Computational modelling of flows in porous scaffold materials using a lattice Boltzmann method

Ye, Shangjun

Computational modelling of flows in porous scaffold materials using a lattice Boltzmann method

Shangjun Ye

2010

Supervisors: Professor Wen Wang
Dr. Alexander Vikhansky

A thesis submitted in partial fulfillment of the requirements for the Degree of Philosophy in

School of Engineering and Materials Science
Queen Mary, University of London
Mile End Road
London E1 4NS

Dedicated to my wife Hui Liu and my son Zizai

Acknowledge

Foremost, I would like to sincerely thank my supervisor, Professor Wen Wang, for his encouragement, guidance and support to me throughout the PhD project. His patience, motivation, enthusiasm and immense knowledge not only helped me in all stages of the research and writing up of the PhD thesis, but also set an example that I aim to achieve some day.

I am grateful to my co-supervisor, Dr Alexander Vikhansky, for his detailed and constructive comments on my work, and for his help in the past three and half years.

I am grateful to Dr Yiling Lu for his valuable advice on this project and on a number of projects that I participated in during my PhD study. I would like to thank him and his family for the wonderful time that I enjoyed with them.

I thank Professor David Lee and Dr Jonathan Campbell for their help on micro-CT images of the scaffold materials and a number of valuable discussions on cell & tissue engineering.

I would also like to thank colleagues in Biofluids and Cell Mechanics Laboratory, Dr Wei Li, Dr QianQian Wang, Miss Chanyuan Chai, Mr Zhikai Tan, Mr Yankai Liu, Ms Jingyi Du and Miss Ke Bai for helpful discussions and enjoyable time during the long hours in the laboratory.

It has been a pleasure to share time with my office mates, Pengxiang

Song, Naomi Kwak, Neveen Hosny, and Henry Clarke.

My deepest gratitude goes to my family. My parents Liuzhong Ye and Dongfeng Gu, my wife Hui Liu, and my son Zizai Ye who are the source of my strength. They have unselfishly sacrificed their own quality of life and tolerated my absence from the family in order to pursuit research in the UK. Without their encouragement, understanding and love, it would have been impossible for me to complete the work.

Finally, I would like to thank all who have supported me during my PhD study.

Abstract

Porous scaffold materials have been widely used in biological tissue engineering. It is known that fluid flow in porous media significantly increases the supply of oxygen and other nutrients to cells seeded in the porous material, and speeds up the clearance of metabolic end products. Local shear stress distribution is a function of media flow rate, viscosity and the porous scaffold micro-structure. This research project aims to investigate fluid movement in porous structures by using a lattice Boltzmann method. This new numerical method models the fluid as a collection of identical particles with collision and propagation procedures, and has been shown as an alternative and efficient numerical solver of Navier-Stokes equations, in particular for flows in complex geometries. The numerical scheme is verified using flow in a two-dimensional channel, as well as in three-dimensional ducts with constant shapes, where analytical solutions are available. 2D porous structures originated from micro-CT images are then used to study the flow and wall shear stress distribution. One of the advantages of the lattice Boltzmann method is that the shear stress can be computed directly from the local distribution function and has the same accuracy with the velocity profile. Fluid patterns and wall shear stress distribution in 3D porous structures, which

are reconstructed from the micro-tomographic slices, have been investigated under different flow rates, viscosity and geometrical structures. Results from this project demonstrate that lattice Boltzmann method is suitable for flow modelling in scaffold materials. It provides detailed information on localized velocity and stress distributions, which can be used to improve the design of the scaffold for cell and tissue engineering.

Contents

Lists of Figures

Lists of Tables

1 Introduction

1.1 Cell culture in tissue engineering

Cell culture is the process to grow cells isolated from living tissues, especially of the animal, in the laboratory, by controlling the supply of nutrients and other conditions (Bhatia 1999). The history of cell culture can date back to 19th century. In 1885, Roux removed a portion of the medullary plate of a chicken embryo and kept it alive in a warm saline solution for several days, which showed that it might be possible to culture cells in vitro (Alberts *et al* 2002). Ross Granville Harrison, working at Johns Hopkins Medical School and then at Yale University, published his results from 1907-1910, establishing the methodology of tissue culture (Schiff 2002). Cell culture techniques were advanced significantly in the 1940s and 1950s to support the research in virology. The Salk polio vaccine, one of the first products mass-produced using cell culture techniques was made possible by the work of John Franklin Enders, Thomas Huckle Weller, and Frederick Chapman Robbins, who were awarded the Nobel Prize for Physiology or Medicine in 1954 “for their discovery of the ability of poliomyelitis viruses to grow in cultures of various types of tissue” (Hargittai 2002). From then on, cell culture became a routine laboratory technique. A Colombian woman, Claudia Castillo, who had suffered a tuberculosis infection, became the first person to receive a whole organ transplantation using her own stem cells. This operation, cooperated by scientists and surgeons from Britain, Italy and Spain, gives the patients the choice to get engineering replacements for their damaged organs, such as the bowel or bladder (Macchiarini *et al* 2008).

Cell culture includes both the extraction and implantation processes. In the first place, cells need to be isolated from the tissue for ex vivo culture, which is usually called extraction. Then, they are often implanted or 'seeded' into an artificial structure, typically known as scaffolds. Usually, the scaffold is kept in an incubator which maintains an optimal temperature and humidity for cell growth. For mammalian cells, temperature is typically set at 37 °C, the relative humidity is >95% and a slightly acidic pH is achieved by maintaining a CO_2 level at 5% (Langer & Vacanti 1993).

3D scaffolds can potentially provide flow-induced mechanical stimulation and allow cells to synthesis 3D multilayered extracellular matrix (ECM) (Bancroft *et al* 2002). The scaffolds, in order to supply oxygen and nutrients to the cells as well as the transport of metabolite, are normally porous media with a high porosity and adequate pore size. Meanwhile, some researchers are trying to graft porous scaffold made by biodegradable material in vivo to avoid a second surgery (Hollister 2005; 2006).

Mechanical stimulus are widely used to stimulate cell proliferation and differentiation (Bakker *et al* 2001; Bancroft *et al* 2002; Gutierrez & Crumpler

2008; McAllister & Frangos 1999). Flow rate, wall shear stress, porosity as well as permeability can impact on cell activities (Hollister 2005). At the same flow rate, effects of the wall shear stress can be singled out by altering the fluid viscosity. Using this approach, Bakker *et al* demonstrated that the shear stress is one of the most important mechanical factors on cell proliferation (Bakker *et al* 2001). However, the detailed distribution of the wall shear stress in a scaffold material and the mechanisms involved need much more investigation.

Bancroft found that minor increases in the shear stress was relevant to the augment of the mineralization of the scaffolds compared to the static control sample, and further increases in the shear stress did not improve this effect(Bancroft *et al* 2002). Cartmell *et al* also reported that cell proliferation had been enhanced by the increase of shear stress from the static control, and would be inhibited by further increase of the flow rate (i.e. shear stress, as in the Stokes flow, fluid shear stress is proportional to the flow rate) (Cartmell *et al* 2003).

Lappa presented a numerical model in which the cell growth rate is proportional to the shear stress (Lappa 2003). The validity of the model is yet to be established, since experiment results have shown that the shear stress can enhance as well as inhibit the cell proliferation.

1.2 Porous media

Porous media consist of solid (often called the frame or matrix) and pores (voids) with a liquid or gas. Most pores are usually interconnected to each other so that the fluid can move through them. Many natural substances, such as rocks, bushes, biological tissues and man-made materials (e.g. cements, foams and ceramics) can be considered as porous media.

Porosity and permeability are two of the primary properties which indicate the capability of the storage and movement of fluid in porous media.

Porosity, presenting the storage capability of fluid, is defined as the ratio of the volume of voids to the total volume.

$$\phi = \frac{V_v}{V_t} \tag{1.1}$$

where ϕ, V_v, V_t are the porosity, the volume of void space, the total volume of material, respectively. Obviously, the range of ϕ is between 0 and 1.

Alternatively, the porosity can be got from the density of the current porous medium and the substance.

$$\phi = 1 - \frac{\rho_b}{\rho_s} \tag{1.2}$$

where ρ_b, ρ_s are the bulk density and the substantial density, respectively.

Permeability indicates the ability of the flow through a porous medium, which is influenced by the packing, shape and pore size distribution. For example, a porous medium with a high porosity may be impermeable, if the voids are isolated from each other. It is hard to calculate the permeability theoretically, even with detailed information on the structure. The commonly used measurement is based on flow experiments using the Darcy’s law, as detailed below.

In 1856, Henry Darcy, by conducting experiments of flow through sand, found the relationship between the pressure and the fluid discharge, which is known as the Darcy’s law (Darcy 1856). Figure 1.1 is the demonstration of the experiment: the tube, fully filled with sand, is placed horizontally to avoid the influence of gravity; a reservoir is connected to the left pipe to drive the flow by supplying a constant water pressure, another tank is connected to the right pipe to get the total volume of water during a given time. Using this trivial setup, Darcy found the discharge rate through the porous medium was proportional to the pressure drop, which can be written as

$$\dot{Q} = \frac{KA}{\mu}\frac{P_a - P_b}{L} \tag{1.3}$$

where $\dot{Q}$ is the discharge, K is the permeability, A is the area of cross-section, μ is the dynamic viscosity of fluid, P_a, P_b are the pressure at the inlet and outlet, respectively, L is the length of pipe filled with porous media. By dividing both sides of Equation (1.3) with the surface area A, Darcy’s law can be written in a differential from as

$$\vec{V} = -\frac{K}{\mu}\nabla P \tag{1.4}$$

where $\vec{V}$ is the average velocity through the porous media, ∇P is the pressure gradient, the negative sign means that the flow is from high pressure to low pressure, the opposite direction of pressure gradient.

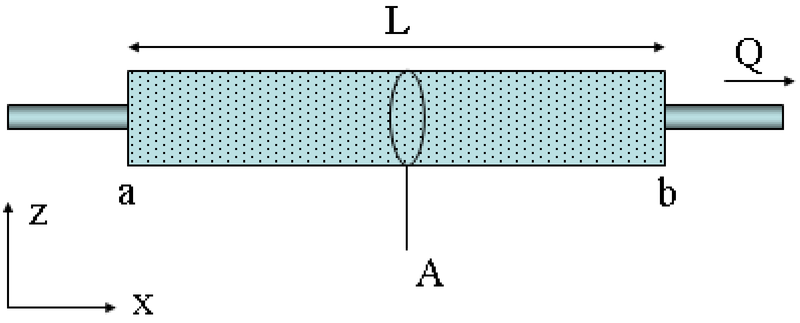


Figure 1.1 The illusion of experiment setup by Darcy in 1856.

Darcy's law is widely used to study flows in homogeneous porous media at low Reynolds numbers (Blokhra & Khajuria 1991; Sen 1989). However, if the Reynolds number is not very small, nonlinear correction term is needed to consider the inertial effect of the flow (Firdaouss *et al* 1997).

1.3 A Brief introduction on the lattice Boltzmann method

Following the introduction of the famous game, "Game of Life" (Gardner 1970), devised by a British mathematician John Conway in 1970, some researchers found the flow simulation can also be mimicked by cellular automaton (Frisch *et al* 1986b; Hardy *et al* 1976). In the so-called FHP model, named by the initials of the authors, two simple steps were applied to the particles: collision and propagation. This scheme was found to be a discretization to the Boltzmann equation, and therefore can introduce the Navier-Stokes equations by Chapman-Enskog expansion (Koelman 1991b; Qian *et al* 1992). Their works, named as lattice gas cellular automata or LGCA for short, offered a new way to model the fluid flow.

To get rid of the noise in FHP (Frisch *et al* 1987) or four-dimensional face-centered-hyper-cubic (Dhumieres *et al* 1986) lattice in lattice gas cellular automata, McNamara suggested to replace the Boolean variables which represent the presence or absence of the particles (Mcnamara & Zanetti 1988). Later, the particle-collision operator has been replaced by the Bhatnagar-Gross-Krook (BGK for shorter) approximation by Koelman (Koelman 1991a), Qian (Qian *et al* 1992) and others. In around 1996, Sterling and He found that lattice Boltzmann equation is a special discretization scheme of Boltzmann equation, which finally established the lattice Boltzmann method on the solid foundation of the kinetic theory (He & Luo 1997b; Sterling & Chen 1996).

After that, the lattice Boltzmann method have been rapidly developed to solve problems such as, multiphase flows (Grunau *et al* 1993; Premnath & Abraham 2005), blood flows (Krafczyk *et al* 1998; Zhang *et al* 2008), flow in porous media (Chen *et al* 1991b; Olson & Rothman 1997), non-Newtonian flows (Boyd *et al* 2006; Yoshino *et al* 2007) and so on.

Figure 1.2 is the annual number of publications using the lattice Boltzmann method. The exponent increase implies the potential application of this method and the rising interesting from investigators. The 2010 data is only to August 2010 with delayed update of the database.

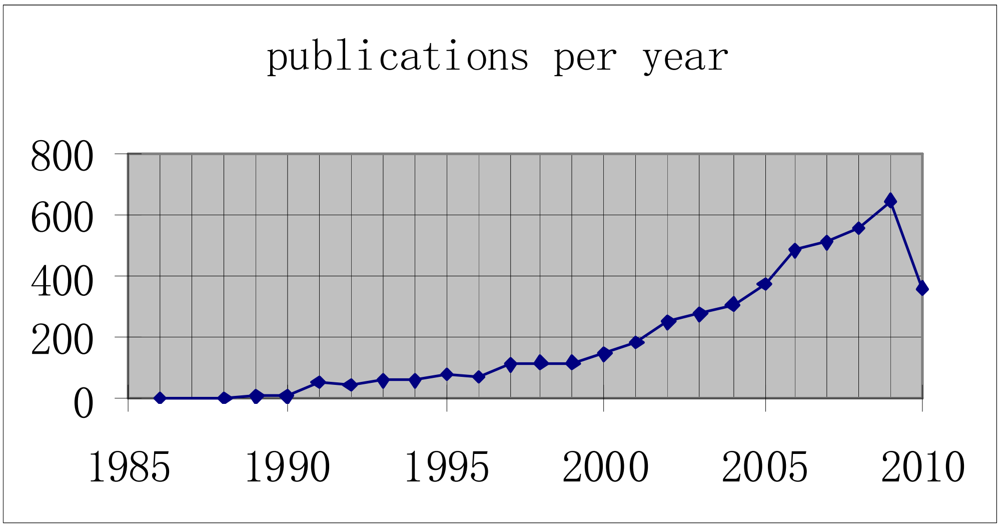


Figure 1.2 Annual number of publications using the lattice Boltzmann method. The graph is generated using the ISI Web of Science digital databases in August 2010.

1.4 Aim and objectives

This project aims to apply the lattice Boltzmann method to model fluid velocity and shear stress distribution in highly porous scaffold materials that are used in cell & tissue engineering and to investigate effects of parameters, such as the flow rate, pressure on detailed flow patterns in the scaffolds.

1.4.1 Methodology - why the lattice Boltzmann method?

The tradition numerical method such as the finite volume method (FVM), finite element method (FEM), finite difference method (FDM) have been well developed to solve Navier-Stokes equation in flow simulation problems (Kumar & Naidu 1995; Nadobny *et al* 2007; Shibeshi & Collins 2005). They can provide detailed information on flow in blood vessels (Sun *et al* 2009; Torii *et al* 2009), with encouraging progresses on patient- specific CFD simulations based on computational tomography images (Cheng *et al*). The effects of wall shear stress on cell growth has been studied by these methods (Lappa 2003). The Lappa's model assumed the cells as an additional fluid phase and calculated the multiphase flow field using a finite volume method, with the volume-of-fluid (VOF) method and level-set method to capture the interface. On top of other limitations in the traditional methods, the simulation needs to re-mesh the grid to accommodate the cell growth, which adds to extra computation requirement.

Unlike the convention methods above, the lattice Boltzmann method does not need to re-mesh the grid at all. Besides, LBM is second-order accurate for the shear stress as well as the velocity. So it has been extensively used to solve for flows in porous media or in fluid-solid coupling problems (Premnath & Abraham 2005; Spaid & Phelan 1997; Zhang *et al* 2008). However, it is necessary to employ interpolation or extrapolation for the treatment of the solid boundary, which will affect LBM's accuracy. Furthermore, it was found that the lattice Boltzmann method provided comparable accuracy under lower expenses in terms of CPU time compared to commercially available finite volume/element software (Geller *et al* 2006). Moreover, the accuracy of the shear stress calculation using the lattice Boltzmann method is of the second order, due to the fact that shear stress calculation is independent of the velocity (Kruger *et al* 2009).

Cellular automaton has been used to investigate cell proliferation and immigration (Cheng *et al* 2006; Lee *et al* 1995), in which the convection-diffusion process of the nutrition supply is solved by finite difference method (Chung *et al* 2010). Since the lattice Boltzmann method is a special case of cellular automaton (Chen & Doolen 1998), it is intrinsically compatible with the general cellular automaton.

1.4.2 Objectives of the project

The first objective of this study is to develop an efficient program based on the lattice Boltzmann method that can read the micro-CT images of real porous media, reconstruct the porous structure, model the fluid flow under given parameters such as boundary condition, density and viscosity.and output the calculation results to other post-process software. To achieve this, Matlab (Version 2008a, licensed), a high-level language including the package of graphs, is used for the preprocess; FORTRAN 95, a general-purpose programming language which is especially suitable for high-performance scientific computation, is used in the main part of the software to reduce the simulation time; Tecplot (Version 360, licensed), another commercial software, is used for the flow analysis and the presentation of the results.

2D simulation is carried out first and is then extended to 3D simulation based on realistic geometries of a porous scaffold using the lattice Boltzmann method.

The other objective of the project is to develop a numerical model that is capable to simulation the cell growth based on the relationship to mechanical factors such as pressure gradient and wall shear stress.

In all studies, validation of the program has been carried out using either available analytical/asymptotic results or other numerical softwares. The porous geometry is taken from the micro-CT images in our own laboratory.

1.5 A brief overview of the structure of the thesis

The thesis is organized in the following way:

Chapter 1 reviews the background on cell culture and the lattice Boltzmann method. It also lays out the objectives of this report and the advantage of using the lattice Boltzmann method in the study.

Chapter 2 gives details on the development of the lattice Boltzmann method, and derives the Navier-Stokes equations from the lattice Boltzmann method by Chapman-Enskog expansion. In addition, different treatments of boundary conditions are also presented.

In Chapter 3, the program developed based on the lattice Boltzmann method is validated by modeling Poiseuille flows in a planar channel and in a rectangular duct.

2D flow simulation in a porous structure based on a 2D micro-CT image is presented in Chapter 4.

Chapter 5 presents cell proliferation study in the 2D structure, a simplified relationship between the cell growth and local shear stress is proposed. The purpose of this chapter is to demonstrate the capacity of our program based on the lattice Boltzmann method, rather than to demonstrate the actual cell proliferation process.

3D flow simulation in a reconstructed porous scaffold based on micro-CT images is presented in Chapter 6. In the preliminary study, a small volume in the middle of scaffold is used, rather than the whole scaffold.

Following the list of references, the main code of 2D lattice Boltzmann method developed in FORTRAN program language is given in Appendix.

2 Lattice Boltzmann Method

2.1 Introduction

The lattice Boltzmann method originates from the lattice gas cellular automaton, which is a discrete particle with some certain rules. The first lattice gas cellular automata can be dated back in 1973, when Hardy, de Pazzis and Pomeau (known as HPP model after the initials of three authors) proposed a simple two dimensional model on a square lattice (Hardy *et al* 1973). Today, the HPP model is mainly of historical interest because it does not lead to the correct Navier-Stokes equations in macroscopic limit due to the deficiency of rotational symmetry (Wolf-Gladrow 2000). However, the collision and propagation processes introduced by HPP model are key features for all lattice gas cellular automata models for the fluid simulation ever since, including the lattice Boltzmann method. The evolution equation of the lattice gas cellular automata is

$$n_i(\vec{x}+\vec{e}_i,t+1)=n_i(\vec{x},t)+\Omega_i \qquad (2.1)$$

where $n_i(\vec{x},t)$ is a set of Boolean variables describing the presence and absence of particles, $\vec{e}_i$ is the local particle velocities, and Ω_i are the local collision rules predefined based on the particles status.

In 1986, by changing the square lattice to hexagonal lattice and by introducing a more complex collision rules, Frisch *et al* proposed a very simple lattice gas cellular automata method which was able to account for the complexity of real fluid flows (Frisch *et al* 1986b). This was also independently proposed by Wolfram in the same year (Wolfram 1986). The so-called FHP model meshes the 2D computation domain with hexagonal lattice. Each lattice sites up to six particles with identity mass and these particles can only move along one of the six directions which are the line connections to the neighboring lattice as shown in Figure 2.1. In a time step, these particles travel to their neighbors and certain collision rules are implemented based on the state of the lattice. There are two collision rules in their pioneering work: one for the two body collision and another for the three body collision, as depicted in Figure 2.2. It has been shown that this simple model obeys the incompressible Navier-Stokes equations after coarse-grain process (Koelman 1991b; Qian *et al* 1992). The computation fluid dynamics community shows great interests in lattice gas cellular automata, and more than a thousand papers have cited Frisch *et al*’s work so far, according to the “Web of Science” database. A four-dimensional face-centered-hyper-cubic (FCHC) lattice was proposed by d’Humieres *et al* to model the 3D fluid dynamics (d'Humieres *et al* 1986).

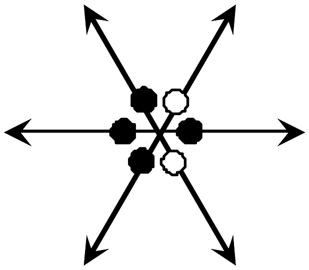
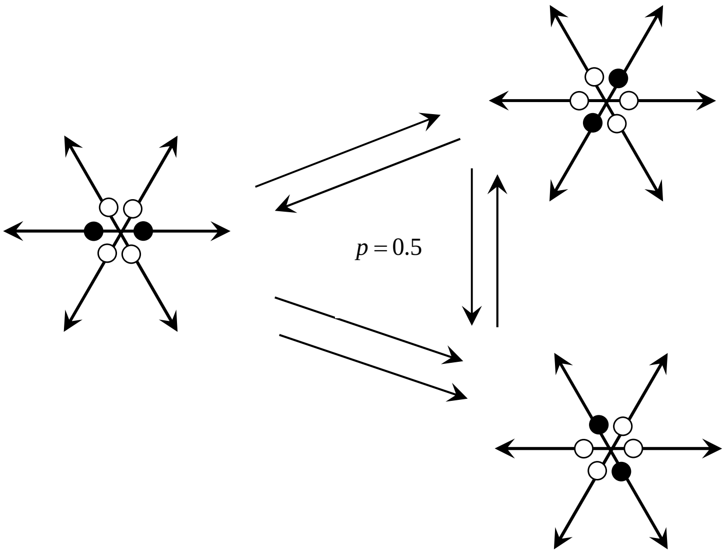


Figure 2.1 A sketch of FHP lattice. Up to six particles can be sited in the lattice. There are four fluid particles in current lattice presented by black solid circles and two voids presented by white circles. The arrows are the moving directions of the particles.



a) 2-particle head-on collision rules

b) 3-particle collision rules

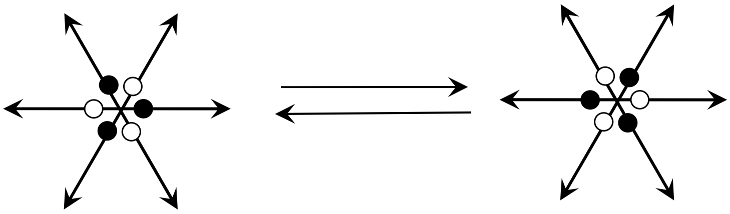


Figure 2.2 The collision rules for FHP model. a) If there are 2 fluid particles (dark solid circle) in the lattice opposite to each other, after collision, these 2 fluid particles will be rotated by 60°to left or right with equal probability. b) If there are 3 fluid particles with 120°to each other, after collision, these 3 fluid particles will change their positions with the voids.

Despite the remarkable achievement in the late 80s, a number of serious intrinsic problems remain unresolved in the lattice gas algorithms. For example, the exclusion principle in which only two statuses (occupied or unoccupied, fluid or void) are allowed at a certain node leads to a Fermi-Dirac distribution for the local equilibrium particles instead of Maxwell distribution in fluid dynamics (Frisch *et al* 1986a). The usage of Boolean variables can be easily implemented with parallel computers without any round-off error; on the other hand, coarse grain process is required to get the macroscopic variables, such as density and velocity. The artificial collision rules are very hard to construct and usually do not have any physical meanings (Wolf-Gladrow 2000).

To reduce and to remove the statistic fluctuations of lattice gas cellular automata (Orszag & Yakhot 1986), McNamara and Zanetti introduced lattice Boltzmann model by replacing the Boolean variables with Feimi-Dirac distribution functions which is real variables (Mcnamara & Zanetti 1988). A linear collision operator was proposed to simplify the collision rules by assuming that the distribution was very close to the local equilibrium state (Higuera & Jimenez 1989). The Bhatnagar-Gross-Krook (short as BGK afterwards) operator (Bhatnagar *et al* 1954), suggested by several groups independently (Chen *et al* 1991a; Qian *et al*. 1992), has been popularly adapted in the lattice Boltzmann simulation. In this thesis, the lattice BGK model has been applied to derive the Navier-Stokes equations and to mimic the flow in porous scaffolds.

2.2 Lattice Boltzmann method to Navier-Stokes equations

The lattice Boltzmann method with BGK approximation can be written as

$$F_i(\vec{\boldsymbol{x}}+\vec{\boldsymbol{c}}_i\Delta t, t+\Delta t) = F_i(\vec{\boldsymbol{x}},t) - \frac{1}{\tau}\left(F_i(\vec{\boldsymbol{x}},t) - F_i^{eq}(\rho,\vec{u})\right) \tag{2.2}$$

where $F_i(\vec{x},t)$ is the distribution function in site $\vec{x}$ at time t, τ is the dimensionless relaxation time, c_i is the lattice velocity defined as the ratio to the lattice length and step time Δt, and $F_i^{eq}(\rho,\vec{u})$ is the equilibrium distribution function.

Analog to the lattice gas cellular automata, the implement of lattice Boltzmann method can be separated into two steps, which are usually called collision step and streaming (propagation) step, respectively.

$$\begin{aligned} &\text{collision:} \quad \tilde{F}_i(\vec{\boldsymbol{x}},t) = F_i(\vec{\boldsymbol{x}},t) - \frac{1}{\tau}\left(F_i(\vec{\boldsymbol{x}},t) - F_i^{eq}(\rho,\vec{u})\right) \\ &\text{streaming:} \qquad F_i(\vec{\boldsymbol{x}}+\vec{\boldsymbol{c}}_i\Delta t, t+\Delta t) = \tilde{F}_i(\vec{\boldsymbol{x}},t) \end{aligned} \tag{2.3}$$

where $\tilde{F}_i(\vec{x},t)$ is the post-collision value. The collision step can be implemented locally, and particles travel to their corresponding neighbour lattice in the streaming step.

The macroscopic values, local mass density ρ and the momentum density j, are defined as the sum over the distribution at each lattice.

$$\begin{aligned} \rho &= \sum_i F_i(x,t) \\ j &= \rho u = \sum_i c_i F_i(x,t) \end{aligned} \tag{2.4}$$

To derive the Navier-Stokes equations from lattice Boltzmann equation, Chapman-Enskog expansion is used by assuming the time for diffusion process is much slower than that of convention process (Chapman *et al* 1970; Rivet & Frisch 1986). Likewise, the distribution function $F_i(\vec{x},t)$ are expanded around the equilibrium distributions $F_i^{(0)}(x,t)$.

$$F_i(x,t) = F_i^{(0)}(x,t) + \varepsilon F_i^{(1)}(x,t) + \varepsilon^2 F_i^{(2)}(x,t) + o(\varepsilon^3) \tag{2.5}$$

where ε is a small parameter commonly used in asymptotic analysis.

Also, it is assumed that only the zero-th term has the contribution to the density ρ and momentum j. other perturbations ($F_i^{(1)}(x,t)$, $F_i^{(2)}(x,t)$, and etc) do not contribute to the mass and momentum at all.

$$\begin{aligned} &\sum_i F_i^{(0)}(x,t) = \rho, \quad \sum_i c_i F_i^{(0)}(x,t) = j \\ &\sum_i F_i^{(n)}(x,t) = 0, \quad \sum_i c_i F_i^{(n)}(x,t) = 0, \quad n > 0 \end{aligned} \tag{2.6}$$

The small parameter ε can be Knudsen number which is the ratio between the mean free path and the characteristic length scale of the flow or the time step Δt, here we adapt the setting in He and Luo's work. (He & Luo 1997a).

By introducing the following expansions

$$\begin{aligned} &F_i(x+\Delta x, t+\Delta t) = \sum_{n=0}^{\infty} \frac{\varepsilon^n}{n!} D_t^n F_i(x,t) \\ &\partial_t = \sum_{n=0}^{\infty} \varepsilon^n \partial_t^{(n)} \end{aligned} \tag{2.7}$$

where $D_t \equiv (\partial_t + c_i \cdot \nabla)$ and $\varepsilon = \Delta t$, we can rewrite the lattice Boltzmann equation (2.2) in the consecutive order of the parameter ε as follows

$$O(\varepsilon^0): \quad F_i^{(0)} = F_i^{eq} \tag{2.8}$$

$$O(\varepsilon^1): \quad D_t^{(0)} F_i^{(0)} = -\frac{1}{\tau} F_i^{(1)} \tag{2.9}$$

$$O(\varepsilon^2): \quad \partial_t^{(1)} F_i^{(0)} + D_t^{(0)} F_i^{(1)} + \frac{1}{2}\left(D_t^{(0)}\right)^2 F_i^{(0)} = -\frac{1}{\tau} F_i^{(2)} \tag{2.10}$$

Substituting Equation (2.9) into Equation (2.10), we can simplify it as

$$O(\varepsilon^2): \quad \partial_t^{(1)} F_i^{(0)} + \left(\frac{2\tau - 1}{2\tau}\right) D_t^{(0)} F_i^{(1)} = -\frac{1}{\tau} F_i^{(2)} \tag{2.11}$$

The n-th lattice tensor is defined as

$$E^{(n)} = \sum_i W_i \vec{c}_{i,1} \vec{c}_{i,2} \ldots \vec{c}_{i,n} \tag{2.12}$$

where $\vec{c}_{i,k}$ is the $k-$th lattice velocity of c_i, and its value depends on the specific model used. For D2Q9 model which will be mentioned in the following section, the lattice tensors have the following properties:

$$\begin{aligned} &E^{(0)}=1\\ &E^{(2)}=\frac{1}{3}c^2\delta_{ij}\\ &E^{(4)}=\frac{1}{9}c^4\delta_{ijkl}\\ &E^{(2k+1)}=0 \quad k>0 \end{aligned} \tag{2.13}$$

where δ_{ij} and δ_{ijkl} are the Kronecker delta function with two and four indices, respectively.

$$\begin{aligned} &\delta_{ij}=\begin{cases}1 & i=j\\ 0 & i\neq j\end{cases}\\ &\delta_{ijkl}=\delta_{ij}\delta_{kl}+\delta_{ik}\delta_{jl}+\delta_{il}\delta_{jk} \end{aligned} \tag{2.14}$$

By applying the properties of the lattice tensor $E^{(n)}$, we have

$$\begin{aligned} &\sum_i F_i^{eq}=\rho\\ &\sum_i c_{i\alpha}F_i^{eq}=\rho u_\alpha\\ &\sum_i c_{i\alpha}c_{i\beta}F_i^{eq}=\frac{1}{3}c^2\rho\delta_{\alpha\beta}+\rho u_\alpha u_\beta\\ &\sum_i c_{i\alpha}c_{i\beta}c_{i\gamma}F_i^{eq}=\frac{1}{3}c^2\rho(\delta_{\alpha\beta}u_\gamma+\delta_{\beta\gamma}u_\alpha+\delta_{\gamma\alpha}u_\beta) \end{aligned} \tag{2.15}$$

where the Greek index α,β represent the spatial axis, and Latin index i represents one of the 9 lattice components.

The first two equations are the same as the first two of Equation (2.6) if Equation (2.8) is introduced. Sum up all the 9 components of F_i in Equation (2.9), and using the properties of lattice tensors stated in Equation (2.15), we get

$$\partial_t^{(0)}\rho+\partial_\alpha(\rho u_\alpha)=0 \tag{2.16}$$

and

$$\partial_t^{(0)}\rho u_\alpha + \partial_\beta \Pi_{\alpha\beta}^{(0)} = 0 \tag{2.17}$$

where $\Pi_{\alpha\beta}^{(0)} = \sum_i c_{i\alpha} c_{i\beta} F_i^{eq}$ is called as the zero-th momentum flux tensor.

Following the same step, for Equation (2.11), we get

$$\partial_t^{(1)}\rho = 0 \tag{2.18}$$

and

$$\partial_t^{(1)}\rho u_\alpha + \left(\frac{2\tau - 1}{2\tau}\right)\partial_\beta \Pi_{\alpha\beta}^{(1)} = 0 \tag{2.19}$$

where $\Pi_{\alpha\beta}^{(1)} = \sum_i c_{i\alpha} c_{i\beta} F_i^{(1)}$ is named as the first order momentum flux tensor, which can be expressed with the aid of Equation (2.9) and Equation (2.16) as

$$\begin{aligned}
\Pi_{\alpha\beta}^{(1)} &= \sum_i c_{i\alpha} c_{i\beta} F_i^{(1)} = -\tau \sum_i c_{i\alpha} c_{i\beta} D_t^{(0)} F_i^{(0)} \\
&= -\tau \sum_i \left(\partial_t^{(0)} c_{i\alpha} c_{i\beta} F_i^{(0)} + \nabla \cdot c_{i\alpha} c_{i\beta} c_{i\gamma} F_i^{(0)}\right) \\
&= -\tau \sum_i \left[\partial_t^{(0)}\left(\frac{1}{3}c^2 \rho \delta_{\alpha\beta} + \rho u_\alpha u_\beta\right) + \nabla \cdot \left(\frac{1}{3}c^2 \rho (\delta_{a\beta} u_\gamma + \delta_{\beta\gamma} u_\alpha + \delta_{\gamma\alpha} u_\beta)\right)\right] \\
&= -\tau \sum_i \left[\partial_t^{(0)}\left(\frac{1}{3}c^2 \rho \delta_{\alpha\beta} + \rho u_\alpha u_\beta\right) + \frac{1}{3}c^2 \left(\partial_\gamma \rho u_\gamma \delta_{\alpha\beta} + \partial_\alpha \rho u_\beta + \partial_\beta \rho u_\alpha\right)\right] \\
&= -\tau \sum_i \left[\partial_t^{(0)}\left(\rho u_\alpha u_\beta\right) + \frac{1}{3}c^2 \left(\partial_\alpha \rho u_\beta + \partial_\beta \rho u_\alpha\right)\right]
\end{aligned} \tag{2.20}$$

It is easy to find that the first term is of order $O(M^3)$, and can be neglected. Therefore, by assembling all the expressions we got so far, we have

$$(\partial_t^{(0)} + \varepsilon \partial_t^{(1)})\rho + \partial_\alpha \rho u_\alpha = 0 \tag{2.21}$$

and

$$(\partial_t^{(0)} + \varepsilon \partial_t^{(1)})\rho u_\alpha + \partial_\beta \left(\Pi_{\alpha\beta}^{(0)} + \varepsilon \frac{2\tau - 1}{2\tau} \Pi_{\alpha\beta}^{(1)} \right) = 0 \tag{2.22}$$

By setting $\varepsilon = 1$ and assuming that the density variation is very small, after some simple manipulations, we get the so-called continuity equation and Navier-Stokes equations, respectively.

$$\frac{\partial \vec{u}}{\partial t} + \nabla \cdot \vec{u} = 0 \tag{2.23}$$

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \nabla \cdot \vec{u} = -\nabla p + \nu \nabla^2 \vec{u} \tag{2.24}$$

with the kinetic viscosity as

$$\nu = \frac{2\tau - 1}{6} \tag{2.25}$$

From the derivation, we can also get the relationship between shear rate tensor $\varepsilon_{\alpha\beta}$ and the first order momentum flux tensor $\Pi_{\alpha\beta}^{(1)}$ as

$$\varepsilon_{\alpha\beta} = \frac{1}{2}\left(\partial_\beta u_\alpha + \partial_\alpha u_\beta\right) = -\frac{3}{2\rho\tau} \Pi_{\alpha\beta}^{(1)} \tag{2.26}$$

Thus the stress tensor $\tau_{a\beta}$ of Newtonian fluid is

$$\begin{aligned} \tau_{a\beta} &= -p\delta_{\alpha\beta} + 2\mu\varepsilon_{\alpha\beta} \\ &= -p\delta_{\alpha\beta} - \left(1 - \frac{1}{2\tau}\right)\Pi_{\alpha\beta}^{(1)} \end{aligned} \tag{2.27}$$

From Equation (2.27), we can find that the shear stress can be calculated from the differences between local distribution function and equilibrium values, and is not relevant to the calculation of velocity values.

