UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **FINITE DIFFERENCE OF THERMAL LATTICE BOLTZMANN SCHEME FOR THE SIMULATION OF NATURAL CONVECTION HEAT TRANSFER**

SESI PENGAJIAN: **2009/2010**

Saya **SITI AISYAH-AWANIS BINTI MOHD YUSOFF (870201-29-5546)**
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

☐	**SULIT**	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD**	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
√	**TIDAK TERHAD**	

Disahkan oleh:

_____ (TANDATANGAN PENULIS)

_____ (TANDATANGAN PENYELIA)

Alamat Tetap:

D/A HIM MOTOR, KG PINTU GERBANG, JLN TAWANG, 16020 BACHOK , KELANTAN

MUHAMAD ZUHAIRI BIN SULAIMAN
(Nama Penyelia)

Tarikh: **20 NOVEMBER 2009**

Tarikh: : **20 NOVEMBER 2009**

CATATAN:
- * Potong yang tidak berkenaan.
- ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
- ♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

FINITE DIFFERENCE OF THERMAL LATTICE BOLTZMANN SCHEME FOR THE SIMULATION OF NATURAL CONVECTION HEAT TRANSFER

SITI AISYAH-AWANIS BINTI MOHD YUSOFF

Report submitted in partial fulfilment of the requirements
for the award of the degree of
Bachelor of Mechanical Engineering

Faculty of Mechanical Engineering
UNIVERSITI MALAYSIA PAHANG

NOVEMBER 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: 20 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 concurrently submitted for award of other degree.

Signature
Name: SITI AISYAH-AWANIS BINTI MOHD YUSOFF
ID Number: MA06039
Date: 20 NOVEMBER 2009

ACKNOWLEDGEMENT

In the name of Allah s.w.t, the most Gracious, the Ever Merciful Praise is to Allah, Lord of the Universe and Peace and Prayers be upon His final prophet and Messenger Muhammad s.a.w.

I am heartily thankful to my supervisor, Mr. Muhamad Zuhairi bin Sulaiman, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of this project. I also would like to express very special thanks to my former supervisor, Mr. Mohd Rosdzimin bin Abdul Rahman for the suggestions and co-operation throughout the study. I also sincerely thanks for the time spent proofreading and correcting my many mistakes.

I reserve my sincere thanks for my family members. I am deeply indebted to my father, Mohd Yusoff bin Mat Leh, my mother, Rabiah binti Abdullah, my sister, Siti Mariam-Nabilah and my youngest sister, Siti Fatimah Najibah for their never ending love, dedication, support and faith in me. Without them, this project would not been done.

Special thanks should be given to my committee members. Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

ABSTRACT

In this thesis, a method of lattice Boltzmann is introduced. Lattice Boltzmann method (LBM) is a class of computational fluid dynamics (CFD) methods for fluid simulation. Objective of this thesis is to develop finite difference lattice Boltzmann scheme for the natural convection heat transfer. Unlike conventional CFD methods, the lattice Boltzmann method is based on microscopic models and macroscopic kinetic equation. The lattice Boltzmann equation (LBE) method has been found to be particularly useful in application involving interfacial dynamics and complex boundaries. First, the general concept of the lattice Boltzmann method is introduced to understand concept of Navier-Strokes equation. The isothermal and thermal lattices Boltzmann equation has been directly derived from the Boltzmann equation by discretization in both time and phase space. Following from this concept, a few simple isothermal flow simulations which are Poiseulle flow and Couette flow were done to show the effectiveness of this method. Beside, numerical result of the simulations of Porous Couette flow and natural convection in a square cavity are presented in order to validate these new thermal models. Lastly, the discretization procedure of Lattice Boltzmann Equation (LBE) is demonstrated with finite difference technique. The temporal discretization is obtained by using second order Rungge-Kutta (modified) Euler method from derivation of governing equation. The discussion and conclusion will be presented in chapter five.

ABSTRAK

Di dalam tesis ini, kaedah kekisi *Boltzmann* diperkenalkan. Kaedah kekisi *Boltzmann* (LBM) ialah dikelaskan daripada kaedah pengiraan dinamik bendalir berkomputer (CFD) untuk simulasi bendalir. Objektif tesis ini ialah untuk membangunkan kaedah pembezaan terhingga kekisi *Boltzmann* untuk pemanasan semulajadi pemindahan haba. Tidak seperti kaedah CFD, kaedah kekisi *Boltzmann* ialah berdasarkan model mikroskopik dan persamaan kinetik makroskopik. Kaedah persamaan kekisi *Boltzmann* ditemui untuk kegunaan terutamanya di dalam aplikasi yang melibatkan hubungkait dinamik dan garisan sempadan yang rumit. Permulaannya, konsep umum kaedah kekisi *Boltzman* diperkenalkan untuk memahami konsep persamaan *Navier-Strokes* iaitu dengan diskritasikan persamaan *Boltzmann* terhadap masa dan ruang fasa, persamaan untuk isoterma dan terma kekisi *Boltzmann* dapat diterbitkan. Berdasarkan konsep ini, contoh yang mudah daripada pengaliran isoterma ialah pengaliran *Poiseulle* dan pengaliran *Couette* telah dijalankan untuk menunjukkan keberkesanan kaedah ini. Selain itu, keputusan berangka daripada simulasi pengaliran *Porous Couette* dan pemanasan semulajadi di dalam ruang segiempat diperkenalkan untuk mengesahkan model terma yang baru. Akhir sekali, diskritasikan prosedur persamaan kekisi *Boltzmaan* diperkenalkan dengan menggunakan teknik pembezaan terhingga. Hasil daripada pemberolehan persaman *governing*, masa diskritasi ini diperolehi dengan menggunakan kaedah susuan kedua *Rungge-Kutta(modified) Euler*. Perbincangan dan kesimpulan akan dibentangkan di dalam bab lima.

TABLE OF CONTENTS

CHAPTER 5 CONCLUSION AND RECOMMENDATIONS

LIST OF TABLE

LIST OF FIGURES

LIST OF SYMBOLS

R	Gas constant
t	Time
T	Temperature
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
υ	Shear viscosity
β	Thermal expansion coefficient
ε	Internal energy
ρ	Density
T_∞	Infinity temperature
T_f	Film temperature
T_s	Surface temperature
A	Area of contact A

η	Proportionally constant
χ	Thermal diffusivity
Ω	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
LGCA	Lattice gas cellular automata
PBC	Periodic Boundary condition
PDEs	Partial differential equations

CHAPTER 1

INTRODUCTION

1.1 NAVIER-STROKES EQUATION

The Navier-strokes equation can derive as 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 derivation of the Navier–Stokes equations begins with the conservation of mass, momentum, and energy being written for an arbitrary control volume. These equations describe how the velocity, pressure, temperature, and density of a moving fluid are related. The mathematical relationship governing fluid flow is the famous continuity equation and Navier-strokes equation is given by

$$\nabla . \boldsymbol{u} = 0 \tag{1.1}$$

$$\frac{\partial u}{\partial t} + \boldsymbol{u}\nabla . \boldsymbol{u} = -\nabla P + \left(\frac{2\tau - 1}{6} \right) \nabla^2 \boldsymbol{u} \tag{1.2}$$

The Navier-Stroke equation is nonlinear partial differential equations in almost every real situation and so complex that there is currently no analytical solution to them except for a small number of special cases. 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. (C.S. Nor Azwadi, 2007)

Nowadays, the use of a computer is necessary to determine the fluid motion of a particular problem because fluid related problems arising in science and engineering are extremely complex by nature.

1.2 COMPUTATIONAL FLUID DYNAMICS

Computational Fluid Dynamic (CFD) is a based on the fundamental governing equation of fluid dynamics-the continuity, momentum and energy equations. The most fundamental consideration in CFD is how one treats a continuous fluid in a discretized fashion on a computer. One method is to discretize the spatial domain into small cells to form a volume mesh or grid, and then apply a suitable algorithm to solve the Navier-Strokes equation or an equation derived from them. To exactly simulate fluid flow in a computer it would be necessary to solve Navier-Stroke equation with infinite accuracy .In reality, numerical researchers must choose a method to discretize the problem. Some of the general numerical methods used in computational fluid dynamics are described here.

1.3 LATTICE BOLTZMANN METHOD (LBM)

The Lattice Boltzmann method (LBM) is a recently developed method for simulating fluid flows and modeling physics in fluids. Unlike the traditional CFD methods, which solve the conservation equations of macroscopic properties (i.e., mass, momentum, and energy) numerically, LBM models the fluid consisting of fictive particles, and such particles perform consecutive propagation and collision processes over a discrete lattice mesh. Due to its particulate nature and local dynamics, LBM has several advantages over other conventional CFD methods, especially in dealing with complex boundaries, incorporating of microscopic interactions, and parallelization of the algorithm. It is also

known as an alternative approach to the well-known finite difference, finite element and finite volume technique for solving the Navier-Strokes equations. 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 Boltzmann 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 LBM recognizes that Boltzmann's transport equation is a computational tool that can be solved on the lattice .The collision term of this equation can be simplified using the Bhatnagar-Gross-Krook (BGK) approximation where the distribution function relaxes to a local equilibrium with a constant relaxation time. 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. (Xiaoyi He and Li-Shi Luo ,1997)

Although LBM approach treats gases and liquids as systems consisting of individual particles, the primary goal is to build the connection between the microscopic and macroscopic dynamics, rather than to deal with macroscopic dynamics directly. In other words, the goal is to derive macroscopic equations from microscopic dynamics by means of statistic, rather than to solve macroscopic equation.

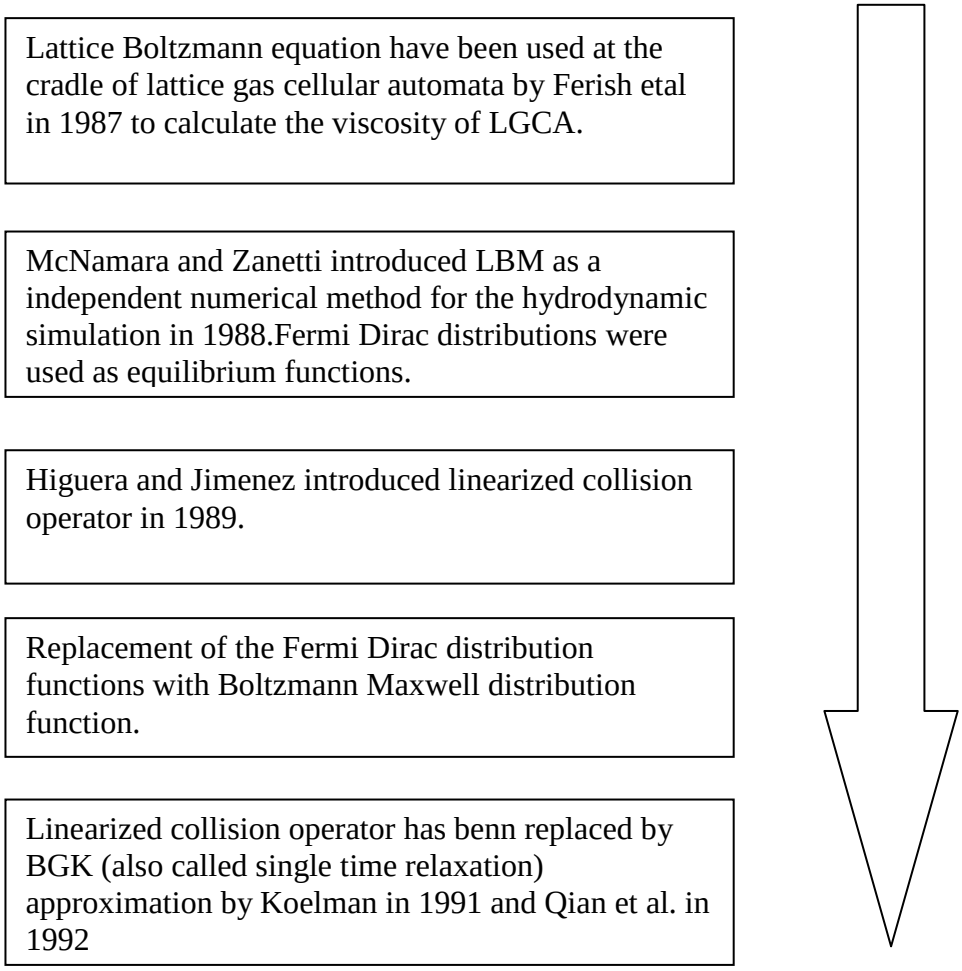


Figure 1.1: Historically stages in the development of lattice Boltzmann model

Source: Wolf Gladrow, 2000

1.4 CLASSICAL CFD VERSUS LATTICE BOLTZMANN METHODS

The conventional simulation of fluid flow and other physical processes generally starts from non linear partial differential equation (PDEs). These PDEs are discretized by finite differences, finite element finite volume or spectral methods. The resulting

algebraic equations of ordinary differential equation are solved by standard numerical methods. In LBM, the starting point is a discrete microscopic model which by construction conservation equation of mass and momentum for Navier-Stokes equation. The derivation of the corresponding macroscopic equation requires multi-scale analysis. (Wolf Gladrow, 2000)

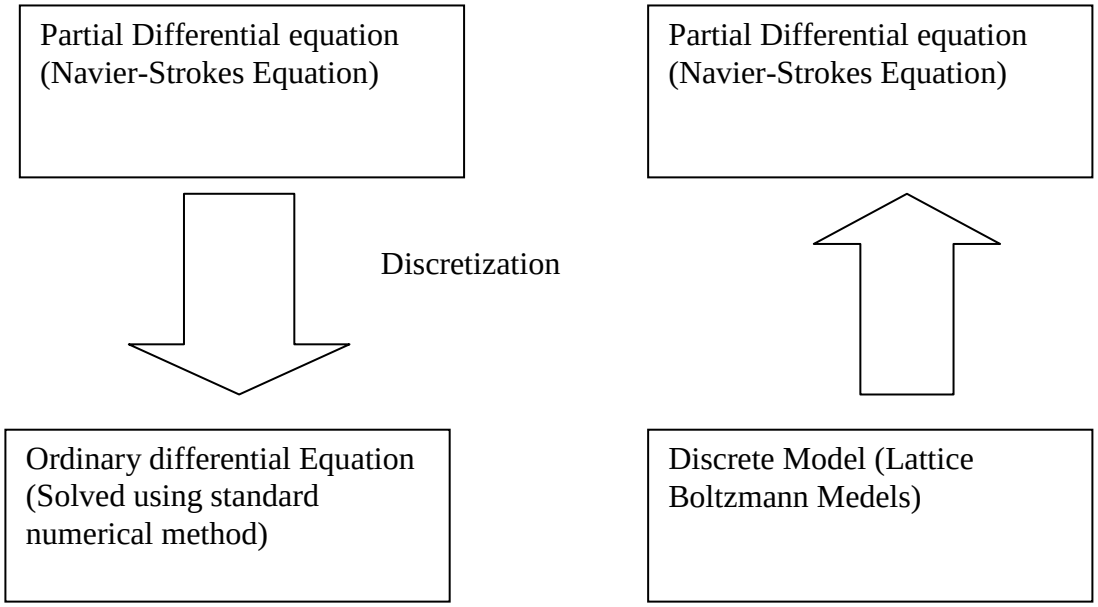


Figure 1.2: Classical CFD versus LBM

1.5 PROJECT OBJECTIVE

To develop finite difference Lattice Boltzmann Scheme for the Natural Convection Heat Transfer

1.6 PROJECT SCOPES

The first project scope is to analysis heat transfer limit to natural convection only. The second project scope is the problem will be test at Rayleigh number, Ra $=10^3$ to 10^5.This limitation due to Lattice Boltzmann Method (LBM) cam perform well at low Rayleigh number and at high Rayleigh number LBM having a problem. This selection high Rayleigh number is to show that this scheme can simulate problem at high Rayleigh number. For the last project scope is to simulate natural convection in a

square cavity. Detail characteristic numerical value of the flow will be carrying out; isotherm line, stream line and average of Nusselt number where they occur will be compared.

