Validation of the Lattice Boltzmann Method for Direct Numerical Simulation of Wall-Bounded Turbulent Flows

by

Dustin John Bespalko

A thesis submitted to the
Department of Mechanical and Materials Engineering
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen's University
Kingston, Ontario, Canada
September 2011

Abstract

In this work, the lattice Boltzmann method (LBM) was validated for direct numerical simulation (DNS) of wall-bounded turbulent flows. The LBM is a discrete-particle-based method that numerically solves the Boltzmann equation as opposed to conventional DNS methods that are based on the Navier-Stokes (NS) equations. The advantages of the LBM are its simple implementation, its ability to handle complex geometries, and its scalability on modern high-performance computers.

An LBM code was developed and used to simulate fully-developed turbulent channel flow. In order to validate the results, the turbulence statistics were compared to those calculated from a conventional NS-based finite difference (FD) simulation. In the present study, special care was taken to make sure the computational domains for LBM and FD simulations were the same. Similar validation studies in the literature have used LBM simulations with smaller computational domains in order to reduce the computational cost. However, reducing the size of the computational domain affects the turbulence statistics and confounds the results of the validation.

The turbulence statistics calculated from the LBM and FD simulations were found to agree qualitatively; however, there were several significant deviations, particularly in the variance profiles. The largest discrepancy was in the variance of the pressure fluctuations, which differed by approximately 7%. Given that both the LBM and FD simulations resolved the full range of turbulent scales and no models were used, this error was deemed to be significant.

The cause of the discrepancy in the pressure variance was found to be the compressibility of the LBM. The LBM allows the density to vary, while the FD method does not since it solves the incompressible form of the NS equations. The effect of the compressibility could be reduced by lowering the Mach number, but this would come at the cost of significantly increasing the computational cost. Therefore, the conclusion of this work is that, while the LBM is capable of producing accurate solutions for incompressible turbulent flows, it is significantly more expensive than conventional methods for simple wall-bounded turbulent flows.

Acknowledgements

I would like to thank Professor Andrew Pollard for his supervision of this project. His careful guidance and helpful suggestions have been an enormous help throughout my thesis. I will always be thankful for the contribution he has made to my development as a researcher.

I would also like to acknowledge the support of Professor Mesbah Uddin. This thesis has benefited tremendously from his knowledge and expertise.

Thanks to Professor Ugo Piomelli for providing the finite difference code used in this work, and to Dr. Liang Wei for allowing me access to the data from his discontinuous Galerkin finite element simulation.

I would also like to thank the past and present members of the Computational and Experimental Fluid Dynamics Laboratory: Christopher Ball, Andrew Duncan, Dr. Hachimi Fellouah, Dr. Guillaume Fournier, Dr. François Golanski, Alireza Mahdavifar, Dr. Hassan Raiesi, Hamed Sadeghi, Frank Secretain, Dr. Abdul-Monsif Shinneeb, Professor Mesbah Uddin, and Dr. Liang Wei. I have valued your friendship over the past years, and our discussions have greatly influenced my work.

Financial support for this thesis was provided by the Natural Science and Engineering Research Council (NSERC), the Ontario Graduate Scholarship (OGS) program, and Sun Microsystems of Canada. The computational resources were provided by the High Performance Computing Virtual Laboratory (HPCVL).

Finally, I would like to especially thank my family for all of their love, encouragement, and support during my studies. To my parents Rose and Doug, brothers Ryan and Dylan, and grandparents Nola and John - I couldn't have done it without you. Shelly and Augustine - I love you.

Contents

List of Figures

Nomenclature

Latin Symbols

$\vec{c}$	particle velocity $[m/s]$.
$\vec{c}_0$	particle peculiar velocity $[m/s]$.
$\vec{c}_i$	discrete particle velocity $[m/s]$.
c_s	sound speed $[m/s]$.
E	total energy per unit mass $[J/kg]$.
e	internal energy per unit mass $[J/kg]$.
F_N	N-particle probability density distribution.
f	single-particle mass density distribution $[kg \cdot s^3/m^6]$.
f^{eq}	single-particle equilibrium mass density distribution $[kg \cdot s^3/m^6]$.
f_i	discrete single-particle mass density distribution $[kg/m^3]$.
f_i^{eq}	discrete single-particle equilibrium mass density distribution $[kg/m^3]$.
$\tilde{f}_i$	post-collision single-particle mass density distribution $[kg/m^3]$.
$\vec{g}$	acceleration of body force $[m^2/s]$.
g_i	forcing function in the lattice Boltzmann equation $[kg/m^3]$.
I	identity matrix.
H	Hamiltonian $[kg \cdot m^2/s]$.
$\mathcal{H}$	negative of the probability of the current state of the system.
I	the identity matrix.
J	collision operator $[kg \cdot s^2/m^6]$.

k	thermal conductivity $[W/m \cdot K]$.
k_B	the Boltzmann constant $[1.3806503 \times 10^-23 kg \cdot m^2/s^2 \cdot K]$.
L	Lagrangian $[kg \cdot m^2/s]$.
L_x	length of simulation domain in the streamwise direction $[m]$.
L_y	length of simulation domain in the spanwise direction $[m]$.
L_z	length of simulation domain in the wall-normal direction $[m]$.
l	characteristic length scale $[m]$.
m	particle mass $[kg]$.
N	number of particles.
N_t	number of snapshots.
N_x	number of grid nodes in the x-direction.
N_y	number of grid nodes in the y-direction.
N_z	number of grid nodes in the z-direction.
n	number of threads.
P	total stress tensor $[N/m^2]$.
$\vec{p}$	particle momentum $[kg \cdot m/s]$.
p	pressure $[N/m^2]$.
p_w	wall pressure $[N/m^2]$.
$\overline{p}$	mean pressure $[N/m^2]$.
$\vec{p}$	momentum vector $[kg \cdot m/s]$.
$\vec{Q}$	heat flux vector $[W/m^2]$.
R	specific gas constant $[J/kg \cdot K]$.
s_f	serial fraction.
T	temperature $[K]$.
T'	fluctuating component of temperature $[K]$.
t	time $[s]$.
t_1	serial execution time $[s]$.

t_n	parallel execution time with n threads $[s]$.
t_p	time required to run the parallel portion of the code $[s]$.
t_s	time required to run the serial portion of the code $[s]$.
U_0	mean centreline velocity $[m/s]$.
u	streamwise velocity $[m/s]$.
$\vec{u}$	fluid velocity vector $[m/s]$.
u_τ	friction velocity $[m/s]$.
v	spanwise velocity $[m/s]$.
$\vec{v}$	particle velocity vector $[m/s]$.
w	wall-normal velocity $[m/s]$.
W_i	discrete equilibrium distribution weights $[kg/m^3]$.
w_i	quadrature coefficient $[m^3/s^3]$.
$\vec{x}$	position vector $[m]$.
x	streamwise coordinate $[m]$.
y	spanwise coordinate $[m]$.
z	wall-normal coordinate $[m]$.

