LATTICE BOLTZMANN METHOD FOR MICRO CHANNEL AND MICRO ORIFICE FLOWS

By

TAIHO YEOM

Bachelor of Science in Mechanical Engineering

Ajou University

Suwon, South Korea

2005

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2007

LATTICE BOLTZMANN METHOD FOR MICRO

CHANNEL AND MICRO ORIFICE FLOWS

Thesis Approved:

F. W. Chambers

K. A. Sallam

J. D. Jacob

A. Gordon Emslie

Dean of the Graduate College

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

NOMENCLATURE

a	accommodation coefficient
c	particle streaming velocity
c_s	speed of sound
e_α	α th discrete particle velocity
f	single particle distribution function
f_α	α th discrete particle distribution function
f^{eq}	equilibrium distribution function
F	external force
h	height of the open orifice area
H	channel height
k	Boltzmann constant
Kn	Knudsen number
l	orifice length
L	channel length
m	molecular mass
P	pressure
r_f	reflection factor
Re	Reynolds number

t	time
T	temperature
u	mean thermal velocity of the molecule
U	fluid velocity
w_α	weighting factor in the LBM
x	position of the particle
λ	mean free path
μ	dynamic viscosity
v	velocity of the particle
ρ	density
ς	single relaxation time
τ	dimensionless relaxation time
υ	kinematic viscosity

ABBREVIATIONS

BE	Boltzmann equation
BGK	Bhatnagar, Gross, and Krook
DSMC	Direct Simulation Monte Carlo
DVM	Discrete Velocity Model
D2Q9	two dimensional nine velocities
LBE	Lattice Boltzmann Equation
LBGK	Lattice Bhatnagar, Gross, and Krook

LBM	Lattice Boltzmann Method
LGCA	Lattice Gas Cellular Automata
MEMS	Micro-Electro-Mechanical Systems
NS	Navier-Stokes

CHAPTER I

INTRODUCTION

1.1 Background

Recently, there has been rapid development of Micro-Electro-Mechanical Systems (MEMS), which are applicable to a variety of industrial fields and many other scientific applications. Many different types of micron-size systems are under development ranging from sensors, actuators, and electronic circuits to the integrated systems combining these applications. These MEMS also contain many micro-fluidic devices such as micro-channels, micro-pumps, micro-nozzles, and micro-filters. Due to the advent of MEMS and the micro-fluidic devices inside the systems, the study of micro flows has become more important than the past. As a result, many researchers have been investigating micro flows experimentally and numerically. Unlike macro flows, fabricating instruments for micro-scale experiments is obviously difficult work (Arkilic, et al. [1]). Therefore as a more effective and alternative approach, numerical methods have been considered by many researchers.

In micro gas flow, there are some important effects which cannot be predicted in the normal viscous flow based on the continuum hypothesis (Karniadakis, et al. [2]). The first one is a rarefaction effect which is attributed to the original characteristic of the dilute gas itself. Since micro flows have dimensions of the order of 0.1 to 10 μm, at this very small scale a fluid particle can travel a relatively long distance and collide with a boundary before it collides with another particle (Zea and Chambers [3]). Knudsen number (*Kn*), which is the ratio of the mean free path (λ) to the characteristic length (H) of the system, describes this effect. As *Kn* increases this rarefaction effect becomes truly

significant. Another main effect is compressibility, which represents that the pressure distribution along the flow direction is not linear. Basically gas flows are a compressible flow. In addition, there is a need of the large pressure difference to drive a fluid through a relatively long channel compared with macro-scale channels, so this compressibility become important under the large pressure variation between the inlet and outlet (Agrawal and Agrawal [4]). Viscous heating and thermal creep (Karniadakis, et al. [2]) are other important effects but the current work does not cover these effects. Ahmed and Beskok [5] have investigated viscous heating as well as compressibility and rarefaction effects in micro gas flow through micro-filters. The final important effect in micro gas flows is the slip velocity at the boundary wall. The present work is mainly focused on obtaining this slip velocity at the wall by applying proper boundary conditions to gain appropriate results for micro channels and micro filters. Lee and Lin [6] regarded the slip velocity at the boundary wall from the results of LBM simulation as a numerical error produced by the lack of stability of applied boundary conditions. However this slip velocity is obviously an existing phenomenon in the real physics of micro flows. Slip flow already was verified analytically by Karniadakis, et al. [2], who presented a unified flow model for micro flows which is applicable under the wide Knudsen number range including the transitional flow regime. Lee and Lin [6] applied the boundary conditions to generate slip velocity in a physically proper way. Besides, other researchers (Zhang, et al. [7], Tang, et al. [8], Lim, et al. [9]) conducted numerical simulation using various types of boundary conditions to analyze the slip velocity at the wall under various conditions. These papers will be discussed in detail later.

By the Knudsen number (Kn), micro flows can be divided into different flow regimes (Karniadakis, et al. [2]) as can be seen in Table I. For $Kn < 0.01$ the flow can be regarded as continuum flow regime that is governed by continuum hypothesis. But as Kn increases the domination of continuum hypothesis for the flow starts to vanish. For $0.01 < Kn < 0.1$ the flow is called slip flow regime and at this flow regime, there exists a slip velocity at the wall boundary. For $0.1 < Kn < 10$ the flow becomes transitional regime. These two flow regimes of slip and transitional are typical for gas flows in micro flows. Beyond $Kn = 10$, the flow is considered as free molecular flow regime that should be considered by a particle based analysis that solves the equations for motions of molecules.

TABLE I.
Classified flow regimes by various ranges of Knudsen number

Range of Knudsen number	**Flow regime**
$Kn < 0.01$	Continuum flow
$0.01 < Kn < 0.1$	Slip flow
$0.1 < Kn < 10$	Transitional flow
$10 < Kn$	Free molecular flow

A number of numerical analyses have been performed to investigate micro flows through micro-channels and micro-filters. Arkilic, et al. [1] fabricated a micro-channel for micro gas flows and demonstrated that a numerical method, which is based on the compressible Navier-Stokes equations (NS) with the slip boundary condition, matched well with the results from the experiments of fabricated micro-channel. Chen, et al. [10] also used compressible NS equations to conduct numerical analysis applying the slip boundary condition. Another research group, (Ahmed and Beskok [5], Ahmed and Beskok [11]), simulated gas flows in micro-filters numerically with spectral element method NS equations, which utilize high-order velocity slip and temperature jump boundary

conditions. The numerical method that solves the NS equations shows good agreement with the literature in their results. However, they should be incorporated with first-order or higher-order boundary conditions and they are applicable for only limited flow regimes of continuum and slip flow (Tang, et al. [12]). Mott, et al. [13] conducted the simulation of micro flows through micro-filters to verify the scaling laws, which represent the pressure drop along the filter, in the continuum regime with a Direct Simulation Monte Carlo (DSMC) method. The DSMC is a particle-based numerical simulation and it is basically an efficient solution for the high Knudsen number gas flows of complex geometries. However, DSMC calculates properties of numerous cells, which are much smaller than those of the other numerical methods and contains at least 20 molecules that should be simulated in the previous step (Karniadakis, et al. [2]). Therefore, DSMC simulations require much more computational time for the same size of domain than other numerical methods. The Lattice Boltzmann Method (LBM) is another particle-based method that has been spotlighted recently as an effective approach for micro flow simulations. In contrast, the LBM does not calculate properties of each molecule contained in the discrete cell. Thus it is a more efficient method than DSMC. In addition, this newly emerged method can be applied to all flow regimes, from the macro-scale flow to the free molecular flow regime and can be performed with very complex geometries like porous media. Zea and Chambers [3] considered micro gas flows through the micro channel and micro orifice using LBM. Lee and Lin [6], Zhang, et al. [7], Tang, et al. [8], Lim, et al. [9], Tang, et al. [12] simulated numerical experiments based on LBM for 2D micro channel gas flows. Jeong, et al. [14] and Agrawal and Agrawal [4] also conducted LBM simulation for 3D micro-channel gas flows. In addition, Chen and

Doolen [15] have reviewed a wide range of applications of the LBM. Important effects of micro flows mentioned previously can be observed well from results of the many research groups that performed LBM simulations. However, there are some inconsistencies among the results. On that account, more efforts should be devoted to develop advanced LBM approaches.

1.2 Problem statement and presentation

Zea and Chambers [3] performed 2D LBM simulation for micro channel and micro orifice flows. The micro orifice was the basic configuration of the micro filter. Their simulations yielded compressibility and density distributions along the channel and filter that were matched well with the literature, but did not simulate the slip velocity at the wall of the micro channel accurately. For the current LBM simulation of micro channels, 21×2100 lattice nodes are used to make the ratio of length to height 100. Implementations are primarily performed in the slip flow regime to examine the slip velocity at the wall. To provide the slip velocity at the wall of the micro channel and micro filter, the no-slip bounce back, reflection factor (Tang, et al. [8]) and accommodation coefficients (Zhang, et al. [7]) are applied to the boundary conditions. For the micro orifice simulations, the ratio which is the ratio of the open orifice area to the total area was selected as 0.6 to compare the results with the literature. The shape of the orifice was selected as a square. Micro orifice simulations are also conducted in the slip flow regime with the proper boundary conditions obtained for the micro channel simulations. The air was assumed to flow through the micro orifice under isothermal conditions throughout the computations.

1.3 Objectives of research.

The objective of this project is to investigate important micro gas flow features including slip velocity, compressibility, and rarefaction effects for micro channel and micro orifice flows using the LBM simulation.

CHAPTER II

LATTICE BOLTZMANN METHOD

2.1 Fundamentals of the method.

The Lattice Boltzmann Method (LBM) is a newly emerging method for the numerical solution of viscous flow problems, which is different from the traditional solution with Navier-Stokes equations. The LBM solves the simplified Boltzmann Equation (BE). The original BE can be expressed as:

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} + F\frac{\partial f}{\partial v} = Q(f_1, f_2) \tag{2.1.1}$$

The term on the left-hand side represents the kinetic behavior of a particle following a trajectory influenced by the external force F. The state of this particle can be expressed by the particle distribution function $f(x,v,t)$, the most important variable in the LBM, where x denotes the position of a particle, and v denotes the velocity of the particle. The first term on the left-hand side is the rate of change of $f(x,v,t)$ and the second term is the convection of particles with the velocity v caused by the external force F (Karniadakis, et al. [2]). The term on the right-hand side is called the collision operator that represents the collisions of particles. This collision operator is given by:

$$Q(f_1, f_2) = \int_{R^3}\int_{S^+} |V \cdot n|(f_{1'}f_{2'} - f_1 f_2)dndv_1 \tag{2.1.2}$$

In the above equation, 1 and 2 represent two particles before the collision and $1'$ and $2'$ represent two particles after collision. Also, V indicates the relative velocity of v_2 to the

v_1. The integration is calculated in the three-dimensional velocity space R^3 and the hemisphere S^+, which implies the volume where the particles are moving after collisions (Karniadakis, et al. [2]). The collision term truly makes the BE difficult to solve analytically and numerically because of its complex nonlinear integral term. Thus many approaches to simplify this part have been studied. One of the most widely used methods is the Bhatnagar, Gross, and Krook (BGK) model (Bhatnagar, et al. [16]). The BGK model simplifies the collision term as:

$$Q(f_1, f_2) = -\frac{f - f^{eq}}{\varsigma} \tag{2.1.3}$$

where $f \equiv f_1$ and f^{eq} is the equilibrium distribution function which is called the Maxwell-Boltzmann distribution function. ς denotes the single relaxation time which is associated with the distribution function f and adjusts the approach rate to the equilibrium state in the local relaxation process (Tang, et al. [8]). The equilibrium distribution function is expressed by:

$$f^{eq} = \frac{m}{(2\pi kT)^{3/2}} \exp\left[-\frac{mu^2}{2kT}\right] \tag{2.1.4}$$

where k is the Boltzmann constant, and m, u and T are the molecular mass, mean thermal velocity of molecules and temperature, respectively. The BE with the BGK model can be represented as:

$$\frac{\partial f}{\partial t} + v \cdot \nabla f = -\frac{f - f^{eq}}{\varsigma} \tag{2.1.5}$$

The direct origin of the LBM can be found in the Lattice Gas Cellular Automata (LGCA) method which was developed by Frisch, et al. [17]. Some drawbacks of LGCA method led to the LBM by replacing the number of particles, which is called Boolean number, with the single-particle distribution function discretized in velocity and time space. Qian, et al. [18] introduced several sets of lattice systems including the two dimensional nine velocities (D2Q9) and three dimensional nineteen velocities (D3Q19). The current simulation adopted the D2Q9 square lattice system. The single-particle distribution function with the discrete velocity is rewritten as $f_\alpha(x,t) = f(x,e_\alpha,t)$, that is related with the αth velocity in the D2Q9 square lattice system (Figure 1). The discrete velocities in the D2Q9 are given by:

$$\begin{aligned} &e_0 = 0 \\ &e_\alpha = c\left(\cos(\alpha-1)\pi/4\ ,\ \sin(\alpha-1)\pi/4\right) && for\ \alpha = 1,2,3,4 \\ &e_\alpha = \sqrt{2}c\left(\cos(\alpha-1)\pi/4\ ,\ \sin(\alpha-1)\pi/4\right) && for\ \alpha = 5,6,7,8 \end{aligned} \tag{2.1.6}$$

In the above, $c = \Delta x/\Delta t$ is the particle streaming velocity. Δx and Δt indicate the constant lattice space and the time steps, respectively. Thus $f_\alpha(x,t) = f(x,e_\alpha,t)$ can be interpreted as the distribution function with the velocity e_α at the position of x at time t. With the discrete distribution function, velocity set and the BGK model, the BE becomes:

$$\frac{\partial f_\alpha}{\partial t} + e_\alpha \cdot \nabla f_\alpha = -\frac{f_\alpha - f_\alpha^{eq}}{\varsigma} \tag{2.1.7}$$

This form of BE, which is called the discrete velocity model (DVM) (Yu, et al. [19]), can be solved numerically easily compared with the original BE, with its complex nonlinear collision term and Boolean nature.

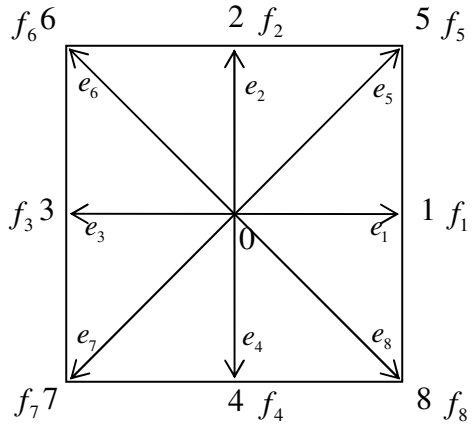

Figure 1. Two dimensional nine velocity square lattice system (D2Q9 model).

To solve the DVM (Eq. 2.1.7) numerically, the finite difference scheme can be applied to discretize the DVM. The application of the finite difference scheme to the DVM with the time step Δt and space step $\Delta x = e_\alpha \Delta t$ gives the formula as:

$$f_\alpha(x + e_\alpha \Delta t, t + \Delta t) - f_\alpha(x,t) = \frac{f_\alpha^{eq}(x,t) - f_\alpha(x,t)}{\tau} \tag{2.1.8}$$

Here $\tau = \varsigma / \Delta t$ denotes the dimensionless relaxation time that is different from the single relaxation time ς in the BE and x is the position in the discretized physical space. This equation is the Lattice Boltzmann Equation (LBE) with the BGK model: the so-called LBGK model. In applying the finite difference scheme, special caution in setting the range of $\varsigma << 1$ is needed for low viscosity flows (Yu, et al. [19]). In the LBGK model, the equilibrium distribution function can be expressed as:

$$f_\alpha^{eq} = \rho w_\alpha \left[1 + \frac{3}{c^2}(e_\alpha \cdot U) + \frac{9}{2c^4}(e_\alpha \cdot U)^2 - \frac{3}{2c^2}U^2 \right] \tag{2.1.9}$$

where U is the fluid velocity and w_α is the weighting factor represented as:

$$\begin{aligned} w_\alpha &= 4/9, \qquad for\ \alpha = 0 \\ w_\alpha &= 1/9, \qquad for\ \alpha = 1,2,3,4 \\ w_\alpha &= 1/36, \qquad for\ \alpha = 5,6,7,8 \end{aligned} \tag{2.1.10}$$

During the calculation of the LBM process, parameters such as density and momentum fluxes can be computed by the following formulas:

$$\begin{aligned} \rho &= \sum_{\alpha=0}^{8} f_\alpha = \sum_{\alpha=0}^{8} f_\alpha^{eq} \\ \rho u &= \sum_{\alpha=0}^{8} e_\alpha f_\alpha = \sum_{\alpha=0}^{8} e_\alpha f_\alpha^{eq} \end{aligned} \tag{2.1.11}$$

In addition, the pressure is given simply as:

$$P = \frac{1}{3}\rho c^2 \tag{2.1.12}$$

The incompressible Navier-Stokes (NS) equations can be derived from LBE using the Chapman-Enskog expansion (Succi [20]). The viscosity, which is one of the most important parameters in viscous fluid dynamics also can be derived from Eq. (2.1.8) as:

$$\upsilon = (\tau - 0.5)c_s^2 \Delta t \tag{2.1.13}$$

where $c_s = c/\sqrt{3}$. This formula indicates that the dimensionless relaxation time τ must always be larger than 0.5. Also, Qian, et al. [18] noticed that τ cannot exceed 2 to ensure numerical stability. Lallemand and Luo [21] considered the difficulty of applying the LBGK model to thermal fluid simulations owing to the dependence on τ, which is not varying parameter in the LBGK model. Because this fixed relaxation time makes the Prandtl number unity when thermal fluids are simulated. They recommended using the general LBE to overcome this drawback of the LBGK model.

2.2 Application to micro gas flows.

To apply the LBM for micro gas flows, special treatment should be considered for the relaxation time (τ) because of its micro-scale and high Knudsen number (*Kn*) flow corresponding to the slip flow regime. In the LBM simulation of macro-scale flows, the difference of density between nodes does not need to be considered. Thus the relaxation time, which is related to density, is fixed through the calculations. In micro-scale applications, on the other hand, the difference of density between nodes can affect the result of simulations significantly by reason of the invalidity of the continuum assumption. As a result, a treatment for the relaxation time is required. The current simulation adopted the method of Nie, et al. [22], who proposed the replacement of τ with τ' as:

$$\tau' = 0.5 + \frac{1}{\rho}(\tau - 0.5) \tag{2.2.1}$$

In the above equation, the ρ can be found by summing distribution functions (Eq. 2.1.11) of each lattice node. Substituting Eq. 2.2.1 to Eq. 2.1.15 gives us the following new expression for the viscosity as:

$$\upsilon = \left(\frac{1}{\rho}(\tau - 0.5)\right) c_s^2 \Delta t \tag{2.2.2}$$

This presents a constant dynamic viscosity $\mu = \rho\upsilon$. The constant dynamic viscosity also implies that current simulations conducted under isothermal condition. Since the mean free path (λ) is linearly proportional to υ, λ can be expressed as:

$$\lambda = \frac{A}{\rho}(\tau - 0.5) \tag{2.2.3}$$

Here $A = 0.388$ is a constant determined by Nie, et al. [22] by comparing numerical results with experiment results. Hence, the relation between the *Kn* and τ is given by:

$$Kn = \frac{\lambda}{H} = \frac{A(\tau - 0.5)}{\rho H} \tag{2.2.4}$$

H denotes the height of the flow domain and the relaxation time (τ) is associated with the viscosity (Eq. 2.2.3). Additionally, the mean free path (λ), which is related with Knudsen number (*Kn*), is linearly proportional to the viscosity. Because of this correlation between *Kn* and τ, the treatment of *Kn* itself can play a role in the handling of τ. Lim, et al. [9] proposed $Kn = \tau / H$ that is on account of the relationship of $\lambda = \tau \Delta x$. In addition, they linked the local *Kn* with Kn_o / P, where subscript o denotes outlet and P is the pressure distribution of the micro channel along the flow direction. Zhang, et al. [7] provided the set of relations of *Kn* with τ for micro flows along with the lattice model of LBM such as D2Q9 and D3Q19 that were mentioned previously. For the D2Q9 model, the *Kn* can be expressed as $Kn = M(\tau - 0.5)/H$, where $M = \sqrt{(8/3\pi)}$.

2.3 Boundary conditions for lattice Boltzmann method.

Many researchers have been applying special boundary conditions to obtain the slip velocity at the wall and capture the other features of micro gas flows in the LBM simulations. The boundary conditions include the no-slip bounce-back, extrapolation scheme, accommodation coefficient scheme, and reflection factor scheme, which will be discussed in this chapter.

Succi [20] presented various boundary conditions for the LBM ranging from the complex boundary condition to the no-slip boundary condition regardless of the scale of the flow. He classified boundary conditions as on-grid and mid-grid for all cases of boundary conditions according to the positions of the wall boundary. First, the on-grid boundary condition has the wall boundary at the middle of the node which is represented by the 1-0-3 line in Figure 1. If this node is a bottom node, the area below the 1-0-3 line is inside the wall and above the 1-0-3 line is the fluid that is neighboring with the wall. However, this on-grid method gives only first-order accuracy for the boundary calculations due to the one-sided relationship with the fluid node. On the other hand, the mid-grid boundary condition guarantees a second order accuracy. In the mid-grid method, at the upper boundary walls, there is an imaginary layer of nodes which represents outside the wall above the 5-2-6 line in Figure 1. Those two layers of nodes are related to each other to calculate the properties of the boundary. The on-grid and mid-grid methods are illustrated in Figure 2.

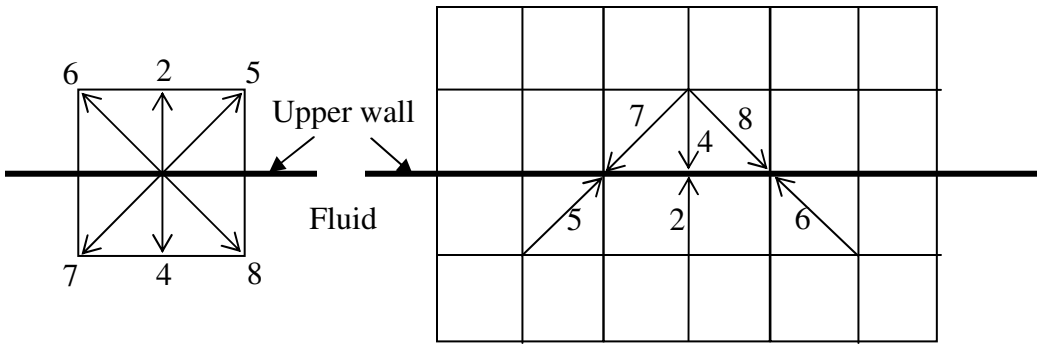

Figure 2. Illustrations of on-grid method (left) and mid-grid method (right) Succi [20].

Chen, et al. [23] proposed the extrapolation scheme boundary condition. It is very similar to the traditional finite difference schemes for macro-scale viscous flows used to obtain the no-slip boundary condition. They simply assume that there is an additional layer (8-4-7 line in Figure 1) of nodes which follows the same calculation as the real fluid node. They related those nodes with the condition of $f_i^{-1} = 2f_i^0 - f_i^1$ where f_i^{-1}, f_i^0, and f_i^1 are the distribution functions on the layer below the wall (8-4-7 line), the wall layer (1-0-3 line), and the fluid layer (5-2-6 line). Unlike previous methods (Noble, et al. [24], Maier, et al. [25]], this extrapolation method does not need special assumptions for the incoming particle distribution function (Chen, et al. [23]). This scheme also provides a second order accuracy for calculations of boundary.

Lim, et al. [9] conducted an LBM simulation for 2D micro-channel gas flows. The remarkable aspect that they used in their process was the varying Knudsen number along the channel with the relation of $Kn = Kn_0 / P(X)$. They also applied on-grid type boundary conditions. In their implementation, the 1-0-3 line (in Figure 1) is regarded as

the wall boundary. At the upper wall, the distribution functions of f_8, f_4 and f_7 are treated as incoming particles from the boundary node to the fluid node and at the lower wall, the distribution functions of f_5, f_2 and f_6 are treated as incoming particles from the fluid node to the boundary node. Thus these distribution functions of incoming particles from boundary nodes are unknown properties which need to be determined by boundary condition treatments. In order to update unknown distribution functions as well as capture the slip velocity at the wall, they used a kind of a specular reflection. Considering the upper wall, f_8, f_4 and f_7 are unknown distribution functions that needed to be specified. The specular boundary conditions used in this implementation are given as :

$$\begin{aligned} f_8(x,y,t) &= f_5(x,y,t) \\ f_4(x,y,t) &= f_2(x,y,t) \\ f_7(x,y,t) &= f_6(x,y,t) \end{aligned} \tag{2.3.1}$$

All these distribution functions are calculated at the same time step. At the inlet and outlet boundary conditions, equilibrium functions (f^{eq}) were adopted to fill blanks of unknown distribution functions. Another special feature they applied for the boundary condition was an extrapolation scheme. They approximate the density and velocity of the boundary node by a second-order polynomial extrapolation. After the approximation, the unknown distribution functions of incoming particles from the boundary node are adjusted by equilibrium functions under the condition that all particles are going to be in

the state of equilibrium following the equilibrium function in their local relaxation process.