1.7 PROJECT BACKGROUND

The lattice Boltzmann Method (LBM) is an alternative approach to the well-known finite difference, finite element and finite volume techniques for solving the Navier-Strokes equations. LB scheme is a scheme evolved from the improvement of lattice gas automata (LGA) and inherits some features from its precursor, the LGA. The implementation of the Bhatnagar-Gross-Krook (BGK) approximation is a improvement to enhance the computational efficiency has been made for the LB method. The algorithm is simple and can also easily modify to allow for the application of other, more complex simulation component. In mathematics, finite difference methods are numerical methods for approximating the solution to difference equation using finite difference equation to approximate derivatives. Finite difference lattice Boltzmann method is obtained using second order Runge-Kutta (modified) Euler Method.

1.8 THESIS OUTLINE

The aim of this thesis is to study the methods of the lattice Boltzmann equations in order by using finite difference method. These subjects, newly emerged in 1980's utilize the statistical mechanics of simple discrete models to simulate complex physical systems. The theory of lattice Boltzmann method in 4-discrete velocity and 9-discrete velocity are reviewed in detail.

In the Chapter two, the concept of a distribution function is considered and the derivation and the theory of the classical Boltzmann equation are discussed briefly. Then the theory of lattice Boltzmann method and its coefficients from the Boltzmann equation (Chapman-Enskog expansion) are also presented.

For the four discrete velocities is found in the isothermal model and for the 9-discrete velocity, happen in thermal model that will discuss in chapter three. By using 4-

type of discrete velocity, we can apply to develop Porous Couette flow problem for the thermal fluids problem. Using 9-discrete velocity model can apply in Poiseulle flow and Couette flow

In Chapter four, the simulation by using Finite difference method was applied among the first approaches applied to the numerical solution of differential equations. This method is directly applied to the differential form of the governing equations. The finite difference is a second order equation upwind scheme.

Finally in chapter five, conclusions and discussion on future studies are presented and also some of recommendations will elaborate in order to improve the project on future.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

For the literature review will consist the theory of Lattice Boltzmann Model (LBM) that a content are governing equation, basic principle of LBM, collide function Bhatnagar-Gross-Krook (BGK), equilibrium distribution function, time relaxation, discretization of microscopic velocity and lastly is a derivation of Navier-Strokes equation. For the derivation of Navier-Strokes equation was already state in Chapter 1. Beside, derivation of two type of boundary condition which are Bounce-back and also periodic also presented. For this chapter also, we state briefing about Isothermal Lattice Boltzmann Model which are Poiseulle Flow and Couette Flow and Thermal Lattice Boltzmann Model which is Porous Couette Flow.

LBM is a computational fluid dynamics (CFD) method or alternative method to simulate the fluid problems especially to simulate the complex fluid flow problems including single and multiphase flow in complex geometries. The main objective to achieve is to create a connection between the microscopic and macroscopic dynamics. From the Navier-Strokes equation, we already know the density, velocity, pressure and etc. So, this is called macroscopic scale. But, for the Boltzmann equation, we only have the particles that moves to one place to another and called microscopic scale

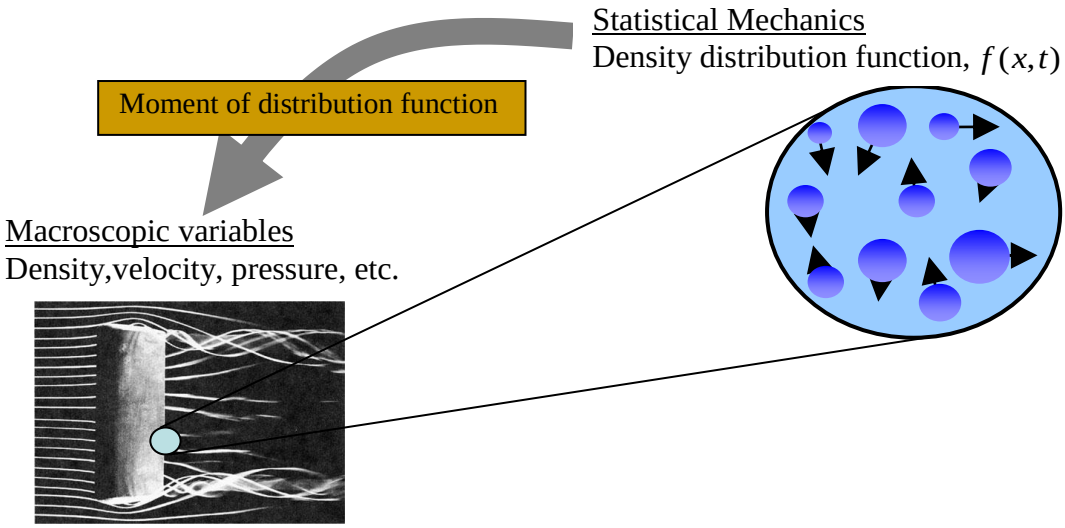


Figure 2.1: General concept of Lattice Boltzmann

Source: C.S. Nor Azwadi, 2007

2.2 GOVERNING EQUATION

From this equation, we can the relation between microscopic and macroscopic dynamics. The Boltzmann equation given is shown below:-

$$f(x+c\Delta t, t+\Delta t) - f(x,t) = \Omega(f) \tag{2.1}$$

Where f = density distribution function

c = microscopic velocity

Ω = collision integral

2.3 BASIC PRINCIPLE

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).Example is from the 'snooker'. From this situation, means when a ball hit to another ball, its can firstly streaming and then it will collision and it become streaming again.

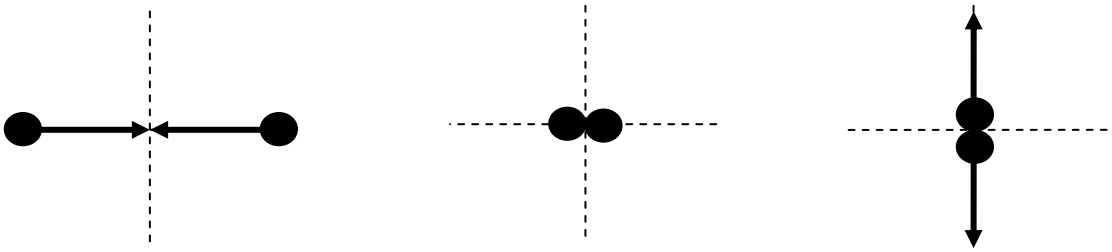


Figure 2.2: Streaming and collision processes

Source: C.S. Nor Azwadi, 2007

It represents the minimal from of Boltzmann kinetic equation (Higuera and Jimenez, 1989) and the result is a very elegant and simple equation. (F.Kuznik et al. 2007). Lattice Boltzmann equation is directly obtained from the lattice gas automata by taking ensemble average with the assumption of random phase and leads to the following equation. (Wolf Gladrow, 2000)

$$f(x+c\Delta t, c+a\Delta t, t+\Delta t) - f(x,c,t) = \Omega(f) \qquad (2.2)$$

Where $f(x,c,t)$ = the single particle distribution function with discrete velocity, c

$\Omega(f)$ = the lattice Boltzmann collision operator

Distribution function $f(x,c,t)$ describe the number of particles at position x,moving with velocity, c at time, t. There are two conditions related to the distribution function; without collisions and with collisions. At a short time,Δt each particle would move from x to $x + c\Delta t$ and each particle velocity would change from c to $c + a\Delta t$ where a is the acceleration due to external forces on a particle at x with a velocity$x$. Hence, the number of molecules $f(x,c,t)dxdc$ is equal to the number of molecules $f(x + c\Delta t, c + a\Delta t, t + \Delta t)dxdc$ for the distribution without collisions. (N. A. C. Sidik, 2007). Therefore;

$$f(x+c\Delta t, c+a\Delta t, t+\Delta t)dxdc - f(x,c,t) = 0 \qquad (2.3)$$

There will be a net difference between the number of molecules $f(x,c,t)dxdc$ and the number of molecules $f(x + c\Delta t, c + a\Delta t, t + \Delta t)dxdc$ if collision occurs between the molecules.

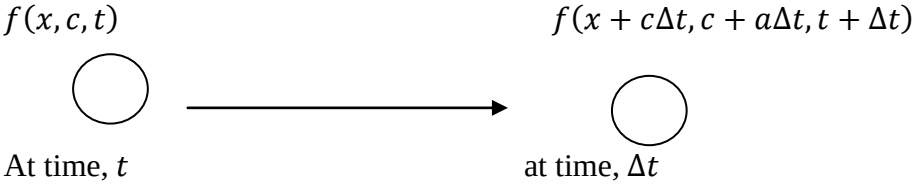
This can be expressed by;

$$f(x+c\Delta t, c+a\Delta t, t+\Delta t)dxdc - f(x,c,t)dxdc = \Omega(f) \qquad (2.4)$$

Where $\Omega(f)dxdcdt$ is the collision operator. On dividing by $dcdt$, and letting dt tends to zero $(dt \sim 0)$ give the Boltzmann equation for f ;

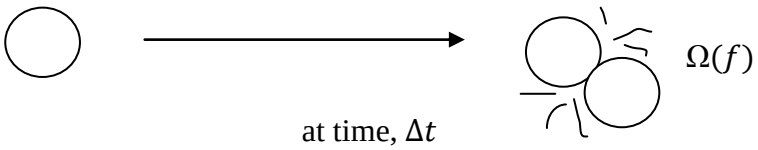
$$\frac{\delta f}{\delta t} + c_a \frac{\delta f}{\delta c_a} + a\frac{\delta f}{\delta c_a} = \Omega(f) \qquad (2.5)$$

Without collision;



With collision;

$f(x, c, t)$ $f(x + c\Delta t, c + a\Delta t, t + \Delta t)$



At time, t at time, Δt

2.4 COLLISION INTEGRAL, Ω

The BGK collision model assumes that the system is at near equilibrium state and the particle distribution function relaxes to its equilibrium state at a constant rate. It term refers to a collision operator used in the Boltzmann Equation and in the Lattice Boltzmann method, a Computational fluid dynamics technique. Boltzmann's equation describes the evolution of molecules in rare gas. If no external force is present, then after a long time the gas should reach an equilibrium state. Boltzmann's equation describes such behavior. Boltzmann conceived the H-theorem to explain how a many-body system to approach equilibrium from an arbitrary non-equilibrium initial state. He derived this theorem based on the assumption of 'molecular chaos' where the two colliding molecule's position and velocities were statistically uncorrelated or unrelated. (S. Succi, 2001)

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

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 related to f^{eq} and τ is a relaxation parameter.

2.5 TIME RELAXATION

To achieve the equilibrium, means τ is a relaxation parameter (time to reach equilibrium state during every collision process) and value of the time relaxation is between (0.5< τ <1). Below shown the time relaxation concept:

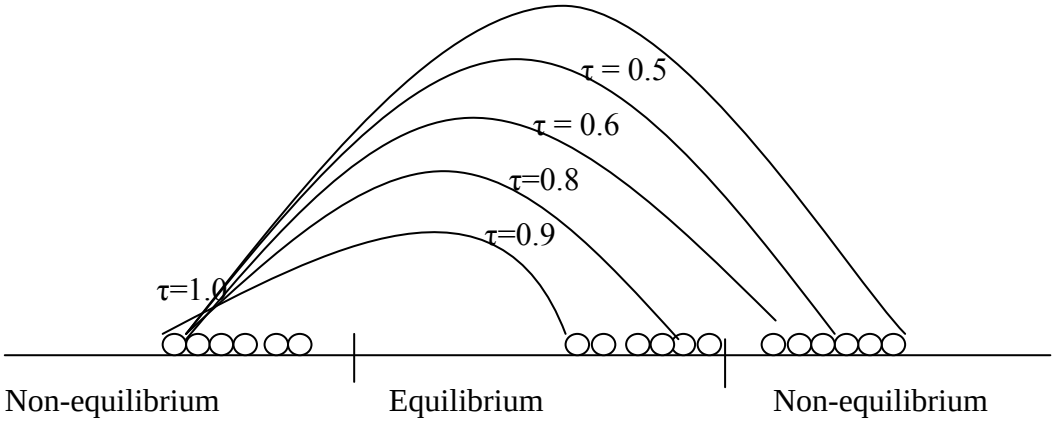


Figure 2.3: Time relaxation concept

The figure above show value of τ = 0.5 is the limit for the relaxation time. The value of time relaxation τ need to be more close to 1. The more close time relaxation τ to 1, the more number of particles exchange to equilibrium state.

2.6 DISCRETIZATION OF MICROSCOPIC VELOCITY

For the discretization 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. Means that for the 9-discrete velocity happen in isothermal fluid flow (poiseulle flow and couette flow) while the 4-discrete velocity happen in thermal fluid flow (porous couette flow).

