A. A. Mohamad

Lattice Boltzmann Method

Fundamentals and Engineering Applications with Computer Codes

Springer

Lattice Boltzmann Method

A. A. Mohamad

Lattice Boltzmann Method

Fundamentals and Engineering Applications with Computer Codes

Springer

Prof. A. A. Mohamad
Department of Mechanical and Manufacturing Engineering
Schulich School of Engineering
The University of Calgary
Calgary, AB
T2N 1N4
Canada
e-mail: mohamad@ucalgary.ca

Present Address
Prof. A. A. Mohamad
College of Engineering
Alfaisal University
Riyadh
KSA

ISBN 978-0-85729-454-8 e-ISBN 978-0-85729-455-5

DOI 10.1007/978-0-85729-455-5

Springer London Dordrecht Heidelberg New York

Cover design: eStudio Calamar S.L.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Simplicity is Embedded in Complexity

Preface

Computational methods have emerged as powerful techniques for investigating and exploring physical and chemical phenomena and for solving real engineering problems. The finite element method (FEM) was first applied to solve a structural problem in 1956 by Turner et al. In the late 1960s, the finite element became a powerful technique for solving partial differential equations, heat transfer, and fluid dynamics problems. Also, at the same time the finite difference method (FDM) was used to solve fluid dynamics problems. In 1980, the finite volume method (FVM) was developed in Imperial College mainly to solve fluid dynamics problems. Since then the FVM has been extensively used to solve transport phenomena problems. Indeed finite difference, finite element, and finite volume methods belong to the same family of weighted residual methods and the only difference between these methods is the nature of the base and weighting functions. In 1988, the Lattice Boltzmann method (LBM) was introduced by McNamara and Zanetti to overcome the drawbacks of the lattice gas cellular automata. Since then the LBM emerged as an alternative powerful method for solving fluid dynamics problems. In traditional computational fluid dynamics methods (CFD), Navier–Stokes equations (NS) solve mass, momentum and energy conservation equations on discrete nodes, elements, or volumes. In other words, the nonlinear partial differential equations convert into a set of nonlinear algebraic equations, which are solved iteratively. In LBM, the fluid is replaced by fractious particles. These particles stream along given directions (lattice links) and collide at the lattice sites. The LBM can be considered as an explicit method. The collision and streaming processes are local. Hence, it can be programmed naturally for parallel processing machines. Another beauty of the LBM is handling complex phenomena such as moving boundaries (multiphase, solidification, and melting problems), naturally, without a need for face tracing method as it is in the traditional CFD.

A few years ago, I started learning LBM methods after many years of experience in finite difference and finite volume methods. As an engineer with little background on kinetic theory of particles, I had difficulty with understanding some of the terminologies used in kinetic theory. However, the beauty of the simplicity

of the method attracted me. I can see the future of the method in dealing with multiphase and multicomponent flows. It is easy to incorporate thermodynamics with the LBM, while such incorporation is a difficult task with the conventional method of solving NS. The most time-consuming process in solving incompressible flows, using traditional CFD methods, is dealing with the pressure term. At each step, the Laplace equation must be solved to satisfy the continuity equation. This process is the most computer resource demanding, especially for unsteady-state problems. In the LBM, there is no need for such a process, where the LBM is an explicit method, by nature. However, there is no method without difficulties and drawbacks.

I think there is a need for a textbook on the subject for engineers and for people willing to use the power of the method with little background in mathematics and physics. The book is written for engineers and scientists willing to apply the LBM for simulating heat, mass, and momentum transfers. I tried to avoid complicated theory and mathematics. However, I worked the mathematics using first- and second- year calculus notation in order not to confuse the reader with fancy notations used in the literature. Also, the book starts with simple one-dimensional problems with step-by-step explanations, which totally clear the way for the reader to understand more complicated issues.

I do believe that learning by working with problems and applications can help with understanding the topic. Engineering students and some science students are not well prepared on the kinetic theory and statistical mechanics. The book tries to cover the fundamentals of kinetic theory and statistical methods before introducing the LBM, which is the backbone of the most molecular and micro analysis of transports. A step-by-step approach is used in the book to help the reader to follow the subject without juggling from reference to reference. However, a list of references and extra reading materials are suggested to help the reader with more in-depth materials. The book is an introduction to the topic of the LBM with emphasis on the applications and a few complete examples with computer codes are included in the text. The reader should be able to produce the results presented in the book by himself. The book can be used as a textbook for senior undergraduate or graduate one-semester courses with hands-on examples. I think, the way the materials are introduced in the book builds the reader's confidence in understanding the topic and coding without any guesses. In some cases, the finite difference method is introduced parallel to the LBM for two reasons. First to show the differences and similarities between the well-established finite difference method and the LBM. Second, to compare the results predicted by the two methods. The finite difference method is easy to understand because it is based on the Taylor series expansion, which most students are exposed to in calculus classes.

Complete computer codes are given with examples. The codes are written in FORTRAN, however, it can be easily translated to other computer languages. The codes are written for clearness and simplicity and not for efficiency. In the first chapter, an introduction to kinetic theory is given, intended to familiarize the reader with concepts of kinetic theory. Chapter 2 introduces the Lattice Boltzmann

equation with general discussion of the method. Chapter 3 deals with the diffusion equation for heat, mass, and momentum. Chapter 4 introduces the advection–diffusion equation without and with the source term. Chapter 5 discusses with examples isothermal fluid flow problems (without heat or mass transfer). Chapter 6 is complementary to Chap. 5, where nonisothermal fluid flow problems are discussed, without and with coupling, i.e., forced and free convection. Chapter 7, introduces multi-relaxation schemes, Chap. 8, gives an overview of dealing with complex flow conditions and references suggested for each topic, such as flow in porous media and reactive flow, combustion, phase change and multiphases, etc.

The author highly appreciates any comments from the readers, via e-mail: mohamad@ucalgary.ca.

Acknowledgments

No matter where we stand now, there were people helped us to get there. There are many unseen hands behind the stage helping us morally or physically to perform on the stage. Writing a book is not an exceptional job. I learned a lot from people I worked with and students I taught. However, without good and relaxed environments nothing can be done as it is supposed to be, especially spending hours and hours in front of books, papers, and the computer without distraction. The smiles of people around you give more energy to the work. Just to mention a few. Working and communication with Dr. Succi were an excellent experience. His insight on the theory of the LBM cannot be underestimated by any means. Some of the visiting professors read the first draft of the book, which was published locally and noticed a few typos. Especial thanks go to Prof. Satti from Um-Alkura University, Saudi Arabia and Prof. Jose Rabi from Brazil. Also, many discussions with the former student of mine Dr. Kuzmin was very fruitful. Dr. Kuzmin developed a passion on the theory of the LBM.

Contents

Chapter 1
Introduction and Kinetics of Particles

1.1 Introduction

There are two main approaches in simulating the transport equations (heat, mass, and momentum), continuum and discrete. In continuum approach, ordinary or partial differential equations can be achieved by applying conservation of energy, mass, and momentum for an infinitesimal control volume. Since it is difficult to solve the governing differential equations for many reasons (nonlinearity, complex boundary conditions, complex geometry, etc.), therefore finite difference, finite volume, finite element, etc., schemes are used to convert the differential equations with a given boundary and initial conditions into a system of algebraic equations. The algebraic equations can be solved iteratively until convergence is insured. Let us discuss the procedure in more detail, first the governing equations are identified (mainly partial differential equation). The next step is to discretize the domain into volume, girds, or elements depending on the method of solution. We can look at this step as each volume or node or element contains a collection of particles (huge number, order of 10^{16}). The scale is macroscopic. The velocity, pressure, temperature of all those particles represented by a nodal value, or averaged over a finite volume, or simply assumed linearly or bi-linearly varied from one node to another. The phenomenological properties such as viscosity, thermal conductivity, heat capacity, etc. are in general known parameters (input parameters, except for inverse problems). For inverse problems, one or more thermophysical properties may be unknown.

On the other extreme, the medium can be considered made of small particles (atom, molecule) and these particles collide with each other. This scale is microscale. Hence, we need to identify the inter-particle (inter-molecular) forces and solve ordinary differential equation of Newton's second law (momentum conservation). At each time step, we need to identify location and velocity of each particle, i.e, trajectory of the particles. At this level, there is no definition of temperature, pressure, and thermo-physical properties, such as viscosity, thermal conductivity, heat capacity, etc. For instance, temperature and pressure are related to the kinetic energy of the particles (mass and velocity) and frequency of particles

A. A. Mohamad, *Lattice Boltzmann Method*,
DOI: 10.1007/978-0-85729-455-5_1, © Springer-Verlag London Limited 2011

bombardment on the boundaries, respectively. This method is called molecular dynamics (MD) simulations. To get an idea, a number of equations need to be solved; 10^3 cm of air at the room conditions contains about 3×10^{22} molecules. One mole of water (16 g, a small cup of water) contains more than 6.0×10^{23} molecules. To visualize this number, if we assume that the diameter of these molecules is 1 mm (tip of a pen, a dot) and these dots are arranged side by side, then the total area that can be covered by these dots is 6.0×10^{11} km^2. The total surface area of the earth is about 5.1×10^9 km^2 and area of the United States is 9.63×10^6 km^2 and area of Africa is 1.22×10^6 km^2. This means that we can cover the total area of the earth by dots made from number of molecules of 16 g of water if the molecule diameter is 1 mm. Furthermore, it is interesting to calculate how many years we need to complete the project of doting the world by using pen tip. For a reasonable fast person, 6 dots per second is a good estimate. Hence, it needs at least about 3×10^{15} years to complete the project.

The question is do we really need to know the behavior of each molecule or atom?

In bookkeeping process we need to identify location (x, y, z) and velocity (c_x, c_y, and c_z are velocity components in x, y and z direction, respectively) of each particle. Also, the simulation time step should be less than the particles collision time, which is in the order of fero-seconds (10^{-12} s). Hence, it is impossible to solve large size problems (order of cm) by MD method. At this scale, there is no definition of viscosity, thermal conductivity, temperature, pressure, and other phenomenological properties. Statistical mechanics need to be used as a translator between the molecular world and the macroscopic world. The question is, is the velocity and location of each particle important for us? For instance, in this room there are billions of molecules traveling at high speed order of 400 m/s, like rockets, hitting us. But, we do not feel them, because their mass (momentum) is so small. The resultant effect of such a "chaotic" motion is almost nil, where the air in the room is almost stagnant (i.e., velocity in the room is almost zero). Hence, the behavior of the individual particles is not an important issue on the macroscopic scale, the important thing is the resultant effects.

Fundamentally, MD is simple and can handle phase change and complex geometries without any difficulties and without introducing extra ingredients. However, it is important to specify the appropriate inter-particle force function. The main drawback or obstacle of using MD in simulation of a relatively large system is the computer resource and data reduction process, which will not be available for us in the seen future.

What about a middle man, sitting at the middle of both mentioned techniques, the lattice Boltzmann method (LBM). The main idea of Boltzmann is to bridge the gap between micro-scale and macro-scale by not considering each particle behavior alone but behavior of a collection of particles as a unit, Fig. 1.1. The property of the collection of particles is represented by a distribution function. The keyword is the distribution function. The distribution function acts as a representative for collection of particles. This scale is called meso-scale.

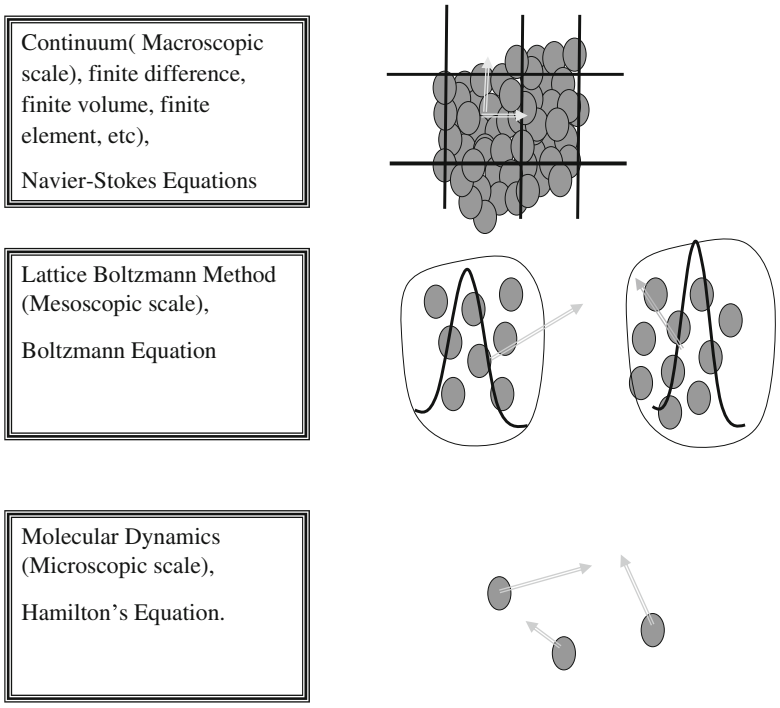

Fig. 1.1 Techniques of simulations

The mentioned methods are illustrated in Fig. 1.1.

LBM enjoys advantages of both macroscopic and microscopic approaches, with manageable computer resources.

LBM has many advantages. It is easy to apply for complex domains, easy to treat multi-phase and multi-component flows without a need to trace the interfaces between different phases. Furthermore, it can be naturally adapted to parallel processes computing. Moreover, there is no need to solve Laplace equation at each time step to satisfy continuity equation of incompressible, unsteady flows, as it is in solving Navier–Stokes (NS) equation. However, it needs more computer memory compared with NS solver, which is not a big constraint. Also, it can handle a problem in micro- and macro-scales with reliable accuracy.

1.2 Kinetic Theory

It is necessary to be familiar with the concepts and terminology of kinetic theory before proceeding to LBM. The following sections are intended to introduce the reader to the basics and fundamentals of kinetic theory of particles. I tried to avoid the detail of mathematics; however more emphasis is given to the physics.

Note that, the word particle and molecule are used interchangeably in the following paragraphs.

1.2.1 Particle Dynamics

As far as we know, the main building block of all the materials in nature is the molecules and sub-molecules. These molecules can be visualized as solid spheres moving "randomly" in conservatory manner in a free space. The motion satisfies conservation of the mass, momentum, and energy. Hence, Newton's second law (momentum conservation) can be applied, which states that the rate of change of linear momentum is equal to the net applied force.

$$\mathbf{F} = \frac{\mathrm{d}(\mathrm{m}\mathbf{c})}{\mathrm{d}t} \tag{1.1}$$

where **F** stands for the inter-molecular and external forces, m is the mass of the particle, **c** is the velocity vector of the particle and t is the time. For a constant mass, the equation can be simplified as,

$$\mathbf{F} = m\frac{\mathrm{d}\mathbf{c}}{\mathrm{d}t} = m\mathbf{a} \tag{1.2}$$

where **a** is the acceleration vector. The position of the particle can be determined from definition of velocity,

$$\mathbf{c} = \frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} \tag{1.3}$$

where **r** is the position vector of the particle relative to the origin, as shown in Fig. 1.2

In the MD simulation, the above equations are solved provided that **F** is a known function.

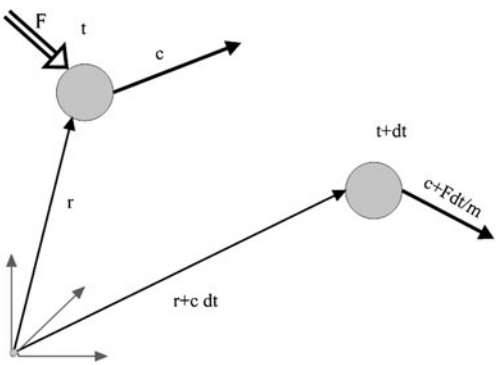

Fig. 1.2 Position and velocity vectors

If an external force, $\mathbf{F}$, is applied to a particle of mass m, the velocity of the particle will change from $\mathbf{c}$ to $\mathbf{c} + \mathbf{F}dt/m$ and its position changes from $\mathbf{r}$ to $\mathbf{r} + \mathbf{c}dt$, see Fig. 1.2. In the absence of an external force, the particle streams (moves) freely from one location to another location without changing its direction and speed, assuming no collision takes place.

The magnitude of the particle velocity increases and interaction between the particles increases as the internal energy of the system increases (for example, heating the system). Increases in the kinetic energy of the molecules are referred as increases in temperature in the macroscopic world. The particles (molecules) are continuously bombarding the container walls. The force exerted by those actions per unit area is referred as pressure in the macroscopic measure. From this simple model, we can see that there is a relationship between temperature and pressure, as the temperature increases, which means the kinetic energy of the molecules increase, we expect that the probability of particles bombarding the container wall, increases.

In the following section, the relationship between pressure, temperature, and kinetic energy will be explored.

1.2.2 Pressure and Temperature

Let us assume that a single particle moving with a speed, c_x (in x-direction), inside a tube of length L and bombarding the ends of the tube, continuously. The force exerted by the particle on an end is equal to the rate of change of the momentum (assuming that the collision is perfect elastic), then

$$F\Delta t = mc_x - (-mc_x) = 2mc_x, \tag{1.4}$$

where Δt is time between hits. Equation 1.4 is an integration of Newton's second law, Eq. 1.2. The time between hits is equal to $2L/c_x$, which is the time needed for the particle to travel from one end to another end and return to the same location, Fig. 1.3. Hence, $2LF/c_x = 2mc_x$, which yields,

$$F = mc_x^2/L \tag{1.5}$$

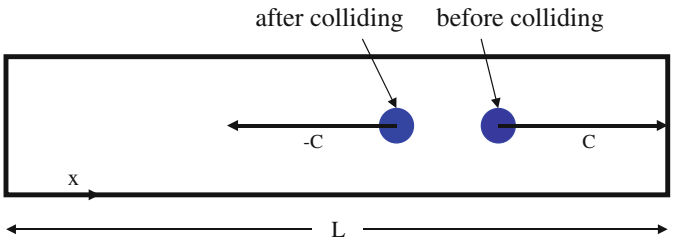

Fig. 1.3 A particle is freely moving in a box

The results can be generalized for N particles. The total force exerted by N particles is proportional to Nmc^2/L. In general, $c^2 = c_x^2 + c_y^2 + c_z^2$, where c_x, c_y and c_z are velocity component in x, y and z-directions, respectively. It is fair to assume that these components are equal, therefore $c^2 = 3c_x^2$. Then, the total force can be written as,

$$F = Nmc^2/(3L) \tag{1.6}$$

The pressure is defined as a force per unit area, perpendicular to the force vector, $P = F/A$. Then, the pressure exerted by N particle on the ends of the tube is equal to,

$$P = Nmc^2/(3LA) = Nmc^2/(3V) \tag{1.7}$$

where V stands for volume, which is equal to LA.

This simple picture of molecular motion, relates the pressure in macroscopic sense to the kinetic energy of the molecules, i.e.,

$$P = (mc^2/2)(2/3)N/V \tag{1.8}$$

In other words, the pressure is related to kinetic energy (KE) as,

$$P = \frac{2}{3}n\text{KE} \tag{1.9}$$

where n is the number of molecules per unit volume.

In this simple model (ideal gas model), we neglected the effect of molecular interaction and effect of the molecular size. However, for a gas at a room temperature, the results are surprisingly true. In a real system, the particle has volume and collision take places between the particles.

Experimentally, it is well-established that for gases far from the critical conditions, the state equation can be expressed as,

$$PV = nRT \tag{1.10}$$

where n is number of moles $= N/N_\text{A}$, where N_A is Avogadro's number and R is the gas constant.

Equating Eqs. 1.8 and 1.10, yields

$$N/N_\text{A}RT = (mc^2/2)(2/3)N \tag{1.11}$$

Introducing Boltzmann constant ($k = R/N_\text{A} = 1.38 \times 10^{-23}$ J/K), we can deduce that kinetic energy, KE, of a gas is

$$\text{KE} = mc^2/2 = (3/2)kT \tag{1.12}$$

It is interesting that the temperature and pressure in the macroscopic world are no more than scale of the kinetic energy of the molecules in the microscopic world.

1.3 Distribution Function

In 1859, Maxwell (1831–1879) recognized that dealing with a huge number of molecules is difficult to formulate, even though the governing equation (Newton's second law) is known. As mentioned before, tracing the trajectory of each molecule is out of hand for a macroscopic system. Then, the idea of averaging came into picture. For illustration purposes, in a class of 500 students (extremely small number, compared with the number of molecules in volume of $1\,\text{mm}^3$), if all the students started asking a question simultaneously, the result is noise and chaos. However, the question can be addressed through a class representative, which can be handled easily and the result may be acceptable by the majority.

The idea of Maxwell is that the knowledge of velocity and position of each molecule at every instant of time is not important. The distribution function is the important parameter to characterize the effect of the molecules; what percentage of the molecules in a certain location of a container have velocities within a certain range, at a given instant of time. The molecules of a gas have a wide range of velocities colliding with each others, the fast molecules transfer momentum to the slow molecule. The result of the collision is that the momentum is conserved. For a gas in thermal equilibrium, the distribution function is not a function of time, where the gas is distributed uniformly in the container; the only unknown is the velocity distribution function.

For a gas of N particles, the number of particles having velocities in the x-direction between c_x and $c_x + \mathrm{d}c_x$ is $Nf(c_x)\mathrm{d}c_x$. The function $f(c_x)$ is the fraction of the particles having velocities in the interval c_x and $c_x + \mathrm{d}c_x$, in the x-direction. Similarly, for other directions, the probability distribution function can be defined as before. Then, the probability for the velocity to lie down between c_x and $c_x + \mathrm{d}c_x$, c_y and $c_y + \mathrm{d}c_y$, and c_z and $c_z + \mathrm{d}c_z$ will be $Nf(c_x)f(c_y)f(c_z)\mathrm{d}c_x\mathrm{d}c_y\mathrm{d}c_z$.

It is important to mention that if the above equation is integrated (summed) over all possible values of the velocities, yields the total number of particles to be N, i.e.,

$$\iiint f(c_x)f(c_y)f(c_z)\,\mathrm{d}c_x\,\mathrm{d}c_y\,\mathrm{d}c_z = 1. \tag{1.13}$$

Since any direction can be x, or y or z, the distribution function should not depend on the direction, but only on the speed of the particles. Therefore,

$$f(c_x)f(c_y)f(c_z) = \Phi(c_x^2 + c_y^2 + c_z^2) \tag{1.14}$$

where Φ is another unknown function, that need to be determined. The value of distribution function should be positive (between zero and unity). Hence, in Eq. 1.14, velocity is squared to avoid negative magnitude. The possible function that has property of Eq. 1.14 is logarithmic or exponential function, i.e.,

$$\log A + \log B = \log(AB) \tag{1.15}$$

or

$$e^A e^B = e^{(A+B)} \tag{1.16}$$

It can be shown that the appropriate form for the distribution function should be as,

$$f(c_x) = Ae^{-Bc_x^2} \tag{1.17}$$

where A and B are constants. The exponential function implies that the multiplication of the functions can be added if each function is equal to the exponent of a function.

For example:

$$F(x) = e^{Bx}, \quad F(y) = e^{Cy},$$

then

$$F(x)F(y) = e^{Bx}e^{Cy} = e^{(Bx+Cy)}.$$

But if $F(x) = Bx$ and $F(y) = Cy$, then $F(x)F(y) = BxCy$, in this case the multiplication of function is not equal to the addition of the functions. Accordingly, it can be assumed that,

$$f(c) = Ae^{-Bc_x^2}Ae^{-Bc_y^2}Ae^{-Bc_z^2} = A^3e^{-Bc^2} \tag{1.18}$$

Multiplying together the probability distributions for the three directions, gives the distribution in terms of the particle speed c. In other words, the distribution function is that giving the number of particles having speed between c and $c + dc$.

It is important to think about the distribution of particles in velocity space, a three-dimensional space (c_x, c_y, c_z), where each particle is represented by a point having coordinates corresponding to the particle's velocity. Thus, all points lying on a spherical surface centered at the origin correspond to the same speed. Therefore, the number of particles having speed between c and $c + dc$ equals the number of points lying between two shells of the sphere, with radii c and $c + dc$, Fig. 1.4.

The volume of the spherical shell is $4\pi c^2 dc$. Therefore, the probability distribution as a function of speed is:

$$f(c)dc = 4\pi c^2 A^3 e^{-Bc^2} dc \tag{1.19}$$

Integration of above function for the given Fig. 1.4 yields eight particles.

The constants A and B can be determined by integrating the probability distribution over all possible speeds to find the total number of particles N, and their total energy E.

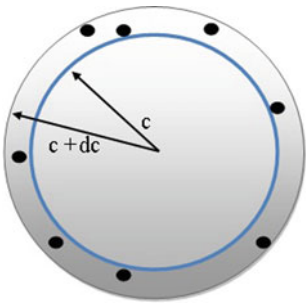

Fig. 1.4 Phase diagram

Since a particle moving at speed c has kinetic energy $\frac{1}{2}mc^2$, we can use the probability distribution function to find the average kinetic energy per particle, as:

$$\overline{\frac{1}{2}mc^2} = \frac{\int_0^\infty \frac{1}{2}mc^2 f(c)\,\mathrm{d}c}{\int_0^\infty f(c)\,\mathrm{d}c} \tag{1.20}$$

The numerator is the total energy, the denominator is the total number of the particles. Notice that the unknown constant A cancels between numerator and denominator. Substituting the value of $f(c)$ in the integrals, yields

$$\overline{\frac{1}{2}mc^2} = \frac{3m}{4B} \tag{1.21}$$

Substituting the value for the average kinetic energy in terms of the temperature of the gas (Eq. 1.12),

$$\overline{\frac{1}{2}mc^2} = \frac{3}{2}kT \tag{1.22}$$

Hence, $B = m/2kT$, so

$$f(c) \propto c^2 \mathrm{e}^{-\frac{mc^2}{2kT}} \tag{1.23}$$

The constant of proportionality is given by integrating over all speeds and setting the result as equal to one (since we factored out the number of particles N in our definition of $f(c)$).

The final result is:

$$f(c) = 4\pi\left(\frac{m}{2\pi kT}\right)^{\frac{3}{2}} c^2 \mathrm{e}^{-\frac{mc^2}{2kT}} \tag{1.24}$$

Note that this function increases parabolically from zero for low speeds, reaches a maximum value and then decreases exponentially. As the temperature increases, the position of the maximum shifts to the right. The total area under the curve is always one, by definition. This equation called Maxwell or Maxwell–Boltzmann distribution function.

The probability of finding a particle that has a specific velocity is zero, because the velocity of particles change continuously over a wide range. The meaningful question is to find the probability of a particle or particles within a range of velocity rather than at a specific velocity. Therefore, Eq. 1.24, need to be integrated in that range of velocity.

Example For air molecules (say, nitrogen) at 0 and 100°C temperatures, calculate the distribution function.

The mass of one molecule of N_2, which is molar mass (28 g/mol or 0.028 kg/mol) divided by Avogadro's number ($6.022 \times 10^{23}\,\text{mol}^{-1}$) gives $4.6496180671 \times 10^{-26}$ kg, which is mass of one N_2 molecule. The Boltzmann constant is 1.38×10^{-23} J/K $(\text{kg m}^2/(\text{s}^2\,\text{K}))$.

Then $m/(2k) = 4.6496180671 \times 10^{-3}/(2 * 1.38) = 1.684644227 \times 10^{-3}$ $(\text{m}^2/(\text{s}^2\,\text{K}))$. □

Figure 1.5 shows the probability of as a function of molecular velocity (c) for N_2 at $T = 273$ and 373 K.

The area under each curve is unity. As the temperature increases, the number of molecules with high velocity increases. The most probable speed is equal to

$$\sqrt{\frac{2kT}{m}} \tag{1.25}$$

This can be obtained by setting the derivative of the distribution function with respect velocity to zero, and then solve for velocity.

The average speed is equal to

$$\langle c \rangle = \sqrt{\frac{8kT}{\pi m}} \tag{1.26}$$

which is the weighted average velocity. It can be obtained by integrating distribution function from zero to infinity, as

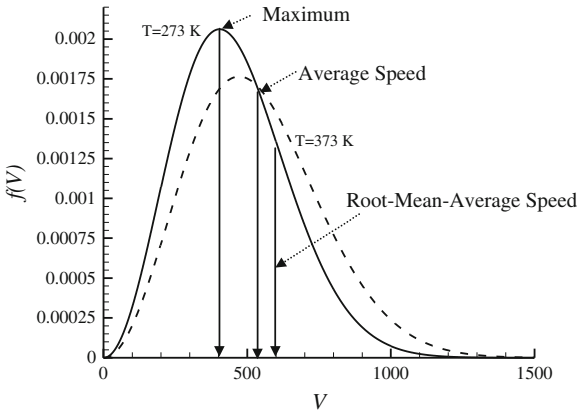

Fig. 1.5 The probability distribution function for nitrogen gas as a function of molecular velocity, c

$$\langle c \rangle = \int_0^\infty c f(c)\, \mathrm{d}c \tag{1.27}$$

The root-mean-average speed is equal to

$$\langle c^2 \rangle = \int_0^\infty c^2 f(c)\, \mathrm{d}c = \frac{3kT}{m} \tag{1.28}$$

The mean average speed is equal to $(c_x^2 + c_y^2 + c_z^2)^{1/2}$ and average speed is equal to $(c_x + c_y + c_z)/3$.

The root mean squared speed of a molecule is

$$c_{\mathrm{rms}} = \sqrt{\langle c^2 \rangle} = \sqrt{\frac{3K_B T}{m}} \tag{1.29}$$

Lighter gases usually have higher molecular speeds than heavy gases (higher molecular weight). Figure 1.6 illustrates the probability function for H_2 (2 kg/kmole), N_2 (28 kg/kmole) and CO_2 (44 kg/kmole).

At room temperature, most N_2 molecules travel at speeds around 500 m/s, while most hydrogen molecules travel at speed around 1,600 m/s (supersonic rockets).

Exercise Calculate the average and rms speed of hydrogen molecule at room temperature (25°C).

Exercise For a Maxwellian distribution function, calculate the total number of molecules striking unit area of a wall per unit time. What will be the pressure on the wall?

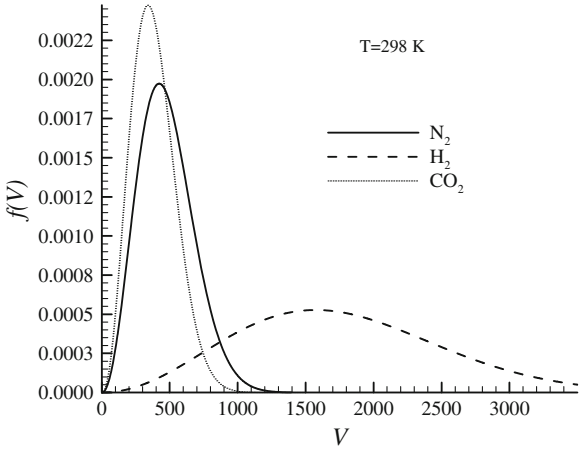

Fig. 1.6 Distribution function for N_2, H_2 and CO_2 at room temperature

1.3.1 Boltzmann Distribution

Boltzmann generalized the Maxwell's distribution for arbitrary large systems. He was the first to realize the deep connection between the thermodynamic concept of entropy and the statistical analysis of possible states of a large system—that the increase in entropy of a system with time is a change in macroscopic variables to those values corresponding to the largest possible number of microscopic arrangements. Boltzmann showed that the numbers of available microscopic states for a given energy are far greater for macroscopic values corresponding to thermal equilibrium. For example, for a given energy there are far more possible microscopic arrangements of gas molecules in which the gas is essentially uniformly distributed in a box than that of all the gas molecules being on the left-hand half of the box. Thus, if a liter of gas over the course of time goes through all possible microscopic arrangements, in fact there is a negligible probability of it all being in the left-hand half in a time the age of the universe. So if we arrange for all the particles to be in the left-hand half by using a piston to push them there, then remove the piston, they will rapidly tend to a uniform distribution spread evenly throughout the box.

Boltzmann proved that the thermodynamic entropy S, of a system (at a given energy E) is related to the number W, of microscopic states available to it by $S = k\log(W)$, k being Boltzmann's constant. There were some ambiguities in counting the number of possible microscopic arrangements which were rather troublesome, but not fatal to the program. For example, how many different velocities can a particle in a box have? This matter was cleared up by the quantum mechanics.

Boltzmann was then able to establish that for any system large or small in thermal equilibrium at temperature T, the probability of being in a particular state at energy E is proportional to $e^{\frac{-E}{kT}}$, i.e.

$$f(E) = Ae^{-E/kT} \tag{1.30}$$

This is called the Boltzmann distribution.

Let us consider kinetic energy of molecules in x-direction, then

$$E = \frac{1}{2}mc_x^2 \tag{1.31}$$

For a normalized probability function, the probability function integrated for all values of velocity (from minus to plus infinity) should be one.

Hence,

$$\int_{-\infty}^{\infty} Ae^{-\frac{mc_x^2}{2kT}}\,dc = 1 \tag{1.32}$$

Therefore,

$$A = \sqrt{\frac{m}{2\pi kT}} \tag{1.33}$$

The probability of finding velocity c_x is

$$f(c_x) = \sqrt{\frac{m}{2\pi kT}} \mathrm{e}^{-\frac{mc_x^2}{2kT}} \tag{1.34}$$

We are interested on probability of three dimensional velocity (c) where

$$c^2 = c_x^2 + c_y^2 + c_z^2 \tag{1.35}$$

The probability of (c) is multiple of probability of each function, i.e.,

$$f(c) = f(c_x) f(c_y) f(c_z) \tag{1.36}$$

which leads to

$$f(c) = \left[\sqrt{\frac{m}{2\pi kT}}\right]^3 \mathrm{e}^{-\frac{m}{2kT}(c_x^2 + c_y^2 + c_z^2)} \tag{1.37}$$

or

$$f(c) = \left(\frac{m}{2\pi kT}\right)^{3/2} \mathrm{e}^{-\frac{mc^2}{2kT}} \tag{1.38}$$

It should be noted that the above equation does not take into account the fact that there are more ways to achieve a higher velocity. In making the step from this expression to the Maxwell speed distribution, this distribution function must be multiplied by the factor $4\pi c^2$ (which is surface area of a sphere in the phase space) to account for the density of velocity states available to particles. Therefore, Maxwell distribution function (Eq. 1.24) is covered. In fact, integration of Maxwell distribution function (Eq. 1.24) over a surface of sphere in phase space yields Eq. 1.37.

An ideal gas has a specific distribution function at equilibrium (Maxwell distribution function). But Maxwell did not mention, how the equilibrium is reached. This was one of the revolutionary contribution of Boltzmann, which is the base of the LBM.

In the next chapter, Boltzmann transport equation, which is a main concern of this book, will be discussed.

Chapter 2
The Boltzmann Equation

Ludwig Eduard Boltzmann (1844–1906), the Austrian physicist whose greatest achievement was in the development of statistical mechanics, which explains and predicts how the properties of atoms and molecules (microscopic properties) determine the phenomenological (macroscopic) properties of matter such as the viscosity, thermal conductivity, and diffusion coefficient. The distribution function (probability of finding particles within a certain range of velocities at a certain range of locations at a given time) replaces tagging each particle, as in molecular dynamic simulations. The method saves the computer resources drastically.

In this chapter, the main concept of the Boltzmann equation is introduced.

2.1 Boltzmann Transport Equation

A statistical description of a system can be explained by distribution function $f(r, c, t)$, where $f(r, c, t)$ is the number of molecules at time t positioned between r and $r + \mathrm{d}r$ which have velocities between c and $c + \mathrm{d}c$, as mentioned in the previous chapter. An external force F acting on a gas molecule of unit mass will change the velocity of the molecule from c to $c + F\mathrm{d}t$ and its position from r to $r + c\mathrm{d}t$ (Fig. 2.1).

The number of molecules, $f(r, c, t)$, before applying the external force is equal to the number of molecules after the disturbance, $f(r + c\mathrm{d}t, c + F\mathrm{d}t, t + \mathrm{d}t)$, if no collisions take place between the molecules. Hence,

$$f(r + c\mathrm{d}t, c + F\mathrm{d}t, t + \mathrm{d}t)\mathrm{d}r\mathrm{d}c - f(r, c, t)\mathrm{d}r\mathrm{d}c = 0 \tag{2.1}$$

However, if collisions take place between the molecules there will be a net difference between the numbers of molecules in the interval $\mathrm{d}r\mathrm{d}c$. The rate of change between final and initial status of the distribution function is called collision operator, Ω. Hence, the equation for evolution of the number of the molecules can be written as,

A. A. Mohamad, *Lattice Boltzmann Method*,
DOI: 10.1007/978-0-85729-455-5_2, © Springer-Verlag London Limited 2011

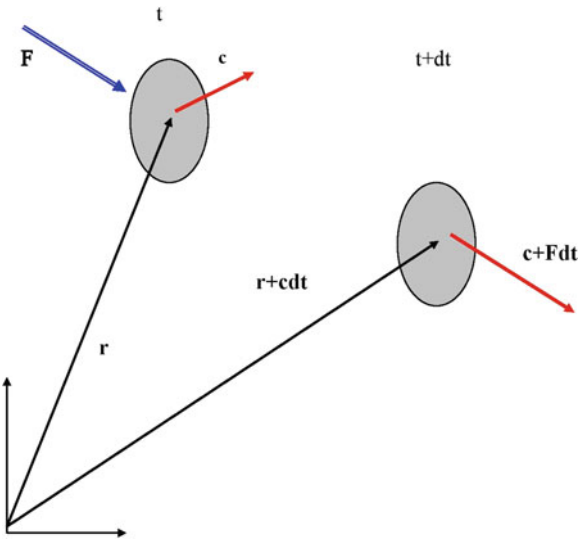

Fig. 2.1 Position and velocity vector for a particle after and before applying a force, F

$$f(r + c\mathrm{d}t, c + F\mathrm{d}t, t + \mathrm{d}t)\mathrm{d}r\mathrm{d}c - f(r, c, t)\mathrm{d}r\mathrm{d}c = \Omega(f)\mathrm{d}r\mathrm{d}c\mathrm{d}t \tag{2.2}$$

Dividing the above equation by $\mathrm{d}t\mathrm{d}r\mathrm{d}c$ and as the limit $\mathrm{d}t \to 0$, yields

$$\frac{\mathrm{df}}{\mathrm{d}t} = \Omega(f) \tag{2.3}$$

The above equation states that the total rate of change of the distribution function is equal to the rate of the collision. Since f is a function of r, c and t, then the total rate of change can be expanded as,

$$\mathrm{d}f = \frac{\partial f}{\partial r}\mathrm{d}r + \frac{\partial f}{\partial c}\mathrm{d}c + \frac{\partial f}{\partial t}\mathrm{d}t \tag{2.4}$$

Dividing by $\mathrm{d}t$, yields

$$\frac{\mathrm{df}}{\mathrm{d}t} = \frac{\partial f}{\partial r}\frac{\mathrm{dr}}{\mathrm{d}t} + \frac{\partial f}{\partial c}\frac{\mathrm{dc}}{\mathrm{d}t} + \frac{\partial f}{\partial t} \tag{2.5}$$

The vector r can be expressed in 3-D Cartesian coordinate system as $r = xi + yj + zk$, where i, j, and k are unit vectors along x, y, and z-direction, respectively.