Zhang, et al. [7] applied the accommodation coefficient (a) to the specular reflection boundary condition based on mid-grid type method to gain a slip velocity in the LBM simulation of gas flows through the micro-channel. The accommodation coefficient weighs the portion between the specular reflection and diffusive reflection. In the specular reflection, the incoming particles to the wall are reflected as light is reflected from a mirror after the collision. However, in the diffusive reflection, the incoming particles are reflected diffusively without a particular angle of reflection. Since the process of the propagation of distribution functions is taking place horizontally within only boundary nodes without the vertical relationship with neighboring fluid nodes this scheme is slightly different from the traditional mid-grid method. Formulas of boundary conditions at the upper wall which employ the accommodation coefficient (a) are expressed by:

$$\begin{aligned} f_8(x,y,t) &= (1-a) f_5(x-\Delta x, y, t-\Delta t) \\ f_7(x,y,t) &= (1-a) f_6(x+\Delta x, y, t-\Delta t) \\ f_4(x,y,t) &= a f_5(x-\Delta x, y, t-\Delta t) + a f_6(x+\Delta x, y, t-\Delta t) + f_2(x,y,t-\Delta t) \end{aligned} \tag{2.3.2}$$

When the accommodation coefficient (a) is 1, the formulas represent diffusive reflection. In the case that the accommodation coefficient (a) is 0, the formulas represent specular reflection, which is described above in the approach of (Lim, et al. [9]). However, Zhang, et al. [7] take the value of f_5, f_2 and f_6 from the corresponding distribution function at

the previous time step while Lim, et al. [9] take values of f_5, f_2 and f_6 from distribution functions at the same time step as used for f_8, f_4 and f_7.

Tang, et al. [8] simulated 2D micro-channel gas flows by LBM. They focused on the concept that both no-slip bounce-back and specular reflection might not be able to explain the momentum exchange and friction of the real wall reasonably and physically in the micro gas flows. Thus they proposed the reflection coefficient (r_c) that weighs the portion of the contribution of the no-slip bounce-back and specular reflection respectively. These boundary conditions for the upper wall, which are based on the on-grid method, are:

$$\begin{aligned} f_8(x,y,t) &= r_c f_6(x,y,t) + (1-r_c) f_5(x,y,t) \\ f_7(x,y,t) &= r_c f_5(x,y,t) + (1-r_c) f_6(x,y,t) \\ f_4(x,y,t) &= f_2(x,y,t) \end{aligned} \tag{2.3.3}$$

They verified boundary conditions with the reflection coefficient (r_c) by calculating the sums of momentum along the x and y directions. When the sum of y-momentum of the particles after the collision step becomes 0 ($\sum M_y = f_2 - f_4 + f_5 - f_8 + f_6 - f_7 = 0$), this indicates there is no vertical momentum exchange of particles after the collision step. Thus v-velocities at the wall becomes 0 regardless of the value of reflection coefficient (r_c). Whereas, along the sum of x-momentum of particles after the collision, the wall can be regarded as an extremely rough wall at $r_c = 0.5$ and as an absolutely smooth wall at $r_c = 0$. The sum of x-momentum for the reflected particles from the bottom wall after the

collision can be calculated as $\sum M_{x-re} = f_5 - f_6 = (1-2r_c)(f_8 - f_7)$. Thus when $r_c = 0.5$, the sum of momentum becomes 0. This means that the particles reflected back diffusively from the wall without a certain angle of reflection. Tang, et al. [8] regarded this state of wall as an extremely rough wall that particles are reflected perfectly diffusively. See Tang, et al. [8] for more details. In their simulations, when $r_c = 0.7$ the slip velocity at the wall was predicted well.

Succi [26] investigated the slip boundary condition with the reflection coefficient (r) and the slip coefficient (s) which is very similar to that used by Tang, et al. [8] for 3D micro-channel flows. This method, however, is based on the mid-grid scheme:

$$\begin{aligned} f_8(x,y,t) &= rf_6(x+\Delta x, y-\Delta y, t) + sf_5(x-\Delta x, y-\Delta y, t) \\ f_7(x,y,t) &= rf_5(x-\Delta x, y-\Delta y, t) + sf_6(x+\Delta x, y-\Delta y, t) \\ f_4(x,y,t) &= f_2(x, y-\Delta y, t) \end{aligned} \tag{2.3.4}$$

where r is the reflection coefficient and $s = 1-r$ is the slip coefficient. Results from this LBM simulation show that when r goes down below 0.01 slip flows surely take place. Thus he set this value as a critical reflection coefficient. In these simulations, he primarily focused on the range of $0 \le r \le 0.01$.

CHAPTER III

NUMERICAL APPROACH

3.1 The LBM code solution technique

This chapter illustrates the computational procedure of the program “Micro flows_LBM” for current micro flow simulations. Zea and Chambers [3] have adopted and modified the program “anb” developed by Bernsdorf [27] at the C&C Research Laboratories, NEC Europe to perform the micro flow simulations. The program “Micro flows_LBM” is based on the program “anb” and the boundary conditions have been modified from the Zea and Chambers [3]’s program. The program “Micro flows_LBM” is written in FORTRAN 77. The program is composed with three initializing sub-routines and five main sub-routines. To get perfectly converged results, 30000 iteration steps were conducted for micro channel and micro orifice flow simulations.

3.1.1 Sub-routine “read_parameters” and “read_obstacles”

In the sub-routines “read_parameters” and “read_obstacles”, the program reads the initializing parameters and geometry information from the input files “anb.par” and “anb.obs” respectively. The program “Micro flows_LBM” starts its process with these two sub-routines. The configuration file “anb.par” consists of four lines and each line indicates the number of iterations, the density per link, the acceleration term, and relaxation time in order. Among these parameters, the density and the relaxation time are related to the Knudsen number. Thus the density and relaxation time were arranged to change the *Kn*. The obstacle file “anb.obs” has the geometry information of the flows with the two rows of data. The first row represents the x-coordinate of the obstacle nodes, which are interpreted as the obstacle or wall boundary. The second row represents the y-coordinate of the obstacle nodes.

3.1.2 Sub-routine “init_density”

In the program, the values of particle distribution functions f_α are stored in the cell “node (α, x, y)”. α indicates the nine particles in the discrete lattice nodes by the two dimensional nine velocity (D2Q9) model (Figure 1). The initial values of particle distribution functions f_α are imposed to the cell “node (α, x, y)” in the sub-routine “init_density”. The initial values are decided from the equilibrium distribution functions f_α^{eq} (Eq. 2.1.9) with zero initial velocity and it becomes the product of the density and weighting factors (Eq. 2.1.10) of each particles.

3.1.3 Sub-routine “redistribute”

The flows implemented in the current simulations are basically pressure-driven flows. Thus the special inlet boundary condition is used to impose the pressure difference between the inlet and outlet of the channel in the sub-routine “redistribute”. The parameter “accel” from the input file “anb.par” is multiplied to the weighting factor. Thus the product of the density, “accel”, and weighting factors is used to increase or decrease the values of the particle distribution functions of the inlet lattice nodes. At the beginning of every iteration, the program compares the magnitude between the product term and $f_{3,6,7}$. If the values of $f_{3,6,7}$ are greater than the product of the density, “accel”, and weighting factors the product is added to the $f_{1,5,8}$ and subtracted from $f_{3,6,7}$. This addition and subtraction gives an acceleration to the flow from the inlet to the outlet. However, This redistribution method does not allow direct control of the Reynolds number at the inlet of the channel.

3.1.4 Sub-routine “propagate”

The values of particle distribution functions stored in the cell “node (α , x, y)” propagate to the neighboring lattice nodes and stored in temporary cell “n_hlp”. Between the outlet and inlet and between the upper wall nodes and bottom wall nodes, the periodic boundary conditions (Succi [20]) are used.

3.1.5 Sub-routine “bounceback”

In the sub-routine “bounceback”, the boundary conditions for the solid wall or obstacles are incorporated. The propagated particle distribution functions from the fluid nodes to the wall nodes are repositioned rotating to the opposite direction. Then the particle distribution functions from the fluid nodes are sent back to the fluid nodes at the next propagation step. This method illustrates the mechanism that incoming particles to the wall are bounced back into the fluid. You can see more details of the boundary conditions in the sections 3.2 and 3.3 for the current micro flow simulations.

3.1.6 Sub-routine “relaxation”

In this sub-routine, the particle distribution functions after the relaxation process are calculated by the Lattice Boltzmann Equation (Eq 2.1.8). First, the program calculates the integrated local density and the velocity components using Eq 2.1.11. Then the equilibrium distribution functions are computed by Eq 2.1.9. Finally, the particle distribution functions of the equilibrium state, which will be used as the initial values of the distribution functions at the new iteration step, can be calculated by Eq 2.1.8. For the relaxation time of the Eq 2.1.8, new relaxation time model for the micro flows (Nie, et al. [22]) is adopted. Only fluid nodes are considered in the sub-routine “relaxation”.

3.1.7 Sub-routine “write_results”

In this sub-routine, the velocity and pressure are calculated by Eq 2.1.11 and Eq 2.1.12 after the last iteration step is completed. We can find the results of u-velocity, v-velocity and pressure from the output files “ux_array.dat”, “vy_array.dat”, and “pressure.dat” respectively. We can open these output files with Microsoft Excel program. Figure 3 shows the flow chart of the program.

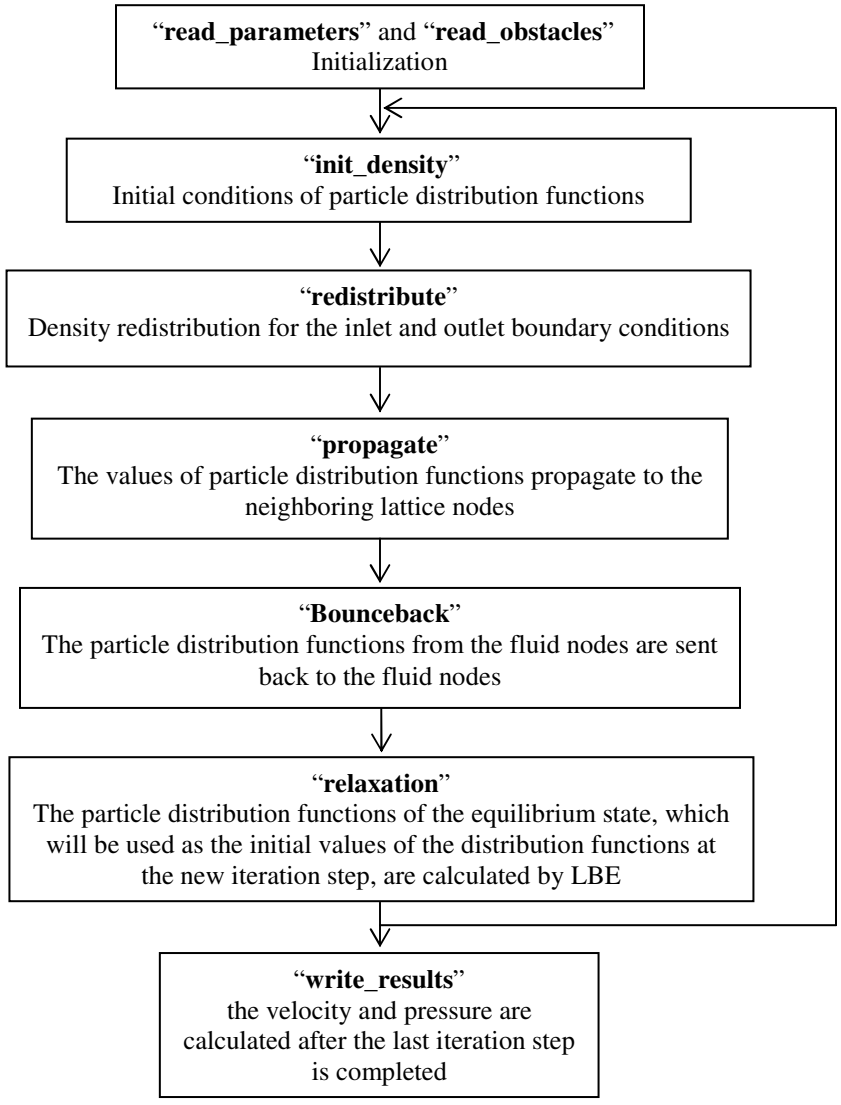

Figure 3. Flow chart of the program “Micro flows_LBM”.

3.2 Application to the micro channel

In the LBM simulation of micro flows, the most important concern is obtaining a correct slip velocity at the wall. This can be done by modeling suitable boundary conditions at the wall that satisfy physical phenomena in micro flows. In general, for LBM simulations, the diffusively scattering particle model has been widely used for very small Kn corresponding to the continuum flow regime (Zhang, et al. [7]). This model is based on the interpretation that reflected particles from the wall after the collision are relaxed along the Maxwell-Boltzmann distribution function. However, in the micro flow LBM simulation, particles are not always reflected diffusively satisfying the Maxwell-Boltzmann distribution function (Zhang, et al. [7]). Thus more general boundary conditions are required for the LBM of micro flows. The present LBM simulation adopts both the no-slip bounce-back boundary condition and slip bounce-back boundary condition to investigate micro channel flows. Figure 4 illustrates a diagram of the micro channel used in current implementations. The ratio of length to height of the channel is maintained as 100 through computations using 21 lattice nodes in the y-direction and 2100 lattice nodes in the x-direction. Tang, et al. [8] conducted the micro channel LBM simulations varying the ratio of length to height from 10 to 1000. However, in their simulations, lattice nodes in the y-direction seemed to be fixed as 10 throughout computations. Nie, et al. [22] simulated the micro channel flows with 10 lattice nodes for the height maintaining the ratio as 100. Lim, et al. [9] also used 10 and 21 lattice nodes in channel height with the fixed ratio 10. On the other hand, Lee and Lin [6] applied relatively finer grids of between 10 and 320 lattice nodes. However, there are still few

results of LBM micro flow simulations using finer grids of more than 20. Thus the simulations implemented by finer grids are highly encouraged in the future work, but the coarse grid is judged adequate for the present work. The dimensions of micro channels used in present study and in the literatures are arranged in Table II.

TABLE II.
The dimensions of the micro channels

	Length (Number of nodes)	**Height (Number of nodes)**	***L/H***
Present study	2100	21	100
Tang, et al. [8]	12 μm ~ 1200 μm	1.2 μm	10 ~ 1000
Nie, et al. [22]	1000	10	100
Lim, et al. [9]	100, 210	10, 21	10
Lee and Lin [6]	N/A	10 ~320	20 ~ 160

The dashed lines in Figure 4 and Figure 5 indicate the real physical wall of the micro channel. There are also imaginary nodes outside the wall to play a role of taking back reflected particles from the wall into fluid nodes. This scheme is based on the mid-grid method (Succi [20]). The height of the channel can be presented as $H = (N_Y - 2)\Delta y$ on the mid-grid method and $H = (N_Y - 1)\Delta y$ on the on-grid method. In the LBM of the square lattice node, used in the current project, the grid scale is based on the relation of $\Delta x = \Delta y = \Delta t = 1$ (Yu, et al. [19]). Therefore *H* becomes 19 on account of $N_Y = 21$. In the no-slip bounce back boundary condition of the current computational study, incoming particles to the wall are bounced away facing an incoming direction of particles. The unknown distribution functions that need to be determined from the boundary condition are f_7, f_4 and f_8 of the wall nodes.

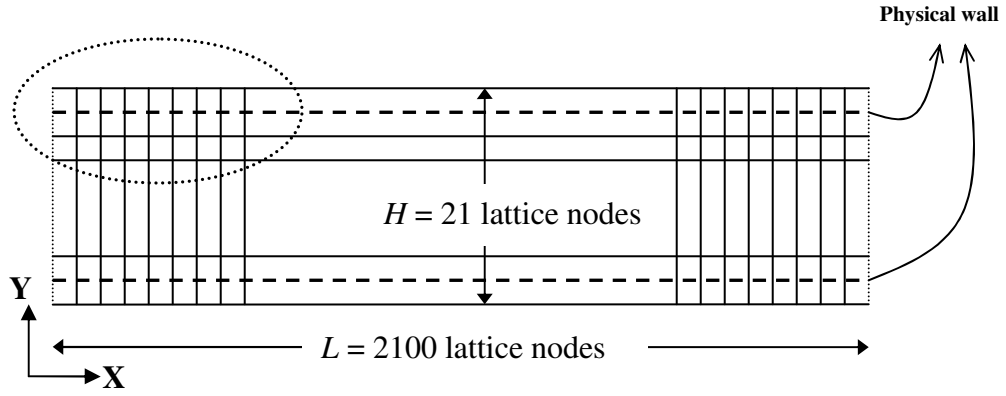

Figure 4. Diagram of the micro channel with *L/H* = 100. Circled area is described again in Figure 4 with the magnified diagram.

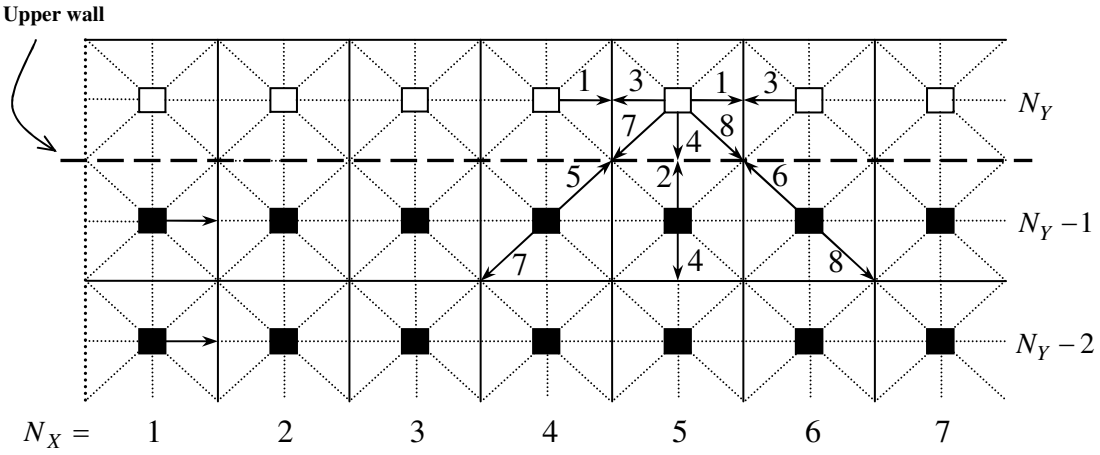

Figure 5. Configurations of lattice nodes based on mid-grid model (Succi [20]) for the upper wall region: □, imaginary nodes outside the wall; ■, fluid nodes.

In Figure 5, the distribution functions of incoming particles, f_5, f_2 and f_6, to the wall are stored in f_7, f_4 and f_8 of the wall node (5, N_Y) respectively. Then they are propagated to adjacent fluid nodes at the next time step. This no-slip bounce-back boundary condition for the upper wall can be expressed as:

$$
\begin{aligned}
f_4(x,y,t) &= f_2(x, y-\Delta y, t-\Delta t) \\
f_7(x,y,t) &= f_5(x-\Delta x, y-\Delta y, t-\Delta t) \\
f_8(x,y,t) &= f_6(x+\Delta x, y-\Delta y, t-\Delta t)
\end{aligned} \tag{3.2.1}
$$

For the slip boundary condition, the reflection factor (Tang, et al. [8]) and the accommodation coefficient (Zhang, et al. [7]), which are discussed in the literature review section of section 2.1, are applied to get more accurate results. The reflection factor (r_f) (Tang, et al. [8]) weighs the proportion of the contribution of no-slip bounce-back and specular schemes respectively as mentioned in the previous chapter. The unknown distribution functions of the slip boundary condition with r_f of the upper wall node are found as:

$$
\begin{aligned}
f_4(x,y,t) &= f_2(x, y-\Delta y, t-\Delta t) \\
f_7(x,y,t) &= r_f f_5(x-\Delta x, y-\Delta y, t-\Delta t) + (1-r_f) f_6(x+\Delta x, y-\Delta y, t-\Delta t) \\
f_8(x,y,t) &= r_f f_6(x+\Delta x, y-\Delta y, t-\Delta t) + (1-r_f) f_5(x-\Delta x, y-\Delta y, t-\Delta t)
\end{aligned} \tag{3.2.2}
$$

The accommodation coefficient (a) (Zhang, et al. [7]) is incorporated to determine the portion of diffusive reflection and specular reflection. As a approaches 1 the boundary condition becomes diffusive reflection and as a approaches 0 the boundary condition becomes specular reflection. On the other hand, unlike the reflection factor and no-slip bounce-back schemes, unknown distribution functions f_7 and f_8 of the wall node are not influenced by distribution functions f_5 and f_6, which are coming from the opposite direction and having the nature of no-slip. The slip bounce-back boundary condition with a for the upper wall can be represented by:

$$
\begin{aligned}
f_4(x,y,t) &= af_5(x-\Delta x, y-\Delta y, t-\Delta t) + af_6(x+\Delta x, y-\Delta y, t-\Delta t) \\
&\quad + f_2(x, y-\Delta y, t-\Delta t) \\
f_7(x,y,t) &= (1-a) f_6(x+\Delta x, y-\Delta y, t-\Delta t) \\
f_8(x,y,t) &= (1-a) f_5(x-\Delta x, y-\Delta y, t-\Delta t)
\end{aligned}
\tag{3.2.3}
$$

3.3 Application to the micro orifice.

The reflection factor (r_f) is also applied to the boundary condition of the micro orifice flow because it shows excellent results in the micro channel simulations. Figure 6 presents a schematic of the micro orifice, which simulates an array of micro filter fibers. The factors such as geometry and dimension of the hole and the edge-shape of the filter can affect the result of the filtration. In the current simulation, the two dimensional square orifice is used. Also, north and south boundaries of the channel part are periodic sides that are neighbored with other arrays of micro filter fiber. The ratio of the open orifice area to the total area (*h/H*) is selected as 0.6 with 2 μm height in order to compare results with the work of Ahmed and Beskok [5]. In addition, 20 lattice nodes in the y-direction and 340 lattice nodes in the x-direction are used setting the ratio of length to height as 17. Ahmed and Beskok [5] used various dimensions for their micro filters. The dimensions of the micro orifice and micro filters used in current study and in the literature are specified in Table III.

TABLE III.
The dimensions of the micro orifice and micro filters

	***H* (μm)**	***h* (μm)**	***l* (μm)**	***L/H***
Present study	2	1.2	0.8	17
Ahmed and Beskok [5]	6	3.6	2.4	17
	4	2.4	1.6	17
	2	1.2	0.8	17
	1	0.6	0.4	17

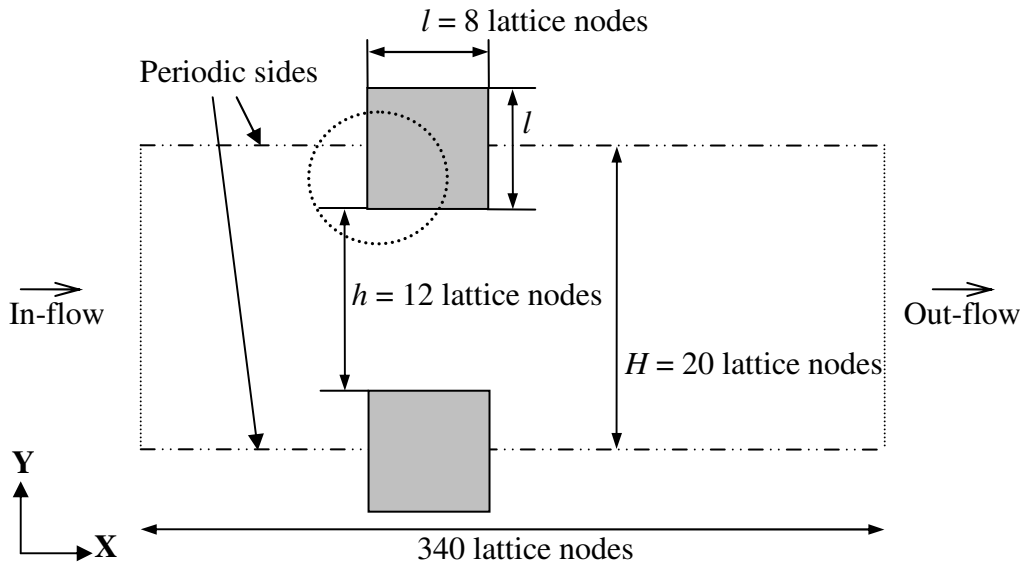

Figure 6. Diagram of the micro orifice. the ratio of the open orifice area to the total area (*h/H*) is 0.6. Circled area is described again in Figure 6 with the magnified diagram.