2.6.1. Isothermal Fluid Flow

Figure below (Figure 2.4) shows the Lattice Boltzmann Isothermal Model (Poiseulle Flow and Couette flow)

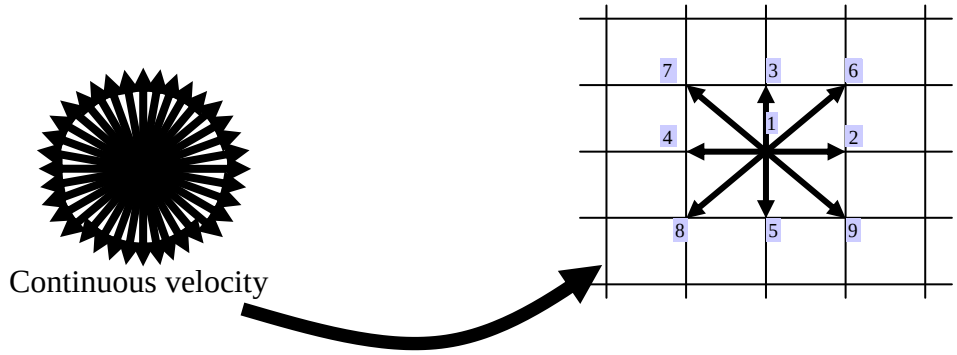


Figure 2.4: Lattice Boltzmann Isothermal Model

Source: C.S. Nor Azwadi, 2007

2.6.1.1 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:

$$\nabla . \boldsymbol{u} = 0 \tag{2.7}$$

$$\frac{\partial u}{\partial t} + \boldsymbol{u}\nabla.\boldsymbol{u} = -\nabla P + \left(\frac{2\tau - 1}{6}\right)\nabla^2 \boldsymbol{u} \tag{2.8}$$

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

$$\upsilon = \frac{2\tau - 1}{6} \tag{2.9}$$

2.6.2 Thermal Fluid Flow

Figure 2.5 is shown the Lattice Boltzmann Thermal Model means Porous Couette Flow

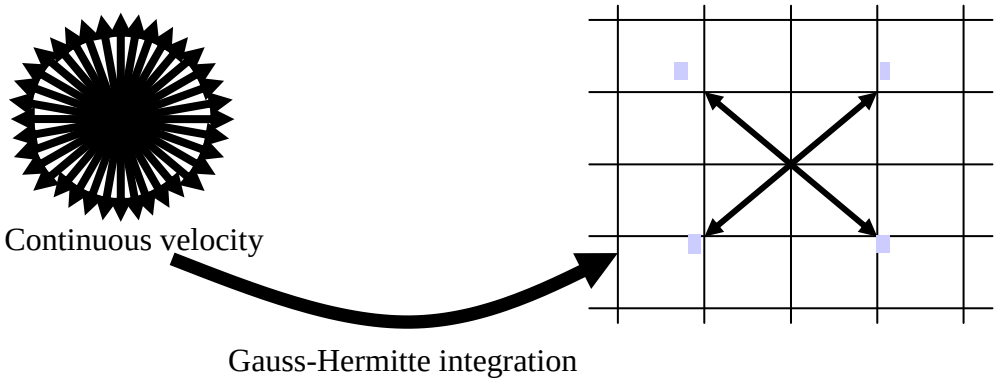


Figure 2.5: Lattice Boltzmann Thermal Model

Source: C.S. Nor Azwadi, 2007

2.6.2.1. The Macroscopic Equation for Thermal

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

$$\frac{\partial T}{\partial t} + \nabla.(\boldsymbol{u}T) = (\tau_g - \frac{1}{2})\nabla^2 T \tag{2.10}$$

Where:

$$\tau_f = 3\upsilon + \frac{1}{2} \tag{2.11}$$

$$\tau_g = \chi + \frac{1}{2} \tag{2.12}$$

2.7 BHATNAGAR-GROSS-KROOK (BGK)

BGK is a combination step between streaming process and collision process. These two processes are repeating one after another until all the distribution function relaxes to the equilibrium distribution function.（S. Succi, 2001)

$$\underbrace{f(x + c\Delta t, t + \Delta t) - f(x,t)}_{\text{Streaming process}} = \underbrace{-\frac{f - f^{eq}}{\tau}}_{\text{Collision process}} \tag{2.13}$$

$f_{i:}$ density distribution function

τ: relaxation parameter

f^{eq}: equilibrium distribution

2.8 BOUNDARY CONDITION

The boundary conditions are responsible to determining these unknown distributions. Basically, there are two ways to define boundary conditions; placing the boundary on grid modes or placing the boundary on links (Xiaoyi he et al. 1995). Lattice Boltzmann method have several types of boundary conditions.

2.8.1 Bounce-Back

The initial approach to simulate boundary was to follow the methods used in the lattice Gas Approach. The simplest boundary condition is called bounce back boundary conditions where all the distribution functions at the boundaries back along to the link they arrived. The bounce-back boundary condition for lattice Boltzmann simulations is evaluated for flow about an infinite periodic array of cylinders. The solution is compared with results from a more accurate boundary condition formulation for the lattice Boltzmann method and with finite difference solutions. The bounce-back boundary condition is used to simulate boundaries of cylinders with both circular and octagonal cross-sections. Figure 2.6 below show the schematic plot of the bounce-back boundary condition. (Gallivan, Martha A., Noble, David R, Georgiadis, John G, Buckius, Richard O, 1997)

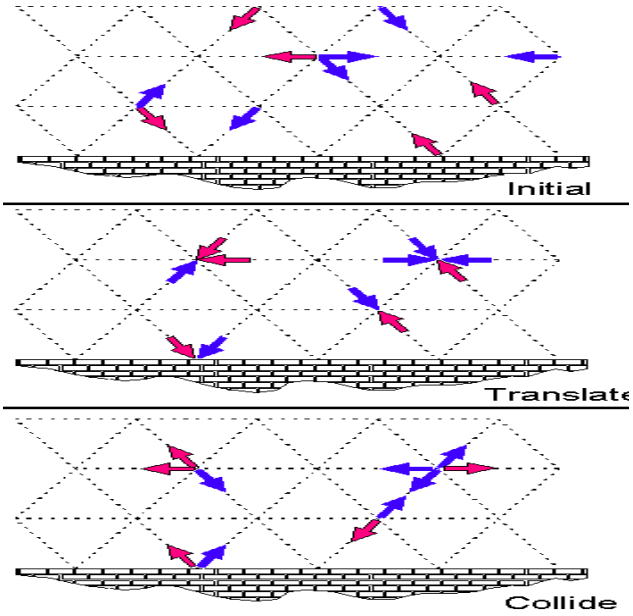


Figure 2.6: Schematic plot of bounce back boundary condition

Source: Frisch et al., 1986

2.8.2 Periodic Boundary

Periodic boundary conditions typically intended to isolate bulk phenomena from the actual boundaries of the real physical system and consequently they are adequate for physical phenomena where surface effect play a negligible role. Periodic boundary conditions are applied directly to the particle populations, and not to macroscopic flow variables. They are generally useful for capturing flow invariance in a given direction. If a uniform body force is used instead of an imposed pressure gradient, periodic conditions can be used in place of macroscopic inflow/outflow conditions in the stream wise direction [Robert S. Maier et al.1996]. This boundary condition can be implementing by bring the same distribution function that leaving the outlet to the inlet. This can review with reference to Figure 2.7 that show a particles flow.

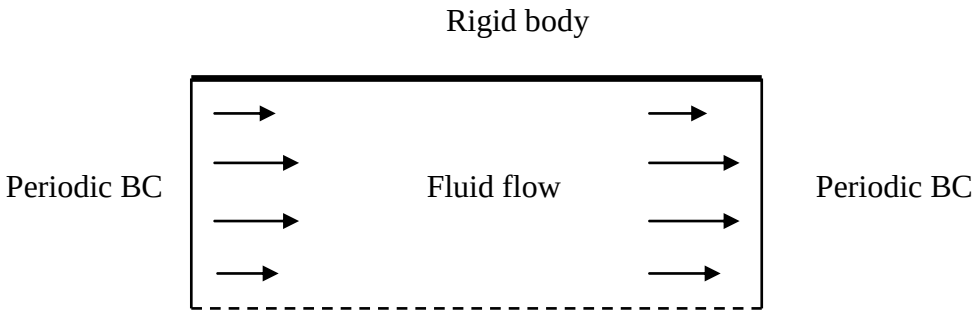


Figure 2.7: Periodic boundary condition

2.9 THEORY OF POISEULLE FLOW

The Hagen-Poiseuille equation is a physical law that describes slow viscous incompressible flow through a constant circular cross-section. It is also known as the Hagen-Poiseuille law, Poiseuille law and Poiseuille equation. The Hagen Poiseuille equation can be derived from the Navier-Stokes equations. The derivation of Poiseuille's Law is surprisingly simple, but it requires an understanding of viscosity. When two layers of liquid in contact with each other move at different speeds, there will

be a force between them. (S. P. Sutera, R. Skalak,1993). This force is proportional to the area of contact *A*, the velocity difference in the direction of flow $\Delta v_x/\Delta y$, and a proportionality constant η and is given by

$$F_{viscosity,top} = -\eta A \frac{\Delta \upsilon_x}{\Delta y} \tag{2.14}$$

The negative sign is in there because we are concerned with the faster moving liquid (top in figure).

2.10 THEORY OF COUETTE FLOW

In fluid dynamics, Couette flow refers to the laminar flow of a viscous fluid in the space between two parallel plates, one of which is moving relative to the other. The flow is driven by virtue of viscous drag force acting on the fluid and the applied pressure gradient parallel to the plates.

Couette flow is frequently used in undergraduate physics and engineering courses to illustrate shear-driven fluid motion. The simplest conceptual configuration finds two infinite, parallel plates separated by a distance *h*. One plate, say the top one, translates with a constant velocity u_0 in its own plane. Neglecting pressure gradients, the Navier-Stokes equations simplify to

$$\frac{d^2u}{dy^2} = 0 \tag{2.15}$$

Where *y* is a spatial coordinate normal to the plates and *u (y)* is the velocity distribution. This equation reflects the assumption that the flow is *uni-directional*. That is only one of the three velocity components (*u*, *v*, *w*) is non-trivial. If *y* originates at the lower plate, the boundary conditions are *u (0)* = 0 and *u (h)* = u_0. The exact solution found by integrating twice and solving for the constants using the boundary condition. (B.R. Munson, D.F. Young, and T.H. Okiishi, 2002)

$$u(y) = u_0 \frac{y}{h} \tag{2.16}$$

2.11 DEFINITION OF RAYLEIGH NUMBER, REYNOLDS NUMBER AND PRANDTL NUMBER

2.11.1 Rayleigh Number

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 describes the relationship between buoyancy and viscosity within a fluid and the Prandtl number which describes the relationship between momentum diffusivity and thermal diffusivity.

For free convection near a vertical wall, this number is

$$Ra_x = Gr_x \,\mathrm{Pr}\,\mathrm{Pr} = \frac{g\beta}{\nu\alpha}(T_s - T_\infty)x^3 \tag{2.17}$$

In the above, the fluid properties Pr, ν, α and β are evaluated at the film temperature, which is defined as

$$T_f = \frac{T_s + T_\infty}{2} \tag{2.18}$$

2.11.2 Reynolds 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.

For flow in a pipe or a sphere moving in a fluid the diameter is generally used today. The velocity may also be a matter of convention in some circumstances, notably stirred vessels.

$$\mathrm{Re} = \frac{\rho VD}{\mu} = \frac{VD}{\nu} = \frac{QD}{\nu A} \tag{2.19}$$

2.11.3 Prandtl Number

The Prandtl number is a dimensionless number approximating the ratio of momentum diffusivity (kinematics viscosity) and thermal diffusivity. It is defined as:

$$\mathrm{Pr} = \frac{\nu}{\alpha} = \frac{\mu C_p}{k} \tag{2.20}$$

Typical values for *Pr* are:

i. Around 0.7-0.8 for air and many other gases
ii. Around 0.16-0.7 for mixture of noble gases or noble gases with hydrogen
iii. Around 7 for water
iv. Between 100 and 40000 for engine oil
v. Between 4 and 5 for R-12 refrigerant
vi. Around 0.015 for mercury

Prandtl number is related to the thickness of the momentum and thermal boundary layer. When *Pr* is small, 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.

CHAPTER 3

METHODOLOGY

For this chapter, we will discuss about the simulation result of poiseulle flow, Couette flow and Porous Couette Flow. We also discuss about finite difference method to simulate natural convection in a square cavity.

3.1 ALGORITHM

The algorithm flowchart for LBM is shown in Figure 3.1. It consists of two processes; advection process and collision process. The initial values of density distribution f are specified at each grid point. Then, the system evolves in the following steps.

i. The advection term is solved by applying the streaming process of the density distribution function.
ii. Then the collision process is solved by BGK collision model.
iii. Next step is to define the boundary conditions based on the bounce back boundary conditions.
iv. The convergence criterion is based on steady state conditions and will discuss in Chapter 4.

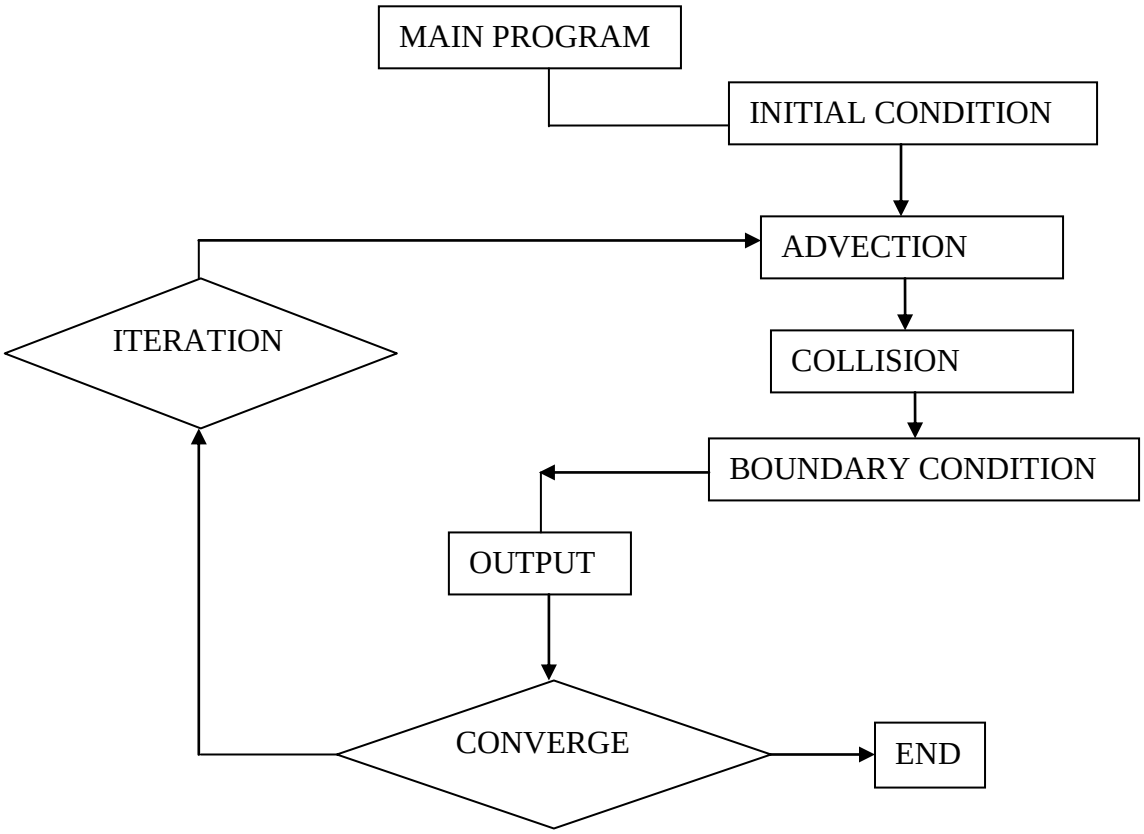
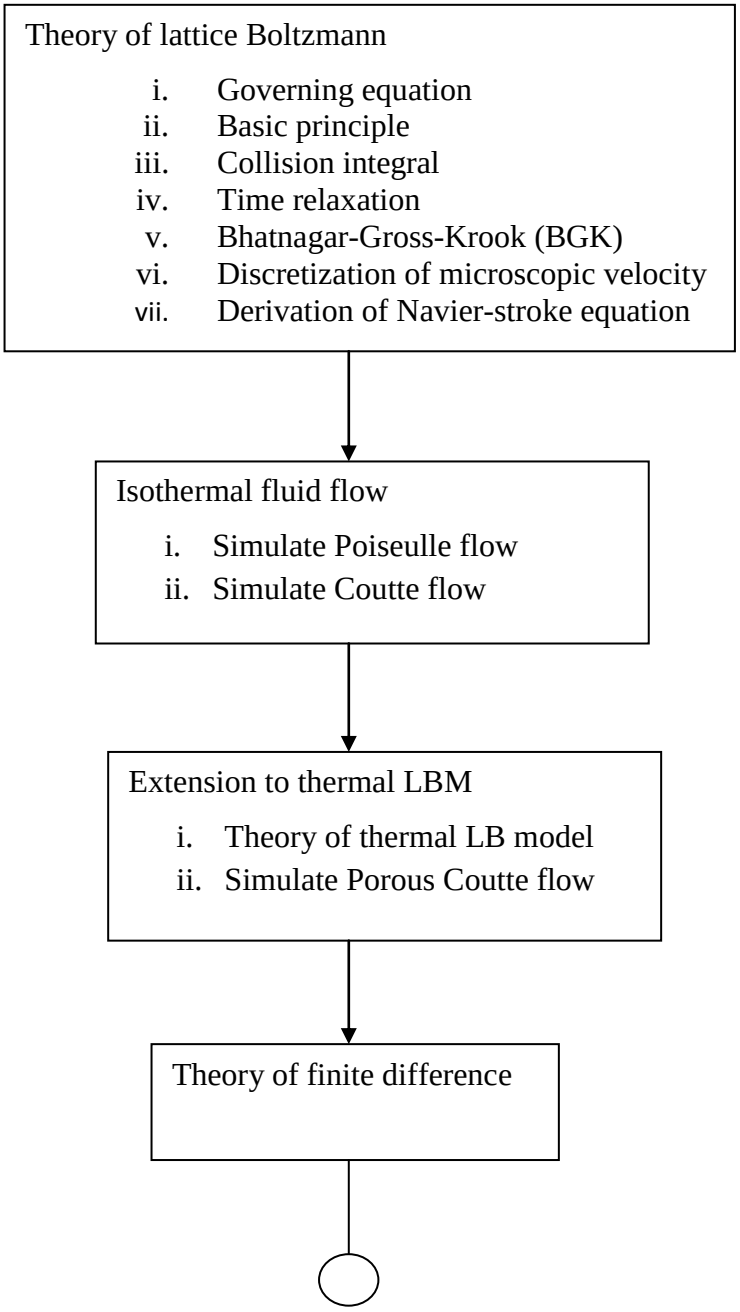