Equation 2.5 can be written as,

$$\frac{\mathrm{df}}{\mathrm{d}t} = \frac{\partial f}{\partial r}c + \frac{\partial f}{\partial c}a + \frac{\partial f}{\partial t} \tag{2.6}$$

where a is equal to $\mathrm{d}c/\mathrm{d}t$, the acceleration and can be related to force F by Newton's second law, $a = F/m$.

Therefore, the Boltzmann transport equation (2.3) can be written as,

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial r} \cdot c + \frac{F}{m} \cdot \frac{\partial f}{\partial c} = \Omega \tag{2.7}$$

The Ω is a function of f and need to be determined to solve the Boltzmann equation.

For system without an external force, the Boltzmann equation can be written as,

$$\frac{\partial f}{\partial t} + c \cdot \nabla f = \Omega \tag{2.8}$$

Note that c and ∇f are vectors.

Equation 2.8 is an advection equation with a source term (Ω), or advection with a reaction term, which can be solved exactly along the characteristic lines that is tangent to the vector c, if Ω is explicitly known. The problem is that Ω is a function of f and Eq. 2.8 is an integro-differential equation, which is difficult to solve.

The relation between the above equation and macroscopic quantities such as fluid density, ρ, fluid velocity vector u, and internal energy e, is as follows

$$\rho(r,t) = \int mf(r,c,t)\,\mathrm{d}c \tag{2.9}$$

$$\rho(r,t)u(r,t) = \int mcf(r,c,t)\,\mathrm{d}c \tag{2.10}$$

$$\rho(r,t)e(r,t) = \frac{1}{2}\int mu_a^2 f(r,c,t)\,\mathrm{d}c \tag{2.11}$$

where m is the molecular mass and u_a the particle velocity relative to the fluid velocity, the peculiar velocity, $u_a = c - u$.

Equations 2.9, 2.10, and 2.11 are conservation of mass, momentum, and energy, respectively.

From the kinetic theory, as discussed before, the internal energy can be expressed as,

$$e = \frac{3}{2m}k_B T \tag{2.12}$$

2.1.1 Example 2.1

In a road intersection, if there is no car collisions, the cars from section A move (stream) to A′ and cars from section B move to B′ without any problem. This case represents Eq. 2.8 with $\Omega = 0$, see Fig. 2.2. However, if there is a collision at the intersection, then cars at section A cannot smoothly move to A′ or from B to B′. The time needed to reach smooth traffic (equilibrium conditions) depend on the type of the collision and the response of the policeman (relaxation time). In this case Ω is not zero.

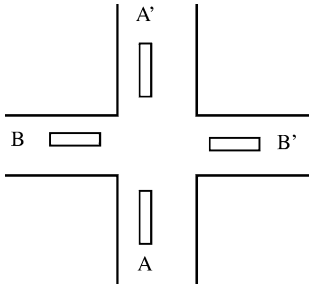

Fig. 2.2 Traffic intersection

2.2 The BGKW Approximation

It is difficult to solve Boltzmann equation because the collision term is very complicated. The outcome of two body collisions is not likely to influence significantly, the values of many measured quantities (Cercignani, 1990). Hence, it is possible to approximate the collision operator with simple operator without introducing significant error to the outcome of the solution. Bhatnagar, Gross and Krook (BGK) in 1954 introduced a simplified model for collision operator. At the same time Welander (1954), independently, introduced similar operator. The collision operator is replaced as,

$$\Omega = \omega(f^{\mathrm{eq}} - f) = \frac{1}{\tau}(f^{\mathrm{eq}} - f) \tag{2.13}$$

where $\omega = 1/\tau$

The coefficient ω is called the collision frequency and τ is called relaxation factor. The local equilibrium distribution function is denoted by f^{eq}, which is Maxwell–Boltzmann distribution function.

After introducing BGKW approximation, the Boltzmann equation (Eq. 2.8, without external forces) can be approximated as,

$$\frac{\partial f}{\partial t} + c \cdot \nabla f = \frac{1}{\tau}(f^{\mathrm{eq}} - f) \tag{2.14}$$

In Lattice Boltzmann method, the above equation is discretized and assumed it is valid along specific directions, linkages. Hence, the discrete Boltzmann equation can be written along a specified direction as,

$$\frac{\partial f_i}{\partial t} + c_i \nabla f_i = \frac{1}{\tau}(f_i^{\mathrm{eq}} - f_i) \tag{2.15}$$

The above equation is the working horse of the lattice Boltzmann method and replaces Navier–Stokes equation in CFD simulations. It is possible to derive Navier–Stokes equation from Boltzmann equation. We can make comments as the followings about Eq. 2.15:

1. The equation is a linear partial differential equation.
2. The equation looks like an advection equation with a source term.
3. The right-hand side of the equation represents the advection (streaming).
4. The left-hand side term represents the collision process, source term.

Equation 2.15 can be discretized as

$$f_i(r + c_i\Delta t, t + \Delta t) = f_i(r, t) + \frac{\Delta t}{\tau}[f_i^{\mathrm{eq}}(r, t) - f_i(r, t)] \tag{2.16}$$

The local equilibrium distribution function with a relaxation time determine the type of problem needed to be solved. The beauty of this equation lies in its simplicity and can be applied for many physics by simply specifying a different equilibrium distribution function and source term (external force). Adding a source term (force term) to the above equation is straightforward. However, there are a few concerns, which will be discussed in the following chapters. Also, the details of implementing the above equation for different problems, such as momentum, heat and mass diffusion, advection–diffusion without and with external forces, will be presented in the following chapters.

It is possible to use finite difference or finite volume to solve partial differential equation (2.15). Some authors used this approach to solve fluid dynamic problems on non-uniform grids. The main focus of the book is to solve Eq. 2.15 in two steps, collision and streaming.

In LBM, the solution domain need to be divided into lattices. At each lattice node, the factitious particles (distribution function) reside. Some of these particles streams (move) along specified directions to the neighboring nodes. The number of direction, linkage, depends on the lattice arrangement. Different lattice arrangements will be discussed in the following section.

2.3 Lattice Arrangements

The common terminology used in LBM is to refer to the dimension of the problem and the number of speed is using *DnQm*, where *n* represent the dimension of the problem (1 for 1-D, 2 for 2-D and 3 for 3-D) and *m* refers to the speed model, number of linkages. In the following paragraphs different lattice arrangements used for 1-D, 2-D, and 3-D will be discussed.

2.3.1 One-Dimensional

In general, two models can be used for lattice arrangements, called D1Q3Q and D1Q5Q, as shown in Fig. 2.3. D1Q3 is the most popular one. The black nodes are the central node, while the gray nodes are neighboring nodes. The factitious

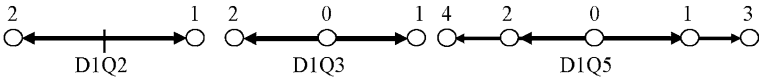

Fig. 2.3 Lattice arrangements for 1-D problems

particles stream from the central node to neighboring nodes through linkages with a specified speed, called lattice speed.

2.3.1.1 D1Q3 and D1Q2

For D1Q3, there are three velocity vectors (c_0, c_1 and c_2) for f_0,$f_1$and f_2, which equal to 0, 1, and -1, respectively. Note that we assumed that $\mathrm{d}x = \mathrm{d}t$, otherwise, $c_1 = \Delta x/\Delta t$ and $c_2 = -\Delta x/\Delta t$,where Δx and Δt are the linkage length and time step, respectively. For this arrangement, the total number of factitious particles at any instant of time cannot exceed three particles. One stagnant particle (zero velocity) resides on the central site. The other two particles move either to the left or to the right node in the streaming process. The weighting factors, ω_i, has values of 4/6, 1/6 and 1/6 for f_0,$f_1$and f_2, respectively. The speed of sound,c_s, in lattice units for D1Q3 is $1/\sqrt{3}$. It is also possible to use other arrangement called D1Q2. The weighting factors,ω_i, has values of 1/2 and 1/2 for $f_1$and f_2, respectively. The speed of sound for this arrangement is $1/\sqrt{2}$. These schemes, D1Q2 and D1Q3, are mostly used. However, it is possible to involve more sites and use higher order schemes, such as, D1Q5.

2.3.1.2 D1Q5

For this arrangement, the total number of factitious particles at any instant of time cannot exceeded five particles. The weighting factors, ω_i, are 6/12, 2/12, 2/12, 1/12, and 1/12 for f_0,f_1,f_2,f_3, and f_4, respectively. The speed of the sound in lattice units is $1/\sqrt{3}$.

2.3.2 Two-Dimensional

2.3.2.1 D2Q5 and D2Q4

D2Q5 model has four velocity vectors issued from the central nodes, Fig. 2.4. One of the particle resides at the central node, hence its speed is zero, noted as $c(0,0)$.The distribution function $f_1$and f_2 move with $c(1,0)$ and $c(-1,0)$ (to the east and west), respectively, while $f_3$and f_4 move with speed $c(0,1)$ and $c(0,-1)$ (to the north and south), respectively. Note that it is assumed that $\Delta x = \Delta y = \Delta t$.

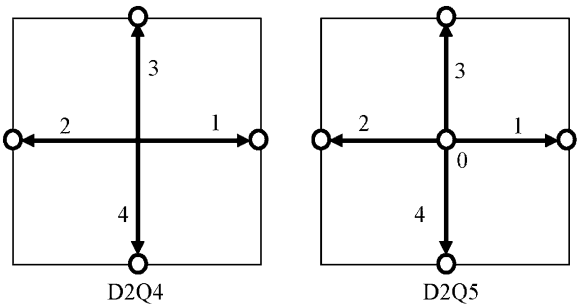

Fig. 2.4 Lattice arrangements for 2-D problems, D2Q4 and D2Q5.

The weighting factors for f_0, f_1, f_2, f_3, and f_4 are 2/6, 1/6, 1/6, 1/6, and 1/6, respectively. It is worthy to mention that this arrangement cannot be used to simulate fluid flows. This issue will be discussed latter on. D2Q4 has four velocity vectors and there is no particle residing on the centre node. The weighting factor for each direction is 1/4. We will discuss implementations and applications of each scheme in the following chapters. The numbering of lattice links is arbitrary. However, for computer programming algorithm, proper numbering may reduce simplifying the computer code.

2.3.2.2 D2Q9

This model is very common, especially for solving fluid flow problems. It has high velocity vectors, with the central particle speed being zero, 2.5. The speeds are $c(0,0), c(1,0), c(0,1), c(-1,0), c(0,-1), c(1,1), c(-1,1), c(-1,-1)$ and $c(1,-1)$ for $f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7$ and f_8, respectively. The weighting factors for corresponding distribution functions are 4/9, 1/9, 1/9, 1/9, 1/9, 1/36, 1/36, 1/36, and 1/36 (Fig. 2.5).

2.3.3 Three-Dimensional

In general two models are used in simulation of three dimensional problems, D3Q15 and D3Q19.

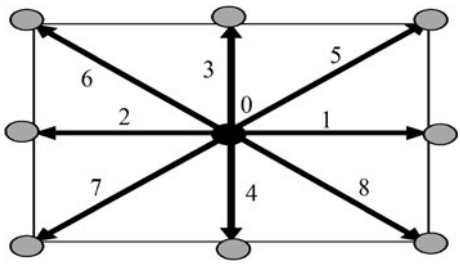

Fig. 2.5 Lattice arrangements for 2-D problems, D2Q9.

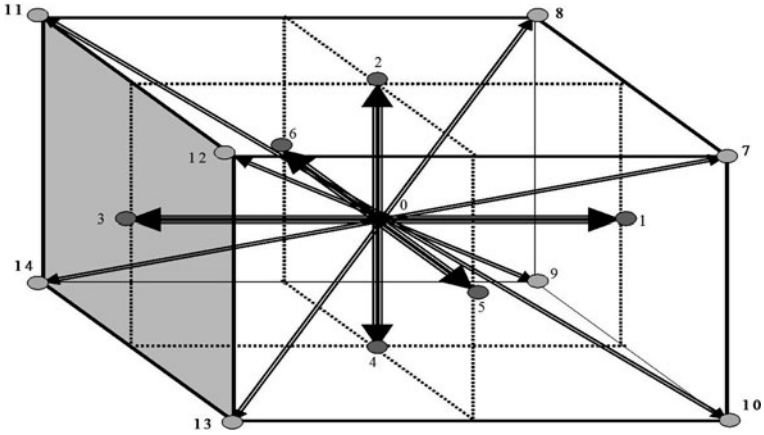

Fig. 2.6 Lattice arrangements for 3-D problems, D3Q15.

2.3.3.1 D3Q15

In this model 15 velocity vectors are used, Fig. 2.6, the central distribution function, f_0 has zero speed. D3Q15 is most commonly used for 3-D simulations.

Notice that nodes 1, 2, 3, and 4 are on the center of the east, north, west, and south faces, respectively. Nodes 5 and 6 are on the center of the front and back faces, respectively. The nodes 7, 8, 9, 10, 11, 12, 13, and 14 are on the corners of the lattice. The 15 velocity vectors for the distribution functions of $f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}$ and f_{14} are

$c(0,0,0), c(1,0,0), c(0,1,0), c(-1,0,0), c(0,-1,0), c(0,0,1), c(0,0,-1),$
$c(1,1,1), c(1,1,-1), c(1,-1,-1), c(1,-1,1), c(-1,1,-1), c(-1,1,1),$
$c(-1,-1,1)$ and $c(-1,-1,-1)$, respectively. The weighting factors are
16/72 for f_0
8/72 for f_1 to f_6 and
1/72 for f_7 to f_{14}.

2.3.3.2 D3Q19

This model has 19 velocity vectors, with central vector of speed zero, Fig. 2.7.

The weighting factors are:

for f_0 is 12/36
for f_1 to $f_6$is 2/36
for f_7 to f_{18} is 1/36.

It is possible to use D3Q7 or D2Q6 similar to D2Q5 and D2Q4 for advection–diffusion problems.

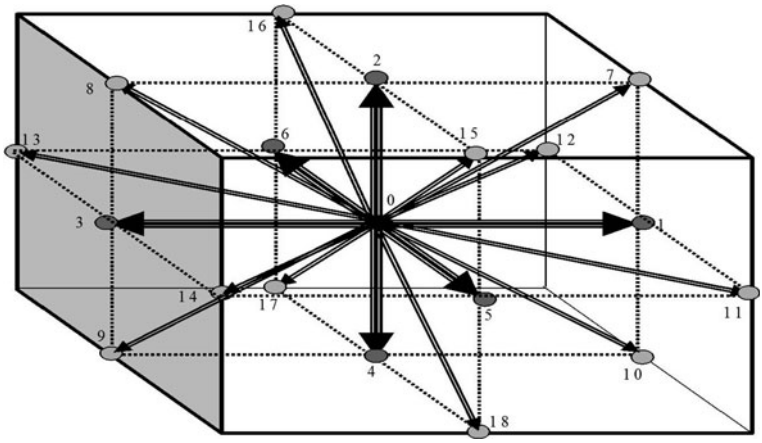

Fig. 2.7 Lattice arrangements for 3-D problems, D3Q19.

2.4 Equilibrium Distribution Function

The key element in applying LBM for different problems is the equilibrium distribution function, f^{eq}. As we will see in the following chapters that the procedure of solving diffusion, advection–diffusion, momentum, and energy equations is the same. The difference mainly depends on the equilibrium distribution function. In fact, different physical problems (such as a wave propagation problem, etc.) can be solved by LBM provided that a proper equilibrium distribution function is used.

For particles moving in a medium with macroscopic velocity $\mathbf{u}$. The normalized Maxwell's distribution function (Eq. 1.38) can be written as,

$$f = \frac{\rho}{2\pi/3} \mathrm{e}^{-\frac{3}{2}(\mathbf{c}-\mathbf{u})^2} \tag{2.17}$$

which can be written as,

$$f = \frac{\rho}{2\pi/3} \mathrm{e}^{-\frac{3}{2}(c^2)} \mathrm{e}^{3(\mathbf{c}\cdot\mathbf{u}-u^2)/2} \tag{2.18}$$

where $c^2 = \mathbf{c}\cdot\mathbf{c}$ and $u^2 = \mathbf{u}\cdot\mathbf{u}$. Recall that Taylor series expansion for e^{-x} is,

$$\mathrm{e}^{-x} = 1 - x + x^2/2 - x^3/(3!)\cdots \tag{2.19}$$

Hence Eq. 2.18 can be expanded around the stationary state as,

$$f = \frac{\rho}{2\pi/3} \mathrm{e}^{-\frac{3}{2}(c^2)} \left[1 + 3(\mathbf{c}\cdot\mathbf{u}) - \frac{3}{2}u^2 + \cdots\right] \tag{2.20}$$

The general form of the equilibrium distribution function can be written as,

$$f_i^{\mathrm{eq}} = \Phi\omega_i\left[A + Bc_i \cdot u + C(c_i \cdot u)^2 + Du^2\right] \tag{2.21}$$

where u is the macroscopic flow velocity vector; A, B, C, and D are constants and need to be determined based on the conservation principle (mass, momentum, and energy). Φ stands for scalar parameter, such as density (ρ), temperature (thermal energy density), or species concentration, which is equal to summation of all the distribution functions, i.e.,

$$\Phi = \sum_{i=0}^{i=n} f_i^{\mathrm{eq}} \tag{2.22}$$

where n is the number of the lattice links. For example, in a diffusion process, the flow is stagnant, velocity is zero, then Eq. 2.17 can be reduced to:

$$f_i^{\mathrm{eq}} = \Phi A\omega_i \tag{2.23}$$

It turns out that A is equal to unity, where $\sum_{i=0}^{i=n} \omega_i = 1$. In the following c3hapters more details will be given about the equilibrium distribution function.

Chapter 3
The Diffusion Equation

3.1 Diffusion Equation

One dimensional diffusion equation can be written as

$$\frac{\partial \phi}{\partial t} = \alpha \frac{\partial^2 \phi}{\partial x^2} \tag{3.1}$$

the dependent variable ϕ (such as temperature, species, momentum, etc.) diffuses in an infinite medium in both directions (to the left and right, x^+ and x^-) without any preference. The rate of diffusion depends on the parameter α, where α stands for thermal diffusion coefficient, mass diffusion coefficient or kinematics viscosity, for energy, mass, and momentum diffusion, respectively. The diffusion process becomes faster as the parameter α increases. Order of magnitude analysis yields,

$$\frac{1}{\tau} \approx \alpha \frac{1}{\ell^2} \tag{3.2}$$

where τ and l are time and length scales, respectively.

For a given time, the scalar ϕ diffusive to distance l is proportional to

$$\ell \approx \sqrt{\alpha \tau} \tag{3.3}$$

Equation 3.1 has second derivative in space, therefore, the diffusion takes place in both directions and requires two boundary conditions. Also, Eq. 3.1 has a first derivative in time, the diffusion is one directional in time, in other words, the diffusion at any point depends on the previous time and no information can be transferred from the future time. Also, it requires an initial condition to solve Eq. 3.1.

For instance, if a drop of ink is added to a glass filled with water, the ink diffuses in all directions and concentration of the ink at any spot in the water decreases as the time proceeds until it reaches the equilibrium condition.

A. A. Mohamad, *Lattice Boltzmann Method*,
DOI: 10.1007/978-0-85729-455-5_3, © Springer-Verlag London Limited 2011

The concentration of ink at any location depends on the pervious time and on the concentration at the boundary of that spot.

3.1.1 Example 3.1

One-dimensional heat diffusion equation can be written as,

$$\rho C \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) \tag{3.4}$$

where T is temperature, ρ, C and k are density, specific heat and thermal conductivity of the medium, respectively. The heat diffusion due to molecular action, depends on thermal conductivity (k), density (ρ) and specific heat (C). For a constant k, the above equation can be written as

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{3.5}$$

where α is the thermal diffusivity ($k/\rho C$). Hence, the only parameter that controls the diffusion of heat is the thermal diffusivity of the medium, which is a property of the material.

3.1.2 Example 3.2

One-dimensional momentum diffusion can be written as,

$$\frac{\partial u}{\partial t} = v \frac{\partial^2 u}{\partial y^2} \tag{3.6}$$

where u is velocity, v is kinematic viscosity of the fluid. For fluid between parallel plates (Fig. 3.1) if the upper plate is set to a motion, the momentum diffuses according to the above equation. The rate of momentum diffusion depends on the

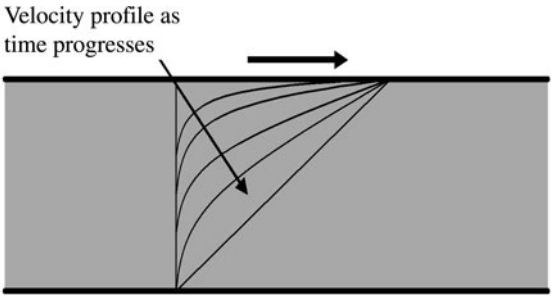

Fig. 3.1 Momentum diffusion in a fluid confined between two parallel plates due to the motion of the upper lid

viscosity of the fluid. The time needed for the bottom plate to sense the motion of the upper plate depends on the viscosity of the fluid (high viscosity means less sensation time), and the distance between the plates (Eq. 3.3).

3.2 Finite Differences Approximation

In the following section, numerical solution using finite difference approximation for one dimensional diffusion problem will be discussed in detail. Extending the method of solution to two and three dimensional problems is a straightforward procedure.

The main objectives of the approximation of the diffusion equation using the finite difference are two folds; first to show the difference and similarity between finite difference approximation and lattice Boltzmann method (LBM); also, to compare the results of the two methods.

As a first step, the domain need to be divided (discretized) into equal segments (it is not necessary for the length of each segment to be equal, however for simplicity we use equal segments) as shown in Fig. 3.2. The nodes are labeled as $0, 1, 2, 3, \ldots, i-2, i-1, i, i+1, i+2, \ldots n$.

An explicit finite difference approach can be used, forward in time and central differences in space. Approximating the diffusion equation at a node i, yields,

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} \tag{3.7}$$

The above equation can be rearranged as:

$$T_i^{n+1} = T_i^n + \frac{\alpha \Delta t}{\Delta x^2}\left(T_{i+1}^n - 2T_i^n + T_{i+1}^n\right) \tag{3.8}$$

Equation 3.8 can be reformulated as,

$$T_i^{n+1} = T_i^n\left(1 - \frac{2\alpha\Delta t}{\Delta x^2}\right) + \frac{2\alpha\Delta t}{\Delta x^2}\left(\frac{T_{i+1}^n + T_{i+1}^n}{2}\right) \tag{3.9}$$

Let

$$\omega = \frac{2\alpha\Delta t}{\Delta x^2} \tag{3.10}$$

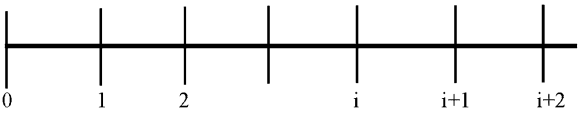

Fig. 3.2 One dimensional node distribution

then Eq. 3.9 can be written as

$$T_i^{n+1} = T_i^n(1-\omega) + \omega\left(0.5T_{i+1}^n + 0.5T_{i-1}^n\right) \tag{3.11}$$

For stability condition, the coefficients of the right-hand side terms must be positive. Hence, the term $(1-\omega)$ must be greater than or equal to zero, which implies that

$$\Delta t \leq \frac{\Delta x^2}{2\alpha}$$

The finite difference approximation for the diffusion equation is manipulation in the form of Eq. 3.11 to compare the finite difference formulation with LBM.

Let us examine Eq. 3.11, the last term $(0.5[T_{i+1}^n + T_{i-1}^n])$ is an average of temperatures around T_i. In other words, the average term represents the equilibrium value of T_i. Since, T_i is at mid-distance from T_{i+1} and T_{i-1}, then at the equilibrium state, the value of T_i should be the average of it's neighbors values. It is appropriate to write the term $0.5[T_{i+1}^n + T_{i-1}^n]$ as T_i^{eq}, then Eq. 3.11, can be rewritten as,

$$T_i^{n+1} = T_i^n(1-\omega) + \omega T_i^{\text{eq}} \tag{3.12}$$

In the next section we can see that Eq. 3.12 resembles LB equation (3.18).

3.3 The Lattice Boltzmann Method

The kinetic equation for the distribution function (temperature distribution, species distributions, 12, etc.), $f_k(x,t)$ can be written as:

$$\frac{\partial f_k(x,t)}{\partial t} + c_k \cdot \frac{\partial f_k(x,t)}{\partial x} = \Omega_k \tag{3.13}$$

$k = 1, 2$ (for one dimensional problem, D1Q2).

The left-hand side terms represent the streaming process, where the distribution function streams (advects) along the lattice link with velocity

$$c_k = \frac{\Delta x}{\Delta t} \tag{3.14}$$

The right-hand term, Ω_k, represents the rate of change of distribution function, f_k, in the collision process.

BGK approximation for the collision operator can be approximated as,

$$\Omega_k = -\frac{1}{\tau}\left[f_k(x,t) - f_k^{\text{eq}}(x,t)\right] \tag{3.15}$$

The term τ represents a relaxation time toward the equilibrium distribution (f_k^{eq}), which is related to the diffusion coefficient on the macroscopic scale. The relation will be discussed latter on.

The kinetic LB equation (3.13) with BGK approximation can be discretized as,

$$\frac{f_k(x,t+\Delta t)-f_k(x,t)}{\Delta t}+c_k\cdot\frac{f_k(x+\Delta x,t+\Delta t)-f_k(x,t+\Delta t)}{\Delta x} = -\frac{1}{\tau}\left[f_k(x,t)-f_k^{eq}(x,t)\right] \tag{3.16}$$

Note that $\Delta x = c_k \Delta t$, substituting in Eq. 3.16, which can be simplified to:

$$f_k(x+\Delta x,t+\Delta t)-f_k(x,t) = -\frac{\Delta t}{\tau}\left[f_k(x,t)-f_k^{eq}(x,t)\right] \tag{3.17}$$

The above equation is the working horse for the diffusion problem in one dimensional space, which can be reformulated as,

$$f_k(x+\Delta x,t+\Delta t) = f_k(x,t)[1-\omega]+\omega f_k^{eq}(x,t) \tag{3.18}$$

where

$$\omega = \frac{\Delta t}{\tau}$$

is called a relaxation time.

It is interesting to note that the above Eq. 3.17 is similar to finite difference equation (3.12).

Equation 3.17 represents a number of equations for different k values ($k = 1$ and 2), in each direction.

Figure 3.3 shows schematic diagram for three nodes with necessary linkages, central and neighboring nodes, where $c_1 = c, c_2 = -c$. Figure 3.4 shows lattices distribution for 1-D, D1Q2.

The dependent variable in Eq. 3.1, ϕ, can be related to the distribution function f_i, as,

$$\phi(x,t) = \sum_{k=1}^{2} f_k(x,t) \tag{3.19}$$

The equilibrium distribution function f_k^{eq} (see Eq. 2.19), can be chosen as

$$f_k^{eq} = w_k \phi(x,t) \tag{3.20}$$

where w_k stands for the weighting factor in the direction k. The weighting factor should satisfy the following criterion,

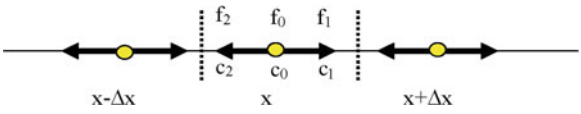

Fig. 3.3 Schematic diagram for 1-D lattice arrangement

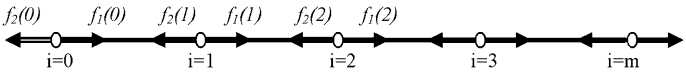

Fig. 3.4 Lattices for 1-D, diffusion problem

$$\sum_{k=1}^{2} w_k = 1 \tag{3.21}$$

The distribution function $f_k^{\mathrm{eq}}(x,t)$ defined in Eq. 3.20 can be summed along all directions and yields,

$$\sum_{i=0}^{2} f_k^{\mathrm{eq}}(x,t) = \sum_{i=0}^{2} w_k \phi(x,t) = \phi(x,t) \tag{3.22}$$

The relation between α and ω can be deduced from multi-scale expansion by using Chapman–Enskog expansion, which yields

$$\alpha = \frac{\Delta x^2}{\Delta t D}\left(\frac{1}{\omega} - \frac{1}{2}\right) \tag{3.23}$$

D is dimension of the problem ($D = 1$ for one dimensional problem, $D = 2$ for two dimensional problem and $D = 3$ for three dimensional problem). The weight factor for one-dimensional problem is $w_k = 1/2$, for $k = 1$ and 2.

In the following section Chapman–Enskog expansion will be illustrated for one dimensional diffusion equation. We will show how the relationship between diffusion coefficient and relaxation time can be established, step by step.

3.4 Equilibrium Distribution Function

For diffusion problem, it is appropriate that equilibrium distribution functions be assumed constants, where no macroscopic velocity is involved. Let,

$$f_i^{\mathrm{eq}} = A_i \tag{3.24}$$

Equilibrium distribution function must satisfy the conservation of mass and momentum,

$$\sum_{i=1}^{2} f_i^{\mathrm{eq}} = \Theta \tag{3.25}$$

and

$$\sum_{i=1}^{2} f_i^{\mathrm{eq}} c_i = 0 \tag{3.26}$$

Hence,

$$\sum_{i=1}^{2} A_i = A_1 + A_2 = \Theta \tag{3.27}$$

Also,

$$\sum_{i=1}^{2} A_i c_i = A_1 c_1 + A_2 c_2 = A_1 - A_2 = 0 \tag{3.28}$$

where $c_1 = 1$ and $c_2 = -1$. Hence $A_1 = A_2$ and from Eq. 3.51, $A_1 = \frac{\Theta}{2}$. In general,

$$A_i = \omega_i \Theta \tag{3.29}$$

Hence,

$$f_i^{\mathrm{eq}} = \omega_i \Theta \tag{3.30}$$

In the following sections, the relationship between LBM and macro-scale system will be discussed. In other words, the relationship between diffusion coefficient and LBM relaxation time will be established by the multi-scale analysis.

3.5 Chapman–Enskog Expansion

One dimensional diffusion equation in Cartesian coordinate can be written as,

$$\frac{\partial T(x,t)}{\partial t} = \Gamma \frac{\partial^2 T(x,t)}{\partial x^2} \tag{3.31}$$

where $T(x,t)$ is the dependent parameter (for instance, temperature for heat diffusion, velocity for momentum diffusion, species concentration for mass diffusion). It is assumed that diffusion coefficient, Γ, is a constant. The diffusion equation can be scaled spatially as, x is set to x/ϵ, where ϵ is a small parameter (Knudson number). Introducing the scale to diffusion equation yields,

$$\frac{\partial T(x,t)}{\partial t} = \Gamma \frac{\epsilon^2 \partial^2 T(x,t)}{\partial x^2} \tag{3.32}$$

Hence, in order to keep both sides of the equation at the same order of magnitude, the time t, must be scaled by $1/\epsilon^2$, i.e., t should be set to t/ϵ^2. For one-dimensional problem with two lattice velocity components ($c_i, i = 1, 2$), the temperature can be expressed as,

$$T(x,t) = \sum_{i=1}^{i=2} f_i(x,t) = f_1(x,t) + f_2(x,t) \tag{3.33}$$

From now and on, we write f_i instead of $f_i(x,t)$ for simplicity. Since the diffusion equation is linear, it is fair to assume that equilibrium distribution function is related to T (see pervious section for details) as,

$$f_i^{\text{eq}} = w_i T(x,t) \tag{3.34}$$

By taking summation of both sides of the above equation,

$$\sum_i f_i^{\text{eq}} = T = w_1 T + w_2 T \tag{3.35}$$

hence, $w_1 + w_2 = 1$, i.e., $w_1 = w_2 = \frac{1}{2}$, then,

$$f_1^{\text{eq}} = w_1 T(x,t) \quad \text{and} \quad f_2^{\text{eq}} = w_2 T(x,t) \tag{3.36}$$

where the weight factors must satisfy the relation

$$\sum_{i=1}^{i=2} w_i = 1$$

The distribution function can be expanded in terms of a small parameter, ϵ as

$$f_i(x,t) = f_i^0 + \epsilon f_i^1 + \epsilon^2 f_i^2 + \cdots \tag{3.37}$$

where f_i^0 is distribution function at the equilibrium conditions, equal to f_i^{eq} By summing above equation, it leads to,

$$\sum_{i=1}^{i=2} f_i(x,t) = \sum_{i=1}^{i=2} [f_i^0 + \epsilon f_i^1 + \epsilon^2 f_i^2 + \cdots] \tag{3.38}$$

Since, $f_1 + f_2 = T(x,t)$ and $f_1^0 + f_2^0 = T(x,t)$, hence other expanded term in the above equation should be zero, i.e.,

$$\sum_{i=1}^{i=2} f_i^1 = 0 \tag{3.39}$$

$$\sum_{i=1}^{i=2} f_i^2 = 0 \tag{3.40}$$

The updated distribution function $f_i(x + c_i\Delta t, t + \Delta t)$ is expanded using Taylor series,

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) + \frac{\partial f_i}{\partial t}\Delta t + \frac{\partial f_i}{\partial x}c_i\Delta t + 1/2\Delta t^2\left(\frac{\partial^2 f_i}{\partial t^2} + 2\frac{\partial^2 f_i}{\partial t \partial x}c_i + \frac{\partial^2 f_i}{\partial x^2}c_i c_i\right) + O(\Delta t)^3 \quad (3.41)$$

Introducing scaling for the above equation, i.e., $\frac{1}{\partial t}$ is replaced by $\frac{\epsilon^2}{\partial t}$ and $\frac{1}{\partial x}$ is replaced by $\frac{\epsilon}{\partial x}$, yields:

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) + \epsilon^2\frac{\partial f_i}{\partial t}\Delta t + \epsilon\frac{\partial f_i}{\partial x}c_i\Delta t + 1/2\Delta t^2\left(\frac{\epsilon^4\partial^2 f_i}{\partial t^2} + 2\frac{\epsilon^3\partial^2 f_i}{\partial t \partial x}c_i + \frac{\epsilon^2\partial^2 f_i}{\partial x^2}c_i c_i\right) + O(\Delta t)^3 \quad (3.42)$$

Lattice Boltzmann equation can be written as,

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) + \frac{\Delta t}{\tau}[f_i^0(x,t) - f_i(x,t)] \quad (3.43)$$

Substituting Eq. 3.42 into above equation and retaining terms up to order of ϵ^2 , yields:

$$\frac{1}{\tau}[f_i^0(x,t) - f_i(x,t)] = \epsilon^2\frac{\partial f_i}{\partial t} + \epsilon\frac{\partial f_i}{\partial x}c_i + 1/2\epsilon^2\Delta t\left(\frac{\partial^2 f_i}{\partial x^2}c_i c_i\right) + O(\Delta t)^2 + O(\epsilon^3) \quad (3.44)$$

where f_i^0 is f_i^{eq}. Expanding $f_i = f_i^0 + \epsilon f_i^1 + \epsilon^2 f_i^2 + \cdots$ and substituting in the above equation, yields,

$$-\frac{1}{\tau}(\epsilon f_i^1 + \epsilon^2 f_i^2 + \cdots) = \epsilon^2\frac{\partial f_i^0}{\partial t} + \epsilon\frac{\partial f_i^0}{\partial x}c_i + \epsilon^2\frac{\partial f_i^1}{\partial x}c_i + \frac{\Delta t}{2}\epsilon^2\frac{\partial f_i^0}{\partial x}c_i c_i + O(\epsilon^3)+)(\Delta t^2) \quad (3.45)$$

Equating term from both sides of the above equation of same order of ϵ:

Terms order of ϵ:

$$-\frac{1}{\tau}f_i^1 = \frac{\partial f_i^0}{\partial x}c_i \quad (3.46)$$

Latter on we may need derivative of f_i^1 with respect of x, hence by taking the derivative of above equation, gives:

$$-\frac{1}{\tau}\frac{\partial f_i^1}{\partial x} = \frac{\partial^2 f_i^0}{\partial x^2}c_i \quad (3.47)$$

Terms order of ϵ^2:

$$-\frac{1}{\tau}f_i^2 = \frac{\partial f_i^0}{\partial t} + \frac{\partial f_i^1}{\partial x}c_i + \frac{\Delta t}{2}\frac{\partial^2 f_i^0}{\partial x^2}c_i c_i \tag{3.48}$$

Substituting the derivative of f_i^1 with respect of x in the above equation, yields,

$$-\frac{1}{\tau}f_i^2 = \frac{\partial f_i^0}{\partial t} - \tau\frac{\partial^2 f_i^0}{\partial x^2}c_i c_i + \frac{\Delta t}{2}\frac{\partial^2 f_i^0}{\partial x^2}c_i c_i \tag{3.49}$$

To recover continuum diffusion equation (1.1), the above equation need to be summed over all states, $i = 1, 2$.

$$\sum_i\left[-\frac{1}{\tau}f_i^2\right] = \sum_i\left[\frac{\partial f_i^0}{\partial t} - \tau\frac{\partial^2 f_i^0}{\partial x^2}c_i c_i + \frac{\Delta t}{2}\frac{\partial^2 f_i^0}{\partial x^2}c_i c_i\right] \tag{3.50}$$

The first term on the left-hand side (LHS) is zero. The first term on the right-hand side (RHS), will be

$$\sum_i \frac{\partial f_i}{\partial t} = \frac{\partial \sum_i f_i}{\partial t} = \frac{\partial \Theta}{\partial t} \tag{3.51}$$

The term of,

$$\sum_i\left[\frac{\partial^2 f_i^0}{\partial x^2}c_i c_i\right] = \frac{\partial^2}{\partial x^2}\left(\sum_i [f_i^0 c_i c_i]\right) = \frac{\partial^2 \Theta}{\partial x^2} \tag{3.52}$$

Hence Eq. 3.50 can be simplified as,

$$0 = \frac{\partial \Theta}{\partial t} - \left(\tau - \frac{\Delta t}{2}\right)\frac{\partial^2 \Theta}{\partial x^2} \tag{3.53}$$

By comparing the above equation with continuum diffusion equation, the relaxation parameter, τ, can be related to the diffusion coefficient as,

$$\Gamma = \tau - \frac{\Delta t}{2} \tag{3.54}$$

Hence, a relationship is established between macro and meso-scales.

