A. A. Mohamad

Lattice Boltzmann Method

Fundamentals and Engineering Applications with Computer Codes

Second Edition

Springer

Lattice Boltzmann Method

A. A. Mohamad

Lattice Boltzmann Method

Fundamentals and Engineering Applications with Computer Codes

Second Edition

Springer

A. A. Mohamad
Department of Mechanical and
Manufacturing Engineering, Schulich
School of Engineering
University of Calgary
Calgary, AB, Canada

ISBN 978-1-4471-7422-6 ISBN 978-1-4471-7423-3 (eBook)
https://doi.org/10.1007/978-1-4471-7423-3

Library of Congress Control Number: 2018963038

This Springer imprint is published by the registered company Springer-Verlag London Ltd. part of Springer Nature.
The registered company address is: The Campus, 4 Crinan Street, London, N1 9XW, United Kingdom

Simplicity is Embedded in Complexity
The world is no more than a collection of digits and pixels.

Preface to the Second Edition

Since the first edition of this book was published, many advancements in LBM have taken place. Also, I have received a number of requests, mostly from students who have begun working in LBM, to add more material to the book. Most requests have been for more on the theory and applications. However, readers have greatly appreciated the simplicity of the approach and the inclusion of computer codes, which I have tried to maintain in this version of the book. Hence, a few sections on the theory have been added to the book, which make the book a stand–alone source for those who need to understand the basic theory of LBM. I assume that for readers who are interested to diving into the topic to a deep level, the book will be help them to dive safely. Also, all the chapters of the book have been revised and edited. However, since most students and researchers are using Matlab software for simulations, in the current edition, all codes have been rewritten in Matlab. However, the codes have been written in a way that they can easily be converted into another language, such as Fortran, Python, or and Julia. We recognize that Matlab is slow compared with Fortran or Julia. However, it is easy and friendly to use. For large problems (huge number of lattices and/or 3D), I recommend the use of other computer languages, such as Fortran or C.

Calgary, Canada A. A. Mohamad

Preface to the First Edition

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, Clough, Martin and Topp. In the late 1960s, the finite element method 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 finite volume method 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 has emerged as an alternative powerful method for solving fluid dynamics problems. In traditional computational fluid dynamics (CFD) methods, Navier–Stokes (NS) equations solve mass, momentum and energy conservation equations on discrete nodes, elements or volumes. In other words, the nonlinear partial differential equations are converted into a set of nonlinear algebraic equations, which are solved iteratively. In the method of lines, the partial differential equations are converted into a system of ordinary differential equations, mainly in time, by discretizing the equations in the spatial domain. In LBM, the fluid is replaced by distribution functions of fractious particles. These particles stream along a given direction (lattice links) and collide at the lattice sites. The LBM can be considered an explicit method. The collision and streaming processes are local. Hence, it naturally can be programmed 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 methods as is the case 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 could see the future of the method in dealing with multiphase and multicomponent flows. Furthermore, it is easy to incorporate thermodynamics with LBM, while such incorporation is a difficult task with the conventional method of solving NSE. The most time-consuming process in solving incompressible flows, using traditional CFD methods, is dealing with the pressure term. At each updating step, the Laplace equation must be solved to satisfy the continuity equation. This process is the most demanding of computer resources especially for unsteady-state problems. In LBM, there is no need for such a process, where LBM is an explicit method, by nature. However, there is no method without difficulties and drawbacks.

I thought there was 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 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 clear the way for the reader to understand more complicated issues.

I do believe that learning by working with problems and applications can help in understanding the topic. Engineering students and some science students are not well prepared in the kinetic theory and statistical mechanics. The book tries to cover the fundamentals of kinetic theory and statistical methods before introducing LBM, which is the backbone of most molecular and microanalysis 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 was suggested to direct the reader to more in-depth materials. The book is an introduction to the topic of 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 with little effort. The book can be used as a textbook for a senior undergraduate or graduate one-semester course with hands-on examples. I think that the way in which the materials are introduced in the book builds confidence in the reader in understanding the topic and coding without any guesses. In some cases, the finite difference method is introduced parallel to LBM for two reasons, first to show the differences and similarities between the well-established finite difference method and LBM, and second, to compare the results predicted by the two methods. The finite difference method is easy to understand because it is based on Taylor series expansion, to which most students have been exposed in their first-year calculus class.

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 clarity and simplicity and not for efficiency. The first chapter, an introduction to kinetic theory, is given, intended to familiarize the reader with concepts of kinetic theory. The second chapter introduces the lattice Boltzmann equation with a general discussion of the method. The third chapter deals with the diffusion equations for heat, mass and momentum. The fourth chapter introduces advection–diffusion equations without and with a source term. The fifth chapter discusses examples of isothermal fluid flow problems (without heat or mass transfer). The sixth chapter is complementary to the fifth chapter, in which non-isothermal fluid flow problems are discussed, without and with coupling, i.e. forced and free convection. The seventh chapter 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.

The author would greatly appreciate any comments from the readers, via email: mohamad@ucalgary.ca.

Acknowledgements

No matter where we stand now, there were people who helped us to get there. There are many unseen hands behind the scenes helping us morally or physically to perform on stage. Writing a book is no exception. I learned a lot from colleagues with whom I have worked and from students I have taught.

I have received many encouraging comments from a number of students and researchers, especially from newcomers to the topic. I should mention that a former Ph.D. student of mine, Dr. Saleh Bawazeer, made many comments and corrected bugs in the Matlab codes.

However, without an environment both stimulating and relaxing, nothing can be done as it ought, especially when one must spend hours and hours in front of books, papers, and computer without distraction. The smiles of those around you give energy to the work. Just to mention a few, family members, as usual, come first.

Contents

Chapter 1
Introduction and Kinetics of Particles

1.1 Introduction

There are two main approaches to solving the transport equations (heat, mass, and momentum) computationally: continuous and discrete. In the continuous approach, ordinary or partial differential equations can be obtained by applying conservation of energy, mass, and momentum with an infinitesimal control volume. Since it is difficult to solve the governing differential equations for many reasons (nonlinearity, complex boundary conditions, complex geometry, etc.), one uses finite difference, finite volume, and finite element methods, among others, to convert the governing differential equations with a given boundary and initial conditions to a system of algebraic equations. Those equations can be solved iteratively until convergence is ensured. Let us discuss the procedure in more detail for a given problem in which the governing equations need to be identified (mainly partial differential equations). This step is called mathematical modeling, which depends on the physics of the problem (and perhaps on the chemistry as well). The next step is to discretize the domain into finite volumes, grids, or elements, depending on the method of the solution. We can consider this step as assigning to each of the finite volumes or nodes or elements a collection of particles (a large number, on the order of 10^{16}). The scale is macroscopic. The velocity, pressure, and temperature of all the particles are represented by a nodal value, or averaged over a finite volume, or simply assumed to vary linearly or bilinearly from one node to another. The phenomenological properties such as viscosity, thermal conductivity, and heat capacity are in general known parameters (input parameters, except for inverse problems). For inverse problems, one or more thermophysical properties may be unknown.

At the other extreme, the medium can be assumed to be made of small particles (atoms, molecules), and those particles collide with each other. This scale is the microscale. Hence we need to identify the interparticle (intermolecular) forces and solve an ordinary differential equation of Newton's second law (momentum conservation). At each time step, we need to identify the location and velocity of each particle, i.e., the trajectory of the particles. Also, we need to ensure that the particles

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3_1

(molecules) do not reside one inside the other. At this level of simulation, there is no definition of temperature, pressure, and thermophysical properties such as viscosity, thermal conductivity, and heat capacity. For instance, temperature and pressure are related to the kinetic energy of the particles (mass and velocity) and the frequency at which particles bombard the boundaries, respectively. This method is called molecular dynamics simulation (MD). To get an idea of the number of equations needed to be solved, consider that 10 cubic centimeters of air at normal room conditions contain about 3×10^{22} molecules. To visualize this number, if we assume that the diameter of these molecules is one millimeter (tip of a pen, a dot) and these dots are arranged side by side, then one square kilometer would hold 10^{12} molecules, and so the total area that could be covered by these dots is $3.0 \times 10^{10}\,\mathrm{km}^2$. The total surface area of the Earth is about $5.1 \times 10^{9}\,\mathrm{km}^2$, and the area of the United States is $9.63 \times 10^{6}\,\mathrm{km}^2$, while area of Africa is $1.22 \times 10^{6}\,\mathrm{km}^2$. This means that we could cover the total surface area of the Earth six times with dots equal to the number of molecules available for us if the molecular diameter were 1 mm. Furthermore, it is interesting to calculate how many years we would need to complete the project of dotting the world using the tip of a pen. For a reasonably fast person, six dots per second is a good estimate. It would then take at least 3×10^{14} years to complete the project.

The second question is, do we really need to know the location and velocity of each molecule or atom?

In the bookkeeping process, we need to identify the location (x, y, z) and velocity (c_x, c_y, and c_z are the velocity components in x, y, and z directions, respectively) of each particle. Also, the simulation time step should be less than the particles' collision time, which is on the order of a ferosecond (10^{-12} s). Hence it is impossible to solve a large problem (order of meters) by the MD method. At this scale, there are no definitions of viscosity, thermal conductivity, temperature, pressure, and other phenomenological properties. Statistical mechanics must be used as a translator between the molecular world and the macroscopic world. We must ask whether the velocity and location of each particle are important to us. For instance, in this room there are billions of molecules traveling with high speed, some of them with speeds on the order of 400 m/s, like rockets hitting us. Yet we do not feel them, because their mass (momentum) is so small. The resultant effect of such a "chaotic" motion is almost nil when the air in the room is almost still (i.e., average air 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 changes and complex geometries without any difficulties and without introducing extra ingredients. However, it is important to specify the appropriate interparticle force function. The main drawback of using MD in simulating a relatively large system is computer resources, which will not be adquate for such a simulation for the foreseeable future. Also, the amount of data for a simulation outcome of MD is huge, which necessitates an extensive data storage system and extraordinary efforts for its analysis.

What about a middleman, sitting in the middle of both above-mentioned simulation techniques: the lattice Boltzmann method (LBM)? Boltzmann's main idea was to bridge the gap between the microscale and macroscale by not considering each

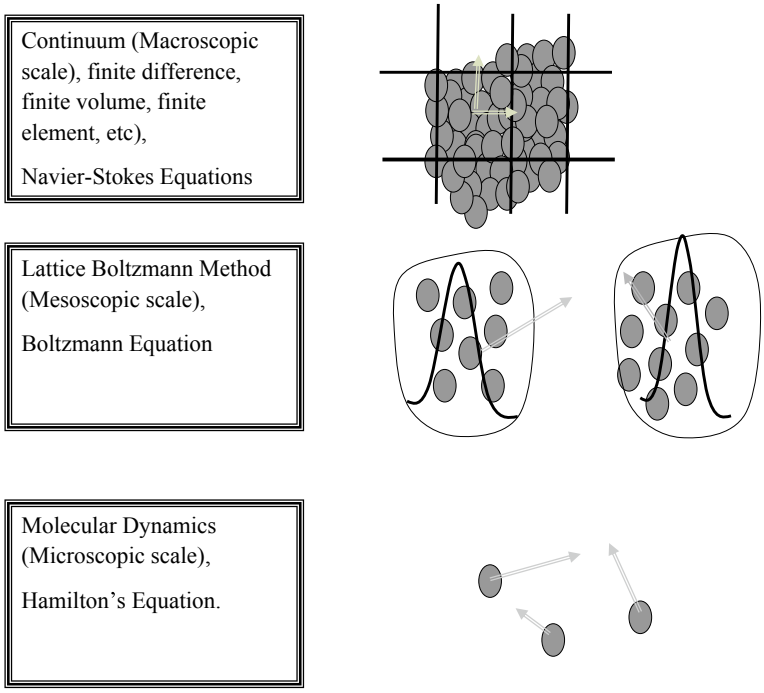


Fig. 1.1 Techniques of simulations

particle's behavior in isolation, but to treat the behavior of a collection of particles as a distribution function; Fig. 1.1. The property of the collection of particles is represented by a distribution function. The distribution function acts as a representative for a collection of particles. This scale is called the mesoscale. However, the LBM can model a wide range of problems from macroscale to nanoscale (for gaseous flow of Knudsen number of order one). The above-mentioned methods are illustrated in Fig. 1.1.

The lattice Boltzmann method (LBM) enjoys advantages of both the macroscopic and microscopic approaches, with manageable computer resources, and has many advantages compared with the conventional computational fluid dynamics method (CFD).

It is easy to apply it to complex domains; it is relatively easy to treat multiphase and multicomponent flows without needing to trace the interfaces between different phases. Furthermore, it can naturally be adapted to parallel processing due to the locality and explicit nature of the method. Moreover, there is no need to solve the Laplace equation at each time step to satisfy the continuity equation of incompressible unsteady flows, as is the case in solving the Navier–Stokes (NS) equation. However, it needs more computer memory compared with the NS solver, which is not a large constraint. Also, it can handle a problem at the macro-, micro-, and to some extent nanoscales with reliable accuracy. Moreover, it is easy to code compared with Navier–Stokes solvers.

The main objective of this chapter is to familiarize the reader with the basics of kinetic theory, distribution functions, and other terminologies and concepts used in the book. I have found that newcomers to LBM without background in either statistical mechanics or probability have had difficulty in understanding the concept of the distribution function.

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 the kinetic theory of particles. I have tried to avoid much of the mathematical detail, though in exchange, more emphasis is given to the physics.

Note that the words "particle" and "molecule" are used interchangeably in the following paragraphs.

1.2.1 Particle Dynamics

As far as we understand, the main building blocks of all matter in nature are molecules and submolecules. These molecules can be visualized as solid spheres moving "randomly" in free space in accordance with conservation laws. The motion satisfies conservation of 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.

$$\mathbf{F} = \frac{d(m\mathbf{c})}{dt}, \tag{1.1}$$

where $\mathbf{F}$ stands for the intermolecular and external forces, m is the mass of the particle, $\mathbf{c}$ is the velocity vector of the particle, and t is the time. For a constant mass, the equation can be simplified as

$$\mathbf{F} = m\frac{d\mathbf{c}}{dt} = m\mathbf{a}, \tag{1.2}$$

where $\mathbf{a}$ is the acceleration vector. The position of the particle can be determined from the definition of velocity,

$$\mathbf{c} = \frac{d\mathbf{r}}{dt}, \tag{1.3}$$

where $\mathbf{r}$ is the position vector of the particle relative to the origin, as shown in Fig. 1.2. Note that we are neglecting the rotation of the molecules. In molecular dynamics

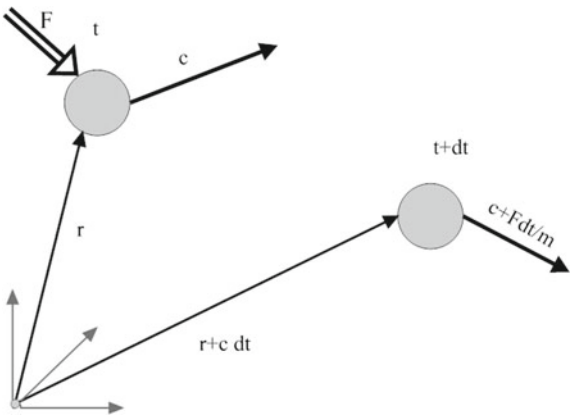


Fig. 1.2 Position and velocity vectors

(MD) simulation, the above equations can be solved, provided that **F** is a known function.

If an external force **F** is applied to a particle of mass m, the velocity of the particle will change from **c** to $\mathbf{c} + \mathbf{F}dt/m$, and its position changes from **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 without changing its direction and speed, assuming that no collision takes place.

The magnitudes of the particles' velocities increase, and interaction between particles increases as the internal energy of the system increases (for example, heating the system). Increases in the kinetic energy of the molecules is referred to as increases in temperature in the macroscopic world. The particles (molecules) are continuously bombarding the walls of a container. The net force exerted by those actions per unit area is referred to as pressure in macroscopic measure. From this simple model, we can see that there is a relation between temperature and pressure: as the temperature increases, which means that the kinetic energy of the molecules increases, we expect that the probability of particles bombarding the container walls increases, too (ideal gas law, $PV = mRT$).

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

1.2.2 Pressure and Temperature

Assume that a single particle is moving with speed c_x (in the 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 perfectly elastic), and then

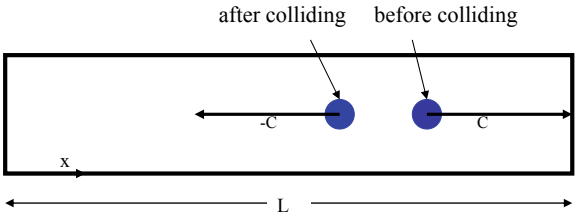


Fig. 1.3 A particle is freely moving in a box

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

where Δt is time between hits. Equation (1.4) is an integral of Newton's second law, Eq. (1.1). The time between hits is equal to $2L/c_x$, which is the time needed for the particle to travel from one end to the other and return to the same location; Fig. 1.3. Hence $2LF/c_x = 2mc_x$, which yields

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

The results can be generalized to 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 the velocity components in the x, y, and z directions, respectively. It is reasonable to assume that these components are equal, and 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 the force per unit area perpendicular to the force vector, $P = F/A$. Then the pressure exerted by N particles 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 the 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

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

where $\tilde{n}$ is the number of molecules per unit volume.

In this simple model (ideal gas model), we have neglected the effect of molecular interaction and the effect of the molecular size. However, for a gas at a room temperature, the result is surprisingly accurate. In a real system, each particle has a volume, and collisions take place between the particles.

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

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

where n is number of moles (equal to N/N_A), N_A is Avogadro's number, and R is the gas constant.

Equating (1.8) and (1.10) yields

$$(N/N_A)RT = (mc^2/2)(2/3)N\,. \tag{1.11}$$

By introducing the Boltzmann constant ($k = R/N_A = 1.38 \times 10^{-23}$ J/K), we can deduce that the kinetic energy KE of a gas is given by

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

It is interesting that the temperature Eq. (1.12) and pressure Eq. (1.9) in the macroscopic world are no more than scalings of the kinetic energy of the molecules in the microscopic world.

Another concept that needs to be clarified is the distribution function, which is the topic of the next section.

1.3 Phase Space

Coordinate systems (Cartesian, cylindrical, spherical) are usually used to describe the positions of the particles of a dynamical system as a function of time. To completely describe the dynamics of a system, it is important to know the position and velocity of its particles as a function of time. A coordinate system comprising position and velocity (or momentum, i.e., velocity multiplied by the mass of the system) can be used to express the dynamics of a system. Such a system is called phase space. In general, the dynamics of a point in the phase space is defined by three positions (say x, y, z) and three velocity components (u, v, w) or three momenta. The trajectory in the phase space represents the motion's evolution of the point over time. The following example illustrates a simple dynamical system.

Example: A system is composed of a spring vertically fixed at one end with the other end free to move. A body of mass m is attached to the free end. The spring stretches until the weight of the body balances the force exerted by the spring ($k\,l$, where k is the spring constant, or stiffness, and l is the stretched length of the spring due to the weight of the body), which is in the equilibrium condition. If the body is slightly displaced from the equilibrium state either upward or downward by a distance x, the spring will oscillate. The system's motion can be described by Newton's second law:

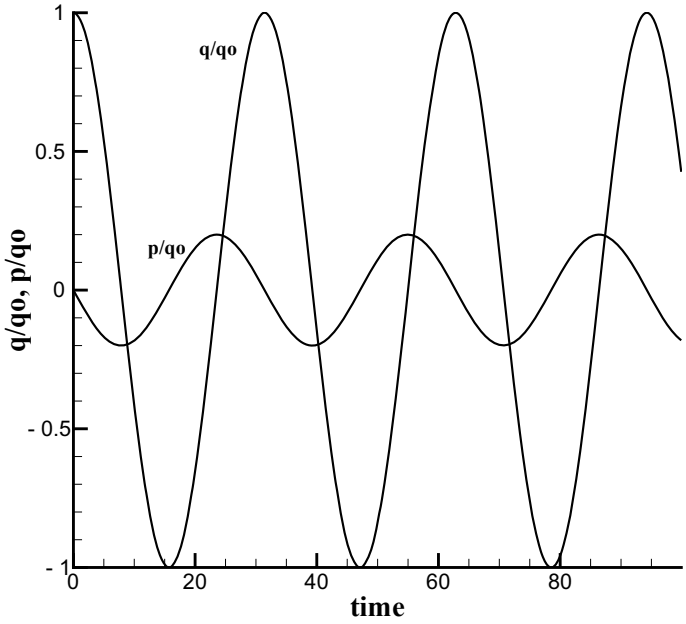


Fig. 1.4 q/q_o and p/q_o as functions of time for $\omega = 0.2$ and mass equal to 1.0

$$\frac{d(mv)}{dt} = -kx \tag{1.13}$$

Most textbooks on dynamics use q instead of x and p for the momentum, mv, where v is the velocity of the mass, $v = \frac{dq}{dt}$. Using this notation, the above equation can be rewritten as

$$\frac{dp}{dt} = k\, q, \quad \frac{dq}{dt} = \frac{p}{m}\,. \tag{1.14}$$

The solution for the above equation for a mass stretched through a distance q_o is

$$q = q_o\, \cos(wt) \quad \text{and} \quad p = -mwq_o\, \sin(wt)\,, \tag{1.15}$$

where $w = (k/m)^{1/2}$.

Figure 1.4 shows the variation of location of the mass q and the velocity of the mass p/m as a function of time.

However, there is another way to represent the dynamics of a moving body, namely by plotting the coordinates q and p, resulting in what is called a phase diagram. Each point on the diagram gives the location and velocity of the body. A point moves along an orbit, and the diagram is plotted by eliminating time t from the above equations. This can be done by squaring both sides of the equations and adding them. Since $\sin(wt)^2 + \cos(wt)^2 = 1$, it follows that

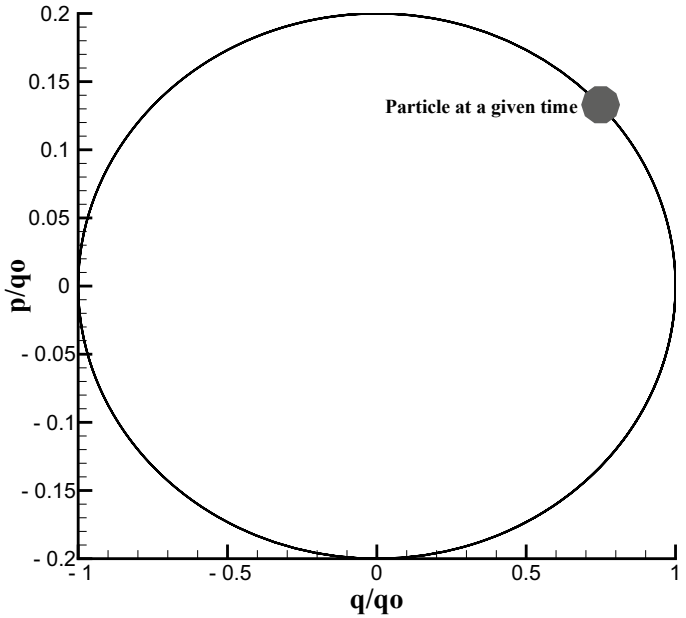


Fig. 1.5 Phase space, q/q_o versus p/q_o for $\omega = 0.2$ and mass equal to 1.0

$$p^2 + mkq^2 = mkq_o^2 \,. \tag{1.16}$$

This is the equation of an ellipse for $mk \neq 1$, and it represents a circle for $mk = 1$. Indeed, we can define a new variable if we wish, say $\xi = \sqrt{mk}q$. Then the above equation can be converted into the equation of a circle regardless of the values of mk. Figure 1.5 shows the phase space diagram for a spring–body system. A point on the diagram gives the location and velocity of the body at an instant of time, as mentioned before. In fact, if we had a video of the diagram, we could identify the location and velocity of the body at a given time.

An alternative approach to solving this problem is to use the energy method. When the spring is stretched a distance q_o from the equilibrium condition, potential energy is stored in the spring (potential energy $= \frac{kq_o^2}{2}$). As the spring is released from rest, $v_o = 0$ at time $t = 0$, part of the potential energy is converted into kinetic energy ($\frac{mv^2}{2}$ or $\frac{p^2}{2m}$). Since energy is conserved, assuming that there is no dissipation, the total energy is equal to the sum of the kinetic and potential energies at any instant of time, i.e.,

$$\frac{p^2}{2m} + \frac{kq^2}{2} = \frac{kq_o^2}{2} \,. \tag{1.17}$$

Rearranging the equation yields,

$$p^2 + mkq^2 = mkq_o^2 \,, \tag{1.18}$$

which is the same result as obtained before.

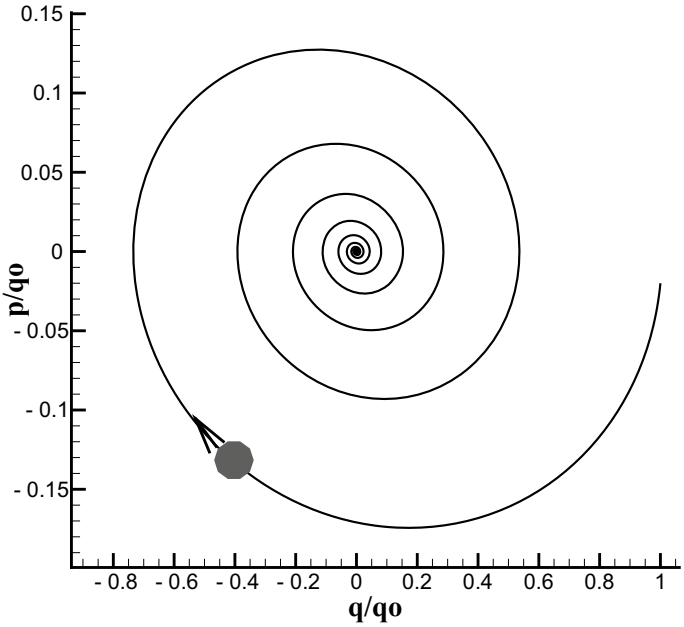


Fig. 1.6 Phase diagram for a dissipative system

Figure 1.5 shows a system without energy dissipation, which is an ideal system. In reality, energy dissipates (is converted to heat, due to friction) as the process continues until all the stored energy dissipates and the system reaches an equilibrium condition (balance between gravitational force and force exerted by the spring). Figure 1.6 shows the phase space diagram of a dissipative system.

It is difficult to visualize a phase space in higher dimensions; however, the concept is the same.

1.3.1 Molecular Distribution in Phase Space

In three-dimensional Cartesian space, the location of a point is usually identified by its x, y, and z coordinates. Suppose a small box of volume $\Delta V = \Delta x\ \Delta y\ \Delta z$ located at x, y, and z contains a number of molecules denoted by $\Delta^3 N$. The superscript refers to the three-dimensional space. The location of the box in space can be identified by a vector $\mathbf{r}$, which is called the position vector. The density of particles (number of particles per unit volume) in the box is $\Delta^3 N/\Delta^3 r$, $\Delta^3 r = \Delta x \Delta y \Delta z$. In the limit, the volume becomes infinitesimally small, and the particle number density can be represented as

$$n(\mathbf{r}) = \lim_{\Delta^3 r \to 0} \frac{\Delta^3 N}{\Delta^3 r} = \frac{d^3 N}{d^3 r}\,, \tag{1.19}$$

where $d^3 r = dxdydx$.

The number of particles in the volume can be written in terms of density as

$$d^3N = n(\mathbf{r})d^3r = n(\mathbf{r})dxdydx\,. \tag{1.20}$$

The above concept can be extended to higher dimensions in the momentum space (phase space). In other words, in the phase space, the particle's dynamics is identified by its location (three dimensions, x, y, and z coordinate) and its velocity, v_x, v_y, and v_z (x, y, and z velocity components). Hence the number of particles that can be found at a location between x and $x + dx$, y and $y + dy$, and z and $z + dz$ that have velocities between v_x and $v_x + dv_x$, between v_y and $v_y + dv_y$, and between v_z and $v_z + dv_z$ is d^6N. The system is in a six-dimensional space, which is called a hyperspace. Note that $N(\mathbf{r}, \mathbf{v})$ is a function of the position vector $\mathbf{r}$ and velocity vector $\mathbf{v}$. The density of the particles in the hyperspace can be written as

$$f(\mathbf{r}, \mathbf{v}) = \lim_{\Delta\mathbf{r}, \Delta\mathbf{v} \to 0} \frac{\Delta^6 N}{\Delta^3 r \Delta^3 v}\,. \tag{1.21}$$

Hence

$$d^6N(\mathbf{r}, \mathbf{v}) = f(\mathbf{r}, \mathbf{v})d^3r d^3v = f(\mathbf{r}, \mathbf{v})dxdydzdv_x dv_y dv_z\,. \tag{1.22}$$

The density of particles in a box of size $dxdydz$ can be found by integrating f over all possible velocity ranges, i.e.,

$$n(\mathbf{r}) = \int_{-\infty}^{\infty} dv_x \int_{-\infty}^{\infty} dv_y \int_{-\infty}^{\infty} dv_z f(\mathbf{r}, \mathbf{v})d^3r = \int\int\int f(\mathbf{r}, \mathbf{v})d^3r\,. \tag{1.23}$$

Let us define a normalized distribution function as

$$\tilde{f}(\mathbf{r}, \mathbf{v}) = \frac{f(\mathbf{r}, \mathbf{v})}{n(\mathbf{r})} \tag{1.24}$$

from Eq. (1.23). Then

$$\int\int\int \tilde{f}(\mathbf{r}, \mathbf{v})d^3r = 1\,. \tag{1.25}$$

Hence the average value of a quantity Q in an infinitesimal volume $dxdydz$ (usually referred to as d^3r) is

$$< Q >= \int\int\int Q(\mathbf{v})\tilde{f}(\mathbf{r}, \mathbf{v})d^3v\,. \tag{1.26}$$

The above integration is over all possible velocities.

1.4 Distribution Function: An Example

Let us illustrate the concept of the distribution function with an example. We assume that a policeman is standing in a road with very heavy traffic, where the velocity is limited to 80 km/h and he/she is measuring the velocity of the passing cars. It is almost impossible to say that any particular cars has a speed of exactly 80 km/h. We may get one or a very few cars whose speed is very near to 80 km/h, but not exactly. Most of the cars have velocities between 70 and 90 km/h, and a few are below and above this range. We do not expect any car to have speed 20 km/h or below, or 120 km/h or above. The best idea for tracking the velocities is to divide the speed domain into intervals of say 10 km/h. Hence the speed intervals are 20–30 (inclusive), 30–40, 40–50, 50–60, 60–70, 70–80, 80–90, 90–100, 100–110, and 110–120, i.e., $\Delta w = 10$. For a given period of time (say an hour), the policeman has counted the cars within the given ranges as follows: 2, 5, 16, 22, 54, 60, 52, 30, 16, and 5, respectively (the total number of cars, N, is 262). No car was recorded with a speed below 20 km/h or above 120 km/h. Figure 1.7 shows a plot of the distribution function of the number of cars in the various speed ranges. We are free to choose a smaller interval, say 5 km/h instead of 10 km/h. Decreasing the interval size gives a smooth distribution function. However, the number of cars in each interval decreases, which yields a uniform (flat) distribution function.

Let ΔN_k represent the number of cars with speeds in the range between $w_k = k\Delta w$ and $w_{k+1} = (k+1)\Delta w$, where Δw is the selected speed interval. For instance, the number of cars (ΔN_1) between 20 km/h, ($k = 1$) and 30 km/h, ($k = 2$) is 2, as in the previous example. The fraction of cars per unit interval, which is called the normalized distribution function, is

$$f_k = \frac{\Delta N_k}{N \Delta w}, \tag{1.27}$$

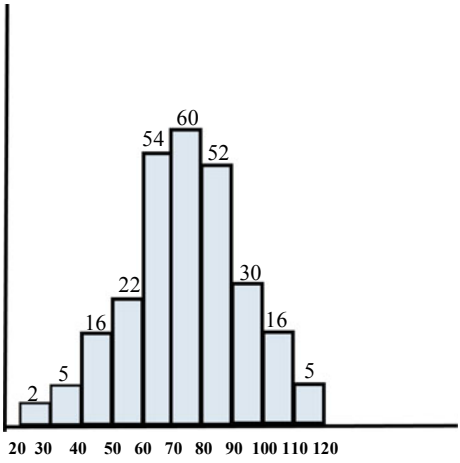


Fig. 1.7 Distribution function for the number of cars within a range of velocity

where N is the total number of cars. For the data given in the previous example,

$$f_1 = \frac{2}{262 * 10} = 0.0007633,\ f_2 = \frac{5}{262 * 10} = 0.001908,\ f_3 = \frac{16}{262 * 10} = 0.0061.... \tag{1.28}$$

The normalized distribution function f is a function of Δw. As Δw decreases, a smoother distribution function can be obtained, but not a flat function.

The reader may wonder why we did not define the fraction of the distribution function as $f_k = \frac{\Delta N_k}{N}$ instead $f_k = \frac{\Delta N_k}{N \Delta w}$. Such a choice would be inappropriate, because as the interval Δw decreases, the number of cars in each interval certainly decreases, and the distribution function would become flat.

Note that $\sum_{k=1}^{N} f_k \Delta w = 1$. In the limit of $\Delta w \to 0$,

$$\sum_{k=0}^{\infty} f_k \Delta w = \int_0^{\infty} f_k dw = 1\,, \tag{1.29}$$

which means that area under the curve f as a function of w is unity.

The average speed of all the cars can be calculated by multiplying the average of the cars' speeds for a given interval by the number of the cars in that interval, divided by the total number of cars. The results are summed over all intervals:

$$< w >= \sum_{k=0}^{\infty} \frac{w_k + w_{k+1}}{2} \frac{\Delta N_k}{N} = \sum_{k=0}^{\infty} (w_k + \Delta w/2) \frac{\Delta N_k}{N \Delta w} \Delta w\,. \tag{1.30}$$

Hence

$$< w >= \sum_{k=0}^{\infty} (w_k + \Delta w/2)\, f_w \Delta w\,. \tag{1.31}$$

As $\Delta w \to 0$, the above equation can be rewritten as

$$< w >= \int_0^{\infty} w_k f_w dw\,. \tag{1.32}$$

In general, the average value of any quantity Q is defined as

$$< Q >= \int_0^{\infty} Q(w) f(w) dw\,. \tag{1.33}$$

In other words, the average value of a quantity can be obtained by multiplying the value of the quantity by the probability density function (distribution function) and performing summation (integration) over all the values.

1.5 The Maxwell–Boltzmann Distribution Function

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

Maxwell's idea was that the knowledge of the velocity and position of each molecule at every instant of time is not that important. The distribution function is the important parameter for characterizing the effect of the molecules (the percentage of molecules in a certain location of a container that have velocities within a certain range, at a given instant of time). The molecules of a gas have a wide range of velocities as they collide with each other, with the faster molecules transferring momentum to the slower ones, and so on. The result of a collision is that momentum is conserved. For a gas in thermal equilibrium, the distribution function should not be a function of time when 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 $c_x + dc_x$ is $Nf(c_x)dc_x$. The function $f(c_x)$ is the fraction of the particles having velocities in the interval from c_x to $c_x + dc_x$ in the x-direction. Similarly, for other directions, the probability distribution function can be defined as before. Then the probability of the particles' velocities lying between c_x and $c_x + dc_x$, c_y and $c_y + dc_y$, and c_z and $c_z + dc_z$ is $Nf(c_x)f(c_y)f(c_z)dc_xdc_ydc_z$.

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

$$\iiint f(c_x)f(c_y)f(c_z)dc_xdc_ydc_z = 1. \tag{1.34}$$

Note that f is normalized by N.

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,

$$f(c_x)f(c_y)f(c_z) = \Phi(c_x^2 + c_y^2 + c_z^2)\,, \tag{1.35}$$

where Φ is a function to be determined. The value of the distribution function should be positive (between zero and unity). Hence in the equation above, the velocity is squared to avoid a negative magnitude. The possible functions that have the property of converting multiplication to addition are the logarithmic and exponential functions,

i.e.,

$$LogA + LogB = Log(AB) \tag{1.36}$$

and

$$e^A e^B = e^{(A+B)}\,. \tag{1.37}$$

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

$$f(c_x) = Ae^{-Bc_x^2}\,, \tag{1.38}$$

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

For example:

$F(x) = e^{Bx}$, $F(y) = e^{Cy}$, 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 which case the product of the functions is not equal to the sum of the functions.

Accordingly, it can be assumed that

$$f(c) = Ae^{-Bc_x^2}Ae^{-Bc_y^2}Ae^{-Bc_z^2} = A^3e^{-Bc^2}\,. \tag{1.39}$$

The probability distribution for the three directions gives the distribution in terms of a particle's speed c. In other words, the distribution function is the number of particles having speed between c and $c + dc$.

It is important to think about the distribution of particles in the 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 the two shells of the sphere, with radii c and $c + dc$; Fig. 1.8.

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

$$f(c)dc = 4\pi c^2 A^3 e^{-Bc^2}dc\,. \tag{1.40}$$

The constant A can be determined by integrating the probability distribution function over all possible speeds, i.e.,

$$\int_0^\infty f(c)dc = 4\pi A^3 \int_0^\infty c^2 e^{-Bc^2}dc = 1\,. \tag{1.41}$$

The integration can be carried out by parts as

$$\int_0^\infty c^2 e^{-Bc^2}dc = -\left[\frac{c}{2B}e^{-Bc^2}\right]_0^\infty + \frac{1}{4B}\int_0^\infty e^{-Bc^2}dc = \frac{1}{4B}\sqrt{\frac{\pi}{B}}\,. \tag{1.42}$$

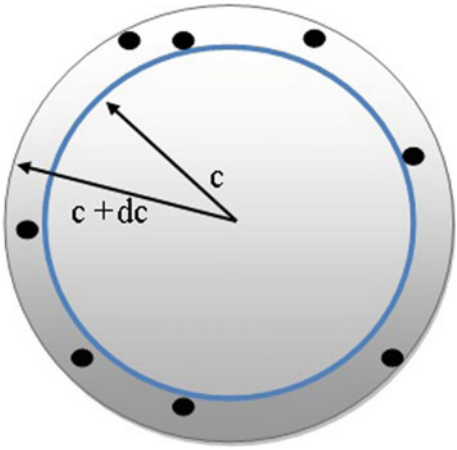


Fig. 1.8 Phase diagram

Therefore, $A = \sqrt{\frac{B}{\pi}}$.

The constant B can be found by calculating the average translational kinetic energy of the molecules for a gas in equilibrium, i.e.,

$$< E_k >= \int_0^\infty \frac{1}{2} mc^2 f(c) dc \,. \tag{1.43}$$

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

$$< \frac{1}{2} mc^2 >= \frac{\int_0^\infty \frac{1}{2} mc^2 f(c) dc}{\int_0^\infty f(c) dc} \,. \tag{1.44}$$

The numerator is the total energy, and the denominator is equal to one. Note that the unknown constant A cancels between numerator and denominator. Substituting the value of $f(c)$ into the integrals yields

$$< \frac{1}{2} mc^2 >= \frac{3m}{4B} \,. \tag{1.45}$$

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

$$< \frac{1}{2} mc^2 >= \frac{3}{2} kT \,. \tag{1.46}$$

Hence $B = m/2kT$, so

$$f(c) \propto c^2 e^{-\frac{mc^2}{2kT}} \,. \tag{1.47}$$

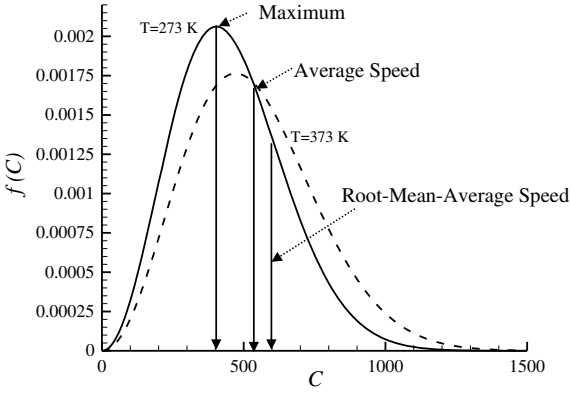


Fig. 1.9 Probability distribution of nitrogen gas as a function of molecular velocity, c

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

The final result is

$$f(c) = 4\pi \left(\frac{m}{2\pi kT}\right)^{\frac{3}{2}} c^2 e^{-\frac{mc^2}{2kT}} . \tag{1.48}$$

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

The probability of finding a particle that has a specific velocity is zero, because the velocities of the particles change continuously over a wide range. The meaningful question is to find the probability of a particle or particles being within a range of velocities rather than at a specific velocity. Therefore, Eq. (1.48) needs to be integrated over that range of velocity.

Example:

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

The mass of one molecule of N_2, which is the 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 the mass of one N_2 molecule. The Boltzmann constant is 1.38×10^{-23} J/K (kg m^2/(s^2.K)).

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

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

The area under each curve is unity. As the temperature increases, the number of molecules that have high velocities increases. The most probable speed is equal to (Fig. 1.9)

$$\sqrt{\frac{2kT}{m}}\,. \tag{1.49}$$

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

The average speed is equal to

$$< c >= \sqrt{\frac{8kT}{\pi m}}\,, \tag{1.50}$$

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

$$< c >= \int_0^\infty cf(c)dc\,. \tag{1.51}$$

The root-mean-average speed is equal to

$$< c^2 >= \int_0^\infty c^2 f(c)dc = \frac{3kT}{m}\,. \tag{1.52}$$

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

The root-mean-squared (rms) speed of a molecule is

$$c_{rms} = \sqrt{< c^2 >} = \sqrt{\frac{3kT}{m}}\,. \tag{1.53}$$

Lighter gases usually have faster molecular speeds than heavier gases (higher molecular weight). Figure 1.10 illustrates the probability function for H_2 (2 kg/kmol), N_2 (28 kg/kmol), and CO_2 (44 kg/kmol).

At room temperature, most N_2 molecules travel at speeds around 500 m/s, while most hydrogen molecules travel at speeds around 1600 m/s (supersonic rockets).

Exercise 1: Calculate the average and rms speed of a hydrogen molecule at room temperature (25 °C).

Exercise 2: For a Maxwellian distribution function, calculate the total number of molecules striking a unit area of a wall per unit time. What will be the pressure on the wall?

Exercise 3: Using a Taylor series, expand Eq. (1.48) in terms of c up to three terms.

1.5.1 Boltzmann's Contribution

Ludwig Boltzmann generalized Maxwell's distribution for arbitrarily 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 number of available microscopic states for a given energy is 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 those in which all the gas molecules are in 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 over a time period 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 and then remove the piston, they will rapidly tend to a uniform distribution spread evenly throughout the box (Fig. 1.10).

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 that were rather troublesome but were not fatal to the program. For example, how many different velocities can a particle in a box have? This matter was cleared up by 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 $e^{-E/kT}$, i.e.,

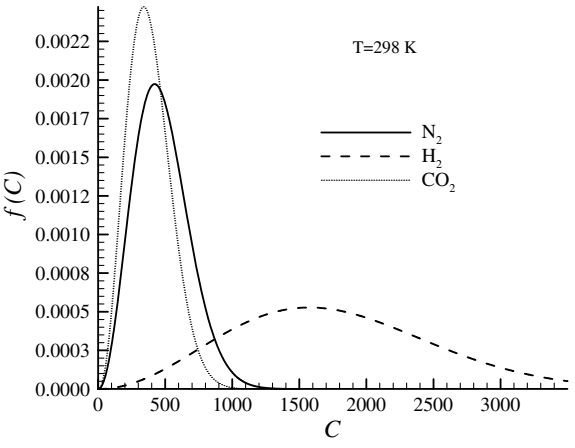


Fig. 1.10 Distribution function for N_2, H_2, and CO_2 at room temperature

$$f(E) = Ae^{-E/kT} . \tag{1.54}$$

This is called the Boltzmann distribution.

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

$$E = \frac{1}{2}mc_x^2 . \tag{1.55}$$

For a normalized probability function, the probability function integrated over all values of the velocity (from minus to plus infinity) should yield the value one. Hence

$$\int_{-\infty}^{\infty} Ae^{-\frac{mc_x^2}{2kT}} dc = 1 , \tag{1.56}$$

and therefore,

$$A = \sqrt{\frac{m}{2\pi kT}} . \tag{1.57}$$

The probability of velocity being c_x is

$$f(c_x) = \sqrt{\frac{m}{2\pi kT}} e^{-\frac{mc_x^2}{2kT}} . \tag{1.58}$$

We are interested in the probability of a three-dimensional velocity c, where

$$c^2 = c_x^2 + c_y^2 + c_z^2 . \tag{1.59}$$

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

$$f(c) = f(c_x) f(c_y) f(c_z) , \tag{1.60}$$

which leads to

$$f(c) = \left[\sqrt{\frac{m}{2\pi kT}}\right]^3 e^{-\frac{m}{2kT}(c_x^2+c_y^2+c_z^2)} , \tag{1.61}$$

or

$$f(c) = \left(\frac{m}{2\pi kT}\right)^{3/2} e^{-\frac{mc^2}{2kT}} . \tag{1.62}$$

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 the surface area of a sphere in the phase space) to account for the density of velocity states available to particles. Therefore, the Maxwell distribution function Eq. (1.48) is covered. In fact, integration of the

Maxwell distribution function Eq. (1.48) over the surface of a sphere in phase space yields Eq. (1.62).

Also, notice that in deriving the above equation, it is assumed that the gas in the container is not macroscopically moving relative to the container. If the gas is moving with a velocity **u** relative to the container, in this case the vector **c** must be replaced with $\mathbf{c} - \mathbf{u}$, and hence

$$f(c) = \left[\sqrt{\frac{m}{2\pi kT}}\right]^3 e^{-\frac{m}{2kT}[(c_x-u_x)^2+(c_y-u_y)^2+(c_z-u_z)^2]}, \tag{1.63}$$

where u_x, u_y, and u_z are macroscopic velocity components in the x, y, and z directions, respectively.

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

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

Exercise: using a Taylor series, expand Eq. (1.63).

1.6 Liouville's Theorem

In classical mechanics, mass is a conservative quantity. The total rate of change of the mass of a system is zero. The continuity equation for fluid mechanics can be written in Cartesian coordinates as

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} + \frac{\partial \rho w}{\partial z} = 0, \tag{1.64}$$

where ρ, u, v, and w are density and the x-, y-, and z-components of velocity, respectively. The above equation states that the rate of change of the mass of a system is due to the influx and outflux of a flow. For an incompressible flow (ρ is a constant), the above equation can be written as

$$\nabla \bullet \vec{V} = 0. \tag{1.65}$$

The above equation implies that the local volume dilation rate is zero.

In phase space, Liouville's theorem states that the rate of change of a volume is zero. An ensemble of particles moves along a set of trajectories without interference. In a volume with many particles moving along trajectories without collision, the density (number of particles per unit volume) in phase space is constant. In general, the density is a function of time t, position q and momentum p (mass multiplied by

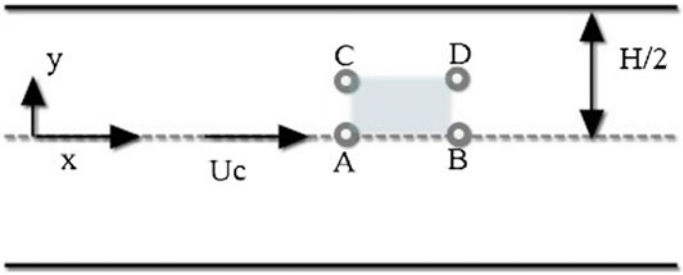


Fig. 1.11 Volume is conserved

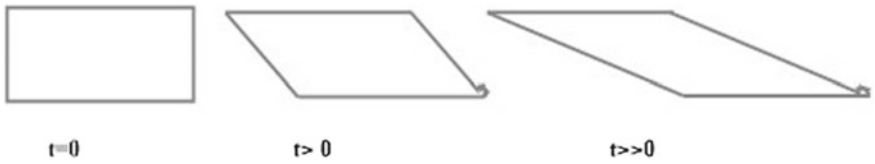


Fig. 1.12 Volume is conserved

velocity). Using the chain role of differentiation yields

$$d\rho = \frac{\partial \rho}{\partial t} dt + \frac{\partial \rho}{\partial q} dq + \frac{\partial \rho}{\partial p} dp \,. \tag{1.66}$$

Dividing the above equation by dt and setting $\frac{d\rho}{dt} = 0$ results in

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho}{\partial q}\frac{dq}{dt} + \frac{\partial \rho}{\partial p}\frac{dp}{dt} = 0 \,. \tag{1.67}$$

The above equation is called Liouville's equation. The term $\frac{dq}{dt}$ is the velocity, and the term $\frac{dp}{dt} = \frac{d(mV)}{dt}$ is the force F (Newton's second law). Hence the above equation can be written in a more generalized form as

$$\frac{\partial \rho}{\partial t} + V.\nabla_q \rho + F.\nabla_p \rho = 0 \tag{1.68}$$

where $\nabla_q = \frac{\partial}{\partial q_1} + \frac{\partial}{\partial q_2} + \frac{\partial}{\partial q_3} + \cdots$ and $\nabla_p = \frac{\partial}{\partial p_1} + \frac{\partial}{\partial p_2} + \frac{\partial}{\partial p_3} + \cdots$.

To illustrate Liouville's theorem, let us solve a very simple problem (Figs. 1.11 and 1.12).

Example:
The velocity profile for a steady, incompressible, fully developed flow in a channel is

$$u = u_c(1 - (2y/H)^2) \,, \tag{1.69}$$

where u_c is the velocity at $y = 0$ (centerline velocity, maximum velocity), y is the vertical coordinate system from the center of the channel, H is the channel height. Flow in an ensemble contains four particles A, B, C, D. The x and y locations of those particles are selected randomly. At time 0, let the locations of these particles be as follows: $A(0, 0)$, $B(1, 0)$, $C(0, H/4)$, $D(1, H/4)$, i.e., the location of A is at $x = 0$ and $y = 0$. The location of B is at $x = 1$, $y = 0$. The location of C is at $x = 0$ and $y = H/4$, etc. The volume (area) enclosing those particles is 1.0 multiplied by $H/4$. Do particles are moving with their velocities. Let $u_c = 1$. Then the particles A and B are moving at a constant velocity equal to 1. Particles C and D at the same y location are moving with a constant velocity of 0.75. At a period of time, say $t = 5$, The x locations of those particles will be calculated as initial x plus the velocity multiplied by time. Therefore, A will move to the location $x = 5$, B moves to $x = 6$, C moves to $x = 3.75$, and D moves to $x = 4.75$. The y locations of those particles will not change. Even if the shape is distorted, the volume (area) is the same (1.0 multiplied by $H/4$). The is volume always the same, even if the one dimension becomes attenuated as time increases.

Chapter 2
The Boltzmann Equation

Ludwig Eduard Boltzmann (1844–1906), the Austrian physicist whose greatest achievement was in the development of statistical mechanics, explains and predicts how the properties of atoms and molecules (microscopic properties) determine the phenomenological (macroscopic) properties of matter such as viscosity, thermal conductivity, and the 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 dynamics simulations. The method offers an enormous savings in computer resources.

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 a distribution function $f(r, c, t)$ giving the number of molecules at time t positioned between r and $r + dr$ that have velocities between c and $c + dc$, as mentioned in the previous chapter. An external force F that acts on a gas molecule of unit mass will change the velocity of the molecule from c to $c + Fdt$ and its position from r to $r + cdt$ (Fig. 2.1).