Figure 3.1: Original LBM algorithm flowchart

3.2 FLOW CHART FOR THE THESIS



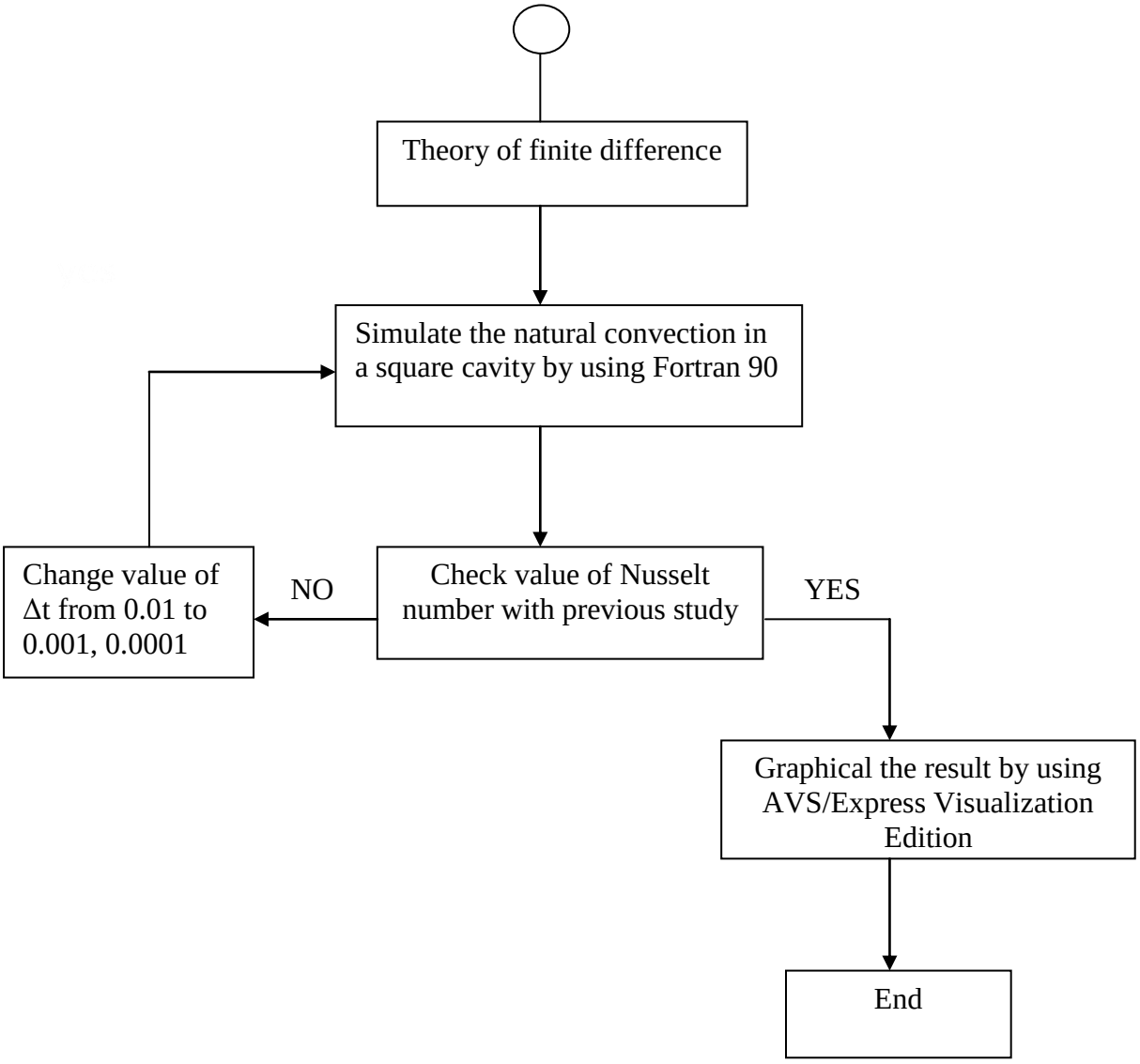


Figure 3.2: Flow chart for the relation in the thesis

3.3 SIMULATION RESULT OF 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. In this case the pressure gradient is set between the inlet and outlet end of the channel. This is done by setting the density (proportional to the pressure) at slightly different values between the two ends. The velocity is not set to any value. The well-known bounce-back boundary condition is applied at the top and bottom wall.

Letting the system evolved, it is observed that it reaches a steady state corresponding to the parabolic solution of the channel flow. The criterion of steady state is set by

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

Where M and N are mesh number in x and y direction respectively. It usually takes a few thousand iterations to reach a steady state depending on the value of the viscosity and the boundary conditions. Two type of measurement were taken in the simulation. One is the measurement of velocity, u .The other is the measurement of pressure along the channel. All the measurements were taken after the steady state is attained. (C.S. Nor Azwadi, 2007)

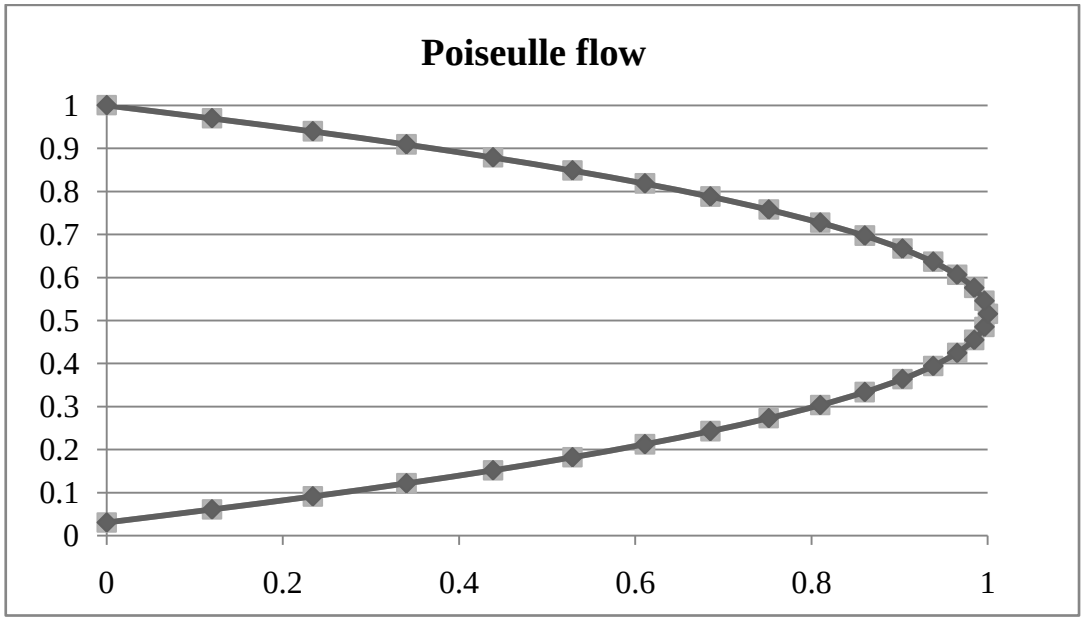


Figure 3.3: Poiseulle flow graph

The density change between the two ends along the centerline of the channel is a straight line as shown in Figure 3.3 .In the inset the velocity profile across the channel is displayed for the stationary state. The figure corresponds to a simulation using lattice size (4x33) means XD = 4, YD = 33 (corresponding to L = 30) and $\tau_f = 0.55$

The combination of results shows that, not only the velocity profile is correct (parabolic shape) but also the pressure distribution is linear along the channel length.

3.4 SIMULATION RESULT OF COUETTE FLOW

For this section, 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 expect 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. The channel is L= 40.

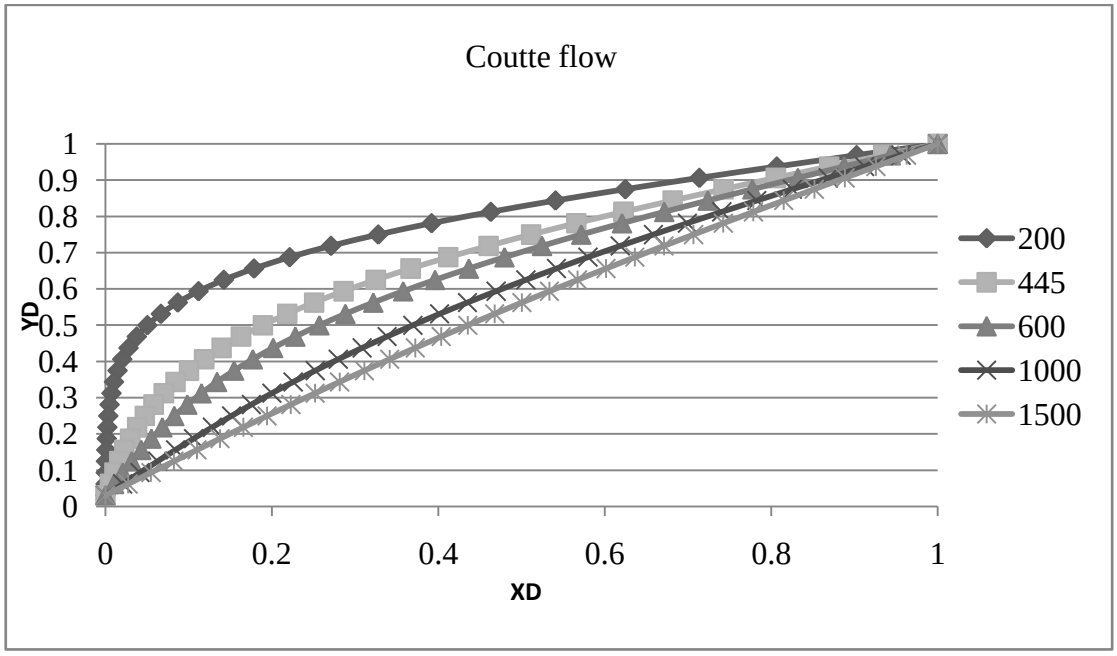


Figure 3.4: Couette flow graph

Figure 3.4 shows a sequence of normalized velocity profiles for different times. The lattice size for this experiment is (4x32) where are XD = 4, YD = 32 and the relaxation time is τ_f = 1.000. The velocity profiles are drawn at times t= 200, 445, 600, 1000, 1500 in LBM units. Periodic boundary conditions are implemented in the x-direction. The solution for the steady state case well known and corresponds to the velocity increasing linearly from zero at the bottom to U at the top plate.

3.5 SIMULATION RESULT OF POROUS COUETTE FLOW

For this section, we shall apply the newly developed model to simulate heat transfer in porous coquette flow problem.

Consider two infinite parallel plates separated by a distance of L. The upper cool plate at temperature T_C moves at speed U and the lower hot plate at temperature T_H is stationary. A constant normal flow of fluid is injected through the bottom hot plate and withdrawn at the same rate from the upper plate. The analytical solution of the velocity field in the steady state is given by

$$u = U(\frac{e^{\mathrm{Re}\frac{y}{L}} - 1}{e^{\mathrm{Re}} - 1}) \tag{3.2}$$

Where Re is the Reynolds number based on the inject velocity,v_o. The temperature profile in the steady state satisfies

$$T = T_c + \Delta T\left(\frac{e^{\mathrm{Pr}\,\mathrm{Pr}\,\mathrm{Re}\frac{y}{L}} - 1}{e^{\mathrm{Pr}\,\mathrm{Pr}\,\mathrm{Re}} - 1}\right) \tag{3.3}$$

