COMPARISON OF THE HYBRID AND THERMAL LATTICE-BOLTZMANN METHODS

A Thesis
Presented to
The Academic Faculty

by

Jonathan D. Olander

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
Woodruff School of Mechanical Engineering

Georgia Institute of Technology
August 2009

COMPARISON OF THE HYBRID AND THERMAL LATTICE-BOLTZMANN METHODS

Approved by:

Professor Samuel Graham,
Committee Chair
Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Professor Cyrus Aidun, Advisor
Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Professor Yogendra Joshi
Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: 18 August 2009

ACKNOWLEDGEMENTS

I would like to thank Professor Aidun for his patience and guidance while I completed my research.

I want to thank Tim Maher for all of his help and guidance with programming. His teaching is what allowed me to move forward with my thesis.

I would like to thank Daniel Reazor, Jonathan Clausen, and Reza Khiabani for their assistance in helping me understand the material and in debugging the code that I attempted to write.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

BGK	Bhatnagar-Gross-Krook.
CFD	computational fluid dynamics.
D2Q9	abbreviation for a discretization of two dimensions and 9 velocities.
LBM	lattice-Boltzmann method.
LGA	lattice gas automaton.
MRT	Multiple Relaxation Times.
PPDF	particle probability distribution function.
TLBM	thermal lattice-Boltzmann method.
$\mathbf{a}$	external force.
α	thermal diffusion rate.
D_0	dimension of space.
δ_t	time step.
$\mathcal{D}$	diffusivity.
$\mathbf{e}_i$	particle velocity vectors.
ϵ	internal energy density.
f	particle probability distribution function.
f^{eq}	equalibrium particle probability distribution function.
$\mathbf{g}$	gravitational force.
g	thermal energy distribution function.
g^{eq}	equalibrium thermal energy distribution.
$\mathcal{G}$	interaction strength.
K	Knudsen number.
k_B	Boltzmann constant.
$\mathbf{L}$	square cavity wall length.
λ	Fourier constant.

N_A	Avogadro's number.
ν	kinematic fluid viscosity.
Ω_i	Bhatnagar-Gross-Krook collision operator.
p	pressure.
Π	stress tensor.
$\mathbf{Pr}$	Prandtl Number.
ψ_2	function of number density for temperature.
Ψ	stream function.
q	heat dissipation term.
R	ideal gas constant.
$\mathbf{Re}$	Reynolds number.
$\rho(\mathbf{x})$	fluid particle density at location $\mathbf{x}$.
t	time.
T	temperature.
τ	the single relaxation time.
τ_v	viscous relaxation time.
τ_T	thermal relaxation time.
θ	dimensionless temperature.
$\mathbf{u}$	macroscopic fluid velocity.
u	x-direction velocity.
U	fluid velocity.
v	y-direction velocity.
ε	kinetic energy.
ω	particle vorticity.
$\mathbf{W}$	width of cavity.
$\mathbf{x}$	location of a particle.
Z_i	viscous heating effects.

SUMMARY

The thermal lattice-Boltzmann method will be compared to the hybrid method in cavity flow situations following a flow simulations with the lattice-Boltzmann method. Lid-driven cavity flow is a well documented flow type and is simulated for comparison purposes of the thermal lattice-Boltzmann method and the hybrid method that utilized the energy equation. It is practical to simulate cavity flow due to the distinct flow patterns that emerge as the flow develops. Using the governing differential equations cavity flow has been simulated by Bozeman et al. [8]. The lattice-Boltzmann method has also been shown to accurately model cavity flow by Hou et al. [23]. The thermal lattice-Boltzmann method has also been shown to accurately model the thermal flow in a lid-driven cavity flow [27, 16, 5].

The flow and temperature field results will be validated through simulations of Couette flow. The flow will also be compared to the velocity data Ghia et al. generated through simulation of cavity flow [18]. The thermal lattice-Boltzmann method will be compared to the temperature fields generated by the energy equation of the hybrid method after the energy equation results are validated by the simulation of Couette flow. The effect of the Reynolds number is considered as the two methods for determining thermal flows are compared. To determine which method is better suited from computer simulations the two will be compared for computational demands and the speed of both convergence and computation.

The convergence criteria is maintained for the lattice-Boltzmann method, thermal lattice-Boltzmann method, and hybrid method. For each method the equations are applied to each point in the lattice structure once to compete a time-step.

The results will shows that the two methods produce very similar results. However, the thermal lattice-Boltzmann method is much more intensive computationally. It is safe to conclude that the hybrid method is a much more practical method to use in solving thermal flows. The hybrid method is better suited for simulations based on the fact that it is less computationally intensive and accurate.

CHAPTER I

INTRODUCTION

Lattice-gas automata (LGA), lattice-Boltzmann models (LBM), and thermal lattice-Boltzmann models (TLBM) are all relatively new methods that are very promising for the solution of nonlinear partial differential equations. The field of research began in 1986 after the publication of a paper by Frisch, Hasslacher, and Pomeau [17]. The paper showed that following collisions that conserve mass and momentum in the macroscopic limit leads to the Navier-Stokes equation. The condition for this process is that the lattice has symmetry. From the work done by Frisch et al. sprung the lattice-Boltzmann method, which has the luxury of being much more flexible than the original lattice-gas cellular automata.

The lattice-Boltzmann method (LBM) is considered an alternative to the various methods for solving the Navier-Stokes equations. The lattice-Boltzmann method has been successfully implemented in many fluid dynamics problems. The lattice-Boltzmann method originated from the lattice gas automata (LGA), a boolean fluid model. The LGA model simulates the motion of fluids with colliding particles on a symmetrical lattice. The average fluid variables, most notably the density and velocity, satisfy equations similar to the Navier-Stokes equations. The improvement of the lattice-Boltzmann method over the lattice gas automata is that the lattice-Boltzmann method deals with the averaged distribution functions. By dealing with the averaged distribution function the statistical averaging step in the original LGA is eliminated. The model was simplified even further by adding the collision model of Bhatnagar-Gross-Krook (BGK) to the lattice-Boltzmann equation. With the Bhatnagar-Gross-Krook model many complex fluid phenomena can be modeled, such as capillary action, multiple phase flows, and nonlinear diffusion.

Thermal lattice-Boltzmann models began to develop after the publication of a paper by McNamara and Alder in 1993 [30]. In McNamara's paper a thermal lattice-Boltzmann model was developed; however, the method suffered from the need of additional velocities to maintain stability of the calculations. Since 1993 the method has been further developed and simplified.

The thermal lattice-Boltzmann method (TLBM) has not had the same level of success as the lattice-Boltzmann models due to problems of instability with the current models. The primary current models are the multi-speed approach by McNamara and Alder [30], the passive-scalar approach by Shan [38], and the thermal distribution model proposed by He et al. [21]. Peng et al. simplified the thermal distribution model for incompressible thermal flows by removing the complicated gradient operator term for the temperature. Another method was proposed by Khiabani et al. in which the energy equation was combined with the lattice-Boltzmann method [25]. The hybrid method solves thermal flow problems while avoiding the complications added using a rigorous thermal lattice-Boltzmann method as proposed by He.

1.1 Applications

In the Pulp and Paper science the modeling of heat transfer is a valuable ability. Applications include, but are not limited to, paper drying, coating drying, and hot pressing. Being able to model such processes could save a great deal of money by determining the benefits are worth the purchase of expensive equipment.

One of the most common drying processes is running the paper around steam-heated rollers. The rollers are heated by steam and then the energy is transfered from the rollers to the paper. The energy then causes the water in the paper to evaporate. Figure 1.1 shows an example of a Fourdrinier machine.

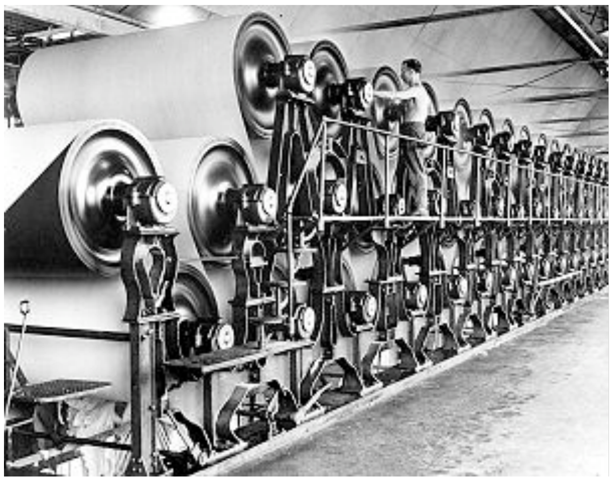

Figure 1.1: A classic example of a Fourdrinier machine

After the paper is formed in a Fourdriner machine it is run over multiple steam-heated rollers. Figure 1.1 shows an example of the drying section. Modeling the section before installing it could insure that the proper number of steam-heated rollers are installed.

Drying is also important for the coating of the paper. Often times the paper is coated with either a powder or a chemical to change the properties of the paper. Often the coating is added to dry paper so a two phase simulation can be run to simulate the addition and drying of the coating agent.

Before the drying section there is often a press. The press attempts to press the water out of the paper and supporting mesh. However, after the press when the paper expands it re-absorbs some water. To attempt to minimize the post-press absorption of water the press is sometimes heated.

1.2 Goals and Objectives

The thermal lattice-Boltzmann method will be compared to the hybrid method in cavity flow situations. Lid-driven cavity flow is a well documented flow type. It is practical to simulate cavity flow due to the distinct flow patterns that emerge as the flow develops. Using the governing differential equations cavity flow has been simulated by Bozeman et al. [8]. The lattice-Boltzmann method has also been shown to accurately model cavity flow by Hou et al. [23]. The thermal lattice-Boltzmann method has also been shown to accurately model the thermal flow in a lid-driven cavity flow [27, 16, 5].

The thermal lattice-Boltzmann method will be compared to the temperature fields generated by the energy equation of the hybrid method. The effect of the Reynolds number is considered as the two methods for determining thermal flows are compared. To determine which method is better suited from computer simulations the two will be compared for computational demands and the speed of both convergence and computation.

The results shows that the two methods produce very similar results. However, the thermal lattice-Boltzmann method is much more intensive computationally. It is safe to conclude that the hybrid method is a much more practical method to use in solving thermal flows. The hybrid method is better suited for simulations based on the fact that it is less computationally intensive and accurate.

1.3 Lattice-Boltzmann Method

The lattice-Boltzmann Equation method is alternative numerical approach to typical computational fluid dynamics (CFD). As stated before, the lattice-Boltzmann Equation method originated from the lattice gas automaton (LGA). In the standard lattice gas automaton model the number of particles at a site with a given velocity is limited to zero or one. Also, the densities calculated with the lattice gas automaton model exhibit a large amount of statistical noise. The noise is accounted for by performing coarse-grain averaging. The need for the coarse grain averaging, as well as other limitations, has led to the development of the lattice-Boltzmann models.

Figure 1.2 shows an example of a two-dimensional lattice gas automaton defined on a square lattice. The lattice is defined as having particle speeds of 0, 1, and $\sqrt{2}$. Node 0 in the figures carries the defined speed of 0. The other speeds are based on the distance between them and node 0. In Figure 1.2, the left figure shows the

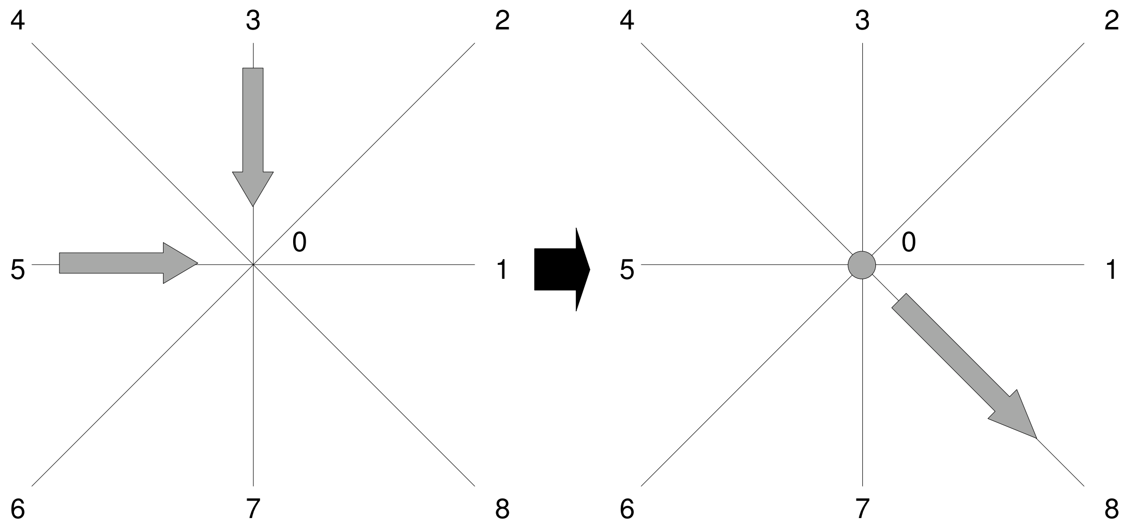

Figure 1.2: A two body collision

.

input of two particles colliding each at a speed of 1. The right part of Figure 1.2 shows the output of a stationary particle and one of speed $\sqrt{2}$. While the lattice gas automaton model would operate with boolean operators to determine the input and output of Figure 1.2, the lattice-Boltzmann method model uses real-valued variables in the particle probability distribution function (PPDF).

In the LBM the movement of particles is assumed to be discrete in both time and space. Being discrete in both time and space means that there is a set of directions with given velocities within the method. Also, each interaction between the particles takes place after a given time-step. After the time-step a particle will arrive at a new site and undergo a collision. During the collision the particles are assumed to conserve both their number and momentum. The transport equation for a single particle probability distribution function is given by the equation

$$(\partial_t + \mathbf{e}_i \cdot \nabla_{\mathbf{x}} + \mathbf{a} \cdot \nabla_{\mathbf{e}_i}) f(\mathbf{x}, \mathbf{e}_i, t) = (\partial_t f_{\text{coll}}) \tag{1.1}$$

where t is the time, f is the particle probability distribution function, $\mathbf{e}_i$ is the particle velocity vectors, $\mathbf{a}$ is the external force, and $\mathbf{x}$ is the particle's location. From Equation 1.1 the lattice-Boltzmann Equation is derived. A common form of the lattice-Boltzmann Equation is

$$f_i(\mathbf{x}+\mathbf{e}_i, t+1) - f_i(\mathbf{x}, t) = -\frac{1}{\tau}[f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)] \tag{1.2}$$

where τ is a single relaxation time. As in Equation 1.1, the $f_i(\mathbf{x}, t)$ in Equation 1.2 represents the single-particle distribution function, and the $f_i^{(0)}(\mathbf{x}, t)$ is the equilibrium distribution time. The right side of Equation 1.2 is known as the Bhatnagar-Gross-Krook (BGK) collision operator and is designated as the symbol Ω_i. The substitution of Ω_i into Equation 1.2 will simplify it to

$$f_i(\mathbf{x}+\mathbf{e}_i, t+1) = f_i(\mathbf{x}, t) + \Omega_i \tag{1.3}$$

In a two-dimensional space there are nine distance velocity vectors for the lattice-Boltzmann method. Equation 1.3 is discretized using a D2Q9 lattice (two dimensions, nine velocities) as shown in Figure 1.3. The discrete velocities of the locations shown

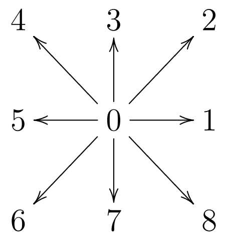

Figure 1.3: Schematic plot of velocity directions for a typical point

in Figure 1.3 are given by

$$\mathbf{e}_i = \begin{cases} \mathbf{e}_i = (0,0) & i = 0 \\ \mathbf{e}_i = \left(\cos(\frac{i-1}{4}\pi), \sin(\frac{i-1}{4}\pi)\right) & i = 1,3,5,7 \\ \mathbf{e}_i = \left(\frac{2}{\sqrt{2}}\cos(\frac{i-1}{4}\pi), \frac{2}{\sqrt{2}}\sin(\frac{i-1}{4}\pi)\right) & i = 2,4,6,8 \end{cases} \tag{1.4}$$

Using the discrete velocities, the discrete lattice-Boltzmann Equations are

$$f_i^{eq}(\mathbf{x},t) = \begin{cases} \frac{4}{9}\rho(\mathbf{x})\left[1 - \frac{3}{2}\mathbf{u}^2\right] & i = 0 \\ \frac{1}{9}\rho(\mathbf{x})\left[1 + 3(\mathbf{e}_i \cdot \mathbf{u}) + \frac{9}{2}(\mathbf{e}_i \cdot \mathbf{u})^2 - \frac{3}{2}\mathbf{u}^2\right] & i = 1,3,5,7 \\ \frac{1}{36}\rho(\mathbf{x})\left[1 + 3(\mathbf{e}_i \cdot \mathbf{u}) + \frac{9}{2}(\mathbf{e}_i \cdot \mathbf{u})^2 - \frac{3}{2}\mathbf{u}^2\right] & i = 2,4,6,8 \end{cases} \tag{1.5}$$

where $\rho(\mathbf{x})$ is the fluid particle density at node $\mathbf{x}$, and $\mathbf{u}$ is the macroscopic fluid velocity. It is important to note that the discrete lattice-Boltzmann method equations in Equation 1.5 only hold true while the distance between points in the distribution function is equal to the time-step being used.

1.3.1 Collision Interval Theory

The simplification that occurs from Equation 1.2 to Equation 1.3 is based on the assumption that for a small time-step the particle probability distribution function, f, is nearly equal to the equilibrium value of the particle probability distribution function, f_i^{eq}. Following the assumption that the distribution functions are nearly equation, the equilibrium particle probability distribution function can be calculated as

$$f_i^{eq} = \frac{\rho}{(2\pi RT)^{D_0/2}} \exp\left[-\frac{(\mathbf{e}-\mathbf{u})^2}{2RT}\right] \tag{1.6}$$

where D_0 is the dimension of space, R is the ideal gas constant, and T is the temperature.

The assumption of Equation 1.6 is how the Bhatnagar-Gross-Krook collision operator, Ω_i, is derived from $\partial_t f_{\text{coll}}$ of Equation 1.1 [7].

1.3.2 Link to Hydrodynamics

The density ρ and the macroscopic fluid velocity $\mathbf{u}$ are determined from the fist two moments. The moments are given by the first moment equation

$$\rho(\mathbf{x},t) = \sum_i f_i(\mathbf{x},t) \tag{1.7}$$

and the second moment equation

$$\rho(\mathbf{x},t)\mathbf{u}(\mathbf{x},t) = \sum_i f_i(\mathbf{x},t)\mathbf{e}_i \tag{1.8}$$

and the third moment equation

$$\frac{\rho(\mathbf{x},t)DRT}{2} = \sum_i \frac{(\mathbf{e}_i - \mathbf{u}(\mathbf{x},t))^2}{2} f_i(\mathbf{x},t) \tag{1.9}$$

Using the first two moment equations, Equations 1.7 and Equation 1.8, the density and macroscopic fluid velocity can be calculated.

Initially the equilibrium probability distribution function, f^{eq}, is calculated using Equation 1.5 and the initial density. The $t+1$ of Equation 1.3 represents a very small time-step immediately after a collision. Boundary conditions are set upon the fluid and cause movement. A new probability distribution function, f, can be calculated. Comparing the two distributions, as shown in Equation 1.2, a post collision distribution function is determined. In the next time-step, the post collision distribution function, $f_i(\mathbf{x}+\mathbf{e}_i, t+1)$, becomes the equilibrium distribution function, f^{eq}.

1.3.3 Bounce-back

For all flow situations boundary conditions must be applied. In the lattice-Boltzmann method the interaction of a fluid particle with a solid particle is performed using the bounce-back method. The bounce-back method is used to model the interaction of the fluid with the walls of the cavity in the lid-driven cavity flow. The bounce-back method takes the velocity directions of a typical point in the lattice structure, shown in Figure 1.3, and reverses the orientation. The new velocity direction of the point in the lattice structure is shown in Figure 1.4.

The bounce-back satisfies both the no-penetration and the no-slip boundary conditions. During the simulation the calculated velocity should be very nearly zero from the points in the lattice structure that go through the bounce-back operation. Opposite walls should not interact. In a cavity where the top has a designated velocity

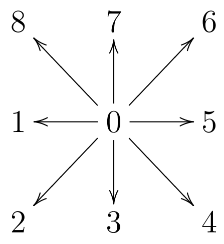

Figure 1.4: Schematic plot of velocity directions for a typical point

the velocity should not be passed to the bottom of the cavity with periodic boundary conditions. The bounce-back operation prevents the periodic boundary conditions to have opposite walls interact.

CHAPTER II

THERMAL LATTICE-BOLTZMANN METHOD

The primary methods for thermal lattice-Boltzmann models currently fall into one of three categories: the multi-speed approach [30], the passive-scalar approach [38], and the thermal energy distribution model [21]. The multi-speed approach uses the same distribution function to define the macroscopic velocity, pressure, and temperature. However, it requires additional velocities to connect each lattice structure. There are two distributions in the passive-scalar model: one for the velocity and density and another for the temperature. The thermal energy distribution model uses the energy distribution function as derived from the Boltzmann equation. In the thermal energy distribution model there is a complicated gradient term that is usually attempted to be simplified out to keep the equations simple.

2.1 Multi-speed Approach

McNamara and Alder created the multi-speed method to solve thermal problems in conjunction with the lattice-Boltzmann Method [30]. In the multi-speed approach only the density distribution function is used. However, to obtain the temperature distribution at the macroscopic level, additional speeds are necessary. McNamara first set up the equations expressing the microscopic conservation of mass momentum, and energy. To solve the conservation equations for velocities using the lattice-Boltzmann method, the moments are taken (i.e. Equations 1.7 and 1.8). McNamara and Alder again found the moments but of higher order than before to account for the additional equation for the conservation of energy in their multi-speed application of the lattice-Boltzmann method to hydrodynamics. The third moment found expresses the microscopic conservation of energy, Equation 2.1, as developed by Nourgaliev [33].

$$\rho(\mathbf{x},t)\varepsilon = \sum_i f_i(\mathbf{x},t)\mathbf{e}_i \tag{2.1}$$

where the kinetic energy. ε, is given by

$$\varepsilon = \frac{D_0 k_B T}{2} = \frac{D_0 RT}{2N_A} \tag{2.2}$$

where N_A is Avogadro's number and k_B is the Boltzmann Constant.

From the stress and energy moments, McNamara and Alder use the lattice-Boltzmann equations and the equations they derived for the dissipative corrections to the ideal fluid to derive the macroscopic thermo-fluid equations of motion. In order to guarantee linear independence of the moment equation used there is a need for at least 13 different particle velocities for a two dimensional simulation. The lattice-Boltzmann method needs 9 velocities and an additional 4 velocities for the energy equations. The multi-speed approach is numerically instable and the temperatures are limited to a narrow range [28].

In 1994, McNamara and Alder's research was supplemented by some research done by Chen, Ohashi, and Akiyama [10]. Chen et al.'s article follows McNamara's development of the multi-speed thermal model and then attempts to numerically model various flows and compare them with theoretical results. They also employed one extra moment tensor in an attempt to stablize the method. Their results showed a more stable simulation of theoretical flows but suffered from the need for even more velocities than the approach developed by McNamara and Alder.

Around the same time as the paper was published, Alexander, Chen, and Sterline published "Lattice-Boltzmann thermohydrodynamics" [3]. The "Lattice-Boltzmann thermohydrodynamics" paper follows a similar methodolgy to that employed by McNamara and Alder. Alexander et al. attempted to use the energy conservation equation and apply it to the lattice-Boltzmann equations [3]. The results are more stable than those of McNamara's, but it forces the introduction of several very complex terms and calculations. “Lattice-Boltzmann thermohydrodynamics” is similar to McNamara and Alder’s multi-speed approach, but it is more of a precurser to He, Chen, and Doolen’s thermal energy distribution model.

In 1995, McNamara attempted to stabalize the multi-speed method further by using Lax-Wendroff advection to provide an adjustable time-step [28, 29]. The Lax-Wendroff method is a numerical method for the solution of partial differential equations based on finite differences. McNamara attempted to understand the instability from the previous numerical schemes by applying the Lax-Wendroff method to the advection equation, Equation 2.3, and then compare these results to particle probability evolution equation, Equation 1.2.

$$\frac{\partial f_i}{\partial t} = -\mathbf{e}_i \cdot \nabla f_i \tag{2.3}$$

Using the Lax-Wendroff method, McNamara found a way to decrease the time-step to less than one and modify the constants of the lattice-Boltzmann equation to improve the stability of the thermal lattice-Boltzmann method. While the results proved to be more stable than before, they were still lacked sufficient stability.