The number of molecules $f(r, c, t)$ before the external force is applied is equal to the number of molecules after the disturbance, $f(r + cdt, c + Fdt, t + dt)$, if no collisions take place between the molecules. Hence

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

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

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3_2

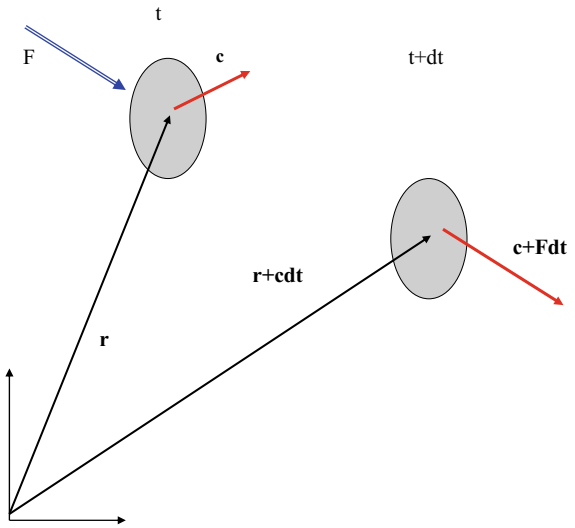


Fig. 2.1 Position and velocity vector for a particle after and before applying a force F

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

Dividing the above equation by $dtdrdc$ and taking the limit $dt \to 0$ yields

$$\frac{df}{dt} = \Omega(f)\,. \tag{2.3}$$

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

$$df = \frac{\partial f}{\partial r}dr + \frac{\partial f}{\partial c}dc + \frac{\partial f}{\partial t}dt\,. \tag{2.4}$$

Dividing by dt yields

$$\frac{df}{dt} = \frac{\partial f}{\partial r}\frac{dr}{dt} + \frac{\partial f}{\partial c}\frac{dc}{dt} + \frac{\partial f}{\partial t}\,. \tag{2.5}$$

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

Equation (2.5) can be written as

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

where a is equal to dc/dt, and the acceleration can be related to the force F by Newton's second law, $a = F/m$.

The Boltzmann transport equation (2.3) can be written as

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

Here Ω is a complex function of f and must be known in order to solve the Boltzmann equation.

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

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

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 are 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 integrodifferential equation that is difficult to solve.

The collision operator Ω can be expressed as the rate of increase in the number of molecules in the domain due to the collision (gain) minus the rate of loss in the number of molecules due to the collision (lost). To illustrate this concept, let us consider an example. Assume that there are 100 molecules in a given space (between r and $r + dr$). Those molecules have velocities in the range, say 2 m/s (c) and 4 m/s ($c + dc$). Due to collision, say ten molecules increased their to more than 4 m/s, and five molecules' velocities decreased below 2 m/s. Also, in the same volume, there were other molecules with velocity not in the range 2–4 m/s (out of our velocity domain), let us say three molecules' velocities come into the range between 2 and 4 m/s due to collision. Now the gain is three molecules, the loss is fifteen ($10 + 5$), and the net is -12 molecules. Since the system is dynamic, it is better to represent the results as a rate of change.

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

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

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

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

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

Equations (2.9)–(2.11) are conservation of mass, momentum, and energy, respectively.

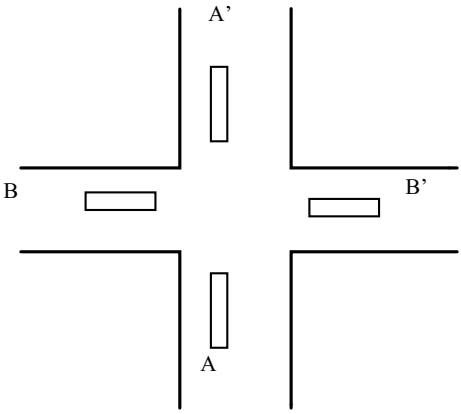


Fig. 2.2 Traffic intersection

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 *Illustrative Example 2.1*

At a road intersection, if there are no collisions, the cars from section A move (stream) to A', and cars from section B move to B' without any problem. This case is represented in 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 condition) depends on the type of collision and the response of the police officer (relaxation time). In this case, Ω is not zero.

2.2 The BGK Approximation

It is difficult to solve the 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 a simple operator without introducing a significant error in the outcome of the solution. In 1954, Bhatnagar, Gross, and Krook (BGK) introduced a simplified model for the collision operator. At the same time, Welander (1954) independently introduced a similar operator. The collision operator is replaced by

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

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 the Maxwell–Boltzmann distribution function.

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

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

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

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

The above equation is the workhorse of the lattice Boltzmann method and replaces the Navier–Stokes equation in CFD simulations. It is possible to derive the Navier–Stokes equation from the Boltzmann equation. We can comment on Eq. (2.15) as follows:

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

Equation (2.15) can be discretized as

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

The local equilibrium distribution function with a relaxation time determines the type of problem to be solved. The beauty of this equation lies in its simplicity, and it can be applied to many physical situations 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, and advection–diffusion without and with external forces, will be presented in the following chapters.

It is possible to use a finite difference or finite volume to solve partial differential equation (2.15). Some authors have used this approach to solve fluid dynamics problems on nonuniform grids. The main focus of this book is to solve Eq. (2.15) in two steps: collision and streaming.

A few steps need to be done to solve a problem using LBM. In any numerical method, the solution domain needs to be discretized either to nodes, elements, control volume, or lattices. In LBM, the solution domain needs to be discretized into lattices.

The following sections will discuss in some detail the lattice arrangements and setting the weighting factors for the lattice linkages.

2.3 Lattice Arrangements

The common terminology used in LBM is to refer to the dimension of the problem and the number of streaming directions (speed) used as $DnQm$, where n represents 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, the 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, three models can be used for 1-D lattice arrangements, called D1Q2, D1Q3, and D1Q5, as shown in Fig. 2.3. D1Q3 is the most popular one. The black node is the central node, while the gray nodes are neighboring nodes. For the model D1Q2, there is no particle residing on the site, while for D1Q3 and D1Q5, a particle is residing on the site all the time. Hence it has zero velocity (stationary particle). The factitious particles stream from the central node to neighboring nodes through linkages with a specified speed, called the lattice speed.

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 are equal to 0, 1, and -1, respectively. Note that we have assumed that $\Delta x = \Delta t$; 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. The total number of factitious particles at any instant of time is three. 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. Since those particles are streaming the same distance, they have the same weighting factors. The weighting factors ω_i have values of $4/6$, $1/6$, and $1/6$ for f_0, f_1, and f_2, respectively. In the following section, we discuss in detail how we can determine the weighting factors and sound speed. The speed of sound, c_s, in lattice units for D1Q3 is $1/\sqrt{3}$. It is also possible to use another arrangement called D1Q2. The weighting factors ω_i have 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.

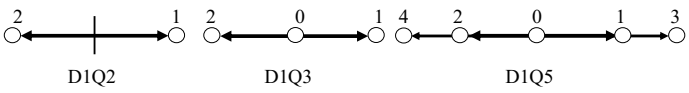


Fig. 2.3 Lattice arrangements for 1-D problems

However, it is possible to involve more sites and use higher-order schemes, such as D1Q5. Note that D1Q3 is more stable than D1Q2.

Example: One-dimensional problem with two lattice velocities, one streaming to the right, $c_1 = 1$, and the other streaming to the left, $c_2 = -1$. First condition: $w_1 + w_2 = 1$; second condition: $w_1 c_1 + w_2 c_2 = 0$. Hence $w_1 = w_2$, and from the first condition, $w_1 = w_2 = 1/2$. Third condition:

$$\sum w_i c_{i\alpha} c_{i\beta} = c_s^2 \delta_{\alpha\beta}\,, \tag{2.17}$$

$$w_1 c_{1\alpha} c_{1\beta} + w_2 c_{2\alpha} c_{2\beta} = c_s^2\,.\delta_{\alpha\beta} \tag{2.18}$$

The Kronecker delta, $\delta_{\alpha\beta}$, is equal to 1 if $\alpha = \beta$; otherwise, it is zero. In all cases, we have one dimension, $c_{11} = c_1$ and $c_{22} = c_2$; therefore, $c_s^2 = 1/2$.

D1Q5

For this arrangement, the total number of factitious particles at any instant of time is 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 sound in lattice units is $1/\sqrt{3}$.

2.3.2 Two-Dimensional

D2Q5 and D2Q4

The D2Q5 model has four velocity vectors issuing from the central nodes; Fig. 2.4. One of the particles resides at the central node; hence its speed is zero, denoted by $c(0, 0)$. The distribution functions $f_1$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 $\Delta x = \Delta y = \Delta t$. 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 worth mentioning that this arrangement cannot be used to simulate fluid flows. This issue will be discussed latter on. D2Q4 has four velocity vectors, and there are no particles residing on the center 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 a computer programming algorithm, proper numbering may simplify the computer code. We have found that D2Q5 is more stable than D2Q4.

D2Q9

This model is very common, especially for solving fluid flow problems. It has nine velocity vectors, with the central particle speed equal to zero; Fig. 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 the corresponding distribution functions are 4/9, 1/9, 1/9, 1/9, 1/9, 1/36, 1/36, 1/36, and 1/36.

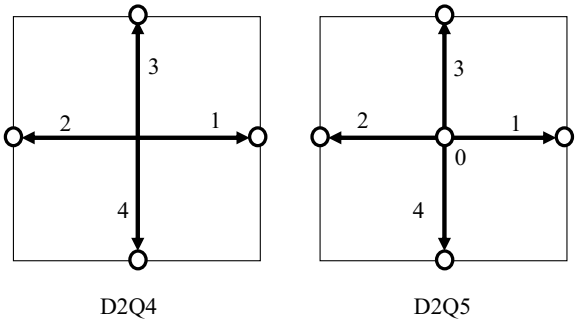


Fig. 2.4 Lattice arrangements for 2-D problems, D2Q4 and D2Q5

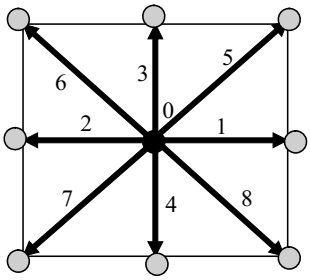


Fig. 2.5 Lattice arrangements for 2-D problems, D2Q9

2.3.3 Three-Dimensional

In general, two models are used in simulating three-dimensional problems: D3Q15 and D3Q19.

D3Q15

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

Note that nodes 1, 2, 3, and 4 are at the centers 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}.

D3Q19

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

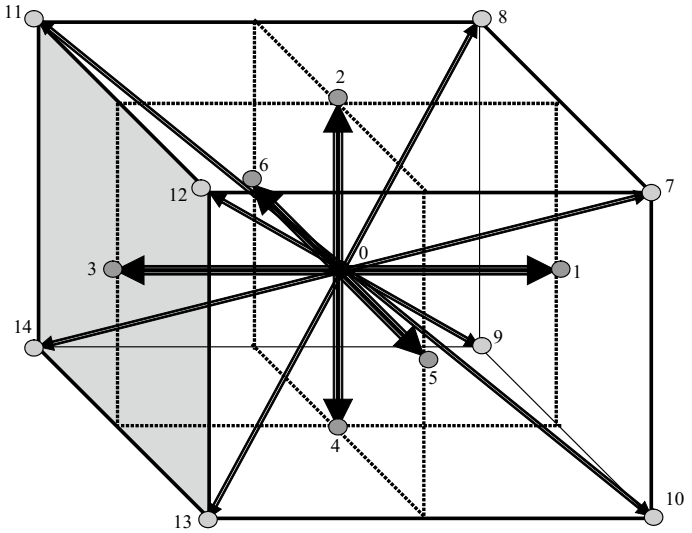


Fig. 2.6 Lattice arrangements for 3-D problems, D3Q15

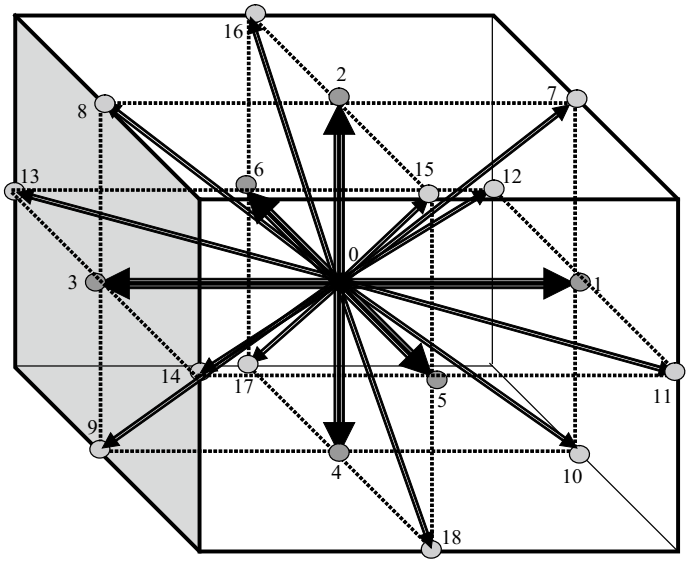


Fig. 2.7 Lattice arrangements for 3-D problems, D3Q19

The weighting factors are as follows:
for f_0 it is 12/36,
for f_1 to f_6 it is 2/36, and
for f_7 to f_{18} it is 1/36.

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

The next step is to determine the weighting factor for each lattice linkage.

2.3.4 Summary

For 1-D problems, it is best to use D1Q3, which is found to be more stable than using D1Q2. For 2-D, it is recommended to use D2Q5 for energy and species conservation equations. Of course, a greater number of lattice links can be used, such as D2Q9, without any problem. However, for fluid flow, at least D2Q9 must be used to ensure a Galilean invariant. For 3-D, it is recommended to use D3Q7 for energy equation and other scalar quantities, such as the species conservation equation. For fluid flow, D3Q17 or a higher number of lattice links must be used, such as D3Q19, which is commonly used.

2.4 Weighting Factors

In LBM, the solution domain needs to be divided into lattices, as mentioned before. The factitious particles (distribution function) reside at each lattice node,, usually more than one. Be aware that the method is different from classical methods such as the finite difference method. For example, in the finite difference method, velocity components, temperature, etc., are unknowns on each node. In LBM, there are a few distribution functions (particles) f_1, f_2, and so on that depend on the dimension and nature of the problem and are unknown. Some of these particles stream (move) along specified directions to the neighboring nodes. The number of directions, linkages, depends on the lattice arrangement, as mentioned before.

Figure 2.5 illustrates a lattice node with eight particles (distribution functions) streaming with velocity c in different directions, and a particle is residing on the site. We use the notation $c_{i\alpha}$ to represent the stream velocity, with the subscript i to identify the particle and α to identify the velocity components in the x- and y-directions, in 2-D. For example, particle 1 streams to the east and has only an x-component. On the other hand, particle 2 streams to the northeast and has x- and y-components.

The isotropy of lattices must be conserved in selecting the weighting factor for each lattice direction. Also, the lattice speed of sound should be properly calculated within the constraints of isotropy. In general, the weighting factors are needed for weighting each distribution function. The distribution function residing on the site has more weight than other distribution functions. A distribution function streaming a short distance should have more weight than a function streaming to a longer distance. The distribution functions streaming the same distance, regardless of their direction, should have the same weight. Also, the sum of all weighting factors should

be one. If you divide a pie into pieces, the sum of the pieces must be equal to the whole pie, regardless of how you divide the pie.

The following conditions must be satisfied:

The sum of all weighting factors must equal one:

$$\sum w_i = 1\,. \tag{2.19}$$

Symmetry must be conserved (independent of the rotation of the lattices),

$$\sum w_i c_{i\alpha} = 0, \tag{2.20}$$

isotropy,

$$\sum w_i c_{i\alpha} c_{i\beta} = c_s^2 \delta_{\alpha\beta}, \tag{2.21}$$

$$\sum w_i c_{i\alpha} c_{i\beta} c_{i\gamma} = 0, \tag{2.22}$$

$$\sum w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} = c_s^4 (\delta_{\alpha\beta}\delta_{\gamma\delta} + \delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma}), \tag{2.23}$$

$$\sum w_i c_{i\alpha} c_{i\beta} c_{i_\gamma} c_{i\delta} c_{i\epsilon} = 0, \tag{2.24}$$

and so on.

In the following sections, examples will be given to illustrate how to apply the above equations to calculate the weighting factors and lattice sound speed.

For example, D1Q3:

$$c_i = 0, \pm 1\,. \tag{2.25}$$

By substituting the lattice velocity in Eqs. (19), (20), and (21), the following equations are obtained:

$$w_0 + w_1 + w_2 = 1\,, \tag{2.26}$$

$$w_1 - w_2 = 0\,, \tag{2.27}$$

$$w_1 + w_2 = c_s^2\,. \tag{2.28}$$

The previous equations can be solved to find the weights as a function of c_s^2:

$$w_0 = 1 - c_s^2 \tag{2.29}$$

$$w_1 = w_2 = \frac{c_s^2}{2}\,. \tag{2.30}$$

It should be noted that when $c_s^2 = 1$, the weight of the rest distribution will vanish, and the D1Q2 model is obtained. However, if we use Eq. (23), a different value of c_s^2 can be obtained:

$$w_1 + w_2 = 3c_s^4 \,. \tag{2.31}$$

The value of c_s^2 can be found by solving Eq. (31) with Eqs. (29) and (30). By considering the positive real nontrivial solution, we end up with $c_s^2 = 1/3$ and $w_o = 2/3$, $w_1 = w_2 = 1/6$. Selecting $c_s^2 = 1/3$ is preferable, because it satisfies more symmetry and isotropy conditions.

Another example, D1Q5: let us assume that D1Q5 has velocities $c_i = 0, \pm 1, \pm a$, which need to be determined.

We use the equations for the symmetry and isotropy conditions:

$$w_0 + w_1 + w_2 + w_3 + w_4 = 1 \,, \tag{2.32}$$
$$w_1 - w_2 + w_3 - w_4 = 0 \,, \tag{2.33}$$
$$w_1 + w_2 + a^2 w_3 + a^2 w_4 = c_s^2 \,, \tag{2.34}$$
$$w_1 - w_2 + a^3 w_3 - a^3 w_4 = 0 \,, \tag{2.35}$$
$$w_1 + w_2 + a^4 w_3 + a^4 w_4 = 3c_s^4 \,. \tag{2.36}$$

Solving the above equations yields $w_1 = w_2$, $w_3 = w_4$, and

$$w_0 = \frac{3c_s^4 - c_s^2(1 + a^2) + a^2}{a^2} \,, \tag{2.37}$$
$$w_1 = w_2 = \frac{c_s^2(a^2 - 3c_s^2)}{2(a^2 - 1)} \,, \tag{2.38}$$
$$w_3 = w_4 = \frac{c_s^2(3c_s^2 - 1)}{2a^2(a^2 - 1)} \,. \tag{2.39}$$

To obtain a solution, D1Q4 is set to $w_o = 0$ and solving Eq. (2.37) yields

$$c_s^2 = \frac{\pm\sqrt{a^4 - 10a^2 + 1} + a^2 + 1}{6} \,. \tag{2.40}$$

The minimum integer value of a that gives a positive real nontrivial c_s^2 is $a = 4$. So the D1Q4 model is achieved with $\boldsymbol{\xi}_i = 0, \pm 1, \pm 4$ and $c_s^2 = (\pm\sqrt{97} + 17)/6$.

The best value of c_s^2 for D1Q5 can be found using a higher momentum:

$$w_1 + w_2 + a^6 w_3 + a^6 w_4 = 15c_s^6 \,. \tag{2.41}$$

The value of c_s^2 can be found by solving Eqs. (2.32)–(2.41) and taking the positive real nontrivial solution:

$$c_s^2 = \frac{\pm\sqrt{9a^4 - 42a^2 + 9} + 3a^2 + 3}{30} \,. \tag{2.42}$$

Table 2.1 The weighting factors w_i and the speed of sound squared (c_s^2) for the D1Q2, D1Q3 models

Indices (i)		0	1	2
c		0	1	-1
Model	c_s^2			
D1Q2	1		1/2	1/2
D1Q3a	1/2	2/4	1/4	1/4
D1Q3b	1/3	4/6	1/6	1/6
D1Q3c	1/6	10/12	1/12	1/12

Table 2.2 The weighting factors w_i and the speed of sound squared (c_s^2) for the D2Q4, D2Q5 models

	i	0	1	2	3	4
	c_x	0	1	0	-1	0
	c_y	0	0	1	0	-1
Model	c_s^2					
D2Q4	1/2	0	1/4	1/4	1/4	1/4
D2Q5a	1/3	2/6	1/6	1/6	1/6	1/6
D2Q5b	1/4	4/8	1/8	1/8	1/8	1/8
D2Q5c	1/6	8/12	1/12	1/12	1/12	1/12

The minimum integer value of a that gives a positive real nontrivial c_s^2 is $a = 3$. So the D1Q5 model is achieved with $c_i = 0, \pm 1, \pm 3$, and $c_s^2 = (\pm\sqrt{90} + 15)/15$.

The same procedure can be followed to find the speed of sound and the weights for other models such as D2Q5 and D2Q9.

Tables 2.1 and 2.2 show some of the possible values for w_i and c_s for one- and two-dimensional problems, which are illustrated in the following sections.

Another gradient of LBM is to determine the equilibrium distribution function.

2.5 Equilibrium Distribution Function

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

For particles moving in a medium with macroscopic velocity $\mathbf{u}$, the Maxwell distribution function Eq. (1.63) can be written as

$$f^{eq} = \frac{\rho}{(2\pi RT)^{D/2}} exp\left(-\frac{(\vec{c}-\vec{u})^2}{2RT}\right) \tag{2.43}$$

$$= \frac{\rho}{(2\pi RT)^{D/2}} exp\left(-\frac{\vec{c}.\vec{c}-2\vec{c}.\vec{u}+\vec{u}.\vec{u}}{2RT}\right) \tag{2.44}$$

$$= \frac{\rho}{(2\pi RT)^{D/2}} exp\left(-\frac{\vec{c}.\vec{c}}{2RT}\right) exp\left(-\frac{-2\vec{c}.\vec{u}+\vec{u}.\vec{u}}{2RT}\right), \tag{2.45}$$

where D is the dimension of the problem, e.g., $D = 2$ for the two-dimensional problem. The exponential function can be expanded using a Taylor series:

$$exp(x) = 1 + x + x^2/2! + x^3/3! + \cdots . \tag{2.46}$$

So, Eq. (2.45) becomes

$$f^{eq} = \frac{\rho}{(2\pi RT)^{D/2}} exp\left(-\frac{\vec{c}.\vec{c}}{2RT}\right)\left[1 - \frac{-2\vec{c}.\vec{u}+\vec{u}.\vec{u}}{2RT} + \frac{(-2\vec{c}.\vec{u}+\vec{u}.\vec{u})^2}{4R^2T^2} + \cdots\right]. \tag{2.47}$$

Ignoring terms of order $O(u^3)$ and higher, the equilibrium distribution function can be approximated as

$$f^{eq} = \frac{\rho}{(2\pi RT)^{D/2}} exp\left(-\frac{\vec{c}.\vec{c}}{2RT}\right)\left[1 - \frac{-2\vec{c}.\vec{u}+\vec{u}.\vec{u}}{2RT} + \frac{(\vec{c}.\vec{u})^2}{2R^2T^2}\right]. \tag{2.48}$$

The well-known distribution function can be recovered by the following two substitutions: $RT = c_s^2$ and $W(\vec{c}) = \exp\left(-\frac{\vec{c}.\vec{c}}{2RT}\right)\ (2\pi RT)^{-D/2}$,

$$f^{eq} = \rho W(\vec{c})\left[1 + \frac{2\vec{c}.\vec{u}-\vec{u}.\vec{u}}{2c_s^2} + \frac{(\vec{c}.\vec{u})^2}{2c_s^4}\right]. \tag{2.49}$$

The equilibrium distribution function along a discretized velocity direction is

$$f_i^{eq} = \rho w_i\left[1 + \frac{2\vec{c}.\vec{u}-\vec{u}.\vec{u}}{2c_s^2} + \frac{(\vec{c}.\vec{u})^2}{2c_s^4}\right] + O(u^2). \tag{2.50}$$

A higher-order representation is

$$f_i^{eq} = \rho w_i\left[1 + \frac{2\vec{c}.\vec{u}-\vec{u}.\vec{u}}{2c_s^2} + \frac{(\vec{c}.\vec{u})^2}{2c_s^4} + \frac{(\vec{c}.\vec{u})^3}{2c_s^6} - \frac{(\vec{c}.\vec{u})u^2}{2c_s^4}\right] + O(u^4). \tag{2.51}$$

For a stationary medium ($u = 0$), $f_i^{eq} = \rho w_i$.

Another item left to discuss is the force term in LBM, which is the topic of the following section.

2.6 The Source Term

The BGK–Boltzmann transport equation with force term can be written as follows:

$$\left[\frac{\partial}{\partial t} + c_\alpha \frac{\partial}{\partial x_\alpha} + \frac{F_\alpha}{m}\frac{\partial}{\partial c_\alpha}\right] f(\boldsymbol{x}, \boldsymbol{c}, t) = \frac{1}{\tau}\left(f^{eq}(\boldsymbol{x}, \boldsymbol{c}) - f(\boldsymbol{x}, \boldsymbol{c}, t)\right), \quad (2.52)$$

where c_α, F_α, m, and f are the microscopic velocity, the external force, the mass, and the density function, respectively.

The force term $\frac{F}{m}\frac{\partial f}{\partial c_\alpha}$ cannot be evaluated, since the dependency of f on c_α is unknown. According to the assumption of the BGK collision, which states that f can deviate only slightly from equilibrium, it is safe to assume that $f \simeq f^{eq}$. Differentiation of Eq. (2.43) with respect c_α yields

$$\frac{\partial f(\boldsymbol{x}, \boldsymbol{c}, t)}{\partial c_\alpha} \approx \frac{\partial f^{eq}(\boldsymbol{x}, \boldsymbol{c})}{\partial c_\alpha} = -\frac{c_\alpha - u_\alpha}{RT} f^{eq}(\boldsymbol{x}, \boldsymbol{c}). \quad (2.53)$$

Therefore, the discretized BGK–Boltzmann equation can be rewritten as

$$\left[\frac{\partial}{\partial t} + c_\alpha \frac{\partial}{\partial x_\alpha}\right] f(\boldsymbol{x}, \boldsymbol{c}, t) = \frac{1}{\tau}\left(f^{eq}(\boldsymbol{x}, \boldsymbol{c}) - f(\boldsymbol{x}, \boldsymbol{c}, t)\right) + \frac{\boldsymbol{F}}{m}.\frac{\boldsymbol{c} - \boldsymbol{u}}{RT} f^{eq}(\boldsymbol{x}, \boldsymbol{c}). \quad (2.54)$$

For $m = 1$ and $RT = c_s^2$, the force term ends up as $\boldsymbol{F}.\frac{(\boldsymbol{c}-\boldsymbol{u})}{c_s^2} f_{eq}$ (He et al. 1998).

However, the force term should be added to the discretized LBM as $F_\alpha(1 - \frac{1}{2\tau})\frac{c_\alpha - u_\alpha}{\rho c_s^2} f_i^{eq}$. Also, the velocity should be calculated as

$$u_\alpha = \sum f_i c_{i\alpha} + \frac{F_\alpha}{2\rho}. \quad (2.55)$$

Guo et al. (2002) suggested the following formula for the source term:

$$\omega_i F_\alpha (1 - \frac{1}{2\tau})\left[\frac{c_{i\alpha} - u_\alpha}{c_s^2} + \frac{c_{i\alpha} c_{i\beta} u_\alpha}{c_s^4}\right]. \quad (2.56)$$

The velocity should be calculated as in Eq. (2.55).

Note that if $u = 0$ (for modeling the source term in heat diffusion problems, for example), the above-mentioned models become identical.

In the following chapters, more details will be given about the equilibrium distribution function and the force term with applications.

Chapter 3
Similarities and Scaling

In the previous chapters, the theory of LBM applications was discussed. However, before discussing the applications, it is important to create a link between the lattice domain (moment space) and the physical domain. Such a link can be established via similarities (nondimensionalizing the governing equations). It is very common in the thermal sciences (fluid mechanics and heat transfer) to use geometric and dynamic similarities for building a prototype and reducing data. In most engineering practices, the engineers build a model, either small or large depending on the problem, before building a full-sized the prototype. For instance, before building a full-sized airplane or automobile, tests are done on a model, computationally and experimentally.

Most engineering and scientific problems involve a few parameters (geometric properties, boundary and initial conditions, etc.). It is a very time-consuming job to do experiments or even analysis of a problem with many parameters. As simple example, consider an engineer interested in studying fluid flow in a pipe with the goal of documenting the pressure drop as a function of other governing parameters. From the physics of the problem, it is known that the pressure drop in a pipe depends on the mass flow rate (flow velocity), the density and viscosity of the working fluid, the diameter and length of the pipe, and perhaps other parameters such as the roughness of the pipe's surface. The momentum equation (Navier–Stokes) for a fully developed incompressible Newtonian flow in a pipe without body forces can be written as

$$0 = -\frac{1}{\rho}\frac{dp}{dz} + \frac{\nu}{r}\frac{d}{dr}\left(r\frac{du}{dr}\right), \tag{3.1}$$

where z, r, p, ρ, ν, and u are the axial coordinate, radial coordinate, pressure, density, viscosity, and axial velocity of the fluid, respectively. The boundary conditions are at $r = 0$, $\frac{du}{dr} = 0$ and at $r = r_o$, $u = 0$, where r_o is the outer radius of the pipe. Imagine someone doing experiments to correlate the pressure drop for different fluids. The kind of fluid needs to be varied in order to address the effect of density and viscosity. The pipe diameter and pipe length need to be varied in order to address the effect of the pipe's diameter and length. Furthermore, the mass flow rate needs to be varied for each of the experiments. However, nondimensionalizing the equation helps to reduce

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3_3

drastically the number of the parameters that must be varied in the experiment. Let $P = \frac{p}{\rho u_r^2}$, $R = \frac{r}{r_o}$, $Z = \frac{z}{r_o}$, and $U = \frac{u}{u_r}$. The governing equation will be

$$0 = -\frac{dP}{dZ} + \frac{1}{ReR}\frac{d}{dR}\left(R\frac{dU}{dR}\right) \tag{3.2}$$

with boundary conditions $R = 0$, $dU/dR = 0$ and $R = 1$, $U = 0$. In the above equation, $Re = \frac{u_r r_o}{\nu}$ is the Reynolds number, which is dimensionless. The reference velocity, u_r, stands for the inlet velocity or average velocity. It is clear that pressure drop is a function of only two parameters, Re and the dimensionless radius, R. Furthermore, if dP/dZ is a constant, we can simplify the above further by introducing a new variable $U^* = U/(dP/dZ)$, resulting in

$$0 = -1 + \frac{1}{ReR}\frac{d}{dR}\left(R\frac{dU^*}{dR}\right). \tag{3.3}$$

Notice that Re is the only parameter that must be varied. Also, the range of R is from 0 to 1, regardless of the size of the actual diameter of the pipe.

The bottom line: in LBM simulation we need to use a similar concept to transfer information from the physical space (problem) to the momentum space (lattice space) and vice versa. In the following sections, utilizing the similarity concept will be used to establish a relationship between the LBM domain, which is called the lattice domain, and the physical domain.

In general, for an isothermal fluid flow, the following scaling (nondimensional) parameters arise:

Time: $t^* = \frac{tU}{L}$, Length: $L^* = l/L$,
Reynolds number: $Re = \frac{UL}{\nu}$,

where the superscript refers to a nondimensional parameter, while U, L, t are the velocity, length, and time scales, respectively. To ensure the mapping between two domains, those parameters must be the same for the physical and LBM domains. In the following section, the concept is further explained.

3.1 Heat Diffusion

Heat (mass) diffusion in a two-dimensional stagnant medium with a rectangular geometry with constant properties is governed by

$$\rho c\frac{\partial T}{\partial t} = k\frac{\partial^2 T}{\partial x^2} + k\frac{\partial^2 T}{\partial y^2}, \tag{3.4}$$

where ρ, c, and k are density, specific heat, and thermal conductivity, respectively. Initially, the domain is at room temperature T_o. At time zero, the bottom and left-

hand walls are exposed to high temperature T_h, while the other boundaries are kept at room temperature. The domain has height $H = 10$ cm, and length $L = 20$ cm.

Let us select H for the length scale. Also, let us define a dimensionless temperature, $\theta = \frac{T-T_o}{T_h-T_o}$. Note that regardless of the values of T_h and T_o, the value of θ ranges between 0 and 1. The above equation can be rewritten as

$$\frac{\partial \theta}{\partial t} = \frac{\alpha}{H^2}\left(\frac{\partial^2 \theta}{\partial x^{*2}} + \frac{\partial^2 \theta}{\partial y^{*2}}\right), \tag{3.5}$$

where $\alpha = k/(\rho c)$, x^*, and y^* are thermal diffusivity, x/H, and y/H, respectively. By setting $\tau = \frac{t\alpha}{H^2}$, which is dimensionless time, the above equation can be rewritten as

$$\frac{\partial \theta}{\partial \tau} = \left(\frac{\partial^2 \theta}{\partial x^{*2}} + \frac{\partial^2 \theta}{\partial y^{*2}}\right) \tag{3.6}$$

with initial condition $\theta = 0$ and boundary conditions $x^* = 0$, $\theta = 1$, $x^* = L/H$, $\theta = 0$. For $y^* = 0$, we have $\theta = 1$, and for $y^* = 1$, we have $\theta = 1$.

In LBM, the so-called lattice units are used instead of physical units. For example, in the above example, H was selected as the length scale. The aspect ratio is L/H. In LBM, the number of lattices in the y-direction, say, is M (it can be any number), which depends on the accuracy of the solution and may depend on other parameters. However, to satisfy geometric similarity, the number of lattices in x-direction, N, must satisfy the following condition: $L/H = N/M$. If a heat source term S is associated with Eq. (3.4), which has a unit of W/m^3, then the nondimensionalized source term S^* will be $\frac{SH^2}{k(T_h-T_o)}$.

Also, τ, needs to be matched:

$$\frac{t\alpha}{H^2}|_{physical} = \frac{t\alpha}{M^2}|_l \tag{3.7}$$

Note: the subscript l stands for lattice (LBM domain).

The left-hand-side term is known from the problem definition. The right-hand term needs to be matched with the left-hand-side term. It is possible to select any value for $\alpha|_l$ (of course within the limit of stability; this issue will be discussed later on). By this process, the mapping in time between two domains is ensured.

3.2 Fluid Flow

For problems involving flow, the Navier–Stokes (NS) equation needs to be solved. As a first step, NS needs to be nondimensionalized. In general, besides geometrical similarity, dynamic similarity must be satisfied. For isothermal flow, the Reynolds number needs to be matched with the lattice Reynolds number. The Reynolds number is $Re = \frac{UH}{\nu}$, while the lattice Reynolds number is $Re_l = \frac{U_l M}{\nu_l}$, where M is the number

of lattices along the reference length. Note that U_l and ν_l need to be selected to ensure the stability of the LBM and to some extent have nothing to do with flow velocity and fluid viscosity, for example, flow in a two-dimensional duct with height $H = 10\,\text{cm}$ and length $L = 50\,\text{cm}$. The inlet velocity U is 0.5 m/s, and the fluid kinematic viscosity ν is 0.00002. Then one has $Re = 0.5 * 0.1/0.00002 = 2500$. Therefore, Re_l must be set to 2500. Let us say we have used 100 lattices in the H-direction; we need to choose both U_l and ν_l. We will learn later on that we have to use a small value for U_l to keep the flow incompressible; let us say we select 0.1 for U_l. Hence v_l can be calculated from $2500 = \frac{0.1*100}{\nu_l}$, i.e., $\nu_l = 0.004$. Also, we shall learn later that there is a constraint on selecting the value of ν_l, which is related to the stability of the scheme. However, if we find that the value of ν_l is low, then we have to increase the number of lattices in the H-direction, say from 100 to 400, or change the value of the lattice velocity.

For nonisothermal flows, the Rayleigh number Ra and the Prandtl number Pr come into the picture. The same procedure needs to be followed in selecting Ra_l and Pr_l.

The issue of translating information from the physical domain to the so-called lattice domain and vice versa will become clearer in the following chapters.

Now we have all the ingredients of LBM. In the following chapters, we shall solve many problems from different branches of engineering and the natural sciences.

Example Air flows through a duct with height $H = 0.2$ m, and length $L = 3$ m. The inlet velocity U_i is 0.03 m/s. The kinematic viscosity of air is $\nu = 1.5 \times 10^{-5}$. Explain how to solve the problem using convectional CFD and LBM.

Solution. In convectional CFD, the normal procedure is to nondimensionalize the problem. Therefore, the reference length and velocity need to be specified for the given problem. Let us take the height of the channel and inlet velocity as reference length and velocity scales, respectively. Hence, the nondimensional height of the problem will be 1, and the ratio with the length of the channel will be L/H (aspect ratio), which is $3/0.2 = 15$. The nondimensional inlet velocity will be 1. Examining the governing equation, which is the Navier–Stokes equation, yields a nondimensional parameter called the Reynolds number, $U_i H/\nu = 400$. If we are using the finite volume method, for example, then we solve the problem with $Re = 400$ and with a duct of height 1.0 and length 15. The inlet condition is 1.0 for velocity. For LBM, we need to match the geometric and flow parameters. In this problem, the aspect ratio and Reynolds number are given by

$$L/H = M/N \tag{3.8}$$

and

$$\frac{U_i H}{\nu} = 400 = \frac{U_l N}{\nu_l}, \tag{3.9}$$

where N and M are the numbers of lattices in the height and length directions, respectively. Also, U_l and ν_l are the lattice inlet velocity and lattice viscosity, respectively.

First, the inlet velocity should be small, to ensure an incompressible flow, i.e., the inlet velocity can be chosen about 0.1 or less. We need to match between physical Re, which is 400, and the LBM Re_{LB}, i.e.,

$$Re_l = 400 = \frac{U_l N}{\upsilon_l}\,, \tag{3.10}$$

where N is the number of lattices in the reference length direction, which is the height. For higher resolution, we need to select more lattices. The number of lattices along the length of the duct is controlled by the matching aspect ratio, i.e.,

$$\frac{L}{H} = 15 = \frac{N}{M}\,, \tag{3.11}$$

where M is the number of lattices along the duct.

Let us return to Re. As we mentioned, we must choose U_l to be small, less than 0.1, to ensure incompressibility. Now, either we choose the number N of lattices along the height of the duct or the viscosity ν_l. Later on, we shall see that the lattice viscosity is related to the relaxation time τ by

$$\nu_l = \frac{1}{3}(\tau - 0.5)\,, \tag{3.12}$$

where τ must be greater than 0.5.

3.3 Important Remarks

It is important to realize that the geometric similarity between the physical problem and the LB domain should be kept (such as aspect ratio). For example, if the physical domain is $L = 20$ and $H = 5$ units in the x- and y-directions, respectively, and we select H as the reference, then we have $L/H = 4$. Therefore, in the LB domain, the number of lattices in the x-direction should be four times the number of lattices in the y-direction.

Also, the controlling dimensionless parameters should be matched between the physical problem and the LB domain. For example, for fluid flow problems, the Reynolds number should be the same. The characteristic velocity for the LB domain will not necessarily match the characteristic velocity of the physical problem. In fact, the characteristic velocity for LB should be a small value to keep the Mach number in the incompressible domain, which can be selected randomly as long as the value of the Mach number is less than about 0.2.

Chapter 4
Boundary Conditions

Applying boundary conditions to different problems using LBM requires attention because it is different from the application of a boundary condition as in the classical CFD method. However, this chapter may be revisited when one is applying LBM to specific problems in the following chapters.

The distribution functions streaming out of the domain are known from the streaming process. However, the distribution functions toward the domain at the boundaries are unknown. Those functions need to be calculated.

As we know, there are a few types of boundary conditions: the values of the function are given (Dirichlet), the derivative of the function is known (Neumann), or one may have a combination of these (mixed). The following sections discuss those kinds of boundary conditions. However, in the conventional CFD method, the boundary conditions are explicitly known and can be directly applied. In LBM, the story is different, and the general practice is to use five distribution functions, Q5, for the energy (temperature) and species (concentration) conservation equations and to use nine, Q9, for the momentum equation (velocity). Therefore, the boundary conditions cannot be applied directly. There is a need for translating the given boundary conditions to the streaming-in distribution functions. In the following section, the boundary conditions for temperature and species concentration will be introduced first, followed by the boundary conditions for the momentum equation (velocity and pressure).

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3_4

4.1 Boundary Conditions for Energy and Species Conservation Equations

4.1.1 Dirichlet Boundary Condition, the Value of the Function Is Known

For instance, in heat transfer, the temperature is given at the boundary. For fluid flow problems, the velocity components are known at the boundary. As mentioned before, the distribution functions inward to the domain of interest need to be calculated. Let us take the left boundary, as an example, and assume that ϕ is a known scalar quantity at the boundary (such as temperature for heat transfer problems).

4.1.2 One-Dimensional Problems

For 1-D problems and for the left-hand boundary, the only unknown distribution function is f_1, for D1Q2 and D1Q3. For D1Q2, we have

$$\phi = f_1 + f_2 \,. \tag{4.1}$$

Therefore,

$$f_1 = \phi - f_2 \,. \tag{4.2}$$

For D1Q3: $f_1 = \phi - (f_0 + f_2)$.

Another approach:

At the boundary, the system is in equilibrium, and therefore, the nonequilibrium part of the distribution functions must be zero. Hence, for D1Q2,

$$f_1^{eq} - f_1 + f_2^{eq} - f_2 = 0, \tag{4.3}$$

where $f_1^{eq} = w_1\phi$ and $f_2^{eq} = w_2\phi$; w_1 and w_2 are the weighting factors. For D1Q2, $w_1 = w_2 = 0.5$. Then f_1 can be calculated as

$$f_1 = w_1\phi + w_2\phi - f_2 \,. \tag{4.4}$$

The above equation is the same as Eq. (4.2).

4.1.3 Two-Dimensional Problems

For D2Q4 or D2Q5, at the left boundary, the distribution function f_1 needs to be calculated. As mentioned before, the system is in equilibrium at the boundary, whence

$$f_1^{eq} - f_1 + f_2^{eq} - f_2 = 0\,. \tag{4.5}$$

Therefore,

$$f_1 = w_1\phi + w_2\phi - f_2\,. \tag{4.6}$$

For D2Q4, $w_1 = w_2 = 0.25$, whence $f_1 = 0.5\phi - f_2$.

For D2Q9, the distribution functions f_1, f_5, and f_8 are unknown. The equilibrium conditions should be imposed on the boundary, i.e.,

$$f_1 = f_1^{eq} + f_2^{eq} - f_2, \tag{4.7}$$

$$f_5 = f_5^{eq} + f_{7eq} - f_7, \tag{4.8}$$

and

$$f_8 = f_8^{eq} + f_6^{eq} - f_6. \tag{4.9}$$

4.1.4 Fluid Mechanics Problems

For D2Q9, and for a given velocity at the boundary, the density ρ is equal to the sum of the distribution functions,

$$\rho = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8\,. \tag{4.10}$$

The momentum equations in the x- and y-directions are

$$\rho u = f_1 + f_5 + f_8 - f_6 - f_3 - f_7 \tag{4.11}$$

and

$$\rho v = f_5 + f_2 + f_6 - f_7 - f_4 - f_8, \tag{4.12}$$

respectively. The x- and y-components of the velocity are u and v, respectively. We have four unknowns (three distribution functions, f_1, f_5, f_8, and an extra unknown ρ) and three equations. Therefore, another equation is needed to solve the problem. The equilibrium condition will be imposed normal to the boundary, i.e.,

$$f_1 - f_1^{eq} = f_3 - f_3^{eq}. \tag{4.13}$$

The equilibrium distribution functions f_1^{eq} and f_3^{eq} can be calculated as

$$f_1^{eq} = \frac{\rho}{9}\left[1 + 3u + \frac{9}{2}u^2 - \frac{3}{2}(u^2 + v^2)\right] \tag{4.14}$$

and

$$f_3^{eq} = \frac{\rho}{9}\left[1 - 3u + \frac{9}{2}u^2 - \frac{3}{2}(u^2 + v^2)\right]. \tag{4.15}$$

Hence

$$\boxed{f_1 = f_3 + \frac{2}{3}\rho u}. \tag{4.16}$$

The above four equations can be manipulated to yield

$$\boxed{\rho = \frac{1}{1-u}[f_0 + f_2 + f_4 + 2(f_3 + f_6 + f_7)]}, \tag{4.17}$$

$$\boxed{f_5 = f_7 - \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho u + \frac{1}{2}\rho v}, \tag{4.18}$$

and

$$\boxed{f_8 = f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho u - \frac{1}{2}\rho v}. \tag{4.19}$$

Note: If the velocity components at the wall are zero, which is called a **nonslip condition**, a bounce-back is usually used, which assumes that the particles hitting the wall reflect back, i.e., $f_1 = f_3$, $f_5 = f_7$, and $f8 = f_6$. Note that if $u = v = 0$, then the above equations resemble the bounce-back condition because $f_2 = f_4$ as well.

In some fluid flow problems, the pressure is known at the boundary, since the pressure is related to the density in LBM, $p = \rho/c^2$, where $c = \sqrt{3}$ for D2Q9. In such a case, we solve for u:

$$\boxed{u = 1 - \frac{1}{\rho}[f_0 + f_2 + f_4 + 2(f_3 + f_6 + f_7)]}. \tag{4.20}$$

Since u is known, we use Eqs. (4.16), (4.18), and (4.19) to calculate the unknown distribution functions. Detailed implementations will be given later on, when we discuss fluid flow problems.

4.2 Von Neumann: The Derivative of the Function Is Known

In some problems, the value of the function is unknown yet the derivative of the function is known at the boundary. For instance, heat flux at the boundary is known. Practical applications include electric power imposed at the boundary, solar intensity, convective boundary conditions, and a laser beam hitting an opaque surface. The generic formula for such a boundary condition is

$$\frac{\partial \phi}{\partial x} = a\phi + b\,, \tag{4.21}$$

where a and b are known constants and x is a coordinate normal to the boundary. If $a = 0$ and $b = 0$, then the flux at the boundary is zero (adiabatic condition). If $a = 0$, then the flux is constant at the boundary.

The flux in any direction is the difference between the distribution functions along that direction. For example, for D1Q2, the flux in the positive x-direction is $f_1 - f_2$. Hence Eq. (4.21) yields

$$f_1 - f_2 = a(f_1 + f_2) + b\,. \tag{4.22}$$

Solving for f_1 for the left boundary condition yields

$$f_1 = \frac{f_2(1+a) + b}{1-a}, \quad a \neq 1\,. \tag{4.23}$$

Also, it is possible to use a second-order finite difference method,

$$\frac{\phi_1 - \phi_3}{2} = a\phi_1 + b\,. \tag{4.24}$$

Solving for ϕ_1 yields

$$\phi_1 = \frac{\phi_3 + 2b}{1-2a}\,. \tag{4.25}$$

Since ϕ_3 is known, ϕ_1 (at the boundary) can be calculated from the above equation. Hence the missing distribution function related to ϕ_1 can be formulated.

Chapter 5
The Diffusion Equation

5.1 Diffusion Equation

The one-dimensional diffusion equation can be written as

$$\frac{\partial \phi}{\partial t} = \alpha \frac{\partial^2 \phi}{\partial x^2} . \tag{5.1}$$

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

$$\frac{1}{\tau} \approx \alpha \frac{1}{\ell^2} , \tag{5.2}$$

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

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

$$\ell \approx \sqrt{\alpha \tau} . \tag{5.3}$$

Equation 5.1 has a second spatial derivative; therefore, the diffusion takes place in both directions, and it requires two boundary conditions. Also, Eq. (5.1) has a first time derivative, and the diffusion is unidirectional 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 the equation.

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3_5

As an example, if a drop of ink is added to a glass filled with water, the ink diffuses in all directions and the concentration of the ink at any spot in the water decreases as the time proceeds until it reaches the equilibrium condition. The concentration of the ink at any location depends on the previous time and on the concentration at the boundary of that spot.

5.1.1 Examples

The one-dimensional heat diffusion equation can be written as

$$\rho C \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right), \tag{5.4}$$

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 the thermal conductivity (k), density (ρ), and specific heat (C). For a constant k, the above equation can be written as

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}, \tag{5.5}$$

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.

Assume that a plate has thickness L, initially at temperature T_o, and at time zero, the left boundary ($x = 0$) is set to a high temperature T_h. The above equation can be nondimensionalized by defining a new variable, $\phi = \frac{T - T_o}{T_h - T_o}$, where T_o and T_h are the initial temperature (or temperature at the right-hand side of the boundary of the domain, $x = L$) and temperature at the left-hand side of the boundary of the domain, $x = 0$ (hot temperature), respectively. The dimensionless coordinate system is $x^* = x/L$, and the dimensionless time is $\tau = t\alpha/L^2$. Therefore, the dimensionless heat diffusion equation is

$$\frac{\partial \phi}{\partial \tau} = \frac{\partial^2 \phi}{\partial x^{*2}}. \tag{5.6}$$

The initial condition $\tau = 0$ is $\phi = 0$, and the boundary conditions are at $x^* = 0$, $\phi = 1$ and at $x^* = 1$, $\phi = 0$.

Fluid Mechanics: The one-dimensional momentum diffusion can be written as

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial y^2}, \tag{5.7}$$

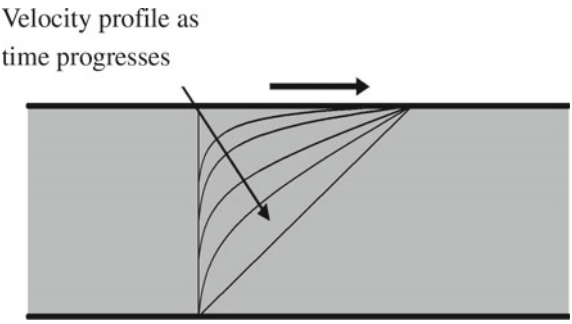


Fig. 5.1 Momentum diffusion in a fluid confined between two parallel plates due to the motion of the upper lid

where u is velocity, and ν is the kinematic viscosity of the fluid. For a fluid between parallel plates (Fig. 5.1), if the upper plate is set in motion, the momentum diffuses according to the above equation. The rate of momentum diffusion depends on the viscosity of the fluid. The time needed for the bottom plate to sense the motion of the upper plate depends inversely on the viscosity of the fluid (high viscosity means less sensation time) and the distance between the plates,

$$\tau \approx \ell^2/\nu\,, \tag{5.8}$$

and so on.

5.2 Approximation of Finite Differences

In the following section, the numerical solution using finite difference approximation of a 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 finite differences are twofold: first, to show the differences and similarities between finite difference approximation and the lattice Boltzmann method, and also to compare the results of the two methods.

As a first step, the domain needs to be divided (discretized) into equal segments (it is not necessary that the length of each segment be equal; however, for simplicity, we use equal segments), as shown in Fig. 5.2. The nodes are labeled 0, 1, 2, 3, . . . , i-2, $i-1, i, i+1, i+2, \ldots, n$.

Fig. 5.2 One-dimensional node distribution

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

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} . \tag{5.9}$$

The above equation can be rearranged as

$$T_i^{n+1} = T_i^n + \frac{\alpha \Delta t}{\Delta x^2} \left(T_{i+1}^n - 2T_i^n + T_{i+1}^n \right) . \tag{5.10}$$

Equation 5.10 can be reformulated as

$$T_i^{n+1} = T_i^n \left(1 - \frac{2\alpha \Delta t}{\Delta x^2} \right) + \frac{2\alpha \Delta t}{\Delta x^2} \left(\frac{T_{i+1}^n + T_{i+1}^n}{2} \right) . \tag{5.11}$$

Let

$$\Omega = \frac{2\alpha \Delta t}{\Delta x^2}; \tag{5.12}$$

then Eq. (5.11) can be written as

$$T_i^{n+1} = T_i^n \left(1 - \Omega \right) + \Omega \left(0.5 T_{i+1}^n + 0.5 T_{i-1}^n \right) . \tag{5.13}$$

For the 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

$$\Delta t \leq \frac{\Delta x^2}{2\alpha} . \tag{5.14}$$

The finite difference approximation for the diffusion equation is manipulated into the form of Eq. (5.13) to compare the finite difference formulation with LBM.

