HIGH DENSITY RATIO MULTI-COMPONENT LATTICE BOLTZMANN FLOW MODEL FOR FLUID DYNAMICS AND CUDA PARALLEL COMPUTATION

by

Jie Bao

BS, Beijing University of Aeronautics and Astronautics, China, 2004

Submitted to the Graduate Faculty of

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2010

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Jie Bao

It was defended on

Feburary 26th, 2010

and approved by

Dr. Anne M. Robertson, Associate Professor, Department of Mechanical Engineering & Materials Science

Dr. Sung K. Cho, Associate Professor, Department of Mechanical Engineering & Materials Science

Dr. Joseph J. McCarthy, Associate Professor, Department of Chemical & Petroleum Engineering

Dissertation Director: Dr. Laura Schaefer, Associate Professor,

Department of Mechanical Engineering& Materials Science

Copyright © by Jie Bao

2010

HIGH DENSITY RATIO MULTI-COMPONENT LATTICE BOLTZMANN FLOW MODEL FOR FLUID DYNAMICS AND CUDA PARALLEL COMPUTATION

Jie Bao, PhD

University of Pittsburgh, 2010

The lattice Boltzmann equation (LBE) method is a promising technique for simulating fluid flows and modeling complex physics in fluids, and can be modified for solving general nonlinear partial differential equations (NPDEs). The LBE method has recently attracted more and more attention since it may help us to better understand the mechanisms of the complicated physical phenomena and dynamic processes modeled by NPDEs.

In this dissertation, firstly, we developed a second-order accurate mass conserving boundary condition (BC) for the LBE method. Through several cases, the results show that our mass conserving BC will not result in the constant mass leakage that occurs for the other BCs in some cases. Additionally, it increases the efficiency and stability of the method for cases that involve relatively large magnitudes of body force.

Secondly, we developed a multi-component and multi-phase LBE method for high density ratios. Multi-component multi-phase (MCMP) flow is very common in engineering or industrial problems and in nature. Because the lattice Boltzmann equation (LBE) model is based on microscopic models and mesoscopic kinetic equations, it offers many advantages for the study of multi-component or multi-phase flow problems. While the original formulation of Shan and Chen's (SC) model can incorporate some multiple phase and component scenarios, the density ratio of the different components is greatly restricted (less than approximately 2.0). This obviously limits the applications of this MCMP LBE model. Hence, based on the original SC

MCMP model and the improvements in the single-component multi-phase (SCMP) flow model reported by Yuan and Schaefer, we have developed a new model that can simulate a MCMP system with a high density ratio.

Finally, we developed a parallel computation LBE method based on Compute Unified Device Architecture (CUDA). CUDA offers a great economic alternative way to increase the calculation speed of LBE method instead of using a supercomputer. We present how to apply CUDA to the LBE method, including boundary condition treatments, single phase flow, thermal problems, and multi-phase cases. Through the results of several numerical experiments, our model with the help of CUDA can offer an improvement of a 10-30 times faster speed than that of a traditional single thread CPU code.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Roman Letters

a , $\vec{a}$	Acceleration; attraction parameter in equation of state
b	Repulsion parameter in equation of state
c	Lattice speed; lattice spacing
c_0	Constant in equation of state
c_s	Lattice sound speed
D	Dimension of space
$\vec{e}_\alpha$	Lattice velocity vector
f_α	Particle distribution function
f_α^{eq}	Equilibrium particle distribution function
$\vec{F}$	Force per unit mass
g	Gravitational constant; fluid-fluid interaction strength
g_f	Intensity of fluid-fluid interaction
g_w	Intensity of fluid-solid interaction
H	Height of the domain
L	Length of the domain
nx , ny , nz	Lattice size in the x, y and z directions
p	Pressure
Ra	Rayleigh number

Re	Reynolds number
t	Time
T	Temperature
$\vec{u}$, u	Fluid velocity
$\vec{u}^{eq}$	Equilibrium velocity
$\vec{x}$	Position

Greek Letters

δt	Time step
δx , δy	Lattice constant
λ	Relaxation time
μ	Dynamic viscosity
ν	Kinematic viscosity
$\vec{\xi}_\alpha$	Discrete velocity set
ρ	Density
ρ_w	Wall density
τ	Relaxation time
ψ	Effective mass

Abbreviations

BC	Boundary condition
BGK	Bhatnagar-Gross-Krook
CFD	Computational fluid dynamics

CUDA	Compute unified device architecture
EOS	Equation of state
FH	Filippova and Hänel
LB	Lattice Boltzmann
LBE	Lattice Boltzmann equation
LBGK	Lattice BGK
MLS	Mei, Luo, and Shyy
N-S	Navier-Stokes
NPDE	nonlinear partial differential equation
PDE	Partial differential equation
PDF	Particle distribution function
SC	Shan & Chen

ACKNOWLEDGEMENTS

During my Ph.D. study work, many people helped me in numerous ways. Let me try to thank them without, I hope, forgetting anyone.

First and foremost, I with to acknowledge my advisor, Dr. Laura Schaefer, for supporting me, encouraging me and guiding me throughout my whole Ph. D. study and research work with her brilliant advice, patience and kindness. Thank you.

My thanks are extended to the members of my thesis committee, Drs. Anne M. Robertson, Joseph J. McCarthy and Sung K. Cho. Thank you for your time and valuable advice. I am very thankful to the faculty and staff members in the Department of Mechanical Engineering, who have made my studies here valuable and enjoyable.

Special thanks go to Dr. Peng Yuan for the cooperation and helpful suggestions and discussions.

Thanks also go to Zijing Zeng for the useful suggestions and discussions.

I feel lucky to have made a lot of new friends here. My life becomes happy in Pitt due to the following people: Lifeng Qin, Di Xu, Yunjun Zhao, Chunhua Fu, Dr. Hongbin Cheng, Dr. Qingming Chen, Tony Kerzmann, Veronica Miller, Michael Ikeda, Dr. Florian Zink.

I reserve my sincere thanks for my family members. I am deeply indebted to my father: Qixiong Bao, and my mother Yufang Ding for their support and endless love through my whole life and the long journey of study. Words cannot express the gratitude I have to what they have

done for me in the past. I am also thankful for the love and encouragements provided by my wife's parents.

Finally, I would like to thank my wife Kun Li for support and faith in me. Thank you for sharing with me all the exciting and difficult times of living and studying in a foreign country.

1.0 INTRODUCTION

The lattice Boltzmann equation (LBE) method is a promising technique for simulating fluid flows and modeling complex physics in fluids. Unlike conventional computational fluid dynamics (CFD) methods, the LBE model is based on microscopic models and mesoscopic kinetic equations in which the collective behavior of the particles in a system is used to simulate the continuum mechanics of the system. Due to this kinetic nature, the LBE method has been found to be particularly useful in applications involving interfacial dynamics and complex boundaries, such as multi-phase or multi-component flows [1]. Besides that, the LBE model is straightforward to program and intrinsically parallel [2, 3, 4].

Additionally, although the LBE method can be considered to be a simplified fictitious molecular dynamics model designed for solving fluid problems, it also demonstrates the potential to simulate a nonlinear system in other areas. Some early work shows that the LBE method has been extended successfully to simulate some evolution equations [5, 6, 7, 8, 9, 10], and recent research shows that the LBE method can be used to solve more generalized nonlinear partial differential equations (NPDEs). Chai, Shi and Zheng's work proves that the LBE model can recover 6^{th}-order NPDEs [11]. NPDEs play an important role in different fields of physics and mathematics [12, 13], and the LBE method has therefore attracted more and more attention since it may help us to better understand the mechanisms of the complicated physical phenomena and dynamic processes modeled by NPDEs.

1.1 INTRODUCTION TO THE LATTICE BOLTZMANN EQUATION METHOD

The LBE model is derived from the continuum Boltzmann equation, which is an integro-differential equation, and describes the evolution of a single-particle distribution function $f\left(\vec{x},\vec{\xi},t\right)$ in the physical-momentum space. Because of the high dimensions of the distribution and the complexity in the collision integral, direct solution of the full Boltzmann equation is a formidable task for both analytical and numerical techniques [14]. In 1954, Bhatnagar, Gross and Krook developed the Boltzmann-BGK equation which is an important simplification of the original Boltzmann equation [15]. The Boltzmann-BGK equation takes the form:

$$\frac{\partial f}{\partial t}+\xi\cdot\nabla f=-\frac{1}{\lambda}\left[f-f^{0}\right] \tag{1.1}$$

For solving f numerically, the Boltzmann-BGK equation is first discretized in the momentum space using a finite set of velocities $\vec{\xi}_{\alpha}$:

$$\frac{\partial f_{\alpha}}{\partial t}+\xi_{\alpha}\cdot\nabla f_{\alpha}=-\frac{1}{\lambda}\left[f_{\alpha}-f_{\alpha}^{(eq)}\right] \tag{1.2}$$

where $f_{\alpha}\left(\vec{x},t\right)\equiv f\left(\vec{x},\xi_{\alpha},t\right)$ and $f_{\alpha}^{(eq)}\left(\vec{x},t\right)\equiv f^{(0)}\left(\vec{x},\xi_{\alpha},t\right)$ are the distribution function and the equilibrium distribution function of the α th discrete velocity ξ_{α}, respectively. The equilibrium distribution function can be expressed in the form:

$$f_{\alpha}^{eq}=\rho w_{\alpha}\left[1+\frac{3}{c^{2}}\vec{e}_{\alpha}\cdot\vec{u}+\frac{9}{2c^{4}}\left(\vec{e}_{\alpha}\cdot\vec{u}\right)^{2}-\frac{3}{2c^{2}}\vec{u}\cdot\vec{u}\right] \tag{1.3}$$

where w_{α} is the weighting factor [16], $c=\frac{\delta x}{\delta t}$ is the lattice speed, $\vec{e}_{\alpha}$ is the discrete velocity set, and $\vec{u}$ and ρ are the macroscopic velocity and density.

The weighting factors and discrete velocities for the D2Q9 model are, for example:

$$\vec{e}_\alpha = \begin{cases} (0,0), & \alpha = 0; \\ (\pm 1,0)c, (0,\pm 1)c, & \alpha = 1,2,3,4; \\ (\pm 1,\pm 1)c, & \alpha = 5,6,7,8. \end{cases} \tag{1.4}$$

$$w_\alpha = \begin{cases} 4/9, & \alpha = 0; \\ 1/9, & \alpha = 1,2,3,4; \\ 1/36, & \alpha = 5,6,7,8. \end{cases} \tag{1.5}$$

For a 3-D LBE model, the weighting factors and discrete velocities for D3Q19 (a widely used state space) are:

$$\vec{e}_\alpha = \begin{cases} (0,0,0), & \alpha = 0; \\ (\pm 1,0,0)c, (0,\pm 1,0)c, (0,0,\pm 1)c & \alpha = 1,2,...,6; \\ (\pm 1,\pm 1,0)c, (\pm 1,0 \pm 1)c, (0,\pm 1,\pm 1)c, & \alpha = 7,8,...,18. \end{cases} \tag{1.6}$$

$$w_\alpha = \begin{cases} 1/3, & \alpha = 0; \\ 1/18, & \alpha = 1,2,...,6; \\ 1/36, & \alpha = 7,8,...,18. \end{cases} \tag{1.7}$$

After discretizing the PDF in momentum space, the number of possible particle spatial positions and microscopic momenta are reduced to 9 for the 2-D problem, as shown in Figure 1.1. For 3-D flow, there are several cubic lattice models, such as the D3Q15, D3Q19 and D3Q27 model, as shown in Figure 1.2.

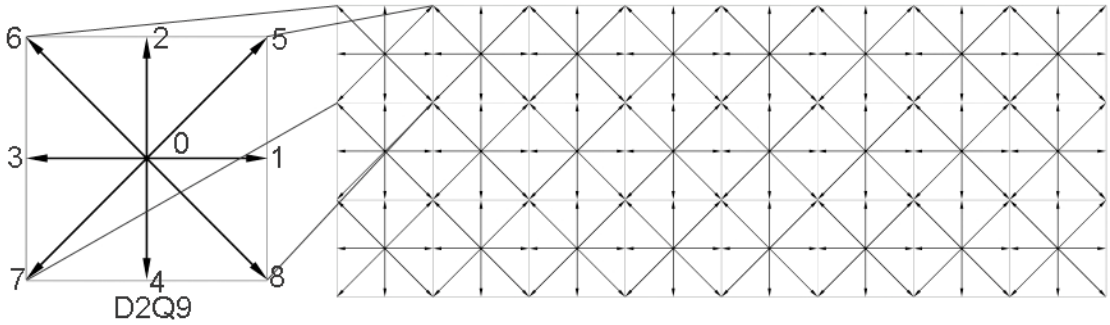


Figure 1.1: The 2D grids for LBM

Based on the D2Q9, D3Q15, D3Q19 and D3Q27 frame definition, the density and the velocity can be defined as:

$$\rho = \sum_{\alpha=0}^{b} f_\alpha \tag{1.8}$$

$$\vec{u} = \frac{1}{\rho}\sum_{\alpha=0}^{b} f_\alpha \vec{e}_\alpha \tag{1.9}$$

where b represents the total number of possible particle spatial positions. To solve $f_\alpha(\vec{x},t)$, equation (1.2) needs to be further discretized in physical space $\vec{x}$ and time t, so the completely discretized form of Boltzmann-BGK equation is:

$$f_\alpha(\vec{x}+\vec{e}_\alpha\Delta t, t+\Delta t) = f_\alpha(\vec{x},t) - \frac{1}{\tau}\left[f_\alpha(\vec{x},t) - f_\alpha^{eq}(\vec{x},t)\right] \tag{1.10}$$

where τ is the non-dimensional relaxation time. Appendix C shows the steps from continuum Boltzmann equation to LBE model in detail. This equation often can be solved using the following two steps [17]:

Collision: $$\widetilde{f}_\alpha(\vec{x},t) = f_\alpha(\vec{x},t) - \frac{1}{\tau}\left[f_\alpha(\vec{x},t) - f_\alpha^{eq}(\vec{x},t)\right] \tag{1.11}$$

Streaming: $$f_\alpha(\vec{x}+\vec{e}_\alpha\Delta t, t+\Delta t) = \widetilde{f}_\alpha(\vec{x},t) \tag{1.12}$$

After the streaming step (equation (1.12)), we can substitute the newly calculated PDF $f_\alpha(\vec{x}+\vec{e}_\alpha\Delta t, t+\Delta t)$ into equation (1.11) by replacing the old $f_\alpha(\vec{x},t)$. Every loop is a time step, and after many time steps, which may be over 10,000 for some particularly complicated phenomena, the program will converge.

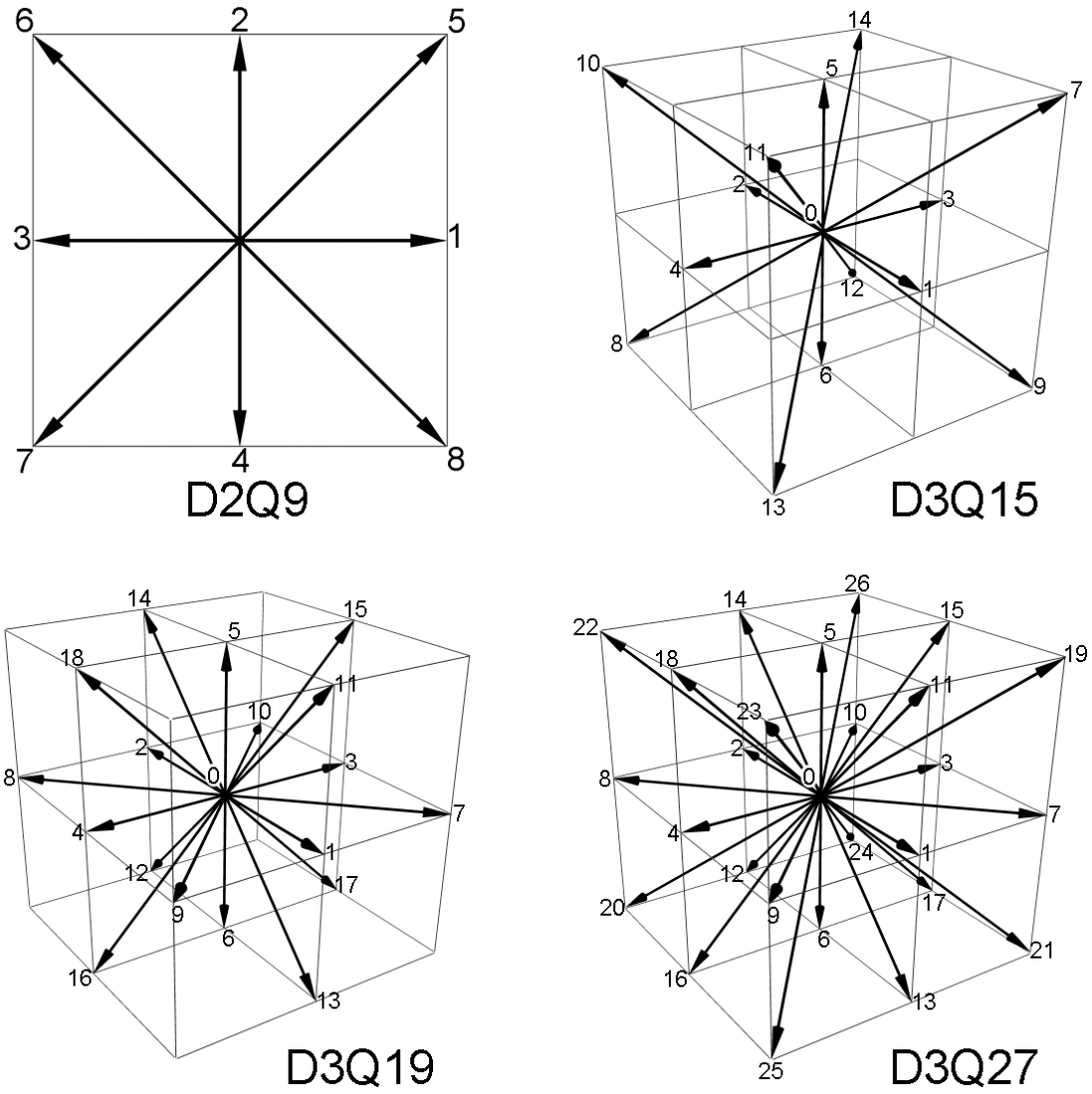


Figure 1.2: 2D square lattice and 3D cubic lattice for LBM

1.2 INTRODUCTION TO THE MULTI-PHASE LBE METHOD

It is commonly accepted that the separation of different phases or components is microscopically due to the long-range interaction between the molecules of a fluid [18]. This interaction can be expressed as:

$$\vec{F}(\vec{x}) \cong -c_0 \psi(\vec{x}) g \nabla \psi(\vec{x}) \tag{1.13}$$

where c_0 is a constant depending on the lattice structure. For the D2Q9 and D3Q19 lattices, $c_0 = 6.0$, and for the D3Q15 lattice, $c_0 = 10.0$. The coefficient for the strength of the interparticle force is g, with $g < 0$ representing an attractive force between particles and $g > 0$ a repulsive force. $\psi(\vec{x})$ is the effective mass, which is a function of local density and can be varied to reflect different fluid and fluid mixture behaviors, as represented by various equations of state (EOS). This equation is derived from the original Shan and Chen (SC) model. Although that work only used the interparticle forces of nearest neighbor sites, it can be extended to include other neighboring sites as long as the gradient term $\nabla \psi$ is properly specified. We use both the nearest and next-nearest sites to evaluate this gradient term, which gives a six-point scheme for two dimensions:

$$\frac{\partial \psi(i,j)}{\partial x} = c_1[\psi(i+1,j) - \psi(i-1,j)] + c_2[\psi(i+1,j+1) - \psi(i-1,j+1) + \psi(i+1,j-1) - \psi(i-1,j-1)] \tag{1.14a}$$

$$\frac{\partial \psi(i,j)}{\partial y} = c_1[\psi(i,j+1) - \psi(i,j-1)] + c_2[\psi(i+1,j+1) - \psi(i+1,j-1) + \psi(i-1,j-1) - \psi(i-1,j-1)] \tag{1.14b}$$

where c_1 and c_2 are the weighting coefficients for the nearest and next nearest sites, respectively.

In addition to the interparticle forces, if the problem includes a solid wall boundary, the interaction between the fluid and solid interface needs to be considered, so the forces applied on a particle that contacts the solid wall are:

$$\vec{F}_w(\vec{x}) = -\rho\ (\vec{x})\sum_{x'} G_w(\vec{x},\vec{x}')\rho_w(\vec{x}')(\vec{x}'-\vec{x}) \tag{1.15}$$

where $G_w(\vec{x},\vec{x}')$ reflects the intensity of the fluid-solid interaction, and $\rho_w(\vec{x}')$ is the wall density, which equals one at the wall and zero in the fluid. Furthermore, in addition to interparticle and wall forces, the body force can be defined as:

$$\vec{F}_b(\vec{x}) = \rho(\vec{x})\vec{a} \tag{1.16}$$

The viscosity and the surface tension are two additional important factors for specifying fluid characteristics. The viscosity is defined in the LBE model as:

$$\upsilon = \left(\tau - \frac{1}{2}\right)c_s^2\delta t \tag{1.17}$$

where c_s is the speed of sound in the LBE model. Hence, the viscosity can be changed by choosing a different relaxation time τ. In order to adjust the surface tension, an additional force term should be introduced into the fluid-fluid interaction, and is defined as:

$$\vec{F}_s = \kappa\psi(\rho)\nabla\nabla^2\psi(\rho) \tag{1.18}$$

where κ determines the strength of the surface tension [2].

Hence, the total force on each particle can be expressed as:

$$\vec{F}_{total} = \vec{F}_i + \vec{F}_w + \vec{F}_b + \vec{F}_s \cdots\cdots \tag{1.19}$$

All of these forces can be incorporated into the model by shifting the velocity in the equilibrium distribution. This means that the velocity $\vec{u}$ in equation (1.3) is replaced by

$$\vec{u}^{\,eq} = \vec{u}_i + \frac{\tau_i \vec{F}_{total,i}}{\rho_i(\vec{x})} \tag{1.20}$$

The effective mass ψ, which was mentioned in the introduction to calculating interparticle forces and surface tension, is the mechanism for incorporating a more sophisticated EOS. As stated previously, the effective mass $\psi(\vec{x}) = \psi(\rho(\vec{x}))$ is a function of the local density, and can be defined as:

$$\psi(\rho) = \sqrt{\frac{2\left(p - c_s^2 \rho\right)}{c_0 g}} \tag{1.21}$$

where p is the pressure. The choice of EOS can reflect the relationship between the pressure, temperature and density. In Yuan and Schaefer's work [19], five different EOS were tested in this model, and it was found that that Peng-Robinson (P-R) EOS provided the maximum increase in the density ratio of SCMP flows while maintaining small spurious currents around the interface. Hence, we used the P-R EOS in our following multi-phase flow research, where the P-R EOS is expressed as:

$$P = \frac{\rho RT}{1 - b\rho} - \frac{a\alpha(T)\rho^2}{1 + 2b\rho - b^2\rho^2} \tag{1.22}$$

$$\alpha(T) = \left[1 + \left(0.37464 + 1.5422\omega - 0.26992\omega^2\right)\left(1 - \sqrt{T/T_c}\right)\right]^2 \tag{1.23}$$

with $a = \frac{0.45724 R^2 T_c^2}{P_c}$ and $b = \frac{0.0778 RT_c}{P_c}$, where a is the attraction parameter, b is the volumetric or repulsion parameter, and ω is the acentric factor. T_c and P_c are the critical temperature and critical pressure, respectively.

1.3 INTRODUCTION TO THE THERMAL LBE METHOD

As stated previously, the LBE method can be used to solve generalized NPDEs. Hence, the LBE model can also be applied to solving thermal problems. In a single-phase thermal fluid system, if the viscous and compressive heating effects are negligible, the temperature field satisfies a much simplified passive-scalar equation:

$$\frac{\partial T}{\partial t} + \vec{u} \cdot \nabla T = \nabla \cdot (\alpha \nabla T) + \psi \tag{1.24}$$

where $\vec{u}$ is the macroscale velocity, α is the thermal diffusivity, and ψ is the source term. To solve equation (1.24) using the LBE method, we firstly define a particle distribution function (PDF) $f_\alpha^T(\vec{x},t)$ for temperature, which is the same as the dynamic PDF. The temperature then can be found using the following relationship:

$$T = \sum_{\alpha=0}^{b} f_\alpha^T \tag{1.25}$$

Through equations (1.26) and (1.27) below, we can solve the temperature PDF $f_\alpha^T(\vec{x},t)$ numerically:

$$f_\alpha^T(\vec{x} + \vec{e}_\alpha \Delta t, t + \Delta t) = f_\alpha^T(\vec{x},t) - \frac{1}{\tau_T}\left[f_\alpha^T(\vec{x},t) - f_\alpha^{T(eq)}(\vec{x},t)\right] \tag{1.26}$$

$$f_\alpha^{T(eq)} = Tw_\alpha\left[1 + \frac{3}{c^2}\vec{e}_\alpha \cdot \vec{u} + \frac{9}{2c^4}(\vec{e}_\alpha \cdot \vec{u})^2 - \frac{3}{2c^2}\vec{u} \cdot \vec{u}\right] \tag{1.27}$$

where τ_T is the dimensionless single relaxation time for temperature. The temperature variance results in a buoyancy force, which is expressed as:

$$\rho\vec{G} = \rho\beta g(T - T_0)\vec{j} \tag{1.28}$$

To incorporate the buoyancy force, we substitute equation (1.28) into equation (1.19) ($\vec{F}_{total} = \vec{F}_i + \vec{F}_w + \vec{F}_b + \vec{F}_s \cdots\cdots$), calculate the total force that is applied on the particles, and then calculate the shifted velocity by equation (1.20) ($\vec{u}^{\,eq} = \vec{u}_i + \frac{\tau_i \vec{F}_{total,i}}{\rho_i(\vec{x})}$). Finally, the velocity $\vec{u}$

in equation (1.27) is replaced by this shifted velocity. Through including the buoyancy force, the LBE method can be used for a number of systems with thermal effects, such as natural convection problems.

2.0 INTRODUCTION TO BOUNDARY CONDITIONS FOR THE LBE METHOD

Boundary conditions (BCs) play a very important role in numerical simulation. A BC not only affects the accuracy and stability of a computational method, but it is also one of the characteristics that determine the adaptability of a CFD method. In this section, several BCs are discussed that include most of the necessary treatments for dealing with popular practical situations, such as the solid wall BC and the open BC.

2.1 BOUNCE BACK SOLID WALL BOUNDARY CONDITIONS

In LBE simulations, to some extent, developing accurate and efficient BCs is as important as developing an accurate computation scheme itself, since they will influence the stability of the computation. The most common and simplest solid wall BC is the bounce-back boundary condition. In this BC, when a particle distribution streams to a wall node, it scatters back to the fluid node along its incoming link. However, the bounce-back BC only gives first order numerical accuracy. To improve it, many BCs have been proposed in the past [20, 21, 22, 23], such as the halfway bounce-back scheme [24, 25], which is easy to implement and gives second-order accuracy for a straight wall. In this scheme, the wall is placed halfway between a fluid node and a bounce-back node. The order of accuracy for a boundary condition in the LBE model can be tested by calculating the slope of the L_2-norm error, which is defined as:

$$E_2 = \frac{\left\{\int_0^H \left[u_{LBE}(y) - u_{exact}(y)\right]^2 dy\right\}^{1/2}}{\left[\int_0^H u_{exact}^2(y) dy\right]^{1/2}} \tag{2.1}$$

where u_{LBE} is the LBE solution of the velocity.

Compared with other second order boundary treatments, the halfway bounce-back does not require any extrapolation, and is therefore easy to implement. Figure 2.1 shows the application procedure of both the fullway and halfway bounce-back BC. For the halfway bounce-back BC, in only one time step, a fluid particle goes to the boundary site, reverses its velocity and comes back, while the fullway bounce-back condition needs two time steps to go forth and back. From the foundation of these simple BCs treatments, the LBE method can be used to address many complicated geometries, which is one of the strengths of the LBE model compared to traditional simulation methods.

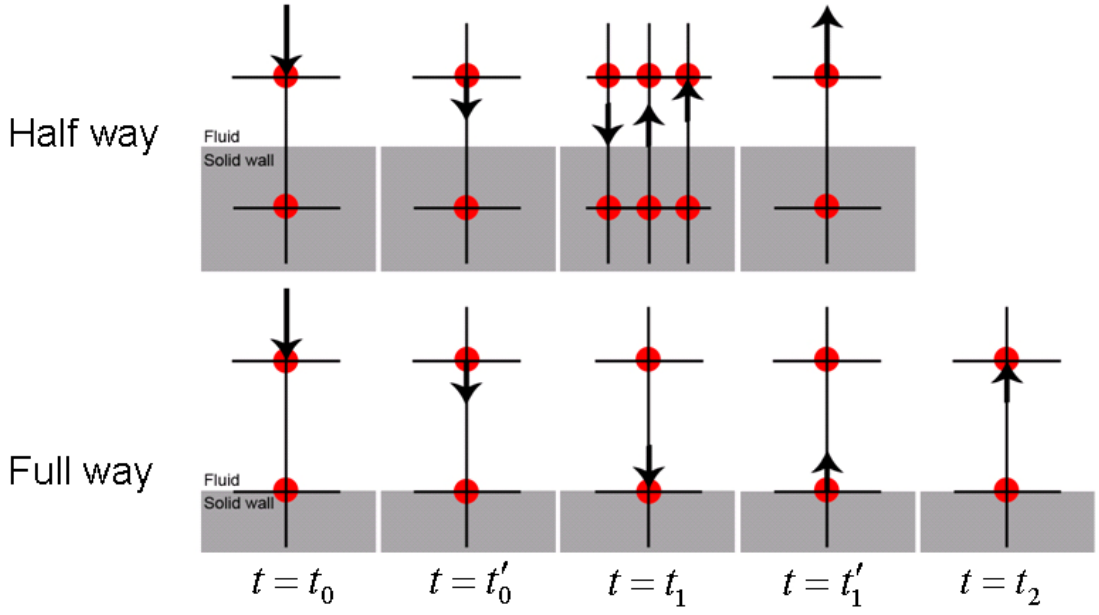
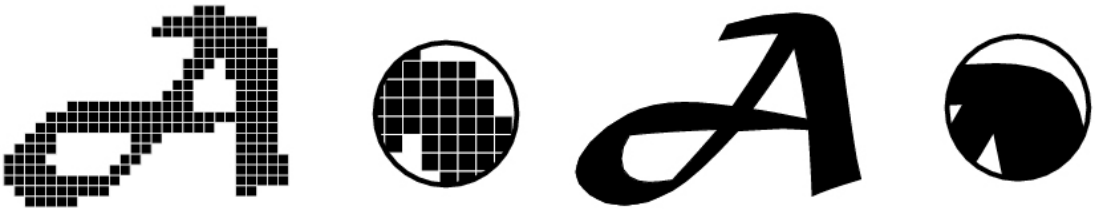


Figure 2.1: Demonstration of half way and full way bounce back boundary condition

Figure 2.2a shows an example of LBE grids for an object with a complicated geometry. From the enlargement, we can see that LBE meshes use squares to match the curved boundary. Hence, the higher the resolution (which means more grids), the better the matching between the simulation model and the real object that we are studying (such as in Figure 2.2b). This is similar to a bitmap or pixmap in computer graphics, which is a type of memory organization for an image file format that used to store digital images. Hence, for a 2D problem, for generating the grids for LBE simulation, we can firstly generate a black and white bitmap file (BMP) to describe the whole simulation domain, where a black pixel represents a solid object point and a white one a fluid space (or vice versa). Next, this image file is converted to a 2D matrix that only includes the digits 1 and 0, where 1 is black and 0 is white. The command "imread('C:\ image.bmp')" in MATLAB can easily deal with this conversion. Finally, this matrix file is defined as a solid boundary wall file in the LBE model program. By following this method, most 2D complicated geometries can be imported into the LBE simulation program.