2.3 2D lattice Boltzmann model

From the derivation of the Navier-Stokes equation in the last section, we have found the importance of the lattice tensor $E^{(n)}$. A proper lattice tensor should meet the requirement of isotropy of the 2nd and 4th ranks (Wolfram 1986). For example, the lack of isotropy of rank 4 of the HPP lattice tensor fails to yield the Navier-Stokes equations in the macroscopic limit (Wolf-Gladrow 2000).

D2Q9 model is the most popular two-dimension lattice Boltzmann model with a rest particle in the center and 8 active particles with different directions.

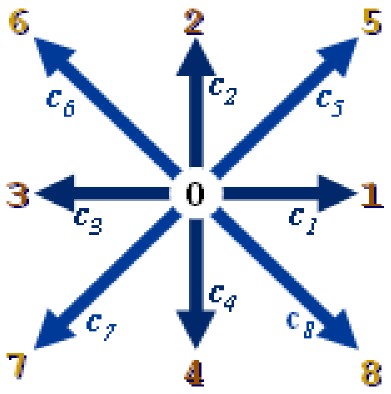


Figure 2.3 The 9-component distribution function.

As depicted in Figure 2.3, the 9 components of different directions in D2Q9 model can be expressed as

$$c_i = \begin{cases} (0,0) & i=0 \\ (\cos[(i-1)\pi/2], \sin[(i-1)\pi/2])c & i=1,2,3,4 \\ (\cos[(i-5)\pi/2+\pi/4], \sin[(i-5)\pi/2+\pi/4])\sqrt{2}c & i=5,6,7,8 \end{cases} \quad (2.28)$$

The subscript index is traditionally from 0 to 8.

The equilibrium distribution of "virtual" fluid particles should obey the

Maxwell-Boltzmann distribution

$$F^{eq} = \rho \left(\frac{m}{2\pi k_B T} \right)^{3/2} \exp(-mu^2 / 2k_B T) \qquad (2.29)$$

where m is the mass of the particle, k_B is the Boltzmann constant, T is the temperature, and u is velocity. If the velocity is very small, we can expand it in Taylor series as

$$F_i^{eq}(\rho, \vec{u}) = W_i \left\{ \rho + \rho \frac{m}{k_B T} c_i \cdot \vec{u} + \rho \frac{m}{2k_B T} \left[\frac{m}{k_B T} (c_i \cdot \vec{u})^2 - \vec{u}^2 \right] \right\} \qquad (2.30)$$

or more explicitly

$$\begin{aligned} F_i &= \frac{4}{9} \rho \left[1 - \frac{3u^2}{2c^2} \right] & i &= 0 \\ F_i &= \frac{1}{9} \rho \left[1 + 3\frac{c_i \cdot \vec{u}}{c^2} + \frac{9(c_i \cdot \vec{u})^2}{2c^4} - \frac{3u^2}{2c^2} \right] & i &= 1, 2, 3, 4 \qquad (2.31) \\ F_i &= \frac{1}{36} \rho \left[1 + 3\frac{c_i \cdot \vec{u}}{c^2} + \frac{9(c_i \cdot \vec{u})^2}{2c^4} - \frac{3u^2}{2c^2} \right] & i &= 5, 6, 7, 8 \end{aligned}$$

with $\frac{k_B T}{m} = \frac{c^2}{3}$. We will give a brief introduction to the calculation of the weight value W_i in the following section.

2.4 3D lattice Boltzmann method

The extension of 2D lattice Boltzmann model to 3D one is rather straightforward. One just needs to choose a certain 3D lattice mode and calculate the corresponding weight. d’Humieres *et al* proposed a multispeed lattice-gas cellular automata over a cubic lattice with 19 velocities which is called D3Q19 model (d'Humieres *et alet al* 1986). As shown in Figure 2.4, there are three different speeds in D3Q19 model: 1 rest particle in the center with zero speed, 6 particles with 1 lattice speed (black arrows), and 12 particles with $\sqrt{2}$ lattice speed (red arrows). The lattice velocity can be written as

$$\vec{c}_i = c \times \begin{cases} (0,0) & i=0 \\ (\pm1,0,0),(0,\pm1,0),(0,0,\pm1) & i=1,2,\ldots,5,6 \\ (\pm1,\pm1,0),(\pm1,0,\pm1),(0,\pm1,\pm1) & i=7,8,\ldots,17,18 \end{cases} \tag{2.32}$$

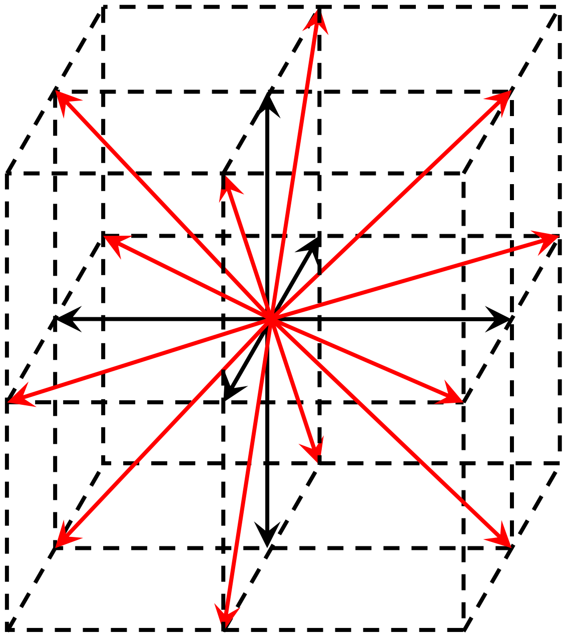


Figure 2.4 The lattice velocities of the D3Q19 model.

The local equilibrium distribution F_i^{eq}, which is a function of local value of density and velocity, can be derived theoretically by applying the maximum entropy principle under the conservation of mass and momentum (Karlin *et al* 1998). Alternatively, Keolman proposed a general and simple method to calculate the weight W_i for particles with different lattice speed

(Koelman 1991b). First, The Taylor expansion of equilibrium distribution F_i^{eq} up to second order of velocity from the Maxwell-Boltzmann distribution function (2.29) can be written as

$$F_i^{eq}(\rho,\vec{u}) = W_i \left\{ \rho + \rho \frac{m}{k_B T} c_i \cdot \vec{u} + \rho \frac{m}{2k_B T} \left[\frac{m}{k_B T} (c_i \cdot \vec{u})^2 - \vec{u}^2 \right] \right\} \qquad (2.33)$$

where W_i is the weight factor, ρ is the local density, m is the mass, k_B is the Boltzmann constant as recalled.

The velocity momentum tensor up to fourth order should equal to those of the continuum Boltzmann distribution, which leads to

$$\begin{aligned} &\sum_\alpha F_\alpha^{eq} = \int f(u)du = \rho \\ &\sum_\alpha c_{i\alpha} c_{j\alpha} F_\alpha^{eq} = \int f(u) u_i u_j du = \rho \frac{k_B T}{m} \delta_{ij} \\ &\sum_\alpha c_{i\alpha} c_{j\alpha} c_{k\alpha} c_{n\alpha} F_\alpha^{eq} = \int f(u) u_i u_j u_k u_n du \\ &= \rho \left(\frac{k_B T}{m} \right)^2 \left(\delta_{ij}\delta_{kn} + \delta_{ik}\delta_{jn} + \delta_{in}\delta_{kj} \right) \end{aligned} \qquad (2.34)$$

After some manipulations, we can get the solution to the constraint Equation (2.34) as

$$\begin{aligned} &W_i = \begin{cases} \frac{1}{3} & i = 0 \\ \frac{1}{18} & i = 1,2,...,5,6 \\ \frac{1}{36} & i = 7,8,...,17,18 \end{cases} \\ &\frac{k_B T}{m} = \frac{c^2}{3} \end{aligned} \qquad (2.35)$$

2.5 Boundary conditions

The implement of boundary conditions is necessary and very important to any numerical simulation, and will influence the accuracy and stability of the schemes. Extensive tests have been carried out to investigate the behavior of various boundary conditions (Ahrenholz *et al* 2006; Kao & Yang 2008; Mei *et al* 1999; Pan *et al* 2006; Wagner & Pagonabarraga 2002; Zou & He 1997). Unlike the finite element method or finite volume method, which solves the velocity directly from Navier-Stokes equations, in lattice Boltzmann method, we are going to solve the distribution function $F(x,t)$. It is a challenge to implement the traditional velocity or pressure conditions. Even for the no-slip boundary condition at the solid surface, more research is needed. In this section, we will give a brief introduction to the treatments of various boundary conditions using D2Q9 model.

2.5.1 Velocity and pressure boundary condition

A widely used scheme to deal with velocity and pressure boundary condition was proposed by Zou and He, by extending the bounce back scheme to the non-equilibrium part in 1997 (Zou & He 1997). One single case, i.e. an east boundary treatment is introduced here, while the others can be implemented similarly.

a. Velocity inlet condition in the east boundary

As an example, if we want to apply a velocity inlet boundary condition in the east boundary as shown in Figure 2.5. After propagation, the values of $F_0, F_2, F_3, F_4, F_6, F_7$ are known since they travel from the neighbor lattices in the inner domain, while the values of F_1, F_5, F_8 are unknown. Suppose the horizontal velocity $u = u_0$ and vertical velocity $v = v_0$. 4 equations are needed to work out the denisty ρ and distribution function components F_1, F_5, F_8.

From the density formula, we have

$$\rho = \sum_{i=0}^{8} F_i \tag{2.36}$$

The formulas of x- and y- direction velocities give another two

$$\rho u_0 = \sum_{i=0}^{8} c_{ix} F_i \tag{2.37}$$

$$\rho v_0 = \sum_{i=0}^{8} c_{iy} F_i \tag{2.38}$$

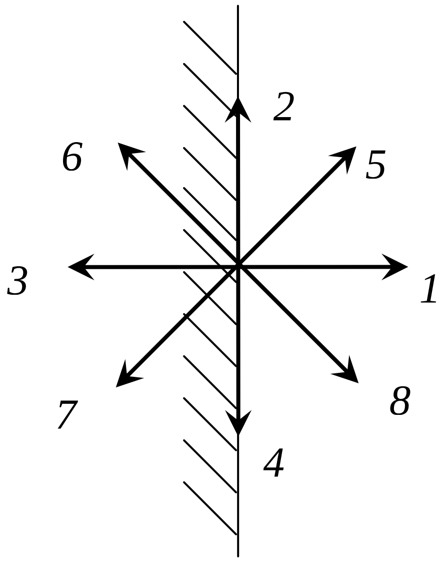


Figure 2.5 Illustration of the lattices after streaming at the east boundary, the distribution functions F_1, F_5, F_8 are unknown.

Zou and He proposed the fourth equation by assuming that the bounce back condition of non-equilibrium parts holds in the direction normal to the boundary

$$f_1 - f_1^{eq} = f_3 - f_3^{eq} \tag{2.39}$$

After some manipulations, we can work out

$$\begin{aligned}
&\rho = \frac{2(F_3 + F_6 + F_7) + (F_0 + F_2 + F_4)}{1 - u_0} \\
&F_1 = F_3 + \frac{2}{3}\rho u_0 \\
&F_5 = F_7 - \frac{1}{2}\left(F_2 - F_4\right) + \frac{\rho u_0}{6} + \frac{\rho v_0}{2} \\
&F_8 = F_6 + \frac{1}{2}\left(F_2 - F_4\right) + \frac{\rho u_0}{6} - \frac{\rho v_0}{2}
\end{aligned} \tag{2.40}$$

b. Pressure inlet condition in the west boundary

To implement the pressure inlet condition $p = p_{in}$, first, we need convert it into density ρ_{in} which is based on the extremely simple state equation of ideal gas. Besides, the value of the velocity tangent to the boundary, the y-component of velocity v_0 here, should also be given to close the equation. Analog to the analysis of velocity inlet condition above, we can get the four equations for the pressure inlet condition

$$\begin{aligned}
&\rho = \sum_{i=0}^{8} F_i \\
&\rho u_0 = \sum_{i=0}^{8} c_{ix} F_i \\
&\rho v_0 = \sum_{i=0}^{8} c_{iy} F_i = 0 \\
&f_1 - f_1^{eq} = f_3 - f_3^{eq}
\end{aligned} \tag{2.41}$$

And the unknown variables u_0, F_1, F_5, F_8 can be solved

$$\begin{aligned}
&u_0 = 1 - \frac{2(F_3 + F_6 + F_7) + (F_0 + F_2 + F_4)}{\rho_{in}} \\
&F_1 = F_3 + \frac{2}{3}\rho u_0 \\
&F_5 = F_7 - \frac{1}{2}\left(F_2 - F_4\right) + \frac{\rho u_0}{6} \\
&F_8 = F_6 + \frac{1}{2}\left(F_2 - F_4\right) + \frac{\rho u_0}{6}
\end{aligned} \tag{2.42}$$

2.5.2 Wall boundary condition

Unlike traditional methods that apply the velocity (Dirichlet) or flux (Von Newman) conditions directly on the boundary, there is no corresponding, physically based boundary condition for the distribution function F_i in the mesoscopic level. For the regular meshes used in lattice Boltzmann method, truncation error will raise from the exact solid boundary, especially for the curved boundary. These make it non-trivial to implement the accurate wall boundary condition in lattice Boltzmann method, and the challenge remains an open one (Latt *et al* 2008). The bounce back scheme, which is a particularly straightforward approach from lattice gas cellular automata, was proposed at the very beginning of LBM to model no-slip conditions on solid surfaces (Lavallee *et al* 1991). In this scheme, as implied by its name, the "virtual" particle which reaches the solid boundary will reverse immediately. The bounce back condition was found only first-order in numerical accuracy at the wall boundary (Cornubert *et al* 1991; Ginzbourg & Adler 1994; Ziegler 1993). To improve the accuracy of zigzag approximation of curved boundary, several schemes have been proposed using interpolation/extrapolation treatment (Chun & Ladd 2007; Junk & Yang 2005; Kao & Yang 2008; Verschaeve 2009). A brief introduction will be given to the treatments of wall boundary condition.

a. Bounce back scheme

The process is illustrated in Figure 2.6. At time step t, the distribution functions F_4, F_7, F_8 travel to their corresponding solid neighbors in the steaming process, and immediately reflect back at the solid boundary, at the time step $t + \Delta t$, these vectors meet at the original lattice with same magnitude but opposite directions. It was found that if the boundary was moved to the half mesh unit between the fluid and solid interface, the bounce back scheme is second order accurate for straight wall and flow in the porous media (Pan *et al* 2006).

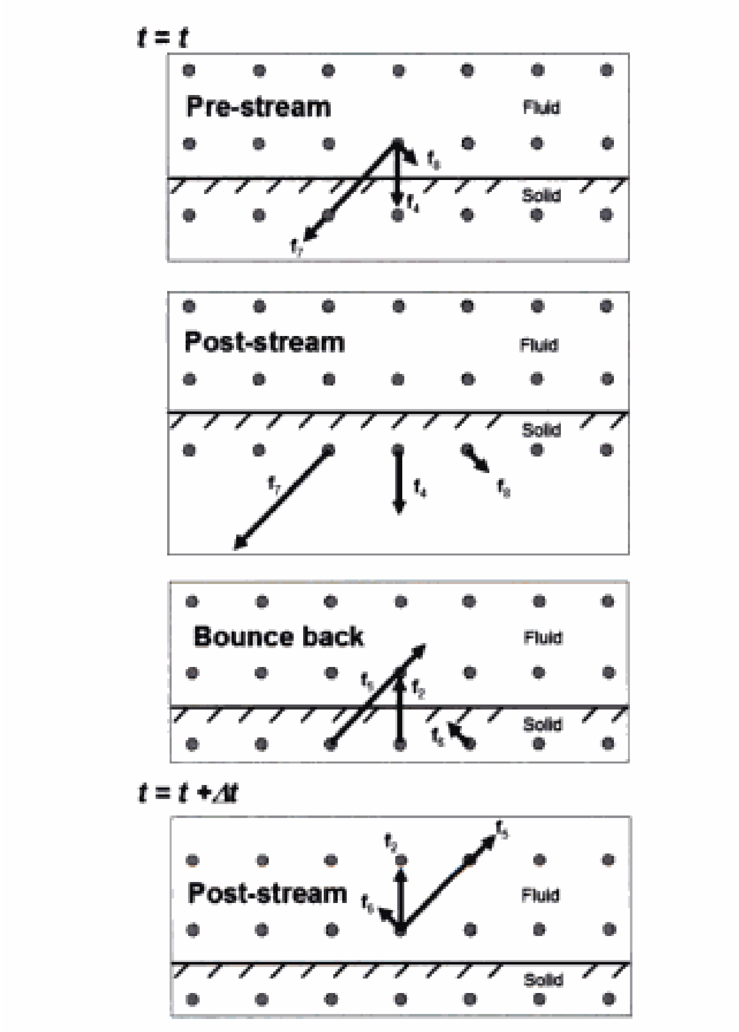


Figure 2.6 The treatment of bounce back scheme.

As already mentioned, the numeric implement of bounce back scheme on node (i, j) illustrated in Figure 2.6 is shown in Table 2.1, where $temp$ is a temporal variable, $F_k(x_{ij}, t)$ is the distribution function on node (i, j) as defined before.

Table 2.1 Algorithm of bounce back scheme.

$temp = F_1(x_{ij}, t), \quad F_1(x_{ij}, t) = F_3(x_{ij}, t), \quad F_3(x_{ij}, t) = temp$
$temp = F_2(x_{ij}, t), \quad F_2(x_{ij}, t) = F_4(x_{ij}, t), \quad F_4(x_{ij}, t) = temp$

$temp = F_5(x_{ij},t), \quad F_5(x_{ij},t) = F_7(x_{ij},t), \quad F_7(x_{ij},t) = temp$
$temp = F_6(x_{ij},t), \quad F_6(x_{ij},t) = F_8(x_{ij},t), \quad F_8(x_{ij},t) = temp$

b. Bouzidi’s scheme

However, if the boundary is curved (Figure 2.7), the simple bounce back scheme will treat it as some zig-zag approximation, which will obviously introduce inaccuracy result. To solve this issue, Bouzidi proposed an interpolation approach to catch the accurate solid boundary for the no-slip boundary condition (Bouzidi *et al* 2001). Later, Lallemand *et al* applied the same method to the moving boundary problem (Lallemand & Luo 2003).

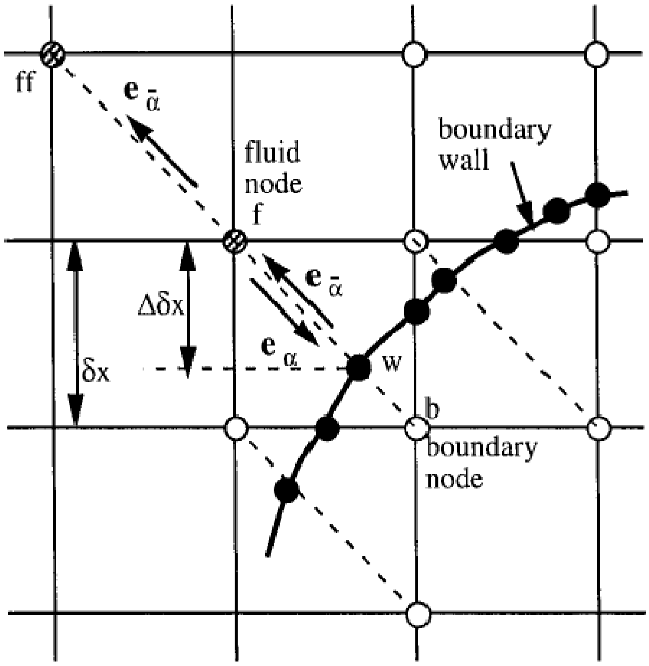


Figure 2.7 A sketch of curved boundary.

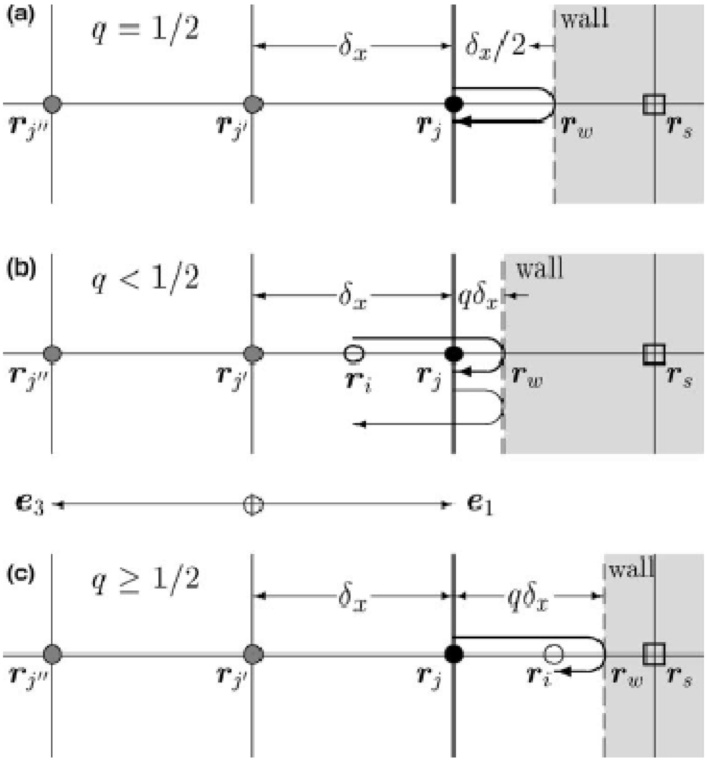


Figure 2.8 Illustration of the Bouzidi's boundary treatment for a rigid wall located arbitrarily between two grid sites in one dimension.

For the sake of simplicity, the 2D boundary in Figure 2.7 can be projected into 1D. This treatment is very intuitive and can be expanded to 3D problem. In Figure 2.8, a wall r_w is placed between node r_j and r_s, the shadow area is solid. The parameter q indicates the faction of fluid part.

$$q = \frac{|r_j - r_w|}{|r_j - r_s|} \tag{2.43}$$

In the tradition bounce back scheme (Figure 2.8 a), distribution function $F_1(x,t)$ at node r_j with lattice velocity c_1, travels from left to right, will hit the wall at $x = r_w$, and bounce back to r_j, which is the value of F_3 at the next time step t^{n+1}. An analogue implement is taken In the Bouzidi's scheme.

The virtual particle will travel a specific distance (Δx) as the same treatment of propagation step. The only difference is that a rigid bounce back will occur when the particle reaches the wall. To avoid extrapolation, which will induce numerical considering the numerical stability, Bouzidi's scheme, illustrated in Figure 2.8, can be classified into 2 situations (Figure 2.8 b and c). If $q < 1/2$ depicted in Figure 2.8b, at time step t^n, The distribution function F_1 at the grid r_j with the lattice velocity pointing to r_s would end up at r_j after back back from r_w. The distance from r_i to r_j is $\left(1-2q\right)\delta x$. Because r_i does not locate at the grid, the value of F_3 at the grid r_j can be interpolated by the points $r_{j'}, r_j, r_i, r_{j''}$. Using linear interpolation, the value of F_3 is

$$F_{i'}(r_j, t+1) = 2q\tilde{F}_i(r_j, t) + (1-2q)\tilde{F}_i(r_{j'}, t) \tag{2.44}$$

where i' is the opposite direction of i, the tilde over F is the after-collision, before-propagation distribution function.

To achieve second order interpolation, the value of F_3 can be written as

$$\begin{gathered} F_{i'}(r_j, t+1) = q(1+q)\tilde{F}_i(r_j, t) + (1-4q^2)\tilde{F}_i(r_{j'}, t) \\ -q(1-2q)\tilde{F}_i(r_{j''}, t) \end{gathered} \tag{2.45}$$

Similarly, the linear and quadratic interpolation of $q > 1/2$ can be written as

$$F_{i'}(r_j, t+1) = \frac{1}{2q}\tilde{F}_i(r_j, t) + \frac{2q-1}{2q}\tilde{F}_i(r_{j'}, t) \tag{2.46}$$

$$\begin{gathered} F_{i'}(r_j, t+1) = \frac{1}{q(1+2q)}\tilde{F}_i(r_j, t) + \frac{2q-1}{q}\tilde{F}_i(r_{j'}, t) \\ -\frac{2q-1}{2q+1}\tilde{F}_i(r_{j''}, t) \end{gathered} \tag{2.47}$$

c. Filippova and Hänel (FH) scheme

On the other hand, Filippova and Hänel proposed another linear interpolation method which take the distribution function at node r_s instead of $r_{j'}$ to the wall boundary condition treatment (Filippova & Hanel 1998).

$$F_{i'}(r_j, t+1) = (1-\chi)\tilde{F}_i(r_j, t) + \chi F_i^*(r_s, t) \tag{2.48}$$

where χ is a interpolation weight, F_j^* is called as friction equilibrium distribution function.

$$F_i^*(r_s, t) = w_i \rho \left[1 + \frac{3}{c^2}(c_i \cdot u_{sf}) + \frac{9}{2c^4}(c_i \cdot u_s)^2 - \frac{3}{2c^2}u_s^2 \right] \tag{2.49}$$

where u_s is the velocity at the solid node r_s, u_{sf} is a ‘correction’ velocity. More details of the derivation can be found in (Mei *et al* 1999).

Filippova and Hänel chose the following values of χ and u_{sf}

when $q < 1/2$:

$$u_{sf} = u_j, \quad \chi = \frac{\omega(2q-1)}{1-\omega} \tag{2.50}$$

and $q \geq 1/2$

$$u_{sf} = \frac{q-1}{q}u_j + \frac{1}{q}u_s, \quad \chi = \omega(2q-1) \tag{2.51}$$

where $\omega = \frac{1}{\tau}$.

To improve the numerical stability, Mei and Luo (Mei *et al* 1999) suggested the following values for χ and u_{sf}.

$$\begin{aligned} q<\frac{1}{2}: \quad & u_{sf}=u_{j'}, \quad \chi=\frac{\omega(2q-1)}{1-2\omega} \\ q\geq\frac{1}{2}: \quad & u_{sf}=\frac{2q-3}{2q}u_j+\frac{3}{2q}u_s, \quad \chi=\frac{2\omega(2q-1)}{2+\omega} \end{aligned} \tag{2.52}$$

Kao and Yang’s recent work showed FH scheme is second order accurate (Kao & Yang 2008).

2.6 Summary

To overcome the intrinsic drawbacks of lattice gas cellular automata, the Maxwell-Boltzmann distribution functions of “virtual” particles and the linear BGK operator have been introduced by scientists in a lattice Boltzmann method (Chen *et al* 1991a; Higuera & Jimenez 1989; Mcnamara & Zanetti 1988; Qian & *et al*. 1992). The macroscopic Navier-Stokes equations can be derived from the microscopic lattice Boltzmann method by applying Chapman-Enskog expansion (Chen & Doolen 1998; He & Luo 1997a), which also requires the isotropy of 2nd and 4th ranks of the lattice tensor. The weights of equilibrium distributions of a specific lattice model can be determined from the conservation of density and momentums (Koelman 1991b). The bounce back boundary condition originated from the LGCA, and was found to be second order accurate if the wall aligns at the middle of the interface of fluid and solid lattice (Pan *et al* 2006). The bounce back scheme was applied to the non-equilibrium part to introduce the pressure and velocity boundary conditions (Zou & He 1997).

3 Flow Simulation Validation

In this chapter, simulations based on the lattice Boltzmann method of flow in two simple geometries are carried out: Poiseuille flow is simulated to investigate the accuracy of the lattice Boltzmann method in two dimensional flows; and flow though an infinite duct with rectangular shape is used to verify the code in three dimensional conditions.

3.1 Poiseuille flow

Let us consider a channel with a width in the y-direction of $2h$. The pressure gradient along the x-direction is $-dp/dx$. The dynamic viscosity of fluid is μ. The mesh of Poiseuille flow is shown in Figure 3.1.

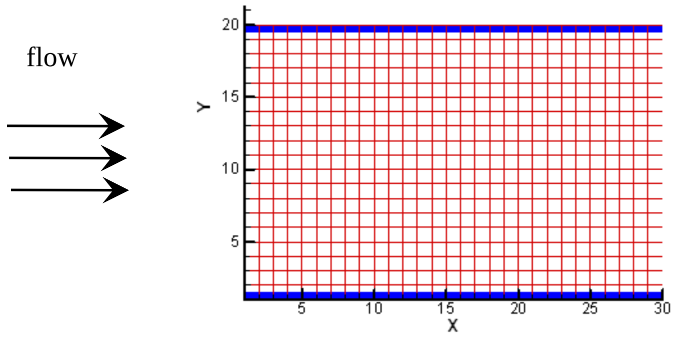


Figure 3.1 the mesh of channel flow. The blue block indicates the solid wall, which is half way from the grid

Assuming that the flow is steady and impressible, we can write the governing equation as

$$\mu \frac{d^2u}{dy^2} = -\frac{dp}{dx} \tag{3.1}$$

where u is the x-component velocity. With the no-slip boundary condition,

$$\begin{array}{ll} y = 0 & u = 0 \\ y = 2h & u = 0 \end{array} \tag{3.2}$$

The theoretical solution for this problem can be easily got by integrating Equation (3.1) as

$$u = -\frac{1}{2\mu}\frac{dp}{dx}(2hy - y^2) \tag{3.3}$$

and for Newtonian fluid, the shear stress is

$$\begin{aligned} \tau_{xy} &= \mu\frac{du}{dy} = \frac{dp}{dx}(h-y) \\ \tau_{xx} &= \mu\frac{du}{dx} = 0 \\ \tau_{yy} &= \mu\frac{dv}{dy} = 0 \end{aligned} \tag{3.4}$$

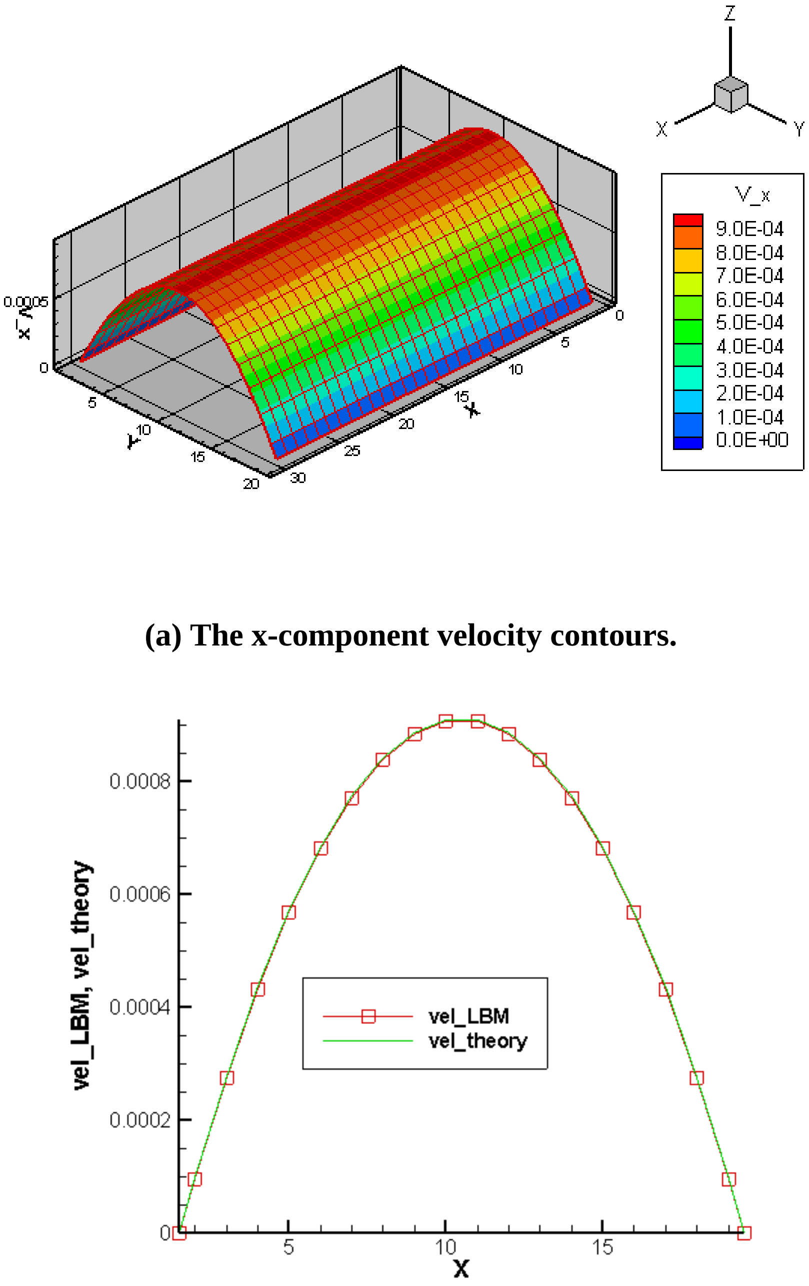
3.1.1 Numerical simulation

In the lattice Boltzmann method, the lattice length is usually considered as the base unit of length, and the physical quantities as length need to be converted to lattice unit prior to the simulation based on the preserve of non-dimensional Reynolds number. If the domain of length L has N lattice units, the space unit can be simply defined as $\delta_x = L/N$. For convenience, the unit lattice length is used in this thesis. To validate the lattice Boltzmann code, in current simulation, the half-height is set as 9 lattices, the length of the channel is 30 lattices, the driven pressure drop is $-1\times10^{-4}N$ per unit lattice, the relaxation parameter ω is set to 1.3, and the density of the “virtual” particles is 0.8 kg per cubic unit lattice. From these parameters, we can work out the Reynolds number as

$$Re = \frac{Uh}{\nu} = -\frac{dp}{dx}\frac{h^3}{4\rho\nu^2} = 2.83 \tag{3.5}$$

where $U = -\frac{dp}{dx}\frac{h^2}{4\mu}$ is the average velocity, which can be easily derived from the theoretical velocity Equation (3.3).

