end do

 else if(k .eq. 2) then
  do i = 2,xd; do j = 1,yd - 1
   ii = i - 1; jj = j + 1
   g(k,ii,jj) = tmpg(k,i,j)
  end do; end do

 else if(k .eq. 3) then
  do i = 2,xd; do j = 2,yd
   ii = i - 1; jj = j - 1
   g(k,ii,jj) = tmpg(k,i,j)
  end do; end do

 else if(k .eq. 4) then
  do i = 1,xd-1; do j = 2,yd
   ii = i + 1; jj = j - 1
   g(k,ii,jj) = tmpg(k,i,j)
  end do; end do

 end if
end do


return
end
```

```
      subroutine boundary1
(f,f0,rho,u0,cd,xd,yd,g,g0,temp,dd,x1,x2,y1,y2)
!-----------------------------------------------------------------------
      real*8  f(1:cd,1:xd,1:yd),f0(1:cd,1:xd,1:yd)
      real*8  g(1:dd,1:xd,1:yd),g0(1:dd,1:xd,1:yd)
      real*8  temp(1:xd,1:yd)
      real*8  u0
      real*8  rho(1:xd,1:yd)

      ! BOUNCE BACK boundary conditon bottom
      do i = 2,xd-1           !bottom
      f(4,i,1) = f(8,i,1)
      f(3,i,1) = f(7,i,1)
      f(5,i,1) = f(9,i,1)

      ! no slip bounce back
      g(1,i,1) = g(3,i,1)
      g(2,i,1) = g(4,i,1)
      end do

      do i = 2,xd-1           !upper
      f(7,i,yd) = f(3,i,yd)
      f(9,i,yd) = f(5,i,yd)
      f(8,i,yd) = f(4,i,yd)

      ! no slip bounce back
      g(3,i,yd) = g(1,i,yd)
      g(4,i,yd) = g(2,i,yd)
      end do

      ! left and right boundary condition
      do j = 2,yd-1           !left boundary
      f(2,1,j) = f(6,1,j)
      f(3,1,j) = f(7,1,j)
      f(9,1,j) = f(5,1,j)

      g(1,1,j) = g(3,1,j)
      g(4,1,j) = g(2,1,j)
      end do

      do j = 2,yd-1           !right boundary
      f(5,xd,j) = f(9,xd,j)
      f(6,xd,j) = f(2,xd,j)
      f(7,xd,j) = f(3,xd,j)

      g(2,xd,j) =g(4,xd,j)
      g(3,xd,j) =g(1,xd,j)

      end do

      ! four corner boundary condition
      f(9,1,yd) = f(5,1,yd)
      f(8,1,yd) = f(4,1,yd)
      f(2,1,yd) = f(6,1,yd)
      g(4,1,yd) = g(2,1,yd)

      f(7,xd,yd) = f(3,xd,yd)
      f(6,xd,yd) = f(2,xd,yd)
      f(8,xd,yd) = f(4,xd,yd)
      g(3,xd,yd) = g(1,xd,yd)
      f(3,1,1) = f(7,1,1)
      f(2,1,1) = f(6,1,1)
      f(4,1,1) = f(8,1,1)
      g(1,1,1) = g(3,1,1)

      f(5,xd,1) = f(9,xd,1)
      f(4,xd,1) = f(8,xd,1)
      f(6,xd,1) = f(2,xd,1)
      g(2,xd,1) = g(4,xd,1)

      return
      end


   subroutine calc1
(cx,cy,f,ux,uy,rho,u0,rho0,cd,xd,yd,temp,g,tc,th,dd,dx,dy,x1,x2,y1,y2)
!-----------------------------------------------------------------------
      real*8  cx(1:cd), cy(1:cd), f(1:cd,1:xd,1:yd)
      real*8  dx(1:dd), dy(1:dd)
      real*8  temp(1:xd,1:yd),g(1:dd,1:xd,1:yd)
      real*8      ux(1:xd,1:yd), uy(1:xd,1:yd), rho(1:xd,1:yd)
      real*8  u0,rho0,th,tc
      integer i, j, k