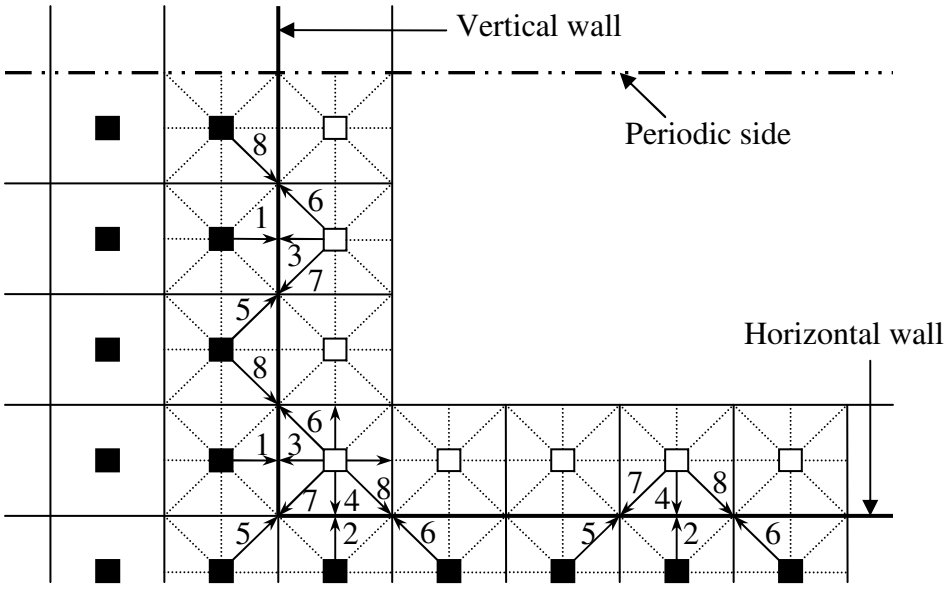

Figure 7. Configurations of lattice nodes for the micro orifice flow: □, imaginary nodes inside the orifice; ■, fluid nodes.

Figure 7 is the magnified part from the circled area of Figure 6 showing the configuration of lattice nodes and the relationship between the distribution functions of the nodes. Since the flow direction varies along the position of the orifice wall, rearrangement of the

distribution functions is required in order to apply the reflection factor (r_f) to the boundary condition of the micro orifice. For instance, in the case of vertical wall nodes of the orifice, Figure 7, the flow direction is top to bottom along the orifice wall. Thus the unknown distribution functions that need to be determined are f_3, f_6 and f_7. These are different from the case of horizontal north wall of micro channel. The boundary condition with r_f for vertical wall nodes of the micro orifice can be expressed as:

$$\begin{aligned} f_3(x,y,t) &= f_1(x-\Delta x, y, t-\Delta t) \\ f_6(x,y,t) &= r_f f_8(x-\Delta x, y+\Delta y, t-\Delta t) + (1-r_f) f_5(x-\Delta x, y-\Delta y, t-\Delta t) \\ f_7(x,y,t) &= r_f f_5(x-\Delta x, y-\Delta y, t-\Delta t) + (1-r_f) f_8(x-\Delta x, y+\Delta y, t-\Delta t) \end{aligned} \qquad (3.3.1)$$

In the above formulas, unknown distribution functions f_3, f_6 and f_7 are influenced by distribution functions f_1, f_5 and f_8 of neighboring fluid nodes.

For the corner wall node of the orifice, more special treatments are needed because the corner wall nodes are affected by both vertical direction flow and horizontal direction flow. For example, the distribution function f_7 of the corner wall node can be correlated with f_8 (from vertical flow), f_6 (from horizontal flow) and f_5 (as a no-slip bounce-back component). Other distribution functions f_6 and f_8 of the corner wall node also can be handled the identical way. The boundary condition for the horizontal wall node does not necessitate any more special treatment, as it is the same as the micro channel horizontal

wall nodes. The boundary condition with r_f of the corner wall node for the micro orifice is presented as:

$$
\begin{aligned}
f_3(x,y,t) &= f_1(x-\Delta x, y, t-\Delta t) \\
f_4(x,y,t) &= f_2(x, y-\Delta y, t-\Delta t) \\
f_6(x,y,t) &= r_f f_8(x-\Delta x, y+\Delta y, t-\Delta t) + 0.5(1-r_f) f_5(x-\Delta x, y-\Delta y, t-\Delta t) \\
&\quad +0.5(1-r_f) f_7(x+\Delta x, y+\Delta y, t-\Delta t) \\
f_7(x,y,t) &= r_f f_5(x-\Delta x, y-\Delta y, t-\Delta t) + 0.5(1-r_f) f_8(x-\Delta x, y+\Delta y, t-\Delta t) \\
&\quad + 0.5(1-r_f) f_6(x+\Delta x, y-\Delta y, t-\Delta t) \\
f_8(x,y,t) &= r_f f_6(x+\Delta x, y-\Delta y, t-\Delta t) + 0.5(1-r_f) f_5(x-\Delta x, y-\Delta y, t-\Delta t) \\
&\quad + 0.5(1-r_f) f_7(x+\Delta x, y+\Delta y, t-\Delta t)
\end{aligned}
\tag{3.3.2}
$$

3.4 Analytical solution.

Karniadakis, et al. [2] developed the unified flow model for micro flows based on the compressible Navier-Stokes (NS) equation considering velocity scaling and rarefaction effects. This unified model is valid for the wide Knudsen number (Kn) regime from the continuum to the transitional flow regime and independent of the gas type. In addition, the unified flow model has shown excellent agreement with results of the Direct Simulation Monte Carlo (DSMC) method and solutions of the linearized Boltzmann equation (Karniadakis, et al. [2]). The unified flow model was also validated in micro filter flows at various *Kn* between 0.023 and 0.185 (Ahmed and Beskok [5]). Besides, the experimental results for micro flow are not available in the literature for micro channel flows and micro orifice flows under conditions matching the current simulation (Zea and Chambers [3]). On this account, the unified flow model of (Karniadakis, et al. [2]) is the best analytical solution for the present study at this time. Their non-dimensionalized velocity profile (U^*) for the micro channel flow, which is a function only of the *y* to the channel height and the Knudsen number (Kn), is:

$$U^*(y, Kn) \equiv \left[\frac{-(y/h)^2 + (y/h) + Kn/(1-bKn)}{(1/6) + Kn/(1-bKn)} \right] \tag{3.4.1}$$

When $b = 0$ in the above equation the unified model corresponds to the first order accuracy boundary condition. When $b = -1$, as used in current implementations, the model has second order accuracy for a wide range of *Kn*. The unified flow model for

micro channels also provides the pressure distribution (Karniadakis, et al. [2]). The pressure distribution function is expressed as:

$$\tilde{P}^2 - 1 + 2(6+\overline{\alpha})\frac{2-\sigma_v}{\sigma_v} Kn_o(\tilde{P}-1) + 2(6b+\overline{\alpha})\frac{2-\sigma_v}{\sigma_v} Kn_o^2 \log_e\left(\frac{\tilde{P}-bKn_o}{1-bKn_o}\right) = B\left(1-\frac{x}{L}\right) \tag{3.4.2}$$

Where $\tilde{P}(x) = P(x)/P_o$, which represents the pressure at x coordinate normalized with the exit pressure. B is a constant with a value that makes $\tilde{P}(0) = P_i/P_o$. The P_i and P_o denote the pressure at inlet and outlet respectively. The constant σ_v is the tangential momentum accommodation coefficient that represents tangential momentum exchange of particles with the wall. When the surface is rough with the characteristic length scale of molecules, the particles reflected from the wall diffusively and σ_v becomes close to 1 (Lee and Lin [6]). The real engineering cases such as air or CO_2 on machined brass or shellac have $\sigma_v = 1$ (Lee and Lin [6]). Thus σ_v is selected as 1, which represents diffuse reflection, for the current simulation. σ_v is expressed as $\sigma_v = (m_i - m_r)/(m_i - m_w)$. Here m_i, m_r and m_w are the tangential momentum of incoming, reflected particles and the wall respectively (Karniadakis, et al. [2]). m_w is 0 for the stationary wall. The tangential momentum accommodation coefficient σ_v and the accommodation coefficient (a) of Zhang, et al. [7] for the boundary conditions both act similarly and weigh the proportion of diffusive reflection and specular reflection. However they are not directly related. The

constant $\overline{\alpha}$ has the value of 2.2 that was determined for nitrogen in a finite-length channel with the ratio $L/h = 20$ (Karniadakis, et al. [2]).

CHAPTER IV

RESULTS AND DISCUSSION

4.1 Simulation results of micro channel flows.

In the LBM simulations of micro channel flows, three different Knudsen numbers (*Kn*) of 0.00194, 0.0194 and 0.194, which indicate the continuum, slip and transitional flow regimes respectively, were used. For boundary conditions at micro channel walls, the no-slip bounce-back, accommodation coefficient (a) (Zhang, et al. [7]) and reflection factor (r_f) (Tang, et al. [8]) were applied to obtain a slip velocity at the boundary wall. Velocities from the result are non-dimensionalized by the mean velocity with the y coordinate non-dimensionalized by the height of channel. Results at the location *x/L* = 0.9 are presented to show the fully developed velocity profile of the flow. Velocity profiles are analyzed by comparing with the analytical unified flow result of Karniadakis, et al. [2] which is discussed in section 4.3. First of all, simulations were conducted to acquire proper slip velocity at the wall in the slip flow regime. Thus for *Kn* = 0.0194, the accommodation coefficient and reflection factor were incorporated into the bounce-back boundary condition.

Figure 8 shows the results of the accommodation coefficient scheme. Overall, the accommodation coefficient scheme does not provide very good results. The profiles do not agree well with the analytical result. When $a = 0.3$ the velocity profile does not have a parabolic shape especially around the wall area and it has a huge deviation in the center of the flow. As the accommodation coefficient increases, the velocity profile approaches the parabolic shape and the slip velocity decreases. When $a = 0.99$, which indicates

almost a diffusive reflection, the velocity profile still has deviations in the center and wall area of the flow.

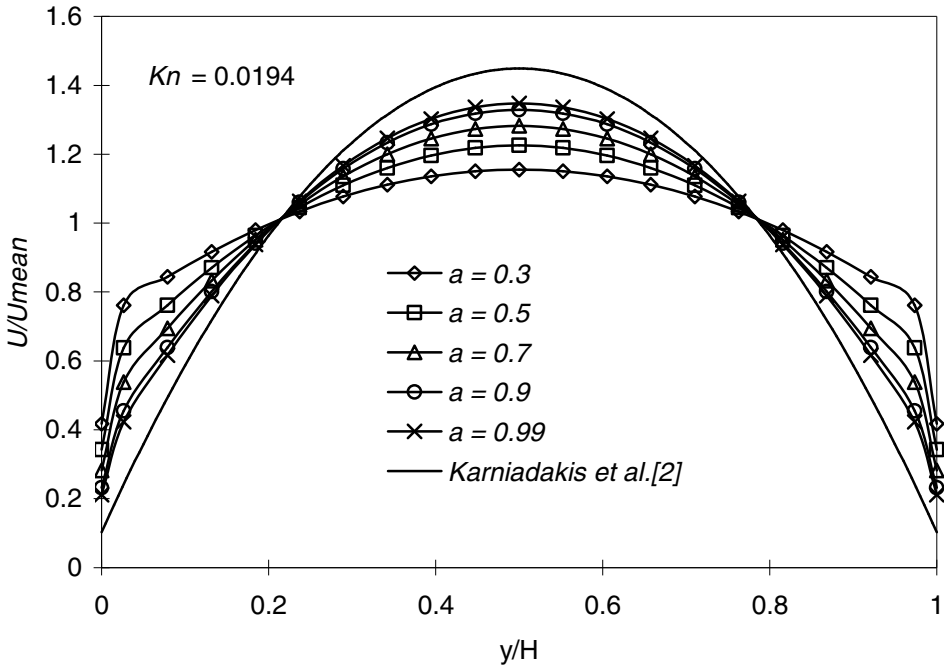

Figure 8. Non-dimensionalized velocity profiles with various values of the accommodation coefficient (*a*) for *Kn* = 0.0194.

A diffusive reflection implies natural movement of particles after collisions obeying the Maxwell-Boltzmann distribution function. For diffusive phenomena there should be enough particle distribution functions to represent real physical phenomena with large numbers of particles. However, in the LBM, only nine particle distribution functions represent movements of all particles due to the discretized velocity model such as D2Q9. Thus the differences of the accommodation coefficient scheme from the analytical result can be attributed in part to this trait of the LBM.

The results of the reflection factor boundary condition are presented in Figure 9. Velocity profiles for the reflection factor scheme show similar patterns as the accommodation

coefficient scheme as the reflection factor increases. When $r_f = 0.3$, there are large deviations in the center and wall area of the flow. As the reflection factor increases, the velocity profile approaches the result of Karniadakis, et al. [2]. In the case of $r_f = 0.7$, deviations of the maximum velocity and slip velocity are approximately 2.38% and 38% each. In the case of $r_f = 0.85$, the velocity profile is in very good agreement with the analytical result showing approximately only 0.3% and 2.34% deviations in the maximum velocity and slip velocity.

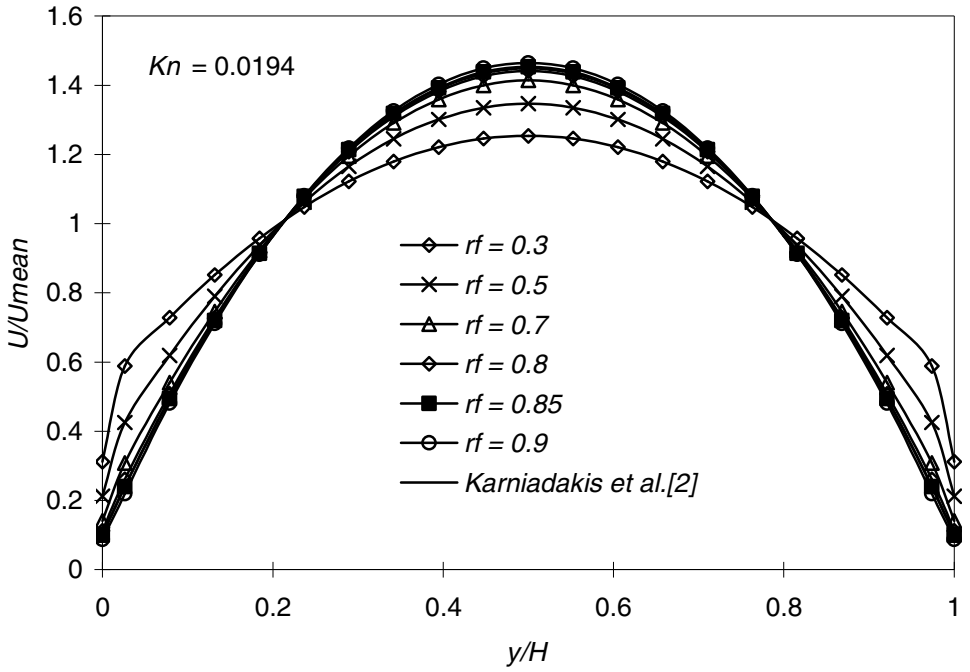

Figure 9. Non-dimensionalized velocity profiles with various values of the reflection factor for *Kn* = 0.0194.

Figure 10 illustrates the case of $r_f = 0.85$ again comparing with the no-slip bounce-back scheme under the *Kn* = 0.0194. It is shown that the velocity profile applied with no-slip bounce-back boundary condition also provides slip velocity at the wall in the slip flow regime. The no-slip bounce-back boundary condition originated for the case of no-slip

velocity at the wall. However, the velocity profile of the no-slip bounce-back shows approximately 35.65% underestimated slip velocity in the wall. In the maximum velocity, the profile overestimates the velocity approximately 2.52%. The current simulations were not conducted to determine the relation between the *Kn* and r_f through the entire slip flow regime changing both parameters. Since increasing r_f shows a trend of making slip velocity decrease in the case of *Kn* = 0.0194, One may speculate that the optimum reflection factor may decrease with increasing *Kn* in the slip flow regime.

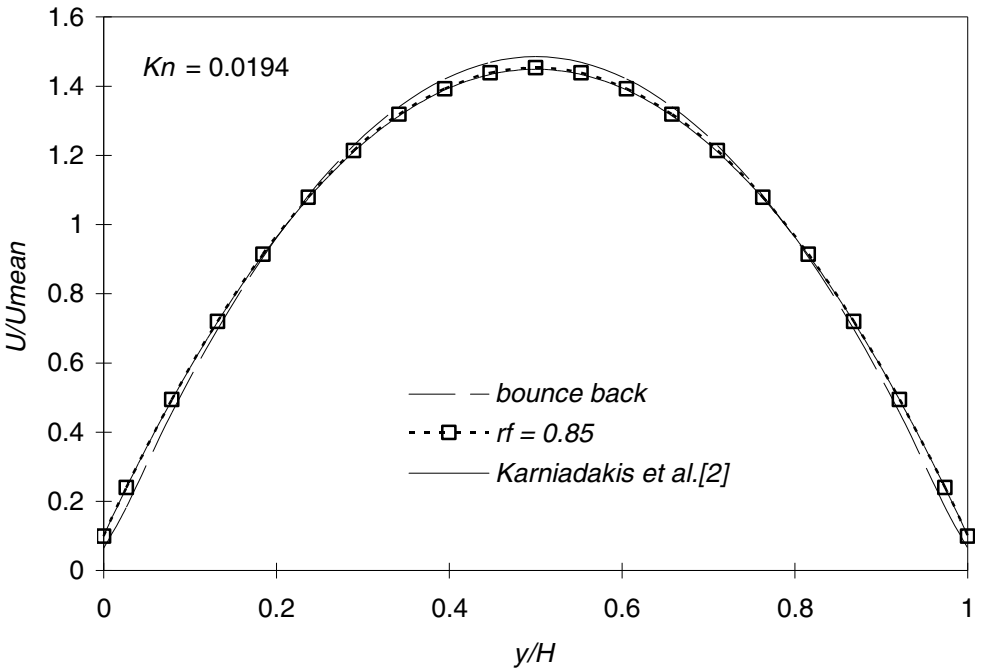

Figure 10. Non-dimensionalized velocity profiles of the reflection factor of 0.85 and no-slip bounce-back schemes for *Kn* = 0.0194.

Additionally, simulations were performed with *Kn* = 0.00194 and *Kn* = 0.194. These simulations were used to assess the capability of the LBM using the boundary conditions applied to slip flow for the continuum and transitional flow regimes. Figure 11 shows the comparison of velocity profiles between the no-slip bounce back and the reflection factor

of 0.85 for Kn = 0.00194, corresponding to the continuum flow regime. These two boundary condition schemes yield consistent velocity profiles in the continuum flow regime, although there are small deviations from the analytical result around the wall area including a slip velocity at the wall. In the very low Knudsen number flow, the slip velocity should be very small. However, the current simulations calculate noticeably large slip velocity at the wall in the continuum flow regime.

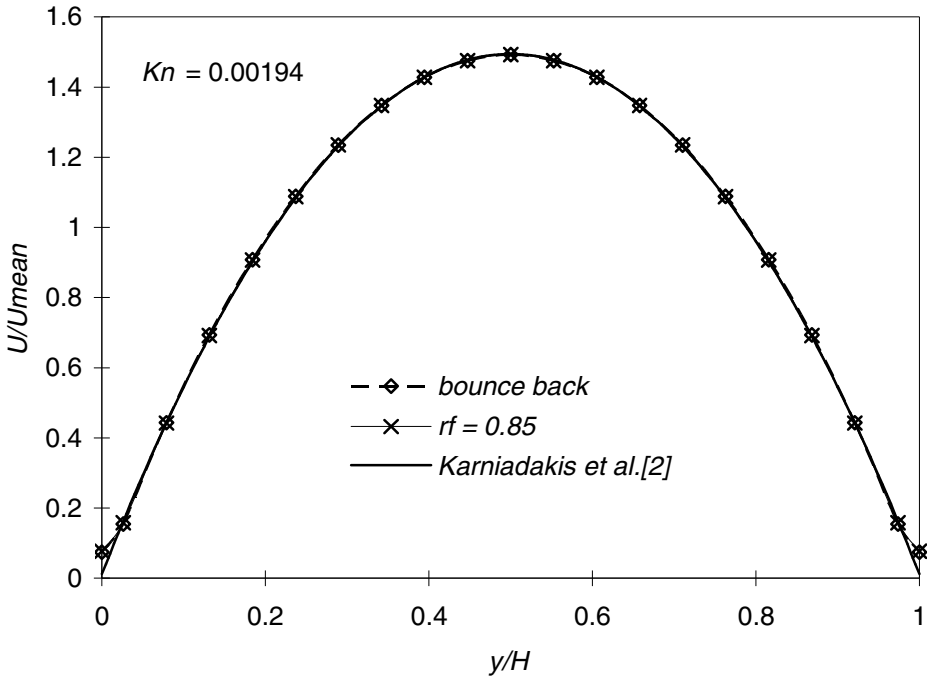

Figure 11. Non-dimensionalized velocity profiles with the no-slip bounce-back and reflection factor boundary conditions for Kn = 0.00194 corresponding to the continuum flow regime.

The comparison of results between the no-slip bounce back, reflection factor and accommodation coefficient boundary for Kn = 0.194 are presented in Figure 12. The results of current LBM simulations for the transitional flow regime are not well predicted. This discrepancy may be attributed to the single relaxation time model used in the current simulations. The single relaxation time model represents only one collision step for particles until they approach the equilibrium state after collisions. In the low Knudsen

number flow, there are more particles in the same area compared with the high Knudsen number flow. Thus the space that particles can move before they collide with other particles is relatively small in the low Knudsen number flow. So the particles can relax with small relaxation time and just one collision step. On the other hand, in the high Knudsen number flows, due to the rarefied effect, several particle collision steps can be reasonably expected before approaching the equilibrium state. Hence more sophisticated relaxation time models should be developed to improve the performance of the LBM simulation for high Knudsen number flows.

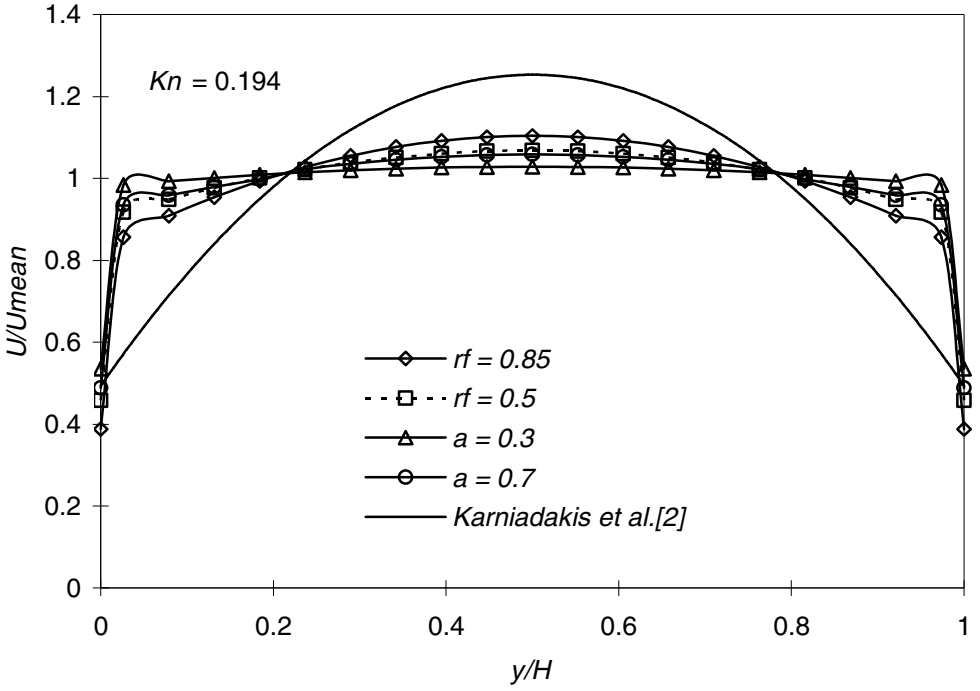

Figure 12. Non-dimensionalized velocity profiles with the no-slip bounce-back, reflection factor and accommodation coefficient boundary conditions for *Kn* = 0.194 corresponding to the transitional flow regime.