Greek Symbols

δ	channel half-width $[m]$.
Δt	time step $[s]$.
Δx	grid spacing $[m]$.
δ_ν	viscous length scale $[m]$.
ϵ	expansion parameter.
η	Kolmogorov length scale $[m]$.
κ	wavenumber $[1/m]$.
λ	bulk viscosity $[kg/m \cdot s]$.
μ	dynamic viscosity $[kg/m \cdot s]$.

ν	kinematic viscosity $[m^2/s]$.
ρ	density $[kg/m^3]$.
σ	particle diameter $[m]$.
τ	shear stress tensor N/m^2.
τ_w	wall shear stress $[N/m^2]$.
ϕ_{ij}	interparticle potential $[N \cdot m]$.
ψ	arbitrary function of particle velocities.
Ω	collision cross-sectional area $[m^2]$.
ω	relaxation frequency $[1/s]$.
$\hat{\omega}$	relaxation frequency multiplied by the time step.

Brackets, Superscripts, and Subscripts

$\langle\phi\rangle$	Reynolds ensemble average.
$\{\phi\}$	Favre ensemble average.
ϕ'	fluctuating component of a Reynolds decomposition.
ϕ''	fluctuating component of a Favre decomposition.
ϕ^+	normalization by the viscous scales.
ϕ^*	normalization by the largest scales.
$\tilde{\phi}$	post-collision value.

Non-dimensional Numbers

Kn	Knudsen number.
Ma	Mach number.
Re	Reynolds number.
Re_τ	Friction Reynolds number.

Acronyms

BGK	Bhatnager-Gross-Krook.
CFD	Computational Fluid Dynamics.
CPU	Central Processing Unit.
D3Q19	Three-dimensional lattice with thirteen velocities.
DBE	Discrete Boltzmann Equation.
DG	Discontinuous Galerkin.
DNS	Direct Numerical Simulation.
FD	Finite Difference.
HPCVL	High Performance Computing Virtual Laboratory.
LBE	Lattice Boltzmann Equation.
LBM	Lattice Boltzmann Method.
LES	Large Eddy Simulation.
LETOT	Large Eddy Turn Over Time.
LGCA	Lattice Gas Cellular Automata.
NS	Navier-Stokes.
RANS	Reynolds Averaged Navier-Stokes.

Chapter 1

Introduction

Fluid flow is governed by the Navier-Stokes (NS) equations. These equations yield either laminar or turbulent flows depending on the Reynolds number, which is the ratio of the inertial and viscous forces in the flow. If the Reynolds number is low, the viscous forces dominate and small perturbations in the flow are immediately damped out and the flow remains laminar. At higher Reynolds numbers, the inertial forces are more important and the same perturbations instead amplify and cause a transition to a turbulent flow.

In a fully turbulent flow, the largest turbulent structures (called "eddies") derive their energy from the mean shear and transport it to successively smaller scales in what is known as the turbulence energy cascade. The energy is eventually dissipated by the smallest scales. At this level, the turbulent fluctuations are small enough that they are damped out by viscous forces.

While a number of analytical solutions exist for simple laminar flows, there are no known solutions for turbulence. For this reason, the Navier-Stokes equations must be solved numerically for turbulent flows. A direct numerical simulation (DNS) is a simulation that solves the NS equations without the use of any models other than those of a numerical nature [21]. In order to be considered a DNS, a simulation must resolve the full range of scales in a turbulent flow. The computational domain must be large enough to accommodate the largest scales and the grid resolution must be sufficiently fine to capture the energy content of the smallest scales [40].

Unfortunately, the width of the spectrum of turbulent scales increases with the Reynolds number. The ratio of the Kolmogorov length scale to the integral scale, which are the length scales for the largest and smallest scales, respectively, is proportional to $Re^{-3/4}$. Thus, since the resolution requirements are roughly the same in each coordinate direction, the computational requirements for a DNS scale approximately with $Re^{9/4}$. For this reason, DNS simulations are typically limited to low-Reynolds number flows.

In a large-eddy simulation (LES) the smallest scales in the flow are replaced with a model while the large scales are accurately resolved. The dissipation of the smallest scales is accounted for by introducing an eddy viscosity. In a Reynolds averaged Navier-Stokes (RANS) model, all of the turbulent motions are modelled. The influence of turbulence on the mean flow is included by modelling the behaviour of the Reynolds stresses in the ensemble averaged NS equations.

While LES and RANS simulations are less computationally expensive, and thus are capable of simulating higher Reynolds number flows, they are not as reliable as direct numerical simulations. Furthermore, since these methods do not resolve all of the turbulent scales, they are not useful for studying the detailed structures of turbulence. Thus, DNS has become a useful and necessary research tool in the fundamental study of turbulent flows.

Direct numerical simulations are often performed using fully-spectral, or pseudo-spectral methods. These methods are used because of their superior accuracy and computational efficiency. Spectral methods transform the Navier-Stokes equations into spectral space in order to represent the flow field as a finite set of basis functions. The governing equations are then solved in spectral space. Since the derivatives in the Navier-Stokes equations do not need to be approximated in these methods, they are essentially free of truncation errors, and thus are very accurate. Another advantage of spectral methods is that the solution of the Poisson equation for pressure (for incompressible flows), which is usually a very computationally expensive operation, is reduced to a simple division in spectral space. This makes spectral codes very computationally efficient. Spectral methods work very well for problems with simple geometry, especially those with periodic boundaries. However, the implementation of these methods is considerably more difficult for problems with complex geometries because choosing appropriate basis functions that satisfy the boundary conditions is not easy. For this reason, finite difference and finite volume methods are often used for problems with complex geometries.