For a 3D problem, the 3D object can be modeled in any CAD software, and then exported into an Initial Graphics Exchange Specification (IGES) file, which records the coordinates of solid object points in a 3D space. Each solid object point's coordinate is read and recorded as 1, which is then put it in to the corresponding position in a 3D matrix. Finally, as in the 2D problem, this matrix file is defined as a solid boundary wall file in the LBE model program.



a b

Figure 2.2: Demonstration of 2-D LBE grids, (a) low resolution, (b) high resolution

2.2 CURVED SOLID WALL BOUNDARY CONDITIONS

For a curved wall, the bounce back boundary condition may result in obvious jagged boundaries when the grid's resolution is not high enough, and therefore additional errors will be introduced. Hence, for dealing with a curved solid wall without requiring high mesh resolution, Filipova and Hänel (FH) proposed a curved wall BC [26], which later was improved by Mei, Luo and Shyy (MLS) [27, 28]. Bao, Yuan and Schaefer further refined this technique to create a mass conserving boundary condition [29] which solves the mass leakage problem of the original BC when a strong body force is present in a system.

As shown in Figure 2.3, $\vec{e}_\alpha$ and $\vec{e}_{\bar{\alpha}}$ denote directions opposite to each other, $\vec{x}_b$ is a boundary node, and $\vec{x}_f$ is a fluid node. The curved wall is located between a boundary node and fluid node, with $\Delta = \dfrac{\left|\vec{x}_f - \vec{x}_w\right|}{\left|\vec{x}_f - \vec{x}_b\right|}$ denoting the fraction of an intersected link in the fluid region. Obviously, $0 \le \Delta \le 1$. In order to finish the streaming step, we need to know $\tilde{f}_{\bar{\alpha}}(\vec{x}_b, t)$ at

boundary node $\vec{x}_b$, where $\widetilde{f}_{\bar{\alpha}}$ denotes the post-collision state of the distribution function. FH proposed the following treatment for $\widetilde{f}_{\bar{\alpha}}(\vec{x}_b,t)$ on curved boundaries:

$$\widetilde{f}_{\bar{\alpha}}(\vec{x}_b,t) = (1-\chi)\widetilde{f}_{\alpha}(\vec{x}_f,t) + \chi f_{\alpha}^{(*)}(\vec{x}_b,t) + 2w_{\alpha}\rho\frac{3}{c^2}\vec{e}_{\bar{\alpha}}\cdot\vec{u}_w \tag{2.2}$$

where $\vec{u}_w \equiv \vec{u}(\vec{x}_w,t)$ is the velocity at the wall, χ is the weighting factor that controls the linear interpolation between $\widetilde{f}_{\alpha}(\vec{x}_f,t)$ and $f_{\alpha}^{(*)}(\vec{x}_b,t)$, and $f_{\alpha}^{(*)}(\vec{x}_b,t)$ is given by a fictitious equilibrium distribution:

$$f_{\alpha}^{(*)}(\vec{x}_b,t) = w_{\alpha}\rho(\vec{x}_w,t)\left[1+\frac{3}{c^2}\vec{e}_{\alpha}\cdot\vec{u}_{bf}+\frac{9}{2c^4}(\vec{e}_{\alpha}\cdot\vec{u}_f)^2-\frac{3}{2c^2}\vec{u}_f\cdot\vec{u}_f\right] \tag{2.3}$$

where $\rho(\vec{x}_w,t)$ is called the wall density. In equation (2.3), $\vec{u}_f \equiv \vec{u}(\vec{x}_f,t)$ is the fluid velocity near the wall, $\vec{u}_{bf}$ is to be chosen, and the weighting factor χ depends on $\vec{u}_{bf}$.

$$\vec{u}_{bf} = [1-3/(2\Delta)]\vec{u}_f + 3/(2\Delta)\vec{u}_w \text{ and } \chi = (2\Delta-1)/(\tau+1/2) \text{ for } \Delta \geq 1/2 \tag{2.4}$$

$$\vec{u}_{bf} = \vec{u}_{ff} = \vec{u}_f(\vec{x}_f+\vec{e}_{\bar{\alpha}}\delta t,t) \text{ and } \chi = (2\Delta-1)/(\tau-2) \quad \text{for } \Delta < 1/2 \tag{2.5}$$

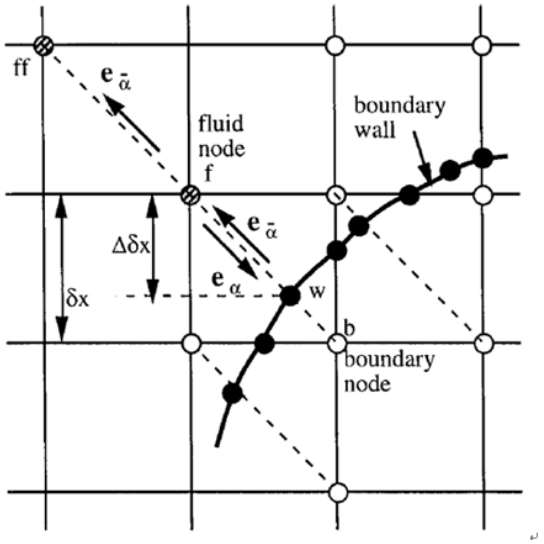


Figure 2.3: Layout of the lattice and curved wall boundary

It must be determined what $\rho(\vec{x}_w, t)$ term guarantees mass conservation. We discuss this in the context of the D2Q9 model. Shown in Figure 2.4 are the known and unknown PDFs of a flat boundary site at the lower wall boundary after the streaming step. The outgoing PDFs are f_4, f_7, f_8, which are known, and the incoming PDFs are f_2, f_5, f_6, which are unknown. Mass conservation requires that $f_2 + f_5 + f_6 = f_4 + f_7 + f_8$. It is assumed that all of the outgoing PDFs also satisfy equation (2.3), with an unknown $\rho(\vec{x}_w, t)$ term. Then summing these together, we find:

$$f_4 + f_7 + f_8 = \frac{1}{6}\rho(\vec{x}_w, t)[1 - 3u_{bf}^y + 3(u_f^y)^2] \tag{2.6}$$

where u_{bf}^y is the *y*-component of $\vec{u}_{bf}$ and u_f^y is the *y*-component of $\vec{u}_f$. Therefore $\rho(\vec{x}_w, t)$ will be:

$$\rho(\vec{x}_w, t) = 6\frac{f_4 + f_7 + f_8}{1 - 3u_{bf}^y + 3(u_f^y)^2} \tag{2.7}$$

By substituting the expression of $\rho(\vec{x}_w, t)$ into equation (2.3), the unknown PDFs f_2, f_5, f_6 can be obtained. It is straightforward to show that this boundary treatment indeed satisfies the condition $\sum_{outgoing} f = \sum_{incoming} f$, so the total mass is conserved. The simulation results that will be shown in Chapter 3 demonstrate the improved performance of our BC.

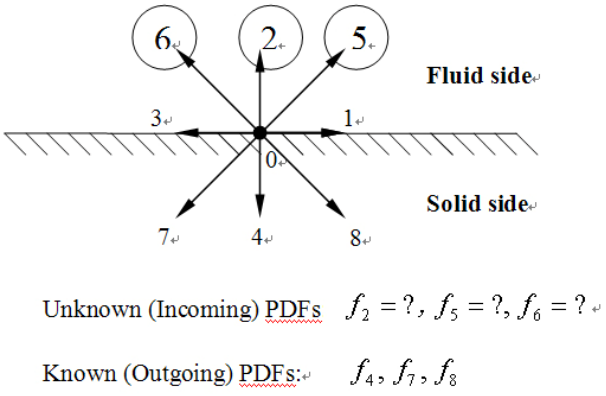


Figure 2.4: PDFs of a flat boundary site at the lower wall boundary after the streaming step

2.3 THE OPEN BOUNDARY CONDITION

Open boundaries generally include inlets/outlets, periodical boundaries, lines of symmetry and infinity. The most commonly used open boundary conditions are introduced in this section.

2.3.1 Periodical Boundary Condition

The periodical BC is the most basic open BC. The periodical BC can be applied directly to the PDFs, and not to the macroscopic flow variables, which means the PDFs coming out of one boundary will enter into the opposite boundary.

The periodical BC can be used as an inflow/outflow BC in the streamwise direction. For example, with periodical BCs at the inlet and outlet, the uniform body force or constant pressure

gradient can be included in the simulation procedure after the collision step, which is expressed as follows:

$$\widetilde{f}_{\alpha\_inlet} = \widetilde{f}_{\alpha\_outlet} - w_\alpha \frac{3}{c^2}\frac{dp}{dx} e_\alpha \cdot \vec{x} \tag{2.8}$$

where $\frac{dp}{dx}$ is the constant pressure gradient, $\vec{x}$ is the unit vector in the x (streamwise) direction, α denotes the direction of the unknown PDF, and ~ denotes the post-collision state here and hereafter.

2.3.2 Extrapolation Boundary Condition

Besides the periodical BC, we can use the zero derivative condition for an inflow/outflow boundary. Supposing $i = 1$ is the inlet boundary and $i = nx$ is the outlet boundary, in the 2-D case, the zero derivative condition can be expressed in the following form:

$$\widetilde{f}_\alpha(i=1, j) = \widetilde{f}_\alpha(i=2, j) \tag{2.9}$$

$$\widetilde{f}_\alpha(i=nx, j) = \widetilde{f}_\alpha(i=nx-1, j) \tag{2.10}$$

We also can use extrapolation to find PDFs at the inflow/outflow boundary; e.g., instead of using the periodical treatment, the following simple extrapolation can be used:

$$\widetilde{f}_{\alpha}(i=1,j)=2\widetilde{f}_{\alpha}(i=2,j)-\widetilde{f}_{\alpha}(i=3,j) \tag{2.11}$$

$$\widetilde{f}_{\alpha}(i=nx,j)=2\widetilde{f}_{\alpha}(i=nx-1,j)-\widetilde{f}_{\alpha}(i=nx-2,j) \tag{2.12}$$

2.3.3 Inlet Boundary Condition

The PDFs at the inlet can be obtained by applying bounce-back method [30, 31], from which the specified velocity or pressure can be recovered. In these approaches, usually the inlet boundary is placed half way between the inlet boundary node and the first fluid node, as shown in Figure 2.5. If the velocity profile is known at the inlet, the standard bounce-back scheme for unknown PDFs at the inlet is:

$$\widetilde{f}_{\bar{\alpha}\_inlet}=\widetilde{f}_{\alpha}+2w_{\alpha}\rho\frac{3}{c^{2}}e_{\bar{\alpha}}\cdot u_{inlet} \tag{2.13}$$

where w_{α} is the weighting factor; and e_{α} and $e_{\tilde{\alpha}}$ denote directions opposite to each other.

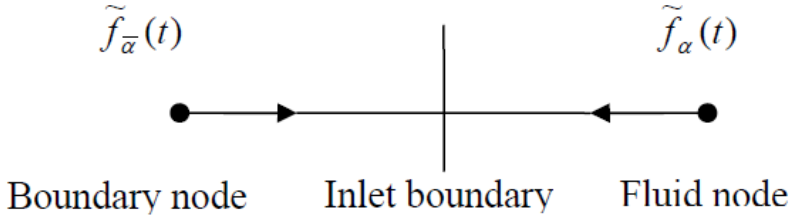


Figure 2.5: Layout of the inlet boundary

In some cases, the inlet is not placed in the middle of two nodes as shown in Figure 2.6, where the exact position of inlet is recorded by $\Delta\delta x$. Yu [32] proposed the boundary treatment for these cases. In this approach, the unknown PDFs at the inlet were decomposed to an equilibrium part and non-equilibrium part and then computed separately, i.e.

$\widetilde{f}_{\bar{\alpha}\_inlet} = \widetilde{f}^{(eq)}_{\bar{\alpha}\_inlet} + \widetilde{f}^{(neq)}_{\bar{\alpha}\_inlet}$. An example is shown in Figure 2.6. By using linear interpolation and setting $\widetilde{f}^{(neq)}_{3,I} = \widetilde{f}^{(neq)}_{3,B}$ the unknown PDF can be obtained:

$$f^{(eq)}_{1,B} = f^{(eq)}_{1,I} + \frac{\Delta}{1+\Delta}\left[f^{(eq)}_{1,C} - f^{(eq)}_{1,I}\right] \tag{2.14}$$

$$f^{(neq)}_{1,B} = f^{(neq)}_{1,I} + \frac{\Delta}{1+\Delta}\left[f^{(neq)}_{1,C} - f^{(neq)}_{1,I}\right] \tag{2.15}$$

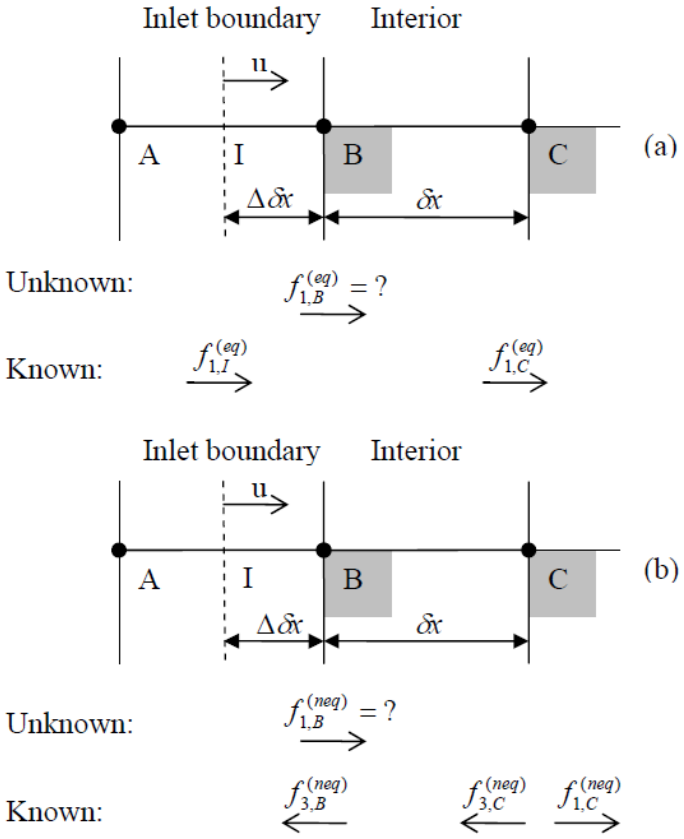


Figure 2.6: Configuration of PDFs used to construct the inlet boundary condition: (a) Configuration of equilibrium PDFs at the inlet, (b) Configuration of non-equilibrium PDFs at the inlet (the gray color denotes a fluid node)

2.3.4 Thermal Boundary Condition

Two different thermal boundary conditions are commonly encountered in temperature-dependent flows:

i) Isothermal wall: Suppose the temperature is fixed as T_w at the bottom wall. In the D2Q9 context as shown in Figure 2.7a, after streaming, f_2, f_5, and f_6 are unknowns. These unknown PDFs can be assumed to equal their equilibrium distribution, with ρ replaced by some unknown temperature T'. Summing these three PDFs together, we have:

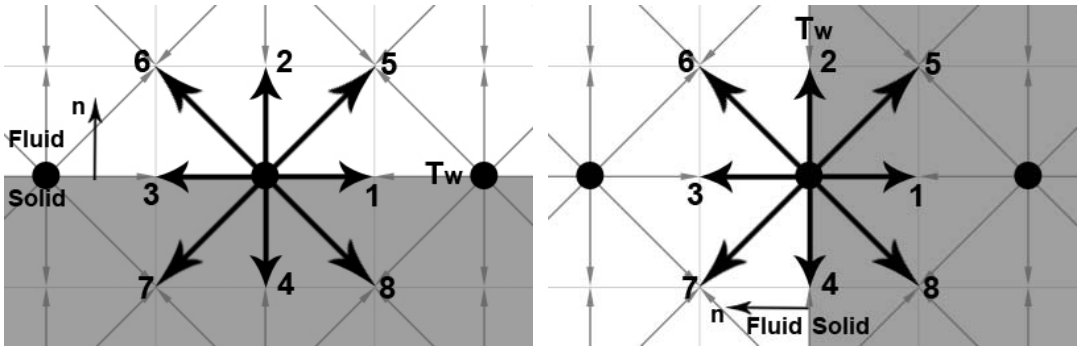
$$f_2 + f_5 + f_6 = \frac{1}{6}T'\left(1 + 3u_y + 3u_y^2\right) \tag{2.16}$$

where u_y is the velocity normal to the wall. Meanwhile, for the isothermal wall, $\sum_{\alpha=0}^{8} f = T_w$. Substituting equation (2.16) into this, T' can be calculated as:

$$T' = \frac{6}{1 + 3u_y + 3u_y^2}\left(T_w - f_0 - f_1 - f_3 - f_4 - f_7 - f_8\right) \tag{2.17}$$

Finally, f_2, f_5, and f_6 can be obtained by substituting T' into equation (1.3) ($f_\alpha = T' w_\alpha \left[1 + \frac{3}{c^2}\vec{e}_\alpha \cdot \vec{u} + \frac{9}{2c^4}\left(\vec{e}_\alpha \cdot \vec{u}\right)^2 - \frac{3}{2c^2}\vec{u} \cdot \vec{u}\right]$). For another case shown in Figure 2.7b, the solid wall is the right wall, and following the method outlined above, we can find that:

$$T' = \frac{6}{1 - 3u_x + 3u_x^2}\left(T_w - f_0 - f_1 - f_2 - f_4 - f_5 - f_8\right) \tag{2.18}$$



a b

Figure 2.7: Sketch of thermal boundary condition for D2Q9

From these two simple 2-D cases, it is shown that the normal direction of the wall (shown as arrow *n* in Figure 2.7) is required to derive the equation used for calculation of T'. For a different direction, the resulting equation is also different. For the 2-D case, the Q9 discretization requires 32 different equations for the possible normal directions of the solid wall. For the 3-D case, the Q15, Q19, and Q27 discretizations need 98, 162, and 338 different boundary equations, respectively. If objects with complicated geometry are involved in the simulation problems, such as porous media problems, it would greatly reduce the efficiency of the code, especially for the parallel method. To overcome this, based on the general formulation of this method, we have developed a common equation:

$$T' = \frac{T_w - \sum_{\alpha} f_\alpha T_\alpha}{\sum_{\alpha} f_\alpha^T \left(1 - T_\alpha\right)} \tag{2.19}$$

where $f_\alpha^T = w_\alpha \left[1 + \frac{3}{c^2} \vec{e}_\alpha \cdot \vec{u} + \frac{9}{2c^4} \left(\vec{e}_\alpha \cdot \vec{u} \right)^2 - \frac{3}{2c^2} \vec{u} \cdot \vec{u} \right]$ (2.20)

$T_\alpha = 0$ if $x + \vec{e}_\alpha \Delta x$ is a fluid node, and $T_\alpha = 1$ if $x + \vec{e}_\alpha \Delta x$ is a solid node. Then PDFs for thermal boundary can be calculated by following equation:

$$f'_\alpha = T' w_\alpha \left(1 - T_\alpha\right)\left[1 + \frac{3}{c^2}\vec{e}_\alpha \cdot \vec{u} + \frac{9}{2c^4}\left(\vec{e}_\alpha \cdot \vec{u}\right)^2 - \frac{3}{2c^2}\vec{u} \cdot \vec{u}\right] + f_\alpha T_\alpha \tag{2.21}$$

where f' stands for the PDFs after the boundary condition treatment, and f denotes the PDFs after the streaming step. Equations (2.19)-(2.21) can be directly used for the Q15, Q19, and Q27 schema for 3-D cases.

ii) Heat flux BC: Another common thermal boundary condition is the assumption of a constant heat flux on/at a surface. The formulation of these BCs is directly analogous to those of the isothermal case. After each streaming step, the temperature of the inner domain can be obtained by equation $T = \sum_{\alpha=0}^{b} f_\alpha^T$. A second-order finite difference scheme is used to find the temperature on the wall. For example, for the bottom wall at $y = 0$, $\left.\frac{\partial T}{\partial y}\right|_{i,1} = \frac{4T_{i,2} - T_{i,3} - 3T_{i,1}}{2\Delta y}$.

After finding the wall temperature, the same procedure as described in the isothermal wall case is used to calculate the unknown PDFs.

3.0 A MASS CONSERVING BOUNDARY CONDITION FOR THE LBE METHOD

As mentioned in section 2.2, in the introduction to curved boundary condition treatments, we have developed a mass conserving boundary condition for the LBE method when a body force such as gravity is applied to fluids. When body forces are present, both FH and MLS BCs can result in mass leakage, which is an unphysical reduction of the total mass of the system. In this section we demonstrate the performance of our improved boundary condition method through some example cases.

3.1 FULLY DEVELOPED FLOW IN A 2-D CURVED PIPE

The FH, MLS, and our newly refined mass conserving BC were all designed for a curved boundary, so we tested our new boundary condition for steady flow in a curved pipe, which is very common in practical thermal fluid systems. As shown in Figure 3.1, the fluid flows into the pipe through cross section A and flows out though cross section B. A body force in the form of gravity is applied on the fluid along the negative y-direction.

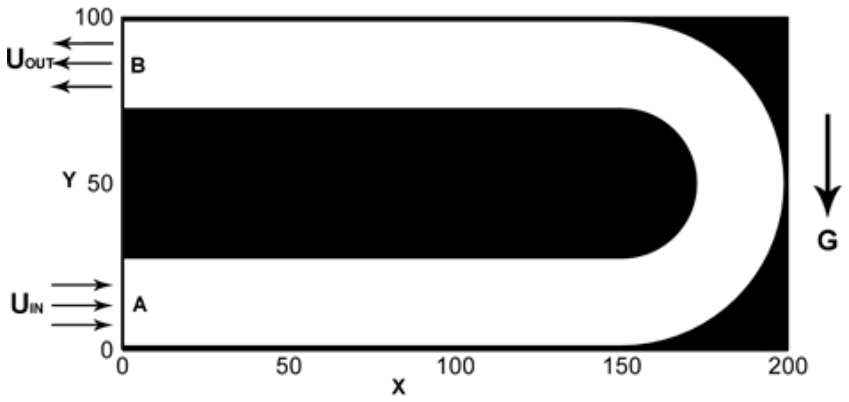
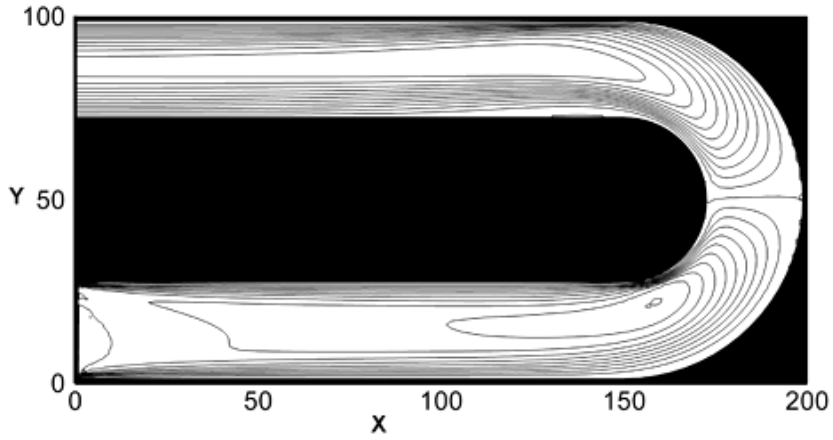
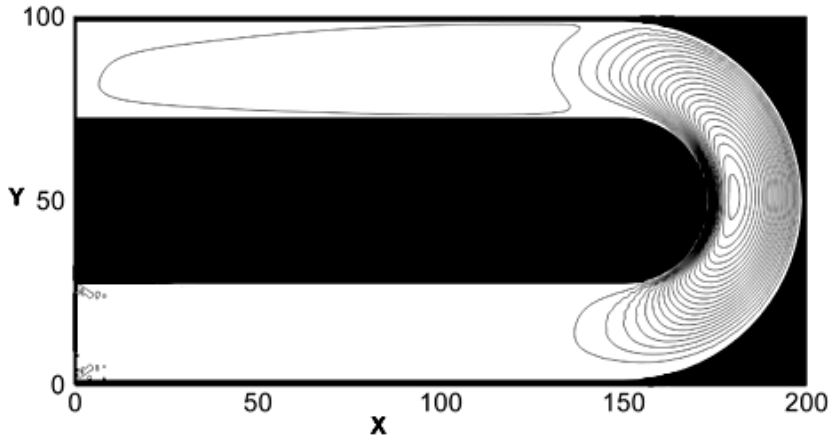


Figure 3.1: Schematic computational area for flow in a curved pipe

On the wall of the pipe, equations (2.2)-(2.7) are used to update the boundary conditions for the $\widetilde{f}_{\bar{\alpha}}(\vec{x}_b, t)$ functions. Figure 3.2 shows the resulting x-velocity and y-velocity contours using the mass conserving BC, respectively. Figure 3.3 shows how the total system mass changes with time. Our new boundary condition can cause the total system mass to converge to a constant value, while the MLS BC demonstrates a steady mass leakage.



a



b

Figure 3.2: Flow in a curved pipe (a) x-velocity contour and (b) y-velocity contour

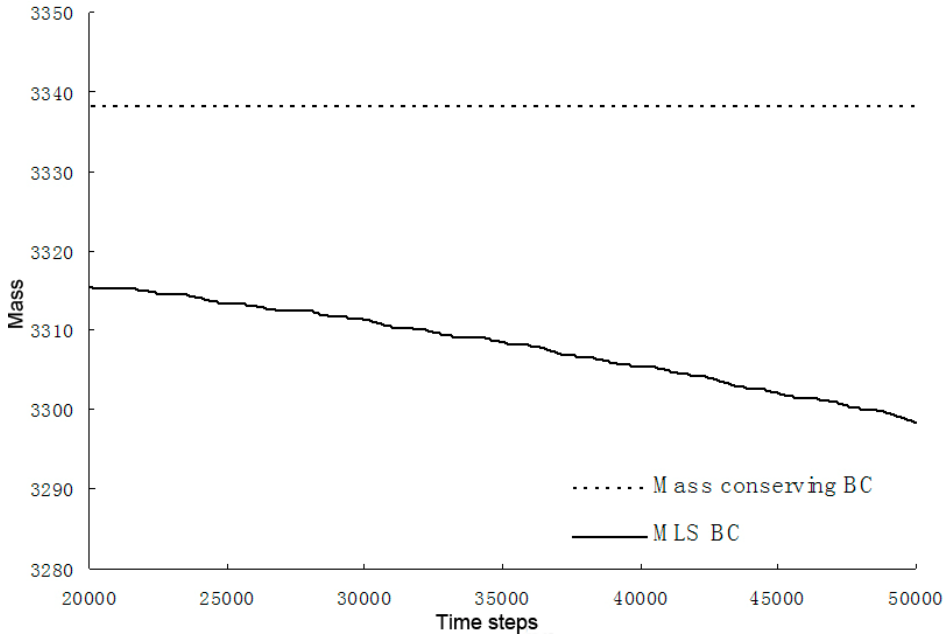


Figure 3.3: System mass change with time using the mass conserving BC and MLS BC

3.2 STEADY AND UNSTEADY FLOW OVER A CIRCULAR CYLINDER

From the tests in Section 3.1, our mass conserving BC shows the expected ability of avoiding constant mass leakage. To further examine the accuracy of the mass conserving BC method on a curved wall, we conducted a final set of tests concerns 2-D steady and unsteady flows around a circular cylinder placed in a rectangular channel. For steady-state flow, the problem at Re=10 is tested. For unsteady flow, the simulation is at Re=100, resulting in a periodical vortex sheet.

As shown in Figure 3.4, for steady-state problems, the circular cylinder is placed in a domain with 105×70 nodes, and the center of the cylinder is at the origin (0, 0) of the grids. The

periodical boundary condition treatment is used for the upper and lower boundaries. A uniform velocity is used at the inlet, and the extrapolation boundary condition is used at the outlet.

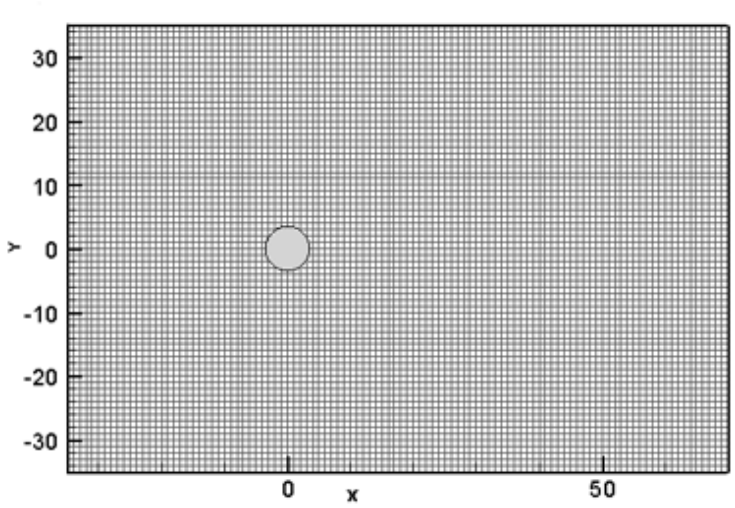


Figure 3.4: Schematic of the computational area for steady flow over a circular cylinder. The cylinder has a diameter of 7 lattice units. The cylinder center-to-center distance (H) is 70 lattice units

On the surface of the circular cylinder, the mass conserving BC treatment is used to update the boundary conditions for the $\tilde{f}_{\bar{\alpha}}(\vec{x}_b, t)$ values. Figure 3.5 shows the streamline and the contour of the pressure at $\mathrm{Re} = 2Vr/\upsilon = 10$ using a radius (r) of 3.5. Figure 3.6 shows the velocity profile $u(x=0, y)/V$ for $H/r = 20$ at $\mathrm{Re} = 10$. The MLS BC was also used in the same conditions and grids for comparison. The finite difference solution is obtained using body-fitted coordinates and are distributed along the upper surface of the circle over 200 grid points. Figure 3.7 shows the centerline (y=0) velocity variations, upstream and downstream, at Re=10. The results using the MLS BC and finite different results are also presented for comparison.

As can be seen, the mass conserving solid wall BC results agree well with the result of the finite difference solution, and have a similar order of accuracy as those for the MLS BC. The

difference in the results between the MLS and mass conserving BC is around 0.01% of the MLS BC's results (this difference cannot clearly be seen from the figures). Hence, our mass conserving BC would not reduce the accuracy for problems with geometries that include curved geometry boundaries.