2.2 Passive-Scalar Approach

Shan et al. used the advection-diffusion equation to simulate the temperature field while simultaneously solving the lattice-Boltzmann equation for mass and momentum conservation [38]. Shan et al. saw the need for an improvement to McNamara et al.’s method for simulating thermal effects simultaneously with fluid flows. McNamara et al.’s method suffers from numerical instability, especially in three dimensions. His paper also declares that when inter-particle forces are included in the multiphase model that the energy conservation becomes even more complicated. Due to complications, it is easiest to deal with fluids that can be assumed to behave as an ideal gas.

Shan et al. dealt with the multiple component lattice-Boltzmann equation model.

Instead of having a second fluid, the second component simulates a passive temperature field. The lattice-Boltzmann equations were solved for as explained previously with Equations 1.2 through 1.8. The lattice-Boltzmann method equations must satisfy Equation 1.7 and Equations 2.4 and 2.5

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.4}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} + \mathbf{g} \tag{2.5}$$

where $\mathbf{u}$ is given by Equation 2.6

$$\mathbf{u} = \mathbf{u}_1 + \frac{\mathbf{g}}{2} \tag{2.6}$$

where $\mathbf{g}$ is gravitational force, p is the pressure, and ν is the kinematic viscosity. The kinematic viscosity, ν, is found using Equation 2.7 [23].

$$\nu = \frac{2\tau - 1}{6} \tag{2.7}$$

Equation 2.7 is the kinematic viscosity's relationship to the relaxation time used in the lattice-Boltzmann method. The second component of the multiple component lattice-Boltzmann model, or the temperature field, must satisfy the "passive-scalar" equation.

$$\frac{\partial \theta}{\partial t} + \mathbf{u} \cdot \nabla \theta = \nabla \cdot (\mathcal{D} \nabla \theta) \tag{2.8}$$

where $\mathcal{D}$ is the diffusivity and is given by the equation

$$\mathcal{D} = \frac{1}{3} \left\{ \tau (1 + 9\mathcal{G}\psi_2 \frac{d\psi_2}{df_2} - \frac{1}{2} \right\} \tag{2.9}$$

where $\mathcal{G}$ is the interaction strength and ψ_2 is a function of the number density of the second component–the temperature component for the multiple component lattice-Boltzmann equation model in discussion. To simplify the simulation, $\mathcal{G}$ is set to zero. With the interaction strength, $\mathcal{G}$, being set to zero, Equation 2.8 becomes

$$\frac{\partial \theta}{\partial t} + \mathbf{u} \cdot \nabla \theta = \nu \nabla^2 \theta \tag{2.10}$$

where θ is the dimensionless temperature and defined as

$$\theta = \frac{T - T_{initial}}{T_{boundary} - T_{initial}} \tag{2.11}$$

The "passive-scalar" approach was used to model Rayleigh-Bénard convection. It simulated the temperature field accurately when compared with theoretical results, but suffered from the fact that it doubled the resources needed for a typical lattice-Boltzmann model simulation.

The entire method is based on the fact that the macroscopic temperature satisfies the evolution equation that is the same as the "passive-scalar". The conditions that must be met to apply this method are that both the heat dissipation and the compression work are negligible [21]. The "passive-scalar" approach would be much more useful if the method could incorporate heat dissipation and compression work.

Lallemand and Luo attempted to improve the stability of the thermal lattice-Boltzmann method in 2003 by combining the multi-speed approach and the passive-scalar approach [26]. They first solved the mass and momentum conservation equations using a multi-speed lattice-Boltzmann equation where a two dimentional lattice is solved using 13 discrete velocities. They then use a finite difference technique and the momentum to solve for the energy and thus the temperature. Lallemand and Luo conclude that the instability of previous thermal lattice-Boltzmann models comes from the collision operator. Lallemand is a proponent of the Multiple-Relaxation-Times (MRT) model over the Bhatnagar-Gross-Krook model. The Multiple-Relaxation-Times model for the thermal lattice-Boltzmann method is also attempted by d'Humiéres et al. in 2002 [11] and Treeck et al. in 2006 [44]. Lallemand and Luo's model was applied to turbulent conditions by Treeck et al. by extending the model with a Smagorinski subgrid scale model.

2.3 Thermal Energy Distribution Model

To solve thermo-hydrodynamic problems using the lattice-Boltzmann model He et al. introduced an internal energy density distribution function to simulate the temperature field [21]. The macroscopic density and velocity fields are still simulated as explained earlier using the density distribution function (Equations 1.2 through 1.8). The thermal model for the lattice-Boltzmann method is derived by discretizing the continuous evolution equation for the internal energy distribution. It is similar to the "passive-scalar" method in the fact that it also implements an independent distribution function for the temperature evolution. It is an improvment to previous methods because it is able to simulate viscous heat dissipation and compression work by directly simulating the evolution of internal energy.

As before, the evolution of a single-particle density distribution in a system follows the Boltzmann equation, Equation 1.2. The additional moment that is needed for the thermal calculations, following the notation of Equations 1.7 and 1.8, is

$$\rho\epsilon = \int \frac{(\mathbf{e}-\mathbf{u})^2}{2} f d\mathbf{e} \tag{2.12}$$

where ϵ is the internal energy density and is given by the equation

$$\epsilon = \frac{D_0 RT}{2} \tag{2.13}$$

Also, the Boltzmann equations still must satisfy the continuity and momentum equations at the Navier-Stokes level, as shown in Equations 2.4 and 2.5. The single-relaxation-time lattice-Boltzman Bhatnagar-Gross-Krook model, the method that has been explained up to this point, suffers from the fact that the thermal conductivity can not be adjusted independently of the kinematic viscosity. To account for this shortfall and calculate the internal energy or temperature with arbitrary Prandtl number, an internal energy density distribution function, g, is introduced

$$g = \frac{(\mathbf{e}-\mathbf{u})^2}{2} f \tag{2.14}$$

The internal energy density distribution, g, is given that name because it is equation to the internal energy density, $\rho\epsilon$, when integrated over the velocity space. Incorporating Equation 2.14 with Equation 1.3 the Equation becomes

$$\partial_t g + (\mathbf{e} \cdot \nabla) g = \frac{(\mathbf{e} - \mathbf{u})^2}{2} \Omega(f) - fq \tag{2.15}$$

where q is the heat dissipation term and is given by the equation:

$$q = (\mathbf{e} - \mathbf{u}) \cdot [\partial_t \mathbf{u} + (\mathbf{e} \cdot \nabla)\mathbf{u}] \tag{2.16}$$

The new collision model based on Equation 2.15 is

$$\frac{(\mathbf{e} - \mathbf{u})^2}{2} \Omega(f) = -\frac{g - g^{eq}}{\tau_c} \tag{2.17}$$

where

$$g^{eq} = \frac{\rho(\mathbf{e} - \mathbf{u})^2}{2(2\pi RT)^{\frac{D}{2}}} \exp\left[-\frac{(\mathbf{e} - \mathbf{u})^2}{2RT}\right] \tag{2.18}$$

Equation 2.18, the equation for internal energy equilibrium distribution, g^{eq}, is very similar to the equilibrium particle probability distribution, f^{eq}.

$$f^{eq} = \frac{\rho}{(2\pi RT)^{\frac{D}{2}}} \exp\left[-\frac{(\mathbf{e} - \mathbf{u})^2}{2RT}\right] \tag{2.19}$$

Using Equations 2.18 and 2.12, the new moment used for calculating the macroscopic variables is

$$\rho\epsilon = \int g d\mathbf{e} \tag{2.20}$$

where ϵ is still calculated using Equation 2.13.

Using the Chapman–Enskog multiscale expansion, the time derivative is expanded to $\partial_t = K\partial_{t_0} + K^2\partial_{t_1} + ...$ The K is the Knudsen number, the mean free path divided by the hydrodynamic length scale, is assumed to be small. Both the density distribution and the internal energy distribution function are similarily expanded.

$$f = f^{eq} + Kf^{(1)} + K^2 f^{(1)} + ... \tag{2.21}$$

$$g = g^{eq} + Kg^{(1)} + K^2 g^{(1)} + ... \tag{2.22}$$

The space derivative used in these situations is $\nabla = K\nabla_1$. The first-order Chapman–Enskog approximation of Equation 2.15 is

$$\partial_{t_0} g^{eq} + (\mathbf{e}\cdot\nabla_1) g^{eq} = -\frac{g^{(1)}}{\tau} - f^{eq} q \tag{2.23}$$

The integral of Equation 2.23 of velocity space is

$$\partial_{t_0}(\rho\epsilon) + \nabla_1\cdot(\rho\mathbf{u}\epsilon) = -p\nabla_1\cdot\mathbf{u} \tag{2.24}$$

which is he macroscopic energy equation for Euler fluids. The second-order Chapman–Enskog approximation of Equation 2.15 is

$$\partial_{t_1} g^{eq} + \left[\partial_{t_0} + (\mathbf{e}\cdot\nabla_1)\right] g^{(1)} = -\frac{g^{(2)}}{\tau} - f^{(1)} q \tag{2.25}$$

where

$$f^{(1)} = -\tau\left[\partial_{t_0} f^{eq} + \nabla_{(1)}\cdot(\epsilon f^{eq})\right] \tag{2.26}$$

Equation 2.26 is is the standard first-order nonequilibrium deviation of the density distribution [23]. Combining $f^{(1)}$ from Equation 2.26 and $g^{(1)}$ from Equation 2.23 into Equation 2.25 and then integrating over the velocity space gives the equation

$$\partial_{t_1}(\rho\epsilon) = \nabla_1\cdot(\rho\alpha\nabla\epsilon) + \Pi \tag{2.27}$$

where Π is the stress tensor and is given by

$$\Pi = \rho\nu(\nabla\mathbf{u} + \mathbf{u}\nabla) \tag{2.28}$$

and α is the thermal diffusion rate and given by the equation

$$\alpha = \frac{(D_0+2)\tau RT}{D_0} \tag{2.29}$$

After integrating Equation 2.15 in one time-step the equation becomes

$$\begin{aligned}
g(\mathbf{x}+\epsilon\delta_t,\epsilon,t+\delta_t) - g(\mathbf{x},\epsilon,t) = \\
-\frac{\delta_t}{2\tau}\left[g(\mathbf{x}+\epsilon\delta_t,\epsilon,t+\delta_t) - g^{eq}(\mathbf{x}+\epsilon\delta_t,\epsilon,t+\delta_t)\right] \\
-\frac{\delta_t}{2} f(\mathbf{x}+\epsilon\delta_t,\epsilon,t) q(\mathbf{x}+\epsilon\delta_t,\epsilon,t) - \frac{\delta_t}{2\tau}\left[g(\mathbf{x},\epsilon,t) - g^{eq}(\mathbf{x},\epsilon,t)\right] \\
-\frac{\delta_t}{2} f(\mathbf{x},\epsilon,t) q(\mathbf{x},\epsilon,t)
\end{aligned} \tag{2.30}$$

where δ_t is the time-step. To simplify the equation, a new variable is introduced.

$$\bar{g} = g + \frac{\delta_t}{2\tau}(g - g^{eq}) + \frac{\delta_t}{2} f q \tag{2.31}$$

Using the new variable from Equation 2.31 and simplifying Equation 2.30, the evolution equation for $\bar{g}$ is

$$\begin{aligned} &\bar{g}_i(\mathbf{x} + \mathbf{e}_i\delta_t, t + \delta_t) - \bar{g}_i(\mathbf{x}, t) = \\ &\qquad - \frac{\delta_t}{\tau + 0.5\delta_t}\left[\bar{g}_i(\mathbf{x}, t) - g_i^{eq}(\mathbf{x}, t)\right] - \frac{\tau}{\tau + 0.5\delta_t} f_i(\mathbf{x}, t) q_i(\mathbf{x}, t)\delta_t \end{aligned} \tag{2.32}$$

The discrete velocities, $\mathbf{e}_i$, are the same as with the discretized lattice-Boltzmann equations and are shown again in Equation 2.33

$$\mathbf{e}_i = \begin{cases} \mathbf{e}_i = (0, 0) & i = 0 \\ \mathbf{e}_i = \left(\cos(\frac{i-1}{4}\pi), \sin(\frac{i-1}{4}\pi)\right) & i = 1, 3, 5, 7 \\ \mathbf{e}_i = \left(\frac{2}{\sqrt{2}}\cos(\frac{i-1}{4}\pi), \frac{2}{\sqrt{2}}\sin(\frac{i-1}{4}\pi)\right) & i = 2, 4, 6, 8 \end{cases} \tag{2.33}$$

Using the discrete velocities, the discrete internal energy dnsity equilibrium distribution is

$$g_i^{eq}(\mathbf{x}, t) = \begin{cases} -\frac{2\rho(\mathbf{x})\epsilon}{3}\mathbf{u}^2 & i = 0 \\ \frac{\rho(\mathbf{x})\epsilon}{9}\left[\frac{3}{2} + \frac{3}{2}(\mathbf{e}_i \cdot \mathbf{u}) + \frac{9}{2}(\mathbf{e}_i \cdot \mathbf{u})^2 - \frac{3}{2}\mathbf{u}^2\right] & i = 1, 3, 5, 7 \\ \frac{\rho(\mathbf{x})\epsilon}{36}\left[3 + 6(\mathbf{e}_i \cdot \mathbf{u}) + \frac{9}{2}(\mathbf{e}_i \cdot \mathbf{u})^2 - \frac{3}{2}\mathbf{u}^2\right] & i = 2, 4, 6, 8 \end{cases} \tag{2.34}$$

Equations 2.33 and 2.34 are linked to hydrodynamics using the equation:

$$\rho\epsilon = \sum_i \bar{g}_i - \frac{\delta_t}{2}\sum_i f_i q_i \tag{2.35}$$

The Equations 2.33, 2.34, and 2.35 constitute the lattice-Boltzmann thermal equation.

Peng et al. noticed that the thermal energy distribution model was largely being neglected due to the complications that arose from the complexity of the evolution equation, Equation 2.32 [34]. Peng et al. decided to simplify the model by neglecting the last term of the evolution equation because compression work done by pressure and viscous heat dissipation can be neglected for incompressible flow. The simplified thermal evolution equation is

$$g_i(\mathbf{x} + \mathbf{e}_i\delta_t, t + \delta_t) - g_i(\mathbf{x}, t) = -\frac{1}{\tau}\left[g_i(\mathbf{x}, t) - g_i^{eq}(\mathbf{x}, t)\right] \tag{2.36}$$

All other parts of He's lattice-Boltzmann thermal equations stay the same. The simplification of the thermal energy distribution model still provided accurate results without the complicated gradient term of the evolution equation.

D'Orazio and Succi used the same simplification as Peng et al. when working with the thermal energy distribution model–they neglected the complicated gradient term of the evolution equation [13]. However, they did add a forcing term to the evolution equation to account for viscous heating.

$$g_i(\mathbf{x} + \mathbf{e}_i\delta_t, t + \delta_t) - g_i(\mathbf{x}, t) = -\frac{\delta_t}{\tau + 0.5\delta_t}\left[g_i(\mathbf{x}, t) - g_i^{eq}(\mathbf{x}, t)\right]\frac{\delta_t\tau}{\tau + 0.5\delta_t}Z_i f_i \tag{2.37}$$

where Z_i is the term that represents the effects of viscous heating and is given by the equation

$$Z_i = \frac{[\mathbf{e}_i - \mathbf{u}(\mathbf{x}, t)] \cdot [\mathbf{u}(\mathbf{x} + \mathbf{e}_i\delta_t, t + \delta_t) - \mathbf{u}(\mathbf{x}, t)]}{\delta_t} \tag{2.38}$$

Equations 2.33 and 2.34 are linked to hydrodynamics using D'Orazio and Succi's method with the equation:

$$\rho\epsilon = \sum_i \bar{g}_i - \frac{\delta_t}{2}\sum_i f_i Z_i \tag{2.39}$$

The discretized equations of the thermal probability distribution remain the same as those developed by He in D'Orazio and Succi's study. The calculations for viscous are only applicable for turbulent flows, which are indicated by the Reynolds number of the flow.

Lü et al. took the work done by He et al. and the simplifications proposed by Peng et al. for incompressible cases and applied them to a hexagonal lattice [27]. Lü added the ability to test the thermal energy distribution model under various Prandtl

Table 1: Equilibrium Density Distribution Function Constants

	$i=0$	$i=1,3,5,7$	$i=2,4,6,8$
$A=$	$\rho-\frac{8}{3}\rho\epsilon(1-\epsilon)+12\lambda\epsilon^2$	$\frac{\rho\epsilon(3-4\epsilon)}{6}-3\lambda\epsilon^2$	$\frac{\rho\epsilon(4\epsilon-1)}{18}+\lambda\epsilon^2$
$B=$	0	$\frac{\rho(1-4/3\epsilon)}{2}$	$\frac{\rho(4\epsilon-1)}{18}$
$C=$	$\frac{4}{3}\rho(2\epsilon-1)$	$\frac{\rho(\epsilon-3/4)}{3}$	$\frac{\rho(1/2-2\epsilon)}{18}$
$D=$	0	$\rho(1-2\epsilon)$	$\frac{\rho(6\epsilon-1)}{27}$
$E=$	0	$-\frac{2\rho}{9}$	$\frac{2\rho}{81}$

numbers. The Prandtl number, **Pr**, is the ratio between the kinematic viscosity, ν, and the thermal diffusion rate, α, as shown in Equation 2.40.

$$\mathbf{Pr}=\frac{\nu}{\alpha} \tag{2.40}$$

The kinematic viscosity, ν, and the thermal conductivity, α, are given respectively in Equation 2.41

$$\begin{aligned}\nu&=\epsilon\left(\tau-\tfrac{1}{2}\right)\\ \alpha&=(2\rho\epsilon+18\lambda\epsilon)\left(\tau-\tfrac{1}{2}\right)\end{aligned} \tag{2.41}$$

where λ is a constant parameter that is introduced according to the Fourier law while determining constants for f_i^{eq}. The equilibrium density distribution function is given as

$$f_i^{eq}=A+B\mathbf{e}_i\cdot\mathbf{u}+C\mathbf{u}^2+D(\mathbf{e}_i\cdot\mathbf{u})^2+E(\mathbf{e}_i\cdot\mathbf{u})^3 \tag{2.42}$$

The coefficients for the equilibrium density distribution of Equation 2.42 are given in Table 1. Using Equation 2.41, the equation for the Prandtl number becomes

$$\mathbf{Pr}=\frac{\rho}{2\rho+18\lambda} \tag{2.43}$$

In order for the equations to satisfy the equilibrium distribution functions the constant A must follow the constraints presented in Table 2. Based on the constraints in Table 2, the constant parameter must be in the range of

$$-\frac{\rho}{9}<\lambda<\frac{\rho}{9}\cdot\frac{2(1-\epsilon)}{\epsilon} \tag{2.44}$$

Table 2: Equilibrium Constraints

$i = 0$	$i = 1, 3, 5, 7$	$i = 2, 4, 6, 8$
$A < \rho$	$A > 0$	$A > 0$

and the internal energy density must be in the range of

$$\frac{\mathbf{Pr}}{2\mathbf{Pr} + 1} < \epsilon < \frac{3\mathbf{Pr}}{2\mathbf{Pr} + 1} \tag{2.45}$$

By adding the Prandtl number into the simulation, Lü et al. was able to model thermal fluids more accurately because the kinematic viscosity and the thermal conductivity were no longer directly proportional to the local density.

2.4 Advancements to the Thermal lattice-Boltzmann Method

Zhang and Chen published a paper in 2003 that details the implementation of a thermal lattice-Boltzmann model in a multiphase flow [47]. Zhang and Chen solved the density and momentum conservation equations using using a multiphase lattice-Boltzmann method. The evolution of the temperature was then solved for using a scalar energy transport equation, very similar or the "passive-scalar" approach. However, Zhang's model is limited by the fact that it only applies to ideal gases and fails to take into account many variables such as surface tension.

Shi et al. developed the thermal lattice-Boltzmann discretized equations using a different relation of the thermal energy distribution function, g, to the particle probability distribution function, f [40]. The relation proposed by Shi et al. is given by

$$g = \frac{(\mathbf{e} - \mathbf{u})^2}{3R} f \tag{2.46}$$

Using Equation 2.46 the new equilibrium thermal energy probability distribution becomes

$$g^{eq} = \frac{(\mathbf{e} - \mathbf{u})^2}{3R} f^{eq} = \frac{\rho(\mathbf{e} - \mathbf{u})^2}{3R(2\pi RT)^{\frac{3}{2}}} \exp\left[-\frac{(\mathbf{e} - \mathbf{u})^2}{2RT}\right] \tag{2.47}$$

Using Taylor expansion up to $\mathbf{u}^2$, Equation 2.19, the equilibrium particle probability distribution, becomes

$$f^{eq} = \rho \left(\frac{1}{2\pi RT}\right)^{3/2} \exp\left(-\frac{\mathbf{e}^2}{2RT}\right) \left[1 + \frac{(\mathbf{e}\cdot\mathbf{u})}{RT} + \frac{(\mathbf{e}\cdot\mathbf{u})^2}{2R^2T^2} - \frac{\mathbf{u}^2}{2RT}\right] \tag{2.48}$$

and Equation 2.18, the equalibrium thermal energy probability distribution, becomes

$$\begin{aligned} g^{eq} &= \rho T \left(\frac{1}{2\pi RT}\right)^{3/2} \exp\left(-\frac{\mathbf{e}^2}{2RT}\right) \\ &\times \left[\frac{\mathbf{e}^2}{3RT} + \left(\frac{(\mathbf{e}^2}{3RT} - \frac{2}{3}\right)\frac{(\mathbf{e}\cdot\mathbf{u})}{RT} + \left(\frac{\mathbf{e}^2}{3RT} - \frac{4}{3}\right)\frac{(\mathbf{e}\cdot\mathbf{u})^2}{2R^2T^2} - \left(\frac{\mathbf{e}^2}{3RT} - \frac{2}{3}\right)\frac{\mathbf{u}^2}{2RT}\right] \end{aligned} \tag{2.49}$$

After simplifying Equation 2.49 by canceling out the higher order terms the equalibrium thermal energy probability distribution is

$$\begin{aligned} g^{eq} &= \rho T \left(\tfrac{1}{2\pi RT}\right)^{3/2} \exp\left(-\tfrac{\mathbf{e}^2}{2RT}\right) \left[1 + \tfrac{(\mathbf{e}\cdot\mathbf{u})}{RT} + \tfrac{(\mathbf{e}\cdot\mathbf{u})^2}{2R^2T^2} - \tfrac{\mathbf{u}^2}{2RT}\right] \\ &= T f^{eq} \end{aligned} \tag{2.50}$$

For low Mach numbers, the equation for g^{eq} can be reduced even further by neglecting the terms on an order of $\mathbf{u}^2$.

$$g_i^{eq}(\mathbf{x}, t) = \begin{cases} \frac{4}{9}\rho T(\mathbf{x}) & i = 0 \\ \frac{1}{9}\rho T(\mathbf{x})\left[1 + 3(\mathbf{e}_i \cdot \mathbf{u})\right] & i = 1, 3, 5, 7 \\ \frac{1}{36}\rho T(\mathbf{x})\left[1 + 3(\mathbf{e}_i \cdot \mathbf{u})\right] & i = 2, 4, 6, 8 \end{cases} \tag{2.51}$$

The new discretized equations for the thermal lattice-Boltzmann method are linked to hydrodynamics through the equation

$$\rho T = \sum_i g_i \tag{2.52}$$

Finally, the evolution equation used is esentially the same as that proposed by D'Oraizo and Succi. Therefore, the Equations 2.33, 2.51, and 2.52 constitute the lattice-Boltzmann thermal equation. The model proposed by Shi incorporates the viscous heat dissipation but does not include buoyancy effects. It is important to recognize that the viscous heat dissipation is not included in the calculations when it is not applicable. Viscous heat dissipation occurs when the flow becomes turbulent which is indicated by the Reynolds number.