Let us examine Eq. (5.13), in which the last term $(0.5[T_{i+1}^n + T_{i-1}^n])$ is the average of the temperatures around T_i. In other words, the average term represents the equilibrium value of T_i. Since T_i is midway between T_{i+1} and T_{i-1}, it follows that at the equilibrium state, the value of T_i should be the average of its neighboring values. It is appropriate to write the term $0.5[T_{i+1}^n + T_{i-1}^n]$ as T_i^{eq}. Then Eq. (5.13) can be rewritten as

$$T_i^{n+1} = T_i^n \left(1 - \Omega \right) + \Omega T_i^{eq} . \tag{5.15}$$

In the next section, we can see that Eq. (5.15) resembles the LB equation (5.21).

5.3 The Lattice Boltzmann Method

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

$$\frac{\partial f_k(x,t)}{\partial t} + c_k . \frac{\partial f_k(x,t)}{\partial x} = \Omega_k \,, \tag{5.16}$$

$k = 1, 2$ (for the 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

$$c_k = \frac{\Delta x}{\Delta t} \,. \tag{5.17}$$

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

The BGK approximation for the collision operator can be approximated as

$$\Omega_k = -\frac{1}{\tau}\left[f_k(x,t) - f_k^{eq}(x,t)\right] . \tag{5.18}$$

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 later on.

The kinetic LB equation (5.16) with the BGK approximation can be discretized as

$$\frac{f_k(x,t+\Delta t) - f_k(x,t)}{\Delta t} + c_k . \frac{f_k(x+\Delta x, t+\Delta t) - f_k(x,t+\Delta t)}{\Delta x} =$$

$$-\frac{1}{\tau}\left[f_k(x,t) - f_k^{eq}(x,t)\right] . \tag{5.19}$$

Note that $\Delta x = c_k \Delta t$, and substituting in Eq. (5.19), it can be simplified to

$$f_k(x+\Delta x, t+\Delta t) - f_k(x,t) = -\frac{\Delta t}{\tau}\left[f_k(x,t) - f_k^{eq}(x,t)\right] . \tag{5.20}$$

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

$$f_k(x+\Delta x, t+\Delta t) = f_k(x,t)\,[1-\Omega] + \Omega\, f_k^{eq}(x,t) \tag{5.21}$$

where

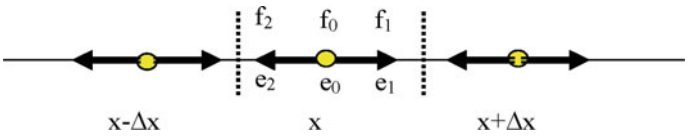


Fig. 5.3 Schematic diagram for a 1-D lattice arrangement

$$\Omega = \frac{\Delta t}{\tau} \tag{5.22}$$

is called a relaxation time.

It is interesting to note that Eq. (5.21) above is similar to the finite difference equation (5.15).

Equation 5.21 represents the number of equations for different k values ($k = 1$ and 2) in each direction. Then what is the difference between the FD and LBM schemes? First, updating the value of f_i in Eq. (5.21) does not depend on the neighboring values (i.e., on f_{i+1} and f_{i-1}, while) it does in Eq. (5.15). Hence the LBM process is solely local. The effect of neighbors is injected by the streaming processes. The stability of FD and LBM is conditional, because both of the schemes are explicit in time. However, the stability of LBM is much better than the stability of FD.

Figure 5.3 shows a schematic diagram for three nodes with necessary linkages and central and neighboring nodes, where $c_1 = c, c_2 = -c$. The dependent variable, ϕ, in Eq. (5.1) can be related to the distribution function f_i as

$$\phi(x,t) = \sum_{k=1}^{2} f_k(x,t)\,. \tag{5.23}$$

The equilibrium distribution function f_k^{eq} (see Eq. 5.21) can be chosen as

$$f_k^{eq} = w_k \phi(x,t)\,, \tag{5.24}$$

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

$$\sum_{k=1}^{2} w_k = 1\,. \tag{5.25}$$

The distribution function $f_k^{eq}(x,t)$ defined in Eq. (5.24) can be summed along all directions, and this yields

$$\sum_{i=0}^{2} f_k^{eq}(x,t) = \sum_{i=0}^{2} w_k \phi(x,t) = \phi(x,t) \tag{5.26}$$

The relation between α and ω can be deduced from a multiscale expansion using a Chapman–Enskog expansion, which yields

$$\alpha = \frac{\Delta x^2 c_s^2}{\Delta t}\left(\frac{1}{\Omega} - \frac{1}{2}\right). \tag{5.27}$$

The weight factor for the one-dimensional problem, $D1Q2$, is $w_k = 1/2$ for $k = 1$ and 2.

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

5.4 Equilibrium Distribution Function

Recall the equilibrium distribution function (Eq. 2.50),

$$f_k^{eq} = \rho w_k \left[1 + \frac{2\vec{c}.\vec{u} - \vec{u}.\vec{u}}{2c_s^2} + \frac{(\vec{c}.\vec{u})^2}{2c_s^4}\right]. \tag{5.28}$$

For the diffusion problem, the macroscopic velocity is zero; then

$$f_k^{eq} = \rho w_k \,. \tag{5.29}$$

Hence

$$f_k^{eq} = \omega_k \phi \,, \tag{5.30}$$

where ϕ is the dependent variable, such as temperature. In the following sections, the relationship between LBM and macroscale systems will be discussed. In other words, the relationship between the diffusion coefficient and LBM relaxation time will be established by multiscale analysis.

5.5 Chapman–Enskog Expansion

The one-dimensional diffusion equation in Cartesian coordinates can be written as

$$\frac{\partial T(x,t)}{\partial t} = \Gamma \frac{\partial^2 T(x,t)}{\partial x^2} \,, \tag{5.31}$$

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 the 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 (Knudsen number). Introducing the scale to the diffusion equation yields

$$\frac{\partial T(x,t)}{\partial t} = \Gamma \frac{\epsilon^2 \partial^2 T(x,t)}{\partial x^2} . \tag{5.32}$$

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

$$T(x,t) = \sum_{i=1}^{i=2} f_i(x,t) = f_1(x,t) + f_2(x,t) . \tag{5.33}$$

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 the equilibrium distribution function is related to T (as mentioned before) as

$$f_i^{eq} = w_i T(x,t) . \tag{5.34}$$

By summing both sides of the above equation, we obtain

$$\sum_i f_i^{eq} = T = w_1 T + w_2 T ; \tag{5.35}$$

henc, $w_1 + w_2 = 1$, i.e., $w_1 = w_2 = \frac{1}{2}$. Then

$$f_1^{eq} = w_1 T(x,t) \; and \; f_2^{eq} = w_2 T(x,t) , \tag{5.36}$$

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

$$f_i(x,t) = f_i^o + \epsilon f_i^1 + \epsilon^2 f_i^2 + \cdots , \tag{5.37}$$

where f_i^o is the distribution function at the equilibrium conditions, equal to f_i^{eq}. By summing the above equation, we get

$$\sum_{i=1}^{i=2} f_i(x,t) = \sum_{i=1}^{i=2} [f_i^o + \epsilon f_i^1 + \epsilon^2 f_i^2 + \cdots] . \tag{5.38}$$

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

$$\sum_{i=1}^{i=2} f_i^1 = 0\,. \tag{5.39}$$

$$\sum_{i=1}^{i=2} f_i^2 = 0\,. \tag{5.40}$$

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

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) + \frac{\partial f_i}{\partial t}\Delta t + \frac{\partial f_i}{\partial x}c_i\Delta t + 1/2\Delta t^2\left(\frac{\partial^2 f_i}{\partial t^2} + 2\frac{\partial^2 f_i}{\partial t\partial x}c_i + \frac{\partial^2 f_i}{\partial x^2}c_i c_i\right) + O(\Delta t)^3\,. \tag{5.41}$$

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

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) + \epsilon^2\frac{\partial f_i}{\partial t}\Delta t + \epsilon\frac{\partial f_i}{\partial x}c_i\Delta t + 1/2\Delta t^2(\frac{\epsilon^4\partial^2 f_i}{\partial t^2} + 2\frac{\epsilon^3\partial^2 f_i}{\partial t\partial x}c_i + \frac{\epsilon^2\partial^2 f_i}{\partial x^2}c_i c_i) + O(\Delta t)^3\,. \tag{5.42}$$

The lattice Boltzmann equation can be written as

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) + \frac{\Delta t}{\tau}[f_i^o(x,t) - f_i(x,t)]\,. \tag{5.43}$$

Substituting Eq. (5.42) into the above equation and retaining terms up to order ϵ^2 yields

$$\frac{1}{\tau}[f_i^o(x,t) - f_i(x,t)] = \epsilon^2\frac{\partial f_i}{\partial t} + \epsilon\frac{\partial f_i}{\partial x}c_i + 1/2\epsilon^2\Delta t(\frac{\partial^2 f_i}{\partial x^2}c_i c_i) + O(\Delta t)^2 + O(\epsilon^3)\,, \tag{5.44}$$

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

$$-\frac{1}{\tau}(\epsilon f_i^1 + \epsilon^2 f_i^2 + ..) = \epsilon^2\frac{\partial f_i^o}{\partial t} + \epsilon\frac{\partial f_i^o}{\partial x}c_i + \epsilon^2\frac{\partial f_i^1}{\partial x}c_i + \frac{\Delta t}{2}\epsilon^2\frac{\partial f_i^o}{\partial x}c_i c_i + O(\epsilon^3)+)(\Delta t^2)\,. \tag{5.45}$$

We equate the terms on both sides of the above equation of the same order of ϵ:

Terms order of ϵ:

$$-\frac{1}{\tau} f_i^1 = \frac{\partial f_i^o}{\partial x} c_i \,. \tag{5.46}$$

Later on, we may need the derivative of f_i^1 with respect of x; hence by taking the derivative of the above equation, we get

$$-\frac{1}{\tau} \frac{\partial f_i^1}{\partial x} = \frac{\partial^2 f_i^o}{\partial x^2} c_i \,. \tag{5.47}$$

Terms order of ϵ^2:

$$-\frac{1}{\tau} f_i^2 = \frac{\partial f_i^o}{\partial t} + \frac{\partial f_i^1}{\partial x} c_i + \frac{\Delta t}{2} \frac{\partial^2 f_i^o}{\partial x^2} c_i c_i \,. \tag{5.48}$$

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

$$-\frac{1}{\tau} f_i^2 = \frac{\partial f_i^o}{\partial t} - \tau \frac{\partial^2 f_i^o}{\partial x^2} c_i c_i + \frac{\Delta t}{2} \frac{\partial^2 f_i^o}{\partial x^2} c_i c_i \,. \tag{5.49}$$

To recover the continuum diffusion equation (5.1), the above equation needs to be summed over all states, $i = 1, 2$:

$$\sum_i [-\frac{1}{\tau} f_i^2] = \sum_i \left[\frac{\partial f_i^o}{\partial t} - \tau \frac{\partial^2 f_i^o}{\partial x^2} c_i c_i + \frac{\Delta t}{2} \frac{\partial^2 f_i^o}{\partial x^2} c_i c_i \right]. \tag{5.50}$$

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

$$\sum_i \frac{\partial f_i}{\partial t} = \frac{\partial \sum_i f_i}{\partial t} = \frac{\partial \Theta}{\partial t} \,. \tag{5.51}$$

The term of

$$\sum_i [\frac{\partial^2 f_i^o}{\partial x^2} c_i c_i] = \frac{\partial^2}{\partial x^2} (\sum_i [f_i^o c_i c_i] = \frac{\partial^2 \Theta}{\partial x^2} \,. \tag{5.52}$$

Hence Eq. (5.50) can be simplified as

$$0 = \frac{\partial \Theta}{\partial t} - (\tau - \frac{\Delta t}{2}) \frac{\partial^2 \Theta}{\partial x^2} \,. \tag{5.53}$$

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

$$\Gamma = \tau - \frac{\Delta t}{2} \,. \tag{5.54}$$

Hence, a relationship is established between the macro- and mesoscales.

5.5.1 Normalizing and Scaling

As discussed in Chapter 3, it is more appropriate to solve specific problems in nondimensionalized (normalized) forms than in dimensional form, for a few reasons. The controlling parameters can be grouped into a few nondimensional parameters; hence, the results can be presented in a more general form. Also, the scaling from continuum mechanics (macroscale) to the lattice Boltzmann method (mesoscale) becomes clear. For example, Eq. (5.1) is the dimensional form for the diffusion equation; α has units of m^2/s, x has units of m, and ϕ has units of $^\circ C$, or K, or other units depending on what ϕ represents. However, let us assume that Eq. (5.1) represents the heat diffusion equation. Then ϕ represents temperature, and it has units of degrees Celsius ($^\circ C$). We can scale the temperature by $\theta = \frac{\phi - \phi_c}{\phi_h - \phi_c}$, where ϕ_c and ϕ_h are colder and hotter temperatures bounding the system. Then 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

$$\frac{\partial \theta}{\partial t^*} = \alpha^* \frac{\partial^2 \theta}{\partial x^{*2}} , \tag{5.55}$$

where t^* is nondimensional time ($t^* = t/T$), and T is total time or final time; α^* is a nondimensional parameter ($\alpha^* = \alpha T/L^2$). The values of θ and x range from 0 to 1.0, regardless of the range of the dimensional variables. The equation can be solved easily with LBM. Using a conventional numerical technique such as the finite difference method, for example, we discretize the space domain into nodes, and then the distance between two nodes is Δx, and the time domain is also discretized into Δt. If the domain is discretized into 100 nodes (0, 1, 2, . . . , 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 a stable solution if an explicit method is used. In LBM, the domain also needs to be discretized; however, the most popular method is to use unity for Δx and Δt, i.e., $\Delta x = 1$ and $\Delta t = 1$. The question remains as to the number of lattices and time steps for a given time and dimension of the domain. We can select any total number of lattices, for example $N = 500$, and then the nondimensional distance x at any site i is $i/500$. For instance, for site number 50, the value of x is $50/500 = 0.1$. The time scale needs to be worked out. The only parameter that needs to be matched between the physical (macro) and momentum (meso) domains is the parameter $\alpha^* = \frac{\alpha T}{L^2}$. Hence for LBM, we need to match the above-mentioned dimensionless time, i.e.,

$$\frac{\alpha \Delta t M}{(\Delta x N)^2}|_{macro} = \frac{\alpha \Delta t M_{lb}}{(\Delta x N_{lb})^2}|_{lb} , \tag{5.56}$$

where M and N are the numbers of time and space steps. Since $\Delta x = 1$ and $\Delta t = 1$ in LBM, the above equation becomes

$$\frac{\alpha \Delta t M}{(\Delta x N)^2}|_{macro} = \frac{\alpha M_{lb}}{N_{lb}^2}|_{lb} , \tag{5.57}$$

where M and N are the numbers of nodes (or lattices) and time steps, respectively. The left-hand side of the above equation is known. For the right-hand terms, we have the freedom to select two of the parameters. We can select for α_{lb} any value within the range such that τ will be within the stability limits. However, if we are dealing with variable α, then α_{lb} needs to be selected to match the same form of α_{macro}. The value of N needs to be selected to ensure a good resolution and accurate data.

For example, for the macro problem, $\alpha = 0.25, \Delta t = 0.05, \Delta x = 0.1$. The system length is one, and then $N = 1/0.1 = 10$. If the total time is 2.0, then $M = 2.0/0.05 = 40$. To simulate the problem, we need to select α, M_{lb}, and N_{lb} for a given lattice arrangement. For the given values,

$$\frac{\alpha \Delta t M}{(\Delta x N)^2} = \frac{0.25 * 0.05 * 40}{(0.1 * 10)^2} = 0.5\,. \tag{5.58}$$

Hence

$$0.5 = \frac{\alpha M_{lb}}{N_{lb}^2}\,. \tag{5.59}$$

For 100 lattices ($N_{lb} = 100$), we hence have $M_{lb} = 20000$ for $\alpha_{lb} = 0.25$.

In the following sections, a few practical examples are introduced to present applications of the diffusion equation with codes.

5.5.2 *Heat Diffusion in an Infinite Slab Subjected to a Constant Temperature*

A slab is 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 predicted results of both methods, lattice Boltzmann (LBM) and finite difference (FDM), for $\alpha = 0.25$.

Solution:
Let us divide the domain of integration into $\Delta x = 1.0$ and $\Delta t = 1.0$. (Fig. 5.4)

Equation 5.27 can be used to calculate Ω:
$\Omega = \frac{1}{\alpha/c_s^2 + 0.5}$. Then $\Omega = 0.8$ for D1Q3, where $c_s^2 = 1/3$.

The temperature is a function of x and t. Equation 5.33 yields

$$T(x,t) = \sum_{k=1}^{2} f_k(x,t) \tag{5.60}$$

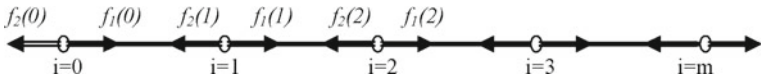


Fig. 5.4 Lattices for the 1-D diffusion problem

and from Eq. (5.34), we have $f_k^{eq}(x,t) = w_k T(x,t)$, and hence $f_1^{eq}(x,t) = 0.5T(x,t)$ and $f_2^{eq}(x,t) = 0.5T(x,t)$.

The lattice Boltzmann method consists of two steps: collisions and streaming.

The collisions step is

$$f_k(x, t + \Delta t) = f_k(x,t)\,[1 - \Omega] + \Omega\, f_k^{eq}(x,t)\,, \tag{5.61}$$

where $k = 1$ and 2. The streaming step is

$$f_k(x + \Delta x, t + \Delta t) = f_k(x, t + \Delta t)\,. \tag{5.62}$$

In order not to have to save the previous values of $f_k(x,t)$ and to avoid overwriting updated values, the stream (updating) for $f_1(x,t)$ should be from $i = m$ to $i = 0$, while the stream (updating) for $f_2(x,t)$ should be from $i = 0$ to $i = m$, as shown in Fig. 5.5.

As illustrated in Fig. 5.5, 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, overwriting updated values by old values can be avoided.

The codes for D1Q2, D1Q3a, and D1Q3b LBM, and the finite difference method are given in the appendix.

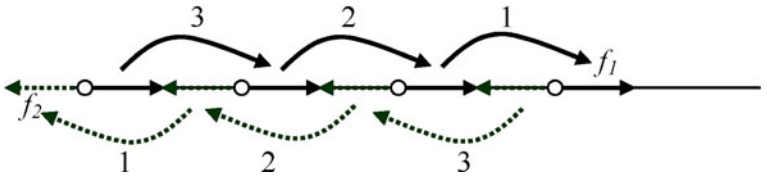


Fig. 5.5 Algorithm for streaming processes

5.5.3 Heat Flux Calculation

The heat flux in a macroscale system can be expressed as

$$q'' = -k\frac{\partial T}{\partial n}\,, \tag{5.63}$$

where k is the thermal conductivity of the medium, and n is the normal direction to the boundary or the surface. In LBM, the nondimensionalized heat flux is expressed at any latex location by applying the first moment, i.e., for D1Q2 and D1Q3,

$$\frac{q''}{k} = \frac{1}{c_s^2\tau}(f_1 - f_2)\,. \tag{5.64}$$

Or using finite difference approach,

$$\frac{q''}{k} = -(T(1) - T(0))\,. \tag{5.65}$$

Note that $\Delta x = 1$.

5.5.4 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 needs to be known, while f_1 can be obtained from the streaming process.

The following sections discuss the treatment of different boundary conditions.

A: Constant-temperature boundary condition, Dirichlet boundary condition

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

$$f_1^{eq}(0,t) - f_1(0,t) + f_2^{eq}(0,t) - f_2(0,t) = 0 \tag{5.66}$$

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

Therefore, at $x = 0.$, $f_1(0) + f_2(0) = T_w$, and from the streaming processes, $f_2(0) = f_2(1)$, and then $f_1(0)$ can be found as $f_1(0) = T_w - f_2(0)$ for D1Q2 and $f_1 = T_w - f_2 - f_0$ for D1Q3.

B: 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 $f_1(m) = f_1(m-1)$ and $f_2(m) =$

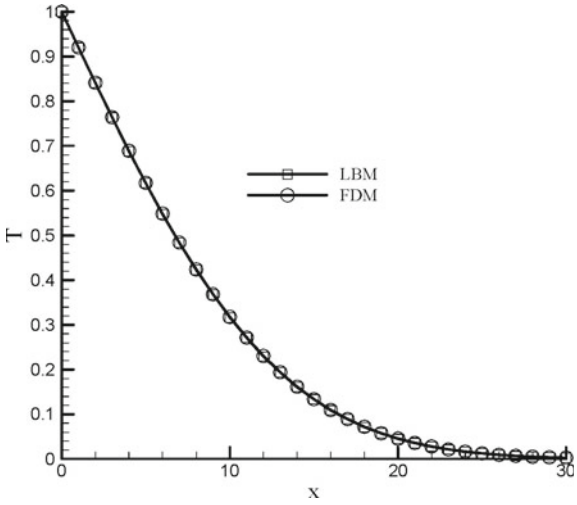


Fig. 5.6 Comparison of predicted results obtained by LBM and FDM for the slab of constant temperature problem

$f_2(m-1)$, where m denotes the last lattice node. Or simply use $f_2(m) = f_1(m)$, which ensures zero flux at the boundary (Fig. 5.6).

Variable diffusivity: The diffusivity of some materials, such as composite materials, may vary along the x-axis. Or the diffusivity is a function of temperature. This kind of problem can be solved similarly to the method of solution discussed before. See the appendix for a code for solving a problem with variable diffusivity.

5.5.5 Constant Heat Flux Example

This is the same example as before, except that the left-hand boundary is exposed to a constant heat flux of $100 W/m^2$ instead of a constant temperature. The thermal conductivity of the slab is $20 W/mK$.

Solution:

The procedure is the same as for the previous example, except that we need to determine $f_1(0)$ at $x = 0$. Fourier's law states that the heat flux is proportional to the temperature gradient:

$$q'' = -k\frac{\partial T}{\partial x}, \tag{5.67}$$

where k is the thermal conductivity of the material.

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

$$q'' = -k\frac{T(1) - T(0)}{dx}. \tag{5.68}$$

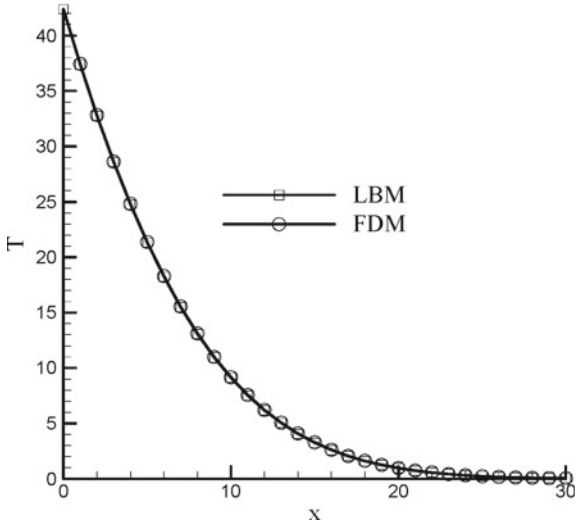


Fig. 5.7 A comparison of results predicted by LBM and FDM for a slab subjected to a constant heat flux at the boundary

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

$$f_1(0) = f_1(1) + f_2(1) - f_2(0) + \frac{q'' dx}{k}. \tag{5.69}$$

Hence $f_1(0)$ can be calculated from the above equation, where $f_2(0)$ can be obtained from the streaming step as before.

Or using momentum balance, as discussed before, at $x = 0$, we have

$$f_1(0) = f_2(0) + \frac{q'' \tau c_s^2}{k}. \tag{5.70}$$

The computer code is given in the appendix. It is the same computer code as discussed before (see the appendix for codes), except the boundary conditions should be changed. Figure 5.7 compares the predicted results of LBM and FDM.

5.6 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, the differential equation for such a problem can be written as

$$\rho C \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) + q_g \,, \tag{5.71}$$

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

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} + S \,, \tag{5.72}$$

where $S = \frac{q_g}{\rho C}$.

In lattice Boltzmann, correctly adding the source term depends on the lattice arrangement. For D1Q2, the source term can be added to the LB equation as

$$f_k(x, t + \Delta t) = f_k(x, t)\,[1 - \Omega] + \Omega\, f_k^{eq}(x, t) + \Delta t\, w_k S c_s^2 \,, \tag{5.73}$$

where S is the force or source term ($S = \frac{q_g}{\rho C}$), as mentioned before. Notice that units of S is units of temperature (^oC) per unit of time. The value of w_k is 0.5 for both w_1 and w_2.

For D1Q3, the source term is added as before, for example,

$$f_k(x, t + \Delta t) = f_k(x, t)\,[1 - \Omega] + \Omega\, f_k^{eq}(x, t) + \Delta t\, w S \,, \tag{5.74}$$

where $\tau = 1/\Omega$. We found that D1Q3b gives better results than D1Q3a. Matlab codes are given in the appendix.

5.7 Axisymmetric Diffusion

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

$$\frac{\partial T}{\partial t} = \frac{\alpha}{r^k} \frac{\partial}{\partial r}\left(r^k \frac{\partial T}{\partial r}\right), \tag{5.75}$$

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

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial r^2} + \frac{k\alpha}{r} \frac{\partial T}{\partial r} \,. \tag{5.76}$$

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

An analytic solution for the heat conduction (steady state) in a cylindrical shell without heat generation can be written as

$$\theta = \frac{ln(r/r_i)}{ln(r_o/r_i)} \,, \tag{5.77}$$

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

The BGK lattice Boltzmann equation with source term can be written as Eq. (5.71) with $\omega = \frac{\Delta t}{\alpha+0.5}$. The source term S can be treated as either

$$\frac{\alpha n}{r}\frac{T_{j+1} - T_{j-1}}{2\Delta r_j} \tag{5.78}$$

or

$$\frac{\alpha n}{r}\frac{f_{2,j+1} - f_{1,j-1}}{\tau} \tag{5.79}$$

where $\tau = \alpha + 0.5$.

For more details about solving axisymmetric (cylindrical or spherical with or without heat generation) diffusion problems with LBM, we refer to a 2009 paper by Mohamad, (*Progress in Computational Fluid Dynamics, an International Journal*, Volume 9, Number 8/2009, pp. 490–494).

5.8 Two-Dimensional Diffusion Equation

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

For illustrative purposes, three types of lattice arrangements will be discussed, namely, D2Q4, D2Q5, and D2Q9. The arrangement D2Q5 is a simple extension of the 1-D problem, while D2Q9 is more involved. The main objective of introducing D2Q9 is for future analysis of fluid flow problems using Navier–Stokes equations. Hence, it aims to build a background for more complicated problems, mainly for fluid flow problems. *Note that D2Q5 is sufficient and sufficiently stable to solve scalar variables, such as temperature (energy equation) and species concentration (species conservation equation).*

5.8.1 D2Q4

The common terminology used in LBM refers to the dimension of the problem and the speed model. Thus we have DnQm, where n represents the dimension of the problem (2 for 2-D and 3 for 3-D), and m refers to the speed model, as mentioned before.

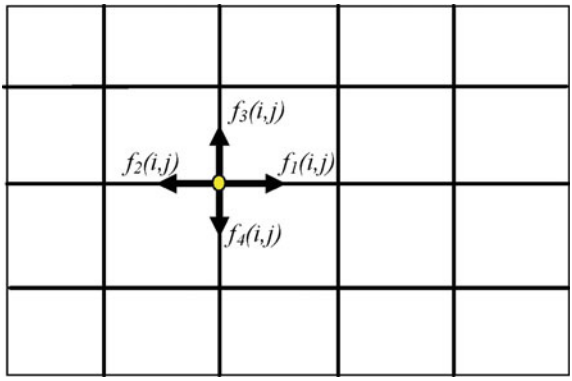


Fig. 5.8 Lattices for the 2-D diffusion problem

The diffusion phenomena have no directional preference, and only the gradient determines the diffusion process, unlike the advection process, where the information follows the direction of the flow. Full isotropy can be ensured by a 90^o rotation of the coordinates; hence it is sufficient to use only four streaming directions; see Fig. 5.8.

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. The collisions step without a forcing function is

$$f_k(x, y, t + \Delta t) = f_k(x, y, t)\,[1 - \Omega] + \Omega\, f_k^{eq}(x, y, t)\,, \tag{5.80}$$

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

The streaming step is

$$f_k(x + \Delta x, y + \Delta y, t + \Delta t) = f_k(x, y, t + \Delta t) \tag{5.81}$$

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

In the 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 $f_4(i, j - 1)$. We have $w(1) = w(2) = w(3) = w(4) = 1/4$.

Indeed, there is nothing new here compared with the 1-D problem. In programming, one needs to take care not to overwrite values of $f\prime s$; other than that, every step is similar to the 1-D algorithm. We have $w_i = 0.25$ for $i = 1, 2, 3,$ and 4. Also,

$$\alpha = \frac{\Delta x^2 c_s^2}{\Delta t}\left(\frac{1}{\Omega} - \frac{1}{2}\right), \tag{5.82}$$

where $c_s^2 = 1/2$.

5.8.2 D2Q5

The situation here is 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

$$\alpha = \frac{\Delta x^2 c_s^2}{\Delta t}\left(\frac{1}{\Omega} - \frac{1}{2}\right), \tag{5.83}$$

where $c_s^2 = 1/3$.

Other than that, every step and algorithm is the same as for D2Q4, though of course there is no streaming for the particle at the center of the lattice (f_0). The codes for D2Q5 and finite difference are given in the appendix. It should be mentioned that the main controlling parameter is the nondimensional diffusion coefficient α^*, as discussed before. Hence, the scaling between the macro system and LBM should be as follows:

$$\frac{\alpha T_t}{L^2}|_{macro} = \frac{\alpha T_t c_s^2}{N^2}|_{LBM}, \tag{5.84}$$

where T_t is the total time of the simulation, which is equal to the time increment ($dt = 1$) multiplied by the number of time steps; L is the length scale for the macro system, and N is the number of the lattices in the L direction.

5.9 Boundary Conditions

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

5.9.1 The Value of the Function Is Given at the Boundary

For example, the temperature at the left and bottom boundaries is given, i.e., $T = C_1$ at $x = 0$ and $T = C_2$ at $y = 0$, (C_1 and C_2 are constants), $f_1 + f_2 + f_3 + f_4 = T$ (for D2Q4). Then the unknowns are f_1 and f_3, because f_2 and f_4 can be obtained from the streaming step, i.e., $f_2(i, j) = f_2(i + 1, j)$ and $f_4(i, j) = f_4(i, j + 1)$, *and* f_1 and f_3 can be calculated using the 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., $f_1 = 0.5C_1 - f_2$, and similarly, $f_3 = 0.5C_2 - f_4$. This ensures the flux conservation, which conserves the physics of the problem. However, if T is given on the right boundary, then f_2 must be determined similarly.

Alternatively, since the temperature T_b is given at the boundary and from the definition,

$$T_b = \sum_i f_i \,, \tag{5.85}$$

the unknown distribution function can be determined. For example, for D2Q5 at the left-hand boundary, f_1 is unknown and can be calculated as

$$f_1 = T_b - f_0 - f_2 - f_3 - f_4 \,. \tag{5.86}$$

5.9.2 Adiabatic Boundary Conditions

Consider, for instance,

$$\frac{\partial T}{\partial x} = 0 \,. \tag{5.87}$$

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

$$\frac{T(i+1, j) - T(i, j)}{\Delta x} = 0 \tag{5.88}$$

(first-order accuracy). It can be simplified as $T(i, j) = T(i+1, j)$, and hence

$$\begin{aligned} f_1(i, j) + f_2(i, j) + f_3(i, j) + f_4(i, j) = f_1(i+1, j) + f_2(i+1, j) \\ + f_3(i+1, j) + f_4(i+1, j) \,. \end{aligned} \tag{5.89}$$

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.

Alternatively, the difference between the ingoing and outgoing distribution functions must be zero. As an example, for D2Q5, the left-hand boundary is adiabatic, and therefore, $f_1 - f_2 = 0$, i.e., $f_1 = f_2$ at the boundary (bounce-back).

5.9.3 Constant Flux Boundary Condition

Consider, for instance,

$$k\frac{\partial T}{\partial x} = q \,. \tag{5.90}$$

The finite difference approximation for the above boundary condition is

$$k\frac{T(i+1, j) - T(i, j)}{\Delta x} = q \,, \tag{5.91}$$

which can be rewritten as

$$T(i, j) = T(i + 1, j) - \frac{q\Delta x}{k}. \tag{5.92}$$

Therefore,

$$f_1(i, j) = f_1(i + 1, j) - \frac{q\Delta x}{k}. \tag{5.93}$$

Other boundary conditions can be formulated similarly, i.e., using the finite difference approach. It is also possible to use higher-order approximations. Alternatively, as discussed before, the following equation can be used at the boundary:

$$f_1(boundary) = f_2(boundary) + \frac{q''\tau c_s^2}{k} \tag{5.94}$$

5.10 Two-Dimensional Heat Diffusion in a Plate

A two-dimensional square slab as shown in Fig. 5.9 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 using the LB and FD methods. The thermal diffusivity is 0.25, and $xl = yl = 1.0$. The computer codes can be found in the appendix.

Note that for the stability condition, the time step for FDM is 0.2. There is no problem in using a time step of 1.0 with the LBM method. Hence, LBM is much faster and more efficient than FDM. Codes are given in the appendix. Figure 5.10 compares the predicted results of LBM and FDM.

5.10.1 D2Q9

For fluid flow problems (Navier–Stokes equation), where the advection term (nonlinear term) is important, the D2Q4 or D2Q5 model is insufficient to ensure macroscopic isotropy. It is common to use D2Q9. Even D2Q5 is sufficient for us for diffusion problems; however, it is illustrative to introduce D2Q9 for a diffusion problem. The knowledge obtained in solving diffusion problems using D2Q9 can be easily extended for advection–diffusion and Navier–Stokes problems.

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

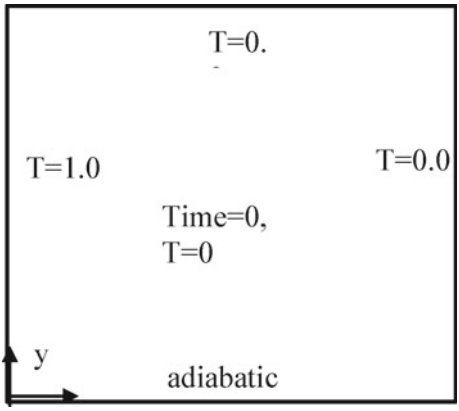


Fig. 5.9 A square (2-D) domain with coordinate system

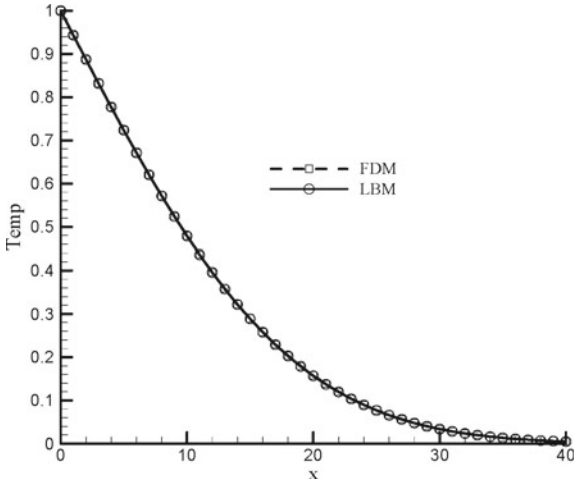


Fig. 5.10 Temperature distribution along the middle line ($y = 0.5$)

$$\alpha = \frac{\Delta x^2}{3\Delta t}\left(\frac{1}{\Omega} - \frac{1}{2}\right). \tag{5.95}$$

The domain of interest should be divided into lattices, as shown in Fig. 5.11.

The Solution Procedure:

Note that the numbering of f is arbitrary. For simplicity of coding, it is more convenient to use the numbering of streaming links, as shown in Fig. 5.11.

As usual, the LBM consists of two steps: collisions and streaming. The collisions step without forcing function is

$$f_k(x, y, t + \Delta t) = f_k(x, y, t)\,[1 - \Omega] + \Omega\, f_k^{eq}(x, y, t)\,, \tag{5.96}$$

$k = 0, \ldots, 8$.

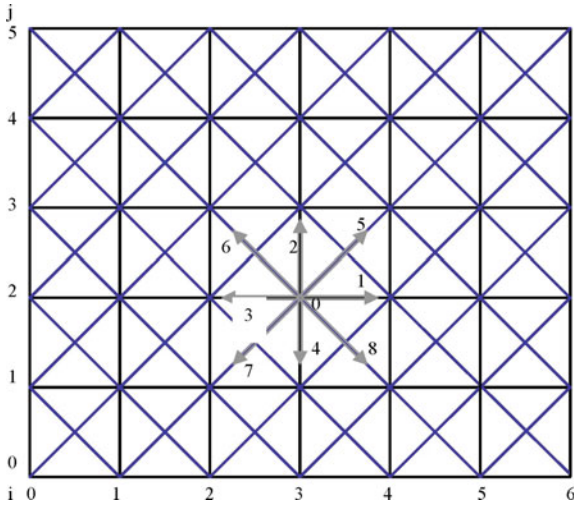
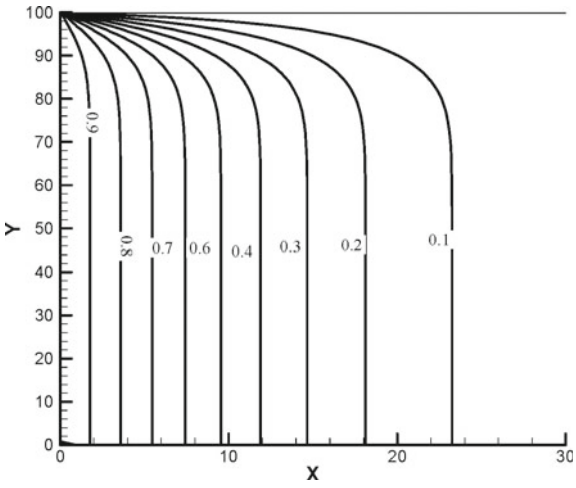


Fig. 5.11 Lattice arrangement for D2Q9



Isotherms, comparison between predictions of LBM and FDM. It is difficult to observe the difference between the predictions of the two methods

The streaming step is

$$f_k(x + \Delta x, y + \Delta y, t + \Delta t) = f_k(x, y, t + \Delta t), \tag{5.97}$$

$k = 1, \ldots, 8$.

In the 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, one should be careful not to override new values in the streaming process.

The values of the weighting factors are as follows:

- $w(0) = 4/9$;
- $w(1) = w(2) = w(3) = w(4) = 1/9$;
- $w(5) = w(6) = w(7) = w(8) = 1/36$.

The values of the streaming velocity along the lines are as follows:

$c_o(0, 0), c_1(c, 0), c_2(0, c), c_3(-c, 0), c_4(0, -c), c_5(c, c), c_6(-c, c), c_7(-c, -c)$ and $c_8(c, -c)$ where $c = \Delta x/\Delta t$.

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

$$f_k(i, j) = w(k)\phi(i, j)\,. \tag{5.98}$$

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

$$\phi(x, t) = \sum_{k=1}^{2} f_k(x, t)\,. \tag{5.99}$$

5.10.2 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: $f_5^{eq} - f_5 + f_7^{eq} - f_7 = w_5\phi_{wall} + w_7\phi_{wall}$, f_7 is known from the streaming process, and then $f_5 = w_5\phi_{wall} + w_7\phi_{wall} - f_7$. The other unknown streaming functions can be evaluated similarly.

The adiabatic boundary condition can be treated as discussed before. For details, see the computer code in the appendix. Computer code is given in the appendix for the same problem as discussed before for D2Q5, i.e., heat diffusion in a square plate.

5.10.3 Constant Flux Boundary Conditions

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

$$k\frac{\partial T}{\partial x} = q\,. \tag{5.100}$$

The above equation can be approximated using finite difference methods, which yield

$$k\frac{T(i + 1, j) - T(i, j)}{\Delta x} = q\,. \tag{5.101}$$

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

$$k\frac{T(1, j) - T(0, j)}{\Delta x} = q\,, \tag{5.102}$$

which can be rearranged as

$$T(0, j) = T(1, j) + \frac{q\Delta x}{k}\,. \tag{5.103}$$

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

5.11 Problems

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

2. Solve the one-dimensional problem with convective boundary condition.

3. Solve the diffusion problem for the plate shown in Fig. 5.12, using LBM. The plate's left-hand vertical boundary is maintained at $T = 1.0$, and the right-hand vertical boundary at $T = 0.0$. The other boundaries are assumed to be adiabatic. Plot steady-state isotherms and compare the result with FDM.

4. In certain applications, the thermal conductivity of the material may be a function of temperature. For instance, $\alpha = 0.25 + 0.5T^2$, where α is the thermal diffusivity of the material and T is dimensionless temperature. Solve the heat diffusion equation in such a material. The domain of the solution is 2 units in length and the initial temperature was $T = 0$. The temperature of the left-hand boundary is suddenly set to $T = 1.0$. Compare the results obtained by LBM and FDM at times 50 and 100. Hint: the relaxation time or relaxation frequency becomes a function of temperature.

5. Anisotropic heat diffusion: heat or mass diffusion in wood, crystal, sedimentary rocks, and fibers has a preferable direction. In other words, the heat or mass diffusion coefficient is directionally dependent. The heat diffusion equation in the two-dimensional domain can be expressed as

$$\rho c\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k_x\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y\frac{\partial T}{\partial y}\right). \tag{5.104}$$

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

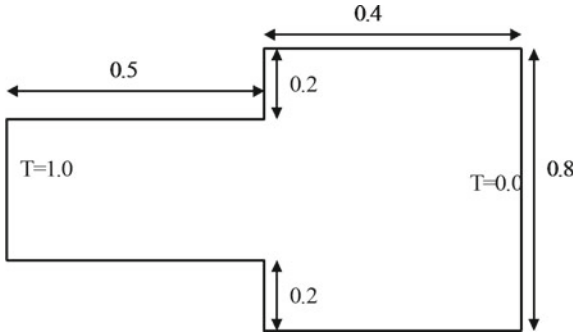


Fig. 5.12 Sketch of problem 4

$$\alpha_x = \frac{k_x}{\rho c} = \frac{\Delta x^2 c_s^2}{\Delta t}\left(\frac{1}{w_x} - 0.5\right) \tag{5.105}$$

and

$$\alpha_y = \frac{k_y}{\rho c} = \frac{\Delta y^2 c_s^2}{\Delta t}\left(\frac{1}{w_y} - 0.5\right) \tag{5.106}$$

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

6. One-dimensional heat diffusion in an inhomogeneous medium without heat generation with variable properties can be simulated by solving the following equation:

$$\frac{\partial(\rho c T)}{\partial t} = \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right), \tag{5.107}$$

where ρ, c, and k are functions of x. Initially, the medium is at a constant temperature, zero, and then suddenly, the left-hand boundary is exposed to a heat source at temperature equal to unity. The other boundary is kept at zero temperature. Estimate the temperature through the medium at times 1, 10, and 20. Hint: use the enthalpy concept $h = \rho c T$. The governing equation can be written as

$$\frac{\partial(h)}{\partial t} = \frac{\partial}{\partial x}\left(k\frac{\partial h/(\rho c)}{\partial x}\right). \tag{5.108}$$

Manipulate the right-hand side as differential and source terms.

Problem: A wall is exposed to solar flux,

$$I = I_o sin(\pi t/36000), \tag{5.109}$$

where I_o is maximum flux, say 1000 W/m^2. Time t is in seconds. The wall's inner surface is kept at $T = 22°$C. The thermal diffusivity of the wall is 4.0×10^{-7}. The wall thickness is 0.3 m. Calculate the temperature at the middle of the wall as a function of temperature. Compare your results with using the finite difference method (FDM).

Chapter 6
The Laplace, Poisson, and Biharmonic Equations

The Poisson and Laplace equations arise in many engineering applications, such as the potential theory of hydrodynamics and electromagnetism. Also, in solving problems in incompressible flow, the pressure equation (Poisson equation) needs to be solved in updating the Navier–Stokes solver. In solid mechanics, the Poisson equation arises in the analysis of small transverse deflections of a tensed membrane subjected to a transverse load. If the membrane is replaced by a linearly elastic plate with bending stiffness, the transverse deflections must obey the inhomogeneous biharmonic equation. The biharmonic equation arises also in the analysis of two-dimensional elasticity problems formulated in terms of the Airy stress function. In the following sections, the solution of the above-mentioned equations by the lattice Boltzmann method (LBM) will be undertaken, and the results will be compared with available analytic solutions or with conventional numerical methods predictions.

6.1 Laplace and Poisson Equations

The general form of the Poisson equation can be written as

$$\nabla^2 \Phi = S\,. \tag{6.1}$$

If S is set to zero, the equation is called the Laplace equation. Hence solving the Poisson equation is more general than solving the Laplace equation, which is a special case of the Poisson equation.

Equation (6.1) is of elliptic type, whereby the solution is dictated by the boundary conditions and the source term. One may say that the boundary data propagate into the interior of the domain instantaneously. The procedure to solve the Poisson equation is similar that used in solving the diffusion equation, as discussed previously. Since most numerical methods solve such problems iteratively, the speed of the solution's convergence depends primarily on the number of iterations needed to advance the boundary conditions to the domain.

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3_6

6.2 LBM Solution

The single relaxation LBM algorithm can be written as

$$f_i(x + c\Delta t + \Delta t) - f_i(x, t) = \frac{\Delta t}{\tau}[f_i^{eq}(x, t) - f_i(x, t)] + \left(1 - \frac{1}{2\tau}\right) w_i c_s^2 S \tag{6.2}$$

or

$$f_i(x + c\Delta t, t + \Delta t) = f_i(x, t)[1 - \Delta t/\tau] + \frac{\Delta t}{\tau} f_i^{eq} + \left(1 - \frac{1}{2\tau}\right) w_i c_s^2 S\,, \tag{6.3}$$

where the subscript i is the streaming direction, w_i is the weighting factor, c_s is the speed of sound, which is a constant depending on the lattice arrangement, and τ is the relaxation time. The increment Δt represents a collision step. The same procedure as that discussed in the previous chapter can be followed. As explained before, the solution of the Poisson equation requires that the domain reach equilibrium conditions instantaneously. A careful examination of Eq. (6.3) shows that the updated value of the distribution function (left-hand side of the equation) is some sort of averaging of old values of the distribution function and the equilibrium distribution function, for τ not equal to one. However, setting τ equal to Δt eliminates the contribution of the old value of the distribution function. Hence it is reasonable to set $\tau = \Delta t$ in Eq. (6.3). Therefore, Eq. (6.3) reduces to

$$f_i(x + c\Delta t, t + \Delta t) = f_i^{eq} + \left(1 - \frac{1}{2\tau}\right) w_i c_s^2 S\,. \tag{6.4}$$

Equation (6.4) is the basic driving force for solving the Poisson equation. The equilibrium distribution function is

$$f_i^{eq} = w_i \Phi\,, \tag{6.5}$$

where Φ is the dependent variable.

The same procedure is used to solve the diffusion equation, which must be followed by setting the relaxation parameter τ to 1.0. The following sections will present a few examples solved by LBM, and the predictions are compared either with the conventional finite difference method or the analytic solution.

Example 1: In the first example, we consider the following case of a constant source term:

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = 0.05\,, \tag{6.6}$$

with Φ set to zero at the boundaries of the domain of interest. The domain of interest is a square with unit length. The point $(x, y) = (0, 0)$ is located at the bottom left corner.

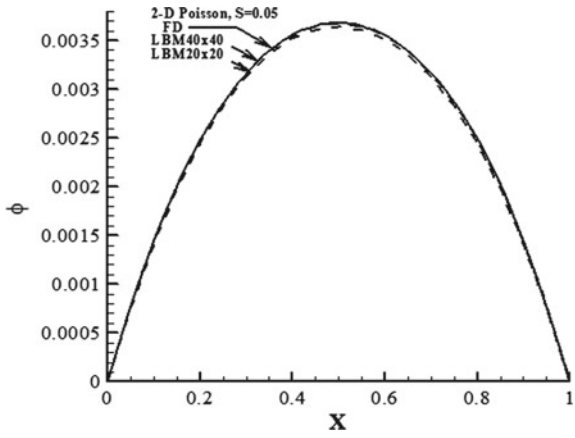


Fig. 6.1 Poisson equation with $S = 0.5$

For LBM, it is appropriate to rearrange the above equation as

$$0 = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} - 0.05\,. \tag{6.7}$$

In other words, the time derivative is set to zero.

For comparison purposes with the finite difference method, there is a need to properly scale the LB equation. The above equation is scaled by a length scale of one, while the LB equation is scaled by N, which is the number of lattices in the x-direction (nondimensional x equal to x/N). Therefore, the source term 0.05 should be scaled to $0.05/N^2$. In other words, the source term used with LBM is $0.05/N^2$ and not 0.05. Figure 6.1 shows a comparison between prediction of LBM using 20×20 and 40×40 lattices and the finite difference (FD) prediction using 80×80 nodes. The difference between predictions of LBM with a 40×40 lattice and FD is not noticeable. However, using a 20×20 lattice, the LBM prediction is slightly different from the FD predictions.

Example 2:
In a second example, the Poisson equation with a nonlinear source term is tested, namely

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = \Phi^2 + sin(2\pi x)cos(\pi y) + 0.1\,. \tag{6.8}$$

In the above equation, the source term S is

$$S = \Phi^2 + sin(2\pi x)cos(\pi y) + 0.1\,. \tag{6.9}$$

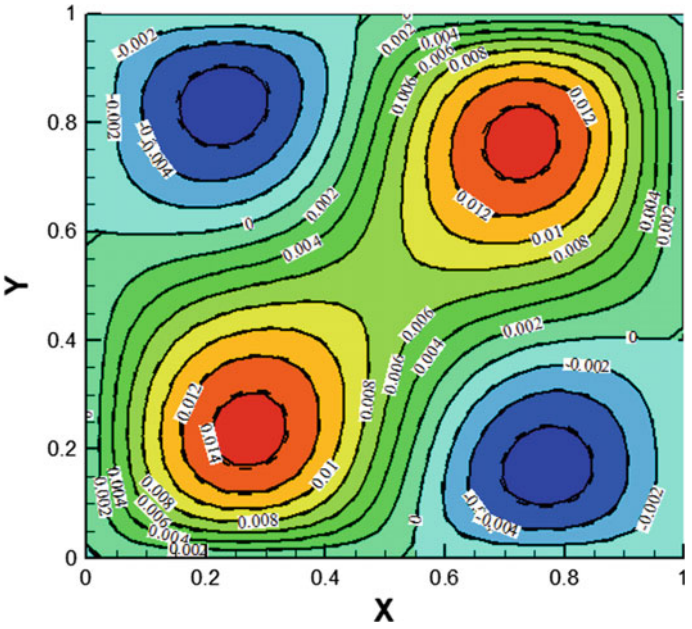


Fig. 6.2 Nonlinear poisson equation

The same boundary conditions as in the previous example are imposed (the dependent variable values at the boundary are set to zero). Figure 6.2 shows contour plots from the LBM (solid lines) and FD (dashed lines) predictions. Also, the source term must be scaled by $1/N^2$, as mentioned before. The difference between the predictions of both methods is insignificant.