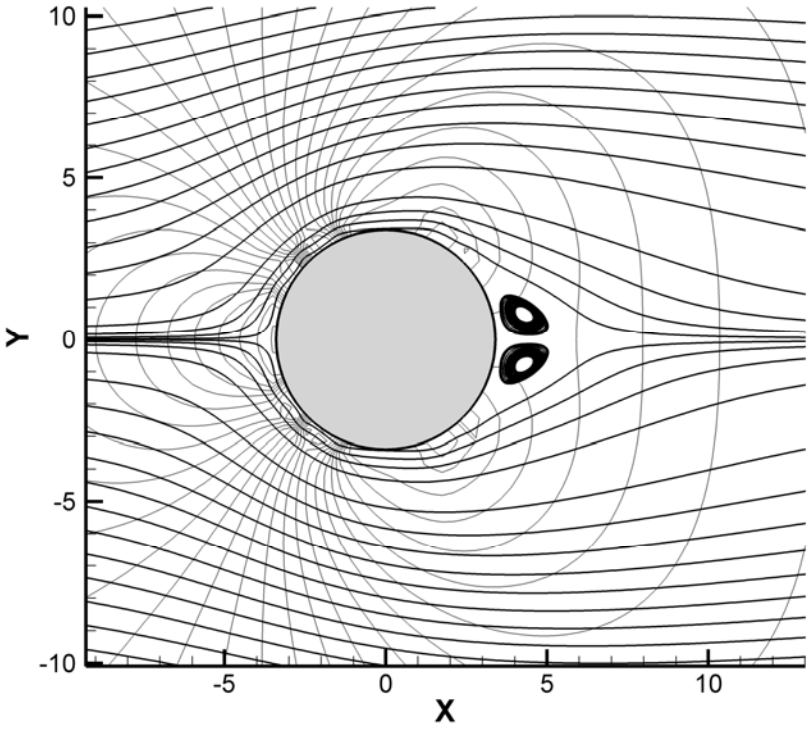


Figure 3.5: Steady flow around a cylinder at Re=10, with streamlines and pressure contours

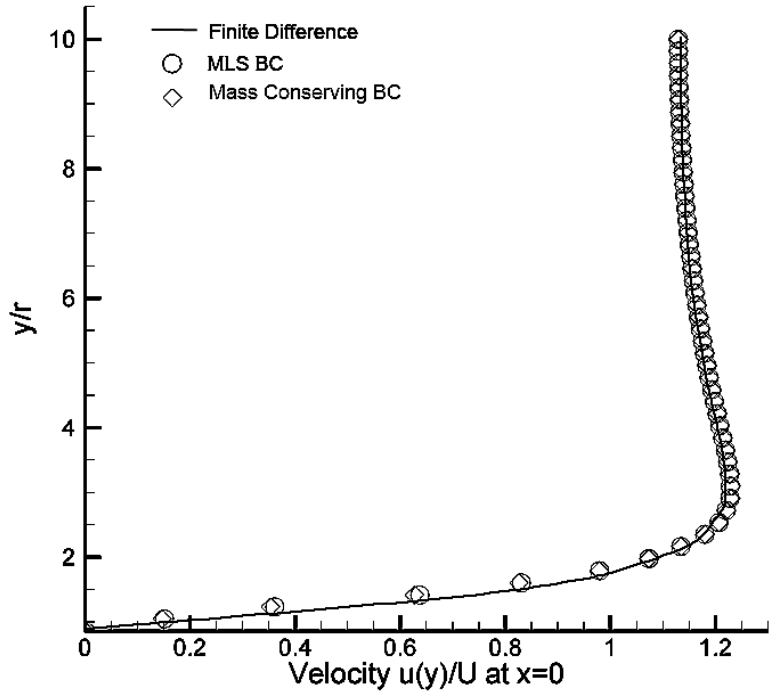


Figure 3.6: Comparison of the velocity profiles at x=0 for the MLS BC, mass conserving BC and finite difference result for Re=10

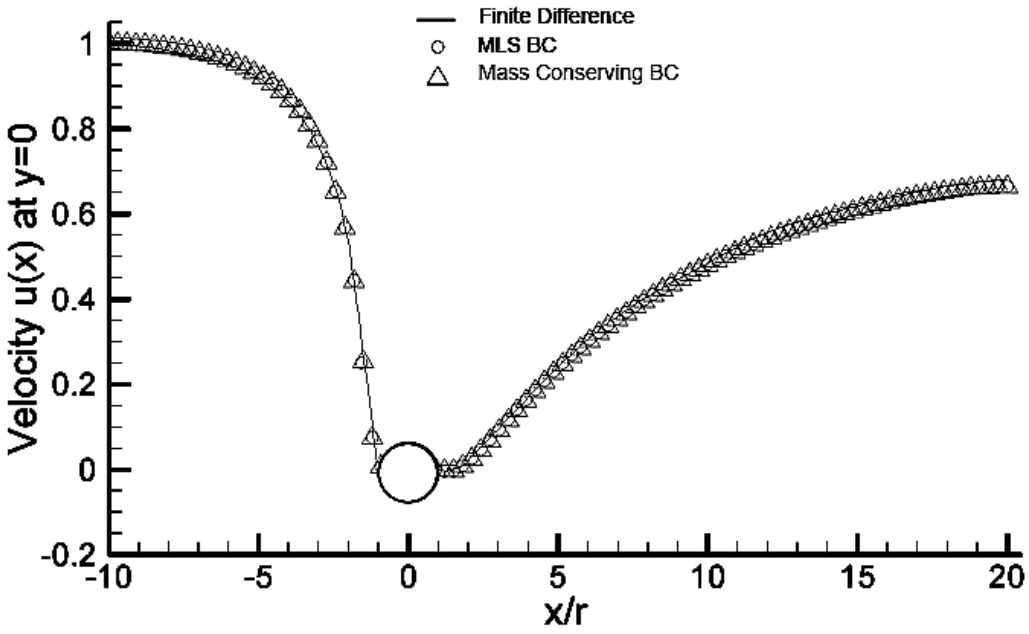
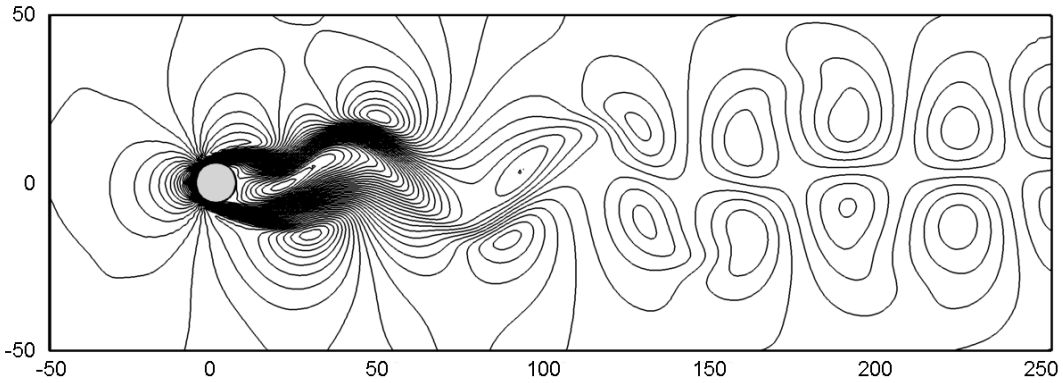
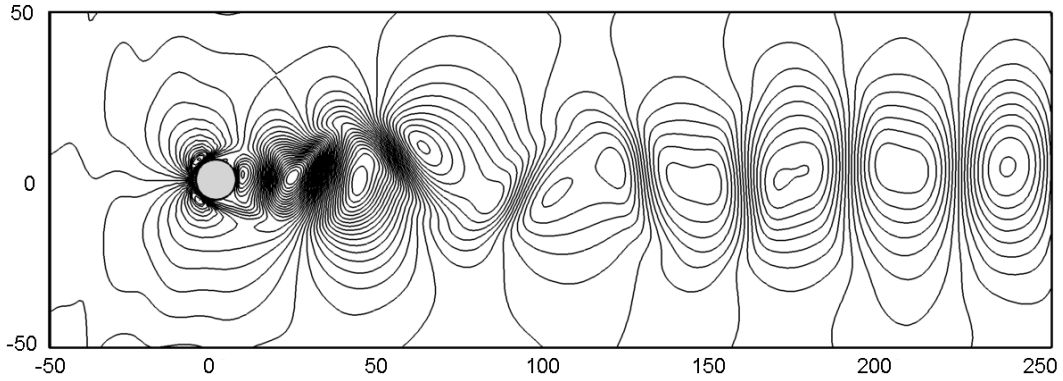


Figure 3.7: Comparison of the velocity variation at the centerline (y=0) for the MLS BC, mass conserving BC and finite difference result for Re=10

For unsteady flow, the problem is tested in a 300×100 node regime, with a radius of r=6. The periodical boundary condition treatment is used for the upper and lower boundaries. Again, a uniform velocity is given at the inlet, and the extrapolation boundary condition is used at the outlet. The results of the developed periodical flow are shown in Figure 3.8 for the x-velocity and y-velocity contours, respectively. Our mass conserving BC shows necessary capacity of describing and predicting unsteady flow problems. Like other curved BCs, such as FH and MLS, our BC does not cause obvious jagged velocity and pressure distribution near the solid boundary wall in this low grid resolution case (the radius of the cylinder is only 6 grids).



a



b

Figure 3.8: Unsteady flow around a cylinder at Re=100: (a) x-velocity contour; (b) y-velocity contour

3.3 CONCLUSION AND DISCUSSION

In summary, we have proposed a second-order accurate mass conserving boundary condition for the LBE method. Our mass conserving BC will not result in the constant mass

leakage that occurs for other both first- and second-order BCs in some cases. We can derive an expression for the mass leakage of the MLS BC, which can be calculated by:

$$mass\ leakage = 0.5 * (nx + 1) * time\ interval * [\rho(ny - 1) - \rho(1)]g \tag{3.1}$$

This shows that the larger the gravity or body force, the larger the mass leakage, so our BC is a good alternative for solving problems with large magnitudes of body forces, such as gravity or a magnetic force.

Moreover, when there is the presence of an "artificial gravity" for certain problems, such as a thermal problem with natural convection, there are also benefits from using our mass conserving BC. For example, the Rayleigh number, which is important in many thermal-fluid problems, is defined as $Ra = \frac{g\beta\Delta T(ny)^3}{\nu\alpha}$, where β is the thermal expansion coefficient, and ny is the lattice size in the y direction. Hence, in such problems, when we want to simulate a large Rayleigh number, we can increase the gravity instead of using more grid units in the y-direction, because increasing gravity does not significantly add to the computation load. By doing so, the efficiency of the code can be obviously increased when a relatively larger Rayleigh number is needed. Although increasing the grids in the y-direction may seem more direct and efficient, because it is a cubic in the equation, the increasing of grids can lead to a large increase in computational time and money for many cases.

Additionally, because each lattice must be square or cubic (for a 3D problem), to maintain the ratio in the simulation domain, when ny is increased, we also need to also increase nx and nz. The Rayleigh number commonly ranges from 10^3 to 10^8 for practical engineering problems. Hence, adjusting the Rayleigh number by only changing ny would make the simulation very inefficient. Furthermore, the amount of available physical memory and the

operating system's addressing capability [33] strictly limits the number of grids. For three-dimensional problems, especially for multi-component, multi-phase flows or a thermal system, the memory usage often reaches that limit. When using the MLS BC, increasing the grids also increases the mass leakage, as seen in equation (3.1). Since our BC allows the requirement of mass conservation to be exactly met, the stability of the model is greatly increased. With this accurate and stable BC treatment, LBE method can be used at more practical problems such as multi-component multi-phase system. In the flowing chapters, we will introduce more practical applications.

4.0 THE LBE METHOD FOR MULTI-COMPONENT MULTI-PHASE FLOW WITH HIGH DENSITY RATIOS

Multi-component multi-phase (MCMP) flow is very common in engineering or industrial problems and in nature. Because the lattice Boltzmann equation (LBE) model is based on microscopic models and mesoscopic kinetic equations, it offers many advantages for the study of multi-component or multi-phase flow problems. While the original formulation of Shan and Chen's (SC) model can incorporate some multiple phase and component scenarios, the density ratio of the different components is greatly restricted (less than approximately 2.0). This obviously limits the applications of this MCMP LBE model. Hence, based on the original SC MCMP model and the improvements in the single-component multi-phase (SCMP) flow model reported by Yuan and Schaefer [19], we have developed a new model that can simulate a MCMP system with a high density ratio.

4.1 INTRODUCTION TO MCMP FLOW WITH HIGH DENSITY RATIOS

As in the single component multi-phase flow case, the separation of different components in MCMP flows is also due to the long-range interaction between the molecules of the fluid [18], so the interaction force must be revised to also include two parts for a multi-component fluid. One contribution is the interaction between molecules of the same component, and another is the

interaction between molecules from different components. In a similar manner as in equation (1.13), these two parts can be expressed as:

$$\vec{F}_{i,i}(\vec{x}) \cong -c_0 \psi_i(\vec{x}) g_{ii} \nabla \psi_i(\vec{x}) \tag{4.1}$$

$$\vec{F}_{i,j}(\vec{x}) \cong -c_0 \psi_i(\vec{x}) g_{ij} \nabla \psi_j(\vec{x}) \quad i \neq j \tag{4.2}$$

$\vec{F}_{i,i}$ is the force between the different particles of component i, and $\vec{F}_{i,j}$ indicates the force between the component i and component j.

To increase the density ratio between different components, one first should increase the density ratio for the different phases of each single component. Work has already been done on increasing the density ratio for single-component multi-phase (SCMP) flows. For example, as reported by Swift [34], the maximum density ratio obtained using the free-energy-based approach is less than 10:1, and the largest density ratio tested in the He, Chen and Zhang (HCZ) approach is 40:1 [35]. These are improvements, but are still not large enough for most practical problems. This is because the idea gas EOS and original SC model show unrealistic pressure-density relationship. They give a high compressibility for the liquid phase, which is even higher than the vapor phase. Yuan and Schaefer [19] found, hence, that is possible to simulate SCMP flows with a density ratio that can reach 1,000:1 by using a more accurate EOS such as the van der Waals, Peng-Robinson, or Carnahan-Starling EOS [36], and all of these EOSs are easy to apply to the LBE model. To do so, first, the pressure term in equation (1.21) ($\psi(\rho) = \sqrt{\frac{2\left(p - c_s^2 \rho\right)}{c_0 g}}$) is replaced by the pressure formulation in an EOS (using Peng-Robinson as example, $P = \frac{\rho RT}{1 - b\rho} - \frac{a\alpha(T)\rho^2}{1 + 2b\rho - b^2\rho^2}$). Secondly, the effective mass is substituted into equation (1.13) for calculation of the interparticle forces ($\vec{F}(\vec{x}) \cong -c_0 \psi(\vec{x}) g \nabla \psi(\vec{x})$). Finally,

equation (1.20) is used to calculate the shifted velocity ($\vec{u}^{eq} = \vec{u}_i + \frac{\tau_i \vec{F}_{total,i}}{\rho_i(\vec{x})}$). Achieving a density ratio of up to 1,000:1 means that the LBE model can work well for the simulation of most single-component vapor-liquid flows.

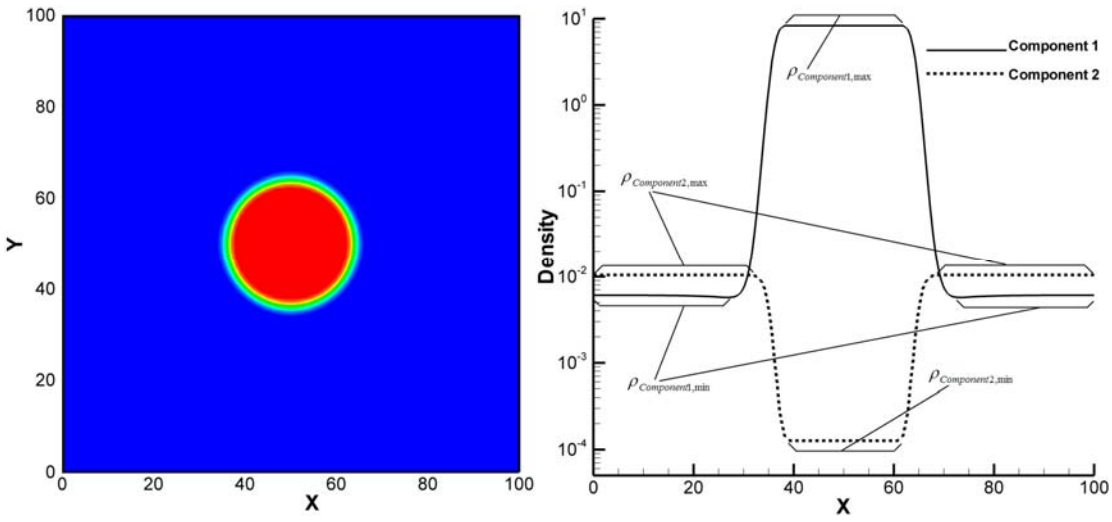
Unlike in the original SC model, though, the coefficient of interaction strength within a component (g_{ii}) here cannot control the overall interaction strength. (Indeed, it is canceled out when we substitute equation (1.21) into equation (4.1) and (4.2). The only requirement for g_{ii} is to ensure that the whole term inside the square root in equation (1.21) is positive. However, we have found that the coefficient of interaction strength between different components g_{ij} is very important for creating and extending the MCMP LBE model. Firstly, when equation (1.21) is substituted into equation (4.2), g_{ij} is not eliminated. Secondly, from equation (4.2), it can be seen that g_{ij} affects the magnitude of the interparticle force between different components $F_{i,j}$. The behavior of the interaction between the different components is primarily controlled by this force, so interaction can be adjusted through changing the value of g_{ij}. From our tests, this force plays a critical role in adjusting the system density ratio, which we will explain and demonstrate in the following sections.

4.2 SIMULATION OF AN EQUILIBRIUM DROPLET WITHOUT A BODY FORCE AND EXTERNAL FORCES

The first example is the simulation of a circular droplet in a 100×100 2D square domain for a liquid-gas system without body forces. A periodical boundary condition is applied on the

two vertical boundaries ($x = 0$ and $x = 100$), and a bounce back treatment is used for the upper and bottom boundaries ($y = 0$ and $y = 100$). For an initial condition for component 1 (the liquid component), we specified a low density value ($\rho = 0.005$) for most of the simulation region, except for a circular area at the center of the simulation space, where a high density value was set ($\rho = 1$). For component 2 (the gas component), density was set to a relatively higher value than component 1 ($\rho = 0.01$) at most areas of the simulation domain, except the center circle area (corresponding to the liquid component), where an extremely low density was used ($\rho = 0.0001$). Generally, if the problem can reach a converged result, the shape of the high density area is not important. A square, triangle, or any random geometry will not affect the final result. Because a droplet always tends to become the shape that has smallest surface area, for this 2D case, the droplet will always become a circle. Hence, setting the initial shape to a circle increases the problem's convergence speed, since as with most numerical methods, the less difference between the initial conditions and final result, the faster the model converges. Figure 4.1a shows the density contour of component 1 (the liquid component), and Figure 4.1b shows the comparison of the density of the two components along the center line ($y = 50,\ 0 \le x \le 100$) on a $\log_{10}$ scale. For convenience, we have highlighted several segments on these lines, which are the maximum density of component 1, minimum density of component 1, maximum density of component 2, and minimum density of component 2, noted as $\rho_{Component1,\max}$, $\rho_{Component1,\min}$, $\rho_{Component2,\max}$, and $\rho_{Component2,\min}$, respectively. The density variation in each segment is very small compared to the density change at the interface of the different components, so this small variation can be neglected. In Figure 4.1b, the density of component 1 in the droplet ($\rho_{Component1,\max}$) is on the order of 10^1, while the density of component 2 around the droplet

($\rho_{Component2,\max}$) is on the order of 10^{-2}. Hence the density ratio of these two components is around 1,000. In general, the density ratio of a system refers to the ratio between the maximum densities of the two components ($\frac{\rho_{Component1,\max}}{\rho_{Component2,\max}}$).



a b

Figure 4.1: (a) Density contour for a circular droplet, (b) Comparison of the densities of the two components along the center line ($y = 50$, $0 \le x \le 100$)

4.2.1 Maximum Force Ratio

As detailed in Section 4.1, the balance of the weights of the two parts $\vec{F}_{i,i}$ and $\vec{F}_{i,j}$ directly affects the density ratio of the different components. The forces $\vec{F}_{i,i}$ and $\vec{F}_{i,j}$ vary in the simulation domain and reach a maximum value around the interface of the two components. Figure 4.2 shows the distribution of the ratio of $F_{1,1-x}$ and $F_{1,1,\max}$ in the test area, where $F_{1,1-x}$ is

the x-direction of the interparticle force of component 1 and $F_{1,1,\max}$ is the maximum force in the entire test region.

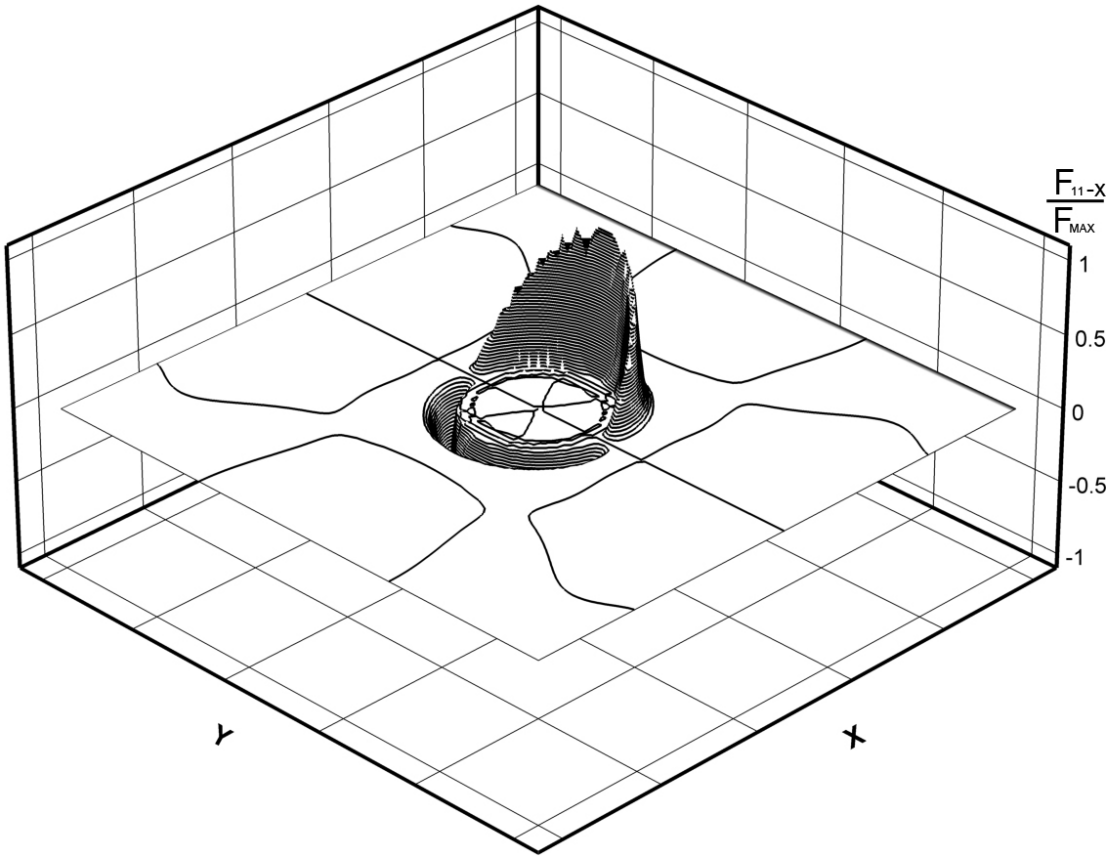


Figure 4.2: Distribution of the ratio of $F_{1,1-x}$ and $F_{1,1,\max}$

Hence, we find that the ratio of the maximum forces ($\frac{|F_{1,1\max}|}{|F_{1,2\max}|}$) is a key factor, and examine how this ratio affects the density ratio of the two components.

This simulation is for an equilibrium droplet in a 100×100 2D square domain without body forces. Figure 4.3 shows four tests in this simulation domain with the $\frac{|F_{1,1\max}|}{|F_{1,2\max}|}$ ratio set

equal to 260, 75, 56, and 47. The solid line is the density distribution of component 1 along the center line ($y = 50$, $0 \le x \le 100$), and the dotted line is the density distribution of component 2 along the center line. The density ratios of the two components are 278, 83, 53, and 44, respectively.

Realistically, two unmixable components cannot occupy the same physical space at the same time. For example, in a water-oil system, water cannot flow into the space that the oil already occupies. This means that the density of water should be zero in the area where the density of oil takes precedence. For a numerical model, it is almost impossible to have a density distribution with a zero value (or an extremely small value near zero). What can be done is to reduce the value of $\rho_{Component1,\min}$ and $\rho_{Component2,\min}$, or to increase the ratio $\frac{\rho_{component2,\max}}{\rho_{component1,\min}}$ and $\frac{\rho_{component1,\max}}{\rho_{component2,\min}}$. As shown in Figure 4.1b and Figure 4.3, the ratio $\frac{\rho_{component1,\max}}{\rho_{component2,\min}}$ is on the order of 10^3 which is relatively high. However, the ratio $\frac{\rho_{component2,\max}}{\rho_{component1,\min}}$ is only on the order of 10^0 to 10^2 for different cases with different system density ratios. Many actual MCMP flow systems can exhibit a ratio of more than 10^3 , so even the improvement shown by our simulation is not high enough for some cases. This is caused by the limitation on the density ratio in the SCMP LBE model. For example, for a water-air system at a standard state ($T = 0^\circ C$, $P = 760 mmHg$, and $g = 9.80665 m/s^2$) the density of water vapor in the air is around $0.02 kg/m^3$, while the density of water is around $1 \times 10^3 kg/m^3$. This means that, for such a system, the density ratio of a single-component $\frac{\rho_{component1,\max}}{\rho_{component1,\min}}$ should be 5×10^4 , and currently no SCMP LBE model can

stably reach such a high density ratio. Fortunately, the density of water vapor in the air is higher than this standard state in many practical and engineering problems (as high as $1kg/m^3$), such as in fog and most multi-phase thermal problems. Hence, our MCMP LBE model can be applied to study these systems and problems. For a further extension of our MCMP LBE model, it may depend on a more powerful SCMP LBE model.

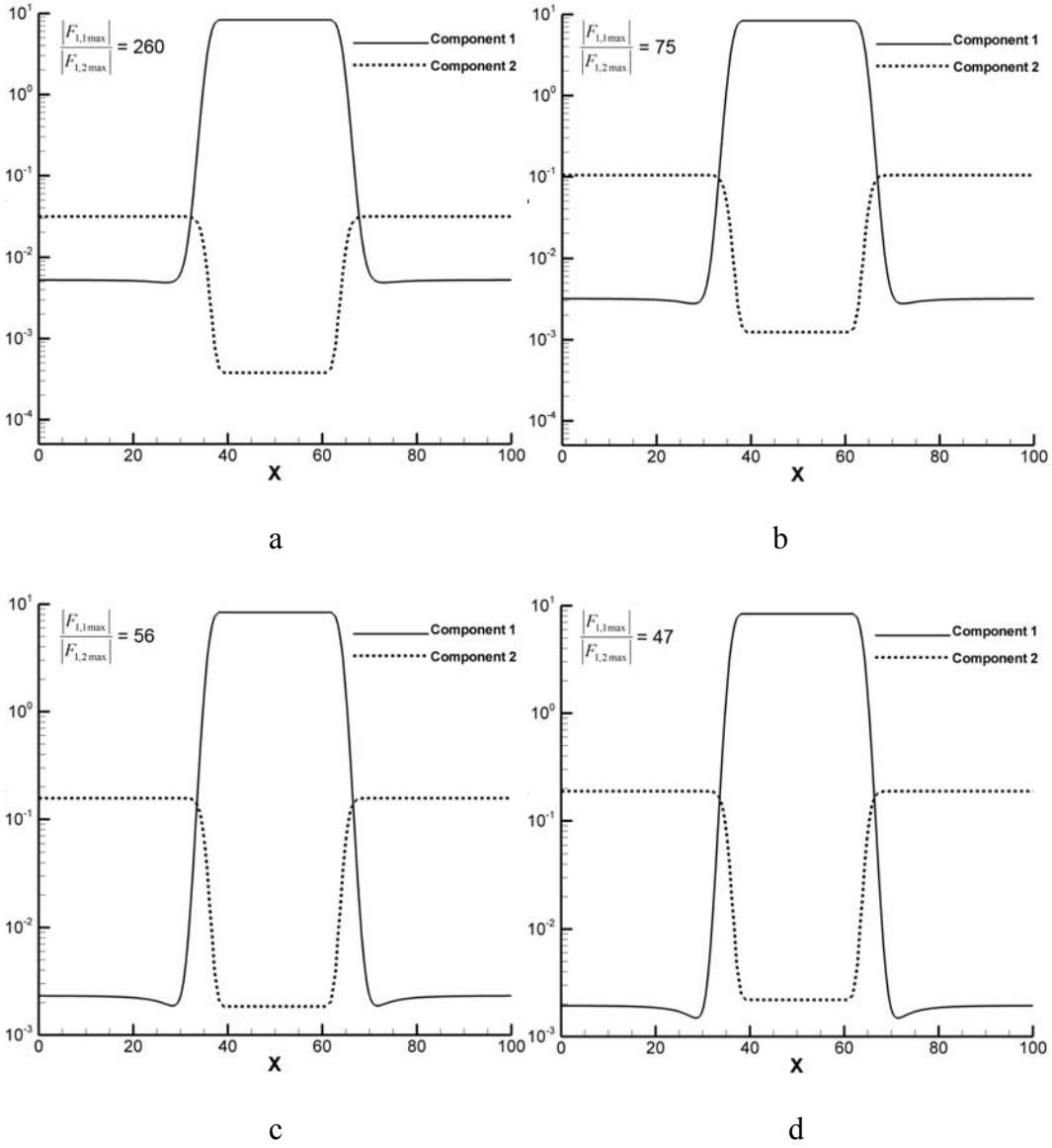


Figure 4.3: Comparison of the density of the two components along the center

line ($y = 50$, $0 \le x \le 100$) for (a) $\frac{|F_{1,1\max}|}{|F_{1,2\max}|} = 260$, (b) $\frac{|F_{1,1\max}|}{|F_{1,2\max}|} = 75$,

(c) $\frac{|F_{1,1\max}|}{|F_{1,2\max}|} = 56$ and (d) $\frac{|F_{1,1\max}|}{|F_{1,2\max}|} = 47$

Figure 4.4 shows the relationship between the density ratio and maximum force ratio. From Figure 4.4, we can see that by changing the ratio of the maximum force, the density ratio can be adjusted from 1 to 1,000. This is a substantial range that can cover many kinds of liquid-gas or liquid-liquid two component systems. One should notice that the forces $\vec{F}_{1,1}$ and $\vec{F}_{1,2}$ are in conflict when they are of the same order. Therefore, if the density ratio should be around 1, the interparticle force between the different components $\vec{F}_{1,2}$ is 1 or 2 orders of magnitude higher than the force $\vec{F}_{1,1}$ (or $\vec{F}_{1,1}$ can be directly neglected). This coincides with the actual physical phenomenon. When the density ratio of two components is very large, such as in a water-air system, the formation and position of the interface are mainly controlled by the tension of the water (as well as the body force and the interaction between the fluid and solid wall, if those are present). The interparticle force between the different components is very weak, especially for steady-state behavior. With a decrease in the density ratio, the interparticle force becomes more important. When the density ratio is near 1.0, as in a water-oil system, the formation and position of the interface is mainly caused by the interparticle force between the different components instead of the tension, body, force or other forces applied on each single component.

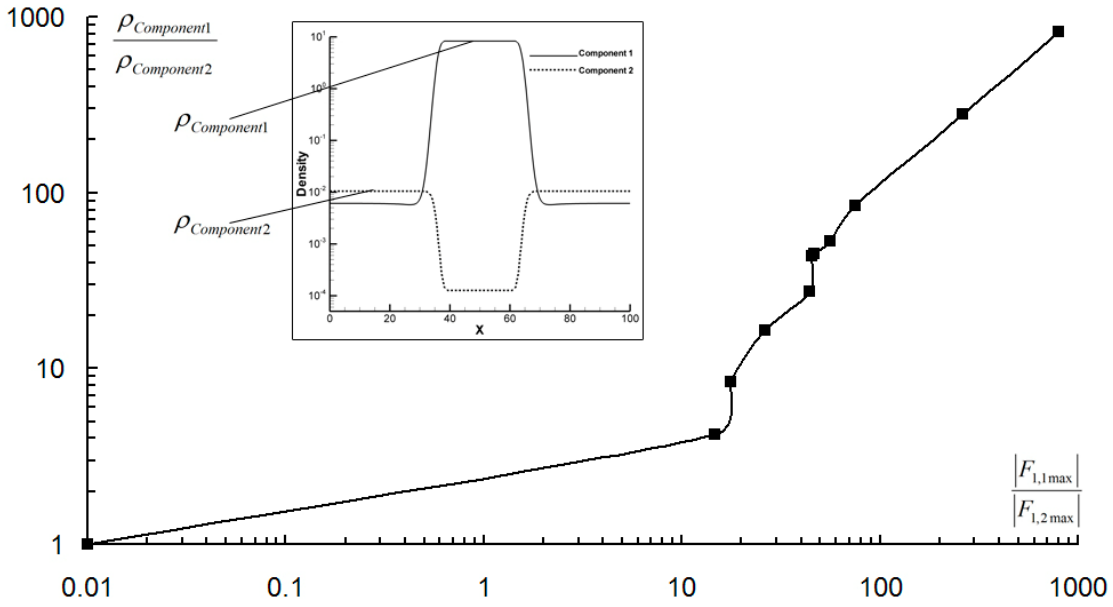
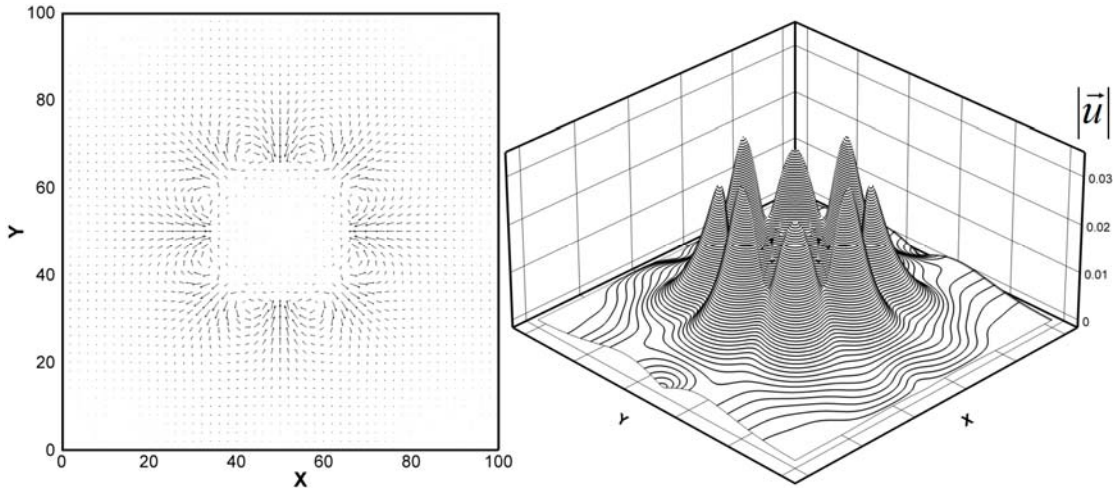


Figure 4.4: Density ratio $\frac{\rho_{Component1}}{\rho_{Component2}}$ variation with the ratio of the maximum force $\frac{|F_{1,1\max}|}{|F_{1,2\max}|}$

4.2.2 Spurious Current

A spurious current around the interface is an unphysical velocity that exists in the LBE model. Much research has been focused on discovering why the spurious current exists and on reducing it. One of the explanations is that the anisotropy of the discretized momentum space causes the spurious current [37]. Hence, by modifying the LBE model, the spurious current can be decreased, but cannot be eliminated. Our MCMP LBE model performs well in reducing the spurious current to an acceptably small value. Figure 4.5a shows the spurious current vectors in

the same 2D simulation domain for a density ratio of around 1,000, and Figure 4.5b shows the magnitude of the spurious current.



a b

Figure 4.5: (a) Vector of spurious current around the droplet, (b) magnitude of the spurious current

We can see that, for a density ratio of around 1,000, the spurious current is still under 0.032 (the main current magnitude is on the order of 10^{-13} for this equilibrium test case). For a dynamic problem, the main stream would be 10^{-1} to 10^{1}, so this spurious current is not easy to be noticed. This spurious current looks large compared to the surrounding velocity, but it is as low as or even lower than most SCMP or MCMP LBE models. The spurious current in most popular and widely used SCMP or MCMP LBE models may be around 0.05 to 0.1 for the same equilibrium test with a much smaller density ratio (on the order of 10^{0}). Furthermore, the area affected by the spurious current is very small in our MCMP LBE model. For the case with a 1,000:1 density ratio, the velocity is reduced to 10^{-13} at $2r$ far away from the droplet center (r

is the radius of the droplet), as shown in Figure 4.5b. For the cases with a smaller density ratio (smaller than 300:1), the affected area is also much smaller. In those cases, the spurious current disappears around $1.2r \sim 1.3r$ away from the droplet center. Figure 4.6 shows the relationship between the spurious current and the density ratio using our MCMP LBE model. It can be seen that the spurious current is tightly controlled for the full range of a density ratio that covers most kinds of liquid-gas and liquid-liquid systems.