Figure 3.2 (a) is a 3D color graph of velocity in the x direction, and (b) is the x-component velocity profile in a cross-section. The stream line in Figure 3.3 shows that the flow is parallel to the channel wall, which is supported by the near-zero y-direction velocity distribution shown in Figure 3.4. The parabolic shape along the cross-section in Figure 3.2 (b) agrees to the theoretical profile very well and the L^2-norm error is about 1×10^{-3} which will be defined later. However, the y-component velocity, although very small in magnitude, has a parabolic-like distribution as shown in Figure 3.4 indicating that high order of distribution function series (i.e. $F_i^{(2)}, F_i^{(3)}, \cdots$) may induce some minor errors to the velocity results. Nevertheless, the maximum error is approximately 5×10^{-15} unit lattice per second， which is negligibly small.



(a) The x-component velocity contours.

(b) The comparison with the LBM results and the theoretic result in a y-axis cross-section

Figure 3.2 x-velocity of Poiseuille flow in a channel

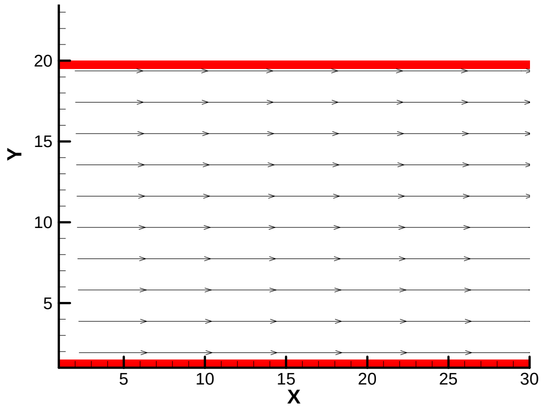


Figure 3.3 the streamline of Poiseuille flow in a channel

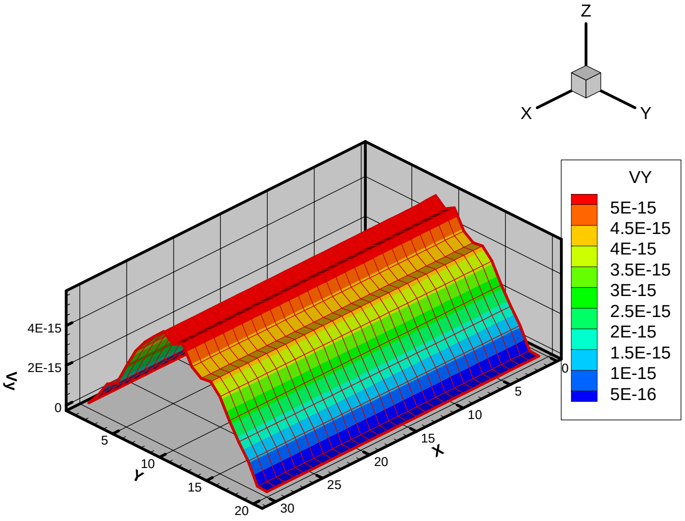


Figure 3.4 The y- velocity contours

3.1.2 Error analysis

In LBM calculation, L^2 -norm error of velocity is introduced to indicate the convergence. The definition of the L^2 -norm error is

$$E_2 = \frac{\left(\sum_{i=1}^{n} (\phi_{LBM} - \phi_{exact})^2 \right)^{1/2}}{\left(\sum_{i=1}^{n} \phi_{exact}^2 \right)^{1/2}} \tag{3.6}$$

where E_2 is the L^2-norm error, ϕ_{LBM} is the result of LBM, which can be velocity or shear stress, ϕ_{exact} is the theoretical velocity and wall shear stress given in Equation (3.3) and (3.4). A summation is taken for all the fluid nodes. Figure 3.5 shows the L^2-norm errors of velocity and shear stress τ_{xy}. We can find that the velocity converges after ~ 15,000 iterations and the shear stress after ~60,000 iterations. The residual of the velocity is approximately 1×10^{-3}, i.e. the difference between the simulation results and theoretic solution is less than 0.1%. Surprisingly, the residual of shear stress in this simulation reaches 10^{-14}, almost the machine precision of the floating point number we used in the program, which is a much precise results compared to the velocity. From Equation 2.27, we know that the calculation of shear stress does not rely on the value of velocity, which allows the shear stress to have highly accurate results. This is indeed one of the advantages of the LBM and is not possible for conventional numerical methods. Besides, the shear stress is linear to y-coordinate, while the velocity profile has a parabolic profile, which will result in additional benefits to the accuracy of shear stress.

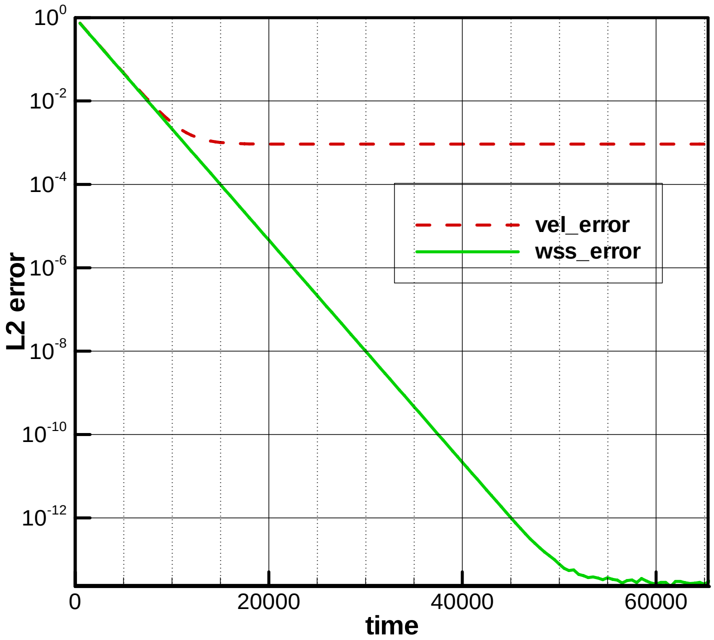


Figure 3.5 The L^2 velocity and shear stress τ_{xy} errors against the calculating time step.

The theoretic shear stress components τ_{xx}, τ_{yy} are zero, so the L^2-norm error will be infinity. However, we can use the standard error as an indicator,

$$E_s = \left(\frac{\sum_{i=1}^{n} (\phi_{LBM})^2}{n} \right)^{1/2} \tag{3.7}$$

The standard errors are shown in Figure 3.6. The error of *x*-directional normal component, τ_{xx}, rises from around 10^{-7} at the beginning of calculation, to 3.8×10^{-6} at 8000 time step and remains steady afterwards. The error of y-directional normal component τ_{yy}, remains close to zero at ~ 10^{-17}.

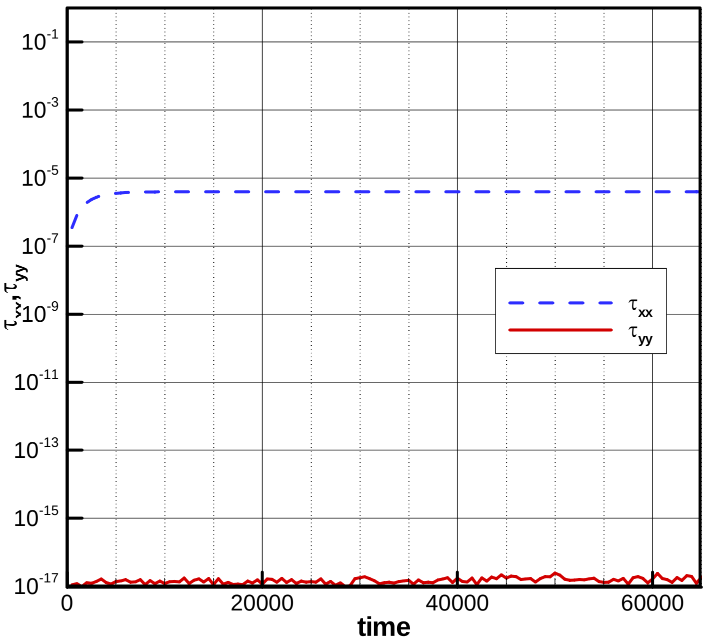


Figure 3.6 The standard errors of shear stresses τ_{xx}, τ_{yy} against the calculating time step.

3.2 Flow in a rectangular duct

3.2.1 Theoretical consideration on duct flow

Flow velocity in a straight, infinite duct with a constant shape is unidirectional and varies only with the y and z axis, as shown in Figure 3.7. The continuity and momentum equations for an incompressible flow are

Continuity: $$\frac{\partial u}{\partial x} = 0 \tag{3.8}$$

Momentum: $$-\frac{dp}{dx} + \mu\left(\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) = 0 \tag{3.9}$$

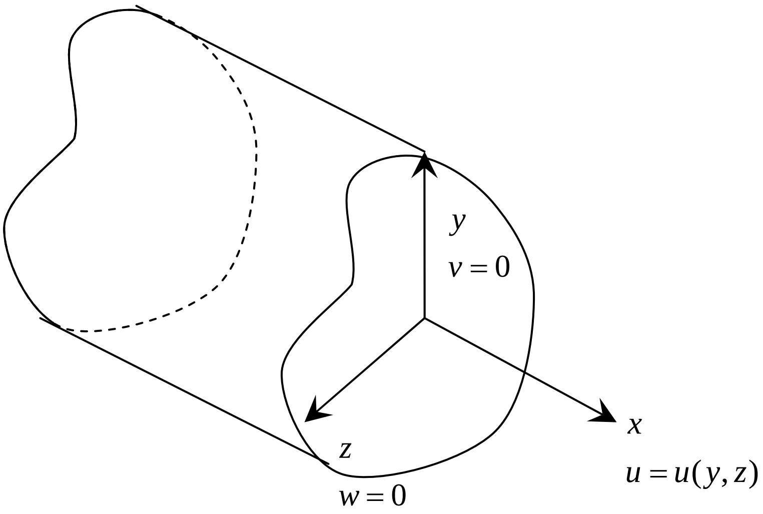


Figure 3.7 The sketch of steady laminar flow in infinite duct with arbitrary but constant cross-section, the velocity in *y* and *z* direction is zero, and *x*-direction velocity is a function of *y* and *z*.

For the fully developed flow in an arbitrary duct, the shear stress should balance with the net pressure difference,

$$\oint_s \tau_w ds = -A\frac{dp}{dx} \tag{3.10}$$

From the definition of mean shear stress, we have

$$\overline{\tau}_w = \frac{1}{S}\oint_s \tau_w ds \tag{3.11}$$

where S is the perimeter of section.

The stress tensor for an incompressible Newtonian viscous fluid in Cartesian coordinate system is (White 1991)

$$\sigma_{ij} = -p\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{3.12}$$

where *p* is the pressure, δ_{ij} is the Kronecker delta as recalled, which is unity if the subscript *i* and *j* are equal, and zero otherwise. μ is the dynamic viscosity, and u_i is the component of velocity. The stress tensor can be written more explicitly in matrix form as

$$\bar{\bar{\sigma}} = \begin{bmatrix} -p + 2\mu \frac{\partial u}{\partial x} & \mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) & \mu\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right) \\ \mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) & -p + 2\mu \frac{\partial v}{\partial y} & \mu\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right) \\ \mu\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right) & \mu\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right) & -p + 2\mu \frac{\partial w}{\partial z} \end{bmatrix} \tag{3.13}$$

where the notation ‘=’ over σ indicates it is a tensor, and (u, v, w) are the velocity components in *x*, *y* and *z* axis, respectively.

For the full developed laminar flow in an infinite duct of arbitrary constant cross-section, the velocity is purely axial and depends only on the y and z direction, i.e. $v = w = 0$ and $u = u(y, z)$, the shear stress tensor is simplified as

$$\bar{\bar{\sigma}} = \begin{bmatrix} -p & \mu \frac{\partial u}{\partial y} & \mu \frac{\partial u}{\partial z} \\ \mu \frac{\partial u}{\partial y} & -p & 0 \\ \mu \frac{\partial u}{\partial z} & 0 & -p \end{bmatrix} \tag{3.14}$$

The normal vector $\vec{n}$ at point (x_0, y_0, z_0) on the cross-section $f(y, z) = 0$ is given by

$$\vec{n}=\frac{\left(0,\quad \left.\frac{\partial f}{\partial y}\right|_{(x_0,y_0,z_0)},\quad \left.\frac{\partial f}{\partial z}\right|_{(x_0,y_0,z_0)}\right)}{\sqrt{\left(\left.\frac{\partial f}{\partial y}\right|_{(x_0,y_0,z_0)}\right)^2+\left(\left.\frac{\partial f}{\partial z}\right|_{(x_0,y_0,z_0)}\right)^2}} \qquad (3.15)$$
$$\equiv\left(0,\quad \cos\theta,\quad \sin\theta\right)$$

where θ is the angle between normal vector and y axis.

From Cauchy's stress theorem (Irgens 2008), The shear stress at a specific point $\left(x_0,y_0,z_0\right)$ is

$$\vec{\sigma}=\overline{\overline{\sigma}}\cdot\vec{n}=\left.\begin{pmatrix}\mu\frac{\partial u}{\partial y}\cos\theta+\mu\frac{\partial u}{\partial z}\sin\theta\\ -p\cos\theta\\ -p\sin\theta\\ \end{pmatrix}\right|_{(x_0,y_0,z_0)} \qquad (3.16)$$

The tangent shear stress, generally called wall shear stress, which is one component of $\vec{\tau}$, can be obtained as

$$\vec{\tau}=\vec{\sigma}-(\vec{n}\cdot\vec{\sigma})\vec{n}=\left.\begin{pmatrix}\mu\frac{\partial u}{\partial y}\cos\theta+\mu\frac{\partial u}{\partial z}\sin\theta\\ 0\\ 0\\ \end{pmatrix}\right|_{(x_0,y_0,z_0)} \qquad (3.17)$$

We have got the theoretic wall shear stress expression for the infinite duct with constant shape. Not surprisingly, the wall shear stress in *y* and *z* direction is zero, and the pressure *p* has no contribution to it.

From the derivation in Chapter 2, we know that the shear stress in lattice Boltzmann method can be expressed by a distribution function as

$$\sigma_{ij}=-p\delta_{ij}-\left(1-\frac{1}{2\tau}\right)\sum_{\alpha=0}^{18}c_{\alpha i}c_{\alpha j}\left(F_\alpha-F_\alpha^{eq}\right) \qquad (3.18)$$

if D3Q19 model is applied.

Now, let's consider a straight, rectangular duct with width $2a$ and height $2b$, as shown in Figure 3.8. The centre of the coordinate system is located at the centre of the channel of the entrance plane. The velocity profile of Stokes flow in the rectangle duct can be achieved by solving Equations (3.8) and (3.9) with boundary conditions

$$u(\pm a, z) = u(y, \pm b) = 0 \tag{3.19}$$

The solution can be expressed as (White 1991)

$$\begin{aligned} u(y,z) = {} & \frac{16a^2}{\mu\pi^3}\left(-\frac{dp}{dx}\right)\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1)^3}\left(1-\frac{\cosh[(2n+1)\pi z/2a]}{\cosh[(2n+1)\pi b/2a]}\right) \\ & \times\cos[(2n+1)\pi y/2a] \end{aligned} \tag{3.20}$$

and the *y*- and *z*- velocities are zero.

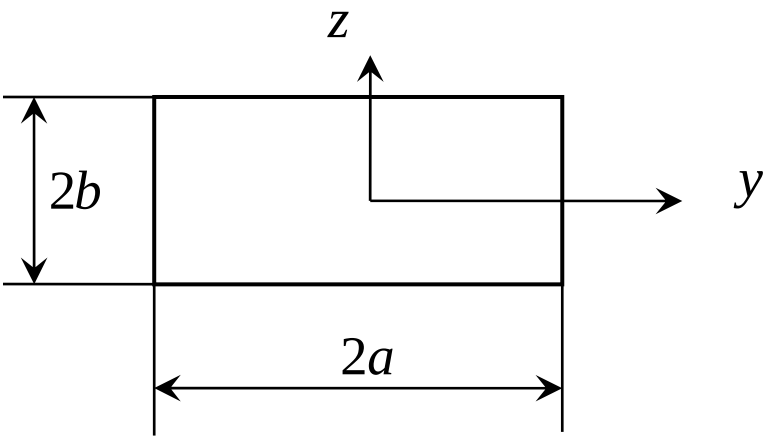


Figure 3.8 The cross-section of a rectangular duct.

From Equation (3.17), $\frac{\partial u}{\partial y}$ and $\frac{\partial u}{\partial z}$ are required to work out the shear stress, which can be manipulated from Equation (3.20) as

$$\frac{\partial u}{\partial y}=\frac{8a}{\mu\pi^2}\left(-\frac{dp}{dx}\right)\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)^2}\left(1-\frac{\cosh[(2n+1)\pi z/2a]}{\cosh[(2n+1)\pi b/2a]}\right) \\ \times\sin[(2n+1)\pi y/2a] \tag{3.21}$$

and

$$\frac{\partial u}{\partial z}=\frac{8a}{\mu\pi^2}\left(-\frac{dp}{dx}\right)\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)^2}\left(\frac{\sinh[(2n+1)\pi z/2a]}{\cosh[(2n+1)\pi b/2a]}\right) \\ \times\cos[(2n+1)\pi y/2a] \tag{3.22}$$

Since the pressure does not contribute to the tangent wall shear stress which we are interested in (see Equation (3.17) for details), we will investigate the stress tensor without the pressure, which can be calculated totally from the distribution function. So these 6 components for the rectangular flow are

$$\begin{aligned} &\tau_{xx}=2\mu\frac{\partial u}{\partial x}=0, && \tau_{yy}=2\mu\frac{\partial v}{\partial y}=0 \\ &\tau_{zz}=2\mu\frac{\partial w}{\partial z}=0, && \tau_{xy}=\mu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)=\mu\frac{\partial u}{\partial y} \\ &\tau_{yz}=\mu\left(\frac{\partial w}{\partial y}+\frac{\partial v}{\partial z}\right)=0 && \tau_{zx}=\mu\left(\frac{\partial w}{\partial x}+\frac{\partial u}{\partial z}\right)=\mu\frac{\partial u}{\partial z} \end{aligned} \tag{3.23}$$

3.2.2 Numerical simulation by the lattice Boltzmann method

In the lattice Boltzmann simulation, the width and height of the fluid domain are set equally at 30 lattice units, and the length of the duct at 5 lattice units. The pressure drop per unit lattice is $-1\times10^{-5}N$, the relaxation parameter ω is set at 1.2 and the density of the "virtual" particles is 1 kg per cubic unit lattice. From these parameters, we can work out the kinetic viscosity of the experimental fluid is

$$\nu = \frac{1}{3}(\frac{1}{\omega} - \frac{1}{2}) = \frac{1}{9} \tag{3.24}$$

and the Reynolds number is

$$Re = \frac{Uh}{\nu} = 1.61 \tag{3.25}$$

Bounce back condition is implemented at the wall boundary. A constant pressure difference dp/dx condition is applied in the x direction. For sake of simplicity, the velocity is set as zero and the density is set to be uniform for the whole flow region in the initial condition treatment. The calculation will converge quicker for a better guess of velocity (Skordos 1993). As shown in Figure 3.9, we can see that after 3400 time steps of simulation, convergence is achieved with the largest L^2-norm error is less than 1×10^{-6}. The overall calculation takes about 30 seconds in an Intel Pentium D CPU 3.4GHz with 3GB memory computer. The error of x-velocity drops from 10^{-3} to less than 1×10^{-6}. The errors of y-velocity and z-velocity, follow the same trend and drop at the same rate due to the identical geometry setting.

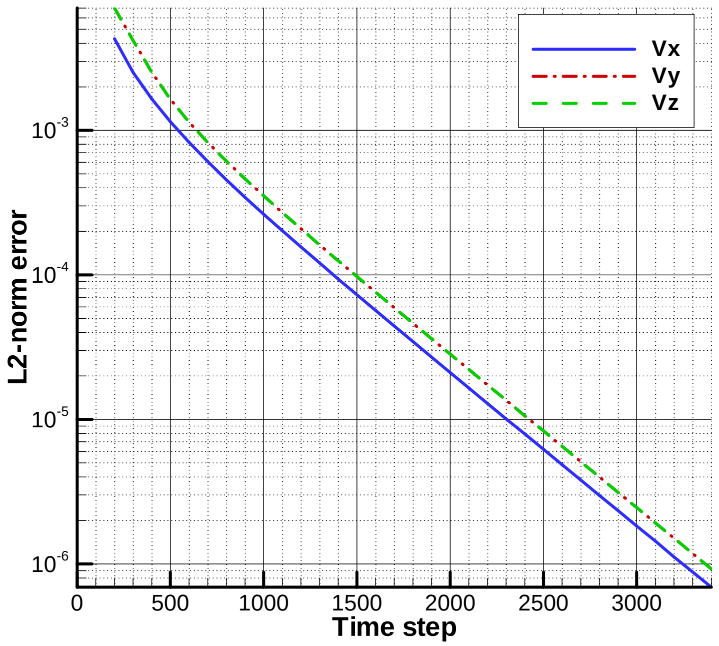


Figure 3.9 The L^2-norm error of x-, y- and z-direction of velocity.

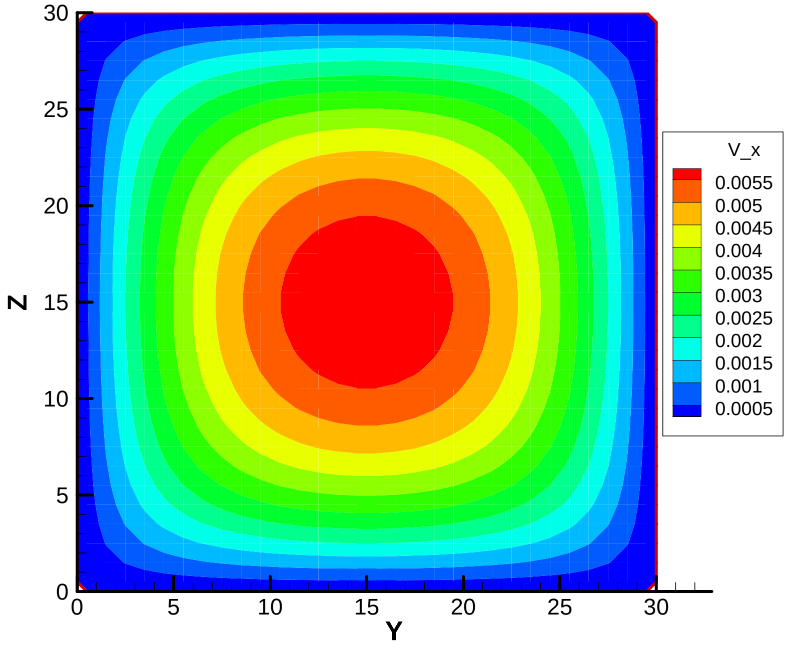


Figure 3.10 The *x*-velocity contours in the cross-section *x*=3

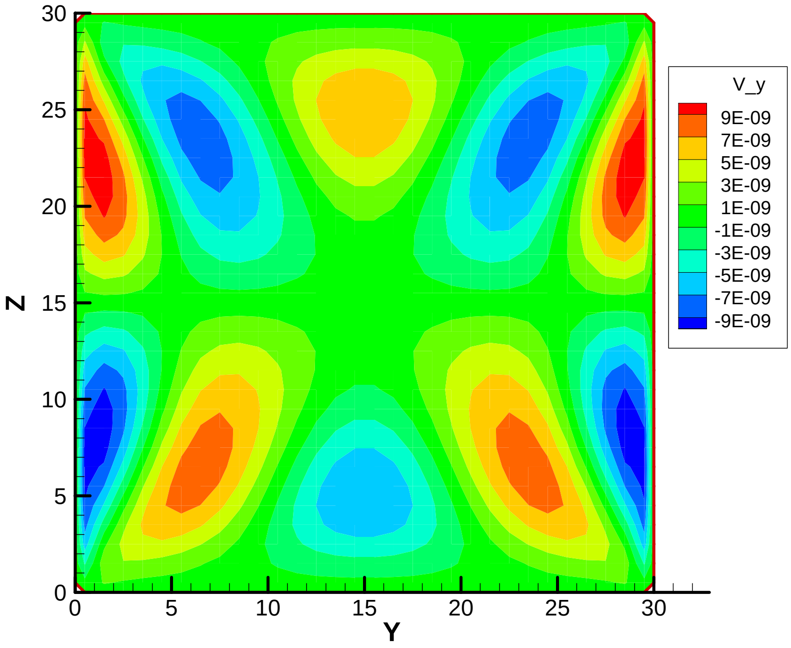


Figure 3.11 The *y*-velocity contours in a cross-section *x*=3

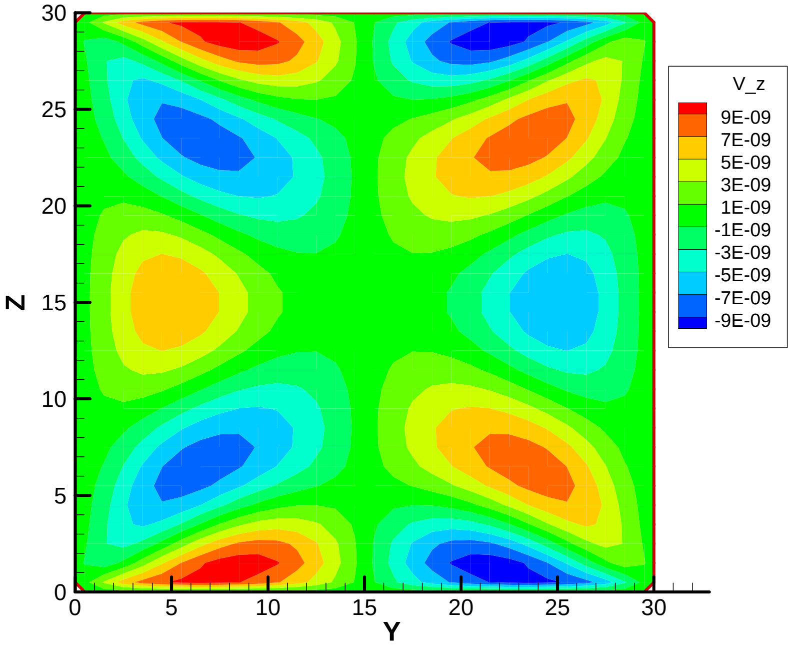


Figure 3.12 The *z*-velocity contours in a cross-section *x*=3

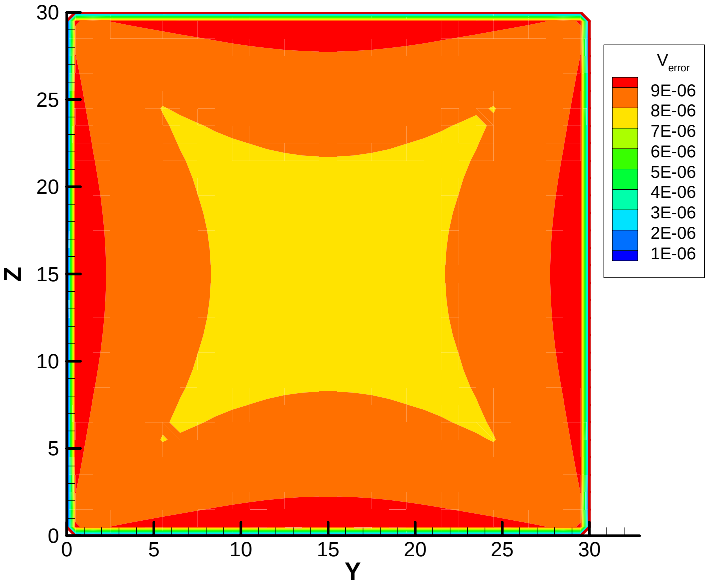
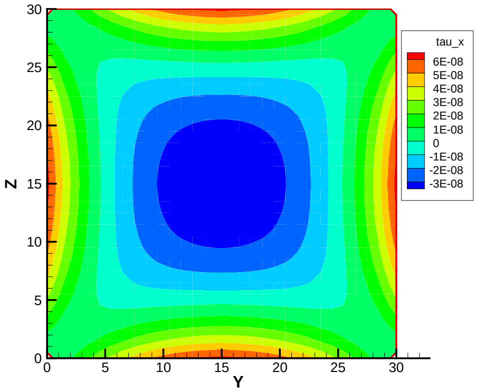
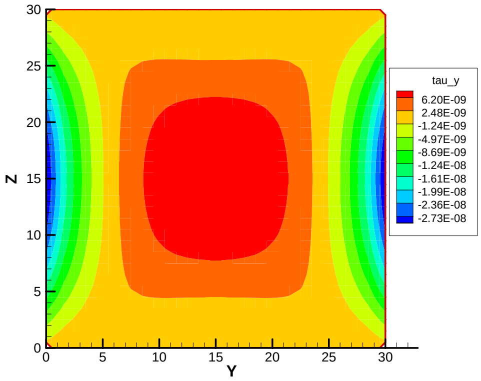


Figure 3.13 The contours of *x*-velocity error between lattice Boltzmann results and analytic solution

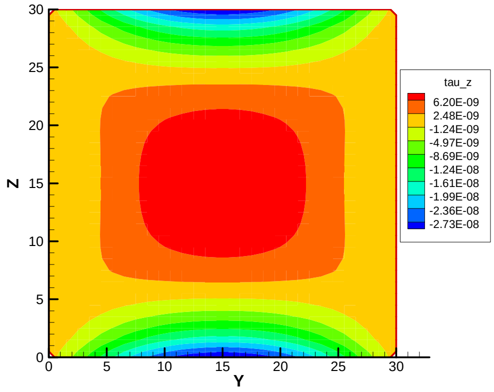
Figure 3.10 ~ Figure 3.12 are the contours of the *x*-, *y*- & *z*- velocities. The *x*-velocity near the center is alike to the cylindrical pipe flow, where the boundary effect is weak. The *y*- & *z*- velocities have got anti-symmetric distributions along the midline *z* = 15 and *y* = 15 respectively, while the analytic solution is zero at the whole region. The overall standard errors for y-velocity and z-velocity are both 4.1×10^{-9}, which is negligibly small. The contours of x-velocity error $\left(V_x^{theory} - V_x^{LBM}\right)$ are depicted in Figure 3.13, where the maximum error occurs near the middle of the boundary walls at 9×10^{-6} unit lattice per second, which is approximately 1% of the analytic results there. In the center, the difference between the LBM result and analytic one is $\sim 7.8\times10^{-6}$ unit lattice per second, which represents a relative error of

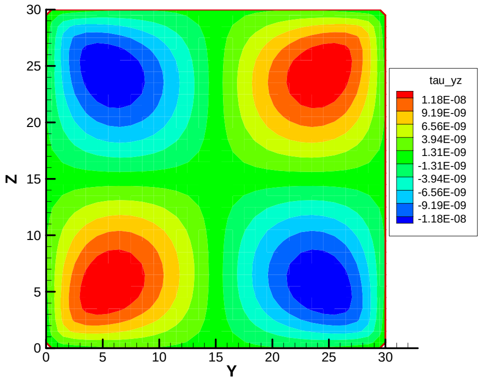
~ 0.13% . The L^2-norm error for the x-velocity is 2.5×10^{-3}, which demonstrates exceptionally satisfactory accuracy of the lattice Boltzmann method.





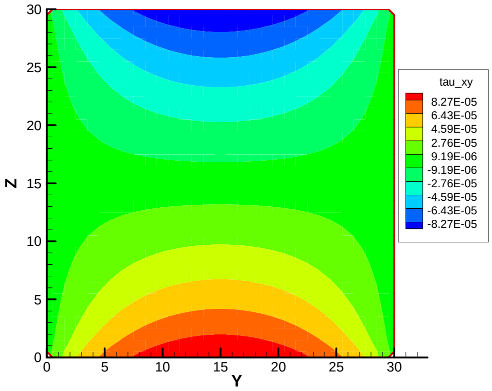
(a) The shear stress component τ_{xx} **(b) The shear stress component** τ_{yy}

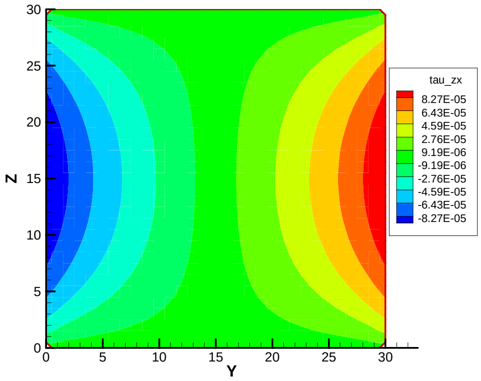




(c) The shear stress component τ_{zz} **(d) The shear stress component**

τ_{yz}





(e) The shear stress component τ_{xy} **(f) The shear stress component**

τ_{zx}

Figure 3.14 Contours of the 6 components of the shear stress in the cross-section *x* = 3.

Figure 3.14 (a) – (f) present contours of the 6 components of the shear stress. From Equation (3.23), we know 4 of them should be zero, and the residual of which is of the order 10^{-8} in the lattice Boltzmann results. The contour pattern of τ_{xx} is very similar to the *x*-velocity, and the maximum value of is 6×10^{-8}, with the standard error of 6.65×10^{-8}. The contours of τ_{yy} and τ_{zz} are the same if they change the coordinates, with the same value of 6.2×10^{-9} at the center, 1/10 of the maximum of τ_{xx}. The standard errors are both 2.18×10^{-8}. τ_{yz} is anti-symmetric for the midlines *y* = 15 and *z* = 15, and the maximum of which is 1.18×10^{-8}, with the standard error of 1.70×10^{-8}.

The non-zero components τ_{xy} and τ_{zx} are very similar, so only τ_{xy} is used to compare with the analytic solution in Equation (3.21). As depicted in Figure 3.15, the lattice Boltzmann results and analytic ones have very good agreement in the majority part of the rectangular colored by green, where the difference is smaller than 5×10^{-8}. At the four corners, the error raises to 2×10^{-6} mainly caused by the linear interpolation there.