Azwadi and Tanahashi took the method developed by Shi et al. and attempted to lessen the computational power needed by reducing the number of velocities from 9 to 4 in a two dimensional space and from 27 to 8 in a three dimensional space. In the two dimensional space, the discretized thermal energy equation is given by

$$g_i^{eq}(\mathbf{x},t) = \frac{1}{4}\rho T(\mathbf{x})\left[1+3(\mathbf{e}_i \cdot \mathbf{u})\right] \qquad i=1,2,3,4 \tag{2.53}$$

The lattice structure and the discretized equations for the density distribution function, f^{eq}, remains the same in the calculations. The typical lattice structure for the density distribution function is shown in the left part of Figure 2.4. The right part of Figure 2.4 shows the set up of the lattice structure used by Azwadi and Tanahashi for their thermal energy distribution function. Azwadi and Tanahashi used a similar

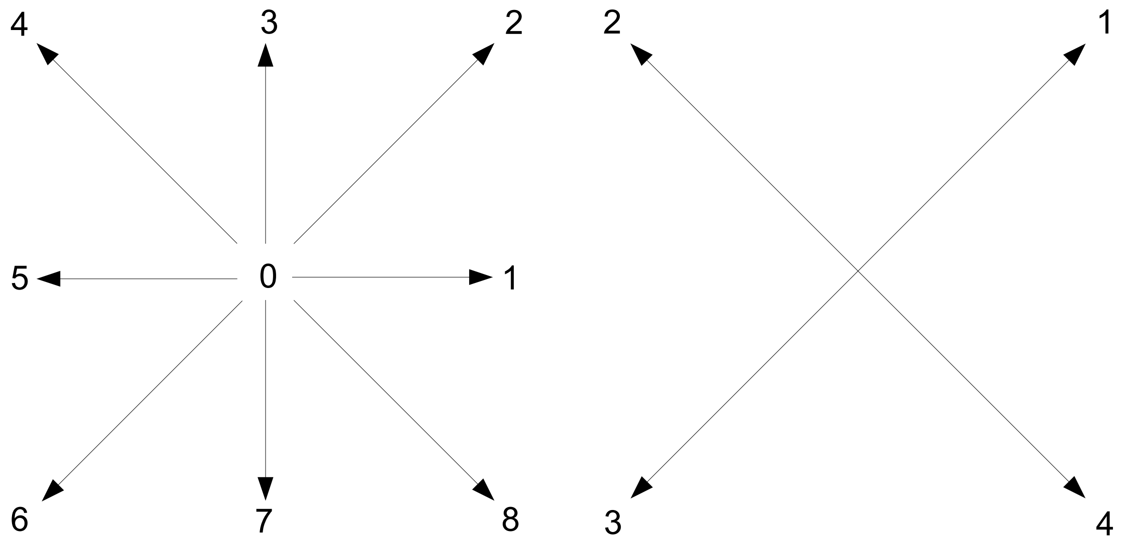

Figure 2.1: Lattice of the density probability distribution and the lattice of the thermal probability distribution

structure as shown in Figure 2.4 for three dimensions where the eight corners of the cube are the only velcities used in the thermal energy distribution function. Both of the studies with the newly defined lattice structures produced accurate results without as much computing power as previously needed for implementing the thermal lattice-Boltzmann method.

There has also been research done on implementing the thermal lattice-Boltzmann model as proposed by He in microflow situations by Shu et al. [41]. In order to implement the thermal lattice-Boltzmann model it was necesarry to redefine the relatation times and implement diffuse scattering and temperature jump boundary conditions. The numerical simulations in the microflow study compared well with experimental results.

CHAPTER III

LID-DRIVEN CAVITY FLOW: OVERVIEW AND IMPLEMENTATION

3.1 Basic Principles

Lid-driven cavity flow is the fluid motion that is the result of a plate moving over a cavity. Figure 3.1 shows the basic setup for a two dimensional case of lid-driven cavity flow. The moving top wall is at a given temperature, T_1, and given speed, U.

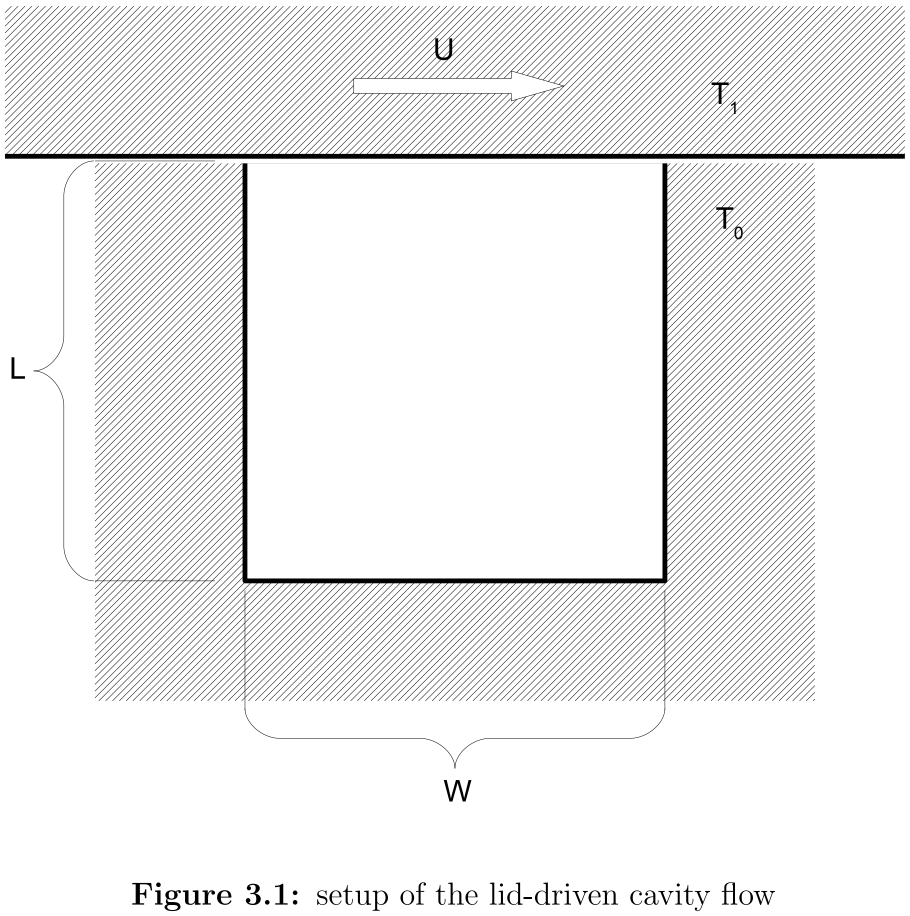

Figure 3.1: setup of the lid-driven cavity flow

The stationary walls that surround the fluid are at another temperature, T_0. The cavity has a height of L and a width of W. In the current simulation a square cavity is simulated and L is used for both height and width.

The primary method for solving fluid flows currently is through the Navier-Stokes equations. In 2005, Erturk et al. performed a study on lid-driven cavity flow by solving non-linear Navier-Stokes equations in an iterative fashion [16]. In a two-dimensional, incompressible, and axisymmetric flow it is useful to simplify the Navier-Stokes equations by using the stream function, Ψ, and the vorticity, ω. The vorticity is given by the equation

$$\vec{\omega} = \nabla \times \mathbf{u} \tag{3.1}$$

and the stream function is given by the equations

$$\frac{\partial \Psi}{\partial y} = u \qquad \text{and} \qquad -\frac{\partial \Psi}{\partial x} = v \tag{3.2}$$

where u is the velocity in the x-direction on the coordinate axis, and v is the velocity in the y-direction on the coordinate axis. Combining Equations 3.1 and 3.2 the continuity equation becomes

$$-\omega = \frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \Psi}{\partial y^2} = 0 \tag{3.3}$$

The vorticity in situations with laminar flow, which applies to flow with Reynolds numbers less than approximately 2100, is equal to zero. Vorticity is the local rotation of fluid elements. The low Reynolds number type flows are considered irrotational flows. The non-dimensional Navier-Stokes equation is given by the equation

$$\frac{1}{\mathbf{Re}}\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right) = \frac{\partial \Psi}{\partial y}\frac{\partial \omega}{\partial x} - \frac{\partial \Psi}{\partial x}\frac{\partial \omega}{\partial y} \tag{3.4}$$

The Navier-Stokes equation is a non-linear system. To solve the equation Erturk used an iterative method. To apply an iterative method pseudo time derivatives are assigned to Equations 3.3 and 3.4. The new equations are

$$\frac{\partial \Psi}{\partial t} = \frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \Psi}{\partial y^2} \tag{3.5}$$

$$\frac{\partial \omega}{\partial t} = \frac{1}{\mathbf{Re}}\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right) + \frac{\partial \Psi}{\partial y}\frac{\partial \omega}{\partial x} - \frac{\partial \Psi}{\partial x}\frac{\partial \omega}{\partial y} \tag{3.6}$$

Equations 3.5 and 3.6 are then discretized and solved computationally. The streamlines from the results at a Reynolds number of 1000 are presented in Figure 3.1. As

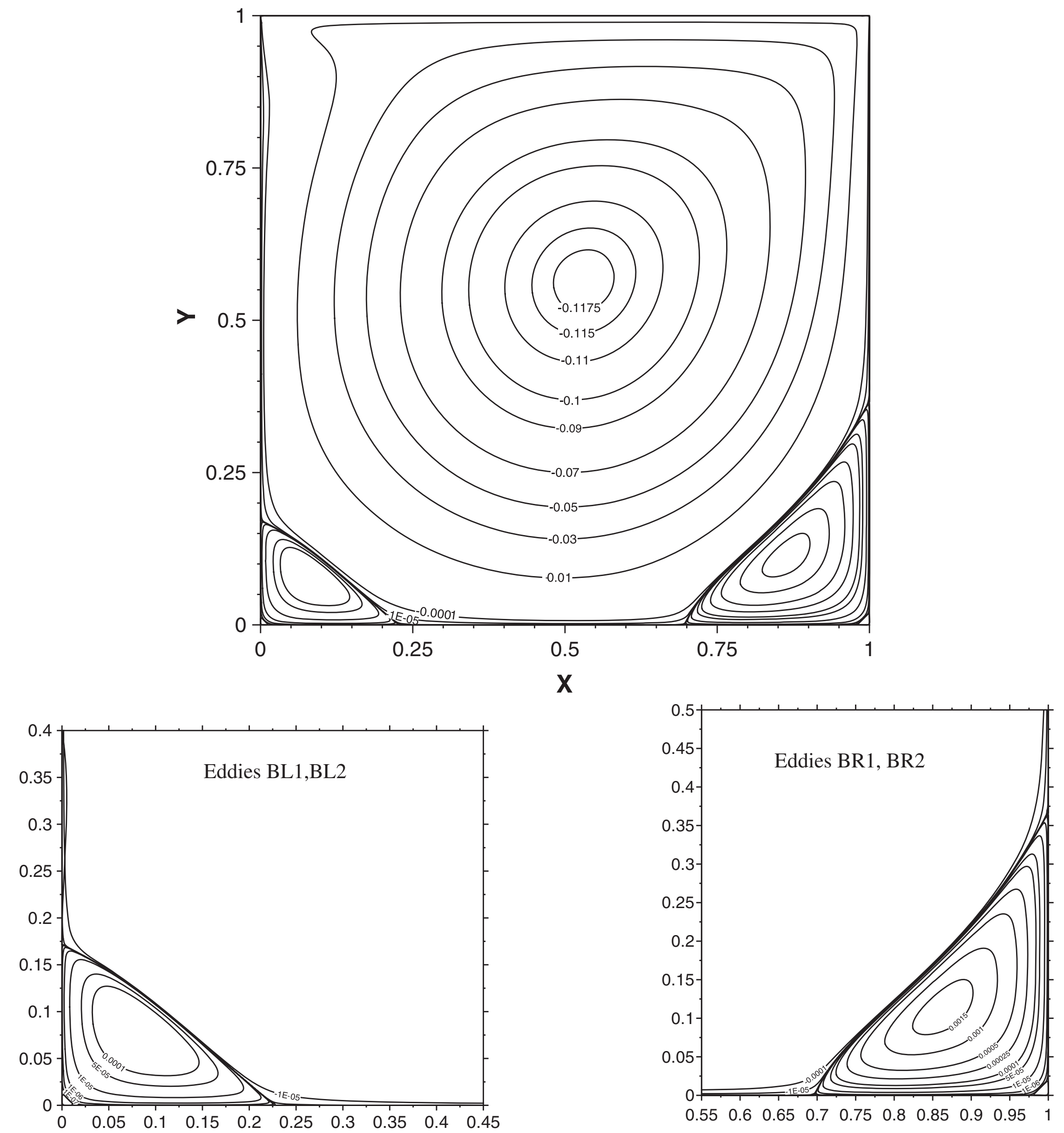

Figure 3.2: Cavity flow with a Reynolds number of 1000 [16]

the top plate moves to the right, the fluid in the cavity begins to turn clockwise. As the flow develops, counter-clockwise flow begins to develop in the bottom two corners. However, the Reynolds number is not high enough to create turbulence or a counter-current flow in the upper left of the cavity.

Figure 3.1 presents a good visualization of the streamlines for cavity flow. The dominate counter-clockwise is clearly visible as well as the cavitation in the bottom corners that spins in a clockwise direction. The higher the Reynolds number, which correlates directly with the velocity of the top plate, the more the cavitation will grow. Also, for larger Reynolds numbers the primary vortex will move closer to the center of the cavity. Figure 3.1 shows that for a Reynolds number of 1000 the vortex is close to the center of the cavity. For smaller Reynolds number values the vortex will move closer to the upper-right corner of the cavity.

3.2 Simulation

In the lattice-Boltzmann method the units do not correlate directly with more standard units that would be used in equations such as the energy equation. There are equations that relate the variables used in the lattice-Boltzmann method to typical fluid properties. The relaxation time is related to the kinematic viscosity, ν, by Equation 3.7 [23].

$$\nu = \frac{2\tau_v - 1}{6} \tag{3.7}$$

The Reynolds number, **Re**, is given by Equation 3.8.

$$\mathbf{Re} = \frac{U \times L}{\nu} \tag{3.8}$$

In Equation 3.8 the U is the velocity of the top plate and the L is the depth of the cavity. Instead of conventional units, L is measured in the number of nodes in the lattice structure. The velocity, U, and the length, L, are the values used to non-dimensionalize the energy equation and make sure that the energy equation produces the same results as the thermal lattice-Boltzmann method.

With the conventional single relaxation time method, there is no way to adjust the Prandtl number for the flows. Lü et al. addressed the rigidity of the Prandtl number in thermal lattice-Boltzmann simulations by using Fourier constants [27]. An alternative method to Lü et al.'s is to just use two different relaxation times, one for the lattice-Boltzmann method and one for the thermal lattice-Boltzmann method. To differentiate between the two relaxation times, they are given different notations. The viscous relaxation time becomes τ_v and the thermal relaxation time becomes τ_T. Using these notations for the relaxation times, the lattice-Boltzmann equation becomes

$$f_i(\mathbf{x}+\mathbf{e}_i, t+1) - f_i(\mathbf{x}, t) = -\frac{1}{\tau_v}[f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)] \tag{3.9}$$

and the thermal lattice-Boltzmann equation becomes

$$g_i(\mathbf{x}+\mathbf{e}_i, t+1) - g_i(\mathbf{x}, t) = -\frac{1}{\tau_T}[g_i(\mathbf{x}, t) - g_i^{eq}(\mathbf{x}, t)] \tag{3.10}$$

The discretized equations for both the lattice-Boltzmann method and the thermal lattice-Boltzmann method do not change. The directions for both the lattice-Boltzmann method and the thermal lattice-Boltzmann method are the same.

$$\mathbf{e}_i = \begin{cases} \mathbf{e}_i = (0,0) & i=0 \\ \mathbf{e}_i = \left(\cos(\frac{i-1}{4}\pi), \sin(\frac{i-1}{4}\pi)\right) & i=1,3,5,7 \\ \mathbf{e}_i = \left(\frac{2}{\sqrt{2}}\cos(\frac{i-1}{4}\pi), \frac{2}{\sqrt{2}}\sin(\frac{i-1}{4}\pi)\right) & i=2,4,6,8 \end{cases} \tag{3.11}$$

The discretized lattice-Boltzmann equations are

$$f_i^{eq}(\mathbf{x}, t) = \begin{cases} \frac{4}{9}\rho(\mathbf{x})\left[1-\frac{3}{2}\mathbf{u}^2\right] & i=0 \\ \frac{1}{9}\rho(\mathbf{x})\left[1+3(\mathbf{e}_i\cdot\mathbf{u})+\frac{9}{2}(\mathbf{e}_i\cdot\mathbf{u})^2-\frac{3}{2}\mathbf{u}^2\right] & i=1,3,5,7 \\ \frac{1}{36}\rho(\mathbf{x})\left[1+3(\mathbf{e}_i\cdot\mathbf{u})+\frac{9}{2}(\mathbf{e}_i\cdot\mathbf{u})^2-\frac{3}{2}\mathbf{u}^2\right] & i=2,4,6,8 \end{cases} \tag{3.12}$$

The discretized thermal lattice-Boltzmann equations, as developed by Shi et al. are

$$g_i^{eq}(\mathbf{x}, t) = \begin{cases} \frac{4}{9}\rho T(\mathbf{x}) & i=0 \\ \frac{1}{9}\rho T(\mathbf{x})\left[1+3(\mathbf{e}_i\cdot\mathbf{u})\right] & i=1,3,5,7 \\ \frac{1}{36}\rho T(\mathbf{x})\left[1+3(\mathbf{e}_i\cdot\mathbf{u})\right] & i=2,4,6,8 \end{cases} \tag{3.13}$$

These equations are linked to hydrodynamics by the equations

$$\rho(\mathbf{x}, t) = \sum_i f_i(\mathbf{x}, t) \tag{3.14}$$

$$\rho(\mathbf{x},t)\mathbf{u}(\mathbf{x},t) = \sum_i f_i(\mathbf{x},t)\mathbf{e}_i \tag{3.15}$$

and

$$\rho(\mathbf{x},t)T(\mathbf{x},t) = \sum_i g_i(\mathbf{x},t) \tag{3.16}$$

The thermal diffusion rate, α, can be calculated using Equation 3.17 [4].

$$\alpha = \tau_T - \frac{1}{2} \tag{3.17}$$

The Prandtl number can now be defined as

$$\mathbf{Pr} = \frac{\nu}{\alpha} = \frac{\frac{2\tau_v - 1}{6}}{\tau_T - \frac{1}{2}} \tag{3.18}$$

The two relaxation times were used in the setup that was run for the comparison of the thermal lattice-Boltzmann a hybrid methods. This allowed a greater deal of flexibility and accuracy. The Prandtl number, being the ratio of kinematic viscosity and thermal diffusivity, is more of a fluid property than a quantifier of flow or heat dissipation. Unlike the Reynolds number and Grashof number, the Prandtl number has no length scale. The lack of a length scale means that the Prandtl number is dependent on the fluid and the fluid state. The ability to prescribe a Prandtl number to the thermal lattice-Boltzmann method allows the modeling the heat transfer and movement of a specific fluid.

The Prandtl number also allows the thermal lattice-Boltzmann method to be more closely modeled to the hybrid method as developed by Khiabani et al. [25]. In this simulation, the thermal lattice-Boltzmann method is compared to the non-dimensionalized hybrid method.

3.2.1 The Hybrid Method for Solving for Temperature

The hybrid method obtains the temperature field by solving the energy equation, Equation 3.19.

$$\rho c_p \left(\frac{\partial T}{\partial t} + \vec{u} \cdot \nabla T \right) = k \nabla^2 T \tag{3.19}$$

The dimensionless temperature, with reference to the notation in Figure 3.1, is defined as

$$\theta = \frac{T - T_0}{T_1 - T_0} \tag{3.20}$$

In the simulation the lattice units for time, location, and velocity were converted to non-dimensional terms using a reference velocity and length scale before being used. The dimensionless energy equation, using the dimensionless units, becomes

$$\frac{\partial \theta}{\partial t} + \vec{u}^* \cdot \nabla \theta = \frac{1}{\mathbf{Re} \times \mathbf{Pr}} \nabla^2 \theta \tag{3.21}$$

The non-dimensional energy equation is then discretized for implementation in an iterative code. Each of the partials from the non-dimensional energy equation can be broken down into algebraic equations. The partial of theta with respect to time becomes

$$\frac{\partial \theta}{\partial t} = \frac{\theta(x, y, t+1) - \theta(x, y, t)}{\Delta t^*} \tag{3.22}$$

The partial with respect to the x-direction becomes

$$\frac{\partial \theta}{\partial x} = \frac{\theta(x+1, y, t) - \theta(x-1, y, t)}{2 \times \Delta x^*} \tag{3.23}$$

The partial with respect to the y-direction, or even the z-direction for a three dimensional lattice, is the same as that shown in Equation 3.23 except that the “x” must be substituted with a “y” or “z” respectively.

Finally, the second derivative with respect to a direction, such as the x-direction, is shown in Equation 3.24.

$$\frac{\partial^2 \theta}{\partial x^2} = \frac{\theta(x-1, y, t) + \theta(x+1, y, t) - 2 \times \theta(x, y, t)}{(\Delta x^*)^2} \tag{3.24}$$

Using Equation 3.22, 3.23, and 3.24 the non-dimensional energy equation is discretized and used to calculate the temperature profile of the flow in an iterative fashion. By removing the gradient terms of Equation 3.21 and substituting the equivalent partials the energy equation becomes

$$\frac{\partial \theta}{\partial t} + u^* \frac{\partial \theta}{\partial x} + v^* \frac{\partial \theta}{\partial y} = \frac{1}{\mathbf{Re} \times \mathbf{Pr}} \left(\frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \theta}{\partial y^2} \right) \tag{3.25}$$

The u and v are the non-dimensional velocities. The non dimensional velocity is chosen to be the velocity of the fluid at a given location divided by the top plate velocity. The Δx and Δy must also be non-dimensional. To make the lengths non-dimensional they are divided by the length L. As described before L is measured in the number of nodes in the lattice structure in a given direction.

$$u^* = \frac{u}{U} \tag{3.26}$$

$$\Delta x^* = \frac{1}{L} \tag{3.27}$$

$$\Delta t^* = \frac{U}{L} \tag{3.28}$$

All of the terms in Equations 3.21 through 3.25 use the non-dimensional terms given in Equations 3.26 through 3.27.

Equation 3.25 can be combined with Equations 3.22, 3.23, and 3.24 to find the equation that is used for iteration.

3.3 Implementation

Both the thermal lattice-Boltzmann method and the method that solves the energy equation, the hybrid method, were run with the same boundary conditions. The boundary conditions are dimensionless, as stated before. The dimensionless temperature values used in the discretized thermal lattice-Boltzmann method are simply scaled from zero to one. Using this scale, and a viscous relaxation time close to one, both methods should display very similar results. The boundary conditions are

$$\begin{aligned} \theta = T_0 = 0 \quad &\text{at} \quad x = 0, \text{and L} \\ \theta = T_0 = 0 \quad &\text{at} \quad y = 0 \\ \theta = T_1 = 1 \quad &\text{at} \quad y = \text{L} \end{aligned} \tag{3.29}$$

For cavity flow, all four boundaries are held at a constant temperature. For a different scheme, such as Couette flow, the non-bounded parts of the lattice structure would need boundary conditions. The typical boundary condition for a non-bounded boundary is adiabatic. In the simulation of a square cavity the width is the same as the height. Equation 3.29 uses the same length, L, for both the height and width of the square cavity.

The simulations run to compare the thermal lattice-Boltzmann method and the hybrid method were performed in C#. C# allows for a convenient user interface to change the parameters. For purposes of comparison it was desirable to be able to run the simulation at varying Reynolds numbers. The simulation of the flow is a simple implementation of the lattice-Boltzmann method. There are no body forces added to the flow. The lack of body forces means that the code is only suited to simulate incompressible flow of a uniform material.

All of the boundaries of the simulation are flat walls. The flat walls allows for a simple bounce-back to be performed rather than the more complicated bounce-back procedures that have been developed for curved boundaries [19, 32, 49]. A simple bounce-back at a flat surface is sufficient for stable results.

The code first initializes the lattice structure that is going to be used. During the initialization there are three matrices created for the probability distributions. There is one matrix for the particle probability distribution function, one for the equilibrium particle probability distribution function, and one for the particle probability distribution function after a time-step. The code for the simulation is presented in the Appendix.

The thermal lattice-Boltzmann method requires the exact same three matrices. In addition to the six distribution matrices there is a matrix for the velocity in the x-direction, a matrix for the velocity in the y-direction, a matrix for the temperature as calculated by the energy equation, a matrix for the density of the fluid, and a matrix for the temperature as calculated from the thermal lattice-Boltzmann method. There is also a boolean matrix that designates whether the node is solid or a fluid. "On" is designated as being a solid node in the lattice structure. Following the initialization, there is an iterative structure with the order of operations being as follows:

1. The propigation of the distribution functions, probability (f^{eq}) and thermal (g^{eq}), is performed in the directions as shown in Figure 1.3.

2. The calculation of velocity and temperature is performed based off of the distribution functions. The operation for determining the velocity and temperature is determined by the Equations 3.14, 3.15, and 3.16. During this step the boundary conditions are also set The boundary conditions include the wall temperatures, the top plate temperature, and the top plate velocity.

3. The re-calculation of both the equilibrium particle probability distribution function and the equilibrium thermal distribution function is performed following Equations 3.12 and 3.13.

4. The collision process is performed for both the probability and the thermal lattices using the respective Equations 3.9 and 3.10.

5. A simple bounce-back is performed at all of the solid boundaries.

The bounce-back procedure does not include the top plate of the cavity structure, only the three walls of the cavity. Prior to the bounce-back, all boundaries are periodic. The bounce-back prevents the node on opposite walls from interacting. Bounce-back also is a way of implementing a no-slip boundary condition. The no-slip boundary condition means that immediately next to the boundary the fluid will have the same velocity as the wall. Instead of performing a bounce-back operation on the top row of the lattice (the top plate of the cavity), the velocity is set to the velocity of the top plate instead of being calculated by Equation 3.15.

The final step of the code is to export the data to files that could be read by Tecplot 360 or Paraview.