In finite difference and finite volume methods, a computational grid is introduced in physical space and the flow field is represented by its values on the discrete grid points. In the finite difference method, the derivatives in the Navier-Stokes equations at any particular grid point are approximated in terms of the neighbouring grid points. The number of neighbours considered depends on the order of the discretisation scheme used. For the finite volume method, the Navier-Stokes equations are integrated over a control volume (defined by the computational grid). The fluxes entering the control volume are then estimated using a discretisation scheme. In both of these methods there are truncation errors introduced by approximating the derivatives (for the finite difference case), or the fluxes (for the finite volume case). This truncation error is a problem when turbulent flows are simulated because it effectively reduces the maximum wavenumber that is resolved by the grid. Thus, compared to a spectral method, a finite difference or finite volume code must use a finer mesh in order to reduce the truncation error. Additionally, both of these methods solve the Poisson equation for pressure in physical space; this can take up to 80% of the total computational time. Thus, these methods are also much less computationally efficient compared to spectral methods. So, while finite difference and finite volume codes are easier to implement than spectral codes for problems with complex geometries, the computational time required to run these simulations can be as much as an order of magnitude larger.

In this work, the lattice Boltzmann method (LBM) is investigated as a possible alternative to these methods for direct numerical simulations of wall-bounded turbulent flows. The LBM is a discrete-particle-based method that solves the Boltzmann equation using a finite difference method. Instead of solving for the macroscopic flow variables, such as density, velocity, and temperature, the LBM solves for what is referred to as the single particle distribution. The single particle distribution represents the expected mass density of particles at a position $\vec{x}$, travelling with a velocity $\vec{c}$. Once the single particle distribution is known, its moments can be calculated to determine the equivalent macroscopic state.

The LBM has a number of attractive features that make it a candidate for simulating complex turbulent flows [12]:

- The convection operator in the LBM is linear. This is in contrast to the second-order nonlinearity in the Navier-Stokes equations that is very time consuming to solve numerically. The LBM recovers the same nonlinear behaviour as the NS equations through a relaxation process that models the particle interactions.

- The LBM discretises the Boltzmann equation in velocity space and only retains a minimal set of particle velocities that are needed to recover the proper macroscopic behaviour. This makes the LBM considerably more computationally efficient than other particle-based methods.

- It can be shown that the LBM is isotropic to second-order, and thus the orientation of the computational grid only has a small effect on the solution. This is ideal for computations of turbulent flows due to the three-dimensional nature of turbulent structures. If the orientation of the computational grid influences the solution, the orientation of these structures will be affected.

- The pressure in the LBM is computed from the ideal gas law, and so it does not require the solution of a Poisson equation to determine the pressure. This is often the most expensive operation in a finite difference method.

- The LBM is also very simple to implement. It is completely explicit and thus does not require any matrix inversion. Each time step can be split into a collision and streaming step. The particle distributions are first relaxed towards their equilibrium values in the collision step, then the post-collision distributions are advected to their neighbouring lattice sites in the streaming step. This simple implementation reduces the development time required for LBM codes and makes them simple to modify for complex geometries.

- The implementation of the LBM is very amenable to parallelization on modern high-performance computers. The collision step is embarrassingly parallel by definition, and the streaming step requires very little transfer of data between processors. For this reason, the LBM has very good parallel performance. This is important for DNS since the computational cost necessitates the use of parallel computers.

There are, however, a number of limitations to the LBM [12]:

- Though the LBM is restricted to solving incompressible problems ($Ma < 0.3$), it is in fact a compressible method. This means that the density does vary and there are sound waves present in the simulation. Thus, the time step must be made very small in order to resolve the fast-moving sound waves. Methods that solve the incompressible form of the NS equations can use much larger time steps.

- While the LBM allows for density variations, and is therefore a compressible method, it does not have enough degrees of freedom to allow for temperature variations. For this reason, the LBM cannot be compared to methods that solve the fully-compressible NS equations.

- Due to the underlying lattice, the LBM is restricted to a uniform cubic lattice. If the lattice is made non-uniform, or stretched in a particular coordinate direction, an interpolation must be done to determine the values of the single particle distribution functions on the lattice sites. Adding such an interpolation adds computational complexity and reduces the formal order of accuracy of the method.

The LBM combines many of the desirable features of spectral and finite difference methods. The computational efficiency of the LBM is comparable to that of spectral codes (on a per node basis), and the ease of implementing complex boundaries is comparable to finite difference methods.

1.1 Literature Review

The LBM was first applied as an improvement to Lattice Gas Cellular Automata (LGCA) simulations. In this method, fluid flow is modelled by tracking the movement of particles on a discrete lattice. It uses the same streaming and collision processes as the modern LBM, but the collision operator is based on a discrete set of collision rules instead of a relaxation process. This method is able to recover the macroscopic motion of the fluid, but is plagued by noise and is difficult to extend to three dimensions [52]. McNamara and Zanetti [38] proposed replacing the boolean occupation numbers with mean occupation numbers (equivalent to single particle distribution) to correct the problems with noise, but retained the same collision rules as the LGCA. These were the first simulations performed with the LBM. However, since this method used the collision rules from LGCA models, it was not possible to change the viscosity. To solve this problem, Higuera and Jiménez [25] proposed a linear collision operator that contained a few tunable parameters that could be adjusted to decrease the viscosity. The linearised collision operator relaxes the pre-collision particle distributions at a particular site towards their local equilibrium distribution. The relaxation times used are thus related to the transport coefficients such as viscosity and thermal conductivity. A special case of the linearised collision operator utilizes the Bhatnagar-Gross-Krook [8] (BGK) assumption of a single relaxation time, which further increases the computational efficiency of the collision process. The use of the BGK approximation was first proposed independently by Qian [45] and Chen *et al.* [13]. Later, a form of the equilibrium distribution was given by Koelman [33] that resulted in an LBM scheme that worked on arbitrary regular lattices. These distributions eliminated unphysical effects, such as violation of Galilean invariance and pressures that were dependent on the velocity. These advances allowed the extension of the LBM to three dimensions, which was previously difficult for the LGCA models [15]. Eventually, the LBM was established as a numerical method completely independent of the LGCA. Despite its evolution from LGCA models, He and Luo [23] showed that the LBM can be derived directly from the Boltzmann equation by discretising it in both velocity space and physical space. They also demonstrated that the discretisation of the Boltzmann equation in velocity space is completely independent of the discretisation in physical space.