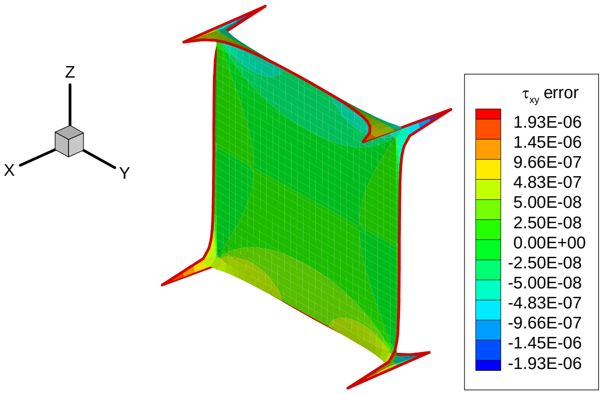


Figure 3.15 The contours of differences of τ_{xy} between lattice Boltzmann results and analytic solution based on Equation (3.21)

3.2.3 Effects of the lattice density

We have performed a mesh-independent study to check the accuracy of lattice Boltzmann method. In all cases, the aspect ratio $\kappa = b/a$ is 1, the length of channel is 5 lattice units with periodic boundary condition applied. The pressure gradient per unit lattice is -1×10^{-5}. A number of different mesh densities (length×width×height) are examined in the computational domain, i.e $5\times5\times5$, $5\times10\times10$, $5\times15\times15$, $5\times20\times20$, $5\times30\times30$, $5\times40\times40$ and $5\times50\times50$ respectively. By compared to the analytic velocity profiles, we can find the relative errors with different lattice densities. As shown in Figure 3.16, the error drops from 3.7% at 5 lattice units to 0.14% at 30 lattice units, and remains almost the same as the unit number increases further. Even at 10 lattice units, the error is less than 1%, which can be considered as satisfactory in most calculations.

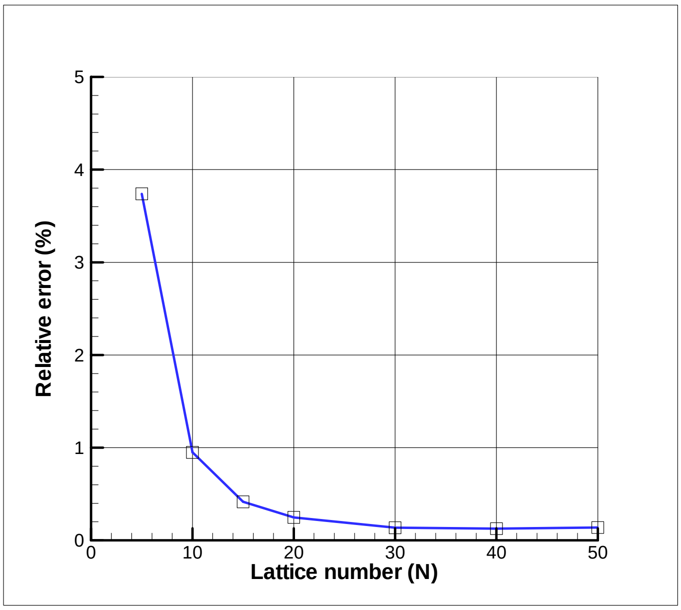


Figure 3.16 The relative errors with different number of lattice densities in simulations.

3.3 Summary

In this chapter, we have conducted a 2D Poiseuille flow between two infinite planar planes and in a 3D rectangular duct to verify our lattice Boltzmann program. For the 2D Poiseuille flow, the length of channel is set as 30, and the width at 20. Driven by a steady pressure gradient, the parabolic velocity profile simulated matches the analytic results very well, and the L^2 norm of the x-velocity is less than 1×10^{-3}. The shear stress τ_{xy}, on the other hand, reaches 1×10^{-14} after ~60,000 iterations, which is many orders of magnitude smaller than the error in the velocity, but takes 4-fold increase in time to reach convergence. Standard errors are used to study the error in cases with zero analytic results. The standard errors of τ_{xx} and τ_{yy} are 3.8×10^{-6} and 10^{-17} respectively. This simulation not only validates our LBM program, but also provides an example that the lattice Boltzmann method makes possible much higher accuracy in shear stress calculation.

A cubiod with 30 lattice units in both height and width is used for the 3D flow simulation. To speed up the calculation, only 5 lattice units are used along the flow direction with periodic boundary condition to mimic the infinite duct flow. The x-velocity by lattice Boltzmann method shows very good agreement to the analytical results, with a relative error of 0.13% at the center. The error of shear stress τ_{xy} in most part is less than 5×10^{-8}, and the error raises at the four corners due to the linear interpolation. The accuracy of current shear stress calculation can be improved further by applying high order interpolation. The standard errors of y- and z-velocities are 4.1×10^{-9}, with the same order of the zero shear stress components: τ_{xx}, τ_{yy}, τ_{zz} and τ_{yz}.

A mesh-independent study has been performed to check the accuracy of lattice Boltzmann method. Even with only 10 lattice units, the error is as low as below 1%. The results can be considered to be very good in most engineering numerical calculations. As the lattice density increases, the relative error decreases further, but no apparent improvement can be seen when the lattice unit number is above 30.

Again, for flow in a duct, we have also seen that the lattice Boltzmann method provides accurate results. The fact that highly accurate shear stress can be calculated independently of the velocity makes it a highly desirable method to study flow in porous scaffolds.

4 2D simulation of flow in porous scaffolds using the lattice Boltzmann method

4.1 Flow around an array of solid square cylinders

We start with creeping flow through an array of square cylinders in this chapter. As shown in Figure 4.1, the square cylinders align in a rectangular with 3 squares in a column and 4 squares in a row. The dimension of each square is 8×8 lattice units. The computation domain is the rectangle with a length of 64 lattice units, and a width of 48 lattice units. Periodic condition both in the *x* and *y* direction are applied. The flow around these square cylinders is driven by a pressure gradient in the *x*-direction, with $dp/dx=-10^{-5}N$. The Reynolds number is around 1.

Based on the average velocity (flow rate at the outlet divided by the width) as shown in Figure 4.2, the simulation is found to converge in less than 5000 time steps, which is much quicker than that in the simulation of Poiseuille flow in Chapter 3. One possible reason for this is that the amount of convergent time step is dependent on the difference between distribution function F_i and the equilibrium distribution function F_i^{eq} (i.e. $F_i-F_i^{eq}$).

The closer the two values, the faster the calculation will converge. Low velocity used in the simulation using the lattice Boltzmann method is clearly beneficial for flow simulation in porous media.

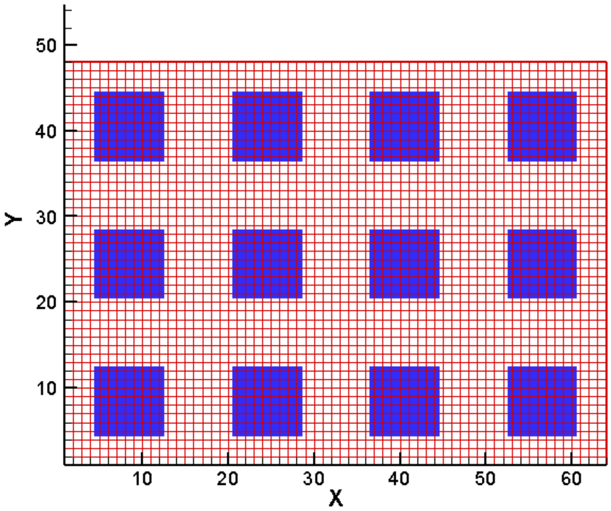


Figure 4.1 A sketch of the flow domain with an array of square cylinders.

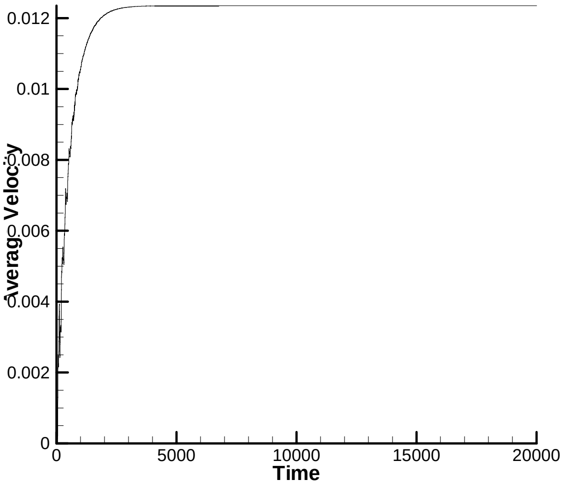


Figure 4.2 The average velocity against time steps to show the convergence of LBM calculation.

The streamlines in Figure 4.3 show that the flow is mainly through the channels between two rows of square blocks. The vector contour in Figure 4.4 implies the effects of blocks to form a sub-channel Poiseuille flow between the two neighbour rows. From the vectors along the outlet, we can see that the velocity behind the block is very low, and the velocity profile in the sub-channel is parabolic-like. This hypothesis can be supported by the velocity distribution at the cross-section $x = 24$ in Figure 4.6. When the pressure gradient changes, as shown in Figure 4.5, the average velocity changes linearly forming a straight line passing through the origin point. This agrees to results predicted by the Darcy’s law. In addition, from the slope we can get the permeability of this porous structure. To verify the LBM program, Fluent is used to simulate flow in the same structure under identical conditions, except that more dense meshes (double density) are used in Fluent to ensure accurate result. Figure 4.6 shows the velocity comparison between results by Fluent and LBM at two cross-sections, $x = 16$, which lies in the middle of the fluid region between two column blocks, and $x = 24$, which lies in the middle of a column squares. Good agreement is seen in the results.

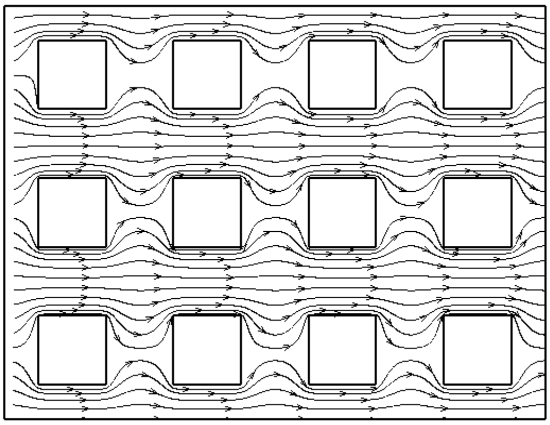


Figure 4.3 The streamline of flow around an array of square cylinders in LBM simulation.

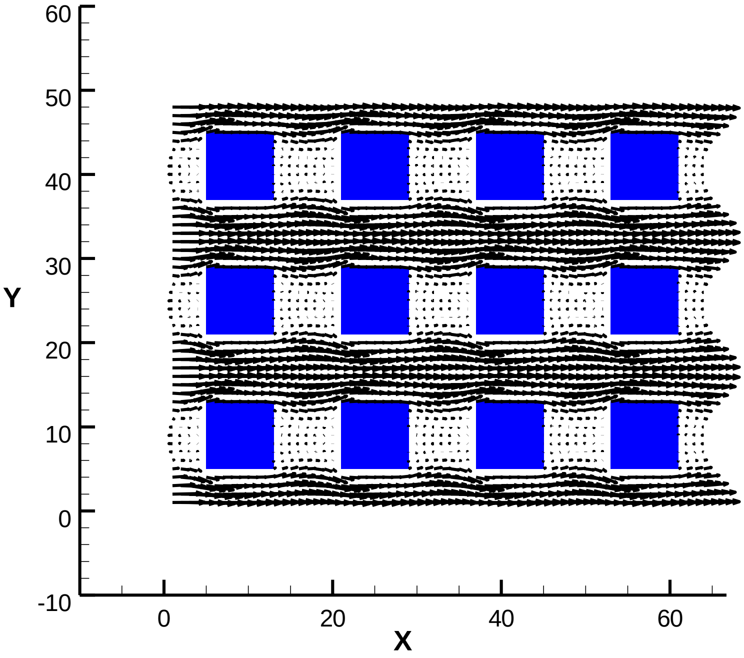


Figure 4.4 The velocity vectors of flow around an array of square cylinders in LBM simulation.

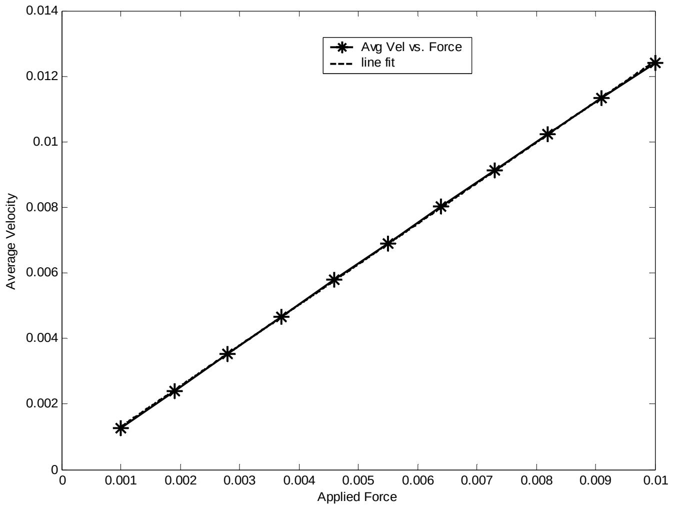


Figure 4.5 The average velocity against the pressure gradient.

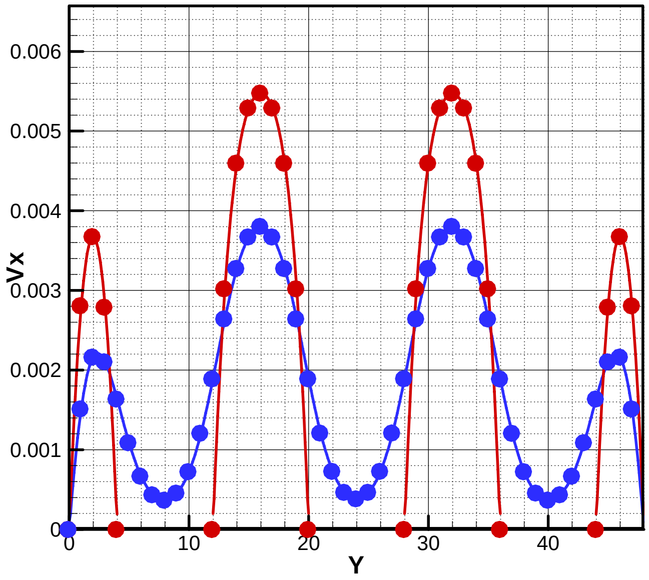


Figure 4.6 Comparison in the velocity calculated using Fluent (curves) and LBM (solid circles) at the *x* = 16 (blue) and *x* = 24 (red) cross-sections respectively. .

4.2 Flow simulation in a more realistic porous structure

Micro-CT uses x-ray to get the structural information at fine cross-sections of a 3D-object, with the pixel size of the image at micrometer resolution. In this project, the porous scaffold used in cell engineering is a porous disk with a diameter of 30mm. A slice image of the disc taken by the micro-CT is shown in Figure 4.7. The black color is void and the white is for the solid frame. The size of this image is 650×650 pixels. For the sake of simplicity, we assume 1 pixel represents 0.05mm (since the diameter of the disc is 30mm and the image covers slightly larger area than the disc). Since the real geometry is a disk, voids exist at four corners. A rectangular part of the image is chosen, outlined in red box in Figure 4.7 and given in Figure 4.8. The resolution of the rectangular region is 552×422 pixels, represented the physical dimensions of $27.6\,mm \times 21.1\,mm$.

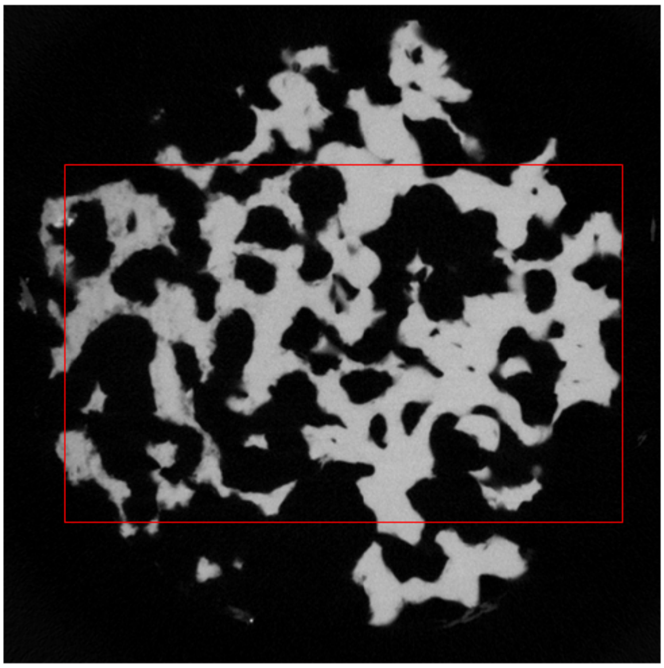


Figure 4.7 A 2D micro-CT image of the porous scaffold used in cell culture in the laboratory. The size of this image is 650×650 pixels; the black colour is void and the white is solid.

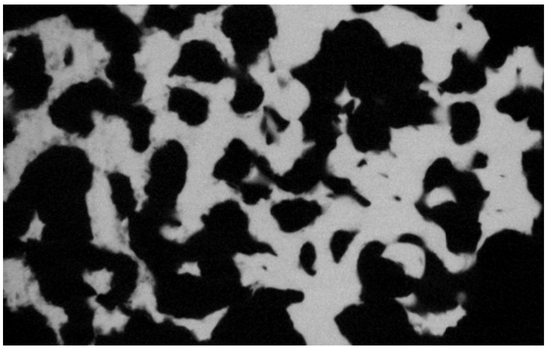


Figure 4.8 The rectangular section of the image in Figure 4.7 for the simulation.

It is clear that the image above is part of a *y*-*z* cross-section of the scaffold, whereas the flow is in the *x*-axial direction. Connectivity in a 3D scaffold can be significantly reduced in a 2D slice. In order to simulate flow in a complex 2D structure that is based on a 2D image of a real scaffold material, the image above is edited to create more connections. The structure is then put in a channel, as shown in Figure 4.9, where flow from the left to the right is simulated using the lattice Boltzmann method. The whole flow region is meshed by 1000×422 lattice units, too dense to clarify a single grid. A 22×22 part has been magnified to show the details of the mesh.

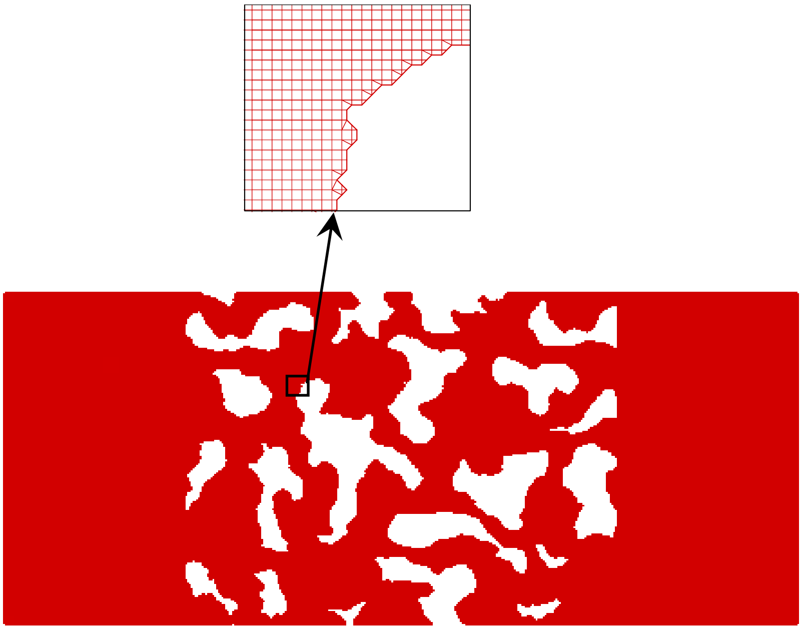


Figure 4.9 The mesh for flow in a more realistic 2D porous scaffold. The resolution for the calculation is 1000×422 lattice units. A closer look of a 22×22 section is provided as an insert to show details of the lattice mesh.

In the simulation, Zou & He velocity inlet boundary condition is applied at the entrance, and the inlet velocity is set to $0.001\, m/s$. The pressure outlet boundary condition is applied at the right side of the channel exit (Zou & He 1997). Following the Poiseuille flow, the L^2-norm error is used to indicate the convergence of the calculation. Slightly different from our previous study, we calculate the L^2-norm every successive 500 time steps (see Equation (4.1) for comparison), in order to consider the effect of complex geometry and the slow convergence of shear stress.

$$E_2 = \frac{\left(\sum_{i=1}^{n}(\phi^{t+500} - \phi^{t})^2\right)^{1/2}}{\left(\sum_{i=1}^{n}\left(\phi^{t+500}\right)^2\right)^{1/2}} \tag{4.1}$$

where ϕ represents the value of ρ, u_x or u_y . ϕ^{t+500}, ϕ^{t} are the values at time step t+500, or t, respectively. Though the geometry is rather complex, and the computation domain is more than 700 times larger than that for the Poiseuille flow, The L^2-norm residual is less than 10^{-6} after 15000 iterations, the same as the Poiseuille flow we discussed in Chapter 3. After 36000 iterations, The L^2-norm of y-velocity is below 10^{-7} and treated to be convergent.

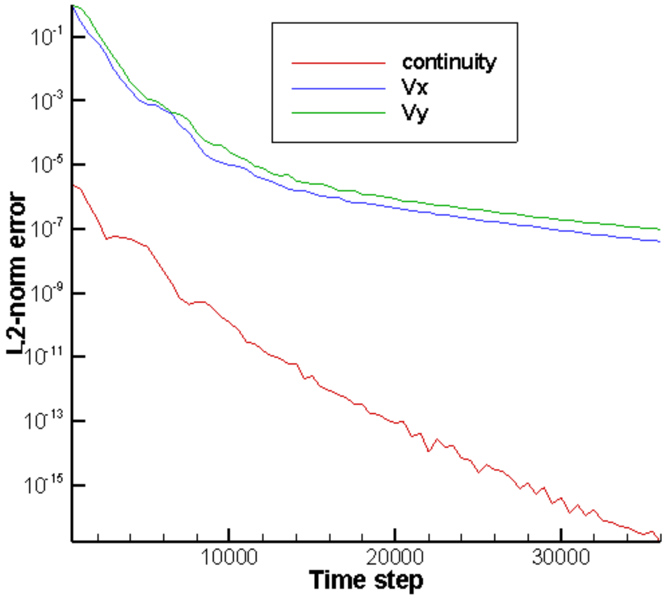
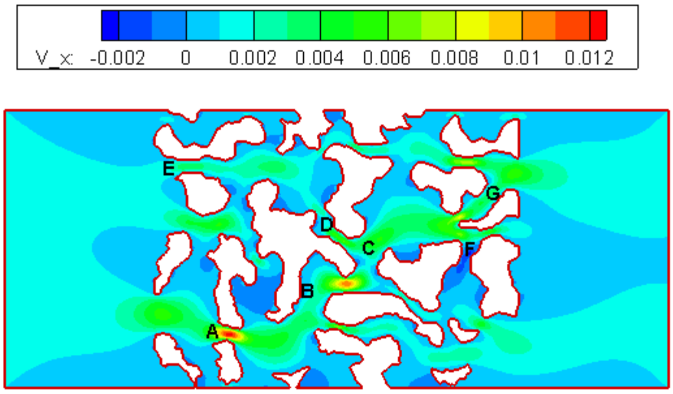
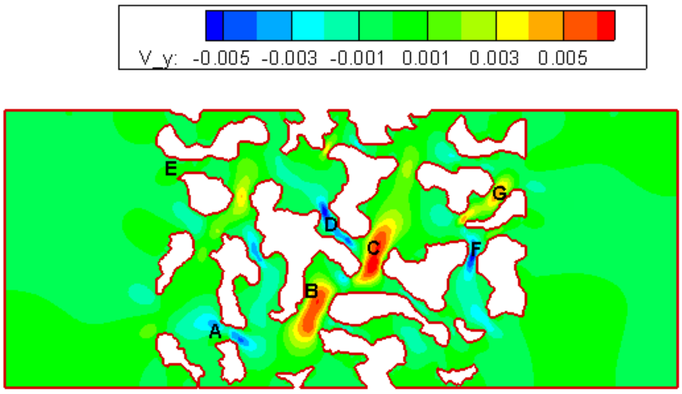


Figure 4.10 The L^2-norm of mass and velocity in the calculation

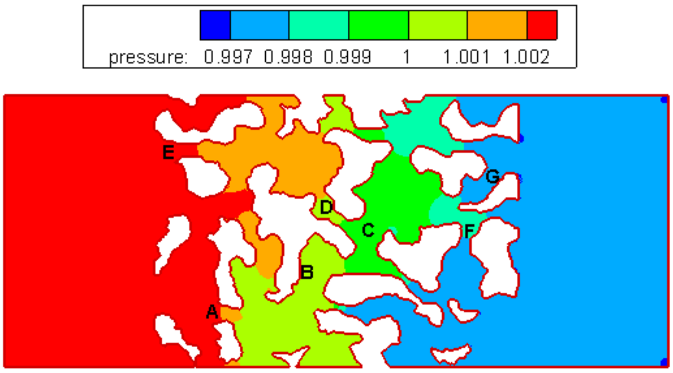
Figure 4.11 (a) and (b) are the x- and y- velocity contours for the flow respectively. It can be seen that the flow forms patterns to pass through easy connections with low energy cost (i.e. pressure drop). Along that flow path, the narrower the channel, the higher the velocity value is. There exist two major channels (A-B-C-G and E-D-C-G) in the porous structure, the maximum value of the x- velocity lies at point A. For the y-velocity, after pass through the narrow channel A, the fluid moves upwards to B & then to C, this induces an increase in the y- velocity in channels B and C. In the mean time, negative y-velocity occurs in channels D and F as the fluid flows downwards. Figure 4.11 (c) gives the pressure contour in the flow domain. Due to the creeping nature of the flow, pressure change is very small in the flow domain. Pressure drops occur mainly in the narrow channels where the resistance to flow is high.



(a) *x*-axis velocity contours



(b) *y*-axis velocity contours



(c) The contours of pressure

Figure 4.11 2D flow and pressure simulation by LBM in a more realistic porous scaffold.

Streamlines are presented in Figure 4.12, which give details on the exact flow path from any given point in the domain. The density of streamlines, in addition, can be used as an indicator to the velocity magnitude. It can be seen clearly that the connection A-B-C-G is the main channel for fluid movement. The shear stress contour is presented in Figure 4.13. As mentioned earlier, the shear stress calculation in LBM does not depend on the velocity and has the same 2nd order accuracy. Regions with large value of shear stress exist in the narrow areas along the main connection channels, including points A - G as indicated in the figure.

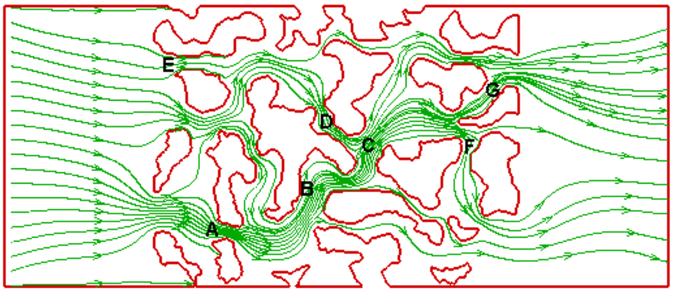


Figure 4.12 Streamline ditribution in the porous scaffold by LBM

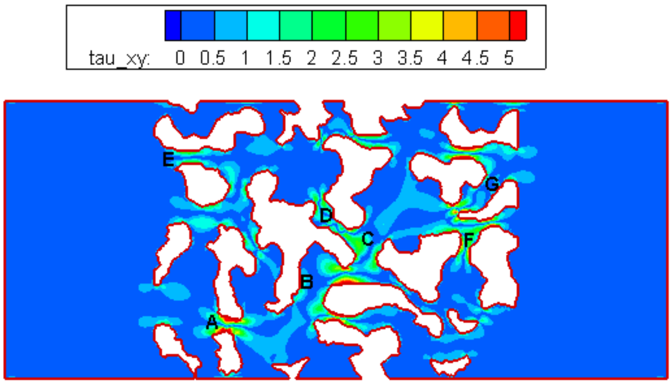


Figure 4.13 Contours of the shear stress component τ_{xy} distribution in the porous scaffold by LBM

To examine the accuracy of the lattice Boltzmann method for flow in the more complex porous material, we have also simulated the flow in above geometry using Fluent. 2 cross-sections, denoted by light brown and dark brown planes respectively in Figure 4.14, have been picked up to compare the results.

Results by the lattice Boltzmann method and by Fluent at y = 10mm (i.e. the central line along the x-axis) is given in Figure 4.15. The x- velocity and y-velocity are compared in Figure 4.15 (a) and (b), and show very good agreement. The x-velocity is 0.001 m/s at x = 0 due to the velocity boundary condition at the entrance; at the exit, the value is great than 0.001 m/s, indicating that more flow occurs near the centre of the scaffold than near the wall boundary. 2 peak values of approximately 0.003m/s are seen at x = 19 mm and x = 26 mm, where the narrow channels meet. The negative velocity at *x* = 35mm is caused by the local channel structure near point F. The y-velocity is zero at the entrance and the exit due to the fixed boundary conditions. The velocity profiles indicate satisfactory agreement between results by LBM and Fluent. For velocity profile in the *x* = 25 mm section, shown in Figure 4.16 (a) & (b), similar conclusion can be made.

For the shear stress distribution, as shown in Figure 4.15 (c) and Figure 4.16 (c), there are good agreements in results by LBM and Fluent, but the magnitude of peak values of the shear stress is smaller as calculated by the lattice Boltzmann method than that by Fluent. We are confident that our LBM program is capable of calculate the shear stress in a complex porous structure at second order accuracy and more importantly, doing so independently of the velocity.

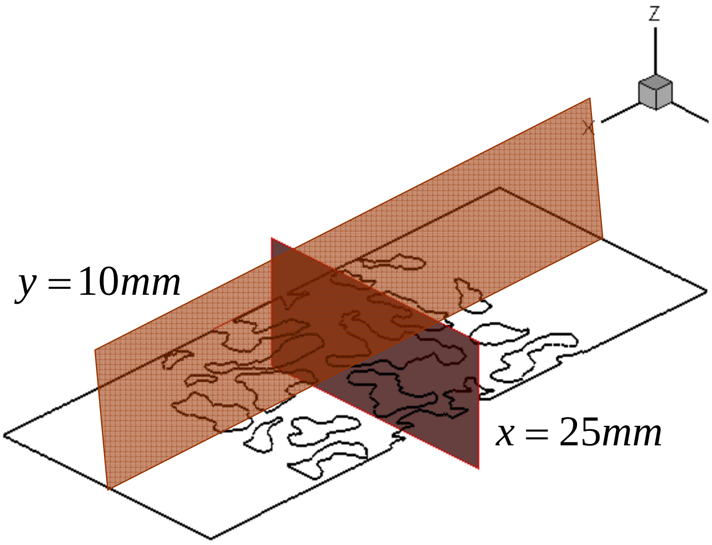
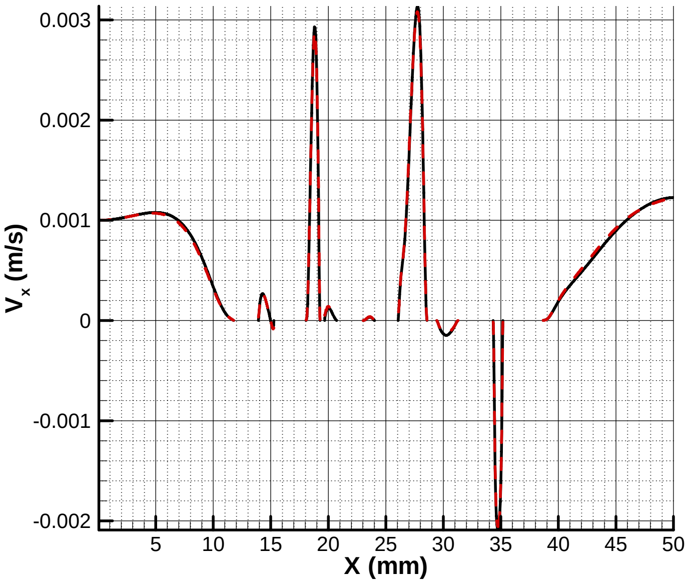
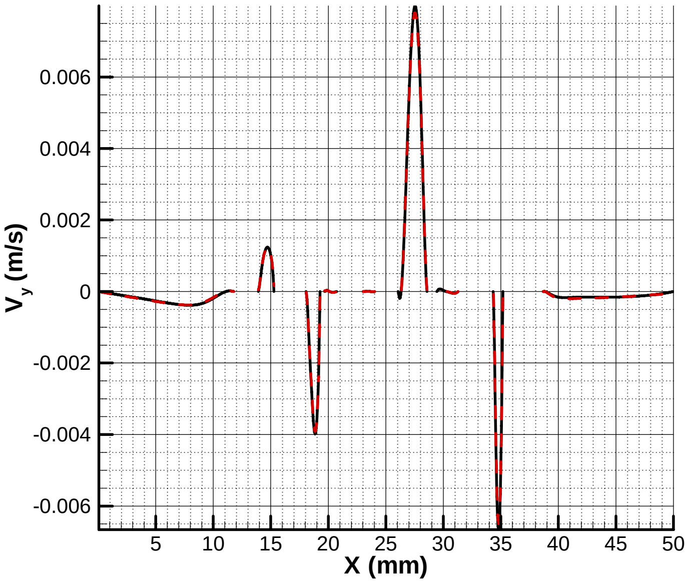


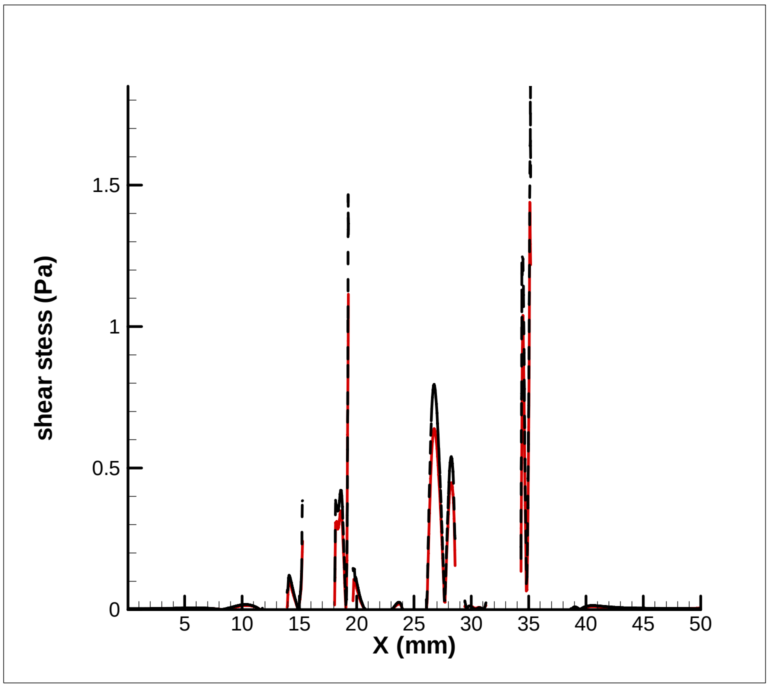
Figure 4.14 The cross-sections used to compare results by LBM and by Fluent results (light brown: *y*=10mm; dark brown: *x*=25mm.)