Where $\Delta T=T_H-T_c$ is the temperature difference between the hot and cool walls. $Pr=v/\chi$ is the Prandtl number. Another dimensionless parameter id the Rayleigh number defined by

$$Ra = \frac{g\beta\Delta TL^3}{\upsilon\chi} \tag{3.4}$$

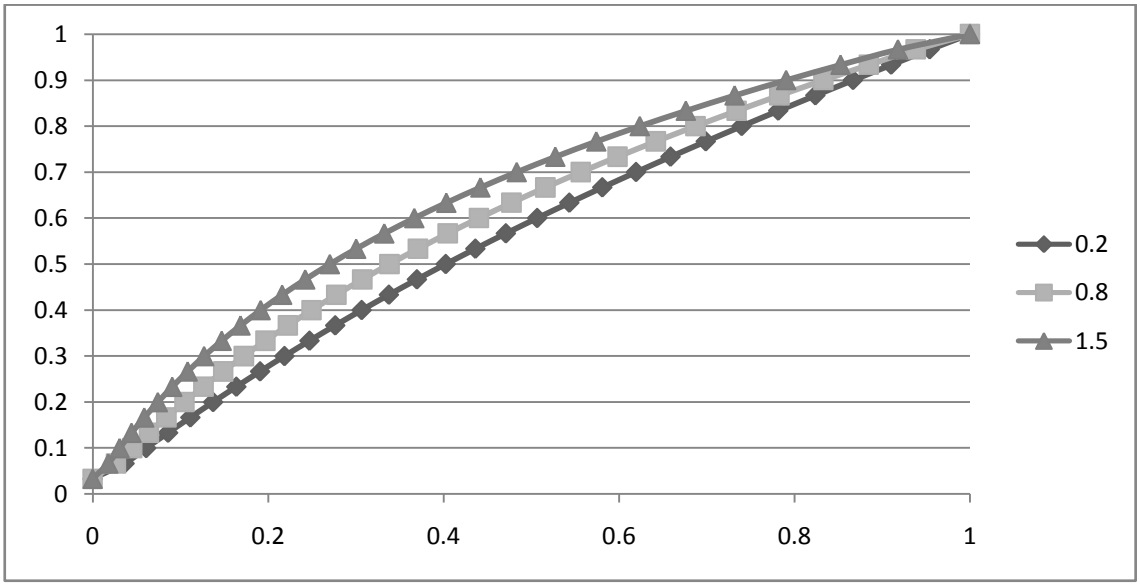


Figure 3.5: Temperature profile at Ra=100 and Re=10 and Pr=0.2, 0.8, and 1.5

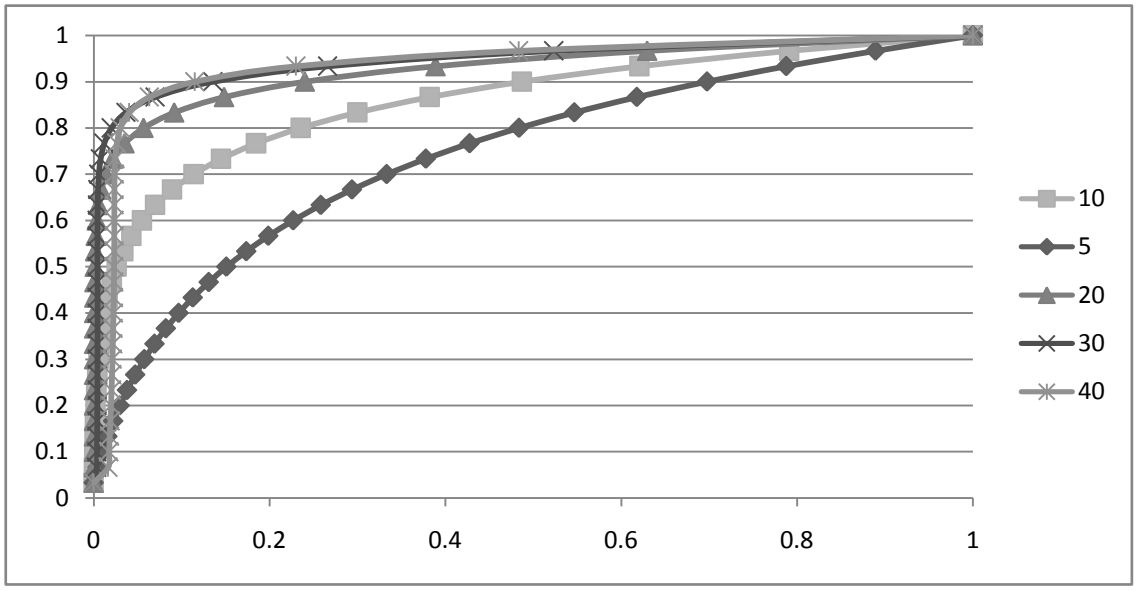


Figure 3.6: Temperature profile at Pr = 0.71 and Ra = 100 and Re=5, 10, 20, and 30

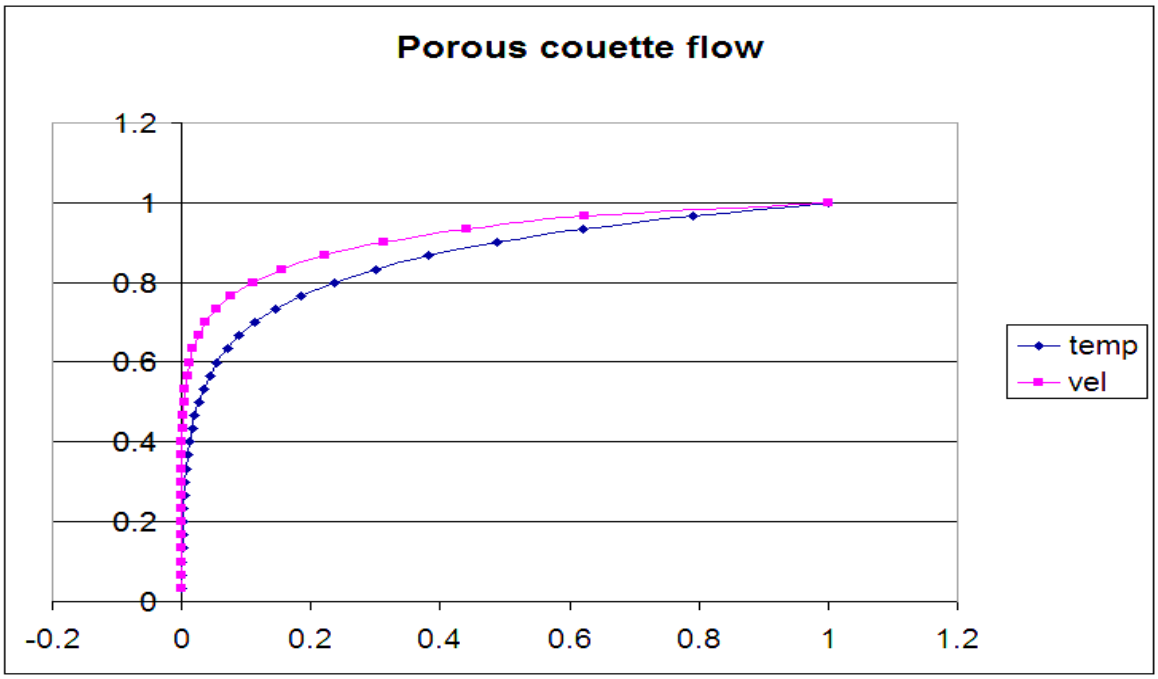


Figure 3.7: Velocity and temperature profile at Re=10, Pr = 0.71 and Ra = 60000

Periodic boundary conditions are used at the entrance and exit of the channel and the non-equilibrium bounce back boundary conditions for the velocity. The normalized temperature profile for Ra=100, Re=10 and Pr =0.2, 0.8, and 1.5 are shown in figure 3.5. For the figure 3.6 shows the result for the Pr=0.71, Ra= 100 and Re= 5, 10, 20 and 30. They agree well with the analytical solution. To show that this model is suitable and numerically stable for a wide range of Rayleigh number, the computations for Ra=10 till Ra=60000 at Pr=0.71 and Re=10 have been done. The results are shown in figure 3.7. The results presented above indicating that the numerical stability of the new model is higher than the previous proposed 4-velocity lattice model.

CHAPTER 4

RESULT AND DISCUSSION

4.1 FINITE DIFFERENCE LATTICE BOLTZMANN METHOD

One possible way to release the constraint of the lattice symmetry is to use the fnite difference scheme for the lattice Boltzmann equation. For this section, the discretization procedure of LBE is demonstrated with finite difference technique. The temporal discretization is obtained using second order Rungge-Kutta (modified) Euler method. The time evolution of particle distributions is then derived by

$$f_i^{n+\frac{1}{2}} = f_i^n + \frac{\Delta t}{2}\left(-c_i.\nabla f_i^n - \frac{f_i^n - f_i^{eq,n}}{\tau_f} \right) \tag{4.2}$$

$$f_i^{n+1} = f_i^n + \Delta t\left(-c_i.\nabla f_i^{n+\frac{1}{2}} - \frac{f_i^{n+\frac{1}{2}} - f_i^{eq,n+\frac{1}{2}}}{\tau_f} \right) \tag{4.3}$$

The second order Equations of 4.4~4.7 upwind schemes can be applied to calculate the spatial gradient in equation 4.1;

$$\frac{\partial f_i}{\partial t} + c_i.\nabla f_i(x,t) = -\frac{1}{\tau_f}(f_i - f_i^{eq}) \tag{4.1}$$

$$c_{ix}\partial_x f_i = c_{ix}\frac{3f_i(x,y) - 4f_i(x-\Delta x,y) + f_i(x-2\Delta x,y)}{2\Delta x}, c_{ix} > 0 \tag{4.4}$$

$$c_{ix}\partial_x f_i = -c_{ix}\frac{3f_i(x,y)-4f_i(x+\Delta x,y)+f_i(x+2\Delta x,y)}{2\Delta x}, c_{ix}<0 \quad (4.5)$$

$$c_{iy}\partial_y f_i = c_{iy}\frac{3f_i(x,y)-4f_i(x,y-\Delta y)+f_i(x,y-2\Delta y)}{2\Delta y}, c_{iy}>0 \quad (4.6)$$

$$c_{iy}\partial_y f_i = -c_{ix}\frac{3f_i(x,y)-4f_i(x,y+\Delta y)+f_i(x,y+2\Delta y)}{2\Delta y}, c_{iy}<0 \quad (4.7)$$

The combination of these specifics space and time discretization results in second order in space and second order in time. (C.S.Nor Azwadi)

4.2 FINITE DIFFERENCE THERMAL LATTICE BOLTZMANN METHOD

The evolution of internal energy density distribution is given by;

$$\frac{\partial g_i}{\partial t} + c_i.\nabla g_i = -\frac{1}{\tau_g}(g_i - g_i^{eq}) \quad (4.8)$$

The time space can also be discretised using second order Rungge-Kutta (modified) Euler method. The time evolution of particle distributions is then given by;

$$g_i^{n+\frac{1}{2}} = g_i^{n} + \frac{\Delta t}{2}\left(-c_i.\nabla_{g_i}{}^{n} - \frac{g_i^{n} - g_i^{eq,n}}{\tau_g}\right) \quad (4.9)$$

$$g_i^{n+1} = g_i^{n} + \Delta t\left(-c_i.\nabla_{g_i}{}^{n+\frac{1}{2}} - \frac{g_i^{n+\frac{1}{2}} - g_i^{eq,n+\frac{1}{2}}}{\tau_g}\right) \quad (4.10)$$

4.3 NATURAL CONVECTION IN A SQUARE CAVITY

The natural convection in a square cavity is brought to test the validity of finite difference thermal lattice Boltzmann model using the newly developed 4-velocity model for the internal energy density equilibrium density distribution function. The problem is a two dimensional square cavity with side wall is a difference temperature which are a hot wall on the left and the cold wall on the right side. Meanwhile, the top side and the bottom side walls being adiabatic. The temperature difference between walls show a temperature gradient in a fluid and also consequently density difference induces a fluids motion. This called convection.

In this simulation, the Boussinesq approximation is applied to the buoyancy force term. With this approximation, it assumed that all fluid properties β and υ can be considered as constant in the body force term except for the temperature dependence of the density in the gravity term.

$$\rho G = \rho\beta g(T - T_m)\, j \tag{4.11}$$

Where β is the thermal expansion coefficient, g is the acceleration due to gravity, T_m is the average temperature and j is the vertical direction opposite to that of gravity. The dynamical similarity depends on two dimensionless parameters; the prandtl number, Pr and the Rayleigh number, Ra defined by

$$\Pr = \frac{\upsilon}{\chi} \tag{4.12}$$

$$Ra = \frac{g\beta\Delta TL^3}{\upsilon x} \tag{4.13}$$

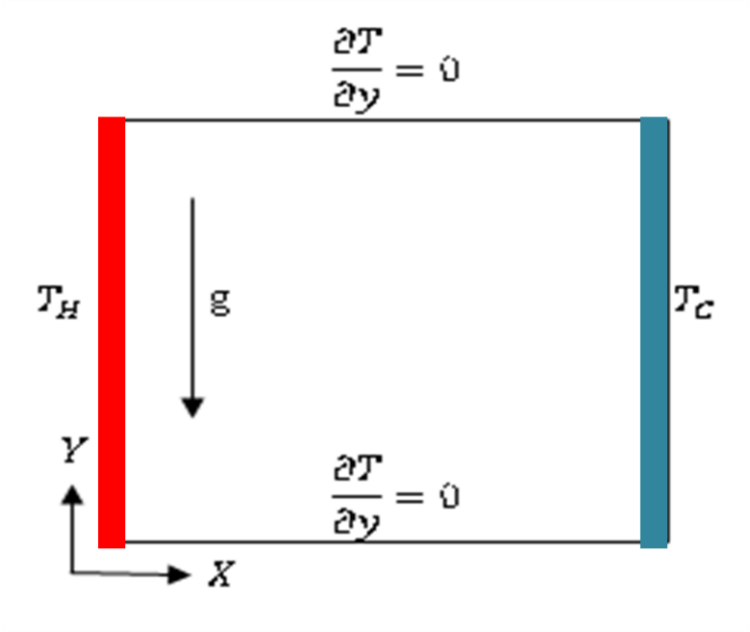


Figure 4.1: The schematic for natural convection in a square cavity.

Source: C.S. Nor Azwadi, 2007

The simulation were done for Rayleigh number from Ra=10^3 to Ra=10^5.The mesh size of 51x51, 101x101, 201x201 are used for Ra=10^3, Ra=10^4 and Ra=10^5 respectively. In this simulation, Prandlt number, Pr is set to be 0.71.The benchmark result of the natural convection in a square cavity was based on the solution of Navier Strokes equation by Davis (1983). Figure below show the streamlines and isotherms for the simulation at Rayleigh number, Ra=10^3 to Ra=10^5

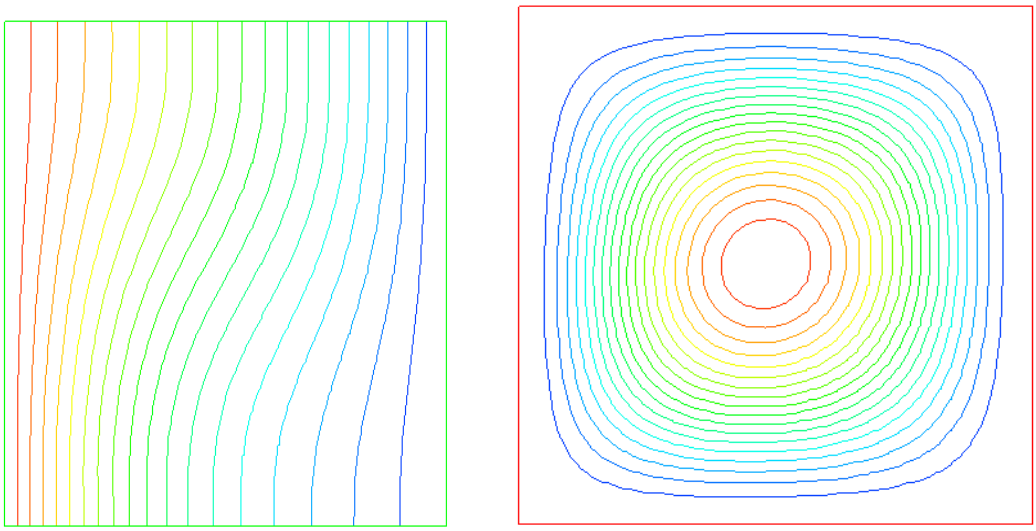


Figure 4.2: Streamline and isotherms for the simulation of natural convection in a square cavity for $Ra=10^3$

At the beginning of the simulation for $Ra= 10^3$, vortex appear at the center of the cavity with circular shape. The isotherms are almost vertically parallel to the wall indicating that conduction mode heat transfer mechanism is dominant. From figure 4.2, streamline and isotherms can see clearly.