To test the validity of the code it was set up to simulate Couette flow. Couette flow is the laminar flow of a viscous fluid between to infinite parallel plates. A realistic simulation of this kind of flow is the flow between two concentric, rotating cylinders. Couette flow with no pressure or body forces creates a flow with a linear velocity profile between the velocity of the top and bottom plate. The temperature field should have the same profile as the velocity but perhaps on a different scale.

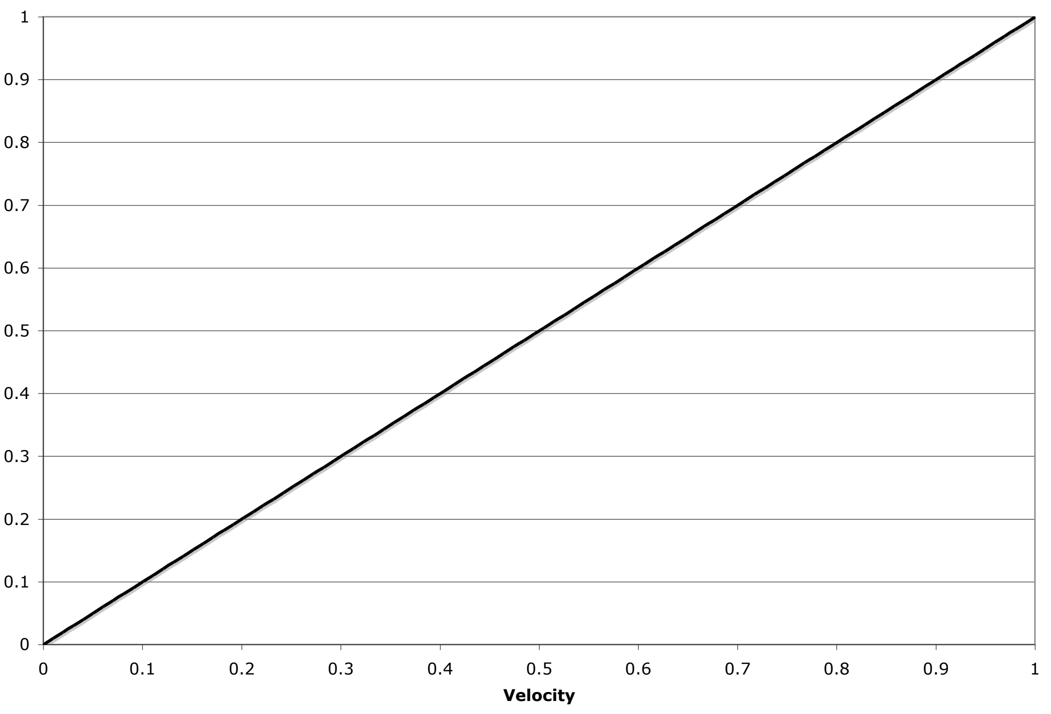

Figure 3.3: The normalized velocity profile for Couette flow

Figure 3.3 shows the normalized velocity profile of Couette flow with the top plate being the high velocity and the bottom plate being the low velocity. The velocity profile shown in Figure 3.3 verifies that the process is basically correct, albeit simple.

3.3.1 Implementation of Cavity Flow

From the beginning of the implementation of the cavity flow it was obvious that the velocities next to the moving top plate were not accurate when the viscous relaxation time, τ_v, was not equal to one or very nearly one. All lattice-Boltzmann schemes are unstable when τ_v is 0.5. The instability is demonstrated by Equation 3.7. If τ_v is set to 0.5 the kinematic fluid viscosity, ν, becomes null.

Another problem encountered was the long processing time. To make sure that the Reynolds number being implemented in the iterative process was exactly the desired one the viscous relaxation time was calculated from the combination of Equations 3.7 and 3.8.

$$\tau_v = \frac{1}{2} + \frac{3 \times U \times L}{\mathbf{Re}} \tag{3.30}$$

In the lattice-Boltzmann scheme the length L is the number of nodes in the lattice structure instead of the height of the cavity in metric units.

Another constraint that must be imposed is that the top plate velocity must be relatively small. The Bhatnagar-Gross-Krook scheme for the lattice-Boltzmann method only apples to speeds that are significantly less than the speed of sound. The speed of sound in the the Bhatnagar-Gross-Krook method is $\frac{1}{\sqrt{3}}$. To satisfy the speed constraint and Equation 3.30 for the viscous relaxation time the lattice structure must be relatively large. The code developed for the comparison of the thermal lattice-Boltzmann method and the hybrid method initially ran two lattice-Boltzmann schemes and the hybrid method's energy equation all at the same time. The process of running all three schemes at the same time is memory intensive for the computer. In order to obtain results in a timely fashion it was acceptable to have some error that could be easily rationalized. Eventually the simulation was made to run each method separately so the number of time-steps to converge for each method could be recorded.

The convergence criteria is maintained for the lattice-Boltzmann method, thermal lattice-Boltzmann method, and hybrid method. For each method the equations are applied to each point in the lattice structure once to compete a single time-step.

The parameters for the simulations are presented in Table 3 reveal that both the thermal and the viscous relaxation times at a Reynolds number of 100 are above one.

Table 3: Simulation parameters for various Reynolds Numbers

$ReynoldsNumber$	100	500	1000
$x - dimensions$	256	1000	256
$y - dimensions$	256	1000	256
$Prandtlnumber$	0.35	0.72	0.3
τ_T	1.01	.63875	0.585
τ_v	1.0355	0.7997	0.5765
U	0.07	0.05	0.10
$Time - Steps$	49541	300000	3683573

Values of the relaxation times being slightly above one provide stable results for the velocity and thermal lattice-Boltzmann method.

The closer the relaxation times are to 0.5 the larger the jump is between the boundaries and the values in the lattice structure. The errors are most pronounced in the simulations of a Reynolds number of 1000.

Figure 3.4 is supposed to illustrate two simulations of a Reynolds number of 1000. The current simulation in Figure 3.4 is obviously of a larger Reynolds number than the Ghia et al.'s simulation. The maximum magnitude of negative velocities much are larger in the current simulation than they should be as shown from the results of Ghia et al.'s simulation in the same figure. It is easy to see the large difference in magnitude between the velocities in the two figures.

From a comparison of the horizontal centerlines from the current simulation and Ghia et al.'s results, shown in Figure 3.5, it shows that a larger Reynolds number creates a maximum negative magnitude much larger than Ghia's simulations. Again, the difference in velocity is indicative of a Reynolds number that is too large for the current simulation.

3.3.2 Implementation of the thermal lattice-Boltzmann method

The thermal lattice-Boltzmann method is run the same way as the lattice-Boltzmann method but it calculates the temperature instead of the velocity. If the thermal

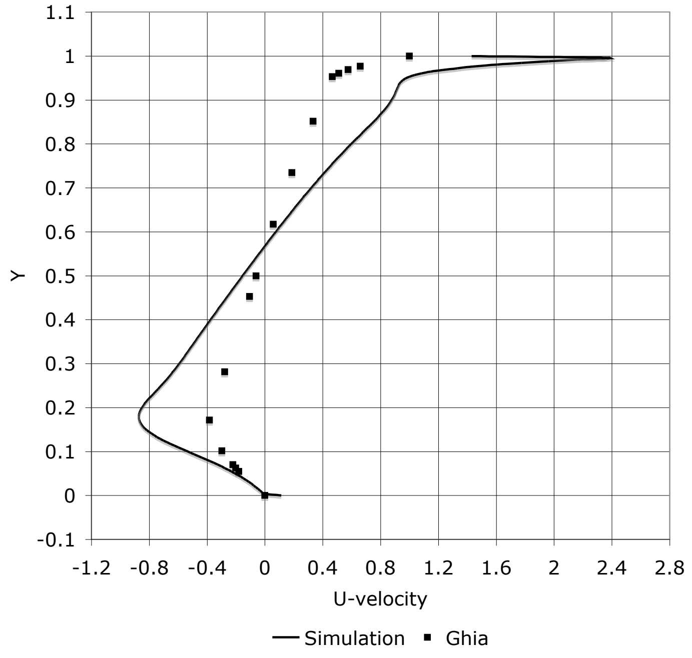

Figure 3.4: Profile of the velocity from a vertical centerline from a 256 × 256 simulation of cavity flow with Re=1000 [18]

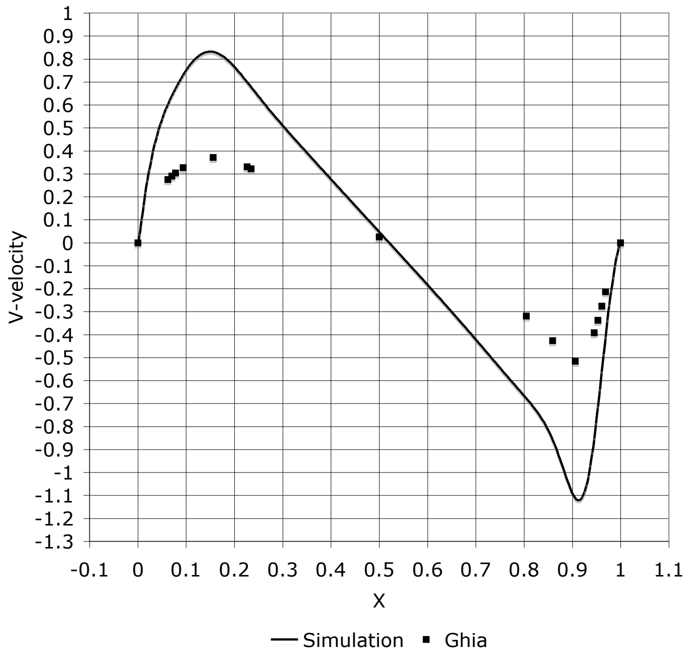

Figure 3.5: Profile of the velocity from a horizontal centerline from a 256 × 256 simulation of cavity flow with Re=1000 [18]

lattice-Boltzmann method is run simultaneously with the lattice-Boltzmann method the memory usage is double what is used when running the lattice-Boltzmann method alone.

There have been some methods developed to lessen the memory usage of the thermal lattice-Boltzmann method. Azwadi et al. implemented a two dimensional method with four velocities [4]. The idea of using four velocities for solving for the temperature is something that McNamara et al. hinted when explaining the passive scalar method [30].

Figure 3.6 shows the discrepancies that occur between the temperatures generated by the thermal lattice-Boltzmann method and the hybrid method when a larger thermal relaxation time is used. In the case shown in Figure 3.6 the Prandtl number was chosen to be slightly less than one and as a result of the constraints of Equation 3.18 the thermal relaxation time must be very large. Due to the large thermal relax-

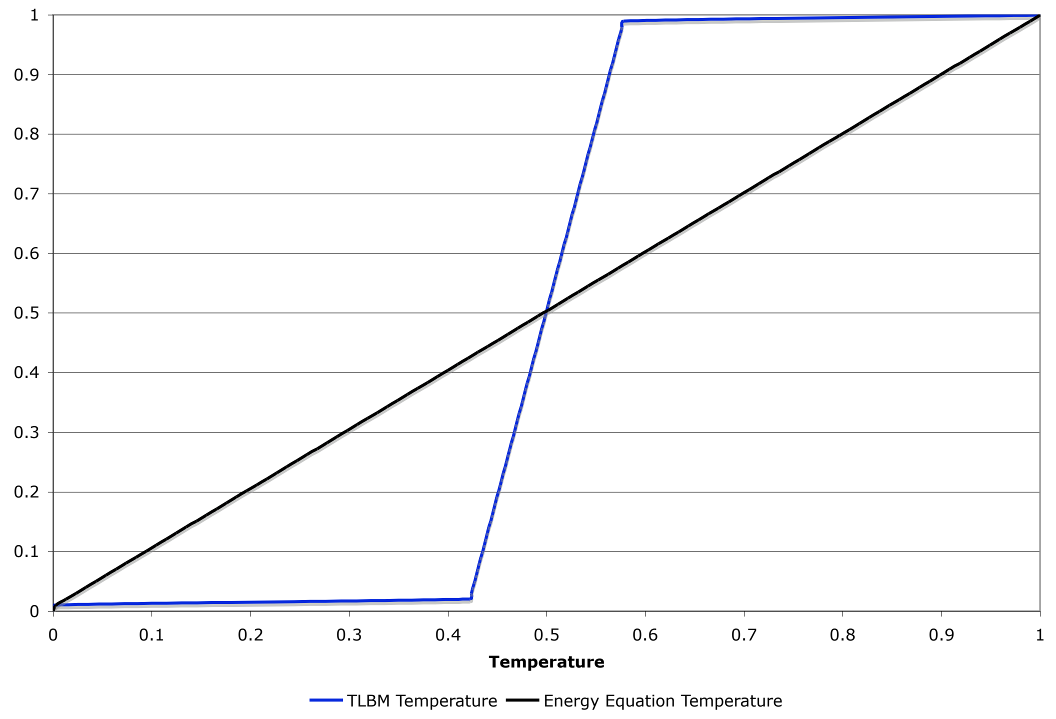

Figure 3.6: The normalized temperature profiles for Couette flow from both thermal lattice-Boltzmann method and the energy equation

ation time the collision operator in the thermal lattice-Boltzmann equation, the right hand side of Equation 3.10, becomes very small. The change in the thermal energy distribution is very small due to the small collision operator and the end result is that it takes much longer to reach a steady state.

The simulation of Couette Flow shown in Figure 3.6 is of an unrealistic fluid. To provide more practical results it was attempted to created a simulation where a Prandtl number is chosen and the thermal relaxation time is calculated based on the chosen Prandtl number. The calculation of the thermal relaxation time is done with the equation

$$\tau_T = \frac{1}{2} + \frac{2\tau_v - 1}{6 \times \mathbf{Pr}} \tag{3.31}$$

Equation 3.31 shows that the thermal lattice-Boltzmann method will be more accurate if a smaller Prandtl number is chosen. The thermal lattice-Boltzmann method will be most stable for a thermal relaxation time close to one. If the viscous relaxation time is close to one the most stable Prandtl number will be $\frac{1}{6}$.

There is a very large jump in temperature at the boundaries for both the simulations of a Reynolds number of 500 and 1000. The jump in temperature is due to the thermal relaxation time being less than one. The same problem was encountered with boundaries of the velocity simulations. There is also the difference in scale between the temperatures from the simulations of the thermal lattice-Boltzmann method and the energy equation. The difference in scale is due to either the jump in temperature at the boundaries or the two models do not the same fluid with the same thermal diffusivity.

From Figures 3.8 and 3.7 it is difficult to tell where the error is coming from. The boundaries must be correct to verify if the two models are modeling the same fluid.

Despite the errors in the boundary conditions in Figure 3.7 the temperatures in the middle section, between a height of $y = 0.2$ to $y = 0.8$ are very close between the two different models. The models produce different profiles though. The difference

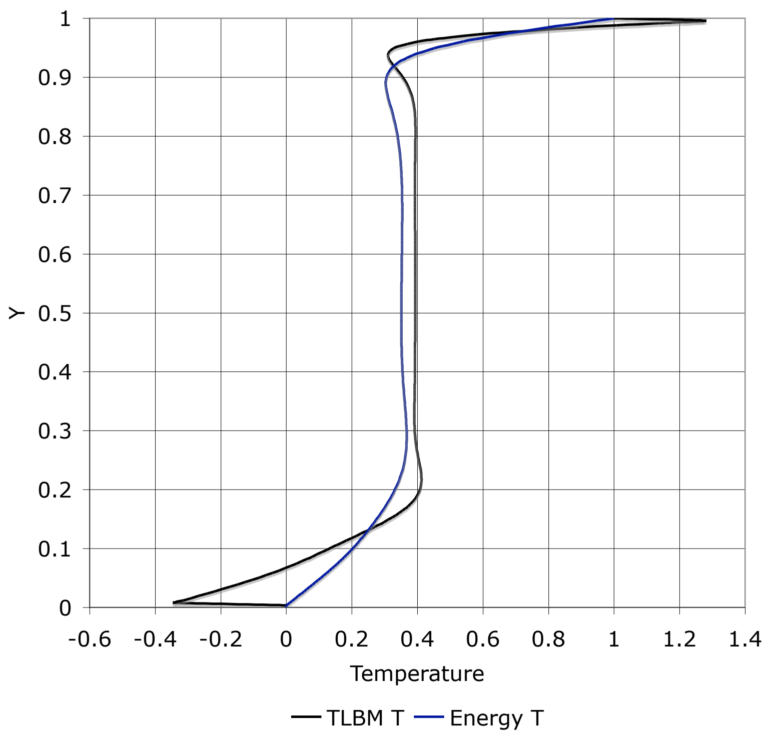

Figure 3.7: Profile of the temperature from a vertical centerline from a 256×256 simulation of cavity flow with Re=1000

in profile shapes implies that the differences between the two models was more likely due to different fluids being modeled. The same differences and implications of Figure 3.7 are also illustrated in Figure 3.8.

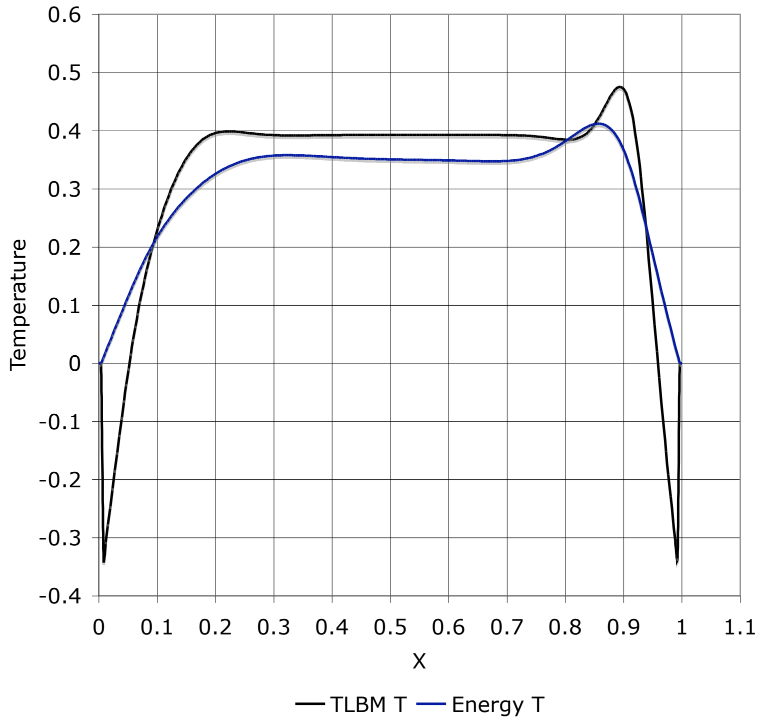

Figure 3.8: Profile of the temperature from a horizontal centerline from a 256×256 simulation of cavity flow with Re=1000

Shi et al.'s equations for the thermal lattice-Boltzmann method seems to produce a believable temperature field for the velocity field but it does [40]. The boundaries of the thermal lattice-Boltzmann method need to be corrected to properly compare the thermal lattice-Boltzmann and hybrid methods.

3.4 Modifications

3.4.1 Velocity

It has been shown that calculating the relaxation times from the lattice size, speed of the top plate, and Reynolds number does not provide accurate results. To meet the desired Reynolds numbers the relaxation times must either be close close to 0.5

or the lattice structure must be very large. The relaxation times close to 0.5 create a significant jump in values from the top row of the lattice structure to the next row.

In the case of a viscous relaxation time close to 0.5, the jump in velocity means that the simulation is no longer approximating the desired Reynolds number. The larger velocity indicates a similarly proportioned increase in the Reynolds number of the cavity flow. The thermal relaxation time close to 0.5 causes a jump in the temperature. The jump in temperature applies only to the thermal lattice-Boltzmann method. The small thermal relaxation time causes the thermal lattice-Boltzmann method to no longer model the same thermal flow as the hybrid method.

To correct the errors caused by relaxation times less than one it is necessary to adjust the boundary conditions for both the lattice-Boltzmann method and the thermal lattice-Boltzmann method. To correct the error in the velocity the top plate velocity is simply multiplied by the viscous relaxation. The recalculation of the top plate velocity must occur after the viscous relaxation time calculation. The viscous relaxation time is based on the lattice dimensions, the desired Reynolds number, and the velocity of the top plate. The top plate velocity will decrease but due to the jump in velocity just below the top plate the simulation will run as if the top plate is actually at the velocity it was set to prior to the adjustment. The boundary conditions for the hybrid method that implements the energy equation do not need to be changed.

The simulations have also shown a discrepancy in velocity at the bottom of the cavity in the cavity-flow simulations. The problem arrises from all of the boundaries being treated as periodic. The code was implemented with periodic boundary conditions to simulate Couette flow. While the bounce-back method does prevent the data from penetrating into the lattice structure, it does not prevent the skewing of results at the boundaries. To change the code and no longer have periodic boundary conditions it is only necessary to prevent propagation of probability distribution function,

f^{eq}, in the directions that point towards a cavity wall.

3.4.2 Temperature

The adjustment made to the top plate velocity does not work for the top plate temperature in the thermal lattice-Boltzmann simulation. The simplest way to ensure stable results from the thermal lattice-Boltzmann method is to ensure that the thermal relaxation time is nearly one. The Prandtl number is calculated based on the thermal relaxation time being set to 0.99 using Equation 3.18, reiterated here.

$$\mathbf{Pr} = \frac{\nu}{\alpha} = \frac{\frac{2\tau_v - 1}{6}}{\tau_T - \frac{1}{2}} \tag{3.32}$$

While more efforts would be needed to stabilize the thermal lattice-Boltzmann method, setting the thermal relaxation time close to one provides stable results for the comparison of the method to the hybrid method using the energy equation.

From the simulation of a Reynolds number equal to 100 with the parameters shown in Table 3 it was shown that for thermal relaxation times of one or greater the energy equation solution for temperature is very unstable. Using the same code for the other simulations of Table 3 results were obtained from to solution of the Energy equation. For the case of a Reynolds number of 100 there were no results.

To verify that the thermal lattice-Boltzmann method equations were working properly a simulation using Azwadi et al.'s equations and compared to the results of Shi et al.'s equations. The results shown in Figure 3.9 show that the two methods produce practically the same results as the lines of the profiles overlap.

The energy equation being used for the hybrid method is based on Khiabani et al.'s research and is stated in Equation 3.33.

$$\frac{\partial \theta}{\partial t} + \vec{u} \cdot \nabla \theta = \frac{1}{\mathbf{Re} \times \mathbf{Pr}} \nabla^2 \theta \tag{3.33}$$

It appears that thermal diffusivity may be too small for the thermal lattice-Boltzmann method and conduction heat transfer does not have enough influence over the temperature. The other possibility for the difference between the thermal lattice-Boltzmann

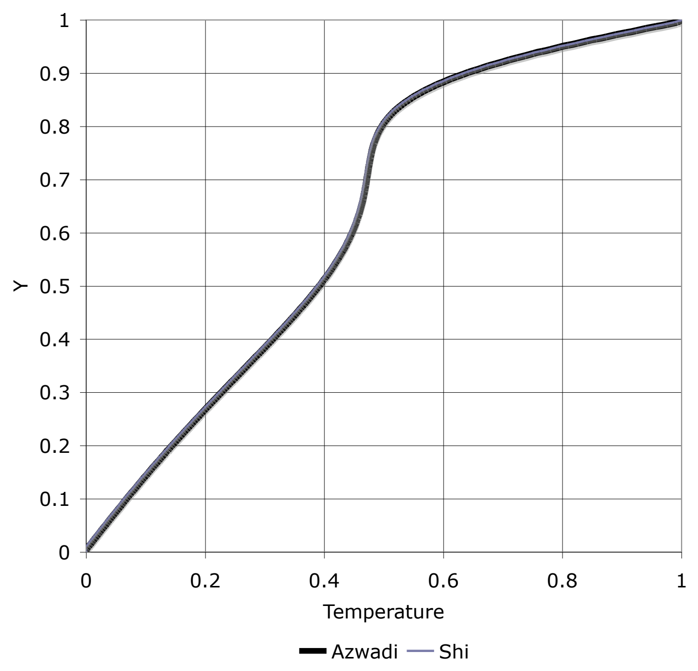

Figure 3.9: Comparison of the temperature profiles from the vertical centerlines of Shi et al.'s and Azwadi et al.'s thermal lattice-Boltzmann methods

and hybrid methods is that the velocity components of the thermal lattice-Boltzmann equations are not large enough to accurate temperatures based on the velocity field and fluid properties. Due to the fact that Azwadi et al. and Shi et al. both dropped higher order terms to simplify the thermal lattice-Boltzmann equations it is more likely that the differences are due to the velocity not having enough influence over the temperatures [4, 40].

CHAPTER IV

RESULTS

4.1 Velocity

Following the corrections to the boundary conditions and the adjustment of the top plate velocity based on the viscous relaxation time the simulations were run again. The results match other simulations much better and do not show the errors at the boundaries.

Figure 4.1 shows the vertical centerline of the velocity from the current simulation and the simulation from Ghia et al [18]. Both figures are of simulations of a Reynolds number of 100. The results from Figure 4.1 illustrate that the current model is getting the basic form of the flow correct. Another reason for possible differences is that the dimensions of the two simulations are different. The simulation from Ghia et al. have the dimensions of 129×129 while the simulation for the current study have the dimensions of 256×256.