(a) The *x*-component velocity

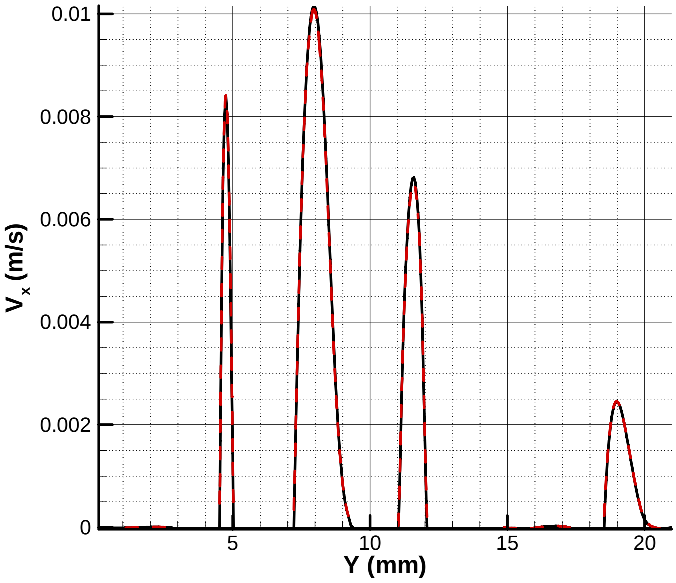


(b) The *y*-component velocity

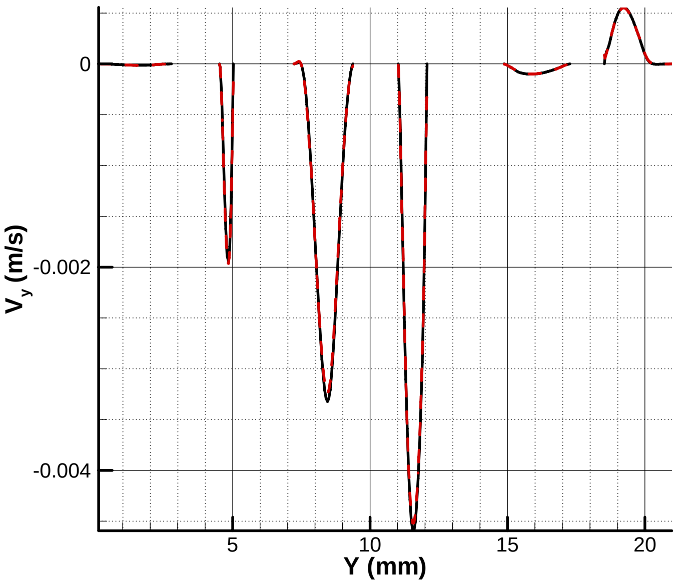


(c) The magnitude of shear stress

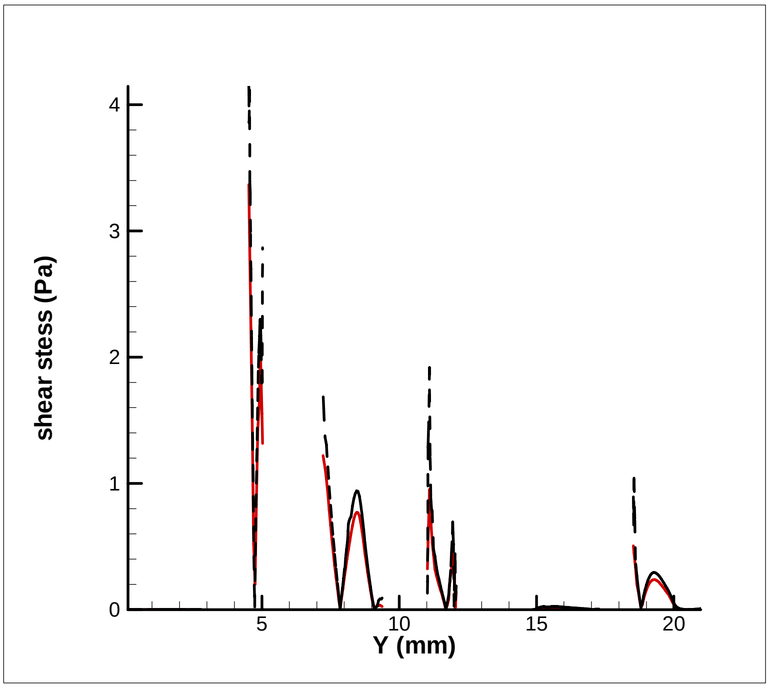
Figure 4.15 Velocity and shear stress profiles at *y* = 10 mm section, i.e. the central line along the *x*-axis. The red solid line is result by LBM and the black dashed line is that by Fluent.



(a) The *x*-component velocity



(b) The *y*-component velocity



(c) The magnitude of shear stress

Figure 4.16 Velocity and shear stress profiles at *x* = 25 mm section, i.e. the central line along *y*-axis. The red solid line is result by LBM and the black dashed line is that by Fluent.

4.3 Summary

In this chapter, two 2D flow simulations have been studied using the lattice Boltzmann method: the creeping flow around an array of square cylinder and a porous scaffold originated from a micro-CT image. It takes less than 5000 iterations to reach the convergence for the first geometry. For the latter geometry, the maximum residual for mass and velocities is less than 10^{-6} after 15000 iterations, and is as low as 10^{-7} after 36000 iterations. This represents a similar iteration steps as in Poiseuille flow discussed in Chapter 3, although the geometry is much more complex and the computation domain is approximately 700 times larger. These results show that the convergence of the lattice Boltzmann method is irrelevant to the complexity of geometry and the mesh size, which make it a powerful tool to simulate of flow in complex geometries, such as a porous scaffold.

For the creeping flow around an array of square cylinders, velocity profiles by Fluent and the lattice Boltzmann method show very good agreement. For the complex porous geometry, originated from micro-CT images, the lattice Boltzmann simulation provides very similar results in velocity to those by Fluent. The magnitude of the peak shear stress by the lattice Boltzmann method is smaller than that by Fluent. The error may be caused by, in our view, the calculation of velocity gradients in Fluent, or the interpolation process in the lattice Boltzmann method. The underestimate of shear stress by LBM was also reported by using a finite difference formula to calculate the wall shear stress from the velocity (Porter *et al* 2005).

Results from this chapter set the foundation to mimic the proliferation of cells seeded in the porous scaffold, when a relation between cell growth and local shear stress is known (or assumed), as well as for simulation of flow in 3D porous scaffolds in the following two chapters.

5 2D simulation of cell proliferation in scaffolds

5.1 Introduction

The process of cell proliferation, called cell cycle, has four major phases (Smith & Martin 1973). G_1 phase is marked by synthesis of various enzymes that are required for DNA replication. The cell then enters the S phase, where DNA replication occurs. The G_2 phase is the gap between DNA replication and division, the cell continue to grow, and significant proteins are produced, which are required during the process of mitosis. The fourth phase is M phase, where a cell separates the duplicated chromosomes into two identical daughter cells.

There are a small number of preliminary studies on the relationship between the fluid simulation and cell proliferation using coarse porous samples (Bancroft *et al* 2002; Cartmell *et al* 2003). Experimental results show that with the increase of shear stress, cell proliferation rate will be enhanced at low values, but after a certain critical value, cell proliferation will be inhibited.. Based on these findings, a simplified hypothetical Gaussian relationship between the cell proliferation rate and the shear stress is proposed in the current study. Details on the model will be discussed in the next section.

Cell growth alters the geometry of the scaffold. This effect is particularly significant in regions where the dimension of the cell is compatible to the width of the micro-channel in the porous material. Therefore, it creates another dimension of complexity in numerical simulation. In conventional methods, the altered geometry of the solid boundary will require re-meshing of the flow domain, which can be computing intensive. The lattice Boltzmann method, on the other hand, generates the mesh of the solid structure and the fluid domain together. As cell grows and the solid boundary evolves, one only needs to change the label of node from fluid to solid accordingly in regions with new cells. Therefore the lattice Boltzmann method is intrinsically much more efficient in solving the coupled problem.

In this chapter, a pilot study on the interaction between flow and cell proliferation is carried out based on a shear stress dependent cell proliferation model. The lattice Boltzmann method is used to calculate the flow field and shear stress distribution following every stage of cell division, which itself is dependent on, among other factors, the local shear stress. It needs to be emphasized that the purpose of the pilot study is to demonstrate the capacity of our method using LBM in studying flow-structure interaction with an active boundary, rather than to confirm or to mimic cell proliferation process in scaffolds. The Gaussian function of cell proliferation rate with the shear stress, although supported qualitatively by available experimental observations, can only be regarded as a crude assumption.

5.2 Cell proliferation model

In our model, cell proliferation is affected by 3 factors: the cell cycle time, contact inhibition and wall shear stress factor.

The cell cycle time is assumed as a constant T_c, which means the new born cell will grow and divide into two cells after a period of T_c. The typical cell cycle time is around 12~24 hours, which may vary between different cells and under different stimuli (Olariu *et al* 2007). At each time step, the state of all the cells will increase by $\Delta t / T_c$ as

$$p(t+\Delta t) = p(t) + \frac{\Delta t}{T} \tag{5.1}$$

where $p(t)$ is the cell state function, where $p(t)=0$ denotes a new born cell, and the cell will divide into two if $p(t) \geq 1$, Δt is the time interval. As an example to illustrate Equation (5.1)., let's consider a cell with a cell cycle $T_c = 24h$ and the time interval $\Delta t = 4h$. At its birth, t = 0 and the cell state function $p(0)=0$. Assuming there is no other factors affecting the cell growth, the cell state function increases $1/6$ at each step and the cell will divide into 2 cells after 6 time steps, when

$$p(6\Delta t) = 6 \times \frac{\Delta t}{T_c} = 1 \tag{5.2}$$

When neighboring cells touch each other and there is no room for cell division, the cell proliferation stops. This phenomenon, known as 'contact inhibition', has been observed both on flat plates (Folkman & Moscona 1978;

Takahashi & Suzuki 1996) and in 3D porous scaffolds (Risbud *et al* 2002; Shigematsu *et al* 1999). This effect can be built easily into our model by a conditional check,

$$p(t+\Delta t)=\begin{cases} 0, & if\ \ contact\ inhibition\ occurs \\ p(t)+\Delta p, & if\ \ not \end{cases} \tag{5.3}$$

where Δp is the change of cell state by the cell cycle or wall shear stress.

Most cells sense and respond to mechanical stimuli, including pressure and shear stresses. In the bone, for example, cells react to mechanical stimulation induced by the interstitial fluid movement (Fritton & Weinbaum 2009). The cell proliferation will be enhanced by the increase of the flow induced shear stress compared to the static control sample (Bancroft *et al* 2002). However, further increases in the shear stress will inhibit the proliferation and cause the cell to be detached and washed away from the substance (Alvarez-Barreto *et al* 2007; Cartmell *et al* 2003). The recommend shear stress has been suggested to be 0.8-3 Pa in the lacunar-canalicular system (Maes *et al* 2009; Stolberg & McCloskey 2009). To model this effect, which we call it ‘shear stress dependent cell proliferation rate’, we assumed a Gaussian function as shown in Figure 5.1.

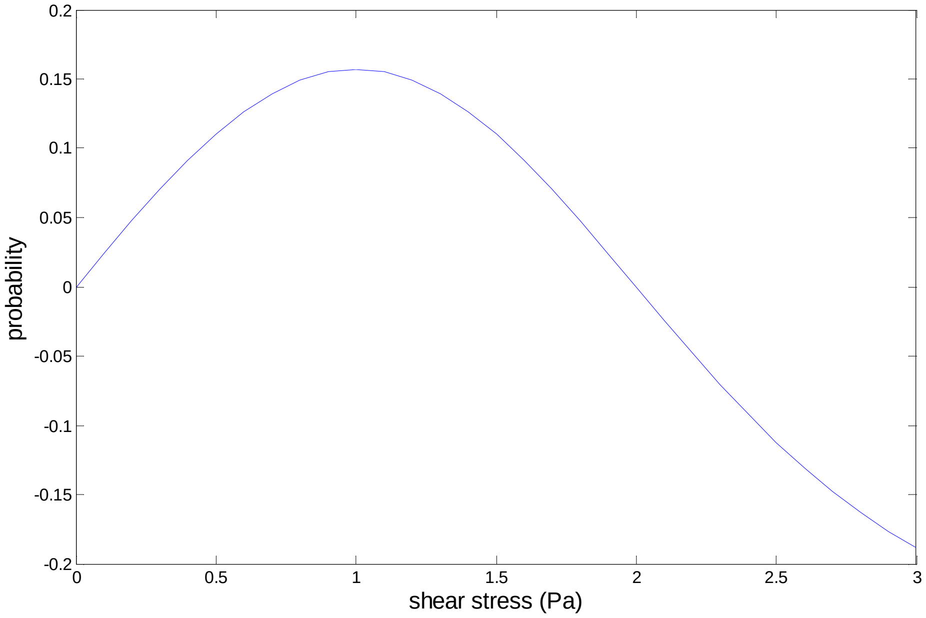


Figure 5.1 The Gaussian function of cell proliferation with fluid shear stress.

In the above model, at zero shear stress, there is no shear-enhanced effect. At low magnitude of the shear stress, e.g. 0.0 < τ < 2.0 Pa, cell proliferation is sped up when cells are subjected to a shear stress, with the maximum effect at τ = 1.0 Pa. Further increase in shear stress from 2 Pa results in cell proliferation being inhibited,

$$\theta = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(\tau-\tau_0)^2}{2\sigma^2}\right) - \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{\tau_0{}^2}{2\sigma^2}\right) \qquad (5.4)$$

where θ is the value of ‘shear stress dependant cell growth rate’, σ is the standard deviation, τ_0 is the expected value, τ is the local shear stress. It can be easily found that when the shear stress equal to τ_0, the θ reaches its maximum value.

In our model, N cells are randomly cultured in the porous scaffold at $t=0$ with randomly given values of $p_i, i=1,\cdots,N$. Each cell will occupy a rectangular space with a lattice unit δ_x, as depicted in Figure 5.2.

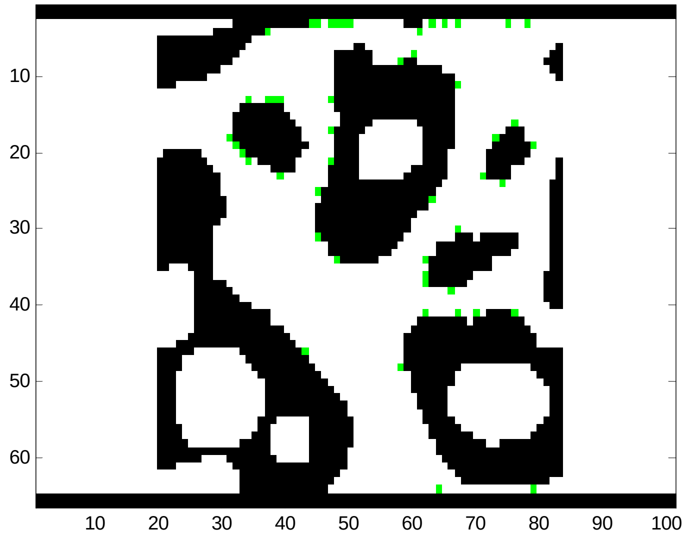


Figure 5.2 An illustration of cells randomly seeded in a porous structure.

At each time step $t=t^n$, shear stress will be calculated by the lattice Boltzmann method as the fluid domain may have changed. Then, the influence of shear stress will be calculated by Equation (5.4) and added to cell state function, a parameter that records the percentage of the cell cycle that has elapsed since last cell division. The overall change of cell state function together with cell cycle effect can be written as

$$p_i(t^n) = p_i(t^n - \Delta t) + \theta_i + \frac{\Delta t}{T_c}, \quad i=1,\cdots,N \tag{5.5}$$

where $t^n - \Delta t$ denotes the previous time step.

Before the implement of cell division, the condition check will be applied for the contact inhibition effect. If the cell state function of a cell reaches 1 or above, the cell divides and the state function of the two daughter cells is set to 0. One of the daughter cells will stay in the original lattice, and the other one will occupy one of the available lattices, randomly, of the 8 neighbouring lattices.

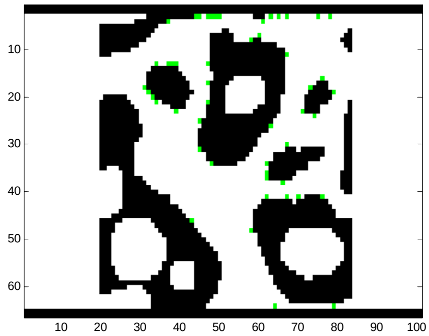
$$\begin{cases} p_i(t^n) = 0 & \text{cell stays in the old position} \\ p_{N+1}(t^n) = 0 & \text{the new born cell in the neighbor} \end{cases} \tag{5.6}$$

where the subscript N+1 denotes the increase of cell amount.

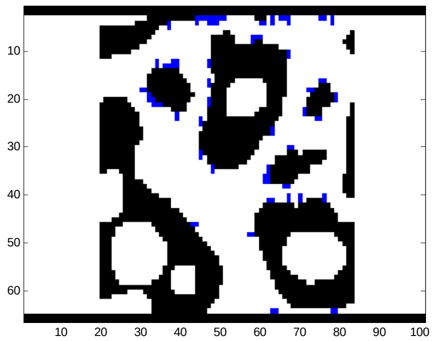
5.3 Results and discussion

Two different cases have been investigated, as examples, in this study. In the first case, the effect of wall shear stress dominates the cell proliferation, which can be seen as a simplified model in which the supply of the nutrition is very limited, so that only regions with high velocity/shear stress have enough supply for cell growth and division.

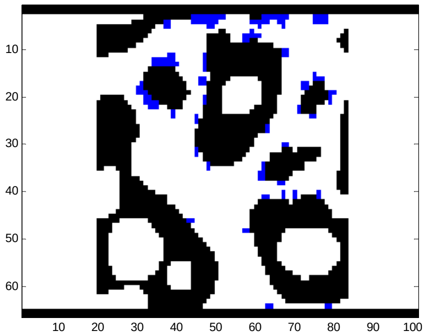
In the initial step, 50 cells are randomly cultured in a porous scaffold. We record the development of cells in the scaffold and present results when the total cell number increases to 100, 150 …and to 300 in Figure 5.3. It can be seen in the Figure 5.2 that the small gap near the bottom wall is very narrow and becomes blocked by the new cells. As a result, cell growth becomes very slow there. In contrast, the channel near the top wall is the main stream for fluid movement with relatively high shear stresses. Cell proliferation is significantly enhanced, e.g. Figure 5.2 (a) – (d), until the channel is blocked totally in Figure 5.2 (e). Following that, as shown in Figure 5.2 (f), the channel in the middle becomes the main stream for fluid movement, and cell proliferation is significantly enhanced there.



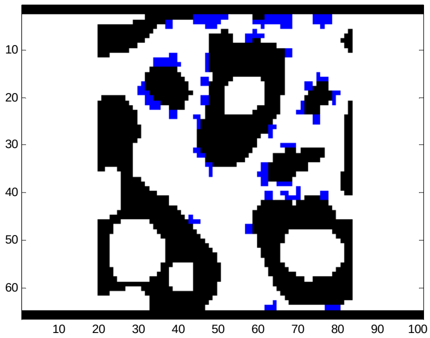
(a) Cell number *N*=50



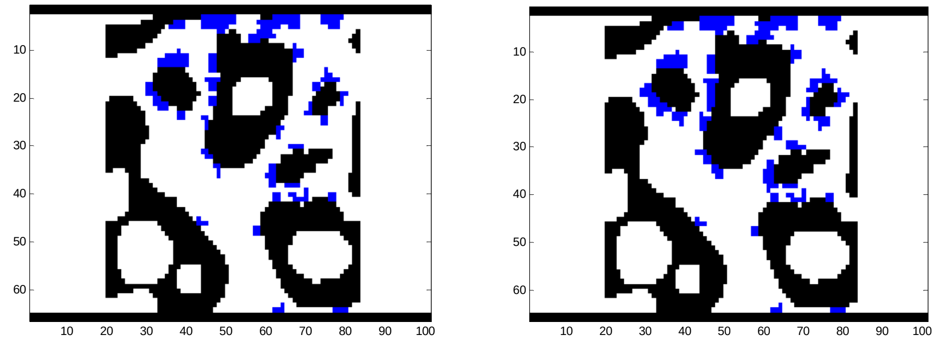
(b) Cell number *N*=100



(c) Cell number *N*=150



(d) Cell number *N*=200

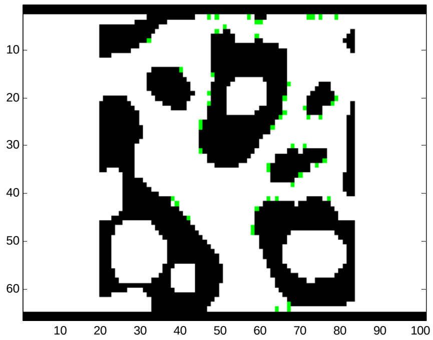


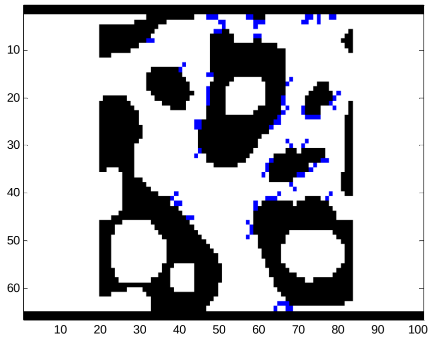
(e) Cell number *N*=250 **(f) Cell number *N*=300**

Figure 5.3 Cell proliferation in a porous scaffold, where the effect of wall shear stress is predominant. White is the fluid, black is the solid, and blue is the cell, green in (a) is the original 50 mother cells

In the second case, we assume that the effect of wall shear stress on cell proliferation rate is negligibly small. In the initial step, 50 cells are randomly seeded into the same porous scaffold as in the earlier case. We show the state of cell distribution following every cell cycle in Figure 5.4. At the end of the first cell cycle, as shown in Figure 5.4 (b), the number of cells is doubled. The same doubling process happens again at the end of the second cell cycle, as shown in Figure 5.4 (c). By now the cell density in certain regions becomes dense. At the end of the third cell cycle, only 167 new cells have the extra room to grow, and 33 cells are inhibited from dividing into two daughter cells due to the ‘contact inhibition’ condition. During the fourth cell cycle time, 229 cells divide, which is only 62.4% of the total number of mature cells. As cell cycle carried on, there are only 37.8% cells divide during the fifth cell cycle.

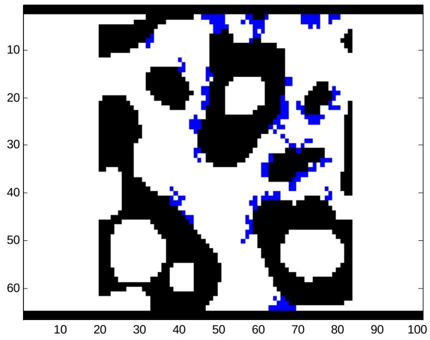
Cell growth occupies available spaces in a few cell cycles and the cell proliferation rate decreases very quickly as more and more cells are prohibited from division.

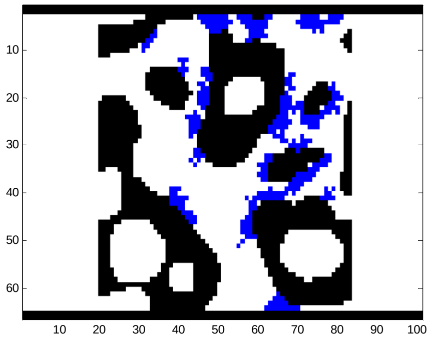




(a) initial state, cell number **(b) by the end of the 1st cell cycle**

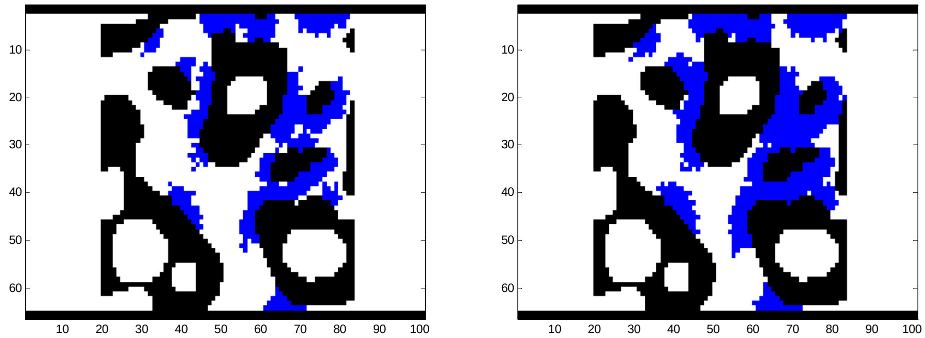
***N*=50** ***N*=100**





(c) by the end of the 2nd cell cycle **(d) by the end of the 3rd cell cycle**

***N*=200** ***N*=367**



(e) by the end of the 4th cell cycle

***N*=596**

(f) by the end of the 5th cell cycle

***N*=821**

Figure 5.4 Cell proliferation in a porous scaffold at different time points. In the figure, effects of wall shear stress on cell growth are assumed to be negligible.

6 Flow in a 3D porous scaffold material

Most cells sense and respond to mechanical stimuli, including pressure and shear stresses. In the bone, for example, cells react to mechanical stimulation induced by the interstitial fluid movement (Fritton & Weinbaum 2009). The flow induced shear stress has been suggested to be 0.8-3 Pa in the lacunar-canalicular system (Maes *et al* 2009; Stolberg & McCloskey 2009), while the growth of tissue cells was inhibited by the shear stress higher than 2.6 Pa (Alvarez-Barreto *et al* 2007).

However, it is not a simple task to get the detail information on shear stress in a complex porous structure. Theoretical and experimental methods have been applied to estimate shear stress inside 3D porous scaffolds (Cioffi *et al* 2006; Wang & Tarbell 2000). By assuming that flow through the scaffolds with an idealized pore structure of varying tortuosity obeyed Darcy's law, Botchwey *et alet al* estimated shear stresses within their microcarrier scaffolds with high aspect ratio rotation (Botchwey *et al* 2003). While this approach provides an order of magnitude estimate of the average shear stress, the distribution of shear stresses within complex 3D porous materials is yet to be determined. The lattice Boltzmann method may provide a useful tool in addressing this challenge. In this chapter, we will apply the lattice Boltzmann method to investigate flow in a 3D porous scaffold.

6.1 Imaging process

Micro-CT has been successfully used to reconstruct the micro-structure of porous scaffolds (Cartmell *et al* 2004; Kuhn *et al* 1990; Unser *et al* 1995), which is an efficient tool to quantitatively measure the distribution of micro-structure. There are several general protocols to reconstruct 3D objects, mostly based on interpolation schemes (Tinku & Ping-Sing; Unser *et al* 1995). The resolution of images is 650×650 pixels for a diameter of 30 mm porous disk. Recall that for the sake of simplicity, we assume 1 pixel represents 0.05mm (since the diameter of the disc is 30mm and the image covers slightly larger area than the disc). Figure 6.1 is a grayscale micro-CT slice image of the porous disk. The dark color indicates the pore and the light color is the solid phase. The 3D porous disk shown in Figure 6.2 is reconstructed using a Matlab program developed inhouse which is based on the Parallel-beam CT reconstruction algorithm (Hu 1999). A close look of the scaffold can be found in Figure 6.3, where a small volume of the structure in the middle of the scaffold ($128\times128\times128$ pixels, i.e. $6.4\,mm\times 6.4\,mm\times 6.4\,mm$) is shown following surface smoothing.

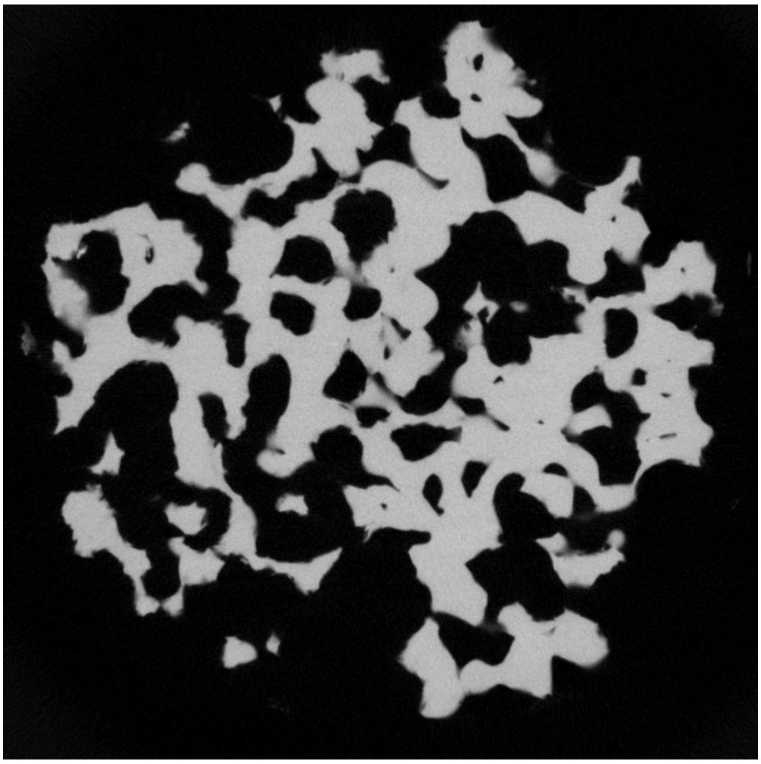


Figure 6.1 A 2D micro-CT slice of the porous disk.

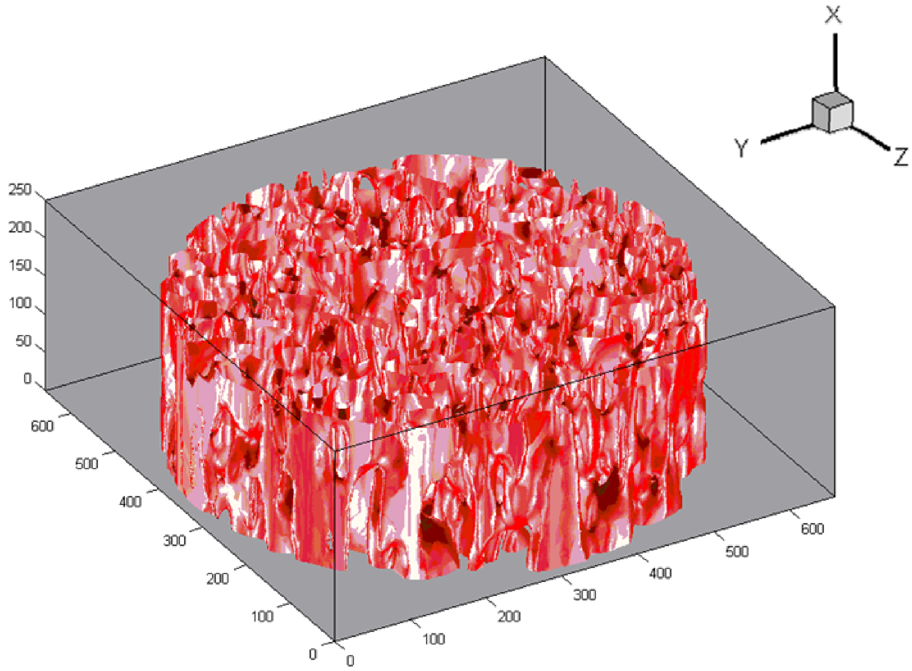


Figure 6.2 Reconstructed porous scaffold disk.

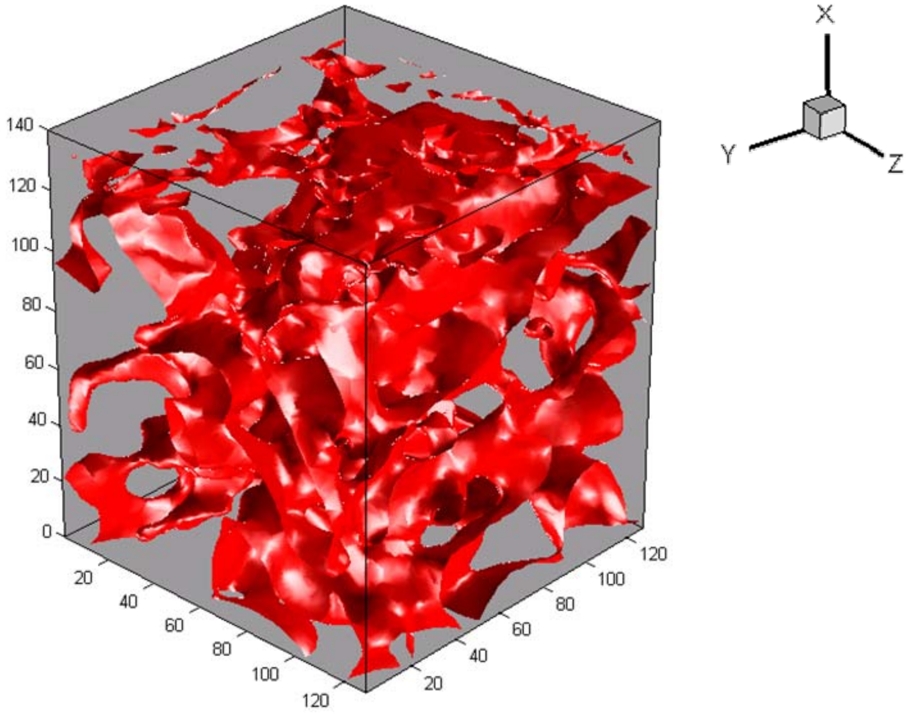


Figure 6.3 A representative volume of the structure in the middle of the 3D scaffold with $128 \times 128 \times 128$ **pixels, or** $6.4\,mm \times 6.4\,mm \times 6.4\,mm$ **in physic units.**

6.2 3D results based on LBM simulation

In the last section, we have constructed the 3D porous scaffold. However, it is still beyond the capability of computers in most research groups to mimic flow through the whole porous disk (Maes *et al* 2009; Porter *et al* 2005). In our current study, a small volume in the middle of the scaffold is chosen to carry out flow simulation to demonstrate that our method based on LBM works in 3D structures. A $64\times64\times64$ lattice units, which represents a volume of $3.2mm\times3.2mm\times3.2mm$ has been selected in the center of the porous disk in order to avoid the end effect in flow simulation. The driven pressure drop is $-1\times10^{-5}\,Pa$, the relaxation parameter ω is set at 1.2, and the density of the "virtual" particles is 1 kg per cubic unit lattice. From these parameters, we can work out the Reynolds number is

$$\mathrm{Re}=\frac{Uh}{\nu}=0.58 \tag{6.1}$$

Due to the asymmetric geometry, it is not appropriate to implement the symmetric or periodic boundary conditions. Instead, no-slip bounce back conditions are applied at the solid structure (Porter *et al* 2006). Flow in the void is driven by a constant pressure difference dp/dx. Fluid with a constant density is assumed initially. As shown in Figure 6.4, after 4600 time steps of calculation, numerical convergence is achieved with the largest L^2-norm error being smaller than 1×10^{-6}. The errors of *x*-, *y*- and *z*- velocities, drop at nearly equal rate from 10^{-2} to less than 1×10^{-6}.