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      do i = 1,xd
            do j = 1,yd
                  rho(i,j) = f(1,i,j); ux(i,yd)=0.0; uy(i,j) = 0.0;
ux(i,j) = 0.0
                  do k = 2,cd
                        ux(i,j) =  ux(i,j) + f(k,i,j)*cx(k)
                        uy(i,j) =  uy(i,j) + f(k,i,j)*cy(k)
                        rho(i,j)=rho(i,j)+f(k,i,j)


                  end do
            end do
      end do

      do i = 1,xd
            do j = 1,yd
                  temp(i,j) = g(1,i,j);
                  do k = 2,dd
                        temp(i,j) = temp(i,j) + g(k,i,j)
                  end do
            end do
      end do



!!!!!!!!!!!!!!!!!!!!!!!!!
      do i = 1,xd
                  do j = 1,yd
                        if (i.eq.1) then
                              ux(i,j) = 0.0
                              uy(i,j) = 0.0
                              temp(i,j) = th
                        else
                        if (rho(i,j) .ne. 0.) then
                        ux(i,j) =  ux(i,j)/rho(i,j)
                        uy(i,j) =  uy(i,j)/rho(i,j)
                        temp(i,j) = temp(i,j)/rho(i,j)
                  else
                        ux(i,j)  = 0.
                        uy(i,j)  = 0.
                        temp(i,j) = 0.
                        end if
                        end if
                  end do
            end do


      do i= 1,xd
                  temp(i,1)=temp(i,2)                 !adiabatic right
wall
                  temp(i,yd)=temp(i,yd-1)                         !cold
left wall
      end do