Since the LBM, unlike the LGCA, is capable of altering the viscosity, it is suitable for simulating turbulent flows. One of the first simulations of turbulence using the LBM was performed by Benzi and Succi [4]. They simulated two-dimensional forced isotropic turbulence, and compared the time evolution of total energy and enstrophy, as well as the energy spectra, to the results from a spectral code. The LBM showed good agreement with the results from the spectral code, and recovered the inertial range in the energy spectrum. Furthermore, they compared the computational efficiency of the LBM and the spectral code and found them to be approximately equal. On a per time-step basis, the efficiency of the LBM is greater than the spectral code, but since the LBM is compressible it requires a smaller time step to maintain stability. These two factors were found to offset each other. Chen *et al.* [15] performed a simulation of three-dimensional decaying isotropic turbulence. This simulation was performed on a computational grid with $128 \times 128 \times 128$ lattice sites and periodic boundaries in all three directions. The LBM results were once again compared to results from a spectral code. The initial conditions for the spectral code were determined using a Gaussian random distribution that generated a specific initial energy spectrum. These initial conditions were then transformed from spectral to physical space and used to initialise the particle distributions for the LBM simulation. The energy spectra were then compared after the turbulence was allowed to decay. These results showed good agreement with the spectral code, except for a small discrepancy at high wavenumbers. Luo *et al.* [36] studied three-dimensional decaying isotropic turbulence using the LBM, and compared the energy spectrum as well as the mean kinetic energy and dissipation rate to results from a pseudo-spectral method. They also found a discrepancy between the two methods for high wavenumbers which was attributed to the fact that the LBM is slightly more dissipative than the spectral code because it is only second-order accurate in time and space.

Martinez *et al.* [37] used the LBM to simulate a two-dimensional decaying shear layer and compared the results to those computed using a spectral method. The spectral code used to simulate the shear layer solved the vorticity equation; it was initialised with two spectral representations of delta functions for vorticity at two locations in the domain. Random noise was superimposed according to a pre-defined energy spectrum for the initial state. The initial conditions for the spectral method were transformed from spectral space to physical space in order to be implemented in the LBM simulation. Then, the time evolution of a number of quantities were compared for the two simulations. These quantities included the enstrophy, palinstrophy, and the fourth-order enstrophy. All of these quantities showed good agreement between the LBM and spectral results. The energy spectra were also compared. The findings from the energy spectra were similar to those found by [4], [15], and [36]. In addition to these quantities, Martinez *et al.* compared the vorticity contour plots at selected snapshots in time for both methods. They found that the LBM matched very well to the spectral results early in the simulation, but drifted slightly over time. However, roughly the same vortical structures were observed in both simulations, although the exact positions and times were slightly different. The LBM relaxed to a "sinh - Poisson" state, which is expected for this flow. The differences in vortical structures were explained by errors in reconciling the LBM time scale with the spectral time scale. They also state that these errors are in part due to the compressibility of the LBM simulation. This argument is supported by the fact that the errors between the spectral results (incompressible) and the LBM decrease as the Mach number is decreased in the LBM simulation.

As a result of the success of these early turbulence simulations, the LBM has become a useful tool for direct numerical simulation. Benzi *et al.* [3] used the LBM to study the extended self-similarity of anisotropic turbulence. They were able to use the LBM to study the dependence of the shear rate on the scaling exponents for anisotropic turbulent flows. More recently, Yu and Girimaji [54] studied decaying isotropic turbulence with and without frame rotation, and homogeneous shear turbulence [53]. In both of these papers, the LBM was verified with existing DNS methods and found to be reliable. Djenidi [17] performed an LBM simulation of grid-generated turbulence. While the simulation performed was not large enough to capture the largest scale structures, and the resolution was not sufficient to resolve the smallest scales, Djenidi reported a good agreement with experimental results.

In all of the LBM simulations of turbulent flows discussed above, none included wall boundaries. This is because simulations with periodic or symmetry boundaries are easily implemented in the LBM, and are ideal for comparison with spectral methods. However, for flows with complex geometries, the accuracy of the wall boundary conditions is of primary importance and therefore must be verified. The simplest and most widely studied wall-bounded flow is that of fully-developed turbulent channel flow. The first DNS of turbulent channel flow was computed by Kim, Moser, and Moin in 1987 [32]. They simulated a channel at a Reynolds number of 3300 (based on the mean centerline velocity and the channel half-width) using a fully spectral method to solve the incompressible Navier-Stokes equations. The simulation used Fourier series for the basis functions in the homogeneous direction (stream-wise and span-wise) and Chebychev polynomials for the basis functions in the wall-normal direction. In this simulation, no subgrid scale model was used; the authors took care to ensure that the smallest scales were adequately resolved. Later, Moser *et al.* [41] redid the simulation with a slightly different computational domain and also ran similar simulations for higher Reynolds numbers. These simulations have become the standard database used by the DNS community to evaluate the performance of other DNS techniques for wall-bounded turbulent flows.

In the literature there are a number of simulations of turbulent channel flow using the LBM. The first simulation of fully-developed channel flow was computed by Eggels [18]. This simulation was performed at a Reynolds number significantly lower than the simulation of Moser *et al.* and had limited resolution as well as a limited extent in the homogeneous directions. The simulation was performed with a thermal LBM, and included a temperature gradient across the channel in the wall-normal direction. Despite these differences, the turbulence statistics up to second-order showed good qualitative agreement with the database. Amati *et al.* [1] also performed a simulation of turbulent channel flow, but their analysis was mainly focused on scaling laws (they do not report any statistics). However, in their later work [2], they simulated a channel with a Reynolds number of 3300 and show a reasonable agreement with the database of Moser *et al.* [41]. This simulation is under resolved and also has a limited extent in the homogeneous directions. However, the second-order statistics show good directional agreement with the database results. The largest discrepancy in the LBM results is an over-prediction of the streamwise velocity fluctuations.