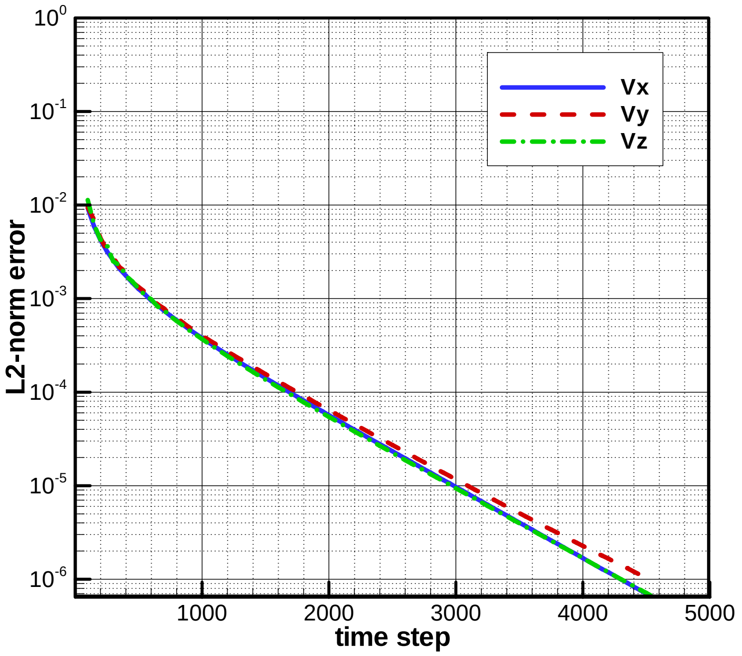


Figure 6.4 The L^2-norm error of *x*-, *y*- and *z*- velocity.

The contours of velocity magnitude are shown in Figure 6.5 as an overview of the flow along the x-direction. The void in the center represents the solid phase. The high velocity is shown in red colour, which lies mainly in the left hand of the obstacle.

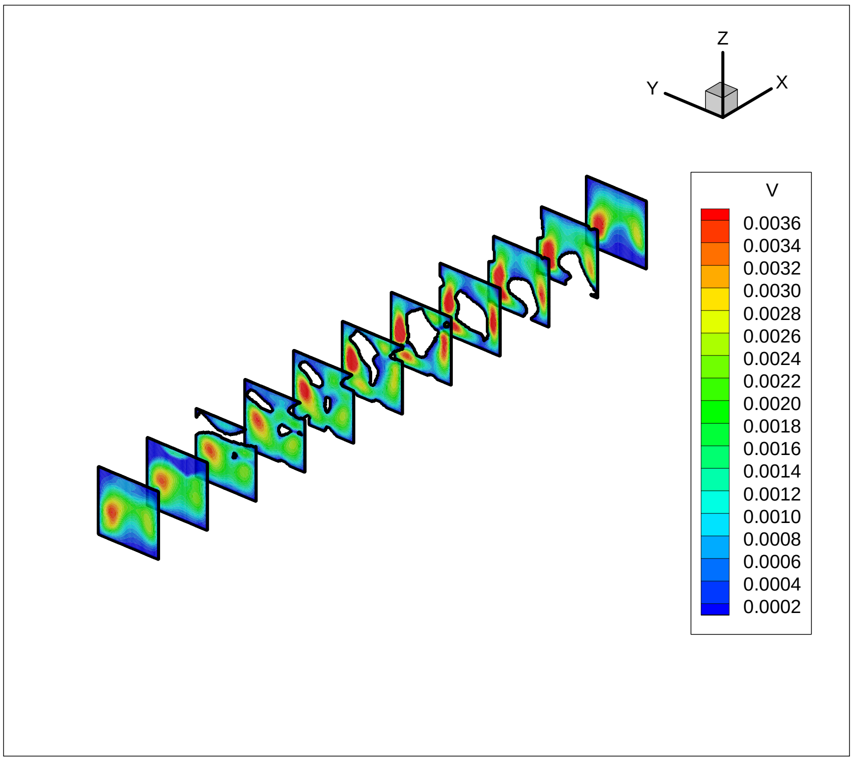


Figure 6.5 Contours of velocity magnitude at different cross-sections along the flow direction.

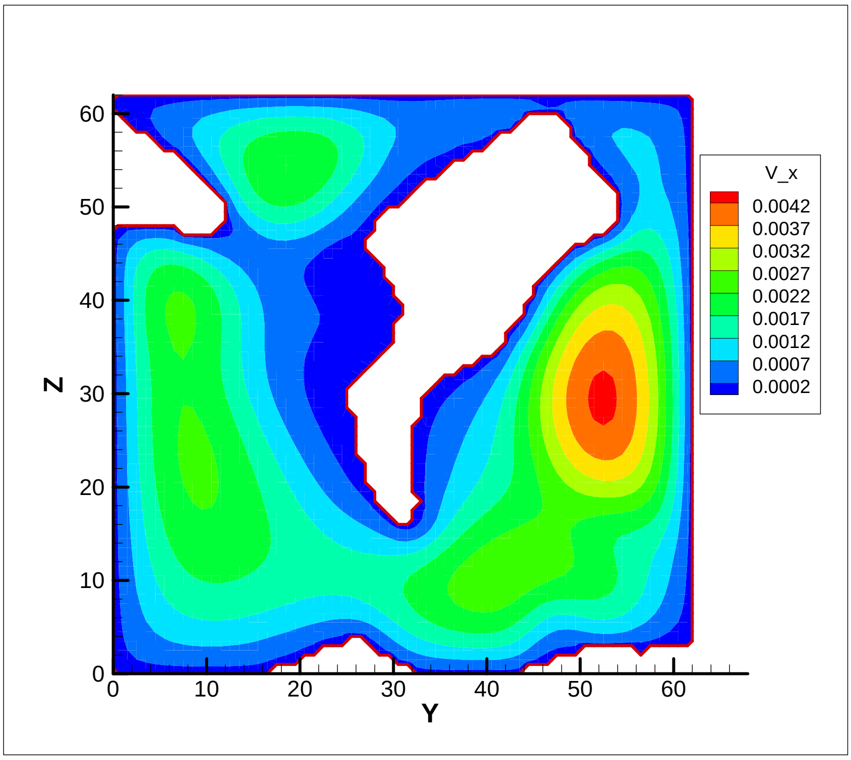


Figure 6.6 Contours of the *x*-velocity in the middle plane, *x* = 32.

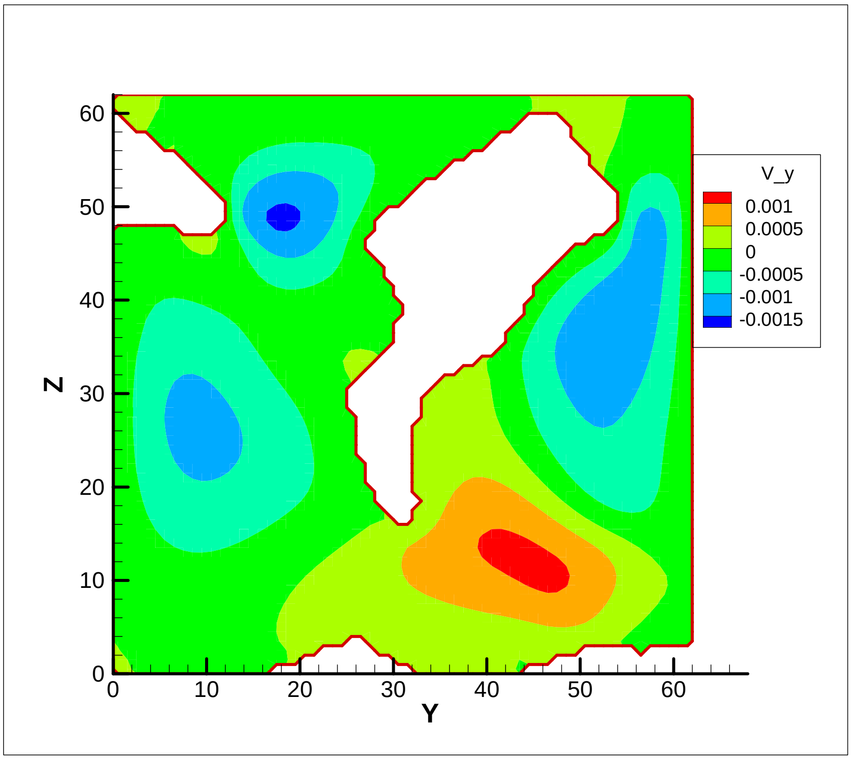


Figure 6.7 Contours of the *y*-velocity in the middle plane, *x* = 32.

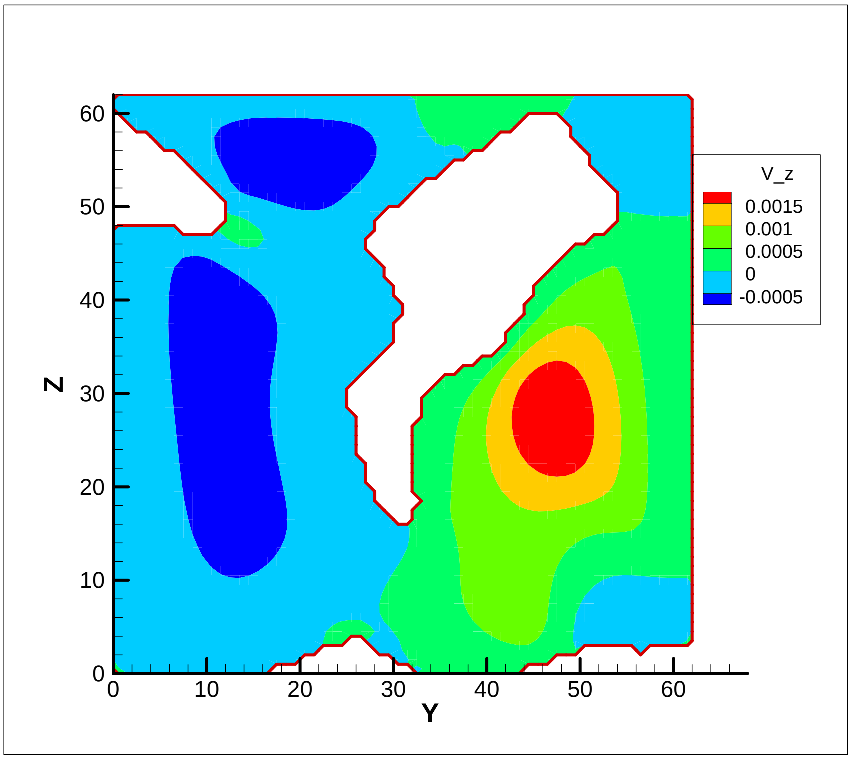


Figure 6.8 Contours of the *z*-velocity in the middle plane, *x* = 32.

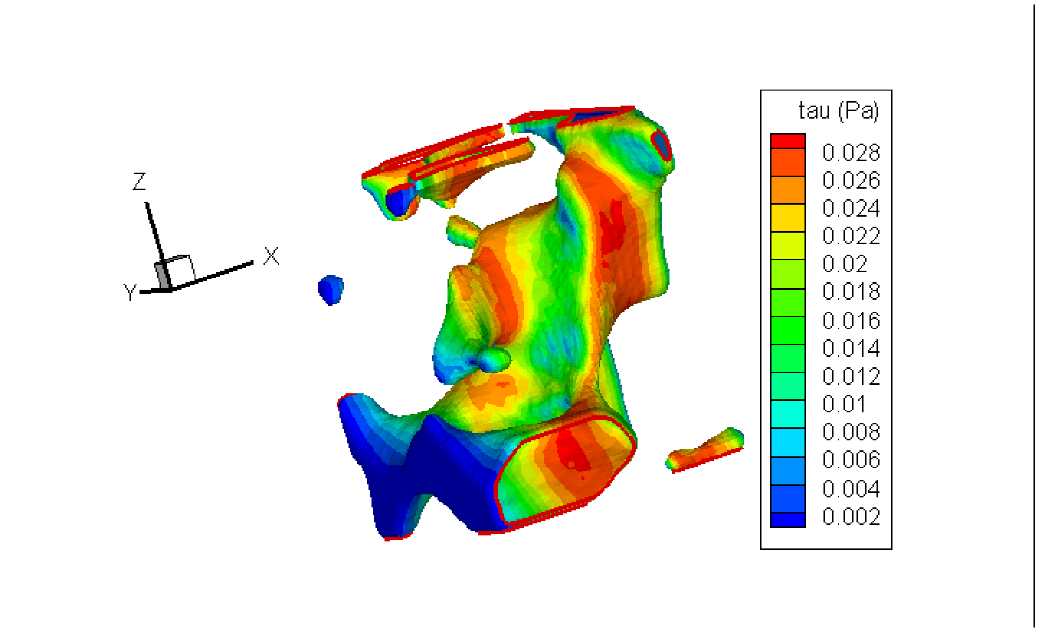


Figure 6.9 Contours of the wall shear stress on the scaffold.

Figure 6.6 to Figure 6.8 are the contours of the three velocity components in the midplane x = 32. The maximum value of x-, y- and z- velocity is of the same order at ~0.0015 *m/s*. This indicates that the portion of the porous scaffold chosen for flow simulation is randomly structured with no preferred direction of distribution. The wall shear stress contour on the scaffold material is shown in Figure 6.9. Near the entrance, the surface of the solid is normal to the flow direction, and the wall shear stress is close to zero due to the static flow in that region. The shear stress reaches its maximum value of ~0.028 Pa in regions with rapid variation in velocity.

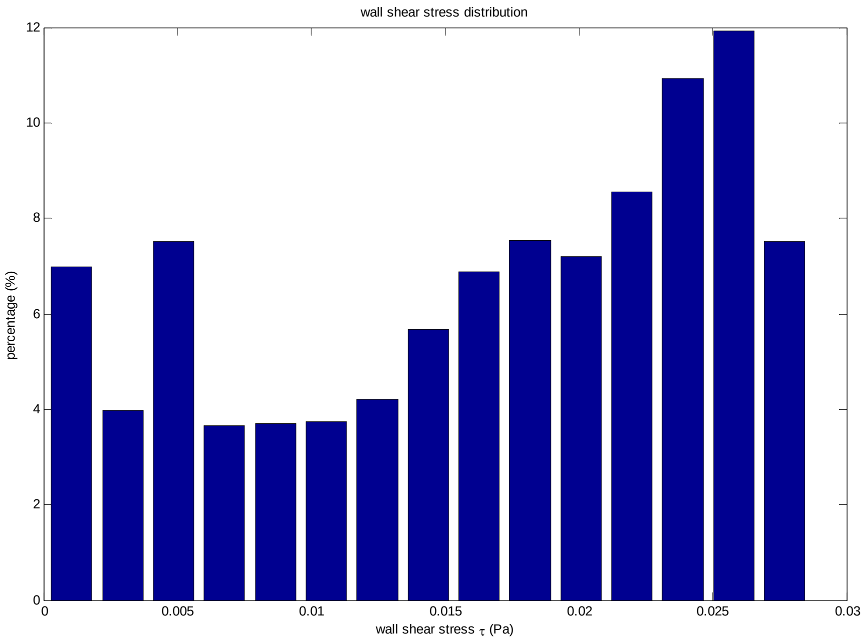


Figure 6.10 The distribution of wall shear stress values in the porous scaffold.

Figure 6.10 is the histogram of the distribution of wall shear stress. It is seen that in all regions of the scaffold, wall shear stress is low, i.e. below 0.03 Pa. Nearly 50% of the scaffold surface is subjected to shear stress equal or greater than 0.02 Pa. Less than 25% of the scaffold surface has shear stress values < 0.01 Pa. The shear stress distribution can help us to identify the range of the shear stress that seeded cells experience and, with information on cell locations within the scaffold, provides information on population of cells under different shear stress magnitude. Clearly, the distribution of the shear stress is not unique and is dependent on scaffold structure and flow conditions.

6.3 Darcy's law

For flow with low Reynolds number, its governing equation can be simplified from the Navier-Stokes equations to the Stokes equation,

$$\frac{\partial \vec{u}}{\partial t} + \nabla p = \mu \nabla^2 \vec{u} \tag{6.2}$$

The linear equation dictates proportional change in the velocity with the pressure gradient increases, i.e.

$$\frac{\partial (\alpha \vec{u})}{\partial t} + \alpha (\nabla p) = \mu \nabla^2 (\alpha \vec{u}) \tag{6.3}$$

It is obvious that

$$\nabla p = b \vec{u} \tag{6.4}$$

where b is the ratio which depends on the structure of the porous media. Equation (6.4) still holds by replacing the velocity u with the average velocity U in a cross-section along x-axis.

$$\frac{dp}{dx} = \frac{\mu}{K} U \tag{6.5}$$

where K is the permeability. This well-known linear relationship is first reported by Henry Darcy in 1856, by conducting a number of experiments of water through beds of sand (Darcy 1856).

When we investigate the relationship between the pressure drop and the average velocity, we achieved following results, shown in Table 6.1:

Table 6.1 The average velocity under different pressure gradient in lattice unit.

Pressure gradient (Pa)	Average velocity (m/s)
1.00E-07	1.03E-05
2.50E-07	2.58E-05
5.00E-07	5.16E-05
1.00E-06	1.03E-04
2.50E-06	2.59E-04
5.00E-06	5.19E-04
1.00E-05	1.04E-03
2.00E-05	2.12E-03
4.00E-05	4.38E-03

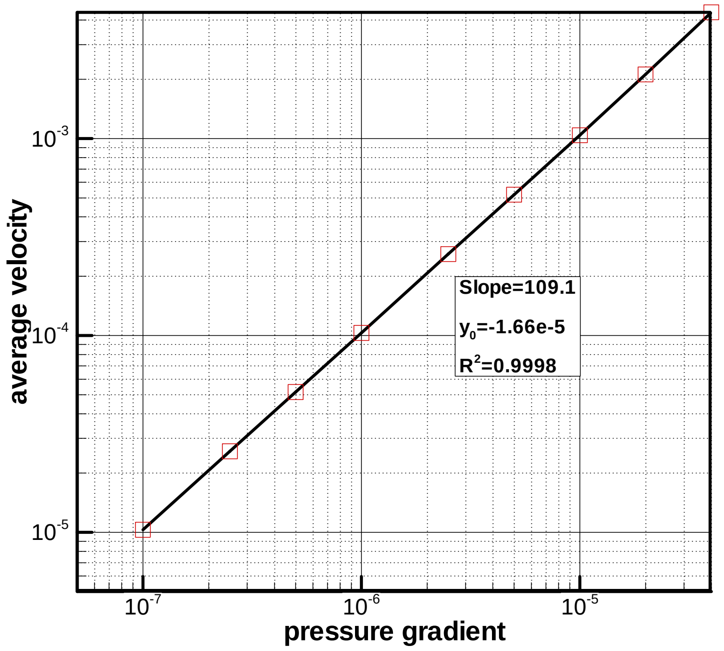


Figure 6.11 The average velocity against the pressure gradient.

By fitting the average velocity as a linear function of pressure gradient, we have got the slope of this fitting as 109.1, the point at which this line intersects the average velocity is -1.66×10^{-5} *m/s*, the correlation coefficient of these two columns is 0.9998, as depicted in Figure 6.11. We can easily calculate the permeability of the scaffold using Darcy's law as

$$\begin{aligned} K &= (\Delta x)^2 \times slope \\ &= (0.05\times10^{-3})^2 \times 109.1 = 2.73\times10^{-7}\,m^2 \end{aligned} \tag{6.6}$$

6.4 Relationship between the pressure drop and the average wall shear stress

Analog to the discussion of average velocity, it is easy to find that the average wall shear stress is proportional to the pressure gradient as well from the Stokes equation (6.2). For the flow through a periodic square array of cylinders, the average wall shear stress is a function of the pressure drop $\frac{dp}{dx}$, permeability K and porosity ϕ as (Wang & Tarbell 1995)

$$\bar{\tau} = -\sqrt{K}\,\frac{dp}{dx}\,\frac{4}{\pi}\,\frac{1-0.319285\phi^2-0.043690\phi^4}{\sqrt{\left(1-\phi-0.305828\phi^4\right)\left(1+\phi-0.305828\phi^4\right)}} \tag{6.7}$$

A similar linear expression has been assumed recently as

$$\bar{\tau} = B\frac{\mu}{\sqrt{K}}U \tag{6.8}$$

and the value of factor B was found as $B = 1.07 \pm 0.03$ (Roman *et al* 2010).

In our study, we assume the average shear stress is proportional to the applied pressure gradient instead of average velocity

$$\bar{\tau} = \bar{B}\sqrt{K}\,\frac{dp}{dx} \tag{6.9}$$

This assumption can be beneficial from the less parameters taken into account (2 parameters from 3). We have simulated flow in 4 additional cubes with $64 \times 64 \times 64$ lattice units from different part of the porous disk under different pressure gradient to work out their permeability. The permeability of the 5 cases (in total) are shown in Figure 6.12. The permeability of the case 2 is almost 2 times greater than that of the case 4, although they have very similar porosity. This implies that the Kozeny–Carman equation (McCabe *et al* 2005) does not holds at the mesoscopic level. The resistant force through a porous media varies due to the difference of micro-structure between these cases.

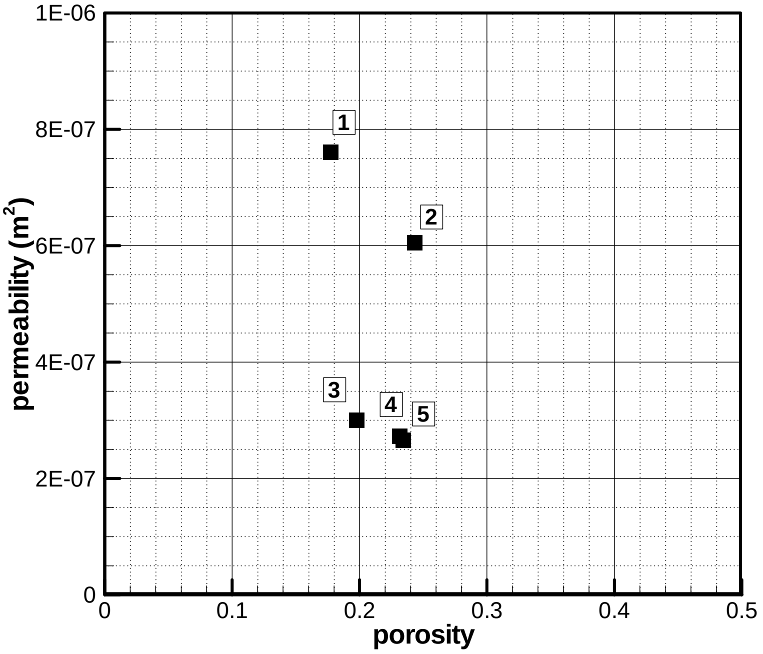


Figure 6.12 Permeability against porosity of 5 cubic volumes in the scaffold disc.

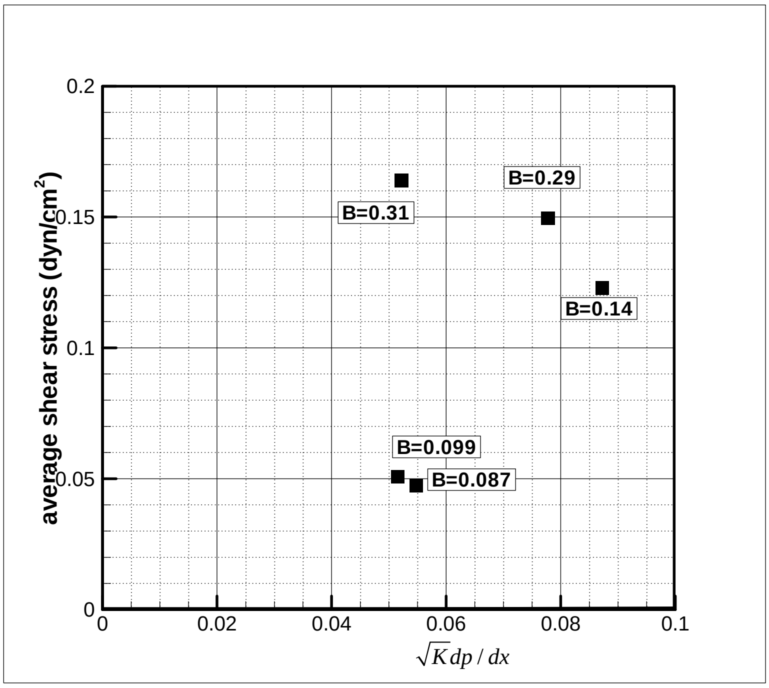


Figure 6.13 The average shear stress against $\sqrt{K}dp/dx$ of the 5 different objects, labelled with the value of corresponding coefficient $\overline{B}$.

On the other hand, we can estimate the $\overline{B}$ value in Equation (6.9) by working out the average shear stresses $\overline{\tau}$ and the corresponding value of $\sqrt{K}dp/dx$. In Figure 6.13, we present results based on the 5 cases. The average value of B is 0.16 with the standard deviation of 0.08.

6.5 Summary

We have developed a program to reconstruct the 3D porous scaffold from the micro-CT images. The lattice Boltzmann method is then applied to simulation flow in a small cubic portion of the porous scaffold. The detailed information on the velocity and the wall shear stress distributions has been presented. A near perfect linear relationship is found between the average velocity and the pressure gradient, largely due to the fact that our study is focused on flow in the porous scaffold with very low flow velocity. From the relation, we can calculate the permeability of the porous structure.

While the Kozeny–Carman equation predicts that the permeability has a simple relation with the porosity, our study has demonstrated that it may not hold true in the microscopic level, although the permeability has been found to be of the same order of magnitude, i.e. $10^{-7} m^2$ in all cases. Furthermore, we have tried to find an empirical equation to estimate the average wall shear stress, by assuming a linear relationship between it and the value of $\sqrt{K}\frac{dp}{dx}$. If such a relation exists, the slope between the two, based on our study is approximately 0.16 ± 0.08. One possible limitation in our estimation is in the fact that the size of the cubic structure we used to simulate flow is fairly small. Wall effects, as imposed to surround the cubic structure may have had a strong effect on the value of the slop we estimated.

7 Conclusion

The lattice Boltzmann method is based on the linear BGK collision operator, and can lead to the Navier-Stokes equations by applying the Chapman-Enskog expansion under low velocity. In our current study, the magnitude of Reynolds number is of the order of unity, which makes the lattice Boltzmann method eligible in our simulations.

We have developed computer programs based on the lattice Boltzmann method in FORTRAN 95, a general-purpose programming language which is especially suitable for high-performance scientific computation. Two dimensional and three dimensional programs, with D2Q9 model of 9 different lattice velocities and D3Q19 of 19 different lattice velocities, respectively, have been developed in separate programs.

We have carried out simulation of 2D Poiseuille flow between two infinite planar planes and of flow in a 3D rectangular duct to validate the computer programs. For the 2D Poiseuille flow, the L^2-norm residual between the numerical simulation and the analytic solution is approximately 1×10^{-3} for the velocity, and is significantly lower, i.e. $\sim10^{-14}$ for the shear stress. Besides, the average standard error of normal shear stress in the x-axis τ_{xx} is 3.8×10^{-6}, 1/1000th of the velocity error. The much smaller error in the shear stress than that of the velocity supports the theoretical derivation, in which the calculation of the shear stress is directly from the distribution function, and is independent of the velocity.

The 3D lattice Boltzmann program is verified by the simulation of flow in an infinite duct with rectangular shape. The L^2-norm differences between two consecutive time steps for the *x*-, *y*- and *z*- components of velocity are less than 1×10^{-6}. The x-velocity error between lattice Boltzmann simulation and analytic results is ~0.14%, which is of satisfactory accuracy for a mesh of 30×30 or above. A mesh-independent study has been carried out on the computer program based on the lattice Boltzmann method. Even at a fairly low lattice density, e.g. at 10 x 10 lattice units across the domain, the error of velocity is below 1%, which is usually considered as of acceptable accuracy in most engineering numerical calculations. As the lattice density increases, the relative error decreases further, but no apparent improvement can be seen when the lattice unit number is above 30. We have discovered the convergence of shear stress is slower than that of the velocity, which implies that the L^2-norm of shear stress might be a better indicator, if accurate results are required and the computer time is less considered.

To further check the accuracy of our program in complex geometries, the creeping flows around either an array of square cylinder or a porous scaffold originated from micro-CT images have been studied using both the lattice Boltzmann method and Fluent in 2D flow simulations. Fluent, as a commercially available Navier-Stokes equation solver based on the finite volume method, is used here for comparison. Results in both geometries provide very similar results in by LBM and by Fluent. Our study provides further evidence that the lattice Boltzmann method provides satisfactory accuracy with lower expenses in terms of CPU time in comparison to commercial software (Geller *et al* 2006).

It has been reported that cells cultured in 2D monolayers behave differently in a number of ways to those *in vivo*, where they are commonly subjected to 3D environment (Benya & Shaffer 1982; Zhang *et al* 2010). The use of 3D scaffolds can provide a way to overcome this problem by providing flow-induced mechanical stimulation and allowing cells to synthesis 3D extracellular matrix (ECM) (Bancroft *et al* 2002). However, the local velocity and shear stress that are experienced by the cells in a 3D scaffold can be different by a few orders of magnitude under the same flow conditions. Detailed information on local shear stress and its distribution are needed to investigate their effects on cell activities, such as cell growth, proliferation and differentiation (Porter *et al* 2005). In order to achieve this, we have developed a program to reconstruct the 3D porous scaffold from micro-CT images produced in our laboratory, and then selected a small portion of the porous scaffold to simulate 3D flow inside. Currently, it is still beyond the capability of computers in most research groups to mimic flow through the whole porous disk (Maes *et al* 2009; Porter *et al* 2005).

In the current study, a $64\times64\times64$ cubic portion subjected to

$3.2mm \times 3.2mm \times 3.2mm$ in the center of the porous disk is simulated using the lattice Boltzmann method. Detailed distribution on the velocity and wall shear stress has been calculated. A histogram distribution of the wall shear stress gives the percentage of wall surface under a specific range of shear stress. A near perfect linear relationship is found between the average velocity and the pressure gradient in our simulation. This results from the fact that flow in the porous scaffold in our simulation has very low velocity. The permeability of the porous structure can be readily calculated from the velocity-pressure gradient relation by applying the Darcy's law (Darcy 1856).

We have selected additional four $64 \times 64 \times 64$ lattice units from different part of the porous disk and simulated flow in them. In all cases, linear relationship between the average velocity and the pressure gradient holds. Average porosity of the 5 cubic volumes is 0.217, with a standard deviation of 0.025. This shows that the porous disk has a rather uniform porosity at different locations. The permeability in all 5 cases has been found to be of the same order of magnitude, i.e.$10^{-7} m^2$. We further extended our model to consider an empirical equation that has been proposed to estimate the average wall shear stress, by assuming a linear relationship between its value and the value of $\sqrt{K} dp / dx$. Our preliminary results show that if such a relation exists, the slope between these two, based on our study is approximately 0.16 ± 0.08.

There are a number of studies that link cell proliferation rate to the shear stress. In the current study, we have developed a numerical model based on the LBM to account for the time-history effect of the shear stress on cell growth with evolving geometries (i.e. as cell growths). It is intended as a preliminary study based on a hypothetical relation between the cell proliferation rate and the accumulative effect of the shear stress. In the model, effects of the cell cycle and other factors, such as contact-inhibition are modelled by a ‘cell state probability function’, *p*. A cell will divide into 2 daughter cells when the *p* value reaches 1 or above. This study is designed to demonstrate the capacity of the LBM model to simulate such a process, and to reveal the intrinsic advantage of the LBM method over other conventional numerical methods in tacking moving boundary problems.

There are a number of limitations in the current study. They include:

The lattice Boltzmann method, as a new numerical methods, is still under development. Several issues have been raised by researchers, e.g. the viscosity is limited in a certain range due to numerical stability (Worthing *et al* 1997). It is very difficult to implement a ‘good’ boundary condition (i.e. second order, mass-conserved, numerical stable) for curved walls. Some schemes with high order accuracy does not conserve the mass, while the bounce back condition only have one-order accuracy in general case (Chun & Ladd 2007; Verschaeve 2009).

It has been reported that the shear stress is underestimated near the wall

(Porter *et al* 2005). However, the error caused by the interpolation and calculation of normal vector in the zig-zag boundary has not been investigated.

Due to the capacity of computers used in the study, we have only simulated flow in a small cubic portion of the porous disk. The effect of the imposed wall surrounding (side) walls of the cube has not been properly investigated. In future studies, a bigger portion or the whole porous disc can be considered.

To estimate the average wall shear stress, we have assumed a linear relationship between $\bar{\tau}$ and $\sqrt{K}\dfrac{dp}{dx}$. Parameters, such as the average pore size, surface area, tortuosity, need to be taken into consideration in future studies.

The proposed model on the interaction between the flow shear stress and cell proliferation needs to be carefully considered and improved using experimental data.

In conclusion, we have developed a new numerical model based on the lattice Boltzmann method to study flow in porous scaffolds. It provides satisfactory results with a number of intrinsic advantages over conventional numerical methods, and can be further developed into a robust tool to investigate performance of scaffold materials in 3D cell culture.

References

Ahrenholz B, Tolke J, Krafczyk M. 2006. Lattice-Boltzmann simulations in reconstructed parametrized porous media. *Int J Comput Fluid D* 20:369-77

Alberts B, Johnson A, Lewis J, Raff M, Roberts K, Walter P. 2002. *Molecular Biology of the Cell, Fourth Edition*: Garland

Alvarez-Barreto J, Linehan S, Shambaugh R, Sikavitsas V. 2007. Flow Perfusion Improves Seeding of Tissue Engineering Scaffolds with Different Architectures. *Annals of Biomedical Engineering* 35:429-42

Bakker AD, Soejima K, Klein-Nulend J, Burger EH. 2001. The production of nitric oxide and prostaglandin E-2 by primary bone cells is shear stress dependent. *Journal of Biomechanics* 34:671-7

Bancroft GN, Sikavitsast VI, van den Dolder J, Sheffield TL, Ambrose CG, et alet al. 2002. Fluid flow increases mineralized matrix deposition in 3D perfusion culture of marrow stromal osteloblasts in a dose-dependent manner. *Proceedings of the National Academy of Sciences of the United States of America* 99:12600-5

Benya PD, Shaffer JD. 1982. Dedifferentiated chondrocytes reexpress the differentiated collagen phenotype when cultured in agarose gels. *Cell* 30:215-24

Bhatia S. 1999. *Microfabrication in tissue engineering and bioartificial organs*. Boston: Kluwer Academic Publishers. xx, 145 p. pp.