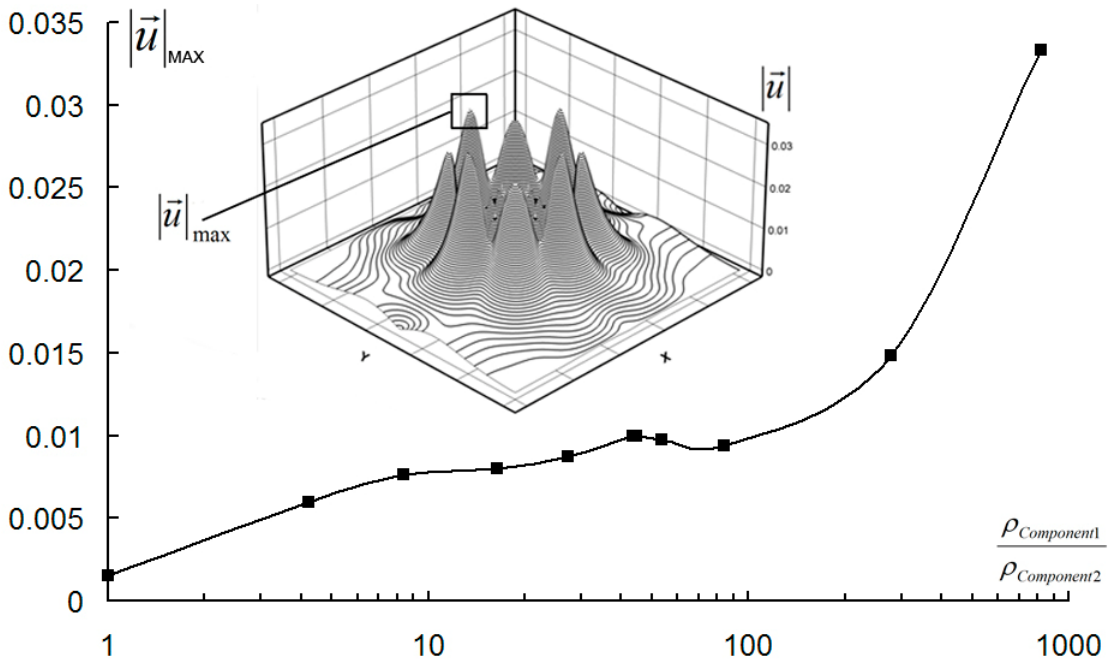


Figure 4.6: The maximum magnitude of the spurious currents varying with the density ratio

4.2.3 Convergence Speed

Convergence is another important criteria for any numerical model. Hence, we tested the convergence speed of our MCMP LBE model for the same 2D case. Figure 4.7 shows the residual with respect to the time steps. The residual here is defined as the difference between the maximum magnitude of the spurious current of two nearby time steps:

$$residual = \left|\vec{u}\right|_{\max}^{n} - \left|\vec{u}\right|_{\max}^{n-1} \tag{4.3}$$

The case shown in Figure 4.7 is again for a density ratio of around 1,000. We can see that our MCMP LBE model converges quite quickly. Before 5,000 steps, the residual is already smaller than 1.0×10^{-6}, and near 30,000 steps, the residual reaches 1.0×10^{-16}, which is the ultimate precision for most double precision computational code, while some other SCMP or MCMP may need over 50,000 steps to reach the similar magnitude of residual. For those cases with a smaller density ratio, the model converges even faster.

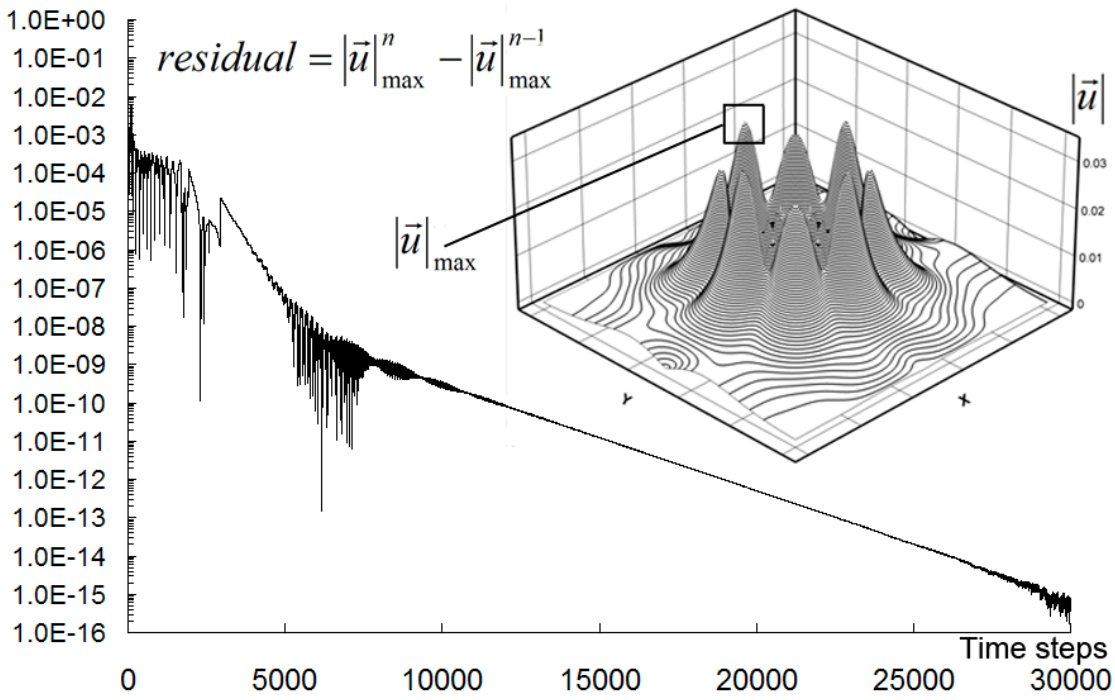


Figure 4.7: The residual changes with time steps

4.3 SIMULATION OF AN EQUILIBRIUM DROPLET AFFECTED BY A BODY FORCE AND EXTERNAL FORCES

One of the strengths of the LBE model, besides its ability to simulate MCMP flow, is its flexibility for applying body and external forces on a flow. Hence, in this second example, the boundary conditions and the number of grids are the same as in Section 3.2, but gravity is also added along the negative y-direction in the simulation domain, and the interaction between fluids and the solid boundary is considered. Equations (1.15) and (1.16) are used to calculate these two forces. The value of G_w controls the force between the fluid nodes and solid nodes. A positive G_w represents a repulsive force, which means that the solid wall repels the fluid and acts as a

hydrophobic surface. A negative G_w represents an attractive force, or hydrophilic surface. The contact angle can clearly reflect the property of the interaction between the fluid and the solid surface (smaller than 90° for a hydrophilic surface, greater than 90° for a hydrophobic surface). Figure 4.8 shows the position of the interface of the two components. Figure 4.8a is a droplet on a hydrophilic surface whose contact angle (θ) is about 55° and Figure 4.8b shows a droplet on a hydrophobic surface with a 107° contact angle. Our MCMP LBE model does not require an extra step or calculation for incorporation of external and body forces. The methodology is the same as in the MCMP LBE model discussed in the previous section, and the simulation results clearly reflect the effects of these forces. Our MCMP LBE model inherits the original LBE model's flexibility in dealing with external and body forces. It should be noted that the contact angle is controlled by both of the fluids near the solid-fluid interface. For a high density ratio, these co-effects are not strong, and the contact angle is mainly controlled by the fluid with the higher density. For a small density ratio, these co-effects become strong.

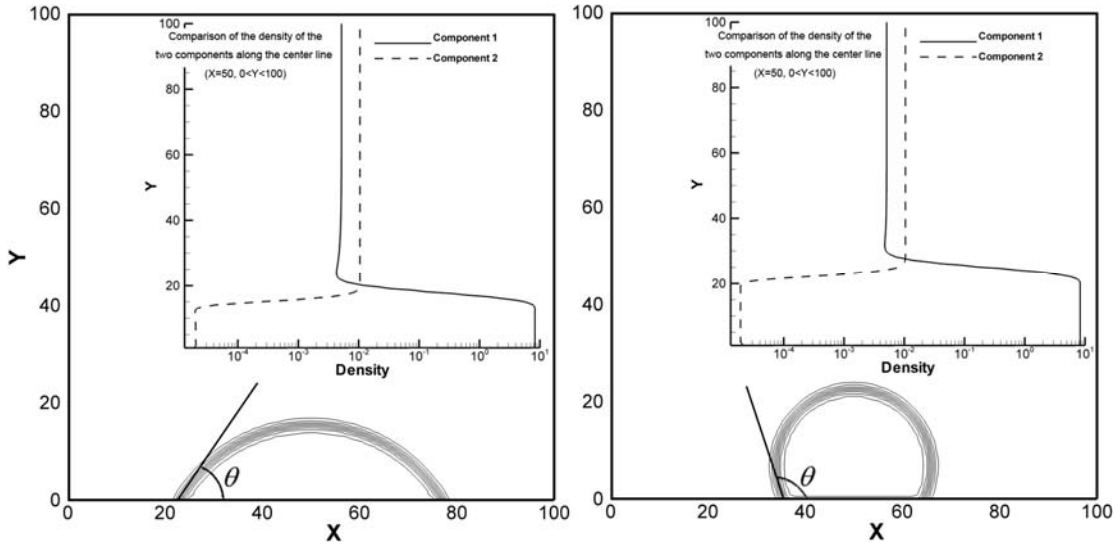
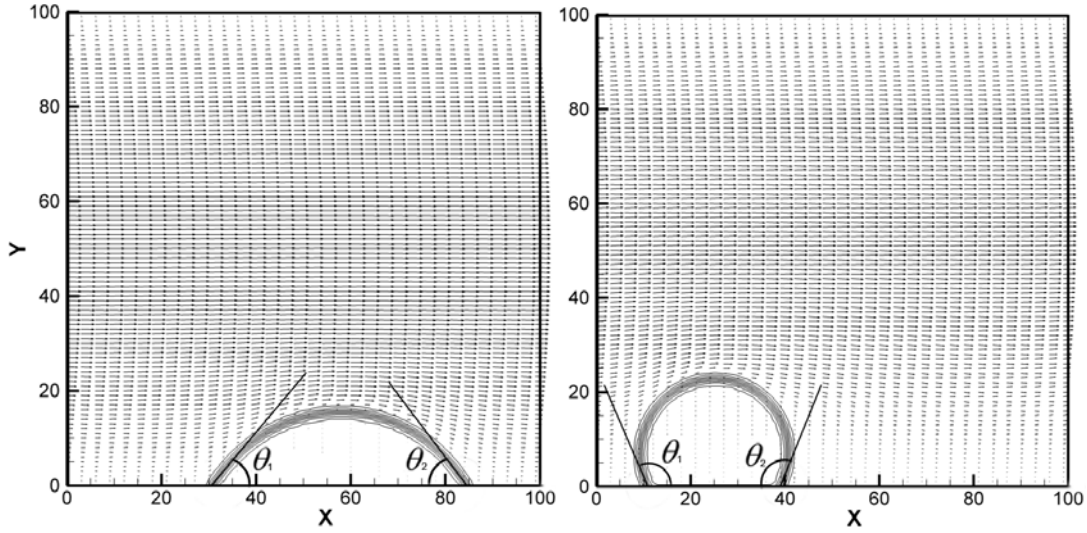


Figure 4.8: Position of the interface of the two components on a hydrophilic and hydrophobic surface, and comparison of the density of the two components along the center line ($x = 50,\ 0 \le y \le 100$)

4.4 SIMULATION OF A DROPLET IN A FLOW

This third example is a further refinement of the second one. Based on Section 3.3, we add a pressure drop along the positive x-direction, where $\frac{dP}{dx}$ equals 2×10^{-5} in lattice units. The system density ratio is about 300:1. Figure 4.9 shows the position of the interface of the two components and the velocity vector in the simulation domain. The main current magnitude is on the order of 10^{-1}, and the spurious current is too small to be observed compared to the main current. As expected for a hydrophilic surface, in Figure 4.9a, the two contact angles θ_1 (50°)

and θ_2 (56°) are different. This difference reflects the contact angle hysteresis phenomena. In Figure 4.9b, the two contact angles θ_1 (112°) and θ_2 (112°) are almost the same, which means that the repulsive forces at the two contact points are nearly equal, and the drag force is very small. This coincides with the real physical phenomena. The drag force on a hydrophobic surface is very small. For the ideal hydrophobic surface, the drag force should be zero.



a b

Figure 4.9: Position of interface of the two components and velocity vector,

(a) hydrophilic surface, (b) hydrophobic surface

4.5 CONCLUSIONS AND DISCUSSION

The theoretical basis presented in Section 3.1 and the simulation results detailed in Sections 3.2-3.4 show that applying a more realistic EOS for at least one component (particularly

the liquid component) instead of using the SC EOS is the precondition for increasing the density ratio of a MCMP flow system. Additionally, the balance between the forces $\vec{F}_{i,i}$ and $\vec{F}_{i,j}$ plays an important role in adjusting the potential density ratio for a MCMP system. In previous MCMP LBE models, the maximum density ratio was severely limited, partly due to the lack of attention paid to the interaction between molecules of the same component ($\vec{F}_{i,i}$). We found that by increasing the fraction of $\vec{F}_{i,i}$, the density ratio can increase substantially. All of the results in Section 4.2-4.4 are for a simple case in order to clearly demonstrate our MCMP LBE model's performance. To fully demonstrate the capabilities of our model, more realistic systems or problems should be tested, such as those with a more complicated solid boundary, a more sophisticated interface, dynamic problems, and moving boundary conditions. However, the ability to simulate these systems can be limited by computational power and/or time. To overcome this hurdle, we have further expanded our code for parallel calculations using a graphic processing unit, as described in the following chapter.

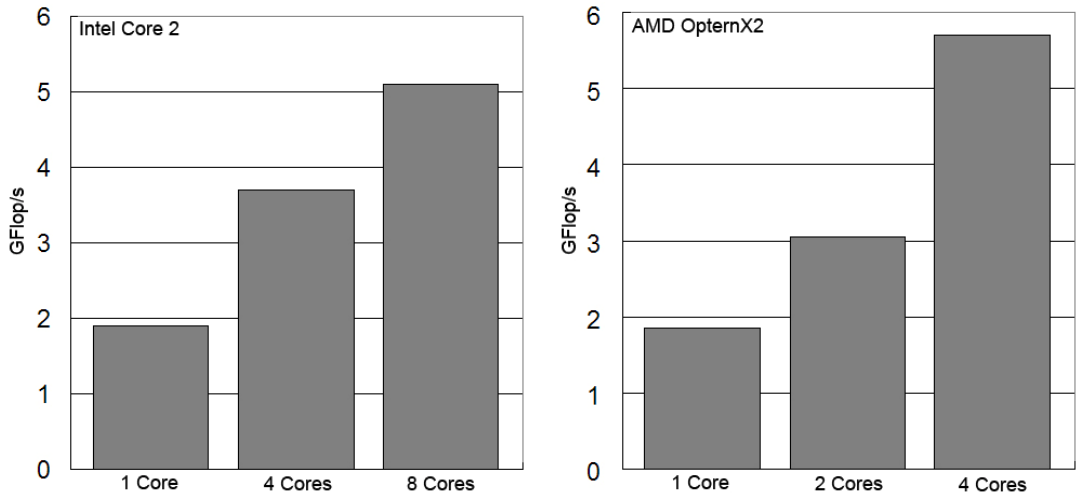
5.0 APPLICATION OF THE COMPUTE UNIFIED DEVICE ARCHITECTURE (CUDA) TO THE LATTICE BOLTZMANN EQUATION METHOD

Parallel computation on a supercomputer is a great way to increase the speed of a numerical simulation program. However, both the programming environment and the costs of a supercomputer may limit the application of these computational tools. Additionally, some recent research shows that the LBE method cannot benefit as much as conventional CFD methods from traditional parallel computation on supercomputers. In order to overcome these difficulties, we will use NVIDIA's Compute Unified Device Architecture (CUDA) technique, which offers an alternative way to increase the calculation speed of the LBE method. In this chapter, we will introduce this new computational tool, explain how this technique can benefit the LBE method, and demonstrate its performance through several practical cases.

5.1 BRIEF INTRODUCTION TO PARALLEL COMPUTATION FOR THE LBE METHOD

Although the development of new computer techniques has increased the efficiency of numerical simulation in recent years, the time cost of many computational problems is still unaffordable. This time consumption problem becomes serious for the 3-dimensional, multi-component and thermal lattice Boltzmann model. One obvious way to reduce the calculation

time is through parallel computation. However, recent research [38] has shown that the multi-thread LBE model on a traditional multi-core or multi-CPU computer system does not lead to a large increase in solution speed. This is because the LBE model requires a greater memory bandwidth than traditional CFD methods, and traditional multi-core or multi-CPU computer system cannot offer enough memory bandwidth for the 3-D LBE model. Memory bandwidth is the rate at which data can be read from or stored into a semiconductor memory by a processor. Figure 5.1a [38] shows that a system based on the Intel Core 2 CPU [39], which is the most popular CPU for personal computers and mainstream workstations, can only increase the speed by about 2.68 times when the number of cores increases from 1 to 8. For a system based on the AMD OpternX2 [40], shown in Figure 5.1b [38], which is widely used in mainstream and high-end workstations or servers, a 3-time increase in speed can be achieved when the number of cores increases from 1 to 4. The GFlop/s in the figures is a measure of a computer's performance, and it is an acronym meaning billion floating point operations per second. Hence, an increase in the number of cores cannot offer a linear speed increase. Also, an increase in the number of cores causes the price of the system to rise exponentially. Figure 5.2 shows the average price of a multi-core system in the current market. Although some newly developed multi-core computer systems, such as the IBM CELL system [41], partially solve the memory bandwidth problem, its impressive computational efficiency comes with a high price and a difficult programming environment that is a major departure from conventional programming.



a b

Figure 5.1: Comparison of runtime performance for a different number of cores on a 64X64X64 LBE problem[38] for the (a) Intel Core 2 and (b) AMD OpternX2

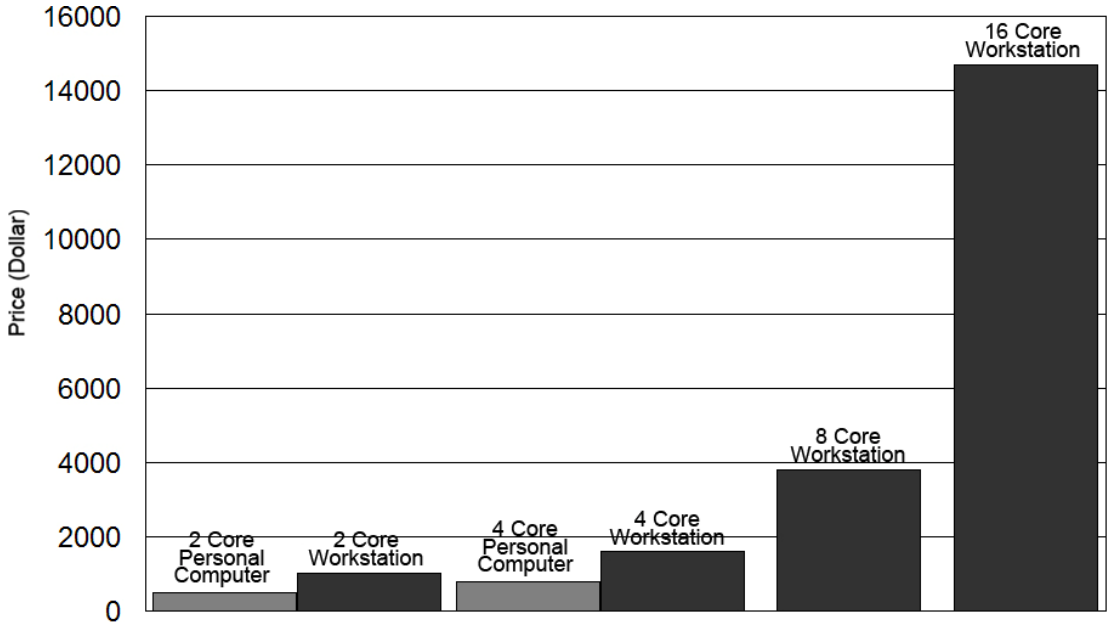


Figure 5.2: Price comparison for multi-core computer systems

Additionally, most supercomputers with over hundreds of CPUs to thousands of CPUs are based on the architecture of interconnecting a collection of computers via a high-speed network or switching fabric [42], as shown in Figure 5.3. The speed of these kinds of interconnection is significantly slower than the internal connection between the CPU and memory in a single computer. Hence, the speed increase of the parallel LBE model also would be greatly limited by the bandwidth between CPUs on a supercomputer.

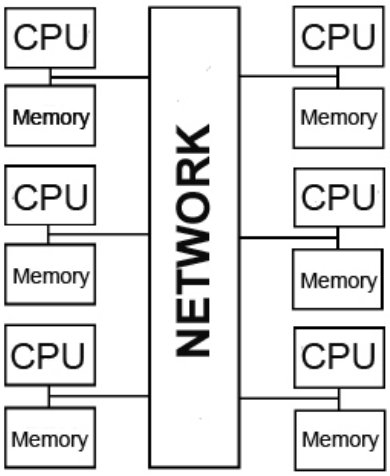


Figure 5.3: Modern supercomputer architecture

5.2 INTRODUCTION TO THE CUDA LBE METHOD

For the purpose of increasing simulation speed without a correspondingly large increase in cost, we have developed a multi-thread parallel lattice Boltzmann equation model based on the Compute Unified Device Architecture (CUDA). CUDA is a new hardware and software architecture for issuing and managing computations on a graphic processor unit (GPU) as a data-

parallel computing device [43]. With multiple cores driven by very high memory bandwidth, today's GPUs offer incredible resources for both graphics and non-graphics processing. Figure 5.4 [43] shows the comparison of floating-point operation capacity between GPUs and CPUs. The main reason behind GPUs' evolutionary advantage over traditional CPUs is that the GPU is specialized for graphics rendering [44, 45], which requires computing-intensive, highly parallel computation, and therefore is designed such that more transistors are devoted to data processing rather than data caching and flow control. Figure 5.5 [43] shows the scheme of structures of a CPU and GPU. The structure of a GPU obviously shows that the GPU is especially well-suited to address problems that can be expressed as data-parallel computations (the same program is executed on many data elements in parallel) with high arithmetic intensity (the ratio of arithmetic operation to memory operations).

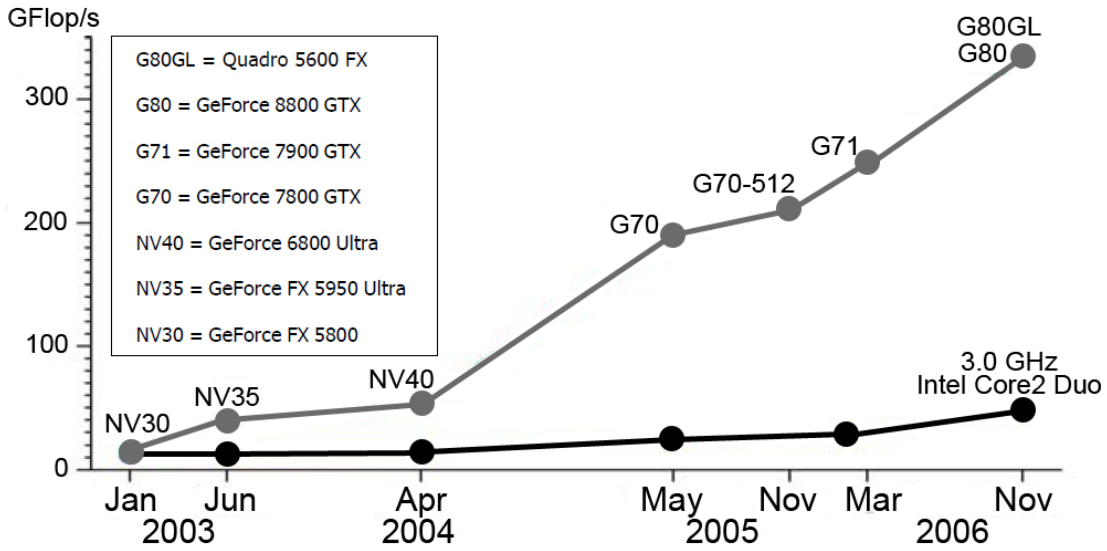
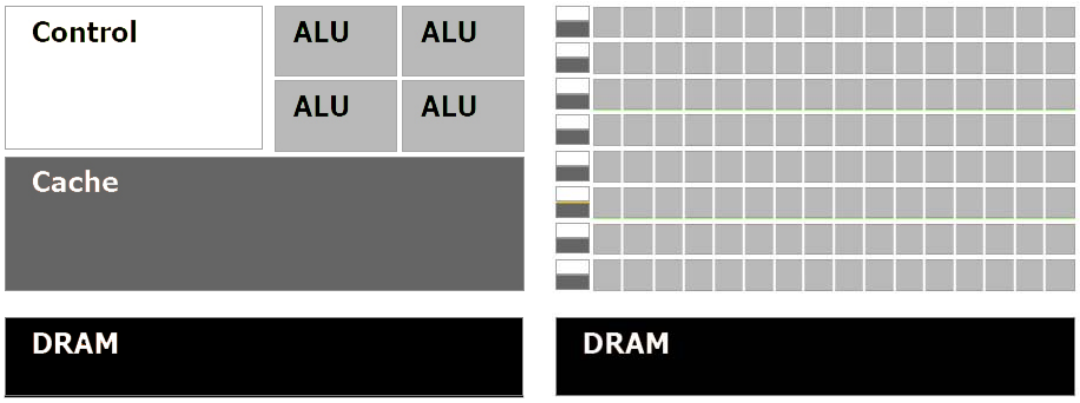


Figure 5.4: Comparison of floating-point operation capacity between GPUs and CPUs



a b

Figure 5.5: Structural organization of a CPU and a GPU, (a) CPU, (b) GPU

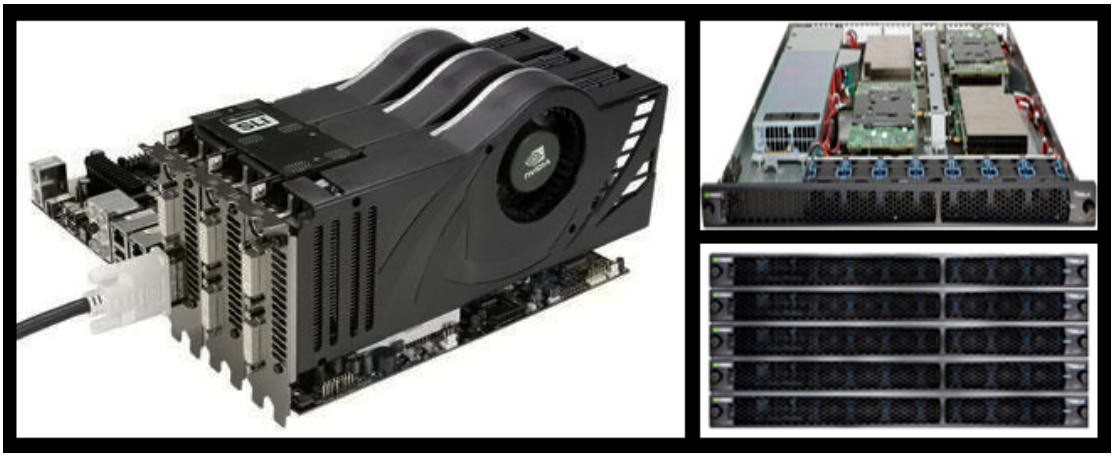
With the development of the GPU and the optimization of software for that programming environment in the last two years, CUDA has become a very powerful and popular parallel computational solution for various kinds of practical applications. More than 5 million NVIDIA 8 and higher series graphic cards were sold, and all of them are suited for CUDA operations. Table 1 shows the price of several kinds of 8 and higher series GPUs with the memory bandwidth, memory capacity available, and number of streaming processors. The prices cover a wide range. The cheapest one, the 8400 GS, which is widely used in laptops, also can offer 16 processor cores, and is very useful for the beginning research and study of CUDA. The 8800 and 9800 can offer a great speed advantage over a CPU, and that speed comparison will be shown in the subsequent result sections.

Additionally, all of these GPUs can work in parallel, which means that 2-4 graphic cards can work together by a technique called Scalable Link Interface (SLI) [46], and offer 2 to 4 times the memory space and 2-4 times the number of stream processors to satisfy the computational load requirement. All of these extension abilities are available on a common personal computer. Figure 5.6a shows an example of such system. For higher computational load

requirements, NVIDIA also created the Tesla Computing System [47], which integrated 4 GPUs in a rack mounted server case, as shown in Figure 5.6b, and this system can be further stacked for higher computing capacity.