Recently, Lammers *et al.* [34] simulated a channel with a much narrower domain than Moser *et al.* [41], but with a fine enough grid resolution to resolve the Kolmogorov scales. The statistics from the LBM simulation showed a good qualitative agreement with the data of Moser *et al.* [41], but there were significant differences between the two simulations. The most significant difference was that the LBM over-predicted the variance of the pressure fluctuations by approximately 25%. Lammers *et al.* suggested that the over-prediction could be caused by spurious pressure fluctuations that are inherent to the LBM, or due to the compressibility of the LBM. However, it is also possible that the over-prediction of the pressure variance is due to narrower domain used in the LBM simulation. The domain used for the LBM simulation was four times narrower in the spanwise direction than the domain used in the simulation of Moser *et al.*. Furthermore, the two-point correlations calculated from the spectral simulation reveal that, while the velocity fluctuations have a long correlation length in the streamwise direction, the pressure field is highly correlated in the spanwise direction. In fact, the domain used by Moser *et al.* is only just wide enough for the spanwise two-point correlations to decay to zero within half of the domain. Since the computational domain for the LBM simulation performed by Lammers *et al.* was not the same as the domain used by Moser *et al.*, it is impossible to separate errors that are inherent to the LBM from errors caused by the narrower computational domain.

1.2 Objective

The objective of the present work was to validate the LBM and assess its suitability for DNS of incompressible wall-bounded turbulent flows. Before the LBM can be used in a predictive capacity, it must be rigorously validated against existing DNS databases. As stated in the previous section, there have been a number of studies in the literature that have attempted to validate the LBM for DNS of turbulent flows. The findings of these studies have been promising for free shear flows, such as homogeneous isotropic turbulence, but the LBM has not yet been adequately validated for wall-bounded turbulence.

In this work, an LBM code was developed and used to simulate fully-developed turbulent channel flow. The results were then compared to those from a conventional Navier-Stokes-based finite difference (FD) simulation. The parameters for the LBM and FD simulations were based on the $Re_\tau = 180$ spectral simulation of Moser *et al.* [41]. The LBM results were compared to those from the FD simulation instead of the spectral simulation of Moser *et al.* for a number of reasons. First, since the FD method is second-order accurate in space and time, its formal order of accuracy is comparable to the LBM than the spectral method, which does not suffer from truncation error. Second, running the FD simulation with an "in house" code allowed for the study of statistics that are not freely available from the database of Moser *et al.*. This allowed for a more complete analysis of the results.

Unlike the simulation of Lammers *et al.* [34], special care was taken to ensure that the dimensions of the computational domains for both simulations were identical. This removed the influence of the domain size on the turbulence statistics and allowed for a proper validation of the LBM.

1.3 Contributions

This study is the definitive validation of the LBM for direct numerical simulation of wall-bounded turbulent flows. Other validations of the LBM for wall-bounded flows in the literature ([19], [34], and [44]) have either used computational domains that were too small, or did not have sufficient grid resolution to be considered a DNS.

In this work, the results from a well-resolved LBM DNS of turbulent channel flow were compared to those from a conventional Navier-Stokes finite difference simulation with an identical computational domain. Thus, any difference in the turbulence statistics obtained can be attributed to differences in the methods themselves. This is the first time that the turbulence statistics for an LBM simulation of a wall-bounded turbulent flow have been rigorously compared to the results from a conventional Navier-Stokes solver.

This study also provides insight into the performance of LBM codes on shared memory machines. An LBM code was developed and parallelized with OpenMP so that it could be run in parallel on a shared memory machine. This work discusses the implementation issues associated with parallelizing the LBM and presents the resulting parallel performance.

Finally, by comparing the performance of the LBM and FD simulations, this work examines the suitability of the LBM for DNS of wall-bounded turbulent flows. This includes not only an analysis of the accuracy of the LBM, but a comparison of the computational efficiency and the ease of implementation for both the LBM and FD methods. While it is difficult to extend the results from performance studies to other problems, this work provides a benchmark for the performance that can be expected from the LBM for simple wall-bounded turbulent flows.

1.4 Thesis Layout

This thesis is divided into seven chapters. The second chapter gives a brief introduction to kinetic theory. This chapter is intended to give the reader the required background in kinetic theory to understand the origin of the important features of the LBM. The third chapter outlines the implementation of the LBM code used in this work, and presents the verification results for Poiseuille and duct flow. The fourth chapter discusses the parallel implementation of the LBM code and the achieved parallel performance. The fifth chapter outlines the methodology used to validate the LBM, and the sixth chapter discusses the results. The final chapter states the conclusions drawn from this study and makes recommendations for future work.

Chapter 2

Theory

In the field of computational fluid dynamics (CFD), the majority of the simulation methods used are based on the Navier-Stokes equations, which are the governing equations for continuum fluid dynamics. In contrast, the lattice Boltzmann method is a discrete-particle-based method that solves for approximate solutions to the Boltzmann equation. Thus, in order to work with the LBM, a background in kinetic theory is invaluable.

There are many textbooks and academic papers available on the topic of kinetic theory as well as a large number on the LBM. However, most texts concerning kinetic theory do not cover the LBM, and texts on the LBM often give little background on the details of kinetic theory and the Boltzmann equation. Therefore, the purpose of this chapter is to cover most, if not all, of the important results in kinetic theory that form the backbone of the lattice Boltzmann method. For the most part, rigorous derivations of the equations presented in this chapter are beyond the scope of this work, but the physical reasoning and the assumptions used are clearly stated. References to helpful texts are given for the interested reader.

The first section of this chapter defines kinetic theory and demonstrates how problems that can be solved using continuum methods can also be solved using discrete particle techniques. The next section derives the Boltzmann equation starting from the Hamiltonian equations that describe the dynamics of an N-body system. This derivation shows the systematic generalisation of the N-body problem to the Boltzmann equation in order to reduce the computational complexity. Then, a multiscale expansion is applied to the Boltzmann equation to show that it is equivalent to the Navier-Stokes equations for continuum fluid mechanics.

2.1 Kinetic Theory of Gases

Depending on the detail required, fluid motion can be described using a number of different models. For most cases of engineering interest, the fluid is assumed to behave as a continuum and the Navier-Stokes equations are used as the governing equations. The continuum assumption is valid when the macroscopic properties of interest (pressure, velocity, temperature, etc.) are insensitive to the exact position and velocity of the individual particles that make up the fluid. However, if this assumption breaks down, it becomes necessary to either statistically model the motion of the particles, or directly model interactions between molecules, atoms, and even electrons. Figure 2.1 illustrates a number of models of fluid motion and gives the names of their corresponding governing equations.