    return
      end
```

# C

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Lattice Boltzmann Method     !
!   based on Square Lattice     !
!     OLD's MODEL               !
!   Natural convection          !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
   program natural_convection

    parameter(cd = 9, xd = 81, yd = 81, dd = 4) !UBAH XD, YD
=121,151,181,201
      real*8  cx(1:cd), cy(1:cd)
      real*8  dx(1:dd), dy(1:dd)
      real*8  f(1:cd,1:xd,1:yd), f0(1:cd,1:xd,1:yd),
ff(1:cd,1:xd,1:yd)
      real*8  g(1:dd,1:xd,1:yd), g0(1:dd,1:xd,1:yd),
gg(1:dd,1:xd,1:yd)
      real*8  ux(1:xd,1:yd), uy(1:xd,1:yd), rho(1:xd,1:yd)
      real*8  temp(1:xd,1:yd),sumf,sumg
      real*8  vel(1:xd,1:yd)
      real*8  u0, rho0, pi,tauv,th,tc,pt,ra,nyu,ok,di,tauc,ri,re

      u0 = 0.0                ! wall velocity
      rho0 = 1.0              ! constant density
      pt = 0.71               ! Prandtl number
      ra = 10000              ! rayleigh number
      th = 1.0                ! left hot wall
      tc = 0.0                ! right cool/initial wall temperature

      write(*,*) 'calculation start'
      write(*,*) 'nstep?'                                     !
iteration step for display
      read(*,*) nstep
      write(*,*) 'nstep =',nstep

      gra = (0.0577**2)/(yd)

      nyu = (gra*((yd-1)**3)*(th-tc)*pt/ra)**0.5
      tauv = 3*nyu + 0.5
      di    = nyu/pt    !!!!!!!!!!!!!!!!!NEED MODIFICATION
      tauc = di+0.5

      write(*,*) 'nyu  =',nyu
      write(*,*) 'di   =',di
      write(*,*) 'Tauv =',tauv
      write(*,*) 'Tauc =',tauc
      write(*,*) 'gra =',gra
      write(*,*) 'everything is ok?,press 0 if ok'
      read (*,*) ok

      call initial1
(cx,cy,f,f0,g,g0,ux,uy,rho,temp,u0,rho0,th,tc,cd,xd,yd,dx,dy,dd,x1,x2,
y1,y2)
```

```
      do iter = 1, 1000

            call collide1 (f,f0,tauv,cd,xd,yd,g,g0,tauc,dd)

            call force (cx,cy,ux,uy,temp,f,f0,gra,cd,xd,yd)

            call stream1 (f,cd,xd,yd,g,dd)

            call boundary1
(f,f0,rho,u0,cd,xd,yd,g,g0,temp,dd,x1,x2,y1,y2)

            call calc1
(cx,cy,f,ux,uy,rho,u0,rho0,cd,xd,yd,temp,g,tc,th,dd,dx,dy,x1,x2,y1,y2)

            if (mod(iter,nstep) .eq.0) then
                  write(*,*) 'time step=', iter
                  write(*,*) 'velocity=',ux(xd/4,yd/4)
                  write(*,*) 'temperature= ',temp(xd/4,yd/4)
            end if

            call equilibrium1
(cx,cy,f0,ux,uy,rho,cd,xd,yd,dx,dy,g0,temp,dd)
!!!!!!!!!!!!!!    check for convergence
      if (mod(iter,nstep) .eq.0) then
            write (*,*)'sum_f = ',sumf
                  do k = 1,cd
                        do i = 1,xd
                              do j = 1,yd
                                    ff(k,i,j) = f(k,i,j)
                           end do
                        end do
                  end do

                  write (*,*)'sum_g = ',sumg
                  do k = 1,4
                        do i = 1,xd
                              do j = 1,yd
                                    gg(k,i,j) = g(k,i,j)
                           end do
                        end do
                  end do

    endif

      if (mod(iter,nstep) .eq.1) then
                  sumf = 0.
                  do k = 1,cd
                        do i = 1,xd
                              do j = 1,yd
                                    sumf = sumf +(f(k,i,j)-
ff(k,i,j))**2
                              end do
                        end do
                  end do

                  sumf = (sumf/9.0*(yd)*(xd))**0.5

                  !***** if converge*****::!
                  if (sumf .le. 1.0d-3) then
                  write(*,*)'solution_f converge'
!                       go to 100
                  end if

                  sumg = 0.
                  do k = 1,4
                        do i = 1,xd
                              do j = 1,yd
                                    sumg = sumg +(g(k,i,j)-
gg(k,i,j))**2
                              end do
                        end do
                  end do

                  sumg = (sumg/4.0*(yd)*(xd))**0.5