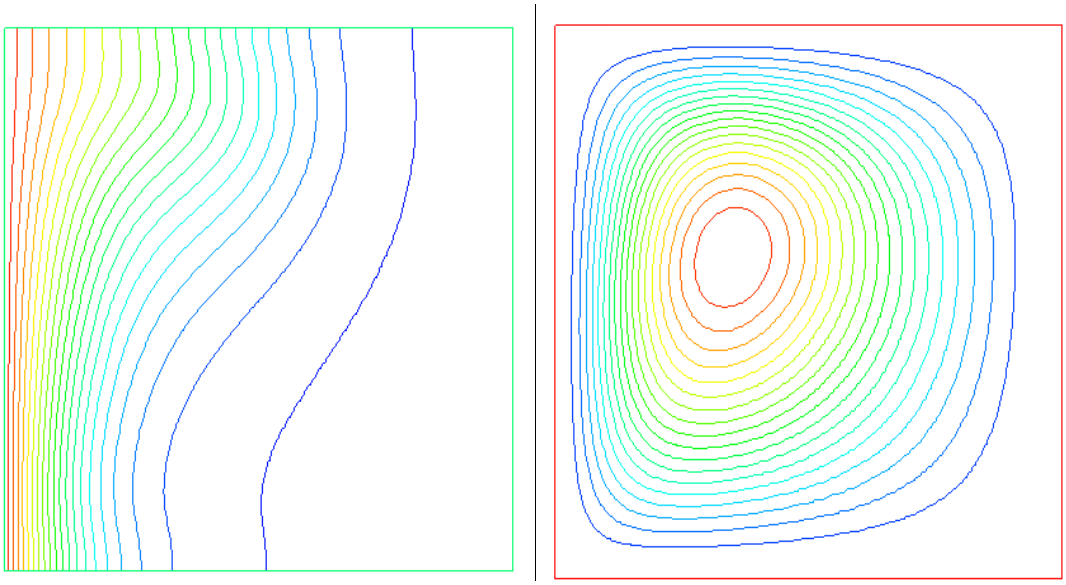


Figure 4.3: Streamline and isotherms for the simulation of natural convection in a square cavity for $Ra=10^4$

For Ra= 10^4 from figure 4.3, circular vortex at the center of cavity was distorted and its shape change to horizontal oval due to the connection effect. Meanwhile, isotherms start to be horizontally parallel to the wall at the cavity center. This effect due to the heat transfer mechanisms are mixed conduction and convection.

4.4 EXPECTED RESULT FOR THE SIMULATION OF NATURAL CONVECTION IN A SQUARE CAVITY AT RA=10^5

Figure 4.4 show that the streamline and isotherms for the simulation at Ra=10000 that have done by Azwadi. It is show that the increasing the Rayleigh number to Ra=10^5, a vortex oval shape appear at the left of the cavity when the system achieve equilibrium condition. All isotherms are almost horizontally parallel to the wall indicating that the convection is the main heat transfer mechanism.

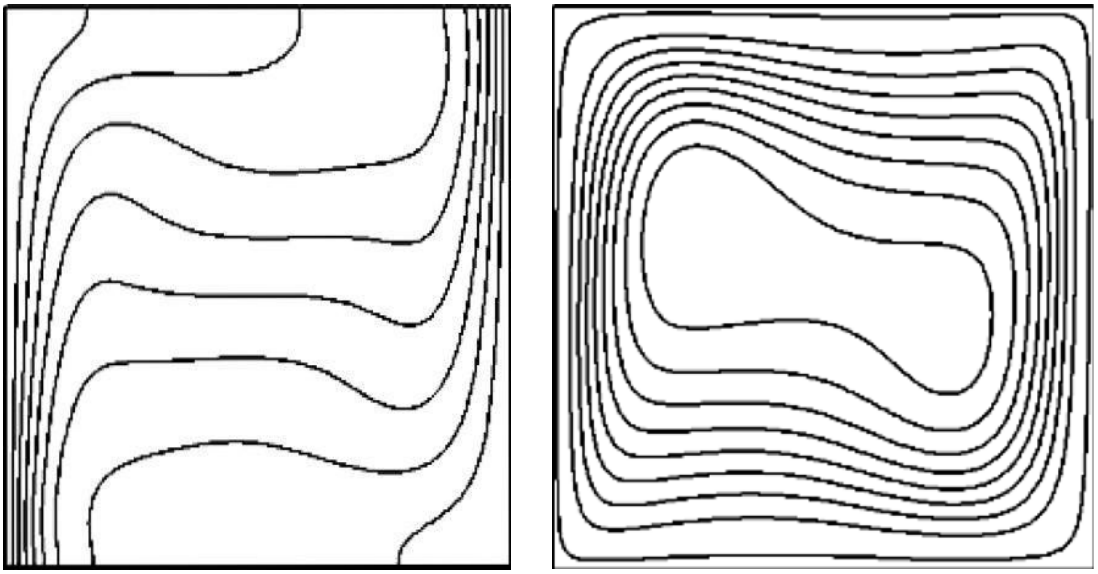


Figure 4.4: Streamline and isotherms for the simulation of natural convection in a square cavity for Ra=10^5

Table shows the comparison the average Nusselt number throughout the cavity by using finite difference scheme for $Ra=10^3$ to $Ra=10^6$

Table 4.1: Comparison among the present result with other LBM.

Rayleigh number, ra		10^3	10^4	10^5	10^6
Nu_{ave}	X.He et al (1998)	1.117	2.244	4.520	-
	Peng et al.(2003)	1.117	2.235	4.511	-
	Davis (1883)	1.116	2.234	4.510	8.798
	Azwadi (2007)	1.117	2.236	4.549	8.723
	Rosdzimin (2008)	1.116	2.201	4.249	-
	Present (2009)	1.116	1.737	-	-

From the table 4.1 shows the predicted results are compared with the results that obtained by original double distribution function thermal lattice Boltzmann scheme (X.He et al 1998), the simplified scheme (Peng et al. 2003), the solution by Navier-Strokes equation (Davis 1983), the simplified thermal (Azwadi 2007) and lastly by using CIP-LBM scheme by (Rosdzimin 2008).

For all the value of Rayleigh number have been considered in the present analysis where the average Nusselt number have been predicted ±0.5 error compare to the previous result and can be accepted for the real engineering applications.

The evolution of lattice Boltzmann equations have been discretised using second order upwind finite difference. From that equation, as expected that the boundary layer is thicker than the velocity boundary layer for different Rayleigh number simulations. The flow patterns that including the boundary layers and vortices can be seen clearly in Figure 4.2, Figure 4.3 and also in Figure 4.4. The result obtained demonstrate that by using double distribution function thermal lattice Boltzmann model is very efficient procedure to study flow and heat transfer in a differentially heated square enclosure.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 CONCLUSION

In chapter one, have been introduced about the Navier-stokes equation and introduced the lattice Boltzmann method (LBM). In literature review, the theory of lattice Boltzmann method (LBM) from the Boltzmann equation has been discussed. The theory of the classical Boltzmann equation is also discussed.

In methodology, the algorithm of the advection and collision process was explained detail in chapter three. Simulation result for the isothermal flow which are Poiseulle flow and Couette flow have been discuss. Beside, the simulation result for the thermal flow which is Porous Couette flow also has been performed well.

Chapter four concerned with the combination of the finite difference scheme with the lattice Boltzmann method. 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.

Objective for this thesis was achieved and already discussed in chapter four. Simulation of natural convection in a square cavity by using finite difference method is successfully done for the $Ra=10^3$, $Ra=10^4$. In this thesis, the advection term in both density and internal energy density equations has been discretised by using second order upwind scheme and applied in the simulation of natural convection in a square cavity.

By using finite difference scheme, the thickness of thermal boundary layer decrease as Rayleigh number increase.

For the simulation at Rayleigh number, $Ra=10^5$ occur some problem. Simulation at high Rayleigh number will take a longer time to simulate. In this case, $Ra=10^5$ need a longer time, more than one month to simulate. Because of lack of time, the simulation is not done yet. To increase the simulation time, we were forced to apply small value of Δt. This problem proposed to be solved in recommendation for future work.

5.2 RECOMMENDATIONS

Modification of finite difference lattice Boltzmann scheme can be done by solving the non advection terms using higher order Rungge-Kutta method in order to reduce the simulation time by increasing the accuracy of time. Using this advantage, finite difference lattice Boltzmann scheme can be extended for simulation of any fluid flows problem or heat transfer problem using the non uniform grid size.

REFERENCES

B.R. Munson, D.F. Young, and T.H. Okiishi (2002). *Fundamentals of Fluid Mechanics*, John Wiley and Sons. ISBN 0-471-44250-X

C. S. N. Azwadi and T. Tanahashi (2006). *Simplified Thermal Lattice Boltzmann in Incompressible Limit*. International Journal of Modern Physics B. Vol. 20, No. 17 pp 2437-2449

C. S. N. Azwadi and T. Tanahashi (2007). *Three-Dimensional Thermal Lattice Boltzmann Simulation of Natural Convection in a Cubic Cavity*. International Journal of Modern Physics B. Vol. 21, No. 1 pp 87-96

C. S. N. Azwadi and T. Tanahashi (2008). *Simplified Finite Difference Thermal Lattice Boltzmann Method*. International Journal of Modern Physics B. Vol. 22, No.22 pp 3865-3876

D. V. Davis (1983). *Natural Convection of air in a square cavity: A benchmark numerical solution*. International Journal Numerical Method Fluid, 3, pp249-26472

Gallivan, Martha A., Noble,David R, Georgiadis, John G, Buckius, Richard O (1997), *An Evaluation of the Bounce-Back Boundary Condition for Lattice Boltzmann Simulations*. International Journal of Numerical Methods in Fluids, vol. 25, Issue 3, pp.249-263

Dieter A. Wolf-Gladrow (2000). *Lattice Gas Cellular Automata and Lattice Boltzmann Models*, An Introduction. Springer

Dieter Wolf-Gladrow 1 (1995). *A Lattice Boltzmann Equation for Diffusion*. Journal of Statistical Physics, Vol. 79, Nos. 5/6, 1995

S. P. Sutera, R. Skalak (1993). *The history of Poiseuille's law*. Annual Review of Fluid Mechanics, Vol. 25, pp. 1-19

Succi, Sauro (2001), *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University

Xiaoyi He and Li-Shi Luo (1997). *A priori derivation of the lattice Boltzmann equation*. Physical Review E, volume 55, number 6

Xiaoyi He, Qisu Zou, Li-Shi Luo, and Micah Dembo (1997). *Analytic Solutions of Simple Flows and Analysis of Nonslip Boundary Conditions for the Lattice Boltzmann BGK Model*. Journal of Statistical Physics, Vol. 87, Nos. ½

Xiaoyi He, Shiyi Chen, and Gary D. Doolen (1998) *A Novel Thermal Model for the Lattice Boltzmann Method in Incompressible Limit*. Journal Of Computational Physics 146, 282–300. Article No. Cp986057

APPENDIX A1

GANNT CHART FOR 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														

APPENDIX A2

GANNT CHART FOR FINAL YEAR PROJECT 2

PROJECT ACTIVITIES	**W 2**	**W 3**	**W4**	**W5**	**W6**	**W7**	**W8**	**W9**	**W10**	**W 11**	**W12**	**W13**	**W14**	**W15**
Theory of finite difference method														
Simulate the natural convection in a square cavity for														
1.Ra=10^3														
2=Ra=10^4														
3=Ra=10^5														
Graphical the result by using AVS														
Presentation														
Submit Report														