Figure 4.3 shows the vertical centerline of the velocity from the current simulation and the simulation from Ghia et al [18]. Both figures are of simulations of a Reynolds number of 400.

Figure 4.6 is from the same simulation of a Reynolds number of 1000. Figure 4.6 again includes the results from the current study and Ghia et al.'s study. The two sets of results re-emphasize the fact the the velocity generated from the simulation is basically correct except for being scaled a little differently.

The resulting velocity profiles from the current simulation are nearly identical to the Ghia et al. simulations. Ghia et al.'s simulations are not devoid of errors. The most notable error is from Figure 4.4 where close to the right wall there is a

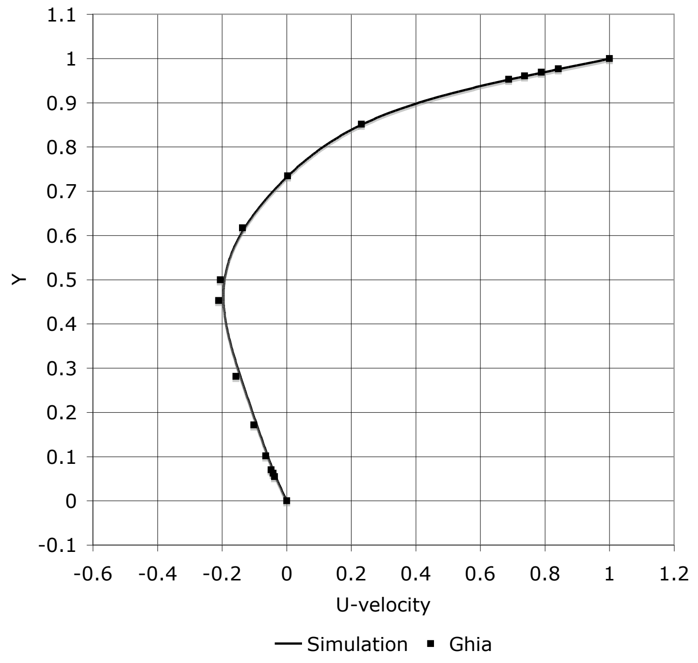

Figure 4.1: Profile of the velocity from a vertical centerline from a 256 × 256 simulation of cavity flow with Re=100 [18]

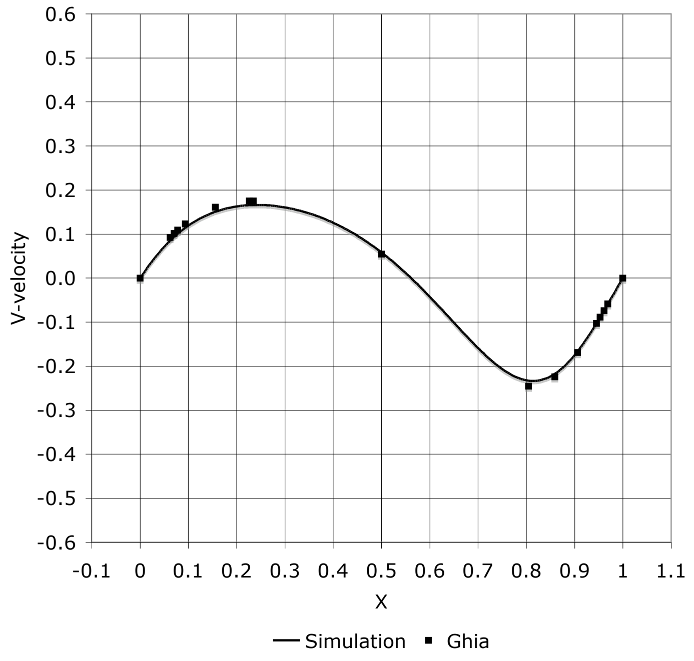

Figure 4.2: Profile of the velocity from a horizontal centerline from a 256 × 256 simulation of cavity flow with Re=100 [18]

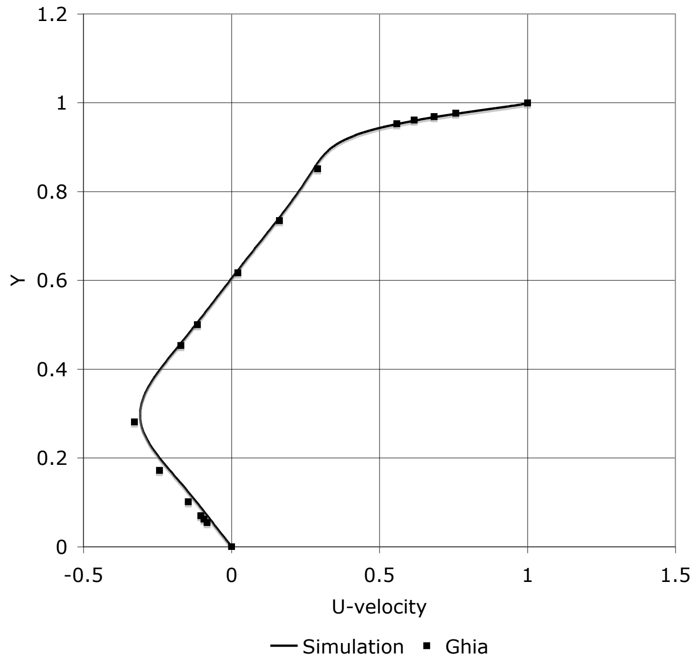

Figure 4.3: Profile of the velocity from a vertical centerline from a 256 × 256 simulation of cavity flow with Re= 400 [18]

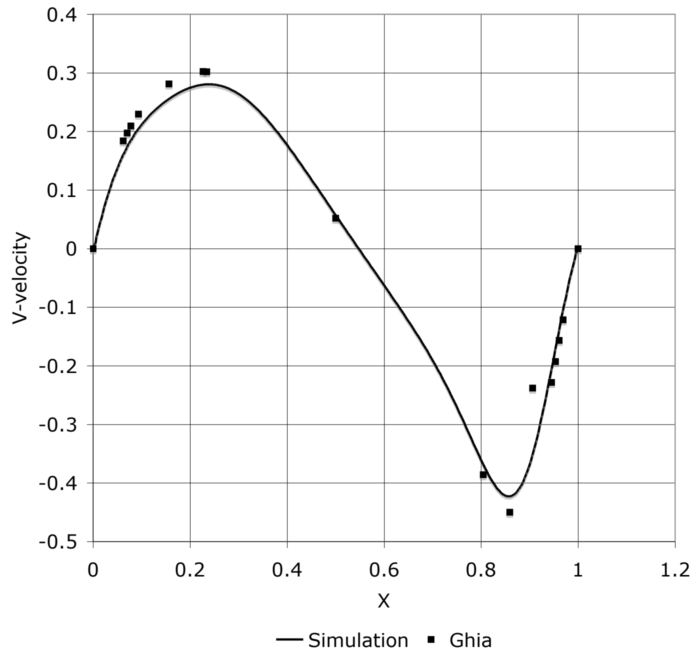

Figure 4.4: Profile of the velocity from a horizontal centerline from a 256 × 256 simulation of cavity flow with Re= 400 [18]

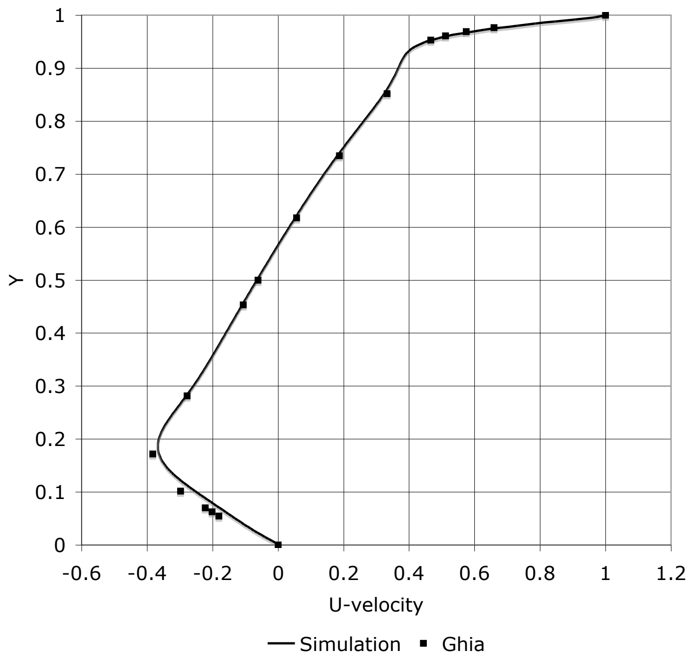

Figure 4.5: Profile of the velocity from a vertical centerline from a 256 × 256 simulation of cavity flow with Re=1000 [18]

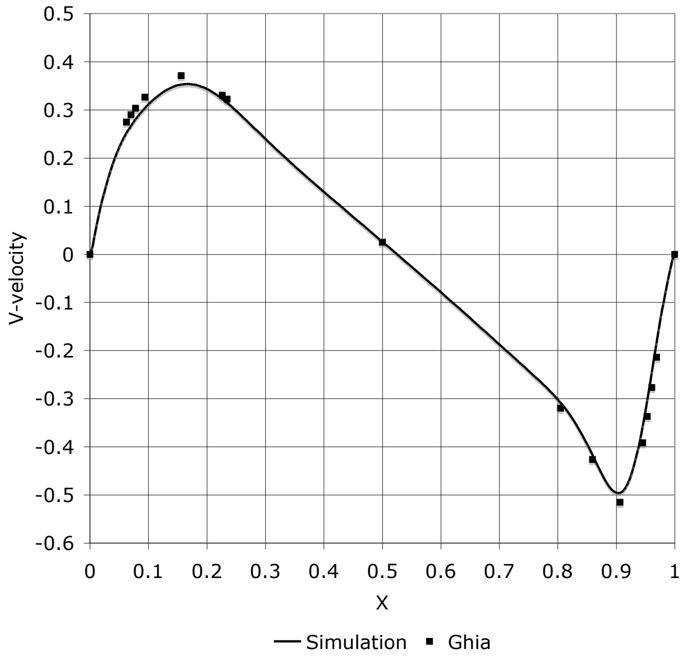

Figure 4.6: Profile of the velocity from a horizontal centerline from a 256 × 256 simulation of cavity flow with Re=1000 [18]

discrepancy between two of the points.

The results of Ghia et al.'s simulation are very similar the results of the current simulation for all three Reynolds numbers, 100, 400, and 1000. The very slight differences between the results of Ghia et al.'s simulation and the current simulation could be from the different grid sizes. Ghia et al.'s simulations were with a grid size of 129×129. Another possible explanation for the differences is that the velocity of the top plate used for the Bhatnagar-Gross-Krook lattice-Boltzmann method, 0.05, is too large. The velocities converged in the simulation but the large velocity may have caused some unknown problems.

It is easy to see the similarities between the two flows when comparing the stream functions of the flow. The center of the vortex created by the moving top plate is a good indication that the flow is correct for the given Reynolds number.

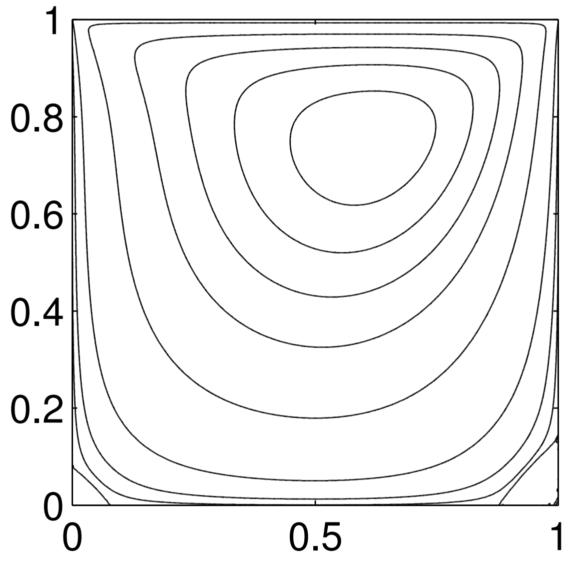

Figure 4.7: Stream function from Lü et al. simulation of cavity flow with dimensions of 257×297 and a Re=100 [27]

Figures 4.7 and 4.8 have the same center for the vortex. The flow pattern of the two simulations is very similar. One of the problems visible in 4.8 is the stream function ending at the wall. The possible problems that could cause such errors would

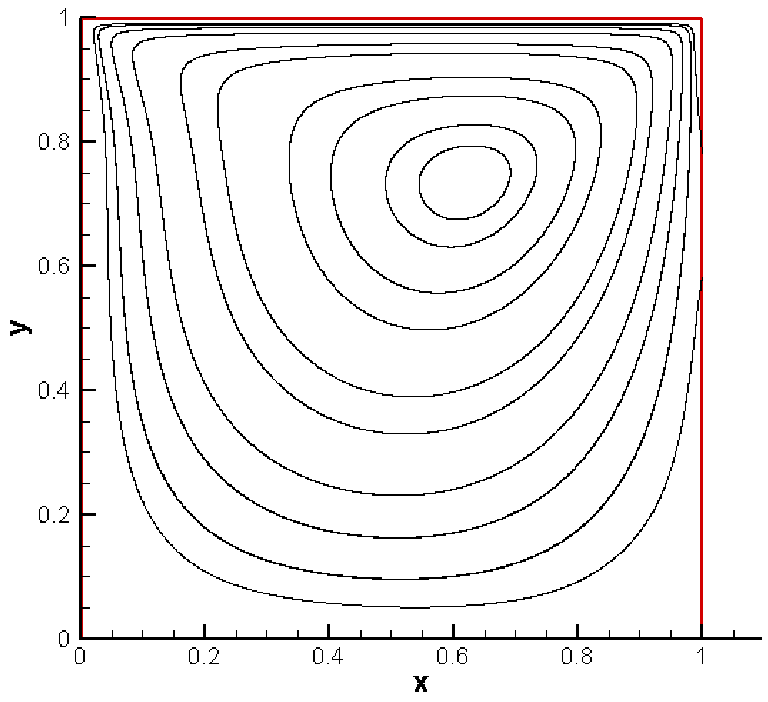

Figure 4.8: Stream function from the simulation of cavity flow with dimensions of 256×256 and a Re=100

be that the boundaries do not meet the no-slip boundary conditions or a large grid near the boundary. The problem is not the with the boundary conditions. At the boundaries of the simulation the bounce-back method is applied. The bounce-back method automatically applies the no-slip condition.

To improve the results it is possible to decrease the distance between the nodes in the lattice structure. The simulation that was performed was done with a constant distance between the nodes of the lattice structure.

Figures 4.9 and 4.10 also agree very well. The simulations illustrated in the two figures have a Reynolds number of 400. The center of the vortex is in nearly the same location in Figure 4.10 as in Figure 4.9. Also, the counter-current flows in the bottom two corners are about the same size between the two figures.

Figures 4.11 and 4.12 look very similar. The center of the two vortices are in about the same location and the counter-current flow in the bottom corners are of

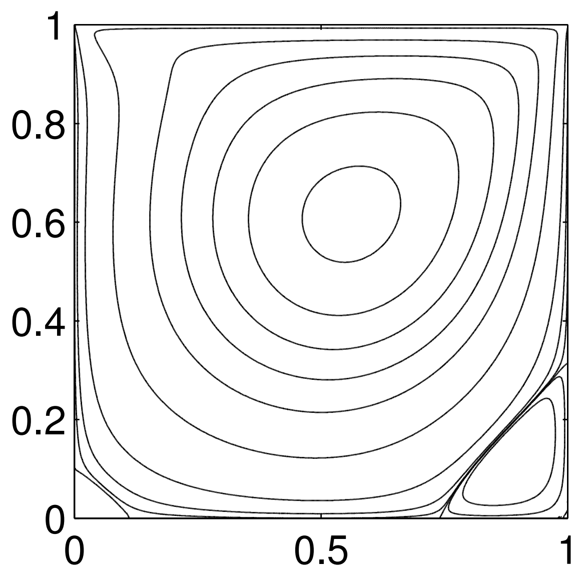

Figure 4.9: Stream function from Lü et al. simulation of cavity flow with dimensions of 257 × 297 and a Re=400 [27]

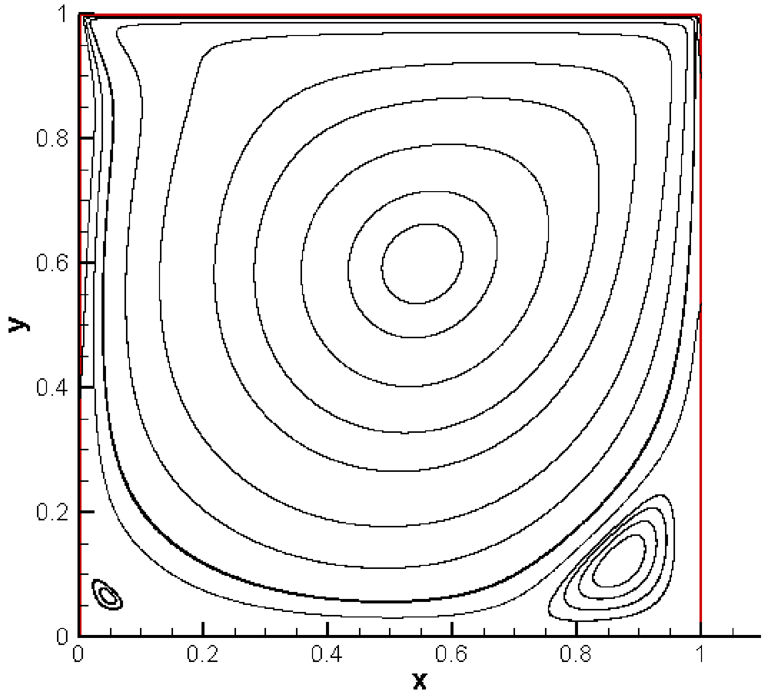

Figure 4.10: Stream function from the simulation of cavity flow with dimensions of 256 × 256 and a Re=400

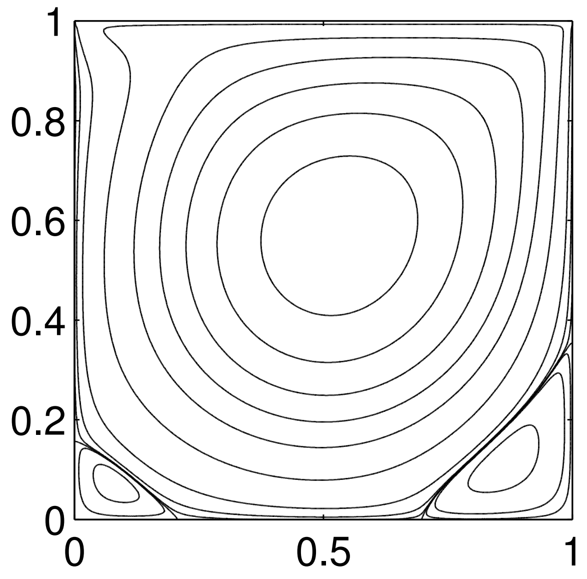

Figure 4.11: Stream function from Lü et al. simulation of cavity flow with dimensions of 257×297 and a Re=1000 [27]

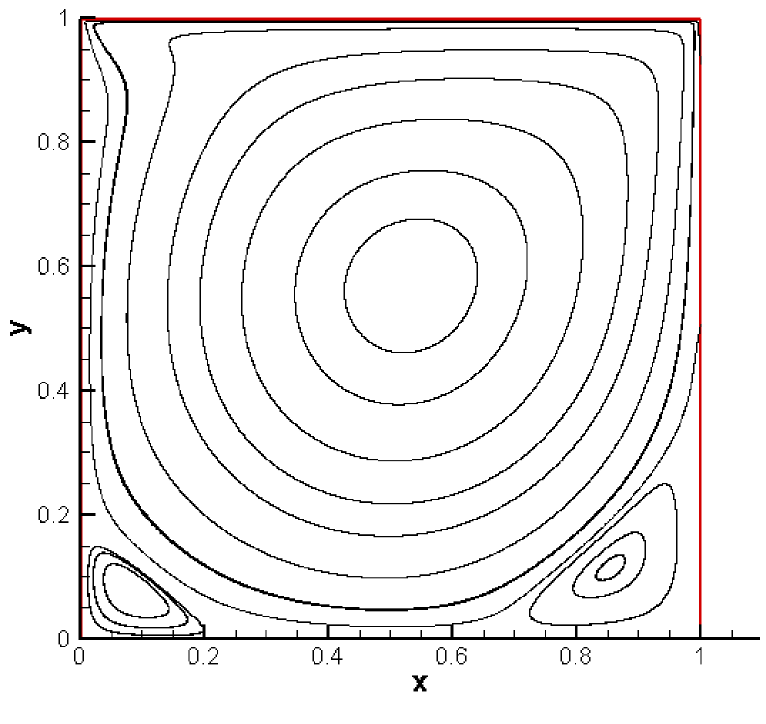

Figure 4.12: Stream function from the simulation of cavity flow with dimensions of 256×256 and a Re=1000

comparable size. The flow is also starting to pull away from the wall near the upper left corner in both figures. For even larger Reynolds numbers some counter-current flow will form there as well.

The changes to the velocity simulation appear to agree with other simulations of the same type of flow. It is important to have verified the velocity field of the simulation because both of the methods for calculating temperature are effected by the velocity. If the velocity field is incorrect then both of the temperature fields would be incorrect.

Verifying the velocity field also verifies the methodology used for simulating the lattice-Boltzmann method. If the velocity field is correct then the methodology for the lattice-Boltzmann method used in the code is most likely correct. Verifying the methodology for the lattice-Boltzmann method also verifies the methodology for the thermal lattice-Boltzmann method. The two methods are performed the exact same way. The differences between the lattice-Boltzmann method and the thermal lattice-Boltzmann method are the equations used to calculate the equilibrium distribution function and the relation to convert the distributions to the values of density, velocity, and temperature.

4.2 Temperature

Following the correction to the boundaries of the thermal lattice-Boltzmann method the resulting temperature profiles are very similar. The three Reynolds numbers that were simulated all produced results showing the hybrid method utilizing the energy equation calculating nearly the same temperature profile as the thermal lattice-Boltzmann method.

Figures 4.13 and 4.14 show the temperatures generated by the hybrid method and the thermal lattice-Boltzmann method of the horizontal centerline and vertical centerline respectively. The simulation with a Reynolds number of 100 also has the largest Prandtl number of the three Reynolds numbers used.

Table 4: Prandtl Numbers for Temperature Simulations

$Reynolds Number$	$Prandtl Number$
100	0.26
500	0.052
1000	0.026

The Prandtl number for the Reynolds number of 100 is 0.26. The Prandtl numbers for the three simulations of Reynolds number of 100, 500, and 1000 are presented in Table 4.

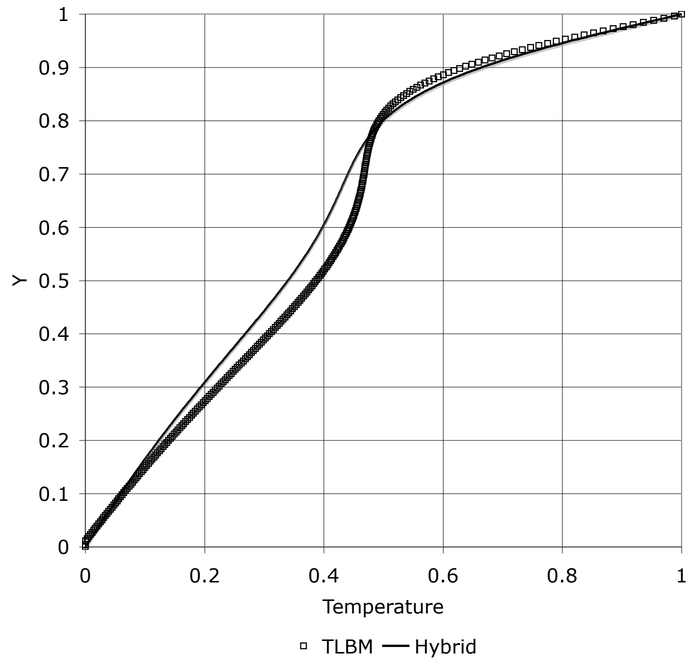

Figure 4.13: Profile of the temperature from a vertical centerline from a 256 × 256 simulation of cavity flow with Re=100

Figures 4.15 and 4.16 show the temperatures generated by the hybrid method and the thermal lattice-Boltzmann method of the horizontal centerline and vertical centerline respectively. The Prandtl number for the simulation with a Reynolds number of 500 is 0.052.