Bhatnagar PL, Gross EP, Krook M. 1954. A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. *Physical Review* 94:511-25

Blokhra RL, Khajuria M. 1991. Flow through Porous-Media .13. Hydrodynamic Flow Studies of Aqueous-Solutions of Methanol, Sucrose, Sodium-Bicarbonate and Their Mixtures through Antimony Oxide Membrane. *Journal of Membrane Science* 62:211-8

Botchwey EA, Pollack SR, El-Amin S, Levine EM, Tuan RS, Laurencin CT. 2003. Human osteoblast-like cells in three-dimensional culture with fluid flow. *Biorheology* 40:299-306

Bouzidi M, Firdaouss M, Lallemand P. 2001. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of Fluids* 13:3452-9

Boyd J, Buick J, Green S. 2006. A second-order accurate lattice Boltzmann non-Newtonian flow model. *J. Phys. A-Math. Gen.* 39:14241-7

Cartmell S, Huynh K, Lin A, Nagaraja S, Guldberg R. 2004. Quantitative microcomputed tomography analysis of mineralization within three-dimensional scaffolds in vitro. *Journal of Biomedical Materials Research Part A* 69A:97-104

Cartmell SH, Porter BD, Garcia AJ, Guldberg RE. 2003. Effects of medium perfusion rate on cell-seeded three-dimensional bone constructs in vitro. *Tissue Engineering* 9:1197-203

Chapman S, Cowling TG, Burnett D. 1970. The mathematical theory of non-uniform gases; an account of the kinetic theory of viscosity, thermal conduction and diffusion in gases. [Cambridge, Eng.]: Cambridge University Press. xxiv, 423 p. pp.

Chen S, Chen H, Martnez D, Matthaeus W. 1991a. Lattice Boltzmann model for simulation of magnetohydrodynamics. *Phys Rev Lett* 67:3776-9

Chen S, Doolen GD. 1998. Lattice Boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.* 30:329-64

Chen SY, Diemer K, Doolen D, Eggert K, Fu C, et alet al. 1991b. Lattice Gas Automata for Flow through Porous-Media. *Physica D* 47:72-84

Cheng G, Youssef BB, Markenscoff P, Zygourakis K. 2006. Cell Population Dynamics Modulate the Rates of Tissue Growth Processes. *Biophysical journal* 90:713-24

Cheng Z, Tan FPP, Riga CV, Bicknell CD, Hamady MS, et alet al. Analysis of Flow Patterns in a Patient-Specific Aortic Dissection Model. *Journal of Biomechanical Engineering* 132:051007-9

Chun B, Ladd AJC. 2007. Interpolated boundary condition for lattice Boltzmann simulations of flows in narrow gaps. *Phys Rev E* 75:066705

Chung CA, Lin T-H, Chen S-D, Huang H-I. 2010. Hybrid cellular automaton modeling of nutrient modulated cell growth in tissue engineering constructs. *Journal of Theoretical Biology* 262:267-78

Cioffi M, Boschetti F, Raimondi MT, Dubini G. 2006. Modeling evaluation of the fluid-dynamic microenvironment in tissue-engineered constructs: A micro-CT based model. *Biotechnology and Bioengineering* 93:500-10

Cornubert R, d'Hum D, es, Levermore D. 1991. A Knudsen layer theory for lattice gases. *Phys. D* 47:241-59

d'Humieres D, Lallemand P, Frisch U. 1986. Lattice Gas Models for 3d

Hydrodynamics. *Europhysics Letters* 2:291-7

Darcy H. 1856. Les Fontaines Publiques de la ville de Dijan. *V. Dalmont, Paris*:647

Dhumieres D, Lallemand P, Frisch U. 1986. Lattice Gas Models for 3d Hydrodynamics. *Europhys Lett* 2:291-7

Filippova O, Hanel D. 1998. Grid refinement for lattice-BGK models. *J Comput Phys* 147:219-28

Firdaouss M, Guermond JL, LeQuere P. 1997. Nonlinear corrections to Darcy's law at low Reynolds numbers. *Journal of Fluid Mechanics* 343:331-50

Folkman J, Moscona A. 1978. Role of cell shape in growth control. *Nature* 273:345-9

Frisch U, d'Humieres D, Hasslacher B, Lallemand P, Pomeau Y, Rivet JP. 1986a. *Lattice gas hydrodynamics in two and three dimensions*. Medium: ED; Size: Pages: 56 pp.

Frisch U, d'Humières D, Hasslacher B, Lallemand P, Pomeau Y, Rivet JP. 1987. Lattice Gas Hydrodynamics in Two and Three Dimensions. *Complex Systems* 1:649-707

Frisch U, Hasslacher B, Pomeau Y. 1986b. Lattice-Gas Automata for the Navier-Stokes Equation. *Physical Review Letters* 56:1505-8

Fritton SP, Weinbaum S. 2009. Fluid and Solute Transport in Bone: Flow-Induced Mechanotransduction. *Annu Rev Fluid Mech* 41:347-74

Gardner M. 1970. Fantastic Combinations of John Conways New Solitaire

Game Life. *Scientific American* 223:120-&

Geller S, Krafczyk M, Tolke J, Turek S, Hron J. 2006. Benchmark computations based on lattice-Boltzmann, finite element and finite volume methods for laminar flows. *Comput Fluids* 35:888-97

Ginzbourg I, Adler PM. 1994. Boundary flow condition analysis for the three-dimensional lattice Boltzmann model. *J. Phys. II France* 4:191-214

Grunau D, Chen SY, Eggert K. 1993. A Lattice Boltzmann Model for Multiphase Fluid-Flows. *Physics of Fluids a-Fluid Dynamics* 5:2557-62

Gutierrez RA, Crumpler ET. 2008. Potential effect of geometry on wall shear stress distribution across scaffold surfaces. *Annals of Biomedical Engineering* 36:77-85

Hardy J, Depazzis O, Pomeau Y. 1976. Molecular-Dynamics of a Classical Lattice Gas - Transport Properties and Time Correlation-Functions. *Phys Rev A* 13:1949-61

Hardy J, Pomeau Y, de Pazzis O. 1973. Time evolution of a two-dimensional model system. I. Invariant states and time correlation functions. *Journal of Mathematical Physics* 14:1746-59

Hargittai I. 2002. *The road to Stockholm : Nobel Prizes, science, and scientists*. Oxford ; New York: Oxford University Press. xvii, 342 p., [24] p. of plates pp.

He XY, Luo LS. 1997a. Lattice Boltzmann model for the incompressible Navier-Stokes equation. *Journal of Statistical Physics* 88:927-44

He XY, Luo LS. 1997b. A priori derivation of the lattice Boltzmann equation.

Phys Rev E 55:R6333-R6

Higuera FJ, Jimenez J. 1989. Boltzmann Approach to Lattice Gas Simulations. *EPL (Europhysics Letters)* 9:663

Hollister SJ. 2005. Porous scaffold design for tissue engineering. *Nature Materials* 4:518-24

Hollister SJ. 2006. Porous scaffold design for tissue engineering (vol 4, pg 518, 2005). *Nature Materials* 5:590-

Hu, H. 1999. Multi-slice helical CT: Scan and reconstruction. Medical Physics **26**(1): 5-18.

Irgens F. 2008. *Continuum mechanics*. Berlin: Springer. xviii, 661 p. pp.

Junk M, Yang Z. 2005. One-point boundary condition for the lattice Boltzmann method. *Phys Rev E* 72:066701

Kao PH, Yang RJ. 2008. An investigation into curved and moving boundary treatments in the lattice Boltzmann method. *Journal of Computational Physics* 227:5671-90

Karlin IV, Gorban AN, Succi S, Boffi V. 1998. Maximum Entropy Principle for Lattice Kinetic Equations. *Physical Review Letters* 81:6

Koelman JMVA. 1991a. A Simple Lattice Boltzmann Scheme for Navier-Stokes Fluid-Flow. *Europhys Lett* 15:603-7

Koelman JMVA. 1991b. A Simple Lattice Boltzmann Scheme for Navier-Stokes Fluid Flow. *EPL (Europhysics Letters)* 15:603

Kruger T, Varnik F, Raabe D. 2009. Shear stress in lattice Boltzmann simulations. *Phys Rev E* 79:046704

Krafczyk M, Cerrolaza M, Schulz M, Rank E. 1998. Analysis of 3D transient blood flow passing through an artificial aortic valve by Lattice-Boltzmann methods. *Journal of Biomechanics* 31:453-62

Kuhn JL, Goldstein SA, Feldkamp LA, Goulet RW, Jesion G. 1990. Evaluation of a microcomputed tomography system to study trabecular bone structure. *Journal of Orthopaedic Research* 8:833-42

Kumar BVR, Naidu KB. 1995. Finite-Element Analysis of Nonlinear Pulsatile Suspension Flow Dynamics in Blood-Vessels with Aneurysm. *Computers in Biology and Medicine* 25:1-20

Lallemand P, Luo LS. 2003. Lattice Boltzmann method for moving boundaries. *J Comput Phys* 184:406-21

Langer R, Vacanti JP. 1993. Tissue Engineering. *Science* 260:920-6

Lappa M. 2003. Organic tissues in rotating bioreactors: Fluid-mechanical aspects, dynamic growth models, and morphological evolution. *Biotechnology and Bioengineering* 84:518-32

Latt J, Chopard B, Malaspinas O, Deville M, Michler A. 2008. Straight velocity boundaries in the lattice Boltzmann method. *Phys Rev E* 77:056703

Lavallee P, Boon JP, Noullez A. 1991. Boundaries in lattice gas flows. *Physica D: Nonlinear Phenomena* 47:233-40

Lee Y, Kouvroukoglou S, McIntire LV, Zygourakis K. 1995. A cellular automaton model for the proliferation of migrating contact-inhibited cells.

Biophysical journal 69:1284-98

Macchiarini P, Jungebluth P, Go T, Asnaghi MA, Rees LE, et alet al. 2008. Clinical transplantation of a tissue-engineered airway. *Lancet* 372:2023-30

Maes F, Van Ransbeeck P, Van Oosterwyck H, Verdonck P. 2009. Modeling fluid flow through irregular scaffolds for perfusion bioreactors. *Biotechnology and Bioengineering* 103:621-30

McAllister TN, Frangos JA. 1999. Steady and transient fluid shear stress stimulate NO release in osteoblasts through distinct biochemical pathways. *Journal of Bone and Mineral Research* 14:930-6

McCabe WL, Smith JC, Harriott P. 2005. *Unit operations of chemical engineering*. Boston: McGraw-Hill. xxv, 1140 p. pp.

Mcnamara GR, Zanetti G. 1988. Use of the Boltzmann-Equation to Simulate Lattice-Gas Automata. *Phys Rev Lett* 61:2332-5

Mei RW, Luo LS, Shyy W. 1999. An accurate curved boundary treatment in the lattice Boltzmann method. *Journal of Computational Physics* 155:307-30

Nadobny J, SArntenings M, Diehl D, Stetter E, Brinker G, Wust P. 2007. Evaluation of MR-induced hot spots for different temporal SAR modes using a time-dependent finite difference method with explicit temperature gradient treatment. *Ieee Transactions on Biomedical Engineering* 54:1837-50

Olariu A, Cleaver KM, Cameron HA. 2007. Decreased neurogenesis in aged rats results from loss of granule cell precursors without lengthening of the cell cycle. *The Journal of Comparative Neurology* 501:659-67

Olson JF, Rothman DH. 1997. Two-fluid flow in sedimentary rock: Simulation, transport and complexity. *Journal of Fluid Mechanics* 341:343-70

Orszag SA, Yakhot V. 1986. Reynolds Number Scaling of Cellular-Automaton Hydrodynamics. *Physical Review Letters* 56:1691

Pan CX, Luo LS, Miller CT. 2006. An evaluation of lattice Boltzmann schemes for porous medium flow simulation. *Comput Fluids* 35:898-909

Porter B, Zauel R, Stockman H, Guldberg R, Fyhrie D. 2005. 3-D computational modeling of media flow through scaffolds in a perfusion bioreactor. *Journal of Biomechanics* 38:543-9

Premnath KN, Abraham J. 2005. Lattice Boltzmann model for axisymmetric multiphase flows. *Physical Review E* 71:-

Qian YH, Dhumieres D, Lallemand P. 1992. Lattice BGK Models for Navier-Stokes Equation. *Europhys Lett* 17:479-84

Qian YH, et alet al. 1992. Lattice BGK Models for Navier-Stokes Equation. *EPL (Europhysics Letters)* 17:479

Risbud MV, Karamuk E, Moser R, Mayer J. 2002. Hydrogel-Coated Textile Scaffolds as Three-Dimensional Growth Support for Human Umbilical Vein Endothelial Cells (HUVECs): Possibilities as Coculture System in Liver Tissue Engineering. *Cell Transplantation* 11:369-77

Rivet JP, Frisch U. 1986. LATTICE GAS AUTOMATA IN THE BOLTZMANN APPROXIMATION. *Comptes Rendus De L Academie*

Des Sciences Serie Ii 302:267-&

Roman SV, Samuel BV, Vassilios IS, Dimitrios VP. 2010. Distribution of flow-induced stresses in highly porous media. *Applied Physics Letters* 97:024101

Schiff JA. 2002. An unsung hero of medical research. *Yale Alumni Magazine*

Sen PN. 1989. Unified Model of Conductivity and Membrane-Potential of Porous-Media. *Physical Review B* 39:9508-17

Shibeshi SS, Collins WE. 2005. The rheology of blood flow in a branched arterial system. *Applied Rheology* 15:398-405

Shigematsu M, Watanabe H, Sugihara H. 1999. Proliferation and differentiation of unilocular fat cells in the bone marrow. *Cell Struct Funct* 24:89-100

Skordos PA. 1993. Initial and boundary conditions for the lattice Boltzmann method. *Phys Rev E* 48:4823

Smith JA, Martin L. 1973. Do Cells Cycle? Proceedings of the National Academy of Sciences of the United States of America 70:1263-7

Spaid MAA, Phelan FR. 1997. Lattice Boltzmann methods for modeling microscale flow in fibrous porous media. *Physics of Fluids* 9:2468-74

Sterling JD, Chen SY. 1996. Stability analysis of lattice Boltzmann methods. *J Comput Phys* 123:196-206

Stolberg S, McCloskey KE. 2009. Can shear stress direct stem cell fate? *Biotechnology Progress* 25:10-9

Sun N, Leung JH, Wood NB, Hughes AD, Thom SA, et alet al. 2009. Computational analysis of oxygen transport in a patient-specific model of abdominal aortic aneurysm with intraluminal thrombus. *Br J Radiol* 82:S18-23

Takahashi K, Suzuki K. 1996. Density-Dependent Inhibition of Growth Involves Prevention of EGF Receptor Activation by E-Cadherin-Mediated Cell-Cell Adhesion. *Experimental Cell Research* 226:214-22

Tinku A, Ping-Sing T. Computational foundations of image interpolation algorithms. *Ubiquity* 2007:1-17

Torii R, Keegan J, Wood NB, Dowsey AW, Hughes AD, et alet al. 2009. The effect of dynamic vessel motion on haemodynamic parameters in the right coronary artery: a combined MR and CFD study. *Br J Radiol* 82:S24-32

Unser M, Aldroubi A, Eden M. 1995. Enlargement or reduction of digital images with minimum loss of information. *Image Processing, IEEE Transactions on* 4:247-58

Verschaeve JCG. 2009. Analysis of the lattice Boltzmann Bhatnagar-Gross-Krook no-slip boundary condition: Ways to improve accuracy and stability. *Phys Rev E* 80:036703

Wagner AJ, Pagonabarraga I. 2002. Lees-Edwards boundary conditions for lattice Boltzmann. *Journal of Statistical Physics* 107:521-37

Wang DM, Tarbell JM. 1995. Modeling interstitial flow in an artery wall allows estimation of wall shear stress on smooth muscle cells. *J Biomech Eng* 117:358-63

Wang S, Tarbell J. 2000. Effect of fluid flow on smooth muscle cells in a 3-dimensional collagen gel model. *Arteriosclerosis, thrombosis, and vascular biology* 20:2220-5

White FM. 1991. *Viscous fluid flow*. New York: McGraw-Hill. xxi, 614 p. pp.

Wolf-Gladrow DA. 2000. Lattice-gas cellular automata and lattice Boltzmann models : an introduction. Berlin ; New York: Springer. ix, 308 p. pp.

Wolfram S. 1986. Cellular Automaton Fluids .1. Basic Theory. *Journal of Statistical Physics* 45:471-526

Worthing RA, Mozer J, Seeley G. 1997. Stability of lattice Boltzmann methods in hydrodynamic regimes. *Phys Rev E* 56:2243

Yoshino A, Hotta Y, Hirozane T, Endo M. 2007. A numerical method for incompressible non-Newtonian fluid flows based on the lattice Boltzmann method. *J. Non-Newton. Fluid Mech.* 147:69-78

Zhang JF, Johnson PC, Popel AS. 2008. Red blood cell aggregation and dissociation in shear flows simulated by lattice Boltzmann method. *Journal of Biomechanics* 41:47-55

Zhang Y, Li T-S, Lee S-T, Wawrowsky KA, Cheng K, et alet al. 2010. Dedifferentiation and Proliferation of Mammalian Cardiomyocytes. *PLoS ONE* 5:e12559

Ziegler DP. 1993. Boundary conditions for lattice Boltzmann simulations. *Journal of Statistical Physics* 71:1171-7

Zou QS, He XY. 1997. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Physics of Fluids* 9:1591-8