Table 1. NVIDIA Geforce graphic cards

GPU Engine	Processor Cores	Memory Bandwidth (GB/sec)	Standard Memory Configuration (MB)	Price (Dollar)
Geforce 280 GTX	240	141.7	1024	450
Geforce 9800 GTX	128	70.4	512	160
Geforce 8800 GTX	128	86.4	768	170
Geforce 8600 GTS	32	32	256	50
Geforce 8400 GS	16	6.4	128 or 256	30



a b

Figure 5.6: (a) NVIDIA SLI system; (b) NVIDIA Tesla Computing System

As stated above, the memory bandwidth has limited the multi-thread LBE model's speed in the traditional multi-CPU system. Figure 5.7 shows the comparison of memory bandwidth of different computer systems and GPU systems. Most desktop systems and workstations can only offer less than 20 GB/s of memory bandwidth, while the GPU system can easily offer more than 50 GB/s of memory bandwidth, and the current high-end graphic card, NVIDIA Geforce GTX280, has 141.7 GB/s of memory bandwidth. The latest of NVIDIA's graphic cards is the GT300 (named "Fermi") and can offer 256GB/s bandwidth. This was demonstrated in June 2009, and it will be available on the market in the spring of 2010. Additionally, CUDA features a parallel data cache or on-chip shared memory with very fast general read and write access, so applications can take advantage of it by minimizing overfetch and round-trips to DRAM (the memory on the graphic card), and therefore becoming less dependent on DRAM memory bandwidth. Overfetch is a side effect of prefetch and tends to waste bandwidth on the shared bus. In modern electronic signal techniques, the bridge device may transmit several read requests before receiving the first read return. This behavior is known as prefetch and tends to improve the performance of the device. However, due to prefetch, a read return packet may arrive after it has indicated that it will no longer accept returned data, which is called overfetch. The usage of on-chip shared memory will be explained in detail in the following sections.

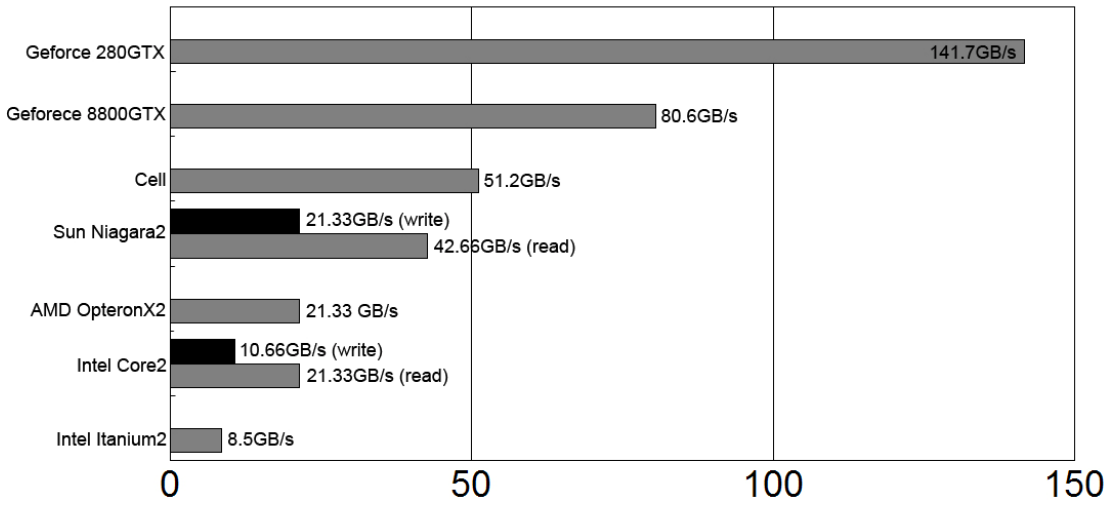


Figure 5.7: Comparison of memory bandwidth for different system

5.3 INTRODUCTION TO THE CUDA PROGRAMMING STRATEGY FOR THE LBE METHOD

Since device memory is of much higher latency and lower bandwidth than on-chip memory, device memory accesses should be minimized when implementing the LBE model. A typical programming pattern is shown below:

1. Load data from device memory to shared memory,

2. Process the data in shared memory, and

3. Write the results back to device memory.

For our computational fluid dynamics problems, the programming strategy is shown in Figure 5.8, and can be described as follows:

1. Set initial value on all grids and deposit on system memory,

2. Copy all the data in the whole simulation domain to device memory (memory on the graphic card)[i],

3. Copy one block of data to shared memory[ii],

4. Process the data in shared memory,

5. Write the results back to device memory[iii],

6. Copy the data in another block to shared memory,

7. Process the data in shared memory,

8. Write the results back to device memory,

9. When all the data in the device memory are updated, repeat step 3-6, and

10. When all of the values have converged, copy all of the data back to system memory for output into a file or display on the screen[iv].

---

[i] The example code is in appendix A1.
[ii] The example code is in appendix A2.
[iii] The example code is in appendix A3.
[iv] The example code is in appendix A4.

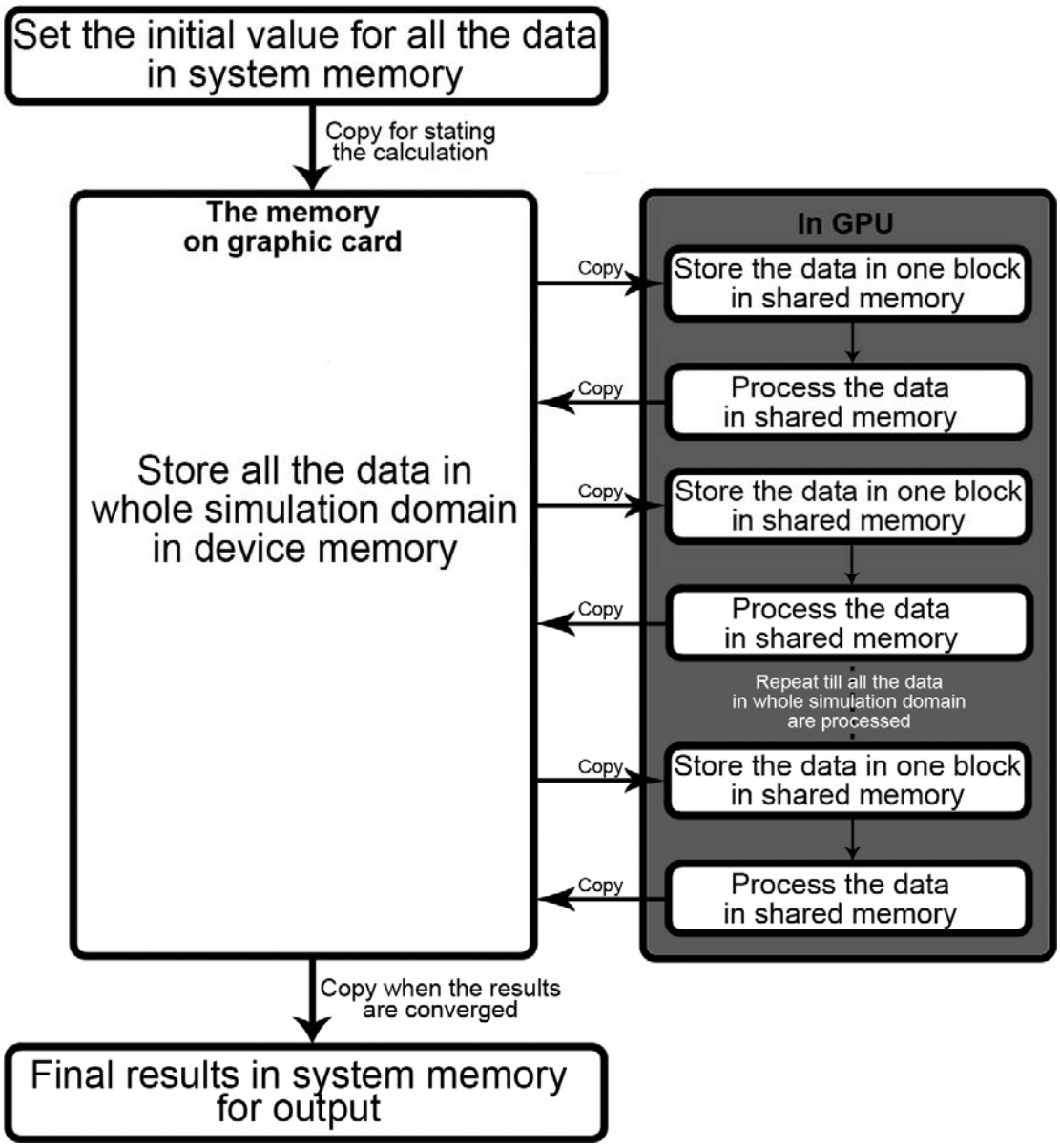


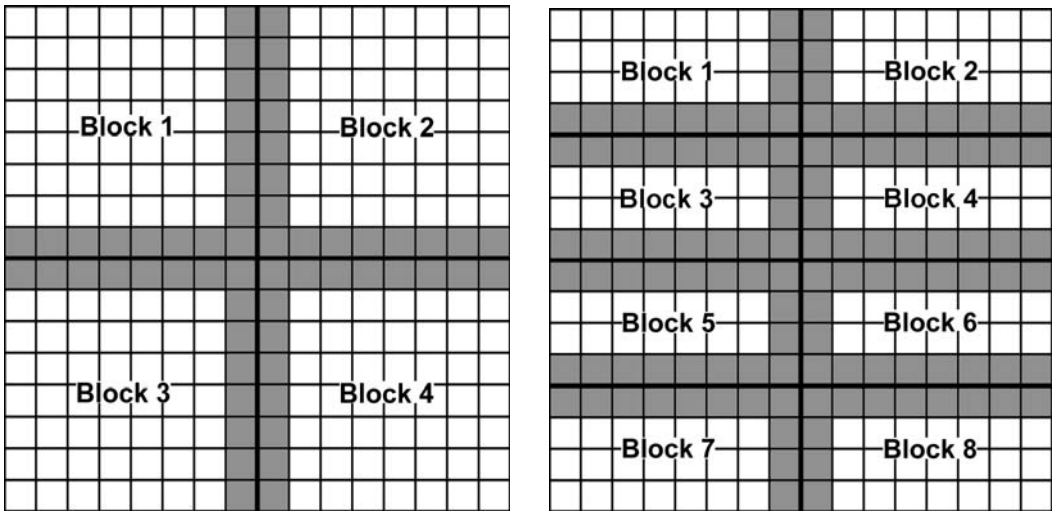
Figure 5.8: General programming strategy for numerical simulation based on CUDA

The general and traditional idea of parallel computation is to divide the whole simulation domain into several small blocks. The number of blocks depends on the number of CPUs or cores. If all of the calculation is completely local, this method is efficient and easy. The phrase "completely local" here means that updating the value on one grid only needs its own previous data, and does not need the data of other grids near by it. However, many computational models

are not completely local, such as the finite difference method. The Lattice Boltzmann equation model is not completely local as well, especially for the multi-component or multi-phase LBE model. Figure 5.9 shows a simulation domain which is divided in 4 blocks, and each thread is in charge of calculating the value in one block. Each small squire grid in figure stands for a lattice. If the simulation model is non-completely local, it is unavoidable that calculating the value on the edge of each block requires the data in other blocks. For the traditional parallel computation, the number of threads is only 2, 4, 8, or 16 at most for most practical cases. Hence, the number of interfaces of different blocks is not very large, and they can be manipulated when implementing the calculation on these interfaces. However, for a CUDA program, there may be around 100 threads, so the work load of dealing with the calculation of data on the block interfaces is too heavy, and may be even heavier than the load of updating the data on the whole simulation domain.

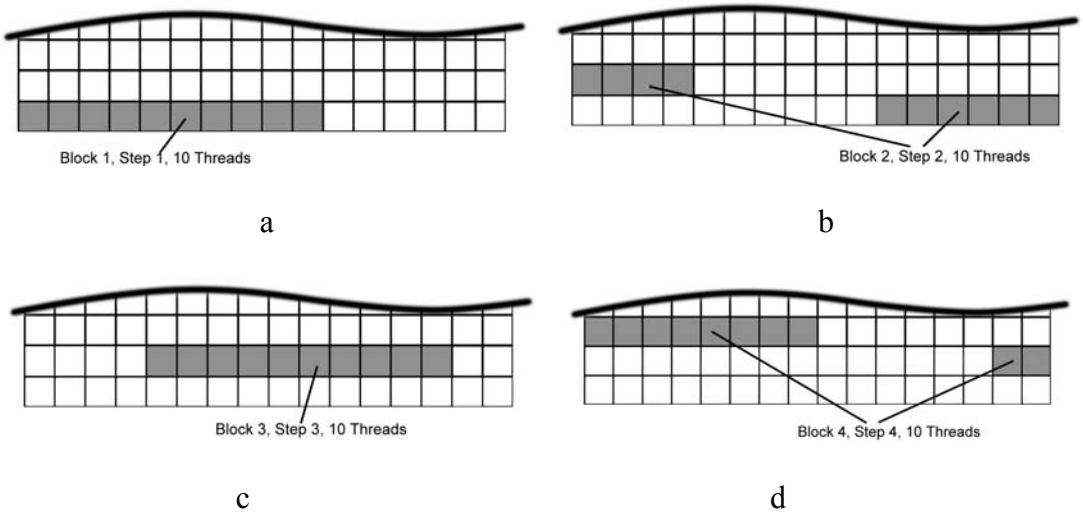
Figure 5.10 demonstrates the parallel strategy of our program, similar as figure 5.9, and each small squire grid stands for a lattice. In one calculation cycle, each thread only calculates the data on one grid, and each thread reads the data to shared memory for the calculation of one certain grid only. This method can also avoid encountering the block interface problem. In order to demonstrate this clearly, the number of threads is set to 10 in the example. In the first step, the 10 threads update the value on the grid of blocks 1 to 10 (they are in grey), as shown in Figure 5.10a, and in the second step, the 10 threads update the value on the grid of blocks 11-20 (shown in grey), as seen in Figure 5.10b. By continuing this method (Figure 5.10c and d), all of the data in the whole simulation domain can be updated. Through this method, it is easier to deal with the division of the simulation domain, especially when using a large number of threads. The work load of each thread can be more easily regulated than in the traditional division method shown in

Figure 5.9. Furthermore, the shape of each block is not required to be rectangular in our method, so it is more efficient to deliver the work load to all of the threads that are available on the hardware.



a b

Figure 5.9: Division of blocks: a, 4 blocks; b, 8 blocks



a b

c d

Figure 5.10: Demonstration of block division for CUDA

For the simplest case, single-component single-phase flow, the simulation can be divided into two steps, a collision and a streaming step, as described in equation (1.10). To clarify this separation, it can be divided into two parts that represent the collision and streaming steps:

Collision: $\tilde{f}_\alpha(\vec{x},t) = f_\alpha(\vec{x},t) - \frac{1}{\tau}\left[f_\alpha(\vec{x},t) - f_\alpha^{eq}(\vec{x},t)\right]$

Streaming: $f_\alpha(\vec{x} + \vec{e}_\alpha \Delta t, t + \Delta t) = \tilde{f}_\alpha(\vec{x},t)$

where $f_\alpha^{eq} = \rho w_\alpha \left[1 + \frac{3}{c^2}\vec{e}_\alpha \cdot \vec{u} + \frac{9}{2c^4}(\vec{e}_\alpha \cdot \vec{u})^2 - \frac{3}{2c^2}\vec{u} \cdot \vec{u}\right]$

The collision step is completely local, but the streaming step (equation (1.12)) is not. Figure 5.11 illustrates the meaning of streaming step. It can be seen as a simple copy step. Each component ($f_1, f_2, ..., f_8$) is copied to the nearby grids correspondingly after a streaming step. For the CUDA program, all of the data must be copied from the device memory to shared memory for processing, so the streaming step can be done during this copying procedure[v]. Figure 5.12 shows the process for the streaming step and collision step[vi].

[v] The example code is in appendix A5 .
[vi] The example code is in appendix A6.

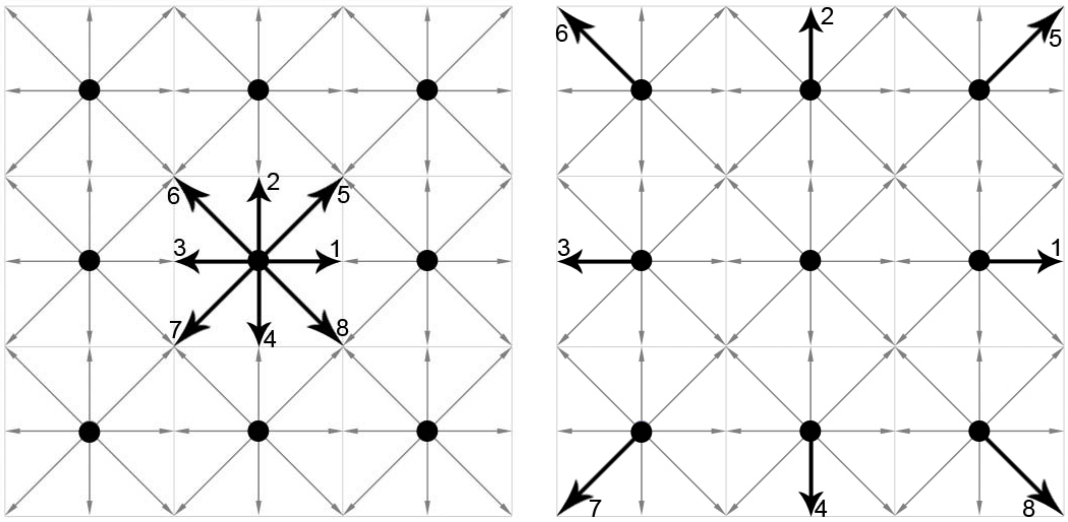


Figure 5.11: Streaming step: (a) time step 0; (b) time step 1

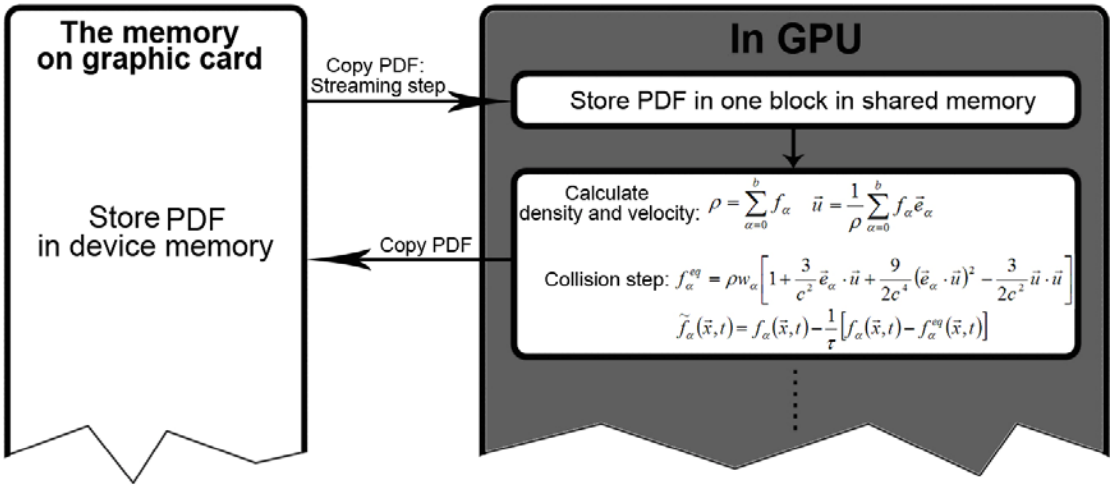


Figure 5.12: Demonstration of the streaming step and collision step based on the CUDA LBE model

For multi-phase or multi-component flow, the force between particles needs to be calculated. Equation (1.14a) and (1.14b) show that the calculation of the gradient term causes a

highly non-local computation, which depends on the data of the six nearby grids. Hence, we must copy all of the six points' data from device memory to shared memory for the calculation of this gradient term. If the gradient term in the grey grids (using 10 threads as an example) is required to be calculated, as shown in Figure 5.13, the values of the effective mass in the grids circled by the bold black line are needed to be copied from device memory to shared memory[vii]. After calculating this gradient term, the value of $\nabla\psi$ can be copied back to device memory or restored in temporary shared memory for calculating the interparticle forces by equation (1.13) and afterwards. When calculating interparticle forces on each grid (i,j,k) by equation (1.13), each thread reads the effective mass $\psi_{i,j,k}$ and gradient of effective mass $\nabla\psi_{i,j,k}$ that are stored in device memory. Hence this step is highly parallel. By substituting this interparticle force into equation (1.20) ($\vec{u}^{\,eq} = \vec{u}_i + \frac{\tau_i \vec{F}_{total,i}}{\rho_i(\vec{x})}$) for the shifted velocity, this step also depends only on local data, so it can be highly parallel as well. Figure 5.14 shows the steps of this copy process and the work flow of the calculation of the interparticle forces, modified velocity, and updated PDF[viii].

[vii] The example code is in appendix A7.
[viii] The example code is in appendix A8.

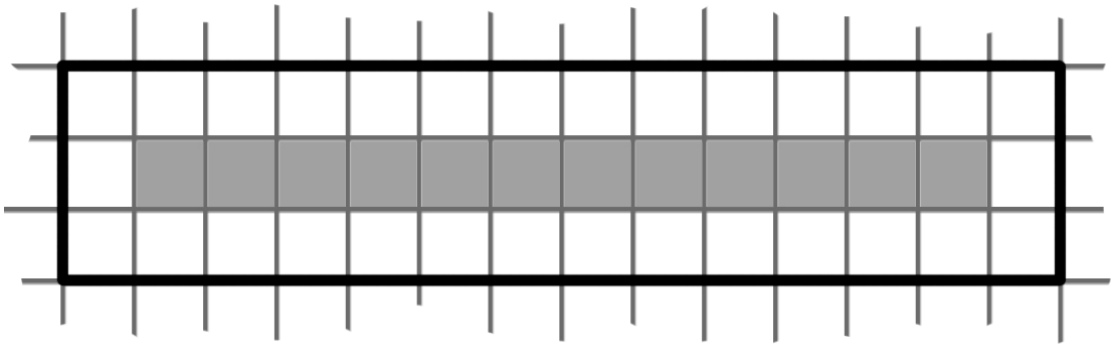


Figure 5.13: Grids for calculating the gradient term of effective mass

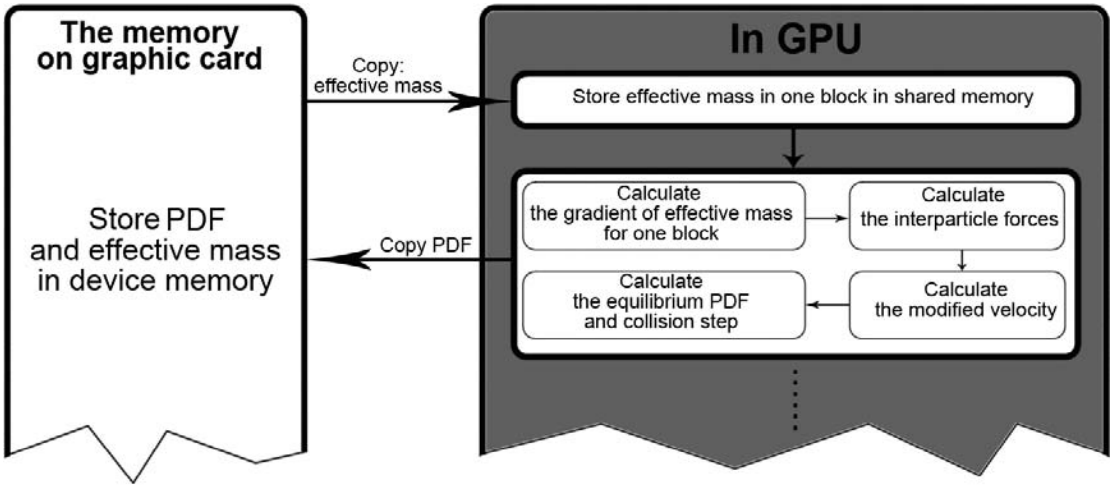


Figure 5.14: Demonstration of MCMP LBE model based on CUDA programming

Additionally, when incorporating temperature-dependent behavior into the LBE method, two sets of particle distribution functions are needed, one for the dynamic part and one for thermal part. The two parts are coupled together through the buoyancy force $\rho\vec{G} = \rho\beta g\left(T - T_0\right)\vec{j}$. The buoyancy force is then utilized in calculating the shifted velocity, which is explained in Section 1.3. Hence the programming strategy for the thermal LBE model is very similar to the single component single phase LBE model and is shown in Figure 5.15.

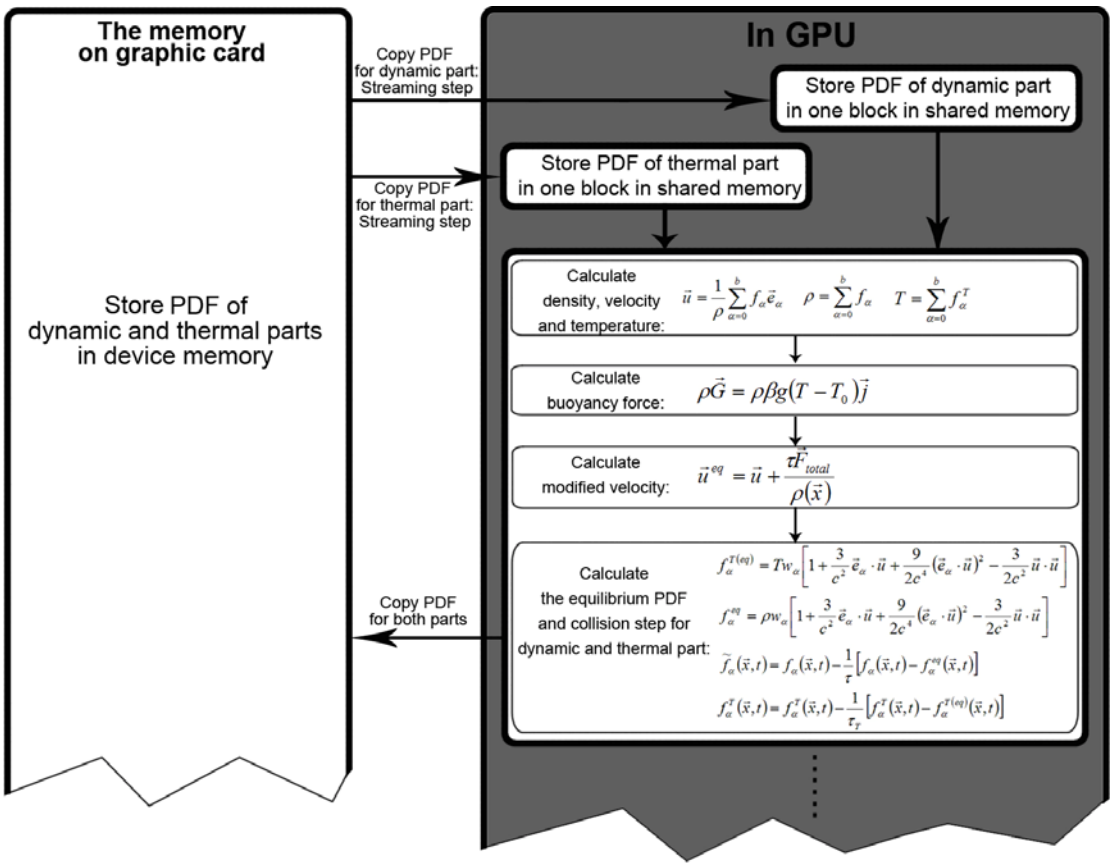


Figure 5.15: Demonstration of the Thermal LBE model based on CUDA programming

5.4 EXPERIMENTAL METHODOLOGY AND RESULTS

5.4.1 Methodology

Our tests were run on two different NVIDIA GPUs, an 8800 GTS and a 260 GTX. They represent the middle range of performance of the two generations of NVIDIA GPUs for CUDA, show the most advanced CUDA technique at their time (8800 GTS for 2006-2007 and 260 GTX

for 2008-present), and both were the most popular GPUs because of their market price. The specifications for each of these GPUs are found in Table 2.

Table 2. GPU Specifications

	Main Clock	RAM	Mem. Clock	Mem. Bus Width	Cores
8800 GTS	1.2GHz	320MB	1.8GHz	256bit	96
260 GTX	1.4GHz	896MB	2.1GHz	448bit	216

For comparison, we also performed test runs on two different Intel dual-core CPU systems, an E6600 (2.4GHz) and an E8500 (3.16GHz). We used the Compaq Fortran compiler 6.6b and the Visual Studio 2005 C++ compiler for the CPU code, and used NVCC release 2.0 for the GPU code.

5.4.2 Single Component Single Phase Flow

For fully examining the performance of the CUDA LBE program, we have run a series of tests for a 2-D Poiseuille flow case. Firstly, different size simulation domains were tested, where the numbers of lattice points for each case are 64X64, 128X128, 256X256, 512X512 and 1024X1024. Figure 5.16 shows the velocity profile from both our CUDA LBE model and the analytical solution ($u = -\frac{1}{4\eta}\frac{\Delta P}{\Delta x}\left(R^2 - r^2\right)$). All of the runs were tested on the E6600 and E8500 for the CPU code and on the 8800 GTS and 260 GTX for the GPU code, and the results are shown in Figure 5.17. The calculation time is the code spend for 100 time steps, which is in ms. From the results, we can see that the speed advantage of the GPU increases with the number of

grids. It can also be observed that the E6600 and E8500 were released on the market approximately 2 years apart. However, the speed increase of the CPU over that two years is limited (around 15%-20%), because there was no substantive change in the CPU architecture. Conversely, the speed increase between the two generations of GPUs is obvious, and is around a 100% increase, because the 260 GTX has more than double the streaming processors as the 8800 GTS, as shown in Table 2. The two GPUs (8800 GTS and 260 GTX) were installed in both the E6600 and E8500 systems for testing how CPU speed affects the GPU code. Figure 5.17 shows that the GPU code depends little on the CPU speed.

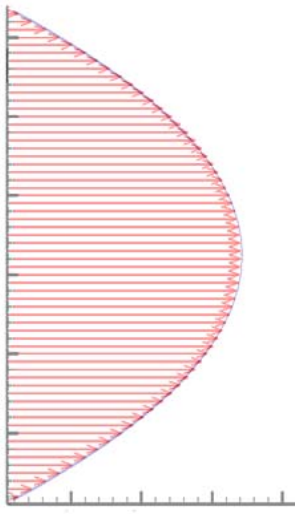


Figure 5.16: Comparison of velocity profile from CUDA LBE model (red vectors) and analytical solution (blue line)

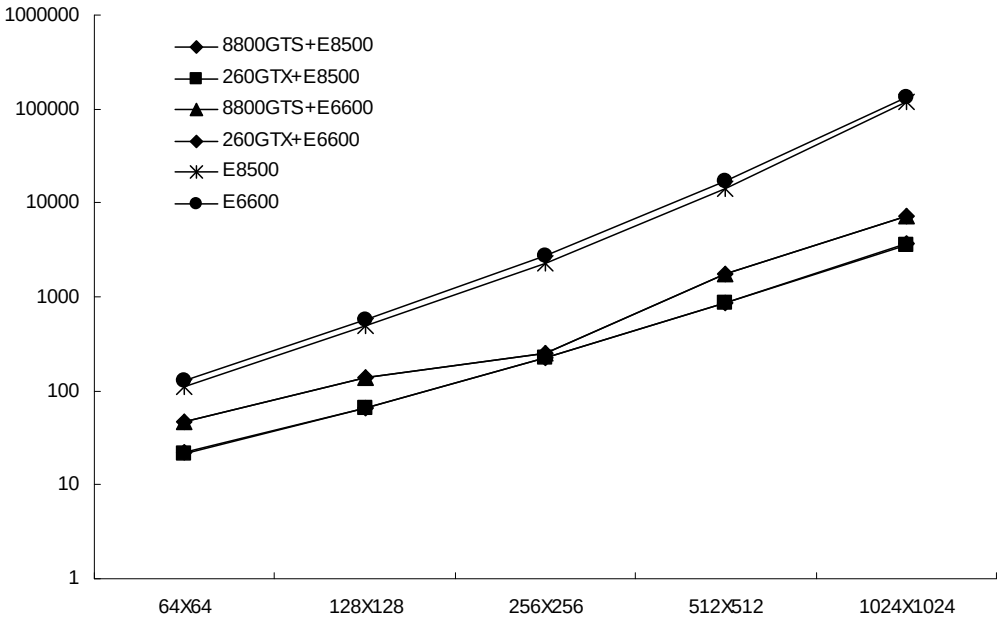


Figure 5.17: Calculation time comparison between GPUs and CPUs

We also studied how the size of the block affects the code efficiency. For a 1024X1024 2-D Poiseuille flow, we used block sizes of 2X2, 4X4, 8X8, and 16X16. From the results shown in Figure 5.18, it can be seen that the larger block size has a much higher efficiency than the small blocks, such as 2X2 and 4X4. However, it should be noted that the block size is limited by the hardware capacity, such as the size of shared memory and the total number of streaming processors in the GPUs, so in most complicated practical problems, the block size cannot be very large.