One of the purposes of this project is to confirm the compressibility effect for micro flows. This can be performed by investigating the pressure distribution along the flow direction of the micro channel. Since the reflection factor scheme showed proper velocity

profile that agreed well with the literature, the pressure distribution for the case of $r_f = 0.85$ and Kn = 0.0194 was investigated. The decreasing pressure distribution along the flow direction, which is non-dimensionalized by outlet pressure, is shown in Figure 13. The x coordinate is normalized by the length of the micro channel. Due to decreasing pressure and density, centerline velocities increase along the micro channel.

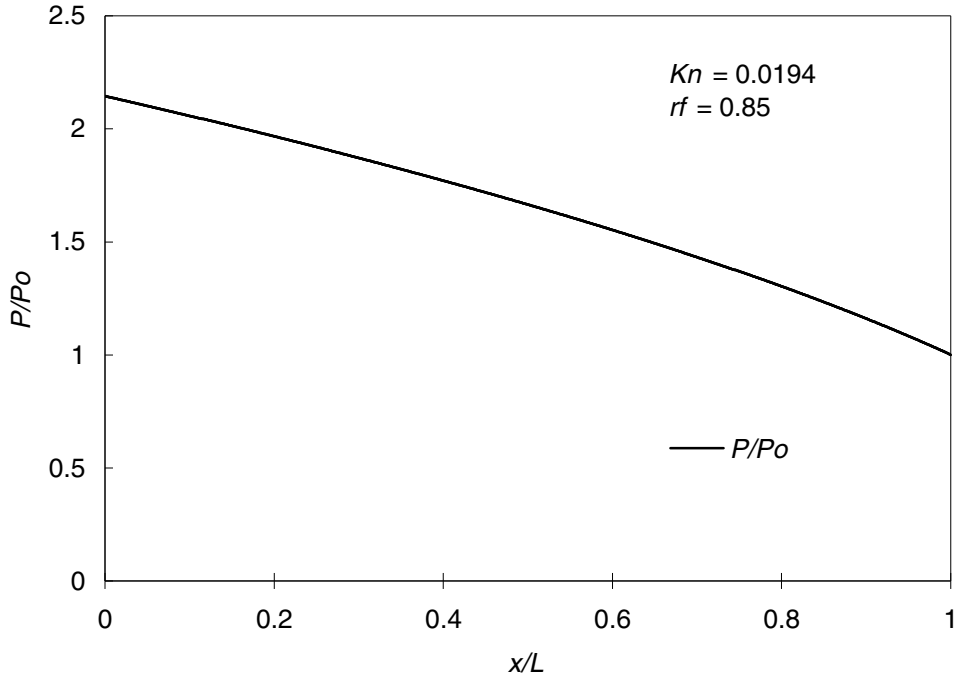

Figure 13. Non-dimensionalized pressure distribution at the center of the flow along the x-direction of the micro channel. The pressure is non-dimensionalized by outlet pressure.

Figure 14 shows increasing velocity along the channel which is a typical feature of micro channel flow (Lim, et al. [9]). Similar results can be found in many papers (Agrawal and Agrawal [4], Lee and Lin [6], Lim, et al. [9], Jeong, et al. [14]). The micro flows implemented in the current project are pressure driven flows. To impose the pressure difference between the inlet and outlet the density redistribution method is used for the inlet boundary condition, which increases or decreases the density of the inlet according

to the densities of the inlet nodes at the end of each iteration time step. However the density redistribution method does not seem to provide accurate results at the inlet and outlet of the micro channel. We can see small jumped velocities in the inlet and outlet of the velocity profile in Figure 14.

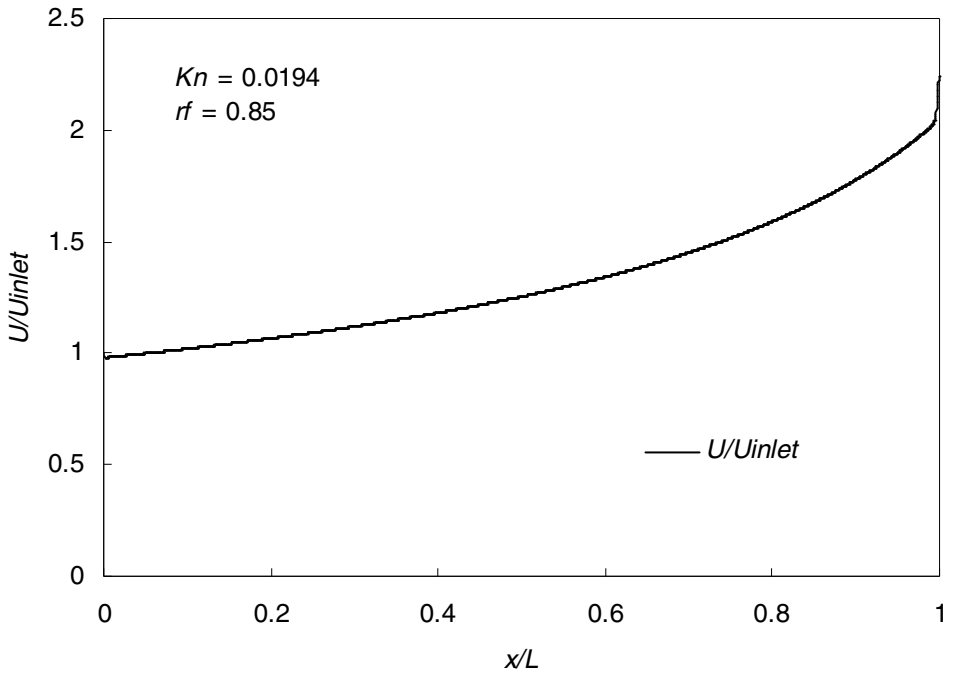

Figure 14. Non-dimensionalized velocity distribution at the center of the flow along the x-direction of the micro channel. The velocity is non-dimensionalized by inlet velocity.

In order to ensure the compressibility effect is simulated accurately, the difference between the non-linear pressure distribution of micro channel flows and the corresponding linear pressure distribution of the incompressible flow can be considered. This is the traditional comparison used in the literature. This non-linearity of the pressure distribution along the channel is presented in Figure 15. Here, $Pl = P_o + (P_i - P_o)(1-x)$ is the linear pressure distribution which is a function of only x coordinate. P_i and P_o

indicate inlet and outlet pressure respectively. The LBM simulation differences from the linear pressure distribution are in excellent consistency with the analytical result of Karniadakis, et al. [2] in the figure. Another feature of these results is the location where the maximum pressure deviation is indicated. The maximum point is shifted toward the outlet at approximately x/L = 0.58. In the results of Lee and Lin [6], Lim, et al. [9], the non-linearity decreases when the Knudsen number increases. This means that the compressibility effects decrease in more rarefied gas flows.

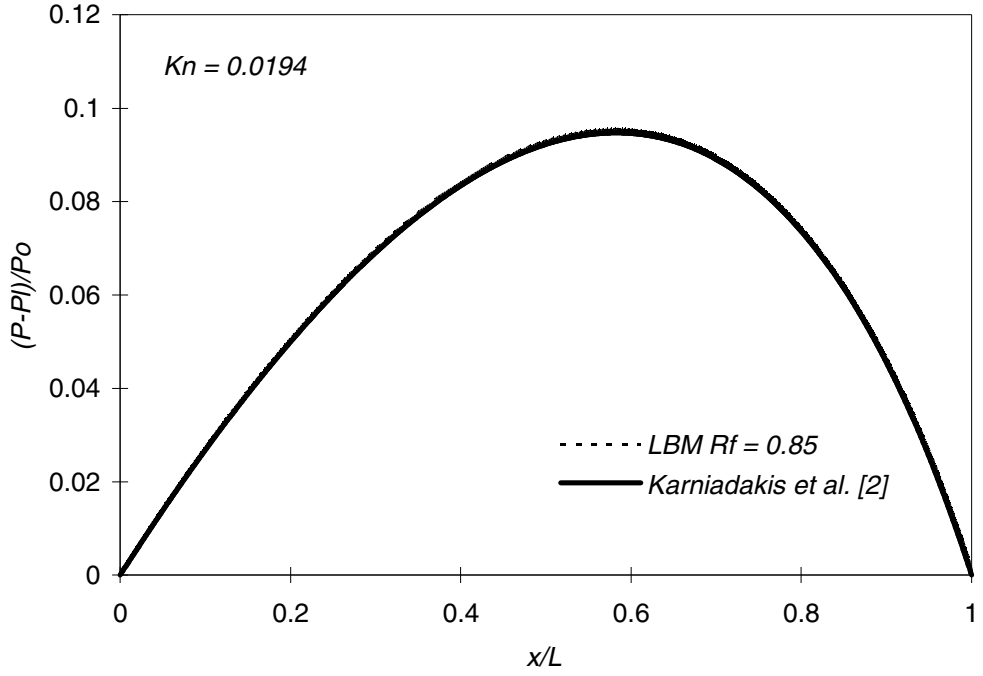

Figure 15. Non-linearity of the pressure distribution along the micro channel.

4.2 Simulation results of micro orifice flows.

The main purpose in conducting simulations of gas flows through the micro orifice, which simulates an array of micro filter fibers, is to investigate the action of flows passing the micro filter. 2 μm for channel height and 1.2 μm for the height of open orifice area are used to keep the ratio of the open orifice are to the total area as 0.6. The length from the inlet to the outlet of the channel is 34 μm. The schematic view of the micro orifice is illustrated in Figure 6. The simulations are performed at atmospheric conditions, with air assumed to flow through the micro orifice. The temperature of the micro orifice and surrounding areas is kept at 298 K, so the simulation has isothermal conditions throughout the computations. The reflection factor boundary condition, which shows excellent agreement with the analytical result in the simulations of micro channel flows, is applied to the wall of orifices in the limit of slip flow regime. The simulations are conducted for Kn = 0.02628 and at five different Reynolds numbers. In the simulations of micro orifice flows, the Knudsen number is calculated using the orifice open area height.

Figures 16, 17 and 18 present velocity, density and pressure variations along the micro orifice at five different Reynolds numbers with Kn = 0.02628. The orifice is located at the x-lattice nodes between 87 and 94. The dimension of velocity is m/s. In the current LBM code, particle streaming velocity (c) in the Eq. 3.1.6 is set as 1, which is different from the real particle streaming velocity under the atmospheric conditions. The real particle streaming velocity can be calculated as $c = \sqrt{3}c_s$ where c_s is the speed of sound under the simulation condition. The velocity of the flow is decided by Eq. 3.1.11. Since the velocity

has the same ratio to the particle streaming velocity (c and e_α have same dimensions from Eq. 3.1.6) the real velocity of the flows can be calculated by multiplying the results of the LBM by $\sqrt{3}c_s$.

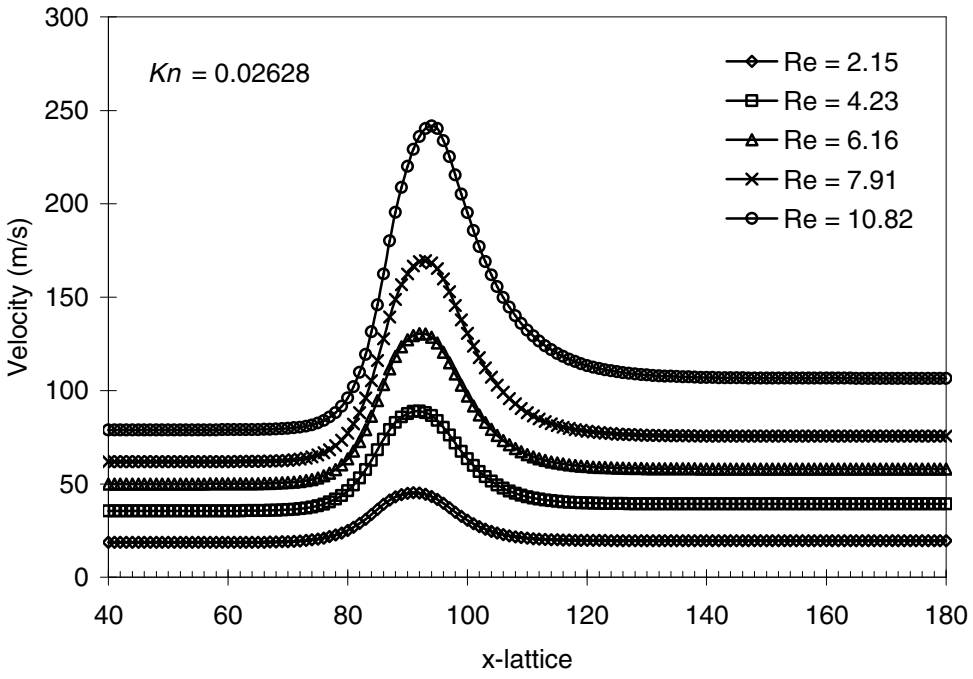

Figure 16. Velocity distributions along the flow direction of the micro orifice at five different Reynolds numbers.

In Figure 16, as the flow approaches the orifice it accelerates and hits its maximum velocity. However maximum velocities occur at different locations for the different Reynolds numbers. For example, when the Re = 10.82 the flow reaches its maximum velocity at x-lattice = 94 while the maximum velocity occurs at x-lattice = 91 when the Re = 2.15. This means that the flow at high Reynolds number accelerates for a somewhat longer time compared with the low Reynolds number flow. This phenomenon, which owes to the compressibility effect, can be more severe for the high Reynolds number flow than for the low Reynolds number flow. After the flow reaches its maximum

velocity, it begins to slow down and reaches constant velocity again. Another indicator of the compressibility effect can be found here. There are deviations between the incoming constant velocity to the orifice and the outgoing constant velocity from the orifice for the different Reynolds numbers. At Re = 10.82 the outgoing velocity is increased about 35.11% compared to the incoming velocity. However at Re = 2.15 the flow shows only approximately a 5% difference between incoming and outgoing velocities. Thus it is confirmed that different densities resulting from the compressibility effect produce greater changes at high Reynolds number. Similar behavior also can be observed in the results of Ahmed and Beskok [5]. The velocity distributions are correlated very well with the density and pressure distributions in Figures 17 and 18. The density distribution is a direct evidence of the compressibility effect (Ahmed and Beskok [5]).

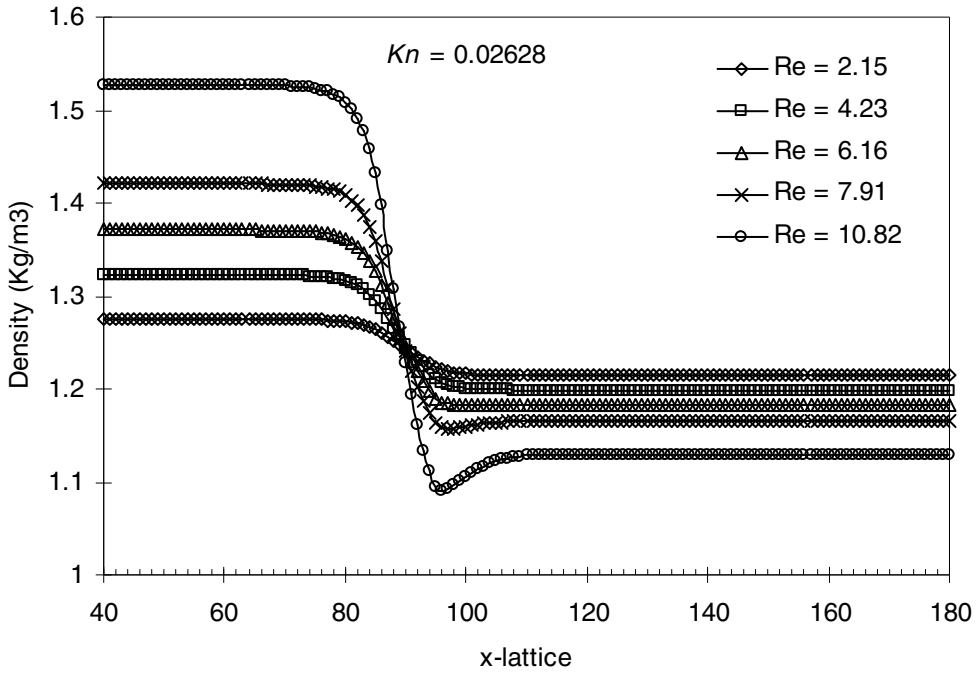

Figure 17. Density distributions along the flow direction of the micro orifice at five different Reynolds numbers.

Figure 17 shows that the largest density-drop take place at the highest Reynolds number of 10.82. The densities maintain nominally constant values until the flows come to the orifice, and densities drop to minimum values after the orifice. However the location where the minimum density takes place does not correspond to the location where the maximum velocity is captured. At lower Reynolds numbers of 2.15, 4.23 and 6.16, after densities reach their minimum they remain as constant without upturning regions. On the other hand, at the higher Reynolds number of 7.91 and 10.82, there are small upturning regions in which densities increase, reaching constant values after a short distance. This is unexpected behavior which is different than the results of Ahmed and Beskok [5]. This feature requires additional study.

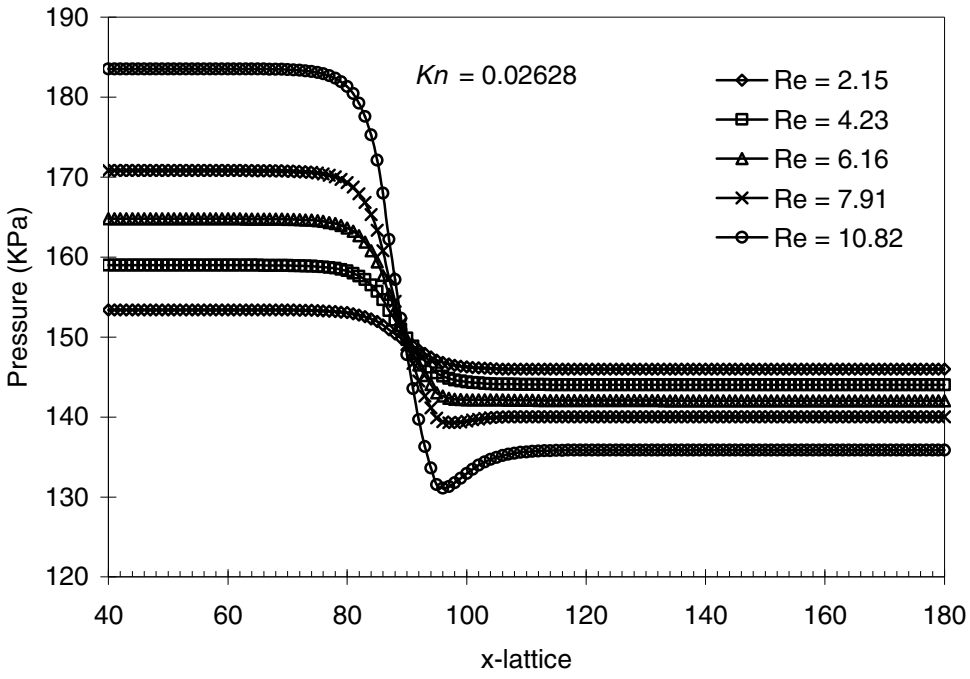

Figure 18. Pressure distributions along the flow direction of the micro orifice at five different Reynolds numbers.

The pressure distributions in Figure 18 show the same pattern as the density distributions. The largest pressure drop occurs at the highest Reynolds number. In addition, there are upturning regions of the pressure right after the orifice for two higher Reynolds number cases. However unlike the density distribution, these upturning regions of the pressure also can be found in the results of Ahmed and Beskok [5].

For more detailed analysis, upstream velocities of the orifice and velocities inside the orifice along the y direction are investigated as Ahmed and Beskok [5] did. Simulation results at Re = 6.16 are selected to be investigated. Figure 19 presents streamwise u-velocity profiles at various locations upstream from the orifice. Again, the micro orifice region is between x-lattice 87 and 94.

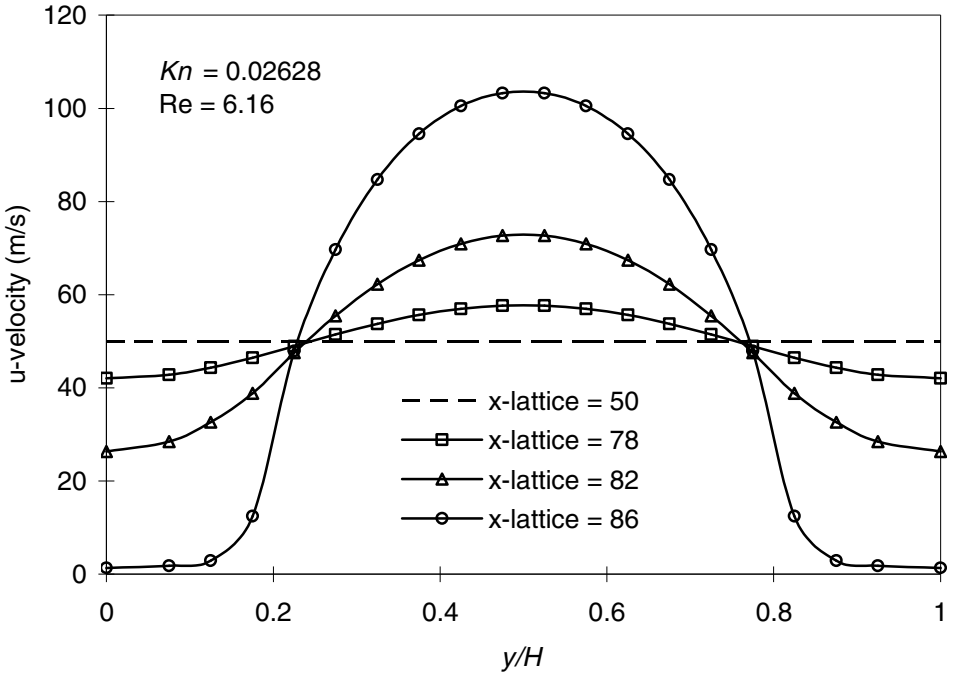

Figure 19. Streamwise u-velocity profiles at various upstream locations of the micro orifice.

The orifice open area is between *y/H* = 0.2 and *y/H* = 0.8. At x-lattice = 50, which is far away from the orifice, the flow has a uniform streamwise velocity. As the flow approaches the orifice, the profile begins to develop a parabolic shape. The center of the flow starts to accelerate and wall area of the flow starts to decelerate due to the blockage of the orifice. At x-lattice = 86, which is just before the inlet of the orifice, u-velocities around the wall area have almost vanished due to the orifice. The streamwise v-velocity profile at x-lattice = 86 is presented in Figure 20. According to this v-velocity profile from just before the orifice, the flow starts to accelerate to the y-direction following the vertical wall of the orifice as expected. However v-velocity starts to decrease after the end of the vertical wall and becomes zero at the center of the orifice where the flow is accelerated to the x-direction.

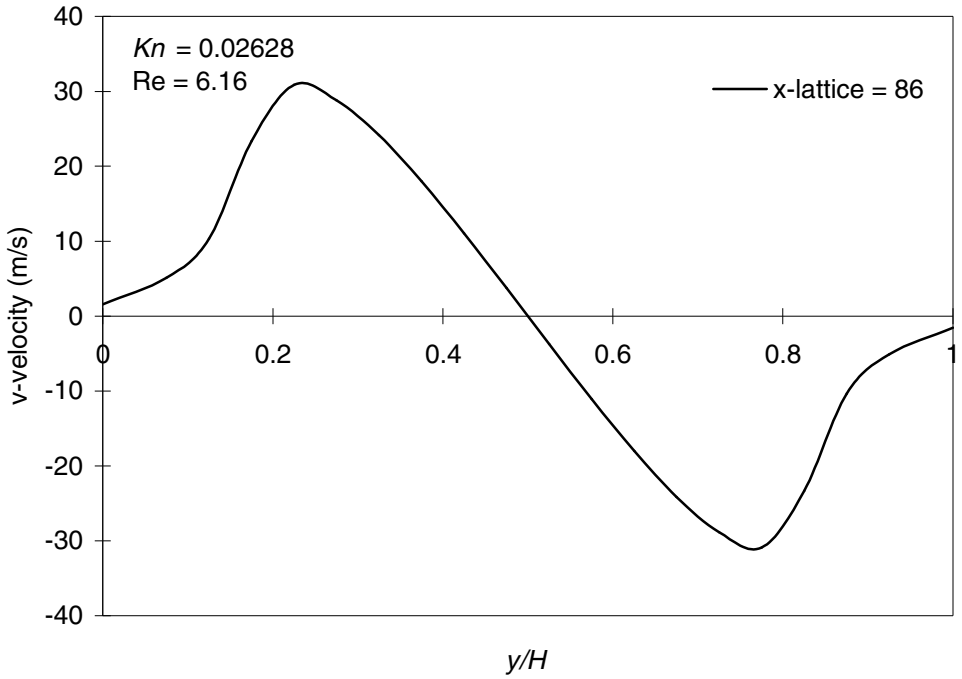

Figure 20. Streamwise v-velocity profile at x-lattice = 86.