Appendix: Main code of 2D lattice Boltzmann method

```
Program LBM
use ComPara, only: CP_max_t, CP_frame, CP_tol, Flag_BC_force, CP_time_step, CP_L2_err, lx, ly, gamma
use Node_data_D2Q9, only: macro, macro_temp
!use CellCulture
implicit none
!integer ierr
real*8 :: tic, toc, tic1, toc1
!real*8 t0,t1,t2
integer time,max_t,frame,ti, min_t
integer i, j, k, ierr
  character*40 flog
  character Flag_resume
!-----------initialize
  call CPU_time(tic)
  call init_parameters_comment

! call read_geometry


  call init_density
  !if(FLAG_BC_Pressure .OR. FLAG_BC_Velocity) then
  call init_BC
  !end if
! call init_CellCulture
  call init_FE
  call cal_FE_fluid
  call write_fluid_structure
! call FE_category
! call output_neutral
! stop
! call write_cell_structure
  call CPU_time(toc)
  write(*, '(/1x,a,e9.3,a)') "Elapsed time for initialization is ",toc-tic," seconds."
  write(*,'(/1x,a)')"============================================================"
```

```
!-----------LOOP
  call CPU_time(tic)
max_t=CP_max_t
frame=CP_frame
ti=0
fLog=".\Results\lbm.log"
!open(9,file=flog)
!stop
  write(*                                  ,'(/1x,a)')"========================MAIN
LOOP==========================="
  write(9                                  ,'(/1x,a)')"========================MAIN
LOOP==========================="
  CP_L2_err=1.d0
  !cell_generation=0
!do while(cell_no<cell_max_no .and. cell_generation<cell_max_generation)
! judge to resume or not
        write(* ,'(/1x,a)')"Do you want to resume your calculation? (y) or (n)"
        read(*,'(a)')Flag_resume
        !min_t=1
        if( Flag_resume=='y') then
                call read_cas_file(min_t)
                write(*,*)"Resume calculation from time step=", min_t
                open(9,file=fLog, STATUS = "OLD", ACTION = "WRITE", IOSTAT = ierr, ACCESS
="APPEND")
                write(9,*)"resume calculation"
        else
                open(9,file=fLog, STATUS = "NEW", ACTION = "WRITE", IOSTAT = ierr)
                min_t=1
        end if

        !main loop

        do time=min_t,max_t+min_t
                if(maxval(CP_L2_err)>CP_tol) then
                        call collision
                        call propagation
                        call bounceback
                        call BC_treatment
                        call cal_macro
                        if( time==min_t) then
                                !call CPU_time(tic1)
                                macro_temp=macro
                                !call CPU_time(toc1)
                                !write(*, '(/1x,a,e9.3,a)') "Elapsed time for copy macro is ",toc1-tic1," seconds."
```

```
                    end if
                    if(mod(time,CP_time_step)==0) then

write(* ,'(/1x,a)')"========================================================"
                          ti=ti+1
                          call cal_L2_error
                          call check(time,tic,toc,ti)
      !                   call write_fluid_cell_vel(cell_generation)
                          call FE_cal_vel_wss
                          call write_fluid_vel_wss(1)
                          call write_cas_file(time)
                    end if
      !             write(*,*) time
              end if

      end do
      !initial the cell
!       if( cell_generation==0) then
!             call init_CellCulture
!             call write_cell_structure(cell_generation)
!       end if
!      call write_fluid_cell_vel(cell_generation)
!       call Cell_growth
!      call write_cell_structure(cell_generation)
!      CP_L2_err=1.d0
!      write(*,*)"cell_generation=",cell_generation
!      stop
!end do
  call del_mem
  close(9)
end program LBM

!> define the common parameters


Module ComPara
!-----------LBM parameters----------------
integer :: lx, ly !<computation domain
real*8    :: density, omega, force !< density, relexation parameter, body force
integer :: CP_max_t, CP_frame, CP_time_step
real*8    :: CP_tol
real*8    :: CP_L2_err(3)
! the precondition parameters
! cf. I:\LBM\Reference\Optimization\PhysRevE.70.066706-1.pdf for more details
```

```
real*8    :: gamma




!-----------mathematic constant------------
real*8, parameter:: PI              = 3.14159265358979323846d0 !< accurate enough :P
real*8, parameter:: CP_Cs        = 0.57735026918962576451d0
real*8, parameter:: CP_Cs_sq = 0.33333333333333333333d0
real*8, parameter:: CP_bct      = 0.5d0 !< for the upper and bottom boundary, esp. for Poiseuille flow

!----------functions------------------------
real*8 , external :: my_mod
logical, external :: fun_no_fracture
logical, external :: fun_is_fluid
integer, external :: fun_point_number
integer, external :: fun_4point_case
real*8 , external :: fun_distance
real*8 , external :: fun_theta
integer, external :: FE_node_xy
real*8 , external :: fun_cal_rate

!----------BC flag--------------------
logical :: FLAG_Debug_Poisseuille=.True.
logical :: FLAG_BC_Pressure=.True.
logical :: FLAG_BC_Velocity=.True.
logical :: FLAG_BC_FH            =.True.
logical :: FLAG_BC_force      =.True.
real*8    :: BC_Pressure_inlet
real*8    :: BC_Pressure_outlet

end module ComPara

Module FE
implicit none
!-----------Output: finite element---------
!real*8 ,ALLOCATABLE, dimension(:, :) :: FE_node !< x,y coordinate for each all point
!integer,ALLOCATABLE, dimension(:, :) :: FE_seq    !< connectivity
real*8 ,ALLOCATABLE, dimension(:, :) :: FE_fluid_node !< x,y coordinate for each fluid point
integer,ALLOCATABLE, dimension(:, :) :: FE_fluid_seq    !< fluid connectivity
!integer,ALLOCATABLE, dimension(:, :, :) :: FE_node_rate    !< 4- sequence of fluid interface particles
!integer,ALLOCATABLE, dimension(:, :) :: node_xy              !< sequency of fluid particles
real*8 ,ALLOCATABLE, dimension(:, :) :: FE_wss_node
integer,ALLOCATABLE, dimension(:, :) :: FE_wss_seq
real*8 ,ALLOCATABLE, dimension(:, :) :: FE_wss_area
```

```
real*8 ,ALLOCATABLE, dimension(:, :) :: FE_wss_sort
integer,ALLOCATABLE, dimension(:, :) :: FE_fluid_seq_1
!<----------flag ----------------------------
!integer
!----------count----------------------------
!integer :: FE_seq_number=0          !< the number of fluid meshes
integer :: FE_number(4)=0      !< the number of fluid particles
integer :: FE_node_number(4)=0 !< the mount of 4-direction interface
integer :: FE_seq_fluid_number=0 !< the mount of total fluid nodes
integer :: FE_total_node_number=0
integer :: FE_wss_node_number=0
!integer :: FE_wss_seq_number =0
integer :: FE_wss_sort_i=10
!functions
end Module FE

Module Node_Data_D2Q9
implicit none
Type Node
!>NodeType
!!0 - fluid
!!1 - solid
!!2 - fluid, interface
!!3 - solid, interface
integer :: NodeType=0
!record of queue
integer :: rate_ij =0
! macroscopic density
!real*8    :: density=0.0d0
! x,y-direction velocity
!real*8    :: velocity(2)=0.0d0
end type Node

!define the parameter in the node
real*8 :: D2Q9_weight(0:8),D2Q9_c(0:8,2)
real*8 ,ALLOCATABLE, dimension(:, :, :) :: fbar
real*8 ,ALLOCATABLE, dimension(:, :, :) :: feq
real*8 ,ALLOCATABLE, dimension(:, :, :) :: macro
real*8 ,ALLOCATABLE, dimension(:, :, :) :: macro_temp
type(Node), pointer ,dimension(:,        :) :: pNode
!integer,ALLOCATABLE, dimension(:, :      ) :: NodeType
end module Node_Data_D2Q9

! Boundary Condition module
```

```
Module BoundaryCondition
implicit none
type BC_Velocity
! Zou & He (POF 1997)
! i,j -- position
! index: id
!            1 -- east
!            2 -- north
!            3 -- west
!            4 -- south
! u_x,u_y: given velocity
integer :: i,j,id
real*8 :: u_x,u_y
end type BC_Velocity

Type BC_Pressure
! Zou & He (POF 1997)
! i,j -- position
! index: id
!            1 -- east
!            2 -- north
!            3 -- west
!            4 -- south
! u_1: given velocity along the boundary
! rou:
integer :: i,j,id
real*8 :: u_1=0.d0, rou
end type BC_Pressure
!> Filippova and Hanel boundary treatment
Type BC_FH
!!NodeType
!!0 - fluid
!!1 - solid
!!2 - fluid, interface
!!3 - solid, interface
!integer :: NodeType=0
!!Boundary condition type
!!x,y-direction distance to the node
!real*8 ::rate(8)=0.0d0

integer :: i,j
integer :: id(0:4)=0
real*8 :: rate(8)=0.d0
end type BC_FH
```

```
type(BC_velocity), pointer:: pBC_velocity(:)
type(BC_pressure), pointer:: pBC_pressure(:)
type(BC_FH),             pointer:: pBC_FH(:)
end module BoundaryCondition

! Cell Culture module
Module CellCulture
implicit none
type cell
integer :: i,j
real*8 :: p, wss
end type cell
type(cell), pointer::pCell(:)
integer:: cell_no, cell_x_min, cell_x_max
integer:: cell_max_no, cell_max_generation, cell_generation
real*8 :: cell_max_wss, cell_dead_wss
real*8 , external :: fun_wss_pro
end module


subroutine init_parameters_comment
!//////////////////////////////////////////////////////////////////
! initialize the parameter
use ComPara
use Node_Data_D2Q9
use FE, only: FE_wss_sort_i
implicit none
integer :: ierr
real*8    :: t0
namelist /parameter/ lx,ly, density, omega, force, &
  FLAG_Debug_Poisseuille, &
  FLAG_BC_Pressure, FLAG_BC_Velocity, FLAG_BC_FH, FLAG_BC_force, &
  CP_max_t, CP_frame, CP_time_step, CP_tol, &
  FE_wss_sort_i

write(*,'(/1x,a)')"============================================================"
    write(*,'(/1x,a)')"initize the parameter from parameter.in"
    open(1, file='.\input\parameter.in', STATUS = "OLD", ACTION = "READ", &
    IOSTAT = ierr)
    if( ierr==0) then
        ! computation domain
          read(1,*)
        read(1,*)lx
```

```
              read(1,*)
            read(1,*)ly
              read(1,*)
            read(1,*)density
              read(1,*)
            read(1,*)omega
              read(1,*)
            read(1,*)force
            ! boundary condition flag
!             read(1,*)
!            read(1,*)BC_Pressure_inlet
!             read(1,*)
!            read(1,*)BC_Pressure_outlet
!            ! Boundary condition
              read(1,*)
            read(1,*)FLAG_Debug_Poisseuille
              read(1,*)
            read(1,*)FLAG_BC_Pressure
              read(1,*)
            read(1,*)FLAG_BC_Velocity
              read(1,*)
            read(1,*)FLAG_BC_FH
              read(1,*)
            read(1,*)FLAG_BC_force



            ! temporal parameter
              read(1,*)
            read(1,*)CP_max_t
              read(1,*)
            read(1,*)CP_frame
              read(1,*)
              read(1,*)CP_time_step
              read(1,*)
            read(1,*)CP_tol
!             read(1,*)
!            read(1,*)FE_wss_sort_i
      end if
      close(1)
      !-------report the parameter
      open(1, file='.\results\parameter.out')
      write(1,nml=parameter)
      close(1)
      !-------allocate fbar
```

```
    allocate(fbar(lx,ly,0:8),stat=ierr)
    if( ierr/=0) then
        write(*,'(/1x,a)')"Allocate fbar FAILED."
        stop
    end if
    !-------allocate feq
    allocate(feq(lx,ly,0:8),stat=ierr)
    if( ierr/=0) then
        write(*,'(/1x,a)')"Allocated feq FAILED."
        stop
    end if

    !-------allocate macro
    allocate(macro(lx,ly,3),stat=ierr)
    if( ierr/=0) then
        write(*,'(/1x,a)')"Allocated macro FAILED."
        stop
    end if
    !-------allocate macro_temp
    allocate(macro_temp(lx,ly,3),stat=ierr)
    if( ierr/=0) then
        write(*,'(/1x,a)')"Allocated macro_temp FAILED."
        stop
    end if
    !-------allocate NodeType
    allocate(pNode(lx,ly),stat=ierr)
    if( ierr/=0) then
        write(*,'(/1x,a)')"Allocated pNode FAILED."
        stop
    end if
    !-------init weight and lattice unit velocity
    t0=1.d0/36.d0
    D2Q9_weight(0)=t0*16.d0
    D2Q9_weight(1:4)=t0*4.d0
    D2Q9_weight(5:8)=t0
    D2Q9_c((/0,2,4/),1)=0.d0
    D2Q9_c((/1,5,8/),1)=1.d0
    D2Q9_c((/3,6,7/),1)=-1.d0
    D2Q9_c((/0,1,3/),2)=0.d0
    D2Q9_c((/2,5,6/),2)=1.d0
    D2Q9_c((/4,7,8/),2)=-1.d0

end subroutine init_parameters_comment
```

```
!!//////////////////////////////////////////////////////////////////
!initialize the variable of each node
subroutine init_density
use ComPara, only : density, lx, ly, CP_cs_sq, BC_Pressure_inlet, gamma
use Node_Data_D2Q9

implicit none
integer i,j,k
real*8 :: u_n(8),u_x, u_y, u_squ, d_loc, u_max
write(*,'(/1x,a)') 'initializing the equilibrium distribution function feq and fbar'
macro(:,:,2)=BC_Pressure_inlet
macro(:,:,3)=0.d0
!macro(:,:,1)=dsqrt(macro(:,:,2)*macro(:,:,2)+macro(:,:,3)*macro(:,:,3))
!u_max=MAXVAL(macro(:,:,1))
!gamma=(10.d0*dsqrt(3.d0)*u_max)**2
macro(:,:,1)=density



do j=1, ly, 1
    do i=1, lx, 1
           d_loc= macro(i,j,1)
           u_x   = macro(i,j,2)
           u_y   = macro(i,j,3)

            u_n(1) =     u_x
            u_n(2) =           u_y
            u_n(3) = - u_x
            u_n(4) =         - u_y
            u_n(5) =     u_x + u_y
            u_n(6) = - u_x + u_y
            u_n(7) = - u_x - u_y
            u_n(8) =     u_x - u_y
            u_squ = u_x * u_x + u_y * u_y

!...........zero velocity density

            feq(i,j,0) = D2Q9_weight(0) * d_loc * (1.d0 - u_squ / (2.d0 * CP_cs_sq))
            do k=1,8
            feq(i,j,k) = D2Q9_weight(k)* d_loc * (1.d0 + u_n(k) / CP_cs_sq &
                    + u_n(k) ** 2.d0 / (2.d0 * CP_cs_sq ** 2.d0)   &
                    - u_squ / (2.d0 * CP_cs_sq) )
            end do
    end do
end do
```

```
fbar(:,:,:)=feq(:,:,:)
!write(1,*) 'initial density'
end subroutine init_density

!////////////////////////////////////////////////////////////////////
! initial boundary condition from different file
subroutine init_BC
use ComPara, only: FLAG_BC_Velocity, FLAG_BC_Pressure,    flag_BC_FH
!use BoundaryCondition
implicit none
!integer i,j, ierr, p_i,v_i
write(*,'(/1x,a)') 'Initializing Boundary Conditions...'
if( flag_BC_Pressure==.True.) then
  call init_Pressure
end if
if( flag_BC_Velocity==.True.) then
  call init_Velocity
end if
if( flag_BC_FH==.True.) then
  call init_FH
end if
end subroutine init_BC
!////////////////////////////////////////////////////////////////////
! initial pressure from BC_pressure.in file
subroutine init_Pressure
use ComPara, only: ly
use BoundaryCondition
implicit none
integer ierr, p_i, i
real*8 BC_Pressure_outlet, BC_Pressure_inlet
write(*,'(/1x,a)') 'initializing pressure and velocity boundary condition'
!init pressure outlet
! Poisseuille flow
if( FLAG_Debug_Poisseuille==.True.) then
!       inlet
        !p_i=2*(ly-2)
        p_i=ly-2
        BC_Pressure_inlet=0.8d0
        BC_Pressure_outlet=0.79d0
          if(FLAG_BC_Pressure) then


                allocate(pBC_pressure(p_i),stat=ierr)
                if(ierr /= 0) then
                        write(*,'(/1x,a)')"Allocated pBC_pressure FAILED."
                end if
                do i=1,ly-2
                        pBC_pressure(i)%i=lx
                        pBC_pressure(i)%j=i+1
                        pBC_pressure(i)%id=1
                        pBC_pressure(i)%u_1=0.0d0
                        pBC_pressure(i)%rou=BC_Pressure_outlet
                end do
                do i=ly-1,p_i
                        pBC_pressure(i)%i=1
                        pBC_pressure(i)%j=i-ly+2
                        pBC_pressure(i)%id=3
                        pBC_pressure(i)%u_1=0.0d0
                        pBC_pressure(i)%rou=BC_Pressure_inlet
                end do
        end if
else

        open(1, file='.\input\BC_pressure.in', STATUS = "OLD", ACTION = "READ", &
        IOSTAT = ierr)
        if( ierr /= 0) then
                write(*,'(/1x,a)')"Open file BC_pressure.in FAILED."
                STOP
        end if
        read(1,*)p_i
        allocate(pBC_pressure(p_i),stat==ierr)
        if(ierr /= 0) then
                write(*,'(/1x,a)')"Allocated pBC_pressure FAILED."
        end if
!     # read comments
        read(1,*)
        do i=1,p_i
                read(1,*) pBC_Pressure(i)%i, pBC_Pressure(i)%j,pBC_Pressure(i)%id, pBC_Pressure(i)%u_1,
pBC_Pressure(i)%rou
        end do
end if
end subroutine init_Pressure

!//////////////////////////////////////////////////////////////////////
! initial velocity from BC_velocity.in file
subroutine init_Velocity
!init velocity inlet
!v_i=2*(ly-2)
implicit none
```

```
integer v_i,i
logical flag_BC_velocity_DEBUG
flag_BC_Velocity_DEBUG=.False.
If( flag_BC_Velocity_DEBUG) then
      v_i=ly-2
!     if(FLAG_BC_Velocity) then
      allocate(pBC_velocity(v_i),stat=ierr)
      if(ierr /= 0) then
            write(*,'(/1x,a)')"Allocated pBC_Velocity is unsuccessful."
      end if
      do i=1,ly-2
            pBC_velocity(i)%i=1
            pBC_velocity(i)%j=i+1
            pBC_velocity(i)%id=3
            pBC_velocity(i)%u_x=1.d-3
            pBC_velocity(i)%u_y=0.d0
      end do
!       do i=ly-1,v_i
!             pBC_velocity(i)%i=lx
!             pBC_velocity(i)%j=i-ly+2
!             pBC_velocity(i)%id=1
!             pBC_velocity(i)%u_x=1.d-3
!             pBC_velocity(i)%u_y=0.d0
!       end do
else

      open(1, file='.\input\BC_velocity.in', STATUS = "OLD", ACTION = "READ", &
      IOSTAT = ierr)
      if( ierr /= 0) then
            write(*,'(/1x,a)')"Open file BC_velocity.in FAILED."
            STOP
      end if
      read(1,*)v_i
      allocate(pBC_Velocity(v_i),stat==ierr)
      if(ierr /= 0) then
            write(*,'(/1x,a)')"Allocated pBC_pressure FAILED."
      end if
!       # read comments
      read(1,*)
      do i=1,v_i
            read(1,*) pBC_Velocity(i)%i, pBC_Velocity(i)%j,pBC_Velocity(i)%id, pBC_Velocity(i)%u_x,
pBC_Velocity(i)%u_y
      end do
end if
end subroutine init_Velocity

!///////////////////////////////////////////////////////////////////////////
subroutine init_FH
!///////////////////////////////////////////////////////////////////////////
!initialize the geometry
use ComPara
use Node_Data_D2Q9
use BoundaryCondition
use FE, only: FE_number, FE_node_number
implicit none
integer ierr
integer i,j,k, FH_number(0:4), temp , temp1!, temp2


write(*,'(/1x,a)')"Reading geometry from .\input\FH_indicator.dat & FH_rate_file.dat "

    !-----------open the indicator.dat file
    open(1,file='.\input\FH_indicator.dat', STATUS = "OLD", ACTION = "READ", &
    IOSTAT = ierr)
    if( ierr/=0) then
        write(*,'(/1x,a)')"read file .\input\FH_indicator.dat FAILED."
        stop
    end if
    open(2,file='.\input\FH_rate_file.dat', STATUS = "OLD", ACTION = "READ", &
    IOSTAT = ierr)
    !----------read file
    read(1, *)
    read(1, *)FH_number(0)
    ! allocate node
    allocate(pBC_FH(0:FH_number(0)),stat=ierr)
    if( ierr/=0) then
        write(*,'(/1x,a)')"Allocate pBC_FH FAILED."
        stop
    end if
    ! read NodeType from indicator.dat
    read(1, *)FH_number(1)
    read(1, *)FH_number(2)
    read(1, *)FH_number(3)
    read(1, *)FH_number(4)
!       temp1=0
!       temp2=0
```

```
    do i=1,lx
        do j=1,ly
            !write(*,*) i,j

            read(1,*)pNode(i,ly+1-j)%NodeType
            select case(pNode(i,ly+1-j)%NodeType)
                case(0)
                    FE_number(1)=FE_number(1)+1
                    pNode(i,ly+1-j)%rate_ij=FH_number(3)+FH_number(4)+FE_number(1)
                case(1)
                    FE_number(2)=FE_number(2)+1

pNode(i,ly+1-j)%rate_ij=FE_number(2)+FH_number(3)+FH_number(4)+FH_number(1)

                case(2)
                    FE_number(3)=FE_number(3)+1
                    pNode(i,ly+1-j)%rate_ij=FE_number(3)
                    pBC_FH(FE_number(3))%id(0)=FE_number(3)
                    read(2, *) pBC_FH(FE_number(3))%i
                    read(2, *) temp1
                    pBC_FH(FE_number(3))%j=ly+1-temp1
                    read(2, '(8e14.9)') pBC_FH(FE_number(3))%rate(:)
                    do k=1,4
                        if(pBC_FH(FE_number(3))%rate(k)>0.d0) then
                            FE_node_number(k)=FE_node_number(k)+1
                            pBC_FH(FE_number(3))%id(k) =FE_node_number(k)
                        end if
                    end do
                case(3)
                    FE_number(4)=FE_number(4)+1
                    pNode(i,ly+1-j)%rate_ij=FE_number(4)+FH_number(3)
                    temp=pNode(i,ly+1-j)%rate_ij
                    read(2, *) pBC_FH(temp)%i
                    read(2, *) temp1
                    pBC_FH(temp)%j=ly+1-temp1
                    read(2, '(8e14.9)') pBC_FH(temp)%rate(:)
            end select

        end do
    end do


    close(1)
    close(2)
    !-----check the input file
    if( maxval(ABS(FE_number(1:4)-FH_number(1:4)))>0) then
        write(*,*)"in subroutine read_geometry, input file header is wrong"
        stop
    end if




!write(1,'(/1x,a)')"end of subroutine init_geo."
!write(1,'(/1x,a)')"------------------------------------------ "
!write(*,'(/1x,a)')"end of subroutine read_geo."
!write(*,'(/1x,a)')"------------------------------------------ "
end subroutine init_FH


!//////////////////////////////////////////////////////////////////
subroutine read_cas_file(time)
use ComPara
use Node_data_D2Q9
implicit none
integer i,j,k
integer time
  character*40 flog
  write(*,'(/1x,a)') 'read feq from lbm2d.cas'
  fLog=".\input\lbm2d.cas"
  open(10,file=fLog,STATUS='OLD')
  ! write the time step
  read(10,'(I10)')time
do j=1,ly,1
    do i=1,lx,1
        read(10,'(9e20.11)')(feq(i,j,k),k=0,8)
    end do
end do
!write(9,*)feq(20,30,:)
close(10)
fbar(:,:,:)=feq(:,:,:)
end subroutine read_cas_file

!//////////////////////////////////////////////////////////////////
!collision
subroutine collision
use ComPara, only: lx, ly, omega, fun_is_fluid
use Node_data_D2Q9, only: feq, fbar
```

```
implicit none
integer i,j,k
real*8 max_tau,t1(0:8),t2(0:8),t3(0:8)
!calculate equillibium function feq
! call cal_feq

do j=1,ly,1
   do i=1,lx,1
      if(fun_is_fluid(i,j)) then
         !do k=0,8,1
         ! Reference
         ! JFM Y Li, etc
         ! 2004, vol. 519, pp. 273
            t1(:)=feq(i,j,:)
            t2(:)=fbar(i,j,:)
              t3=1.d0-t1/t2
             max_tau=MAXVAL(t3)
              max_tau=DMAX1(max_tau, omega)
            feq(i,j,:)=(1.0d0-max_tau)*t2+max_tau*t1
         !end do
      end if
   end do
end do
!write(1,*) 'collision finish'
!write(*,*) 'collision finish'
end subroutine collision
!///////////////////////////////////////////////////////////////////
!propagation
subroutine propagation
!integer time
use ComPara, only: lx, ly
use Node_data_D2Q9, only: feq, fbar
implicit none
integer i, j, k, x_e, x_w, y_n, y_s

do j=1,ly,1
   do i=1,lx,1
!.........compute upper and right next neighbour nodes with regard
!              to periodic boundaries
               y_n = mod(j,ly) + 1
               x_e = mod(i,lx) + 1

!.........compute lower and left next neighbour nodes with regard to
!              periodic boundaries
```

```
          y_s = ly - mod(ly + 1 - j, ly)
          x_w = lx - mod(lx + 1 - i, lx)

!.........zero
          fbar(i    ,j    ,0) = feq(i,j,0)
!.........east
          fbar(x_e,j    ,1) = feq(i,j,1)
!.........north
          fbar(i    ,y_n,2) = feq(i,j,2)
! .........west
          fbar(x_w,j    ,3) = feq(i,j,3)
! .........south
          fbar(i    ,y_s,4) = feq(i,j,4)
! .........north-east
          fbar(x_e,y_n,5) = feq(i,j,5)
! .........north-west
          fbar(x_w,y_n,6) = feq(i,j,6)
! .........south-west
          fbar(x_w,y_s,7) = feq(i,j,7)
! .........south-east
          fbar(x_e,y_s,8) = feq(i,j,8)

  end do
end do

!write(1,*) 'propagation'

end subroutine propagation
!////////////////////////////////////////////////////////////////////
!implement bounceback scheme in the solid particles
subroutine bounceback
use ComPara, only: lx, ly, fun_is_fluid
use Node_data_D2Q9, only: feq, fbar, pNode

implicit none
integer i,j, x_e, x_w, y_n, y_s
real*8 :: temp(0:8)



do j=1,ly,1
  do i=1,lx,1
      !debug
```

```
!solid or solid interface
    if(fun_is_fluid(i,j)==.False.) then
          y_n = mod(j,ly) + 1
          x_e = mod(i,lx) + 1
          y_s = ly - mod(ly + 1 - j, ly)
          x_w = lx - mod(lx + 1 - i, lx)

!.........east
          fbar(x_e,j    ,1) = fbar(i,j,3)
!.........north
          fbar(i    ,y_n,2) = fbar(i,j,4)
! .........west
          fbar(x_w,j    ,3) = fbar(i,j,1)
! .........south
          fbar(i    ,y_s,4) = fbar(i,j,2)
! .........north-east
          fbar(x_e,y_n,5) = fbar(i,j,7)
! .........north-west
          fbar(x_w,y_n,6) = fbar(i,j,8)
! .........south-west
          fbar(x_w,y_s,7) = fbar(i,j,5)
! .........south-east
          fbar(x_e,y_s,8) = fbar(i,j,6)
    end if
    end do
  end do



end subroutine bounceback

!/////////////////////////////////////////////////////////////////////
!implement bounceback scheme in the solid particles
subroutine BC_treatment
use ComPara, only: Flag_BC_FH, Flag_BC_force, Flag_BC_Pressure, FLAG_BC_Velocity
!use Node_data_D2Q9, only: feq, fbar, pNode
!use BoundaryCondition
implicit none
! add force
if( Flag_BC_force) then
    call add_force
end if
! implement the velocity & pressure boundary
if( Flag_BC_Pressure .OR. Flag_BC_Velocity) then
    call BC_treatment_vp
```

```
end if
! implement the FH boundary
if( Flag_BC_FH) then
        call BC_treatment_FH
end if
end subroutine BC_treatment
!//////////////////////////////////////////////////////////////////////
!adding force 0.01 pascal/lu
subroutine add_force
use ComPara, only: lx, ly, force, fun_is_fluid, gamma
use Node_data_D2Q9, only: fbar, feq
implicit none
real*8 ::    t1,t2
integer i,j
t1=force/3.d0 /gamma
t2=force/12.d0 /gamma
do j=1,ly,1
    do i=1,lx,1
        if(fun_is_fluid(i,j)) then !
                if(fbar(i,j,3)>t1 .and. fbar(i,j,6) >t2 .and. fbar(i,j,7)>t2) then
                    fbar(i,j,1)=fbar(i,j,1)+t1
                    fbar(i,j,3)=fbar(i,j,3)-t1
                    fbar(i,j,5)=fbar(i,j,5)+t2
                    fbar(i,j,6)=fbar(i,j,6)-t2
                    fbar(i,j,7)=fbar(i,j,7)-t2
                    fbar(i,j,8)=fbar(i,j,8)+t2
                end if
        end if
    end do
end do
!write(1,*) 'add force 0.01 pascal/lu'
end subroutine add_force
!//////////////////////////////////////////////////////////////////////
!treat the velocity and pressure
subroutine BC_treatment_vp
use ComPara, only: Flag_BC_Pressure, FLAG_BC_Velocity
use Node_data_D2Q9, only: fbar
use BoundaryCondition
implicit none
integer :: time

integer :: i,j,k,p_i,v_i
!u_n -- normal velocity
real*8    :: u_n, rou_in, ftemp(0:8),u_t
```

```
!pressure Boundary treatment

 if(FLAG_BC_Pressure) then
    p_i=size(pBC_pressure)

    do k=1,p_i
        i=pBC_pressure(k)%i
        j=pBC_pressure(k)%j
        u_t=pBC_pressure(k)%u_1
        rou_in=pBC_pressure(k)%rou
        !Zou and He pressure boundary on East side
        if(pBC_pressure(k)%id==1) then
            ftemp(:)=fbar(i,j,:)
            u_n=1.d0-(2.d0*(ftemp(1)+ftemp(5)+ftemp(8))+(ftemp(0)+ftemp(2)+ftemp(4)))/rou_in
            fbar(i,j,3)=ftemp(1)+2.d0/3.d0*rou_in*u_n
            fbar(i,j,6)=ftemp(8)-.5d0*(ftemp(2)-ftemp(4))+rou_in*u_n/6.d0+0.5d0*rou_in*u_t
            fbar(i,j,7)=ftemp(5)+.5d0*(ftemp(2)-ftemp(4))+rou_in*u_n/6.d0-0.5d0*rou_in*u_t
        end if
        !Zou and He pressure boundary on West side
        if(pBC_pressure(k)%id==3) then
            ftemp(:)=fbar(i,j,:)
            u_n=1.d0-(2.d0*(ftemp(3)+ftemp(6)+ftemp(7))+(ftemp(0)+ftemp(2)+ftemp(4)))/rou_in
            fbar(i,j,1)=ftemp(3)+2.d0/3.d0*rou_in*u_n
            fbar(i,j,5)=ftemp(7)-.5d0*(ftemp(2)-ftemp(4))+rou_in*u_n/6.d0+0.5d0*rou_in*u_t
            fbar(i,j,8)=ftemp(6)+.5d0*(ftemp(2)-ftemp(4))+rou_in*u_n/6.d0-0.5d0*rou_in*u_t
        end if
        !Zou and He pressure boundary on North side
        if(pBC_pressure(k)%id==2) then
            ftemp(:)=fbar(i,j,:)
            u_n=1.d0-(2.d0*(ftemp(2)+ftemp(5)+ftemp(6))+(ftemp(0)+ftemp(1)+ftemp(3)))/rou_in
            fbar(i,j,4)=ftemp(2)+2.d0/3.d0*rou_in*u_n
            fbar(i,j,7)=ftemp(5)+.5d0*(ftemp(1)-ftemp(3))+rou_in*u_n/6.d0-0.5d0*rou_in*u_t
            fbar(i,j,8)=ftemp(6)-.5d0*(ftemp(1)-ftemp(3))+rou_in*u_n/6.d0+0.5d0*rou_in*u_t
        end if
        !Zou and He pressure boundary on South side
        if(pBC_pressure(k)%id==4) then
            ftemp(:)=fbar(i,j,:)
            u_n=1.d0-(2.d0*(ftemp(4)+ftemp(7)+ftemp(8))+(ftemp(0)+ftemp(1)+ftemp(3)))/rou_in
            fbar(i,j,2)=ftemp(4)+2.d0/3.d0*rou_in*u_n
            fbar(i,j,5)=ftemp(7)-.5d0*(ftemp(1)-ftemp(3))+rou_in*u_n/6.d0+0.5d0*rou_in*u_t
            fbar(i,j,6)=ftemp(8)+.5d0*(ftemp(1)-ftemp(3))+rou_in*u_n/6.d0-0.5d0*rou_in*u_t
        end if
    end do
end if
```

```
!velocity Boundary treatment

 if(FLAG_BC_Velocity) then
    v_i=size(pBC_velocity)

    do k=1,v_i
        i=pBC_velocity(k)%i
        j=pBC_velocity(k)%j

        !Zou and He velocity boundary on East side
        if(pBC_velocity(k)%id==1) then
            u_n=-pBC_velocity(k)%u_x
            u_t=pBC_velocity(k)%u_y
            ftemp(:)=fbar(i,j,:)
            rou_in=(2.d0*(ftemp(1)+ftemp(5)+ftemp(8))+(ftemp(0)+ftemp(2)+ftemp(4)))/(1.d0-u_n)

            fbar(i,j,3)=ftemp(1)+2.d0/3.d0*rou_in*u_n
            fbar(i,j,6)=ftemp(8)-.5d0*(ftemp(2)-ftemp(4))+rou_in*u_n/6.d0+0.5d0*rou_in*u_t
            fbar(i,j,7)=ftemp(5)+.5d0*(ftemp(2)-ftemp(4))+rou_in*u_n/6.d0-0.5d0*rou_in*u_t
        end if
        !Zou and He velocity boundary on West side
        if(pBC_velocity(k)%id==3) then
            u_n=pBC_velocity(k)%u_x
            u_t=pBC_velocity(k)%u_y
            ftemp(:)=fbar(i,j,:)
            rou_in=(2.d0*(ftemp(3)+ftemp(6)+ftemp(7))+(ftemp(0)+ftemp(2)+ftemp(4)))/(1.d0-u_n)
            fbar(i,j,1)=ftemp(3)+2.d0/3.d0*rou_in*u_n
            fbar(i,j,5)=ftemp(7)-.5d0*(ftemp(2)-ftemp(4))+rou_in*u_n/6.d0+0.5d0*rou_in*u_t
            fbar(i,j,8)=ftemp(6)+.5d0*(ftemp(2)-ftemp(4))+rou_in*u_n/6.d0-0.5d0*rou_in*u_t
        end if
        !Zou and He velocity boundary on North side
        if(pBC_velocity(k)%id==2) then
            u_n=-pBC_velocity(k)%u_y
            u_t=pBC_velocity(k)%u_x
            ftemp(:)=fbar(i,j,:)
            rou_in=(2.d0*(ftemp(2)+ftemp(5)+ftemp(6))+(ftemp(0)+ftemp(1)+ftemp(3)))/(1.d0-u_n)
            fbar(i,j,4)=ftemp(2)+2.d0/3.d0*rou_in*u_n
            fbar(i,j,7)=ftemp(5)+.5d0*(ftemp(1)-ftemp(3))+rou_in*u_n/6.d0-0.5d0*rou_in*u_t
            fbar(i,j,8)=ftemp(6)-.5d0*(ftemp(1)-ftemp(3))+rou_in*u_n/6.d0+0.5d0*rou_in*u_t
        end if
        !Zou and He velocity boundary on South side
        if(pBC_velocity(k)%id==4) then
```

```
            u_n=pBC_velocity(k)%u_y
            u_t=pBC_velocity(k)%u_x
            ftemp(:)=fbar(i,j,:)
            rou_in=(2.d0*(ftemp(4)+ftemp(7)+ftemp(8))+(ftemp(0)+ftemp(1)+ftemp(3)))/(1.d0-u_n)
            fbar(i,j,2)=ftemp(4)+2.d0/3.d0*rou_in*u_n
            fbar(i,j,5)=ftemp(7)-.5d0*(ftemp(1)-ftemp(3))+rou_in*u_n/6.d0+0.5d0*rou_in*u_t
            fbar(i,j,6)=ftemp(8)+.5d0*(ftemp(1)-ftemp(3))+rou_in*u_n/6.d0-0.5d0*rou_in*u_t
        end if
    end do
end if

end subroutine BC_treatment_vp

!//////////////////////////////////////////////////////////////////////
!BCTreatment(time)
subroutine BC_Treatment_FH
use ComPara, only: lx, ly
use Node_data_D2Q9, only: pNode
use BoundaryCondition
implicit none
integer :: y_n, x_e, y_s, x_w, y_nn, x_ee, y_ss, x_ww, i, j, k
!real*8 :: tmp, xi, u_sf, f_star, c_squ, u_f, u_squ
integer :: f1_i, minus_i, r_ff_x, r_ff_y, r_s_x, r_s_y




!write(1,*) 'BCTreatment'
!BC treatment by F_H method
!!debug
!write(*,*) '(10,1)'
!write(*,*) pNode(10,1)%node(0:8)
!write(*,*) '(10,2)'
!write(*,*) pNode(10,2)%node(0:8)
do k=1, size(pBC_FH)-1
    i=pBC_FH(k)%i
    j=pBC_FH(k)%j
    if(pNode(i,j)%NodeType==2) then

    !debug
    !write(*,*) i,j

    !fluid interface
!.........compute upper and right next neighbour nodes with regard
!          to periodic boundaries
```

```
          y_n = mod(j,ly) + 1
          y_nn= mod(y_n,ly)+1
          x_e = mod(i,lx) + 1
          x_ee= mod(x_e,lx)+1

!.........compute lower and left next neighbour nodes with regard to
!         periodic boundaries

          y_s = ly - mod(ly + 1 - j, ly)
          y_ss= ly - mod(ly+1-y_s,ly)
          x_w = lx - mod(lx + 1 - i, lx)
          x_ww= lx - mod(lx+1-x_w,lx)

      !BC treatment
      !###########################################
      !# 1 rate(1)
      !###########################################



      if(pBC_FH(k)%rate(1)>0.d0 ) then
        !###########################################
        !deal with rate(1)
        f1_i=1
        !x_ff
        r_ff_x=x_w
        r_ff_y=j
        r_s_x=x_e
        r_s_y=j
        minus_i=3
        call cal_BC_FH(i,j, 1, f1_i,r_ff_x, r_ff_y,r_s_x, r_s_y, minus_i)
      end if
      if(pBC_FH(k)%rate(3)>0.d0) then
        f1_i=3
        r_ff_x=x_e
        r_ff_y=j
        r_s_x=x_w
        r_s_y=j
        minus_i=1
        call cal_BC_FH(i,j, 3, f1_i,r_ff_x, r_ff_y,r_s_x, r_s_y, minus_i)
      end if

      !###########################################
      !# 2 rate(2)
```

```
!###########################################



if(pBC_FH(k)%rate(2)>0.d0 ) then
   !###########################################
   !deal with rate(1)
   f1_i=2
   !x_ff
   r_ff_x=i
   r_ff_y=y_s
   r_s_x=i
   r_s_y=y_n
   minus_i=4
   call cal_BC_FH(i,j, 2, f1_i,r_ff_x, r_ff_y,r_s_x, r_s_y, minus_i)
end if
if(pBC_FH(k)%rate(4)>0.d0) then
   f1_i=4
   r_ff_x=i
   r_ff_y=y_n
   r_s_x=i
   r_s_y=y_s
   minus_i=2
   call cal_BC_FH(i,j, 4, f1_i,r_ff_x, r_ff_y,r_s_x, r_s_y, minus_i)
end if

!###########################################
!# 3 rate(3)
!###########################################



if(pBC_FH(k)%rate(5)>0.d0 ) then
   !###########################################
   !deal with rate(1)
   f1_i=5
   !x_ff
   r_ff_x=x_w
   r_ff_y=y_s
   r_s_x=x_e
   r_s_y=y_n
   minus_i=7
   call cal_BC_FH(i,j, 5, f1_i,r_ff_x, r_ff_y,r_s_x, r_s_y, minus_i)
end if
if(pBC_FH(k)%rate(7)>0.d0) then
   f1_i=7
```

```
            r_ff_x=x_e
            r_ff_y=y_n
            r_s_x=x_w
            r_s_y=y_s
            minus_i=5
            call cal_BC_FH(i,j, 7, f1_i,r_ff_x, r_ff_y,r_s_x, r_s_y, minus_i)
        end if



        !############################################
        !# 4 rate(4)
        !############################################



            if(pBC_FH(k)%rate(6)>0.d0 ) then
                !############################################
                !deal with rate(1)
                f1_i=6
                !x_ff
                r_ff_x=x_e
                r_ff_y=y_s
                r_s_x=x_w
                r_s_y=y_n
                minus_i=8
                call cal_BC_FH(i,j, 6, f1_i,r_ff_x, r_ff_y,r_s_x, r_s_y, minus_i)
            end if
            if(pBC_FH(k)%rate(8)>0.d0) then
                f1_i=8
                r_ff_x=x_w
                r_ff_y=y_n
                r_s_x=x_e
                r_s_y=y_s
                minus_i=6
                call cal_BC_FH(i,j, 8, f1_i,r_ff_x, r_ff_y,r_s_x, r_s_y, minus_i)
            end if

    end if

end do
!!debugging
!write(*,*) '(10,1)'
!write(*,*) pNode(10,1)%node(0:8)
!write(*,*) '(10,2)'
!write(*,*) pNode(10,2)%node(0:8)
```

```
!write(1,*) 'BCTreatment'
!write(*,*) 'BCTreatment'
end subroutine BC_Treatment_FH
!///////////////////////////////////////////////////////////////////
!used for Loop BC treatment
subroutine cal_BC_FH(i,j, rate, f1_i,r_ff_x, r_ff_y,r_s_x, r_s_y, minus_i)
use ComPara, only:    CP_cs_sq, fun_cal_rate, omega
use Node_data_D2Q9, only: fbar, macro, D2Q9_c, D2Q9_weight

implicit none
integer :: i,j, rate, f1_i, r_ff_x, r_ff_y, r_s_x, r_s_y, minus_i
real*8    :: tmp, xi, u_sf, u_w, f_star, c_squ, u_f, u_squ
        u_w=0.d0
        c_squ = CP_cs_sq
        tmp = fun_cal_rate(i,j,rate)
        ! 0<q<0.5
        if(tmp<0.5d0 .AND. tmp>0.d0) then
            xi=omega*(2.d0*tmp-1.d0)/(1.d0-2.d0*omega)
            ! here is u_sf*c_i instead of u_sf
            ! i=1
            u_sf=(D2Q9_c(f1_i,1)*macro(r_ff_x,r_ff_y,2) +D2Q9_c(f1_i,2)*macro(r_ff_x,r_ff_y,3))
        ! 0.5<=q<=1
        else if (tmp>=0.5d0 .AND. tmp<1.d0) then
            xi=2.d0*omega*(2.d0*tmp-1.d0)/(2.d0+omega)
            u_sf=(1.d0-1.5d0/tmp)*(D2Q9_c(f1_i,1)*macro(i,j,2) &
            +D2Q9_c(f1_i,2)*macro(i,j,3))+1.5d0/tmp*u_w
        else
            write(*,*) 'check pNode rate(k)',i,j,rate,tmp
        end if
        u_f=D2Q9_c(f1_i,1)*macro(i,j,2)+D2Q9_c(f1_i,2)*macro(i,j,3)
        u_squ=(macro(i,j,2) ** 2.d0+macro(i,j,3)**2.d0)
        f_star=D2Q9_weight(f1_i)* macro(i,j,1) * (1.d0 + u_sf / c_squ &
              + u_f ** 2.d0 / (2.d0 * c_squ ** 2.d0) - u_squ / (2.d0 * c_squ))
        fbar(i,j,minus_i)=(1.d0-xi)*fbar(i,j,f1_i)+xi*f_star
!        pNode(r_s_x,r_s_y)%node(minus_i)=(1.d0-xi)*pNode(i,j)%node(f1_i)+xi*f_star

end subroutine cal_BC_FH


!///////////////////////////////////////////////////////////////////
!calculate the macroscope variables and the equilibrium functions feq
subroutine cal_macro
use ComPara, only: lx, ly, fun_is_fluid, density, CP_cs_sq, gamma
use Node_data_D2Q9, only: fbar, feq, macro, D2Q9_weight
```

```
implicit none
integer i,j,k
real*8 :: u_n(8),u_x, u_y, u_squ, c_squ, d_loc, u_max
 c_squ = CP_cs_sq
! calculate the gamma
!macro(:,:,1)=dsqrt(macro(:,:,2)*macro(:,:,2)+macro(:,:,3)*macro(:,:,3))
!u_max=MAXVAL(macro(:,:,1))
!gamma=(10.d0*dsqrt(3.d0)*u_max)**2
!write(*,*) "gamma=",gamma
do i=1,lx,1
    do j=1,ly,1
        !------------calculate the macroscopic variables
        d_loc=sum(fbar(i,j,:))
        if( d_loc>0.0d0 .and. fun_is_fluid(i,j)) then
            macro(i,j,1)=d_loc
            macro(i,j,2)= ((fbar(i,j,1)+fbar(i,j,5)+fbar(i,j,8)) &
           -(fbar(i,j,3 ) + fbar(i,j,6 ) + fbar(i,j,7 ))) / d_loc
            macro(i,j,3)=((fbar(i,j,2)+fbar(i,j,5)+fbar(i,j,6))&
           -(fbar(i,j,4 ) + fbar(i,j,7 ) + fbar(i,j,8 )))/ d_loc
       else
            macro(i,j,1)=density
            macro(i,j,2:3)=0.d0
        end if
        !------------upgrade the equilibrium functions feq
        d_loc=macro(i,j,1)
        u_x    =macro(i,j,2)
        u_y    =macro(i,j,3)
        u_n(1) =    u_x
        u_n(2) =          u_y
        u_n(3) = - u_x
        u_n(4) =        - u_y
        u_n(5) =    u_x + u_y
        u_n(6) = - u_x + u_y
        u_n(7) = - u_x - u_y
        u_n(8) =    u_x - u_y
        u_squ = u_x * u_x + u_y * u_y
        feq(i,j,0) = D2Q9_weight(0) * d_loc * (1.d0 - u_squ / (2.d0 * c_squ))
        do k=1,8
            feq(i,j,k)=D2Q9_weight(k) * d_loc * (1.d0 + u_n(k) / c_squ &
                    + u_n(k) ** 2.d0 / (2.d0 * c_squ ** 2.d0) &
                    - u_squ / (2.d0 * c_squ) )
        end do

    end do
end do


end subroutine cal_macro



!////////////////////////////////////////////////////////////////////////
!check(time)
subroutine check(time,tic,toc,ti)
use ComPara, only: lx, ly, CP_L2_err
use Node_data_D2Q9, only: macro

implicit none
integer :: time,ti
real*8    :: tic,toc
real*8 :: d_loc,t
  character*13 ::    char_time
  integer :: temp(4)

  call CPU_time(toc)
  d_loc=toc-tic
  t=d_loc/float(3600)!xxx.xxx hours left
  temp(1)=int(t)
  t=(t-float(temp(1)))*float(60)!xxx.xxx mins left
  temp(2)=int(t)
  t=(t-float(temp(2)))*float(60)
  temp(3)=int(t)
  t=(t-float(temp(3)))*float(100)
  temp(4)=int(t)
  write(char_time,'(I4.4,a,I2.2,a,I2.2,a,I2.2)')temp(1),":",temp(2),":",temp(3),":",temp(4)

d_loc=sum(macro(:,:,1))

if(mod(ti,25)==1) then
        write(* ,'(/,a,4x,a,2x,a,2x,a)')"time(sec)", "iteration","    density","rsd-density    rsd-x              rsd-y"
!          write(9    ,'(/,a,4x,a,2x,a,2x,a)')"time(sec)",    "iteration","        density","rsd-density        rsd-x
rsd-y"
end if

write(*,'(a,i8,f12.4,3e12.3)') char_time, time,    d_loc, CP_L2_err(1:3)
write(9,'(a,i8,f12.4,3e12.3)') char_time, time,    d_loc, CP_L2_err(1:3)

end subroutine check
```

```
!////////////////////////////////////////////////////////////////////
!calculate the L2 relative error
subroutine cal_L2_error
use ComPara, only: lx, ly, CP_L2_err, fun_is_fluid
use Node_data_D2Q9, only: macro, macro_temp, pNode
implicit none
integer :: i,j
real*8    :: err1(3), err2(3)
!real*8 :: d_loc,t
err1=0.d0
err2=0.d0
do i=1,lx
        do j=1,ly
                if(fun_is_fluid(i,j)) then
                        err1(:)=err1(:)+(macro(i,j,:)-macro_temp(i,j,:))**2
                        err2(:)=err2(:)+(macro(i,j,:))**2
                end if
        end do
end do
  CP_L2_err(:)=dsqrt(err1(:)/err2(:))
  macro_temp=macro

end subroutine cal_L2_error
!///////////////////////////////////////////////////////////
! function my_mod
real*8 function my_mod(a,p)
implicit none
real*8 a,p !, my_mod
my_mod=a-int(a/p)*p
end function my_mod



!///////////////////////////////////////////////////////////
! function fun_is_fluid(i,j)
! determine whether point is fluid or not



logical function fun_is_fluid(i,j)
!use ComPara
use Node_data_D2Q9, only: pNode
implicit none
integer :: i,j
!logical fun_no_fracture
if(pNode(i,j)%NodeType==0 .OR. pNode(i,j)%NodeType==2 .OR. pNode(i,j)%NodeType==6) then
```

```
  fun_is_fluid=.True.
  else
  fun_is_fluid=.False.
end if

end function fun_is_fluid

!///////////////////////////////////////////////////////////
! calculate the rate(i,j,k)
real*8 function fun_cal_rate(i,j,k)
!use ComPara
use Node_data_D2Q9, only: pNode
use BoundaryCondition, only: pBC_FH
implicit none
integer :: i,j,k


if(pNode(i,j)%NodeType==2 .OR. pNode(i,j)%NodeType==3) then
  fun_cal_rate=pBC_FH(pNode(i,j)%rate_ij)%rate(k)
  else
  fun_cal_rate=0.d0
end if

end function fun_cal_rate

!///////////////////////////////////////////////////////////
! function fun_4point_case(i,j)
! determine whether point is fluid or not


integer function fun_4point_case(i,j)
use ComPara, only: fun_cal_rate, fun_point_number
use Node_data_D2Q9, only: pNode
implicit none
integer :: i,j

        if( fun_cal_rate(i,j,5)<=0.5d0 &
          .AND. fun_cal_rate(i+1,j,6)<=0.5d0 &
          .AND. fun_cal_rate(i+1,j+1,7)<=0.5d0 &
          .AND. fun_cal_rate(i,j+1,8)<=0.5d0 ) then
          if( pNode(i,j)%NodeType==2) then
          if(fun_cal_rate(i,j,5)>0.d0) then
          fun_4point_case=1
          else
```

```
     fun_4point_case=2
     end if
     else if ( pNode(i,j)%NodeType==3) then
     if(fun_cal_rate(i,j,5)>0.d0) then
     fun_4point_case=3
     else
     fun_4point_case=4
     end if
     end if
    else
    open(1, file='.\results\fun_4point_case.dat',ACCESS ='append')
    write(1,*) "fun_4point_case",i,j
    close(1)
     write(*,*) "in function fun_4point_case: rate(5:8)>0.5d0 has been found in", i,j
    end if


end function fun_4point_case



!//////////////////////////////////////////////////////////
!function fun_distance(x1,y1,x2,y2)
! calculate the distance between (x1,y1) and (x2, y2)
!
real*8 function fun_distance(x1,y1,x2,y2)
implicit none
real*8 :: x1, x2, y1, y2
fun_distance=DSQRT((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2))


end function fun_distance
```