3.5.1 Normalizing and Scaling

It is more appropriate to solve specific problems in non-dimensionalized (normalized) forms rather than in dimensional form for a few reasons, the

controlling parameters can be grouped into a few non-dimensional parameters, hence the results can be presented in more general forms. Also, the scaling from continuum mechanics (macro-scale) to LBM (meso-scale) becomes clear. For example, Eq. 3.1 is dimensional form for diffusion equation, α has units of m^2/s, x has unit of m and ϕ has units of °C, or K, or other units depending on what ϕ represents. However, let's assume that Eq. 3.1 represents heat diffusion equation, then ϕ represents temperature and has units of centigrade (°C). We can scale the temperature by $\theta = \frac{\phi - \phi_c}{\phi_h - \phi_c}$, where ϕ_c and ϕ_h are cold and hot temperatures binding the system. The x can be scaled by the length of the system L, i.e., nondimensionalized $x^* = x/L$. The nondimensional heat diffusion equation can be written as,

$$\frac{\partial \theta}{\partial t^*} = \frac{\partial^2 \theta}{\partial x^{*2}} \tag{3.55}$$

where t^* is non-dimensional time ($t^* = \alpha t/L^2$). The equation is independent of $\alpha; \theta$ and x ranges from 0 to 1.0, regardless of the range of the dimensional equation or the physical material of the system. More important, the equation can be solved easily with LBM considering the following scaling. In conventional numerical technique, say finite difference method (FDM) for example, we discretize the space domain into nodes and the distance between two nodes is Δx, and time domain also discretizes into Δt. If the domain descritzed into 100 nodes $(0, 1, 2, \ldots, 99, 100)$, then $\Delta x = 0.01$. Let us set Δt to 0.0001. Note that, Δt and Δx must be properly chosen to ensure stable solution if an explicit method is used. In LBM, the domain need to discretized also, however, the most popular method is to use unity for Δx and for Δt, i.e., $\Delta x = 1$ and $\Delta t = 1$. The question is what will be the number of lattices and time steps for total dimensional time of 5 (for example). We can select any total number of lattices, for example $N = 500$, then the nondimensional distance, x, at any site (i), is $i/500$. For instance site number 50, the x is 50/500=0.1. Time scale need to be worked out. Remember that t^* is equal to $\alpha t/L^2$ which in lattice unit should equal to $T\Delta t/\tau N^2$ where τ is relaxation time in LBM, T is total time and N is the total number of lattice sites. For the given example, we are interested in the results at dimensionless time of $t^* = 2.0$, hence,

$$2.0 = \frac{T\Delta t}{\tau N^2} \tag{3.56}$$

Since we select $N = 500, \Delta t = 1.0$, then have two free variables (τ and T), where T is total time steps in LBM units. If we select $\tau = 0.5$ m then $T = 250,000$. Hence, the calculations should be carried out for 250,000 times to represent the results of 2.0 nondimensional time. For FDM, the number of time steps is 2/0.0001=20,000. In general, LBM needs less computer time than the FDM for the diffusion problems.

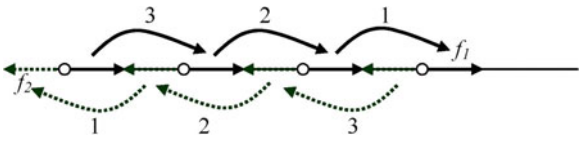

Fig. 3.5 Algorithm for streaming processes

3.5.2 Heat Diffusion in an Infinite Slab Subjected to a Constant Temperature

A slab initially at temperature equal to zero, $T = 0.0$. For time $t \geqslant 0$, the left surface of the slab is subjected to a high temperature and equal to unity, $T = 1.0$. The slab length is 100 units. Calculate the temperature distribution in the slab for $t = 200$. Compare the results of both methods, LBM and FDM, for $\alpha = 0.25$.

Solution Let us divide the domain of integration into $\Delta x = 1.0$ and $\Delta t = 1.0$.

Equation 3.23, can be used to calculate ω : $0.25 = (1/\omega - 0.5)$, then $\omega = 1.333333$.

The temperature is a function of x and t. Equation 3.19 states that

$$T(x,t) = \sum_{k=1}^{2} f_k(x,t) \tag{3.57}$$

and from Eq. 3.20, $f_k^{eq}(x,t) = w_k T(x,t)$, hence $f_1^{eq}(x,t) = 0.5T(x,t)$ and $f_2^{eq}(x,t) = 0.5T(x,t)$.

LBM consists of two steps, collision and streaming.

The collision step is:

$$f_k(x, t + \Delta t) = f_k(x,t)[1 - \omega] + \omega f_k^{eq}(x,t)$$

where $k = 1$ and 2.

The streaming step is:

$$f_k(x + \Delta x, t + \Delta t) = f_k(x, t + \Delta t).$$

In order not to save the pervious values of $f_k(x,t)$, and to avoid over writing updated values, the stream (updating) for $f_1(x,t)$ should be from $i = m$ to $i = 0$, while stream (updating) for $f_2(x,t)$ should be from $i = 0$ to $i = m$, as shown in Fig. 3.5.

As illustrated in Fig. 3.5, f_1 should be swept backward $(m, m-1, \ldots, 4, 3, 2, 1, 0)$ in programming, $i = m,\ 0,\ -1$. While, f_2 should be swept forward $(1, 2, 3\ldots)$ in programming, $i = 0,\ m$. By this procedure the overwriting of the updated values by old values can be avoided.

3.5.3 Boundary Conditions

At $x = 0$, the left-hand boundary condition, the value of f_2 can be obtained from the streaming process, then the unknown is only f_1 at the left boundary. Hence there is a need for an equation to relate f_1 to the known parameters. On the right side, $x = L$, the value of f_2 need to be known, while f_1 can be obtained from streaming process.

The following sections discuss treatment of different boundary conditions.

3.5.3.1 Constant Temperature Boundary Condition, Dirichlet Boundary Condition

The detailed flux balance at the boundary, $x = 0$, for D1Q2 is as follows,

$$f_1^{\mathrm{eq}}(0,t) - f_1(0,t) + f_2^{\mathrm{eq}}(0,t) - f_2(0,t) = 0 \tag{3.58}$$

and,

$$f_1^{\mathrm{eq}}(0,t) = w_1 T_w = 0.5 T_w \quad \text{and} \quad f_2^{\mathrm{eq}}(0,t) = w_2 T_w = 0.5 T_w.$$

Therefore at $x = 0.$, $f_1(0) + f_2(0) = T_w$, and from the streaming processes, $f_2(0) = f_2(1)$, then $f_1(0)$ can be determined as, $f_1(0) = T_w - f_2(0)$. Figure 3.6 shows the comparison of predicted results obtained by LBM and FDM for a slab with a constant temperature boundary condition.

3.5.3.2 Adiabatic Boundary Condition, Zero Flux Condition

The temperature gradient is zero, which implies that $T(m) = T(m-1)$.

Hence,

$$f_1(m) + f_2(m) = f_1(m-1) + f_2(m-1),$$

or

$$f_1(m) = f_1(m-1) \quad \text{and} \quad f_2(m) = f_2(m-1),$$

where m denotes the last lattice node.

3.5.4 Constant Heat Flux Example

Same example as before, except that the left-hand boundary is exposed to a constant heat flux of $100\,\mathrm{W/m^2}$ instead of a constant temperature. The thermal conductivity of the slab is 20 W/mK (Figs. 3.6 and 3.7).

Solution The procedure is same as for previous example, except that we need to determine $f_1(0)$ at $x = 0$.

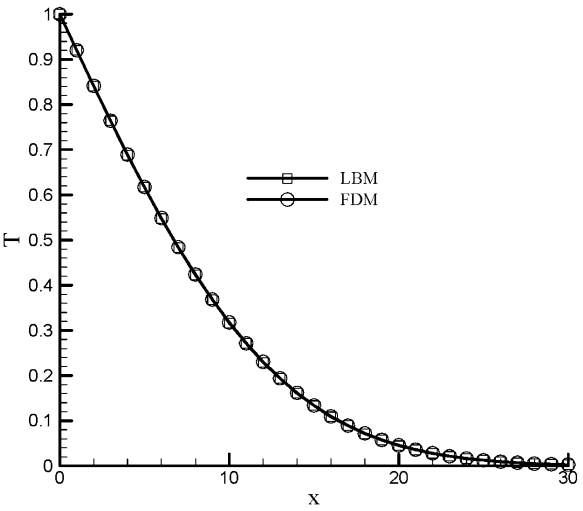

Fig. 3.6 Shows the comparison of predicted results obtained by LBM and FDM, for a slab with a constant temperature problem

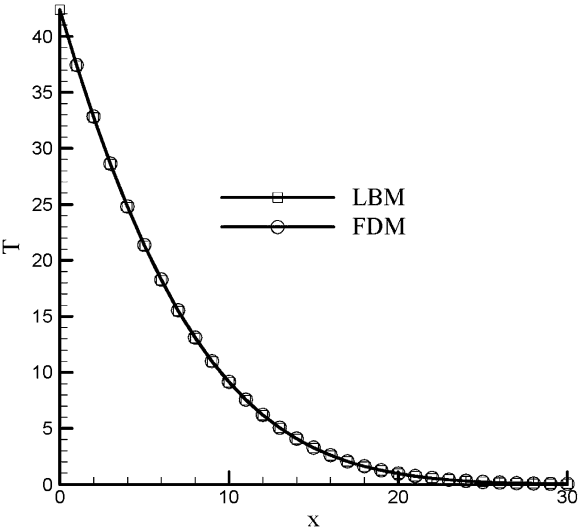

Fig. 3.7 Shows a comparison of results predicted by LBM and FDM for a slab subjected to a constant heat flux at the boundary

Fourier Law states that the heat flux is proportional to the temperature gradient:

$$q'' = -k\frac{\partial T}{\partial x} \tag{3.59}$$

where, k is the thermal conductivity of the material.

The above equation can be approximated at $x = 0$ by finite difference method, as:

$$q'' = -k\frac{T(1) - T(0)}{dx} \tag{3.60}$$

Substituting $T(1) = f_1(1) + f_2(1)$ and $T(0) = f_1(0) + f_2(0)$ in the above equation and solving for $f_1(0)$ gives,

$$f_1(0) = f_1(1) + f_2(1) - f_2(0) + \frac{q'' dx}{k} \tag{3.61}$$

Hence $f_1(0)$ can be calculated from above equation, where $f_2(0)$ can be obtained from streaming step as before. The computer code is given in the Appendix.

Same computer code as discussed before (see Appendix for codes), except the boundary condition should be replaces by:

```
! Boundary condition

f1(0)=f1(1)+f2(1)-f2(0)+flux*dx/tk          ! constant heat flux boundary condition, x=0.

f1(m)=f1(m-1)                               ! adiabatic boundary condition, x=L

f2(m)=f2(m-1)                               ! adiabatic boundary condition, x=L
```

Figure 3.7 shows a comparison of results predicted by LBM and FDM for a slab subjected to a constant heat flux at the boundary.

3.6 Source or Sink Term

The source (sink) term can be manipulated as follows. Let us assume that we have a slab with a uniformly distributed heat source, such as a conductor subjected to a constant electric potential difference (voltage), or a chemical reaction taking place through a plane reactor. Assuming that the heat transfer is one-dimensional, then the differential equation for the such a problem can be written as,

$$\rho C \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) + q_g \tag{3.62}$$

where q_g is the rate of heat generation per unit volume. For a constant thermal conductivity, the equation can be rewritten as,

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} + \frac{q_g}{\rho C} \tag{3.63}$$

In Lattice Boltzmann, the source term can be added to the LB equation as

$$f_k(x, t + \Delta t) = f_k(x, t)[1 - \omega] + \omega f_k^{eq}(x, t) + \Delta t w_k S \tag{3.64}$$

where S is the force or source term, which can be related to q_g as

$$S = \frac{q_g}{\rho C} \tag{3.65}$$

notice that unit of S is unit of temperature (°C) per unit of time. The value of w_k is 0.5 for both w_1 and w_2.

The LBM code can be simply modified, by adding an extra term in calculating f_1 and f_2.

In the LBM codes, the following modifications need to be done,

```
=====================================
      do i=0,m

         rho(i)=f1(i)+f2(i)
              source=qg/(rho(i)*cp)
         feq=0.5*rho(i)
f1(i)=omega*feq+(1.-omega)*f1(i)+dt*0.5*source

                  f2(i)=omega*feq+(1.-omega)*f2(i)+dt*0.5*source
end do
=========================================
```

3.7 Axi-Symmetric Diffusion

One-dimensional radial diffusion in cylindrical or spherical geometries can be expressed as,

$$\frac{\partial T}{\partial t} = \frac{\alpha}{r^k}\frac{\partial}{\partial r}\left(r^k \frac{\partial T}{\partial r}\right) \tag{3.66}$$

where k equal to 1 and 2 for cylindrical and spherical coordinate systems, respectively. The above equation can be expanded as,

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial r^2} + \frac{k\alpha}{r}\frac{\partial T}{\partial r} \tag{3.67}$$

The above equation is similar to diffusion equation in Cartesian coordinate with an extra term, the last term, which can be treated as a source term.

Analytical solution for heat conduction (steady state) in cylindrical shell without heat generation can be written as,

$$\theta = \frac{\ln(r/r_\mathrm{i})}{\ln(r_\mathrm{o}/r_\mathrm{i})} \tag{3.68}$$

where $\theta = \frac{(T-T_\mathrm{i})}{(T_\mathrm{o}-T_\mathrm{i})}$, T_i and T_o are temperature at r_i (inner radius) and r_o (outer radius), respectively.

BGK Lattice Boltzmann equation with source term can be written as Eq. 3.64 with $\omega = \frac{\Delta t}{\alpha+0.5}$. The source term S can be treated either

$$\frac{\alpha n}{r}\frac{T_{j+1}-T_{j-1}}{2\Delta r_j} \tag{3.69}$$

or

$$\frac{\alpha n}{r}\frac{f_{2,j+1}-f_{1,j-1}}{\tau} \tag{3.70}$$

where $\tau = \alpha + 0.5$.

For more details about solving axis-symmetric (cylindrical or spherical with or without heat generation) diffusion problem with LBM, we refer to paper by Mohamad (2009).

3.8 Two-Dimensional Diffusion Equation

The method of solution and equation for two and three dimensional problems are similar to the one dimensional problem. The steps of the solution are exactly the same, except that the number of streaming directions increase and the weighting factors for each direction should be evaluated. In the following section the algorithm of the solution will be discussed.

Three types of lattice arrangements will be discussed, namely D2Q4, D2Q5, and D2Q9. D2Q4 and D2Q5 are simple extension for 1-D problem, while D2Q9 is more involved. The main objective of introducing D2Q9 is for future analysis of fluid flow problems, Navier–Stokes equations. Hence, it aims to build a background for more complicated problems, mainly for fluid flow problems.

3.8.1 D2Q4

The common terminology used in LBM is to refer to the dimension of the problem and the number of speed is using D*n*Q*m*, where *n* represent the dimension of the problem (2 for 2-D and 3 for 3-D) and *m* refers to the speed model, as mentioned before.

Since the diffusion phenomena has no directional preference and only gradient determines the diffusion process, unlike advection process, where the information follows the direction of the flow. Full isotropy can be insured by 90° rotation of the coordinates; hence, it is sufficient to use only four streaming directions, Fig. 3.8.

Note that the numbering of *f* is arbitrary, you can use 1, 2, 3 and 4 instead of 1, 3, 2 and 4.

As usual, the LBM consists of two steps, collisions and streaming.

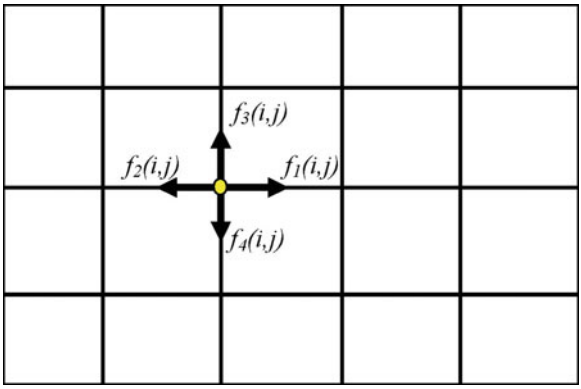

Fig. 3.8 Lattices for 2-D, diffusion problem

The collisions step without forcing function is:

$$f_k(x, y, t + \Delta t) = f_k(x, y, t)[1 - \omega] + \omega f_k^{\text{eq}}(x, y, t) \tag{3.71}$$

$k = 1, 2, 3$ and 4.

The streaming step is:

$$f_k(x + \Delta x, y + \Delta y, t + \Delta t) = f_k(x, y, t + \Delta t) \tag{3.72}$$

$k = 1, 2, 3$ and 4.

In streaming step, for example $f_1(i,j)$ moves to $f_1(i+1,j)$, $f_2(i,j)$ moves to $f_2(i-1,j)$, $f_3(i,j)$ moves to $f_3(i,j+1)$ and $f_4(i,j)$ moves to $f_4(i,j-1)$. $w(1) = w(2) = w(3) = w(4) = 1/4$.

Indeed, nothing new compared with 1-D problem. In programming, one needs to take care of not over writing values of f's, other than that, every step is similar to the 1-D algorithm.

The $w_i = 0.25$ for $i = 1, 2, 3$ and 4. Also,

$$\alpha = \frac{\Delta x^2}{2\Delta t}\left(\frac{1}{\omega} - \frac{1}{2}\right) \tag{3.73}$$

3.8.2 D2Q5

Similar to D2Q4, except that there will be a particle residing at the center of the lattice (f_0). The weight coefficients are, 2/6 for f_0 and 1/6 for f_1, f_2, f_3 and f_4. The diffusion coefficient is,

$$\alpha = \frac{\Delta x^2}{3\Delta t}\left(\frac{1}{\omega} - \frac{1}{2}\right) \tag{3.74}$$

Other than that every step and algorithm is the same as for D2Q4, of course there is no streaming for the particle at the center of the lattice (f_0).

3.9 Boundary Conditions

In the following subsections, different boundary conditions will be discussed in detail.

3.9.1 The Value of the Function is Given at the Boundary

For example, temperature at the left and bottom boundaries are given, i.e., $T = C_1$at $x = 0$ and $T = C_2$ at $y = 0$, (C_1 and C_2 are constants), $f_1 + f_2 + f_3 + f_4 = T$. Then, the unknowns are f_1 and f_3 because f_2 and f_4 can be obtained from streaming step, i.e., $f_2(i,j) = f_2(i+1,j)$ and $f_4(i,j) = f_4(i,j+1) \cdot f_1$ and f_3 can be calculated by using flux conservation equation, i.e., $f_1^{eq} - f_1 + f_2^{eq} - f_2 = 0$, since $w = 0.25$ for all streaming directions and $f_1^{eq} = f_2^{eq} = 0.5C_1$, then $f_1 = 0.25C_1 + 0.25C_1 - f_2$, i.e., $f_1 = 0.5C_1 - f_2$ similarly $f_3 = 0.5C_2 - f_4$. This ensures the flux conservation, which conserve the physics of the problem.

However, if T is given on right boundary =, then f_2 need to be determined, similarly. The above argument is valid for both D2Q4 and D2Q5.

3.9.2 Adiabatic Boundary Conditions, for Instance

$$\frac{\partial T}{\partial x} = 0 \tag{3.75}$$

by using finite difference approach, $\frac{\partial T}{\partial x} = 0$, can be approximated as

$$\frac{T(i+1,j) - T(i,j)}{\Delta x} = 0 \tag{3.76}$$

(first order accurate). It can simplified as $T(i,j) = T(i+1,j)$. Hence,

$$\begin{aligned} f_1(i,j) + f_2(i,j) + f_3(i,j) + f_4(i,j) = {} & f_1(i+1,j) + f_2(i+1,j) \\ & + f_3(i+1,j) + f_4(i+1,j) \end{aligned} \tag{3.77}$$

It is rational to assume that $f_1(i,j) = f_1(i+1,j), f_2(i,j) = f_2(i+1,j)$ and so on.

3.9.3 Constant Flux Boundary Condition

For instance

$$k\frac{\partial T}{\partial x} = q \tag{3.78}$$

The finite difference approximation for the above boundary condition is,

$$k\frac{T(i+1,j) - T(i,j)}{\Delta x} = q \tag{3.79}$$

which can be rewritten as:

$$T(i,j) = T(i+1,j) - \frac{q\Delta x}{k} \tag{3.80}$$

therefore

$$f_1(i,j) = f_1(i+1,j) - \frac{q\Delta x}{k} \tag{3.81}$$

Other boundary conditions can be formulated similarly, i.e., using finite difference approach. It is also possible to use higher order approximations.

3.10 Two-Dimensional Heat Diffusion in a Plate

A two dimensional square slab shown in Fig. 3.9 is subjected to the given boundary conditions. Initially the slab was at zero temperature. For time $\geqslant 0$, the boundary at $x = 0$ is subjected to a high temperature of value 1.0 and other boundaries are kept as before. The length of the domain is 100 units. Determine the temperature distribution in the slab at time = 400 units (s). Compare the results obtained by using LB and FD methods. Thermal diffusivity is 0.25 and $xl = yl = 1.0$. The computer codes can be found in the Appendix.

Note that for stability conditions the time step for FDM is 0.2. There is no such a problem in using a time step of 1.0 with the LBM method. Hence, LBM is much faster and efficient than FDM.

3.10.1 D2Q9

For fluid flow problems (Navier–Stokes equation), where the advection term (nonlinear term) is important, the D2Q4 or D2Q5 model is not sufficient to insure

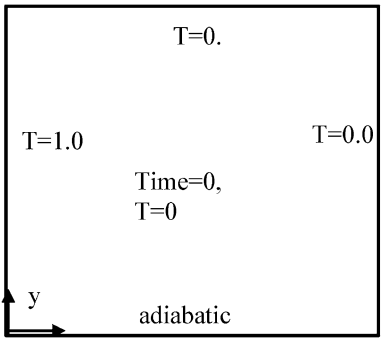

Fig. 3.9 A square (2-D) domain with coordinate system

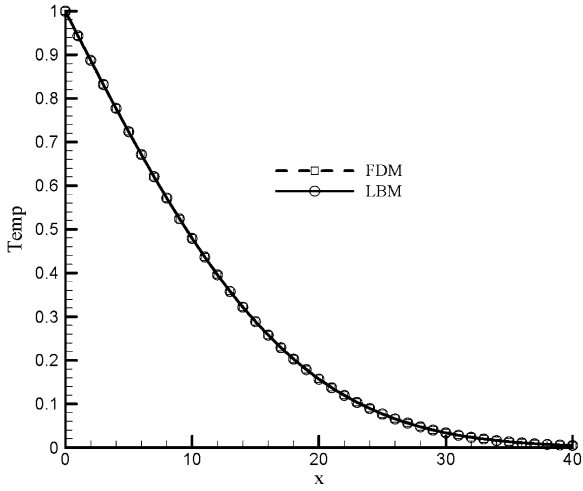

Fig. 3.10 Temperature distribution along the middle line ($y = 0.5$)

the macroscopic isotropy. It is common to use D2Q9. It is illustrative to introduce D2Q9 for a diffusion problem. The knowledge obtained in solving diffusion problem using D2Q9 can be easily extended for advection–diffusion and Navier–Stokes problems.

The macroscopic diffusion coefficient can be related to relaxation factor, ω, as

$$\alpha = \frac{\Delta x^2}{3\Delta t}\left(\frac{1}{\omega} - \frac{1}{2}\right) \tag{3.82}$$

Figures 3.10 and 3.11 compare results obtained by LBM and FDM for the given problem (Fig. 3.9). The domain of interest should be divided into lattices as shown in Fig. 3.12

The Solution Procedure Note that numbering of f is arbitrary. For simplicity of coding, it is more convenient to use numbering of streaming links as shown in Fig. 3.12

As usual, the LBM consists of two steps, collisions and streaming.

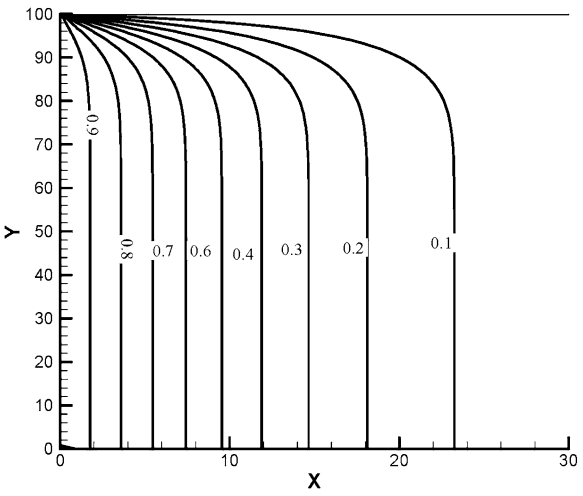

Fig. 3.11 Isotherm comparison between predictions of LBM and FDM, it is difficult to observe the difference between the predictions of both the methods

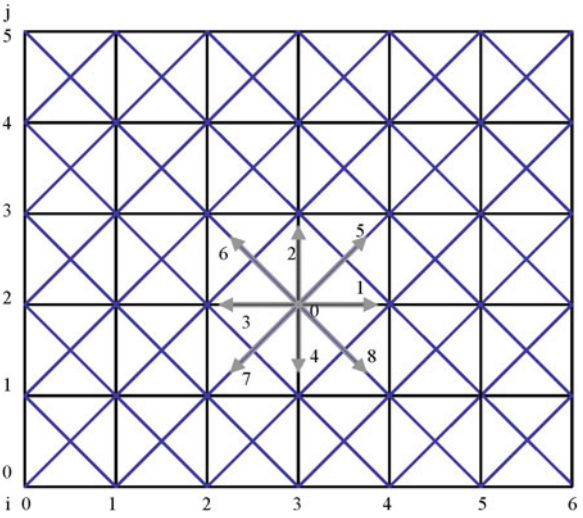

Fig. 3.12 Lattice arrangement for D2Q9

The collisions step without forcing function is:

$$f_k(x, y, t + \Delta t) = f_k(x, y, t)[1 - \omega] + \omega f_k^{\text{eq}}(x, y, t) \tag{3.83}$$

$k = 0, \ldots, 8$.

The streaming step is:

$$f_k(x + \Delta x, y + \Delta y, t + \Delta t) = f_k(x, y, t + \Delta t) \tag{3.84}$$

$k = 1, \ldots, 8$.

In streaming process, for example $f_1(i,j)$ moves to $f_1(i+1,j)$, $f_2(i,j)$ moves to $f_2(i,j+1)$, $f_3(i,j)$ moves to $f_3(i-1,j)$, $f_4(i,j)$ moves to $f_4(i,j-1)$, $f_5(i,j)$

moves to $f_5(i+1,j+1)$, $f_6(i,j)$ moves to $f_6(i-1,j+1)$, $f_7(i,j)$ moves to $f_7(i-1,j-1)$ and $f_8(i,j)$ moves to $f_8(i+1,j-1)$. In programming, attention should be taken to not override new values in the streaming process.

The values of the weighting factors are:

$w(0) = 4/9$

$w(1) = w(2) = w(3) = w(4) = 1/9$ and

$w(5) = w(6) = w(7) = w(8)=1/36$.

The values of the streaming velocity along the lines are: $c_0(0,0), c_1(c,0), c_2(0,c), c_3(-c,0), c_4(0,-c), c_5(c,c), c_6(-c,c), c_7(-c,-c)$ and $c_8(c,-c)$ where $c = \Delta x/\Delta t$.

The initial distribution function $f_i(i,j)$ should be evaluated as,

$$f_k(i,j) = w(k)\phi(i,j) \tag{3.85}$$

The equilibrium function can be calculated using Eq. 3.20, $f_k^{\mathrm{eq}}(i,j) = w(k)\phi(i,j)$ and $\phi(i,j)$ can be calculated as,

$$\phi(x,t) = \sum_{k=1}^{2} f_k(x,t) \tag{3.86}$$

3.10.2 Boundary Conditions

For Dirichlet boundary condition, the flux conservation principle will be used. For instance, the following procedure can be applied to evaluate f_5 at the boundary ($x = 0$), for a given ϕ value (ϕ_{wall}) at the left-hand boundary: $f_5^{\mathrm{eq}} - f_5 + f_7^{\mathrm{eq}} - f_7 = w_5\phi_{\mathrm{wall}} + w_7\phi_{\mathrm{wall}}, f_7$ is known from streaming process, then $f_5 = w_5\phi_{\mathrm{wall}} + w_7\phi_{\mathrm{wall}} - f_7$.

Similarly, the other unknown streaming functions can be evaluated. Adiabatic boundary condition can be treated as discussed before. For details see the computer code in appendix. The computer code is given in appendix for the same problem as discussed before for D2Q5, i.e., heat diffusion in a square plate.

3.10.3 Constant Flux Boundary Conditions

For constant heat flux boundary condition, for instance $x = 0$,

$$k\frac{\partial T}{\partial x} = q. \tag{3.87}$$

The above equation can be approximated using FDMs, which yields,

$$k\frac{T(i+1,j)-T(i,j)}{\Delta x}=q \tag{3.88}$$

At $x = 0$, the equation can be written as

$$k\frac{T(1,j)-T(0,j)}{\Delta x}=q \tag{3.89}$$

which can be rearranged as,

$$T(0,j)=T(1,j)+\frac{q\Delta x}{k} \tag{3.90}$$

Then follow similar procedure as for a constant temperature problem condition with an extra term.

3.11 Problems

1. Extend the LBM for three dimensional problem, solve heat diffusion in a cubic material subjected to a hot vertical wall on one face and other faces kept at constant but cold temperature. Compare finite difference solution with LBM solution at dimensionless time of 40 and after steady state reaches.

2. Solve one dimensional problem with convective boundary condition.

3. Solve diffusion problem for the plate shown in Fig. 3.13, using LBM. The plate left-hand vertical boundary maintained at $T = 1.0$ and right-hand vertical boundary at $T = 0.0$. The other boundaries are assumed to be adiabatic. Plot steady state isotherms and compare it with FDM.

4. In certain applications, the thermal conductivity of the material may be a function of temperature. For instance $\alpha = 0.25 + 0.5T^2, \alpha$, is the thermal diffusivity of the material and T is dimensionless temperature. Solve heat diffusion equation in such a material. The domain of the solution is 2 units in length and initial was at $T = 0$, suddenly the temperature of left-hand boundary is set to

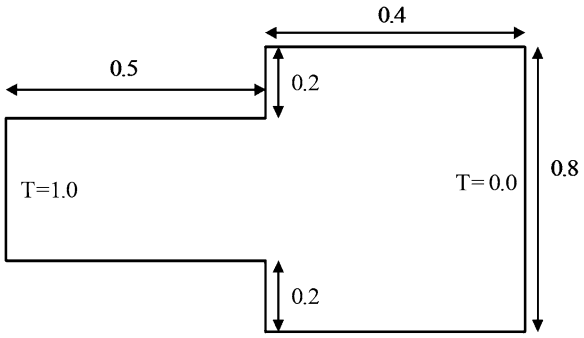

Fig. 3.13 Sketch of the problem 4

$T = 1.0$. Compare results obtained by LBM and FDM at time = 50 and 100. Hint: the relaxation time or relaxation frequency become function of temperature.

5. Anisotropic heat diffusion: heat or mass diffusion in wood, crystal, sedimentary rocks, and fibers have preferable direction. In other words, heat or mass diffusion coefficient is directional dependant. Heat diffusion equation in two dimensional domain can be expressed as,

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right) \tag{3.91}$$

In LBM modelling the relaxation time (or relaxation frequency) is directional dependent. The diffusion coefficients are related to relaxation frequency as,

$$\alpha_x = \frac{k_x}{\rho c} = \frac{\Delta x^2 c_s^2}{\Delta t}\left(\frac{1}{w_x} - 0.5\right) \tag{3.92}$$

and

$$\alpha_y = \frac{k_y}{\rho c} = \frac{\Delta y^2 c_s^2}{\Delta t}\left(\frac{1}{w_y} - 0.5\right) \tag{3.93}$$

c_s equal to $1/\sqrt{2}$ and $1/\sqrt{3}$ for D2Q4 and D2Q5, respectively. In calculating the distribution function that stream in x-direction w_x should be used and w_y should be used in calculating the distribution functions that stream in y-direction. Formulate the problem discussed in Sect. 3.10 for $\alpha_x = 0.5$ and $\alpha_y = 0.1$. Compare LBM and FDM predictions.

Chapter 4
Advection–Diffusion Problems

In this chapter, the physics of advection and advection diffusion will be explained. Lattice Boltzmann method will be discussed for solving different advection–diffusion problems for one and two dimensional cases. Extending the method to three dimension problems is straightforward.

4.1 Advection

The advection is the transport of heat, mass, and momentum by bulk motion of fluid particles. The governing advection equation for one dimensional (1-D) in Cartesian coordinate system can be expressed as,

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = 0. \tag{4.1}$$

where ϕ is a dependent variable (mass, momentum, energy, species, etc.) and u is the advection velocity, fluid flow velocity. The above equation has first order derivative in space and in time; hence it needs only a boundary condition and an initial condition for closing the solution. Equation 4.1 conveys us that the dependent variable, ϕ, advects with u-velocity in space as the time proceeds. The property ϕ is not diffused or distorted, but it just translates with speed u. For instance, if the property ϕ has a given profile as shown in Fig. 4.1 at time, $t = t_0$, then after $t = t_1$, the profile conserves its original shape and only translates (advectes) with a given velocity u, to a different location.

If u is not a function of ϕ, then the above equation is linear and analytical solution can be obtained. For instance if $\phi(0, x) = g(x)$, then the solution is $g(x - ut)$, where $g(x)$ is the profile of the function at $t = 0$.

As an example, if a drop of oil is placed on a water stream, the drop of oil advects without any diffusion. It conserves its shape and density as it moves with the velocity of the water stream.

A. A. Mohamad, *Lattice Boltzmann Method*,
DOI: 10.1007/978-0-85729-455-5_4, © Springer-Verlag London Limited 2011

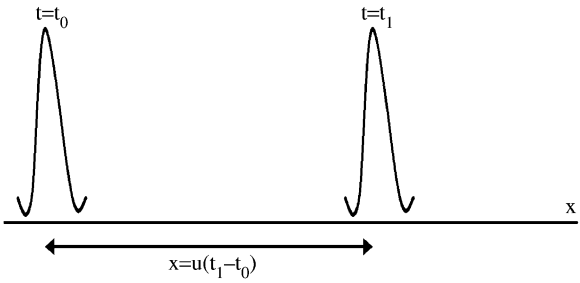

Fig. 4.1 Illustrates advection processes

Three dimensional advection equation can be written as

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} + w\frac{\partial \phi}{\partial z} = 0 \tag{4.2}$$

which is an extension for 1-D equation.

4.2 Advection–Diffusion Equation

The advection–diffusion process is a process where both advection and diffusion take place simultaneously. For instance, if a pollutant or a drop of ink is added to a stream of water, the pollutant or ink concentration decreases (diffuses) as the stream moves away from the source. The phenomenon of advection–diffusion without and with a source or reaction term is very common in the nature and in industrial and engineering applications, usually called transport problem.

The one-dimensional advection–diffusion equation can be expressed in Cartesian coordinate system as,

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = \alpha\frac{\partial^2 \phi}{\partial x^2} \tag{4.3}$$

where α is the diffusion coefficient as discussed in the previous chapter.

Figure 4.2 illustrates the concept of advection–diffusion process. Initial function propagates in space as the time proceeds, at the same time the shape of the function changes (flatten) due to diffusion process.

4.2.1 Finite Difference Method

In numerical methods, advection term may cause problem if not properly approximated. The flow direction affect the information flow, upstream should have effect on the downstream, while effect of the downstream condition on the

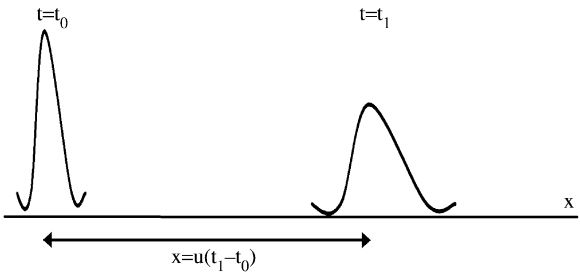

Fig. 4.2 Advection–diffusion processes

upstream condition depends on the strength of the flow (magnitude of flow velocity). In more precious wording, it depends on ratio of advection to diffusion, Peclet number. Hence, using central difference,

$$\left(\frac{\phi(i+1)-\phi(i-1)}{2\Delta x}\right),$$

to approximate the first derivative in space $(\frac{\partial\phi}{\partial x})$ may lead to unstable solution, because we are giving the same weight for both, left and right hand of the function ϕ. In other words, we are not taking into consideration that more information is transferring in the stream-wise direction. In order to take into the consideration the information flow direction, it is more appropriate to use up-wind scheme,

$$\frac{\partial\phi}{\partial x} = \begin{cases} \frac{\phi(i)-\phi(i-1)}{\Delta x} & \text{if } u > 0 \\ \frac{\phi(i+1)-\phi(i)}{\Delta x} & \text{if } u \leq 0 \end{cases} \tag{4.4}$$

Note that Eq. 4.4 is a first order accurate and in general causes artificial diffusion (false diffusion). In other words, the accuracy of the solution depends on the ratio of a numerically introduced diffusion coefficient to the physical diffusion coefficient (α). The remedy for this problem is to use higher order up-wind schemes, which need more nodes to be involved in Eq. 4.4. For more detail, we refer to textbooks on numerical methods for computational fluid dynamics (CFD).

The diffusion term (second order term) can be approximated, as discussed in Chap. 3, i.e., using the central difference scheme.

The code for explicit, finite difference for positive $u = 0.1$ is given in the appendix. Initially, the value of ϕ is zero through the domain and instantly the left-hand side boundary condition for ϕ is set to unity.

For stability reason, Δt and Δx are set to 0.25 and 0.5, respectively. Since u is positive, the first derivative is approximated as it is given in the first line of Eq. 4.4.

Forward difference is used in approximating the time derivative. Again the approximation is first order accurate. Better approximation is possible, such as using Crank–Nikelson method. For details, we refer to numerical method textbooks on the subject.

4.2.2 The Lattice Boltzmann

4.2.2.1 The Lattice Boltzmann Method for Advection–Diffusion Problems

The Lattice Boltzmann equation for advection–diffusion problem is the same as Eq. 3.18,

$$f_k(x + \Delta x, t + \Delta t) = f_k(x, t)[1 - \omega] + \omega f_k^{\text{eq}}(x, t) \tag{4.5}$$

The only difference is an extra term added to the equilibrium distribution function to accommodate the advection effect,

$$f_k^{\text{eq}} = w_k \phi(x, t)\left[1 + \frac{c_k \cdot \vec{u}}{c_s^2}\right] \tag{4.6}$$

where $\vec{u}$ is advection velocity vector. For example for 2-D, the vector $\vec{u} = ui + vj$, where i and j are unit vectors along x and y-direction. And c_k is unit vector along the streaming direction. In general,

$$c_k = \frac{\Delta x}{\Delta t} i + \frac{\Delta y}{\Delta t} j \tag{4.7}$$

For $c_k = 1$, the speed of sound, c_s, is as follows: for D1Q2, D2Q4, and D3Q6,

$$c_s = \frac{1}{\sqrt{2}} \tag{4.8}$$

for D1Q3, D2Q5, D2Q9, and D3Q15,

$$c_s = \frac{1}{\sqrt{3}} \tag{4.9}$$

Same procedure can be followed for diffusion problem. The LBM code is the same as for diffusion problem (Chap. 3), except that an extra term is need to be added to f^{eq}.

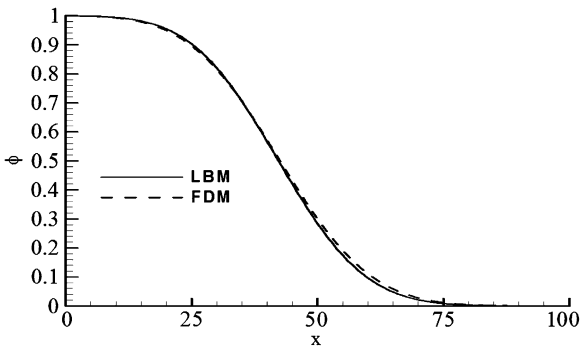

Fig. 4.3 Comparison between predictions of LBM and FDM for advection–diffusion problem

Computer code for 1-D advection–diffusion equation can be found in the appendix for the same problem as we discussed for finite different method.

Figure 4.3 compares the results obtained by using finite difference and LBM method for the mentioned problem. Note that Δx used for FDM is 0.5 compared with $\Delta x = 1.0$ used for LBM to get same results. LBM method is more stable and produce accurate results compared with FDM. FDM needs smaller Δx to produce accurate results, due to numerical errors mentioned before.