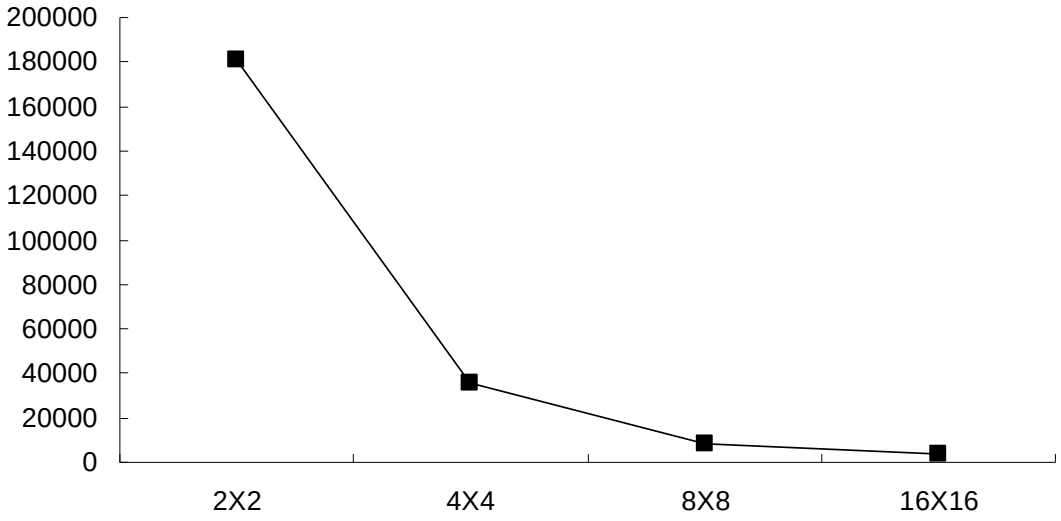
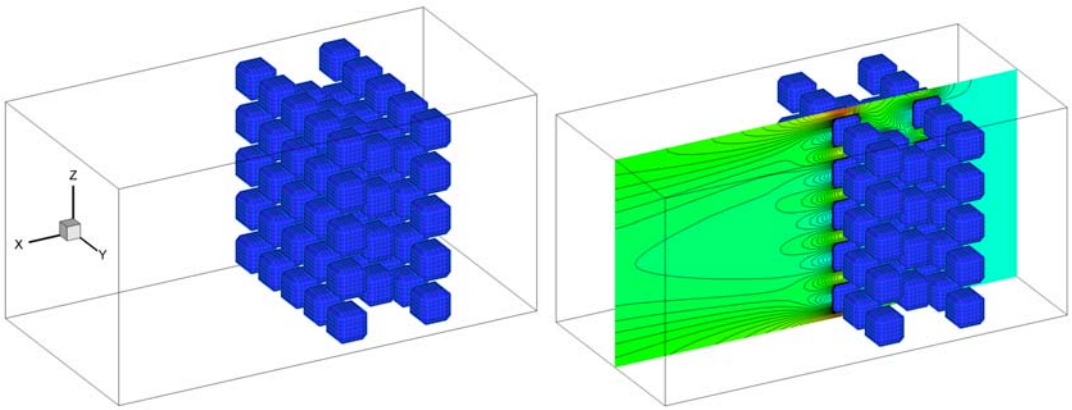


Figure 5.18: Calculation time (in ms) for different size of block

One of the advantages of the LBE method is the simplicity of the solid wall boundary treatment. Hence, objects or solid walls with a complicated geometry can be easily addressed in the LBE model. For testing the efficiency of dealing with solid wall boundary conditions, we tested two 3-D cases with complex geometry walls. Because the 8800GTS GPU has 320MB memory, which is not enough for most 3-D cases, all of our 3-D simulations were only tested on the 260 GTX.

The first case is an external flow with porous media as shown in Figure 5.19a with $50 \times 50 \times 100$ grids, where a periodical boundary condition is used for the boundaries of y=0, y=ymax, z=0, and z=zmax. A uniform inlet velocity is at x=0, and an extended boundary treatment is used for x=xmax. Figure 5.19b shows the velocity profile on a slice of y=25. Because the periodical BC is used for the upper and bottom boundaries, there is a relatively bigger space at the upper and bottom boundary that allows fluid to flow through easily, so the velocity is higher around this reign. This result coincides with the fact that fluid tends to flow to

the space that has less resistance. Our LBE model clearly simulates the velocity distribution, such as the boundary layer, around every cubic object. Figure 5.20 shows the calculation time comparison between GPUs and CPUs. All of the codes run 10,000 time steps to reach steady state.



a b

Figure 5.19: (a) Demonstration of simulation domain, (b) Velocity profile on a slice at y=25

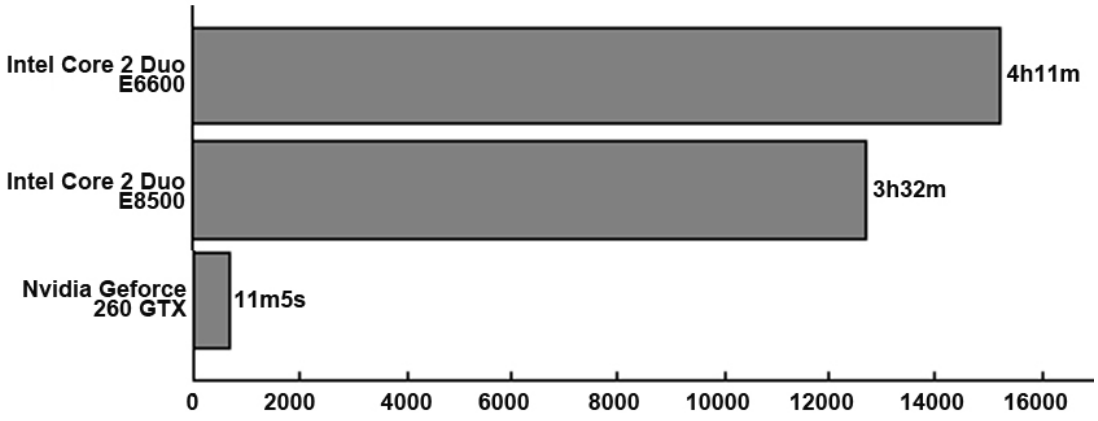
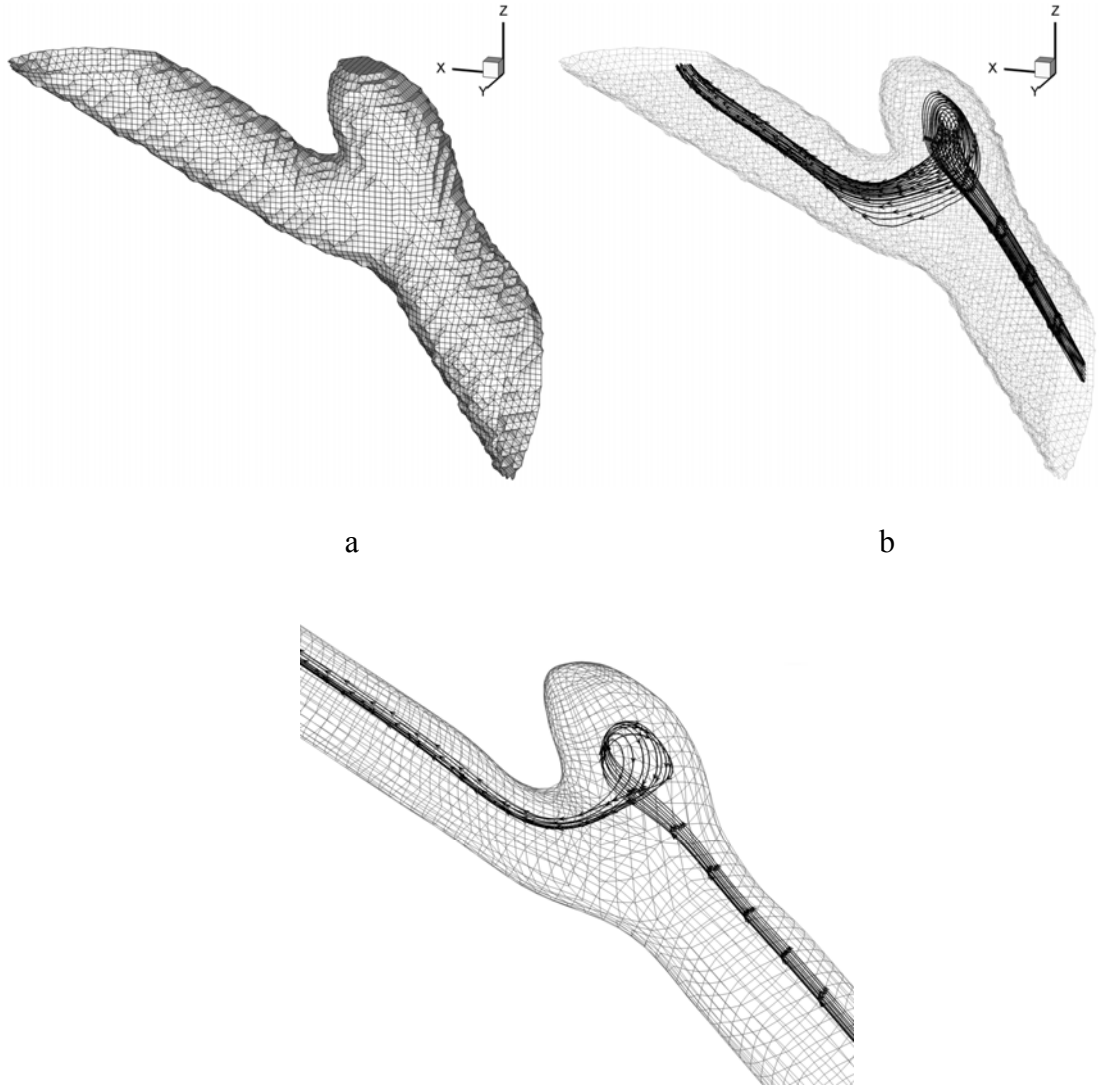


Figure 5.20: Calculation time comparison for 3-D porous media flow

The second case is internal flow in vessel networks, as shown in Figure 5.21a. Evidence has shown that changes in the hemodynamic characteristics of cerebral vessels play an important role in the initiation and development of vascular diseases [48, 49, 50, 51, 52, 53]. CFD simulation on imaging-based patient-specific models are an effective alternative for understanding the vascular blood flow characteristics as well as assessing their roles in the development of vascular diseases [54, 55, 56, 57, 58, 59]. 3D blood vessel models often involve solid walls with complicated geometries. As we discussed in the previous sections, dealing with these solid wall boundary condition is one of the strengths of the LBE model. The 3D blood vessel network also requires a great amount of lattices, so our CUDA parallel LBE program can increase calculation efficiency in this area.

For demonstration of the results and ease of comparison of the results from different methods, we choose a relatively simple blood vessel model. The model is from a section of a real blood vessel through the technique of 3D rotational angiogram (3DRA) [60, 61, 62]. This model has a section of a blood vessel with an aneurysm and includes 37,100 grids. A uniform velocity is set at inlet, and extrapolation boundary condition is set at outlet. Figure 5.21b shows the streamline in this blood vessel with a Reynolds number of around 100 from LBE model. Figure 5.21c show the streamline from ADINA's result. ADINA is a popular commercial CFD software which is based on the finite element analysis (FEA) method, and it is widely used in biological applications. The results from LBE method and ADINA are qualitatively comparable, a part of fluid flows into the aneurysm and flows out after a circulation. This circulation in the aneurysm plays an important role in studying the development of vascular diseases, and our LBE method successfully simulated and tracked this phenomena. Because the limitation of grid resolution for

LBE method, the solid wall boundary is not as smooth as the FEA one, which uses unstructured grids, the results of LBE method and ADINA cannot perfectly match.



a b

c

Figure 5.21: (a) Simulation domain for flow in vessel networks, (b) streamline in blood vessel from the LBE model, (c) streamline in blood vessel from a finite element analysis method

5.4.3 Thermal Single Component Single Phase Flow

With the recent rapid advancements in the development of advanced cooling systems, the geometry of these cooling systems has become more and more complicated. Additionally, heat pipes are increasingly being widely used in cooling systems, so the study of heat transfer in porous media and small capillary tubes has attracted much attention. With its inherent advantages for dealing with complicated solid boundary conditions, the LBE method can also be used for the simulation of these thermal fluid problems. However, because the LBE model for thermal flow requires two sets of PDFs with much more complicated boundary condition treatments compared to isothermal problems, the calculation load for the thermal LBE model is nearly three times that of the isothermal model. Therefore, most common thermal problems often use more than 24 hours of calculation time.

From our previous tests utilizing the GPU code, we found that its efficiency for logical operation is relatively low. Additionally, it is difficult to effectively parallelize logical operation for the GPU. Logical operation here represents the operation of checking if the condition is true or false. A sample of logical operation for CPU code is shown below:

```
for (i=0; i=imax; i++)
{
  if (grid(i)=1)
    {f(i)=...}
      }
```

This sample means that if the condition grid(i)=1 is true, then the program runs the operation {f(i)=…}. The operation {f(i)=…} is only applied on the point “i” that meets the condition grid(i)=1.

For comparison, a sample of logical operation for GPU is shown below.

If (grid(tid)=1)

{ f(tid)=.... }

This sample intends to achieve the same function as the CPU code. “tid” here means the identification of a thread from 0 to imax, which represent all of the threads that are running in parallel. However, in this sample, the program cannot know which thread meets the condition (grid=1) when running the operation in {f(tid)=…}. The program will simply run the operation {f(tid)=…} on all of the threads, even if only one certain thread meets the condition grid(tid)=1.

For the avoidance of logical operations in using the GPU code to determine the position and normal direction of the solid wall, we have developed a generalized series of equations (as described in Section 2.3.4 and summarized in equation (2.19)) for thermal boundary conditions, which allows us to easily parallelize the boundary condition treatment.

In order to demonstrate the efficiency of this treatment, a simulation test was conducted over a domain of $50 \times 50 \times 100$, as shown in Figure 5.22a. A periodical boundary condition is used for the boundaries of y=0, y=ymax, z=0, and z=zmax. A uniform inlet velocity is at x=0, and an extended boundary treatment is used for x=xmax. All surfaces of the solid objects have a uniform temperature=1, and the temperature of environment and inlet is set to 0.1. For examining the efficiency of the GPU code in thermal dynamics problem applications, a simulation for an external flow heat transfer case has been run on both a CPU and GPU. Figure 5.22b shows the temperature contour on a slice of y=20, and Figure 5.23 show the temperature contour on this slice at time steps 200, 400, 800 and 1200. Similar to the isothermal case in Section 5.4.2, the velocity is higher in the regions around the upper and bottom boundaries, so the convective heat transfer is also higher near this area. Hence, the temperature is higher near

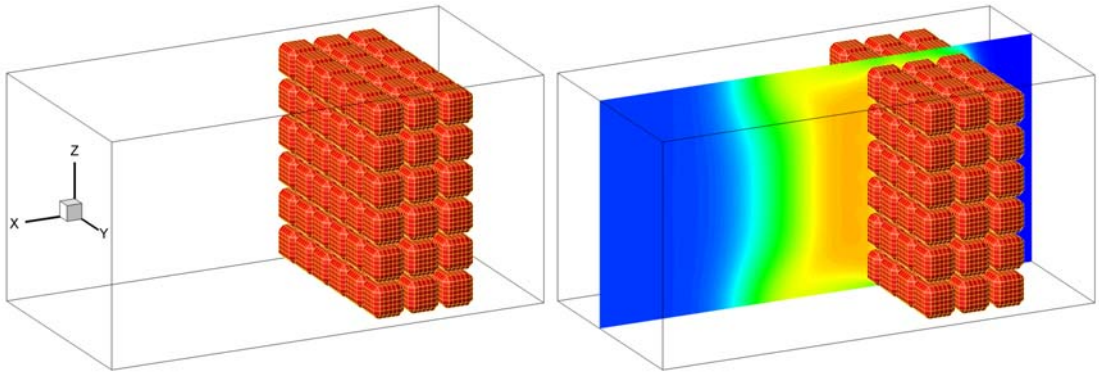
the upper and bottom boundaries at the beginning (time steps 200 and 400). The cubic solid objects keep generating heat, and the heated area keeps spreading, so the region near the center line (z=25) has the longest heated distance at time steps 800 and 1200, though the velocity is lower around this area.

The comparison of simulation time is shown in Figure 5.24. The calculation time is recorded for 1,000 time steps. The thermal LBE model requires more than double the calculation load and resources than the isothermal problems.

Table 3 shows the calculation time comparison between the thermal and isothermal problem. The efficiency of the CPU code is obviously reduced, with each time step lasting around 4.5s, which is about 3.5 times longer than the isothermal LBE model. However, the increase of calculation load does not as greatly affect the efficiency of GPU code, which only requires about 0.14s for one time step, or about 2.1 times longer than the isothermal problem. The GPU code is over 34 times faster than the CPU code.

Table 3. Calculation time comparison between isothermal and thermal LBE models

	Isothermal LBE model	Thermal LBE model
GPU code on 260GTX	0.067s	0.140s
CPU code on E8500	1.272s	4.533s



a b

Figure 5.22: (a) Demonstration of simulation domain, (b) Temperature contour on a slice at y=25

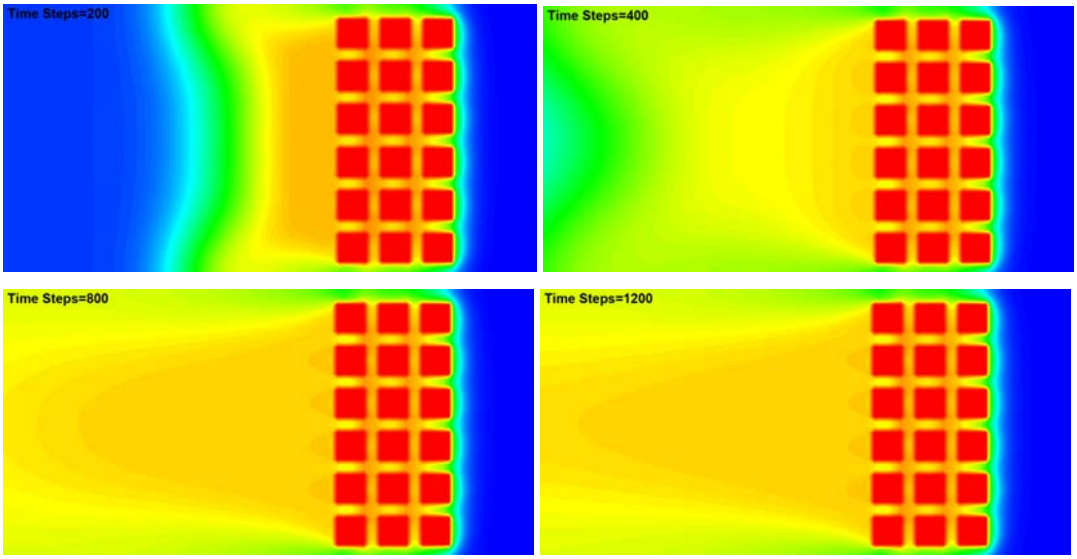


Figure 5.23: Temperature contour at the slice of y=20 for time steps 200, 400, 800 and 1200

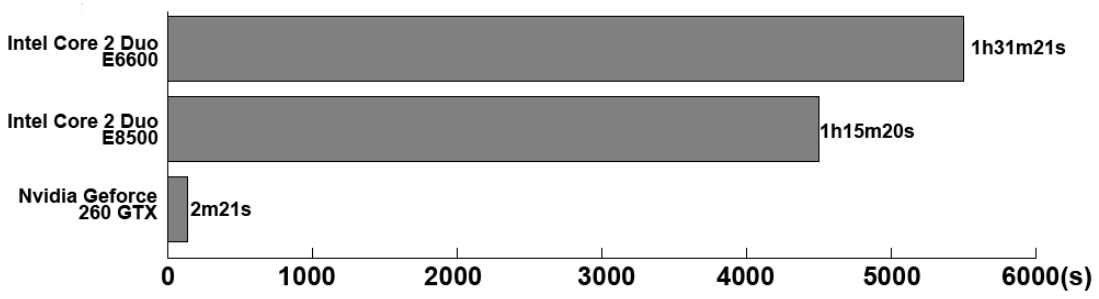


Figure 5.24: Calculation time comparison for 3-D thermal porous media flow

5.4.4 Multi-Phase Flow

The ability to create detailed simulations for multi-phase flow is one of the greatest strengths of the LBE method, as outlined in the previous sections. By applying the parallel computational method that was introduced in Section 5.1 and the boundary condition treatment from Section 1.2, 3-D multi-phase problems can be solved. In order to compare the computation speed between the CPU and GPU code, we developed two 3-D multi-phase cases.

The first simulation case is a test of the interaction between a droplet and a micro-structured surface, as shown in Figure 5.25. The simulation domain is still 100X50X50 grid points. The four vertical boundaries are periodical, and an extended boundary condition is used for upper boundary. A solid wall with convex dots is placed on the bottom of simulation domain. Both hydrophobic and hydrophilic surfaces are tested.

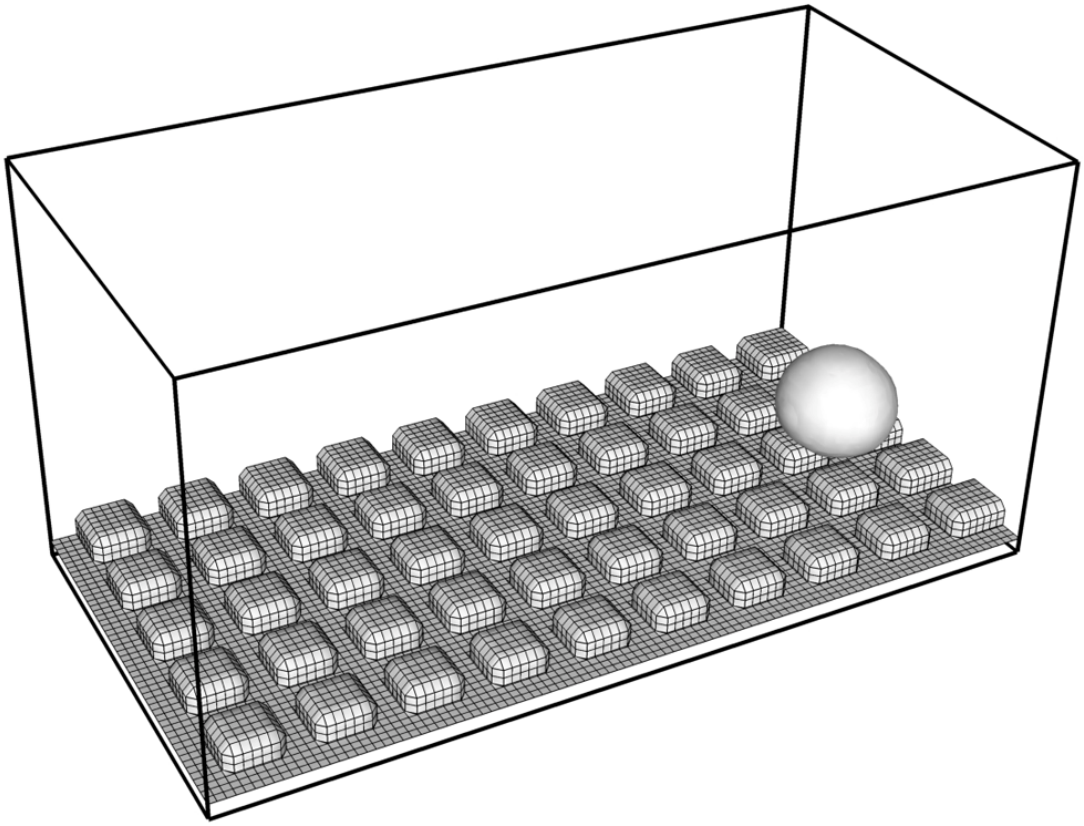
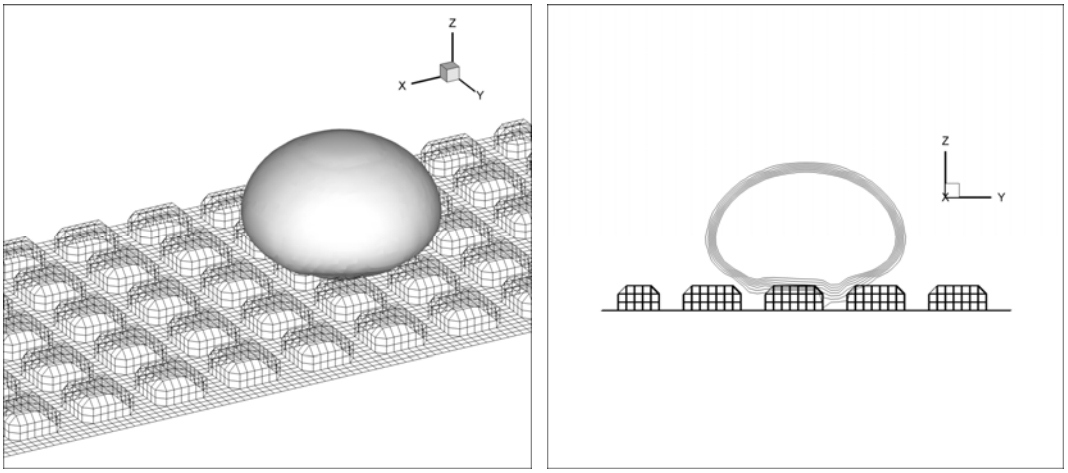


Figure 5.25: A droplet on a micro-structured solid surface

Figure 5.26 shows the hydrophobic surface case. Figure 5.26b show a cross section of the droplet. The simulation results shows that the convex shapes on the hydrophobic surface greatly reduce the contact area between the fluid droplet and solid surface, which reduces the droplet's drag force on the solid surface. This is stated by Cassie-Baxter [63] and Wenzel's theory [64], which asserts that a microstructured surface can amplify the natural tendency of the surface. For the hydrophobic surface, this phenomenon is called the Lotus effect, based on the very high water repellency exhibited by the leaves of the lotus flower [65], which is also known as the self-cleaning property from superhydrophobic micro-nanostructured surfaces discovered in the 1970s [66]. In nature, this superhydrophobic surface is a double structure, which is formed out of a characteristic epidermis and the covering waxes. The epidermis of the lotus plant possesses

papillae that are 10 to 20 μm in height and 10 to 15 μm in width, on which the so-called epicuticular waxes are imposed [67, 68, 69, 70] as shown in Figure 5.27. This can make the contact angle between a droplet and the leaf surface larger than 170 degrees, which means that the droplet's actual contact area is only 0.6% of the drop's surface. Some nanotechnologists have developed treatments, coatings, paints, roof tiles, fabrics and other surfaces that can emulate the structure of the lotus plant leaf, and which can offer the stay-dry or self-clean function that is very useful in practical and engineering applications. Our numerical model offers an efficient method for studying how the geometry or distribution of papillae affect the superhydrophobic effects. Because the number of grid limitation, the size convex dot is a little big to be exactly defined as lotus effect, but our simulation shows the characteristic of superhydrophobic micro-nanostructured surfaces.



a b

Figure 5.26: (a) liquid droplet on hydrophobic surface, (b) a cross section of droplet

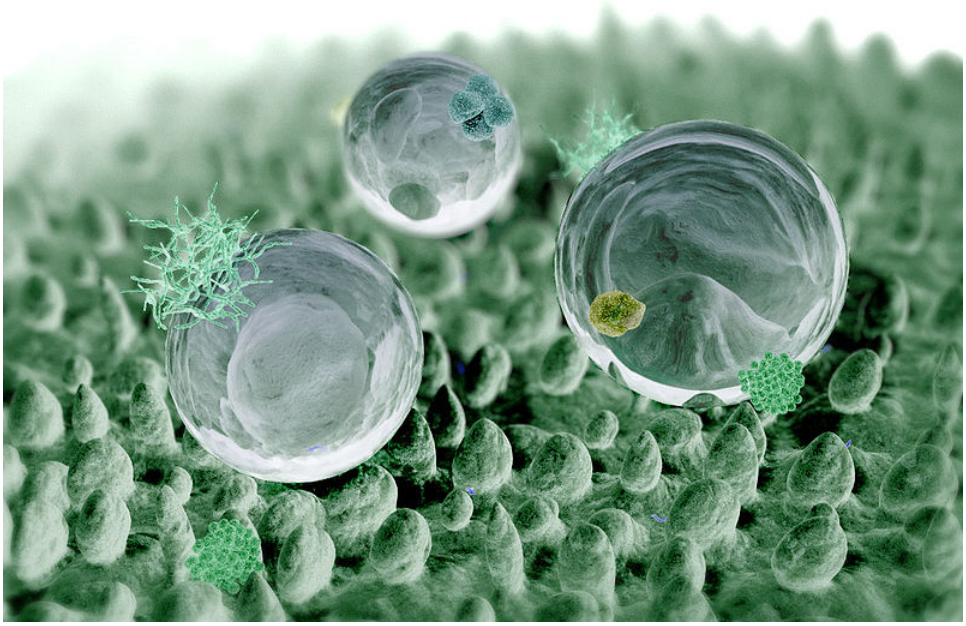
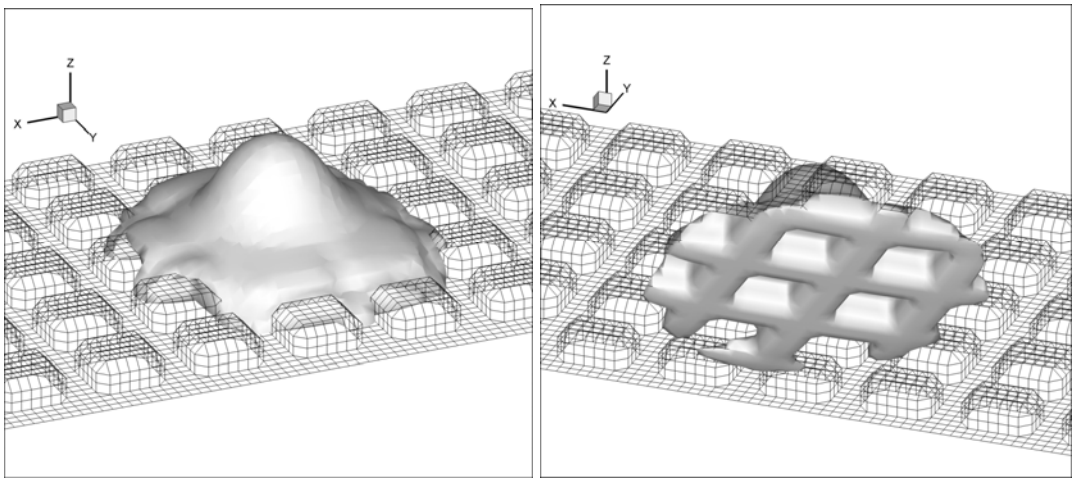


Figure 5.27: Computer graphic of a lotus leaf surface

Figure 5.28 shows the hydrophilic surface case. Because of strong attractive forces between the fluid and solid wall, the droplet tend to spread in the channels as much as possible, as shown in Figure 5.28a, and Figure 5.28b shows that the liquid already fully penetrates into the channels between convex surfaces. Figure 5.29 shows a cross section of the droplet. The micro-structured convex dots on the hydrophilic surface greatly increase the contact area between the liquid and solid, so the drag force is also increased, which also coincides with Wenzel's theory.