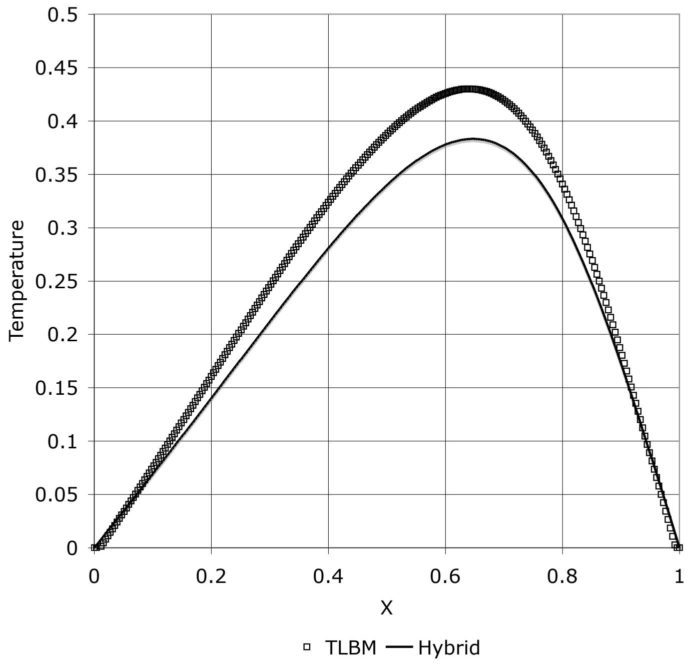

Figure 4.14: Profile of the temperature from a horizontal centerline from a 256×256 simulation of cavity flow with Re=100

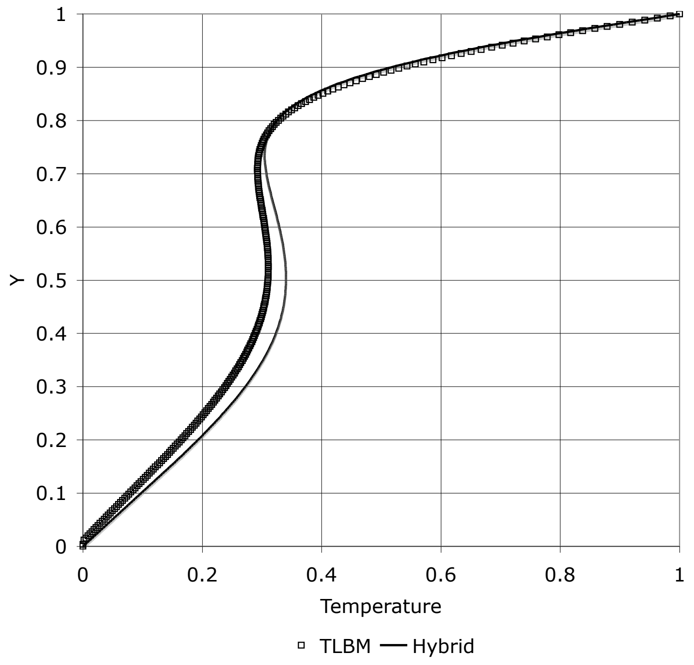

Figure 4.15: Profile of the temperature from a vertical centerline from a 256×256 simulation of cavity flow with Re=500

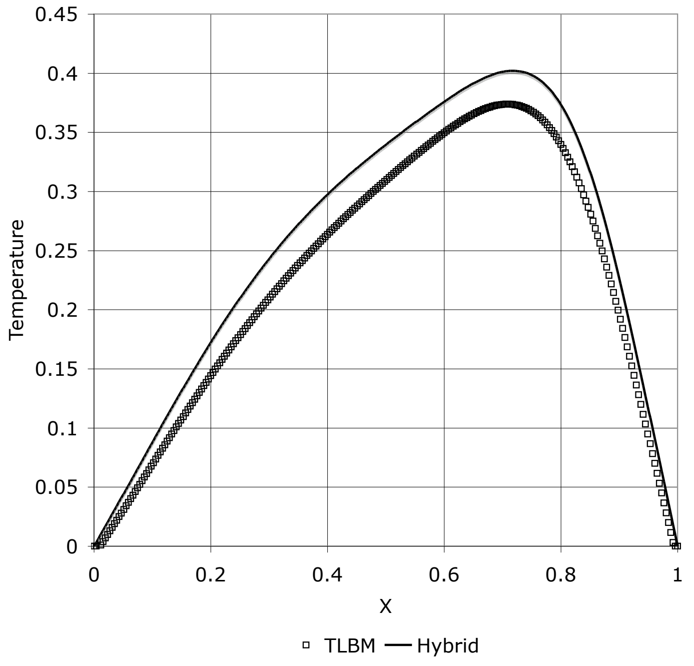

Figure 4.16: Profile of the temperature from a horizontal centerline from a 256×256 simulation of cavity flow with Re=500

Figures 4.17 and 4.18 show the temperatures from the two methods overlapping. The Prandtl number for the simulation with a Reynolds number of 1000 is 0.026.

The fact that both produce reliable results means that the faster, less computationally intensive energy equation is better suited for the job of calculating a temperature field. In both the Reynolds number simulations of 500 and 1000 the hybrid method produces results where the temperature appears more effected by the temperature.

In the equations from Shi et at. for the thermal equilibrium distribution function the higher order terms are dropped to simplify the equations [40]. It is possible that the dropped terms are important to preserve accuracy of the values in the thermal lattice-Boltzmann method. The other possibility for the difference between the thermal lattice-Boltzmann and hybrid methods is the top plate velocity is too large for the Bhatnagar-Gross-Krook lattice-Boltzmann simulation. The thermal lattice-Boltzmann equations used were derived based on the Bhatnagar-Gross-Krook lattice-Boltzmann equations and suffer from the same need for a small reference velocity.

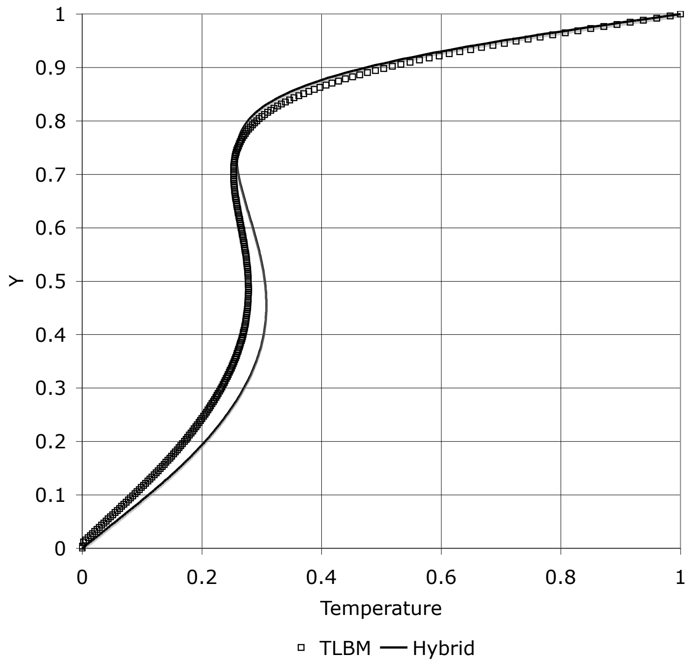

Figure 4.17: Profile of the temperature from a vertical centerline from a 256×256 simulation of cavity flow with Re=1000

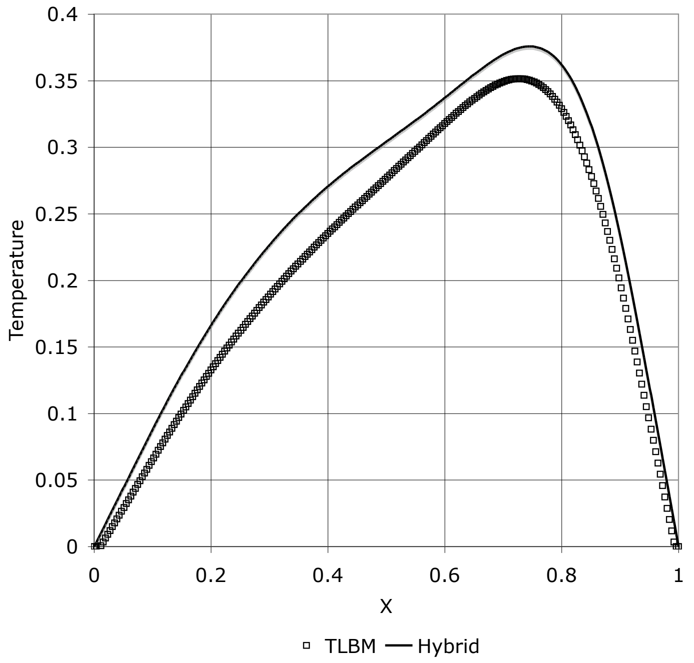

Figure 4.18: Profile of the temperature from a horizontal centerline from a 256×256 simulation of cavity flow with Re=1000

4.2.1 Decreased Velocity to Improve Thermal Lattice-Boltzmann Accuracy

In an attempt to improve the accuracy of the thermal lattice-Boltzmann method the top plate velocities were decreased. The Bhatnagar-Gross-Krook method is most stable for velocity values much less than the speed of sound, $\frac{1}{\sqrt{3}}$. While the simulations with a top plate velocity of 0.05 did converge the velocity may be a little high. The simulations were run again with the lattice size again being 256×256 and the velocity varying between 0.05, 0.01, and 0.005. For all three of the simulations the Reynolds number was maintained at 500. The resulting Prandtl numbers for the three simulations are shown in Table 5.

Table 5: Prandtl Numbers as a result of changing the top plate velocity

$TopPlateVelocity$	$PrandtlNumber$
0.05	0.052
0.01	0.010
0.005	0.0052

Figures 4.15 and 4.16 show the temperature results from a simulation for both the hybrid method and the thermal lattice-Boltzmann method with a top plate velocity of 0.05. These are the results of the previous simulation and the reference point to determine if the lower velocity improves the accuracy of the thermal lattice-Boltzmann method.

Table 5 shows that the simulation with a top plate velocity decreased by a factor of five also decreases the Prandtl number by a factor of five. As a result of the drop in the Prandtl number the thermal centerlines in Figures 4.19 and 4.20 do not exhibit the same shape as the simulation if Figures 4.15 and 4.16.

Despite the difference in shape between the two simulations there is a definite improvement in accuracy with the decrease in velocity. There are still differences

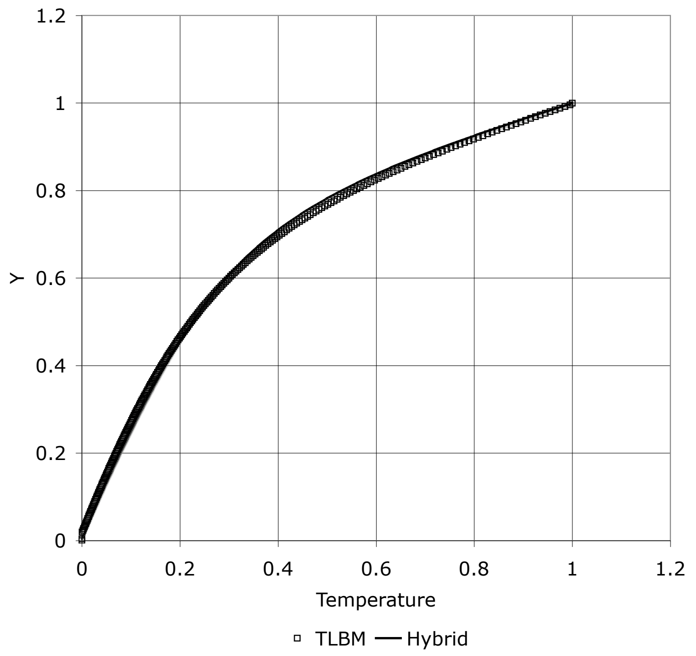

Figure 4.19: Profile of the temperature from a vertical centerline from a 256×256 simulation of cavity flow with Re=500 and a top plate velocity of 0.01

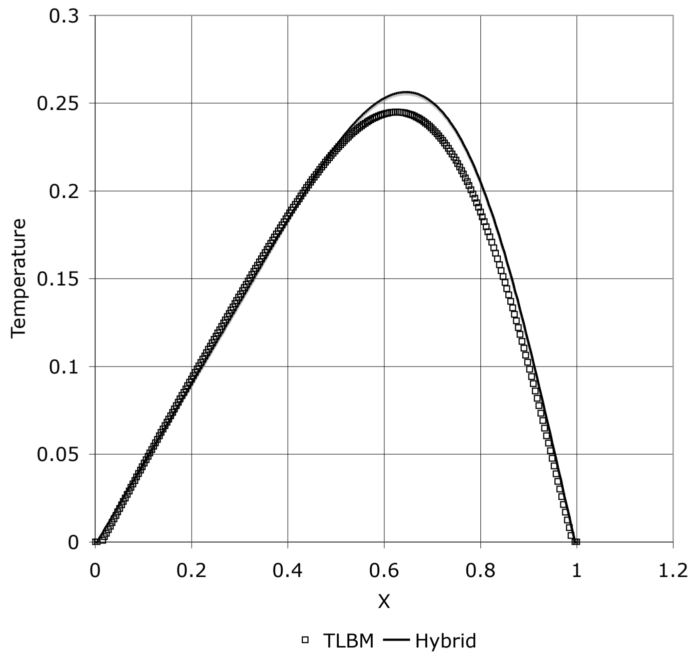

Figure 4.20: Profile of the temperature from a horizontal centerline from a 256×256 simulation of cavity flow with Re=500 and a top plate velocity of 0.01

between the temperatures generated by the hybrid method and the thermal lattice-Boltzmann method but a large portion of the respective centerlines overlap. In the simulation with a top plate velocity of 0.05 the two temperature profiles had a very similar shape but there was usually a small difference between the temperatures of the hybrid method and the temperatures of the thermal lattice-Boltzmann method.

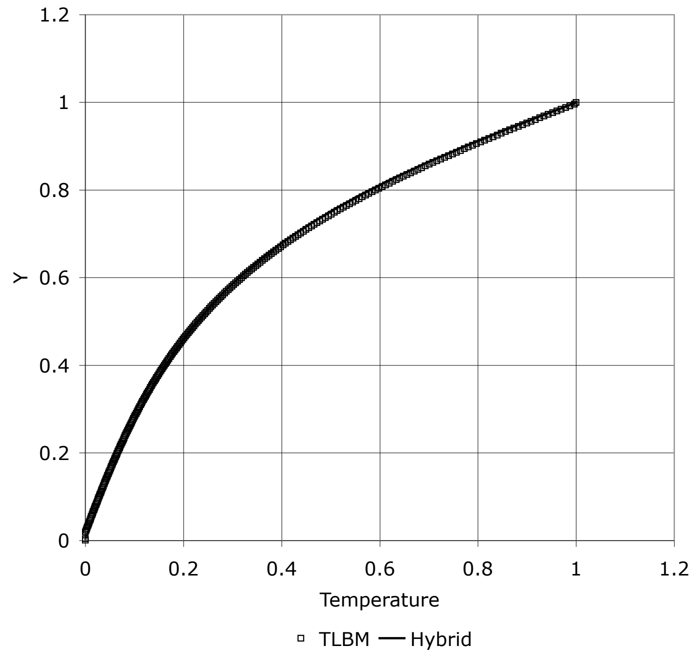

Figure 4.21: Profile of the temperature from a vertical centerline from a 256 $\times$ 256 simulation of cavity flow with Re=500 and a top plate velocity of 0.005

The velocity was decreased further to determine if the accuracy of the thermal lattice-Boltzmann method would improve further. The results of a simulation with a top plate velocity of 0.005 are shown in Figures 4.21 and 4.22. The main improvement clearly illustrated between Figures 4.19 and 4.20 and Figures 4.21 and 4.22 is the maximum temperature in the horizontal centerline. It is easy to see that between Figure 4.20 and Figure 4.22 the difference between the hybrid method temperatures and the thermal lattice-Boltzmann temperatures decreases.

As the top plate velocities decrease the influence of velocity on the thermal flow also decreases. Not only will the change in temperatures be lower due to the velocity

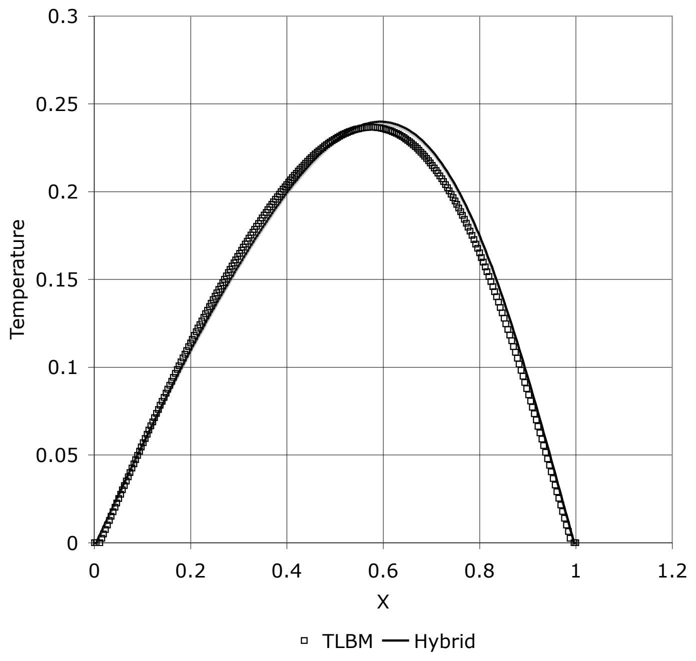

Figure 4.22: Profile of the temperature from a horizontal centerline from a 256×256 simulation of cavity flow with Re=500 and a top plate velocity of 0.005

vector $\vec{u}$ of the energy equation, Equation 3.21, but the resulting lower Prandtl number also means that the thermal diffusion dominates the transfer of heat. Heat conduction is the most dominate form of heat transfer in the simulations with lower top plate velocities and very small Prandtl numbers.

The fact that the decrease in top plate velocity increased the accuracy of the simulations reinforces the idea that the dropped higher order terms in the Shi et al. and the Azwadi et al. simulations are important to the accuracy of the generated temperatures [4, 40].

4.2.2 Increased Order to Improve Thermal Lattice-Boltzmann Accuracy

To test the hypothesis further that the higher orders dropped from the Shi et al. and the Azwadi et al. simulations are important to the accuracy of the generated temperatures the higher order values from the Shi et al. thermal lattice-Boltzmann equations were used to run three more simulations [4, 40]. The new form of Shi et al.'s equation is Equation 4.1.

$$g_i^{eq}(\mathbf{x},t) = \begin{cases} \frac{4}{9}\rho T(\mathbf{x})\left[1-\frac{3}{2}\mathbf{u}^2\right] & i=0 \\ \frac{1}{9}\rho T(\mathbf{x})\left[1+3(\mathbf{e}_i\cdot\mathbf{u})+\frac{9}{2}(\mathbf{e}_i\cdot\mathbf{u})^2-\frac{3}{2}\mathbf{u}^2\right] & i=1,3,5,7 \\ \frac{1}{36}\rho T(\mathbf{x})\left[1+3(\mathbf{e}_i\cdot\mathbf{u})+\frac{9}{2}(\mathbf{e}_i\cdot\mathbf{u})^2-\frac{3}{2}\mathbf{u}^2\right] & i=2,4,6,8 \end{cases} \tag{4.1}$$

Using Equation 4.1 instead of 3.13 simulations were run for Reynolds numbers of 100, 500, and 1000. In order to generate results quickly the lattice size was decreased from 256×256 to 100×100. The resulting Prandtl numbers due to the change in lattice size are presented in Table 6.

Table 6: Prandtl Numbers as a result of a grid size of 100×100

$ReynoldsNumber$	$PrandtlNumber$
100	0.10
500	0.020
1000	0.010

The results of the simulations for Reynolds numbers of 500 and 1000 are presented because they exhibit results that are comparable to other studies. The simulation for a Reynolds number of 500 has the same Reynolds number and a Prandtl number about twice as large as the simulation presented in Figures 4.19 and 4.20. The simulation for a Reynolds number of 1000 has a different Reynolds number but a nearly identical Prandtl number as the simulation presented in Figures 4.19 and 4.20.

As a result of the drop in the Prandtl number the thermal centerlines in Figures 4.23 and 4.24 do not exhibit the same shape as the simulation in Figures 4.15 and 4.16. The results from the new thermal equilibrium probability distribution function of Equation 4.1 are not as accurate at the results from the lower velocity, shown if Figures 4.19 and 4.20, but it does appear more accurate than the earlier results shown in Figures 4.15 and 4.16.

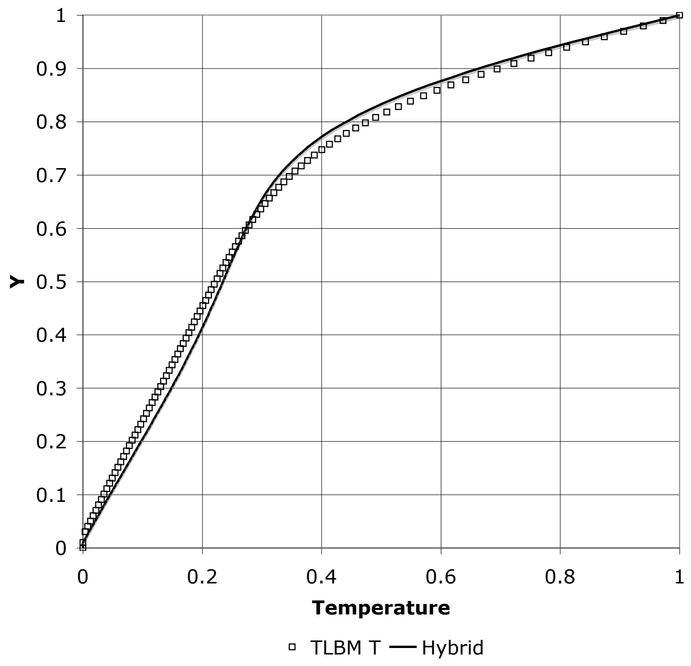

Figure 4.23: Profile of the temperature from a vertical centerline from a 100 × 100 simulation of cavity flow with Re=500

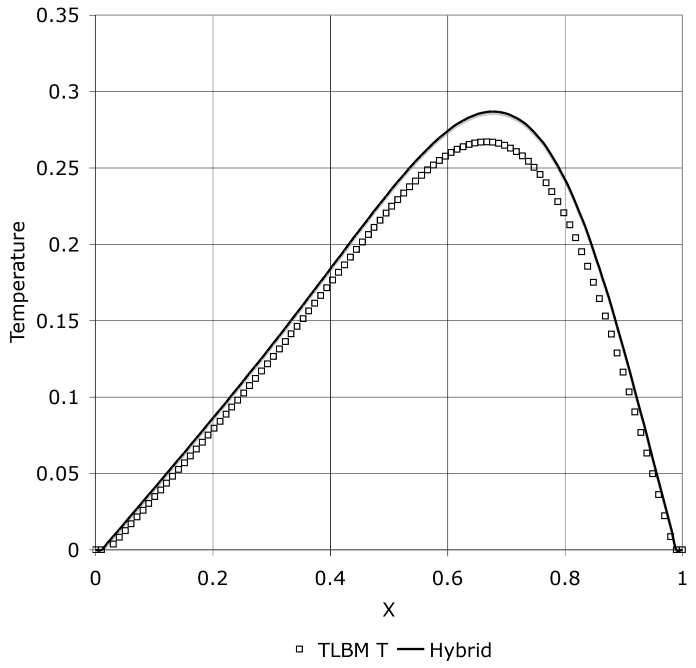

Figure 4.24: Profile of the temperature from a horizontal centerline from a 100×100 simulation of cavity flow with Re=500

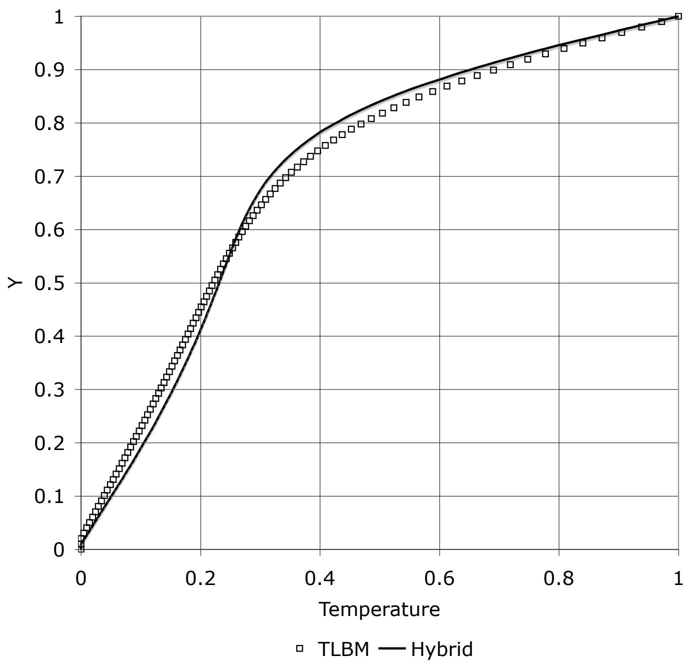

Figure 4.25: Profile of the temperature from a vertical centerline from a 100×100 simulation of cavity flow with Re=1000

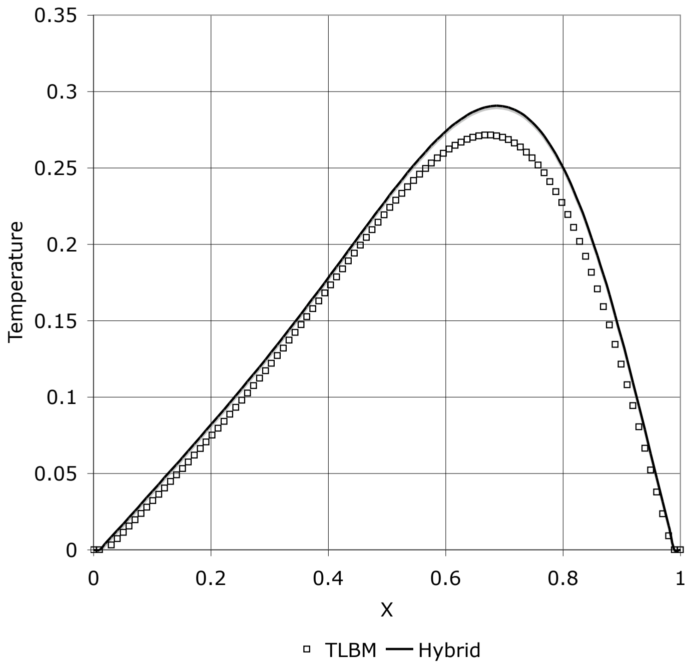

Figure 4.26: Profile of the temperature from a horizontal centerline from a 100×100 simulation of cavity flow with Re=1000

The change in the Prandtl number between the simulations with Reynolds numbers of 500 and 1000 is only a factor of two. The resulting temperature centerlines are nearly identical. Again, the results show that the lower velocity improved results more than the addition of higher order terms.