Problem: Solve the two-dimensional Laplace equation

$$\nabla^2 \Phi = 0 \tag{6.10}$$

subject to following boundary conditions:

$$\Phi(x, 0) = \Phi(0, y) = \Phi(1, y) = 0 \tag{6.11}$$

and

$$\Phi(x, 1) = sin(\pi x). \tag{6.12}$$

Compare with the finite difference solution and the analytic solution, which is

$$\Phi(x, y) = \frac{sin(\pi x) sinh(\pi y)}{sinh(\pi)}. \tag{6.13}$$

6.3 Biharmonic Equation

The biharmonic equation can be written in terms of two coupled Poisson equations if the boundary values of the dependent function and its second derivatives are given. Hence the method developed for solving the Poisson equation should be equally valid for solving the biharmonic equation, i.e.,

$$\nabla^4 \Phi = \nabla^2(\nabla^2 \Phi) = S\,. \tag{6.14}$$

Hence the above equation can be written as two coupled equations, as

$$\nabla^2 \Phi = \Omega \tag{6.15}$$

and

$$\nabla^2 \Omega = S \tag{6.16}$$

with boundary conditions

$$\Phi \text{ and } \Phi'' \text{ known at the boundaries.} \tag{6.17}$$

However, if the first derivative of the function is known at the boundaries, it is not straightforward to use the slitting method. In such a case, we may use the shooting method, by guessing different boundary values of the second derivatives of the function and using interpolation to figure out the correct values of the second derivatives.

Example: let us develop LBM to solve a biharmonic equation with known second derivatives at the boundaries for the following problem:

$$\frac{d^4\Phi}{dx^4} = 1.0\,, \tag{6.18}$$

with the boundary conditions $\Phi = 0$, $\frac{d^2\Phi}{dx^2} = 1/12.$ at $x = 0$ and $x = 1.0$. Figure 6.3 compares the predicted results with the analytic solution, which is

$$\Phi = x^4/24 - x^3/12 + x^2/24. \tag{6.19}$$

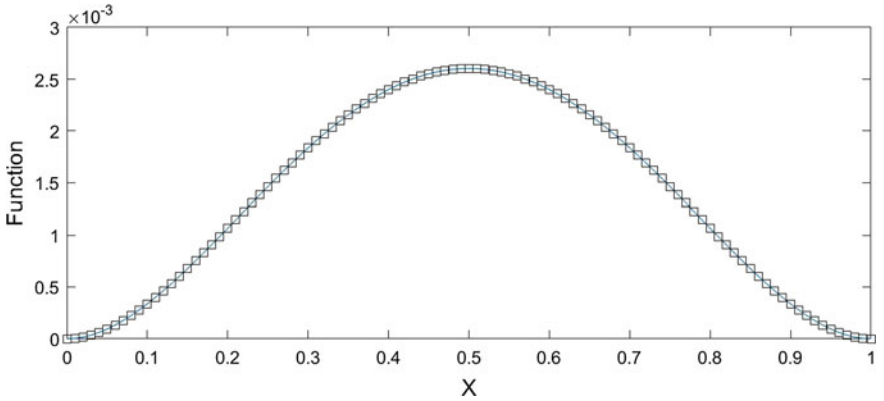


Fig. 6.3 Comparison of prediction with analytic solution

Problem: try to apply the developed method to the following equation:

$$\nabla^4 \Phi = -sin(\pi x) sin(\pi y)\,, \tag{6.20}$$

with boundary conditions $\Phi = \frac{\partial^2 \Phi}{\partial x^2} = \frac{\partial^2 \Phi}{\partial y^2} = 0$ along the boundaries of the same unit square used in the previous example. Its analytic solution is

$$\Phi = -(sin(\pi x) sin(\pi y))/(4\pi^4). \tag{6.21}$$

Chapter 7
Advection–Diffusion Problems

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

7.1 Advection

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

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = 0\,, \tag{7.1}$$

where ϕ is a dependent variable (mass, momentum, energy, species, etc.) and u is the advection velocity, i.e., the fluid flow velocity. The above equation has a first-order derivative in space and in time; hence it needs only a boundary condition and an initial condition for specifying the solution. Equation (7.1) conveys that the dependent variable, ϕ, advects with u-velocity in space as time proceeds. The property ϕ does not diffuse or disperse; it just translates with speed u. For instance, if the property ϕ has a given profile as shown in Fig. 7.1 at time $t = t_o$, then after $t = t_1$, the profile conserves its original shape and only translates (advects) with a given velocity u to a different location.

If u is not a function of ϕ, then the above equation is linear, and an analytic 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 in a stream of water, the oil advects without diffusion. It conserves its shape and density as it moves with the velocity of the water stream.

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3_7

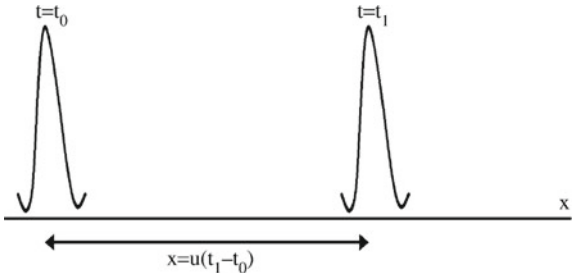


Fig. 7.1 Illustration of advection processes

The three-dimensional advection equation can be written as

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} + w\frac{\partial \phi}{\partial z} = 0\,, \tag{7.2}$$

which is an extension of the 1-D equation.

7.2 Advection–Diffusion Equations

The advection–diffusion process is a process in which 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 nature and in industrial and engineering applications, where it is usually called a transport problem.

The one-dimensional advection–diffusion equation can be expressed in Cartesian coordinates as

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = \alpha\frac{\partial^2 \phi}{\partial x^2}\,, \tag{7.3}$$

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

Figure 7.2 illustrates the concept of the advection–diffusion process. The initial function propagates in space as time proceeds, while the shape of the function simultaneously changes (flattens) due to the diffusion process. However, the area under the carve will be conserved.

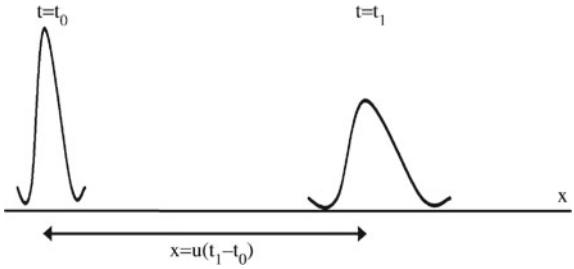


Fig. 7.2 Advection–diffusion processes

7.2.1 The Finite Difference Method

In numerical methods, the advection term may cause problems if it is not properly approximated. The flow direction affects the information flow; the upstream should have an effect on the downstream, while the effect of the downstream condition on the upstream condition depends on the strength of the flow (magnitude of flow velocity) and the magnitude of the diffusion coefficient. In more precise wording, it depends on the ratio of advection to diffusion, called the Peclet number. Hence using the 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 an unstable solution, because we are giving the same weight to both the left and right sides of the function ϕ. In other words, we are not taking into consideration that more information is transferring in the streamwise direction. In order to take into the consideration the direction of information flow, it is more appropriate to use the upwind scheme,

$$\frac{\partial\phi}{\partial x} = \begin{cases} \frac{\phi(i)-\phi(i-1)}{\Delta x} & \text{if } u > 0\,, \\ \frac{\phi(i+1)-\phi(i)}{\Delta x} & \text{if } u \le 0\,. \end{cases} \tag{7.4}$$

Note that Eq. (7.4) is accurate to first order 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 upwind schemes, which requires that more nodes be involved in Eq. (7.4). For more details, we refer to textbooks on numerical methods for computational fluid dynamics (CFD).

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

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

For stability reasons, Δt and Δx are set to 0.25 and 0.5, respectively. Since u is positive, the first derivative is approximated as given in the first line of Eq. (7.4).

Forward differences are used in approximating the time derivative. Again, the approximation is first-order accurate. A better approximation is possible, using, for example, the Crank–Nicolson or Runge–Kutta method. For details, we refer to numerical method textbooks on the subject.

7.2.2 The Lattice Boltzmann Method

The Lattice Boltzmann Method for Advection–Diffusion Problems

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

$$f_k(x + \Delta x, t + \Delta t) = f_k(x, t)\,[1 - \Omega] + \Omega\, f_k^{eq}(x, t)\,. \tag{7.5}$$

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

$$f_k^{eq} = w_k \phi(x, t)\left[1 + \frac{c_k \bullet \vec{u}}{c_s^2}\right], \tag{7.6}$$

where $\vec{u}$ is the advection velocity vector. For example, in two dimensions, the vector $\vec{u}$ is equal to $ui + vj$, where i and j are unit vectors along the x- and y-directions, and c_k is a unit vector along the streaming direction. In general,

$$c_k = \frac{\Delta x}{\Delta t} i + \frac{\Delta y}{\Delta t} j\,. \tag{7.7}$$

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

$$c_s = \frac{1}{\sqrt{2}}, \tag{7.8}$$

while for D1Q3, D2Q5, D2Q9, and D3Q15, we have

$$c_s = \frac{1}{\sqrt{3}}. \tag{7.9}$$

The same procedure can be followed as that used in diffusion problems. The LBM code for this problem is the same as for the diffusion problem in Chap. 3), except that an extra term is added for calculating f^{eq}.

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

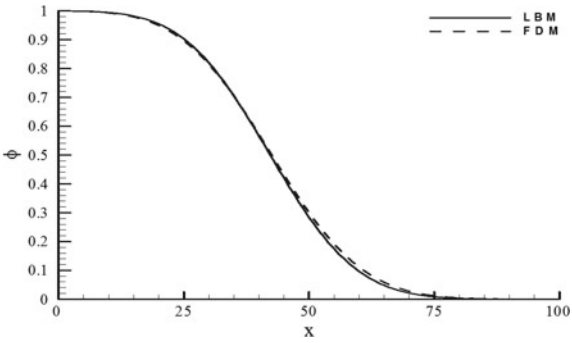


Fig. 7.3 Comparison between predictions of LBM and FDM for the advection–diffusion problem

The thermal diffusivity is set to 0.25, and the velocity to 0.1. An explicit finite difference method is used in marching the solution with time. Hence to ensure a stable solution, the time step and spatial increment are set to 0.25 and 0.5, respectively. For a total time of 400, we use 1600 time steps. For a total domain of length 100, we use 200 nodes. To match the finite difference prediction, the same values for thermal diffusivity and velocity are used in the LBM simulation as for FD. However, Δt and Δx are set to 1.0. Therefore, only 400 time steps are needed. Figure 7.3 compares the results obtained using finite differences and D1Q3, LBM method for the above-mentioned problem. The LBM method is more stable and produces accurate results compared with the finite difference method (FDM).

7.3 Equilibrium Distribution Function

In another approach to constructing an equilibrium distribution function for the advection–diffusion problem, it is appropriate to assume that the equilibrium distribution functions can be approximated as a polynomial in velocity. Let

$$f_i^{eq} = A_i + B_i c_i . u \, . \tag{7.10}$$

The equilibrium distribution function must satisfy conservation of mass and momentum:

$$\sum_{i=1}^{2} f_i^{eq} = \Theta \tag{7.11}$$

and

$$\sum_{i=1}^{2} f_i^{eq} c_i = \Theta u \, . \tag{7.12}$$

Hence Eq. (4.11) yields

$$\sum_{i=1}^{2} A_i = A_1 + B_1 u + A_2 - B_2 u = \Theta\,. \tag{7.13}$$

This implies that

$$A_1 + A_2 = \Theta \tag{7.14}$$

and

$$B_1 u - B_2 u = 0, or \;\; B_1 = B_2\,. \tag{7.15}$$

This is equation (4.12),

$$\sum_{i=1}^{2} (A_i c_i + B_i c_i c_i u) = u\Theta \tag{7.16}$$

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

$$\sum_{i=1}^{2} A_i c_i = 0\,. \tag{7.17}$$

Hence $A_1 = A_2$. In general,

$$A_i = \omega_i \Theta\,, \tag{7.18}$$

where $\omega_i = A_i/\Theta$. Also, from Eq. (4.19), we obtain

$$B_i c_i c_i = \Theta\,. \tag{7.19}$$

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

$$w_i c_i c_i = \frac{1}{c_s^2}\,. \tag{7.20}$$

We then have the following relation:

$$B_i = \frac{w_k}{c_s^2}\Theta\,. \tag{7.21}$$

Hence

$$f_i^{eq} = w_i \Theta \left(1 + \frac{u c_i}{c_s^2}\right)\,. \tag{7.22}$$

In order to relate LBM to the macroscale problem, a multiscale expansion is needed. In the following sections, the Chapman–Enskog expansion for the advection–diffusion problem will be discussed in detail.

7.4 Chapman–Enskog Expansion

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

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = \Gamma \frac{\partial^2 T}{\partial x^2} \tag{7.23}$$

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

$$\frac{\partial T(x,t)}{\partial t} + u\epsilon\frac{\partial T(x,t)}{\partial x} = \Gamma\ \epsilon^2 \frac{\partial^2 T(x,t)}{\partial x^2}\,. \tag{7.24}$$

If we now scale time t by $1/\epsilon^2$ as we did in Chap. 3 for the diffusion equation, the first term on the left-hand side of the equation balances the diffusion term on the right-hand side of the equation, and hence the effect of the advection term will be neglected. If we scale time t by $1/\epsilon$, then 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.,

$$\frac{\partial}{\partial t} = \epsilon\frac{\partial}{\partial t_1} + \epsilon^2\frac{\partial}{\partial t_2}\,. \tag{7.25}$$

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

$$T(x,t) = \sum_{i=1}^{i=2} f_i(x,t) = f_1(x,t) + f_2(x,t)\,. \tag{7.26}$$

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

$$f_i^{eq} = w_i T(x,y)(1 + Ac_i.u) \tag{7.27}$$

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

$$\sum_i f_i^{eq} = T = w_1 T + w_2 T \,. \tag{7.28}$$

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

The weight factors must satisfy the relation

$$\sum_{i=1}^{i=2} w_i = 1 \,, \tag{7.29}$$

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

$$f_i(x,t) = f_i^o + \epsilon f_i^1 + \epsilon^2 f_i^2 + \cdots \,, \tag{7.30}$$

where f_i^o is the distribution function at the equilibrium conditions, equal to f_i^{eq}. Summing the above equation leads to

$$\sum_{i=1}^{i=2} f_i(x,t) = \sum_{i=1}^{i=2} [f_i^o + \epsilon f_i^1 + \epsilon^2 f_i^2 + \cdots] \,. \tag{7.31}$$

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

$$\sum_{i=1}^{i=2} f_i^1 = 0 \,, \tag{7.32}$$

$$\sum_{i=1}^{i=2} f_i^2 = 0 \,. \tag{7.33}$$

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

$$\begin{aligned} f_i(x + c_i \Delta t, t + \Delta t) = f_i(x,t) + \frac{\partial f_i}{\partial t}\Delta t + \frac{\partial f_i}{\partial x} c_i \Delta t \\ + 1/2 \Delta t^2 \left(\frac{\partial^2 f_i}{\partial t^2} + 2 \frac{\partial^2 f_i}{\partial t \partial x} c_i + \frac{\partial^2 f_i}{\partial x^2} c_i c_i \right) + O(\Delta t)^3 \,. \end{aligned} \tag{7.34}$$

Introduction 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 $\epsilon\frac{1}{\partial x}$, keeping terms with ϵ^2 yields

$$f_i(x+c_i\Delta t,t+\Delta t)=f_i(x,t)+\epsilon\frac{\partial f_i}{\partial t_1}\Delta t+\epsilon^2\frac{\partial f_i}{\partial t_2}\Delta t+\epsilon\frac{\partial f_i}{\partial x}c_i\Delta t + \frac{\Delta t^2}{2}\left(\frac{\epsilon^2\partial^2 f_i}{\partial t_1^2}+2\frac{\epsilon^2\partial^2 f_i}{\partial t_1\partial x}c_i+\frac{\epsilon^2\partial^2 f_i}{\partial x^2}c_ic_i\right)+O(\Delta t)^3\,. \tag{7.35}$$

The lattice Boltzmann equation can be written as

$$f_i(x+c_i\Delta t,t+\Delta t)=f_i(x,t)+\frac{\Delta t}{\tau}[f_i^o(x,t)-f_i(x,t)]\,. \tag{7.36}$$

Substituting Eq. (7.35) into the above equation and retaining terms up to order ϵ^2 yields

$$\frac{1}{\tau}[f_i^o(x,t)-f_i(x,t)]=\epsilon\frac{\partial f_i}{\partial t_1}+\epsilon^2\frac{\partial f_i}{\partial t_2}+\epsilon\frac{\partial f_i}{\partial x}c_i + \frac{\epsilon^2\Delta t}{2}\left(\frac{\partial^2 f_i}{\partial t_1^2}+2\frac{\partial^2 f_i}{\partial x\partial t_1}+\frac{\partial^2 f_i}{\partial x^2}c_ic_i\right)+O(\Delta t)^2+O(\epsilon^3)\,, \tag{7.37}$$

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

$$-\frac{1}{\tau}(\epsilon f_i^1+\epsilon^2 f_i^2+\cdots)=\epsilon\frac{\partial f_i^o}{\partial t_1}+\epsilon^2\frac{\partial f_i^1}{\partial t_1}+\epsilon^2\frac{\partial f_i^o}{\partial t_2}+\epsilon\frac{\partial f_i^o}{\partial x}c_i+\epsilon^2\frac{\partial f_i^1}{\partial x}c_i + \frac{\Delta t\epsilon^2}{2}\left[\frac{\partial^2 f_i^o}{\partial t_1^2}+2\frac{\partial^2 f_i^o}{\partial x\partial t_1}c_i+\frac{\partial f_i^o}{\partial x}c_ic_i\right]+O(\epsilon^3)+)(\Delta t^2)\,. \tag{7.38}$$

Collecting terms of order ϵ from both sides of the above equations results in

$$-\frac{1}{\tau}f_i^1=\frac{\partial f_i^o}{\partial t_1}+\frac{\partial f_i^o}{\partial x}c_i\,. \tag{7.39}$$

Later on, we may need the derivative of f_i^1 with respect of t_1 and x; hence by taking the derivative of above equation, we obtain

$$-\frac{1}{\tau}\frac{\partial f_i^1}{\partial t_1}=\frac{\partial^2 f_i^o}{\partial t_1^2}+\frac{\partial^2 f_i^o}{\partial t_1\partial x}c_i \tag{7.40}$$

and

$$-\frac{1}{\tau}\frac{\partial f_i^1}{\partial x} = \frac{\partial^2 f_i^o}{\partial t_1 \partial x} + \frac{\partial^2 f_i^o}{\partial x^2} c_i \,. \tag{7.41}$$

We now multiply Eq. (7.41) by c_i, add it to Eq. (7.40), and rearrange the summed equation to obtain

$$-\frac{1}{\tau}\left(\frac{\partial f_i^1}{\partial t_1} + \frac{\partial f_i^1}{\partial x} c_i\right) = \frac{\partial^2 f_i^o}{\partial t_1^2} + 2\frac{\partial^2 f_i^o}{\partial t_1 \partial x} c_i + \frac{\partial^2 f_i^o}{\partial x^2} c_i c_i \,. \tag{7.42}$$

By integrating Eq. (7.40) and summing over all values of $i = 1, 2$, we obtain

$$-\frac{1}{\tau}\sum_{i=1}^{2} f_i^1 = \sum_{i=1}^{2}\frac{\partial f_i^o}{\partial t_1} + \sum_{i=1}^{2}\frac{\partial f_i^o}{\partial x} c_i \,. \tag{7.43}$$

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

$$0 = \frac{\partial T}{\partial t_1} + \frac{\partial (uT)}{\partial x} \,, \tag{7.44}$$

which is the advection equation. Note that t_1 balance advection term.

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

$$-\frac{1}{\tau} f_i^2 = \frac{\partial f_i^1}{\partial t_1} + \frac{\partial f_i^o}{\partial t_2} + \frac{\partial f_i^1}{\partial x} c_i + \frac{\Delta t}{2}\left[\frac{\partial^2 f_i^o}{\partial t_1^2} + 2\frac{\partial^2 f_i^o}{\partial x \partial t_1} c_i + \frac{\partial^2 f_i^o}{\partial x^2} c_i c_i\right] . \tag{7.45}$$

Substituting Eq. (7.38) into the above equation yields

$$-\frac{1}{\tau} f_i^2 = \frac{\partial f_i^1}{\partial t_1} + \frac{\partial f_i^o}{\partial t_2} + \frac{\partial f_i^1}{\partial x} c_i - \frac{\Delta t}{2\tau}\left[\frac{\partial f_i^1}{\partial t_1} + \frac{\partial f_i^1}{\partial x} c_i\right] . \tag{7.46}$$

Collecting similar terms gives us

$$-\frac{1}{\tau} f_i^2 = \frac{\partial f_i^o}{\partial t_2} + \left(1 - \frac{\Delta t}{2\tau}\right)\left[\frac{\partial f_i^1}{\partial t_1} + \frac{\partial f_i^1}{\partial x} c_i\right] . \tag{7.47}$$

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

$$\sum_i \left[-\frac{1}{\tau} f_i^2\right] = \sum_{i=1}^{2}\frac{\partial f_i^o}{\partial t_2} + \left(1 - \frac{\Delta t}{2\tau}\right)\left[\sum_{i=1}^{2}\frac{\partial f_i^1}{\partial t_1} + \sum_{i=1}^{2}\frac{\partial f_i^1}{\partial x} c_i\right] \tag{7.48}$$

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

$$\sum_{i=1}^{2} \frac{\partial f_i^o}{\partial t_2} = \frac{\partial \sum_{i=1}^{2} f_i^o}{\partial t_2} = \frac{\partial \Theta}{\partial t_2} . \tag{7.49}$$

The term of

$$\sum_i \left[\frac{\partial^2 f_i^o}{\partial x^2} c_i c_i \right] = \frac{\partial^2}{\partial x^2} \left(\sum_i [f_i^o c_i c_i] = \frac{\partial^2 \Theta}{\partial x^2} \right. \tag{7.50}$$

Hence Eq. (7.48) can be simplified as

$$0 = \frac{\partial \Theta}{\partial t} - \left(\tau - \frac{\Delta t}{2} \right) \frac{\partial^2 \Theta}{\partial x^2} . \tag{7.51}$$

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

$$\Gamma = \tau - \frac{\Delta t}{2} . \tag{7.52}$$

7.4.1 Two-Dimensional Advection–Diffusion Problems

The finite difference approximation of

$$\frac{\partial \phi}{\partial t} + u \frac{\partial \phi}{\partial x} + v \frac{\partial \phi}{\partial y} = \alpha \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \right) \tag{7.53}$$

is straightforward for a given velocity field. However, care should be taken if the velocity direction is changing.

The first-order accurate upwind scheme can be written as

$$\frac{\partial \phi}{\partial x} = \begin{cases} \frac{\phi(i,j) - \phi(i-1,j)}{\Delta x} & \text{if } u > 0 , \\ \frac{\phi(i+1,j) - \phi(i,j)}{\Delta x} & \text{if } u \leq 0 , \end{cases} \tag{7.54}$$

and

$$\frac{\partial \phi}{\partial y} = \begin{cases} \frac{\phi(i,j) - \phi(i,j-1)}{\Delta y} & \text{if } \mathrm{v} > 0 \\ \frac{\phi(i,j+1) - \phi(i,j)}{\Delta y} & \text{if } \mathrm{v} \leq 0 \end{cases} \tag{7.55}$$

As an example, let us consider the flow in the square porous domain shown in Fig. 7.4. 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 a temperature of

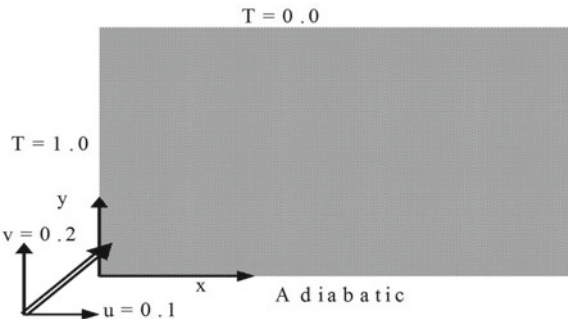


Fig. 7.4 Flow and heat transfer in a porous medium

unity at time 0. Find the temperature distribution in the domain at time 200 for a material of thermal diffusivity 1.0. The velocity field of $u = 0.1$ and $v = 0.2$ is kept constant in the domain.

For the finite difference method, it is sufficient to use rectangular grids of $\Delta x = 1.0$ and $\Delta y = 1.0$. To ensure a stable solution, set $\Delta t = 0.2$ for FDM. The computer code is given in the appendix.

7.5 Two-Dimensional Lattice Boltzmann Method

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

7.5.1 D2Q5

The lattice arrangement for D2Q5 is shown in Fig. 7.5. The stagnant distribution function f_0 is located at the center. The method of solution is similar to the LBM introduced to solve the diffusion equation except for the equilibrium term, as mentioned before.

The lattice Bhatnagar, Gross, Krook (BGK) equationis

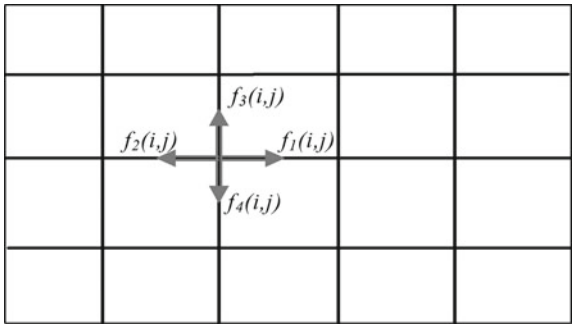


Fig. 7.5 Lattice arrangement for D2Q4

$$f_k(x + \Delta x, t + \Delta t) = f_k(x, t)\,[1 - \Omega] + \Omega\, f_k^{eq}(x, t)\,, \tag{7.56}$$

where k takes values from 1 to 5 (from 0 to 4).

The streaming and collision processes are similar to those explained in Chap. 5. The equilibrium part can be calculated using the following equation:

$$f_k^{eq} = w_k \phi(x, t)\left[1 + \frac{c_k \bullet \mathbf{u}}{c_s^2}\right] \tag{7.57}$$

where $w_k = 1/6$ for $k = 1, 4$ and for $k = 0$, $w_0 = 4/6$. For $\Delta x = \Delta y = \Delta t = 1.0$, we have

$$\frac{c_k \bullet \mathbf{u}}{c_s} = \begin{cases} 3u & k = 1\,, \\ -3u & k = 2\,, \\ 3\mathrm{v} & k = 3\,, \\ -3\mathrm{v} & k = 4\,, \\ 0 & k = 0\,. \end{cases} \tag{7.58}$$

Note that c_k is a unit vector along the streaming line and

$$c_s = \frac{1}{\sqrt{3}} \tag{7.59}$$

for D2Q5.

The computer code (see the appendix) is developed to simulate heat diffusion and advection in a permeable porous medium. The velocity components (u and v) in the domain are known constants ($u = 0.1$, $v = 0.4$). The thermal diffusivity is set to 1.0. The left-hand boundary is set to 1.0, while the top of the domain's boundary is set to zero. The rest of the boundaries are assumed to be adiabatic. Figure 7.6 shows the isotherms predicted for 400 time steps. The interested reader can change the parameters and examine the effect of those parameters on the temperature field (Fig. 7.7).

7.5.2 D2Q9

The lattice arrangement for D2Q9 is shown in (Fig. 7.8), which is the same as in Fig. 5.12.

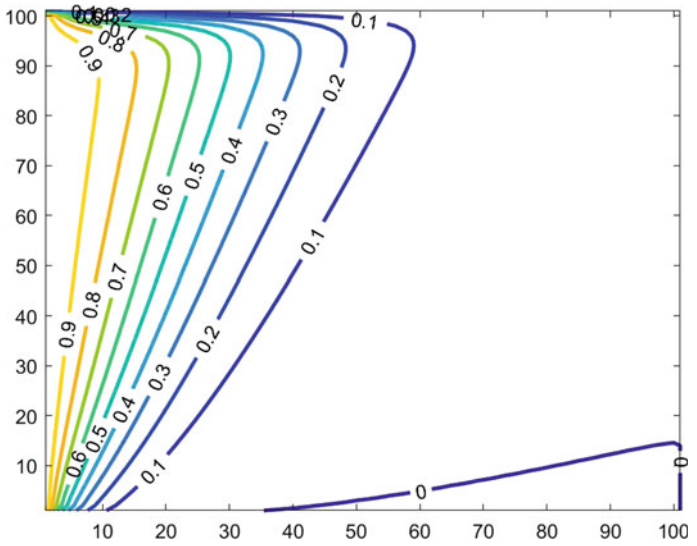


Fig. 7.6 Isotherms, D2Q5

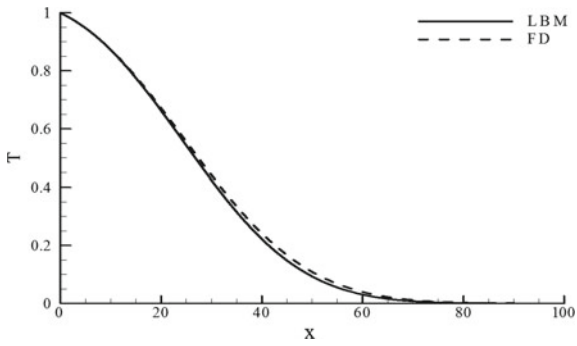


Fig. 7.7 Temperature profile at the midplane ($y = 0.5$) predicted by LBM D2Q5, D2Q9, and FDM

The procedure is the same as discussed before, with the f^{eq} as mentioned before, namely

$$f_k^{eq} = w_k \phi(x,t)\left[1 + \frac{c_k \bullet \mathbf{u}}{c_s^2}\right]. \tag{7.60}$$

Hence $f_1^{eq} = w_1\phi(1.0 + 3.0u)$
$f_2^{eq} = w_2\phi(1.0 + 3.0v)$
$f_3^{eq} = w_3\phi(1.0 - 3.0u)$

$f_4^{eq} = w_4\phi(1.0 - 3.0v)$
$f_5^{eq} = w_5\phi(1.0 + 3.0(u + v))$

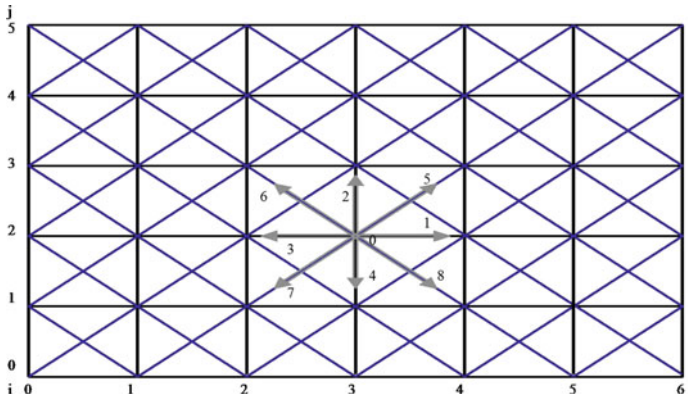


Fig. 7.8 Lattice arrangement for D2Q9

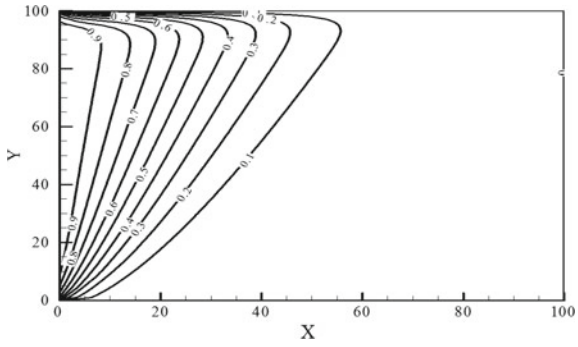


Fig. 7.9 Isotherms at time 400

and so on. For details of implementation and coding, see the code for D2Q9 in the appendix. Notice that in programming, $f1, f2, f3, \ldots$ are notated with subscripts. This is for two reasons: making the programming compact and the code easily extendible to three dimensions. The code simulates heat transfer as in the previous problem, with $v = 0.4$ and $u = 0.1$. The conditions on all boundaries are kept at zero temperature, except that the left side is suddenly changed to unity. The thermal diffusivity of the medium is set to 1.0. The simulation is carried out for 400 time steps. The code is given in the appendix. Figure 7.9 shows a plot of the isotherms at time 400. Figure 7.10 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 nondimensionalized or scaled after the calculation. For instance, the height and length of the domain can be divided by the total height of the domain, the velocity field can be divided by the inlet velocity, and so on.

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

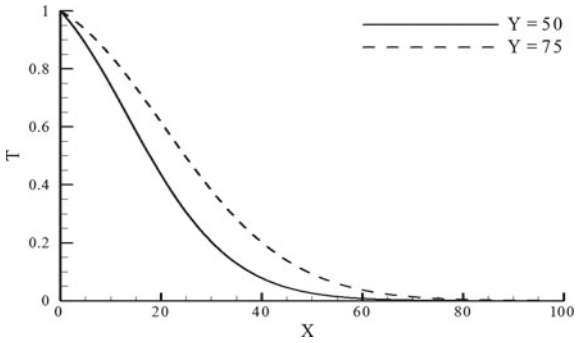


Fig. 7.10 Temperature profiles at mid-height ($Y = 50$) and at three-quarter height ($Y = 75$) for time 400

7.6 Problems

7.6.1 Combustion in a Porous Layer

Porous burners, or combustion in porous media, can be modeled as two separate energy equations, one for the solid phase and one for the gaseous phase. Assume a premixed gaseous fuel with air flowing through a porous layer, as shown in Fig. 7.11. Combustion takes place inside the layer. For simplicity, let us model the combustion as a constant heat source term released at a certain location in the porous medium.

The following model equations need to be solved. The energy equation for gaseous phase,

$$\frac{\partial(\rho_g c_g T_g)}{\partial t} + u\frac{\partial(\rho_g c_g T_g)}{\partial x} = \frac{\partial}{\partial x}\left(k_g \frac{\partial T_g}{\partial x}\right) - h_v(T_g - T_s) + S\,, \tag{7.61}$$

and the energy equation for solid phase,

$$\frac{\partial(\rho_s c_s T_s)}{\partial t} = \frac{\partial}{\partial x}\left(k_s \frac{\partial T_s}{\partial x}\right) + h_v(T_g - T_s), \tag{7.62}$$

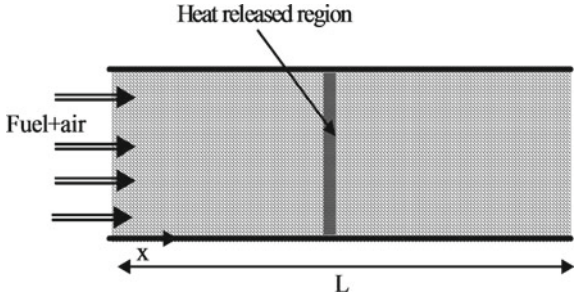


Fig. 7.11 Schematic diagram of porous burner

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

As for the boundary conditions, at the inlet $x = 0$, assume for simplicity that $T_g = T_s = T_{in}$, and at the outlet $x = L$, assume that the temperature gradients are zero.

Modify above code to simulate flow in a porous layer. Hint: use two distribution functions, one for the gas's temperature and the other for the solid's temperature.

7.6.2 Cooling a Heated Plate

A fully developed flow between parallel plates, Fig. 7.12, can be assumed as $u/u_{in} = 1.5(1 - y/H)$. A cold flow with a temperature of 20°C enters the channel. The bottom of the channel is maintained at $T = 80$°C, while the top plate is assumed to be perfectly insulated. Calculate the temperature field between the plates and the rate of heat transfer from the bottom plate.

7.6.3 Coupled Equations with Source Term

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

$$\frac{\partial \Theta}{\partial t} + u\frac{\partial \Theta}{\partial x} = \frac{1}{Pr}\frac{\partial^2 \Theta}{\partial x^2} \tag{7.63}$$

and

$$\frac{\partial \Phi}{\partial t} + u\frac{\partial \Phi}{\partial x} = \frac{1}{Pr\ Sc}\left[\frac{\partial^2 \Phi}{\partial x^2} - \frac{\partial^2 \Theta}{\partial x^2}\right] \tag{7.64}$$

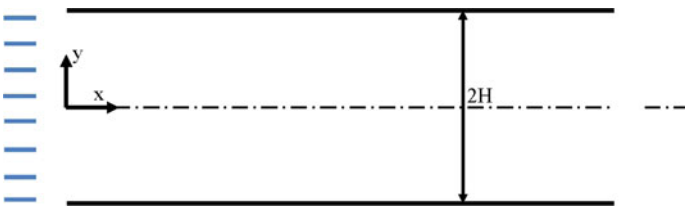


Fig. 7.12 Forced convection problem diagram

respectively. The source term is the second term on the right-hand side of Eq. (4.64) ($\frac{1}{Pr\ Sc}\frac{\partial^2\Theta}{\partial x^2}$). The source (or sink) term needs to be modeled correctly. Hence to solve this kind of problem, it is necessary to use two distribution functions, one for the energy equation and another for the 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

$$S_i = w_i\ \Delta t\ S\,, \tag{7.65}$$

where $\mathrm{S} = -\frac{1}{Pr\ Sc}\frac{\partial^2\Theta}{\partial x^2}$ and w_i is the weighting coefficient. The second derivative can be approximated using a 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 $\Phi = 1.0$ is forced into the domain. Compare the results of D1Q2 and D1Q3. You will notice that the prediction of D1Q2 may be oscillatory, which is physically incorrect. We found that the D1Q2 scheme may produce an oscillatory solution with source term, which is not the case for D1Q3. Therefore, it is recommended to use D1Q3 for 1-D problems, and D2Q5 for 2-D problems.

Chapter 8
Isothermal Incompressible Fluid Flow

In this chapter, fluid flow problems will be explained. The lattice Boltzmann method for solving various isothermal two-dimensional fluid flow problems will be discussed. The implementation of different boundary and flow conditions will be detailed. Extending the method to three-dimensional problems is straightforward.

8.1 Navier–Stokes Equation

In a continuum domain, fluid flow is governed by Navier–Stokes (NS) equations (continuity and momentum equations). For two-dimensional incompressible flow, the conservative form for the NS equation can be written in Cartesian coordinates, without body force, as follows. For the x-momentum,

$$\frac{\partial \rho u}{\partial t} + \frac{\partial (\rho u u)}{\partial x} + \frac{\partial (\rho \mathrm{v} \mathrm{u})}{\partial \mathrm{y}} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu \frac{\partial u}{\partial y}\right), \tag{8.1}$$

and for the y-momentum,

$$\frac{\partial \rho \mathrm{v}}{\partial t} + \frac{\partial (\rho u \mathrm{v})}{\partial x} + \frac{\partial (\rho \mathrm{v} \mathrm{v})}{\partial \mathrm{y}} = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left(\mu \frac{\partial \mathrm{v}}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu \frac{\partial \mathrm{v}}{\partial y}\right). \tag{8.2}$$

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 viscosity.

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

$$\frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho \mathrm{v})}{\partial \mathrm{y}} = 0 \tag{8.3}$$

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3_8

must be satisfied. 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 that arises in solving the incompressible NS. Different techniques are available to solve NS equations, such as the finite difference, finite volume, and finite element methods. The main problems in solving fluid flow problems using NS are twofold. 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 upwind schemes have been used to approximate the advection terms. Higher-order schemes reduce the numerical diffusion problems. However, it decreases the stability limit or adds a dispersion effect. In order to overcome these problems, different correction methods have been used. The main idea is to use a first-order upwind scheme and 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 the updating process, the continuity equation needs to be satisfied. This step requires the solution of the Laplace equation (pressure and pressure correction equations). Solving the Laplace equation consumes significant computer time. In fact, the majority of the time needed to solve NS is spent in solving the Laplace equation.

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

8.2 Lattice Boltzmann

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

8.2.1 The BGK Approximation

The Boltzmann equation is an integrodifferential equation. One of the problems in solving the Boltzmann equation is the complicated nature of the collision integral. The main practice for overcoming this difficulty is to use the BGK (Bhatnagar–

Gross–Krook) approximation, as discussed above. The lattice Boltzmann method simulates incompressible flow with a low Mach number ($Ma = u/c_s$, where u is the macroscopic flow velocity and c_s is the speed of sound) condition and weak variation in density. Hence in simulations, the characteristic flow velocity must be set to small values (order of 0.1) in order to reduce the error of simulation. Therefore, in simulating a flow at high Reynolds number (uN/ν), either the number of lattices N should be increased or the kinematic viscosity ν should be decreased. Care should be taken in using very small values of ν, which may cause stability problems. Note that there is no relationship between flow velocity and velocity used in LBM simulation, as far as Reynolds number is matched between the physical system and the LBM simulation.

D2Q9, BKG–LBM

In the following paragraphs, we shall discuss the BKG–LBM method D2Q9.

The BKG–lattice Boltzmann method is the same as introduced before. The lattice Boltzmann equation can be written as

$$f_k(x + \Delta x, t + \Delta t) = f_k(x, t)\,[1 - \Omega] + \Omega\, f_k^{eq}(x, t)\,, \tag{8.4}$$

where $\Omega = 1/\tau$, τ being the relaxation time. The only difference is an extra term added to the equilibrium distribution term,

$$f_k^{eq} = w_k \rho(x, t) \left[1 + \frac{c_k.\mathbf{u}}{c_s^2} + \frac{1}{2}\frac{(c_k.\mathbf{u})^2}{c_s^4} - \frac{1}{2}\frac{\mathbf{u}^2}{c_s^2}\right], \tag{8.5}$$

where

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

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

and

$$\mathbf{u} = ui + vj\,. \tag{8.6}$$

The above equation ensures second-order accuracy. For greater accuracy, extra terms need to be added. Fourth-order accuracy is achieved with

$$f_k^{eq} = w_k \rho(x, t) \left[1 + \frac{c_k.\mathbf{u}}{c_s^2} + \frac{1}{2}\frac{(c_k.\mathbf{u})^2}{c_s^4} - \frac{1}{2}\frac{\mathbf{u}^2}{c_s^2} + \frac{(c_k.\mathbf{u})^3}{2c_s^6} - \frac{(c_k.\mathbf{u})u^2}{2c_s^4}\right]. \tag{8.7}$$

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

The parameters that control incompressible flow are the Reynolds number and the geometric constraints of the problem. This can be deduced from the NS equations using characteristic length and velocity scales. Hence for an incompressible flow, we need to match dynamic (Reynolds number) and geometric similarities.

8.2.2 Incompressible Model

A modification for the equilibrium distribution function has been proposed by He and Luo to reduce the compressibility effect. They suggest that the density of the fluid can be replaced by

$$\rho = \rho_o + \delta\rho\,, \tag{8.8}$$

where ρ_o is the average fluid density, which is constant, and $\delta\rho$ is the density fluctuation. The modified equilibrium distribution function ends up as

$$f_k^{eq} = w_k \left[\rho + \rho_o \left(3\frac{\mathbf{c_k}.\mathbf{u}}{c_s^2} + \frac{9}{2}\frac{(\mathbf{c_k}.\mathbf{u})^2}{c_s^4} - \frac{3}{2}\left(\frac{\mathbf{u}}{c_s}\right)^2 \right) \right]. \tag{8.9}$$

The density ρ is the momentum density,

$$\rho = \sum_{k=0}^{n} f_k = \sum_{k=0}^{n} f_k^{eq}\,, \tag{8.10}$$

and

$$\rho_o \mathbf{u} = \sum_{k=0}^{n} c_k f_k = \sum_{k=0}^{n} c_k f_k^{eq}\,. \tag{8.11}$$

8.2.3 Mach and Reynolds Numbers

Error analysis of the LBM requires an extensive multiscale 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 of order Ma^2.

The fluid viscosity is related to the relaxation frequency by

$$\nu = \frac{\Delta x^2 c_s^2}{\Delta t}(\tau - 0.5)\,, \tag{8.12}$$

where $c_s = 1/\sqrt{3}$ for D2Q9.

The Reynolds number is $Re = UL/\nu$, where U and L are characteristic velocity and characteristic length in macroscale, respectively.

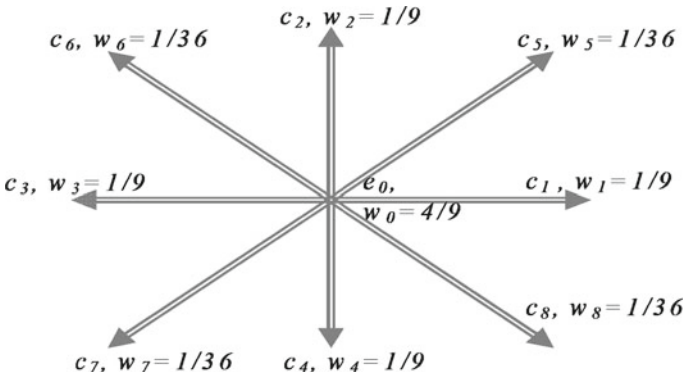


Fig. 8.1 Streaming directions

The above equation can be manipulated (by dividing both sides of the equation by UL), and the result is

$$Ma = \frac{\Delta x}{L\sqrt{3}}(\omega - 0.5)Re \tag{8.13}$$

The value of $L/\Delta x$ is the number of lattices in the direction of the characteristic length, say N. Setting Δx equal to unity is normal practice in LBM, $L = N$, and hence $Re = UN/\nu$ (let us call this Reynolds number the lattice Reynolds number). For an accurate solution, Ma should be kept small; therefore, ω or N should be chosen to ensure a low Ma value. In general, U should be kept at order 0.1, which is unrelated to the macroscopic velocity. In practical applications, the macroscopic Reynolds number should be equal to the LBM Reynolds number. Then U and ν can be arbitrarily selected within the range that ensures the stability of the solution. In the following sections, examples illustrate this issue more clearly.

Two-Dimensional with Nine Velocities, D2Q9

There is no need to elaborate on D2Q9 in detail, because it has already been discussed in the previous chapters. Hence D2Q9 applications for fluid flow will be presented in the following sections.

Figure 8.1 illustrates the streaming direction and values of w and c_k.

The weight coefficients are $w_o = 4/9$, $w_1 = w_2 = w_3 = w_4 = 1/9$ and $w_5 = w_6 = w_7 = w_8 = 1/36$.

The c_k are unit vectors along the lattice streaming directions,
$c_0 = (0, 0)$, the x- and y-components are zero,
$c_1 = (1, 0)$, the x-component is 1 and the y-component is 0,
$c_2 = (0, 1)$, the x-component is 0 and the y-component is 1,
$c_3 = (-1, 0)$, the x-component is -1 and the y-component is 0,
$c_4 = (0, -1)$, the x-component is 0 and the y-component is -1,
$c_5 = (1, 1)$, the x-component is 1 and the y-component is 1,

$c_6 = (-1, 1)$, the x-component is -1 and the y-component is 1,
$c_7 = (-1, -1)$, the x-component is -1 and the y-component is -1,
$c_8 = (1, -1)$, the x-component is 1 and the y-component is -1.

For example, if the x- and y-components of a vector $\mathbf{u}$ are 3 and -2 ($u = 3$, $v = -2$), respectively, then
$c_1.\mathbf{u} = (1, 0).(3, -2) = 3$,
$c_3.\mathbf{u} = (-1, 0).(3, -2) = -3$,
$c_5.\mathbf{u} = (1, 1).(3, -2) = 3 - 2 = 1$,
$c_6.\mathbf{u} = (-1, 1).(3, -2) = -3 - 2 = -5$, and so on.
And $\mathbf{u}^2 = \mathbf{u}.\mathbf{u} = (3, -2).(3, -2) = 9 + 4 = 13$.

Using Eq. 8.5, the values of

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

can be written for $k = 0$ to 8 as

$$\begin{aligned}
&1 - \frac{3}{2}(u^2 + v^2), && k = 0,\\
&1 + 3u + \frac{9}{2}u^2 - \frac{3}{2}(u^2 + v^2), && k = 1,\\
&1 + 3v + \frac{9}{2}v^2 - \frac{3}{2}(u^2 + v^2), && k = 2,\\
&1 - 3u + \frac{9}{2}u^2 - \frac{3}{2}(u^2 + v^2), && k = 3,\\
&1 - 3v + \frac{9}{2}v^2 - \frac{3}{2}(u^2 + v^2), && k = 4,\\
&1 + 3(u + v) + \frac{9}{2}(u + v)^2 - \frac{3}{2}(u^2 + v^2), && k = 5,\\
&1 + 3(-u + v) + \frac{9}{2}(-u + v)^2 - \frac{3}{2}(u^2 + v^2), && k = 6,\\
&1 + 3(-u - v) + \frac{9}{2}(u + v)^2 - \frac{3}{2}(u^2 + v^2), && k = 7,\\
&1 + 3(u - v) + \frac{9}{2}(u - v)^2 - \frac{3}{2}(u^2 + v^2), && k = 8\,.
\end{aligned}$$

Mass and Momentum Conservation

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

$$\rho = \sum_{k=0}^{8} f_k\,. \tag{8.14}$$

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

$$\rho \mathbf{u} = \sum_{\mathbf{k}=\mathbf{0}}^{8} f_k \mathbf{c}_\mathbf{k} \,, \tag{8.15}$$

or

$$\mathbf{u} = \frac{1}{\rho} \sum_{\mathbf{k}=\mathbf{0}}^{8} f_k \mathbf{c}_\mathbf{k} \,. \tag{8.16}$$

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

$$c_s = \frac{1}{\sqrt{3}} \,.$$

The frequency w of the relaxation parameter can be calculated from Eq. 8.12 for a given kinetic viscosity of the fluid.

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

8.3 Boundary Conditions

One of the important issues in LB simulation of flow is accurate modeling of the boundary conditions. Specifying the 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 have been suggested and tested. In the following paragraphs, different boundary conditions are explained. The mean idea is not to review different attempts, but to discuss, from our point of view, the simplest and most robust methods.

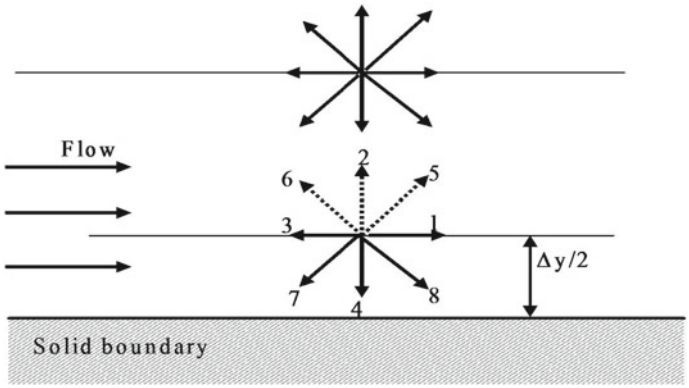


Fig. 8.2 Bounce-back scheme

8.3.1 Bounce-Back

Bounce-back is used to model solid stationary or moving boundary conditions, non-slip conditions, or flow over obstacles. For bounce-back at a boundary, the momentum needs to be adjusted,

$$f_k = f_{k'} + 2\frac{\rho w_i}{c_s^2}(\mathbf{u}_b.\mathbf{c}_k)\,, \tag{8.17}$$

where the direction of k' is the direction opposite to k, and u_b is the boundary velocity.

In the following sentences, the bounce-back method of stationary boundaries will be discussed. The method is quite simple and mainly implies that an incoming particle toward the solid boundary is bounced back into the flow domain. In the literature, a few versions of the 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. 8.2. The distribution functions f_4, f_7, and f_8 are known from the streaming process. It is assumed that when these known distribution functions hit the wall, they bounce back to the solution domain. Therefore, $f_5 = f_7$, $f_2 = f_4$, and $f_6 = f_8$. Note that f_7 at node say (i, j) is equal to f_7 of node $(i + 1, j + 1)$ after streaming, and a similar argument is valid for f_8, which is coming from node $(i - 1, j + 1)$. Hence the boundary condition must be applied after the streaming process.

Bounce-back ensures conservation of mass and momentum at the boundary. Another possible arrangement is shown in Fig. 8.3, with the wall midway between the two lattice sites. However, the domain is extended inside the solid wall. Bounce-back is taking place inside the wall.

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

Notice that f_7 is bouncing back from the left-hand lattice in the solid wall, and f_8 is bouncing back from the 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.