In order to study the computational efficiency of the GPU- versus CPU-based code for these types of structures, these simulations were completed on the E6600 and E8500 CPUs and the 260GTX GPU. Figure 5.30 shows the calculation time of 10,000 steps for the GPU and CPUs. It is clear that the GPU computation offers a clear advantage, with the CUDA parallel LBE model running at around 22 times faster than the traditional CPU code.



a b

Figure 5.28: Droplet on hydrophilic surface

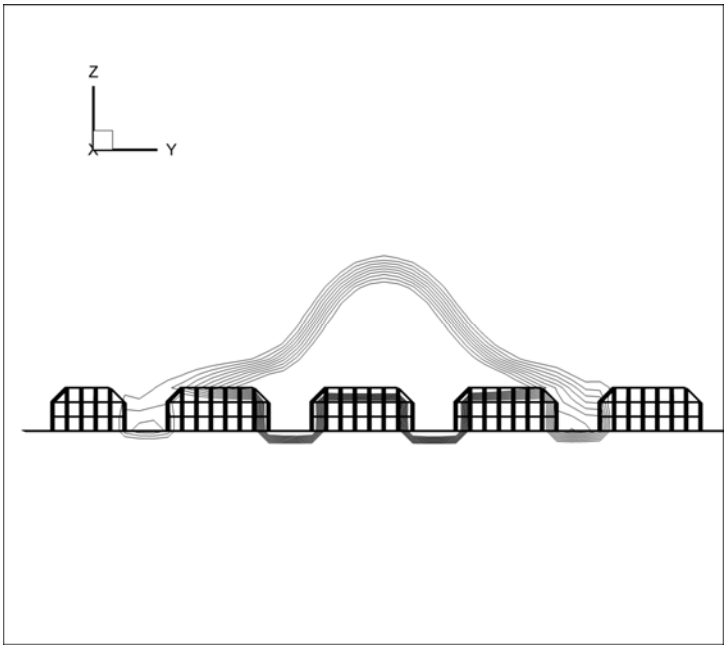


Figure 5.29: A cross section of droplet on hydrophilic surface

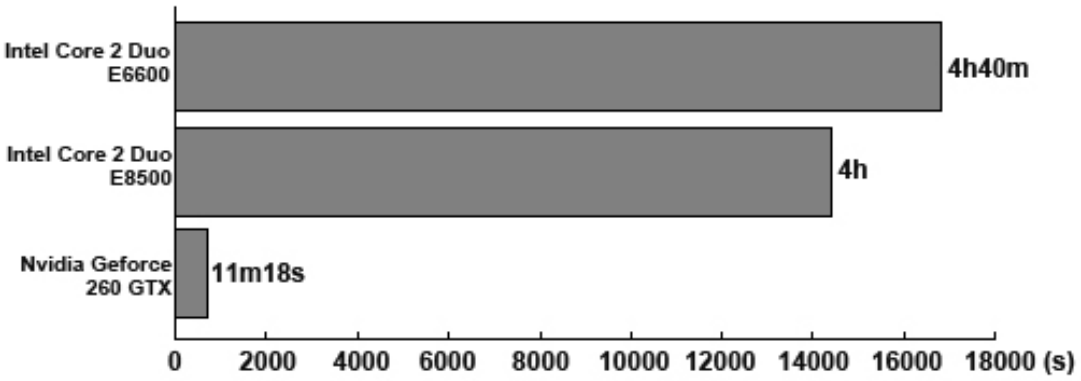
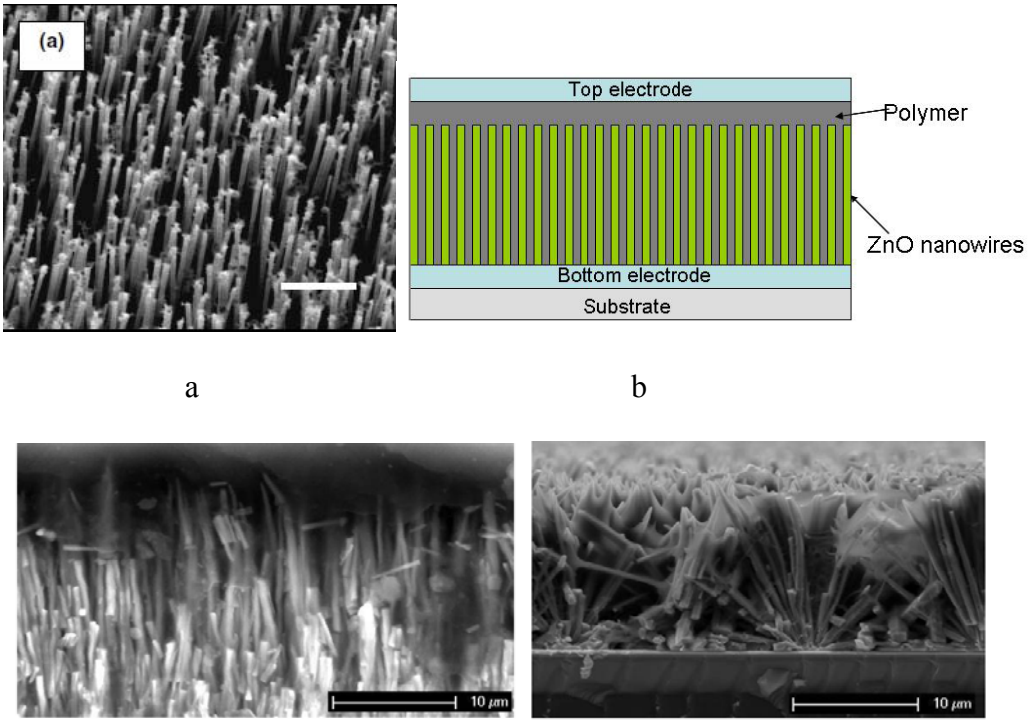


Figure 5.30: Calculation time comparison for 3-D multi-phase flow

This numerical model can be further extended for the simulation of the melding process of nano-composites. Polymer-based nanocomposites are becoming an attractive set of materials due to their multifunctionality and many potential applications [71]. These materials are expected to possess unique electric, magnetic, optical, and mechanical properties which can be significantly different from those of individual material [72]. For polymeric matrix composites, the assembly of composite materials is the key to success. However, the intrinsic van der Waals attraction among nanowires and high surface area and high aspect ratio of nanowires often prevent efficient polymer fill into the matrix, especially for some kind of materials such as ZnO. The focus of this melding process is how to insert a polymer liquid into a host nano-framework [73], such as a nano-wire array [74, 75] as shown in Figure 5.31a, as fully or as deeply as possible. Figure 5.31b shows a schematic diagram of ideal ZnO nanocomposites. Figure 5.31c shows an SEM image of ZnO nanocomposites where pure polyimide PI-2611 is used to fill the matrix, and the polyimide almost cannot insert into the nanowires array. Figure 5.31d shows another SEM image of ZnO nanocomposites where polyimide PI-2611 is diluted by solvent T9039 to 1:2 weight ratios, and the polyimide can partially penetrate into the array [72]. Our LBE model can show how the liquid flows into the nano-wire array, and how deeply the polymer

can be inserted while adjusting the attractive or repulsive interaction between the solid host and liquid polymers. Better understanding and engineering of this interaction can have many practical applications. Therefore, our second case for demonstrating the CUDA multi-phase LBE model gives an example of this problem.



a b

c d

Figure 5.31: (a) SEM image of nanowire array, (b) schematic diagram of ideal ZnO nanocomposite, (c)(d) SEM image of ZnO nanocomposites

Figure 5.32 shows the simulation domain which contains 100X100X50 grids. The small pillars which represent the nanowire array are planted on the bottom wall. The diameter of each pillar is 3 grids. The periodical boundary condition is used on the four vertical boundaries, and

gravity force is along the negative z-direction. A liquid droplet is initially put on the top of the pillars.

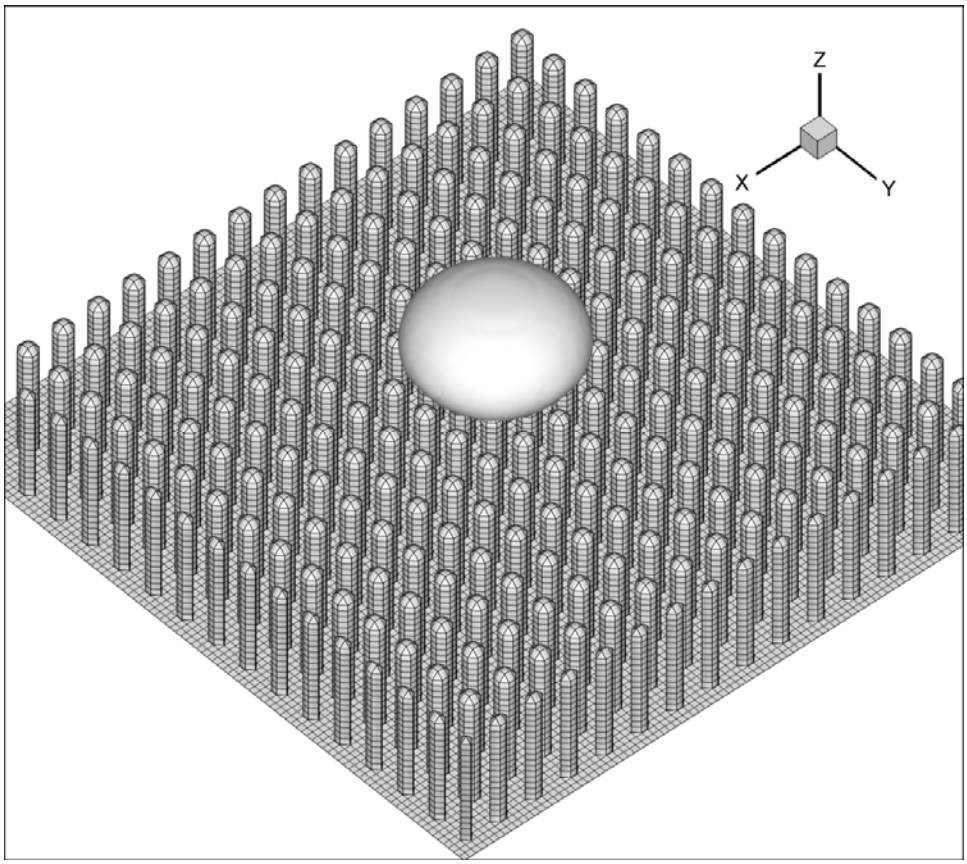
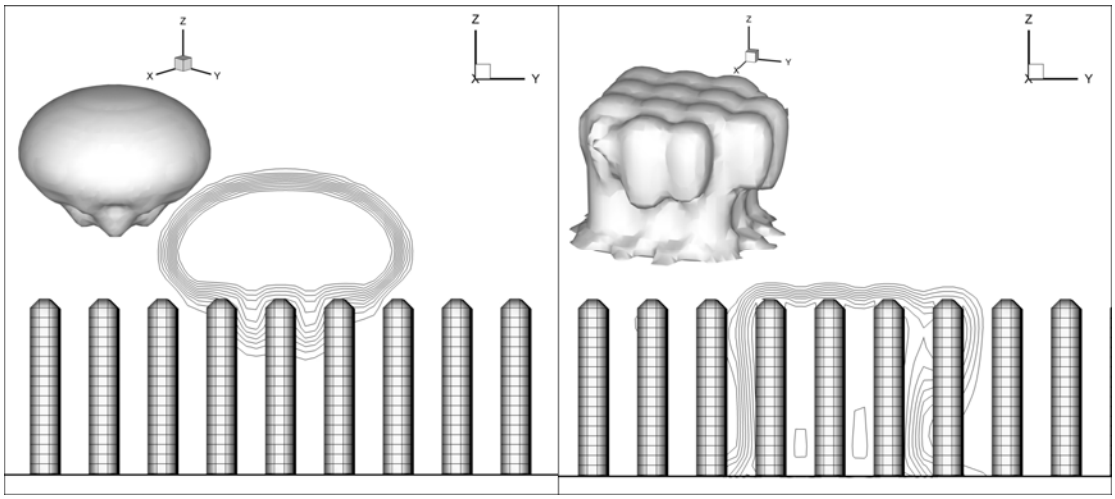


Figure 5.32: Sketch of the simulation domain

The value of G_w in equation (1.15) controls the force between the liquid and solid surface, so it can be used to adjust the melding process. When a strong repulsive force is set between the liquid and solid surface, the droplet will primarily stay on the top of the pillar array, and cannot achieve a strong penetration, as shown in Figure 5.33a. The cross section is along the center line of the droplet, and the small figure at the left upper corner shows the interface of the liquid. On the other hand, for an attractive force between the liquid and solid surface, the droplet

can flow into the space between the pillars as shown in Figure 5.33b. The value of G_w controls the strength of the attractive force, so the droplet can be partially or fully inserted into the pillar array.

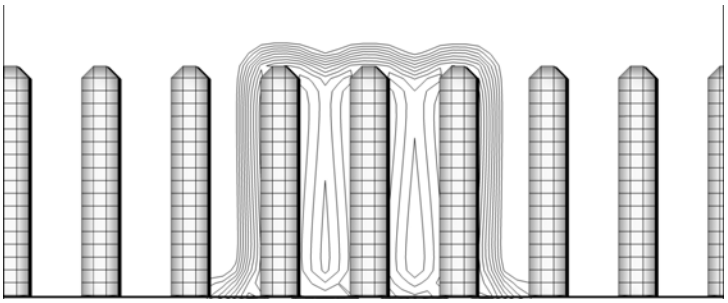


a b

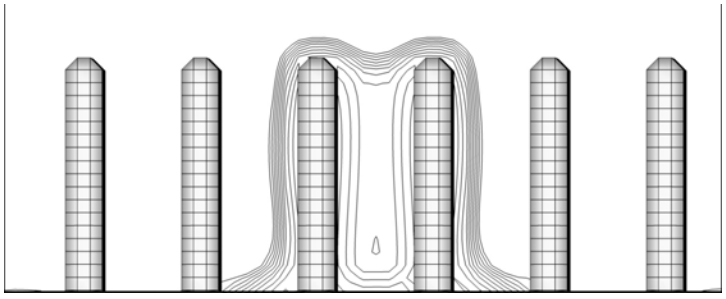
Figure 5.33: (a) Cross section and liquid interface of droplet for repulsive interaction, (b) cross section and liquid interface of droplet for attractive interaction

We also tested how the distance between pillars affects this process. Our test model is obviously ideal compared to a real nanowire array. The distribution of an actual nanowire array is much more complicated. The distance between nanowires and the length of nanowires vary, and nanowires may also not be perfectly orthogonal with the bottom wall. Additionally, some of the nanowires may crossover, and so on. Obviously, all of these variations would affect the melding process. Figure 5.34a and Figure 5.34b show the results for different distances between pillars. Figure 5.34c shows a result for pillar array with varied length. The bigger space between pillars causes the penetrating process to occur more easily. Hence, our numerical model shows

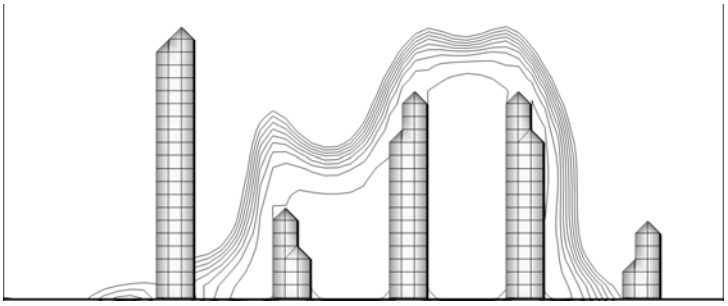
the capacity to adapt complex solid boundary conditions, and also offers a good tool for people to study how each element (such as the length of the nanowires, distribution of the nanowires, and the force between the polymer and nanowires) affects the melding process.



a



b



c

Figure 5.34: Cross section of droplet for different distribution of pillars, (a) distance between pillars is 4 grids, (b) distance between pillars is 8 grids, (c) pillars have different lengths

5.5 CONCLUSIONS AND DISCUSSION

NVIDIA's CUDA technique offers a great choice for parallel computation using the LBE method. From the test results in Chapter 5, it can be seen that the GPU code can offer 10-30 times faster performance over the CPU code. Furthermore, this impressive increase in efficiency is accomplished using a common and inexpensive (under $200) GPU. Additionally, the stability of the system for a traditional CPU-based workstation is critical, since the system often needs to run for several days continuously in a full work load. Hence, these kinds of systems require high-maintenance standard environments, such as cooling systems, a power supply system, and a back up system. All of these requirements increase the cost of a simulation. GPU-based simulations do not require the same critical system stability, because a GPU only spends 1/30 to 1/10 time of a CPU for the same simulation problem. This means that the GPU code can be solved in only 30 minutes to 4 hours for most 3-D LBE method problems, even if multi-phase or thermal dynamics is involved.

From the results in Section 5.4, it can be seen that the CUDA technique shows great adaptability to the LBE model. It is not only used for simple single phase and isothermal flow problems, but also shows obvious efficiency advantages for thermal and multi-phase cases. Our study also shows that with an increase of the calculation load, the efficiency of the GPU LBE model also increases. For single phase isothermal flow problems, the GPU is around 15 times faster than the CPU model. For the high calculation load cases, such as multi-phase and thermal flow problems, the GPU LBE model is about 20 and 34 times faster, respectively. Because we assume our GPU code does not affect the accuracy of LBE method and we only parallelized the original LBE code, we did not pay too much attention on the comparison of the accuracy, especially for the thermal and multi-phase cases.

However, the weakness of GPU model is also obvious. A GPU's memory space is smaller than the system memory. Hence, our current GPU LBE model can only deal with cases with less than 1 million grids, and for thermal or multi-phase problem, the limit on the number of grids is even lower. However, with the development of transistor techniques, the cost of memory is continually decreasing, so it is possible for more and more memory to be integrated onto a graphics card. Currently, some graphics cards with 2GB of memory are available on the market. Furthermore, as stated in Section 5.2, the SLI technique that allows multiple graphics cards working parallel can also provide multiple time available graphic memory space versus a single card.

6.0 CONCLUSIONS AND FUTURE WORK

6.1 MAJOR ACCOMPLISHMENTS

In this dissertation, we have made several contributions to the study of LBE method and parallel computation. The following is a summary of our major accomplishments.

1. Development of a new mass conserving boundary condition for the LBE method

We proposed a second-order accurate mass conserving boundary condition for the LBE method. Several practical cases involving curved solid boundaries are tested for examining the accuracy and robustness of the proposed BC. Compared with the FH and MLS BCs, our new BC will not result in the constant mass leakage seen in other BCs in some special cases, and accuracy is kept at the same level as MLS BCs.

2. Development of a new LBE method for multi-component multi-phase flow with high density ratios

Originally, the SC model was proposed for multi-component and multi-phase system. In their multi-component approach, the different components were included in the model by introduction different sets of PDFs. We found that the biggest problem for this approach is the density ratio cannot be high. The highest density ratio it can achieve is only about 2.0. If the density ratio is higher than that, the simulation will fail. However, in common gas-liquid flows, like an air-water system, the density ratio can be higher than 1,000. In order to simulate such systems, we noticed

the original model's lack of attention paid to the interaction between molecules of the same component ($\vec{F}_{i,i}$). We found that by increasing the fraction of $\vec{F}_{i,i}$, the density ration can increase substantially. All of the results in Chapter 3 clearly demonstrate our MCMP LBE model can handle the two-component system with density ratio of up to approximately 1,000:1, which makes the simulation of most air-water systems possible. Additionally, the spurious current in our model is very small (0.032) in the case with a density ratio of around 1,000, which is smaller than the average level (0.05 to 0.1) of most other SCMP or MCMP LBE model for much lower density ratios.

3. Application of the Compute Unified Device Architecture (CUDA) to the LBE method

Parallel computation is always an obvious way to increase the numerical simulation speed, but many factors limit the application of traditional multi-thread computation, such as the difficulty of the programming environment and costs of the hardware. Besides that, some numerical methods may not benefit much from parallel computation on a traditional multi-CPU system. The LBE method is one of these numerical models. Hence, we developed a series of parallelized LBE models based on the CUDA technique, which is a new hardware and software architecture for issuing and managing computations on a graphic processor unit (GPU) as a data-parallel computing device. In Chapter 5, we introduced the CUDA LBE models for simple single-phase flow, multi-phase flow, and thermal flow, as well as the most common and necessary BC treatments. Through a basic benchmark test and several practical applications, which include blood vessel internal flow, porous media flow, thermal flow, and multi-phase flow on a complex solid boundary wall, our CUDA LBE models show a great increase in efficiency, operating 10-30 times faster than single-thread CPU program. Furthermore, this impressive increase in efficiency is accomplished using a common and inexpensive (under $200) GPU.

6.2 FUTURE WORK

Several possible future tasks are listed below. We have conducted preliminary research in some of these areas.

1. Thermal multi-component multi-phase flow with high density ratio

In industrial problems, heat transfer is almost always involved in multi-component multi-phase systems, such as air-water systems. In Chapter 4, we introduced an isothermal MCMP flow with a high density ratio, and it is highly possible that this can be extended to solve thermal problems. A simple way is adding another set of PDFs for temperature besides the two sets of PDFs for the two components. We have already conducted some basic tests for this method, but it is not currently accurate enough for the application because the result differs significantly from the real physical phenomena. One obvious reason is that the two components have different thermal properties and behavior, especially for two components with a high density difference, and one set of PDFs cannot simulate the different heat transfer phenomena for the two components at the same time. In order to solve this problem, one more additional set of PDFs may be needed, which means that two sets of PDFs will be used for temperature and to simulate the thermal property and behavior of the two different components separately. However, two sets of PDFs for temperature will cause another problem, namely how to determine the unique temperature at each point from two sets of PDFs. Studying the interaction of the two sets of PDFs for temperature will be the critical step for developing a thermal LBE model for MCMP flow with a high density ratio.

2. Further application of the CUDA technique for the LBE method

As introduced in Chapter 5, the CUDA technique was developed by NVIDIA. While it is not a standard programming platform for unified computation, it is currently very popular. However,

all of the CUDA code can only run on NVIDIA's products. The competition between different companies in the unified computation area means that this kind of technique will continually undergo significant changes. Fortunately, a standard unified computation platform, Open Computing Language (OpenCL), is under development by the organization of companies of such as Apple, AMD, NVIDIA, IBM and so on. OpenCL will offer a friendlier user interface, and programmers will not need to pay as much attention to the hardware usage and control as when writing CUDA code, such as control reading, writing, and storing of data on different kinds of memory. Furthermore, an OpenCL program can run on the any unified computing hardware, no matter what manufacturer it is from. Hence, rewriting the CUDA LBE model to the OpenCL platform will help the further development of parallel LBE methods on a unified computing environment. The easier interface will also simplify the difficulty of the programming, which may help in proposing more complicated models for solving practical engineering and industrial problems. Although CUDA provides a great efficiency increase, the accuracy problem can be a concern. Most current CUDA code or other unified computing code are based on single precision calculation, because the hardware can only support single precision, which may cause higher accumulative error than double precision calculations. From our previous work, we also noticed that the higher accumulative error may cause our CUDA code to not be as stable as the CPU code. Fortunately, NVIDIA, AMD, and Intel already claim that their new generation of unified capable products will have the ability to handle double precision calculation. Hence, developing a double precision CUDA or OpenCL LBE model is a prospective direction.

APPENDIX A

EXAMPLE OF CUDA CODES

A.1 COPY DATA FROM SYSTEM MEMORY TO DEVICE MEMORY

```
// copy host memory to device memory
float h_f1[60][60];
float* d_f1;
CUDA_SAFE_CALL( cudaMalloc( (void**) &d_f1, mem_size));
CUDA_SAFE_CALL( cudaMemcpy( d_f1, h_f1,
mem_size,cudaMemcpyHostToDevice) );
```

A.2 COPY DATA FROM DEVICE MEMORY TO SHARED MEMORY

```
// copy device memory to shared memory
#define sf1( index)      CUT_BANK_CHECKER(sf1, index)
__shared__  float sf1[2][30];
const unsigned int tidx = threadIdx.x+1;
```

```
const unsigned int tidy = threadIdx.y;
for (iblock=0; iblock<=1; iblock++)
 {
for (jblock=0; jblock<=28; jblock++)
  {
   sf1[tidy][tidx]=g_f1[60+jblock*120+iblock*29+tidy*60+tidx];
    }
   }
```

A.3 COPY DATA FROM SHARED MEMORY TO DEVICE MEMORY

```
// copy shared memory to device memory
#define sf1( index)      CUT_BANK_CHECKER(sf1, index)
__shared__  float sf1[2][30];
const unsigned int tidx = threadIdx.x+1;
const unsigned int tidy = threadIdx.y;
for (iblock=0; iblock<=1; iblock++)
 {
  for (jblock=0; jblock<=28; jblock++)
  {
  g_ftemp1[60+jblock*120+iblock*29+tidy*60+tidx]=sf1[tidy][tidx];
    }
 }
```

A.4 COPY DATA FROM DEVICE MEMORY TO SYSTEM MEMORY

```
// copy device memory to host memory
CUDA_SAFE_CALL( cudaMemcpy( h_ftemp1, d_ftemp1, mem_size,cudaMemcpyDeviceToHost) );
```

A.5 STREAMING STEP

```
for (iblock=0; iblock<=1; iblock++)
 {
  for (jblock=0; jblock<=28; jblock++)
  {
 sf1[tidy][tidx]=g_ftemp1[60+jblock*120+iblock*29+tidy*60+tidx-1];
 sf2[tidy][tidx]=g_ftemp2[60+jblock*120+iblock*29+(tidy-1)*60+tidx];
 sf3[tidy][tidx]=g_ftemp3[60+jblock*120+iblock*29+tidy*60+tidx+1];
 sf4[tidy][tidx]=g_ftemp4[60+jblock*120+iblock*29+(tidy+1)*60+tidx];
 sf5[tidy][tidx]=g_ftemp5[60+jblock*120+iblock*29+(tidy-1)*60+tidx-1];
 sf6[tidy][tidx]=g_ftemp6[60+jblock*120+iblock*29+(tidy-1)*60+tidx+1];
 sf7[tidy][tidx]=g_ftemp7[60+jblock*120+iblock*29+(tidy+1)*60+tidx+1];
 sf8[tidy][tidx]=g_ftemp8[60+jblock*120+iblock*29+(tidy+1)*60+tidx-1];
 sf0[tidy][tidx]=g_ftemp0[60+jblock*120+iblock*29+tidy*60+tidx];
   }
  }
```

A.6 CALCULATE DENSITY, VELOCITY AND COLLISION STEP

```
//density
srho[tidy][tidx]=sf1[tidy][tidx]+sf2[tidy][tidx]+sf3[tidy][tidx]+sf4[tidy][tidx]+sf5[tidy][ti
dx]+sf6[tidy][tidx]+sf7[tidy][tidx]+sf8[tidy][tidx]+sf0[tidy][tidx];
//velocity
sux[tidy][tidx]=(sf1[tidy][tidx]-sf3[tidy][tidx]+sf5[tidy][tidx]-sf6[tidy][tidx]-
sf7[tidy][tidx]+sf8[tidy][tidx])/srho[tidy][tidx];
suy[tidy][tidx]=(sf2[tidy][tidx]-sf4[tidy][tidx]+sf5[tidy][tidx]+sf6[tidy][tidx]-
sf7[tidy][tidx]-sf8[tidy][tidx])/srho[tidy][tidx];
//equilibriumPDF
sf1[tidy][tidx]=srho[tidy][tidx]/9.0*(1.0+3.0*sux[tidy][tidx]+4.5*sux[tidy][tidx]*sux[tid
y][tidx]-1.5*(sux[tidy][tidx]*sux[tidy][tidx]+suy[tidy][tidx]*suy[tidy][tidx]));
//collision step fro tao=1
g_ftemp1[60+jblock*120+iblock*29+tidy*60+tidx]=sf1[tidy][tidx];
```

A.7 COPY EFFECTIVE MASS FROM DEVICE MEMORY TO SHARED MEMORY

```
for (iblock=0; iblock<=1; iblock++)
 {
  for (jblock=0; jblock<=28; jblock++)
  {
 sfi1[tidy][tidx]=g_fi[60+jblock*120+iblock*29+tidy*60+tidx-1];
```

```
sfi2[tidy][tidx]=g_fi[60+jblock*120+iblock*29+(tidy-1)*60+tidx];
sfi3[tidy][tidx]=g_fi[60+jblock*120+iblock*29+tidy*60+tidx+1];
sfi4[tidy][tidx]=g_fi[60+jblock*120+iblock*29+(tidy+1)*60+tidx];
sfi5[tidy][tidx]=g_fi[60+jblock*120+iblock*29+(tidy-1)*60+tidx-1];
sfi6[tidy][tidx]=g_fi[60+jblock*120+iblock*29+(tidy-1)*60+tidx+1];
sfi7[tidy][tidx]=g_fi[60+jblock*120+iblock*29+(tidy+1)*60+tidx+1];
sfi8[tidy][tidx]=g_fi[60+jblock*120+iblock*29+(tidy+1)*60+tidx-1];
sfi[tidy][tidx]=g_fi[60+jblock*120+iblock*29+tidy*60+tidx];
 }
}
```

A.8 CALCULATE GRADIENT OF EFFECTIVE MASS, INTERPARTICLE FORCES AND MODIFIED VELOCITY

```
//gradient of effective mess and interparticle forces
sforcex[tidy][tidx]=-sfi[tidy][tidx]*gf*(sfi3[tidy][tidx]-sfi1[tidy][tidx]-
0.25*sfi5[tidy][tidx]+0.25*sfi6[tidy][tidx]-0.25*sfi8[tidy][tidx]+0.25*sfi7[tidy][tidx]);
sforcey[tidy][tidx]=-sfi[tidy][tidx]*gf*(sfi4[tidy][tidx]-sfi2[tidy][tidx]-
0.25*sfi5[tidy][tidx]-0.25*sfi6[tidy][tidx]+0.25*sfi8[tidy][tidx]+0.25*sfi7[tidy][tidx]);
//modified velocity
suxc[tidy][tidx]=sux[tidy][tidx]+0.5*sforcex[tidy][tidx]/srho[tidy][tidx];
suyc[tidy][tidx]=suy[tidy][tidx]+0.5*sforcey[tidy][tidx]/srho[tidy][tidx];
```

APPENDIX B

FROM THE LBE TO THE N-S EQUATIONS

Our starting point is the LBGK model. Performing a Taylor series expansion in the time and space with small parameters $\delta x = \delta t$ to the second order, we obtain:

$$\delta t \frac{\partial f_\alpha}{\partial t} + \delta t e_{\alpha k} \frac{\partial f_\alpha}{\partial x_k} + \frac{(\delta t)^2}{2}\left[\frac{\partial^2 f_\alpha}{\partial t^2} + 2e_{\alpha k}\frac{\partial^2 f_\alpha}{\partial t \partial x_k} + e_{\alpha k}e_{\alpha n}\frac{\partial^2 f_\alpha}{\partial x_k \partial x_n}\right] + \frac{1}{\tau}\left(f_\alpha - f_\alpha^{(eq)}\right) = 0 \qquad \text{B.1}$$

In order to derive the N-S equations from LBE, the Chapman-Enskog expansion, which in essence is a standard multi-scale expansion, is used as follows:

$$\frac{\partial}{\partial t} = \varepsilon \frac{\partial}{\partial t_1} + \varepsilon^2 \frac{\partial}{\partial t_2} \qquad \text{B.2a}$$

$$\frac{\partial}{\partial x} = \varepsilon \frac{\partial}{\partial x_1} \qquad \text{B.2b}$$

where ε is a small parameter, $\varepsilon << 1$. Equation A.2a means that the convection time scale t_1 is much smaller than the diffusion time scale t_2 (low-frequencies assumption), and the PDF f_α can be expanded similarly as:

$$f_\alpha = f_\alpha^{(0)} + \varepsilon f_\alpha^{(1)} + \varepsilon^2 f_\alpha^{(2)} + O\left(\varepsilon^3\right) \qquad \text{B.3}$$

Inserting equation B.2 and B.3 into equation B.1, we obtain:

$$\varepsilon^2\left\{\delta t\left[\frac{\partial f_\alpha^{(1)}}{\partial t_1}+\frac{\partial f_\alpha^{(0)}}{\partial t_2}+e_{\alpha k}\frac{\partial f_\alpha^{(0)}}{\partial x_{1k}}\right]+\frac{(\delta t)^2}{2}\left[\frac{\partial^2 f_\alpha^{(0)}}{\partial t_1^2}+2e_{\alpha k}\frac{\partial^2 f_\alpha^{(0)}}{\partial t_1\partial x_{1k}}+e_{\alpha k}e_{\alpha n}\frac{\partial^2 f_\alpha^{(0)}}{\partial x_{1k}\partial x_{1n}}\right]+\frac{1}{\tau}f_\alpha^{(2)}\right\} \quad \text{B.4}$$
$$+\varepsilon\left[\delta t\frac{\partial f_\alpha^{(0)}}{\partial t_1}+\delta t e_{\alpha k}\frac{\partial f_\alpha^{(0)}}{\partial x_{1k}}+\frac{1}{\tau}f_\alpha^{(1)}\right]=-\frac{1}{\tau}\left[f_\alpha^{(0)}-f_\alpha^{(eq)}\right]$$