4.3 Equilibrium Distribution Function

For advection diffusion problem, it is appropriate to assume that the equilibrium distribution functions can be approximated as a polynomial in velocity. Let,

$$f_i^{\text{eq}} = A_i + B_i c_i \cdot u \tag{4.10}$$

Equilibrium distribution function must satisfy the conservation of mass and momentum,

$$\sum_{i=1}^{2} f_i^{\text{eq}} = \Theta \tag{4.11}$$

and

$$\sum_{i=1}^{2} f_i^{\text{eq}} c_i = \Theta u \tag{4.12}$$

Hence Eq. 4.11 yields,

$$\sum_{i=1}^{2} A_i = A_1 + B_1 u + A_2 - B_2 u = \Theta \tag{4.13}$$

This implies that,

$$A_1 + A_2 = \Theta \tag{4.14}$$

and

$$B_1 u - B_2 u = 0 \quad \text{or} \quad B_1 = B_2 \tag{4.15}$$

Eq. 4.12,

$$\sum_{i=1}^{2} (A_i c_i + B_i c_i c_i u) = u\Theta \tag{4.16}$$

where $c_1 = 1$ and $c_2 = -1$. By comparing coefficients of both sides of Eq. 4.19, we obtain,

$$\sum_{i=1}^{2} A_i c_i = 0 \tag{4.17}$$

Hence, $A_1 = A_2$. In general,

$$A_i = \omega_i \Theta \tag{4.18}$$

where $\omega_i = A_i/\Theta$. Also, from Eq. 4.19,

$$B_i c_i c_i = \Theta \tag{4.19}$$

Multiply both sides of Eq. 4.19 by ω_i and from Chapman–Enskog expansion, the following relation can be obtained,

$$w_i c_i c_i = \frac{1}{c_s^2} \tag{4.20}$$

we can get the following relation,

$$B_i = \frac{w_k}{c_s^2} \Theta \tag{4.21}$$

Hence,

$$f_i^{eq} = w_i \Theta \left(1 + \frac{u c_i}{c_s^2} \right) \tag{4.22}$$

In order to relate LBM to macro scale problem, multi-scale expansion is needed. In the following sections, Chapman–Enskog expansion for advection–diffusion problem will be discussed in detail.

4.4 Chapman–Enskog Expansion

For example, the advection–diffusion equation for energy equation without source term (temperature) in Cartesian coordinate can be written as,

$$\frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} = \Gamma \frac{\partial^2 T}{\partial x^2} \tag{4.23}$$

where $T(x, t)$ is dependant parameter (T stands for temperature for energy equation, for velocity for momentum equation, or for species concentration for species conservation equation). It is assumed that diffusion coefficient, Γ, is a constant. The advection–diffusion equation can be scaled spatially as, x is set to x/ϵ, where ϵ is a small parameter. Introducing the scale to advection diffusion equation yields,

$$\frac{\partial T(x,t)}{\partial t} + u\epsilon \frac{\partial T(x,t)}{\partial x} = \Gamma \epsilon^2 \frac{\partial^2 T(x,t)}{\partial x^2} \tag{4.24}$$

Now, if we scale time t, with $1/\epsilon^2$ as we did in Chap. 3 for diffusion equation, then the first term on the left-hand side of the equation balances the diffusion term on the right-hand side of the equation, hence the effect of advection term will be neglected. If we scale time t, with $1/\epsilon$, in this case there will be a balance between the first and second terms on the left-hand side of the equation, and the effect of diffusion term will be neglected. To resolve this issue,we need to introduce dual time scales, i.e.,

$$\frac{\partial}{\partial t} = \epsilon \frac{\partial}{\partial t_1} + \epsilon^2 \frac{\partial}{\partial t_2} \tag{4.25}$$

Hence, the first term in the above equation balances the advection term and the second term balances the diffusion term. The problem is resolved.

For one-dimensional problem with two lattice velocity components ($c_i, i = 1, 2$), the temperature can be expressed as,

$$T(x,t) = \sum_{i=1}^{i=2} f_i(x,t) = f_1(x,t) + f_2(x,t) \tag{4.26}$$

From now and on, we write f_i instead of $f_i(x,t)$ for simplicity. The equilibrium distribution function can be expressed as,

$$f_i^{\text{eq}} = w_i T(x,y)(1 + Ac_i \cdot u) \tag{4.27}$$

where A is a constant. The second term on the right-hand side accounts for advection effect. By taking summation of both sides of the above equation,

$$\sum_i f_i^{\text{eq}} = T = w_1 T + w_2 T \tag{4.28}$$

Notice that the term $Ac_i \cdot u$ cancels in the summation process because $c_1 = 1$ and $c_2 = -1$.

The weight factors must satisfy the relation

$$\sum_{i=1}^{i=2} w_i = 1 \tag{4.29}$$

i.e., $w_1 + w_2 = 1$, which yields, $w_1 = w_2 = 1/2$.

The distribution function can be expanded in terms of a small parameter, ϵ as

$$f_i(x,t) = f_i^0 + \epsilon f_i^1 + \epsilon^2 f_i^2 + \cdots \tag{4.30}$$

where f_i^0 is distribution function at the equilibrium conditions, equal to f_i^{eq} By summing above equation, it leads to,

$$\sum_{i=1}^{i=2} f_i(x,t) = \sum_{i=1}^{i=2} [f_i^0 + \epsilon f_i^1 + \epsilon^2 f_i^2 + \cdots] \tag{4.31}$$

Since, $f_1 + f_2 = T(x,t)$ and $f_1^0 + f_2^0 = T(x,t)$, hence other expanded term in the above equation should be zero, i.e,

$$\sum_{i=1}^{i=2} f_i^1 = 0 \tag{4.32}$$

$$\sum_{i=1}^{i=2} f_i^2 = 0 \tag{4.33}$$

The updated distribution function $f_i(x + c_i\Delta t, t + \Delta t)$ is expanded using Taylor series,

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) + \frac{\partial f_i}{\partial t}\Delta t + \frac{\partial f_i}{\partial x} c_i \Delta t + 1/2\Delta t^2 \left(\frac{\partial^2 f_i}{\partial t^2} + 2\frac{\partial^2 f_i}{\partial t \partial x} c_i + \frac{\partial^2 f_i}{\partial x^2} c_i c_i \right) + O(\Delta t)^3 \tag{4.34}$$

Introducing scaling for the above equation, i.e., $\frac{\partial}{\partial t}$ is replaced by $\epsilon \frac{\partial}{\partial t_1} + \epsilon^2 \frac{\partial}{\partial t_2}$ and $\frac{1}{\partial x}$ is replaced by $\epsilon \frac{1}{\partial x}$, by keeping terms with ϵ^2, yields:

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) + \epsilon \frac{\partial f_i}{\partial t_1}\Delta t + \epsilon^2 \frac{\partial f_i}{\partial t_2}\Delta t + \epsilon \frac{\partial f_i}{\partial x} c_i \Delta t + \frac{\Delta t^2}{2} \left(\frac{\epsilon^2 \partial^2 f_i}{\partial t_1^2} + 2\frac{\epsilon^2 \partial^2 f_i}{\partial t_1 \partial x} c_i + \frac{\epsilon^2 \partial^2 f_i}{\partial x^2} c_i c_i \right) + O(\Delta t)^3 \tag{4.35}$$

Lattice Boltzmann equation can be written as,

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) + \frac{\Delta t}{\tau} [f_i^0(x,t) - f_i(x,t)] \tag{4.36}$$

Substituting Eq. 4.35 into above equation and retaining terms up to order of ϵ^2, yields:

$$\frac{1}{\tau}[f_i^0(x,t) - f_i(x,t)] = \epsilon \frac{\partial f_i}{\partial t_1} + \epsilon^2 \frac{\partial f_i}{\partial t_2} + \epsilon \frac{\partial f_i}{\partial x} c_i + \frac{\epsilon^2 \Delta t}{2} \left(\frac{\partial^2 f_i}{\partial t_1^2} + 2\frac{\partial^2 f_i}{\partial x \partial t_1} + \frac{\partial^2 f_i}{\partial x^2} c_i c_i \right) + O(\Delta t)^2 + O(\epsilon^3) \tag{4.37}$$

where f_i^0 is f_i^{eq}. Expanding $f_i = f_i^0 + \epsilon f_i^1 + \epsilon^2 f_i^2 + \cdots$ and substituting in the above equation, yields,

$$-\frac{1}{\tau}\left(\epsilon f_i^1+\epsilon^2 f_i^2+\cdots\right)=\epsilon\frac{\partial f_i^0}{\partial t_1}+\epsilon^2\frac{\partial f_i^1}{\partial t_1}+\epsilon^2\frac{\partial f_i^0}{\partial t_2}+\epsilon\frac{\partial f_i^0}{\partial x}c_i+\epsilon^2\frac{\partial f_i^1}{\partial x}c_i + \frac{\Delta t\epsilon^2}{2}\left[\frac{\partial^2 f_i^0}{\partial t_1^2}+2\frac{\partial^2 f_i^0}{\partial x\partial t_1}c_i+\frac{\partial f_i^0}{\partial x}c_i c_i\right]+O(\epsilon^3)+)(\Delta t^2) \tag{4.38}$$

Collecting the terms of order of ϵ from both sides of the above equations:

$$-\frac{1}{\tau}f_i^1=\frac{\partial f_i^0}{\partial t_1}+\frac{\partial f_i^0}{\partial x}c_i \tag{4.39}$$

Latter on we may need derivative of f_i^1 with respect to t_1 and x, hence by taking the derivative of above equation, respectively:

$$-\frac{1}{\tau}\frac{\partial f_i^1}{\partial t_1}=\frac{\partial^2 f_i^0}{\partial t_1^2}+\frac{\partial^2 f_i^0}{\partial t_1\partial x}c_i \tag{4.40}$$

and

$$-\frac{1}{\tau}\frac{\partial f_i^1}{\partial x}=\frac{\partial^2 f_i^0}{\partial t_1\partial x}+\frac{\partial^2 f_i^0}{\partial x^2}c_i \tag{4.41}$$

Multiply Eq. 4.41 by c_i, add it to Eq. 4.40 and rearrange the summated equation,

$$-\frac{1}{\tau}\left(\frac{\partial f_i^1}{\partial t_1}+\frac{\partial f_i^1}{\partial x}c_i\right)=\frac{\partial^2 f_i^0}{\partial t_1^2}+2\frac{\partial^2 f_i^0}{\partial t_1\partial x}c_i+\frac{\partial^2 f_i^0}{\partial x^2}c_i c_i \tag{4.42}$$

By integrating Eq. 4.40 and taking the summing over all values of i ($i = 1, 2$) we obtain:

$$-\frac{1}{\tau}\sum_{i=1}^{2}f_i^1=\sum_{i=1}^{2}\frac{\partial f_i^0}{\partial t_1}+\sum_{i=1}^{2}\frac{\partial f_i^0}{\partial x}c_i \tag{4.43}$$

The term on the left-hand side diminishes (equal to zero). The summation of f_i^0 is equal to T. Also, $\sum_{i=1}^{2}f_i^0 c_i = uT$. Hence the above equation, reformulated as,

$$0=\frac{\partial T}{\partial t_1}+\frac{\partial(uT)}{\partial x} \tag{4.44}$$

which is advection equation. Notice that t_1 balance advection term.

Now let us return to Eq. 4.38 and collect order terms of ϵ^2 :

$$-\frac{1}{\tau}f_i^2=\frac{\partial f_i^1}{\partial t_1}+\frac{\partial f_i^0}{\partial t_2}+\frac{\partial f_i^1}{\partial x}c_i+\frac{\Delta t}{2}\left[\frac{\partial^2 f_i^0}{\partial t_1^2}+2\frac{\partial^2 f_i^0}{\partial x\partial t_1}c_i+\frac{\partial^2 f_i^0}{\partial x^2}c_i c_i\right] \tag{4.45}$$

Substituting Eq. 4.38 into the above equation, yields:

$$-\frac{1}{\tau}f_i^2 = \frac{\partial f_i^1}{\partial t_1} + \frac{\partial f_i^0}{\partial t_2} + \frac{\partial f_i^1}{\partial x}c_i - \frac{\Delta t}{2\tau}\left[\frac{\partial f_i^1}{\partial t_1} + \frac{\partial f_i^1}{\partial x}c_i\right] \tag{4.46}$$

Collecting similar terms,

$$-\frac{1}{\tau}f_i^2 = \frac{\partial f_i^0}{\partial t_2} + \left(1 - \frac{\Delta t}{2\tau}\right)\left[\frac{\partial f_i^1}{\partial t_1} + \frac{\partial f_i^1}{\partial x}c_i\right] \tag{4.47}$$

Summing the above equation over all states, $i = 1, 2$,

$$\sum_i\left[-\frac{1}{\tau}f_i^2\right] = \sum_{i=1}^{2}\left[\frac{\partial f_i^0}{\partial t_2} + \left(1 - \frac{\Delta t}{2\tau}\right)\right]\left[\sum_{i=1}^{2}\frac{\partial f_i^1}{\partial t_1} + \sum_{i=1}^{2}\frac{\partial f_i^1}{\partial x}c_i\right] \tag{4.48}$$

The left-hand side (LHS) is zero. The first term on the right-hand side (RHS), will be

$$\sum_{i=1}^{2}\frac{\partial f_i^0}{\partial t_2} = \frac{\partial \sum_{i=1}^{2} f_i^0}{\partial t_2} = \frac{\partial \Theta}{\partial t_2} \tag{4.49}$$

The term of,

$$\sum_i\left[\frac{\partial^2 f_i^0}{\partial x^2}c_i c_i\right] = \frac{\partial^2}{\partial x^2}\left(\sum_i [f_i^0 c_i c_i]\right) = \frac{\partial^2 \Theta}{\partial x^2} \tag{4.50}$$

Hence Eq. 4.48 can be simplified as,

$$0 = \frac{\partial \Theta}{\partial t} - \left(\tau - \frac{\Delta t}{2}\right)\frac{\partial^2 \Theta}{\partial x^2} \tag{4.51}$$

By comparing above equation with continuum diffusion equation, the relaxation parameter, τ, can be related to the diffusion coefficient as,

$$\Gamma = \tau - \frac{\Delta t}{2} \tag{4.52}$$

4.4.1 Two-Dimensional Advection–Diffusion Problems

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} = \alpha\left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right) \tag{4.53}$$

The finite difference approximation of the above equation is straightforward for a given velocity field. However, care should be taken if the velocity direction is changing.

First order accurate up-wind scheme can be written as,

$$\frac{\partial \phi}{\partial x} = \begin{cases} \frac{\phi(i,j)-\phi(i-1,j)}{\Delta x} & \text{if } u > 0 \\ \frac{\phi(i+1,j)-\phi(i,j)}{\Delta x} & \text{if } u \leq 0 \end{cases} \tag{4.54}$$

and

$$\frac{\partial \phi}{\partial y} = \begin{cases} \frac{\phi(i,j)-\phi(i,j-1)}{\Delta y} & \text{if } v > 0 \\ \frac{\phi(i,j+1)-\phi(i,j)}{\Delta y} & \text{if } v \leq 0 \end{cases} \tag{4.55}$$

As an example, let us consider flow in a square porous domain shown in Fig. 4.4. Initially the domain is at zero temperature. The upper boundary of the domain is kept at zero temperature, while the bottom and vertical right boundaries are assumed to be adiabatic. The left vertical wall is subjected to unity temperature at time $= 0$. Find the temperature distribution in the domain at time $= 200$, for material of diffusivity of 1.0. The velocity field of $u = 0.1$ and $v = 0.2$ is kept constant in the domain.

For finite difference it is sufficient to use rectangular grids of $\Delta x = 1.0$ and $\Delta y = 1.0$.

To ensure stable solution $\Delta t = 0.2$, for FDM. The computer code is given in Appendix.

4.5 Two-Dimensional Lattice Boltzmann Method

In the following sections, two approaches will be discussed, namely D2Q4 and D2Q9.

4.5.1 D2Q4

The lattice arrangement for D2Q4 is shown in Fig. 4.5. The method is similar to LBM introduced to solve diffusion equation except that the equilibrium term, as mentioned before.

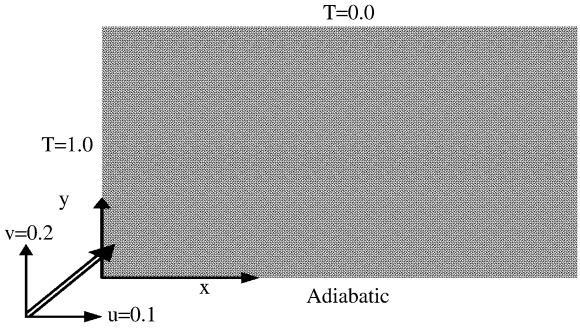

Fig. 4.4 Flow and heat transfer in a porous medium

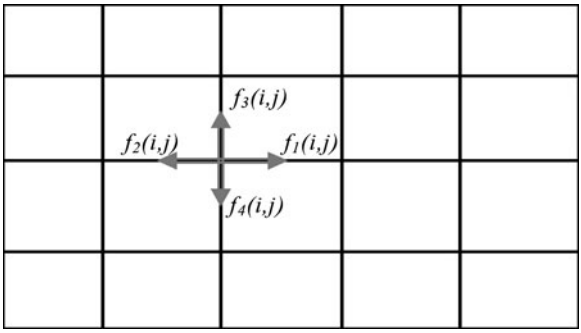

Fig. 4.5 Lattice arrangement for D2Q4

The Lattice BGK equation,

$$f_k(x + \Delta x, t + \Delta t) = f_k(x,t)[1 - \omega] + \omega f_k{}^{\text{eq}}(x,t) \tag{4.56}$$

The k takes from 1 to 4.

The streaming and collision processes are similar to what is explained in previous chapter, Chap. 3.

The equilibrium part can be calculated using Eq.4.6,

$$f_k^{\text{eq}} = w_k \phi(x,t)\left[1 + \frac{c_k \cdot \vec{u}}{c_s^2}\right] \tag{4.57}$$

where $w_k = 0.25$ for $k = 1, 4$

For $\Delta x = \Delta y = \Delta t = 1.0$

$$\frac{c_k \cdot \vec{u}}{c_s} = \begin{cases} 2u & k = 1 \\ -2u & k = 2 \\ 2v & k = 3 \\ -2v & k = 4 \end{cases} \tag{4.58}$$

Notice that c_k is unit vector along the streaming line and

$$c_s = \frac{c_k}{\sqrt{2}} \tag{4.59}$$

for D2Q4.

The computer code (see Appendix) is clear and there is no need to elaborate more on the streaming and collision processes evaluations because it is similar to procedure discussed in Chap. 3. Also, setting boundary conditions is same as in Sect. 3.6.4. Figure 4.6 shows the temperature profile at the mid-plane $y = 0.5$ predicted by LBM and FDM.

4.5.2 D2Q9

The lattice arrangement for D2Q9 is shown below Fig. 4.7 which is the same as in Fig. 3.12.

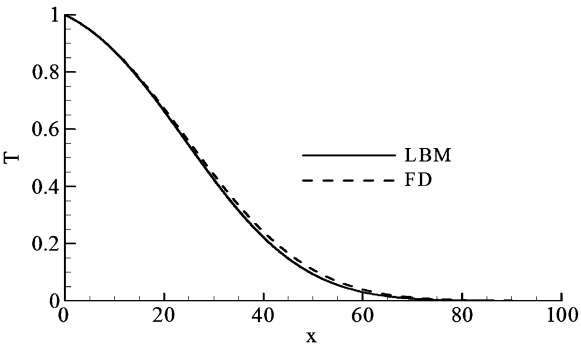

Fig. 4.6 Temperature profile at the mid-plane ($y = 0.5$) predicted by LBM and FDM

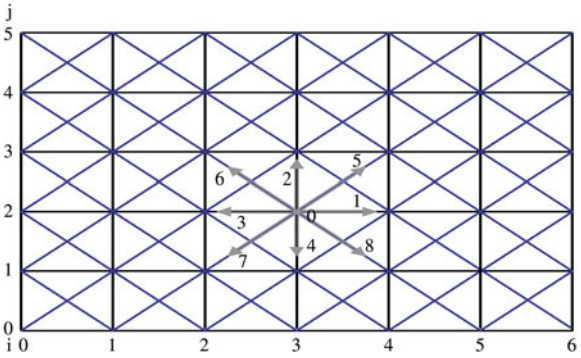

Fig. 4.7 Lattice arrangement for D2Q9

The procedure is the same as discussed in Sect. 3.4.3, except the f^{eq} should account for the advection term,

$$f_k^{eq} = w_k \phi(x,t)\left[1 + \frac{c_k \cdot \vec{u}}{c_s^2}\right] \tag{4.60}$$

Hence,

$$\begin{aligned} f_1^{eq} &= w_1 \phi(1.0 + 3.0u) \\ f_2^{eq} &= w_2 \phi(1.0 + 3.0v) \\ f_3^{eq} &= w_3 \phi(1.0 - 3.0u) \\ f_4^{eq} &= w_4 \phi(1.0 - 3.0v) \\ f_5^{eq} &= w_5 \phi(1.0 + 3.0(u+v)) \end{aligned}$$

etc.

For details of implementation and coding, see the code for D2Q9 in the appendix. Notice that in programming, $f_1, f_2, f_3, \ldots$ used with subscripts for two reasons, compact programming and the code can be easily extended to 3-D. The code simulates heat transfer as in previous problem, with $v = 0.4$ and $u = 0.1$. The conditions on all boundaries are kept at zero temperature, except the left side is suddenly changed to unity. The thermal diffusivity of the medium is set to 1.0. The simulation is carried for 400 time steps. The code is given in Appendix.

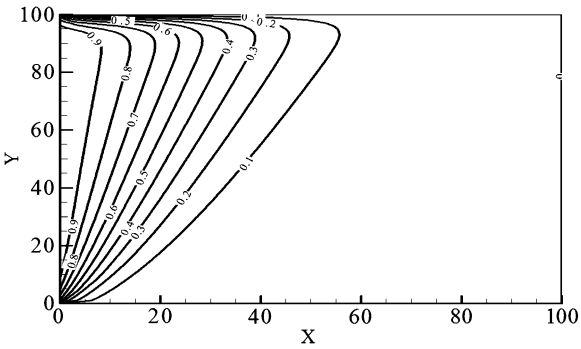

Fig. 4.8 Isotherms at time = 400

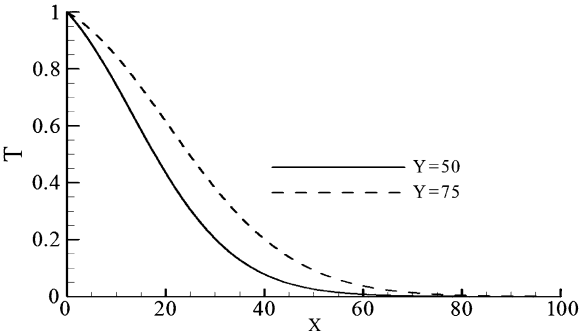

Fig. 4.9 Temperature profiles at the mid-height ($Y = 50$) and at the third quarter height ($Y = 75$) for time = 400

Figure 4.8 shows the isotherm plot at time = 400. Figure 4.9 shows the temperature profiles at mid-height ($Y = 50$) and at height $Y = 75$, for time = 400.

Note that, the scale of the dimension of the problem can be non-dimensionalized or scaled after the calculation. For instance, the height and length of the domain can be divided to the total height of the domain, velocity field can be divided to the inlet velocity, and so on.

At this stage the reader should feel comfortable to solve advection–diffusion problems with and without source terms. The source or sink term can be added as discussed in Chap. 3.

4.6 Problems

4.6.1 Combustion in Porous Layer

Porous burner, or combustion in porous media can be model as a two separate energy equations, one for solid phase and other for gaseous phase. Assume a premixed gaseous fuel with air flows through a porous layer, as shown in Fig. 4.10 combustion takes place inside the layer. For simplicity, let us model the combustion as a constant heat source term released at certain location in the porous medium.

The following model equations need to be solved;

Energy equation for gaseous phase:

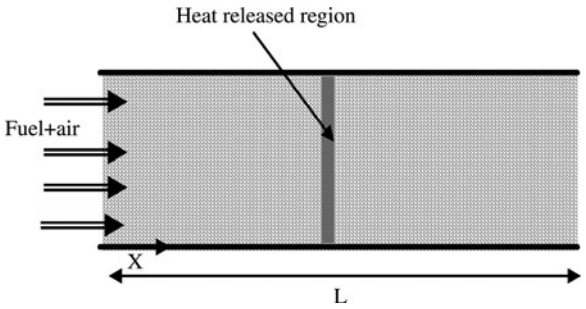

Fig. 4.10 Schematic diagram of porous burner

$$\frac{\partial(\rho_g c_g T_g)}{\partial t} + u\frac{\partial(\rho_g c_g T_g)}{\partial x} = \frac{\partial}{\partial x}\left(k_g \frac{\partial T_g}{\partial x}\right) - h_v(T_g - T_s) + S \tag{4.61}$$

Energy equation for solid phase:

$$\frac{\partial(\rho_s c_s T_s)}{\partial t} = \frac{\partial}{\partial x}\left(k_s \frac{\partial T_s}{\partial x}\right) + h_v(T_g - T_s) \tag{4.62}$$

where ρ, c, T, and h_v are density, specific heat, temperature, and volumetric heat transfer coefficient, respectively. The subscript g and s stand for gas and solid phases, respectively. The source term, S, has value only at the location of heat release and zero at other locations.

The boundary condition:

At inlet, $x = 0$, for simplicity assume that $T_g = T_s = T_{in}$ and at the outlet, $x = L$, assume that the gradient of the temperatures are zero. Modify above code to simulate a flow in a porous layer.

Hint use two distribution functions, one for gas temperature and other for solid temperature.

4.6.2 Cooling a Heated Plate

1. Fully developed flow between parallel plates can be assumed as $u/u_{in} = 1.5(1 - y/H)$. A cold flow with temperature of 20°C enters the channel. The bottom of the channel maintained at $T = 80$°C, while the top plate is assumed to be perfectly insulated. Calculate the temperature field between the plates and rate of heat transfer from the bottom plate (Fig. 4.11).

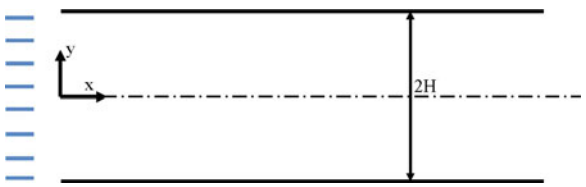

Fig. 4.11 Forced convection problem diagram

4.6.3 Coupled Equations with Source Term

In this section we consider advection–diffusion equation with a source term, which arises in modeling Soret effect. The non-dimensional energy and species conservation equations can be written as,

$$\frac{\partial \Theta}{\partial t} + u\frac{\partial \Theta}{\partial x} = \frac{1}{Pr}\frac{\partial^2 \Theta}{\partial x^2} \tag{4.63}$$

and

$$\frac{\partial \Phi}{\partial t} + u\frac{\partial \Phi}{\partial x} = \frac{1}{PrSc}\left[\frac{\partial^2 \Phi}{\partial x^2} - \frac{\partial^2 \Theta}{\partial x^2}\right] \tag{4.64}$$

respectively. The source term is the second term on the right-hand side of Eq. 4.64 ($\frac{1}{PrSc}\frac{\partial^2 \Theta}{\partial x^2}$) is the source (or sink) term need to be modelled correctly. Hence to solve this kind of problem, it is necessary to use two distribution functions, one for energy equation and another for species conservation equation, such as f_i and g_i for temperature and species concentration, respectively. The source term can be added to the species equation as,

$$S_i = w_i \, \Delta t \, S \tag{4.65}$$

where $S = -\frac{1}{PrSc}\frac{\partial^2 \Theta}{\partial x^2}$ and w_i is the weighting coefficient. The second derivative can be approximated by using second order central difference. Use D1Q2 and D1Q3 to solve the above problem for $Pr = 1.2$ and $Sc = 2.5$ for a given velocity $u = 0.5$. The flow initially was at zero temperature and zero species concentration. Then, suddenly a hot and polluted fluid with $\Theta = 1.0$ and $\Phi = 1.0$ forced to the domain. Compare results of D1Q2 and D1Q3. You will notice that prediction of D1Q2 may be oscillatory, which is physically incorrect. We found that D1Q2 scheme may produce oscillatory solution with source term, which is not the case for D1Q3. Therefore, it is recommended to use D1Q3 for 1-D and D2Q5 for 2-D problems.

Chapter 5
Isothermal Incompressible Fluid Flow

In the following chapter, fluid flow problems will be explained. Lattice Boltzmann method (LBM) will be discussed to solve different isothermal, two dimensional fluid flow problems. Implementation of different boundary and flow conditions will be detailed. Extending the method for three dimensional problems is straightforward.

5.1 Navier–Stokes Equation

In continuum domain, fluid flow is governed by Navier–Stokes equations, NS (continuity and momentum equation).

For incompressible flow, two-dimensional, the conservative form for NS equation can be written in Cartesian coordinate system, without body force, as,

x-momentum

$$\frac{\partial \rho u}{\partial t}+\frac{\partial(\rho u u)}{\partial x}+\frac{\partial(\rho v u)}{\partial y}=-\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu \frac{\partial u}{\partial y}\right) \tag{5.1}$$

y-momentum

$$\frac{\partial \rho v}{\partial t}+\frac{\partial(\rho u v)}{\partial x}+\frac{\partial(\rho v v)}{\partial y}=-\frac{\partial p}{\partial y}+\frac{\partial}{\partial x}\left(\mu \frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu \frac{\partial v}{\partial y}\right) \tag{5.2}$$

The left-hand side represents the advection term, or total acceleration of fluid parcels. The first term on the right-hand side is the pressure gradient term. The last two terms on the right-hand side represent the shear force due to viscous effect.

These two equations have three unknowns, u, v, and p. However, the continuity equation must be satisfied,

$$\frac{\partial(\rho u)}{\partial x}+\frac{\partial(\rho v)}{\partial y}=0. \tag{5.3}$$

A. A. Mohamad, *Lattice Boltzmann Method*,
DOI: 10.1007/978-0-85729-455-5_5, © Springer-Verlag London Limited 2011

Mathematically the problem is closed for given boundary and initial conditions. However there is no explicit equation for pressure. This is one of the difficulties arising in solving incompressible NS. Different techniques are available to solve NS equations, such as finite difference, finite volume, finite element, etc. The main problems in solving fluid flow problem using NS are two folds. First, the advection term is nonlinear, which needs special treatment to avoid stability and numerical diffusion problems. To overcome this problem, second or higher order up-wind schemes have been used to approximate advection terms. Higher order schemes reduce the numerical diffusion problems, however, it decreases the stability limit or adds dispersion effect. In order to overcome these problems, differed correction methods have been used. The main idea is to use first order up-wind scheme and added extra terms as a source. Second, for incompressible flow, there is no explicit equation for pressure, as mentioned before. Hence at each time step in up dating process, the continuity equation need to be satisfied. This step requires solution of Laplace equation (pressure and pressure correction equations). Solving Laplace equation consumes significant computer time. In fact, the major time needed to solve NS is spent in solving Laplace equation.

For solving NS equation we refer to textbook on computational fluid dynamics, see the reference section. For two dimensional problems, it is possible to introduce stream function, and vorticity–stream function (VSF) method can be easily formulated. However, treatment of the boundary conditions is not straightforward.

5.2 Lattice Boltzmann

Lattice Boltzmann equation is linear, in fact the nonlinearity is imbedded (implicitly) in the left-hand side to LBE. The nonlinear advection term in macroscopic approach is replaced by linear streaming process in LBM, similar to characteristic methods for solving compressible flows. There is no need to solve Poisson equation at each time step, as in macroscopic approach to satisfy continuity equation, which drastically reduces the computational time. LBM can be considered an explicit method, where there is no need to solve simultaneous equations, every time step. Since, the collision and streaming processes are local, therefore the LBM method can be easily used in parallel processor machines.

5.2.1 The BGK Approximation

Boltzmann equation is an integro-differential equation. One of the problems in solving the Boltzmann equation is the complicated nature of the collision integral. The main practice to overcome the difficulty of BE is to use BGK (Bhatnagar–Gross–Krook) approximation, as discussed before. Lattice Boltzmann simulates incompressible flow under low Mach number ($Ma = u/c_s$, where is u is the

macroscopic flow velocity and c_s is the speed of sound) condition and weak variation in density. Hence, in simulation, the characteristic flow velocity must be set to small values (order of 0.1) in order to reduce error of simulation. Therefore, in simulating high Reynolds number (uN/v), either number of lattices N, should be increased or kinematics viscosity, v, should be decreased. Care should be taken in using very small values of v, which may cause stability problem.

5.2.1.1 D2Q9, BKG–LBM

In the following paragraphs D2Q9, BKG–LBM will be discussed. The BKG–LBM is the same as introduced before. The lattice Boltzmann can be written as,

$$f_k(x + \Delta x, t + \Delta t) = f_k(x, t)[1 - \omega] + \omega f_k^{\text{eq}}(x, t) \tag{5.4}$$

The only difference is extra terms added to the equilibrium distribution term,

$$f_k^{\text{eq}} = w_k \rho(x, t)\left[1 + \frac{c_k \cdot \mathbf{u}}{c_s^2} + \frac{1}{2}\frac{(c_k \cdot \mathbf{u})^2}{c_s^4} - \frac{1}{2}\frac{\mathbf{u}^2}{c_s^2}\right] \tag{5.5}$$

where

$$c_{\text{s}} = \frac{c_k}{\sqrt{3}},$$

$$c_k = \frac{\Delta x}{\Delta t} i + \frac{\Delta y}{\Delta t} j,$$

and

$$\mathbf{u} = ui + vj \tag{5.6}$$

Note that the above equation is same as Eq. 4.6, with extra terms, which are necessary for momentum conservation equation, nonlinearity in velocity. Also, it is necessary to use D2Q9 arrangement to conserve isotropy, Galilean invariance, and velocity independent pressure of the incompressible NS equation.

The parameters that control incompressible flow are Reynolds number and geometrical constrain of the problem. This can be deduced from the NS equations by using characteristic length and velocity scales. Hence, for any incompressible flow, we need to match Reynolds number and geometric similarity.

5.2.1.2 Mach and Reynolds Numbers

Error analysis of LBM needs extensive multi-scale expansion. However, in this section the detailed analysis is omitted and only practical applications are emphasized.

Analysis shows that LBM resembles Navier–Stokes equations for incompressible flow for low Mach number. The error in LBM is order of Ma^2.

The fluid viscosity is related to the relaxation frequency as,

$$v = \frac{\Delta x^2}{3\Delta t}(\omega - 0.5) \tag{5.7}$$

Reynolds number, $Re = UL/v$, where U and L are characteristic velocity and characteristic length in macro-scale, respectively.

Equation 5.7 can be manipulated (by dividing both sides of the equation to UL), and yields,

$$Ma = \frac{\Delta x}{L\sqrt{3}}(\omega - 0.5)Re \tag{5.8}$$

The value of $L/\Delta x$ is the number of lattices in the direction of the characteristic length, say N. For Δx of unity as a normal practice in LBM, $L = N$, hence $Re = UN/v$ (let us call this Reynolds number, a lattice Reynolds number). For accurate solution, Ma, should be kept small, therefore, ω or N should be chosen to insure low Ma value. In general, U should be in the order of 0.1 or 0.2, which is not related to the macroscopic velocity. In practical applications, the macroscopic Reynolds number should equal the LBM Reynolds number. Then, U and v can be arbitrarily selected within the range that insures the stability of the solution. The following sections, examples illustrate this issue more clearly.

5.2.1.3 Two Dimensional with Nine Velocities, D2Q9

There is no need to elaborate on D2Q9 in detail, because it is already discussed in Chaps. 3 and 4. Hence, D2Q9 applications for fluid flow will be as layout in the following sections.

Figure 5.1 illustrates the streaming direction and values of w and c_k.

The weight coefficients are:

$w_0 = 4/9, w_1 = w_2 = w_3 = w_4 = 1/9$ and $w_5 = w_6 = w_7 = w_8 = 1/36$.

c_k are unit vectors along the lattice streaming directions,

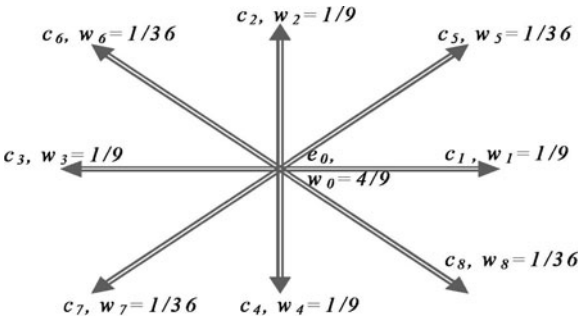

Fig. 5.1 Streaming directions

$c_0 = (0, 0)$, x and y-components are zeros,
$c_1 = (1, 0)$, x-component is 1 and y-component is 0,
$c_2 = (0, 1)$, x-component is 0 and y-component is 1,
$c_3 = (-1, 0)$, x-component is -1 and y-component is 0,
$c_4 = (0, -1)$, x-component is 0 and y-component is -1,
$c_5 = (1, 1)$, x-component is 1 and y-component is 1,
$c_6 = (-1, 1)$, x-component is -1 and y-component is 1,
$c_7 = (-1, -1)$, x-component is -1 and y-component is -1,
$c_8 = (1, -1)$, x-component is 1 and y-component is -1.

For example, if x- and y-components of a vector $\mathbf{u}$ are 3 and -2 ($u = 3$, $v = -2$), respectively, then

$c_1 \cdot \mathbf{u} = (1, 0) \cdot (3, -2) = 3,$
$c_3 \cdot \mathbf{u} = (-1, 0) \cdot (3, -2) = -3,$
$c_5 \cdot \mathbf{u} = (1, 1) \cdot (3, -2) = 3 - 2 = 1$
$c_6 \cdot \mathbf{u} = (-1, 1) \cdot (3, -2) = -3 - 2 = -5$ and so on.

And $\mathbf{u}^2 = \mathbf{u} \cdot \mathbf{u} = (3, -2) \cdot (3, -2) = 9 + 4 = 13.$
Using Eq. 5.5, the values of,

$$\left[1 + \frac{c_k \cdot \mathbf{u}}{c_s^2} + \frac{1}{2}\frac{(c_k \cdot \mathbf{u})^2}{c_s^4} - \frac{1}{2}\frac{\mathbf{u}^2}{c_s^2}\right]$$

can be written for $k = 0$–8 as,

$$1 - \frac{3}{2}(u^2 + v^2) \quad k = 0$$
$$1 + 3u + \frac{9}{2}u^2 - \frac{3}{2}(u^2 + v^2) \quad k = 1$$
$$1 + 3v + \frac{9}{2}v^2 - \frac{3}{2}(u^2 + v^2) \quad k = 2$$
$$1 - 3u + \frac{9}{2}u^2 - \frac{3}{2}(u^2 + v^2) \quad k = 3$$
$$1 - 3v + \frac{9}{2}v^2 - \frac{3}{2}(u^2 + v^2) \quad k = 4$$
$$1 + 3(u + v) + \frac{9}{2}(u + v)^2 - \frac{3}{2}(u^2 + v^2) \quad k = 5$$
$$1 + 3(-u + v) + \frac{9}{2}(-u + v)^2 - \frac{3}{2}(u^2 + v^2) \quad k = 6$$
$$1 + 3(-u - v) + \frac{9}{2}(u + v)^2 - \frac{3}{2}(u^2 + v^2) \quad k = 7$$
$$1 + 3(u - v) + \frac{9}{2}(u - v)^2 - \frac{3}{2}(u^2 + v^2) \quad k = 8$$

5.2.1.4 Mass and Momentum Conservations

The summation of distribution functions at each lattice site represents the macroscopic fluid density,

$$\rho = \sum_{k=0}^{8} f_k \tag{5.9}$$

The momentum can be represented as an average of the lattice (microscopic) velocities c_k, weighted by the distribution function,

$$\rho \mathbf{u} = \sum_{\mathbf{k}=\mathbf{0}}^{8} f_k \mathbf{c_k} \tag{5.10}$$

or

$$\mathbf{u} = \frac{1}{\rho} \sum_{\mathbf{k}=\mathbf{0}}^{8} f_k \mathbf{c_k} \tag{5.11}$$

The pressure is given by $p = \rho/3$, which postulates a constant sound speed of

$$c_s = \frac{c_k}{\sqrt{3}}$$

The frequency of relaxation parameter, w can be calculated from Eq. 5.7, for a given kinetic viscosity of the fluid.