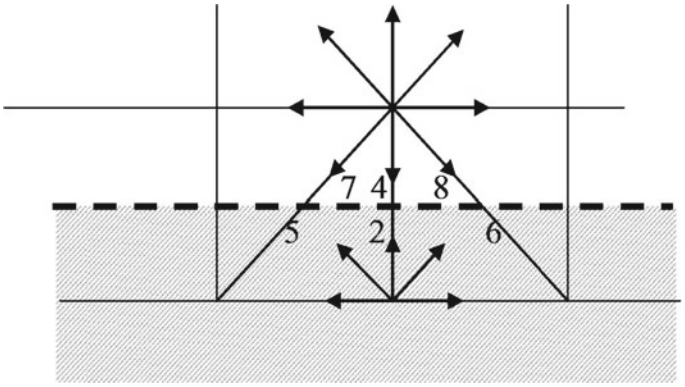


Fig. 8.3 Bounce-back scheme II

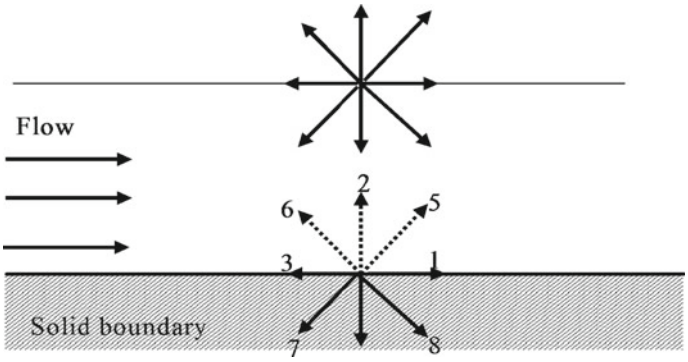


Fig. 8.4 Simple bounce-back scheme

Another scheme, the simple bounce-back scheme (III), is shown in Fig. 8.4; the lattices are located directly at the solid surface and not at the middle plane, as shown in Fig. 8.4.

Here we 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 the 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 schemes I and II are not that difficult, and the reader may modify the code for these kinds of bounce-back schemes.

Bounce-back can be applied to all lattices on solid surfaces in modeling flow over an obstacle. For instance, flow in a porous medium, Fig. 8.5, can be simulated by discrete solids embedded in a fluid flow. Bounce-back should be used for nodes on solid surfaces, with the velocity in and on the solid set to zero.

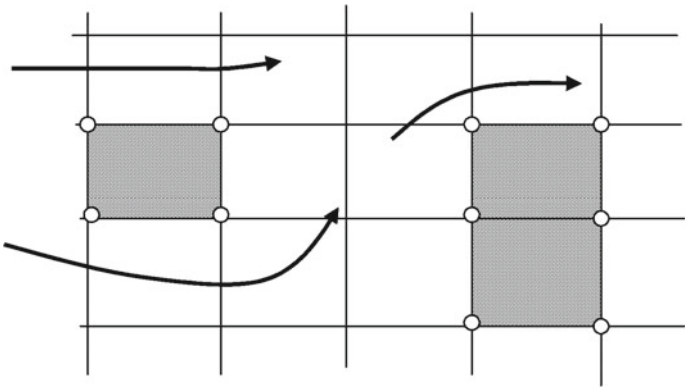


Fig. 8.5 Flow through a porous medium, discrete model

8.3.2 Boundary Condition with Known Velocity

It is very common in practical applications that we know the velocity component at the boundaries, for example, the inlet velocity for a channel flow. Zhu and He described a method to calculate three unknown distribution functions based on Eqs. (8.14) and (8.15) with equilibrium conditions assumed to be normal to the boundary.

Equation 8.14 can be written as

$$\rho = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8 \,. \tag{8.18}$$

Equation 5.15 can be written for the x-component as

$$\rho u = f_1 + f_5 + f_8 - f_6 - f_3 - f_7 \,, \tag{8.19}$$

and for the y-component as

$$\rho \mathrm{v} = f_5 + f_2 + f_6 - f_7 - f_4 - f_8 \,. \tag{8.20}$$

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

Let us illustrate the above concept with an example. Figure 8.6 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.

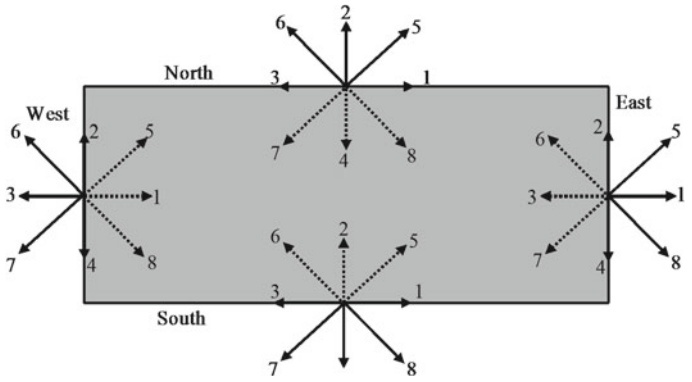


Fig. 8.6 Distribution functions at the boundaries of a domain

West Boundary

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

Then Eq. 8.14 can be written as

$$\rho_w = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8 \,. \tag{8.21}$$

Equation 8.15 can be written as

$$\rho_w u_w = f_1 + f_5 + f_8 - f_6 - f_3 - f_7 \,, \tag{8.22}$$

and Eq. 8.15 can be written as

$$\rho_w \mathrm{v_w} = f_5 + f_2 + f_6 - f_7 - f_4 - f_8 \,. \tag{8.23}$$

The equilibrium condition normal to the boundary yields

$$f_1 - f_1^{eq} = f_3 - f_3^{eq} \,, \tag{8.24}$$

where f^{eq} can be calculated from Eq. 8.9,

$$f_1^{eq} = \frac{1}{9}\rho_w \left[1 + 3u_w + \frac{9}{2}u_w^2 - \frac{3}{2}(u_w^2 + \mathrm{v_w^2})\right] \,, \tag{8.25}$$

and

$$f_3^{eq} = \frac{1}{9}\rho_w \left[1 - 3u_w + \frac{9}{2}u_w^2 - \frac{3}{2}(u_w^2 + \mathrm{v_w^2})\right] \,. \tag{8.26}$$

Substituting the above equations into Eq. (8.24) yields

$$f_1 = f_3 + \frac{2}{3}\rho_w u_w \,. \tag{8.27}$$

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

Solving Eqs. (8.23–8.27) for the four unknowns ρ_w, f_1, f_5, and f_8 yields the following equations:

$$\rho_w = \frac{1}{1-u_w}\left[f_0 + f_2 + f_4 + 2(f_3 + f_6 + f_7)\right] , \tag{8.28}$$

$$f_5 = f_7 - \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_w u_w + \frac{1}{2}\rho_w \mathrm{v_w} \,, \tag{8.29}$$

$$f_8 = f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_w u_w - \frac{1}{2}\rho_w \mathrm{v_w} \,. \tag{8.30}$$

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

East Boundary:

$$\rho_E = \frac{1}{1+u_E}\left[f_0 + f_2 + f_4 + 2(f_1 + f_5 + f_8)\right] , \tag{8.31}$$

$$f_3 = f_1 - \frac{2}{3}\rho_E u_E \,, \tag{8.32}$$

$$f_7 = f_5 + \frac{1}{2}(f_2 - f_4) - \frac{1}{6}\rho_E u_E - \frac{1}{2}\rho_E \mathrm{v_E} \tag{8.33}$$

$$f_6 = f_8 - \frac{1}{2}(f_2 - f_4) - \frac{1}{6}\rho_E u_E + \frac{1}{2}\rho_E \mathrm{v_E} \,. \tag{8.34}$$

North Boundary:

$$\rho_N = \frac{1}{1+\mathrm{v}_N}\left[f_0 + f_1 + f_3 + 2(f_2 + f_6 + f_5)\right] , \tag{8.35}$$

$$f_4 = f_2 - \frac{2}{3}\rho_N v_N \,, \tag{8.36}$$

$$f_7 = f_5 + \frac{1}{2}(f_1 - f_3) - \frac{1}{6}\rho_N v_N - \frac{1}{2}\rho_N u_N \,, \tag{8.37}$$

$$f_8 = f_6 + \frac{1}{2}(f_3 - f_1) + \frac{1}{2}\rho_N u_N - \frac{1}{6}\rho_N \mathrm{v}_N \,. \tag{8.38}$$

South Boundary:

$$\rho_S = \frac{1}{1 - \mathrm{v}_S}\left[f_0 + f_1 + f_3 + 2(f_4 + f_7 + f_8)\right] \,, \tag{8.39}$$

$$f_2 = f_4 + \frac{2}{3}\rho_S v_S \,, \tag{8.40}$$

$$f_5 = f_7 - \frac{1}{2}(f_1 - f_3) + \frac{1}{6}\rho_S v_S + \frac{1}{2}\rho_S u_S \,, \tag{8.41}$$

$$f_6 = f_8 + \frac{1}{2}(f_1 - f_3) + \frac{1}{6}\rho_S v_S - \frac{1}{2}\rho_S u_S \,. \tag{8.42}$$

8.3.3 Equilibrium and Nonequilibrium Distribution Functions

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

$$f = f^{eq} + f^{neq} \tag{8.43}$$

For nodes on the boundary, see Fig. 8.7, the above equation can be written as

$$f_w = f_f + (f_w^{eq} - f_f^{eq}) \,. \tag{8.44}$$

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

$$f_1 = f_3 + (f_1^{eq} - f_3^{eq}) \tag{8.45}$$

which equal

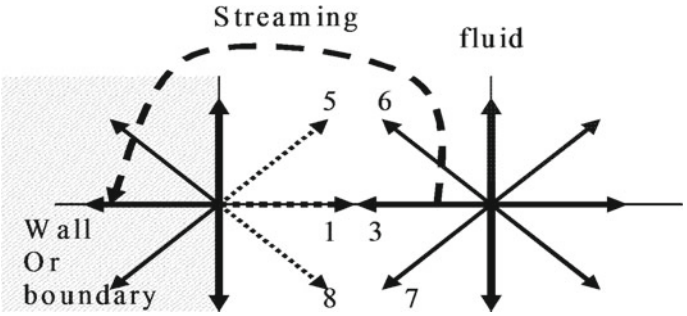


Fig. 8.7 A node on the boundary and another node adjacent to the boundary node

$$f_1 = f_3 + \frac{2}{3}\rho_w u_w, \tag{8.46}$$

and similarly for f_5 and f_8,

$$f_5 = f_7 + (f_5^{eq} - f_7^{eq}) \tag{8.47}$$

and

$$f_8 = f_6 + (f_8^{eq} - f_6^{eq})\,. \tag{8.48}$$

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

8.3.4 Open Boundary Condition

In some problems, the outlet velocity is unknown. The normal practice is to use extrapolation for the unknown distribution functions. For instance, if the east boundary condition (right-hand side) of Fig. 8.6 represents the outlet condition, then f_3, f_6, and f_7 need to be calculated at the east boundary, $i = n$. A second-order polynomial can be used as

$$f_{3,n} = 2.f_{3,n-1} - f_{3,n-2}\,, \tag{8.49}$$

$$f_{6,n} = 2.f_{6,n-1} - f_{6,n-2}\,, \tag{8.50}$$

and

$$f_{7,n} = 2.f_{7,n-1} - f_{7,n-2}. \tag{8.51}$$

However, in some cases the above boundary conditions lead to an unstable solution. Using first-order interpolation showed greater stability than the second-order

interpolation scheme, as

$$f_{3,n} = f_{3,n-1}; \quad f_{6,n} = f_{6,n1} \quad \text{and} \quad f_{7,n} = f_{7,n-1}\,. \tag{8.52}$$

Another approach is to assume that the pressure at the boundary is known, i.e., density ρ. Hence the density at the boundary needs 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 specific the missing distribution functions:

$$u_x = -1 + \frac{f_0 + f_2 + f_4 + 2\,(f_1 + f_5 + f_8)}{\rho_{outlet}}\,, \tag{8.53}$$

$$f_3 = f_1 - \frac{2}{3}\rho_{outlet} u_x\,, \tag{8.54}$$

$$f_7 = f_5 + \frac{1}{2}(f_2 - f_4) - \frac{1}{6}\rho_{outlet} u_x\,, \tag{8.55}$$

$$f_6 = f_8 - \frac{1}{2}(f_2 - f_4) - \frac{1}{6}\rho_{outlet} u_x\,. \tag{8.56}$$

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

8.3.5 *Periodic Boundary Condition*

It becomes necessary to apply a periodic boundary condition to isolate repeating flow conditions, for instance flow over a bank of tubes as shown in Fig. 8.8. 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 along these boundaries. The distribution functions that are leaving line a are the same as the distribution functions entering from line b and conversely.

The distributions functions f_4, f_7, and f_8 are unknowns on the line a, and f_2, f_5, and f_6 are unknowns 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 $f_{6,b} = f_{6,a}$.

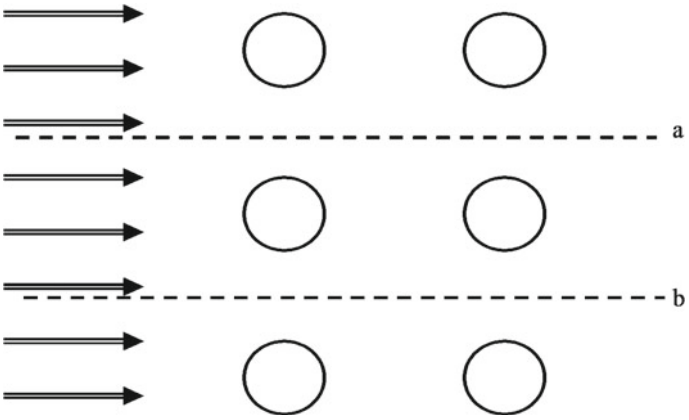


Fig. 8.8 Illustration of the periodic boundary conditions

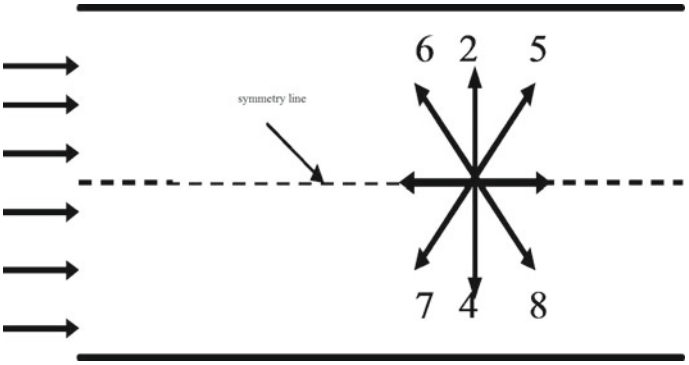


Fig. 8.9 Illustration of symmetry boundary conditions

8.3.6 Symmetry Condition

Many practical problems exhibit 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, in flow in a channel as depicted in Fig. 8.9, the flow above the symmetry line is the mirror image of the flow below the symmetry line. Therefore, integration should be carried out only for one part of the domain, and the symmetry condition should 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 $f_6 = f_8$.

8.4 Stream Function

For two-dimensional problems, it is more illustrative to use a stream function to present the predicted results. A stream function is defined as (consult any fluid mechanics textbook)

$$u = \frac{\partial \psi}{\partial y} \tag{8.57}$$

and

$$v = -\frac{\partial \psi}{\partial x} \tag{8.58}$$

where ψ, u, and v are a stream function and the x-component and y-component velocities, respectively. The value stream function can be set to zero along any impermeable boundary. One of the aobve-mentioned equations can be integrated by considering the average value of the velocity; see the codes in the appendix or any textbook on fluid dynamics or computational fluid dynamics.

8.5 High Reynolds Number and Turbulence

For a high Reynolds number (low Prandtl number for energy transfer), the kinematic viscosity usually has small values, which causes stability problems. In order to overcome such a problem, an additional term can be added to the physical viscosity, similar to what we did in $k - \epsilon$ turbulent modeling (turbulent viscosity or turbulent Prandtl number). Hence

$$\Omega = 1/\tau = \frac{1}{3\nu + 0.5 + A}\,. \tag{8.59}$$

The parameter A can be selected in the range of 0.2–0.4. However, in order to compensate for the effect of A, an extra force term needs to be added to the LB equation,

$$F_i = 3w_i(1 - \frac{\omega_f}{2})\boldsymbol{F}.\boldsymbol{u} + 4.5\frac{w_i\omega_f^2 A}{1 - \omega_f A}\boldsymbol{Q}_i : \sum_{j=0}^{8}(f_j - f_j^{eq})\boldsymbol{c}_j\boldsymbol{c}_j\,. \tag{8.60}$$

The explicit version of the above equation is

$$\Pi^{neq} = \begin{bmatrix} \sum_{j=0}^{8}(f_j - f_j^{eq})c_{jx}c_{jx} & \sum_{j=0}^{8}(f_j - f_j^{eq})c_{jx}c_{jy} \\ \sum_{j=0}^{8}(f_j - f_j^{eq})c_{jx}c_{jy} & \sum_{j=0}^{8}(f_j - f_j^{eq})c_{jy}c_{jy} \end{bmatrix}, \tag{8.61}$$

$$Q_i = \begin{bmatrix} 1 - c_{ix}^2 & -c_{ix}c_{iy} \\ -c_{ix}c_{iy} & 1 - c_{iy}^2 \end{bmatrix}, \tag{8.62}$$

$$F_i = 3w_i(1 - \frac{\omega_f}{2})(F_x u_x + F_y u_y) + 4.5\frac{w_i\omega_f^2 A}{1 - \omega_f A}(\Pi_{1,1}^{neq} Q_{i,1,1} + \Pi_{1,2}^{neq} Q_{i,1,2} + \Pi_{2,1}^{neq} Q_{i,2,1} + \Pi_{2,2}^{neq} Q_{i,2,2}). \tag{8.63}$$

For more details about this method, we refer to the paper by Yang, Shi, and Chai entitled "Generalized modification in the lattice Bhatngar–Gross–Krook model for incompressible Navier–Stokes equations and convection–diffusion equations," *Physical Review E* 90:1 (2014), 013309.

Turbulent Flow: The three-dimensional LBM can be used as a direct numerical simulation (DNS). However, the method needs an extensive lattice, which is not economical. The subgrid scale developed by Smagorinsky in 1963 can be economically used to simulate turbulent flow. The method is to add the eddy viscosity to the physical viscosity, and hence

$$\tau = 3(\nu + \nu_T + 0.5), \tag{8.64}$$

where ν_T is the eddy viscosity,

$$\nu_T = (C_s\Delta)^2\sqrt{2\sum_{i,j} S_{ij}S_{ij}}, \tag{8.65}$$

where S_{ij} is the strain-rate tensor,

$$S_{ij} = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}), \tag{8.66}$$

C_s is the Smagorinsky coefficient (0.1–0.4), Δ is the grid scale. Hence for modeling turbulent flow using LES, one must calculate τ based on Eq. (8.64) at each time step, because τ is no longer a constant.

8.6 Flow in Porous Media

In general, there are two approaches to modeling flow through porous media: detail microscopic and volume average. In the microscopic approach, a porous medium is modeled as solid particles randomly or regularly placed in a fluid. A bounce-back scheme is usually used on the surface of the solid particles. In the volume average method, the fluid and solid are considered a continuum medium, and the momentum balance equation is applied by considering extra source terms to account for the presence of a solid medium. The modified Navier–Stokes equation is given as follows:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}.\Delta)(\mathbf{u}/\epsilon) = -\frac{1}{\rho}\Delta(\epsilon p) + \nu_e \Delta^2 \mathbf{u} + \mathbf{F}\,, \tag{8.67}$$

where ρ, ϵ, and ν_e are fluid density, porosity, and effective viscosity, respectively. The effect of the solid is represented by the force $\mathbf{F}$,

$$\mathbf{F} = -\frac{\epsilon\nu}{K}\mathbf{u} - \frac{\epsilon F_e}{\sqrt{K}}|\mathbf{u}|\mathbf{u} + \epsilon\mathbf{G}\,, \tag{8.68}$$

where ν is the fluid viscosity. Note that fluid viscosity may be different from effective viscosity. However, in most analyses, we assume that they have the same value. Here $\mathbf{G}$ stands for body force, such as a buoyancy force. The geometric F_e and permeability K are functions of porosity,

$$F_e = \frac{1.75}{\sqrt{150\epsilon^2}} \tag{8.69}$$

and

$$K = \frac{\epsilon^3 d_p^2}{150(1-\epsilon)^2}\,, \tag{8.70}$$

where d_p is the effective solid particle diameter.

The problem is usually solved in nondimensional form, and hence Eq. 8.67 can be written as

$$\frac{\partial \mathbf{U}}{\partial t^*} + (\mathbf{U}.\Delta)(\mathbf{U}/\epsilon) = -\Delta(p^*) + \frac{1}{Re}\Delta^2 \mathbf{u} + \mathbf{F}^*\,, \tag{8.71}$$

where

$$F^* = -\frac{\mathbf{U}}{Da} - \frac{\mathbf{U}.\mathbf{U}}{Re\sqrt{Da}} + G^* \tag{8.72}$$

where the superscript * represents a nondimensional term, and Re and Da are the Reynolds number and Darcy number, respectively.

Hence standard LBM can be used with a source term.

8.7 Computer Coding

The coding is quite simple and similar to the codes developed in Chap. 7 for D2Q9, except that extra terms should be added to f^{eq} as represented in Eq. (8.9).

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
==========================================================
```

8.8 Examples

8.8.1 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 1.2×10^{-3} m^2/s). The lid is set to motion with speed of 6.0 m/s.

In isothermal fluid flow, it is important to match Reynolds number and geometrical aspect ratio.

For the above problem, Re = U_{lid}* H/ ν = 6.0*0.2/0.00012 = 1000.

In LBM, we are free to use any values for U_{lid} and viscosity provided that $Re = 1000$. In order to reduce the compressibility effect of LBM, set $U_{lattice}$ to 0.1 and viscosity to 0.01, then we need 100 lattices in the H-direction, because $Re = 1000 = U_{lattice}, N/\nu_{lattice} = 0.1 * N/0.01$, and then $N = 100$. Since the cavity is square, i.e., the aspect ratio is unity, 100 nodes need to be adopted in the y-direction too.

The computer code is given in the appendix. Two versions of the code for the flow in a lid-driven cavity are given in the appendix. One code does not call any external functions (subroutines in FORTRAN). The second version is the main code with a few functions. The advantage of the second version is that when the code becomes long, the debugging process becomes difficult. Moreover, using the same code for different problems becomes much easier. In most cases, boundary conditions need to be changed in the boundary function (or boundary subroutine) and so on. The collision process, streaming, evaluating density, velocity components, and output data are the same for most of problems. Then there is no need to repeat them for each problem; just call the function in Matlab.