There are improvements to accuracy with the addition of higher order terms to the thermal equilibrium distribution function but it is inconclusive as to whether the improvements are due to the decrease in the Pradtl number or the addition of the higher order terms. As stated before, the lower Prandtl number lessens the velocity's effect and increases the effect of conduction heat transfer.

4.3 Comparison of thermal lattice-Boltzmann method and the Hybrid Method

An advantage of the lattice-Boltzmann method is that the code can be performed on parallel processors. It is possible to use parallel processors because the time evolution operator is very local in nature. The computational time is independent of the Reynolds number but very heavily dependent on the size of the lattice structure. As the Reynolds number increase either the lattice structure must also increase in size or the computations will become more unstable as the viscous relaxation time approaches 0.5.

For the purposes of the study presented the benefit of the computational time being independent of Reynolds number is negated by the desire to study larger Reynolds numbers of 500 and 1000. There is a balance between stability and computational time for the study of lid-driven cavity flow. To obtain results in a timely fashion accuracy has been compromised.

The ability to use multiple processors also applies to the thermal lattice-Boltzmann method. The two methods, the lattice-Boltzmann method and the thermal lattice-Boltzmann method, are performed the exact same way.

There have also been many studies about implementing the thermal lattice-Boltzmann method with different thermal effects. Shi et al. and Lü et al. have both performed studies on implementing viscous dissipation directly with the thermal lattice-Boltzmann model [40, 27]. There have also been proposed methods for using the thermal lattice-Boltzmann method to model turbulence, micro-flow, two phase flow, and convection.

Earlier in the passive-scalar approach to solving thermal flows it was not possible to simulate thermal effects with the model. Since the development of the passive-scalar model there have been developments that allow most thermal effects to be modeled through modification of the scalar energy transport equation. Zhang et al. implemented a scalar energy transport equation that included a variable heat conductivity for two-phase flow and a term to include viscous dissipation and surface tension.

A huge benefit of the use of the energy function over the thermal lattice-Boltzmann method is that it is much less intensive computationally. In two dimensions the energy equation takes approximately $\frac{1}{9}^{th}$ the amount of computational power as the D2Q9 lattice-Boltzmann model. This allows the temperatures to be calculated much more quickly.

As mentioned before, methods have been developed to lessen the computational power needed to run simulations of the thermal lattice-Boltzmann method. Azwadi et al. developed a thermal lattice-Boltzmann method that uses four velocities instead of nine [4]. The results of the four velocity thermal lattice-Boltzmann method were shown by Figure 3.9 to produce the same results as the nine velocity thermal lattice-Boltzmann method developed by Shi et al. [40].

4.3.1 Convergence Times

To test which method was better the code was set to solve the lattice-Boltzmann method to completion, the thermal lattice-Boltzmann method to completion, and

Table 7: Number of time-steps needed for convergence of the three simulated methods

$Reynolds Number$	100	500	1000
$Lattice - Boltzmann Method$	66371	170611	418009
$Thermal Lattice - Boltzmann Method$	83242	107436	116030
$Hybrid Method$	6840	32084	39592

finally the energy equation to completion. Solving one of the methods to completion is based on a convergence criteria. The change in the velocity or temperature is divided by the total velocity or temperature and compared to the convergence criteria. Once the convergence criteria is met for one method the program will begin to process the next method.

Table 7 shows that the energy equation solves the temperature distribution much faster than the thermal lattice-Boltzmann equation. The energy equation takes fewer time-steps to process and is less memory intensive. Being less memory intensive means that each time-step also processes much faster. The hybrid method is thus a great deal faster at solving for the temperature field.

The hybrid method converges about ten times faster than the thermal lattice-Boltzmann method of the Reynolds number of 100. The hybrid method of the low Reynolds number flow probably converges faster relative to the thermal lattice-Boltzmann method than the larger Reynolds numbers due to less change in the temperatures occurring.

One factor that probably contributes to the hybrid method converging more quickly than the thermal lattice-Boltzmann method is that the hybrid method is changed based on the difference between the temperatures while the thermal lattice-Boltzmann method is based on a maximum of $\frac{5}{9}$ of the probability distribution being replaced. The hybrid method temperatures are going to be effected more when there is a large difference than the same difference using the thermal lattice-Boltzmann method. The combination of less overall change in temperature and the ability of the hybrid method to converge more quickly make the hybrid much more effective at simulating temperatures for low Reynolds number simulations.

CHAPTER V

CONCLUSION

A study was performed to determine whether the hybrid method of calculating the temperature for a flow using the energy equation is better than the available thermal lattice-Boltzmann methods. By knowing the available options and the pros and cons of each option, and educated decision could be made about what method would be best to use in simulations.

First, research was done on the available options for calculating temperatures of a flow field. The flow field is being generated by the lattice-Boltzmann method. An overview of the lattice-Boltzmann method was presented followed by overviews of the various thermal lattice-Boltzmann methods.

After an overview of the theory of the lattice-Boltzmann and thermal lattice-Boltzmann methods the lid-driven cavity flow is explained. Lid-driven cavity flow was the flow model used to compare and contrast the different methods for calculating the temperature fields. Finally the theory behind the hybrid method using the energy equation is explained.

The results of the lid-driven cavity flow simulations show that the thermal lattice-Boltzmann method and the hybrid method of calculating thermal flows produce similar results. The hybrid method is less computationally intensive and more practical for simulation purposes. The study also illustrates that the hybrid method is more stable. Based on the constraints needed to ensure the stability of the thermal lattice-Boltzmann method the hybrid method is better suited for modeling a broad range of fluids.

In the future it would be useful to compare the two methods as various thermal effects such as viscous dissipation are included in the simulations. Also, in situations where either time or computational power is not a constraint, larger lattice structures could be used to provide more detailed results that could illustrate more subtle differences in the results of the two methods. The new form of Shi et al.'s equation, Equation 4.1, should also be run for a lattice size of 256×256. The larger lattice size is the same as some of the simulations run the current study and would solidify conclusions that the higher order terms are important to accuracy for the thermal lattice-Boltzmann method.

APPENDIX A

CODE FOR THE LATTICE-BOLTZMANN METHOD

The code used to simulate the flows and temperatures is written in C#. The code compiles to an executable Windows file with a very convenient user interface. From the interface it is possible to edit the minimum number of time-steps taken, the interval between the code printing the data, the dimensions of the lattice structure, the velocity of the top plate, the initial temperature of the fluid, the temperature of the top plate, the temperature of the cavity, the Reynolds number of the flow, the Prandtl number of the fluid, the density of the fluid, and where the flow is cavity of Couette.

A.1 Initialization

The code begins by pulling parameters from the user interface. The parameters are:

1. The number of lattice points in the x and y directions
2. The number of time-steps between each printing operation
3. The minimum number of time-steps the operation will run before finishing
4. The Reynolds number for the flow
5. The cavity's top plate velocity
6. The cavity's top plate temperature
7. The cavity's wall temperature
8. The initial temperature of the fluid

9. The initial fluid density

The thermal relaxation time is specified and then the viscous relaxation time and the Prandtl number are calculated from 3.30 and 3.18, respectively.

Matrices are set up for holding the thermal lattice-Boltzmann based temperatures, the energy based temperatures, the velocities, the densities, three distribution functions for each the lattice-Boltzmann method and the thermal lattice-Boltzmann method, and boolean matrix to determine which lattice points are solid. The distribution functions are filled with the respective equilibrium distribution equations, Equations 3.12 and 3.13. The velocity matrix is filled with null and the temperature matrices are filled with the initial temperature. The boolean matrix, "SOLID," is filled with the desired ones and zeros to construct a cavity.

```
public void InitalizeMatricies()
        {
            TauViscous = 0.5 + 3.0 * TopPlateVelocity * (double)(YDirLength - 1) / ReynoldsNumber;
            while (TauViscous < 0.501)
            {
                YDirLength += 50;
                XDirLength += 50;
                TauViscous = 0.5 + 3.0 * TopPlateVelocity * (double)(YDirLength - 1) / ReynoldsNumber;
            }
            TauThermal = 0.99;
            PrandtlNumber = (2 * TauViscous - 1) / (2*TauThermal - 1);
            TopPlateVelocity = TopPlateVelocity * TauViscous;
            TLBMTopPlateTemp = 1.0;
            for (int i = 0; i < XDirLength; i++)
            {
                for (int j = 0; j < YDirLength; j++)
                {
                    double v_x, v_y, Temp, vsq, v0, v1, v2, v3, v4, v5, v6, v7, v8;
                    Temp = InitialTemp;
                    v_x = 0.0;
                    v_y = 0.0;
                    vsq = v_x * v_x + v_y * v_y;
                    v0 = v_x + v_y;
                    v1 = v_x;
```

```
v2 = v_x + v_y;
v3 = v_y;
v4 = -v_x + v_y;
v5 = -v1;
v6 = -v2;
v7 = -v3;
v8 = -v4;
SuperMatrix[i, j].particleProbability[0] = 4.0 * Rho * (1.0 - 1.5 * vsq) / 9.0;
SuperMatrix[i, j].particleProbability[1] = Rho * (1 + 3.0 * v1 +
4.5 * v1 * v1 - 1.5 * vsq) / 9.0;
SuperMatrix[i, j].particleProbability[2] = Rho * (1.0 + 3.0 * v2 +
4.5 * v2 * v2 - 1.5 * vsq) / 36.0;
SuperMatrix[i, j].particleProbability[3] = Rho * (1.0 + 3.0 * v3 +
4.5 * v3 * v3 - 1.5 * vsq) / 9.0;
SuperMatrix[i, j].particleProbability[4] = Rho * (1.0 + 3.0 * v4 +
4.5 * v4 * v4 - 1.5 * vsq) / 36.0;
SuperMatrix[i, j].particleProbability[5] = Rho * (1.0 + 3.0 * v5 +
 4.5 * v5 * v5 - 1.5 * vsq) / 9.0;
SuperMatrix[i, j].particleProbability[6] = Rho * (1.0 + 3.0 * v6 +
4.5 * v6 * v6 - 1.5 * vsq) / 36.0;
SuperMatrix[i, j].particleProbability[7] = Rho * (1.0 + 3.0 * v7 +
4.5 * v7 * v7 - 1.5 * vsq) / 9.0;
SuperMatrix[i, j].particleProbability[8] = Rho * (1.0 + 3.0 * v8 +
4.5 * v8 * v8 - 1.5 * vsq) / 36.0;
SuperMatrix[i, j].thermalProbability[0] = Rho * Temp * 4.0 / 9.0;
SuperMatrix[i, j].thermalProbability[1] = Rho * Temp * (1.0 + 3.0 * v1) / 9.0;
SuperMatrix[i, j].thermalProbability[2] = Rho * Temp * (1.0 + 3.0 * v2) / 36.0;
SuperMatrix[i, j].thermalProbability[3] = Rho * Temp * (1.0 + 3.0 * v3) / 9.0;
SuperMatrix[i, j].thermalProbability[4] = Rho * Temp * (1.0 + 3.0 * v4) / 36.0;
SuperMatrix[i, j].thermalProbability[5] = Rho * Temp * (1.0 + 3.0 * v5) / 9.0;
SuperMatrix[i, j].thermalProbability[6] = Rho * Temp * (1.0 + 3.0 * v6) / 36.0;
SuperMatrix[i, j].thermalProbability[7] = Rho * Temp * (1.0 + 3.0 * v7) / 9.0;
SuperMatrix[i, j].thermalProbability[8] = Rho * Temp * (1.0 + 3.0 * v8) / 36.0;
for (int k = 0; k < 9; k++)
{
    SuperMatrix[i, j].f_timestep[k] = SuperMatrix[i, j].particleProbability[k];
    SuperMatrix[i, j].g_timestep[k] = SuperMatrix[i, j].thermalProbability[k];
    SuperMatrix[i, j].f_equilibrium[k] = SuperMatrix[i, j].particleProbability[k];
    SuperMatrix[i, j].g_equilibrium[k] = SuperMatrix[i, j].thermalProbability[k];
}
double Theta = (InitialTemp - BoxTemp) / (TopPlateTemp - BoxTemp);
```

```
                //EnergyFunction(i, j, 0);
                SuperMatrix[i, j].energy = InitialTemp;
                if (FlowIndex == 0)
                {
                    if (i == 0 && j < (YDirLength - 1))
                        SuperMatrix[i, j].SOLID = 1;
                    if (i == (XDirLength - 1) && j < (YDirLength - 1))
                        SuperMatrix[i, j].SOLID = 1;
                    if (j == 0)
                        SuperMatrix[i, j].SOLID = 1;
                }

                if (FlowIndex == 1)
                {
                    if (j == 0)
                        SuperMatrix[i, j].SOLID = 1;
                }
            }
        }
    }//end InitalizeMatricies()
```

A.2 Order of Operations

In the code, to allow the order of operations to be easily changed, every operation was done with a function inside of a class. A function was also designed to run the other functions in the class.

```
public void LBM_Run(int step)
        {
            count++;
            VelocityChange = 0;
            VelocityChangedenom = 0;
            for (int j = 0; j < YDirLength; j++)
            {
                for (int i = 0; i < XDirLength; i++)
                {
                    LBMPropigation(i, j, count);
                }
            }
            for (int j = 0; j < YDirLength; j++)
            {
                for (int i = 0; i < XDirLength; i++)
                {
                    LBMVelocityAndBC(i, j, count);
                    LBMEquilibriumEQs(i, j, SuperMatrix[i,j].rho, SuperMatrix[i, j].UX,
                    SuperMatrix[i, j].UY, SuperMatrix[i, j].temperature, count);
                }
            }
            if (step % PrintInterval == 0 && step > 1)
            {
                LBMPrinting(step);
            }
            for (int j = 0; j < YDirLength; j++)
            {
                for (int i = 0; i < XDirLength; i++)
                {
                    if (SuperMatrix[i, j].SOLID == 0)
                        LBMCollisions(i, j, count);
                    if (SuperMatrix[i, j].SOLID == 1)
                        LBMBounceBack(i, j, count);
                }
            }
            VelocityChange = VelocityChange / VelocityChangedenom;
            if (step < 2)
                VelocityChange = 10;
        }
```

The first step is the streaming step of the lattice-Boltzmann method. In the code it is called "LBMPropigation." During the propagation phase the boundaries are treated as periodic and the bounce-back method combined with the collision operator not being applied in the solid nodes keeps the opposite walls from having an effect on the fluids in contact with the opposite wall.

The next step, "LBMVelocityAndBC," recalculates the velocity at each point in the lattice structure and applies the velocity of the top plate to the top row of the matrix. In "LBMEquilibriumEQs" the equilibrium distribution function is recalculated based on the new velocities calculated in "LBMVelocityAndBC." Then the time-step is tested against the printing criteria to determine if it should print out the data. If the criteria is met the data is output in both an Excel file and a TecPlot360 file.

Then the collision operator, "LBMCollision," is applied to all the fluid nodes, and the bounce-back operation, "LBMBounceBack" is applied to all of the solid nodes.

Finally, the convergence criteria is calculated. If the "VelocityChange" ever drops below a certain value the data will be outputted and the operation will cease.

A.3 LBMPropigation

```
private void LBMPropigation(int i, int j, int count)
        {
            int rowAbove = 0;
            int rowBelow = 0;
            int columnRight = 0;
            int columnLeft = 0;
            columnRight = (i + 1) % XDirLength;
            columnLeft = (i + (XDirLength - 1)) % XDirLength;
            rowAbove = (j + 1) % YDirLength;
            rowBelow = (j + (YDirLength - 1)) % YDirLength;
            SuperMatrix[i, j].particleProbability[0] = SuperMatrix[i, j].f_timestep[0];
            SuperMatrix[columnRight, j].particleProbability[1] = SuperMatrix[i, j].f_timestep[1];
            SuperMatrix[columnRight, rowAbove].particleProbability[2] = SuperMatrix[i, j].f_timestep[2];
            SuperMatrix[i, rowAbove].particleProbability[3] = SuperMatrix[i, j].f_timestep[3];
            SuperMatrix[columnLeft, rowAbove].particleProbability[4] = SuperMatrix[i, j].f_timestep[4];
            SuperMatrix[columnLeft, j].particleProbability[5] = SuperMatrix[i, j].f_timestep[5];
            SuperMatrix[columnLeft, rowBelow].particleProbability[6] = SuperMatrix[i, j].f_timestep[6];
            SuperMatrix[i, rowBelow].particleProbability[7] = SuperMatrix[i, j].f_timestep[7];
            SuperMatrix[columnRight, rowBelow].particleProbability[8] = SuperMatrix[i, j].f_timestep[8];
        }
```

A.4 LBMVelocityAndBC

```
public void LBMVelocityAndBC(int i, int j, int count)
        {
            int YDirLengthLessOne = YDirLength - 1;
            int XDirLengthLessOne = XDirLength - 1;
            double temp_UX = 0;
            double temp_UY = 0;
            double temp_magnitude = 0;
            double super_magnitude = 0;
            double temp_rho = 0;
            if (SuperMatrix[i,j].SOLID == 0)
            {
                temp_UX = SuperMatrix[i, j].UX;
                temp_UY = SuperMatrix[i, j].UY;
                temp_magnitude = Math.Sqrt(temp_UX * temp_UX + temp_UY * temp_UY);
                SuperMatrix[i, j].UX = 0;
                SuperMatrix[i, j].UY = 0;
                SuperMatrix[i, j].rho = 0;
                for (int k = 0; k < 9; k++)
                {
                    temp_rho += SuperMatrix[i, j].particleProbability[k];    //density calculation
                }
                SuperMatrix[i, j].rho = temp_rho;
                SuperMatrix[i, j].UX = ((SuperMatrix[i, j].particleProbability[1] +
                SuperMatrix[i, j].particleProbability[2] + SuperMatrix[i, j].particleProbability[8]) -
                (SuperMatrix[i, j].particleProbability[4] + SuperMatrix[i, j].particleProbability[5] +
                SuperMatrix[i, j].particleProbability[6])) / SuperMatrix[i, j].rho;
                SuperMatrix[i, j].UY = ((SuperMatrix[i, j].particleProbability[2] +
                SuperMatrix[i, j].particleProbability[3] + SuperMatrix[i, j].particleProbability[4]) -
                (SuperMatrix[i, j].particleProbability[6] + SuperMatrix[i, j].particleProbability[7] +
                SuperMatrix[i, j].particleProbability[8])) / SuperMatrix[i, j].rho;
            }
            if (j == YDirLengthLessOne)
            {
                SuperMatrix[i, j].UX = TopPlateVelocity;
                SuperMatrix[i, j].UY = 0.0;
            }
            if (count > 2)
            {
                super_magnitude = Math.Sqrt(SuperMatrix[i, j].UX * SuperMatrix[i, j].UX +
                SuperMatrix[i, j].UY * SuperMatrix[i, j].UY);
                VelocityChange += Math.Abs(temp_UX - SuperMatrix[i, j].UX);
                VelocityChangedenom += Math.Abs(SuperMatrix[i, j].UX);
            }
```

A.5 LBMEquilibriumEQs

```
private void LBMEquilibriumEQs(int i, int j, double Rho, double v_x, double v_y, double TempIn, int count)
        {
            double Temp, vsq, v0, v1, v2, v3, v4, v5, v6, v7, v8;
            Temp = TempIn;
            vsq = v_x * v_x + v_y * v_y;
            v0 = v_x + v_y;
            v1 = v_x;
            v2 = v_x + v_y;
            v3 = v_y;
            v4 = -v_x + v_y;
            v5 = -v1;
            v6 = -v2;
            v7 = -v3;
            v8 = -v4;
            SuperMatrix[i, j].f_equilibrium[0] = Rho * (1.0 - 1.5 * vsq) * 4.0 / 9.0;
            SuperMatrix[i, j].f_equilibrium[1] = Rho * (1.0 + 3.0 * v1 + 4.5 * v1 * v1 - 1.5 * vsq) / 9.0;
            SuperMatrix[i, j].f_equilibrium[2] = Rho * (1.0 + 3.0 * v2 + 4.5 * v2 * v2 - 1.5 * vsq) / 36.0;
            SuperMatrix[i, j].f_equilibrium[3] = Rho * (1.0 + 3.0 * v3 + 4.5 * v3 * v3 - 1.5 * vsq) / 9.0;
            SuperMatrix[i, j].f_equilibrium[4] = Rho * (1.0 + 3.0 * v4 + 4.5 * v4 * v4 - 1.5 * vsq) / 36.0;
            SuperMatrix[i, j].f_equilibrium[5] = Rho * (1.0 + 3.0 * v5 + 4.5 * v5 * v5 - 1.5 * vsq) / 9.0;
            SuperMatrix[i, j].f_equilibrium[6] = Rho * (1.0 + 3.0 * v6 + 4.5 * v6 * v6 - 1.5 * vsq) / 36.0;
            SuperMatrix[i, j].f_equilibrium[7] = Rho * (1.0 + 3.0 * v7 + 4.5 * v7 * v7 - 1.5 * vsq) / 9.0;
            SuperMatrix[i, j].f_equilibrium[8] = Rho * (1.0 + 3.0 * v8 + 4.5 * v8 * v8 - 1.5 * vsq) / 36.0;
        }
```

A.6 LBMCollision

```
private void LBMCollisions(int i, int j, int count)
        {
            for (int k = 0; k < 9; k++)
            {
                SuperMatrix[i, j].f_timestep[k] = SuperMatrix[i, j].particleProbability[k] -
                (SuperMatrix[i, j].particleProbability[k] - SuperMatrix[i, j].f_equilibrium[k]) /
                TauViscous;
            }
        }
```

A.7 LBMBounceBack

```
private void LBMBounceBack(int i, int j, int count)
        {
            double temporary = 0;
            temporary = SuperMatrix[i, j].f_timestep[1];
            SuperMatrix[i, j].f_timestep[1] = SuperMatrix[i, j].f_timestep[5];
            SuperMatrix[i, j].f_timestep[5] = temporary;
            temporary = SuperMatrix[i, j].f_timestep[2];
            SuperMatrix[i, j].f_timestep[2] = SuperMatrix[i, j].f_timestep[6];
            SuperMatrix[i, j].f_timestep[6] = temporary;
            temporary = SuperMatrix[i, j].f_timestep[3];
            SuperMatrix[i, j].f_timestep[3] = SuperMatrix[i, j].f_timestep[7];
            SuperMatrix[i, j].f_timestep[7] = temporary;
            temporary = SuperMatrix[i, j].f_timestep[4];
            SuperMatrix[i, j].f_timestep[4] = SuperMatrix[i, j].f_timestep[8];
            SuperMatrix[i, j].f_timestep[8] = temporary;
        }
```

A.8 LBMPrinting

```
internal void LBMPrinting(int step)
        {
            StreamWriter sw;
            if (step == PrintInterval)
            {
                sw = new StreamWriter("LBM_parameters_" + step.ToString() + ".txt", false);
                sw.WriteLine("X Direction Length = " + XDirLength.ToString());
                sw.WriteLine("Y Direction Length = " + YDirLength.ToString());
                sw.WriteLine("Reynolds Number = " + ReynoldsNumber.ToString());
                sw.WriteLine("Prandtl number = " + PrandtlNumber.ToString());
                sw.WriteLine("Tau Viscous = " + TauViscous.ToString());
                sw.WriteLine("Tau Thermal = " + TauThermal.ToString());
                sw.WriteLine("Top Plate Velocity = " + TopPlateVelocity.ToString());
                sw.WriteLine("Distance Between Nodes = " + DistranceBetweenPoints.ToString());
                sw.WriteLine("Number of Steps = " + step.ToString());
                sw.Close();
            }
            // TecPlot360 //
            int XDirLengthLessTwo = XDirLength - 2;
            int YDirLengthLessTwo = YDirLength - 2;
            sw = new StreamWriter("LBM_TecPlot_" + step.ToString() + ".plt");
            sw.WriteLine("variables=" + "\t" + "x" + "\t" + "y" + "\t" + "U" + "\t" + "V");
            sw.WriteLine("ZONE" + "\t" + "i=" + XDirLength.ToString() +
            "\t" + "j=" + YDirLength.ToString());
            for (int i = 0; i < (XDirLength); i++)
            {
                for (int j = 0; j < (YDirLength); j++)
                {
                    double x = (double)i / (double)(XDirLength - 1);
                    double y = (double)j / (double)(YDirLength - 1);
                    sw.WriteLine(x.ToString() + "\t" + y.ToString() + "\t" +
                    SuperMatrix[i, j].UX.ToString() + "\t" + SuperMatrix[i, j].UY.ToString());
                }
            }
            sw.Close();
            // Excel //
            sw = new StreamWriter("LBM_Verticle_" + step.ToString() + ".xls");
            sw.WriteLine("Y" + "\t" + "U");
            for (int j = 0; j < YDirLength; j++)
            {
                int i = XDirLength / 2;
                double y = (double)j / (double)(YDirLength - 1);
                sw.WriteLine(y.ToString() + "\t" + SuperMatrix[i, j].UX.ToString());
            }
            sw.Close();
            sw = new StreamWriter("LBM_Horizontal_" + step.ToString() + ".xls");
            sw.WriteLine("X" + "\t" + "V");
            for (int i = 0; i < XDirLength; i++)
            {
                int j = YDirLength / 2;
                double x = (double)i / (double)(XDirLength - 1);
                sw.WriteLine(x.ToString() + "\t" + SuperMatrix[i, j].UY.ToString());
            }
            sw.Close();
        }
```

APPENDIX B

CODE FOR THE THERMAL LATTICE-BOLTZMANN METHOD

The code for the thermal lattice-Boltzmann method follows the exact same methodology as the lattice-Boltzmann method. The difference is in the probability distribution equation and the values being calculated and reset are the temperatures. The parameters needed to run the thermal lattice-Boltzmann method were initialized at the same time as the parameters for the lattice-Boltzmann method.