                  !***** if converge*****::!
                  if (sumg .le. 1.0d-3) then

                  write(*,*)'solution_g converge'
!                       go to 100
                  end if

            if (sumg .le. 1.0d-3 .and. sumf .le. 1.0d-3) then
            write(*,*)'both solution converge'

            go to 100
            end if

            end if

      end do
100         bnu = 0.0
      open (unit=30,file='u
vel1.dat',status='replace',action='write',iostat=ierror)
      write(30,*)'thermal diffusivity      ',di
      write(30,*)'Prandtl number           ',pt
      write(30,*)'hydro relax. time        ',tauv
      write(30,*)'termo relax. time        ',tauc
      write(30,*)'solution converge at     ',iter

      do j = 1,yd

      write(30,*) ux((xd+1)/2,j)*(yd-1)/di

      end do

      close(30)

      open (unit=31,file='v
vel1.dat',status='replace',action='write',iostat=ierror)
      do i = 1,xd

      write(31,*) uy(i,(yd+1)/2)*(yd-1)/di

      end do

      close(31)
      open
(unit=32,file='variables.dat',status='replace',action='write',iostat=i
error)
      write(32,*)'x-vel, y-vel, temp'
      do j = 1,yd
      do i = 1,xd

      write(32,*) ux(i,j)*(yd-1)/di,uy(i,j)*(yd-1)/di,temp(i,j)

      end do
      end do

      close(32)
      end

      subroutine initial1
(cx,cy,f,f0,g,g0,ux,uy,rho,temp,u0,rho0,th,tc,cd,xd,yd,dx,dy,dd,x1,x2,
y1,y2)
!----------------------------------------------------------------------
      real*8  cx(1:cd), cy(1:cd), w(1:cd)
      real*8  dx(1:dd), dy(1:dd)
      real*8  f(1:cd,1:xd,1:yd), f0(1:cd,1:xd,1:yd)
      real*8  g(1:dd,1:xd,1:yd), g0(1:dd,1:xd,1:yd)
      real*8  ux(1:xd,1:yd), uy(1:xd,1:yd), rho(1:xd,1:yd)
      real*8  temp(1:xd,1:yd)
      real*8  u0,  rho0, pi,tc,th

      pi = 4.0*atan(1.0)
    cx(1) =  0.0
    cy(1) =  0.0
    do k = 2,9
      w(k) = 1.
      if(mod(k, 2) .eq. 1.) w(k) = sqrt(2.)
            cx(k) = w(k)*cos((k-2)*pi/4.0)
        cy(k) = w(k)*sin((k-2)*pi/4.0)
    enddo

      dx(1) = 1.0
      dy(1) = 0.0
      dx(2) = 0.0
      dy(2) = 1.0
      dx(3) = -1.0
      dy(3) = 0.0
      dx(4) = 0.0
      dy(4) = -1.0

      do i = 1,xd
                  do j = 1,yd
                        rho(i,j) = 1.0

                        if (i.eq.1) then
                              ux(i,j) = 0.0
                              uy(i,j) = 0.0
                              temp(i,j) = th
                        else
                              ux(i,j) = 0.0
                              uy(i,j) = 0.0
                              temp(i,j) = tc
                        end if
                  end do
            end do
      call equilibrium1 (cx,cy,f0,ux,uy,rho,cd,xd,yd,dx,dy,g0,temp,dd)

      do k = 1,cd
            do i = 1,xd
                  do j = 1,yd
                        f(k,i,j) = f0(k,i,j)
                  end do
            end do
      end do

      do k = 1,dd
            do i = 1,xd
                  do j = 1,yd
                        g(k,i,j) = g0(k,i,j)
                  end do
            end do
      end do

      return
      end

    subroutine equilibrium1
(cx,cy,f0,ux,uy,rho,cd,xd,yd,dx,dy,g0,temp,dd)
!----------------------------------------------------------------------
      real*8  cx(1:cd), cy(1:cd), f0(1:cd,1:xd,1:yd)
      real*8  dx(1:dd), dy(1:dd)
      real*8  g0(1:dd,1:xd,1:yd)
    real*8  ux(1:xd,1:yd), uy(1:xd,1:yd), rho(1:xd,1:yd)
      real*8  temp(1:xd,1:yd)
      real*8  u2, tmp
      integer i, j, k, m

    do i = 1,xd
            do j = 1,yd
                        u2 = ux(i,j)**2 + uy(i,j)**2
                        f0(1,i,j) = rho(i,j)*(1. - 3./2.*u2)*4./9.
                        do k = 1,4
                        m = k*2    ; tmp = cx(m)*ux(i,j) +
cy(m)*uy(i,j)
                        f0(m,i,j) = rho(i,j)*(1. + 3.*tmp +
9./2.*tmp**2 - 3./2.*u2)/9.
                        m = k*2 + 1; tmp = cx(m)*ux(i,j) +
cy(m)*uy(i,j)
                        f0(m,i,j) = rho(i,j)*(1. + 3.*tmp +
9./2.*tmp**2 - 3./2.*u2)/36.
                        end do
            end do