```
==========================================================
```

The results of simulations are shown in Figs. 8.10 and 8.11 for streamlines and velocity at different sections along the cavity, which compare will with FVM results.

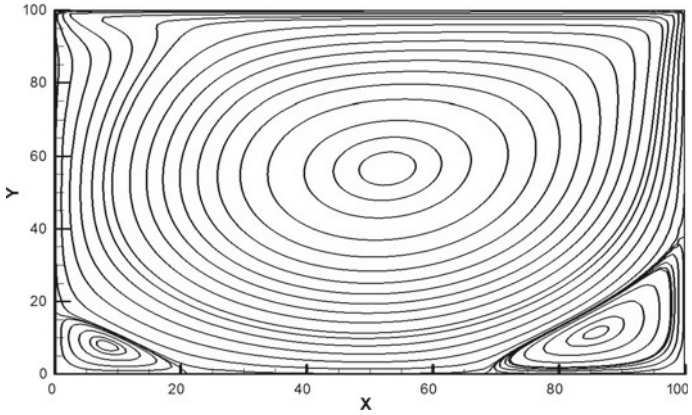


Fig. 8.10 Streamlines for lid-driven cavity, $Re = 1000$

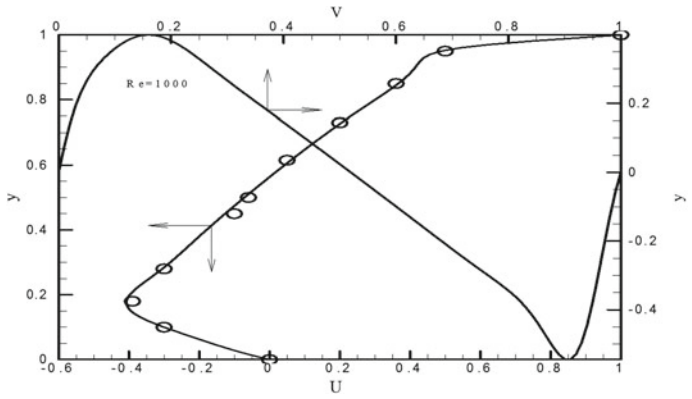


Fig. 8.11 Velocity profiles at the midsection and at the mid-height of the cavity

Figure 8.11 U- and V-velocities at $X = 0.5$ and $Y = 0.5$ of the height of the cavity, respectively, $Re = 1000$.

The results are quite good compared with benchmark solutions. The symbols in Fig. 8.11 denote the benchmark solution obtained by FVM.

8.8.2 Developing Flow in a Two-Dimensional Channel

Water flows between two parallel plates (the channel in Fig. 8.12) 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.

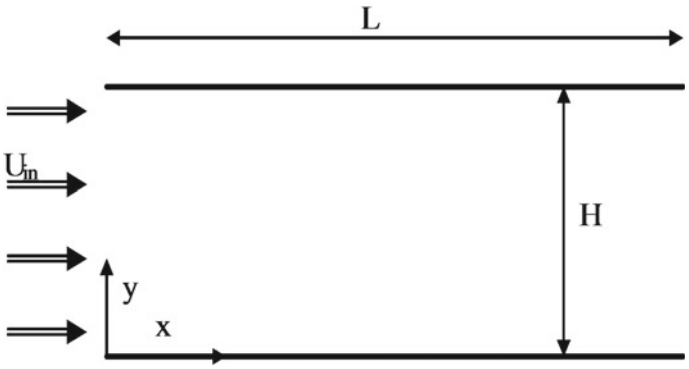


Fig. 8.12 Flow in a channel

The density and kinematic viscosities of water are 1000 kg/m^3 and 10^{-6} m^2/s, respectively.

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

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

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

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

Let us use $U_{lattice} = 0.2$, and 40 lattices in the y-direction; then the lattice viscosity should be calculated to keep $Re = 400$, $Re = U_{lattice}N/\nu_{lattice} = 0.1x80/v_{lattice}$, and then $v_{lattice} = 0.02$. The aspect ratio is $25 = M/N$, since $N = 40$, and therefore we need to use 1000 lattices in the x-direction.

Figure 8.13 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 evident 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 times the average velocity, $u/u_{in} = 1.5(1 - y/H)$.

The code for a lid-driven cavity needs to be slightly modified.

The Computer Code for Flow in a Channel

The code is the same as for a lid-driven cavity, except the boundary condition function and results function need to be modified. In the main code, the call should be as [f] = boundary (nx, ny, f, uo, rho);

It was not necessary to pass ρ to the function boundary for a lid-driven cavity. However, it is done for generality. Also, the output results (result function) need to be modified for plotting the u-velocity. In the main code, nx and ny should be set to different values to reflect the aspect ratio of the channel. However, all these changes are done in the appendix.

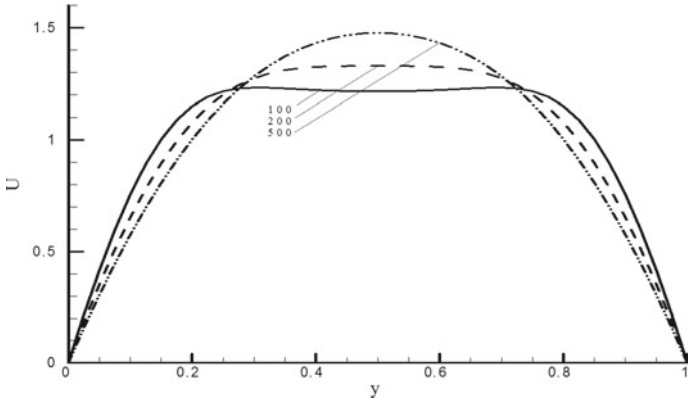


Fig. 8.13 Velocity profiles at different cross sections, $x = 100$, 200, and 300

8.8.3 Flow over Obstacles

Flow over obstacles is very common in engineering and geophysical applications, such as flow through a porous medium, macroscopic analysis. 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 previous 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). Another external function is added to the main code, called obstc. The other functions are the same. However, for decoration, it is a good idea to set the u-velocity and v-velocity for the solid block in the result function.

I encourage the reader to examine the code in the appendix.

Figure 8.14 shows the streamlines with main recirculation region on the shadow of the step. The streamlines are produced using Tecplot software.

Figure 8.15 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.

Example: try to modify the code to handle an L-shaped obstacle in a duct. The L-shape is located at $x = 100$ and $y = 20$ extended to $x = 140$ and $y = 30$.

Figure 8.16 shows streamlines of the problem. Note that the resolution is not very good, because the number of lattices used is insufficient for good resolution.

Flow over a Square Bluff Body

This an interesting example to illustrate von Karmann vortex formation downstream of a bluff body. The bluff body is placed at the center of a duct. The duct has nx =

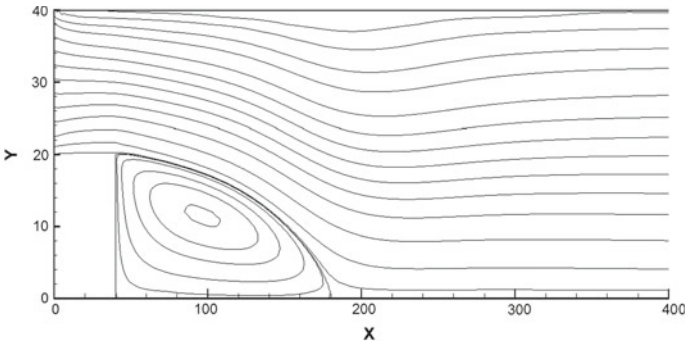


Fig. 8.14 Streamlines in a channel with back-facing step, $Re = 400$

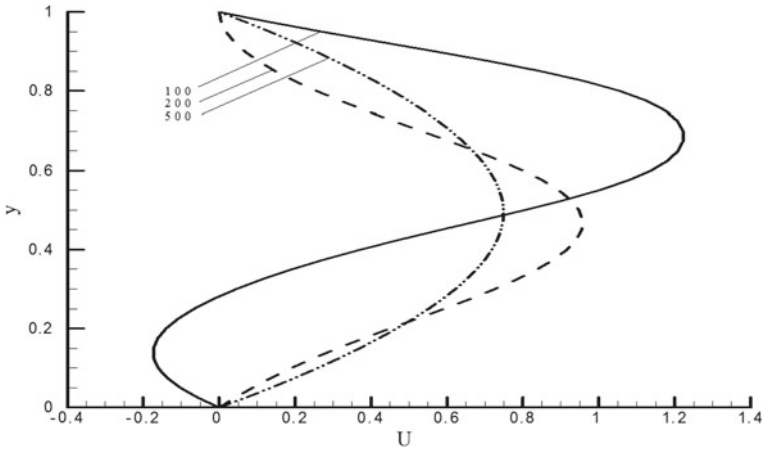


Fig. 8.15 Velocity profiles at different cross sections, $Re = 400$

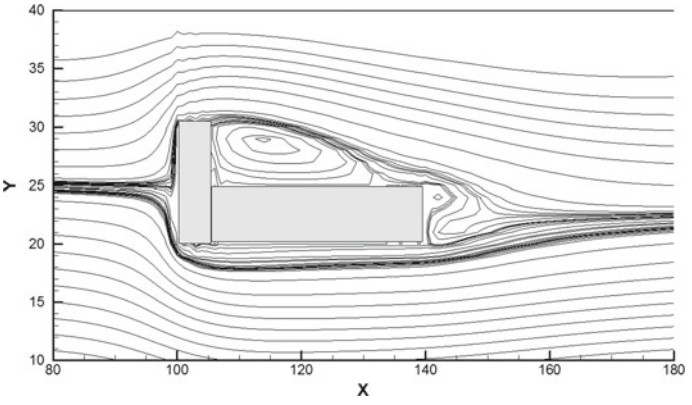


Fig. 8.16 Streamlines in a channel with an L-shaped obstacle

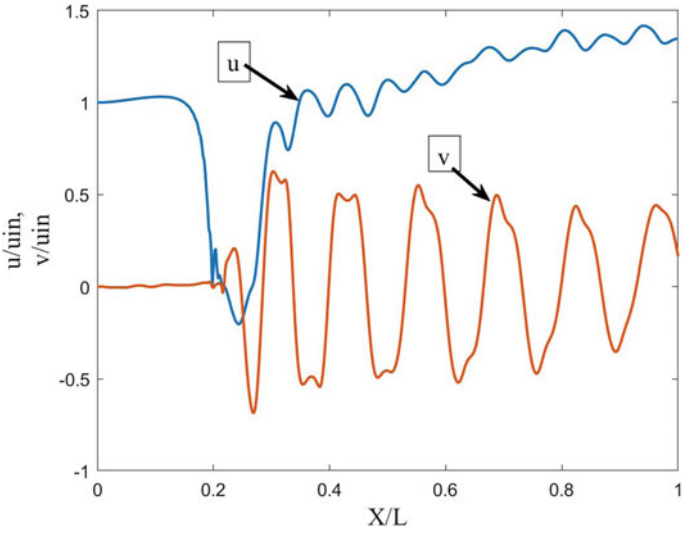


Fig. 8.17 U- and V-velocities at the middle of a channel with square bluff body

500 and ny = 80 lattice, aspect ratio L/H = 6.25. The number of lattices used for the bluff body is 10 × 10. The ilet velocity is set to 0.1, and viscosity is set to 0.01; the Reynolds number is based on the bluff body width, $Re = \frac{0.1*10}{100} = 100$. Figure 8.17 shows the u- and v-velocities at the middle of the duct as a function of the x-axis. The above-mentioned examples clearly show the ability of LBM to model complex flow conditions.

8.9 Vorticity and Stream Function Approach

Another approach to solving two-dimensional fluid flow problems is to use the vorticity–stream function approach (VSF). The pressure gradient term in the Navier–Stokes equation can be eliminated by subtracting the cross differentiation of the x- and y-momentum equations from each other. The nondimensional governing equations can be written as

$$\frac{\partial \Omega}{\partial t} + \frac{\partial}{\partial x}\left(u\frac{\partial \Omega}{\partial x}\right) + \frac{\partial}{\partial y}\left(v\frac{\partial \Omega}{\partial y}\right) = \frac{1}{Re}\left(\frac{\partial^2 \Omega}{\partial x^2} + \frac{\partial^2 \Omega}{\partial y^2}\right) \tag{8.73}$$

and

$$\frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \Psi}{\partial y^2} = -\Omega\,, \tag{8.74}$$

where Ω and Ψ are vorticity and stream functions, respectively. Equation 8.73 is the advection–diffusion equation, which can be solved using methods introduced in Chapter 4, for instance, using D2Q4 or D2Q5. Equation 8.74 is the Laplace equation with source term, which can be solved either by finite differences or by the method introduced in Chapter 3 using D2Q4 or D2Q5 with different distribution functions. Hence there is no need to elaborate on the solution techniques. Moreover, axisymmetric fluid flow problems (flow in a pipe, for example) can by solved with the above-mentioned method with an extra source term in the above equations. This can be an exercise for the reader.

8.10 Hexagonal Grid

In the previous sections, attention was given to rectangular lattice arrangements. However, it is possible to use hexagonal lattice arrangements for the previously mentioned problems. Figure 8.18 shows a 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

$$f_i^{eq} = w_i \rho (1 + 4\, c_i.u_i + 8(c_i.u_i)^2 - 2u_i.u_i)\,. \tag{8.75}$$

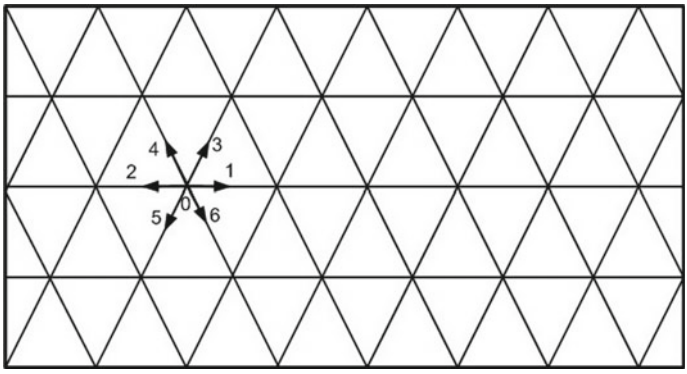


Fig. 8.18 Hexagonal lattice arrangement

8.11 Problems

1. Flow through an expanding channel, channels with different cross sections connected together. Suppose the Reynolds number based on entrance height is 500. The channel is open to the atmosphere from the right-hand side. Determine the streamlines and velocity profiles at different cross sections. Select the number of lattices with inlet velocity 0.1 and viscosity 0.01 (in lattice units), for a length to height ratio of 5 for the first section, and the second section having the same length as the first section. Compare LBM predicted results with predictions of different methods, such as finite difference, finite volume, or finite element methods.

2. A confined water jet is impinging on a plate as shown Fig. 8.19. For the Reynolds number based on the 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 times the jet width. Use the LES turbulent model.

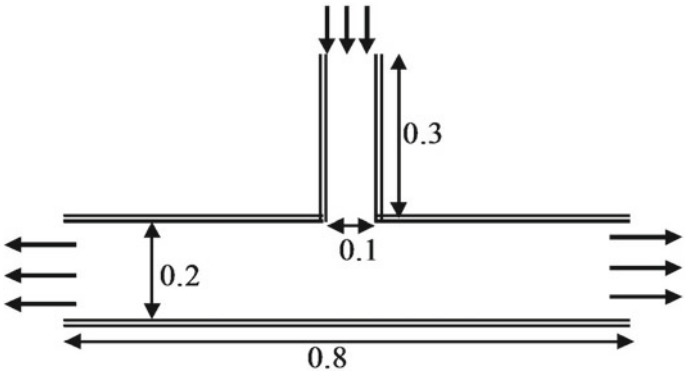


Fig. 8.19 Confined jet between two parallel plates

Chapter 9
Nonisothermal Incompressible Fluid Flow

In Chap. 8, isothermal fluid flow problems were discussed. In many applications, heat and/or mass transfer is associated with flows and convection. In this chapter, two kinds of nonisothermal flows will be discussed, namely forced and natural convection. In forced convection, the energy equation can be solved after one has obtained 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.

9.1 Navier–Stokes and Energy Equations

For two-dimensional incompressible flow, the conservative form for the Naiver–Stokes (NS) equation can be written in Cartesian coordinates, with body force as follows:

x-momentum:

$$\frac{\partial \rho u}{\partial t} + \frac{\partial (\rho u u)}{\partial x} + \frac{\partial (\rho \mathrm{v} u)}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu \frac{\partial u}{\partial y}\right) + F_x \tag{9.1}$$

y-momentum:

$$\frac{\partial \rho \mathrm{v}}{\partial t} + \frac{\partial (\rho u \mathrm{v})}{\partial x} + \frac{\partial (\rho \mathrm{v} \mathrm{v})}{\partial \mathrm{y}} = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left(\mu \frac{\partial \mathrm{v}}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu \frac{\partial \mathrm{v}}{\partial y}\right) + F_y \tag{9.2}$$

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

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3_9

The energy equation can be written as

$$\frac{\partial \rho cT}{\partial t} + \frac{\partial (\rho cuT)}{\partial x} + \frac{\partial (\rho cvT)}{\partial y} = \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + q''', \qquad (9.3)$$

where q''' represents the 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 the Reynolds (Re) and Prandtl (Pr) numbers and the geometric aspect ratio. Hence in LBM simulation, we need to match those parameters. For natural convection (with Boussinesq approximations), the controlling parameters are the Grashof (Gr) or Rayleigh number (Ra), Pr, and the geometric aspect ratio. For more details on the physics and mathematical modeling of convection, textbooks on heat transfer may be consulted.

9.2 Forced Convection, D2Q9

Since there is no coupling between momentum and the scalar equation (energy and/or mass transfer equation), the method used in Chaps. 7 and 8 can be combined. 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:

$$f_k(x + \Delta x, t + \Delta t) = f_k(x, t)\,[1 - \omega_m] + \omega_m\, f_k^{eq}(x, t)\,. \qquad (9.4)$$

The methodology has been explained in the previous chapters; hence there is no need for repetition.

For scalar functions (temperature, species concentration), the following equation needs to solved:

$$g_k(x + \Delta x, t + \Delta t) = g_k(x, t)\,[1 - \omega_s] + \omega_s\, g_k^{eq}(x, t)\,. \qquad (9.5)$$

The equilibrium distribution function is as in the advection problem (Eq. 9.5),

$$g_k^{eq} = w_k \phi(x, t)\left[1 + \frac{c_k \bullet \vec{u}}{c_s^2}\right]. \qquad (9.6)$$

Observe that ω is different for the momentum and scalar equations.

For momentum, we have

$$\omega_m = \frac{1}{3.\nu + 0.5}\,, \tag{9.7}$$

where v is the kinematic viscosity, and for the scalar equation, we have

$$\omega_s = \frac{1}{3.\alpha + 0.5}\,, \tag{9.8}$$

where α is the diffusion coefficient (thermal diffusion or mass diffusion coefficient). Different boundary conditions for the scalar equation were discussed in the previous chapters.

To illustrate the forced convection problem, let us work a few examples. A heated lid-driven cavity and flow through a heated channel will be worked out with full computer code (MATLAB code in the appendix).

9.3 Heated Lid-Driven Cavity

Consider a heated lid-driven cavity in the same problem as in Chap. 8, except that the lid is kept at a high temperature compared with the other boundaries. An enclosure is filled with air of viscosity $v = 0.01$ (lattice units), and the lid moves with velocity $U_{lid} = 0.1$. For a Reynolds number of 1000, we need to use 100×100 lattices in the x- and y-directions (nx = 101, ny = 101). We have $Pr = v/\alpha = 0.01/\alpha = 0.71$ (air), and hence $\alpha = 0.014$. The lid is heated to a normalized temperature $(\mathrm{T\text{-}T}_{coldwall}/(\mathrm{T}_{wall}\text{–}\mathrm{T}_{coldwall})$ of 1.0, and the east and west walls are kept at a cold temperature normalized to 0.0, while the bottom of the cavity is thermally isolated. The computer code is given in the appendix.

It is worth mentioning how the boundary conditions are set for the energy equation. The temperature T_w of the lid (upper boundary) is given. The unknown distribution functions at the upper boundary are f_4, f_7, and f_8. The flux balance at the boundary should be ensured; hence

$$f_4^{neq} + f_2^{neq} = 0\,, \tag{9.9}$$

i.e.,

$$f_4^{eq} - f_4 + f_2^{eq} - f_2 = 0. \tag{9.10}$$

Then

$$f_4 = f_4^{eq} + f_2^{eq} - f_2\,, \tag{9.11}$$

which is

$$f_4 = w_4 T_w + w_2 T_w - f_2, \tag{9.12}$$

and similarly for f_7 and f_8,

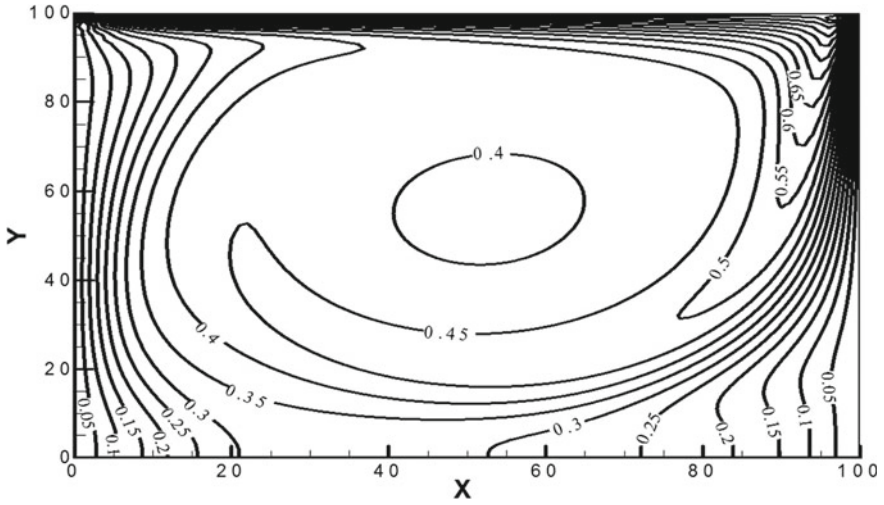


Fig. 9.1 Isotherms for the heated lid-driven cavity, Re = 1000, Pr = 0.71

$$f_7 = w_7 T_w + w_5 T_w - f_5; \quad f_8 = w_8 T_w + w_6 T_w - f_6. \tag{9.13}$$

For adiabatic boundaries, the flux is zero. There are two approaches; one may either set the values of the distribution functions to the values of the adjacent lattice sites or use bounce-back boundary conditions. For example, the bottom of the cavity is adiabatic, and hence

$$f_2 = f_4; \quad f_5 = f_7; \quad f_6 = f_8. \tag{9.14}$$

The isotherms are shown in Fig. 9.1.

The temperature profile along the height of the cavity at three cross sections, first, middle, and third quarter planes (x = 25, 50, and 75) are given in Fig. 9.2.

9.4 Forced Convection Through a Heated Channel

A cold fluid flows through the heated walls of a channel. The walls of the channel are kept at constant temperature. Water flows through a channel of height 6.0 cm and length 120 cm. 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.

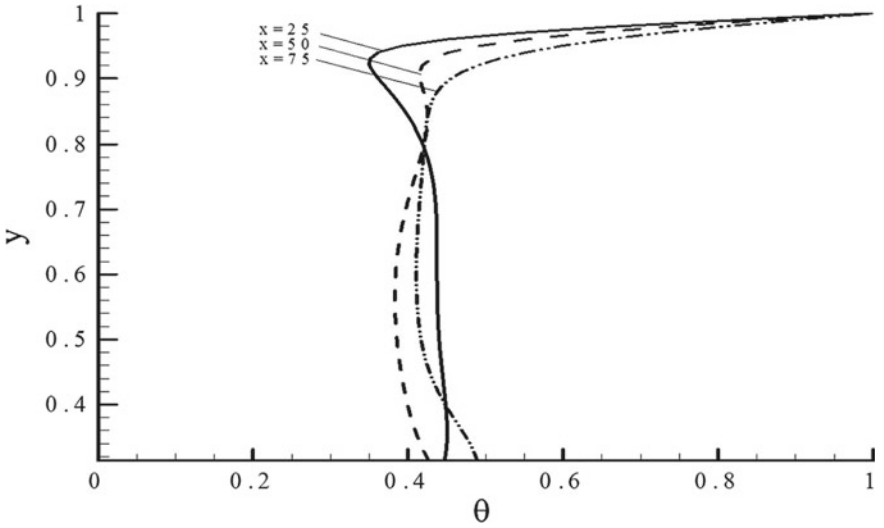


Fig. 9.2 Temperature profiles along the height of the cavity at different cross sections

Solution:

$T_{in} = 20\,°C$ $U_{in} = 0.001$ m/s,

$T_w = 80\,°C$,

L = 1.20 m,

H = 0.06 m.

The viscosity and thermal diffusivity of the water at mean temperature (50 °C) are about 6.0×10^{-7} m^2/s and 1.58×10^{-7} m^2/s, respectively.

The Reynolds number is given by U_{in} H/ν=0.001 ∗0.06/(6.0×10^{-7}) = 100.

The Prandtl number Pr = $\nu/\alpha = 6.0 \times 10^{-7}/1.58 \times 10^{-7}$ is about 3.8.

Let us nondimensionalize the temperature and introduce $\theta = (T - T_{in})/(T_w - T_{in})$. And let use H for the length scale.

At x = 0, we have $\theta_{in} = 0$, while at y = 0 and y/H = 1.0, we have $\theta_w = 1.0$. At x = L, we have $\dfrac{\partial \theta}{\partial x} = 0$.

If we set U_{lm} to 0.1, and use $N_y = 40$ lattices in the y-direction, then for $Re = 100$, we would have $\nu = 0.04$, $L/H = 20$, and then $N_x = 800$.

Figures 9.3 and 9.4 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 illustrate clearly the application of LBM to forced convection.

As an exercise, try to give a different thermal diffusivity to a part of the channel along the x-axis to simulate the conjugate effect. At the interface use the bounce-back scheme.

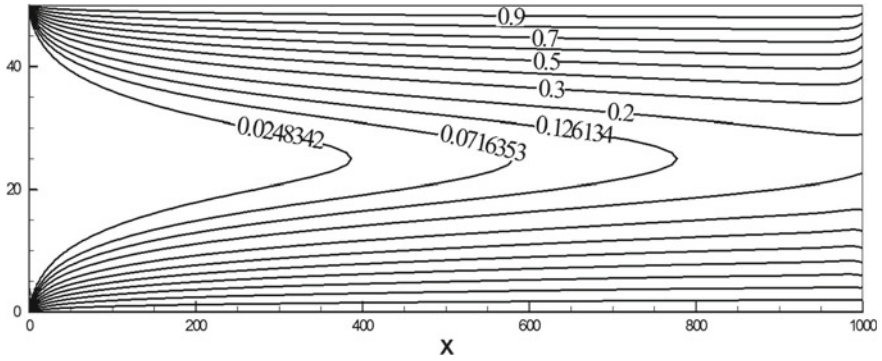


Fig. 9.3 Isothermals for forced convection in a heated channel

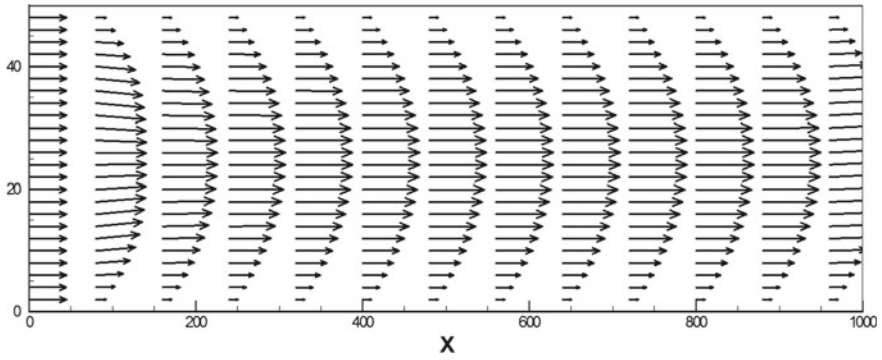


Fig. 9.4 Velocity vectors for forced convection in a heated channel

9.5 Conjugate Heat Transfer

In some problems, we are interested in solving for conduction in a solid coupled with convection in a fluid. For instance, in flow in a macrochannel, 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 a different thermal diffusivity for the solid from that assigned to 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 for a steady-state condition. The following results were 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 a few times greater than the thermal diffusivity of the fluid. Figures 9.5 and 9.6 show the isotherms and velocity vectors predicted by LBM. MATLAB code is given in the appendix.

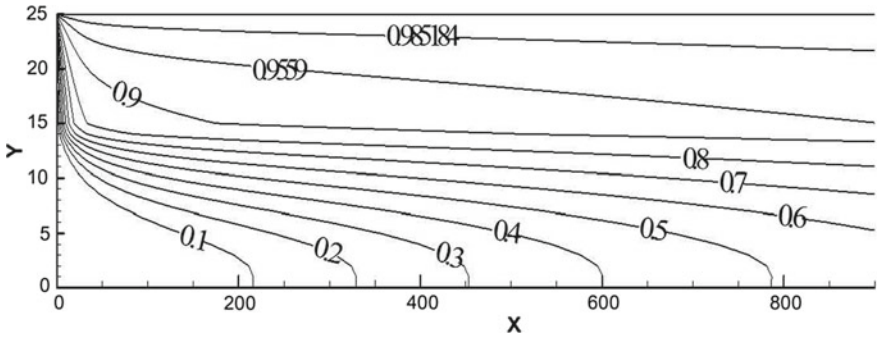


Fig. 9.5 Isotherms for conjugate convection problem in a heated channel

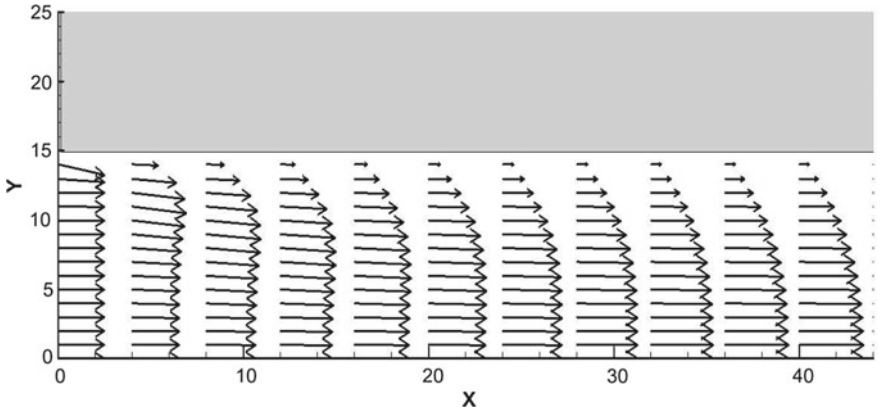


Fig. 9.6 Velocity vectors for forced conjugate convection in a heated channel

9.6 Natural Convection

For natural convection, the momentum and energy equations are coupled, and both should be updated simultaneously. The flow is driven by temperature or mass gradient, i.e., by the buoyancy force. Hence there is an extra force term that must be considered in solving the LB equation.

Then $F_x = \rho g_x$ and $F_y = \rho g_y$.

For natural convection, the Boussinesq approximation yields

$$F_x = \rho_o g_x \beta (T - T_{ref}) \tag{9.15}$$

and

$$F_y = \rho_o g_y \beta (T - T_{ref}) \,. \tag{9.16}$$

Then the nondimensional form of the momentum equations (9.1 and 9.2) can be written as

$$\frac{\partial u}{\partial t}+\frac{\partial(uu)}{\partial x}+\frac{\partial(\mathrm{v}u)}{\partial y}=-\frac{\partial p}{\partial x}+[\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}]+Ra_x\theta\,. \tag{9.17}$$

The y-momentum is given by

$$\frac{\partial \mathrm{v}}{\partial t}+\frac{\partial(u\mathrm{v})}{\partial x}+\frac{\partial(\mathrm{vv})}{\partial \mathrm{y}}=-\frac{\partial p}{\partial y}+[\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}]+Ra_y\theta\,, \tag{9.18}$$

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 the Rayleigh numbers in the x- and y-directions. And $\theta=\frac{T-T_{\mathrm{cold}}}{T_{hot}-T_{\mathrm{cold}}}$.

The nondimensional energy equation without heat source can be written as

$$\frac{\partial\theta}{\partial t}+\frac{\partial(u\theta)}{\partial x}+\frac{\partial(\mathrm{v}\theta)}{\partial y}=\frac{1}{\mathrm{Re}\,\mathrm{Pr}}[\frac{\partial^2\theta}{\partial x^2}+\frac{\partial^2\theta}{\partial y^2}]\,. \tag{9.19}$$

Note that H, H^2/v, v/H, and $(T_{hot}-T_{\mathrm{cold}})$ 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

$$F=3w(k)g_x\beta\theta e_x+3w(k)g_y\beta\theta e_y\,. \tag{9.20}$$

There is another way to add a force term to the LBE, by modifying the velocity in calculating the equilibrium distribution functions.

Newton's second law of motion states that

$$\mathbf{F}=m\mathbf{a}=m\frac{d\mathbf{u}}{dt}\,, \tag{9.21}$$

where $\mathbf{a}$ and $\mathbf{u}$ are the acceleration and velocity vectors.

Then

$$\Delta\mathbf{u}=\frac{\tau\mathbf{F}}{\rho}\,, \tag{9.22}$$

where τ is the relaxation time.

The velocity should be modified by $\Delta\,\mathbf{u}$ in calculating equilibrium distribution functions only, $\mathbf{u}^{eq}=\mathbf{u}+\Delta\mathbf{u}$.

9.6.1 Example: Natural Convection in a Differentially Heated Cavity

A differentially heated cavity is used as a benchmark solution for testing CFD codes. The problem involves a square cavity filled with air (Pr = 0.7) and heated from the left-hand side wall and cooled from the right-hand side wall. The 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. 9.7 with coordinate system and boundary conditions.

We need to calculate $g\beta$ from the definition of the Rayleigh number, $Ra = \frac{g\beta\Delta T H^3}{\alpha\upsilon}$. Then $g\beta = Ra\alpha\upsilon/(\Delta T H^3)$. In lattice units, we can choose $\upsilon = 0.03$ (or any other value, but not large, less than 0.1). Since $\Pr = \frac{\upsilon}{\alpha} = 0.7$, we have $\alpha = 0.03/0.7$, and 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 the H-direction.

Note that the velocity scale in natural convection is proportional to $\sqrt{g\beta\Delta T H}$; in order to ensure that the problem is in the incompressible regime, the value of $\sqrt{g\beta\Delta T H}$ is less than about 0.1. For the above-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 the H-direction.

The complete computer code is given in the appendix.

Figure 9.8 shows streamlines for $\mathrm{Ra} = 10^5$ and Pr = 0.71. The results match the finite volume method predictions. The streamlines shows perfect skew-symmetry, which is one criterion of natural convection in a differentially heated cavity (Fig. 9.9).

For $\mathrm{Ra} = 10^6$, change Ra to 10^6 instead of 10^5 in the code. The boundary layer becomes thinner, as shown in Fig. 9.10.

For higher Ra, more lattices were needed, because the boundary layer becomes thinner as the Rayleigh number increases.

9.7 Flow and Heat Transfer in Porous Media

Fluid flow and heat and mass transfer in porous media have many applications. The literature on the topic is 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,

$$\mathbf{F} = -\frac{\nu}{K}\mathbf{u} - \frac{\beta}{\sqrt{K}}|\mathbf{u}|\mathbf{u}\,, \tag{9.23}$$

where K is the permeability of the medium. For a medium uniformly filled with spherical particles of diameter d_p, K can be expressed as

$$K = \frac{\epsilon^3 d_p}{150(1-\epsilon)^2}\,. \tag{9.24}$$

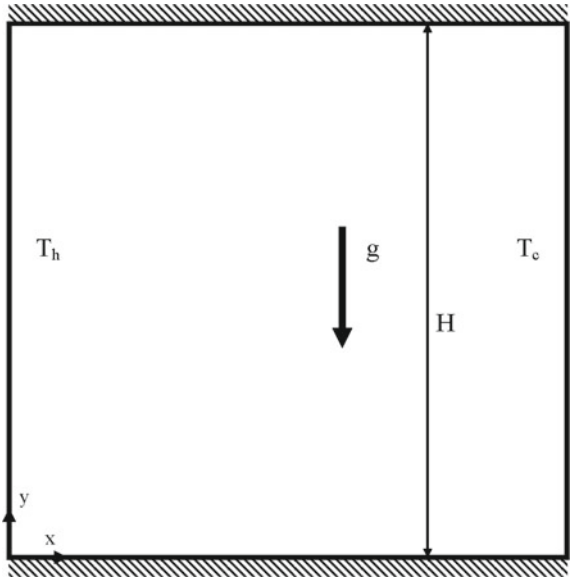


Fig. 9.7 Sketch of the natural convection problem

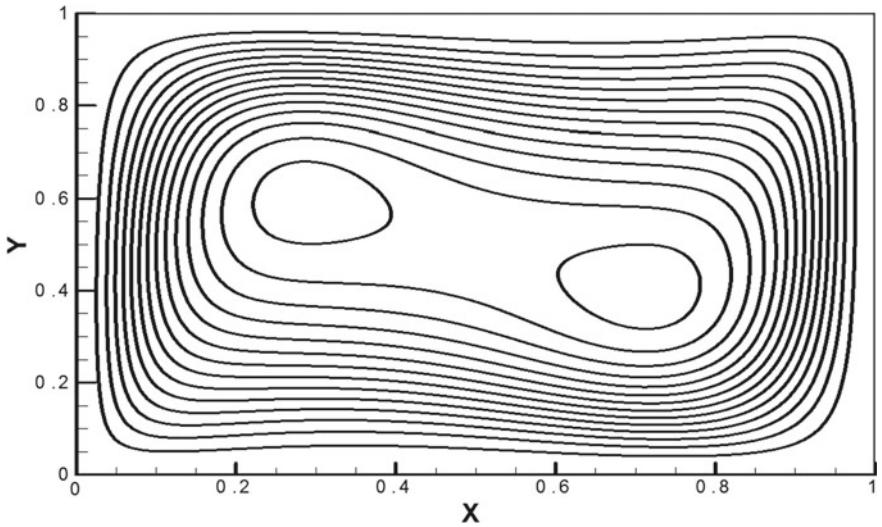


Fig. 9.8 Streamlines for natural convection in a square cavity, $Ra = 10^5$, $Pr = 0.71$

The constant β is usually set to $1.75/\sqrt{150\epsilon}$, where ϵ is the porosity of the medium. The vector **u** is the flow velocity vector. The first term in Eq. (9.23) is called the Darcy term, which accounts for the linear pressure drop due to the presence of a porous medium. The second nonlinear term is called the Forchheimer term, due to

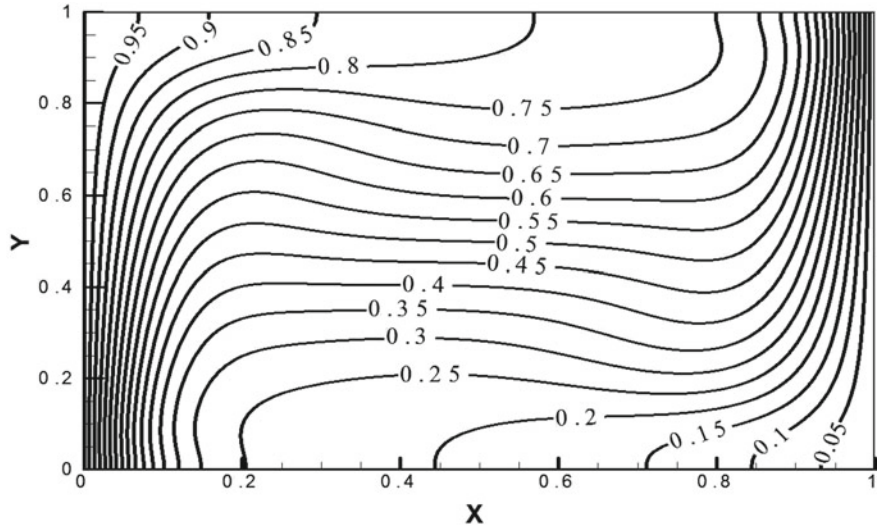


Fig. 9.9 Isotherm for natural convection in a square cavity, $Ra = 10^5$, $Pr = 0.71$

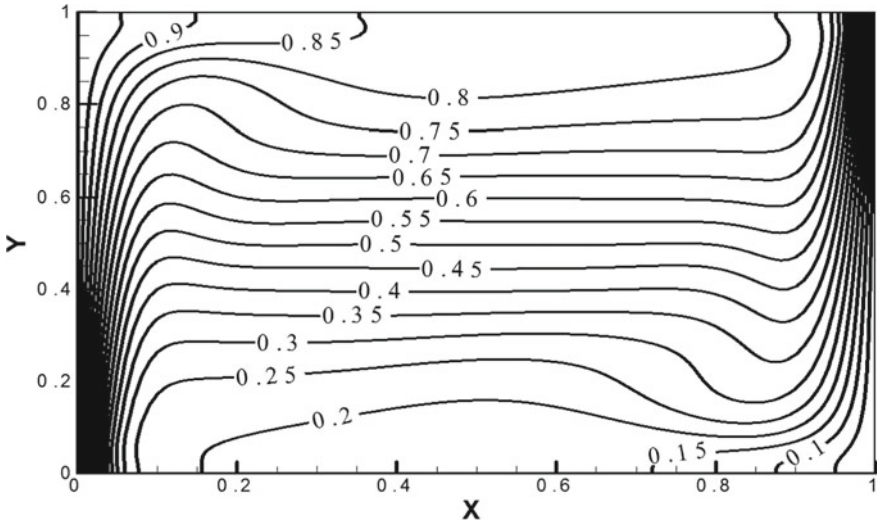


Fig. 9.10 Isotherms for natural convection in a square cavity, $Ra = 10^6$, $Pr = 0.71$

the nonlinear effect of the porous medium. It is useful to define the Darcy number in porous media applications as $Da = K/H^2$, 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 $-w(k)[9\frac{\nu}{K}(u\ c_x + v\ c_y) + \frac{\beta}{\sqrt{K}}(|u|uc_x + |v|vc_y)]$.

9.8 Flow and Heat Transfer in Axisymmetric Geometries

Many problems in engineering involve axisymmetric flow, such as flow in pipes, nozzles, and annular geometries. The mass and momentum transfers for axisymmetric flow without source term are as follows: Continuity equation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial u_r}{\partial r} + \frac{\partial u_z}{\partial z} = -\frac{u_r}{r} . \tag{9.25}$$

The r-momentum equation is

$$\frac{\partial u_r}{\partial t} + u_r \frac{\partial u_r}{\partial r} + u_z \frac{\partial u_r}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial r} + \nu\left(\frac{\partial^2 u_r}{\partial r^2} + \frac{\partial^2 u_r}{\partial z^2}\right) + \nu\left(\frac{1}{r}\frac{\partial u_r}{\partial r} - \frac{u_r}{r^2}\right) . \tag{9.26}$$

The z-momentum equation is

$$\frac{\partial u_z}{\partial t} + u_r \frac{\partial u_z}{\partial r} + u_z \frac{\partial u_z}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} + \nu\left(\frac{\partial^2 u_z}{\partial r^2} + \frac{\partial^2 u_z}{\partial z^2}\right) + \nu\left(\frac{1}{r}\frac{\partial u_z}{\partial r}\right) . \tag{9.27}$$

The last source (sink) terms in the above equations need to injected into the lattice Boltzmann method. A few authors have attempted to incorporate those source terms in LBM. The method introduced by J.G. Zhou (J.G. Zhou, 2008, Axisymmetric lattice Boltzmann method, *Physical Review E* 78, 036701) is simple. The lattice Boltzmann method is modified as

$$f_i(x + e_i\Delta t, t + \Delta t) = f_i(x,t) + \frac{1}{\tau}(f_i^{eq} - f_i) + F_1\Delta t + \frac{\Delta x^2}{6\Delta t}c_{ii}F_i , \tag{9.28}$$

where

$$F_1 = -\frac{\rho}{9}\frac{u_r}{r} \tag{9.29}$$

and

$$F_i = \rho\left(-\frac{u_i u_r}{r} + \frac{\nu}{r}\frac{\partial u_i}{\partial r} - \frac{\nu u_i}{r^2}\delta_{ir}\right) . \tag{9.30}$$

The reader can modify the code for channel flow to simulate flow in a pipe, for example. Also, the interested reader can consult the above-mentioned paper for more detail on analysis and the Chapman–Enskog expansion method. The energy equation also includes a source term,

$$\frac{\partial T}{\partial t} + u_r \frac{\partial T}{\partial r} + u_z \frac{\partial T}{\partial z} = \alpha\left(\frac{\partial^2 T}{\partial r^2} + \frac{\partial^2 T}{\partial z^2}\right) + \alpha\frac{1}{r}\frac{\partial T}{\partial r} . \tag{9.31}$$

The derivative term can be approximated using the finite difference method.

Chapter 10
Multi-Relaxation Schemes

The single-relaxation scheme has been discussed extensively, and many problems were solved in the pervious chapters. There is a claim that multi-relaxation schemes offer greater stability and accuracy than the single-relaxation scheme. This chapter is devoted to explaining the multi-relaxation-time scheme (MRT).

10.1 Multi-Relaxation Method, MRT

The collision operator can be generalized as

$$f_i(x + c\Delta t, t + \Delta t) - f_i(x, t) = -\Omega[f_i(x, t) - f_i^{eq}(x, t)]\,, \tag{10.1}$$

where Ω 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. (10.1) can transformed into the following form:

$$f_i(x + c\Delta t, t + \Delta t) - f_i(x, t) = -\mathbf{M}^{-1}\mathbf{S}[\mathbf{m}(x, t) - \mathbf{m}^{eq}(x, t)]\,, \tag{10.2}$$

where $\mathbf{m}(x, t)$ and $\mathbf{m}^{eq}$ are vectors of moments, $\mathbf{m} = (m_o, m_1, m_2, \ldots m_n)^T$.

The relaxation matrix $\mathbf{S}$ is a diagonal matrix.

The mapping between velocity and moment spaces can be performed by linear transformation,

$$\mathbf{m} = \mathbf{M}\mathbf{f} \quad \text{and} \quad \mathbf{f} = \mathbf{M}^{-1}\mathbf{m}\,. \tag{10.3}$$

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3_10

The matrix $\mathbf{M}$ for D2Q9 is

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{pmatrix}.$$

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

Note that the matrix M is written by assuming that the center of the lattice is zero (0), which is fine if the reader codes in Fortran. In Matlab, indexing zero is not permitted, and therefore in the code (appendix), the index 9 is used for the central point. Hence the first column of the matrix M should be the last column.

The moment vector $\mathbf{m}$ is

$$\mathbf{m} = (\rho, e, \epsilon, j_x, q_x, j_y, q_y, p_{xx}, p_{xy})^T . \tag{10.4}$$

The equilibrium of the moment $\mathbf{m}^{eq}$ is

$$\begin{aligned} m_o^{eq} &= \rho , \\ m_1^{eq} &= -2\rho + 3(j_x^2 + j_y^2) , \\ m_2^{eq} &= \rho - 3(j_x^2 + j_y^2) , \\ m_3^{eq} &= j_x , \\ m_4^{eq} &= -j_x , \\ m_5^{eq} &= j_y , \\ m_6^{eq} &= -j_y , \\ m_7^{eq} &= (j_x^2 - j_y^2) , \\ m_8^{eq} &= j_x j_y , \end{aligned} \tag{10.5}$$

where

$$\begin{aligned} j_x &= \rho u_x = \sum_i f_i^{eq} c_{ix} \,, \\ j_y &= \rho u_y = \sum_i f_i^{eq} c_{iy} \,. \end{aligned} \tag{10.6}$$

For problems with external force term F, the above equations can be modified as

$$\begin{aligned} j_x &= \rho u_x = \sum_i f_i^{eq} c_{ix} - F/2 \,, \\ j_y &= \rho u_y = \sum_i f_i^{eq} c_{iy} - F/2 \,. \end{aligned} \tag{10.7}$$

The diagonal matrix $\mathbf{S}$ is given by

$$\mathbf{S} = \begin{pmatrix} s_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & s_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & s_2 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & s_3 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & s_4 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & s_5 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & s_6 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_7 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_8 & \end{pmatrix}.$$

In compact notation, $\mathbf{S}$ can be written as

$$\mathbf{S} = diag(1.0, 1.4, 1.4, s3, 1.2, s5, 1.2, s7, s8) \,, \tag{10.8}$$

where $s7 = s8 = 2/(1 + 6\nu)$, s_3, and s_5 are arbitrary and 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

$$\mathbf{S} = diag(1, s, s, 1, 1, 1) \,, \tag{10.9}$$

where $s = \frac{1}{5\alpha} + 0.5$, and α is the diffusion coefficient (note that the other elements on $\mathbf{S}$ are arbitrary; here we used 1).

For the advection–diffusion problem, the equilibrium moments are

$$\begin{aligned} m_o^{eq} &= \rho\,, \\ m_1^{eq} &= \rho u_x\,, \\ m_2^{eq} &= \rho u_y\,, \\ m_3^{eq} &= 1\,, \\ m_4^{eq} &= 1\,. \end{aligned} \tag{10.10}$$

In order to employ MRT, the collision subroutine needs to be modified. Other than that, the procedure is the same as for SRM. The code for a lid-driven cavity with MRT collision is given in the appendix. Note that only the main and collision codes are different from the code of SRM.

10.2 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 the Rayleigh number and Prandtl number be set to 10^6 and 0.71, respectively. Also, try to get results for higher Rayleigh numbers, 10^7, 10^8, and 10^9. Compare the predicted results with the prediction from using SRM.

10.3 Two-Relaxation-Time (TRT) Scheme

The two-relaxation-time scheme was developed by Ginzburg (see the paper, Ginzburg, I., *Adv. Water Resource.* 28(11) (2005), 1171, for more details). The distribution function can be split into two parts, symmetric and antisymmetric, as

$$f_i^s = \frac{1}{2}(f_i + f_{-i}) \tag{10.11}$$

and

$$f_i^a = \frac{1}{2}(f_i - f_{-i}) \tag{10.12}$$

respectively. The distribution functions f_i and f_{-i} can be constructed by adding and subtracting the above equations, i.e., $f_i = f_i^a + f_i^s$ and $f_{-i} = f_i^a - f_i^s$. Also, $f_i^a = f_{-i}^a$ and $f_i{}^s = -f_{-i}{}^s$, where f_{-i} is the distribution function moving opposite to f_i. The updating of the collision equation without source term can be expressed as

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) - \frac{1}{\tau_s}\left(f_i^s(x,t) - f_i^{seq}\right) - \frac{1}{\tau_a}\left(f_i^a(x,t) - f_i^{aeq}\right). \tag{10.13}$$

Note that if τ_a is equal to τ_s, then Eq. (10.3) becomes a single-relation BGK scheme. The streaming process is the same as for the single-relation scheme. Equation (10.3) can be simplified using Eq. (10.1) and (10.2) as

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) - \frac{1}{2}(\omega_s + \omega_a) f_i^{seq} - \frac{1}{2}(\omega_s - \omega_a) f_i^{aeq} \tag{10.14}$$

where $\omega_a = 1/\tau_a$ and $\omega_s = 1/\tau_s$. The symmetric relaxation coefficient ω_s tunes the viscosity, i.e., for D2Q9, for example,

$$\omega_s = \frac{1}{3\,\nu + 0.5}\,. \tag{10.15}$$

The asymmetric relaxation coefficient ω_a is chosen to minimize the viscosity dependence on the slip velocity, such as

$$\omega_a = \frac{8(2 - \omega_s)}{8 - \omega_s}\,. \tag{10.16}$$

The computer code is quite similar to the single-relaxation method (SRM). The reader is encouraged to solve a few of the previous problems using the TRT method and then compare the results of both schemes, TRT and SRM. However, the TRT scheme is not as popular as SRM and MTR.

Chapter 11
Complex Flows

Many real-life engineering problems are more complex than what has been described in the previous chapters. However, the materials covered in those chapters are building blocks for dealing with more complex flows. For instance, boiling and condensation take place in many industrial and power plant systems. Flows of oil–water–solid are very common in oil extraction and oil sand processes. And water treatment by macro- and microbubbles, combustion in furnaces, incineration of waste materials, are a few examples worthy of mention. The question is how to model and solve such problems. The natural way is to incorporate complex physics in the source term. In fact, LBM is more powerful for dealing with multiphase and multicomponent flows than the Navier–Stokes (NS) equation of continuum mechanics.

The main reason for this disparity is that such problems involve thermodynamics, which can be naturally injected into LBM, while there is no simple way to combine thermodynamics with Navier–Stokes. The strategy of blaming extra physics to the source term is commonly used in the finite volume method. Most commercial software solves the general transport equation. The general transport equation in two-dimensional Cartesian coordinates can be written as

$$\frac{\partial \rho \Phi}{\partial t} + \frac{\partial \rho u \Phi}{\partial x} + \frac{\partial \rho v \Phi}{\partial y} = \frac{\partial}{\partial x}\left(\Gamma \frac{\partial \Phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\Gamma \frac{\partial \Phi}{\partial y}\right) + S\,. \tag{11.1}$$

The continuity equation can be recovered if we set $\Phi = 1$ and $S = 0$. The X-moment equation can be recovered by setting $\Phi = u$, $\Gamma = \mu$, and $S = -\frac{\partial p}{\partial x} + F_x$, and the energy equation can be recovered by setting $\Phi = c_p T$, $\Gamma = k$, and so on.

The source term can be incorporated into LBM as an external force, as discussed before. The beauty of LBM in treating multiphase flows is that there is no need to trace the interface between phases, as is the case in NS. Hence, multiphase coding is much easier in LBM than in an NS solver. In this book, we avoided discussing such problems for one simple reason. The method is relatively simple to employ, but the reader needs a background in the physics of the problem before applying LBM.

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3_11

I would like to emphasize that understanding the physics of the problem is more essential than solving the problem itself. Even for numerical treatment of a problem, understanding the physics of the problem helps in selecting the correct scheme and analyzing the predicted results. Therefore, the complex flow is left to the interested reader. However, I am sure that the reader can easily extend the previous codes to handle complex phenomena, provided that the reader understands the underlying physics of the problem.

In the following section, the Shan–Chen potential model is introduced to model two-phase flows. The scheme is easy to implement. For more detail about the method, we refer to the original works of Shan–Chen (*Physical Review E* 47(3) (1993), 181, and *J. Statistical Physics* 81(1) (1995), 379).

11.1 Shan–Chen Potential Method

The concept of the Shan–Chen (SC) model will be introduced briefly. The main idea relies on molecular interaction forces. For instance, the interaction force between water molecules is different from the interaction force between molecules of other fluids. Therefore, the interaction force depends on the density of the fluid, where greater density means greater interaction force. The interaction force can be modeled as

$$F(x) = -\int G(x, x - \grave{x})\psi(x)\psi(\grave{x})(\grave{x} - x)d\grave{x}\,, \tag{11.2}$$

where ψ is the effective density, called a pseudopotential. The relationship between density ρ and effective density ψ can be expressed as

$$\psi(\rho) = \rho_o[1 - exp[(-\rho/\rho_o)]\,, \tag{11.3}$$

where ρ_o is a reference density, usually set to unity. Other forms have been used by different authors,

$$\psi(\rho) = \psi_o\, exp(-\rho/\rho_o)\,. \tag{11.4}$$

The force term (Eq. 11.2) can be written as

$$F(x,t) = -G\psi(x,t)\sum_{i=1}^{8} w_i\psi(x + e_i\Delta t, t)e_i\,, \tag{11.5}$$

where G reflects the interaction strength. The state equation can be written as

$$p = \frac{\rho}{3} + \frac{G}{6}(\psi)^2\,. \tag{11.6}$$

To illustrate, the application of the SC model to two immiscible fluids randomly mixed in a container is simulated. The dispersed fluid forms larger droplets within a period of time. For details, see the code in the appendix. The G value is set to -5.0. In Eq. (11.4), ψ_o and ρ_o are set to 1.0.

Bibliography

For those readers who want to read more, I recommend some references, listed in no order of preference. There are many excellent works in the literature, much of it available on the internet. The following references are more or less randomly selected from a pool of excellent sources. Also, I have listed a few papers on multiphase flow for those readers who are interested in modifying the given codes to include multiphysics. Most physics can be included in the force term.

Boundary Conditions

Yu D, Mei R, Lou L, Shyy W. Viscous flow computations with the method of Lattice Boltzmann equation. Prog Aerosp Sci. 2003;39:329–367. This paper reviews different techniques of boundary conditions and introduces a new method of treating inlet boundary conditions based on the doctoral dissertation 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. 1995;7 12:2928–2930.

Chen H, Teixeria C, Molving K. Realization of fluid boundary conditions via discrete Boltzmann dynamics. Int J Modern Phys C. 1–7.

Bouzidi M, Firdaouss M, Lallemand P. Momentum transfer of a Boltzmann-Lattice fluid with boundaries. Phys Fluids. 13 11:3452–3459.

Noble D, Chen S, Georgiadis J, Buckius R. A consistent hydrodynamic boundary condition for the Lattice Boltzmann method. Phys Fluids. 1995;7 1:203–209.

Maier R, Bernard R, Grunau D. Boundary condition for the Lattice Boltzmann method Physic Fluids. 1996;8 7:1788–1801.

Inamuro T, Yoshino M, Ogino F. A non-slip boundary condition for Lattice Boltzmann simulations. Phys Fluids. 1995;7 12:2928–2930.

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3

Review Papers

Chen S, Doolen GD. Lattice Boltzmann method for fluid flows. Ann Rev Fluid Mech. 1998;30:329–364.

Diffusion

Wolf-Gladrow D. A Lattice Boltzmann equation for diffusion. J Stat Phys. 1995;79 5/6:1023–1032.

Ho J, Kuo C, Jiaung W. Twu C. Lattice Boltzmann scheme for hyperbolic heat conduction equation. Numerical Heat Transf Part B. 41:591–607.

Jiaung W, Ho J, Kuo C. Lattice Boltzmann method for the heat conduction problem with phase change. Numerical Heat Transf Part B. 2001;39:167–187.

Van Der Sman R. Diffusion on unstructured triangular grids using Lattice Boltzmann. Future Gener Comput Syst. 2004;20:965–971.

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;48:3648–3659.

Advection–Diffusion

Dawson SP, Chen S, Doolen GD. Lattice Boltzmann computations for reaction-diffusion equations. J Chem Phys 1993;98 2:1514–1523.

Van der Saman RGM, Ernst MH, Berkenbosh AC. Lattice Boltzmann scheme for cooling and packed cut flowers. Int J Heat Mass Transf. 2000;43:577–587.

Kingdon RD, Schofield P. A reaction-flow Lattice Boltzmann model. J Phys A: Math Gen. 1992;25:L907–L910.

Van der Saman RGM, Ernst MH. Convection–diffusion Lattice Boltzmann scheme for irregular lattices. J Comput Phys. 2000;160:766–782.

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 016703:016703-1–016703-8.

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;47:4869–4879.

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 026705:026705-1–026705-17.

Seta T, Eishun T, Okui K. Lattice Boltzmann simulation of natural convection in porous media. Math Comput Simul. 2006;72:195–200.

MRT

Mezrhab A, Moussaoui MA, Jami M, Naji H, Bouzidi M. Double MRT thermal Lattice Boltzmann method for simulation convective flows. Phys Lett A. 2010;374:3499–3507.

Wang L, Guo Z, Zheng C. Multi-relaxation-time Lattice Boltzmann model for axisymmetric flows. Comput Fluids. 2010;39:1542–1548.

Multiphase

Frank X, Funfschilling D, Midoux N, Li H. Bubbles in a viscous liquid: Lattice Boltzmann simulation and experimental validation. J Fluid Mech. 2006;546:113–122.

Shan X, Chen H, Lattice Boltzmann model for simulating flows with multiple phases and components. Phys Rev E 1993;47 3:1815–1819.

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;41:843–854.

Peng Y, Shu C, Chew YT. Lattice kinetic scheme for the incompressible viscous thermal flows on arbitrary meshes. Phys Rev E. 2004;69 016703:016703-1–016703-8.

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;47:4869–4879.

Van Der Sman R. Galilean invariant Lattice Boltzmann scheme for natural convection on square and rectangular lattices. Phys Rev E. 2006:74 026705:026705-1–026705-17.

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;41:843–854.

Li B, Kwok DY. Lattice Boltzmann model of microfluidics in the presence of external forces. J Colloid Interface Sci. 2003;263:144–151.

Guangwu Y. A Lattice Boltzmann equation for waves. J Comput Phys. 2000; 161:61–69.

Buick JM, Greated CA. Gravity in a Lattice Boltzmann model. Phys Rev E. 61 5:5307–5316.

Cates ME, Stratford K, Adhikari R, Stansell P, Desplat J-C, Pagonabarraga I, Wagner AJ. Simulating colloid hydrodynamics with Lattice Boltzmann methods. J Phys Condens Matter. 2004;16;S3903–S3915.

Vahala L, Vahala G, Yepez J. Lattice Boltzmann and quantum lattice gas representation of one-dimensional magnetohydrodynamic turbulence. Phys Lett A. 2003;306:227–234.

Yepez J., Type-II quantum computers. Int J Modern Phys C. 2001;12 9:1273–1284.

Yepez J. Quantum lattice-gas model for the diffusion equation. Int J Modern Phys C. 2001;12 9:1285–1303.

Yamamoto K, He X. Doolen GD. Simulation of combustion field with Lattice Boltzmann method. J Stat Phys. 2002;107 112:367–383.

Chopard B, Luthi PO. Lattice Boltzmann computations and appricationsto physics. Theor Comput Sci 1999;217:115–130.

Li Baoming, 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;46:4235–4244.

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 Nonlinear Sci Numerical Simul. 200:64–68.

Toffoli T, Margolus N. Invertible cellular automata. A Rev Phys D. 1990; 45:229–253.

Wolf-Gladrow D. A Lattice Boltzmann equation for diffusion. J Stat Phys. 1995;79:1023–1032.

Van der Sman RGM, Ernst MH, Berkenbosch AC. Lattice Boltzmann scheme for cooling of packed cut flowers. Int J Heat Mass Transf. 2000;43:577–587.

Ho J-R, Kuo C-P, Jiaung W-S, Twu C-J. Lattice Boltzmann scheme for hyperbolic heat conduction equation. Numerical 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. Numerical Heat Transf. 2001;B39:167–187.

Appendix A
Computer Codes

A.1 Chapter Five

```
% Chapter 5
% LBM- 1-D, diffusion equation D1Q2
clear
m=101;
dx=1.0;
rho=zeros(m);f1=zeros(m);f2=zeros(m); flux=zeros(m);
x=zeros(m);
x(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
alpha=0.25;
omega=1/(alpha+0.5);
twall=1.0;
nstep=200;
for i=1:m
    f1(i)=0.5*rho(i);
    f2(i)=0.5*rho(i);
end
%Collision:
for k1=1:nstep
    for i=1:m
        feq=0.5*rho(i);
        f1(i)=(1-omega)*f1(i)+omega*feq;
        f2(i)=(1-omega)*f2(i)+omega*feq;
    end

    % Streaming:
    for i=1:m-1
        f1(m-i+1)=f1(m-i);
        f2(i)=f2(i+1);
    end
    %Boundary condition:
   f1(1)=twall-f2(1);
   f1(m)=f1(m-1);
   f2(m)=f2(m-1);
   for j=1:m
```


A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3

```
       rho(j)=f1(j)+f2(j);
   end
end
   %Flux:
    for k=1:m
    flux(k)=omega*(f1(k)-f2(k));
    end
figure(1)
plot(x,rho)
    title('Temperature')
    xlabel('X')
    ylabel('T')
figure(2)
plot(x,flux,'o')
   title('Flux, time step=200')
   xlabel('X')
   ylabel('Flux')
================================================================================
% Chapter 5, diffusion
% LBM- 1-D1Q3, diffusion equation
clear
m=101;
w0=4./6.;
w1=1./6.;
c2=1./3.;
w2=w1;
dx=1.0;
rho=zeros(m);f0=zeros(m);f1=zeros(m);f2=zeros(m);
x=zeros(m);fluxq=zeros(m);
x(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
alpha=0.25;
omega=1/(3.*alpha+0.5);
twall=1.0;
nstep=200;
for i=1:m
    f0(i)=w0*rho(i);
    f1(i)=w1*rho(i);
    f2(i)=w1*rho(i);
end
%Collision:
for k1=1:nstep
    for i=1:m
        feq0=w0*rho(i);
        feq=w1*rho(i);
        f0(i)=(1-omega)*f0(i)+omega*feq0;
        f1(i)=(1-omega)*f1(i)+omega*feq;
        f2(i)=(1-omega)*f2(i)+omega*feq;
    end
    % Streaming:
    for i=1:m-1
        f1(m-i+1)=f1(m-i);
        f2(i)=f2(i+1);
    end

    %Boundary condition:
```

```
   f1(1)=twall-f2(1)-f0(1);
   f1(m)=f1(m-1);
   f2(m)=f2(m-1);
   f0(m)=f0(m-1);
   for j=1:m
       rho(j)=f1(j)+f2(j)+f0(j);
   end
end
   %Flux:
    for k=1:m
    flux(k)=omega*(f1(k)-f2(k))/c2;
    end
    for k=1:m-1
        fluxq(k)=rho(k)-rho(k+1);
    end
    fluxq(m)=fluxq(m-1);
figure(1)
plot(x,rho)
    title('Temperature, nstep=200')
    xlabel('X')
    ylabel('T')
figure(2)
plot(x,flux,'o',x,fluxq,'x')
   title('Flux, time step=200')
   xlabel('X')
   ylabel('Flux')
================================================================================
% Chapter 5, D1Q3 with flux boundary condition
% LBM- 1-D1Q3, diffusion equation
clear
m=101;
w0=4./6.;
w1=1./6.;
c2=1./3.;
w2=w1;
dx=1.0;
qf=100.; tk=20.;
rho=zeros(m);f0=zeros(m);f1=zeros(m);f2=zeros(m);
x=zeros(m);fluxq=zeros(m);flux=zeros(m);
x(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
alpha=0.25;
omega=1/(3.*alpha+0.5);

nstep=600;
for i=1:m
    f0(i)=w0*rho(i);
    f1(i)=w1*rho(i);
    f2(i)=w1*rho(i);
end
%Collision:
for k1=1:nstep
    for i=1:m
        feq0=w0*rho(i);
        feq=w1*rho(i);
```

```
        f0(i)=(1-omega)*f0(i)+omega*feq0;
        f1(i)=(1-omega)*f1(i)+omega*feq;
        f2(i)=(1-omega)*f2(i)+omega*feq;
    end

    % Streaming:
    for i=1:m-1
        f1(m-i+1)=f1(m-i);
        f2(i)=f2(i+1);
    end
    %Boundary condition:
   f1(1)=f2(1)+c2*qf/(tk*omega);
   f1(m)=f1(m-1);
   f2(m)=f2(m-1);
   f0(m)=f0(m-1);
   for j=1:m
       rho(j)=f1(j)+f2(j)+f0(j);
   end
end
   %Flux:
    for k=1:m
    flux(k)=omega*(f1(k)-f2(k))/c2;
    end
    for k=1:m-1
        fluxq(k)=rho(k)-rho(k+1);
    end
    fluxq(m)=fluxq(m-1);
figure(1)
plot(x,rho)
    title('Temperature, nstep=200')
    xlabel('X')
    ylabel('T')
figure(2)
plot(x,flux,'o',x,fluxq,'x')
   title('Flux, time step=200')
   xlabel('X')
   ylabel('Flux')
================================================================================

% Chapter 5, source term
% LBM- 1-D1Q3, diffusion equation
clear
m=101;
w0=4./6.;
w1=1./6.;
c2=1./3.;
w2=w1;
dx=1.0;
rho=zeros(m);f0=zeros(m);f1=zeros(m);f2=zeros(m);
x=zeros(m);fluxq=zeros(m);flux=zeros(m);
x(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
rcp=200.0;
qs=1.0;
qsr=qs/rcp;
alpha=0.25;
```

```
omega=1/(3.*alpha+0.5);
twall=1.0;
tk=alpha*rcp;
nstep=200;
for i=1:m
    f0(i)=w0*rho(i);
    f1(i)=w1*rho(i);
    f2(i)=w1*rho(i);
end
%Collision:
for k1=1:nstep
    for i=1:m
        feq0=w0*rho(i);
        feq=w1*rho(i);
        f0(i)=(1-omega)*f0(i)+omega*feq0+qsr*w0;
        f1(i)=(1-omega)*f1(i)+omega*feq+qsr*w1;
        f2(i)=(1-omega)*f2(i)+omega*feq+qsr*w1;
    end

    % Streaming:
    for i=1:m-1
        f1(m-i+1)=f1(m-i);
        f2(i)=f2(i+1);
    end
    %Boundary condition:
   f1(1)=twall-f2(1)-f0(1);
   f1(m)=f1(m-1);
   f2(m)=f2(m-1);
   f0(m)=f0(m-1);
   for j=1:m
       rho(j)=f1(j)+f2(j)+f0(j);
   end
end
   %Flux:
    for k=1:m
    flux(k)=omega*(f1(k)-f2(k))/c2;
    end
    for k=1:m-1
        fluxq(k)=rho(k)-rho(k+1);
    end
    fluxq(m)=fluxq(m-1);
figure(1)
plot(x,rho)
    title('Temperature')
    xlabel('X')
    ylabel('T')
figure(2)
plot(x,flux,'o',x,fluxq,'x')
   title('Flux')
   xlabel('X')
   ylabel('Flux')
================================================================================
% LBM- 1-D1Q3, diffusion equation D1Q3, with variable alpha
clear
m=101;
w0=4./6.;
w1=1./6.;
c2=1./3.;
```

```
w2=w1;
dx=1.0;
rho=zeros(m);f0=zeros(m);f1=zeros(m);f2=zeros(m);
x=zeros(m);fluxq=zeros(m);flux=zeros(m);
tk=zeros(m);dtkdx=zeros(m);cpr=zeros(m);alpha=zeros(m);
omega=zeros(m);
x(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
for i=1:m
    tk(i)=20.+30./(2.*x(i)+1.);
    alpha(i)=tk(i)/100.;
    omega(i)=1/(3.*alpha(i)+0.5);
end
twall=1.0;
nstep=1500;
for i=1:m
    f0(i)=w0*rho(i);
    f1(i)=w1*rho(i);
    f2(i)=w1*rho(i);
end
%Collision:
for k1=1:nstep
    for i=1:m
        feq0=w0*rho(i);
        feq=w1*rho(i);
        f0(i)=(1-omega(i))*f0(i)+omega(i)*feq0;
        f1(i)=(1-omega(i))*f1(i)+omega(i)*feq;
        f2(i)=(1-omega(i))*f2(i)+omega(i)*feq;
    end

    % Streaming:
    for i=1:m-1
        f1(m-i+1)=f1(m-i);
        f2(i)=f2(i+1);
    end
    %Boundary condition:
   f1(1)=twall-f2(1)-f0(1);
   f1(m)=f1(m-1);
   f2(m)=f2(m-1);
   f0(m)=f0(m-1);
   for j=1:m
       rho(j)=f1(j)+f2(j)+f0(j);
   end
end
   %Flux:
    for k=1:m
    flux(k)=tk(k)*omega(k)*(f1(k)-f2(k))/c2;
    end
    for k=1:m-1
        fluxq(k)=tk(k)*(rho(k)-rho(k+1));
    end
    fluxq(m)=fluxq(m-1);
figure(1)
plot(x,rho)
    title('Temperature, nstep=200')
    xlabel('X')
```

```
    ylabel('T')
figure(2)
plot(x,flux,'o',x,fluxq,'x')
   title('Flux')
   xlabel('X')
   ylabel('Flux')
================================================================================

% LBM- 2-D2Q5a, diffusion equation, note that c2=1/3, w0=2/6, others 1/6
clear
m=51;n=51;
xl=1.0;yl=1.0;
dx=xl/(m-1.0); dy=yl/(n-1.0);
w0=2./6.;
w=1./6.;
c2=1./3.;
dx=1.0;
f0=zeros(m,n);f1=zeros(m,n);f2=zeros(m,n);f3=zeros(m,n);f4=zeros(m,n);
rho=zeros(m,n);x=zeros(m);y=zeros(n);fluxq=zeros(m);flux=zeros(m);
Tm=zeros(m);Z=zeros(n,m);
x(1)=0.0; y(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
for j=1:n-1
    y(j+1)=y(j)+dy;
end
alpha=0.25;
omega=1/(3.*alpha+0.5);
twall=1.0;
nstep=400;
for i=1:m
    f0(i,j)=w0*rho(i,j);
    f1(i,j)=w*rho(i,j);
    f2(i,j)=w*rho(i,j);
    f3(i,j)=w*rho(i,j);
    f4(i,j)=w*rho(i,j);
end
%Collision:
for k1=1:nstep
    for j=1:n
    for i=1:m
        feq0=w0*rho(i,j);
        feq=w*rho(i,j);
        f0(i,j)=(1.-omega)*f0(i,j)+omega*feq0;
        f1(i,j)=(1.-omega)*f1(i,j)+omega*feq;
        f2(i,j)=(1.-omega)*f2(i,j)+omega*feq;
        f3(i,j)=(1.-omega)*f3(i,j)+omega*feq;
        f4(i,j)=(1.-omega)*f4(i,j)+omega*feq;
    end
    end

    % Streaming:
    for j=1:n
    for i=1:m-1
        f1(m-i+1,j)=f1(m-i,j);
        f2(i,j)=f2(i+1,j);
    end
```

```
    end
    for i=1:m
        for j=1:n-1
            f3(i,n-j+1)=f3(i,n-j);
            f4(i,j)=f4(i,j+1);
        end
    end
    %Boundary condition:
   for j=1:n
    f1(1,j)=twall-f2(1,j)-f0(1,j)-f3(1,j)-f4(1,j);
   f3(m,j)=-f1(m,j)-f0(m,j)-f2(m,j)-f4(m,j);
   end
   for i=1:m
       f3(i,1)=f3(i,2);
       f4(i,n)=-f0(i,n)-f3(i,n)-f2(i,n)-f1(i,n);
   end

    for j=1:n
   for  i=1:m
       rho(i,j)=f1(i,j)+f2(i,j)+f0(i,j)+f3(i,j)+f4(i,j);
   end
    end
end
%rotating matrix for contour plotting
for j=1:n
    for i=1:m
        Z(j,i)=rho(i,j);
    end
end

  for i=1:n
      Tm(i)=rho(i,(n-1)/2);
  end
figure(1)
plot(x,Tm)
    xlabel('X')
    ylabel('T')
figure(2)
contour(Z)
%   title('Flux')
%   xlabel('X')
%   ylabel('Flux')
================================================================================

%LBM- 2-D2Q9, diffusion equation, note that c2=1/3, w0=4/9, w1-w4, 1/9
% and w5-w8, 1/36
clear
m=51;n=51;
xl=1.0;yl=1.0;
dx=xl/(m-1.0); dy=yl/(n-1.0);
w0=4./9.;
c2=1./3.;
dx=1.0;
f0=zeros(m,n);f=zeros(m,n,8);feq=zeros(m,n,8);f0eq=zeros(m,n);
rho=zeros(m,n);x=zeros(m);y=zeros(n);fluxq=zeros(m);flux=zeros(m);
Tm=zeros(m);Z=zeros(n,m);w(8)=zeros;
x(1)=0.0; y(1)=0.0;
for i=1:m-1
```

```
    x(i+1)=x(i)+dx;
end
for j=1:n-1
    y(j+1)=y(j)+dy;
end
for k=1:4
    w(k)=1./9.;
end
for k=5:8
    w(k)=1./36.;
end
alpha=0.25;
omega=1./(3.*alpha+0.5);
twall=1.0;
nstep=400;

%Collision:
for kk=1:nstep
    for j=1:n
    for i=1:m
        f0eq(i,j)=w0*rho(i,j);
        f0(i,j)=(1.-omega)*f0(i,j)+omega*f0eq(i,j);
    for k=1:8
        feq(i,j,k)=w(k)*rho(i,j);
        f(i,j,k)=(1.-omega)*f(i,j,k)+omega*feq(i,j,k);
    end
    end
    end

      % Streaming:
   f(:,:,1)=circshift( squeeze(f(:,:,1)), [+1,+0] );
   f(:,:,2)=circshift( squeeze(f(:,:,2)), [+0,+1] );
   f(:,:,3)=circshift( squeeze(f(:,:,3)), [-1,+0] );
   f(:,:,4)=circshift( squeeze(f(:,:,4)), [+0,-1] );
   f(:,:,5)=circshift( squeeze(f(:,:,5)), [+1,+1] );
   f(:,:,6)=circshift( squeeze(f(:,:,6)), [-1,+1] );
   f(:,:,7)=circshift( squeeze(f(:,:,7)), [-1,-1] );
  f(:,:,8)=circshift( squeeze(f(:,:,8)), [+1,-1] );
    % End of streaming
    %Boundary condition:
    %left boundary, twall=1.0
   for j=1:n
       f(1,j,1)=w(1)*twall+w(3)*twall-f(1,j,3);
       f(1,j,5)=w(5)*twall+w(7)*twall-f(1,j,7);
       f(1,j,8)=w(8)*twall+w(6)*twall-f(1,j,6);
   end
   %bottom boundary, adiabatic, bounce back
   for i=1:m
       f(i,1,2)=f(i,2,2);
       f(i,1,5)=f(i,2,5);
       f(i,1,6)=f(i,2,6);
   end
   %Top boundary, T=0.0
     for i=1:n
     f(i,m,7)=-f(i,m,5);
     f(i,m,4)=-f(i,m,2);
     f(i,m,8)=-f(i,m,6);
     end
```

```
     %right hand boundary
     for j=1:m
       f(n,j,3)=-f(n,j,1);
       f(n,j,7)=-f(n,j,5);
       f(n,j,6)=-f(n,j,8);
     end

% End of boundary conditions
    for j=1:n
   for  i=1:m
       sumk=0.0;
       for k=1:8
           sumk=sumk+f(i,j,k);
       end
        rho(i,j)=f0(i,j)+sumk;
   end
    end
end
%rotating matrix for contour plotting
for j=1:n
    for i=1:m
        Z(j,i)=rho(i,j);
    end
end
  for i=1:n
      Tm(i)=rho(i,(n-1)/2);
  end
figure(1)
plot(x,Tm)
    xlabel('X')
    ylabel('T')
figure(2)
contour(Z)
%   title('Flux')
%   xlabel('X')
%   ylabel('Flux')
================================================================================
%Chapter 5,
% Finit-Difference 1D, diffusion equation
clear
m=101;
dx=1.;
dx2=dx*dx;
dt=0.5;
alpha=0.50;
Twall=1.0;
T=zeros(m);To=zeros(m);
x=zeros(m);fluxq=zeros(m);
x(1)=0.0;
T(1)=Twall; To(1)=Twall;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
nstep=400;
for kk=1:nstep
for i=2:m-1
    T(i)=To(i)+dt*alpha*(To(i+1)-2.*To(i)+To(i-1))/dx2;
end
```

```
%update
for k=2:m-1
    To(k)=T(k);
end
To(m)=T(m-1);
end
 %Flux:
    for k=1:m-1
    fluxq(k)=(To(k)-To(k+1)/dx);
    end
    fluxq(m)=fluxq(m-1);
figure(1)
plot(x,To)
    title('Temperature')
    xlabel('X')
    ylabel('T')
figure(2)
plot(x,fluxq,'x')
   title('Flux')
   xlabel('X')
   ylabel('Flux')
================================================================================

% Chapter 5
% LBM- 1-Finite Difference, flux boundary condition
clear
m=101;
dx=1.0;
dx2=dx*dx;
dt=0.5;
qf=100.; tk=20.;
T=zeros(m);To=zeros(m);
x=zeros(m);fluxq=zeros(m);
x(1)=0.0;
T(1)=T(2)+qf*dx/tk; To(1)=T(1);
for i=1:m-1
    x(i+1)=x(i)+dx;
end
alpha=0.25;
nstep=800;
for kk=1:nstep
for i=2:m-1
    T(i)=To(i)+dt*alpha*(To(i+1)-2.*To(i)+To(i-1))/dx2;
end
%boundary condition
T(1)=T(2)+qf*dx/tk;
%update
for k=1:m-1
    To(k)=T(k);
end
To(m)=T(m-1);
end
 %Flux:
    for k=1:m-1
    fluxq(k)=(To(k)-To(k+1)/dx);
    end
    fluxq(m)=fluxq(m-1);
figure(1)
```

```
plot(x,To)
    title('Temperature, nstep=400')
    xlabel('X')
    ylabel('T')
figure(2)
plot(x,fluxq,'x')
   title('Flux, time step=400')
   xlabel('X')
   ylabel('Flux')
================================================================================