The corresponding Reynolds number can be calculated as $Re = U \cdot L/v$. U and L are characteristic velocity and length scale of the given flow. For example flow in a duct with uniform inlet velocity u_{in} and height H, Reynolds number usually expressed as $Re = u_{\text{in}} H/v$. In LBM, the Reynolds number need to be matched, hence, if number of lattices in characteristics length direction N, is set to 100 and viscosity of the fluid is v, then U can be selected to match macroscopic Reynolds number. Hence, the actual inlet velocity magnitude is completely ignored. In order to avoid stability problem and for accurate results, the lattice inlet velocity needs to be kept small, in the order of 0.1 or 0.2. For a given Reynolds number and kinematic viscosity, if the calculated velocity is large, then number of the lattice should be increased.

5.3 Boundary Conditions

One of the important and crucial issues in LB simulation of flow is accurate modeling of the boundary conditions. Specifying boundary conditions for Navier–Stokes equations is somehow straightforward. This is not the case for LBM, where the inward distribution functions to the integration domain need to be determined

at the boundaries. Therefore, we need to determine appropriate equations for calculating those distribution functions at the boundaries for a given boundary condition. In the literature different approaches are suggested and tested. In the following paragraphs, different boundary conditions are explained. The mean idea is not to review different attempts, but to discuss, in our point of view, the simple and more robust methods.

5.3.1 Bounce Back

Bounce back is used to model solid stationary or moving boundary condition, non-slip condition, or flow-over obstacles. In the following sentences bounce back method of stationary boundaries will be discussed. The method is quite simple and mainly implies that an incoming particle towards the solid boundary bounces back into flow domain. In the literature, a few versions of bounce back scheme have been suggested. One of the schemes is to locate the wall at half the distance from the lattice sites, as shown in Fig. 5.2. The distribution functions,

f_4, f_7 and f_8 are known from streaming process. It is assumed that, when these known distribution functions hit the wall, bounce back to the solution domain. Therefore,

$f_5 = f_7, f_2 = f_4$ and $f_6 = f_8$. Notice that f_7 at node say (i, j) is equal to f_7 of node $(i + 1, j + 1)$ after streaming, similar argument is valid for f_8, which is coming from node of $(i - 1, j + 1)$. Hence, the boundary condition must be applied after streaming process.

The bounce back ensures conservation of mass and momentum at the boundary. Other possible arrangement is shown in Fig. 5.3, yet the wall at the mid way between two lattice sites. However, the domain is extended inside the solid wall. The bounce back takes places inside the wall.

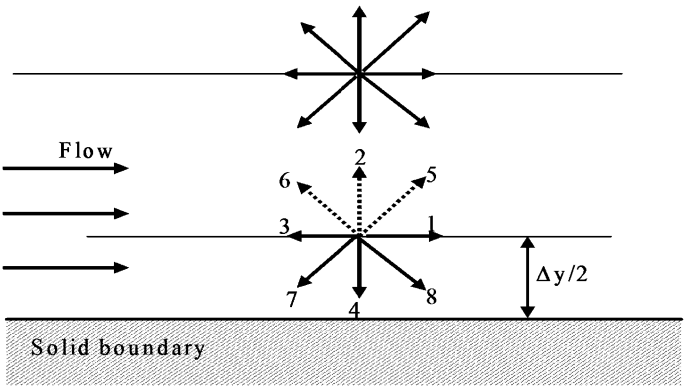

Fig. 5.2 Bounce back scheme

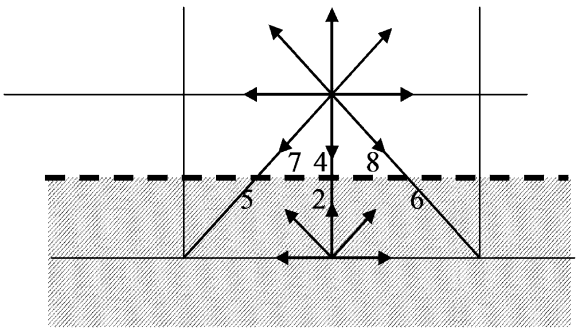

Fig. 5.3 Bounce back, scheme II

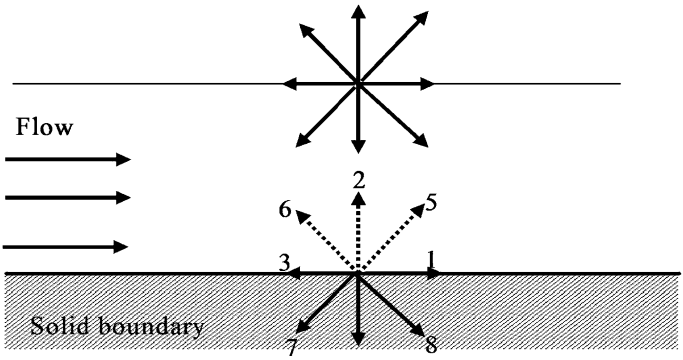

Fig. 5.4 Simple bounce back scheme

Then, $f_2 = f_4, f_5 = f_7$ and $f_6 = f_8$.

Notice that f_7 is bouncing back from left-hand lattice in the solid wall and f_8 bouncing back from right-hand lattice in the solid wall in reference to the main lattice location. It is important to carry the streaming process into the solid wall. Then $f_5 = f_7, f_6 = f_8$ and $f_2 = f_4$ for the same node inside the wall.

Another scheme, yet the simple bounce back scheme (III) is shown in Fig. 5.4, the lattices are located directly at the solid surface and not at middle plane, as shown in Fig. 5.4.

There simply let, $f_5 = f_7, f_2 = f_4$ and $f_6 = f_8$, where f_7, f_4 and f_8 are known from streaming process. Some authors claim that this scheme is first order accurate. This scheme is quite simple compared with the previous schemes and will be adopted in the coding. Using scheme I and II is not that difficult and reader may modify the code for these kinds of bounce back schemes.

Bounce back can be applied to all lattices on the solid surfaces in modeling flow over an obstacle. For instance flow in porous medium, Fig. 5.5, can be simulated by discrete solids embedded in a fluid flow. Bounce back should be used for nodes on the solid surfaces and set the velocity in and on the solid to zero.

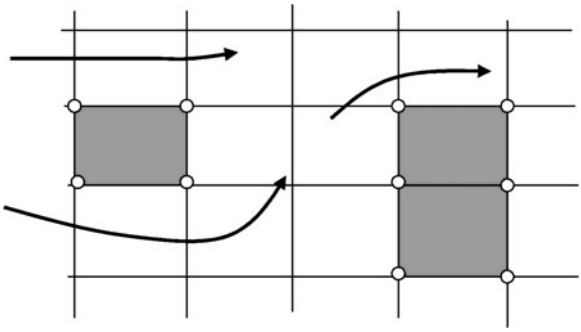

Fig. 5.5 Flow through a porous medium, discrete model

5.3.2 Boundary Condition with Known Velocity

It is very common in practical applications that we know velocity component at the boundary, for example inlet velocity for a channel flow. Zhu and He described a method to calculate three unknown distribution functions based on Eqs. 5.9 and 5.10 with equilibrium conditions assumption, normal to the boundary.

Equation 5.9 can be written as,

$$\rho = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8 \tag{5.12}$$

Equation 5.10 can be written for x-component as,

$$\rho u = f_1 + f_5 + f_8 - f_6 - f_3 - f_7 \tag{5.13}$$

and for y-component as,

$$\rho v = f_5 + f_2 + f_6 - f_7 - f_4 - f_8 \tag{5.14}$$

We have four unknowns, three distribution functions and extra unknown, ρ, at the boundary. However, we have only three equations, we need to construct another equation or condition to solve for four unknowns (three distribution functions and ρ). The condition comes from assumption of equilibrium condition, normal to the surface.

Let us illustrate the above concept by an example. Figure 5.6 shows a domain with east, west, north, and south boundaries. The velocity components are known at these boundaries.

In the figure, the unknown distribution functions are denoted by dotted lines. The known distribution functions after streaming are denoted by solid lines.

5.3.2.1 West Boundary

The x-component velocity, $u = u_w$, and y-component velocity, $v = v_w$, are known, have given values including zero (solid wall).

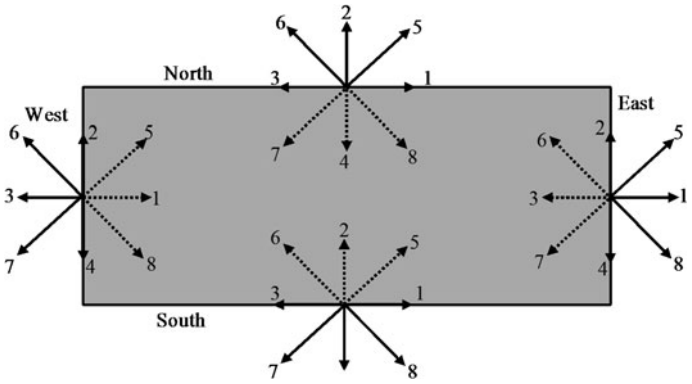

Fig. 5.6 Distribution functions at the boundaries of a domain

Then Eq. 5.12 can be written as,

$$\rho_w = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8 \tag{5.15}$$

Equation 5.13 can be written as,

$$\rho_w u_w = f_1 + f_5 + f_8 - f_6 - f_3 - f_7 \tag{5.16}$$

And Eq. 5.14 can be written as,

$$\rho_w v_w = f_5 + f_2 + f_6 - f_7 - f_4 - f_8 \tag{5.17}$$

The equilibrium condition normal to the boundary, yields

$$f_1 - f_1^{eq} = f_3 - f_3^{eq} \tag{5.18}$$

where f^{eq} can be calculated from Eq. 5.5.

$$f_1^{eq} = \frac{1}{9}\rho_w \left[1 + 3u_w + \frac{9}{2}u_w^2 - \frac{3}{2}(u_w^2 + v_w^2)\right] \tag{5.19}$$

and

$$f_3^{eq} = \frac{1}{9}\rho_w \left[1 - 3u_w + \frac{9}{2}u_w^2 - \frac{3}{2}(u_w^2 + v_w^2)\right] \tag{5.20}$$

Substituting above equations in Eq. 5.18 yields,

$$f_1 = f_3 + \frac{2}{3}\rho_w u_w \tag{5.21}$$

For the west boundary, the unknown distribution functions are f_1, f_5 and f_8

Solving Eqs. 5.15–5.21 for four unknowns, ρ_w, f_1, f_5, and f_8, yield the following equations:

$$\rho_w = \frac{1}{1-u_w}[f_0 + f_2 + f_4 + 2(f_3 + f_6 + f_7)] \tag{5.22}$$

$$f_5 = f_7 - \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_w u_w + \frac{1}{2}\rho_w v_w \tag{5.23}$$

$$f_8 = f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_w u_w - \frac{1}{2}\rho_w v_w \tag{5.24}$$

Hence four unknowns are specified at the west boundary, similar method can be applied for other boundaries. The conditions for other boundaries will be given in the following equations without derivations, following the same procedure as before.

5.3.2.2 East Boundary

$$\rho_E = \frac{1}{1+u_E}[f_0 + f_2 + f_4 + 2(f_1 + f_5 + f_8)] \tag{5.25}$$

$$f_3 = f_1 - \frac{2}{3}\rho_E u_E \tag{5.26}$$

$$f_7 = f_5 + \frac{1}{2}(f_2 - f_4) - \frac{1}{6}\rho_E u_E - \frac{1}{2}\rho_E v_E \tag{5.27}$$

$$f_6 = f_8 - \frac{1}{2}(f_2 - f_4) - \frac{1}{6}\rho_E u_E + \frac{1}{2}\rho_E v_E \tag{5.28}$$

5.3.2.3 North Boundary

$$\rho_N = \frac{1}{1+v_N}[f_0 + f_1 + f_3 + 2(f_2 + f_6 + f_5)] \tag{5.29}$$

$$f_4 = f_2 - \frac{2}{3}\rho_N v_N \tag{5.30}$$

$$f_7 = f_5 + \frac{1}{2}(f_1 - f_3) - \frac{1}{6}\rho_N v_N - \frac{1}{2}\rho_N u_N \tag{5.31}$$

$$f_8 = f_6 + \frac{1}{2}(f_3 - f_1) + \frac{1}{2}\rho_N u_N - \frac{1}{6}\rho_N v_N \tag{5.32}$$

5.3.2.4 South Boundary

$$\rho_S = \frac{1}{1 - v_S}[f_0 + f_1 + f_3 + 2(f_4 + f_7 + f_8)] \tag{5.33}$$

$$f_2 = f_4 + \frac{2}{3}\rho_S v_S \tag{5.34}$$

$$f_5 = f_7 - \frac{1}{2}(f_1 - f_3) + \frac{1}{6}\rho_S v_S + \frac{1}{2}\rho_S u_S \tag{5.35}$$

$$f_6 = f_8 + \frac{1}{2}(f_1 - f_3) + \frac{1}{6}\rho_S v_S - \frac{1}{2}\rho_S u_S \tag{5.36}$$

5.3.3 Equilibrium and Non-Equilibrium Distribution Function

In general, the distribution function can be split into two parts, equilibrium and non-equilibrium,

$$f = f^{eq} + f^{neq} \tag{5.37}$$

For nodes on the boundary, Fig. 5.7, the above equation can be written as,

$$f_w = f_f + (f_w^{eq} - f_f^{eq}) \tag{5.38}$$

For instance, f_1,f_5 and f_8 are unknowns and can be determined as,

$$f_1 = f_3 + (f_1^{eq} - f_3^{eq}) \tag{5.39}$$

which equal to

$$f_1 = f_3 + \frac{2}{3}\rho_w u_w \tag{5.40}$$

similarly for f_5 and f_8,

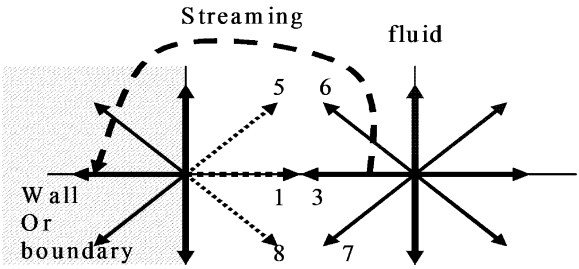

Fig. 5.7 Node on the boundary and another near the boundary

$$f_5 = f_7 + (f_5^{eq} - f_7^{eq}) \tag{5.41}$$

and

$$f_8 = f_6 + (f_8^{eq} - f_6^{eq}) \tag{5.42}$$

For a given velocity at the boundary, this kind of boundary conditions is simple to apply and will be adopted in the following examples.

5.3.4 Open Boundary Condition

In some problems, the outlet velocity is not known. The normal practice is to use extrapolation for the unknown distribution functions. For instance, if the east boundary condition on Fig. 5.6 represent the outlet condition. Then f_3, f_6 and f_7 need to be calculated at the east boundary, $i = n$. Second order polynomial can be used, as:

$$f_{3,n} = 2 \cdot f_{3,n-1} - f_{3,n-2} \tag{5.43}$$

$$f_{6,n} = 2 \cdot f_{6,n-1} - f_{6,n-2} \tag{5.44}$$

and

$$f_{7,n} = 2 \cdot f_{7,n-1} - f_{7,n-2}. \tag{5.45}$$

However, in some cases the above boundary conditions lead to unstable solution. Using first order interpolation showed higher stability than the second order interpolation scheme. Another approach is to assume that the pressure at the boundary is known, i.e., density, ρ. Hence, the density at the boundary need to be set to a constant (say 1.0, for example). For instance if the pressure is given at the outlet, the following equation can be used to specify the missing distribution functions,

$$u_x = -1 + \frac{f_0 + f_2 + f_4 + 2(f_1 + f_5 + f_8)}{\rho_{outlet}} \tag{5.46}$$

$$f_3 = f_1 - \frac{2}{3}\rho_{outlet} u_x \tag{5.47}$$

$$f_7 = f_5 + \frac{1}{2}(f_2 - f_4) - \frac{1}{6}\rho_{outlet} u_x \tag{5.48}$$

$$f_6 = f_8 - \frac{1}{2}(f_2 - f_4) - \frac{1}{6}\rho_{outlet} u_x \tag{5.49}$$

As mentioned ρ_{outlet} can be set to a value, say 1.0.

5.3.5 Periodic Boundary Condition

Periodic boundary condition become necessary to apply to isolate a repeating flow conditions. For instance flow over bank of tubes as shown in Fig. 5.8. The flow conditions above the line a and below the line b are the same. Hence, it is sufficient to model the flow between these two lines and use periodic boundary conditions alone these boundaries. The distribution functions that are leaving line a are the same as the distribution functions entering from line b and visa versa.

The distributions functions, f_4, f_7 and f_8 are unknown on the line a and f_2, f_5 and f_6 are unknown on the line b. The periodic boundary is as follows:

along line a:

$f_{4,a} = f_{4,b}, f_{7,a} = f_{7,a}$ and $f_{8,a} = f_{8,b}$

along line b:

$f_{2,b} = f_{2,a}, f_{5,b} = f_{5,a}$ and $f_{6,b} = f_{6,a}$.

5.3.6 Symmetry Condition

Many practical problems show a symmetry along a line or a plane. Then, it is beneficial to find a solution for only one part of the domain, which saves computer resources. For example, flow in a channel, Fig. 5.9, the flow above the symmetry line is mirror image of the flow below the symmetry line. Therefore the integration should be carried only for one part of the domain, and symmetry condition need to be applied along the symmetry line.

The distribution functions, f_5, f_2 and f_6 are unknowns. The way to construct these functions is to set them equal to their mirror images, i.e., $f_5 = f_8, f_2 = f_4$ and $f_6 = f_8$.

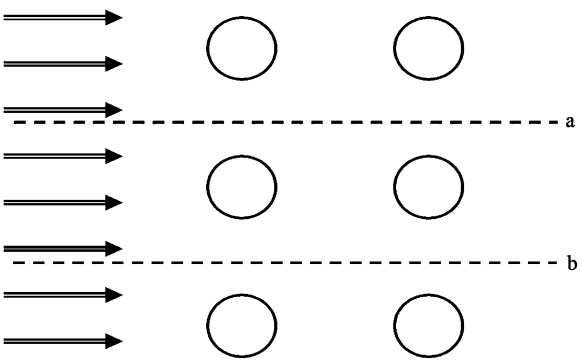

Fig. 5.8 Illustrates the periodic boundary conditions

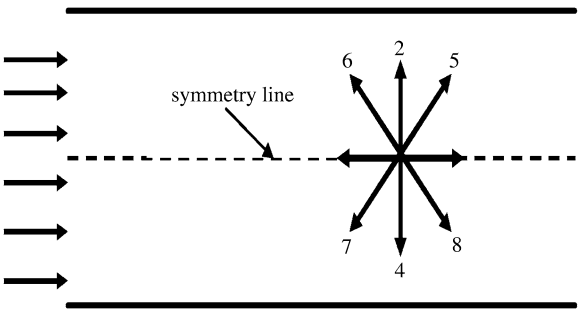

Fig. 5.9 Illustrates the symmetry boundary conditions

5.4 Computer Coding

Indeed the coding is quite simple and similar to the codes developed in Chap. 4 for D2Q9, except that extra terms should be added to f^{eq} as represented in Eq. 5.5.

The algorithm is as follows

```
==============================
Input data
Main Loop
Calculate the equilibrium distribution function
Collision
Streaming
Calculate the Distribution functions at the boundaries
Calculate, Density and Velocity Component
End of the main Loop
Output data
==================================
```

5.5 Examples

5.5.1 Lid Driven Cavity

Lid driven cavity is used as test benchmark problem to test CFD codes. A square cavity of 0.20 m side is filled with engine oil at 15°C (kinematic viscosity is $1.2 \times 10^{-3}\,\mathrm{m}^2/\mathrm{s}$). The lid is set to motion with a speed of 6 m/s.

In isothermal fluid flow it is important to match Reynolds number and geometrical aspect ratio. For the above problem, $Re = U_{\mathrm{lid}} * H/v = 6 * 0.2/0.00012 = 1,000$.

In LBM, we are free to use any values for U_{lid} and viscosity provided that $Re = 1,000$. In order to reduce compressibility effect of LBM, set U_{lattice} to 0.1 and viscosity to 0.01, then we need 100 lattices in H direction because $Re = 1,000 = U_{\mathrm{lattice}} N/v_{\mathrm{lattice}} = 0.1 * N/0.01$, then $N = 100$. Since the cavity is square, i.e., aspect ratio of unity, then 100 nodes need to be adopted in y-direction too.

The computer code is given in the Appendix.

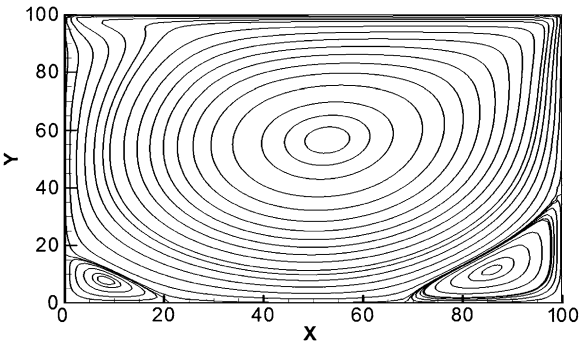

Fig. 5.10 Streamlines for lid driven cavity, $Re = 1{,}000$

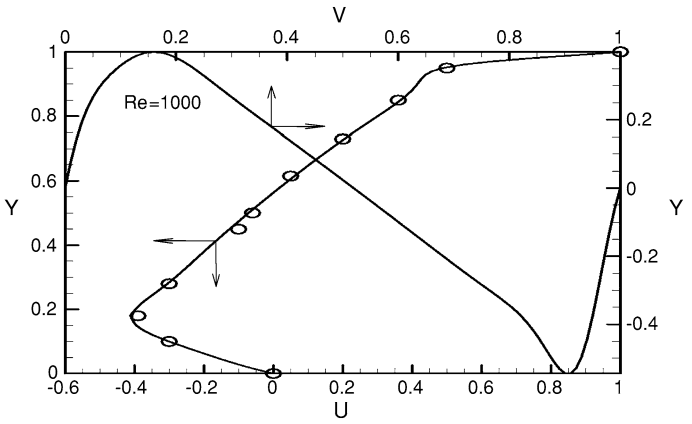

Fig. 5.11 Velocity profiles at the mid-section and at the mid-height of the cavity

The results of simulations are shown in Figs. 5.10 and 5.11 for streamlines and velocity at different sections along the cavity, which are well compared with FVM results.

Figure 5.11 U and V-velocity at $X = 0.5$ and $Y = 0.5$ of the height of the cavity, respectively, $Re = 1{,}000$.

The results are quite well compared with benchmark solutions. The symbols in Fig. 5.11 denotes benchmark solution obtained by FVM.

5.5.2 Developing Flow in a Two-Dimensional Channel

Water flows between two parallel plates (channel Fig. 5.12) with a uniform velocity of 0.02 m/s, the gap between the plates is 2.0 cm, and the length of each plate is 50 cm. Determine the velocity profile and developing length of the flow, etc.

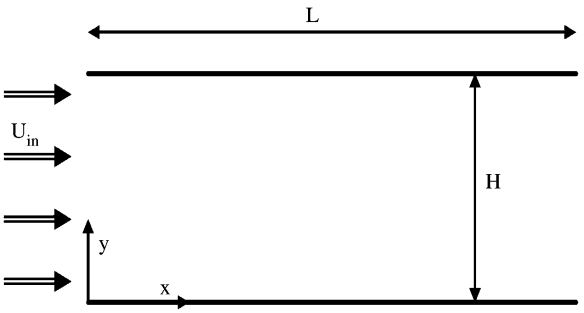

Fig. 5.12 Flow in a channel

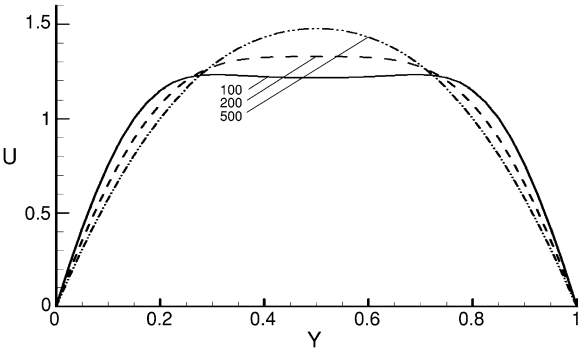

Fig. 5.13 Velocity profiles at different cross sections, $x = 100$, 200 and 300

The density and kinematic viscosity of water are $1,000\,\mathrm{kg/m^3}$ and $10^{-6}\,\mathrm{m^2/s}$, respectively.

Assume that the width of the plates is 1.0 m (unit length).

The Reynolds number is $U_{\mathrm{in}} * h/v = 0.02 * 0.02/10^{-6} = 400$.

The aspect ratio of the problem is $L/H = 50/2 = 25$.

Again, in LBM method we can chose any value for velocity and gap of channel provided that we keep $Re = 400$ and aspect ratio $= 25$.

Let us use $U_{\mathrm{lattice}} = 0.2$, and 40 lattices in y-direction, then lattice viscosity should be calculated to keep $Re = 400$, $Re = U_{\mathrm{lattice}} N/v_{\mathrm{lattice}} = 0.1 \times 80/v_{\mathrm{lattice}}$, then $v_{\mathrm{lattice}} = 0.02$.

Aspect ratio is $25 = M/N$, since $N = 40$, therefore we need to use $1,000$ lattices in x-direction.

Figure 5.13 shows the developing velocity profiles normalized by inlet velocity at three different x-locations ($x/H = 100/40$, $200/40$, $300/40$). The fully developed velocity profile is evidence at $x/H = 300/40$=7.5. It is known that the fully developed velocity profile in a 2-D channel is parabolic with a maximum velocity equal to 1.5 of the average velocity, $u/u_{\mathrm{in}} = 1.5(1 - y/H)$.

The code for lid-driven cavity need to be slightly modified. The subroutine sfbound should be replaced by:

5.5.2.1 The Computer Code for Flow in a Channel

```
==================================
    subroutine sfbound(f,n,m,uo)
    real f(0:8,0:n,0:m)
    real uo
    do j=0,m
    ! bounce back on west boundary
    rhow=(f(0,0,j)+f(2,0,j)+f(4,0,j)+2.*(f(3,0,j)+f(6,0,j)+f(7,0,j)))/(1.-uo)
    f(1,0,j)=f(3,0,j)+2.*rhow*uo/3.
    f(5,0,j)=f(7,0,j)+rhow*uo/6.
    f(8,0,j)=f(6,0,j)+rhow*uo/6.
    end do
    ! bounce back on south boundary
    do i=0,n
    f(2,i,0)=f(4,i,0)
    f(5,i,0)=f(7,i,0)
    f(6,i,0)=f(8,i,0)
    end do
    !bounce back, north boundary
    do i=0,n
    f(4,i,m)=f(2,i,m)
    f(8,i,m)=f(6,i,m)
    f(7,i,m)=f(5,i,m)
    end do
    ! accont for open boundary condition at the outlet
    do j=1,m
    f(1,n,j)=2.*f(1,n-1,j)-f(1,n-2,j)
    f(5,n,j)=2.*f(5,n-1,j)-f(5,n-2,j)
    f(8,n,j)=2.*f(8,n-1,j)-f(8,n-2,j)
    end do
    return
    end
    ============================
```

And the following line should be added to the end of subroutine rhouv:

```
============================
do j=1,m
 v(n,j)=0.0
end do
============================
```

This is necessary because at the outlet, we assume that *y*-component velocity (v) should be zero. This is a rational assumption if we do not know the outlet boundary condition.

5.5.3 Flow over Obstacles

Flow over obstacles is very common in engineering and geophysical applications, such as flow through a porous medium. The solid phase can be simulated as

unconnected obstacles embedded in the flow. The normal practice is to use bounce-back on the solid surfaces. In the following section two problems will be worked out, namely flow over a backward-facing step and flow over a few solid obstacles embedded in the flow.

For the first example, let us modify the pervious example by adding an obstacle at the entrance of the duct. The obstacle height is half of the total width ($H/2$) of the duct and its length is equal to the width of the duct (H).

The subroutines that need to be modified are subroutine sfbound as:

```
    ====================================
subroutine sfbound(f,n,m,uo)
    real f(0:8,0:n,0:m)
    real uo
    do j=21,m
    ! bounce back on west boundary
    rhow=(f(0,0,j)+f(2,0,j)+f(4,0,j)+2.*(f(3,0,j)+f(6,0,j)+f(7,0,j)))/(1.-uo)
    f(1,0,j)=f(3,0,j)+2.*rhow*uo/3.
    f(5,0,j)=f(7,0,j)+rhow*uo/6.
    f(8,0,j)=f(6,0,j)+rhow*uo/6.
    end do
    ! bounce back on south boundary
    do i=0,n
    f(2,i,0)=f(4,i,0)
    f(5,i,0)=f(7,i,0)
    f(6,i,0)=f(8,i,0)
    end do
    ! bounce back, north boundary
    do i=0,n
    f(4,i,m)=f(2,i,m)
    f(8,i,m)=f(6,i,m)
    f(7,i,m)=f(5,i,m)
    end do
    ! account for open boundary condition at the outlet
    do j=1,m
    f(1,n,j)=2.*f(1,n-1,j)-f(1,n-2,j)
    f(5,n,j)=2.*f(5,n-1,j)-f(5,n-2,j)
    f(8,n,j)=2.*f(8,n-1,j)-f(8,n-2,j)
    end do
    ! obstacle at the inlet x=40, y=20
    do i=0,40
    f(2,i,20)=f(4,i,20)
    f(5,i,20)=f(7,i,20)
    f(6,i,20)=f(8,i,20)
    end do
    do j=0,20
    f(1,40,j)=f(3,40,j)
    f(5,40,j)=f(7,40,j)
    f(8,40,j)=f(8,40,j)
    end do
    return
    end
    ========================
```

And we need to add the following to the end of subroutine rhouv:

```
=======================
do j=0,m
v(n,j)=0.0
end do
do j=0,20
do i=0,40
u(i,j)=0.0
v(i,j)=0.0
end do
end do
======================
```

Figure 5.14 shows the streamlines with main recirculation region on the shadow of the step. The flow re-attachment takes place at $x = 180$, $x/H = 4.5$.

Figure 5.15 shows the u-velocity profiles at different cross sections, $x = 100$, 200, and 500, corresponding to $x/H = 2.5$, 5, and 12.5, respectively.

Let us put a L-shaped obstacle in a duct. The L-shape is located at $x = 100$ and $y = 20$ extended to $x = 140$ and $y = 30$.

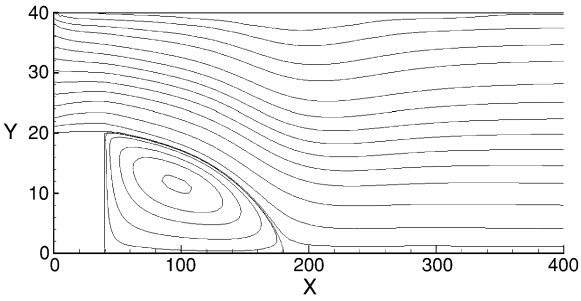

Fig. 5.14 Streamlines in a channel with back-facing step, $Re = 400$.

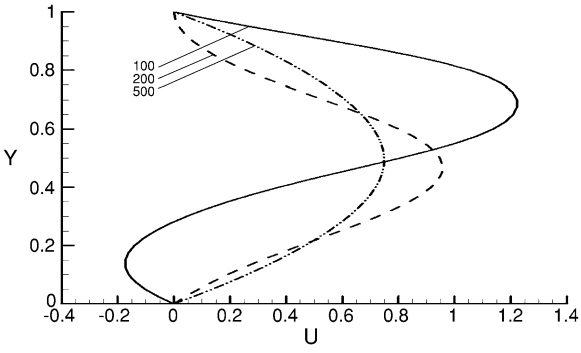

Fig. 5.15 Velocity profiles at different cross sections, $Re = 400$

Figure 5.16 shows streamlines of the problem. Note that the resolution is not that good because number of lattices used are not enough for good resolution.

The modified subroutine sfbound is:

```
=======================
subroutine sfbound(f,n,m,uo)
real f(0:8,0:n,0:m)
real uo
do j=0,m
! bounce back on west boundary
rhow=(f(0,0,j)+f(2,0,j)+f(4,0,j)+2.*(f(3,0,j)+f(6,0,j)+f(7,0,j)))/(1.-uo)
f(1,0,j)=f(3,0,j)+2.*rhow*uo/3.
f(5,0,j)=f(7,0,j)+rhow*uo/6.
f(8,0,j)=f(6,0,j)+rhow*uo/6.
end do
! bounce back on south boundary
do i=0,n
f(2,i,0)=f(4,i,0)
f(5,i,0)=f(7,i,0)
f(6,i,0)=f(8,i,0)
end do
! bounce back, north boundary
do i=0,n
f(4,i,m)=f(2,i,m)
f(8,i,m)=f(6,i,m)
f(7,i,m)=f(5,i,m)
end do
! account for open boundary condition at the outlet
do j=1,m
f(1,n,j)=2.*f(1,n-1,j)-f(1,n-2,j)
f(5,n,j)=2.*f(5,n-1,j)-f(5,n-2,j)
f(8,n,j)=2.*f(8,n-1,j)-f(8,n-2,j)
end do
! obstacle at the inlet x=40, y=20
do i=100,140
f(4,i,20)=f(2,i,20)
f(7,i,20)=f(5,i,20)
f(8,i,20)=f(6,i,20)
end do
do i=100,105
f(2,i,30)=f(4,i,30)
f(5,i,30)=f(7,i,30)
f(6,i,30)=f(8,i,30)
end do
do i=105,140
f(2,i,25)=f(4,i,25)
f(5,i,25)=f(7,i,25)
f(6,i,25)=f(8,i,25)
end do
```

```
do j=20,30
f(3,100,j)=f(1,100,j)
f(7,100,j)=f(5,100,j)
f(6,100,j)=f(8,100,j)
end do
do j=25,30
f(1,105,j)=f(3,105,j)
f(5,105,j)=f(7,105,j)
f(8,105,j)=f(6,105,j)
end do
do j=20,25
f(1,140,j)=f(3,140,j)
f(5,140,j)=f(7,140,j)
f(8,140,j)=f(6,140,j)
end do
return
end
======================
```

and modification for subroutine rhouv is:

```
========================
do j=0,m
v(n,j)=0.0
end do
do j=20,30
do i=100,105
u(i,j)=0.0
v(i,j)=0.0
end do
end do
do j=20,25
do i=105,140
u(i,j)=0.0
v(i,j)=0.0
end do
end do
========================
```

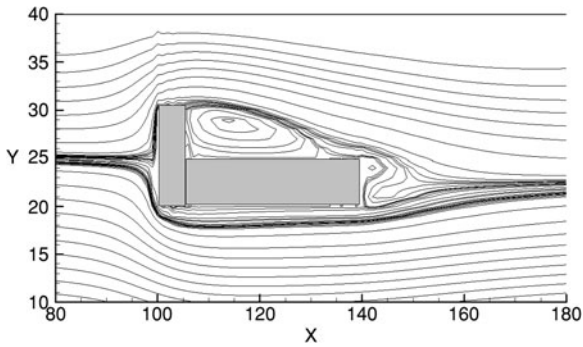

Fig. 5.16 Streamlines in a channel with L-shape obstacle

The mentioned examples clearly show the ability of LBM to model complex flow conditions.

In the following chapter, the non-isothermal problems (forced and natural convection) will be discussed.

5.6 Vorticity and Stream Function Approach

Another approach to solve two dimensional fluid flow problem is to used VSF approach. The pressure gradient term in the Navier–Stokes equation can be eliminated by subtracting the cross differentiation of x and y momentum equations from each other. The non-dimensional governing equations can be written as,

$$\frac{\partial \Omega}{\partial t} + \frac{\partial}{\partial x}\left(u\frac{\partial \Omega}{\partial x}\right) + \frac{\partial}{\partial y}\left(v\frac{\partial \Omega}{\partial y}\right) = \frac{1}{Re}\left(\frac{\partial^2 \Omega}{\partial x^2} + \frac{\partial^2 \Omega}{\partial y^2}\right) \tag{5.50}$$

and

$$\frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \Psi}{\partial y^2} = -\Omega \tag{5.51}$$

where Ω and Ψ are vorticity and stream function, respectively. Equation 5.46 is advection–diffusion equation which can be solved using methods introduced in Chap. 4, for instance using D2Q4 or D2Q5. Equation 5.47 is Laplace equation with source term, which can be solved either by finite difference or by method introduced in Chap. 3 by using D2Q4 or D2Q5 with a different distribution function. Hence, there is no need to elaborate on the solution techniques. Moreover, axis-symmetric fluid flow problem (flow in a pipe, for example) can by solved with the mentioned method with extra source term in the above equations. This can be an exercise to the reader.

5.7 Hexagonal Grid

In the previous sections, the intension was given to rectangular lattice arrangements. However, it is possible to use hexagonal lattice arrangements for the previously mentioned problems. Figure 5.17 shows hexagonal lattice configuration with linkage numbering for ease of programming. The angle between the direction is 60°. For D2Q7, with stationary particle and six moving particles, the weighting factor w_i is 1/2 for $i = 0$ and 1/12 for $i = 1$–7. The streaming velocities are, $c_0 = (0,0), c_1 = (1,0), c_2 = (-1,0), c_3 = (1/2, \sqrt{3}/2), c_4 = (-1/2, \sqrt{3}/2), c_5 = (-1/2, -\sqrt{3}/2)$ and $c_6 = (1/2, -\sqrt{3}/2)$

The equilibrium distribution function is,

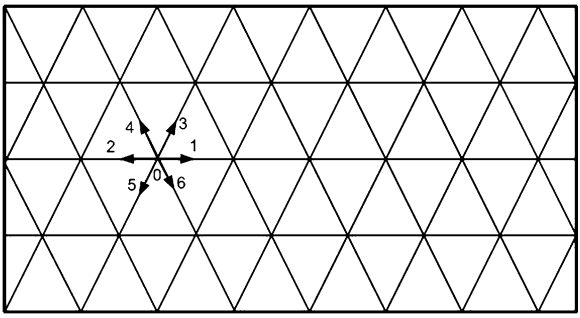

Fig. 5.17 Hexagonal lattice arrangement

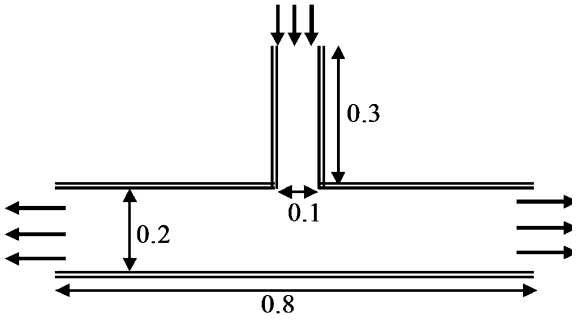

Fig. 5.18 Confined jet between two parallel plates

$$f_i^{\mathrm{eq}} = w_i \rho (1 + 4c_i \cdot u_i + 8(c_i \cdot u_i)^2 - 2u_i \cdot u_i) \tag{5.52}$$

5.8 Problems

1. Flow through an expanding channel as shown in Fig. 3.13. If the Reynolds number based on entrance height (0.4) is 500. The channel is open to atmosphere from the right-hand side. Determine the streamlines and velocity profiles at different cross sections.
2. A confined water jet impinging on a plate as shown in Fig. 5.18. For Reynolds number based on jet width (0.1) of 500, determine the streamlines and velocity profiles at different locations.
3. Also, try to solve an unconfined jet impinging on a plate. The distance between plate and jet is 10 of the jet width.