APPENDIX B

FINITE DIFFERENCE SIMULATION

```
      !************************************************************!
            Finite Difference lattice Boltzmann

      !************************************************************!
program cavity
implicit real*8 (a-h,o-z)
parameter (ij = 330, kkk = 8)
common/var1/f(0:ij,0:ij,0:kkk),feq(0:ij,0:ij,0:kkk),g(0:ij,0:ij,0:k
kk),geq(0:ij,0:ij,0:kkk)
common /var2/ cx(0:kkk),cy(0:kkk),dx(0:kkk),dy(0:kkk)
common /var3/ u(ij,ij),v(ij,ij),temp(ij,ij)
common /var4/ rho(ij,ij)
common /var5/ tmp(0:ij,0:ij,0:kkk),ff(0:ij,0:ij,0:kkk)
common/var6/fn(0:ij,0:ij,0:kkk),fx(0:ij,0:ij,0:kkk),fy(0:ij,0:ij,0:
kkk),fxn(0:ij,0:ij,0:kkk),fyn(0:ij,0:ij,0:kkk)
common/var7/gn(0:ij,0:ij,0:kkk),gx(0:ij,0:ij,0:kkk),gy(0:ij,0:ij,0:
kkk),gxn(0:ij,0:ij,0:kkk),gyn(0:ij,0:ij,0:kkk)
common /con1/ delt,ra, pr, th ,tc,gra
common /con2/ lx,ly
common /con3/ ntin,nstep,totalg
common/con4/pi,tauf,rhoint,delx,dely,delx2,dely2,rtauf,xnyu,chi,tau
g,rtaug
integer lx,ly,ntin,nstep,i,j,k,l
integer ::unit, ierror
character (len=6)::filename

real vel(ij,ij)

lx = 51
ly = 51
ra = 1000.d0
pr = 0.71d0
th = 1.0d0
tc = 0.0d0
ntin = 10
delt = 0.001d0
rhoint = 1.0d0
uinit = 0.0d0

write(*,*)'lx',lx
write(*,*)'ly',ly
write(*,*)'ra',ra
write(*,*)'pr',pr
write(*,*)'th',th
write(*,*)'tc',tc
write(*,*)'ntin',ntin
write(*,*)'delt',delt
write(*,*)'rhoint',rhoint
write(*,*)'uinit',uinit
gra = (0.0557**2)/(lx-1)
xnyu = (gra*((lx-1)**3)*(th-tc)*pr/ra)**0.5
```

```
tauf = 3*xnyu
chi = xnyu/pr
taug = chi
rtauf = 1.0/tauf
rtaug = 1.0/taug

write (6,*) 'tauf = ',tauf
write (6,*) 'rtauf = ',rtauf
write (6,*) 'taug = ',taug
write (6,*) 'rtaug = ',rtaug
write (6,*) 'gra = ',gra
write (6,*) 'is everything ok?'
read (*,*) ok

pi = atan(1.0d0)*4.0d0
write (6,*) 'calculation start'

call initial

do nstep = 1, 5000000

call fin
call output
        call equilibrium

            if (mod(nstep,ntin) .eq. 0) then
            totalg=0.0
            do i = 1,lx
                  do j = 1,ly
                        do k=1,4
                        totalg=totalg+g(i,j,k)
                        end do
                  end do
            end do

                  write(*,*) nstep
                  write(*,10)totalgn-totalg10
format ('Convergence= ',F10.7)

            end if

            if (mod(nstep,ntin) .eq.1) then
            totalgn=0.0
            do i = 1,lx
                  do j = 1,ly
                        do k = 1,4
                        totalgn=totalgn +g(i,j,k)
                        end do
                  end do
            end do
            end if


!***** if converge*****::!
            if (abs(totalgn - totalg) .le. 1.0e-5 )  then
            go to 100
            end if
```

```
            end do

100   write(*,*)'end of iteration'

open(unit=30,file='uvel1.dat',status='replace',action='write',iosta
t=ierror)
write(30,*)' Thermal Diffusivity      ',chi
write(30,*)' Rayleigh Number          ',ra
write(30,*)' Prandtl Number           ',pr
write(30,*)' Hydro Relax. Time        ',tauf
write(30,*)' Termo Relax. Time        ',taug
write(30,*)' Solution Converge at     ',nstep
write(30,*)' Delta t                   ',delt
write(30,*)' Mesh Size                ', lx, ly


do j = 1,ly

write(30,*) u((lx+1)/2,j)*(ly-1)/chi

end do

close(30)

open(unit=31,file='vvel1.dat',status='replace',action='write',iosta
t=ierror)
do i = 1,lx

write(31,*) v(i,(ly+1)/2)*(ly-1)/chi

end do

close(31)

open(unit=32,file='variables.dat',status='replace',action='write',i
ostat=ierror)
write(32,*)'x-vel, y-vel, temp'
do j = 1,ly
do i = 1,lx

write(32,*) u(i,j)*(ly-1)/chi,v(i,j)*(ly-1)/chi,temp(i,j)

            end do
            end do

            close(32)



      stop
      end


=================================================================

            subroutine initial
```

```
======================================================================
implicit real*8 (a-h,o-z)
parameter (ij = 330, kkk = 8)
common/var1/f(0:ij,0:ij,0:kkk),feq(0:ij,0:ij,0:kkk),g(0:ij,0:ij,0:k
kk),geq(0:ij,0:ij,0:kkk)
common /var2/ cx(0:kkk),cy(0:kkk),dx(0:kkk),dy(0:kkk)
common /var3/ u(ij,ij),v(ij,ij),temp(ij,ij)
common /var4/ rho(ij,ij)
common /var5/ tmp(0:ij,0:ij,0:kkk),ff(0:ij,0:ij,0:kkk)
common/var6/fn(0:ij,0:ij,0:kkk),fx(0:ij,0:ij,0:kkk),fy(0:ij,0:ij,0:
kkk),fxn(0:ij,0:ij,0:kkk),fyn(0:ij,0:ij,0:kkk)
common/var7/gn(0:ij,0:ij,0:kkk),gx(0:ij,0:ij,0:kkk),gy(0:ij,0:ij,0:
kkk),gxn(0:ij,0:ij,0:kkk),gyn(0:ij,0:ij,0:kkk)
common /con1/ delt,ra, pr, th ,tc,gra
common /con2/ lx,ly
common /con3/ ntin,nstep,totalg
common/con4/pi,tauf,rhoint,delx,dely,delx2,dely2,rtauf,xnyu,chi,tau
g,rtaug
integer lx,ly,ntin,nstep,i,j,k,l
integer ::unit, ierror
character (len=6)::filename

real w(0:kkk)

delx = 1.0
dely = 1.0
dely2 = dely*dely
delx2 = delx*delx


!setup physical data!
cx(0) = 0.0d0
cy(0) = 0.0d0

do k = 1,8
w(k) = sqrt (2.0d0)
if(mod(k,2) .eq. 1) w(k) = 1.0d0
      cx(k) = w(k)*cos((k-1)*pi/4.0d0)
      cy(k) = w(k)*sin((k-1)*pi/4.0d0)
            end do

dx(1) = 1.0d0
dy(1) = 1.0d0
dx(2) = -1.0d0
dy(2) = 1.0d0
dx(3) = -1.0d0
dy(3) = -1.0d0
dx(4) = 1.0d0
dy(4) = -1.0d0

            do i = 1,lx
                  do j = 1,ly
                        rho(i,j) = rhoint

                        if (i.eq.1) then
                              u(i,j) = 0.0
                              v(i,j) = 0.0
```

```
                        temp(i,j) = th
                  else
                        u(i,j) = 0.0
                        v(i,j) = 0.0
                        temp(i,j) = tc
                  end if

            end do
      end do



      call equilibrium

      do i = 0,lx+1
            do j = 0,ly+1
                  do k = 0,8
                        f(i,j,k) = feq(i,j,k)
                        fx(i,j,k) =0.0
                        fy(i,j,k) =0.0
                  end do
            end do
      end do

      do i = 0,lx+1
            do j = 0,ly+1
                  do k = 1,4
                        g(i,j,k) = geq(i,j,k)
                        gx(i,j,k) =0.0
                        gy(i,j,k) =0.0
                  end do
            end do
      end do
      return
      end


=====================================================================
      subroutine equilibrium
=====================================================================
implicit real*8 (a-h,o-z)
parameter (ij = 330, kkk = 8)
common/var1/f(0:ij,0:ij,0:kkk),feq(0:ij,0:ij,0:kkk),g(0:ij,0:ij,0:k
kk),geq(0:ij,0:ij,0:kkk)
common /var2/ cx(0:kkk),cy(0:kkk),dx(0:kkk),dy(0:kkk)
common /var3/ u(ij,ij),v(ij,ij),temp(ij,ij)
common /var4/ rho(ij,ij)
common /var5/ tmp(0:ij,0:ij,0:kkk),ff(0:ij,0:ij,0:kkk)
common/var6/fn(0:ij,0:ij,0:kkk),fx(0:ij,0:ij,0:kkk),fy(0:ij,0:ij,0:
kkk),fxn(0:ij,0:ij,0:kkk),fyn(0:ij,0:ij,0:kkk)
common/var7/gn(0:ij,0:ij,0:kkk),gx(0:ij,0:ij,0:kkk),gy(0:ij,0:ij,0:
kkk),gxn(0:ij,0:ij,0:kkk),gyn(0:ij,0:ij,0:kkk)
common /con1/ delt,ra, pr, th ,tc,gra
common /con2/ lx,ly
common /con3/ ntin,nstep,totalg
common/con4/
pi,tauf,rhoint,delx,dely,delx2,dely2,rtauf,xnyu,chi,taug,rtaug
```

```
integer lx,ly,ntin,nstep,i,j,k,l
integer ::unit, ierror
character (len=6)::filename

real u2(0:ij,0:ij)


do i = 1,lx
      do j = 1,ly
            u2(i,j) = u(i,j)**2 + v(i,j)**2
            feq(i,j,0) = rho(i,j)*(1.0 - 3.0/2.0*u2(i,j))*4.0/9.0
      do l = 1,4
            k = l*2    ; dir = cx(k)*u(i,j) + cy(k)*v(i,j)
            feq(i,j,k)  =  rho(i,j)*(1.  +  3.*dir  +  9./2.*dir**2  -
3./2.*u2(i,j))/36.
            k = l*2 - 1; dir = cx(k)*u(i,j) + cy(k)*v(i,j)
            feq(i,j,k)  =  rho(i,j)*(1.  +  3.*dir  +  9./2.*dir**2  -
3./2.*u2(i,j))/9.
                              end do
                  end do
            end do

do i = 1,lx
      do j = 1,ly
      do k = 1,4
            tmpg = dx(k)*u(i,j) + dy(k)*v(i,j)
            geq(i,j,k) = rho(i,j)*temp(i,j)*(1 + tmpg )/4
                        end do
                  end do
            end do
            return
            end

=====================================================================
            subroutine fin
=====================================================================
implicit real*8 (a-h,o-z)
parameter (ij = 330, kkk = 8)
common/var1/
f(0:ij,0:ij,0:kkk),feq(0:ij,0:ij,0:kkk),g(0:ij,0:ij,0:kkk),geq(0:ij
,0:ij,0:kkk)
common /var2/ cx(0:kkk),cy(0:kkk),dx(0:kkk),dy(0:kkk)
common /var3/ u(ij,ij),v(ij,ij),temp(ij,ij)
common /var4/ rho(ij,ij)
common /var5/ tmp(0:ij,0:ij,0:kkk),ff(0:ij,0:ij,0:kkk)
common/var6/fn(0:ij,0:ij,0:kkk),fx(0:ij,0:ij,0:kkk),fy(0:ij,0:ij,0:
kkk),fxn(0:ij,0:ij,0:kkk),fyn(0:ij,0:ij,0:kkk)
common/var7/gn(0:ij,0:ij,0:kkk),gx(0:ij,0:ij,0:kkk),gy(0:ij,0:ij,0:
kkk),gxn(0:ij,0:ij,0:kkk),gyn(0:ij,0:ij,0:kkk)
common /con1/ delt,ra, pr, th ,tc,gra
common /con2/ lx,ly
common /con3/ ntin,nstep,totalg
common/con4/
pi,tauf,rhoint,delx,dely,delx2,dely2,rtauf,xnyu,chi,taug,rtaug
integer lx,ly,ntin,nstep,i,j,k,l
integer ::unit, ierror
character (len=6)::filename
```

```
totalf = 0.0
if (nstep .ne. 1) then
      do j = 1,ly
            do k = 0,8

            f(0,j,k) = 2.0*f(1,j,k) - f(2,j,k)
                  f(lx+1,j,k) = 2.0*f(lx,j,k) - f(lx-1,j,k)
                  end do
            end do

            do i = 1,lx
                  do k = 0,8
                  f(i,0,k) = 2.0*f(i,1,k) - f(i,2,k)
                  f(i,ly+1,k) = 2.0*f(i,ly,k) - f(i,ly-1,k)
                  end do
            end do

            do k = 0,8
                  f(0,0,k) = 2.0*f(1,1,k) - f(2,2,k)
                  f(0,ly+1,k) = 2.0*f(1,ly,k) - f(2,ly-1,k)
                  f(lx+1,0,k) = 2.0*f(lx,1,k) - f(lx-1,2,k)
                  f(lx+1,ly+1,k) = 2.0*f(lx,ly,k) - f(lx-1,ly-1,k)
            end do

            do j = 1,ly
                  do k = 0,8

                  fx(0,j,k) = 2.0*fx(1,j,k) - fx(2,j,k)
                  fx(lx+1,j,k) = 2.0*fx(lx,j,k) - fx(lx-1,j,k)
                  end do
            end do

            do i = 1,lx
                  do k = 0,8
                  fx(i,0,k) = 2.0*fx(i,1,k) - fx(i,2,k)
                  fx(i,ly+1,k) = 2.0*fx(i,ly,k) - fx(i,ly-1,k)
                  end do
            end do

            do k = 0,8
                  fx(0,0,k) = 2.0*fx(1,1,k) - fx(2,2,k)
                  fx(0,ly+1,k) = 2.0*fx(1,ly,k) - fx(2,ly-1,k)
                  fx(lx+1,0,k) = 2.0*fx(lx,1,k) - fx(lx-1,2,k)
                  fx(lx+1,ly+1,k = 2.0*fx(lx,ly,k) - fx(lx-1,ly-1,k)
            end do

            do j = 1,ly
                  do k = 0,8

                  fy(0,j,k) = 2.0*fy(1,j,k) - fy(2,j,k)
                  fy(lx+1,j,k) = 2.0*fy(lx,j,k) - fy(lx-1,j,k)
                  end do
            end do

            do i = 1,lx
                  do k = 0,8
```

```
            fy(i,0,k) = 2.0*fy(i,1,k) - fy(i,2,k)
            fy(i,ly+1,k) = 2.0*fy(i,ly,k) - fy(i,ly-1,k)
            end do
      end do

      do k = 0,8
            fy(0,0,k) = 2.0*fy(1,1,k) - fy(2,2,k)
            fy(0,ly+1,k) = 2.0*fy(1,ly,k) - fy(2,ly-1,k)
            fy(lx+1,0,k) = 2.0*fy(lx,1,k) - fy(lx-1,2,k)
            fy(lx+1,ly+1,k) =2.0*fy(lx,ly,k) - fy(lx-1,ly-1,k)
      end do

      do j = 1,ly
            do k = 0,8

                  feq(0,j,k) = 2.0*feq(1,j,k) - feq(2,j,k)
                  feq(lx+1,j,k)=2.0*feq(lx,j,k)- feq(lx-1,j,k)
            end do
      end do

      do i = 1,lx
            do k = 0,8
                  feq(i,0,k) = 2.0*feq(i,1,k) - feq(i,2,k)
                  feq(i,ly+1,k)=2.0*feq(i,ly,k) -feq(i,ly-1,k)
            end do
      end do

      do k = 0,8
            feq(0,0,k) = 2.0*feq(1,1,k) - feq(2,2,k)
            feq(0,ly+1,k) = 2.0*feq(1,ly,k) - feq(2,ly-1,k)
            feq(lx+1,0,k) = 2.0*feq(lx,1,k) - feq(lx-1,2,k)
            feq(lx+1,ly+1,k)=2.0*feq(lx,ly,k)-feq(lx-1,ly-1,k)
      end do
      end if

=====================================================================
      if (nstep .ne. 1) then
            do j = 1,ly
                  do k = 1,4

                        g(0,j,k) = 2.0*g(1,j,k) - g(2,j,k)
                        g(lx+1,j,k) = 2.0*g(lx,j,k) - g(lx-1,j,k)
                  end do
            end do

            do i = 1,lx
                  do k = 1,4
                        g(i,0,k) = 2.0*g(i,1,k) - g(i,2,k)
                        g(i,ly+1,k) = 2.0*g(i,ly,k) - g(i,ly-1,k)
                  end do
            end do

            do k = 1,4
                  g(0,0,k) = 2.0*g(1,1,k) - g(2,2,k)
                  g(0,ly+1,k) = 2.0*g(1,ly,k) - g(2,ly-1,k)
                  g(lx+1,0,k) = 2.0*g(lx,1,k) - g(lx-1,2,k)
                  g(lx+1,ly+1,k) = 2.0*g(lx,ly,k) - g(lx-1,ly-1,k)
```

```
end do

do j = 1,ly
      do k = 1,4

            gx(0,j,k) = 2.0*gx(1,j,k) - gx(2,j,k)
            gx(lx+1,j,k) = 2.0*gx(lx,j,k) - gx(lx-1,j,k)
      end do
end do

do i = 1,lx
      do k = 1,4
            gx(i,0,k) = 2.0*gx(i,1,k) - gx(i,2,k)
            gx(i,ly+1,k) = 2.0*gx(i,ly,k) - gx(i,ly-1,k)
      end do
end do

do k = 1,4
      gx(0,0,k) = 2.0*gx(1,1,k) - gx(2,2,k)
      gx(0,ly+1,k) = 2.0*gx(1,ly,k) - gx(2,ly-1,k)
      gx(lx+1,0,k) = 2.0*gx(lx,1,k) - gx(lx-1,2,k)
      gx(lx+1,ly+1,k)=2.0*gx(lx,ly,k) - gx(lx-1,ly-1,k)
end do

do j = 1,ly
      do k = 1,4

            gy(0,j,k) = 2.0*gy(1,j,k) - gy(2,j,k)
            gy(lx+1,j,k) = 2.0*gy(lx,j,k) - gy(lx-1,j,k)
      end do
end do

do i = 1,lx
      do k = 1,4
            gy(i,0,k) = 2.0*gy(i,1,k) - gy(i,2,k)
            gy(i,ly+1,k) = 2.0*gy(i,ly,k) - gy(i,ly-1,k)
      end do
end do

do k = 1,4
      gy(0,0,k) = 2.0*gy(1,1,k) - gy(2,2,k)
      gy(0,ly+1,k) = 2.0*gy(1,ly,k) - gy(2,ly-1,k)
      gy(lx+1,0,k) = 2.0*gy(lx,1,k) - gy(lx-1,2,k)
      gy(lx+1,ly+1,k) =2.0*gy(lx,ly,k) - gy(lx-1,ly-1,k)
end do

do j = 1,ly
      do k = 1,4

            geq(0,j,k) = 2.0*geq(1,j,k) - geq(2,j,k)
            geq(lx+1,j,k =2.0*geq(lx,j,k) -geq(lx-1,j,k)
      end do
end do

do i = 1,lx
      do k = 1,4
            geq(i,0,k) = 2.0*geq(i,1,k) - geq(i,2,k)
```

```
                        geq(i,ly+1,k) =2.0*geq(i,ly,k)-geq(i,ly-1,k)
                  end do
            end do

            do k = 1,4
                  geq(0,0,k) = 2.0*geq(1,1,k) - geq(2,2,k)
                  geq(0,ly+1,k) = 2.0*geq(1,ly,k) - geq(2,ly-1,k)
                  geq(lx+1,0,k) = 2.0*geq(lx,1,k) - geq(lx-1,2,k)
                  geq(lx+1,ly+1,k =2.0*geq(lx,ly,k)-geq(lx-1,ly-1,k)
            end do
            end if

do i = 1,lx
      do j = 1,ly
            do k = 0,8
            fn(i,j,k)=f(i,j,k)-delt*rtauf*(f(i,j,k)-feq(i,j,k))

            fxn(i,j,k)=fx(i,j,k)-
delt*rtauf*(fx(i,j,k)0.5*(feq(i+1,j,k)-feq(i-1,j,k)))


            fyn(i,j,k)=fy(i,j,k)-delt*rtauf*(fy(i,j,k)-
0.5*(feq(i,j+1,k)-feq(i,j-1,k)))

                        end do
                  end do
            end do

            do i = 1,lx
                  do j = 1,ly
                        do k = 0,8
                              f(i,j,k) = fn(i,j,k)
                              fx(i,j,k) = fxn(i,j,k)
                              fy(i,j,k) = fyn(i,j,k)
                        end do
                  end do
            end do

            do i = 1,lx
                  do j = 1,ly
                        do k = 1,4
                              gn(i,j,k)=g(i,j,k)
delt*rtaug*(g(i,j,k)-geq(i,j,k))
                              gxn(i,j,k)=gx(i,j,k)-
delt*rtaug*(gx(i,j,k)- 0.5*(geq(i+1,j,k)-geq(i-1,j,k)))
                              gyn(i,j,k)=gy(i,j,k)-
delt*rtaug*(gy(i,j,k)- 0.5*(geq(i,j+1,k)-geq(i,j-1,k)))
                        end do
                  end do
            end do

            do i = 1,lx
                  do j = 1,ly
                        do k = 1,4
                              g(i,j,k) = gn(i,j,k)
                              gx(i,j,k) = gxn(i,j,k)
                              gy(i,j,k) = gyn(i,j,k)
```

```
                  end do
            end do
      end do

      do i = 1,lx
            do j = 1,ly
      do k = 0,8
      xx = -cx(k)*delt
      yy = -cy(k)*delt

      zx = sign(1.0,cx(k))
      zy = sign(1.0,cy(k))

      iup = i-int(zx)
      jup = j-int(zy)

      a1 = ((fx(iup,j,k) + fx(i,j,k))*delx*zx - 2.0*(f(i,j,k)
- f(iup,j,k)))/(delx**3*zx)
      e1 = (3.0*(f(iup,j,k) - f(i,j,k)) + (fx(iup,j,k) +
2.*fx(i,j,k))*delx*zx)/(delx*delx)
      b1 = ((fy(i,jup,k) + fy(i,j,k))*dely*zy - 2.0*(f(i,j,k)
- f(i,jup,k)))/(dely**3*zy)
      f1 = (3.0*(f(i,jup,k) - f(i,j,k)) + (fy(i,jup,k) +
2.0*fy(i,j,k))*dely*zy)/dely**2

      d1 = ( - (f(i,j,k) - f(i,jup,k) - f(iup,j,k) +
f(iup,jup,k)) - (fy(iup,j,k) -
fy(i,j,k))*dely*zy)/(delx*dely**2*zx)
      c1 = ( - (f(i,j,k) - f(i,jup,k) - f(iup,j,k) +
f(iup,jup,k)) - (fx(i,jup,k) -
fx(i,j,k))*delx*zx)/(delx**2*dely*zy)
      g1 = ( - (fy(iup,j,k) - fy(i,j,k)) +
c1*delx*delx)/(delx*zx)

      fn(i,j,k) = ((a1*xx+c1*yy+e1)*xx + g1*yy + fx(i,j,k))*xx
+ ((b1*yy+d1*xx+f1)*yy + fy(i,j,k))*yy + f(i,j,k)
      fxn(i,j,k) = (3.0a1*xx + 2.0*(c1*yy+e1))*xx +
(d1*yy+g1)*yy+fx(i,j,k)
      fyn(i,j,k) = (3.0*b1*yy + 2.0*(d1*xx+f1))*yy +
(c1*xx+g1)*xx+fy(i,j,k)
                  end do
            end do
      end do

      do i = 1,lx
            do j = 1,ly
                  do k = 0,8
                        f(i,j,k) = fn(i,j,k)
                        fx(i,j,k) = fxn(i,j,k)
                        fy(i,j,k) = fyn(i,j,k)
                        if (f(i,j,k) <= 0 ) then
                        write (*,*) ' error'
                        end if
                        totalf = totalf + f(i,j,k)
                  end do
            end do
      end do
```

```
            do i = 1,lx
                  do j = 1,ly
                        do k = 1,4
                              xx = -dx(k)*delt
                              yy = -dy(k)*delt

            zx = sign(1.0,dx(k))
            zy = sign(1.0,dy(k))

            iup = i-int(zx)
            jup = j-int(zy)

            a1 = ((gx(iup,j,k) + gx(i,j,k))*delx*zx - 2.0*(g(i,j,k)
- g(iup,j,k)))/(delx**3*zx)
            e1  =  (3.0*(g(iup,j,k)  -  g(i,j,k))  +  (gx(iup,j,k)  +
2.*gx(i,j,k))*delx*zx)/(delx*delx)
            b1 = ((gy(i,jup,k) + gy(i,j,k))*dely*zy - 2.0*(g(i,j,k)
- g(i,jup,k)))/(dely**3*zy)
            f1  =  (3.0*(g(i,jup,k)  -  g(i,j,k))  +  (gy(i,jup,k)  +
2.0*gy(i,j,k))*dely*zy)/dely**2

            d1  =  (  -  (g(i,j,k)  -  g(i,jup,k)  -  g(iup,j,k)  +
g(iup,jup,k))              -              (gy(iup,j,k)              -
gy(i,j,k))*dely*zy)/(delx*dely**2*zx)
            c1  =  (  -  (g(i,j,k)  -  g(i,jup,k)  -  g(iup,j,k)  +
g(iup,jup,k))              -              (gx(i,jup,k)              -
gx(i,j,k))*delx*zx)/(delx**2*dely*zy)
            g1    =    (-    (gy(iup,j,k)    -    gy(i,j,k))    +
c1*delx*delx)/(delx*zx)

            gn(i,j,k) = ((a1*xx+c1*yy+e1)*xx + g1*yy + gx(i,j,k))*xx
+ ((b1*yy+d1*xx+f1)*yy + gy(i,j,k))*yy + g(i,j,k)
                  gxn(i,j,k)  =  (3.0*a1*xx  +  2.0*(c1*yy+e1))*xx  +
(d1*yy+g1)*yy+gx(i,j,k)
            gyn(i,j,k)   =   (3.0*b1*yy   +   2.0*(d1*xx+f1))*yy   +
(c1*xx+g1)*xx+gy(i,j,k)
                        end do
                  end do
            end do

            do i = 1,lx
                  do j = 1,ly
                        do k = 1,4
                              g(i,j,k) = gn(i,j,k)
                              gx(i,j,k) = gxn(i,j,k)
                              gy(i,j,k) = gyn(i,j,k)
                              if (g(i,j,k) <= -0.1 ) then
                              write (*,*) ' error',i,j,k,g(i,j,k)
                              end if
                        end do
                  end do
            end do



            if ( mod(nstep,ntin) == 0) then
```

```
            write (*,60) totalf
60          format ('totalf  = ',F8.2,//)
            write (*,70) totalg
70          format ('totalg  = ',F8.2,//)
            end if
            tempor = 0.0
            do i = 1,lx
                  do j = 1,ly
                  tempor = tempor + temp(i,j)
                  tempave = tempor/(lx*ly)
                  end do
            end do

            do i = 1,lx
                  do j = 1,ly
                        do k = 0,8
                        f(i,j,k) =  f(i,j,k)+3*delt*gra*(cy(k)-
v(i,j))*feq(i,j,k)*(temp(i,j)-tempave)
                        end do
                  end do
            end do


            return
            end

=====================================================================
            subroutine output
=====================================================================
implicit real*8 (a-h,o-z)
parameter (ij = 330, kkk = 8)
common/var1/f(0:ij,0:ij,0:kkk),feq(0:ij,0:ij,0:kkk),g(0:ij,0:ij,0:k
kk),geq(0:ij,0:ij,0:kkk)
common /var2/ cx(0:kkk),cy(0:kkk),dx(0:kkk),dy(0:kkk)
common /var3/ u(ij,ij),v(ij,ij),temp(ij,ij)
common /var4/ rho(ij,ij)
common /var5/ tmp(0:ij,0:ij,0:kkk),ff(0:ij,0:ij,0:kkk)
common/var6/fn(0:ij,0:ij,0:kkk),fx(0:ij,0:ij,0:kkk),fy(0:ij,0:ij,0:
kkk),fxn(0:ij,0:ij,0:kkk),fyn(0:ij,0:ij,0:kkk)
common/var7/gn(0:ij,0:ij,0:kkk),gx(0:ij,0:ij,0:kkk),gy(0:ij,0:ij,0:
kkk),gxn(0:ij,0:ij,0:kkk),gyn(0:ij,0:ij,0:kkk)
common /con1/ delt,ra, pr, th ,tc,gra
common /con2/ lx,ly
common /con3/ ntin,nstep,totalg
common/con4/pi,tauf,rhoint,delx,dely,delx2,dely2,rtauf,xnyu,chi,tau
g,rtaug
integer lx,ly,ntin,nstep,i,j,k,l
integer ::unit, ierror
character (len=6)::filename

            do i = 1,lx
                  do j = 1,ly
                        rho(i,j) = 0.0
                  end do
            end do

            do i = 2,lx-1
```

```
      do j = 2,ly-1
            do k = 0,8
                  rho(i,j) = rho(i,j) + f(i,j,k)
            end do
      end do
end do

do i = 2,lx-1
      rho(i,1) = rho(i,2)
      rho(i,ly) = rho(i,ly-1)
end do

do j = 2,ly-1
      rho(1,j) = rho(2,j)
      rho(lx,j) = rho(lx-1,j)
end do

rho(1,1) = rho(2,2)
rho(lx,1) = rho(lx-1,2)
rho(lx,ly) = rho(lx-1,ly-1)
rho(1,ly) = rho(2,ly-1)

do i= 2,lx-1
      do j = 2,ly-1
            u(i,j) = 0.d0
            v(i,j) = 0.d0
            temp(i,j) = 0.d0
      end do
end do


do  i = 2, lx-1
      do j = 2, ly-1
            do k = 0,8
                  u(i,j)=u(i,j)+ f(i,j,k)*cx(k)/rho(i,j)
                  v(i,j)=v(i,j)+ f(i,j,k)*cy(k)/rho(i,j)
            end do
            do k = 1,4
                  temp(i,j)=temp(i,j)+ g(i,j,k)/rho(i,j)
            end do
      end do
end do

do j = 1, ly
temp(1,j) = th
temp(lx,j) = tc
end do

do i = 1, lx
temp(i,ly) = temp(i,ly-1)
temp(i,1) = temp(i,2)
end do

return
end
```