Figure 21 shows streamwise u-velocity profiles at various locations inside the orifice. The interesting thing is that the flows already have developed the parabolic shape before entering the orifice. The flows at very low Reynolds numbers implemented in the present study have relatively little inertia compared with the macro scale viscous flows so the flows react quickly to the surroundings. This helps explain the parabolic shape of the flow before entering the orifice. Thus there are only slight changes in the shapes of the velocity profiles along the various locations inside the orifice. In addition, as discussed previously, the flow reaches its maximum velocity before the end of the orifice at the x-lattice = 92.

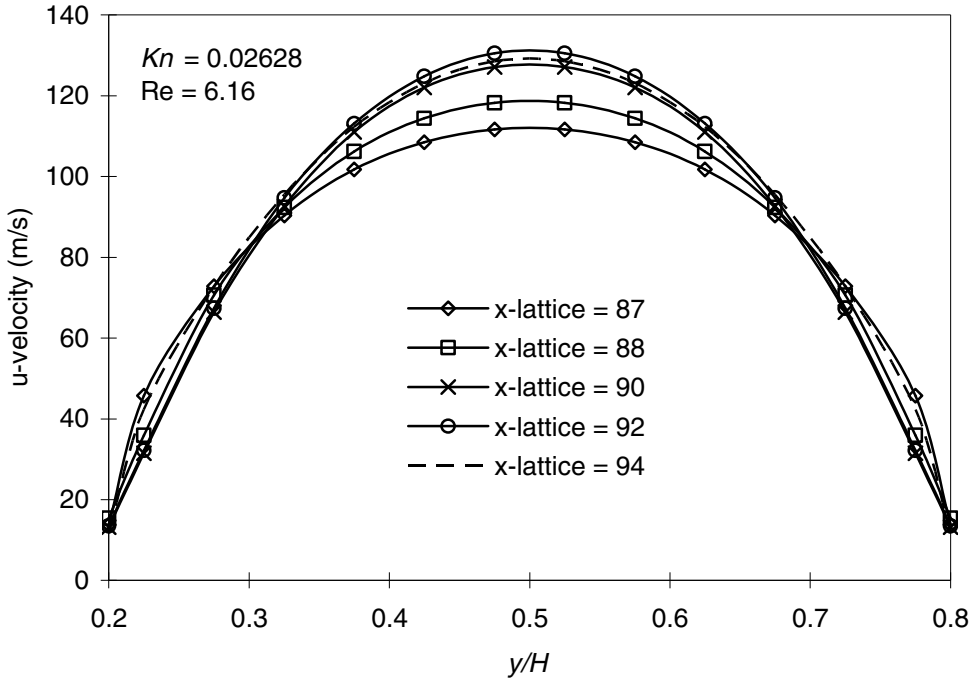

Figure 21. Streamwise u-velocity profiles at various locations inside the orifice.

Figure 22 presents u-velocity profiles at the end of the orifice under various Reynolds numbers. It is shown that as the Reynolds number increases, velocities at all the streamwise locations from the wall to the center of the flow also increase. Overall

behavior of the flows can be observed from Figures 23, 24 and 25. The figures illustrate the velocity vector fields around the orifice from x-lattice = 70 to x-lattice = 120. The x-coordinate is normalized by the length between x-lattice = 70 and x-lattice = 120. The y-coordinate also is normalized by the channel height. When the Re = 7.91 the velocity vector field shows reasonably expected behavior of the flows. As the uniform flows approach the orifice the flows are accelerated to the center of the orifice. Then the flows pass away from the orifice and become uniform flows again after the orifice.

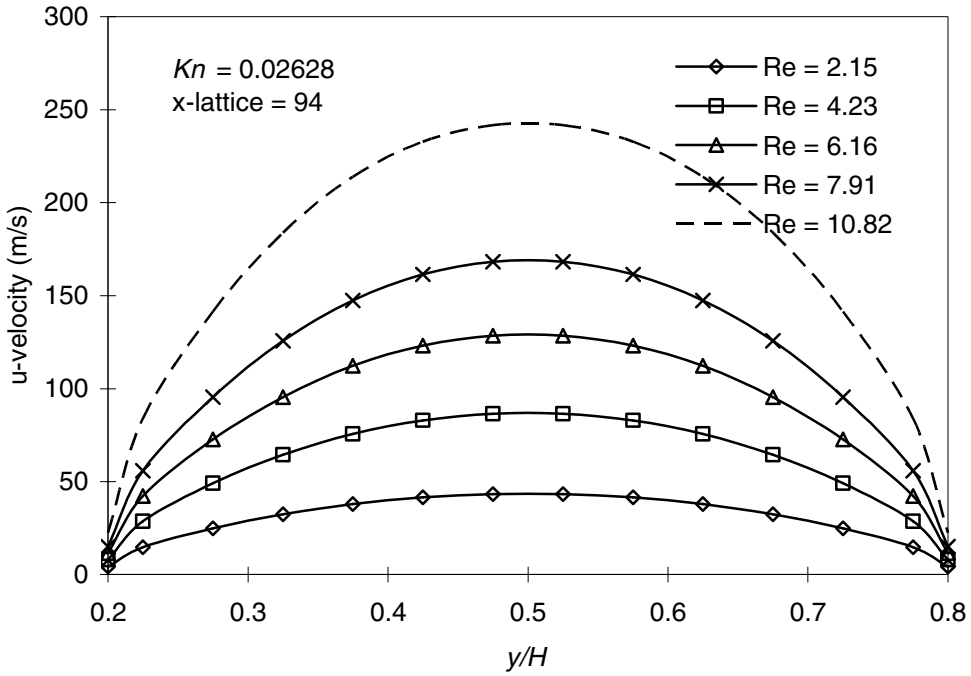

Figure 22. Streamwise u-velocity profiles at the end of the orifice at various Reynolds numbers.

These velocity vector fields do not indicate the strength of the velocities, as all the arrows in the vector fields have the same length. Figures 24 and 25 show interesting features right after the orifice. When the Re = 10.82 and Re = 12.01 the behaviors of the flows before the orifice are almost same with the case of Re = 7.91. However, there are separation areas at the downstream corner of the orifice.

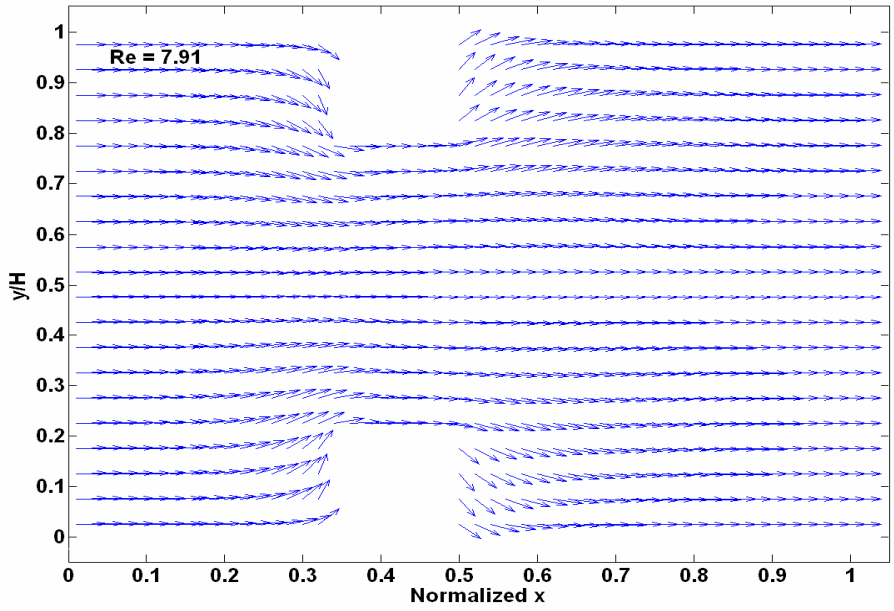

Figure 23. Velocity vector field around the micro orifice at Re = 7.91.

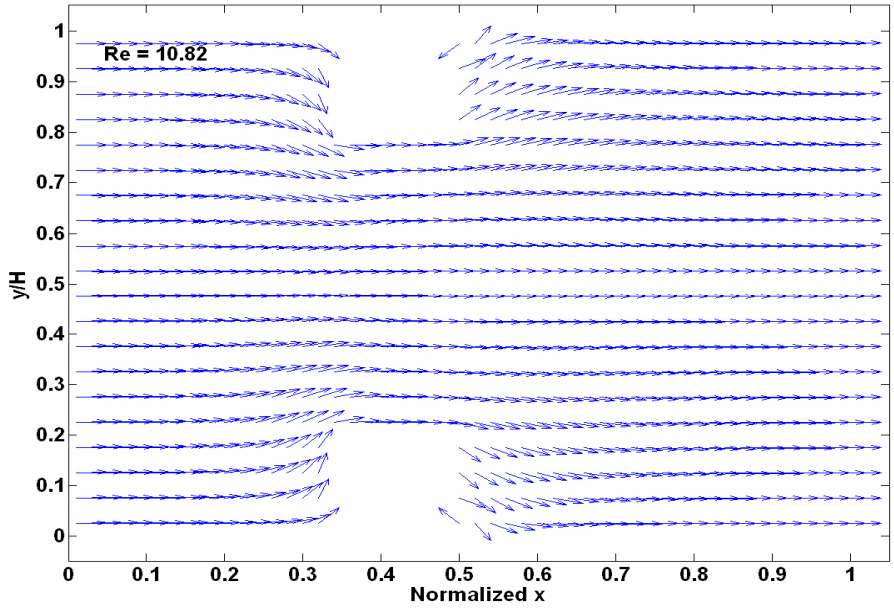

Figure 24. Velocity vector field around the micro orifice at Re = 10.82.

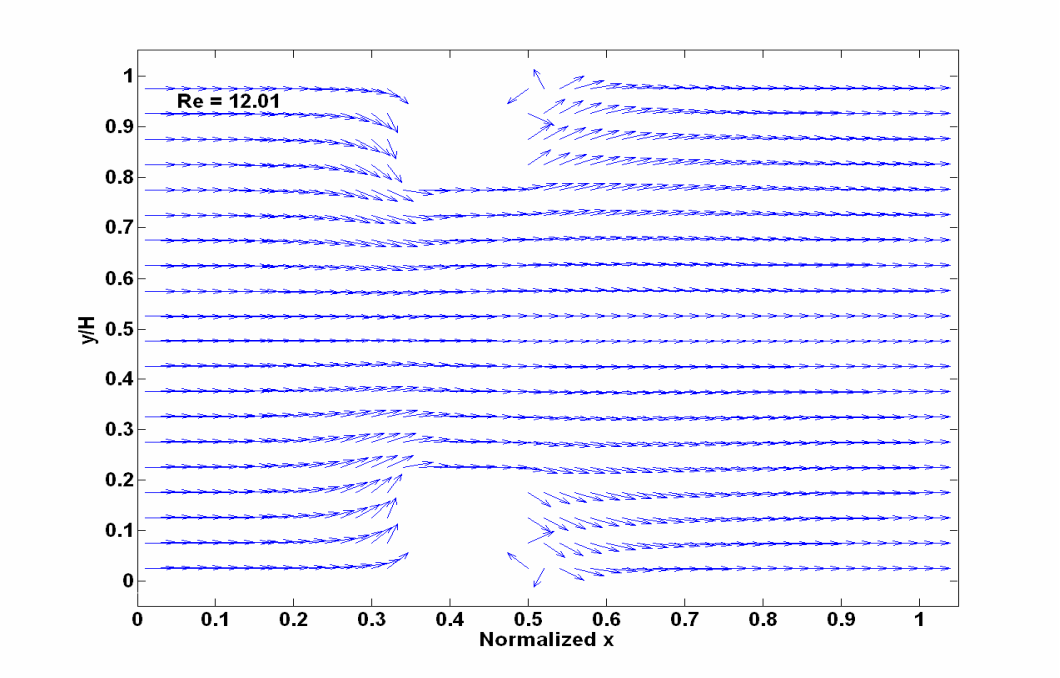

Figure 25. Velocity vector field around the micro orifice at Re = 12.01.

The separation area seems to start to appear at the Re = 10.82. When the Re = 12.01 the separation area is larger than the case of Re = 10.82. These behaviors of the flow need to be studied more in future work. The limited number of lattice points in this region should be noted.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The LBM simulations for isothermal micro gas flows through the micro channel and micro orifice were performed successfully with three types of boundary conditions at Knudsen numbers of 0.00194, 0.0194, and 0.194 and at various Reynolds numbers between 2 and 12. The no-slip bounce back, accommodation coefficient (a), and reflection factor (r_f) schemes were applied to the boundary condition.

First, for the micro channel flows, the reflection factor boundary condition provides very good results which match well with the unified flow model of Karniadakis, et al. [2] while the no-slip bounce-back and accommodation coefficient boundary conditions show deviations from the unified model. In the case of $r_f = 0.85$, the velocity profile is in very good agreement with the analytical result showing approximately only 0.3% and 2.34% deviations in the maximum velocity and slip velocity at Kn = 0.0194. For Kn = 0.00194, the no-slip bounce-back and reflection factor of 0.85 yield consistent velocity profiles although there are deviations from the unified model around the wall area including a slip velocity at the wall. However, For Kn = 0.194, corresponding to the transitional flow regime, the results of the current LBM simulations are not accurate. The pressure profiles from the current LBM simulations confirm the compressibility effect and show excellent consistency with the analytical result of Karniadakis, et al. [2] at Kn = 0.0194

Second, the results of the micro orifice simulations also are in good agreement with the results of Ahmed and Beskok [5]. The compressibility effects can be confirmed well from

the velocity, density, and pressure distributions along the flow direction. From the results, the largest velocity jump, pressure, and density drop through the orifice occurs at the highest Reynolds number of 10.82. This result shows that the compressibility effect becomes significant at the larger Reynolds numbers. On the other hand, the upturning regions of the density and pressure profiles at the end of the orifice are unexpected, interesting results at the higher Reynolds numbers. The behaviors of the flow from the upstream to the downstream of the orifice are predicted well in the current simulations, as shown by the velocity profiles and velocity vector fields.

5.2 Recommendations

The current LBM simulations provide excellent results for isothermal micro gas flows through the micro channel at $Kn = 0.0194$. The simulations also provide good results for the micro orifice in the slip flow regime. However, for the large Knudsen number flows, which fall into the transitional flow regime, the current simulations show very large deviations from the analytical results and appear inaccurate. Thus more studies should be performed to advance the LBM for the large Knudsen number flows corresponding to the transitional and free molecular flow regimes. More sophisticated relaxation time models to replace the single relaxation time model of the current simulation may be an approach to make improvements. In addition, the sophisticated relaxation time model would be able to permit grid refinement around the interesting regions for more accurate results (Zea and Chambers [3]). The upturning region of the density distributions of the micro orifice at high Reynolds numbers larger than 10 and the separation area right after the micro orifice also need further study. Finally, new boundary conditions for the inlet and outlet should be developed replacing the density redistribution method, which makes setting pressure differences and flow rates very complicated at the inlet and outlet of the micro channel.

REFERENCES

1 Arkilic, E. B., Breuer, K. S., and Schmidt, M. A., 1994, "Gaseous Flow in Microchannels," *Proceedings of the 1994 International Mechanical Engineering Congress and Exposition*, Chicago, IL, USA, ASME, FED 197, pp. 57-66.

2 Karniadakis, G., Beskok, A., and Aluru, N., 2005, *Microflows and Nanoflows: Fundamentals and Simulation*, Springer, New York.

3 Zea, I., and Chambers, F. W., 2005, "Lattice Boltzmann Computations of Micro Channel and Micro Orifice Flows," *XI Congreso Internacional Anual de la Sociedad Mexicana de Ingeniería Mecánica*, Morelia, Michoacán, Mexico, Sociedad Mexicana de Ingenieria Mecánica.

4 Agrawal, A., and Agrawal, A., 2006, "Three-Dimensional Simulation of Gaseous Slip Flow in Different Aspect Ratio Microducts," *Physics of Fluids*, 18, 10, pp. 103604/1-11.

5 Ahmed, I., and Beskok, A., 2002, "Rarefaction, Compressibility, and Viscous Heating in Gas Microfilters," *Journal of Thermophysics and Heat Transfer*, 16, 2, pp. 161-170.

6 Lee, T., and Lin, C. L., 2005, "Rarefaction and Compressibility Effects of the Lattice-Boltzmann-Equation Method in a Gas Microchannel," *Physical Review E*, 71, 4, pp. 046706/1-10.

7 Zhang, Y., Qin, R., and Emerson, D. R., 2005, "Lattice Boltzmann Simulation of Rarefied Gas Flows in Microchannels," *Physical Review E*, 71, 4, pp. 047702/1-4.

8 Tang, G. H., Tao, W. Q., and He, Y. L., 2003, "Gas Flow Study in MEMS Using Lattice Boltzmann Method," *First International Conference on Microchannels and Minichannels*, Rochester, New York, ASME, 1, pp. 389-396.

9 Lim, C. Y., Shu, C., Niu, X. D., and Chew, Y. T., 2002, "Application of Lattice Boltzmann Method to Simulate Microchannel Flows," *Physics of Fluids*, 14, 7, pp. 2299-2308.

10 Chen, C. S., Lee, S. M., and Sheu, J. D., 1998, "Numerical Analysis of Gas Flow in Microchannels," *Numerical Heat Transfer; Part A: Applications*, 33, 7, pp. 749-762.

11 Ahmed, I., and Beskok, A., 2001, "Numerical Simulation of Gas Flows in Micro-Filters," *2001 ASME International Mechanical Engineering Congress and Exposition*, New York, NY, ASME, MEMS-Vol. 3, pp. 3077-3083.

12 Tang, G. H., Tao, W. Q., and He, Y. L., 2005, "Lattice Boltzmann Method for Gaseous Microflows Using Kinetic Theory Boundary Conditions," *Physics of Fluids*, 17, 5, pp. 058101/1-4.

13 Mott, D. R., Oran, E. S., and Kaplan, C. R., 2001, "Microfilter Simulations and Scaling Laws," *Journal of Thermophysics and Heat Transfer*, 15, 4, pp. 473-477.

14 Jeong, N., Lin, C. L., and Choi, D. H., 2006, "Lattice Boltzmann Study of Three-

Dimensional Gas Microchannel Flows," *Journal of Micromechanics and Microengineering*, 16, 9, pp. 1749-1759.

15 Chen, S., and Doolen, G. D., 1998, "Lattice Boltzmann Method for Fluid Flows," *Annual Review of Fluid Mechanics*, 30, pp. 329-364.

16 Bhatnagar, P. L., Gross, E. P., and Krook, M., 1954, "A Model for Collision Processes in Gases, I. Small Amplitude Processes in Charged and Neutral One-Component System," *Physical Review*, 94, 3, pp. 511-525.

17 Frisch, U., Hasslacher, B., and Pomeau, Y., 1986, "Lattice-Gas Automata for the Navier-Stokes Equations," *Physical Review Letters*, 56, pp. 1505 -1508.

18 Qian, Y. H., D'Humieres, D., and Lallemand, P., 1992, "Lattice BGK Models for Navier-Stokes Equation," *Europhysics Letters*, 17, 6, pp. 479-484.

19 Yu, D., Mei, R., Luo, L. S., and Shyy, W., 2003, "Viscous Flow Computations with the Method of Lattice Boltzmann Equation," *Progress in Aerospace Sciences*, 39, 5, pp. 329-367.

20 Succi, S., 2001, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*, Oxford University Press, Oxford.

21 Lallemand, P., and Luo, L. S., 2000, "Theory of the Lattice Boltzmann Method: Dispersion, Dissipation, Isotropy, Galilean Invariance, and Stability," *Physical Review E*, 61, 6A, pp. 6546-6562.

22 Nie, X., Doolen, G. D., and Chen, S., 2002, "Lattice-Boltzmann Simulation of Fluid Flows in MEMS," *Journal of Statistical Physics*, 107, 112, pp. 279-289.

23 Chen, S., Martinez, D., and Mei, R., 1996, "On Boundary Conditions in Lattice Boltzmann Methods," *Physics of Fluids*, 8, 9, pp. 2527-2536.

24 Noble, D. R., Chen, S., Georgiadis, J. G., and Buckius, R. O., 1995, "A Consistent Hydrodynamic Boundary Condition for the Lattice Boltzmann Method," *Physics of Fluids*, 7, 1, pp. 203-209.

25 Maier, R. S., Bernard, R. S., and Grunau, D. W., 1996, "Boundary Conditions for the Lattice Boltzmann Method," *Physics of Fluids*, 8, 7, pp. 1788-1801.

26 Succi, S., 2002, "Mesoscopic Modeling of Slip Motion at Fluid-Solid Interfaces with Heterogeneous Catalysis," *Physical Review Letters*, 89, 6, pp. 064502/1-4.