B.1 Order of Operations

```
public void TLBM_Run(int step)
        {
            count++;
            TLBMChange = 0;
            TLBMChangedenom = 0;
            for (int j = 0; j < YDirLength; j++)
            {
                for (int i = 0; i < XDirLength; i++)
                {
                    TLBMPropigation(i, j, count);
                }
            }
            for (int j = 0; j < YDirLength; j++)
            {
                for (int i = 0; i < XDirLength; i++)
                {
                    TLBMTemperatureAndBC(i, j, count);
                    TLBMShiEQs(i, j, SuperMatrix[i, j].rho, SuperMatrix[i, j].UX,
                    SuperMatrix[i, j].UY, SuperMatrix[i, j].temperature, count);
                }
            }
            if (step % PrintInterval == 0 && step > 1)
            {
                TLBMPrinting(step);
            }
            for (int j = 0; j < YDirLength; j++)
            {
                for (int i = 0; i < XDirLength; i++)
                {
                    if (SuperMatrix[i, j].SOLID == 0)
                        TLBMCollisions(i, j, count);
                    if (SuperMatrix[i, j].SOLID == 1)
                        TLBMBounceBack(i, j, count);
                }
            }
            TLBMChange = TLBMChange / TLBMChangedenom;
            if (step < 2)
                TLBMChange = 10;
        }
```

B.2 TLBMPropigation

```
private void TLBMPropigation(int i, int j, int count)
        {
            int rowAbove = 0;
            int rowBelow = 0;
            int columnRight = 0;
            int columnLeft = 0;
            columnRight = (i + 1) % XDirLength;
            columnLeft = (i + (XDirLength - 1)) % XDirLength;
            rowAbove = (j + 1) % YDirLength;
            rowBelow = (j + (YDirLength - 1)) % YDirLength;
            SuperMatrix[i, j].thermalProbability[0] = SuperMatrix[i, j].g_timestep[0];
            SuperMatrix[columnRight, j].thermalProbability[1] = SuperMatrix[i, j].g_timestep[1];
            SuperMatrix[columnRight, rowAbove].thermalProbability[2] = SuperMatrix[i, j].g_timestep[2];
            SuperMatrix[i, rowAbove].thermalProbability[3] = SuperMatrix[i, j].g_timestep[3];
            SuperMatrix[columnLeft, rowAbove].thermalProbability[4] = SuperMatrix[i, j].g_timestep[4];
            SuperMatrix[columnLeft, j].thermalProbability[5] = SuperMatrix[i, j].g_timestep[5];
            SuperMatrix[columnLeft, rowBelow].thermalProbability[6] = SuperMatrix[i, j].g_timestep[6];
            SuperMatrix[i, rowBelow].thermalProbability[7] = SuperMatrix[i, j].g_timestep[7];
            SuperMatrix[columnRight, rowBelow].thermalProbability[8] = SuperMatrix[i, j].g_timestep[8];
        }
```

B.3 TLBMTemperatureAndBC

```
public void TLBMTemperatureAndBC(int i, int j, int count)
        {
            double temp_Temp = SuperMatrix[i, j].temperature;
            int YDirLengthLessOne = YDirLength - 1;
            int XDirLengthLessOne = XDirLength - 1;
            double Trho = 0;
            if (SuperMatrix[i, j].SOLID == 0)
            {
                Trho = SuperMatrix[i, j].thermalProbability[0] +
                SuperMatrix[i, j].thermalProbability[1] + SuperMatrix[i, j].thermalProbability[2] +
                SuperMatrix[i, j].thermalProbability[3] + SuperMatrix[i, j].thermalProbability[4] +
                SuperMatrix[i, j].thermalProbability[5] + SuperMatrix[i, j].thermalProbability[6] +
                SuperMatrix[i, j].thermalProbability[7] + SuperMatrix[i, j].thermalProbability[8];
                SuperMatrix[i, j].temperature = Trho / SuperMatrix[i, j].rho;
            }
            if (j == YDirLengthLessOne)
            {
                SuperMatrix[i, j].temperature = TLBMTopPlateTemp;
            }
            if (SuperMatrix[i, j].SOLID == 1)
            {
                SuperMatrix[i, j].temperature = BoxTemp;
            }
            if (j == 1)
            {
                SuperMatrix[i, j].temperature = BoxTemp;
            }
            if (i == 1 || i == (XDirLengthLessOne - 1) && j < YDirLengthLessOne)
            {
                SuperMatrix[i, j].temperature = BoxTemp;
            }
            if (count > 2)
            {
                TLBMChange += Math.Abs(temp_Temp - SuperMatrix[i, j].temperature);
                TLBMChangedenom += Math.Abs(SuperMatrix[i, j].temperature);
            }
        }
```

In order to get the temperatures to propagate correctly, at the solid boundaries two rows or columns are set to the desired temperature.

B.4 TLBMShiEQs

```
private void TLBMShiEQs(int i, int j, double Rho, double v_x, double v_y, double TempIn, int count)
        {
            double Temp, vsq, v0, v1, v2, v3, v4, v5, v6, v7, v8;
            Temp = TempIn;
            vsq = v_x * v_x + v_y * v_y;
            v0 = v_x + v_y;
            v1 = v_x;
            v2 = v_x + v_y;
            v3 = v_y;
            v4 = -v_x + v_y;
            v5 = -v1;
            v6 = -v2;
            v7 = -v3;
            v8 = -v4;
            SuperMatrix[i, j].g_equilibrium[0] = Rho * Temp * 4.0 / 9.0;
            SuperMatrix[i, j].g_equilibrium[1] = Rho * Temp * (1.0 + 3.0 * v1) / 9.0;
            SuperMatrix[i, j].g_equilibrium[2] = Rho * Temp * (1.0 + 3.0 * v2) / 36.0;
            SuperMatrix[i, j].g_equilibrium[3] = Rho * Temp * (1.0 + 3.0 * v3) / 9.0;
            SuperMatrix[i, j].g_equilibrium[4] = Rho * Temp * (1.0 + 3.0 * v4) / 36.0;
            SuperMatrix[i, j].g_equilibrium[5] = Rho * Temp * (1.0 + 3.0 * v5) / 9.0;
            SuperMatrix[i, j].g_equilibrium[6] = Rho * Temp * (1.0 + 3.0 * v6) / 36.0;
            SuperMatrix[i, j].g_equilibrium[7] = Rho * Temp * (1.0 + 3.0 * v7) / 9.0;
            SuperMatrix[i, j].g_equilibrium[8] = Rho * Temp * (1.0 + 3.0 * v8) / 36.0;
        }
```

The equilibrium equations used for the thermal lattice-Boltzmann calculations were developed by Shi et al. [40].

B.5 TLBMCollisions

```
private void TLBMCollisions(int i, int j, int count)
        {
            for (int k = 0; k < 9; k++)
            {
                SuperMatrix[i, j].g_timestep[k] = SuperMatrix[i, j].thermalProbability[k] -
                (SuperMatrix[i, j].thermalProbability[k] - SuperMatrix[i, j].g_equilibrium[k]) /
                TauThermal;
            }
        }
```

B.6 TLBMBounceBack

```
private void TLBMBounceBack(int i, int j, int count)
        {
            double temporary = 0;
            temporary = 0;
            temporary = SuperMatrix[i, j].g_timestep[1];
            SuperMatrix[i, j].g_timestep[1] = SuperMatrix[i, j].g_timestep[5];
            SuperMatrix[i, j].g_timestep[5] = temporary;
            temporary = SuperMatrix[i, j].g_timestep[2];
            SuperMatrix[i, j].g_timestep[2] = SuperMatrix[i, j].g_timestep[6];
            SuperMatrix[i, j].g_timestep[6] = temporary;
            temporary = SuperMatrix[i, j].g_timestep[3];
            SuperMatrix[i, j].g_timestep[3] = SuperMatrix[i, j].g_timestep[7];
            SuperMatrix[i, j].g_timestep[7] = temporary;
            temporary = SuperMatrix[i, j].g_timestep[4];
            SuperMatrix[i, j].g_timestep[4] = SuperMatrix[i, j].g_timestep[8];
            SuperMatrix[i, j].g_timestep[8] = temporary;
        }
```

B.7 TLBMPrinting

The printing of the data is the exact same methodology as the "LBMPrinting" but the files are renamed. The data was printed at the end of each operation to guarantee that the time-steps to convergence would be saved.

APPENDIX C

CODE FOR THE ENERGY EQUATION

The solution of the energy equation for the temperature is the most simplistic operation in the code. It is done in an explicit fashion.

C.1 Order of Operations

```
public void Energy_Run(int step)
        {
            count++;
            EnergyChange = 0;
            EnergyChangedenom = 0;
            if (step < 2)
            {
                for (int j = 0; j < YDirLength; j++)
                {
                    for (int i = 0; i < XDirLength; i++)
                    {
                        EnergyFunctionInitalize(i, j, count);
                    }
                }
            }
            for (int j = 1; j < (YDirLength - 1); j++)
            {
                for (int i = 1; i < (XDirLength - 1); i++)
                {
                    AidunEnergyFunction(i, j, count);
                }
            }
            if (step % PrintInterval == 0 && step > 1)
            {
                EnergyPrinting(step);
            }
            EnergyChange = EnergyChange / EnergyChangedenom;
            if (step < 2)
            EnergyChange = 10;
        }
```

C.2 EnergyFunctionInitalize

```
 private void EnergyFunctionInitalize(int i, int j, int count)
        {
            for (i = 0; i < XDirLength; i++)
            {
                SuperMatrix[i, 0].energy = 0.0;
                SuperMatrix[i, (YDirLength - 1)].energy = 1.0;
            }
            for (j = 0; j < YDirLength; j++)
            {
                SuperMatrix[0, j].energy = 0.0;
                SuperMatrix[(XDirLength - 1), j].energy = 0.0;
            }
        }
```

C.3 EnergyFunction

```
private void EnergyFunction(int i, int j, int count)
        {
            double temp_Energy = SuperMatrix[i, j].energy;
            double UX = SuperMatrix[i, j].UX / TopPlateVelocity;
            double UY = SuperMatrix[i, j].UY / TopPlateVelocity;
            double deltaX = (1 / XDirLength);
            double deltaY = (1 / YDirLength);
            double TempCurrent = SuperMatrix[i, j].energy;
            double TempLeft = SuperMatrix[(i - 1), j].energy;
            double TempRight = SuperMatrix[(i + 1), j].energy;
            double TempDown = SuperMatrix[i, (j - 1)].energy;
            double TempUp = SuperMatrix[i, (j + 1)].energy;
            double nu = (2 * TauViscous - 1) / 6;
            double chi = (2 * TauThermal - 1) / 6;
            PrandtlNumber = nu / chi;
            double kappa = 1 / (ReynoldsNumber * PrandtlNumber);
            double gradient_x = (TempLeft - TempRight) / (2 * deltaX);
            double gradient_y = (TempDown - TempUp) / (2 * deltaY);
            double gradientsqr_x = (TempLeft + TempRight - 2 * TempCurrent) / (deltaX * deltaX);
            double gradientsqr_y = (TempDown + TempUp - 2 * TempCurrent) / (deltaY * deltaY);
            SuperMatrix[i, j].energy = TempCurrent + kappa * gradientsqr_x +
            kappa * gradientsqr_y + UX * gradient_x + UY * gradient_y;
            if (count > 2)
            {
                EnergyChange += Math.Abs(temp_Energy - SuperMatrix[i, j].energy);
                EnergyChangedenom += Math.Abs(SuperMatrix[i, j].energy);
            }
        }
```

C.4 EnergyPrinting

Again the printing is the same just with different file names. The data was printed at the end of each operation to guarantee that the time-steps to convergence would be saved.

REFERENCES

[1] Abe, T., "Derivation of the lattice boltzmann method by means of the discrete ordinate method for the boltzmann equation," *Journal of Computational Physics*, vol. 131, no. 1, pp. 241 – 246, 1997.

[2] Aidun, C. K., Lu, Y., and Ding, E.-J., "Direct analysis of particulate suspensions with inertia using the discrete boltzmann equation," *Journal of Fluid Mechanics*, vol. 373, pp. 287–311, 1998.

[3] Alexander, F. J., Chen, S., and Sterling, J. D., "Lattice boltzmann thermohydrodynamics," *Phys. Rev. E*, vol. 47, pp. R2249–R2252, Apr 1993.

[4] Azwadi, C. S. N. and Tanahashi, T., "Simplified thermal lattice boltzmann in incompressible limit," *International Journal of Modern Physics*, vol. 20, no. 17, pp. 2437–2449, 2006.

[5] Azwadi, C. S. N. and Tanahashi, T., "Three-dimensional thermal lattice boltzmann simulation of natural convection in a cubic cavity," *International Journal of Modern Physics*, vol. 21, no. 1, pp. 87–96, 2007.

[6] Barios, G., Rechtman, R., Rojas, J., and Tovar, R., "The lattice boltzmann equation for natural convection in a two-dimensional cavity with a partially heated wall," *Journal of Fluid Mechanics*, vol. 522, no. -1, pp. 91–100, 2005.

[7] Bhatnagar, P. L., Gross, E. P., and Krook, M., "A model for collision processes in gases. i. small amplitude processes in charged and neutral one-component systems," *Phys. Rev.*, vol. 94, pp. 511–525, May 1954.

[8] Bozeman, J. D. and Dalton, C., "Numerical study of viscous flow in a cavity," *Journal of Computational Physics*, vol. 12, no. 3, pp. 348 – 363, 1973.

[9] Chen, S. and Doolen, G. D., "Lattice boltzmann method for fluid flows," *Annual Review Fluid Mechanics*, vol. 30, pp. 329–364, 1998.

[10] Chen, Y., Ohashi, H., and Akiyama, M., "Thermal lattice bhatnagar-gross-krook model without nonlinear deviations in macrodynamic equations," *Phys. Rev. E*, vol. 50, no. 4, pp. 2776–2783, 1994.

[11] d Humières, D., Ginzburg, I., Krafczyk, M., Lallemand, P., and Luo, L., "Multiple-relaxation-time lattice boltzmann models in three dimensions," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 360, no. 1792, pp. 437–451, 2002.

[12] Dixit, H. and Babu, V., "Simulation of high rayleigh number natural convection in a square cavity using the lattice boltzmann method," *International Journal of Heat and Mass Transfer*, vol. 49, no. 3-4, pp. 727 – 739, 2006.

[13] D'Orazio, A. and Succi, S., "Simulating two-dimensional thermal channel flows by means of a lattice boltzmann method with new boundary conditions," *Future Generation Computer Systems*, vol. 20, no. 6, pp. 935 – 944, 2004.

[14] Eggels, J. G. M. and Somers, J. A., "Numerical simulation of free convective flow using the lattice-boltzmann scheme," *International Journal of Heat and Fluid Flow*, vol. 16, no. 5, pp. 357 – 364, 1995.

[15] Eggels, J. G. M., "Direct and large-eddy simulation of turbulent fluid flow using the lattice-boltzmann scheme," *International Journal of Heat and Fluid Flow*, vol. 17, no. 3, pp. 307 – 323, 1996.

[16] Erturk, E., Corke, T. C., and Gokcol, C., "Numerical solutions of 2-d steady incompressible driven cavity flow at high reynolds numbers," *International Journal for Numerical Methods in Fluids*, vol. 48, pp. 747–774, March 2005.

[17] Frisch, U., Hasslacher, B., and Pomeau, Y., "Lattice-gas automata for the navier-stokes equation," *Phys. Rev. Lett.*, vol. 56, pp. 1505–1508, Apr 1986.

[18] Ghia, U., Ghia, K. N., and Shin, C. T., "High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method"," *Journal of Computational Physics*, vol. 48, no. 3, pp. 387 – 411, 1982.

[19] Ginzbourg, I. and d'Humières, D., "Local second-order boundary methods for lattice boltzmann models," *Journal of Statistical Physics*, vol. 84, no. 5, pp. 927–971, 1996.

[20] Guo, Z., Zheng, C., Shi, B., and Zhao, T. S., "Thermal lattice boltzmann equation for low mach number flows: Decoupling model," *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 75, no. 3, p. 036704, 2007.

[21] He, X., Chen, S., and Doolen, G. D., "A novel thermal model for the lattice boltzmann method in incompressible limit," *Journal of Computational Physics*, vol. 146, pp. 282–300, 1998.

[22] He, X. and Luo, L.-S., "A priori derivation of the lattice boltzmann equation," *Phys. Rev. E*, vol. 55, pp. R6333–R6336, Jun 1997.

[23] Hou, S., Zou, Q., Chen, S., Doolen, G., and Cogley, A. C., "Simulation of cavity flow by the lattice boltzmann method," *Journal of Computational Physics*, vol. 118, no. 2, pp. 329 – 347, 1995.

[24] Huang, H., Lee, T. S., and Shu, C., "Thermal curved boundary treatment for the thermal lattice boltzmann equation," *International Journal of Modern Physics*, vol. 17, no. 5, pp. 631–643, 2006.

[25] Khiabani, R. H., Joshi, Y., and Aidun, C. K., "Heat transfer in microchannels with suspended solid particles: Lattice-boltzmann based computations," Master's thesis, Georgia Institute of Technology, 2006.

[26] Lallemand, P. and shi Luo, L., "Hybrid finite-difference thermal lattice boltzmann equation," *International Journal of Modern Physics*, vol. 17, no. 1, pp. 41–47, 2003.

[27] Lü, X.-Y., Zhang, C.-Y., Liu, M.-R., Kong, L.-J., and Li, H.-B., "Thermal lattice boltzmann simulation of viscous flow in a square cavity," *International Journal of Modern Physics C: Computational Physics & Physical Computation*, vol. 16, no. 6, pp. 867 – 877, 2005.

[28] McNamara, G., Garcia, A., and Alder, B., "Stabilization of thermal lattice boltzmann models," *Journal of Statistical Physics*, vol. 81, no. 1, pp. 395–408, 1995.

[29] McNamara, G., Garcia, A., and Alder, B., "A hydrodynamically correct thermal lattice boltzmann model," *Journal of Statistical Physics*, vol. 87, no. 5, pp. 1111–1121, 1997.

[30] McNamara, G. and Alder, B., "Analysis of the lattice boltzmann treatment of hydrodynamics," *Physica A*, vol. 194, pp. 218–228, 1993.

[31] McNamara, G. R. and Zanetti, G., "Use of the boltzmann equation to simulate lattice-gas automata," *Phys. Rev. Lett.*, vol. 61, pp. 2332–2335, Nov 1988.

[32] Noble, D. R., Chen, S., Georgiadis, J. G., and Buckius, R. O., "A consistent hydrodynamic boundary condition for the lattice boltzmann method," *Physics of Fluids*, vol. 7, pp. 203–209, 1995.

[33] Nourgaliev, R. R., Dinh, T. N., Theofanous, T. G., and Joseph, D., "The lattice boltzmann equation method: theoretical interpretation, numerics and implications," *International Journal of Multiphase Flow*, vol. 29, no. 1, pp. 117 – 169, 2003.

[34] Peng, Y., Shu, C., and Chew, Y. T., "Simplified thermal lattice boltzmann model for incompressible thermal flows," *Phys. Rev. E*, vol. 68, p. 026701, Aug 2003.

[35] Piaud, B., Blanco, S., Fournier, R., and Clifton, M. J., "Energy-conserving lattice boltzmann thermal model in two dimensions," *Journal of Statistical Physics*, vol. 121, pp. 119–131, Oct 2005.

[36] Seta, T., Takegoshi, E., and Okui, K., "Effects of the body force in the thermal lattice boltzmann method," *I. J. Trans. Phenomena*, vol. 10, pp. 1–15, 2008.

[37] Shan, X. and Doolen, G. D., "Multicomponent lattice-boltzmann model with interparticle interaction," *Journal of Statistical Physics*, vol. 81, no. 1, pp. 379–393, 1995.

[38] Shan, X., "Simulation of rayleigh-b'enard convection using a lattice boltzmann method," in *Physical Review E*, vol. 55, pp. 2780–2788, The Americal Physical Society, 1997.

[39] Shan, X., Yuan, X.-F., and Chen, H., "Kinetic theory representation of hydrodynamics: a way beyond the navier–stokes equation," *Journal of Fluid Mechanics*, vol. 550, pp. 413–441, 2006.

[40] Shi, Y., Zhao, T. S., and Guo, Z. L., "Thermal lattice bhatnagar-gross-krook model for flows with viscous heat dissipation in the incompressible limit," *Phys. Rev. E*, vol. 70, p. 066310, Dec 2004.

[41] Shu, C., Niu, X. D., and Chew, Y. T., "A lattice boltzmann kinetic model for microflow and heat transfer," *Journal of Statistical Physics*, vol. 121, pp. 239–255, October 2005.

[42] Soe, M., Vahala, G., Pavlo, P., Vahala, L., and Chen, H., "Thermal lattice boltzmann simulations of variable prandtl number turbulent flows," *Phys. Rev. E*, vol. 57, pp. 4227–4237, Apr 1998.

[43] Tang, G. H., Tao, W. Q., and He, Y. L., "Thermal boundary condition for the thermal lattice boltzmann equation," *Phys. Rev. E*, vol. 72, p. 016703, Jul 2005.

[44] van Treeck, C., Rand, E., Krafczyk, M., T/:olke, J., and Nachtwey, B., "Extension of a hybrid thermal lbe scheme for large-eddy simulations of turbulent convective flows," *Computers & Fluids*, vol. 35, no. 8-9, pp. 863 – 871, 2006. Proceedings of the First International Conference for Mesoscopic Methods in Engineering and Science.

[45] Wolf-Gladrow, D. A., *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction.* Springer, 2000.

[46] Yuan, P. and Schaefer, L., "A thermal lattice boltzmann two-phase flow model and its application to heat transfer problems–part 2. integration and validation," *Journal of Fluid Engineering*, vol. 128, pp. 151–156, January 2006.

[47] Zhang, R. and Chen, H., "Lattice boltzmann method for simulations of liquid-vapor thermal flows," *Phys. Rev. E*, vol. 67, p. 066711, Jun 2003.

[48] Ziegler, D. P., "Boundary conditions for lattice boltzmann simulations," *Journal of Statistical Physics*, vol. 71, pp. 1171–1177, 1993.

[49] Zou, Q., Hou, S., Chen, S., and Doolen, G. D., "A improved incompressible lattice boltzmann model for time-independent flows," *Journal of Statistical Physics*, vol. 81, no. 1, pp. 35–48, 1995.