APPENDIX C

MESH FOR 2D NATURAL CONVECTION

```
PARAMETER (NNN=230000,       chi = 0.104519187198994)
parameter (xd = 51, yd = 51)
IMPLICIT REAL*8(A-H,O-Z)
integer  node,ok1,ok2
DIMENSION U(NNN),V(NNN),temp(nnn)
DIMENSION Uu(101,101),tempp(101,101),vv(101,101)
open(unit=10,file='nd_number.dat',status='replace',action='write',i
ostat=ierror)
node = 0
do j = 1,yd
      do i = 1,xd
      node = node + 1
      write(10,*) i,j,node
            end do
      end do
      close(10)
open(unit=11,file='variables.dat',status='old',action='read',iostat
=ierror)
read(11,*)
do n = 1,node
read(11,*)u(n),v(n),temp(n)
end do
close(11)
open(unit=11,file='variables.dat',status='old',action='read',iostat
=ierror)
read(11,*)
do j = 1,yd
      do i = 1,xd
      read(11,*)uu(i,j),vv(i,j),tempp(i,j)
            end do
      end do
      close(11)
      do j = 1,yd
            do i = 1,xd
                  uu(i,j) =uu(i,j)*chi/(yd-1)
            end do
      end do
umax = abs(u(1))
do i = 2, node
            ux = abs(u(i))
      if(ux .gt. umax)  then
            umax = ux
write(*,*)'node= ',i,'umax = ',umax
      end if
      end do
      write(*,*)'next for v?'
      read (*,*) ok1
      vmax = abs(v(1))
      do i = 2, node
            uy = abs(v(i))
      if(uy .gt. vmax)  then
      vmax = uy
```

```
      write(*,*)'node= ',i,'vmax = ',vmax
      end if
      end do
      ! calculate nusselt number
      th = 1.0d0
      tc = 0.0d0
            bnu = 0.0
      do i = 1, xd
            do j = 1,yd
             if (i .eq. 1) then
bnu=bnu+(((xd-1)/(chi*(th-tc)*(xd)*(yd)))*((uu(i,j)*tempp(i,j))-
(chi*(tempp(i+1,j)-tempp(i,j)))))
            else if (i .eq. xd) then
bnu=bnu+(((xd-1)/(chi*(thtc)*(xd)*(yd)))*((uu(i,j)*tempp(i,j))-
(chi*(tempp(i,j)-tempp(i-1,j)))))
             else
bnu=bnu+(((xd-1)/(chi*(thtc)*(xd)*(yd)))*((uu(i,j)*tempp(i,j))-
(chi*(0.5*(tempp(i+1,j)-tempp(i-1,j))))))
             end if
             end do
      end do
      write(*,*) 'nusselt number = ',bnu
    STOP
     END
```