As the level of detail increases, so does the computational complexity. For this reason, the model used must be chosen carefully to achieve the required level of accuracy with the minimum computational cost. Therefore, it is important to understand the assumptions of each of the models and the conditions that must be met in order to apply them properly. The dimensionless parameter that determines the limits of

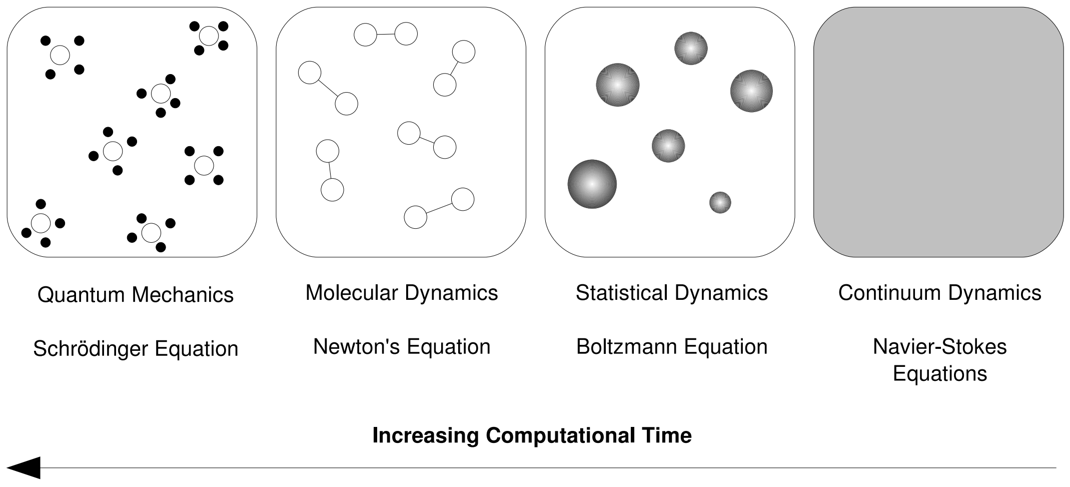


Figure 2.1: Various models of fluid motion and their governing equations.

use for the various fluid models is the Knudsen number (Kn). The Knudsen number is defined as the ratio of the mean free path (λ) to the smallest characteristic flow dimension (l):

$$Kn = \frac{\lambda}{l} \ . \tag{2.1}$$

As illustrated in figure 2.2, continuum models are only valid in the low Knudsen number range [9]. This is because, at low Knudsen numbers, the mean free path (average distance travelled between collisions), is insignificant compared to the characteristic scales in the flow. Therefore, the individual particles have little effect on the macroscopic quantities, and the fluid acts as a continuum. As the Knudsen number increases, the distance between individual particles becomes significant, and the continuum assumption is no longer valid. In this range, discrete-particle-based methods must be used because the conservation equations for continuum models do not form a closed set.

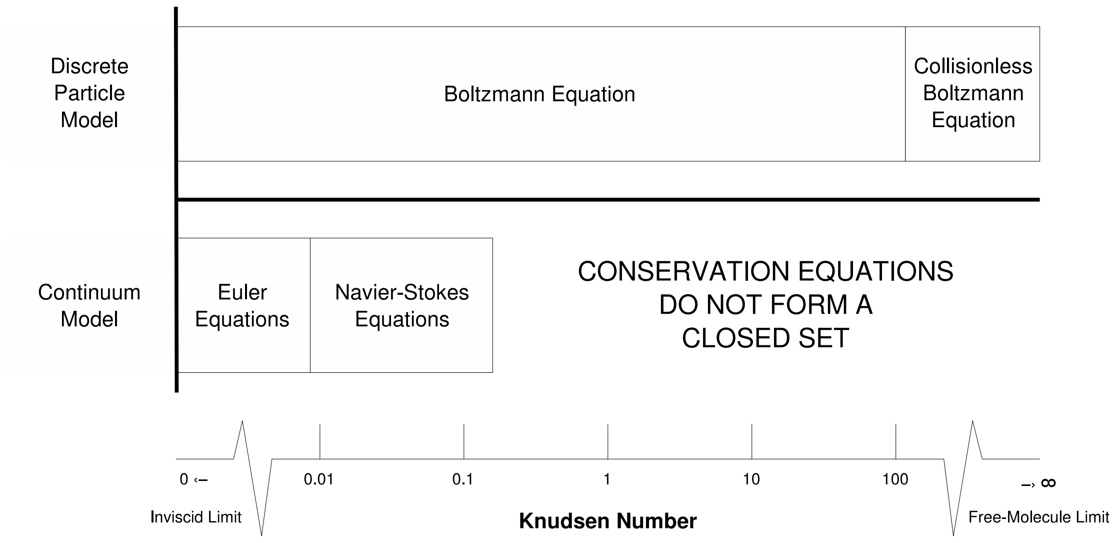


Figure 2.2: Knudsen number ranges for different governing equations of fluid motion. Adapted from [9].

For turbulent flows, the smallest characteristic dimension is the Kolmogorov length scale (η). Since the Kolmogorov length scale is a function of the Reynolds number and the kinematic viscosity of a gas is proportional to the product of the speed of sound and the mean free path, the Knudsen number for a turbulent flow can be expressed in terms of the Mach number and the Reynolds number [49]:

$$\frac{\lambda}{\eta} \approx \frac{Ma}{Re^{1/4}} \quad . \tag{2.2}$$

Incompressible turbulent flows are characterised by low Mach numbers and high Reynolds numbers. Thus, despite the small scales present in a high-Reynolds-number flow, turbulence is a continuum phenomena and can therefore be accurately described by the Navier-Stokes equations. However, as will be shown in section 2.9, the Boltzmann equation is also valid in the continuum region. Therefore, it is possible to describe the motion of a turbulent flow using a discrete-particle-based method. Generally, discrete particle models are associated with a large computational cost due to their large number of degrees of freedom. For this reason, these methods are generally restricted to rarefied gases. However, as shown in the following sections, it is possible to reduce the number of degrees of freedom of discrete-particle-based methods to the point where their computational cost is comparable to conventional continuum methods.