27 Bernsdorf, J., 2001, "Free Lattice Boltzmann Code 'Anb'," http://www.ccrl-nece.de/lba/, accessed May 25, 2007.

APPENDIX A

LATTICE BOLTZMANN CODE FOR MICRO CHANNEL FLOWS

```
program  micro channel
************************************************************************
*    anb                                                               *
*                                                ... means: anb's not best *
*                                 -    -   -                           *
*    A lattice Boltzmann CFD teaching code.                            *
*                                                                      *
*          Joerg BERNSDORF                                             *
*          C&C Research Laboratories, NEC Europe Ltd.                  *
*          Rathausalle 10                                              *
*          D-53757 Sankt Augustin, Germany                             *
*                                                                      *
*          e-mail: bernsdorf@ccrl-nece.de                              *
*          WWW: http://www.ccrl-nece.de/bernsdorf/                     *
*                                                                      *
*    Corrections and hints from Thomas Zeiser and Wolfgang Hamm         *
*    are gratefully acknowledged.                                      *
*                                                                      *
*    Copyright (C) 1998-2001  Joerg BERNSDORF /                        *
*               C&C Research Laboratories, NEC Europe Ltd.             *
*                                                                      *
*    This program is free software; you can redistribute it and/or     *
*    modify it under the terms of the GNU General Public License as    *
*    published by the Free Software Foundation; either version 2 of    *
*    the License, or (at your option) any later version.               *
*                                                                      *
*    This program is distributed in the hope that it will be useful,   *
*    but WITHOUT ANY WARRANTY; without even the implied warranty of    *
*    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the     *
*    GNU General Public License for more details.                      *
*                                                                      *
*    You should have received a copy of the GNU General Public         *
*    License along with this program; if not, write to the Free        *
*    Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139,     *
*    USA.                                                              *
*                                                                      *
*    Last change: 2001/03/29                                           *
************************************************************************
*          The anb code has been modified partly to perform the simulations for *
*          Micro channel gas flows by                                  *
*                                                                      *
*                                                  Taiho Yeom          *
*                                                  Oklahoma State University *
*                                                  May 2007            *
************************************************************************
c.....short introduction
c
c.....anb simulates incompressible viscous flow as governed by the
c     Navier-Stokes equations.
c     But anb is not a Navier-Stokes solver.
c     anb is a lattice Boltzmann solver, which means, the velocity
c     discrete boltzmann equation is solved here, using a single time
c     relaxation (BGK) collision operator.
c
c     For more details on this method read:
c
```

```
c                   Y.H.Quian, D.D'Humieres and P.Lallemand,
c                   Lattice BGK Models for Navier-Stokes Equation
c                   Europhys. Lett., 17 (6), pp. 479-484 (1992)
c
c    Other methods belonging to the lattice gas / lattice Boltzmann
c    familiy exist, but this one is very simple, quite good and used
c    by lots of scientists working on this topic.
c
c    This is a very simple implementation of the lattice BGK scheme,
c    not a very efficient and not a very memory safing one.
c    *  Do not misunderstand this as a good proposal for writing an
c       efficient lattice Boltzmann code !
c    *  Do not use this code for memory- or time-intensive
c       computations, it is not even a research code !
c    *  Don't even think about commercial application !
c
c    This code was written to show beginners in a simple and
c    short way the relevant procedures of a lattice Boltzmann solver,
c    pointing on how everything works "in principle". Nearly all
c    procedures could be implemented other (and better) as it is done
c    here, and even the algorithms used here could be changed to
c    save memory and increase performance. But the code works correct,
c    and we hope it will be good starting point for the first steps
c    in the lattice Boltzmann field. Good luck !
c
     Implicit none
c
c.....parameters
c
c.....grid size in x- and y-dimension
c
     integer  lx,ly
c
c.....ENTER APPROPRIATE VALUES HERE, TAKE CARE: SIZE DOES MATTER !!
c
     parameter(lx=2100,ly=21)
c
c.....variables
c
c.....fluid density per link
c
     real*8  density
c
c.....relaxation parameter
c
     real*8  tau
c
c.....accelleration
c
     real*8  accel
c
c.....maximum number of iterations
c
     integer  t_max
c
c.....linear dimension for Reynolds number
```

```
c
      real*8 r_rey
c
c.....iteration counter
c
      integer time
c
c.....error flag
c
      logical  error
c
c.....obstacle array
c
      logical  obst(lx,ly)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCC
            real*8 ux(lx,ly), vy(lx,ly), pressure(lx,ly)
c
c.....fluid densities
c     a 9-speed lattice is used here, other geometries are possible
c
c     the densities are numbered as follows:
c
c             6   2   5
c              \ | /
c             3 - 0 - 1
c              / | \
c             7   4   8
c
c    the lattice nodes are numbered as follows:
c
c     ^
c     |
c     y
c
c        :   :   :
c
c   3    *   *   *  ..
c
c   2    *   *   *  ..
c
c   1    *   *   *  ..
c                    x ->
c        1   2   3
c
      real*8  node(0:8,lx,ly)
c
c.....help array for temporarely storage of fluid densities
c
      real*8  n_hlp(0:8,lx,ly)
c
c.....average velocity, computed by subroutine 'write_velocity'
c
      real*8  vel
c
c.....startup information message
```

```
c
      write (6,*)
      write (6,*) 'anb 1.0 (2001-03-29)'
      write (6,*) 'Copyright (C) 1998-2001 Joerg Bernsdorf /'
      write (6,*) 'C&C Research Laboratories, NEC Europe Ltd.'
      write (6,*) 'anb comes with ABSOLUTELY NO WARRANTY;'
      write (6,*) 'This is free software, and you are welcome to redistr
     &ibute it'
      write (6,*) 'under certain conditions; see the file COPYING for de
     &tails.'
      write (6,*) 'All rights reserved.'
c
      write (6,*)
      write (6,*) '*************************************************'
      write (6,*) '***              anb starting ...                 ***'
      write (6,*) '*************************************************'
      write (6,*) '*** Precompiled for lattice size lx = ',lx
      write (6,*) '***                            ly = ',ly
      write (6,*) '*************************************************'
      write (6,*) '***'
c
c=======================================================================
c     begin initialisation
c=======================================================================
c
c.....initialize error flag
c
      error = .false.
c
c.....read parameter file
c.....in this file you can enter all relevant parameters,
c     only lattice size must be fixed befor compilation !
c
      call read_parametrs(error,t_max,density,accel,tau,r_rey)
c
c.....if an I/O error occurs while reading the parameter file,
c     the "error"-flag is set "true" and the program stops.
c
      if (error) goto 990
c
c.....read obstacle file
c.....in this file you can enter the x-,and y-coordinates of
c     any obstacles,wall boundaries are also defined here by
c     adding single obstacles.
c
      call read_obstacles(error,obst,lx,ly)
c
c.....if an I/O error occurs reading while the obstacle file,
c     the "error"-flag is set "true" and the program stops.
c
      if (error) goto 990
c
      call init_density(lx,ly,density,node)
c
c.....mean sqaure flow is monitored and stored in the file'anb_qxm.out',
c     one value for each iteration. The file is opened here.
```

```
c
      open(10,file='anb_qxm.out')
c
c=======================================================================
c     end initialisation
c=======================================================================
c=======================================================================
c     begin iterations
c=======================================================================
c
c.....main loop
c
      do 100 time = 1, t_max
c
c.......the integral fluid density is checke each t_max/10 iteration.
c       this is a good indicator, if the program is going to crash ...
c       The integral fluid density should be constant all time ...
c
        if (time .ge. 10 .and. mod(time,t_max/10) .eq. 0) then
          call check_density(lx,ly,node,time)
c
        end if
c
c.......directed flow is induced by density redistribution in the first
c       lattice column. This is not too clever, since the resulting
c       reynolds number can not be controlled and reaching stady state
c       takes quite some time, but it is simple and it works ...
c
        call redistribute(lx,ly,obst,node,accel,density)
c
c.......density propagation: all fluid densities are propagated from
c       non-occupied nodes along the lattice connection lines
c       to their next neighbours, periodic boundary conditiones are
c       applied in each direction.
c
        call propagate(lx,ly,node,n_hlp)
c
c.......bounc back from obstacles: this is the no-slip boundary-
c       condition.
c       The velocity vector of all fluid densities is inverted, so all
c       the fluid densities will be sent back to the node  where they
c       were located before the last propagation step, but with opposite
c       velocity vector
c       ... there exist lots of other possibilities.
c
        call bounceback(lx,ly,obst,node,n_hlp)
c
c.......density relaxation: a single time relaxation with relaxation
c       parameter omega is applied here. This step is only "local",
c       nothing is propagated through the lattice.
c
        call relaxation(density,tau,lx,ly,node,n_hlp,obst)
c
c.......average flow velocity is computed at a cross section in the
c       middle of the channel and written to the file "anb_qx.out"
c       ever iteration. this is a good controll for the convergence of
```

```
c       the program.
c
        call write_velocity(lx,ly,time,obst,node,vel)
c
c.....end of the main loop
c
  100 continue
c
c.....compute fluid-velocities u,v and pressure from velocity
c     distribution, and write to file anb_rs.out.
c
      call write_results(lx,ly,obst,node,density,ux,vy,pressure)
c
c.....compute reynolds number
c
      call comp_rey(lx,ly,obst,node,time,tau,density,r_rey)
c
      goto 999
c
c.....here we get only, if the "error" flag was set "true",  something
c     went wrong reading the files .The program stops with an error
c     message.
c
  990 write (6,*) '!!! error: program stopped during iteration =', time
      write (6,*) '!!!'
c
  999 continue
c
c.....the file for mean square flow output (anb_qx.out) is closed.
c     Look at this file before starting any other evaluating, it tells
c     you, if you reached stady state and if you got reasonable results!
c
      close(10)
c
      write (6,*) '********************    end     ********************'
c
c
      stop
      end
c
c
      subroutine read_parametrs(error,t_max,density,accel,tau,r_rey)
c
**************************************************************************
*                                                                        *
*     Input run-time parameters from file 'anb.par'                      *
*                                                                        *
*           Joerg BERNSDORF                                              *
*           C&C Research Laboratories, NEC Europe Ltd.                   *
*           Rathausalle 10                                               *
*           D-53757 Sankt Augustin, Germany                              *
*                                                                        *
*     Last change: 1999/09/28                                            *
*                                                                        *
**************************************************************************
c
```

```
      implicit none
c
      real*8  density,accel,tau,r_rey
c
      integer  t_max
c
      logical  error
c.....open parameter file
c
      open(unit=10,file='anbpar.txt',STATUS='UNKNOWN' )
c
c.......line 1: number of iterations
c
        read(10,*,err=900) t_max
c
c.......line 2: fluid density per link
c
        read(10,*,err=900) density
c
c.......line 3: density redistribution
c
        read(10,*,err=900) accel
c
c.......line 4: relaxation parameter
c
        read(10,*,err=900) tau
c
c.......line 5: linear dimension (for reynolds number)
c
        read(10,*,err=900) r_rey
c
c.....close parameter file
c
      close(10)
c
c.....information message
c
      write (6,*) '*** Paramters read from file anb.par.'
      write (6,*) '***'
c
      goto 999
c
c.....error message: file read error
c
  900 write (6,*) '!!! Error reading file anb.par'
      write (6,*) '!!!'
c
      goto 990
c
  990 error = .true.
c
  999 continue
c
      return
      end
c
```

```
      subroutine read_obstacles(error,obst,lx,ly)
************************************************************************
*                                                                      *
*     Input obstacle file 'anb.obs'                                    *
*                                                                      *
*           Joerg BERNSDORF                                            *
*           C&C Research Laboratories, NEC Europe Ltd.                 *
*           Rathausalle 10                                             *
*           D-53757 Sankt Augustin, Germany                            *
*                                                                      *
*     Last change: 1998/08/25                                          *
*                                                                      *
************************************************************************
c
      implicit none
c
      integer  lx,ly
c
      logical  error,obst(lx,ly)
c
c.....local variables
c
      integer  x,y
c
c.....no obstacles in obstacle array
c
        do 10 y = 1, ly
          do 10 x = 1, lx
c
   10       obst(x,y) = .false.
c
c.....open obstacle file
c.....the obstacle file is defined by the x- and y- coordinates of the
c     obstacles. Each obstacle has a line of its own. Also boundaries
c     have to be defined here.
c
      open(10,file='channel21x2100.txt ')
c
c.....read obstacle coordinates
c
   20 continue
c
        read(10,*,end=50,err=900) x,y
c
c.......check if obstacle inside domain boundaries
c
        if (x .le. lx .and. y .le. ly) then
c
c.......define obstacle
c
          obst(x,y) = .true.
c
        else
c
          write(6,*) '!!! Obstacle out of range, skipped'
          write(6,*) '!!! lx = ', x, ' , ly = ', y
```

```
        write(6,*) '!!!'
c
       end if
c
     goto 20
c
   50 continue
c
c.....close obstacle file
c
     close(10)
c
     write (6,*) '*** Geometry information read from file anb.obs.'
     write (6,*) '***'
c
     goto 999
c
c.....error message: file read error
c
  900 write (6,*) '!!! Error reading file anb.obs'
     write (6,*) ' '
c
     goto 990
c
  990 error = .true.
c
  999 continue
c
c
     return
     end
c
     subroutine init_density(lx,ly,density,node)
c
**********************************************************************
*                                                                    *
*    Initialize density distribution function n with equilibrium     *
*    for zero velocity                                               *
*                                                                    *
*          Joerg BERNSDORF                                           *
*          C&C Research Laboratories, NEC Europe Ltd.                *
*          Rathausalle 10                                            *
*          D-53757 Sankt Augustin, Germany                           *
*                                                                    *
*    Last change: 1999/09/28                                         *
*                                                                    *
**********************************************************************
c
     implicit none
c
     integer  lx,ly
c
     real*8  density,node(0:8,lx,ly)
c
c.....local variables
c
```

```
      integer  x,y
      real*8  t_0,t_1,t_2
c
c.....compute weighting factors (depending on lattice geometry)
c
      t_0 = density *  4.d0 / 9.d0
      t_1 = density /  9.d0
      t_2 = density / 36.d0
c
c.....loop over computational domain
c
      do 10 x = 1, lx
        do 10 y = 1, ly
c
c.........zero velocity density
c
          node(0,x,y) = t_0
c
c.........equilibrium densities for axis speeds
c
          node(1,x,y) = t_1
          node(2,x,y) = t_1
          node(3,x,y) = t_1
          node(4,x,y) = t_1
c
c.........equilibrium densities for diagonal speeds
c
          node(5,x,y) = t_2
          node(6,x,y) = t_2
          node(7,x,y) = t_2
          node(8,x,y) = t_2
c
   10 continue
c
      return
      end
c
      subroutine check_density(lx,ly,node,time)
**************************************************************************
*                                                                        *
*    compute integral density                                            *
*                                                                        *
*           Joerg BERNSDORF                                              *
*           C&C Research Laboratories, NEC Europe Ltd.                   *
*           Rathausalle 10                                               *
*           D-53757 Sankt Augustin, Germany                              *
*                                                                        *
*    Last change: 2000/01/14                                             *
*                                                                        *
**************************************************************************
c
      implicit none
c
      integer  lx,ly,time
c
      real*8  node(0:8,lx,ly)
```

```
c
c
c.....local variables
c
      integer  x,y,n
c
      real*8 n_sum
c
      n_sum = 0.d0
c
c.....loop over computational domain
c
       do 10 y = 1, ly
         do 10 x = 1, lx
c
c..........loop over all densities
c
           do 10 n = 0, 8
c
c.............sum up densities
c
   10         n_sum = n_sum + node(n,x,y)
c
      write(6,*) '*** Iteration number = ', time
      write(6,*) '*** Integral density = ', n_sum
      write(6,*) '***'
c
      return
      end
c
      subroutine redistribute(lx,ly,obst,node,accel,density)
c
************************************************************************
*                                                                      *
*     density redistribution in first lattice column                   *
*                                                                      *
*           Joerg BERNSDORF                                            *
*           C&C Research Laboratories, NEC Europe Ltd.                 *
*           Rathausalle 10                                             *
*           D-53757 Sankt Augustin, Germany                            *
*                                                                      *
*     Last change: 1999/09/28                                          *
*                                                                      *
************************************************************************
c
      implicit none
c
      integer  lx,ly
c
      logical  obst(lx,ly)
c
      real*8   node(0:8,lx,ly),accel,density
c
c.....local variables
c
      integer  y
```

```
      real*8  t_1,t_2
c
c.....compute weighting factors (depending on lattice geometry) for
c     increasing/decreasing inlet densities
c
      t_1 = density * accel /  9.d0
      t_2 = density * accel / 36.d0

      do 10 y = 1, ly
c
c.......accelerate flow only on non-occupied nodes
c
        if (.not. obst(1,y) .and.
c
c.........check to avoid negative densities
c
     &    node(3,1,y) - t_1 .gt. 0. .and.
     &    node(6,1,y) - t_2 .gt. 0. .and.
     &    node(7,1,y) - t_2 .gt. 0.) then
c
c.........increase east
c
          node(1,1,y) = node(1,1,y) + t_1
c
c.........decrease west
c
           node(3,1,y) = node(3,1,y) - t_1
c
c.........increase north-east
c
          node(5,1,y) = node(5,1,y) + t_2
c
c.........decrease north-west
c
          node(6,1,y) = node(6,1,y) - t_2
c
c.........decrease south-west
c
          node(7,1,y) = node(7,1,y) - t_2
c
c.........increase south-east
c
          node(8,1,y) = node(8,1,y) + t_2
c
        end if
c
   10 continue
c
      return
      end

            subroutine propagate(lx,ly,node,n_hlp)
c
**********************************************************************
*                                                                    *
*     Propagate fluid densities to their next neighbour nodes        *
```

```
*                                                                     *
*          Joerg BERNSDORF                                            *
*          C&C Research Laboratories, NEC Europe Ltd.                 *
*          Rathausalle 10                                             *
*          D-53757 Sankt Augustin, Germany                            *
*                                                                     *
*    Last change: 1999/09/28                                          *
*                                                                     *
***********************************************************************
c
     implicit none
c
     integer  lx,ly
c
     real*8   node(0:8,lx,ly),n_hlp(0:8,lx,ly)
c
c.....local variables
c
     integer  x,y,x_e,x_w,y_n,y_s
c
c.....loop over all nodes
c
     do 10 x = 1, lx
       do 10 y = 1, ly
c
c.........compute upper and right next neighbor nodes with regard
c         to periodic boundaries
c
          y_n = mod(y,ly) + 1
          x_e = mod(x,lx) + 1
c
c.........compute lower and left next neighbor nodes with regard to
c         periodic boundaries

          y_s = ly - mod(ly + 1 - y, ly)
          x_w = lx - mod(lx + 1 - x, lx)
c
c.........density propagation
c
c.........zero: just copy
c
          n_hlp(0,x  ,y  ) = node(0,x,y)
c
c.........east
c
          n_hlp(1,x_e,y  ) = node(1,x,y)
c
c.........north
c
          n_hlp(2,x  ,y_n) = node(2,x,y)
c
c.........west
c
          n_hlp(3,x_w,y  ) = node(3,x,y)
c
c.........south
```

```
c
        n_hlp(4,x  ,y_s) = node(4,x,y)
c
c.........north-east
c
        n_hlp(5,x_e,y_n) = node(5,x,y)
c
c.........north-west
c
        n_hlp(6,x_w,y_n) = node(6,x,y)
c
c.........south-west
c
        n_hlp(7,x_w,y_s) = node(7,x,y)
c
c.........south-east
c
        n_hlp(8,x_e,y_s) = node(8,x,y)
c
   10 continue
c
      return
      end
c
      subroutine bounceback(lx,ly,obst,node,n_hlp)
c
************************************************************************
*                                                                      *
*     Fluid densities are rotated. By the next propagation step, this  *
*     results in a bounce back from obstacle nodes.                    *
*                                                                      *
*           Joerg BERNSDORF                                            *
*           C&C Research Laboratories, NEC Europe Ltd.                 *
*           Rathausalle 10                                             *
*           D-53757 Sankt Augustin, Germany                            *
*                                                                      *
*     Last change: 1999/09/28                                          *
*                                                                      *
************************************************************************
c
      implicit none
c
      integer  lx,ly
c
      logical  obst(lx,ly)
c
      real*8   node(0:8,lx,ly),n_hlp(0:8,lx,ly)
c
c.....local variables

C**********************************************************************
C   This part includes the reflection factor (r_f) and                 *
C         Accommodation coefficient (a) for the boundary conditions    *
C                                                                      *
C   Paper by Tang et al, ICMM2003-1046 : reflection factor             *
C   Paper by Zhang et al, Physical Review E(2005) : accommodation coefficient *
```

```
C                                                                       *
C   This part has been modified by                                      *
C                                                                       *
C                                          Taiho Yeom                   *
C                                          Oklahoma State university    *
C                                          May 2007                     *
C***********************************************************************
      integer  x,y

            real*8 r_f
            real*8 a

            r_f = 0.85
c     a = 0.99
c
c.....loop over all nodes
c
      do 10 x = 1, lx
        do 10 y = 1, ly
c
c.........consider only obstacle nodes
c
          if (obst(x,y)) then
c
c...........rotate all ensities and write back to node
c
c...........east
c
            node(1,x,y) = n_hlp(3,x,y)
c
c...........north
c
            node(2,x,y) = n_hlp(4,x,y)

c           node(2,x,y) = n_hlp(4,x,y) + a*n_hlp(8,x,y)
c     &                 + a*n_hlp(7,x,y)

c...........west
c
            node(3,x,y) = n_hlp(1,x,y)
c
c...........south
c
            node(4,x,y) = n_hlp(2,x,y)

c                 node(4,x,y) = n_hlp(2,x,y) + a*n_hlp(5,x,y)
c     &                 + a*n_hlp(6,x,y)
c
c...........north-east
c
c                       node(5,x,y) = n_hlp(7,x,y)

            node(5,x,y) = r_f*n_hlp(7,x,y) + (1-r_f)*n_hlp(8,x,y)

c                       node(5,x,y) = (1-a)*n_hlp(8,x,y)
c
```

```
c.....north-west

c                       node(6,x,y) = n_hlp(8,x,y)
c
          node(6,x,y) = r_f*n_hlp(8,x,y) + (1-r_f)*n_hlp(7,x,y)

c                 node(6,x,y) = (1-a)*n_hlp(7,x,y)
c
c.....south-west
c
c                       node(7,x,y) = n_hlp(5,x,y)

          node(7,x,y) = r_f*n_hlp(5,x,y) + (1-r_f)*n_hlp(6,x,y)

c                 node(7,x,y) = (1-a)*n_hlp(6,x,y)
c
c.....south-east

c                       node(8,x,y) = n_hlp(6,x,y)
c
          node(8,x,y) = r_f*n_hlp(6,x,y) + (1-r_f)*n_hlp(5,x,y)

c                 node(8,x,y) = (1-a)*n_hlp(5,x,y)
c
        end if
c
   10 continue
c
c
      return
      end

c
      subroutine relaxation(density,tau,lx,ly,node,n_hlp,obst)
c
**************************************************************************
*                                                                        *
*     One-step density relaxation process                                *
*                                                                        *
*           Joerg BERNSDORF                                              *
*           C&C Research Laboratories, NEC Europe Ltd.                   *
*           Rathausalle 10                                               *
*           D-53757 Sankt Augustin, Germany                              *
*                                                                        *
*     Last change: 1999/09/28                                            *
*                                                                        *
**************************************************************************

      implicit none
c
      integer  lx,ly
c
      logical  obst(lx,ly)
c
      real*8   density,tau,node(0:8,lx,ly),n_hlp(0:8,lx,ly)
c
```

```
c.....local variables
c
      integer  x,y,i
c
      real*8  c_squ,t_0,t_1,t_2,u_x,u_y,u_n(8),n_equ(0:8),u_squ,d_loc


C**********************************************************************
C           CORRECTION FOR MICRO FLOWS WITH RELAXION TIME DEPENDING OF RHO
C           XIAOBO ET AL (2002)

            real*8 tauprime

C**********************************************************************
c
c.....weighting factors (depending on lattice geometry)
c
      t_0 = 4.d0 / 9.d0
      t_1 = 1.d0 /  9.d0
      t_2 = 1.d0 / 36.d0
c
c.....square speed of sound
c
      c_squ = 1.d0 / 3.d0
c
c.....loop over all nodes
c.....attention: actual densities are stored after the propagation
c                step in the help-array n_hlp !
c
      do 10 x = 1, lx
        do 10 y = 1, ly
c
c.........only free nodes are considered here
c
          if (.not. obst(x,y)) then
c
c...........integral local density
c
c...........initialize variable d_loc
c
            d_loc = 0.d0
c
            do 20 i = 0, 8
c
              d_loc = d_loc + n_hlp(i,x,y)
c
   20       continue
c

c**********************************************************************
                        tauprime = 0.5 + (tau - 0.5) / d_loc

c**********************************************************************
c...........x-, and y- velocity components
c
            u_x = (n_hlp(1,x,y) + n_hlp(5,x,y) + n_hlp(8,x,y)
```

```
     &          -(n_hlp(3,x,y) + n_hlp(6,x,y) + n_hlp(7,x,y))) / d_loc
c
          u_y = (n_hlp(2,x,y) + n_hlp(5,x,y) + n_hlp(6,x,y)
     &          -(n_hlp(4,x,y) + n_hlp(7,x,y) + n_hlp(8,x,y))) / d_loc
c
c.....square velocity
c
          u_squ = u_x * u_x + u_y * u_y
c
c.....n- velocity compnents (n = lattice node connection vectors)
c.....this is only necessary for clearence, and only 3 speeds would
c.....be necessary
c
          u_n(1) =   u_x
          u_n(2) =         u_y
          u_n(3) = - u_x
          u_n(4) =       - u_y
          u_n(5) =   u_x + u_y
          u_n(6) = - u_x + u_y
          u_n(7) = - u_x - u_y
          u_n(8) =   u_x - u_y
c
c.....equilibrium densities
c.....this can be rewritten to improve computational performance
c.....considerabely !
c
c.....zero velocity density
c
          n_equ(0) = t_0 * d_loc * (1.d0 - u_squ / (2.d0 * c_squ))
c
c.....axis speeds (factor: t_1)
c
          n_equ(1) = t_1 * d_loc * (1.d0 + u_n(1) / c_squ
     &               + u_n(1) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &               - u_squ / (2.d0 * c_squ))
c
          n_equ(2) = t_1 * d_loc * (1.d0 + u_n(2) / c_squ
     &               + u_n(2) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &               - u_squ / (2.d0 * c_squ))
c
          n_equ(3) = t_1 * d_loc * (1.d0 + u_n(3) / c_squ
     &               + u_n(3) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &               - u_squ / (2.d0 * c_squ))
c
          n_equ(4) = t_1 * d_loc * (1.d0 + u_n(4) / c_squ
     &               + u_n(4) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &               - u_squ / (2.d0 * c_squ))
c
c.....diagonal speeds (factor: t_2)
c
          n_equ(5) = t_2 * d_loc * (1.d0 + u_n(5) / c_squ
     &               + u_n(5) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &               - u_squ / (2.d0 * c_squ))
c
          n_equ(6) = t_2 * d_loc * (1.d0 + u_n(6) / c_squ
     &               + u_n(6) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
```

```
     &              - u_squ / (2.d0 * c_squ))
c
           n_equ(7) = t_2 * d_loc * (1.d0 + u_n(7) / c_squ
     &              + u_n(7) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &              - u_squ / (2.d0 * c_squ))
c
           n_equ(8) = t_2 * d_loc * (1.d0 + u_n(8) / c_squ
     &              + u_n(8) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &              - u_squ / (2.d0 * c_squ))
c
c.....relaxation step
c
           do 30 i = 0, 8
c
             node(i,x,y) = n_hlp(i,x,y)
     &                   + (n_equ(i) - n_hlp(i,x,y)) / tauprime
c
   30      continue
c
         end if
c
   10 continue
c
      return
      end
c
      subroutine write_velocity(lx,ly,time,obst,node,vel)
c
************************************************************************
*                                                                      *
*     compute average velocity                                         *
*                                                                      *
*           Joerg BERNSDORF                                            *
*           C&C Research Laboratories, NEC Europe Ltd.                 *
*           Rathausalle 10                                             *
*           D-53757 Sankt Augustin, Germany                            *
*                                                                      *
*     Last change: 1999/09/28                                          *
*                                                                      *
************************************************************************
c
      implicit none
c
      integer  lx,ly,time
c
      logical  obst(lx,ly)
c
      real*8   node(0:8,lx,ly),vel
c
c.....local variables
c
      integer  x,y,i,n_free
c
      real*8  u_x,d_loc
c
c.....loop over channel cross section at half channel length lx/2
```

```
c
      x = int(float(lx) / 2.d0)
c
c.....initialize counter
c
      n_free = 0
      u_x = 0.d0
c
      do 10 y = 1, ly
c
c.......only non-occupied nodes are considered here
c
        if(.not. obst(x,y)) then
c
c.........integral local density
c
c.........initialize variable d_loc
c
          d_loc = 0.d0
c
          do 20 i = 0, 8
c
            d_loc = d_loc + node(i,x,y)
c
   20     continue
c
c.........x-, and y- velocity components
c
          u_x = u_x + (node(1,x,y) + node(5,x,y) + node(8,x,y)
     &               -(node(3,x,y) + node(6,x,y) + node(7,x,y))) / d_loc
c
c.........increase counter
c
          n_free = n_free + 1
c
        end if
c
   10 continue
c
c.....average velocity
c
      vel = u_x / float(n_free)
c
c.....write to file
c
      write(10,*) time, vel
c
      return
      end
c
      subroutine  write_results(lx,ly,obst,node,density,ux,vy,pressure)
c
**********************************************************************
*                                                                    *
*     Output of rsults to file 'anb.dat'                             *
*                                                                    *
```

```
*           Joerg BERNSDORF                                                 *
*           C&C Research Laboratories, NEC Europe Ltd.                      *
*           Rathausalle 10                                                  *
*           D-53757 Sankt Augustin, Germany                                 *
*                                                                           *
*    Last change: 1999/09/28                                                *
*                                                                           *
*****************************************************************************
c
      implicit none
c
      integer  lx,ly, obstacle(lx,ly)
c
      real*8  node(0:8,lx,ly),density
c
      logical  obst(lx,ly)

c.....array for velocities, pressure and obstacle

            real*8 ux(lx,ly), vy(lx,ly), pressure(lx,ly)
c
c.....local variables
c
      integer  x,y,i,obsval
c
      real*8  u_x,u_y,d_loc,press,c_squ
c
c.....square speed of sound
c
      c_squ = 1.d0 / 3.d0
c
c.....open results output files
c
c      open(11,file='anb.dat')

            open(12,file='ux_array.dat')
            open(13,file='vy_array.dat')
            open(14,file='pres_array.dat')
c           open(15,file='obs_array.txt')
c
c.....write header for postprocessing with TECPLOT software
c.....uncomment following line, if this header should be printed
c
c      write(11,*) 'VARIABLES = X, Y, VX, VY, PRESS, OBST'
c     write(11,*) 'ZONE I=', lx, ', J=', ly, ', F=POINT'
c
c.....loop over all nodes
c.....attention: actual densities are stored after the propagation
c               step in the help-array n_hlp !
c
      do 10 y = 1, ly
        do 10 x = 1, lx
c
c.........if obstacle node, nothing is to do ...
c
          if (obst(x,y)) then
```

```
c
c.....obstacle indicator
c
c          obsval = 1
c
c.....velocity components = 0
c
c          u_x = 0.d0
c          u_y = 0.d0
C**********************************************************************
C      Compute the velocity components of the wall nodes                *
C      Modified by                                                      *
C                                                    Taiho Yeom         *
C                                                    Oklahoma State university  *
C                                                    May 2007           *
C**********************************************************************
                        d_loc = 0.d0
c
          do 30 i = 0, 8
c
            d_loc = d_loc + node(i,x,y)
c
  30        continue

c.....x-, and y- velocity components
c
          u_x = (node(1,x,y) + node(5,x,y) + node(8,x,y)
     &          -(node(3,x,y) + node(6,x,y) + node(7,x,y))) / d_loc
c
          u_y = (node(2,x,y) + node(5,x,y) + node(6,x,y)
     &          -(node(4,x,y) + node(7,x,y) + node(8,x,y))) / d_loc
c
c.....pressure = average pressure
c
          press = density * c_squ
c
        else
c**********************************************************************
c.....integral local density
c
c.....initialize variable d_loc
c
          d_loc = 0.d0
c
          do 20 i = 0, 8
c
            d_loc = d_loc + node(i,x,y)
c
  20        continue
c
c.....x-, and y- velocity components
c
          u_x = (node(1,x,y) + node(5,x,y) + node(8,x,y)
     &          -(node(3,x,y) + node(6,x,y) + node(7,x,y))) / d_loc
c
          u_y = (node(2,x,y) + node(5,x,y) + node(6,x,y)
```

```
    &          -(node(4,x,y) + node(7,x,y) + node(8,x,y))) / d_loc
c
c...........pressure
c
          press = d_loc * c_squ
c
c           obsval = 0
c
         end if

c.........small loop to compute velocities, pressure and obstacle in arrays
                                    ux(x,y) = u_x
                                    vy(x,y) = u_y
                                    pressure(x,y) = press
c                                   obstacle(x,y) = obsval
c
c.........write results to file
c
c         write(11,500) x, y, u_x, u_y, press, obsval

c500                format(I2,TR2,I2,TR2,E12.4,TR2,E12.4,TR2,E12.4,TR2,I2)

   10 continue
c
                    write(12,600) ((ux(x,y),y=1,ly),x=1,lx)
                    write(13,600) ((vy(x,y),y=1,ly),x=1,lx)
                    write(14,600) ((pressure(x,y),y=1,ly),x=1,lx)
c                   write(15,700) ((obstacle(x,y),y=1,ly),x=1,lx)

600                 format(21(E12.4,x))
c700                format(21(I2,X))

c.....close file 'anb.dat'
c
c      close(11)
            close(12)
            close(13)
            close(14)
c           close(15)
      return
      end
c
      subroutine comp_rey(lx,ly,obst,node,time,tau,density,r_rey)
c
**********************************************************************
*                                                                    *
*    compute Reynolds number                                         *
*                                                                    *
*           Joerg BERNSDORF                                          *
*           C&C Research Laboratories, NEC Europe Ltd.               *
*           Rathausalle 10                                           *
*           D-53757 Sankt Augustin, Germany                          *
*                                                                    *
*    Last change: 1999/09/28                                         *
*                                                                    *
**********************************************************************
```

```
c
      implicit none
c
      integer  lx,ly,time
c
      real*8  density,node(0:8,lx,ly),tau,r_rey
c
      logical  obst(lx,ly)
c
c.....local variables
c
      real*8  vel,visc,rey
c
c.....compute average velocity
c
      call write_velocity(lx,ly,time,obst,node,vel)
c
c.....compute viscosity
c
      visc = (1.0/3.0)*(2.0*tau-1.0)/(2.0*density)
c
c.....compute Reynolds number
c
c      rey = vel * ly / visc
c
c.....messages
c
       write (6,*) '*** Calculations finished, results:'
       write (6,*) '***'
       write (6,*) '*** viscosity = ', visc
       write (6,*) '*** average velocity = ', vel
       write (6,*) '*** Reynolds number = ', rey
       write (6,*) '***'
       write (6,*) '*** In the file anb.dat, you can find local'
       write (6,*) '*** information about the simulated flow.'
       write (6,*) '***'
       write (6,*) '*** In the file anb_qx.out, you can find the average
     &'
       write (6,*) '*** flow velocity plotted as a function of time.'
       write (6,*) '***'
c
      return
      end
```