Chapter 6
Non-Isothermal Incompressible Fluid Flow

In Chap. 5, isothermal fluid flow problems were discussed. In many applications heat and/or mass transfer were associated with flows, convection. In this chapter, two kinds of non-isothermal flows will be discussed, namely forced and natural convection. In the forced convection, the energy equation can be solved after getting the flow field, i.e., the momentum equation is not coupled with the energy equation. In natural convection, the momentum equation is coupled with the energy equation. Hence, both equations need to be solved simultaneously.

6.1 Naiver–Stokes and Energy Equations

For incompressible flow, two-dimensional, the conservative form for NS equation can be written in Cartesian coordinate system, with body force, as,

x-momentum

$$\frac{\partial \rho u}{\partial t}+\frac{\partial(\rho u u)}{\partial x}+\frac{\partial(\rho v u)}{\partial y}=-\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y}\right)+F_x \tag{6.1}$$

y-momentum

$$\frac{\partial \rho v}{\partial t}+\frac{\partial(\rho u v)}{\partial x}+\frac{\partial(\rho v v)}{\partial \mathrm{y}}=-\frac{\partial p}{\partial y}+\frac{\partial}{\partial x}\left(\mu\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial v}{\partial y}\right)+F_y \tag{6.2}$$

where F_x and F_y are body forces in x and y directions.

The energy equation can be written as,

$$\frac{\partial \rho c T}{\partial t}+\frac{\partial(\rho c u T)}{\partial x}+\frac{\partial(\rho c v T)}{\partial y}=\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+q''' \tag{6.3}$$

A. A. Mohamad, *Lattice Boltzmann Method*,
DOI: 10.1007/978-0-85729-455-5_6, © Springer-Verlag London Limited 2011

where q'''represents heat source per unit volume, such as radiation, heat generation due to chemical reaction, etc.

For forced convection, the body force is not a function of temperature. The main governing parameters for forced convection are, Reynolds (Re) and Prandtl (Pr) numbers and geometrical aspect ratio. Hence, in LBM simulation, we need to match these parameters. For natural convection (with Bosensque approximations), the controlling parameters are Grashof (Gr) or Rayleigh number (Ra), Pr and geometrical aspect ratio. For more details on the physics and mathematical modeling of convection, textbooks on heat transfer need to be consulated.

6.2 Forced Convection, D2Q9–D2Q9

Since there is no coupling between momentum and scalar equation (energy and/or mass transfer equation), the method used in Chaps. 4 and 5 can be combined together. Two different distribution functions need to be solved, for instance f for momentum and g for the scalar variable. Using BKG-LBM, the following equation need to be solved for momentum,

$$f_k(x + \Delta x, t + \Delta t) = f_k(x, t)[1 - \omega_m] + \omega_m f_k^{\text{eq}}(x, t) \tag{6.4}$$

The methodology is well explained in Chap. 5, hence there is no need for repetition.

For scalar function (temperature, species concentration), the following equation need to solved,

$$g_k(x + \Delta x, t + \Delta t) = g_k(x, t)[1 - \omega_s] + \omega_s g_k^{\text{eq}}(x, t) \tag{6.5}$$

The equilibrium distribution function is as in advection problem (Eq. 4.6),

$$g_k^{\text{eq}} = w_k \phi(x, t)\left[1 + \frac{c_k \cdot \vec{u}}{c_s^2}\right] \tag{6.6}$$

Notice that ω is different for momentum and scalar equations. For momentum,

$$\omega_m = \frac{1}{3 \cdot \nu + 0.5} \tag{6.7}$$

where ν is the kinematic viscosity and for the scalar

$$\omega_s = \frac{1}{3 \cdot \alpha + 0.5} \tag{6.8}$$

where α is the diffusion coefficient (thermal diffusion or mass diffusion coefficient).

Different boundary conditions for scalar equation were discussed in Chap. 4.

To illustrate the forced convection problem, let us work a few examples. A heated lid driven cavity and flow through a heated channel problems will be worked out with full computer code.

6.3 Heated Lid-Driven Cavity

Heated lid driven cavity, same problem as in Chap. 5, except that the lid is kept at a higher temperature compared with other boundaries. An enclosure is filled with air of viscosity, $v = 0.02$ (lattice units), and lid moves with velocity, $U_{\text{lid}} = 0.2$. For Reynolds number of 1,000, we need to use 100 x 100 lattices in x and y-directions. $Pr = v/\alpha = 0.02/\alpha = 0.71$ (air), hence $\alpha = 0.02817$. The lid is heated to a normalized temperature, $(T - T_{\text{coldwall}})/(T_{\text{wall}} - T_{\text{coldwall}})$, of 1.0 and east and west walls are kept at cold temperature at (normalized temperature 0.0), while the bottom of the cavity is thermally isolated. The computer code is given in the appendix.

Temperature profile along the height of cavity at three cross sections, first quarter, middle, and third quarter planes ($x = 25$, 50 and 75) are given in Fig. 6.2.

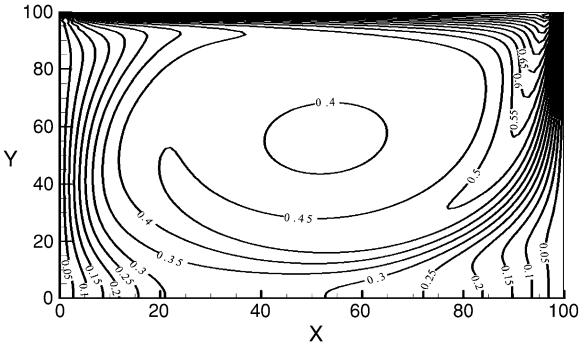

Fig. 6.1 Isotherms for the heated lid driven cavity, $Re = 1{,}000$, $Pr = 0.71$

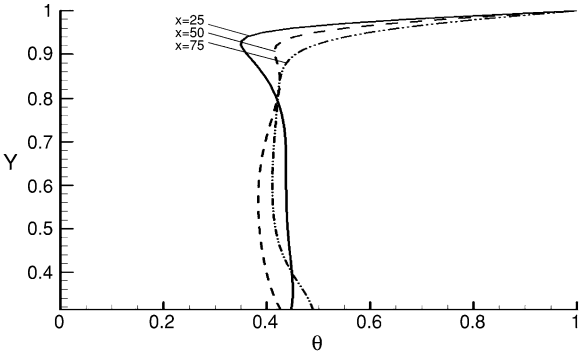

Fig. 6.2 Temperature profiles along the height of the cavity at different cross sections

6.4 Forced Convection Through a Heated Channel

A cold fluid flows through a heated wall channel. The walls of channel are kept at constant temperature. Water flows through a channel of 6.0 cm in height and 120 cm in length. The inlet temperature and velocity of the water are 20°C and 0.006 m/s, respectively. The channel walls are kept at 80°C. Determine the velocity and temperature profiles and rate of heat transfer. Assume that the width of the channel is unity.

Solution
$T_{\text{in}} = 20°\text{C} \quad U_{\text{in}} = 0.001\ \text{m/s}$
$T_{\text{w}} = 80°\text{C}$
$L = 1.20\ \text{m}$
$H = 0.06\ \text{m}$

The viscosity and thermal diffusivity of the water at mean temperature (50°C) are about $6.0 \times 10^{-7}\ \text{m}^2/\text{s}$ and $1.58 \times 10^{-7}\ \text{m}^2/\text{s}$, respectively. Reynolds number $= U_{\text{in}}H/v = 0.001 * 0.06/(6.0 \times 10{-}7 = 100.) \quad Pr = nu/\alpha = 6.0 \times 10^{-7}/1.58 \times 10^{-7}$ is about 3.8

Let us non-dimensionalize the temperature and introduce $\theta = (T - T_{\text{in}})/(T_{\text{w}} - T_{\text{in}})$, And let us use H for length scale.

At $x = 0\ \theta_{\text{in}} = 0$, at $y = 0$ and $y/H = 1.0$, $\theta_{\text{w}} = 1.0$. At $x = L, \frac{\partial\theta}{\partial x} = 0$.

Figures 6.3 and 6.4 show the isotherm distribution and u-velocity vectors in the channel. If you run the code, you should get the same results. These two examples clearly illustrate the application of LBM to forced convection.

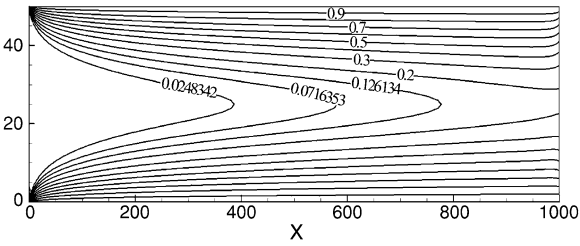

Fig. 6.3 Isothermals for forced convection in a heated channel

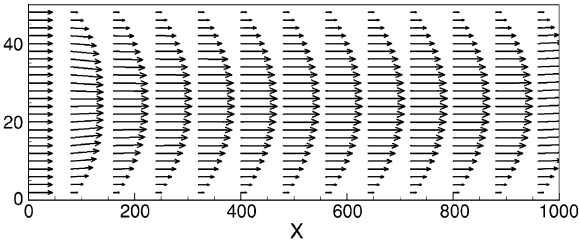

Fig. 6.4 Velocity vectors for forced convection in a heated channel

As an exercise, try to give different thermal diffusivity to a part of the channel along x-axis, to simulate conjugate effect. At the interface use the bounce back scheme.

6.5 Conjugate Heat Transfer

In some problems we are interested in solving conduction in solid, coupled with convection in the fluid. For instance flow in macro-channel, the conduction through the channel wall cannot be neglected. Usually, thermal diffusivity of the solid wall is different from the thermal diffusivity of the fluid. The conjugate problem can be easily handled by assigning different thermal diffusivity for solid, from that of the fluid and use bounce back at the interface. Also, the velocity components in the solid should be set to zero. That is all you need to do. The flux continuity at the interface will be ensured automatically. The following results obtained for the same problem above, forced convection in a channel, with 10 lattices from the bottom and top assigned as a solid wall with thermal diffusivity 10 times higher than the thermal diffusivity of the fluid. Figures 6.5 and 6.6 show the isotherms and velocity vectors predicted by LBM.

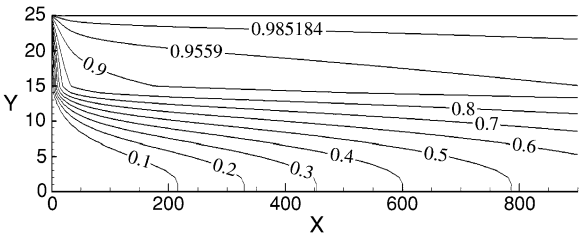

Fig. 6.5 Isotherms for conjugate convection problem in a heated channel

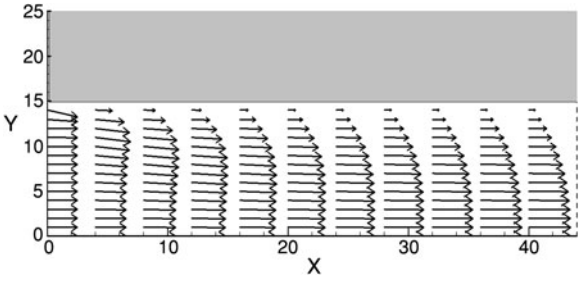

Fig. 6.6 Velocity vectors for forced conjugate convection in a heated channel

6.6 Natural Convection

For natural convection, the momentum and energy equations are coupled, both should be updated simultaneously. The flow is driven by temperature or mass gradient, i.e., buoyancy force. Hence, there is an extra force term needed to be considered in solving LB equation.

Then $F_x = \rho g_x$ and $F_y = \rho g_y$

For natural convection, the Boussinesq approximation yields,

$$F_x = \rho_o g_x \beta (T - T_{\text{ref}}) \tag{6.9}$$

and

$$F_y = \rho_o g_y \beta (T - T_{\text{ref}}) \tag{6.10}$$

Then, the non-dimensional form of the momentum (6.1 and 6.2) equations can be written as,

$$\frac{\partial u}{\partial t} + \frac{\partial (uu)}{\partial x} + \frac{\partial (vu)}{\partial y} = -\frac{\partial p}{\partial x} + \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right] + Ra_x \theta \tag{6.11}$$

y-momentum

$$\frac{\partial v}{\partial t} + \frac{\partial (uv)}{\partial x} + \frac{\partial (vv)}{\partial y} = -\frac{\partial p}{\partial y} + \left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right] + Ra_y \theta \tag{6.12}$$

where $Ra_x(\frac{g_x \beta \Delta T H^3}{\alpha \upsilon})$ and $Ra_y \frac{(g_y \beta \Delta T H^3}{\alpha \upsilon})$ are Rayleigh numbers in x and y directions. And $\theta = \frac{T - T_{\text{cold}}}{T_{\text{hot}} - T_{\text{cold}}}$.

The non-dimensional energy equation without heat source can be written as,

$$\frac{\partial \theta}{\partial t} + \frac{\partial (u\theta)}{\partial x} + \frac{\partial (v\theta)}{\partial y} = \frac{1}{Re\ Pr}\left[\frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \theta}{\partial y^2}\right] \tag{6.13}$$

Note that $H, H^2/v$, v/H and $(T_{\text{hot}} - T_{\text{cold}})$ are used in scaling of length, time, velocity, and temperature.

Force term: The force term can be added as an extra term to the right-hand side of the momentum equation, ρF, as,

$$F = 3w(k) g_x \beta \theta e_x + 3w(k) g_y \beta \theta e_y \tag{6.14}$$

There is another way to add force term to the LBE, by modifying velocity in calculating equilibrium distribution functions.

Newton's second law of motion states that,

$$\mathbf{F} = m\mathbf{a} = m\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} \tag{6.15}$$

where **a** and **u** are acceleration and velocity vectors.

Then,

$$\Delta \mathbf{u} = \frac{\tau \mathbf{F}}{\rho} \tag{6.16}$$

where τ is the relaxation time.

The velocity should be modified by $\Delta\mathbf{u}$ in calculating equilibrium distribution functions only, $\mathbf{u}^{eq} = \mathbf{u} + \Delta\mathbf{u}$.

The subroutine "collision" in the previous computer code need to be slightly modified, as:

```
    ========================================
    subroutine collision(u,v,f,feq,rho,omega,w,cx,cy,n,m)
real f(0:8,0:n,0:m)
real feq(0:8,0:n,0:m),rho(0:n,0:m)
real w(0:8), cx(0:8),cy(0:8)
real u(0:n,0:m), v(0:n,0:m)
    g=10.0                          ! it is better idea to pass g to the subroutine
instead of giving it a value inside the sub.
    beta=0.01                       ! read the above note
    DO i=0,n
DO j=0,m
t1=u(i,j)*u(i,j)+v(i,j)*v(i,j)
DO k=0,8
t2=u(i,j)*cx(k)+v(i,j)*cy(k)


    force=w(k)*g*beta*th(i,j)*cy(k)          ! body force in y-direction only
    feq(k,i,j)=rho(i,j)*w(k)*(1.0+3.0*t2+4.50*t2*t2-1.50*t1)
f(k,i,j)=omega*feq(k,i,j)+(1.-omega)*f(k,i,j)+rho(i,j)*force
END DO
END DO
END DO
return
end

    ========================================
```

6.6.1 *Example: Natural Convection in a Differentially Heated Cavity*

Differentially heated cavity used as a benchmark solution for testing CFD codes. The problem is a square cavity filled with air ($Pr = 0.7$) and heated from the left side wall and cooled from the right-hand side wall. Other connecting boundaries

(bottom and top) are thermally insulated.For $Ra = 10^5$, determine the isotherms, Nusselt number and streamlines.

Solution The cavity sketch is shown in Fig. 6.7 with coordinate system and boundary conditions.

We need to calculate $g\beta$, from Rayleigh number definitions, $Ra = \frac{g\beta\Delta TH^3}{\alpha\upsilon}$, then $g\beta = Ra\alpha\upsilon/(\Delta TH^3)$. In lattice unit, we can choose $\upsilon = 0.03$ (or any other value, but not large, less than 0.1). Since $Pr = \frac{\upsilon}{\alpha} = 0.7$, then $\alpha = 0.03/0.7$, therefore, $g\beta = 10^5 * 0.3^2/(0.7 * N^3)$, assuming that $\Delta T = 1.0$ and $H = N$, where N is the number of lattices in H direction.

Note that the velocity scale in natural convection is proportional to $\sqrt{g\beta\Delta TH}$, in order to ensure the problem in incompressible regime, the value of $\sqrt{g\beta\Delta TH}$ is less than about 0.1. For the mentioned problem $\sqrt{Ra\upsilon\alpha/N^2} = \sqrt{10^5 * 0.03^2/(0.7 * 100^2)} = 0.113$, which is fine. Note that 100 lattices are assumed in H direction.

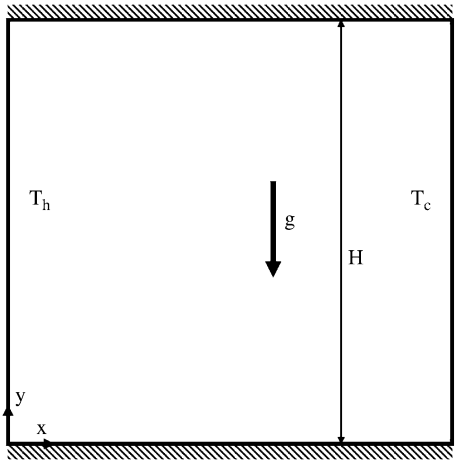

Fig. 6.7 Sketch of natural convection problem

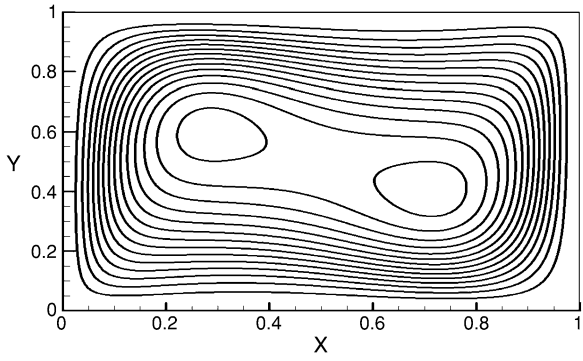

Fig. 6.8 Streamlines for natural convection in a square cavity, $Ra = 10^5, Pr = 0.71$

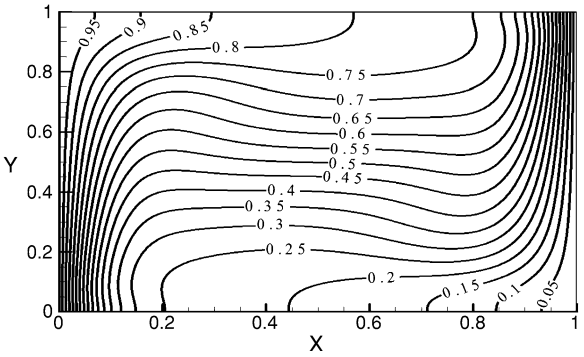

Fig. 6.9 Isotherm for natural convection in a square cavity, $Ra = 10^5$, $Pr = 0.71$

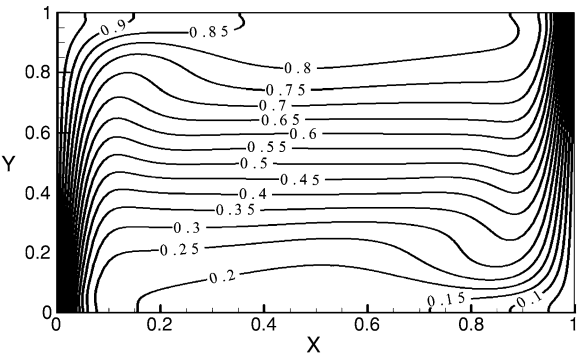

Fig. 6.10 Isotherms for natural convection in a square cavity, $Ra = 10^6, Pr = 0.71$

Complete computer code is given in the appendix.

Figure 6.8 shows streamlines for $Ra = 10^5$ and $Pr = 0.71$. The results match finite volume method predictions. The streamlines show perfect skew-symmetry, which is one criteria for the natural convection in a differentially heated cavity. Isotherms are given in Fig. 6.9.

For $Ra = 10^6$, change Ra to 10^6 instead of 10^5 in the code. The boundary layer becomes thinner, as shown in Fig. 6.10.

For higher Ra, more lattices were needed because the boundary layer becomes thinner as Rayleigh number increases.

6.7 Flow and Heat Transfer in Porous Media

Fluid flow and heat, and mass transfer in porous media has many applications. The literatures on the topic are extensive. However, in this section, the effect of adding porous material to fluid flow is discussed. The body force due to a porous material is given below,

$$\mathbf{F} = -\frac{v}{K}\mathbf{u} - \frac{\beta}{\sqrt{K}}|\mathbf{u}|\mathbf{u} \tag{6.17}$$

where K is the permeability of the medium. For medium uniformly filled with spherical particles of diameter of d_p, K can be expressed as,

$$K = \frac{\epsilon^3 d_\mathrm{p}}{150(1-\epsilon)^2} \tag{6.18}$$

The constant β, usually set to $1.75/\sqrt{150\epsilon}$, where ϵ is the porosity of the medium. The vector $\mathbf{u}$ is the flow velocity vector. First term in Eq. 6.17 is called Darcy term, which accounts for linear pressure drop due to presence of porous medium. The second nonlinear term is called Forchheimer term, due to nonlinear effect of porous medium. It is useful to define Darcy number in porous media applications, $Da = K/H^2$, where H is the characteristic length. In LBM, Da should be used in lattice units, i.e., $Da = K/N^2$, where N is the number of lattices in the direction of the characteristic length. The force term can be added to D2Q9 as, $-w(k)\left[9\frac{v}{K}(uc_x + vc_y) + \frac{\beta}{\sqrt{K}}(|u|uc_x + |v|vc_y)\right]$.

Chapter 7
Multi-Relaxation Schemes

Single relaxation scheme is extensively discussed many problems were solved in the previous chapters. There is a claim that multi-relaxation schemes offer a higher stability and accuracy than the single relaxation scheme. This chapter is devoted to explain the multi-relaxation-time scheme.

7.1 Multi-Relaxation Method (MRT)

The collision operator can be generalized as,

$$f_i(x + c\Delta t, t + \Delta t) - f_i(x, t) = -\mathbf{\Omega}[f_i(x, t) - f_i^{\mathrm{eq}}(x, t)] \tag{7.1}$$

where $\mathbf{\Omega}$ is the collision matrix. The collision step in the velocity space is difficult to perform. It is more convenient to perform the collision process in the momentum space. Hence, Eq. 7.1 can transformed to the following form,

$$f_i(x + c\Delta t, t + \Delta t) - f_i(x, t) = -\mathbf{M}^{-1}\mathbf{S}[\mathbf{m}(x, t) - \mathbf{m}^{\mathrm{eq}}(x, t)] \tag{7.2}$$

where $\mathbf{m}(x, t)$ and $\mathbf{m}^{\mathrm{eq}}$ are vectors of moments, $\mathbf{m} = (m_0, m_1, m_2, \ldots m_n)^{\mathrm{T}}$. The relaxation matrix $\mathbf{S}$ is a diagonal matrix.

The mapping between velocity and moment spaces can be performed by linear transformation,

$$\mathbf{m} = \mathbf{M}\mathbf{f} \quad \text{and} \quad \mathbf{f} = \mathbf{M}^{-1}\mathbf{m}. \tag{7.3}$$

The matrix $\mathbf{M}$ for D2Q9 is

A. A. Mohamad, *Lattice Boltzmann Method*,
DOI: 10.1007/978-0-85729-455-5_7, © Springer-Verlag London Limited 2011

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{pmatrix}$$

The inverse of matrix $\mathbf{M}$ is

$$\mathbf{M}^{-1} = a \begin{pmatrix} 4 & -4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & -1 & -2 & 6 & -6 & 0 & 0 & 9 & 0 \\ 4 & -1 & -2 & 0 & 0 & 6 & -6 & -9 & 0 \\ 4 & -1 & -2 & -6 & 6 & 0 & 0 & 9 & 0 \\ 4 & -1 & -2 & 0 & 0 & -6 & 6 & -9 & 0 \\ 4 & 2 & 1 & 6 & 3 & 6 & 3 & 0 & 9 \\ 4 & 2 & 1 & -6 & -3 & 6 & 3 & 0 & -9 \\ 4 & 2 & 1 & -6 & -3 & -6 & -3 & 0 & 9 \\ 4 & 2 & 1 & 6 & 3 & -6 & -3 & 0 & -9 \end{pmatrix}$$

where $a = 1/36$ the moment vector $\mathbf{m}$ is

$$\mathbf{m} = (\rho, e, \epsilon, j_x, q_x, j_y, q_y, p_{xx}, p_{xy})^{\mathrm{T}}. \tag{7.4}$$

The equilibrium of the moment $\mathbf{m}^{\mathrm{eq}}$ is

$$\begin{aligned} m_0^{\mathrm{eq}} &= \rho \\ m_1^{\mathrm{eq}} &= -2\rho + 3(j_x^2 + j_y^2) \\ m_2^{\mathrm{eq}} &= \rho - 3(j_x^2 + j_y^2) \\ m_3^{\mathrm{eq}} &= j_x \\ m_4^{\mathrm{eq}} &= -j_x \\ m_5^{\mathrm{eq}} &= j_y \\ m_6^{\mathrm{eq}} &= -j_y \\ m_7^{\mathrm{eq}} &= (j_x^2 - j_y^2) \\ m_8^{\mathrm{eq}} &= j_x j_y \end{aligned} \tag{7.5}$$

where,

$$\begin{aligned} j_x &= \rho u_x = \sum_i f_i^{\mathrm{eq}} c_{ix} \\ j_y &= \rho u_y = \sum_i f_i^{\mathrm{eq}} c_{iy}. \end{aligned} \tag{7.6}$$

For problems with external force term F, the above equations can be modified as,

$$\begin{aligned} j_x &= \rho u_x = \sum_i f_i^{\mathrm{eq}} c_{ix} - F/2 \\ j_y &= \rho u_y = \sum_i f_i^{\mathrm{eq}} c_{iy} - F/2. \end{aligned} \tag{7.7}$$

The diagonal matrix $\mathbf{S}$:

$$\mathbf{S} = \begin{pmatrix} s_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & s_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & s_2 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & s_3 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & s_4 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & s_5 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & s_6 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_7 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_8 & \end{pmatrix}$$

in compact notation $\mathbf{S}$ can be written as,

$$\mathbf{S} = \mathrm{diag}(1.0, 1.4, 1.4, s3, 1.2, s5, 1.2, s7, s8) \tag{7.8}$$

where $s7 = s8 = 2/(1 + 6v)$, s_3, and s_5 are arbitrary, can be set to 1.0.

For D2Q5, the matrix $\mathbf{M}$ is:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ -4 & 1 & 1 & 1 & 1 \\ 0 & 1 & -1 & 1 & -1 \end{pmatrix}$$

the diagonal matrix $\mathbf{S}$ is:

$$\mathbf{S} = \mathrm{diag}(1, s, s, 1, 1, 1) \tag{7.9}$$

where $s = \frac{1}{5\alpha} + 0.5$, α is the diffusion coefficient (note that other elements on $\mathbf{S}$ are arbitrary, here we used 1).

For advection–diffusion problem, the equilibrium moments are:

$$\begin{aligned} m_0^{\mathrm{eq}} &= \rho \\ m_1^{\mathrm{eq}} &= \rho u_x \\ m_2^{\mathrm{eq}} &= \rho u_y \\ m_3^{\mathrm{eq}} &= 1 \\ m_4^{\mathrm{eq}} &= 1. \end{aligned} \tag{7.10}$$

In order to imply MRT the collision subroutine need to be modified. Other than that all the procedure is the same as for single relaxation method (SRM). The code for lid-driven cavity with MRT collision is given in the Appendix.

7.2 Problem

Modify the code for the lid-driven cavity, which is given in the Appendix, to solve fluid flow and heat transfer in a differentially heated cavity. Let Rayleigh number and Prandtl number set to 10^6 and 0.71, respectively. Also, try to get results for higher Rayleigh number, $10^7, 10^8$, and 10^9. Compare the results predicted by SRM.

7.3 Two-Relaxation-Time (TRT)

This scheme is developed by Ginzburg (2005, see this paper for more details). The distribution function can be split into two parts, symmetric and antisymmetric, as,

$$f_i^{\mathrm{s}} = \frac{1}{2}(f_i + f_{-i}) \tag{7.11}$$

and

$$f_i^{\mathrm{a}} = \frac{1}{2}(f_i - f_{-i}) \tag{7.12}$$

respectively. The distribution functions f_i and f_{-i} can be constructed by adding and subtracting the above equations, i.e. $f_i = f_i^{\mathrm{a}} + f_i^{\mathrm{s}}$ and $f_{-i} = f_i^{\mathrm{a}} - f_i^{\mathrm{s}}$. Also, $f_i^{\mathrm{a}} = f_{-i}^{\mathrm{a}}$ and $f_i^{\mathrm{s}} = -f_{-i}^{\mathrm{s}}$. f_{-i} is the distribution function moving opposite to f_i. The updating or collision equation without source term can be expressed as,

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) - \frac{1}{\tau_{\mathrm{s}}}(f_i^{\mathrm{s}}(x,t) - f_i^{\mathrm{seq}}) - \frac{1}{\tau_{\mathrm{a}}}(f_i^{\mathrm{a}}(x,t) - f_i^{\mathrm{aeq}}). \tag{7.13}$$

Notice that if τ_{a} is equal to τ_{s} Eq. 7.3 becomes single relation BGK scheme. The streaming process is the same as for single relation scheme. Equation 7.3 can be simplified by using Eqs. 7.1 and 7.2 as,

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) - \frac{1}{2}(\omega_{\mathrm{s}} + \omega_{\mathrm{a}})f_i^{\mathrm{seq}} - \frac{1}{2}(\omega_{\mathrm{s}} - \omega_{\mathrm{a}})f_i^{\mathrm{aeq}} \tag{7.14}$$

where $\omega_{\mathrm{a}} = 1/\tau_{\mathrm{a}}$ and $\omega_{\mathrm{s}} = 1/\tau_{\mathrm{s}}$. The symmetric relaxation coefficient, ω_{s} tunes the viscosity (Eqs. 6.7 or 6.8), i.e. for D2Q9 for example,

$$\omega_{\mathrm{s}} = \frac{1}{3\nu + 0.5}. \tag{7.15}$$

The asymmetric relaxation coefficient, ω_a, is chosen to minimize the viscosity dependence on the slip velocity, such as,

$$\omega_a = \frac{8(2-\omega_s)}{8-\omega_s}. \tag{7.16}$$

The computer code is quite similar to SRM. The reader is encouraged to solve a few previous problems using TRT method and compare the results of both schemes, i.e. TRT and SRM.

Chapter 8
Complex Flows

Many real life and engineering problems are more complex than what explained in the previous chapters. However, the materials covered in the pervious chapters were the building blocks for dealing with more complex flow. For instance, boiling and condensation take place in many industrial and powerplant systems. Flows of oil–water–solid are very common in oil extracting process and oil–sand process, water treatment by macro- and micro- bubbles, combustion in furnaces, incineration or waste material, etc., are few example to mention. The question is, how we can model and solve this kind of problems? The natural way is to incorporate complex physics in the source term. In fact, LBM is more powerful to deal with multi-phase and multi-component flows that using continuum mechanics, Navier–Stokes equation. The main reason for that is because, such problem involved thermodynamics, which can be naturally injected into LBM, while there is no simple means to combine the thermodynamics with NS. The strategy of blaming extra physics to source term is commonly used in finite volume method. Most commercial software solve the general transport equation. The general transport equation in 2-D, cartesian coordinate can be written as,

$$\frac{\partial \rho \Phi}{\partial t} + \frac{\partial \rho u \Phi}{\partial x} + \frac{\partial \rho v \Phi}{\partial y} = \frac{\partial}{\partial x}\left(\Gamma \frac{\partial \Phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\Gamma \frac{\partial \Phi}{\partial y}\right) + S \tag{8.1}$$

The continuity equation can be recovered if we set $\Phi = 1$ and $S = 0$. X-moment equation can be recovered by setting $\Phi = u$, $\Gamma = \mu$ and $S = -\frac{\partial p}{\partial x} + F_x$ and energy equation by setting $\Phi = c_p T$, $\Gamma = k$, and so on.

The source term can be incorporated in LBM as external force, as discussed before. The beauty of LBM in treating multi-phase flows is that there is no need to trace the interface between phase as in the case in NS. Hence, coding of multi-phase is much easier in LBM than in NS solver. In this book, we avoided to discuss this kind of problems for one simple reason. The method is relatively simple to imply but the reader needs background on the physics of the problem,

A. A. Mohamad, *Lattice Boltzmann Method*,
DOI: 10.1007/978-0-85729-455-5_8, © Springer-Verlag London Limited 2011

before applying the LBM. Therefore, the complex flow is left to the readers interest. However, I am sure that the reader can easily extend the pervious codes to handle complex phenomena easily provided that the reader understand the underlying physics of the problem.