2.2 Molecular Dynamics

In molecular dynamics the fluid is modelled as a large number of discrete particles. The particles are simply point masses with no orientation information and thus they do not have rotational or vibrational energy. The dynamics of the system can be described by the governing equations for a Hamiltonian system, which are simply a reformulation of classical mechanics (Newton's equations) [22]:

$$\frac{\partial H}{\partial \vec{x}_i} = -\frac{d\vec{p}_i}{dt} \ , \tag{2.3}$$

$$\frac{\partial H}{\partial \vec{p}_i} = \frac{d\vec{x}_i}{dt} \ , \tag{2.4}$$

$$\frac{\partial H}{\partial t} = -\frac{\partial L}{\partial t} \quad . \tag{2.5}$$

In equations 2.3 - 2.5, $\vec{x}_i$ and $\vec{p}_i$ are the position and momentum of the i^{th} particle, respectively. The Hamiltonian (H) is related to the Lagrangian (L) and can be shown to be equal to the sum of the total kinetic and potential energy of the system. For a system containing N particles, $6N$ first-order differential equations must be solved to obtain the dynamics of the system.

The advantage of the molecular dynamics approach is that the governing equations are simple and easy to solve. However, for most systems of engineering interest, the number of particles is on the order of Avagadro's Number (6.02×10^{23}). For a system this large, the computational requirements are enormous. So, the molecular dynamics model is not feasible for modelling continuum flows. However, the amount of detail provided by this model exceeds that which is required for a continuum flow since the macroscopic state is not sensitive to individual particle positions and velocities. Therefore, it should be possible to remove some of the computational complexity of this method while still recovering the correct macroscopic conservation equations.

2.3 The Liouville Equation

To completely describe the microscopic state of a system, $3N$ positions, and $3N$ momenta are required. In other words, the instantaneous microscopic state of a system can be described as a point in $6N$ dimensional "phase space". If the microscopic state is known, then it is possible to calculate a corresponding macroscopic state. However, since the number of degrees of freedom in the microscopic description is much higher than for the macroscopic description, a macroscopic state does not have a unique microscopic equivalent. In other words, for a given macroscopic state, there are an infinite number of equivalent microscopic states. If all of these equivalent states are plotted in phase space, the microscopic state can be expressed as a probability density function in phase space. Thus, the N-particle probability density is introduced:

$$F_N(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N, \vec{p}_1, \vec{p}_2, \ldots, \vec{p}_N, t) \ . \tag{2.6}$$

The N-particle probability density function represents the probability that the state of the system is located at a particular point in phase space at time t. In other words, it is the probability that particle 1 is located at $\vec{x}_1$ with momentum $\vec{p}_1$, and particle 2 is located at $\vec{x}_2$ with momentum $\vec{p}_2$, and so on. Instead of solving for the motion of each individual particle, the problem is generalised to solving for the evolution of the N-particle probability density function.

To determine the governing equation for the N-particle probability density function, the expression for a change in F_N at a point in phase space is written [22]:

$$dF_N = \frac{\partial F_N}{\partial t} dt + \sum_{i=1}^{N} \frac{\partial F_N}{\partial \vec{x}_i} d\vec{x}_i + \sum_{i=1}^{N} \frac{\partial F_N}{\partial \vec{p}_i} d\vec{p}_i \ . \tag{2.7}$$

Next, equation 2.7 is rewritten to represent the change in F_N along a system trajectory in phase space. Since F_N is constant in time along a trajectory in phase space, the total derivative of F_N is set to zero. This gives the Liouville equation: the governing equation for the N-particle probability density function [22]:

$$\frac{\partial F_N}{\partial t} + \sum_{i=1}^{N} \left[\frac{\partial F_N}{\partial \vec{x}_i} \cdot \frac{d\vec{x}_i}{dt} + \frac{\partial F_N}{\partial \vec{p}_i} \cdot \frac{d\vec{p}_i}{dt} \right] = 0 \ . \tag{2.8}$$

It is important to note that, in the Liouville equation, there is no underlying model for interparticle forces. The interparticle force model can be inserted into the equation by replacing $d\vec{p}_i/dt$ by a model for the forces. The model for the forces is generally based on a potential that is a function of the interparticle radius (ϕ_{ij}):

$$\frac{\partial F_N}{\partial t} + \sum_{i=1}^{N} \left[\frac{\vec{p}_i}{m} \cdot \frac{\partial F_N}{\partial \vec{x}_i} + \left(- \sum_{j \neq i}^{N} \frac{\partial \phi_{ij}}{\partial \vec{x}_i} \right) \cdot \frac{\partial F_N}{\partial \vec{p}_i} \right] = 0 \ . \tag{2.9}$$

2.4 Reduced Probability Distribution Functions

At this point, the molecular dynamics approach has been generalised to a single partial differential equation for the N-particle probability distribution function. This generalisation makes the governing equations easier to deal with, but the system still has an enormous number of degrees of freedom and so requires a large computational effort. The next step to reduce the computational complexity is to write the governing equation in terms of the single-particle probability distribution instead of the

N-particle distribution.

The N-particle probability distribution represents the probability that particle 1 is located at $\vec{x}_1$ with momentum $\vec{p}_1$, and particle 2 is located at $\vec{x}_2$ with momentum $\vec{p}_2$, and so on. The single-particle probability distribution represents the probability that particle 1 is located at $\vec{x}_1$ with momentum $\vec{p}_1$, but includes all of the other possible configurations of the remaining $N-1$ particles. Therefore, to calculate a general R particle distribution function, the N-particle distribution must be integrated over all possible configurations of the remaining $N-R$ particles. Thus, the following expression for the reduced distribution function can be written [22]:

$$F_R(\vec{x}_1, \vec{p}_1, \ldots, \vec{x}_R, \vec{p}_R) = \int F_N d\vec{x}_{R+1} d\vec{p}_{R+1} \ldots d\vec{x}_N d\vec{p}_N \quad . \tag{2.10}$$

The single-particle distribution (F_1) can be calculated using equation 2.10. The next step is to find the governing equation for F_1. This can be done by performing the integration in equation 2.10 to the Liouville equation. After some algebra, the BBGKY hierarchy equation (named for Bogoliubov, Born, Green, Kirkwood, and Yvon) is obtained for the single-particle distribution function [22]:

$$\frac{\partial F_1}{\partial t} + \frac{\vec{p}_1}{m} \cdot \frac{\partial F}{\partial \vec{x}_1} = (N-1) \int \frac{\partial \phi_{12}}{\partial \vec{x}_1} \cdot \frac{\partial F_2}{\partial \vec{p}_1} d\vec{x}_2 \quad . \tag{2.11}$$

For more details on the derivation of this equation, please see [22]. The problem with this equation is that it is not closed. The evolution of the single-particle distribution function depends on the two-particle distribution, and in turn, the two-particle distribution depends on the three particle distribution. However, the single-particle distribution has only 6 degrees of freedom compared to $6N$ for the N-particle distribution function. So, if a suitable method of closing this equation could be found, the computational complexity would be dramatically reduced.