%Chapter 5
% Finit-Difference 1D, diffusion equation with source term
clear
m=101;
dx=1.0;
dx2=dx*dx;
dt=0.5;
qs=1.0;
rcp=200.;
qsr=qs/rcp;
alpha=0.25;
tk=alpha/rcp;
Twall=1.0;
T=zeros(m);To=zeros(m);
x=zeros(m);fluxq=zeros(m);
x(1)=0.0;
T(1)=Twall; To(1)=Twall;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
nstep=400;
for kk=1:nstep
for i=2:m-1
    T(i)=To(i)+dt*alpha*(To(i+1)-2.*To(i)+To(i-1))/dx2+dt*qsr;
end
%update
for k=2:m-1
    To(k)=T(k);
end
To(m)=T(m-1);
end
 %Flux:
    for k=1:m-1
    fluxq(k)=(To(k)-To(k+1)/dx);
    end
    fluxq(m)=fluxq(m-1);
figure(1)
plot(x,To)
    title('Temperature')
    xlabel('X')
    ylabel('T')
figure(2)
plot(x,fluxq,'x')
   title('Flux')
   xlabel('X')
   ylabel('Flux')
================================================================================
```

```
% Finite Difference, 1-D diffusion equation with variable alpha
clear
m=101;
dx=1.0;
dx2=dx*dx;
dt=0.125;
Twall=1.0;
T=zeros(m);To=zeros(m);
x=zeros(m);fluxq=zeros(m);
tk=zeros(m);cpr=zeros(m);
alpha=zeros(m);
x(1)=0.0;
T(1)=Twall; To(1)=Twall;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
for i=1:m
    tk(i)=20.+30.0/(2.*x(i)+1.);
    alpha(i)=tk(i)/100.;
end
nstep=4000;
for kk=1:nstep
   for i=2:m-1
    tke=0.5*(alpha(i+1)+alpha(i));
    tkw=0.5*(alpha(i)+alpha(i-1));
    T(i)=To(i)+dt*(tke*(To(i+1)-To(i))-tkw*(To(i)-To(i-1)))/dx2;
   end
%update
  for k=2:m-1
    To(k)=T(k);
  end
To(m)=T(m-1);
end
 %Flux:
    for k=1:m-1
    fluxq(k)=tk(k)*(To(k)-To(k+1))/dx;
    end
    fluxq(m)=fluxq(m-1);
figure(1)
plot(x,To)
    title('Temperature, nstep=400')
    xlabel('X')
    ylabel('T')
figure(2)
plot(x,fluxq,'x')
   title('Flux, time step=400')
   xlabel('X')
   ylabel('Flux')
```

```
% Finit-Difference 2D, diffusion equation
clear
m=51;n=51;
xl=1.0;yl=1.0;
dx=xl/(m-1.0); dy=yl/(n-1.0);
dx2=dx*dx;
dy2=dy*dy;
dt=0.0001;
alpha=0.25;
```

```
Twall=1.0;
T=zeros(m,n);To=zeros(m,n);Tm=zeros(m);z=zeros(n,m);
x=zeros(m);y=zeros(n);
x(1)=0.0; y(1)=0.0;
%boundary conditions
for i=1:m-1
    x(i+1)=x(i)+dx;
end
for j=1:n-1
    y(j+1)=y(j)+dy;
end
%boundry condition, left vertical boundary
for j=1:m
    T(1,j)=Twall;
    To(1,j)=Twall;
end
nstep=1600;
for kk=1:nstep
 for j=2:n-1
    for i=2:m-1
        xterm=(To(i+1,j)+To(i-1,j))/dx2;
        yterm=(To(i,j+1)+To(i,j-1))/dy2;
        dd=1./dx2+1./dy2;
    T(i,j)=To(i,j)+dt*alpha*(xterm+yterm-2.*To(i,j)*dd);
    end
 end
%update
   for j=2:n-1
     for i=2:m-1
        To(i,j)=T(i,j);
     end
   end
%boundary conditions
   %bottom, adiabatic
   for i=2:m-1
       To(i,1)=To(i,2);
   end
   %right boundary condition T=0
   for j=1:n
       To(m,j)=0.0;
   end
end
for i=1:n
    Tm(i)=To(i,(n-1)/2);
end
%matrix rotation for contour plotting
for j=1:n
for i=1:m
z(j,i)=To(i,j);
end
end
 figure(1)
 plot(x,Tm,'x');
 figure (2)
contour(z)
    title('Temperature')
    xlabel('X')
```

```
    ylabel('Y')
 %  title('Flux')
 %  xlabel('X')
 %  ylabel('Flux')
```

A.2 Chapter Six

```
% Chapter 6
% LBM- D1Q3, BiHarmonic equation, note that c2=1/3, w0=4/6, others 1/6
clear
m=101;
ms=real((m-1)*(m-1))
ms2=ms*ms;
xl=1.0;
dx=xl/(m-1.0);
w0=4./6.;
w=1./6.;
c2=1./3.;
dx=1.0;
f0=zeros(m);f1=zeros(m);f2=zeros(m);
rho=zeros(m);x=zeros(m);gho=zeros(m);an=zeros(m);
Z=zeros(m);
x(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
ss=-1.0/ms;
nstep=100000;
for i=1:m
    f0(i)=w0*rho(i);
    f1(i)=w*rho(i);
    f2(i)=w*rho(i);
end
%Collision:
for k1=1:nstep
   for i=1:m
 %  For variable source term, activate the line below
 %       ss=(sin(pi*i/m))/ms;
        feq0=w0*rho(i);
        feq=w*rho(i);
        f0(i)=feq0+w0*c2*ss/2.0;
        f1(i)=feq+w*c2*ss/2.0;
        f2(i)=feq+w*c2*ss/2.0;
     end

    % Streaming:
    f1(:)=circshift(squeeze(f1(:)),[+1,0]);
    f2(:)=circshift(squeeze(f2(:)),[-1,0]);
    %  OR use the following steps:
  %   for i=1:m-1
 %       f1(m-i+1)=f1(m-i);
 %       f2(i)=f2(i+1);
  %   end
```

```
   %Boundary condition:
   f1(1)=1./12.-f2(1)-f0(1);
   f2(m)=1./12.-f1(m)-f0(m);

   for  i=1:m
       rho(i)=f1(i)+f2(i)+f0(i);
   end

end

%D2w=q
for k1=1:nstep
    for i=1:m
 %  For variable source term, activate the line below
        ss2=-rho(i)/ms;
        geq0=w0*gho(i);
        geq=w*gho(i);
        g0(i)=geq0+w0*c2*ss2/2.0;
        g1(i)=geq+w*c2*ss2/2.0;
        g2(i)=geq+w*c2*ss2/2.0;
     end

    % Streaming:
    g1(:)=circshift(squeeze(g1(:)),[+1,0]);
    g2(:)=circshift(squeeze(g2(:)),[-1,0]);

   %Boundary condition:
  g1(1)=-g2(1)-g0(1);
   g2(m)=-g1(m)-g0(m);

   for  i=1:m
       gho(i)=g1(i)+g2(i)+g0(i);
   end

end

   figure(1)
 plot(x/(m-1),gho)
    xlabel('X')
    ylabel('T')
hold
%analytical solution
for i=1:m
    sx=x(i)/(m-1);
    x4=sx*sx*sx*sx;
    x3=sx*sx*sx;
    x2=sx*sx;
 an(i)=x4/24.-x3/12.+x2/24;
 end
plot(x/(m-1),an,'ks')
================================================================================
% Chapter 6
% LBM- 2-D2Q5, BiHarmonic equation, note that c2=1/3, w0=2/6, others 1/6
clear
m=81;n=81;
ms=real(m-1)*(m-1);
xl=1.0;yl=1.0;
```

```
dx=xl/(m-1.0); dy=yl/(n-1.0);
w0=2./6.;
w=1./6.;
c2=1./3.;
dx=1.0;
pi2=2.*pi;
f0=zeros(m,n);f1=zeros(m,n);f2=zeros(m,n);f3=zeros(m,n);f4=zeros(m,n);
rho=zeros(m,n);x=zeros(m);y=zeros(n);gho=zeros(m,n);
Z=zeros(n,m);
x(1)=0.0; y(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
for j=1:n-1
    y(j+1)=y(j)+dy;
end
ss=1.0/ms;
nstep=10000;
for i=1:m
    f0(i,j)=w0*rho(i,j);
    f1(i,j)=w*rho(i,j);
    f2(i,j)=w*rho(i,j);
    f3(i,j)=w*rho(i,j);
    f4(i,j)=w*rho(i,j);
end
%Collision:
for k1=1:nstep
    for j=1:n
    for i=1:m
 %  For variable source term, activate the line below
       ss=-(sin(pi*i/m)*sin(pi*j/n))/ms;
        feq0=w0*rho(i,j);
        feq=w*rho(i,j);
        f0(i,j)=feq0+w0*ss/6.0;
        f1(i,j)=feq+w*ss/6.0;
        f2(i,j)=feq+w*ss/6.;
        f3(i,j)=feq+w*ss/6.0;
        f4(i,j)=feq+w*ss/6.0;
    end
    end

    % Streaming:
    f1(:,:)=circshift(squeeze(f1(:,:)),[+1,0]);
    f2(:,:)=circshift(squeeze(f2(:,:)),[-1,0]);
    f3(:,:)=circshift(squeeze(f3(:,:)),[0,-1]);
    f4(:,:)=circshift(squeeze(f4(:,:)),[0,+1]);
%  OR use the following steps:
 %   for j=1:n
 %   for i=1:m-1
 %       f1(m-i+1,j)=f1(m-i,j);
 %       f2(i,j)=f2(i+1,j);
 %   end
 %   end
 %   for i=1:m
 %       for j=1:n-1
 %           f3(i,n-j+1)=f3(i,n-j);
 %           f4(i,j)=f4(i,j+1);
 %       end
```

```
 %   end
    %Boundary condition:
   for j=1:n
    f1(1,j)=-f2(1,j)-f0(1,j)-f3(1,j)-f4(1,j);
   f1(m,j)=-f2(m,j)-f0(m,j)-f3(m,j)-f4(m,j);
   end
   for i=1:m
       f4(i,1)=-f0(i,1)-f1(i,1)-f2(i,1)-f3(i,1);
       f3(i,n)=-f0(i,n)-f4(i,n)-f2(i,n)-f1(i,n);
   end

    for j=1:n
   for  i=1:m
       rho(i,j)=f1(i,j)+f2(i,j)+f0(i,j)+f3(i,j)+f4(i,j);
   end
    end
end
%scaling rho

%D2w=q
for k1=1:nstep
    for j=1:n
    for i=1:m
 %  For variable source term, activate the line below
        ss=-rho(i,j)/(ms);
        geq0=w0*gho(i,j);
        geq=w*gho(i,j);
        g0(i,j)=geq0+w0*ss/6.0;
        g1(i,j)=geq+w*ss/6.0;
        g2(i,j)=geq+w*ss/6.;
        g3(i,j)=geq+w*ss/6.0;
        g4(i,j)=geq+w*ss/6.0;
    end
    end

    % Streaming:
    g1(:,:)=circshift(squeeze(g1(:,:)),[+1,0]);
    g2(:,:)=circshift(squeeze(g2(:,:)),[-1,0]);
    g3(:,:)=circshift(squeeze(g3(:,:)),[0,-1]);
    g4(:,:)=circshift(squeeze(g4(:,:)),[0,+1]);
   %Boundary condition:
   for j=1:n
    g1(1,j)=-g2(1,j)-g0(1,j)-g3(1,j)-g4(1,j);
   g1(m,j)=-g2(m,j)-g0(m,j)-g3(m,j)-g4(m,j);
   end
   for i=1:m
       g4(i,1)=-g0(i,1)-g1(i,1)-g2(i,1)-g3(i,1);
       g3(i,n)=-g0(i,n)-g4(i,n)-g2(i,n)-g1(i,n);
   end

    for j=1:n
   for  i=1:m
       gho(i,j)=g1(i,j)+g2(i,j)+g0(i,j)+g3(i,j)+g4(i,j);
   end
    end
end
   for j=1:n
```

```
    for i=1:m
        Z(j,i)=gho(i,j);
    end
end
figure(1)
plot(x/(m-1),gho(:,(n-1)/2))
    xlabel('X')
    ylabel('T')
figure(2)
contour(Z,20)
```

```
================================================================================
```

```
% Chapter 6
% Finit-Difference 1D, Bi-harmonic
clear
m=101;
dx=0.010;
dx4=dx*dx*dx*dx;
ss=5.0;
Twall=0.0;
T=zeros(m);x=zeros(m);Txx=zeros(m);
x(1)=0.0;
T(1)=Twall; To(1)=Twall;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
nstep=10000000;
for kk=1:nstep
    T(1)=0.0; T(2)=0.0;
    T(m)=0.0; T(m-1)=0.0;
  for i=3:m-2
    T(i)=(dx4*ss-(T(i+2)-4.*T(i+1)-4.0*T(i-1)+T(i-2)))/6.0;
  end
end
Txx(1)=2.*(T(2)-T(1))/(dx*dx);
Txx(m)=2.*(T(m-1)-T(m))/(dx*dx);
for  i=2:m-1
Txx(i)=(T(i+1)-2.*T(i)+T(i-1))/(dx*dx);
end
figure(1)
plot(x,T)
    title('Temperature')
    xlabel('X')
    ylabel('T')
```

```
================================================================================
```

```
% Chapter 6
% Finit-Difference 2D, Biharmonic
clear
m=101;n=101;
dx=0.010;
dy=0.010;
dx2=dx*dx;
dy2=dy*dy;
dxdy2=2./dx2+2./dy2;
ss=0.0;
omega=1.80;
```

```
error=5.0;tolr=0.00001;
count=0;
T=zeros(m,n);x=zeros(m);y=zeros(n);Txx=zeros(m,n);To=zeros(m,n);
x(1)=0.0;
y(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
    for j=1:n-1
        y(j+1)=y(j)+dy;
    end
    %boundary conditions
    for j=1:n
        T(1,j)=0.0;
        T(m,j)=0.0;
    end
    for i=1:m
        T(i,1)=0.0;
        T(i,n)=sin(pi*(i-1)/(m-1));
    end
nstep=10000;
while error>tolr
      for i=2:m-1
        for j=2:n-1
          txx=(T(i+1,j)-2.*T(i,j)+T(i-1,j))/dx2;
          tyy=(T(i,j+1)-2.*T(i,j)+T(i,j-1))/dy2;
         T(i,j)=T(i,j)+omega*(txx+tyy-ss)/dxdy2;
        end
      end
  count=count+1;
  error=0.;
  for i=1:m
      for j=1:n
          error=error+abs(T(i,j)-To(i,j));
      end
   end
  To=T;
  count;
end

figure(1)
plot(x,T(:,(n-1)/2))
    title('Temperature')
    xlabel('X')
    ylabel('T')
    for j=1:n
    for i=1:m
        Z(j,i)=T(i,j);
    end
end
figure(2)
contour(Z,20)
================================================================================

% LBM- 2-D2Q5, Laplace equation, note that c2=1/3, w0=2/6, others 1/6
clear
m=101;n=101;
ms=real(m-1)*(m-1);
```

```
xl=1.0;yl=1.0;
dx=xl/(m-1.0); dy=yl/(n-1.0);
w0=2./6.;
w=1./6.;
c2=1./3.;
dx=1.0;
pi2=2.*pi;
f0=zeros(m,n);f1=zeros(m,n);f2=zeros(m,n);f3=zeros(m,n);f4=zeros(m,n);
rho=zeros(m,n);x=zeros(m);y=zeros(n);
Z=zeros(n,m);
x(1)=0.0; y(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
for j=1:n-1
    y(j+1)=y(j)+dy;
end
ss=0.0/ms;
nstep=80000;
for i=1:m
    f0(i,j)=w0*rho(i,j);
    f1(i,j)=w*rho(i,j);
    f2(i,j)=w*rho(i,j);
    f3(i,j)=w*rho(i,j);
    f4(i,j)=w*rho(i,j);
end
%Collision:
for k1=1:nstep
    for j=1:n
    for i=1:m
 %  For variable source term, activate the line below
 %       ss=(rho(i,j)*rho(i,j)+sin(pi2*i/m)*cos(pi*j/n)+0.1)/ms;
        feq0=w0*rho(i,j);
        feq=w*rho(i,j);
        f0(i,j)=feq0+w0*ss/6.0;
        f1(i,j)=feq+w*ss/6.0;
        f2(i,j)=feq+w*ss/6.;
        f3(i,j)=feq+w*ss/6.0;
        f4(i,j)=feq+w*ss/6.0;
    end
    end

    % Streaming:
    f1(:,:)=circshift(squeeze(f1(:,:)),[+1,0]);
    f2(:,:)=circshift(squeeze(f2(:,:)),[-1,0]);
    f3(:,:)=circshift(squeeze(f3(:,:)),[0,-1]);
    f4(:,:)=circshift(squeeze(f4(:,:)),[0,+1]);
%  OR use the following steps:
 %   for j=1:n
 %   for i=1:m-1
 %       f1(m-i+1,j)=f1(m-i,j);
 %       f2(i,j)=f2(i+1,j);
 %   end
 %   end
 %   for i=1:m
 %       for j=1:n-1
 %           f3(i,n-j+1)=f3(i,n-j);
 %           f4(i,j)=f4(i,j+1);
```

```
 %       end
 %   end
    %Boundary condition:
   for j=1:n
    f1(1,j)=-f2(1,j)-f0(1,j)-f3(1,j)-f4(1,j);
   f1(m,j)=-f2(m,j)-f0(m,j)-f3(m,j)-f4(m,j);
   end
   for i=1:m
       f4(i,1)=-f0(i,1)-f1(i,1)-f2(i,1)-f3(i,1);
       f3(i,n)=sin(pi*i/m)-f0(i,n)-f4(i,n)-f2(i,n)-f1(i,n);
   end

    for j=1:n
   for  i=1:m
       rho(i,j)=f1(i,j)+f2(i,j)+f0(i,j)+f3(i,j)+f4(i,j);
   end
    end

end
%rotating matrix for contour plotting
for j=1:n
    for i=1:m
        Z(j,i)=rho(i,j);
    end
end

  figure(1)
plot(x,rho(:,(n-1)/2))
    xlabel('X')
    ylabel('T')
figure(2)
contour(Z,20)

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

LBM- FD, Poission equation with source =0.05, finite-differenc
clear
m=81;n=81;
xl=1.0;yl=1.0;
dx=xl/(m-1); dy=yl/(n-1);
dx2=dx*dx; dy2=dy*dy;
pi2=2.0*pi;
phio=zeros(m,n);x=zeros(m);y=zeros(n);phi=zeros(m,n);
phim=zeros(m);Z=zeros(n,m);
x(1)=0.0; y(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
for j=1:n-1
    y(j+1)=y(j)+dy;
end
cof=2.*(1./dx2+1./dy2);
ss=0.05;
nstep=10000;
for kk=1:nstep
    phio=phi;

 for j=2:n-1
```

```
  for i=2:m-1
      ss=phi(i,j)*phi(i,j)+sin(pi2*x(i))*cos(pi*y(j))+0.1;
     phix=(phi(i+1,j)+phi(i-1,j))/dx2;
     phiy=(phi(i,j+1)+phi(i,j-1))/dy2;
    phi(i,j)=(phix+phiy+ss)/cof;
    end
end
end
   %Boundary condition:
    phi(1,:)=0.0;
    phi(m,:)=0.0;

    phi(:,1)=0.0;
    phi(:,n)=0.0;

  error=norm(phi-phio);
  error
   %rotating matrix for contour plotting
for j=1:n
    for i=1:m
        Z(j,i)=phi(i,j);
    end
end

  for i=1:m
      phim(i)=phi(i,(n-1)/2);
  end
figure(1)
plot(x,phim)
    xlabel('X')
    ylabel('T')
figure(2)
contour(Z,20)
================================================================================
```

A.3 Chapter Seven

```
% FD D1, advection-diffusion equation, Finite Difference
clear
m=201;
tp=zeros(m);tpo=zeros(m);x=zeros(m);
u=0.10;
dx=0.50;
dt=0.25;
alpha=0.25;
mstep=1600;
x(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
twall=1.0;
tp(1)=twall; tpo(1)=twall;
for k1=1:mstep
    for i=2:m-1
        adv=dt*u*(tpo(i)-tpo(i-1))/dx;
```

```
        def=dt*alpha*(tpo(i+1)-2.*tpo(i)+tpo(i-1))/(dx*dx);
        tp(i)=tpo(i)+def-adv;
    end
   tp(1)=twall;
   tp(m)=tp(m-1);
   for i=1:m
        tpo(i)=tp(i);
    end
 end
figure(1)
plot(x,tpo)
    title('Temperature, nstep=400')
    xlabel('X')
    ylabel('T')

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

% LBM- 1-D1Q3, advection-diffusion equation D1Q3
clear
u=0.10;
m=101;
w0=4./6.;
w1=1./6.;
c2=1./3.;
w2=w1;
dx=1.0;
rho=zeros(m);f0=zeros(m);f1=zeros(m);f2=zeros(m);
x=zeros(m);
x(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
alpha=0.25;
omega=1/(alpha+0.5);
twall=1.0;
nstep=400;
for i=1:m
    f0(i)=w0*rho(i);
    f1(i)=w1*rho(i);
    f2(i)=w2*rho(i);
end
%Collision:
for k1=1:nstep
    for i=1:m
        feq0=w0*rho(i);
        feq1=w1*rho(i)*(1.+3.*u);
        feq2=w2*rho(i)*(1.-3.*u);
        f0(i)=(1-omega)*f0(i)+omega*feq0;
        f1(i)=(1-omega)*f1(i)+omega*feq1;
        f2(i)=(1-omega)*f2(i)+omega*feq2;
    end

    % Streaming:
    for i=1:m-1
        f1(m-i+1)=f1(m-i);
        f2(i)=f2(i+1);
    end
    %Boundary condition:
```

```
   f1(1)=twall-f2(1)-f0(1);
   f1(m)=f1(m-1);
   f2(m)=f2(m-1);
   f0(m)=f0(m-1);
   for j=1:m
       rho(j)=f1(j)+f2(j)+f0(j);
   end
end
figure(1)
plot(x,rho)
    title('Temperature, nstep=400')
    xlabel('X')
    ylabel('T')

================================================================================
% LBM- 2-D2Q5a, diffusion equation, note that c2=1/3, w0=2/6, others 1/6
clear
m=101;n=101;
w0=2./6.;
w=1./6.;
c2=1./3.;
dx=1.0;
dy=1.0;
u=0.1; v=0.4;
f0=zeros(m,n);f1=zeros(m,n);f2=zeros(m,n);f3=zeros(m,n);f4=zeros(m,n);
rho=zeros(m,n);x=zeros(m);y=zeros(n);fluxq=zeros(m);flux=zeros(m);
Tm=zeros(m);Z=zeros(n,m);
x(1)=0.0; y(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
for j=1:n-1
    y(j+1)=y(j)+dy;
end
alpha=1.0;
omega=1/(3.*alpha+0.5);
twall=1.0;
nstep=400;
for i=1:m
    f0(i,j)=w0*rho(i,j);
    f1(i,j)=w*rho(i,j);
    f2(i,j)=w*rho(i,j);
    f3(i,j)=w*rho(i,j);
    f4(i,j)=w*rho(i,j);
end
%Collision:
for k1=1:nstep
    for j=1:n
    for i=1:m
        feq0=w0*rho(i,j);
        feq1=w*rho(i,j)*(1.+3.*u);
        feq2=w*rho(i,j)*(1.-3.*u);
        feq3=w*rho(i,j)*(1.+3.*v);
        feq4=w*rho(i,j)*(1.-3.*v);
        f0(i,j)=(1.-omega)*f0(i,j)+omega*feq0;
        f1(i,j)=(1.-omega)*f1(i,j)+omega*feq1;
        f2(i,j)=(1.-omega)*f2(i,j)+omega*feq2;
        f3(i,j)=(1.-omega)*f3(i,j)+omega*feq3;
```

```
        f4(i,j)=(1.-omega)*f4(i,j)+omega*feq4;
    end
    end

    % Streaming:
    for j=1:n
    for i=1:m-1
        f1(m-i+1,j)=f1(m-i,j);
        f2(i,j)=f2(i+1,j);
    end
    end
    for i=1:m
        for j=1:n-1
            f3(i,n-j+1)=f3(i,n-j);
            f4(i,j)=f4(i,j+1);
        end
    end
    %Boundary condition:
   for j=1:n
    f1(1,j)=twall-f2(1,j)-f0(1,j)-f3(1,j)-f4(1,j);
    f2(m,j)=f1(m,j);
   end
   for i=1:m
       f3(i,1)=f4(i,1);
       f4(i,n)=-f0(i,n)-f3(i,n)-f2(i,n)-f1(i,n);
   end

    for j=1:n
   for  i=1:m
       rho(i,j)=f1(i,j)+f2(i,j)+f0(i,j)+f3(i,j)+f4(i,j);
   end
    end
end
%rotating matrix for contour plotting
for j=1:n
    for i=1:m
        Z(j,i)=rho(i,j);
    end
end

  for i=1:n
      Tm(i)=rho(i,(n-1)/2);
  end
figure(1)
plot(x,Tm,'LineWidth',2)
    xlabel('X')
    ylabel('T')
figure(2)
contour(Z,'showText','on','LineWidth',2)
%   title('Flux')
%   xlabel('X')
%   ylabel('Flux')

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

%LBM- 2-D2Q9, diffusion equation, note that c2=1/3, w0=4/9, w1-w4, 1/9
% and w5-w8, 1/36
clear
```

```
m=101;n=101;
xl=1.0;yl=1.0;
w0=4./9.;
c2=1./3.;
dx=1.0;dy=1.0;
f0=zeros(m,n);f=zeros(m,n,8);feq=zeros(m,n,8);f0eq=zeros(m,n);
rho=zeros(m,n);x=zeros(m);y=zeros(n);fluxq=zeros(m);flux=zeros(m);
Tm=zeros(m);Z=zeros(n,m);w(8)=zeros;
x(1)=0.0; y(1)=0.0;
for i=1:m-1
    x(i+1)=x(i)+dx;
end
for j=1:n-1
    y(j+1)=y(j)+dy;
end
for k=1:4
    w(k)=1./9.;
end
for k=5:8
    w(k)=1./36.;
end
alpha=1.00;
u=0.1;v=0.4;
omega=1./(3.*alpha+0.5);
twall=1.0;
nstep=400;

%Collision:
for kk=1:nstep
    for j=1:n
    for i=1:m
        f0eq(i,j)=w0*rho(i,j);
        f0(i,j)=(1.-omega)*f0(i,j)+omega*f0eq(i,j);
        feq(i,j,1)=w(1)*rho(i,j)*(1.+3.*u);
        feq(i,j,2)=w(2)*rho(i,j)*(1.+3.*v);
        feq(i,j,3)=w(3)*rho(i,j)*(1.-3.*u);
        feq(i,j,4)=w(4)*rho(i,j)*(1.-3.*v);
        feq(i,j,5)=w(5)*rho(i,j)*(1.+3.*(u+v));
        feq(i,j,6)=w(6)*rho(i,j)*(1.+3.*(v-u));
        feq(i,j,7)=w(7)*rho(i,j)*(1.-3.*(u+v));
        feq(i,j,8)=w(8)*rho(i,j)*(1.+3.*(u-v));
    for k=1:8
         f(i,j,k)=(1.-omega)*f(i,j,k)+omega*feq(i,j,k);
    end
    end
    end

      % Streaming:
   f(:,:,1)=circshift( squeeze(f(:,:,1)), [+1,+0] );
   f(:,:,2)=circshift( squeeze(f(:,:,2)), [+0,+1] );
   f(:,:,3)=circshift( squeeze(f(:,:,3)), [-1,+0] );
   f(:,:,4)=circshift( squeeze(f(:,:,4)), [+0,-1] );
   f(:,:,5)=circshift( squeeze(f(:,:,5)), [+1,+1] );
   f(:,:,6)=circshift( squeeze(f(:,:,6)), [-1,+1] );
   f(:,:,7)=circshift( squeeze(f(:,:,7)), [-1,-1] );
  f(:,:,8)=circshift( squeeze(f(:,:,8)), [+1,-1] );
    % End of streaming
```

```
    %Boundary condition:
    %left boundary, twall=1.0
   for j=1:n
       f(1,j,1)=w(1)*twall+w(3)*twall-f(1,j,3);
       f(1,j,5)=w(5)*twall+w(7)*twall-f(1,j,7);
       f(1,j,8)=w(8)*twall+w(6)*twall-f(1,j,6);
   end
   %bottom boundary, T=0.0
   for i=1:m
       f(i,1,2)=-f(i,1,4);
       f(i,1,5)=-f(i,1,7);
       f(i,1,6)=-f(i,1,8);
   end
   %Top boundary, T=0.0
     for i=1:n
     f(i,m,7)=-f(i,m,5);
     f(i,m,4)=-f(i,m,2);
     f(i,m,8)=-f(i,m,6);
     end
     %right hand boundary
     for j=1:m
       f(n,j,3)=-f(n,j,1);
       f(n,j,7)=-f(n,j,5);
       f(n,j,6)=-f(n,j,8);
     end

% End of boundary conditions
    for j=1:n
   for  i=1:m
       sumk=0.0;
       for k=1:8
           sumk=sumk+f(i,j,k);
       end
        rho(i,j)=f0(i,j)+sumk;
   end
    end
end
%rotating matrix for contour plotting
for j=1:n
    for i=1:m
        Z(j,i)=rho(i,j);
    end
end

  for i=1:n
      Tm(i)=rho(i,(n-1)/2);
  end
figure(1)
plot(x,Tm, 'LineWidth',2)
    xlabel('X')
    ylabel('T')
figure(2)
contour(Z,'showText','on','LineWidth',2)
%   title('Flux')
%   xlabel('X')
%   ylabel('Flux')

================================================================================
```

A.4 Chapter Eight

Back Flow-Step

```
% Main.m
%LBM- 2-D2Q9, Flow in a channel with step, note that c2=1/3, w9=4/9,
% w1-4=1/9, and w5-w8, 1/36
clear
nx=501;ny=81;
f=zeros(nx,ny,9);feq=zeros(nx,ny,9);
u=zeros(nx,ny);v=zeros(nx,ny);
rho=ones(nx,ny);x=zeros(nx);y=zeros(ny);
Tm=zeros(nx);w(9)=zeros; Tvm=zeros(nx);
w=[1/9 1/9 1/9 1/9 1/36 1/36 1/36 1/36  4/9];
cx = [1 0 -1  0 1 -1 -1  1 0];
cy = [0 1  0 -1 1  1 -1 -1 0];
c2=1./3.;
dx=1.0;dy=1.0;
xl=(nx-1)/(ny-1); yl=1.0;
dx=xl/(nx-1);
dy=yl/(ny-1);
x=(0:dx:xl);
y=(0:dy:yl);
uo=0.1;
alpha=0.01;
Re=uo*(ny-1)/alpha
omega=1./(3.*alpha+0.5);
count=0; tol=1.0e-4; error=10.;erso=0.0;
%setting velocity
    for j=2:ny-1
       u(1,j)=uo;
    end
%Main Loop
 while error>tol
  % Collitions
  [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w);
     % Streaming:
     [f]=stream(f);
    % End of streaming
    %Boundary condition:
    [f]=boundary(nx,ny,f,uo,rho);
    %Obsticale
    [f]=obstc(nx,ny,f,uo,rho);
% Calculate rho, u, v
 [rho,u,v]=ruv(nx,ny,f);
  count=count+1;
   ers=0.;
   for i =1:nx
   for j=1:ny
       ers=ers+u(i,j)*u(i,j)+v(i,j)*v(i,j);
   end
   end
    error=abs(ers-erso);
    erso=ers;
   end
%Plotting data
result(nx,ny,x,y,u,v,uo,rho);
```

```
++++++++++++++++++++++++++++++++
%Boudary conditions for Channel flow
function [f]=boundary(nx,ny,f,uo,rho)
  %right hand boundary
   for j=1:ny
      f(nx,j,3)=f(nx-1,j,3);
      f(nx,j,7)=f(nx-1,j,7);
      f(nx,j,6)=f(nx-1,j,6);
     end
   %bottom, and top boundary, bounce back
   for i=1:nx
       f(i,1,2)=f(i,1,4);
       f(i,1,5)=f(i,1,7);
       f(i,1,6)=f(i,1,8);
       f(i,ny,4)=f(i,ny,2);
       f(i,ny,7)=f(i,ny,5);
       f(i,ny,8)=f(i,ny,6);
       u(i,1)=0.0; v(i,1)=0.0;
       u(i,ny)=0.0; v(i,ny)=0.0;
   end
   %Left boundary, velocity is given= uo
     for j=2:ny-1
     f(1,j,1)=f(1,j,3)+2.*rho(1,j)*uo/3.;
     f(1,j,5)=f(1,j,7)-0.5*(f(1,j,2)-f(1,j,4))+rho(1,j)*uo/6.;
     f(1,j,8)=f(1,j,6)+0.5*(f(1,j,2)-f(1,j,4))+rho(1,j)*uo/6.;
     u(1,j)=uo; v(1,j)=0.0;
     end
 % End of boundary conditions.
end
++++++++++++++++++++++++++++++++++++++++++++
% Collition
function [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w)
for j=1:ny
     for i=1:nx
       t1=u(i,j)*u(i,j)+v(i,j)*v(i,j);
       for k=1:9
          t2=u(i,j)*cx(k)+v(i,j)*cy(k);
         feq(i,j,k)=rho(i,j)*w(k)*(1.0+3.0*t2+4.5*t2*t2-1.5*t1);
         f(i,j,k)=(1.-omega)*f(i,j,k)+omega*feq(i,j,k);
       end
     end
end
end
++++++++++++++++++++++++++++++++
%Obsticale replace at the entrance, Back Fase Flow
function [f]=obstc(nx,ny,f,uo,rho)
  %length of obsticale= nx/4, heigh ny/2
  nxo=(nx-1)/4;
  nyl=(ny-1)/2;
  for i=1:nxo
      f(i,nyl,2)=f(i,nyl,4);
      f(i,nyl,5)=f(i,nyl,7);
      f(i,nyl,6)=f(i,nyl,8);
     end
   %bottom, and top boundary, bounce back
   for j=1:nyl
       f(nxo,j,1)=f(nxo,j,3);
```

```
       f(nxo,j,5)=f(nxo,j,7);
       f(nxo,j,8)=f(nxo,j,8);
   end
   for i=1:nxo
       for j=1:nyl
           u(i,j)=0.0;
           v(i,j)=0.0;
       end
   end

    % End
end
++++++++++++++++++++++++++++++++++++++++
% Plots for channel flow
function result(nx,ny,x,y,u,v,uo,rho)
nxo=(nx-1)/4;
nyl=(ny-1)/2;
for i=1:nxo
    for j=1:nyl
        u(i,j)=0.0;
        v(i,j)=0.0;
    end
end
for j=1:ny
      Tm1(j)=u(51,j)/uo;
      Tm2(j)=u(101,j)/uo;
      Tm3(j)=u(261,j)/uo;
      Tm4(j)=u(301,j)/uo;
      Tm5(j)=u(501,j)/uo;
  end
figure
 plot(Tm1,y,Tm2,y,Tm3,y,Tm4,y,Tm5,y,'LineWidth',1.5)
    xlabel('U')
    ylabel('Y')
    %Stream function calculation
    for j=1:ny
        sx(:,j)=x(:);
    end
    for i=1:nx
        sy(i,:)=y(:);
    end
str=zeros(nx,ny);
for i=1:nx
for j=2:ny
   str(i,j)=str(i,j-1)+0.5*(u(i,j)+u(i,j-1));

end
end
figure
contour(sx,sy,str)
 end
++++++++++++++++++++++++++++++++++
function[rho,u,v]=ruv(nx,ny,f)

rho=sum (f,3);
  for i=1:nx
rho(i,ny)=f(i,ny,9)+f(i,ny,1)+f(i,ny,3)+2.*(f(i,ny,2)+f(i,ny,6)+f(i,ny,5));
   end
```

```
   %calculate velocity compnents
    u = ( sum(f(:,:,[1 5 8]),3) - sum(f(:,:,[3 6 7]),3) )./rho;
   v = ( sum(f(:,:,[2 5 6]),3) - sum(f(:,:,[4 7 8]),3) )./rho;

end
++++++++++++++++++++++++++++++++++++
% Streaming:
   function [f]=stream(f)
   f(:,:,1)=circshift( squeeze(f(:,:,1)), [+1,+0] );
   f(:,:,2)=circshift( squeeze(f(:,:,2)), [+0,+1] );
   f(:,:,3)=circshift( squeeze(f(:,:,3)), [-1,+0] );
   f(:,:,4)=circshift( squeeze(f(:,:,4)), [+0,-1] );
   f(:,:,5)=circshift( squeeze(f(:,:,5)), [+1,+1] );
   f(:,:,6)=circshift( squeeze(f(:,:,6)), [-1,+1] );
   f(:,:,7)=circshift( squeeze(f(:,:,7)), [-1,-1] );
   f(:,:,8)=circshift( squeeze(f(:,:,8)), [+1,-1] );
   end
    % End of streaming
================================================================================
```

Channel Flow

```
%Main.m
%LBM- 2-D2Q9, channel, Re=400, note that c2=1/3, w9=4/9,
% w1-4=1/9, and w5-w8, 1/36
clear
nx=1001;ny=41;
f=zeros(nx,ny,9);feq=zeros(nx,ny,9);
u=zeros(nx,ny);v=zeros(nx,ny);
rho=ones(nx,ny);x=zeros(nx);y=zeros(ny);
Tm=zeros(nx);w(9)=zeros; Tvm=zeros(nx);
w=[1/9 1/9 1/9 1/9 1/36 1/36 1/36 1/36  4/9];
cx = [1 0 -1  0 1 -1 -1  1 0];
cy = [0 1  0 -1 1  1 -1 -1 0];
c2=1./3.;
dx=1.0;dy=1.0;
xl=(nx-1)/(ny-1); yl=1.0;
dx=xl/(nx-1);
dy=yl/(ny-1);
x=(0:dx:xl);
y=(0:dy:yl);
uo=0.2;
alpha=0.02;
Re=uo*(ny-1)/alpha
omega=1./(3.*alpha+0.5);
count=0; tol=1.0e-4; error=10.;erso=0.0;
%setting velocity
    for j=2:ny-1
       u(1,j)=uo;
    end
%Main Loop
 while error>tol
  % Collitions
  [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w);
     % Streaming:
     [f]=stream(f);
    % End of streaming
    %Boundary condition:
```

```
    [f]=boundary(nx,ny,f,uo,rho);
   % Calculate rho, u, v
 [rho,u,v]=ruv(nx,ny,f);
  count=count+1;
   ers=0.;
   for i =1:nx
   for j=1:ny
       ers=ers+u(i,j)*u(i,j)+v(i,j)*v(i,j);
   end
   end
    error=abs(ers-erso);
    erso=ers;
   end
%Plotting data
result(nx,ny,x,y,u,v,uo,rho);
++++++++++++++++++++++++
 %Boudary conditions for Channel flow
function [f]=boundary(nx,ny,f,uo,rho)
  %right hand boundary
   for j=1:ny
      f(nx,j,3)=f(nx-1,j,3);
      f(nx,j,7)=f(nx-1,j,7);
      f(nx,j,6)=f(nx-1,j,6);
     end
   %bottom, and top boundary, bounce back
   for i=1:nx
       f(i,1,2)=f(i,1,4);
       f(i,1,5)=f(i,1,7);
       f(i,1,6)=f(i,1,8);
       f(i,ny,4)=f(i,ny,2);
       f(i,ny,7)=f(i,ny,5);
       f(i,ny,8)=f(i,ny,6);
       u(i,1)=0.0; v(i,1)=0.0;
       u(i,ny)=0.0; v(i,ny)=0.0;
   end
   %Left boundary, velocity is given= uo
     for j=2:ny-1
     f(1,j,1)=f(1,j,3)+2.*rho(1,j)*uo/3.;
     f(1,j,5)=f(1,j,7)-0.5*(f(1,j,2)-f(1,j,4))+rho(1,j)*uo/6.;
     f(1,j,8)=f(1,j,6)+0.5*(f(1,j,2)-f(1,j,4))+rho(1,j)*uo/6.;
     u(1,j)=uo; v(1,j)=0.0;
     end
 % End of boundary conditions.
end
++++++++++++++++++++++++++++++++++
% Collition
function [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w)
for j=1:ny
     for i=1:nx
       t1=u(i,j)*u(i,j)+v(i,j)*v(i,j);
       for k=1:9
          t2=u(i,j)*cx(k)+v(i,j)*cy(k);
         feq(i,j,k)=rho(i,j)*w(k)*(1.0+3.0*t2+4.5*t2*t2-1.5*t1);
         f(i,j,k)=(1.-omega)*f(i,j,k)+omega*feq(i,j,k);
       end
     end
end
```

```
end
++++++++++++++++++++++++++++++++
% Plots for channel flow
function result(nx,ny,x,y,u,v,uo,rho)
for j=1:ny
      Tm1(j)=u(101,j)/uo;
      Tm2(j)=u(201,j)/uo;
      Tm3(j)=u(301,j)/uo;
      Tm4(j)=u(401,j)/uo;
      Tm5(j)=u(601,j)/uo;
  end
figure
 plot(Tm1,y,Tm2,y,Tm3,y,Tm4,y,Tm5,y,'LineWidth',1.5)
    xlabel('U')
    ylabel('Y')
    %Stream function calculation
    for j=1:ny
        sx(:,j)=x(:);
    end
    for i=1:nx
        sy(i,:)=y(:);
    end
str=zeros(nx,ny);
for i=1:nx
for j=2:ny
   str(i,j)=str(i,j-1)+0.5*(u(i,j)+u(i,j-1));

end
end
figure
contour(sx,sy,str)
 end
++++++++++++++++++++++++++++++
function[rho,u,v]=ruv(nx,ny,f)
rho=sum (f,3);
  for i=1:nx
rho(i,ny)=f(i,ny,9)+f(i,ny,1)+f(i,ny,3)+2.*(f(i,ny,2)+f(i,ny,6)+f(i,ny,5));
   end
   %calculate velocity compnents
    u = ( sum(f(:,:,[1 5 8]),3) - sum(f(:,:,[3 6 7]),3) )./rho;
   v = ( sum(f(:,:,[2 5 6]),3) - sum(f(:,:,[4 7 8]),3) )./rho;
end
++++++++++++++++++++++++++++++++++
% Streaming:
   function [f]=stream(f)
   f(:,:,1)=circshift( squeeze(f(:,:,1)), [+1,+0] );
   f(:,:,2)=circshift( squeeze(f(:,:,2)), [+0,+1] );
   f(:,:,3)=circshift( squeeze(f(:,:,3)), [-1,+0] );
   f(:,:,4)=circshift( squeeze(f(:,:,4)), [+0,-1] );
   f(:,:,5)=circshift( squeeze(f(:,:,5)), [+1,+1] );
   f(:,:,6)=circshift( squeeze(f(:,:,6)), [-1,+1] );
   f(:,:,7)=circshift( squeeze(f(:,:,7)), [-1,-1] );
   f(:,:,8)=circshift( squeeze(f(:,:,8)), [+1,-1] );
   end
    % End of streaming
================================================================================
```

Lid-Driven Cavity

```
%main.m
%LBM- 2-D2Q9, Lid-driven Cavity, Re=1000, note that c2=1/3, w9=4/9,
% w1-4=1/9, and w5-w8, 1/36
clear
nx=101;ny=101;
f=zeros(nx,ny,9);feq=zeros(nx,ny,9);
u=zeros(nx,ny);v=zeros(nx,ny);
rho=ones(nx,ny);x=zeros(nx);y=zeros(ny);
w=[1/9 1/9 1/9 1/9 1/36 1/36 1/36 1/36  4/9];
cx = [1 0 -1  0 1 -1 -1  1 0];
cy = [0 1  0 -1 1  1 -1 -1 0];
c2=1./3.;
dx=1.0;dy=1.0;
xl=1.0; yl=1.0;
dx=xl/(nx-1);
dy=yl/(ny-1);
x=(0:dx:xl);
y=(0:dy:yl);
uo=0.10;
alpha=0.01;
Re=uo*(ny-1)/alpha
omega=1./(3.*alpha+0.5);
count=0; tol=1.0e-4; error=10.;erso=0.0;
%setting lid velocity
    u(:,ny)=uo;
%Main Loop
 while error>tol
  % Collitions
  [f]=collision(nx,ny,u,v,cx,cy,omega,f,rho,w);
  % Streaming:
     [f]=stream(f);
    % End of streaming
    %Boundary condition:
    [f]=boundary(nx,ny,f,uo);
% Calculate rho, u, v
 [rho,u,v]=ruv(nx,ny,f);
  count=count+1;
   ers=0.;
   for i =1:nx
   for j=1:ny
       ers=ers+u(i,j)*u(i,j)+v(i,j)*v(i,j);
   end
   end
    error=abs(ers-erso);
    erso=ers;
   end
%Plotting data
result(nx,ny,x,y,u,v,uo,rho);

++++++++++++++++++++++++++++++++++++++++++++
function [f]=boundary(nx,ny,f,uo)
%Boundary condition:
    %left boundary, bounc back
       f(1,:,1)=f(1,:,3);
       f(1,:,5)=f(1,:,7);
```

```
        f(1,:,8)=f(1,:,6);
    %right hand boundary
        f(nx,:,3)=f(nx,:,1);
        f(nx,:,7)=f(nx,:,5);
        f(nx,:,6)=f(nx,:,8);
    %bottom boundary, bounce back
        f(:,1,2)=f(:,1,4);
        f(:,1,5)=f(:,1,7);
        f(:,1,6)=f(:,1,8);
    %Top boundary,moving lid with uo
      for i=2:nx-1
      rhon=f(i,ny,9)+f(i,ny,1)+f(i,ny,3)+2.*(f(i,ny,2)+f(i,ny,6)+f(i,ny,5));
      f(i,ny,4)=f(i,ny,2);
      f(i,ny,8)=f(i,ny,6)+rhon*uo/6.0;
      f(i,ny,7)=f(i,ny,5)-rhon*uo/6.0;
      end
end
++++++++++++++++++++++++++++++++++++++++++++++++++

% Collition
function [f]=collision(nx,ny,u,v,cx,cy,omega,f,rho,w)
for j=1:ny
     for i=1:nx
       t1=u(i,j)*u(i,j)+v(i,j)*v(i,j);
       for k=1:9
          t2=u(i,j)*cx(k)+v(i,j)*cy(k);
         feq(i,j,k)=rho(i,j)*w(k)*(1.0+3.0*t2+4.5*t2*t2-1.5*t1);
         f(i,j,k)=(1.-omega)*f(i,j,k)+omega*feq(i,j,k);
       end
     end
end
end

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
function result(nx,ny,x,y,u,v,uo,rho)
for j=1:ny
      um(j)=u((nx-1)/2,j)/uo;
 end
  for i=1:nx
  vm(i)=v(i,(ny-1)/2)/uo;
  end
 figure
  line(um,y,'LineWidth',1.5)
    xlabel('U')
    ylabel('Y')
  figure
   line(x,vm,'LineWidth',1.5)
   xlabel('X')
   ylabel('V')
figure
quiver(x,y,v/uo,u/uo, 10)
%Stream function calculation
str=zeros(nx,ny);
for i=1:nx
for j=2:ny
   str(i,j)=str(i,j-1)+0.25*(rho(i,j)+rho(i,j-1))*(u(i,j)+u(i,j-1));
end
end
```

```
figure
contour(x,y,str,20)
%or use the following build-in function
figure
startx=0.0:0.1:1;
starty=0.0:0.1:1;
streamline(x,y,v,u,startx,starty)
end
++++++++++++++++++++++++++++++++++++++++++++++++++
function[rho,u,v]=ruv(nx,ny,f)
rho=sum (f,3);
  for i=1:nx
rho(i,ny)=f(i,ny,9)+f(i,ny,1)+f(i,ny,3)+2.*(f(i,ny,2)+f(i,ny,6)+f(i,ny,5));
   end
   %calculate velocity compnents
    u = ( sum(f(:,:,[1 5 8]),3) - sum(f(:,:,[3 6 7]),3) )./rho;
   v = ( sum(f(:,:,[2 5 6]),3) - sum(f(:,:,[4 7 8]),3) )./rho;
end
++++++++++++++++++++++++++++++++++++++++++++++++++
% Streaming:
   function [f]=stream(f)
   f(:,:,1)=circshift( squeeze(f(:,:,1)), [+1,+0] );
   f(:,:,2)=circshift( squeeze(f(:,:,2)), [+0,+1] );
   f(:,:,3)=circshift( squeeze(f(:,:,3)), [-1,+0] );
   f(:,:,4)=circshift( squeeze(f(:,:,4)), [+0,-1] );
   f(:,:,5)=circshift( squeeze(f(:,:,5)), [+1,+1] );
   f(:,:,6)=circshift( squeeze(f(:,:,6)), [-1,+1] );
   f(:,:,7)=circshift( squeeze(f(:,:,7)), [-1,-1] );
   f(:,:,8)=circshift( squeeze(f(:,:,8)), [+1,-1] );
   end
    % End of streaming
================================================================================
```

Flow over an obstacle

```
 %main.m
%LBM- 2-D2Q9, flow over an obstacle, Re=400, note that c2=1/3, w9=4/9,
% w1-4=1/9, and w5-w8, 1/36
clear
nx=501;ny=81;
uo=0.1;
f=zeros(nx,ny,9);feq=zeros(nx,ny,9);utim=zeros(1001);count=zeros(1001);
u=uo*ones(nx,ny);v=zeros(nx,ny);
rho=2.*ones(nx,ny);x=zeros(nx);y=zeros(ny);
Tm=zeros(nx);w(9)=zeros; Tvm=zeros(nx);
w=[1/9 1/9 1/9 1/9 1/36 1/36 1/36 1/36  4/9];
cx = [1 0 -1  0 1 -1 -1  1 0];
cy = [0 1  0 -1 1  1 -1 -1 0];
c2=1./3.;
dx=1.0;dy=1.0;
xl=(nx-1)/(ny-1); yl=1.0;
x=(0:1:nx-1);
y=(0:1:ny-1);
alpha=0.01;
ReH=uo*(ny-1)/alpha
ReD=uo*10./alpha
omega=1./(3.*alpha+0.5);
count(1)=0;
```

```
%setting velocity
    for j=2:ny-1
       u(1,j)=uo;
    end
%Main Loop
 for kk=1:8000
  % Collitions
  [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w);
     % Streaming:
     [f]=stream(f);
    % End of streaming
    %Boundary condition:
    [f]=boundary(nx,ny,f,uo,rho);
    %Obsticale
    [f]=obstc(nx,ny,f,uo,rho);
% Calculate rho, u, v
 [rho,u,v]=ruv(nx,ny,f);
  count(kk)=kk;
  utim(kk)=rho((nx-1)/2,(ny-1)/2);
 end
%Plotting data
result(nx,ny,x,y,u,v,uo,rho,count,utim);

++++++++++++++++++++++++++++++++++++++++++++++++

%Boudary conditions for Channel flow
function [f]=boundary(nx,ny,f,uo,rho)
  %right hand boundary
   for j=1:ny
      f(nx,j,3)=f(nx-1,j,3);
      f(nx,j,7)=f(nx-1,j,7);
      f(nx,j,6)=f(nx-1,j,6);
     end
   %bottom, and top boundary, bounce back
   for i=1:nx
       f(i,1,2)=f(i,1,4);
       f(i,1,5)=f(i,1,7);
       f(i,1,6)=f(i,1,8);
       f(i,ny,4)=f(i,ny,2);
       f(i,ny,7)=f(i,ny,5);
       f(i,ny,8)=f(i,ny,6);
       u(i,1)=0.0; v(i,1)=0.0;
       u(i,ny)=0.0; v(i,ny)=0.0;
   end
   %Left boundary, velocity is given= uo
     for j=2:ny-1
     f(1,j,1)=f(1,j,3)+2.*rho(1,j)*uo/3.;
     f(1,j,5)=f(1,j,7)-0.5*(f(1,j,2)-f(1,j,4))+rho(1,j)*uo/6.;
     f(1,j,8)=f(1,j,6)+0.5*(f(1,j,2)-f(1,j,4))+rho(1,j)*uo/6.;
     u(1,j)=uo; v(1,j)=0.0;
     end
 % End of boundary conditions.
end
++++++++++++++++++++++++++++++++++++
% Collition
function [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w)
for j=1:ny
```