APPENDIX B

LATTICE BOLTZMANN CODE FOR MICRO ORIFICE FLOWS

```
program micro orifice
************************************************************************
*    anb                                                               *
*                                          ... means: anb's not best   *
*                                                              - - -    *
*    A lattice Boltzmann CFD teaching code.                            *
*                                                                      *
*          Joerg BERNSDORF                                             *
*          C&C Research Laboratories, NEC Europe Ltd.                  *
*          Rathausalle 10                                              *
*          D-53757 Sankt Augustin, Germany                             *
*                                                                      *
*          e-mail: bernsdorf@ccrl-nece.de                              *
*          WWW: http://www.ccrl-nece.de/bernsdorf/                     *
*                                                                      *
*    Corrections and hints from Thomas Zeiser and Wolfgang Hamm        *
*    are gratefully acknowledged.                                      *
*                                                                      *
*    Copyright (C) 1998-2001  Joerg BERNSDORF /                        *
*               C&C Research Laboratories, NEC Europe Ltd.             *
*                                                                      *
*    This program is free software; you can redistribute it and/or     *
*    modify it under the terms of the GNU General Public License as    *
*    published by the Free Software Foundation; either version 2 of    *
*    the License, or (at your option) any later version.               *
*                                                                      *
*    This program is distributed in the hope that it will be useful,   *
*    but WITHOUT ANY WARRANTY; without even the implied warranty of    *
*    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.              *
*    See the GNU General Public License for more details.              *
*                                                                      *
*    You should have received a copy of the GNU General Public         *
*    License along with this program; if not, write to the Free        *
*    Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139,     *
*    USA.                                                              *
*                                                                      *
*    Last change: 2001/03/29                                           *
************************************************************************
*          The anb code has been modified partly to perform the simulations for *
*          Micro orifice gas flows by                                  *
*                                                                      *
*                                                 Taiho Yeom           *
*                                                 Oklahoma State University *
*                                                 May 2007             *
************************************************************************
c
c.....short introduction
c
c.....anb simulates incompressible viscous flow as governed by the
c     Navier-Stokes equations.
c     But anb is not a Navier-Stokes solver.
c     anb is a lattice Boltzmann solver, which means, the velocity
c     discrete boltzmann equation is solved here, using a single time
c     relaxation (BGK) collision operator.
c
c     For more details on this method read:
```

```
c
c                   Y.H.Quian, D.D'Humieres and P.Lallemand,
c                   Lattice BGK Models for Navier-Stokes Equation
c                   Europhys. Lett., 17 (6), pp. 479-484 (1992)
c
c    Other methods belonging to the lattice gas / lattice Boltzmann
c    familiy exist, but this one is very simple, quite good and used
c    by lots of scientists working on this topic.
c
c    This is a very simple implementation of the lattice BGK scheme,
c    not a very efficient and not a very memory safing one.
c    *  Do not misunderstand this as a good proposal for writing an
c       efficient lattice Boltzmann code !
c    *  Do not use this code for memory- or time-intensive
c       computations, it is not even a research code !
c    *  Don't even think about commercial application !
c
c    This code was written to show beginners in a simple and
c    short way the relevant procedures of a lattice Boltzmann solver,
c    pointing on how everything works "in principle". Nearly all
c    procedures could be implemented other (and better) as it is done
c    here, and even the algorithms used here could be changed to
c    save memory and increase performance. But the code works correct,
c    and we hope it will be good starting point for the first steps
c    in the lattice Boltzmann field. Good luck !
c
     implicit none
c
c.....parameters
c
c.....grid size in x- and y-dimension
c
     integer  lx,ly
c
c.....ENTER APPROPRIATE VALUES HERE, TAKE CARE: SIZE DOES MATTER !!
c
     parameter(lx=340,ly=20)
c
c.....variables
c
c.....fluid density per link
c
     real*8  density
c
c.....relaxation parameter
c
     real*8  tau
c
c.....accelleration
c
     real*8  accel
c
c.....maximum number of iterations
c
     integer  t_max
c
```

```
c.....linear dimension for Reynolds number
c
c      real*8 r_rey
c
c.....iteration counter
c
      integer time
c
c.....error flag
c
      logical  error
c
c.....obstacle array
c
      logical  obst(lx,ly)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCC
            real*8 ux(lx,ly), vy(lx,ly), pressure(lx,ly)
            real*8 rey(lx,ly),loc_den(lx,ly)
c
c.....fluid densities
c     a 9-speed lattice is used here, other geometries are possible
c
c     the densities are numbered as follows:
c
c             6   2   5
c              \ | /
c             3 - 0 - 1
c              / | \
c             7   4   8
c
c     the lattice nodes are numbered as follows:
c
c     ^
c     |
c     y
c
c        :   :   :
c
c   3    *   *   *  ..
c
c   2    *   *   *  ..
c
c   1    *   *   *  ..
c                    x ->
c        1   2   3
c
      real*8  node(0:8,lx,ly)
c
c.....help array for temporarely storage of fluid densities
c
      real*8  n_hlp(0:8,lx,ly)
c
c.....average velocity, computed by subroutine 'write_velocity'
c
      real*8  vel
```

```
c
c.....startup information message
c
      write (6,*)
      write (6,*) 'anb 1.0 (2001-03-29)'
      write (6,*) 'Copyright (C) 1998-2001 Joerg Bernsdorf /'
      write (6,*) 'C&C Research Laboratories, NEC Europe Ltd.'
      write (6,*) 'anb comes with ABSOLUTELY NO WARRANTY;'
      write (6,*) 'This is free software, and you are welcome to redistr
     &ibute it'
      write (6,*) 'under certain conditions; see the file COPYING for de
     &tails.'
      write (6,*) 'All rights reserved.'
c
      write (6,*)
      write (6,*) '*****************************************************'
      write (6,*) '***              anb starting ...                 ***'
      write (6,*) '*****************************************************'
      write (6,*) '*** Precompiled for lattice size lx = ',lx
      write (6,*) '***                            ly = ',ly
      write (6,*) '*****************************************************'
      write (6,*) '***'
c
c=======================================================================
c     begin initialisation
c=======================================================================
c
c.....initialize error flag
c
      error = .false.
c
c.....read parameter file
c.....in this file you can enter all relevant parameters,
c     only lattice size must be fixed befor compilation !
c
      call read_parametrs(error,t_max,density,accel,tau)
c
c.....if an I/O error occurs while reading the parameter file,
c     the "error"-flag is set "true" and the program stops.
c
      if (error) goto 990
c
c.....read obstacle file
c.....in this file you can enter the x-,and y-coordinates of
c     any obstacles,wall boundaries are also defined here by
c     adding single obstacles.
c
      call read_obstacles(error,obst,lx,ly)
c
c.....if an I/O error occurs reading while the obstacle file,
c     the "error"-flag is set "true" and the program stops.
c
      if (error) goto 990
c
      call init_density(lx,ly,density,node)
c
```

```
c.....mean sqaure flow is monitored and stored in the file'anb_qxm.out',
c     one value for each iteration. The file is opened here.
c
      open(10,file='anb_qxm.out')
c
c=======================================================================
c     end initialisation
c=======================================================================
c=======================================================================
c     begin iterations
c=======================================================================
c
c.....main loop
c
      do 100 time = 1, t_max
c
c.......the integral fluid density is checke each t_max/10 iteration.
c       this is a good indicator, if the program is going to crash ...
c       The integral fluid density should be constant all time ...
c
        if (time .ge. 10 .and. mod(time,t_max/10) .eq. 0) then
c
          call check_density(lx,ly,node,time)
c
        end if
c
c.......directed flow is induced by density redistribution in the first
c       lattice column. This is not too clever, since the resulting
c       reynolds number can not be controlled and reaching stady state
c       takes quite some time, but it is simple and it works ...
c
        call redistribute(time,lx,ly,obst,node,accel,density)

c.......density propagation: all fluid densities are propagated from
c       non-occupied nodes along the lattice connection lines
c       to their next neighbours, periodic boundary conditiones are
c       applied in each direction.
c
        call propagate(lx,ly,node,n_hlp)
c
c.......bounc back from obstacles: this is the no-slip boundary-
c       condition.
c       The velocity vector of all fluid densities is inverted, so all
c       the fluid densities will be sent back to the node  where they
c       were located before the last propagation step, but with opposite
c       velocity vector
c       ... there exist lots of other possibilities.
c
        call bounceback(lx,ly,obst,node,n_hlp)
c
c.......density relaxation: a single time relaxation with relaxation
c       parameter omega is applied here. This step is only "local",
c       nothing is propagated through the lattice.
c
        call relaxation(density,tau,lx,ly,node,n_hlp,obst)
c
```

```
c.......average flow velocity is computed at a cross section in the
c       middle of the channel and written to the file "anb_qx.out"
c       ever iteration. this is a good controll for the convergence of
c       the program.
c
        call write_velocity(lx,ly,time,obst,node,vel)
c
c.....end of the main loop
c
  100 continue
c
c.....compute fluid-velocities u,v and pressure from velocity
c     distribution, and write to file anb_rs.out.
c
      call write_results(lx,ly,obst,node,density,ux,vy,pressure,loc_den)
c
c.....compute reynolds number
c
      call comp_rey(lx,ly,tau,density,ux,rey,node)

c
      goto 999
c
c.....here we get only, if the "error" flag was set "true",  something
c     went wrong reading the files .The program stops with an error
c     message.
c
  990 write (6,*) '!!! error: program stopped during iteration =', time
      write (6,*) '!!!'
c
  999 continue
c
c.....the file for mean square flow output (anb_qx.out) is closed.
c     Look at this file before starting any other evaluating, it tells
c     you, if you reached stady state and if you got reasonable results!
c
      close(10)
c
      write (6,*) '********************    end     ********************'
c
      stop
      end
c
      subroutine read_parametrs(error,t_max,density,accel,tau)
c
**********************************************************************
*                                                                    *
*     Input run-time parameters from file 'anb.par'                  *
*                                                                    *
*           Joerg BERNSDORF                                          *
*           C&C Research Laboratories, NEC Europe Ltd.               *
*           Rathausalle 10                                           *
*           D-53757 Sankt Augustin, Germany                          *
*                                                                    *
*     Last change: 1999/09/28                                        *
*                                                                    *
```

```
**************************************************************************
c
      implicit none
c
      real*8  density,accel,tau
c
      integer  t_max
c
      logical  error
c
c.....open parameter file
c
      open(unit=10,file='anbpar_2um.txt',STATUS='UNKNOWN' )
c
c.......line 1: number of iterations
c
        read(10,*,err=900) t_max
c
c.......line 2: fluid density per link
c
        read(10,*,err=900) density
c
c.......line 3: density redistribution
c
        read(10,*,err=900) accel
c
c.......line 4: relaxation parameter
c
        read(10,*,err=900) tau
c
c.......line 5: linear dimension (for reynolds number)
c
c        read(10,*,err=900) r_rey
c
c.....close parameter file
c
      close(10)
c
c.....information message
c
      write (6,*) '*** Paramters read from file anb.par.'
      write (6,*) '***'
c
      goto 999
c
c.....error message: file read error
c
  900 write (6,*) '!!! Error reading file anb.par'
      write (6,*) '!!!'
c
      goto 990
c
  990 error = .true.
c
  999 continue
c
```

```
c
      return
      end
c
c
      subroutine read_obstacles(error,obst,lx,ly)
c
*****************************************************************************
*                                                                           *
*     Input obstacle file 'anb.obs'                                         *
*                                                                           *
*           Joerg BERNSDORF                                                 *
*           C&C Research Laboratories, NEC Europe Ltd.                      *
*           Rathausalle 10                                                  *
*           D-53757 Sankt Augustin, Germany                                  *
*                                                                           *
*     Last change: 1998/08/25                                               *
*                                                                           *
*****************************************************************************
c
      implicit none
c
      integer  lx,ly
c
      logical  error,obst(lx,ly)
c
c.....local variables
c
      integer  x,y
c
c.....no obstacles in obstacle array
c
       do 10 y = 1, ly
         do 10 x = 1, lx
c
   10       obst(x,y) = .false.
c
c.....open obstacle file
c.....the obstacle file is defined by the x- and y- coordinates of the
c     obstacles. Each obstacle has a line of its own. Also boundaries
c     have to be defined here.
c
      open(10,file='orifice20x340_2um.txt ')
c
c.....read obstacle coordinates
c
   20 continue
c
       read(10,*,end=50,err=900) x,y
c
c.......check if obstacle inside domain boundaries
       if (x .le. lx .and. y .le. ly) then
c
c.......define obstacle
c
         obst(x,y) = .true.
```

```
c
       else
c
         write(6,*) '!!! Obstacle out of range, skipped'
         write(6,*) '!!! lx = ', x, ' , ly = ', y
         write(6,*) '!!!'
c
       end if
c
     goto 20
c
   50 continue
c
c.....close obstacle file
c
     close(10)
c
     write (6,*) '*** Geometry information read from file anb.obs.'
     write (6,*) '***'
c
     goto 999
c
c.....error message: file read error
c
  900 write (6,*) '!!! Error reading file anb.obs'
     write (6,*) ' '
c
     goto 990
c
  990 error = .true.
c
  999 continue
c
     return
     end
c
     subroutine init_density(lx,ly,density,node)
c
**********************************************************************
*                                                                    *
*    Initialize density distribution function n with equilibrium     *
*    for zero velocity                                               *
*                                                                    *
*          Joerg BERNSDORF                                           *
*          C&C Research Laboratories, NEC Europe Ltd.                *
*          Rathausalle 10                                            *
*          D-53757 Sankt Augustin, Germany                           *
*                                                                    *
*    Last change: 1999/09/28                                         *
*                                                                    *
**********************************************************************
     implicit none
c
     integer  lx,ly
c
     real*8  density,node(0:8,lx,ly)
```

```
c
c.....local variables
c
      integer  x,y
      real*8  t_0,t_1,t_2
c
c.....compute weighting factors (depending on lattice geometry)
c
      t_0 = density *  4.d0 / 9.d0
      t_1 = density /  9.d0
      t_2 = density / 36.d0
c
c.....loop over computational domain
c
      do 10 x = 1, lx
        do 10 y = 1, ly
c
c.........zero velocity density
c
          node(0,x,y) = t_0
c
c.........equilibrium densities for axis speeds
c
          node(1,x,y) = t_1
          node(2,x,y) = t_1
          node(3,x,y) = t_1
          node(4,x,y) = t_1
c
c.........equilibrium densities for diagonal speeds
c
          node(5,x,y) = t_2
          node(6,x,y) = t_2
          node(7,x,y) = t_2
          node(8,x,y) = t_2
c
   10 continue
c
      return
      end
c
      subroutine check_density(lx,ly,node,time)
c
************************************************************************
*                                                                      *
*     compute integral density                                         *
*                                                                      *
*           Joerg BERNSDORF                                            *
*           C&C Research Laboratories, NEC Europe Ltd.                 *
*           Rathausalle 10                                             *
*           D-53757 Sankt Augustin, Germany                            *
*                                                                      *
*     Last change: 2000/01/14                                          *
*                                                                      *
************************************************************************
c
      implicit none
```

```
c
      integer  lx,ly,time
c
      real*8  node(0:8,lx,ly)
c
c.....local variables
c
      integer  x,y,n
c
      real*8 n_sum
c
      n_sum = 0.d0
c
c.....loop over computational domain
c
       do 10 y = 1, ly
         do 10 x = 1, lx
c
c..........loop over all densities
c
           do 10 n = 0, 8
c
c.............sum up densities
c
   10         n_sum = n_sum + node(n,x,y)
c
      write(6,*) '*** Iteration number = ', time
      write(6,*) '*** Integral density = ', n_sum
      write(6,*) '***'
c
      return
      end
c
c
      subroutine redistribute(time,lx,ly,obst,node,accel,density)
c
************************************************************************
*                                                                      *
*     density redistribution in first lattice column                   *
*                                                                      *
*           Joerg BERNSDORF                                            *
*           C&C Research Laboratories, NEC Europe Ltd.                 *
*           Rathausalle 10                                             *
*           D-53757 Sankt Augustin, Germany                            *
*                                                                      *
*     Last change: 1999/09/28                                          *
*                                                                      *
************************************************************************
c
      implicit none
c
      integer  lx,ly,time
c
      logical  obst(lx,ly)
c
      real*8   node(0:8,lx,ly),accel,density
```

```
            real*8 d_loc
c
c.....local variables
c
      integer  y
      real*8  t_1,t_2

c.....compute weighting factors (depending on lattice geometry) for
c     increasing/decreasing inlet densities


       do 10 y = 1, ly

      t_1 = density * accel /  9.d0

      t_2 = density * accel / 36.d0


c.......accelerate flow only on non-occupied nodes
c
       if (.not. obst(1,y) .and.
c
c.........check to avoid negative densities
c
     &    node(3,1,y) - t_1 .gt. 0. .and.
     &    node(6,1,y) - t_2 .gt. 0. .and.
     &    node(7,1,y) - t_2 .gt. 0.) then
c
c.........increase east
c
         node(1,1,y) = node(1,1,y) + t_1
c
c.........decrease west
c
          node(3,1,y) = node(3,1,y) - t_1
c
c.........increase north-east
c
         node(5,1,y) = node(5,1,y) + t_2
c
c.........decrease north-west
c
         node(6,1,y) = node(6,1,y) - t_2
c
c.........decrease south-west
c
         node(7,1,y) = node(7,1,y) - t_2
c
c.........increase south-east
c
         node(8,1,y) = node(8,1,y) + t_2
c
       end if
   10 continue
```

```
      return
      end

            subroutine propagate(lx,ly,node,n_hlp)
c
**************************************************************************
*                                                                        *
*    Propagate fluid densities to their next neighbour nodes             *
*                                                                        *
*          Joerg BERNSDORF                                               *
*          C&C Research Laboratories, NEC Europe Ltd.                    *
*          Rathausalle 10                                                *
*          D-53757 Sankt Augustin, Germany                               *
*                                                                        *
*    Last change: 1999/09/28                                             *
*                                                                        *
**************************************************************************
c
      implicit none
c
      integer  lx,ly
c
      real*8   node(0:8,lx,ly),n_hlp(0:8,lx,ly)
c
c.....local variables
c
      integer  x,y,x_e,x_w,y_n,y_s

c
c.....loop over all nodes
c
      do 10 x = 1, lx
       do 10 y = 1, ly

c.........compute upper and right next neighbour nodes with regard
c         to periodic boundaries
c
          y_n = mod(y,ly) + 1
          x_e = mod(x,lx) + 1
c
c.........compute lower and left next neighbour nodes with regard to
c         periodic boundaries

         y_s = ly - mod(ly + 1 - y, ly)
          x_w = lx - mod(lx + 1 - x, lx)
c
c.........density propagation
c
c.........zero: just copy
c
          n_hlp(0,x  ,y  ) = node(0,x,y)
c
c.........east
c
          n_hlp(1,x_e,y  ) = node(1,x,y)
c
```

```
c.........north
c
        n_hlp(2,x  ,y_n) = node(2,x,y)

c.........west
c
        n_hlp(3,x_w,y  ) = node(3,x,y)
c
c.........south
c
        n_hlp(4,x  ,y_s) = node(4,x,y)

c.........north-east
c
        n_hlp(5,x_e,y_n) = node(5,x,y)
c
c.........north-west
c
        n_hlp(6,x_w,y_n) = node(6,x,y)
c
c.........south-west
c
        n_hlp(7,x_w,y_s) = node(7,x,y)
c
c.........south-east

        n_hlp(8,x_e,y_s) = node(8,x,y)

   10 continue
      return
      end
c
c
      subroutine bounceback(lx,ly,obst,node,n_hlp)