Collecting the terms in the ascending order of ε:

$$O(\varepsilon^0): \qquad f_\alpha^{(0)}=f_\alpha^{(eq)} \qquad \text{B.5}$$

$$O(\varepsilon^1): \qquad f_\alpha^{(1)}=-\tau\delta t\left[\frac{\partial f_\alpha^{(0)}}{\partial t_1}+e_{\alpha k}\frac{\partial f_\alpha^{(0)}}{\partial x_{1k}}\right] \qquad \text{B.6}$$

$$O(\varepsilon^2): f_\alpha^{(2)}=-\tau\delta t\left[\frac{\partial f_\alpha^{(1)}}{\partial t_1}+\frac{\partial f_\alpha^{(0)}}{\partial t_2}+e_{\alpha k}\frac{\partial f_\alpha^{(0)}}{\partial x_{1k}}\right]-\tau\frac{(\delta t)^2}{2}\left[\frac{\partial^2 f_\alpha^{(0)}}{\partial t_1^2}+2e_{\alpha k}\frac{\partial^2 f_\alpha^{(0)}}{\partial t_1\partial x_{1k}}+e_{\alpha k}e_{\alpha n}\frac{\partial^2 f_\alpha^{(0)}}{\partial x_{1k}\partial x_{1n}}\right] \qquad \text{B.7}$$

Using equation B.6 and performing some algebra, equation B.7 can be simplified to be:

$$O(\varepsilon^2): \qquad f_\alpha^{(2)}=-\tau\delta t\frac{\partial f_\alpha^{(0)}}{\partial t_2}+\left(\frac{1}{2}-\tau\right)\delta t\left[\frac{\partial f_\alpha^{(1)}}{\partial t_1}+e_{\alpha k}\frac{\partial f_\alpha^{(1)}}{\partial x_{1k}}\right] \qquad \text{B.8}$$

Equation B.5 means that the PDF f_α is not far from the equilibrium state. We can re-write equation B.3 as following:

$$f_\alpha=f_\alpha^{(eq)}+\varepsilon f_\alpha^{(1)}+\varepsilon^2 f_\alpha^{(2)}+O\left(\varepsilon^3\right) \qquad \text{B.9}$$

The equilibrium distribution function satisfies the following constraints:

$$\rho=\sum_\alpha f_\alpha^{(eq)} \qquad \rho\vec{u}=\sum_\alpha \vec{e}_\alpha f_\alpha^{(eq)} \qquad \text{B.10}$$

which lead to the following constrains:

$$\sum_\alpha f_\alpha^{(k)}=0 \qquad \sum_\alpha \vec{e}_\alpha f_\alpha^{(k)}=0 \quad (k=1,2...) \qquad \text{B.11}$$

Summing over all α for both equation B.6 and equation B.8, multiplying by $e_{\alpha n}$ and summing over all α for both equation B.6 and equation B.8, and using the constraints given by equation B.10-11, we can obtain the following macroscopic equations:

$$O(\varepsilon^1): \qquad \frac{\partial \rho}{\partial t_1} + \frac{\partial \rho u_k}{\partial x_{1k}} = 0 \qquad \text{B.12}$$

$$O(\varepsilon^1): \qquad \frac{\partial \rho u_n}{\partial t_1} + \frac{\partial}{\partial x_{1k}} \sum_\alpha e_{\alpha n} e_{\alpha k} \frac{\partial f_\alpha^{(eq)}}{\partial x_{1k}} = 0 \qquad \text{B.13}$$

$$O(\varepsilon^2): \qquad \frac{\partial \rho}{\partial t_2} = 0 \qquad \text{B.14}$$

$$O(\varepsilon^2): \qquad \frac{\partial \rho u_n}{\partial t_2} + \left(1 - \frac{1}{2\tau}\right) \frac{\partial}{\partial x_{1k}} \sum_\alpha e_{\alpha n} e_{\alpha k} \frac{\partial f_\alpha^{(1)}}{\partial x_{1k}} = 0 \qquad \text{B.15}$$

Combining equation B.12 and B.14, the continuity equation is recovered to the second order of ε:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_k}{\partial x_k} = 0 \qquad \text{B.16}$$

Combining equation B.13 and B.15, the momentum equation is recovered to the second order of ε:

$$\frac{\partial \rho u_n}{\partial t} + \frac{\partial}{\partial x_{1k}} \sum_\alpha e_{\alpha n} e_{\alpha k} \frac{\partial f_\alpha^{(eq)}}{\partial x_{1k}} + \left(1 - \frac{1}{2\tau}\right) \frac{\partial}{\partial x_{1k}} \sum_\alpha e_{\alpha n} e_{\alpha k} \frac{\partial f_\alpha^{(1)}}{\partial x_{1k}} = 0 \qquad \text{B.17}$$

Substituting equation B.6 into equation B.17, the momentum equation becomes:

$$\frac{\partial \rho u_n}{\partial t} + \frac{\partial}{\partial x_{1k}} \sum_\alpha e_{\alpha n} e_{\alpha k} \frac{\partial f_\alpha^{(eq)}}{\partial x_{1k}} + \left(1 - \frac{1}{2\tau}\right) \sum_\alpha \left[e_{\alpha n} e_{\alpha k} \frac{\partial^2 f_\alpha^{(eq)}}{\partial t_1 \partial x_{1k}} + e_{\alpha n} e_{\alpha k} \alpha_{\alpha m} \frac{\partial^2 f_\alpha^{(eq)}}{\partial x_{1k} \partial x_{1m}} \right] = 0 \qquad \text{B.18}$$

From equation B.17, we can identify that the momentum flux tensor is:

$$\Pi_{\alpha\beta} = \Pi_{\alpha\beta}^{(0)} + \Pi_{\alpha\beta}^{(1)} = \sum_i e_{i\alpha} e_{i\beta} \left[f_i^{(eq)} + \left(1 - \frac{1}{2\tau}\right) f_i^{(1)} \right] \qquad \text{B.19}$$

with the zero order momentum tensor and the first order momentum tensor given by the following equation:

$$\Pi_{\alpha\beta}^{(0)} = \sum_i e_{i\alpha} e_{i\beta} f_i^{(eq)} \qquad \Pi_{\alpha\beta}^{(1)} = \sum_i e_{i\alpha} e_{i\beta} \left(1 - \frac{1}{2\tau}\right) f_i^{(1)} \qquad \text{B.20}$$

To specify the detailed form of $\Pi_{\alpha\beta}$, the lattice structure and corresponding equilibrium distributions have to be specified. The D2Q9 lattice is considered here. Derivation for other lattice structures can be obtained using similar fashion. For the D2Q9 model, the following identities are observed:

$$\sum_\alpha w_\alpha e_{\alpha m} e_{\alpha n} e_{\alpha k} e_{\alpha j} = \frac{c^4}{9}\left(\delta_{mn}\delta_{kj} + \delta_{mk}\delta_{nj} + \delta_{mj}\delta_{nk}\right) \qquad \text{B.21}$$

$$\frac{\partial}{\partial x_{1m}} \frac{\partial}{\partial x_{1k}} \sum_\alpha w_\alpha e_{\alpha m} e_{\alpha n} e_{\alpha k} e_{\alpha j} \rho u_j = \frac{c^4}{9}\left(\frac{\partial^2}{\partial x_{1m}\partial x_{1m}}(\rho u_n) + 2\frac{\partial^2}{\partial x_{1m}\partial x_{1n}}\rho u_n\right) \qquad \text{B.22}$$

Substituting equation B.6 into equation B.18 and using identity B.22, we find:

$$\sum_\alpha \left[e_{\alpha n} e_{\alpha k} \frac{\partial^2 f_\alpha^{(eq)}}{\partial t_1 \partial x_{1k}} + e_{\alpha n} e_{\alpha k} e_{\varepsilon m} \frac{\partial^2 f_\alpha^{(eq)}}{\partial x_{1k}\partial x_{1m}} \right] = \frac{c^2}{3}\left[\frac{\partial^2}{\partial x_{1m}\partial x_{1m}}(\rho u_n) + 2\frac{\partial^2}{\partial x_{1m}\partial x_{1n}}\rho u_n\right] \qquad \text{B.23}$$

Substituting equation B.23 back into equation B.18, the momentum equation becomes:

$$\frac{\partial \rho u_n}{\partial t} + \frac{\partial \rho u_k u_n}{\partial x_{1k}} = -\frac{\partial p}{\partial x_{1n}} + \left(\tau - \frac{1}{2}\right)\frac{c^2}{3}\left[\frac{\partial^2}{\partial x_{1m}\partial x_{1m}}(\rho u_n) + 2\frac{\partial^2}{\partial x_{1m}\partial x_{1n}}\rho u_n\right] \qquad \text{B.24}$$

where the pressure p and kinematic viscosity v are given by:

$$p \equiv c_s^2 \rho = \frac{\rho}{3} \qquad \text{B.25}$$

$$v \equiv \frac{c^2}{3}\left(\tau - \frac{1}{2}\right) = \frac{2\tau - 1}{6} \qquad \text{B.26}$$

Therefore, the LBE recovers the N-S equation (equation B.27-28) in the incompressible flow limit:

$$\frac{\partial u_\alpha}{\partial x_\alpha} = 0 \qquad \text{B.27}$$

$$\frac{\partial u_\beta}{\partial t} + u_\alpha \frac{\partial u_\beta}{\partial x_\alpha} = -\frac{1}{\rho}\frac{\partial p}{\partial x_\beta} + \nu \nabla^2 u_\beta \qquad \text{B.28}$$

APPENDIX C

FROM THE CONTINUUM BOLTZMANN EQUATION TO THE LBE MODEL

We start from the Boltzmann equation with the BGK approximation:

$$\frac{\partial f}{\partial t} + \vec{\xi} \cdot \nabla f = -\frac{1}{\lambda}\left[f - f^{(0)}\right] \qquad \text{C.1}$$

where $f^{(0)}$ satisfies the Maxwell-Boltzmann distribution function:

$$f^{(0)} \equiv \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\frac{\left(\vec{\xi} - \vec{u}\right)^2}{2RT}\right) \qquad \text{C.2}$$

where R is the gas constant, D is the dimension of the space, and T is the macroscopic temperature. The hydrodynamic variables are the moments of distribution function f:

$$\rho = \int f d\vec{\xi} = \int f^{(0)} d\vec{\xi}\ ,\ \rho\vec{u} = \int \vec{\xi} f d\vec{\xi} = \int \vec{\xi} f^{(0)} d\vec{\xi}\ ,\ \rho\varepsilon = \frac{1}{2}\int \left(\vec{\xi} - \vec{u}\right)^2 f d\vec{\xi} = \int \left(\vec{\xi} - \vec{u}\right)^2 f^{(0)} d\vec{\xi} \qquad \text{C.3}$$

where $\varepsilon = \frac{1}{2} D_0 RT$ and D_0 is the number of the degrees of freedom of a particle.

Integrating equation C.1 over a time interval δt, we find:

$$f\left(\vec{x} + \vec{\xi}\delta t, \vec{\xi}, t + \delta t\right) = e^{-\delta t/\lambda} f\left(\vec{x}, \vec{\xi}, t\right) + \frac{1}{\lambda} e^{-\delta t/\lambda} \int_0^{\delta t} e^{t'/\lambda} f^{(0)}\left(\vec{x} + \vec{\xi}t', \vec{\xi}, t + t'\right) dt' \qquad \text{C.4}$$

Assuming that δt is relatively small and g is locally smooth, and further neglecting the terms of $O(\delta t^2)$ or higher on the right hand side (RHS) of equation C.4, we obtain:

$$f(\vec{x}+\vec{\xi}\delta t,\vec{\xi},t+\delta t)-f(\vec{x},\vec{\xi},t)=-\frac{1}{\tau}\left[f(\vec{x},\vec{\xi},t)-f^{0}(\vec{x},\vec{\xi},t)\right] \quad \text{C.5}$$

Expanding the equilibrium distribution $f^{(0)}$ as a Taylor series in $\vec{u}$:

$$f^{(0)}=\frac{\rho}{(2\pi RT)^{D/2}}\exp\left(-\frac{\vec{\xi}^{2}}{2RT}\right)\left[1+\frac{(\vec{\xi}\cdot\vec{u})^{2}}{RT}+\frac{(\vec{\xi}\cdot\vec{u})^{2}}{2(RT)^{2}}-\frac{\vec{u}^{2}}{2RT}\right]+O(\vec{u}^{3}) \quad \text{C.6}$$

In order to derive the N-S equations, the second order expansion is enough, so we denote:

$$f^{(eq)}=\frac{\rho}{(2\pi RT)^{D/2}}\exp\left(-\frac{\vec{\xi}^{2}}{2RT}\right)\left[1+\frac{(\vec{\xi}\cdot\vec{u})^{2}}{RT}+\frac{(\vec{\xi}\cdot\vec{u})^{2}}{2(RT)^{2}}-\frac{\vec{u}^{2}}{2RT}\right] \quad \text{C.7}$$

and we will use $f^{(eq)}$ instead of $f^{(0)}$ in the following calculations.

To numerically evaluate the hydrodynamic variables in equation C.3, we need to discretized the momentum space $\vec{\xi}$ properly. With an appropriately discretized momentum space, integration in equation C.3 can be approximated by quadrature up to a certain degree of accuracy, i.e.:

$$\rho=\sum_{\alpha}f_{\alpha}=\sum_{\alpha}f_{\alpha}^{(eq)},\ \rho\vec{u}=\sum_{\alpha}\xi_{\alpha}f_{\alpha}=\sum_{\alpha}\xi_{\alpha}f_{\alpha}^{(eq)},\ \rho\varepsilon=\frac{1}{2}\sum_{\alpha}(\vec{\xi}-\vec{u})^{2}f_{\alpha}=\frac{1}{2}\sum_{\alpha}(\vec{\xi}-\vec{u})^{2}f_{\alpha}^{(eq)} \quad \text{C.8}$$

$$\text{where } f_{\alpha}\equiv f_{\alpha}(\vec{x},t)\equiv W_{\alpha}f(\vec{x},\xi_{\alpha},t),\ f_{\alpha}^{(eq)}\equiv f_{\alpha}^{(eq)}(\vec{x},t)\equiv W_{\alpha}f^{(eq)}(\vec{x},\xi_{\alpha},t) \quad \text{C.8}$$

W_{α} is the weight coefficient of the quadrature and $\vec{\xi}_{\alpha}$ is the discrete velocity set.

The above approximation process can be written in the following general form:

$$\int\psi(\vec{\xi})f_{\alpha}^{(eq)}(\vec{x},\xi_{\alpha},t)=\sum_{\alpha}W_{\alpha}\psi(\vec{\xi})f^{(eq)}(\vec{x},\xi_{\alpha},t) \quad \text{C.9}$$

where $\psi(\vec{\xi})$ is the polynomial of $\vec{\xi}$. The above integral has the following common part, which can be evaluated by a Gaussian-type quadrature [76]:

$$I = \int \exp\left(-\frac{\vec{\xi}^2}{2RT}\right)\psi(\vec{\xi})d\vec{\xi} = \sum_{\alpha} W_\alpha \exp\left(-\frac{\xi_\alpha^2}{2RT}\right)\psi(\xi_\alpha) \qquad \text{C.10}$$

We next need to discretized the phase space so that the evolution equation. i.e. C.5, can be numerically evaluated over a discretized physical phase space and time. In doing this, we need to consider two factors. First, the discretized momentum space is coupled to that of physical space such that a lattice structure is obtained. Second, a lattice with necessary symmetries is required to satisfy the conservation constraints in equation C.8 and also to retain the N-S equations. We use the D2Q9 model to illustrate the derivation of the LBE models.

A Cartesian coordinate system is used to recover the D2Q9 model. There fore, we can set: $\psi(\vec{\xi}) = \xi_x^m \xi_y^n$, where ξ_x^m and ξ_y^m are the x and y components of $\vec{\xi}$. The integral in equation C.10 can then be written as:

$$I = \left(\sqrt{2RT}\right)^{(m+n+2)} I_m I_n \qquad \text{C.11}$$

where

$$I_m = \int_{-\infty}^{+\infty} e^{-\xi^2} \xi^m d\xi\,,\ \xi = \xi_x / \sqrt{2RT} \ \text{ or } \ \xi = \xi_y / \sqrt{2RT} \qquad \text{C.12}$$

For the purpose of deriving the D2Q9 model, a third-order Hermite formula [77] is used to evaluate I_m, i.e. $I_m = \sum_{j=1}^{3} \omega_j \xi_j^m$. The three abscissas of the quadrature are:

$$\xi_1 = -\sqrt{3/2}\,,\ \xi_2 = 0\,,\ \xi_3 = \sqrt{3/2} \qquad \text{C.13}$$

and the corresponding weight coefficients are:

$$\omega_1 = \sqrt{\pi}/6\,,\ \omega_2 = 2\sqrt{\pi}/3\,,\ \omega_3 = \sqrt{\pi}/6$$

Then the integral in equation C.11 becomes:

$$I = 2RT\omega_2^2\psi(0) + \sum_{\alpha=1}^{4}\omega_1\omega_2\psi(\xi_\alpha) + \sum_{\alpha=5}^{8}\omega_1^2\psi(\xi_\alpha) \qquad \text{C.14}$$

where ξ_α is the discrete velocity set with a zero-velocity vector for $\alpha = 0$, vectors of $\sqrt{3RT}(\pm 1,0)$ and $\sqrt{3RT}(0,\pm 1)$ for α from 1 to 4, and $\sqrt{3RT}(\pm 1,\pm 1)$ for α from 5 to 8. After the momentum space is discretized with these nine discrete velocities, the physical space needs to be discretized accordingly. This means that the physical space is discretized into a square lattice space with a lattice constant $\delta x = \sqrt{3RT}\delta t$. If we only deal with the isothermal model, then the temperature T has no physical significance here. Thus we can pick δx to be a fundamental quantity, i.e. $\sqrt{3RT} = c \equiv \frac{\delta x}{\delta t}$, or $RT = c_s^2 = \frac{1}{3}c^2$, where c_s is called the sound speed of the model, and c is the lattice speed. Usually the lattice speed is fixed as 1, so for the D2Q9 model, $c_s^2 = 1/3$.

Comparing equation C.10 and C.14, we can identify the weights in equation C.10 as:

$$W_\alpha = 2\pi RT \exp\left(\frac{\xi_\alpha^2}{2RT}\right) w_\alpha \qquad \text{C.15}$$

where w_α is the same as given in equation 1.5 in Section 1.1.

Then the equilibrium distribution function of the D2Q9 model is:

$$f_\alpha^{eq} = W_\alpha f^{(eq)}(\vec{x}, \xi_\alpha, t) = \rho w_\alpha \left[1 + \frac{3}{c^2}\vec{e}_\alpha \cdot \vec{u} + \frac{9}{2c^4}(\vec{e}_\alpha \cdot \vec{u})^2 - \frac{3}{2c^2}\vec{u} \cdot \vec{u}\right] \qquad \text{C.16}$$

where $\vec{e}_\alpha$ is the same as given in equation 4. Model for the other lattice configurations (D2Q7, D3Q27) can be derived in a similar manner [78].

BIBLIOGRAPHY

1. L.-S. Luo, The lattice-gas and lattice Boltzmann methods: Past, Present, and Future, Proceedings of the International Conference on Applied Computational Fluid Dynamics, Beijing, China, pp. 52-83, 2000.

2. R. Benzi, S. Succi, and M. Vergassola, The lattice Boltzmann equation: theory and applications, Phys. Rep. 222, 145-197, 1992.

3. Y. H. Qian, S. Succi, and S. A. Orszag, Recent advances in lattice Boltzmann computing, Annu. Rev. Comput. Phys. 3, 195-242, 1995.

4. S. Chen and G. D. Doolen, Lattice Boltzmann method for fluid flows, Annu. Rev. Fluid Mech. 30, 329-364, 1998.

5. Deng B, Shi BC, Wang GC, A new lattice Bhatnagar–Gross–Krook model for convection-diffusion equation with a source term, Chin Phys Lett 22, 267–70, 2005.

6. Chen S, Dawson SP, Doolen GD, Janecky DR, Lawniczak A. Lattice methods and their applications to reacting systems, Comput Chem Eng, 19, 617–46, 1995.

7. Shen Z, Yuan G, Shen L. Lattice Boltzmann method for Burgers equation, Chin J Comput Phys, 17, 166–72, 2000.

8. Li HB, Huang PH, Liu MR, Kong LJ. Simulation of the MKdV equation with lattice Boltzmann method. Acta Phys Sini, 50, 837–40, 2001.

9. Zhang CY, Tan HL, Liu MR, Kong LJ. A lattice Boltzmann model and simulation of KdV–Burgers equation, Commun Theor Phys, 42, 281–4, 2004.

10. Ma CF. A new lattice Boltzmann model for KdV–Burgers equation. Chin Phys Lett, 22, 2313–5, 2005.

11. Zhenhua Chai, Baochang Shi and Lin zheng, Aunified lattice Boltzmann model for some nonlinear partial differential equations, Chaos, Solitons and Fractals 36, 874-882, 2008.

12. Dodd RK, Eilbeck JC, Gibbon JD, Morris HC. Solitons and nonlinear wave equations. London, Academic Press, 1982.

13. Baochang Shi, Nanzhong He, and Zhaoli Guo, Lattice Boltzmann Model for High-Order Nonlinear Partial Differential Equations, Physics comp, 10, 2009.

14. R. K. Pathria, statistical Mechanics, 2nd edition, Butterworth-Heinemann, 1996.

15. P. L. Bhatnagar, E. P. Gross and M. Krook, Amodel for collision processes in gases. I. Small amplitude processes in charged and neutral one-component system, Phys. Rev., 94, pp. 511-525, 1954.

16. Xiaoyi He and Li-Shi Luo, theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltmann equation, Physical review E, volume 56, Number 6, December 1997.

17. D. d'Humières, I. Ginzburg, M. Krafczyk, P. Lallemand, and L.-S. Luo, Multi-relaxation-time lattice Boltzmann models in three dimensions, Phil. Trans. R. Soc. Lond. A 360, pp. 437-451, 2002.

18. X.-W. Shan and G. D. Doolen, Multicomponent lattice Boltzmann mdoel with interparticle interaction, Journal of Statistical Physics, 81, Nos. ½, 1995.

19. P. Yuan and L. Schaefer, Equations of state in a lattice Boltzmann model, Physics of Fluids, 18, 2006.

20. D. R. Noble, S. Chen, J. G. Georgiadis, and R. O. Buckius, A consistent hydrodynamic boundary condition for the lattice Boltzmann method, Phys. Fluids, 7, pp. 203-209, 1995.

21. P. Ziegler, Boundary conditions for lattice Boltzmann simulations, J. Stat. Phys., 71, pp. 1171-1177, 1993.

22. R. Mei, and W. Shyy, On the finite difference-based lattice Boltzmann method in curvilinear coordinates, J. Compu. Phys. 143, pp. 426-448, 1998.

23. T. Inamuro, M. Yoshino, and F. Ogino, A non-slip boundary condition for lattice Boltzmann simulations, Phys. Fluids, 7, pp. 2928–2930, 1995.

24. A.J.C. Ladd, Numerical simulations of particulate suspensions via a discretized Boltzmann equation, J. Fluid Mech. 271 (1994) 285–339.

25. B.K. Aidun, Y. Lu, Lattice Boltzmann simulations of solid particles suspended in fluid, J. Stat. Phys. 81, 49–61, 1995.

26. Filippova, D. Hel, Grid refinement for lattice-BGK models, J. Comput. Phys. 147, 219–228, 1998.

27. R. Mei, L.-S. Luo, W. Shyy, An accurate curved boundary treatment in the lattice Boltzmann method, J. Comput. Phys. 155, 307–329, 1999.

28. R. Mei, W. Shyy, D. Yu, L.-S. Luo, Lattice Boltzmann method for 3-D flows with curved boundary, J. Comput. Phys. 161, 680–699, 2000.

29. Jie Bao, Peng Yuan, and Laura Schaefer, A mass conserving boundary condition for the lattice Boltzmann equation method, Journal of Computational Physics 227, 8472-8487, 2008.

30. S. Chen, D. Martínez, and R. Mei, On boundary conditions in lattice Boltzmann method, Phys. Fluids 8, pp. 2527-2536, 1996.

31. Q. Zou and X. He, On pressure and velocity boundary conditions for the lattice Boltzmann BGK model, Phys. Fluids 9, pp. 1591-1598, 1997.

32. D. Yu, Viscous Flow Computations with the Lattice Boltzmann Equation Method, Ph.D. thesis, Department of Aerospace Engineering, mechanics and Engineering Science, Univ. of Florida, 2002.

33. J. L. Hennessy and D. A. Patterson, Computer architecture, 4th edition, Morgan Kaufmann, 2006.

34. M. Swift, S. Orlandini, W. Osborn and J. Yeomans, Lattice Boltzmann simulations of liquid-gas and binary-fluid systems, Phys. Rev. E 54, pp. 5041-5052, 1996.

35. R. Zhang, Lattice Boltzmann Approach for Immiscible Multiphase Flow, Ph.D. thesis, Department of Mechanical Engineering, University of Delaware, 2000.

36. Ya Song Wei and Richard J. Sadus, Equation of state of the calculation of fluid-phase equilibria, AIChE Journal, Vol. 46, No.1 2000.

37. M. Sbragaglia, R. Benzi, L. Biferale, S. Succi, K. Sugiyama, and F. Toschi, Generalized lattice Boltzmann method with multirange pseudopotential, Phys. Rev. E 75, 026702, 2007.

38. S. Williams, J. carter, L. Oliker, J. Shalf and K. Yelick, Lattice Boltzmann Simulation Optimization on Leading Multicore Platforms, IPDPS 2008 April 14-28, Miami, Florida, USA.

39. Intel® Core™2 Duo Processors, http://www.intel.com/products/processor/core2duo/index.htm?iid=prod+prod_core2duo

40. AMD Opteron™ Processors for Workstations, http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_8826,00.html

41. The Cell project at IBM Research, http://www.research.ibm.com/cell/.

42. Modern supercomputer architecture http://en.wikipedia.org/wiki/supercomputer

43. NVIDIA CUDA Computer Unified Device Architecture Programming guide, 2007.

44. Appel A. Some techniques for shading machine renderings of solids. Proceedings of the Spring Joint Computer Conference 32, 37–49, 1968.

45. Rendering (computer graphics), http://en.wikipedia.org/wiki/Rendering_(computer_graphics)

46. NVIDIA SLI Technology, http://www.slizone.com/page/home.html

47. NVIDIA Tesla Computing Solutions, http://www.nvidia.com/object/tesla_computing_solutions.html

48. Xomg He, Gray Duckwiler and Daniel J. Valentino, Lattice Boltzmann simulation of cerebral artery hemodynamics, Computers & Fluids 38, 789-796, 2009.

49. Gibbons GH, Dzau VJ. The emerging concept of vascular remodeling. N Engl J Med , 330(20):1431–8, 1994.

50. Kondo S, Hashimoto N, Kikuchi H, Hazama F, Nagata I, Kataoka H. Cerebral aneurysm arising at nonbranching sites. Stroke, 28:398–404, 1997.

51. Malek AM, Alper SL, Izumo S. Hemodynamic shear stress and its role in atherosclerosis. JAMA, 282(21):2035–42, 1999.

52. Ferguson GG. Physical factors in the initiation, growth, and rupture of human intracranial saccular aneurysms. J Neurosurg, 37:666–77, 1972.

53. Parmentier EM, Morton WA, Petschek HE. Platelet aggregate formation in a region of separated blood flow. Phys Fluids, 20:2012–21, 1977.

54. Ku DN, Giddens DP, Zarins CK, et al. Pulsatile flow and atherosclerosis in the human carotid bifurcation – positive correlation between plaque location and low and oscillating shear–stress. Arteriosclerosis, 5(3):293–302, 1985.

55. Ku DN. Blood flow in arteries. Annu Rev Fluid Mech, 29:399–434, 1997.

56. Taylor CA, Hughes TJR, Zarins CK. Finite element modeling of blood flow in arteries. Comput Methods Appl Mech Eng, 158(1-2):155–96, 1998.

57. Berger SA, Jou LD. Flows in stenotic vessels. Annu Rev Fluid Mech, 32:347–82, 2000.

58. Taylor CA. Experimental and computational methods in cardiovascular fluid mechanics. Annu Rev Fluid Mech, 36:197–231, 2004.

59. Cebral JR, Castro MA, Appanaboyina A, Putman CM, Millan D, Frangi AF. Efficient pipeline for image-based patient-specific analysis of cerebral aneurysm hemodynamics: technique and sensitivity. IEEE Trans Med Imaging, 24(4):457–67, 2005.

60. Cavedon,C., Bova, F., and William R. Hendee, Moderator, Three-dimensional rotational angiography (3DRA) adds substantial information to radiosurgery treatment planning of AVM's compared to angio-CT and angio-MR, Med. Phys. Volume 31, Issue 8, pp. 2181-2183, 2004.

61. Z Zeng, H. Zakaria, R Kadirvel, Y Ding, D Lewis, D Kallmes, AM Robertson, CFD Study of the Relationship between wall shear stress and aspect ratio in lactase induced rabbit aneurysm models , Proceedings of the ASME 2008 Summer Bioengineering Conference, SBC-2008, June 25-29, Marco Island, Florida, USA, 2008.

62. Zijing Zeng, Mike Durka, David F. Kallmes, Ramanathan Kadirvel, Yonghong Ding, Debra A. Lewis, Anne M. Robertson, Hemodynamic features ofelastase induced rabbit saccular aneurysms—coupling of biology and hemodynamics, Proceedings of Frontiers of Biomedical Imaging Science conference 2009, Nashville, Tennessee, 2009.

63. Cassie, ABD, Wettability of Porous Surfaces, Trans. Faraday Soc. 40: 546–551. 1944.

64. Wenzel, RN Resistance of Solid Surfaces to Wetting by Water. Ind. Eng. Chem. 28: 988–994. 1936.

65. Lafuma, A.; Quere, D. Superhydrophobic states. Nature Materials 2: 457–460. 2003.

66. Barthlott, Wilhelm; Ehler, N. Raster-Elektronenmikroskopie der Epidermis-Oberflächen von Spermatophyten. Tropische und subtropische Pflanzenwelt (Akad. Wiss. Lit. Mainz) 19: 110. 1977.

67. Neinhuis, C.; Barthlott, W. Characterization and distribution of water-repellent, self-cleaning plant surfaces. Annals of Botany 79: 667–677. 1997.

68. Neinhuis, C.; Barthlott, W. Characterization and distribution of water-repellent, self-cleaning plant surfaces. Annals of Botany 79: 667–677. 1997.

69. Forbes, P. The Gecko's Foot, Bio-inspiration – Engineering New Materials and devices from Nature. London: Fourth Estate. p. 272. 2005.

70. Forbes,P. Self-Cleaning Materials. Scientific American 299 (2): 67–75 2008.

71. C. Sanchez, B. Julian, P. Belleville, and M. Popall, Applications of hybrid organic-inorganic nanocomposites, J. Mater. Chem., Vol. 15, pp. 3559, 2005.

72. Hongbin Cheng, Fabricaion of zinc oxide micro-nanostructure and their applications in gas sensing and nanocomposites, Ph. D. dissertation, University of Pittsburgh, 2009.

73. Michael Wachhold, K. Kasthuri Rangan, Simon J. L. Billinge, Valeri Petkov, Joy Heising and Mercouri G. Kanatzidis, Mesostructured Non-Oxidic Solids with Adjustable Worm-hole Shaped Pores: Metal Germanium Sulfide and Selenide Frameworks Based on the Tetrahedral [Ge4Q10]4- Clusters. Advanced Mater, 12, 85-91, 2000.

74. Prahalad M Parthangal, Richard E Cavicchi and Michael R Zachariah, A universal approach to electrically connecting nanowire arrays using nanoparticles—application to a novel gas sensor architecture, Nanotechnology, 2006.

75. Youngjo Tak and Kijung Yong, A Novel Heterostructure of Co3O4/ZnO Nanowire Array Fabricated by Photochemical Coating Method, J. Phys. Chem. C, 112, 74-79, 2008.

76. P. J. Davis and P. Rabinowitz, Methods of Numerical Integration, 2nd ed., Academic, New York, 1984.

77. N.S. Martys, and H. Chen, Simulation of multicomponent fluids in complex threedimensional geometries by the lattice Boltzmann method, Phys. Rev. E 53, pp. 743-750, 1996.

78. X. He and L.-S. Luo, Theory of the lattice Boltzmann equation: from the Boltzmann equation to the lattice Boltzmann equation, Phys. Rev. E, 56, pp. 6811-6817, 1997.