Appendix A
Computer Codes

A. 1 Diffusion

A.1.1 The LBM Computer Code D1Q2

```
parameter (m=100) !m is the number of lattice nodes
real fo(0:m),f1(0:m),f2(0:m),rho(0:m),feq(0:m),x(0:m)
integer i
open(2,file='result')
dt=1.0
dx=1.0
x(0)=0.0
do i=1,m
x(i)=x(i-1)+dx
end do
csq=dx*dx/(dt*dt)
alpha=0.25
omega=1.0/(alpha/(dt*csq)+0.5)
mstep=200 ! The total number of time steps
twall=1.0 ! Left hand wall temperature
! Initial condition
do i=0,m
rho(i)=0.0 ! Initial value of the domain temperature
f1(i)=0.5*rho(i)
f2(i)=0.5*rho(i)
end do
do kk=1,mstep
```

```
! main loop
! collision process:
do i=0,m
rho(i)=f1(i)+f2(i)
feq(i)=0.5*rho(i)
! since k1=k2=0.5, then feq1=feq2=feq
f1(i)=(1.-omega)*f1(i)+omega*feq(i)
f2(i)=(1.-omega)*f2(i)+omega*feq(i)
end do
!Streaming process:
do i=1,m-1
f1(m-i)=f1(m-i-1) ! f1 streaming
f2(i-1)=f2(i) ! f2 streaming
end do
! Boundary condition
f1(0)=twall-f2(0)! constant temperature boundary condition, x=0
f1(m)=f1(m-1) ! adiabatic boundary condition, x=L
f2(m)=f2(m-1)! adiabatic boundary condition, x=L
end do
! end of the main loop
do i=0,m
    write(2,*)x(i), rho(i)
end do
stop
end
      The Finite Difference Code:
      parameter (m=100)
real dens,fo(0:m),f(0:m)
integer i
open(2,file='finitrs')
    dx=1.0
dt=0.500
alpha=0.25
mstep=400
do i=0,m
fo(i)=0.0
end do
fo(0)=1.0 !initial condition for old value of f at x=0.
f(0)=1.0 ! initial condition for updated value of f at x=0.
fo(m)=fo(m-1) ! initial condition for old value of f at x=L
f(m)=f(m-1) ! initial condition for updated value of f at x=L
```

```
do kk=1,mstep
! main loop
do i=1,m-1
f(i)=fo(i)+dt*alpha*(fo(i+1)-2.*fo(i)+fo(i-1))/(dx*dx)
end do
do i=1,m-1
fo(i)=f(i) ! updating
end do
fo(m)=f(m-1) ! updating the boundary condition at x=L
end do
! end of the main loop
x=0.0
do i=0,m
write(2,*)x,f(i)
x=x+dx
end do
stop
end
```

A.1.2 The LBM Code (2DQ4)

The LBM computer code for the above problem is as follows:

```
=========================================================
! LBM Code for 2-D, diffusion problems, D2Q5
parameter (n=100,m=100)

real f1(0:n,0:m),f2(0:n,0:m),f3(0:n,0:m),f4(0:n,0:m)

real rho(0:n,0:m),feq,x(0:n),y(0:m)

integer i

    open(2,file='results')
open(3,file='midplbm')
!
dx=1.0
dy=dx
dt=1.0
x(0)=0.
y(0)=0.0
do i=1,n
```

```
    x(i)=x(i-1)+dx
end do
    do j=1,m
    y(j)=y(j-1)+dy
end do
    csq=dx*dx/(dt*dt)
alpha=0.25
omega=1.0/(2.*alpha/(dt*csq)+0.5)
mstep=400
do j=0,m
    do i=0,n
    rho(i,j)=0.0              ! initial values of the dependent variable
    end do
    end do
    do j=0,m
    do i=0,n
    f1(i,j)=0.25*rho(i,j)
f2(i,j)=0.25*rho(i,j)
f3(i,j)=0.25*rho(i,j)
f4(i,j)=0.25*rho(i,j)
end do
    end do
    do kk=1,mstep
    do j=0,m
    do i=0,n
    feq=0.25*rho(i,j)
f1(i,j)=omega*feq+(1.-omega)*f1(i,j)
f2(i,j)=omega*feq+(1.-omega)*f2(i,j)
f3(i,j)=omega*feq+(1.-omega)*f3(i,j)
f4(i,j)=omega*feq+(1.-omega)*f4(i,j)
end do
    end do
    ! Streaming
```

```
    do j=0,m
    do i=1,n
    f1(n-i,j)=f1(n-i-1,j)
f2(i-1,j)=f2(i,j)
end do
    end do
    do i=0,n
    do j=1,m
    f3(i,m-j)=f3(i,m-j-1)
f4(i,j-1)=f4(i,j)
end do
    end do
    ! Boundary conditions
    do j=1,m
    f1(0,j)=0.5-f2(0,j)
f3(0,j)=0.5-f4(0,j)
f1(n,j)=0.0
f2(n,j)=0.0
f3(n,j)=0.0
f4(n,j)=0.0
end do
    do i=1,n
    f1(i,m)=0.0
f2(i,m)=0.0
f3(i,m)=0.0
f4(i,m)=0.0
f1(i,0)=f1(i,1)
f2(i,0)=f2(i,1)
f3(i,0)=f3(i,1)
f4(i,0)=f4(i,1)
end do
    do j=0,m
    do i=0,n
    rho(i,j)=f1(i,j)+f2(i,j)+f3(i,j)+f4(i,j)
end do
```

```
    end do
    end do
! end of the main loop
    write(2,*)"VARIABLES =X, Y, T"
    write(2,*)"ZONE ","I=",n+1,"J=",m+1,",",",F=BLOCK"
    do j=0,m
    write(2,*)(x(i),i=0,n)
end do
    do j=0,m
    write(2,*)(y(j),i=0,n)
end do
    do j=0,m
    write(2,*)(rho(i,j),i=0,n)
end do
    do i=0,n
    write(3,*)x(i),rho(i,m/2)
end do
    stop
end
```

==========

A.1.3 The Finite Difference Code (2-D)

==========

```
    ! Finite Difference Code for 2-D
    parameter (n=100,m=100)
    real dens,fo(0:n,0:m),f(0:n,0:m),x(0:n),y(0:m)
    integer i,j
    open(2,file='fin2drs')
open(3,file='midtfdtc')
open(4,file='qfin')
open(5,file='ttfind')
!
dx=1.0
```

```
time=0.0
dy=1.0
dt=0.20
alpha=0.25
mstep=2000
do j=0,m
    do i=0,n
    fo(i,j)=0.0
end do
    end do
    do j=0,n
    fo(0,j)=1.0
f(0,j)=1.0
fo(n,j)=0.0
f(n,j)=0.0
end do
    do i=0,n
    fo(i,0)=0.0
f(i,0)=0.0
! adiabatic bottom boundary
    fo(i,0)=fo(i,1)
f(i,0)=f(i,1)
! zero temp bottom boundary
    fo(i,m)=0.0
f(i,m)=0.0
end do
    do kk=1,mstep
    do j=1,m-1
    do i=1,n-1
    termx=(fo(i+1,j)+fo(i-1,j))/(dx*dx)
termy=(fo(i,j+1)+fo(i,j-1))/(dy*dy)
dd=1./(dx*dx)+1./(dy*dy)
f(i,j)=fo(i,j)+dt*alpha*(termx+termy-2.0*fo(i,j)*dd)
end do
    end do
```

```
    !
do j=1,m-1
    do i=1,n-1
    fo(i,j)=f(i,j)
end do
    end do
    do i=0,n
    ! adiabatic bottom boundary
    f(i,0)=f(i,1)
fo(i,0)=f(i,1)
! zero temp. bottom boundary
    ! f(i,0)=0.0
    ! fo(i,0)=0.0
    end do
    time=time+dt
write(5,*)time,f(5,m/2)
end do
    x(0)=0.0
do i=1,n
    x(i)=x(i-1)+dx
end do
    y(0)=0.0
do j=1,m
    y(j)=y(j-1)+dy
end do
    write(2,*)"VARIABLES =X, Y, T"
    write(2,*)"ZONE ","I=",n+1,"J=",m+1,",","F=BLOCK"
    do j=0,m
    write(2,*)(x(i),i=0,n)
end do
    do j=0,m
    write(2,*)(y(j),i=0,n)
```

```
end do
    do j=0,m
    write(2,*)(f(i,j),i=0,n)
end do
    do j=0,m
    q=f(0,j)-f(1,j)
write(4,*)y(j),q
end do
    do i=0,n
    write(3,*)i,f(i,m/2)
end do
    stop
end
```

==========

The LBM Code for D2Q9 for diffusion problem

==========

```
    ! The LBM code
    parameter (n=100,m=100)
    real f(0:8,0:n,0:m),feq
    real rho(0:n,0:m),x(0:n),y(0:m)
    real w(0:8)
    integer i
    open(2,file='Qresu')
open(3,file='midtlbmtc')
!
dx=1.0
dy=dy
x(0)=0.0
y(0)=0.0
do i=1,n
    x(i)=x(i-1)+dx
end do
```

```
    do j=1,m
    y(j)=y(j-1)+dy
end do
    dt=1.0
tw=1.0
alpha=0.25
csq=(dx*dx)/(dt*dt)
omega=1.0/(3.*alpha/(csq*dt)+0.5)
mstep=400
w(0)=4./9.
do i=1,4
    w(i)=1./9.
end do
    do i=5,8
    w(i)=1./36.
end do
    do j=0,m
    do i=0,n
    rho(i,j)=0.0 ! initial field
    end do
    end do
    do j=0,m
    do i=0,n
    do k=0,8
    f(k,i,j)=w(k)*rho(i,j)
if(i.eq.0) f(k,i,j)=w(k)*tw
    end do
    end do
    end do
    do kk=1,mstep
    do j=0,m
    do i=0,n
```

```
    sum=0.0
do k=0,8
    sum=sum+f(k,i,j)
end do
    rho(i,j)=sum
end do
    end do
    print *,rho(0,m/2)
    do j=0,m
    do i=0,n
    do k=0,8
    feq=w(k)*rho(i,j)
f(k,i,j)=omega*feq+(1.-omega)*f(k,i,j)
end do
    end do
    end do
    ! streaming
    do j=m,0,-1
    do i=0,n
    f(2,i,j)=f(2,i,j-1)
f(6,i,j)=f(6,i+1,j-1)
end do
    end do
    do j=m,0,-1
    do i=n,0,-1
    f(1,i,j)=f(1,i-1,j)
f(5,i,j)=f(5,i-1,j-1)
end do
    end do
    do j=0,m
    do i=n,0,-1
```

```
    f(4,i,j)=f(4,i,j+1)
f(8,i,j)=f(8,i-1,j+1)
end do

    end do

    do j=0,m

    do i=0,n

    f(3,i,j)=f(3,i+1,j)
f(7,i,j)=f(7,i+1,j+1)
end do

    end do

    ! Boundary conditions

    do j=0,m

    f(1,0,j)=w(1)*tw+w(3)*tw-f(3,0,j)
f(5,0,j)=w(5)*tw+w(7)*tw-f(7,0,j)
f(8,0,j)=w(8)*tw+w(6)*tw-f(6,0,j)
f(3,n,j)=-f(1,n,j)
f(6,n,j)=-f(8,n,j)
f(7,n,j)=-f(5,n,j)
end do

    do i=0,n

    f(4,i,m)=-f(2,i,m)
f(7,i,m)=-f(5,i,m)
f(8,i,m)=-f(6,i,m)
f(1,i,0)=f(1,i,1)
f(2,i,0)=f(2,i,1)
f(3,i,0)=f(3,i,1)
f(4,i,0)=f(4,i,1)
f(5,i,0)=f(5,i,1)
f(6,i,0)=f(6,i,1)
f(7,i,0)=f(7,i,1)
f(8,i,0)=f(8,i,1)
end do

    end do

    do j=0,m

    do i=0,n
```

```
    sum=0.0
do k=0,8
    sum=sum+f(k,i,j)
end do
    rho(i,j)=sum
end do
    end do
    print *, rho(0,m/2)
    write(2,*)"VARIABLES =X, Y, T"
    write(2,*)"ZONE ","I=",n+1,"J=",m+1,",","F=BLOCK"
    do j=0,m
    write(2,*)(x(i),i=0,n)
end do
    do j=0,m
    write(2,*)(y(j),i=0,n)
end do
    do j=0,m
    write(2,*)(rho(i,j),i=0,n)
end do
    do i=0,n
    write(3,*)x(i),rho(i,m/2)
end do
    stop
end
```

==========

A.1.4 Chapter 4, Advection-Diffusion

The Finite Difference Code

==========

```
    ! Finite Difference code for 1-D advection-diffusion problem.
    parameter (n=200)
real dens,fo(0:n),f(0:n)
```

```
integer i
open(2,file='finitrs')
!
dx=0.500
dt=0.2500
u=0.10
alpha=0.25
mstep=1600
do i=0,n
fo(i)=0.0
end do
fo(0)=1.0
f(0)=1.0
fo(n)=0.0
f(n)=0.0
do kk=1,mstep
do i=1,n-1
adv=dt*u*(fo(i)-fo(i-1))/dx
f(i)=fo(i)+dt*alpha*(fo(i+1)-2.*fo(i)+fo(i-1))/(dx*dx)-adv
end do
!
do i=1,n-1
fo(i)=f(i)
end do
end do
x=0.0
do i=0,n
write(2,*)x,f(i)
x=x+dx
end do
stop
end
```

==========

Note that the value of dt should be less than or equal to $1./[(2alpha)/(dx*dx)+u/dx]$ to satisfy stability criterion.

LBM Code for 1-D, Advection-Diffusion problem, Chapter 4

==========

```
! LBM for 1-D advection-diffusion
parameter (m=100) !m is the number of lattice nodes
```

```
real dens,fo(0:m),f1(0:m),f2(0:m),rho(0:m),feq1,x(0:m),feq2
integer i
open(2,file='result')
u=0.1
dt=1.0
dx=1.0
x(0)=0.0
do i=1,m
x(i)=x(i-1)+dx
end do
ck=dx/dt
csq=ck*ck
alpha=0.25
omega=1.0/(alpha/(dt*csq)+0.5)
mstep=400 ! The total number of time steps
twall=1.0 ! Left hand wall temperature
! Initial condition
do i=0,m
rho(i)=0.0 ! Initial value of the domain temperature
f1(i)=0.5*rho(i)
f2(i)=0.5*rho(i)
end do
do kk=1,mstep ! main loop
! collision process:
do i=0,m
rho(i)=f1(i)+f2(i)
feq1=0.5*rho(i)*(1.0+u/ck)              ! extra term added to simulate
advection
feq2=0.5*rho(i)*(1.0-u/ck)                              ! w1=w2=0.5
f1(i)=(1.-omega)*f1(i)+omega*feq1
f2(i)=(1.-omega)*f2(i)+omega*feq2
end do
! Streaming process:
do i=1,m-1
f1(m-i)=f1(m-i-1)          ! f1 streaming
f2(i-1)=f2(i)               ! f2 streaming
end do
! Boundary condition
f1(0)=twall-f2(0)                        ! constant temperature boundary
condition, x=0
f1(m)=f1(m-1)                          ! adiabatic boundary condition, x=L
```

```
f2(m)=f2(m-1)                    ! adiabatic boundary condition, x=L
end do                           ! end of the main loop
do i=0,m
   write(2,*)x(i), rho(i)
end do
stop
end
```

==========

Advection-Diffusion D2Q4 Code

==========

```
   ! LBM Code for 2-D, Advection-diffusion problems, D2Q4
   parameter (n=100,m=100)
real f1(0:n,0:m),f2(0:n,0:m),f3(0:n,0:m),f4(0:n,0:m)
real feq1,feq2,feq3,feq4
real rho(0:n,0:m),x(0:n),y(0:m)
integer i
open(2,file='results')
open(3,file='midplbm')
!
u=0.1
v=0.2
dx=1.0
dy=dx
dt=1.0
x(0)=0
y(0)=0.0
do i=1,n
x(i)=x(i-1)+dx
end do
do j=1,m
y(j)=y(j-1)+dy
end do
ck=dx/dt
csq=ck*ck
alpha=1.0
omega=1.0/(2.*alpha/(dt*csq)+0.5)
mstep=200
do j=0,m
do i=0,n
rho(i,j)=0.0 ! initial values of the dependent variable
```

```
end do
end do
do j=0,m
do i=0,n
f1(i,j)=0.25*rho(i,j)
f2(i,j)=0.25*rho(i,j)
f3(i,j)=0.25*rho(i,j)
f4(i,j)=0.25*rho(i,j)
end do
end do
do kk=1,mstep
do j=0,m
do i=0,n
feq1=0.25*rho(i,j)*(1.0+2.0*u/ck)
feq2=0.25*rho(i,j)*(1.0-2.0*u/ck)
feq3=0.25*rho(i,j)*(1.0+2.0*v/ck)
feq4=0.25*rho(i,j)*(1.0-2.0*v/ck)
f1(i,j)=omega*feq1+(1.-omega)*f1(i,j)
f2(i,j)=omega*feq2+(1.-omega)*f2(i,j)
f3(i,j)=omega*feq3+(1.-omega)*f3(i,j)
f4(i,j)=omega*feq4+(1.-omega)*f4(i,j)
end do
end do
! Streaming
do j=0,m
do i=1,n
f1(n-i,j)=f1(n-i-1,j)
f2(i-1,j)=f2(i,j)
end do
end do
do i=0,n
do j=1,m
f3(i,m-j)=f3(i,m-j-1)
f4(i,j-1)=f4(i,j)
end do
end do
! Boundary conditions
do j=1,m
f1(0,j)=0.5-f2(0,j)
f3(0,j)=0.5-f4(0,j)
f1(n,j)=0.0
```

```
f2(n,j)=0.0
f3(n,j)=0.0
f4(n,j)=0.0
end do
do i=1,n
f1(i,m)=0.0
f2(i,m)=0.0
f3(i,m)=0.0
f4(i,m)=0.0
f1(i,0)=f1(i,1)
f2(i,0)=f2(i,1)
f3(i,0)=f3(i,1)
f4(i,0)=f4(i,1)
end do
do j=0,m
do i=0,n
rho(i,j)=f1(i,j)+f2(i,j)+f3(i,j)+f4(i,j)
end do
end do
end do ! end of the main loop

    write(2,*)"VARIABLES =X, Y, T"
write(2,*)"ZONE ","I=",n+1,"J=",m+1,",","F=BLOCK"
do j=0,m
write(2,*)(x(i),i=0,n)
end do
do j=0,m
write(2,*)(y(j),i=0,n)
end do
do j=0,m
write(2,*)(rho(i,j),i=0,n)
end do
do i=0,n
write(3,*)x(i),rho(i,m/2)
end do
stop
end
```

==========

D2Q9 Code

```
=========================================================
    ! LBM -Advection-Diffusion D2Q9
    parameter (n=100,m=100)
real f(0:8,0:n,0:m),feq(0:8)
real rho(0:n,0:m),x(0:n),y(0:m)
real w(0:8)
integer i
open(2,file='Qresu')
open(3,file='midtlbmtc')
!
u=0.1
v=0.4
dt=1.0
dx=1.0
dy=dx
x(0)=0.0
y(0)=0.0
do i=1,n
x(i)=x(i-1)+dx
end do
do j=1,m
y(j)=y(j-1)+dy
end do
tw=1.0
alpha=1.0
ck=dx/dt
csq=ck*ck
omega=1.0/(3.*alpha/(csq*dt)+0.5)
mstep=400
w(0)=4./9.
do i=1,4
w(i)=1./9.
end do
do i=5,8
w(i)=1./36.
end do
density=0.0
do j=0,m
do i=0,n
```

```
do k=0,8
f(k,i,j)=w(k)*density
if(i.eq.0) f(k,i,j)=w(k)*tw
end do
    end do
end do
do kk=1,mstep
do j=0,m
do i=0,n
sum=0.0
do k=0,8
sum=sum+f(k,i,j)
end do
rho(i,j)=sum
end do
end do
print *,rho(0,m/2)
do j=0,m
do i=0,n

    feq(0)=w(0)*rho(i,j)
feq(1)=w(1)*rho(i,j)*(1.0+3.0*u/ck)
feq(2)=w(2)*rho(i,j)*(1.0+3.0*v/ck)
feq(3)=w(3)*rho(i,j)*(1.0-3.0*u/ck)
feq(4)=w(4)*rho(i,j)*(1.0-3.0*v/ck)
feq(5)=w(5)*rho(i,j)*(1.0+3.0*(u+v)/ck)
feq(6)=w(6)*rho(i,j)*(1.0+3.0*(-u+v)/ck)
feq(7)=w(7)*rho(i,j)*(1.0-3.0*(u+v)/ck)
feq(8)=w(8)*rho(i,j)*(1.0+3.0*(u-v)/ck)

    do k=0,8
f(k,i,j)=omega*feq(k)+(1.-omega)*f(k,i,j)
end do
end do
end do
! streaming
do j=m,0,-1
do i=0,n
f(2,i,j)=f(2,i,j-1)
f(6,i,j)=f(6,i+1,j-1)
end do
end do
```

```
do j=m,0,-1
do i=n,0,-1
f(1,i,j)=f(1,i-1,j)
f(5,i,j)=f(5,i-1,j-1)
end do
end do
do j=0,m
do i=n,0,-1
f(4,i,j)=f(4,i,j+1)
f(8,i,j)=f(8,i-1,j+1)
end do
end do
do j=0,m
do i=0,n
f(3,i,j)=f(3,i+1,j)
f(7,i,j)=f(7,i+1,j+1)
end do
end do
! Boundary conditions
! Left boundary condition, the temperature is given, tw
do j=0,m
f(1,0,j)=w(1)*tw+w(3)*tw-f(3,0,j)
f(5,0,j)=w(5)*tw+w(7)*tw-f(7,0,j)
f(8,0,j)=w(8)*tw+w(6)*tw-f(6,0,j)
end do
! Right hand boundary condition, T=0.
do j=0,m
f(6,n,j)=-f(8,n,j)
f(3,n,j)=-f(1,n,j)
f(7,n,j)=-f(5,n,j)
f(2,n,j)=-f(4,n,j)
f(0,n,j)=0.0
end do
! Top boundary conditions, T=0.0
do i=0,n
f(8,i,m)=-f(6,i,m)
f(7,i,m)=-f(5,i,m)
f(4,i,m)=-f(2,i,m)
f(1,i,m)=-f(3,i,m)
f(0,i,m)=0.0
end do
```

```
!Bottom boundary conditions, Adiabatic
! f(1,i,0)=f(1,i,1)
! f(2,i,0)=f(2,i,1)
! f(3,i,0)=f(3,i,1)
! f(4,i,0)=f(4,i,1)
! f(5,i,0)=f(5,i,1)
! f(6,i,0)=f(6,i,1)
! f(7,i,0)=f(7,i,1)
! f(8,i,0)=f(8,i,1)
! T=0.0
do i=0,n
f(2,i,0)=-f(4,i,0)
f(6,i,0)=-f(8,i,0)
f(5,i,0)=-f(7,i,0)
f(1,i,0)=-f(3,i,0)
f(0,i,0)=0.0
end do
end do
do j=0,m
do i=0,n
sum=0.0
do k=0,8
sum=sum+f(k,i,j)
end do
rho(i,j)=sum
end do
end do
print *, rho(0,m/2)
write(2,*)"VARIABLES =X, Y, T"
write(2,*)"ZONE ","I=",n+1,"J=",m+1,",","F=BLOCK"
do j=0,m
write(2,*)(x(i),i=0,n)
end do
do j=0,m
write(2,*)(y(j),i=0,n)
end do
do j=0,m
write(2,*)(rho(i,j),i=0,n)
end do
do i=0,n
write(3,*)x(i),rho(i,m/2)
```

```
end do
stop
end
=======================================================
```

Two-Dimensional, Advection Diffusion, FDM Code

```
=======================================================
    parameter (n=100,m=100)
real dens,fo(0:n,0:m),f(0:n,0:m),x(0:n),y(0:m)
integer i,j
open(2,file='fin2drs')
open(3,file='finitial')
open(4,file='qfin')
open(5,file='ttfind')
!
dx=1.0
time=0.0
dy=1.0
dt=0.20
u=0.1
v=0.2
alpha=1.0
mstep=1000
do j=0,m
do i=0,n
fo(i,j)=0.0
end do
end do
do j=0,n
fo(0,j)=1.0
f(0,j)=1.0
fo(n,j)=0.0
f(n,j)=0.0
end do
do i=0,n
fo(i,0)=0.0
f(i,0)=0.0
! adiabatic bottom boundary
fo(i,0)=fo(i,1)
```

```
f(i,0)=f(i,1)
! zero temp top boundary
fo(i,m)=0.0
f(i,m)=0.0
end do
do kk=1,mstep
do j=1,m-1
do i=1,n-1
termx=(fo(i+1,j)+fo(i-1,j))/(dx*dx)
termy=(fo(i,j+1)+fo(i,j-1))/(dy*dy)
dd=1./(dx*dx)+1./(dy*dy)
advx=u*(fo(i,j)-fo(i-1,j))/dx
advy=v*(fo(i,j)-fo(i,j-1))/dy
advt=dt*(advx+advy)
f(i,j)=fo(i,j)+dt*alpha*(termx+termy-2.0*fo(i,j)*dd)-advt
end do
end do
!
do j=1,m-1
do i=1,n-1
fo(i,j)=f(i,j)
end do
end do
do i=0,n
! adiabatic bottom boundary
! f(i,0)=f(i,1)
! fo(i,0)=f(i,1)
! zero temp. bottom boundary
f(i,0)=0.0
fo(i,0)=0.0
end do
! adiabatic right hand boundary
do j=0,m
f(n,j)=f(n-1,j)
fo(n,j)=fo(n-1,j)
end do
time=time+dt
write(5,*)time,f(5,m/2)
end do
x(0)=0.0
do i=1,n
```

```
x(i)=x(i-1)+dx
end do
y(0)=0.0
do j=1,m
y(j)=y(j-1)+dy
end do
write(2,*)"VARIABLES =X, Y, T"
write(2,*)"ZONE ","I=",n+1,"J=",m+1,",","F=BLOCK"
do j=0,m
write(2,*)(x(i),i=0,n)
end do
do j=0,m
write(2,*)(y(j),i=0,n)
end do
do j=0,m
write(2,*)(f(i,j),i=0,n)
end do
do j=0,m
q=f(0,j)-f(1,j)
write(4,*)y(j),q
end do
stop
end
```

==========

The Computer Code For a Lid-Driven Cavity (Chapter 5)

==========

```
! computer code for lid-driven cavity

parameter (n=100,m=100)
real f(0:8,0:n,0:m)
real feq(0:8,0:n,0:m),rho(0:n,0:m)
real w(0:8), cx(0:8),cy(0:8)
real u(0:n,0:m), v(0:n,0:m)
integer i
open(2,file='uvfield')
open(3,file='uvely')
open(4,file='vvelx')
open(8,file='timeu')
```

```
!
uo=0.10
sumvelo=0.0
rhoo=5.00
dx=1.0
dy=dx
dt=1.0
alpha=0.01
Re=uo*m/alpha
print *, ”Re=”, Re
omega=1.0/(3.*alpha+0.5)
mstep=40000
w(0)=4./9.
do i=1,4
w(i)=1./9.
end do
do i=5,8
w(i)=1./36.
end do
cx(0)=0
cx(1)=1
cx(2)=0
cx(3)=-1
cx(4)=0
cx(5)=1
cx(6)=-1
cx(7)=-1
cx(8)=1
cy(0)=0
cy(1)=0
cy(2)=1
cy(3)=0
cy(4)=-1
cy(5)=1
cy(6)=1
cy(7)=-1
cy(8)=-1
do j=0,m
do i=0,n
rho(i,j)=rhoo
u(i,j)=0.0
```

```
v(i,j)=0.0
end do
end do
do i=1,n-1
u(i,m)=uo
v(i,m)=0.0
end do
! main loop
1 do kk=1,mstep
call collesion(u,v,f,feq,rho,omega,w,cx,cy,n,m)
call streaming(f,n,m)
! ———————————————
call sfbound(f,n,m,uo)
call rhouv(f,rho,u,v,cx,cy,n,m)
print *, u(0,m/2),v(0,m/2),rho(0,m/2),u(n,m/2),v(n,m/2),rho(n,m/2)
write(8,*) kk,u(n/2,m/2),v(n/2,m/2)
END DO
! end of the main loop
call result(u,v,rho,uo,n,m)
stop
end
! end of the main program
subroutine collesion(u,v,f,feq,rho,omega,w,cx,cy,n,m)
real f(0:8,0:n,0:m)
real feq(0:8,0:n,0:m),rho(0:n,0:m)
real w(0:8), cx(0:8),cy(0:8)
real u(0:n,0:m), v(0:n,0:m)
DO i=0,n
DO j=0,m
t1=u(i,j)*u(i,j)+v(i,j)*v(i,j)
DO k=0,8
t2=u(i,j)*cx(k)+v(i,j)*cy(k)
feq(k,i,j)=rho(i,j)*w(k)*(1.0+3.0*t2+4.50*t2*t2-1.50*t1)
f(k,i,j)=omega*feq(k,i,j)+(1.-omega)*f(k,i,j)
END DO
END DO
END DO
return
end
subroutine streaming(f,n,m)
real f(0:8,0:n,0:m)
```

```
! streaming
DO j=0,m
DO i=n,1,-1 !RIGHT TO LEFT
f(1,i,j)=f(1,i-1,j)
END DO
DO i=0,n-1 !LEFT TO RIGHT
f(3,i,j)=f(3,i+1,j)
END DO
END DO
DO j=m,1,-1 !TOP TO BOTTOM
DO i=0,n
f(2,i,j)=f(2,i,j-1)
END DO
DO i=n,1,-1
f(5,i,j)=f(5,i-1,j-1)
END DO
DO i=0,n-1
f(6,i,j)=f(6,i+1,j-1)
END DO
END DO
DO j=0,m-1 !BOTTOM TO TOP
DO i=0,n
f(4,i,j)=f(4,i,j+1)
END DO
DO i=0,n-1
f(7,i,j)=f(7,i+1,j+1)
END DO
DO i=n,1,-1
f(8,i,j)=f(8,i-1,j+1)
END DO
END DO
return
end
subroutine sfbound(f,n,m,uo)
real f(0:8,0:n,0:m)
do j=0,m
! bounce back on west boundary
f(1,0,j)=f(3,0,j)
f(5,0,j)=f(7,0,j)
f(8,0,j)=f(6,0,j)
! bounce back on east boundary
```

```
f(3,n,j)=f(1,n,j)
f(7,n,j)=f(5,n,j)
f(6,n,j)=f(8,n,j)
end do
! bounce back on south boundary
do i=0,n
f(2,i,0)=f(4,i,0)
f(5,i,0)=f(7,i,0)
f(6,i,0)=f(8,i,0)
end do
! moving lid, north boundary
do i=1,n-1
rhon=f(0,i,m)+f(1,i,m)+f(3,i,m)+2.*(f(2,i,m)+f(6,i,m)+f(5,i,m))
f(4,i,m)=f(2,i,m)
f(8,i,m)=f(6,i,m)+rhon*uo/6.0
f(7,i,m)=f(5,i,m)-rhon*uo/6.0
end do
return
end
subroutine rhouv(f,rho,u,v,cx,cy,n,m)
real f(0:8,0:n,0:m),rho(0:n,0:m),u(0:n,0:m),v(0:n,0:m),cx(0:8),cy(0:8)
do j=0,m
do i=0,n
ssum=0.0
do k=0,8
ssum=ssum+f(k,i,j)
end do
rho(i,j)=ssum
end do
end do
do i=1,n
rho(i,m)=f(0,i,m)+f(1,i,m)+f(3,i,m)+2.*(f(2,i,m)+f(6,i,m)+f(5,i,m))
end do
DO i=1,n
DO j=1,m-1
usum=0.0
vsum=0.0
DO k=0,8
usum=usum+f(k,i,j)*cx(k)
vsum=vsum+f(k,i,j)*cy(k)
END DO
```

```
u(i,j)=usum/rho(i,j)
v(i,j)=vsum/rho(i,j)
END DO
END DO
return
end
subroutine result(u,v,rho,uo,n,m)
real u(0:n,0:m),v(0:n,0:m)
real rho(0:n,0:m),strf(0:n,0:m)
open(5, file='streamf')
! streamfunction calculations
strf(0,0)=0.
do i=0,n
rhoav=0.5*(rho(i-1,0)+rho(i,0))
if(i.ne.0) strf(i,0)=strf(i-1,0)-rhoav*0.5*(v(i-1,0)+v(i,0))
do j=1,m
rhom=0.5*(rho(i,j)+rho(i,j-1))
strf(i,j)=strf(i,j-1)+rhom*0.5*(u(i,j-1)+u(i,j))
end do
end do
! ————————————————————
write(2,*)"VARIABLES =X, Y, U, V, S"
write(2,*)"ZONE ","I=",n+1,"J=",m+1,",","F=BLOCK"
do j=0,m
write(2,*)(i,i=0,n)
end do
do j=0,m
write(2,*)(j,i=0,n)
end do
do j=0,m
write(2,*)(u(i,j),i=0,n)
end do
do j=0,m
write(2,*)(v(i,j),i=0,n)
end do
do j=0,m
write(2,*)(strf(i,j),i=0,n)
end do
do j=0,m
write(3,*)j/float(m),u(n/2,j)/uo,u(n/4,j)/uo,u(3*n/4,j)/uo
end do
```

```
do i=0,n
write(4,*) i/float(n),v(i,m/2)/uo
end do
return
end
!============end of the program
```

A.1.5. Chapter Six

A.1.6 Computer Code:

Complete computer code is given below for a heated moving lid cavity

```
=======================================================
parameter (n=100,m=100)
real f(0:8,0:n,0:m)
real feq(0:8,0:n,0:m),rho(0:n,0:m)
real w(0:8), cx(0:8),cy(0:8)
real u(0:n,0:m), v(0:n,0:m)
real g(0:8,0:n,0:m), geq(0:8,0:n,0:m),th(0:n,0:m)
integer i
open(2,file='uvfield')
open(3,file='uvely')
open(4,file='vvelx')
!
cx(:)=(/0.0,1.0,0.0,-1.0,0.0,1.0,-1.0,-1.0,1.0/)
cy(:)=(/0.0,0.0,1.0,0.0,-1.0,1.0,1.0,-1.0,-1.0/)
w(:)=(/4./9.,1./9.,1./9.,1./9.,1./9.,1./36.,1./36.,1./36.,1./36./)
uo=0.2
sumvelo=0.0
rhoo=5.00
dx=1.0
dy=dx
dt=1.0
tw=1.0
th=0.0
g=0.0
visco=0.02
```

```
pr=0.71
alpha=visco/pr
Re=uo*m/alpha
print *, ”Re=”, Re
omega=1.0/(3.*visco+0.5)
omegat=1.0/(3.*alpha+0.5)
mstep=20000
do j=0,m
do i=0,n
rho(i,j)=rhoo
u(i,j)=0.0
v(i,j)=0.0
end do
end do
do i=1,n-1
u(i,m)=uo
v(i,m)=0.0
end do
! main loop
1 do kk=1,mstep
call collesion(u,v,f,feq,rho,omega,w,cx,cy,n,m)
call streaming(f,n,m)
call sfbound(f,n,m,uo)
call rhouv(f,rho,u,v,cx,cy,n,m)
do j=0,m
do i=0,n
sum=0.0
do k=0,8
sum=sum+g(k,i,j)
th(i,j)=sum
end do
end do
end do
! collestion for scalar
call collt(u,v,g,geq,th,omegat,w,cx,cy,n,m)
! streaming for scalar
call streaming(g,n,m)
call gbound(g,tw,w,n,m)
print *, th(n/2,m/2),v(0,m/2),rho(0,m/2),u(n,m/2),v(n,m/2),rho(n,m/2)
END DO
! end of the main loop
```

```
call result(u,v,rho,th,uo,n,m)
stop
end
! end of the main program
subroutine collesion(u,v,f,feq,rho,omega,w,cx,cy,n,m)
real f(0:8,0:n,0:m)
real feq(0:8,0:n,0:m),rho(0:n,0:m)
real w(0:8), cx(0:8),cy(0:8)
real u(0:n,0:m), v(0:n,0:m)
DO i=0,n
DO j=0,m
t1=u(i,j)*u(i,j)+v(i,j)*v(i,j)
DO k=0,8
t2=u(i,j)*cx(k)+v(i,j)*cy(k)
feq(k,i,j)=rho(i,j)*w(k)*(1.0+3.0*t2+4.50*t2*t2-1.50*t1)
f(k,i,j)=omega*feq(k,i,j)+(1.-omega)*f(k,i,j)
END DO
END DO
END DO
return
end
subroutine collt(u,v,g,geq,th,omegat,w,cx,cy,n,m)
real g(0:8,0:n,0:m),geq(0:8,0:n,0:m),th(0:n,0:m)
real w(0:8),cx(0:8),cy(0:8)
real u(0:n,0:m),v(0:n,0:m)
do i=0,n
do j=0,m
do k=0,8
geq(k,i,j)=th(i,j)*w(k)*(1.0+3.0*(u(i,j)*cx(k)+v(i,j)*cy(k)))
g(k,i,j)=omegat*geq(k,i,j)+(1.0-omegat)*g(k,i,j)
end do
end do
end do
return
end
subroutine streaming(f,n,m)
real f(0:8,0:n,0:m)
! streaming
DO j=0,m
DO i=n,1,-1 !RIGHT TO LEFT
f(1,i,j)=f(1,i-1,j)
```

```
END DO
DO i=0,n-1 !LEFT TO RIGHT
f(3,i,j)=f(3,i+1,j)
END DO
END DO
DO j=m,1,-1 !TOP TO BOTTOM
DO i=0,n
f(2,i,j)=f(2,i,j-1)
END DO
DO i=n,1,-1
f(5,i,j)=f(5,i-1,j-1)
END DO
DO i=0,n-1
f(6,i,j)=f(6,i+1,j-1)
END DO
END DO
DO j=0,m-1 !BOTTOM TO TOP
DO i=0,n
f(4,i,j)=f(4,i,j+1)
END DO
DO i=0,n-1
f(7,i,j)=f(7,i+1,j+1)
END DO
DO i=n,1,-1
f(8,i,j)=f(8,i-1,j+1)
END DO
END DO
return
end
subroutine sfbound(f,n,m,uo)
real f(0:8,0:n,0:m)
do j=0,m
! bounce back on west boundary
f(1,0,j)=f(3,0,j)
f(5,0,j)=f(7,0,j)
f(8,0,j)=f(6,0,j)
! bounce back on east boundary
f(3,n,j)=f(1,n,j)
f(7,n,j)=f(5,n,j)
f(6,n,j)=f(8,n,j)
end do
```

```
! bounce back on south boundary
do i=0,n
f(2,i,0)=f(4,i,0)
f(5,i,0)=f(7,i,0)
f(6,i,0)=f(8,i,0)
end do
! moving lid, north boundary
do i=1,n-1
rhon=f(0,i,m)+f(1,i,m)+f(3,i,m)+2.*(f(2,i,m)+f(6,i,m)+f(5,i,m))
f(4,i,m)=f(2,i,m)
f(8,i,m)=f(6,i,m)+rhon*uo/6.0
f(7,i,m)=f(5,i,m)-rhon*uo/6.0
end do
return
end
subroutine gbound(g,tw,w,n,m)
real g(0:8,0:n,0:m)
real w(0:8)
! Boundary conditions
! West boundary condition, T=0.
do j=0,m
g(1,0,j)=-g(3,0,j)
g(5,0,j)=-g(7,0,j)
g(8,0,j)=-g(6,0,j)
end do
! East boundary condition, T=0.
do j=0,m
g(6,n,j)=-g(8,n,j)
g(3,n,j)=-g(1,n,j)
g(7,n,j)=-g(5,n,j)
g(2,n,j)=-g(4,n,j)
g(0,n,j)=0.0
end do
! Top boundary conditions, T=tw=1.0
do i=0,n
g(8,i,m)=tw*(w(8)+w(6))-g(6,i,m)
g(7,i,m)=tw*(w(7)+w(5))-g(5,i,m)
g(4,i,m)=tw*(w(4)+w(2))-g(2,i,m)
g(1,i,m)=tw*(w(1)+w(3))-g(3,i,m)
end do
!Bottom boundary conditions, Adiabatic
```

```
do i=0,n
g(1,i,0)=g(1,i,1)
g(2,i,0)=g(2,i,1)
g(3,i,0)=g(3,i,1)
g(4,i,0)=g(4,i,1)
g(5,i,0)=g(5,i,1)
g(6,i,0)=g(6,i,1)
g(7,i,0)=g(7,i,1)
g(8,i,0)=g(8,i,1)
end do
return
end
subroutine tcalcu(g,th,n,m)
real g(0:8,0:n,0:m),th(0:n,0:m)
do j=1,m-1
do i=1,n-1
ssumt=0.0
do k=0,8
ssumt=ssumt+g(k,i,j)
end do
th(i,j)=ssumt
end do
end do
return
end
subroutine rhouv(f,rho,u,v,cx,cy,n,m)
real f(0:8,0:n,0:m),rho(0:n,0:m),u(0:n,0:m),v(0:n,0:m),cx(0:8),cy(0:8)
do j=0,m
do i=0,n
ssum=0.0
do k=0,8
ssum=ssum+f(k,i,j)
end do
rho(i,j)=ssum
end do
end do
DO i=1,n
DO j=1,m-1
usum=0.0
vsum=0.0
DO k=0,8
```

```
usum=usum+f(k,i,j)*cx(k)
vsum=vsum+f(k,i,j)*cy(k)
END DO
u(i,j)=usum/rho(i,j)
v(i,j)=vsum/rho(i,j)
END DO
END DO
return
end
subroutine result(u,v,rho,th,uo,n,m)
real u(0:n,0:m),v(0:n,0:m),th(0:n,0:m)
real rho(0:n,0:m),strf(0:n,0:m)
open(5, file='streamf')
open(7,file='tprof')
! streamfunction calculations
strf(0,0)=0.
do i=0,n
rhoav=0.5*(rho(i-1,0)+rho(i,0))
if(i.ne.0) strf(i,0)=strf(i-1,0)-rhoav*0.5*(v(i-1,0)+v(i,0))
do j=1,m
rhom=0.5*(rho(i,j)+rho(i,j-1))
strf(i,j)=strf(i,j-1)+rhom*0.5*(u(i,j-1)+u(i,j))
end do
end do
! ————————————————————
write(2,*)"VARIABLES =X, Y, U, V, T"
write(2,*)"ZONE ","I=",n+1,"J=",m+1,",",",F=BLOCK"
do j=0,m
write(2,*)(i,i=0,n)
end do
do j=0,m
write(2,*)(j,i=0,n)
end do
do j=0,m
write(2,*)(u(i,j),i=0,n)
end do
do j=0,m
write(2,*)(v(i,j),i=0,n)
end do
do j=0,m
write(2,*)(th(i,j),i=0,n)
```

```
end do
write(5,*)"VARIABLES =X, Y, ST"
write(5,*)"ZONE ","I=",n+1,"J=",m+1,",","F=BLOCK"
do j=0,m
write(5,*)(i,i=0,n)
end do
do j=0,m
write(5,*)(j,i=0,n)
end do
do j=0,m
write(5,*)(strf(i,j),i=0,n)
end do
do j=0,m
write(3,*)j/float(m),u(5,j)/uo,u(n/2,j)/uo,u(n-10,j)/uo
write(7,*)j/float(m),th(n/4,j),th(n/2,j),th(3*n/4,j)
end do
do i=0,n
write(4,*) i/float(n),v(i,m/2)/uo
end do
return
end
```

==========

A.1.7 Computer Code

The full computer code is:

==========

```
parameter (n=1000,m=50)
real f(0:8,0:n,0:m)
real feq(0:8,0:n,0:m),rho(0:n,0:m)
real w(0:8), cx(0:8),cy(0:8)
real u(0:n,0:m), v(0:n,0:m)
real g(0:8,0:n,0:m), geq(0:8,0:n,0:m),th(0:n,0:m)
integer i
open(2,file='uvfield')
open(3,file='uvely')
open(4,file='vvelx')
```

```
!
cx(:)=(/0.0,1.0,0.0,-1.0,0.0,1.0,-1.0,-1.0,1.0/)
cy(:)=(/0.0,0.0,1.0,0.0,-1.0,1.0,1.0,-1.0,-1.0/)
w(:)=(/4./9.,1./9.,1./9.,1./9.,1./9.,1./36.,1./36.,1./36.,1./36./)
uo=0.1
sumvelo=0.0
rhoo=5.00
dx=1.0
dy=dx
dt=1.0
tw=1.0
th=0.0
g=0.0
visco=0.05
pr=3.8
alpha=visco/pr
Re=uo*m/alpha
print *, ”Re=”, Re
omega=1.0/(3.*visco+0.5)
omegat=1.0/(3.*alpha+0.5)
mstep=10000
do j=0,m
do i=0,n
rho(i,j)=rhoo
u(i,j)=0.0
v(i,j)=0.0
end do
end do
do j=1,m-1
u(0,j)=uo
v(0,j)=0.0
end do
! main loop
1 do kk=1,mstep
call collesion(u,v,f,feq,rho,omega,w,cx,cy,n,m)
call streaming(f,n,m)
call sfbound(f,n,m,uo)
call rhouv(f,rho,u,v,cx,cy,n,m)
! ———————————
do j=0,m
do i=0,n
```

```
sum=0.0
do k=0,8
sum=sum+g(k,i,j)
th(i,j)=sum
end do
end do
end do
! collestion for scalar
call collt(u,v,g,geq,th,omegat,w,cx,cy,n,m)
! streaming for scalar
call streaming(g,n,m)
call gbound(g,tw,w,n,m)
print *, th(n/2,m/2),v(0,m/2),rho(0,m/2),u(n,m/2),v(n,m/2),rho(n,m/2)
END DO
! end of the main loop
call result(u,v,th,uo,n,m)
stop
end
! end of the main program
subroutine collesion(u,v,f,feq,rho,omega,w,cx,cy,n,m)
real f(0:8,0:n,0:m)
real feq(0:8,0:n,0:m),rho(0:n,0:m)
real w(0:8), cx(0:8),cy(0:8)
real u(0:n,0:m), v(0:n,0:m)
DO i=0,n
DO j=0,m
t1=u(i,j)*u(i,j)+v(i,j)*v(i,j)
DO k=0,8
t2=u(i,j)*cx(k)+v(i,j)*cy(k)
feq(k,i,j)=rho(i,j)*w(k)*(1.0+3.0*t2+4.50*t2*t2-1.50*t1)
f(k,i,j)=omega*feq(k,i,j)+(1.-omega)*f(k,i,j)
END DO
END DO
END DO
return
end
subroutine collt(u,v,g,geq,th,omegat,w,cx,cy,n,m)
real g(0:8,0:n,0:m),geq(0:8,0:n,0:m),th(0:n,0:m)
real w(0:8),cx(0:8),cy(0:8)
real u(0:n,0:m),v(0:n,0:m)
do i=0,n
```

```
do j=0,m
do k=0,8
geq(k,i,j)=th(i,j)*w(k)*(1.0+3.0*(u(i,j)*cx(k)+v(i,j)*cy(k)))
g(k,i,j)=omegat*geq(k,i,j)+(1.0-omegat)*g(k,i,j)
end do
end do
end do
return
end
subroutine streaming(f,n,m)
real f(0:8,0:n,0:m)
! streaming
DO j=0,m
DO i=n,1,-1 !RIGHT TO LEFT
f(1,i,j)=f(1,i-1,j)
END DO
DO i=0,n-1 !LEFT TO RIGHT
f(3,i,j)=f(3,i+1,j)
END DO
END DO
DO j=m,1,-1 !TOP TO BOTTOM
DO i=0,n
f(2,i,j)=f(2,i,j-1)
END DO
DO i=n,1,-1
f(5,i,j)=f(5,i-1,j-1)
END DO
DO i=0,n-1
f(6,i,j)=f(6,i+1,j-1)
END DO
END DO
DO j=0,m-1 !BOTTOM TO TOP
DO i=0,n
f(4,i,j)=f(4,i,j+1)
END DO
DO i=0,n-1
f(7,i,j)=f(7,i+1,j+1)
END DO
DO i=n,1,-1
f(8,i,j)=f(8,i-1,j+1)
END DO
```

```
END DO
return
end
subroutine sfbound(f,n,m,uo)
real f(0:8,0:n,0:m)
do j=0,m
! flow in on west boundary
rhow=(f(0,0,j)+f(2,0,j)+f(4,0,j)+2.*(f(3,0,j)+f(6,0,j)+f(7,0,j)))/(1.-uo)
f(1,0,j)=f(3,0,j)+2.*rhow*uo/3.
f(5,0,j)=f(7,0,j)+rhow*uo/6.
f(8,0,j)=f(6,0,j)+rhow*uo/6.
end do
! bounce back on south boundary
do i=0,n
f(2,i,0)=f(4,i,0)
f(5,i,0)=f(7,i,0)
f(6,i,0)=f(8,i,0)
end do
! bounce back, north boundary
do i=0,n
f(4,i,m)=f(2,i,m)
f(8,i,m)=f(6,i,m)
f(7,i,m)=f(5,i,m)
end do
! accont for open boundary condition at the outlet
do j=1,m
f(1,n,j)=2.*f(1,n-1,j)-f(1,n-2,j)
f(5,n,j)=2.*f(5,n-1,j)-f(5,n-2,j)
f(8,n,j)=2.*f(8,n-1,j)-f(8,n-2,j)
end do
return
end
subroutine gbound(g,tw,w,n,m)
real g(0:8,0:n,0:m)
real w(0:8)
! Boundary conditions
! Left boundary condition, the temperature is given, tw
do j=0,m
g(1,0,j)=-g(3,0,j)
g(5,0,j)=-g(7,0,j)
g(8,0,j)=-g(6,0,j)
```

```
end do
! Right hand boundary condition, open
do j=0,m
g(6,n,j)=2.*g(6,n-1,j)-g(6,n-2,j)
g(3,n,j)=2.*g(3,n-1,j)-g(3,n-2,j)
g(7,n,j)=2.*g(7,n-1,j)-g(7,n-2,j)
g(2,n,j)=2.*g(2,n-1,j)-g(2,n-2,j)
g(0,n,j)=2.*g(0,n-1,j)-g(0,n-2,j)
g(1,n,j)=2.*g(1,n-1,j)-g(1,n-2,j)
g(4,n,j)=2.*g(4,n-1,j)-g(4,n-2,j)
g(5,n,j)=2.*g(5,n-1,j)-g(5,n-2,j)
g(8,n,j)=2.*g(8,n-1,j)-g(8,n-2,j)
end do
! Top boundary conditions, T=0.0
do i=0,n
g(8,i,m)=tw*(w(8)+w(6))-g(6,i,m)
g(7,i,m)=tw*(w(7)+w(5))-g(5,i,m)
g(4,i,m)=tw*(w(4)+w(2))-g(2,i,m)
g(1,i,m)=tw*(w(1)+w(3))-g(3,i,m)
end do
!Bottom boundary conditions, Adiabatic
! g(1,i,0)=g(1,i,1)
! g(2,i,0)=g(2,i,1)
! g(3,i,0)=g(3,i,1)
! g(4,i,0)=g(4,i,1)
! g(5,i,0)=g(5,i,1)
! g(6,i,0)=g(6,i,1)
! g(7,i,0)=g(7,i,1)
! g(8,i,0)=g(8,i,1)
! T=0.0
do i=0,n
g(2,i,0)=tw*(w(2)+w(4))-g(4,i,0)
g(6,i,0)=tw*(w(6)+w(8))-g(8,i,0)
g(5,i,0)=tw*(w(5)+w(7))-g(7,i,0)
end do
return
end
subroutine tcalcu(g,th,n,m)
real g(0:8,0:n,0:m),th(0:n,0:m)
do j=1,m-1
do i=1,n-1
```

```
ssumt=0.0
do k=0,8
ssumt=ssumt+g(k,i,j)
end do
th(i,j)=ssumt
end do
end do
return
end
subroutine rhouv(f,rho,u,v,cx,cy,n,m)
real f(0:8,0:n,0:m),rho(0:n,0:m),u(0:n,0:m),v(0:n,0:m),cx(0:8),cy(0:8)
do j=0,m
do i=0,n
ssum=0.0
do k=0,8
ssum=ssum+f(k,i,j)
end do
rho(i,j)=ssum
end do
end do
DO i=1,n
DO j=1,m-1
usum=0.0
vsum=0.0
DO k=0,8
usum=usum+f(k,i,j)*cx(k)
vsum=vsum+f(k,i,j)*cy(k)
END DO
u(i,j)=usum/rho(i,j)
v(i,j)=vsum/rho(i,j)
END DO
END DO
do j=0,m
v(n,j)=0.0
end do
return
end
subroutine result(u,v,th,uo,n,m)
real u(0:n,0:m),v(0:n,0:m),th(0:n,0:m)
2 write(2,*)"VARIABLES =X, Y, U, V, T"
write(2,*)"ZONE ","I=",n+1,"J=",m+1,",","F=BLOCK"
```

```
do j=0,m
write(2,*)(i,i=0,n)
end do
do j=0,m
write(2,*)(j,i=0,n)
end do
do j=0,m
write(2,*)(u(i,j),i=0,n)
end do
do j=0,m
write(2,*)(v(i,j),i=0,n)
end do
do j=0,m
write(2,*)(th(i,j),i=0,n)
end do
do j=0,m
write(3,*)j/float(m),u(5,j)/uo,u(n/2,j)/uo,u(n-10,j)/uo
end do
do i=0,n
write(4,*) i/float(n),v(i,m/2)/uo
end do
return
end
```

==========

Computer code for Natural convection in a differentially heated cavity

==========

```
parameter (n=100,m=100)
real f(0:8,0:n,0:m)
real feq(0:8,0:n,0:m),rho(0:n,0:m)
real w(0:8), cx(0:8),cy(0:8)
real u(0:n,0:m), v(0:n,0:m)
real g(0:8,0:n,0:m), geq(0:8,0:n,0:m),th(0:n,0:m)
integer i
open(2,file='uvfield')
open(3,file='uvely')
open(4,file='tmx')
!
cx(:)=(/0.0,1.0,0.0,-1.0,0.0,1.0,-1.0,-1.0,1.0/)
cy(:)=(/0.0,0.0,1.0,0.0,-1.0,1.0,1.0,-1.0,-1.0/)
w(:)=(/4./9.,1./9.,1./9.,1./9.,1./9.,1./36.,1./36.,1./36.,1./36./)
```

```
uo=0.0
sumvelo=0.0
rhoo=6.00
dx=1.0
dy=dx
dt=1.0
tw=1.0
th=0.0
ra=1.0e5
pr=0.71
visco=0.02
alpha=visco/pr
pr=visco/alpha
gbeta=ra*visco*alpha/(float(m*m*m))
Re=uo*m/alpha
print *, ”Re=”, Re
omega=1.0/(3.*visco+0.5)
omegat=1.0/(3.*alpha+0.5)
mstep=150000
do j=0,m
do i=0,n
rho(i,j)=rhoo
u(i,j)=0.0
v(i,j)=0.0
end do
end do
do i=0,n
u(i,m)=uo
v(i,m)=0.0
end do
! main loop
1 do kk=1,mstep
call collesion(u,v,f,feq,rho,omega,w,cx,cy,n,m,th,gbeta)
call streaming(f,n,m)
call bounceb(f,n,m)
call rhouv(f,rho,u,v,cx,cy,n,m)
! ———————————
! collestion for scalar
call collt(u,v,g,geq,th,omegat,w,cx,cy,n,m)
! streaming for scalar
call streaming(g,n,m)
```

```
call gbound(g,tw,w,n,m)
do j=0,m
do i=0,n
sumt=0.0
do k=0,8
sumt=sumt+g(k,i,j)
end do
th(i,j)=sumt
end do
end do
print *, th(n/2,m/2),v(5,m/2),rho(0,m/2),u(n/2,m/2),v(n/2,m/2),rho(n,m/2)
END DO
! end of the main loop
call result(u,v,rho,th,uo,n,m,ra)
stop
end
! end of the main program
subroutine collesion(u,v,f,feq,rho,omega,w,cx,cy,n,m,th,gbeta)
real f(0:8,0:n,0:m)
real feq(0:8,0:n,0:m),rho(0:n,0:m)
real w(0:8), cx(0:8),cy(0:8)
real u(0:n,0:m), v(0:n,0:m)
real th(0:n,0:m)
tref=0.50
DO i=0,n
DO j=0,m
t1=u(i,j)*u(i,j)+v(i,j)*v(i,j)
DO k=0,8
t2=u(i,j)*cx(k)+v(i,j)*cy(k)
force=3.*w(k)*gbeta*(th(i,j)-tref)*cy(k)*rho(i,j)
if(i.eq.0.or.i.eq.n) force =0.0
if(j.eq.0.or.j.eq.m) force =0.0
feq(k,i,j)=rho(i,j)*w(k)*(1.0+3.0*t2+4.50*t2*t2-1.50*t1)
f(k,i,j)=omega*feq(k,i,j)+(1.-omega)*f(k,i,j)+force
END DO
END DO
END DO
return
end
subroutine collt(u,v,g,geq,th,omegat,w,cx,cy,n,m)
real g(0:8,0:n,0:m),geq(0:8,0:n,0:m),th(0:n,0:m)
```

```
real w(0:8),cx(0:8),cy(0:8)
real u(0:n,0:m),v(0:n,0:m)
do i=0,n
do j=0,m
do k=0,8
geq(k,i,j)=th(i,j)*w(k)*(1.0+3.0*(u(i,j)*cx(k)+v(i,j)*cy(k)))
g(k,i,j)=omegat*geq(k,i,j)+(1.0-omegat)*g(k,i,j)
end do
end do
end do
return
end
subroutine streaming(f,n,m)
real f(0:8,0:n,0:m)
! streaming
DO j=0,m
DO i=n,1,-1 !RIGHT TO LEFT
f(1,i,j)=f(1,i-1,j)
END DO
DO i=0,n-1 !LEFT TO RIGHT
f(3,i,j)=f(3,i+1,j)
END DO
END DO
DO j=m,1,-1 !TOP TO BOTTOM
DO i=0,n
f(2,i,j)=f(2,i,j-1)
END DO
DO i=n,1,-1
f(5,i,j)=f(5,i-1,j-1)
END DO
DO i=0,n-1
f(6,i,j)=f(6,i+1,j-1)
END DO
END DO
DO j=0,m-1 !BOTTOM TO TOP
DO i=0,n
f(4,i,j)=f(4,i,j+1)
END DO
DO i=0,n-1
f(7,i,j)=f(7,i+1,j+1)
END DO
```

```
DO i=n,1,-1
f(8,i,j)=f(8,i-1,j+1)
END DO
END DO
return
end
subroutine bounceb(f,n,m)
real f(0:8,0:n,0:m)
do j=0,m
!west boundary
f(1,0,j)=f(3,0,j)
f(5,0,j)=f(7,0,j)
f(8,0,j)=f(6,0,j)
!east boundary
f(3,n,j)=f(1,n,j)
f(7,n,j)=f(5,n,j)
f(6,n,j)=f(8,n,j)
end do
do i=0,n
!south boundary
f(2,i,0)=f(4,i,0)
f(5,i,0)=f(7,i,0)
f(6,i,0)=f(8,i,0)
!north boundary
f(4,i,m)=f(2,i,m)
f(8,i,m)=f(6,i,m)
f(7,i,m)=f(5,i,m)
end do
return
end
subroutine gbound(g,tw,w,n,m)
real g(0:8,0:n,0:m)
real w(0:8),tw
! Boundary conditions
! West boundary condition, T=1.
do j=0,m
g(1,0,j)=tw*(w(1)+w(3))-g(3,0,j)
g(5,0,j)=tw*(w(5)+w(7))-g(7,0,j)
g(8,0,j)=tw*(w(8)+w(6))-g(6,0,j)
end do
! East boundary condition, T=0.
```

```
do j=0,m
g(6,n,j)=-g(8,n,j)
g(3,n,j)=-g(1,n,j)
g(7,n,j)=-g(5,n,j)
end do
! Top boundary conditions, Adiabatic
do i=0,n
g(8,i,m)=g(8,i,m-1)
g(7,i,m)=g(7,i,m-1)
g(6,i,m)=g(6,i,m-1)
g(5,i,m)=g(5,i,m-1)
g(4,i,m)=g(4,i,m-1)
g(3,i,m)=g(3,i,m-1)
g(2,i,m)=g(2,i,m-1)
g(1,i,m)=g(1,i,m-1)
g(0,i,m)=g(0,i,m-1)
end do
!Bottom boundary conditions, Adiabatic
do i=0,n
g(1,i,0)=g(1,i,1)
g(2,i,0)=g(2,i,1)
g(3,i,0)=g(3,i,1)
g(4,i,0)=g(4,i,1)
g(5,i,0)=g(5,i,1)
g(6,i,0)=g(6,i,1)
g(7,i,0)=g(7,i,1)
g(8,i,0)=g(8,i,1)
g(0,i,0)=g(0,i,1)
end do
return
end
subroutine tcalcu(g,th,n,m)
real g(0:8,0:n,0:m),th(0:n,0:m)
do j=0,m
do i=0,n
ssumt=0.0
do k=0,8
ssumt=ssumt+g(k,i,j)
end do
th(i,j)=ssumt
end do
```

```
end do
return
end
subroutine rhouv(f,rho,u,v,cx,cy,n,m)
real f(0:8,0:n,0:m),rho(0:n,0:m),u(0:n,0:m),v(0:n,0:m),cx(0:8),cy(0:8)
do j=0,m
do i=0,n
ssum=0.0
do k=0,8
ssum=ssum+f(k,i,j)
end do
rho(i,j)=ssum
end do
end do
DO i=0,n
DO j=0,m
usum=0.0
vsum=0.0
DO k=0,8
usum=usum+f(k,i,j)*cx(k)
vsum=vsum+f(k,i,j)*cy(k)
END DO
u(i,j)=usum/rho(i,j)
v(i,j)=vsum/rho(i,j)
END DO
END DO
return
end
subroutine result(u,v,rho,th,uo,n,m,ra)
real u(0:n,0:m),v(0:n,0:m),th(0:n,0:m)
real strf(0:n,0:m),rho(0:n,0:m)
open(5,file='streamt')
open(6,file='nuav')
! streamfunction calculations
strf(0,0)=0.
do i=0,n
rhoav=0.5*(rho(i-1,0)+rho(i,0))
if(i.ne.0) strf(i,0)=strf(i-1,0)-rhoav*0.5*(v(i-1,0)+v(i,0))
do j=1,m
rhom=0.5*(rho(i,j)+rho(i,j-1))
strf(i,j)=strf(i,j-1)+rhom*0.5*(u(i,j-1)+u(i,j))
```

```
end do
end do
! ——————————————————
write(2,*)"VARIABLES =X, Y, U, V"
write(2,*)"ZONE ","I=",n+1,"J=",m+1,",","F=BLOCK"
do j=0,m
write(2,*)(i/float(m),i=0,n)
end do
do j=0,m
write(2,*)(j/float(m),i=0,n)
end do
do j=0,m
write(2,*)(u(i,j),i=0,n)
end do
do j=0,m
write(2,*)(v(i,j),i=0,n)
end do
do j=0,m
write(3,*)j/float(m),u(5,j)/uo,u(n/2,j)/uo,u(n-10,j)/uo
end do
do i=0,n
write(4,*) i/float(n),th(i,m/2)
end do
write(5,*)"VARIABLES =X, Y, S, T"
write(5,*)"ZONE ","I=",n+1,"J=",m+1,",","F=BLOCK"
do j=0,m
write(5,*)(i/float(m),i=0,n)
end do
do j=0,m
write(5,*)(j/float(m),i=0,n)
end do
do j=0,m
write(5,*)(strf(i,j),i=0,n)
end do
do j=0,m
write(5,*)(th(i,j),i=0,n)
end do
! Nusselt number Calculation
snul=0.0
snur=0.0
do j=0,m
```

```
rnul=(th(0,j)-th(1,j))*float(n)
rnur=(th(n-1,j)-th(n,j))*float(n)
snul=snul+rnul
snur=snur+rnur
write(5,*)j/float(m),rnul,rnur
end do
avnl=snul/float(m)
avnr=snur/float(m)
write(6,*)ra,avnl,avnr
return
end
```

==========

MRT code

```
! This code works fine and produce results for lid driven cavity, MRT code.

      parameter (n=100,m=100)
      real f(0:8,0:n,0:m)
      real feq(0:8,0:n,0:m),rho(0:n,0:m)
      real w(0:8), cx(0:8),cy(0:8)
      real u(0:n,0:m), v(0:n,0:m)
      integer i
      real tminv(0:8,0:8),sm(0:8),tm(0:8,0:8),stmiv(0:8,0:8)
      real ev(0:8,0:8)
      open(2,file='uvfield')
      open(3,file='uvely')
      open(4,file='vvelx')
      open(8,file='timeu')
      open(10,file='tmat')
      w(:)=(/4./9.,1./9.,1./9.,1./9.,1./9.,1./36.,1./36.,1./36.,1./36./)
```

```
cx(:)=(/0.,1.,0.,-1.,0.,1.,-1.,-1.,1./)
cy(:)=(/0.,0.,1.,0.,-1.,1.,1.,-1.,-1./)
tm(0,:)=(/1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0/)
tm(1,:)=(/-4.,-1.,-1.,-1.,-1.,2.0,2.0,2.0,2.0/)
tm(2,:)=(/4.0,-2.,-2.,-2.,-2.,1.0,1.0,1.0,1.0/)
tm(3,:)=(/0.0,1.0,0.0,-1.,0.0,1.0,-1.,-1.,1.0/)
tm(4,:)=(/0.0,-2.,0.0,2.0,0.0,1.0,-1.,-1.,1.0/)
tm(5,:)=(/0.0,0.0,1.0,0.0,-1.,1.0,1.0,-1.,-1./)
tm(6,:)=(/0.0,0.0,-2.,0.0,2.0,1.0,1.0,-1.,-1./)
tm(7,:)=(/0.0,1.0,-1.,1.0,-1.,0.0,0.0,0.0,0.0/)
tm(8,:)=(/0.0,0.0,0.0,0.0,0.0,1.0,-1.,1.0,-1./)
 a1=1./36.
tminv(0,:)=(/4.*a1,-4.*a1,4.*a1, 0,0,0,0,0.,0./)
tminv(1,:)=(/4.*a1,-a1,-2.*a1,6.*a1,-6.*a1,0.,0.,9.*a1,0.0/)
tminv(2,:)=(/4.*a1,-a1,-2.*a1,0.,0.0,6.*a1,-6.*a1,-9.*a1,0.0/)
tminv(3,:)=(/4.*a1,-a1,-2.*a1,-6.*a1,6.*a1,0.,0.,9.*a1,0.0/)
tminv(4,:)=(/4.*a1,-a1,-2.*a1,0.,0.,-6.*a1,6.*a1,-9.*a1,0./)
tminv(5,:)=(/4.*a1,2*a1,a1,6.*a1,3.*a1,6.*a1,3.*a1,0.,9.*a1/)
tminv(6,:)=(/4.*a1,2*a1,a1,-6*a1,-3.*a1,6.*a1,3.*a1,0,-9.*a1/)
tminv(7,:)=(/4.*a1,2*a1,a1,-6*a1,-3.*a1,-6*a1,-3*a1,0.,9.*a1/)
tminv(8,:)=(/4.*a1,2*a1,a1,6*a1,3.*a1,-6.*a1,-3.*a1,0.,-9*a1/)
 do i=0,8
 do j=0,8
   sumcc=0.0
   do l=0,8
     sumcc=sumcc+tminv(i,l)*tm(l,j)
```

```
   end do
   ev(i,j)=sumcc
  end do
  end do
  do i=0,8
   print *,(ev(i,j),j=0,8)
  end do

 uo=0.05
rhoo=1.00
dx=1.0
dy=dx
dt=1.0
alpha=0.001
Re=uo*m/alpha
print *, "Re=", Re
 omega=1.0/(3.*alpha+0.5)
 tau=1./omega

sm(:)=(/1.0,1.4,1.4,1.0,1.2,1.0,1.2,tau,tau/)
  do i=0,8
  do j=0,8
   stmiv(i,j)=tminv(i,j)*sm(j)

  end do
  end do
```

```
      do i=0,8
      print *, (stmiv(i,j),j=0,8)
      end do

     mstep=100000
       do j=0,m
     do i=0,n
      rho(i,j)=rhoo
      u(i,j)=0.0
      v(i,j)=0.0
      end do
      end do
      do i=1,n-1
      u(i,m)=uo
      v(i,m)=0.0
      end do
!     main loop
1      do kk=1,mstep
       call collesion(u,v,f,feq,rho,omega,w,cx,cy,n,m,tm,tminv,stmiv)
       call streaming(f,n,m)
! ---------------------------------
       call sfbound(f,n,m,uo)
       call rhouv(f,rho,u,v,cx,cy,n,m)
       print *, u(0,m/2),v(0,m/2),rho(0,m/2),u(n,m/2),v(n,m/2),rho(n,m/2)
      write(8,*) kk,u(n/2,m/2),v(n/2,m/2)
   END DO
```

```
!  end of the main loop
        call result(u,v,rho,uo,n,m)
      stop
      end
!  end of the main program


      subroutine collesion(u,v,f,feq,rho,omega,w,cx,cy,n,m,tm,tminv,stmiv)
      real f(0:8,0:n,0:m)
      real feq(0:8,0:n,0:m),rho(0:n,0:m)
      real w(0:8), cx(0:8),cy(0:8)
      real u(0:n,0:m), v(0:n,0:m)
      real tm(0:8,0:8),tminv(0:8,0:8),stmiv(0:8,0:8)
      real fmom(0:8,0:n,0:m),fmeq(0:8,0:n,0:m)
! Calculate equilibrium moments
   do i=0,n
   do j=0,m
    fmeq(0,i,j)=rho(i,j)
    fmeq(1,i,j)=rho(i,j)*(-2.0+3.0*rho(i,j)*(u(i,j)*u(i,j)+v(i,j)*v(i,j)))
    fmeq(2,i,j)=rho(i,j)*(1.0-3.0*rho(i,j)*(u(i,j)*u(i,j)+v(i,j)*v(i,j)))
    fmeq(3,i,j)=rho(i,j)*u(i,j)
    fmeq(4,i,j)=-rho(i,j)*u(i,j)
    fmeq(5,i,j)=rho(i,j)*v(i,j)
    fmeq(6,i,j)=-rho(i,j)*v(i,j)
    fmeq(7,i,j)=rho(i,j)*(u(i,j)*u(i,j)-v(i,j)*v(i,j))
    fmeq(8,i,j)=rho(i,j)*u(i,j)*v(i,j)
   end do
```

```
  end do
! Calculate Moments
  do i=0,n
  do j=0,m
    do k=0,8
       suma=0.0
      do l=0,8
       suma=suma+tm(k,l)*f(l,i,j)
      end do
       fmom(k,i,j)=suma
    end do
  end do
  end do
! Collision in the moment space
    do i=0,n
    do j=0,m
       do k=0,8
         sumb=0.0
        do l=0,8
          sumb=sumb+stmiv(k,l)*(fmom(l,i,j)-fmeq(l,i,j))
        end do
          f(k,i,j)=f(k,i,j)-sumb
       end do
    end do
    end do
```

```
  return
  end
        subroutine streaming(f,n,m)
     real f(0:8,0:n,0:m)
! streaming
DO j=0,m
  DO i=n,1,-1          !RIGHT TO LEFT
    f(1,i,j)=f(1,i-1,j)
  END DO
  DO i=0,n-1           !LEFT TO RIGHT
    f(3,i,j)=f(3,i+1,j)
  END DO
END DO
DO j=m,1,-1            !TOP TO BOTTOM
  DO i=0,n
    f(2,i,j)=f(2,i,j-1)
  END DO
  DO i=n,1,-1
    f(5,i,j)=f(5,i-1,j-1)
  END DO
  DO i=0,n-1
    f(6,i,j)=f(6,i+1,j-1)
  END DO
END DO
DO j=0,m-1             !BOTTOM TO TOP
  DO i=0,n
```

```
      f(4,i,j)=f(4,i,j+1)
    END DO
    DO i=0,n-1
      f(7,i,j)=f(7,i+1,j+1)
    END DO
    DO i=n,1,-1
      f(8,i,j)=f(8,i-1,j+1)
    END DO
  END DO
  return
  end
      subroutine sfbound(f,n,m,uo)
       real f(0:8,0:n,0:m)
       do j=0,m
  ! bounce back on west boundary
        f(1,0,j)=f(3,0,j)
        f(5,0,j)=f(7,0,j)
        f(8,0,j)=f(6,0,j)
   ! bounce back on east boundary
        f(3,n,j)=f(1,n,j)
        f(7,n,j)=f(5,n,j)
        f(6,n,j)=f(8,n,j)
        end do
   !    bounce back on south boundary
        do i=0,n
          f(2,i,0)=f(4,i,0)
```

```
    f(5,i,0)=f(7,i,0)
    f(6,i,0)=f(8,i,0)
    end do
! moving lid, north boundary
 do i=1,n-1
     rhon=f(0,i,m)+f(1,i,m)+f(3,i,m)+2.*(f(2,i,m)+f(6,i,m)+f(5,i,m))
    f(4,i,m)=f(2,i,m)
    f(8,i,m)=f(6,i,m)+rhon*uo/6.0
    f(7,i,m)=f(5,i,m)-rhon*uo/6.0
 end do

  return
    end

  subroutine rhouv(f,rho,u,v,cx,cy,n,m)
     real f(0:8,0:n,0:m),rho(0:n,0:m),u(0:n,0:m),v(0:n,0:m),cx(0:8),cy(0:8)
     do j=0,m
     do i=0,n
        ssum=0.0
       do k=0,8
         ssum=ssum+f(k,i,j)
       end do
       rho(i,j)=ssum
      end do
    end do
do i=1,n
```

```
        rho(i,m)=f(0,i,m)+f(1,i,m)+f(3,i,m)+2.*(f(2,i,m)+f(6,i,m)+f(5,i,m))
        end do
  DO i=1,n
   DO j=1,m-1
     usum=0.0
     vsum=0.0
     DO k=0,8
       usum=usum+f(k,i,j)*cx(k)
       vsum=vsum+f(k,i,j)*cy(k)
     END DO
     u(i,j)=usum/rho(i,j)
     v(i,j)=vsum/rho(i,j)
   END DO
END DO
return
end
      subroutine result(u,v,rho,uo,n,m)
      real u(0:n,0:m),v(0:n,0:m)
      real rho(0:n,0:m),strf(0:n,0:m)
       open(5, file='streamf')
  !  streamfunction calculations
          strf(0,0)=0.
        do  i=0,n
          rhoav=0.5*(rho(i-1,0)+rho(i,0))
        if(i.ne.0) strf(i,0)=strf(i-1,0)-rhoav*0.5*(v(i-1,0)+v(i,0))
```

```
      do  j=1,m
      rhom=0.5*(rho(i,j)+rho(i,j-1))
        strf(i,j)=strf(i,j-1)+rhom*0.5*(u(i,j-1)+u(i,j))
       end do
       end do
! ----------------------------------


    write(2,*)"VARIABLES =X, Y, U,  V, S"
    write(2,*)"ZONE ","I=",n+1,"J=",m+1,",","F=BLOCK"
    do j=0,m
    write(2,*)(i,i=0,n)
    end do
    do j=0,m
    write(2,*)(j,i=0,n)
    end do
    do j=0,m
    write(2,*)(u(i,j),i=0,n)
    end do
    do j=0,m
     write(2,*)(v(i,j),i=0,n)
     end do
    do j=0,m
      write(2,*)(strf(i,j),i=0,n)
    end do
    do j=0,m
      write(3,*)j/float(m),u(n/2,j)/uo,u(n/4,j)/uo,u(3*n/4,j)/uo
```

```
end do
 do i=0,n
   write(4,*) i/float(n),v(i,m/2)/uo
 end do
return
end
```

=======end of MRT code.==========

Bibliography

Boundary Conditions

Yu D, Mei R, Lou L, Shyy W. Viscous flow computations with the method of lattice Boltzmann equation. Progr Aero Sci. 2003;39:329–67. *This paper reviews different techniques of boundary conditions and introduces a new method of treating inlet boundary conditions base on Ph.D. thesis of the first author. It is recommended to read and adopt the suggested boundary condition.*

Inamuro T, Yoshino M, Ogino F. A non-slip boundary condition of lattice Boltzmann simulations. Phys Fluids. 1995a;7(12):2928–30.

Chen H, Teixeria C, Molvig K. Realization of fluid boundary conditions via discrete Boltzmann dynamics. Int J Mod Phys. C, 1998;9(8):1281–1292.

Bouzidi M, Firdaouss M, Lallemand P. Momentum transfer of a Boltzmann-lattice fluid with boundaries. Phys Fluids. 13(11):3452–59.

Noble D, Chen S, Georgiadis J, Buckius R. A consistent hydrodynamic boundary condition for the lattice Boltzmann method. Phys Fluids. 1995;7(1):203–9.

Maier R, Bernard R, Grunau D. Boundary condition for the lattice Boltzmann method. Phys Fluids. 1996;8(7):1788–801.

Inamuro T, Yoshino M, Ogino F. A non-slip boundary condition for lattice Boltzmann simulations. Phys Fluids. 1995b;7(12):2928–30.

Review Papers

Chen S, Doolen GD. Lattice Boltzmann method for fluid flows. Annu Rev Fluid Mech. 1998;30:329–64.

Diffusion

Wolf-Gladrow D. A Lattice Boltzmann equation for diffusion. J Stat Phys. 1995a;79(5/6): 1023–32.

Ho J, Kuo C, Jiaung W, Twu C, Lattice Boltzmann scheme for hyperbolic heat conduction equation. Numer Heat Transf. Part B 41:591–607

Jiaung W, Ho J, Kuo C. Lattice Boltzmann method for the heat conduction problem with phase change. Numer Heat Transf. 2001a;Part B(39):167–87.

van Der Sman R. Diffusion on unstructured triangular grids using lattice Boltzmann. Future Generat Comput Syst. 2004;20:965–71.
Mishra SC, Lankadasu A, Beronov KN. Application of the lattice Boltzmann method for solving the energy equation of a 2-D transient conduction–radiation problem. Int J Heat Mass Transf. 2005;48:3648–59.
Prog Comput Fluid Dynam Int J. 2009, 9(8):490–94.

Advection–Diffusion

Dawson SP, Chen S, Doolen GD. Lattice Boltzmann computations for reaction–diffusion equations. J Chem Phys. 1993;98(2):1514–23.
van der Saman RGM, Ernst MH, Berkenbosh AC. Lattice Boltzmann scheme for cooling and packed cut flowers. Int J Heat Mass Transf. 2000;43:577–87.
Kingdon RD, Schofield P. A reaction-flow lattice Boltzmann model. J Phys A Math Gen. 1992;25:L907–10.
van der Saman RGM, Ernst MH. Convection–diffusion lattice Boltzmann scheme for irregular lattices. J Comput Phys. 2000;160:766–82.

Isothermal Fluid Flow Problems

Peng Y, Shu C, Chew YT. Lattice kinetic scheme for the incompressible viscous thermal flows on arbitrary meshes. Phys Rev E. 2004;69:016703-1–016703-8.
Zhou Y, Zhang R, Staroselsky I, Chen H. Numerical simulation of laminar and turbulent buoyancy-driven flows using a lattice Boltzmann based algorithm. Int J Heat Mass Transf. 2004;47:4869–79.

Convection, Forced and Natural

van der Sman R. Galilean invariant lattice Boltzmann scheme for natural convection on square and rectangular lattices. Phys Rev E. 2006;74:026705-1–026705-17.
Seta T, Eishun T, Okui K. Lattice Boltzmann simulation of natural convection in porous media. Math Comput Simulat. 2006;72:195–200.

MRT

Ginzburg I. Adv Water Resour. 2005, 28(11):1171.
Mezrhab A, Moussaoui MA, Jami M, Naji H, Bouzidi M. Double MRT thermal lattice Boltzmann method for simulation convective flows. Phys Lett A. 2010;374:3499–507.
Wang L, Guo Z, Zheng C. Multi-relaxation-time lattice Boltzmann model for axisymmetric flows. Comput Fluids. 2010;39:1542–48.

Multiphase

Frank X, Funfschilling D, Midoux N, Li H. Bubbles in a viscous liquid: lattice Boltzmann simulation and experimental validation. J Fluid Mech. 2006;546:113–22.

Shan X, Chen H. Lattice Boltzmann model for simulating flows with multiple phases and components. Phys Rev E. 1993;47(3):1815–19.

Yang Z, Dinh T, Nourgaliev R, Sehgal B. Evaluation for the Darcy's law performance for two-fluid flow hydrodynamics in a particle debris bed using a lattice-Boltzmann model. Heat Mass Transf. 2000;36:295–304.

Agarwal S, Verma N, Mewes D. A lattice Boltzmann model for adsorption breakthrough. Heat Mass Transf. 2005;41:843–54.

Li B, Kwok DY. Lattice Boltzmann model of microfluidics in the presence of external forces. J Colloid Interface Sci. 2003;263:144–51.

Guangwu Y. A lattice Boltzmann equation for waves. J Comput Phys. 2000;161:61–9.

Buick JM, Greated CA. Gravity in a lattice Boltzmann model. Phys Rev E. 61(5):5307–16

Cates ME, Stratford K, Adhikari R, Stansell P, Desplat J-C, Pagonabarraga I, Wagner AJ. Simulating collid hydrodynamics with lattice Boltzmann methods. J Phys Condens Matter. 2004;16:S3903–15.

Vahala L, Vahala G, Yepez J. Lattice Boltzmann and quantum lattice gas representation of one-dimensional magnetohydrodynamic turbulence. Phys Lett A. 2003;306:227–34.

Yepez J. Type-II quantum computers. Int J Mod Phys C. 2001a;12(9):1273–84.

Yepez J. Quantum lattice-gas model for the diffusion equation. Int J Mod Phys C. 2001b;12(9):1285–1303.

Yamamoto K, He X, Doolen GD. Simulation of combustion field with lattice Boltzmann method. J Stat Phys. 2002;107(112):367–83.

Chopard B, Luthi PO. Lattice Boltzmann computations and apprications to physics. Theor Comput Sci. 1999;217:115–30.

Baoming Li, Kwok DY. A lattice Boltzmann model for electrokinetic microchannel flow of electrolyte solution in the presence of external forces with the Poisson–Boltzmann equation. Int J Heat Mass Transf. 2003;46:4235–44.

Barrios G, Techtman R, Rojas J, Tovar R. The lattice Boltzmann equation for natural convection in a two-dimensional cavity with a partially heated wall. J Fluid Mech. 2005;522:91–100.

Yan G, Ruan L. Lattice Boltzmann solver of Rossler equation. Commun Nonlin Sci Numer Simul. 200;64–8

Toffoli T, Margolus N. Invertible cellular automata. A rev Phys D. 1990;45:229–53.

Wolf-Gladrow D. A lattice Boltzmann equation for diffusion. J Stat Phys. 1995b;79:1023–32.

van der Sman RGM, Ernst MH, Berkenbosch AC. Lattice Boltzmann scheme for cooling of packed cut flowers. Int J Heat Mass Transf. 2000;43:577–87.

Ho J-R, Kuo C-P, Jiaung W-S, Twu C-J. Lattice Boltzmann scheme for hyperbolic heat conduction equation. Numer Heat Transf. 2002;B41:591–607.

Jiaung W-S, Ho J-R, Kuo C-P. Lattice Boltzmann method for the heat conduction problem with phase change. Numer Heat Transf. 2001b;B39:167–87.

Index