c
***************************************************************************
*                                                                         *
*     Fluid densities are rotated. By the next propagation step, this     *
*     results in a bounce back from obstacle nodes.                       *
*                                                                         *
*           Joerg BERNSDORF                                               *
*           C&C Research Laboratories, NEC Europe Ltd.                    *
*           Rathausalle 10                                                *
*           D-53757 Sankt Augustin, Germany                               *
*                                                                         *
*     Last change: 1999/09/28                                             *
*                                                                         *
***************************************************************************
c
      implicit none
c
      integer  lx,ly
c
      logical  obst(lx,ly)
c
      real*8   node(0:8,lx,ly),n_hlp(0:8,lx,ly)
```

```
c
c.....local variables
C**********************************************************************
C          This part includes the reflection factor(r_f)                *
C           for the boundary conditions of the micro orifice wall.      *
C                                                                       *
C          Paper by Tang et al, ICMM2003-1046 : reflection factor       *
C                                                                       *
C          This part has been modified by                               *
C                                                                       *
C                                                  Taiho Yeom           *
C                                                  Oklahoma State university *
C                                                  May 2007             *
C**********************************************************************
c
      integer  x,y,i

            real*8 r_b

            r_b = 0.85
c
c.....loop over all nodes  (original)

c
      do 10 x = 85, 95
        do 10 y = 1, ly
c
c.........consider only obstacle nodes
c.......... bottom and upper vertical

          if (obst(x,y) .and.
     &                (y .EQ. 1 .or. y .EQ. 2 .or.
     &          y .EQ. 3 .or. y .EQ. 18 .or.
     &          y .EQ. 19 .or. y .EQ. 20)) then
c
c...........rotate all ensities and write back to node
c
c...........east
c
            node(1,x,y) = n_hlp(3,x,y)
c
c...........north
c
            node(2,x,y) = n_hlp(4,x,y)
c
c...........west
c
            node(3,x,y) = n_hlp(1,x,y)
c
c...........south
c
            node(4,x,y) = n_hlp(2,x,y)
c
c...........north-east
c
c                       node(5,x,y) = n_hlp(7,x,y)
```

```
          node(5,x,y) = r_b*n_hlp(7,x,y) + (1-r_b)*n_hlp(6,x,y)
c
c...........north-west

c                         node(6,x,y) = n_hlp(8,x,y)
c
         node(6,x,y) = r_b*n_hlp(8,x,y) + (1-r_b)*n_hlp(5,x,y)
c
c...........south-west
c
c                         node(7,x,y) = n_hlp(5,x,y)

          node(7,x,y) = r_b*n_hlp(5,x,y) + (1-r_b)*n_hlp(8,x,y)
c
c...........south-east

c                         node(8,x,y) = n_hlp(6,x,y)
c
          node(8,x,y) = r_b*n_hlp(6,x,y) + (1-r_b)*n_hlp(7,x,y)


c..... bottom and upper horizon

          else if (obst(x,y) .and.
     &                   (y .EQ. 4 .or. y .EQ. 17).and.
     &                   x .LE. 93 .and. x .GE. 87) then
c
c...........rotate all ensities and write back to node
c
c...........east
c
          node(1,x,y) = n_hlp(3,x,y)
c
c...........north
c
          node(2,x,y) = n_hlp(4,x,y)
c
c...........west
c
          node(3,x,y) = n_hlp(1,x,y)
c
c...........south
c
          node(4,x,y) = n_hlp(2,x,y)
c
c...........north-east
c
c                         node(5,x,y) = n_hlp(7,x,y)

          node(5,x,y) = r_b*n_hlp(7,x,y) + (1-r_b)*n_hlp(8,x,y)
c
c...........north-west

c                         node(6,x,y) = n_hlp(8,x,y)
c
```

```
          node(6,x,y) = r_b*n_hlp(8,x,y) + (1-r_b)*n_hlp(7,x,y)
c
c...........south-west
c
c                         node(7,x,y) = n_hlp(5,x,y)

          node(7,x,y) = r_b*n_hlp(5,x,y) + (1-r_b)*n_hlp(6,x,y)
c
c...........south-east

c                         node(8,x,y) = n_hlp(6,x,y)
c
          node(8,x,y) = r_b*n_hlp(6,x,y) + (1-r_b)*n_hlp(5,x,y)
c

c.............upper conner
c
          else if (obst(x,y) .and.
     &          (y .EQ. 17 .or. y .EQ. 4) .and.
     &          (x .EQ. 94 .or. x .EQ. 86)) then
c
c...........rotate all ensities and write back to node
c
c...........east
c
          node(1,x,y) = n_hlp(3,x,y)
c
c...........north
c
          node(2,x,y) = n_hlp(4,x,y)
c
c...........west
c
          node(3,x,y) = n_hlp(1,x,y)
c
c...........south
c
          node(4,x,y) = n_hlp(2,x,y)
c
c...........north-east
c
c                         node(5,x,y) = n_hlp(7,x,y)

          node(5,x,y) = r_b*n_hlp(7,x,y) + 0.5*(1-r_b)*n_hlp(8,x,y)
     &      + 0.5*(1-r_b)*n_hlp(6,x,y)
c
c...........north-west

c                         node(6,x,y) = n_hlp(8,x,y)
c
          node(6,x,y) = r_b*n_hlp(8,x,y) + 0.5*(1-r_b)*n_hlp(7,x,y)
     &      + 0.5*(1-r_b)*n_hlp(5,x,y)
c
c...........south-west
c
c                         node(7,x,y) = n_hlp(5,x,y)
```

```
          node(7,x,y) = r_b*n_hlp(5,x,y) + 0.5*(1-r_b)*n_hlp(6,x,y)
     &      + 0.5*(1-r_b)*n_hlp(8,x,y)
c
c.....south-east

c                         node(8,x,y) = n_hlp(6,x,y)
c
          node(8,x,y) = r_b*n_hlp(6,x,y) + 0.5*(1-r_b)*n_hlp(5,x,y)
     &      + 0.5*(1-r_b)*n_hlp(7,x,y)

        end if
c
   10 continue


      return
      end
c
      subroutine relaxation(density,tau,lx,ly,node,n_hlp,obst)
c
************************************************************************
*                                                                      *
*     One-step density relaxation process                              *
*                                                                      *
*           Joerg BERNSDORF                                            *
*           C&C Research Laboratories, NEC Europe Ltd.                 *
*           Rathausalle 10                                             *
*           D-53757 Sankt Augustin, Germany                            *
*                                                                      *
*     Last change: 1999/09/28                                          *
*                                                                      *
************************************************************************
c
      implicit none
c
      integer  lx,ly
c
      logical  obst(lx,ly)
c
      real*8   density,tau,node(0:8,lx,ly),n_hlp(0:8,lx,ly)
c
c.....local variables
c
      integer  x,y,i
c
      real*8  c_squ,t_0,t_1,t_2,u_x,u_y,u_n(8),n_equ(0:8),u_squ,d_loc



C**********************************************************************
C           CORRECTION FOR MICRO FLOWS WITH RELAXION TIME DEPENDING OF RHO
C           XIAOBO ET AL (2002)

            real*8 tauprime
```

```
C*******************************************************************
c
c.....weighting factors (depending on lattice geometry)
c
      t_0 = 4.d0 / 9.d0
      t_1 = 1.d0 /  9.d0
      t_2 = 1.d0 / 36.d0
c
c.....square speed of sound
c
      c_squ = 1.d0 / 3.d0
c
c.....loop over all nodes
c.....attention: actual densities are stored after the propagation
c               step in the help-array n_hlp !
c
      do 10 x = 1, lx
        do 10 y = 1, ly
c
c.........only free nodes are considered here
c
          if (.not. obst(x,y)) then
c
c...........integral local density
c
c...........initialize variable d_loc
c
            d_loc = 0.d0
c
            do 20 i = 0, 8
c
              d_loc = d_loc + n_hlp(i,x,y)
c
   20       continue
c
c*******************************************************************

                        tauprime = 0.5 + (tau - 0.5) / d_loc

c*******************************************************************


c...........x-, and y- velocity components
c
            u_x = (n_hlp(1,x,y) + n_hlp(5,x,y) + n_hlp(8,x,y)
     &           -(n_hlp(3,x,y) + n_hlp(6,x,y) + n_hlp(7,x,y))) / d_loc
c
            u_y = (n_hlp(2,x,y) + n_hlp(5,x,y) + n_hlp(6,x,y)
     &           -(n_hlp(4,x,y) + n_hlp(7,x,y) + n_hlp(8,x,y))) / d_loc
c
c...........square velocity
c
            u_squ = u_x * u_x + u_y * u_y
c
c...........n- velocity compnents (n = lattice node connection vectors)
c...........this is only necessary for clearence, and only 3 speeds would
```

```
c.....be necessary
c
      u_n(1) =   u_x
      u_n(2) =         u_y
      u_n(3) = - u_x
      u_n(4) =       - u_y
      u_n(5) =   u_x + u_y
      u_n(6) = - u_x + u_y
      u_n(7) = - u_x - u_y
      u_n(8) =   u_x - u_y
c
c.....equilibrium densities
c.....this can be rewritten to improve computational performance
c.....considerabely !
c
c.....zero velocity density
c
      n_equ(0) = t_0 * d_loc * (1.d0 - u_squ / (2.d0 * c_squ))
c
c.....axis speeds (factor: t_1)
c
      n_equ(1) = t_1 * d_loc * (1.d0 + u_n(1) / c_squ
     &           + u_n(1) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &           - u_squ / (2.d0 * c_squ))
c
      n_equ(2) = t_1 * d_loc * (1.d0 + u_n(2) / c_squ
     &           + u_n(2) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &           - u_squ / (2.d0 * c_squ))
c
      n_equ(3) = t_1 * d_loc * (1.d0 + u_n(3) / c_squ
     &           + u_n(3) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &           - u_squ / (2.d0 * c_squ))
c
      n_equ(4) = t_1 * d_loc * (1.d0 + u_n(4) / c_squ
     &           + u_n(4) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &           - u_squ / (2.d0 * c_squ))
c
c.....diagonal speeds (factor: t_2)
c
      n_equ(5) = t_2 * d_loc * (1.d0 + u_n(5) / c_squ
     &           + u_n(5) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &           - u_squ / (2.d0 * c_squ))
c
      n_equ(6) = t_2 * d_loc * (1.d0 + u_n(6) / c_squ
     &           + u_n(6) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &           - u_squ / (2.d0 * c_squ))
c
      n_equ(7) = t_2 * d_loc * (1.d0 + u_n(7) / c_squ
     &           + u_n(7) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &           - u_squ / (2.d0 * c_squ))
c
      n_equ(8) = t_2 * d_loc * (1.d0 + u_n(8) / c_squ
     &           + u_n(8) ** 2.d0 / (2.d0 * c_squ ** 2.d0)
     &           - u_squ / (2.d0 * c_squ))
c
c.....relaxation step
```

```
c
         do 30 i = 0, 8
c
           node(i,x,y) = n_hlp(i,x,y)
     &                 + (n_equ(i) - n_hlp(i,x,y)) / tauprime
c
   30    continue
c
       end if
c
   10 continue
c
c
     return
     end
c
c
     subroutine write_velocity(lx,ly,time,obst,node,vel)
c
************************************************************************
*                                                                      *
*    compute average velocity                                          *
*                                                                      *
*           Joerg BERNSDORF                                            *
*           C&C Research Laboratories, NEC Europe Ltd.                 *
*           Rathausalle 10                                             *
*           D-53757 Sankt Augustin, Germany                            *
*                                                                      *
*    Last change: 1999/09/28                                           *
*                                                                      *
************************************************************************
c
     implicit none
c
     integer  lx,ly,time
c
     logical  obst(lx,ly)
c
     real*8   node(0:8,lx,ly),vel
c
c.....local variables
c
     integer  x,y,i,n_free
c
     real*8  u_x,d_loc
c
c.....loop over channel cross section at half channel length lx/2
c
     x = int(float(lx) / 2.d0)
c
c.....initialize counter
c
     n_free = 0
     u_x = 0.d0
c
     do 10 y = 1, ly
```

```
c
c.......only non-occupied nodes are considered here
c
       if(.not. obst(x,y)) then
c
c.........integral local density
c
c.........initialize variable d_loc
c
         d_loc = 0.d0
c
         do 20 i = 0, 8
c
           d_loc = d_loc + node(i,x,y)
c
   20     continue
c
c.........x-, and y- velocity components
c
         u_x = u_x + (node(1,x,y) + node(5,x,y) + node(8,x,y)
     &              -(node(3,x,y) + node(6,x,y) + node(7,x,y))) / d_loc
c
c.........increase counter
c
         n_free = n_free + 1
c
       end if
c
   10 continue
c
c.....average velocity
c
      vel = u_x / float(n_free)
c
c.....write to file
c
      write(10,500) time, vel

  500      format(I,E12.4)

c
      return
      end
c
      subroutine  write_results(lx,ly,obst,node,density,ux,vy,pressure,
     &loc_den)
c
*************************************************************************
*                                                                       *
*    Output of rsults to file 'anb.dat'                                 *
*                                                                       *
*           Joerg BERNSDORF                                             *
*           C&C Research Laboratories, NEC Europe Ltd.                  *
*           Rathausalle 10                                              *
*           D-53757 Sankt Augustin, Germany                             *
*                                                                       *
```

```
*     Last change: 1999/09/28                                                  *
*                                                                              *
********************************************************************************
c
      implicit none
c
      integer  lx,ly, obstacle(lx,ly)
c
      real*8  node(0:8,lx,ly),density
c
      logical  obst(lx,ly)

c.....arrays for velocities, pressure and obstacle

            real*8 ux(lx,ly), vy(lx,ly), pressure(lx,ly),loc_den(lx,ly)
c
c.....local variables
c
      integer  x,y,i,obsval
c
      real*8  u_x,u_y,d_loc,press,c_squ
c
c.....square speed of sound
c
      c_squ = 1.d0 / 3.d0
c
c.....open results output files
c
      open(11,file='loc_den.dat')

            open(12,file='ux_array.dat')
            open(13,file='vy_array.dat')
            open(14,file='pres_array.dat')
            open(15,file='obs_array.dat')
c
c.....write header for postprocessing with TECPLOT software
c.....uncomment following line, if this header should be printed
c
c
c      write(11,*) 'VARIABLES = X, Y, VX, VY, PRESS, OBST'
c     write(11,*) 'ZONE I=', lx, ', J=', ly, ', F=POINT'
c
c.....loop over all nodes
c.....attention: actual densities are stored after the propagation
c               step in the help-array n_hlp !
c
      do 10 y = 1, ly
        do 10 x = 1, lx
c
c.........if obstacle node, nothing is to do ...
c
          if (obst(x,y)) then
c
c...........obstacle indicator
c
            obsval = 1
```

```
c
c.....velocity components = 0
c
c          u_x = 0.d0
c          u_y = 0.d0
c
c.....integral local density
c
c.....initialize variable d_loc
c
c*************************************************************************
C      Compute the velocity components of the wall nodes                 *
C      Modified by                                                       *
C                              Taiho Yeom                                *
C                              Oklahoma State university                 *
C                              May 2007                                  *
C*************************************************************************
c
         d_loc = 0.d0
c
         do 20 i = 0, 8
c
           d_loc = d_loc + node(i,x,y)
c
  20     continue
c
c.....x-, and y- velocity components
c
         u_x = (node(1,x,y) + node(5,x,y) + node(8,x,y)
    &         -(node(3,x,y) + node(6,x,y) + node(7,x,y))) / d_loc
c
         u_y = (node(2,x,y) + node(5,x,y) + node(6,x,y)
    &         -(node(4,x,y) + node(7,x,y) + node(8,x,y))) / d_loc
c.....pressure = average pressure
c
         press = density * c_squ

c                d_loc = density
c
c*************************************************************************
        else
c
c.....integral local density
c
c.....initialize variable d_loc
c
         d_loc = 0.d0
c
         do 30 i = 0, 8
c
           d_loc = d_loc + node(i,x,y)
c
  30     continue
c
c.....x-, and y- velocity components
c
```

```
          u_x = (node(1,x,y) + node(5,x,y) + node(8,x,y)
     &         -(node(3,x,y) + node(6,x,y) + node(7,x,y))) / d_loc
c
          u_y = (node(2,x,y) + node(5,x,y) + node(6,x,y)
     &         -(node(4,x,y) + node(7,x,y) + node(8,x,y))) / d_loc
c
c...........pressure
c
          press = d_loc * c_squ
c
          obsval = 0
c
        end if

c.........small loop to compute velocities, pressure and obstacle in arrays
                                  loc_den(x,y) = d_loc
                                  ux(x,y) = u_x
                                  vy(x,y) = u_y
                                  pressure(x,y) = press
                                  obstacle(x,y) = obsval
c
c.........write results to file
c
c         write(11,500) x, y, u_x, u_y, press, obsval

c500                format(I2,TR2,I2,TR2,E12.4,TR2,E12.4,TR2,E12.4,TR2,I2)

   10 continue
                      write(11,600) ((loc_den(x,y),y=1,ly),x=1,lx)
                      write(12,600) ((ux(x,y),y=1,ly),x=1,lx)
                      write(13,600) ((vy(x,y),y=1,ly),x=1,lx)
                      write(14,600) ((pressure(x,y),y=1,ly),x=1,lx)
                      write(15,700) ((obstacle(x,y),y=1,ly),x=1,lx)

600                   format(20(E12.4,X))
700                   format(20(I2,X))

c.....close file 'anb.dat'
c
     close(11)
           close(12)
           close(13)
           close(14)
           close(15)
c
     return
     end
c
     subroutine comp_rey(lx,ly,tau,density,ux,rey,node)
c
**********************************************************************
*                                                                    *
*    compute Reynolds number                                         *
*                                                                    *
*          Joerg BERNSDORF                                           *
*          C&C Research Laboratories, NEC Europe Ltd.                *
```

```
*           Rathausalle 10                                                  *
*           D-53757 Sankt Augustin, Germany                                 *
*                                                                           *
*    Last change: 1999/09/28                                                *
*                                                                           *
*****************************************************************************
c
     implicit none
c
     integer  lx,ly
c
     real*8  density,tau

           real*8 ux(lx,ly)

c.....local variables
c
     integer x,y,i
           real*8  visc, d_loc
           real*8 rey(lx,ly), node(0:8,lx,ly)
           open(16,file='reynolds.dat')

     do 10 y = 1, ly
       do 10 x = 1, lx

                    d_loc = 0.d0
c
          do 20 i = 0, 8
c
           d_loc = d_loc + node(i,x,y)
c
   20      continue
c
c.....compute viscosity
c
     visc = (1.d0/3.d0)*(2.d0*tau-1.d0)/(2.d0*d_loc)

            if (x .GE. 86 .and. x .LE. 94) then

c.....compute Reynolds number at orifice

     rey(x,y) = ux(x,y) * 14.d0 / visc

           else

c.....compute Reynolds number at channel

     rey(x,y) = ux(x,y) * float(ly-2) / visc

           end if

   10      continue

           write(16,800) ((rey(x,y),y=1,ly),x=1,lx)

  800 format(20(E12.4,x))
```

```
c.....messages
c
      write (6,*) '*** Calculations finished, results:'
      write (6,*) '***'
      write (6,*) '*** viscosity = ', visc
c     write (6,*) '*** average velocity = ', vel
c     write (6,*) '*** Reynolds number = ', rey
      write (6,*) '***'
      write (6,*) '*** In the file anb.dat, you can find local'
      write (6,*) '*** information about the simulated flow.'
      write (6,*) '***'
      write (6,*) '*** In the file anb_qx.out, you can find the average
     &'
      write (6,*) '*** flow velocity plotted as a function of time.'
      write (6,*) '***'

            close(16)
c
     return
     end
```

VITA

TAI HO YEOM

Candidate for the Degree of

Master of Science

Thesis: LATTICE BOLTZMANN METHOD FOR MICRO CHANNEL AND MICRO ORIFICE FLOWS

Major Field: Mechanical and Aerospace Engineering

Biographical:

Personal data: Born in Gwangju, South Korea, on August 19, 1979.

Education: Graduated from Gwangju Seoseok High School, Gwangju, South Korea in February 1998; received Bachelor of Science degree in Mechanical Engineering from Ajou university, Suwon, South Korea in August 2005. Completed the requirements for the Master of Science degree in Mechanical and Aerospace Engineering at Oklahoma State University in July, 2007

Experience: Employed by Korean Army as an artillery, February 2000 to April 2002; employed by Oklahoma State University, Department of Mechanical and Aerospace Engineering as a teaching assistant, Spring semester 2006, Fall semester 2006;

Professional Memberships: Member of American Society of Mechanical Engineers (ASME), Member of American Institute of Aeronautics and Astronautics (AIAA).

Name: Taiho Yeom Date of Degree: July, 2007

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: LATTICE BOLTZMANN METHOD FOR MICRO CHANNEL AND MICRO ORIFICE FLOWS

Pages in Study: 119 Candidate for the Degree of Master of Science

Major Field: Mechanical and Aerospace Engineering

Scope and Method of Study: The current project is performed numerically to gain an understanding of micro filtration by simulating gas flows through micro channels and micro orifices. The numerical simulation is based on the Lattice Boltzmann Method (LBM), which has many advantages for micro flows compared with the typical Navier-Stokes equation computation. Both no-slip bounce-back and slip bounce-back boundary conditions are used to get the slip velocity at the wall of a micro channel and a micro orifice. For the micro channel flows, Kn = 0.00194, 0.0194, and 0.194 are used. For the micro orifice flows, Kn = 0.02628 and five different Reynolds numbers between 2 and 12 are used. For the slip bounce-back boundary condition, the reflection factor and the accommodation coefficient are applied to obtain more accurate results.

Findings and Conclusions: The results of the micro channel flows match well with the analytical results in the slip flow regime. The results show that when the reflection factor is 0.85 the velocity and pressure profiles of the micro channel are in very good agreement with the analytical results for Kn = 0.0194. However, for Kn = 0.194, the current simulations with the reflection factor boundary condition show large deviations from the analytical results. In contrast, the accommodation coefficient scheme does not show good results for the current micro flow implementations at either Knudsen number. The reflection factor boundary condition also is applied to simulate micro orifice flows. For the orifice, the ratio of the open orifice area to the total area was selected as 0.6 in order to compare the results with the literature. From the velocity, density and pressure distributions of the micro orifice the compressibility effect is confirmed very well. An upturning region of the density at the end of the orifice and a separation area after the orifice require more study in the future. In addition, a more sophisticated relaxation time model should be developed for high Knudsen number and Reynolds number flows.

ADVISER'S APPROVAL: F. W. Chambers