      end do

      do i = 1,xd
            do j = 1,yd
            u2 = ux(i,j)**2 + uy(i,j)**2
                  do k = 1,dd
                        tmpg = dx(k)*ux(i,j) + dy(k)*uy(i,j)

                        g0(k,i,j) = rho(i,j)*temp(i,j)*(1. +
2.*tmpg)/4.
                  end do
```

```
       end do
      end do

      return
      end


   subroutine collide1 (f,f0,tauv,cd,xd,yd,g,g0,tauc,dd)
!----------------------------------------------------------------------
      real*8  f(1:cd,1:xd,1:yd), f0(1:cd,1:xd,1:yd), tauv
      real*8  g(1:dd,1:xd,1:yd), g0(1:dd,1:xd,1:yd), tauc
      integer i, j, k

    do k = 1,cd
            do i = 1,xd
                  do j = 1,yd
                        f(k,i,j) = f(k,i,j) - (f(k,i,j) -
f0(k,i,j))/tauv
                        if (f(k,i,j) .le. -1) then
                              write(*,*)i,j,'error'
                        endif
                  end do
            end do
      end do

      do k = 1,dd
            do i = 1,xd
                  do j = 1,yd
                        g(k,i,j) = g(k,i,j) - (g(k,i,j) -
g0(k,i,j))/tauc
                        if (g(k,i,j) .le. -1) then
                              write(*,*)i,j,'error'
                        endif
                  end do
            end do
      end do

      return
      end


      subroutine force (cx,cy,ux,uy,temp,f,f0,gra,cd,xd,yd)
!----------------------------------------------------------------------
      real*8 f(1:cd,1:xd,1:yd), temp(1:xd,1:yd),cy(1:cd),cx(1:cd)
      real*8 f0(1:cd,1:xd,1:yd)
      real*8 ux(1:xd,1:yd), uy(1:xd,1:yd)
      integer i, j, k

      tempor = 0.0
      do i = 1,xd
       do j = 1,yd
         tempor = tempor + temp(i,j)
         tempave = tempor/(yd*yd)
         end do
      end do

      do k = 1,cd
      do i = 1,xd
       do j = 1,yd
```

```
          f(k,i,j) = f(k,i,j) + 3*gra*(cy(k)-
uy(i,j))*f0(k,i,j)*(temp(i,j)-tempave)
         end do
        end do
       end do

       return
       end

       subroutine stream1 (f,cd,xd,yd,g,dd)
!-----------------------------------------------------------------------
       real*8  f(1:cd,1:xd,1:yd),g(1:dd,1:xd,1:yd)
     real*8  tmp(1:cd,1:xd,1:yd),tmpg(1:dd,1:xd,1:yd)
       integer i, j, k
!
       do k = 1,cd
        do i = 1,xd
         do j = 1,yd
          tmp(k,i,j) = f(k,i,j)
         end do
        end do
       end do

       do k = 1,dd
        do i = 1,xd
         do j = 1,yd
          tmpg(k,i,j) = g(k,i,j)
         end do
        end do
       end do

       do k = 1,cd
        if(k .eq. 1) then
         do i = 1,xd; do j = 1,yd
          ii = i    ; jj = j
          f(k,ii,jj) = tmp(k,i,j)
         end do; end do

        else if(k .eq. 2) then
         do i = 1,xd-1; do j = 1,yd
          ii = i + 1; jj = j
          f(k,ii,jj) = tmp(k,i,j)
             end do; end do

        else if(k .eq. 3) then
         do i = 1,xd-1; do j = 1,yd - 1
          ii = i + 1; jj = j + 1
          f(k,ii,jj) = tmp(k,i,j)
         end do; end do

        else if(k .eq. 4) then
         do i = 1,xd; do j = 1,yd - 1
          ii = i    ; jj = j + 1
          f(k,ii,jj) = tmp(k,i,j)
         end do; end do

        else if(k .eq. 5) then
         do i = 2,xd; do j = 1,yd - 1
          ii = i - 1; jj = j + 1
          f(k,ii,jj) = tmp(k,i,j)
```

```
  end do; end do

 else if(k .eq. 6) then
  do i = 2,xd; do j = 1,yd
   ii = i - 1; jj = j
   f(k,ii,jj) = tmp(k,i,j)
  end do; end do

 else if(k .eq. 7) then
  do i = 2,xd; do j = 2,yd
   ii = i - 1; jj = j - 1
   f(k,ii,jj) = tmp(k,i,j)
  end do; end do

 else if(k .eq. 8) then
  do i = 1,xd; do j = 2,yd
   ii = i    ; jj = j - 1
   f(k,ii,jj) = tmp(k,i,j)
  end do; end do

 else if(k .eq. 9) then
  do i = 1,xd-1; do j = 2,yd
   ii = i + 1; jj = j - 1
   f(k,ii,jj) = tmp(k,i,j)
  end do; end do
 end if
end do


do k = 1,dd
 if(k .eq. 1) then
  do i = 1,xd-1; do j = 1,yd
   ii = i + 1; jj = j
   g(k,ii,jj) = tmpg(k,i,j)
  end do; end do

 else if(k .eq. 2) then
  do i = 1,xd; do j = 1,yd - 1
   ii = i ; jj = j + 1
   g(k,ii,jj) = tmpg(k,i,j)
  end do; end do