```
     for i=1:nx
       t1=u(i,j)*u(i,j)+v(i,j)*v(i,j);
       for k=1:9
          t2=u(i,j)*cx(k)+v(i,j)*cy(k);
         feq(i,j,k)=rho(i,j)*w(k)*(1.0+3.0*t2+4.5*t2*t2-1.5*t1);
         f(i,j,k)=(1.-omega)*f(i,j,k)+omega*feq(i,j,k);
       end
     end
end
end
++++++++++++++++++++++++++++++++++++++++++++
%Obsticale replace at the entrance, Back Fase Flow
function [f]=obstc(nx,ny,f,uo,rho)
  %length of obsticale= nx/5, and has sides of 10 units
  nxb=(nx-1)/5;
  nxe=nxb+10;
  nyb=((ny-1)-10)/2;
  nyb=35;
  nye=nyb+10;
  for i=nxb:nxe
      f(i,nyb,4)=f(i,nyb,2);
      f(i,nyb,7)=f(i,nyb,5);
      f(i,nyb,8)=f(i,nyb,6);
      f(i,nye,2)=f(i,nye,4);
      f(i,nye,5)=f(i,nye,7);
      f(i,nye,6)=f(i,nye,8);
     end
   %bottom, and top boundary, bounce back
   for j=nyb:nye
       f(nxb,j,3)=f(nxb,j,1);
       f(nxb,j,7)=f(nxb,j,5);
       f(nxb,j,6)=f(nxb,j,8);
       f(nxe,j,1)=f(nxe,j,3);
       f(nxe,j,5)=f(nxe,j,7);
       f(nxe,j,8)=f(nxe,j,8);
   end
   for i=nxb:nxe
       for j=nyb:nye
           u(i,j)=0.0;
           v(i,j)=0.0;
       end
   end

    % End
end
++++++++++++++++++++++++++++++++++++++++++++++++++++
% Plots for channel flow
function result(nx,ny,x,y,u,v,uo,rho,count,utim)
for j=1:ny
      Tm1(j)=u(51,j)/uo;
      Tm2(j)=u(101,j)/uo;
      Tm3(j)=u(261,j)/uo;
      Tm4(j)=u(301,j)/uo;
    end
   for i=1:nx
  umx(i)=u(i,(ny-1)/2)/uo;
  vmx(i)=v(i,(ny-1)/2)/uo;
   end
```

```
   figure
   plot(x/(nx-1),umx,x/(nx-1),vmx,'LineWidth',1.5)
   figure
 plot(Tm1,y,Tm2,y,Tm3,y,Tm4,y,'LineWidth',1.5)
    xlabel('U')
    ylabel('Y')
    figure
    plot(count,utim)

    %Stream function calculation
    for j=1:ny
        sx(:,j)=x(:);
    end
    for i=1:nx
        sy(i,:)=y(:);
    end
str=zeros(nx,ny);
for i=1:nx
for j=2:ny
   str(i,j)=str(i,j-1)+0.5*(u(i,j)+u(i,j-1));

end
end
figure
contour(sx,sy,str)
figure
contour(sx,sy,u,'LineWidth',1.0)
 end
++++++++++++++++++++++++++++++++++++++++++++
function[rho,u,v]=ruv(nx,ny,f)

rho=sum (f,3);
  for i=1:nx
rho(i,ny)=f(i,ny,9)+f(i,ny,1)+f(i,ny,3)+2.*(f(i,ny,2)+f(i,ny,6)+f(i,ny,5));
   end
   %calculate velocity compnents
    u = ( sum(f(:,:,[1 5 8]),3) - sum(f(:,:,[3 6 7]),3) )./rho;
   v = ( sum(f(:,:,[2 5 6]),3) - sum(f(:,:,[4 7 8]),3) )./rho;

end
++++++++++++++++++++++++++++++++++++++++++++++++++
% Streaming:
   function [f]=stream(f)
   f(:,:,1)=circshift( squeeze(f(:,:,1)), [+1,+0] );
   f(:,:,2)=circshift( squeeze(f(:,:,2)), [+0,+1] );
   f(:,:,3)=circshift( squeeze(f(:,:,3)), [-1,+0] );
   f(:,:,4)=circshift( squeeze(f(:,:,4)), [+0,-1] );
   f(:,:,5)=circshift( squeeze(f(:,:,5)), [+1,+1] );
   f(:,:,6)=circshift( squeeze(f(:,:,6)), [-1,+1] );
   f(:,:,7)=circshift( squeeze(f(:,:,7)), [-1,-1] );
   f(:,:,8)=circshift( squeeze(f(:,:,8)), [+1,-1] );
   end
    % End of streaming
================================================================================
```

A.5 Chapter Nine

Heated Lid-Driven Cavity

```
%LBM- 2-D2Q9, Heated Lid-driven Cavity, Re=1000, note that c2=1/3, w9=4/9,
% w1-4=1/9, and w5-w8, 1/36
clear
nx=101;ny=101;
f=zeros(nx,ny,9);g=zeros(nx,ny,9);rhog=zeros(nx,ny);
u=zeros(nx,ny);v=zeros(nx,ny);
rho=ones(nx,ny);x=zeros(nx);y=zeros(ny);
w=[1/9 1/9 1/9 1/9 1/36 1/36 1/36 1/36  4/9];
cx = [1 0 -1  0 1 -1 -1  1 0];
cy = [0 1  0 -1 1  1 -1 -1 0];
c2=1./3.;
dx=1.0;dy=1.0;
xl=1.0; yl=1.0;
dx=xl/(nx-1);
dy=yl/(ny-1);
x=(0:dx:xl);
y=(0:dy:yl);
uo=0.10;
alpha=0.01;
Re=uo*(ny-1)/alpha
omega=1./(3.*alpha+0.5);
pr=0.71;
alphag=alpha/pr;
omegag=1.0/(3.*alphag+0.5);
count=0; tol=1.0e-5; error=10.;erso=0.0;
%setting lid velocity to uo
    u(:,ny)=uo;
%Main Loop
 while error>tol
  % Collitions
  [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w);
  % Streaming:
     [f]=stream(f);
    % End of streaming
    %Boundary condition:
    [f]=boundary(nx,ny,f,uo);
% Calculate rho, u, v
 [rho,u,v]=ruv(nx,ny,f);
 %Temperature
  [g]=gcol(nx,ny,u,v,cx,cy,omegag,g,rhog,w);   %collision
  [g]=stream(g);     %streaming
  [g]=gbound(nx,ny,w,g);    % boundary conditions

  rhog=sum(g,3);
  count=count+1;
   ers=0.;
   for i =1:nx
   for j=1:ny
       ers=ers+u(i,j)*u(i,j)+v(i,j)*v(i,j);
   end
   end
    error=abs(ers-erso);
    erso=ers;
```

```
   end
%Plotting data
result(nx,ny,x,y,u,v,uo,rho,rhog);

++++++++++++++++++++++++++++++++++++++++
function [f]=boundary(nx,ny,f,uo)
%Boundary condition:
    %left boundary, bounc back
       f(1,:,1)=f(1,:,3);
       f(1,:,5)=f(1,:,7);
       f(1,:,8)=f(1,:,6);
   %right hand boundary
       f(nx,:,3)=f(nx,:,1);
       f(nx,:,7)=f(nx,:,5);
       f(nx,:,6)=f(nx,:,8);
   %bottom boundary, bounce back
       f(:,1,2)=f(:,1,4);
       f(:,1,5)=f(:,1,7);
       f(:,1,6)=f(:,1,8);
   %Top boundary,moving lid with uo
     for i=2:nx-1
     rhon=f(i,ny,9)+f(i,ny,1)+f(i,ny,3)+2.*(f(i,ny,2)+f(i,ny,6)+f(i,ny,5));
     f(i,ny,4)=f(i,ny,2);
     f(i,ny,8)=f(i,ny,6)+rhon*uo/6.0;
     f(i,ny,7)=f(i,ny,5)-rhon*uo/6.0;
     end
end
++++++++++++++++++++++++++++++++++++++++++++
% Collition
function [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w)
for j=1:ny
     for i=1:nx
       t1=u(i,j)*u(i,j)+v(i,j)*v(i,j);
       for k=1:9
          t2=u(i,j)*cx(k)+v(i,j)*cy(k);
         feq(i,j,k)=rho(i,j)*w(k)*(1.0+3.0*t2+4.5*t2*t2-1.5*t1);
         f(i,j,k)=(1.-omega)*f(i,j,k)+omega*feq(i,j,k);
       end
     end
end
end

++++++++++++++++++++++++++++++++++++++++

function [g]=gbound(nx,ny,w,g)
%Boundary condition:
      twall=1.0;    %left hand all temperature
    %left boundary, bounc back
       g(1,:,1)=-g(1,:,3);
       g(1,:,5)=-g(1,:,7);
       g(1,:,8)=-g(1,:,6);
   %right hand boundary
       g(nx,:,3)=-g(nx,:,1);
       g(nx,:,7)=-g(nx,:,5);
       g(nx,:,6)=-g(nx,:,8);
   %bottom boundary, bounce back
       g(:,1,2)=g(:,1,4);
       g(:,1,5)=g(:,1,7);
```

```
       g(:,1,6)=g(:,1,8);
 %      g(:,1,1)=g(:,2,1);
 %      g(:,1,3)=g(:,2,3);
 %      g(:,1,4)=g(:,2,4);
       g(:,1,8)=g(:,2,8);
  %Top boundary,moving lid with uo
     g(:,ny,4)=(w(4)+w(2))*twall-g(:,ny,2);
     g(:,ny,8)=(w(8)+w(6))*twall-g(:,ny,6);
     g(:,ny,7)=(w(7)+w(5))*twall-g(:,ny,5);


end
++++++++++++++++++++++++++++++++++++++++++++++++
% Collition temperature
function [g]=gcol(nx,ny,u,v,cx,cy,omegag,g,rhog,w)
for j=1:ny
     for i=1:nx
       t1=u(i,j)*u(i,j)+v(i,j)*v(i,j);
       for k=1:9
          t2=u(i,j)*cx(k)+v(i,j)*cy(k);
         geq(i,j,k)=rhog(i,j)*w(k)*(1.0+3.0*t2);
         g(i,j,k)=(1.-omegag)*g(i,j,k)+omegag*geq(i,j,k);
       end
     end
end
end
++++++++++++++++++++++++++++++++
function result(nx,ny,x,y,u,v,uo,rho,rhog)
for j=1:ny
  for i=1:nx
      xp(i,j)=x(i);
      yp(i,j)=y(j);
  end
end
for j=1:ny
      uvm(j)=u((nx-1)/2,j)/uo;
      tmj(j)=rhog((nx-1)/2,j);
 end
  for i=1:nx
  vvm(i)=v(i,(ny-1)/2)/uo;
  end
  figure
  plot(tmj,y,'LineWidth',2)
figure
 plot(uvm,y,'LineWidth',2)
    xlabel('U')
    ylabel('Y')
  figure
   plot(x,vvm,'LineWidth',1.5)
   xlabel('X')
   ylabel('V')
figure
quiver(x,y,v,u, 10)
%Stream function calculation
str=zeros(nx,ny);
for i=1:nx
for j=2:ny
```

```
   str(i,j)=str(i,j-1)+0.25*(rho(i,j)+rho(i,j-1))*(u(i,j)+u(i,j-1));
end
end

figure
contour(x,y,str,20)
%or use the following build-in function
figure
startx=0.0:0.1:1;
starty=0.0:0.1:1;
streamline(x,y,v,u,startx,starty)
figure
contour(xp,yp,rhog,20)
end
++++++++++++++++++++++++++++++++++++++++++++++++
function[rho,u,v]=ruv(nx,ny,f)
rho=sum (f,3);
  for i=1:nx
rho(i,ny)=f(i,ny,9)+f(i,ny,1)+f(i,ny,3)+2.*(f(i,ny,2)+f(i,ny,6)+f(i,ny,5));
   end
   %calculate velocity compnents
    u = ( sum(f(:,:,[1 5 8]),3) - sum(f(:,:,[3 6 7]),3) )./rho;
   v = ( sum(f(:,:,[2 5 6]),3) - sum(f(:,:,[4 7 8]),3) )./rho;
end

++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  % Streaming:
   function [f]=stream(f)
   f(:,:,1)=circshift( squeeze(f(:,:,1)), [+1,+0] );
   f(:,:,2)=circshift( squeeze(f(:,:,2)), [+0,+1] );
   f(:,:,3)=circshift( squeeze(f(:,:,3)), [-1,+0] );
   f(:,:,4)=circshift( squeeze(f(:,:,4)), [+0,-1] );
   f(:,:,5)=circshift( squeeze(f(:,:,5)), [+1,+1] );
   f(:,:,6)=circshift( squeeze(f(:,:,6)), [-1,+1] );
   f(:,:,7)=circshift( squeeze(f(:,:,7)), [-1,-1] );
   f(:,:,8)=circshift( squeeze(f(:,:,8)), [+1,-1] );
   end
    % End of streaming
================================================================================
```

Heated Channel

```
%LBM- 2-D2Q9, Heated channel, note that c2=1/3, w9=4/9,
% w1-4=1/9, and w5-w8, 1/36
clear
nx=801;ny=41;
f=zeros(nx,ny,9);g=zeros(nx,ny,9);rhog=zeros(nx,ny);
u=zeros(nx,ny);v=zeros(nx,ny);
rho=ones(nx,ny);x=zeros(nx);y=zeros(ny);
w=[1/9 1/9 1/9 1/9 1/36 1/36 1/36 1/36  4/9];
cx = [1 0 -1  0 1 -1 -1  1 0];
cy = [0 1  0 -1 1  1 -1 -1 0];
c2=1./3.;
dx=1.0;dy=1.0;
xl=1.0; yl=1.0;
dx=xl/(nx-1);
dy=yl/(ny-1);
x=(0:1:nx-1);
```

```
y=(0:1:ny-1);
uo=0.10;
alpha=0.04;
Re=uo*(ny-1)/alpha
omega=1./(3.*alpha+0.5);
pr=3.8;
alphag=alpha/pr;
omegag=1.0/(3.*alphag+0.5);
count=0; tol=1.0e-5; error=10.;erso=0.0;
%setting lid velocity to uo
    u(:,ny)=uo;
%Main Loop
 while error>tol
  % Collitions
  [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w);
  % Streaming:
     [f]=stream(f);
    % End of streaming
    %Boundary condition:
    [f]=boundary(nx,ny,f,uo,rho);
% Calculate rho, u, v
 [rho,u,v]=ruv(nx,ny,f);
 %Temperature
  [g]=gcol(nx,ny,u,v,cx,cy,omegag,g,rhog,w);   %collision
  [g]=stream(g);     %streaming
  [g]=gbound(nx,ny,w,g);    % boundary conditions

  rhog=sum(g,3);
  count=count+1;
   ers=0.;
   for i =1:nx
   for j=1:ny
       ers=ers+u(i,j)*u(i,j)+v(i,j)*v(i,j);
   end
   end
    error=abs(ers-erso);
    erso=ers;
   end
%Plotting data
result(nx,ny,x,y,u,v,uo,rho,rhog);
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
%Boudary conditions for Channel flow
function [f]=boundary(nx,ny,f,uo,rho)
  %right hand boundary
   for j=1:ny
      f(nx,j,3)=f(nx-1,j,3);
      f(nx,j,7)=f(nx-1,j,7);
      f(nx,j,6)=f(nx-1,j,6);
     end

   %bottom, and top boundary, bounce back
   for i=1:nx
       f(i,1,2)=f(i,1,4);
       f(i,1,5)=f(i,1,7);
       f(i,1,6)=f(i,1,8);
       f(i,ny,4)=f(i,ny,2);
       f(i,ny,7)=f(i,ny,5);
       f(i,ny,8)=f(i,ny,6);
```

```
       u(i,1)=0.0; v(i,1)=0.0;
       u(i,ny)=0.0; v(i,ny)=0.0;
   end
   %Left boundary, velocity is given= uo
     for j=2:ny-1
     f(1,j,1)=f(1,j,3)+2.*rho(1,j)*uo/3.;
     f(1,j,5)=f(1,j,7)-0.5*(f(1,j,2)-f(1,j,4))+rho(1,j)*uo/6.;
     f(1,j,8)=f(1,j,6)+0.5*(f(1,j,2)-f(1,j,4))+rho(1,j)*uo/6.;
     u(1,j)=uo; v(1,j)=0.0;
     end
 % End of boundary conditions.
end
++++++++++++++++++++++++++++++++++++++++++++++++++++
% Collition
function [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w)
for j=1:ny
     for i=1:nx
       t1=u(i,j)*u(i,j)+v(i,j)*v(i,j);
       for k=1:9
          t2=u(i,j)*cx(k)+v(i,j)*cy(k);
         feq(i,j,k)=rho(i,j)*w(k)*(1.0+3.0*t2+4.5*t2*t2-1.5*t1);
         f(i,j,k)=(1.-omega)*f(i,j,k)+omega*feq(i,j,k);
       end
     end
end
end
++++++++++++++++++++++++++++++++++++++++++++++++
function [g]=gbound(nx,ny,w,g)
%Boundary condition:
      twall=1.0;    %left hand all temperature
    %left boundary, inlet temp. zero
       g(1,:,1)=-g(1,:,3);
       g(1,:,5)=-g(1,:,7);
       g(1,:,8)=-g(1,:,6);
   %right hand boundary, adiabatic
       g(nx,:,3)=g(nx,:,1);
       g(nx,:,7)=g(nx,:,5);
       g(nx,:,6)=g(nx,:,8);
   %bottom boundary, temp. is twall
       g(:,1,2)=(w(2)+w(4))*twall-g(:,1,4);
       g(:,1,5)=(w(5)+w(7))*twall-g(:,1,7);
       g(:,1,6)=(w(6)+w(8))*twall-g(:,1,8);
   %Top boundary, temp. is twall
     g(:,ny,4)=(w(4)+w(2))*twall-g(:,ny,2);
     g(:,ny,8)=(w(8)+w(6))*twall-g(:,ny,6);
     g(:,ny,7)=(w(7)+w(5))*twall-g(:,ny,5);


end
++++++++++++++++++++++++++++++++++++++++++++++++
% Collition
function [g]=gcol(nx,ny,u,v,cx,cy,omegag,g,rhog,w)
for j=1:ny
     for i=1:nx
       t1=u(i,j)*u(i,j)+v(i,j)*v(i,j);
       for k=1:9
          t2=u(i,j)*cx(k)+v(i,j)*cy(k);
```

```
         geq(i,j,k)=rhog(i,j)*w(k)*(1.0+3.0*t2);
         g(i,j,k)=(1.-omegag)*g(i,j,k)+omegag*geq(i,j,k);
       end
     end
end
end
++++++++++++++++++++++++++++++++++++++++++++++
function result(nx,ny,x,y,u,v,uo,rho,rhog)
for j=1:ny
  for i=1:nx
      xp(i,j)=x(i);
      yp(i,j)=y(j);
  end
end
for j=1:ny
      uvm(j)=u((nx-1)/2,j)/uo;
      tmj(j)=rhog((nx-1)/2,j);
 end
   figure
  plot(tmj,y,'LineWidth',2)
figure
 plot(uvm,y,'LineWidth',2)
    xlabel('U')
    ylabel('Y')
%Stream function calculation
str=zeros(nx,ny);
for i=1:nx
for j=2:ny
   str(i,j)=str(i,j-1)+0.25*(rho(i,j)+rho(i,j-1))*(u(i,j)+u(i,j-1));
end
end
figure
contour(xp,yp,str,20)

%or use the following build-in function
figure
contour(xp,yp,rhog,20)

end
++++++++++++++++++++++++++++++++++++++++++++++
function[rho,u,v]=ruv(nx,ny,f)
rho=sum (f,3);
  for i=1:nx
rho(i,ny)=f(i,ny,9)+f(i,ny,1)+f(i,ny,3)+2.*(f(i,ny,2)+f(i,ny,6)+f(i,ny,5));
   end
   %calculate velocity compnents
    u = ( sum(f(:,:,[1 5 8]),3) - sum(f(:,:,[3 6 7]),3) )./rho;
   v = ( sum(f(:,:,[2 5 6]),3) - sum(f(:,:,[4 7 8]),3) )./rho;
end

++++++++++++++++++++++++++++++++++++++++++++++++++
% Streaming:
   function [f]=stream(f)
   f(:,:,1)=circshift( squeeze(f(:,:,1)), [+1,+0] );
   f(:,:,2)=circshift( squeeze(f(:,:,2)), [+0,+1] );
   f(:,:,3)=circshift( squeeze(f(:,:,3)), [-1,+0] );
   f(:,:,4)=circshift( squeeze(f(:,:,4)), [+0,-1] );
   f(:,:,5)=circshift( squeeze(f(:,:,5)), [+1,+1] );
```

```
   f(:,:,6)=circshift( squeeze(f(:,:,6)), [-1,+1] );
   f(:,:,7)=circshift( squeeze(f(:,:,7)), [-1,-1] );
   f(:,:,8)=circshift( squeeze(f(:,:,8)), [+1,-1] );
   end
    % End of streaming
================================================================================
```

Conjugate Problem

```
%LBM- 2-D2Q9, Heated Lid-driven Cavity, Re=1000, note that
c2=1/3, w9=4/9,
% w1-4=1/9, and w5-w8, 1/36
clear
nx=801;ny=41;
f=zeros(nx,ny,9);g=zeros(nx,ny,9);rhog=zeros(nx,ny);
u=zeros(nx,ny);v=zeros(nx,ny);
rho=ones(nx,ny);x=zeros(nx);y=zeros(ny);
w=[1/9 1/9 1/9 1/9 1/36 1/36 1/36 1/36  4/9];
cx = [1 0 -1  0 1 -1 -1  1 0];
cy = [0 1  0 -1 1  1 -1 -1 0];
c2=1./3.;
dx=1.0;dy=1.0;
xl=1.0; yl=1.0;
dx=xl/(nx-1);
dy=yl/(ny-1);
x=(0:1:nx-1);
y=(0:1:ny-1);
uo=0.10;
alpha=0.04;
Re=uo*(ny-1)/alpha
omega=1./(3.*alpha+0.5);
pr=3.8;
alphag=alpha/pr;
omegag=1.0/(3.*alphag+0.5);
count=0; tol=1.0e-5; error=10.;erso=0.0;
%setting lid velocity to uo
    u(:,ny)=uo;
%Main Loop
 while error>tol
  % Collitions
  [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w);
  % Streaming:
     [f]=stream(f);
    % End of streaming
    %Boundary condition:
    [f]=boundary(nx,ny,f,uo,rho);
% Calculate rho, u, v
 [rho,u,v]=ruv(nx,ny,f);
 %Temperature
  [g]=gcol(nx,ny,u,v,cx,cy,omegag,g,rhog,w);   %collision
  [g]=stream(g);     %streaming
  [g]=gbound(nx,ny,w,g);    % boundary conditions

  rhog=sum(g,3);
  count=count+1;
   ers=0.;
   for i =1:nx
   for j=1:ny
```

```
        ers=ers+u(i,j)*u(i,j)+v(i,j)*v(i,j);
    end
    end
     error=abs(ers-erso);
     erso=ers;
    end
%Plotting data
result(nx,ny,x,y,u,v,uo,rho,rhog);


++++++++++++++++++++++++++++++++++++++++++++++
%Boudary conditions for Channel flow
function [f]=boundary(nx,ny,f,uo,rho)
  %right hand boundary
    for j=1:ny
       f(nx,j,3)=f(nx-1,j,3);
       f(nx,j,7)=f(nx-1,j,7);
       f(nx,j,6)=f(nx-1,j,6);
      end
    %bottom, and top boundary, bounce back
    for i=1:nx
        f(i,1,2)=f(i,1,4);
        f(i,1,5)=f(i,1,7);
        f(i,1,6)=f(i,1,8);
        f(i,ny,4)=f(i,ny,2);
        f(i,ny,7)=f(i,ny,5);
        f(i,ny,8)=f(i,ny,6);
        u(i,1)=0.0; v(i,1)=0.0;
        u(i,ny)=0.0; v(i,ny)=0.0;
    end
    %Left boundary, velocity is given= uo
      for j=2:ny-1
      f(1,j,1)=f(1,j,3)+2.*rho(1,j)*uo/3.;
      f(1,j,5)=f(1,j,7)-0.5*(f(1,j,2)-f(1,j,4))+rho(1,j)*uo/6.;
      f(1,j,8)=f(1,j,6)+0.5*(f(1,j,2)-f(1,j,4))+rho(1,j)*uo/6.;
      u(1,j)=uo; v(1,j)=0.0;
      end
      %solid region
      for j=1:11
          for i=1:nx
              u(i,j)=0.0;
              v(i,j)=0.0;
              u(i,ny-j)=0.0;
              v(i,ny-j)=0.0;
          end
      end
      f(:,11,5)=f(:,11,7);
      f(:,11,2)=f(:,11,4);
      f(:,11,6)=f(:,11,8);
      f(:,ny-11,7)=f(:,ny-11,5);
      f(:,ny-11,4)=f(:,ny-11,2);
      f(:,ny-11,8)=f(:,ny-11,6);
 % End of boundary conditions.
end
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
% Collition
function [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w)
```

```
for j=1:ny
     for i=1:nx
       t1=u(i,j)*u(i,j)+v(i,j)*v(i,j);
       for k=1:9
          t2=u(i,j)*cx(k)+v(i,j)*cy(k);
         feq(i,j,k)=rho(i,j)*w(k)*(1.0+3.0*t2+4.5*t2*t2-1.5*t1);
         f(i,j,k)=(1.-omega)*f(i,j,k)+omega*feq(i,j,k);
       end
     end
end
end

++++++++++++++++++++++++++++++++++++++++++++++++++
function [g]=gbound(nx,ny,w,g)
%Boundary condition:
      twall=1.0;    %left hand all temperature
    %left boundary, inlet temp. zero
       g(1,:,1)=-g(1,:,3);
       g(1,:,5)=-g(1,:,7);
       g(1,:,8)=-g(1,:,6);
   %right hand boundary, adiabatic
       g(nx,:,3)=g(nx,:,1);
       g(nx,:,7)=g(nx,:,5);
       g(nx,:,6)=g(nx,:,8);
   %bottom boundary, temp. is twall
       g(:,1,2)=(w(2)+w(4))*twall-g(:,1,4);
       g(:,1,5)=(w(5)+w(7))*twall-g(:,1,7);
       g(:,1,6)=(w(6)+w(8))*twall-g(:,1,8);
   %Top boundary, temp. is twall
     g(:,ny,4)=(w(4)+w(2))*twall-g(:,ny,2);
     g(:,ny,8)=(w(8)+w(6))*twall-g(:,ny,6);
     g(:,ny,7)=(w(7)+w(5))*twall-g(:,ny,5);


end
++++++++++++++++++++++++++++++++++++++++++++++++++
% Collition
function [g]=gcol(nx,ny,u,v,cx,cy,omegag,g,rhog,w)
for j=1:ny
     for i=1:nx
       t1=u(i,j)*u(i,j)+v(i,j)*v(i,j);
       mg=omegag;
       if (j<11) mg=mg*0.1;
       end
           if(j>(ny-11)) mg=mg*0.1;
           end
       for k=1:9
          t2=u(i,j)*cx(k)+v(i,j)*cy(k);
         geq(i,j,k)=rhog(i,j)*w(k)*(1.0+3.0*t2);
         g(i,j,k)=(1.-mg)*g(i,j,k)+mg*geq(i,j,k);
       end
     end
end
end
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
function result(nx,ny,x,y,u,v,uo,rho,rhog)
%solid region velocity decoration
 for j=1:11
```

```
         for i=1:nx
             u(i,j)=0.0;
             v(i,j)=0.0;
             u(i,ny-j)=0.0;
             v(i,ny-j)=0.0;
         end
     end
for j=1:ny
  for i=1:nx
      xp(i,j)=x(i);
      yp(i,j)=y(j);
  end
end
for j=1:ny
      uvm(j)=u((nx-1)/2,j)/uo;
      tmj(j)=rhog((nx-1)/2,j);
      ttj(j)=rhog(nx-40,j);
 end
   figure
  plot(tmj,y,ttj,y,'LineWidth',2)
figure
 plot(uvm,y,'LineWidth',2)
    xlabel('U')
    ylabel('Y')
%Stream function calculation
str=zeros(nx,ny);
for i=1:nx
for j=2:ny
   str(i,j)=str(i,j-1)+0.25*(rho(i,j)+rho(i,j-1))*(u(i,j)+u(i,j-1));
end
end
figure
contour(xp,yp,u,20)

%or use the following build-in function
figure
contour(xp,yp,rhog,20)

end
++++++++++++++++++++++++++++++++++++++++++++++++++++++
function[rho,u,v]=ruv(nx,ny,f)
rho=sum (f,3);
  for i=1:nx
rho(i,ny)=f(i,ny,9)+f(i,ny,1)+f(i,ny,3)+2.*(f(i,ny,2)+f(i,ny,6)+f(i,ny,5));
   end
   %calculate velocity compnents
    u = ( sum(f(:,:,[1 5 8]),3) - sum(f(:,:,[3 6 7]),3) )./rho;
   v = ( sum(f(:,:,[2 5 6]),3) - sum(f(:,:,[4 7 8]),3) )./rho;
end
++++++++++++++++++++++++++++++++++++++++++++++++++
% Streaming:
   function [f]=stream(f)
   f(:,:,1)=circshift( squeeze(f(:,:,1)), [+1,+0] );
   f(:,:,2)=circshift( squeeze(f(:,:,2)), [+0,+1] );
   f(:,:,3)=circshift( squeeze(f(:,:,3)), [-1,+0] );
   f(:,:,4)=circshift( squeeze(f(:,:,4)), [+0,-1] );
   f(:,:,5)=circshift( squeeze(f(:,:,5)), [+1,+1] );
```

```
   f(:,:,6)=circshift( squeeze(f(:,:,6)), [-1,+1] );
   f(:,:,7)=circshift( squeeze(f(:,:,7)), [-1,-1] );
   f(:,:,8)=circshift( squeeze(f(:,:,8)), [+1,-1] );
   end
    % End of streaming
================================================================================
```

Differentially Heated Cavity

```
%LBM- 2-D2Q9, Differentially Heated, note that c2=1/3, w9=4/9,
% w1-4=1/9, and w5-w8, 1/36
clear
nx=81;ny=81;
f=zeros(nx,ny,9);g=zeros(nx,ny,9);rhog=ones(nx,ny);
u=zeros(nx,ny);v=zeros(nx,ny);
rho=ones(nx,ny);x=zeros(nx);y=zeros(ny);
w=[1/9 1/9 1/9 1/9 1/36 1/36 1/36 1/36  4/9];
cx = [1 0 -1  0 1 -1 -1  1 0];
cy = [0 1  0 -1 1  1 -1 -1 0];
c2=1./3.;
dx=1.0;dy=1.0;
xl=1.0; yl=1.0;
dx=xl/(nx-1);
dy=yl/(ny-1);
x=(0:dx:xl);
y=(0:dy:yl);
alpha=0.05;
omega=1./(3.*alpha+0.5);
pr=0.71;
Ra=1.0e5;
alphag=alpha/pr;
ym1=real(ny-1);
gbeta=Ra*alpha*alphag/(ym1*ym1*ym1);
omegag=1.0/(3.*alphag+0.5);
count=0;
%Main Loop
 while count< 70000
  % Collitions
  [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w,rhog,gbeta);
  % Streaming:
     [f]=stream(f);
    % End of streaming
    %Boundary condition:
    [f]=boundary(nx,ny,f);
% Calculate rho, u, v
 [rho,u,v]=ruv(nx,ny,f);
 %Temperature
  [g]=gcol(nx,ny,u,v,cx,cy,omegag,g,rhog,w);   %collision
  [g]=stream(g);     %streaming
  [g]=gbound(nx,ny,w,g);    % boundary conditions

  rhog=sum(g,3);
  count=count+1;
end
%Plotting data
result(nx,ny,x,y,u,v,rho,rhog);

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
function [f]=boundary(nx,ny,f)
%Boundary condition:
    %left boundary, bounc back
       f(1,:,1)=f(1,:,3);
       f(1,:,5)=f(1,:,7);
       f(1,:,8)=f(1,:,6);
   %right hand boundary
       f(nx,:,3)=f(nx,:,1);
       f(nx,:,7)=f(nx,:,5);
       f(nx,:,6)=f(nx,:,8);
   %bottom boundary, bounce back
       f(:,1,2)=f(:,1,4);
       f(:,1,5)=f(:,1,7);
       f(:,1,6)=f(:,1,8);
   %Top boundary,
     f(:,ny,4)=f(:,ny,2);
     f(:,ny,8)=f(:,ny,6);
     f(:,ny,7)=f(:,ny,5);
  end
++++++++++++++++++++++++++++++++++++++++++++++++
% Collition
function [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w,rhog,gbeta)
for j=1:ny
     for i=1:nx
       t1=u(i,j)*u(i,j)+v(i,j)*v(i,j);
       for k=1:9
           force=3.*w(k)*gbeta*rhog(i,j)*rho(i,j)*cy(k);
           if(i==1 || i==nx) force=0.0;
           end
           if(j==1 || j==ny) force =0.0;
           end
           t2=u(i,j)*cx(k)+v(i,j)*cy(k);
         feq=rho(i,j)*w(k)*(1.0+3.0*t2+4.5*t2*t2-1.5*t1);
         f(i,j,k)=(1.-omega)*f(i,j,k)+omega*feq+force;
       end
     end
end
end
++++++++++++++++++++++++++++++++++++++++++++
function [g]=gbound(nx,ny,w,g)
%Boundary condition:
      twall=1.0;    %left hand all temperature
    %left boundary, T=1.0
       g(1,:,1)=twall*(w(1)+w(3))-g(1,:,3);
       g(1,:,5)=twall*(w(5)+w(7))-g(1,:,7);
       g(1,:,8)=twall*(w(8)+w(6))-g(1,:,6);
   %right hand boundary T=0.0
       g(nx,:,3)=-g(nx,:,1);
       g(nx,:,7)=-g(nx,:,5);
       g(nx,:,6)=-g(nx,:,8);
   %bottom boundary,Insulated
       g(:,1,2)=g(:,2,4);
       g(:,1,5)=g(:,2,7);
       g(:,1,6)=g(:,2,8);
 %      g(:,1,4)=g(:,2,4);
 %      g(:,1,7)=g(:,2,7);
```

```
 %      g(:,1,8)=g(:,2,8);
 %      g(:,1,9)=g(:,2,9);
   %Top boundary,Insulated
     g(:,ny,4)=g(:,ny-1,2);
     g(:,ny,8)=g(:,ny-1,6);
     g(:,ny,7)=g(:,ny-1,5);
  %   g(:,ny,1)=g(:,ny-1,1);
  %   g(:,ny,2)=g(:,ny-1,2);
  %   g(:,ny,3)=g(:,ny-1,3);
  %   g(:,ny,5)=g(:,ny-1,5);
  %   g(:,ny,9)=g(:,ny-1,9);

end


++++++++++++++++++++++++++++++++++++++++++++++
% Collition
function [g]=gcol(nx,ny,u,v,cx,cy,omegag,g,rhog,w)
for j=1:ny
     for i=1:nx
       t1=u(i,j)*u(i,j)+v(i,j)*v(i,j);
       for k=1:9
          t2=u(i,j)*cx(k)+v(i,j)*cy(k);
         geq(i,j,k)=rhog(i,j)*w(k)*(1.0+3.0*t2);
         g(i,j,k)=(1.-omegag)*g(i,j,k)+omegag*geq(i,j,k);
       end
     end
end
end
++++++++++++++++++++++++++++++++++++++++++++++++
function result(nx,ny,x,y,u,v,rho,rhog)
for j=1:ny
      uvm(j)=u((nx-1)/2,j);
      tmj(j)=rhog((nx-1)/2,j);
 end
  for i=1:nx
  vvm(i)=v(i,(ny-1)/2);
  end
  figure
  plot(tmj,y,'LineWidth',2)
figure
 plot(uvm,y,'LineWidth',2)
    xlabel('U')
    ylabel('Y')
  figure
   plot(x,vvm,'LineWidth',1.5)
   xlabel('X')
   ylabel('V')
figure
quiver(x,y,v,u, 10)
%Stream function calculation
str=zeros(nx,ny);
for i=1:nx
for j=2:ny
   str(i,j)=str(i,j-1)+0.25*(rho(i,j)+rho(i,j-1))*(u(i,j)+u(i,j-1));
end
end
figure contour(x,y,str)
```

```
%or use the following build-in function
figure
startx=0.0:0.1:1;
starty=0.0:0.1:1;
streamline(x,y,v,u,startx,starty)
figure
contour(x,y,rhog,15)
end
++++++++++++++++++++++++++++++++++++++++++++++++
function[rho,u,v]=ruv(nx,ny,f)
rho=sum (f,3);
  for i=1:nx
rho(i,ny)=f(i,ny,9)+f(i,ny,1)+f(i,ny,3)+2.*(f(i,ny,2)+f(i,ny,6)+f(i,ny,5));
   end
   %calculate velocity compnents
    u = ( sum(f(:,:,[1 5 8]),3) - sum(f(:,:,[3 6 7]),3) )./rho;
   v = ( sum(f(:,:,[2 5 6]),3) - sum(f(:,:,[4 7 8]),3) )./rho;
end
++++++++++++++++++++++++++++++++++++++++
% Streaming:
   function [f]=stream(f)
   f(:,:,1)=circshift( squeeze(f(:,:,1)), [+1,+0] );
   f(:,:,2)=circshift( squeeze(f(:,:,2)), [+0,+1] );
   f(:,:,3)=circshift( squeeze(f(:,:,3)), [-1,+0] );
   f(:,:,4)=circshift( squeeze(f(:,:,4)), [+0,-1] );
   f(:,:,5)=circshift( squeeze(f(:,:,5)), [+1,+1] );
   f(:,:,6)=circshift( squeeze(f(:,:,6)), [-1,+1] );
   f(:,:,7)=circshift( squeeze(f(:,:,7)), [-1,-1] );
   f(:,:,8)=circshift( squeeze(f(:,:,8)), [+1,-1] );
   end
    % End of streaming
================================================================================
```

A.6 Chapter Ten

MRT-Code for Lid-Driven Cavity

```
%LBM-MRT 2-D2Q9, Lid-driven Cavity, Re=1000, note that c2=1/3, w9=4/9,
% w1-4=1/9, and w5-w8, 1/36
clear
nx=101;ny=101;
f=zeros(nx,ny,9);feq=zeros(nx,ny,9);
u=zeros(nx,ny);v=zeros(nx,ny);
rho=ones(nx,ny);x=zeros(nx);y=zeros(ny);
w=[1/9 1/9 1/9 1/9 1/36 1/36 1/36 1/36  4/9];
cx = [1 0 -1  0 1 -1 -1  1 0];
cy = [0 1  0 -1 1  1 -1 -1 0];
c2=1./3.;
dx=1.0;dy=1.0;
xl=1.0; yl=1.0;
dx=xl/(nx-1);
dy=yl/(ny-1);
x=(0:dx:xl);
y=(0:dy:yl);
uo=0.10;
alpha=0.01;
```

```
Re=uo*(ny-1)/alpha
omega=1./(3.*alpha+0.5);
count=0; tol=1.0e-4; error=10.;erso=0.0;
sm=[ 1.4 1.4 1.0 1.2 1.0 1.2 omega omega 1.0];
m1=[  1  1  1  1 1  1  1  1  1 ];
m2=[ -1 -1 -1 -1 2  2  2  2 -4 ];
m3=[ -2 -2 -2 -2 1  1  1  1  4 ];
m4=[  1  0 -1  0 1 -1 -1  1  0 ];
m5=[ -2  0  2  0 1 -1 -1  1  0 ];
m6=[  0  1  0 -1 1  1 -1 -1  0 ];
m7=[  0 -2  0  2 1  1 -1 -1  0 ];
m8=[  1 -1  1 -1 0  0  0  0  0 ];
m9=[  0  0  0  0 1 -1  1 -1  0 ];
mm=[m2;m3;m4;m5;m6;m7;m8;m9;m1]; %saleh
mminv=inv(mm);
msm=mminv*diag(-sm); %saleh
%setting lid velocity
    u(:,ny)=uo;
%Main Loop
 while error>tol
  % Collitions
  [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w,mm,mminv,msm);
  % Streaming:
     [f]=stream(f);
    % End of streaming
    %Boundary condition:
    [f]=boundary(nx,ny,f,uo);
% Calculate rho, u, v
 [rho,u,v]=ruv(nx,ny,f);
  count=count+1;
   ers=0.;
   for i =1:nx
   for j=1:ny
       ers=ers+u(i,j)*u(i,j)+v(i,j)*v(i,j);
   end
   end
    error=abs(ers-erso);
    erso=ers;
   end
%Plotting data
result(nx,ny,x,y,u,v,uo,rho);

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

function [f]=boundary(nx,ny,f,uo)
%Boundary condition:
    %left boundary, bounc back
       f(1,:,1)=f(1,:,3);
       f(1,:,5)=f(1,:,7);
       f(1,:,8)=f(1,:,6);
   %right hand boundary
       f(nx,:,3)=f(nx,:,1);
       f(nx,:,7)=f(nx,:,5);
       f(nx,:,6)=f(nx,:,8);
   %bottom boundary, bounce back
       f(:,1,2)=f(:,1,4);
       f(:,1,5)=f(:,1,7);
       f(:,1,6)=f(:,1,8);
```

```
   %Top boundary,moving lid with uo
     for i=2:nx-1
     rhon=f(i,ny,9)+f(i,ny,1)+f(i,ny,3)+2.*(f(i,ny,2)+f(i,ny,6)+f(i,ny,5));
     f(i,ny,4)=f(i,ny,2);
     f(i,ny,8)=f(i,ny,6)+rhon*uo/6.0;
     f(i,ny,7)=f(i,ny,5)-rhon*uo/6.0;
     end
end
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
% Collition
function [f]=collition(nx,ny,u,v,cx,cy,omega,f,rho,w,mm,mminv,msm)
% Calculate the fmeq
fmeq(:,:,1)= rho.*(-2.+3.*rho.*(u.*u+v.*v));
fmeq(:,:,2)= rho.*( 1.-3.*rho.*(u.*u+v.*v));
fmeq(:,:,3)= rho.* u;
fmeq(:,:,4)=-rho.* u;
fmeq(:,:,5)= rho.* v;
fmeq(:,:,6)=-rho.* v;
fmeq(:,:,7)= rho.*(u.*u-v.*v);
fmeq(:,:,8)= rho.* u.*v;
fmeq(:,:,9)= rho;

for i=1:nx
    for j=1:ny
        for k=1:9
        smf=0.0;
        for N=1:9
            smf=smf+mm(k,N)*f(i,j,N);
        end
        fmom(i,j,k)=smf;
        end
    end
end

for j=1:ny
     for i=1:nx
       for k=1:9
           ssmb=0.0;
           for N=1:9
               ssmb=ssmb+msm(k,N)*(fmom(i,j,N)-fmeq(i,j,N)); %saleh
           end
          f(i,j,k)=f(i,j,k)+ssmb;
       end
     end
end

end
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
function result(nx,ny,x,y,u,v,uo,rho)
for j=1:ny
      um(j)=u((nx-1)/2,j)/uo;
 end
  for i=1:nx
  vm(i)=v(i,(ny-1)/2)/uo;
  end
 figure
  line(um,y,'LineWidth',1.5)
```

```
    xlabel('U')
    ylabel('Y')
  figure
   line(x,vm,'LineWidth',1.5)
   xlabel('X')
   ylabel('V')
figure
quiver(x,y,v/uo,u/uo, 10)
%Stream function calculation
str=zeros(nx,ny);
for i=1:nx
for j=2:ny
   str(i,j)=str(i,j-1)+0.25*(rho(i,j)+rho(i,j-1))*(u(i,j)+u(i,j-1));
end
end
figure
contour(x,y,str,20)
%or use the following build-in function
figure
startx=0.0:0.1:1;
starty=0.0:0.1:1;
streamline(x,y,v,u,startx,starty)
end
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
function[rho,u,v]=ruv(nx,ny,f)
rho=sum (f,3);
  for i=1:nx
rho(i,ny)=f(i,ny,9)+f(i,ny,1)+f(i,ny,3)+2.*(f(i,ny,2)+f(i,ny,6)+f(i,ny,5));
   end
   %calculate velocity compnents
    u = ( sum(f(:,:,[1 5 8]),3) - sum(f(:,:,[3 6 7]),3) )./rho;
   v = ( sum(f(:,:,[2 5 6]),3) - sum(f(:,:,[4 7 8]),3) ).rho;
end
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
% Streaming:
   function [f]=stream(f)
   f(:,:,1)=circshift( squeeze(f(:,:,1)), [+1,+0] );
   f(:,:,2)=circshift( squeeze(f(:,:,2)), [+0,+1] );
   f(:,:,3)=circshift( squeeze(f(:,:,3)), [-1,+0] );
   f(:,:,4)=circshift( squeeze(f(:,:,4)), [+0,-1] );
   f(:,:,5)=circshift( squeeze(f(:,:,5)), [+1,+1] );
   f(:,:,6)=circshift( squeeze(f(:,:,6)), [-1,+1] );
   f(:,:,7)=circshift( squeeze(f(:,:,7)), [-1,-1] );
   f(:,:,8)=circshift( squeeze(f(:,:,8)), [+1,-1] );
   end
    % End of streaming
================================================================================
```

A.7 Chapter Eleven

SC-Model

```
%LBM-Sc-model 2-D2Q9, note that c2=1/3, w9=4/9,
% w1-4=1/9, and w5-w8, 1/36
```

```
%           6   2  5
%              \ | /
%           3 -- 9-- 1
%              / | \
%           7   4   8
clear all;
clc;
G=-5.0;
rho_cr=log(2);
rho_liq=1.95;
rho_gas=0.15;
Nsteps=2000;
nx=128;ny=128;
noput=10;
f=zeros(nx,ny,9);feq=zeros(nx,ny,9);
u=zeros(nx,ny);v=zeros(nx,ny);
uf=zeros(nx,ny);vf=zeros(nx,ny);
forcx=zeros(nx,ny);forcy=zeros(nx,ny);
rho=ones(nx,ny);x=zeros(nx);y=zeros(ny);
w=[1/9 1/9 1/9 1/9 1/36 1/36 1/36 1/36  4/9];
cx = [1 0 -1  0 1 -1 -1  1 0];
cy = [0 1  0 -1 1  1 -1 -1 0];
c2=1./3.;
omega=1.;
%setting
for i=1:nx
    for j=1:ny
      rho(i,j)=rho_cr+0.1*rand();
    end
end
counter_frame=1;
%Main Loop
 for counter=1:Nsteps
 % force terms
     [forcx,forcy]=force(nx,ny,u,v,cx,cy,rho,w,G);
  % Collitions
  [f]=collision(nx,ny,uf,vf,cx,cy,omega,f,rho,w,forcx,forcy);
 % Streaming:
     [f]=stream(f);
    % End of streaming
    %Boundary condition:
    [f]=boundary(nx,ny,f);
% Calculate rho, u, v
 [rho,u,v]=ruv(nx,ny,f);
 for j=1:ny
     for i=1:nx
       uf(i,j)=u(i,j)+0.5*forcx(i,j)/rho(i,j);
       vf(i,j)=v(i,j)+0.5*forcy(i,j)/rho(i,j);
       end
 end
     counter;
 if mod(counter,noput)==0
     imagesc(rho);
     F(counter_frame)=getframe;
     counter_frame=counter_frame+1;
 end
     %  imagesc(rho)
  %  F = getframe(gcf);
```

```
   end
%Plotting data
 movie(F,10)
% imagesc(rho);
disp('Rho_liq=')
disp(mean(mean(rho(nx/2-5:nx/2+5,ny/2-5:ny/2+5))))
disp('Rho_gas=')
disp(mean(mean(rho(1:10,1:10))))

++++++++++++++++++++++++++++++++++++++++++++++++
function [f]=boundary(nx,ny,f)
%Boundary condition:
    %left boundary, bounc back
       f(1,:,1)=f(nx,:,1);
       f(1,:,5)=f(nx,:,5);
       f(1,:,8)=f(nx,:,8);
   %right hand boundary
       f(nx,:,3)=f(1,:,3);
       f(nx,:,7)=f(1,:,7);
       f(nx,:,6)=f(1,:,6);
   %bottom boundary, bounce back
       f(:,1,2)=f(:,ny,2);
       f(:,1,5)=f(:,ny,5);
       f(:,1,6)=f(:,ny,6);
   %Top boundary
     for i=2:nx-1
     f(i,ny,4)=f(i,1,4);
     f(i,ny,8)=f(i,1,8);
     f(i,ny,7)=f(i,1,7);
     end
end
++++++++++++++++++++++++++++++++++++++++++++++++
% Collition
function [f]=collision(nx,ny,uf,vf,cx,cy,omega,f,rho,w,forcx,forcy)
for j=1:ny
     for i=1:nx
       t1=uf(i,j)*uf(i,j)+vf(i,j)*vf(i,j);
       for k=1:9
          t2=uf(i,j)*cx(k)+vf(i,j)*cy(k);
         feq(i,j,k)=rho(i,j)*w(k)*(1.0+3.0*t2+4.5*t2*t2-1.5*t1);
         ffx=(3.*(cx(k)-uf(i,j))+9.*cx(k)*t2)*forcx(i,j);
         ffy=(3.*(cy(k)-vf(i,j))+9.*cy(k)*t2)*forcy(i,j);
         fftot=w(k)*(1.-0.5*omega)*(ffx+ffy);
         f(i,j,k)=(1.-omega)*f(i,j,k)+omega*feq(i,j,k)+fftot;
       end
     end
end
end
++++++++++++++++++++++++++++++++++++++++++++++++
% Force term
function [forcx,forcy]=force(nx,ny,u,v,cx,cy,rho,w,G)
for j=1:ny
    for i=1:nx

        forcxs=0.0;
        forcys=0.0;
        for k=1:8
            newx=1+mod(i-1+cx(k)+nx,nx);
```

```
            newy=1+mod(j-1+cy(k)+ny,ny);
            psi=1-exp(-rho(newx,newy));
            forcxs=forcxs-G*w(k)*psi*cx(k);
            forcys=forcys-G*w(k)*psi*cy(k);
        end
        forcx(i,j)=(1-exp(-rho(i,j)))*forcxs;
        forcy(i,j)=(1-exp(-rho(i,j)))*forcys;
    end
end

end
++++++++++++++++++++++++++++++++++++++++++++++
function[rho,u,v]=ruv(nx,ny,f)
rho=sum (f,3);
 %calculate velocity compnents
    u = ( sum(f(:,:,[1 5 8]),3) - sum(f(:,:,[3 6 7]),3) )./rho;
   v = ( sum(f(:,:,[2 5 6]),3) - sum(f(:,:,[4 7 8]),3) )./rho;
end
++++++++++++++++++++++++++++++++++++++++++++++
% Streaming:
   function [f]=stream(f)
   f(:,:,1)=circshift( squeeze(f(:,:,1)), [+1,+0] );
   f(:,:,2)=circshift( squeeze(f(:,:,2)), [+0,+1] );
   f(:,:,3)=circshift( squeeze(f(:,:,3)), [-1,+0] );
   f(:,:,4)=circshift( squeeze(f(:,:,4)), [+0,-1] );
   f(:,:,5)=circshift( squeeze(f(:,:,5)), [+1,+1] );
   f(:,:,6)=circshift( squeeze(f(:,:,6)), [-1,+1] );
   f(:,:,7)=circshift( squeeze(f(:,:,7)), [-1,-1] );
   f(:,:,8)=circshift( squeeze(f(:,:,8)), [+1,-1] );
   end
    % End of streaming
================================================================================
```

Index

A. A. Mohamad, *Lattice Boltzmann Method*,
https://doi.org/10.1007/978-1-4471-7423-3