 else if(k .eq. 3) then
  do i = 2,xd; do j = 1,yd
   ii = i - 1; jj = j
   g(k,ii,jj) = tmpg(k,i,j)
  end do; end do

 else if(k .eq. 4) then
  do i = 1,xd; do j = 2,yd
   ii = i ; jj = j - 1
   g(k,ii,jj) = tmpg(k,i,j)
  end do; end do

 end if
end do


return
end
```

```
      subroutine boundary1
(f,f0,rho,u0,cd,xd,yd,g,g0,temp,dd,x1,x2,y1,y2)
!-----------------------------------------------------------------------
      real*8  f(1:cd,1:xd,1:yd),f0(1:cd,1:xd,1:yd)
      real*8  g(1:dd,1:xd,1:yd),g0(1:dd,1:xd,1:yd)
      real*8  temp(1:xd,1:yd)
      real*8  u0
      real*8  rho(1:xd,1:yd)

      ! BOUNCE BACK boundary conditon bottom
      do i = 2,xd-1           !bottom
      f(4,i,1) = f(8,i,1)
      f(3,i,1) = f(7,i,1)
      f(5,i,1) = f(9,i,1)

      ! no slip bounce back
      g(2,i,1) = g(4,i,1)
      end do

      do i = 2,xd-1           !upper
      f(7,i,yd) = f(3,i,yd)
      f(9,i,yd) = f(5,i,yd)
      f(8,i,yd) = f(4,i,yd)

      ! no slip bounce back
      g(4,i,yd) = g(2,i,yd)
      end do

      ! left and right boundary condition
      do j = 2,yd-1           !left boundary
      f(2,1,j) = f(6,1,j)
      f(3,1,j) = f(7,1,j)
      f(9,1,j) = f(5,1,j)

      g(1,1,j) = g(3,1,j)
      end do

      do j = 2,yd-1           !right boundary
      f(5,xd,j) = f(9,xd,j)
      f(6,xd,j) = f(2,xd,j)
      f(7,xd,j) = f(3,xd,j)

      g(3,xd,j) =g(1,xd,j)
      end do

      ! four corner boundary condition
      f(9,1,yd) = f(5,1,yd)
      f(8,1,yd) = f(4,1,yd)
      f(2,1,yd) = f(6,1,yd)
      g(4,1,yd) = g(2,1,yd)
      g(1,1,yd) = g(3,1,yd)

      f(7,xd,yd) = f(3,xd,yd)
      f(6,xd,yd) = f(2,xd,yd)
      f(8,xd,yd) = f(4,xd,yd)
      g(3,xd,yd) = g(1,xd,yd)
      g(4,xd,yd) = g(2,xd,yd)
      f(3,1,1) = f(7,1,1)
      f(2,1,1) = f(6,1,1)
      f(4,1,1) = f(8,1,1)
      g(1,1,1) = g(3,1,1)
      g(2,xd,yd) = g(4,xd,yd)

      f(5,xd,1) = f(9,xd,1)
      f(4,xd,1) = f(8,xd,1)
      f(6,xd,1) = f(2,xd,1)
      g(2,xd,1) = g(4,xd,1)
      g(3,xd,yd) = g(1,xd,yd)

      return
      end


   subroutine calc1
(cx,cy,f,ux,uy,rho,u0,rho0,cd,xd,yd,temp,g,tc,th,dd,dx,dy,x1,x2,y1,y2)
!----------------------------------------------------------------------
      real*8  cx(1:cd), cy(1:cd), f(1:cd,1:xd,1:yd)
      real*8  dx(1:dd), dy(1:dd)
      real*8  temp(1:xd,1:yd),g(1:dd,1:xd,1:yd)
      real*8      ux(1:xd,1:yd), uy(1:xd,1:yd), rho(1:xd,1:yd)
      real*8  u0,rho0,th,tc
      integer i, j, k

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


      do i = 1,xd
            do j = 1,yd
                  rho(i,j) = f(1,i,j); ux(i,yd)=0.0; uy(i,j) = 0.0;
ux(i,j) = 0.0
                  do k = 2,cd
                        ux(i,j) =  ux(i,j) + f(k,i,j)*cx(k)
                        uy(i,j) =  uy(i,j) + f(k,i,j)*cy(k)
                        rho(i,j)=rho(i,j)+f(k,i,j)


                  end do
            end do
      end do

      do i = 1,xd
            do j = 1,yd
                  temp(i,j) = g(1,i,j);
                  do k = 2,dd
                        temp(i,j) = temp(i,j) + g(k,i,j)
                  end do
            end do
      end do




!!!!!!!!!!!!!!!!!!!!!!!!!
      do i = 1,xd
                  do j = 1,yd
                        if (i.eq.1) then
                              ux(i,j) = 0.0
                              uy(i,j) = 0.0
                              temp(i,j) = th
                        else
                        if (rho(i,j) .ne. 0.) then
                        ux(i,j) =  ux(i,j)/rho(i,j)
                        uy(i,j) =  uy(i,j)/rho(i,j)
                        temp(i,j) = temp(i,j)/rho(i,j)
                  else
                        ux(i,j)  = 0.
                        uy(i,j)  = 0.
                        temp(i,j) = 0.
                        end if
                        end if
                  end do
            end do


      do i= 1,xd
                  temp(i,1)=temp(i,2)                  !adiabatic right
wall
                  temp(i,yd)=temp(i,yd-1)                          !cold
left wall
      end do

    return
      end
```