2.5 The Boltzmann Equation

In the previous section, the reduced distribution function was introduced and the BBGKY hierarchy was used to derive the governing equations for the single-particle distribution function. However, this equation cannot be solved without the knowledge of the full N-particle distribution function. In this section, an *ad hoc* derivation will be performed to close the governing equation for the single-particle probability distribution function.

The derivation starts with the BBGKY equation for the single-particle distribution. The left side of this equation is simply the Liouville equation in the limit of zero interactions [35]:

$$\frac{\partial F_1}{\partial t} + \vec{c} \cdot \frac{\partial F_1}{\partial \vec{x}} = 0 \ . \tag{2.12}$$

The physical interpretation of this equation is that, in the absence of collisions, the rate of entry of particles into the phase volume $\delta\vec{x}\delta\vec{c}$ is zero. If particle interactions are included, this rate is no longer zero, and the following expression can be written for the number of particles that enter $\delta\vec{x}\delta\vec{c}$ over a time δt [35]:

$$\left(\frac{\partial F_1}{\partial t} + \vec{c}\cdot\frac{\partial F_1}{\partial \vec{x}}\right)\delta\vec{x}\delta\vec{c}\delta t = \delta n \ . \tag{2.13}$$

The number of particles entering the phase volume as a result of collisions can be separated into two groups: the particles that leave the phase volume due to collisions with other particles (δn_-), and the particles that enter the phase volume as a result of a collision (δn_+). The number of particles leaving the phase volume is equal to the sum of the pairs of particles in which one particle is in the phase volume $\delta\vec{x}\delta\vec{c}$ and the other is in the phase volume $\delta\vec{x}_1\delta\vec{c}_1$ such that there will be a collision in time δt. The total number of particle pairs that satisfy this requirement is [35]:

$$\delta n_- = N\int F_2(\vec{x},\vec{x}_1,\vec{c},\vec{c}_1)d\vec{x}_1 d\vec{c}_1 d\vec{x} d\vec{c} \ , \tag{2.14}$$

where the probability density distributions are written as functions of velocity instead of momentum.

If the particles are modelled as identical hard spheres, an elemental volume in physical space ($\delta\vec{x}_1$) can be defined in which particles will collide with those in $\delta\vec{x}\delta\vec{c}$. If the diameter of the particles is σ, then $\delta\vec{x}_1$ is defined by the elemental volume in which particles (on their pre-collision trajectories) will enter the interaction zone that consists of a spherical volume around $(\vec{x}, \vec{c})$ with a radius σ. A sketch of $\delta\vec{x}_1$ is shown in figure 2.3. In this figure, the elemental volume is defined by the relative velocity between the interacting particles and the interaction cross-sectional area. The expression for $\delta\vec{x}_1$ is written below [35]:

$$\delta\vec{x}_1 = |\vec{c}_1 - \vec{c}|\, dtsdsd\theta \ . \tag{2.15}$$

The expression for $\delta\vec{x}_1$ can be substituted into equation 2.14 to get a new expression for δn_-:

$$\delta n_- = \left(N \int F_2(\vec{x}, \vec{x}_1, \vec{c}, \vec{c}_1) d\vec{c}_1 \, |\vec{c}_1 - \vec{c}| \, sdsd\theta \right) d\vec{x} d\vec{c} dt \ . \tag{2.16}$$

The next step is to find an expression for the number of particles that enter $\delta\vec{x}\delta\vec{c}$ as a result of collisions. However, this expression is related to equation 2.14 since it involves the inverse collision (as illustrated in figure 2.4). In the reverse collision, two particles with the same post-collision states as the forward collision collide to recover the pre-collision states. So, the expression for δn_+ becomes [35]:

$$\delta n_+ = N \int F_2(\tilde{\vec{x}}, \tilde{\vec{x}}_1, \tilde{\vec{c}}, \tilde{\vec{c}}_1) d\tilde{\vec{x}}_1 d\tilde{\vec{c}}_1 d\tilde{\vec{x}} d\tilde{\vec{c}} \ , \tag{2.17}$$

where the quantities with tildes in equation 2.17 represent post-collision values.

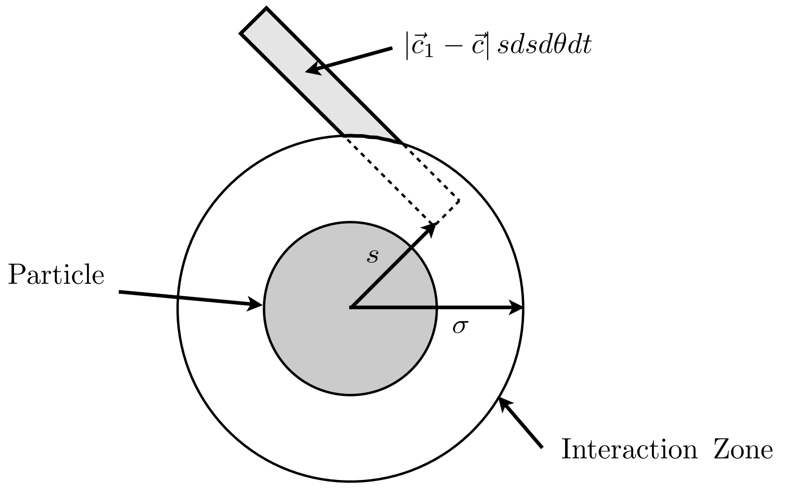


Figure 2.3: Scattering diagram

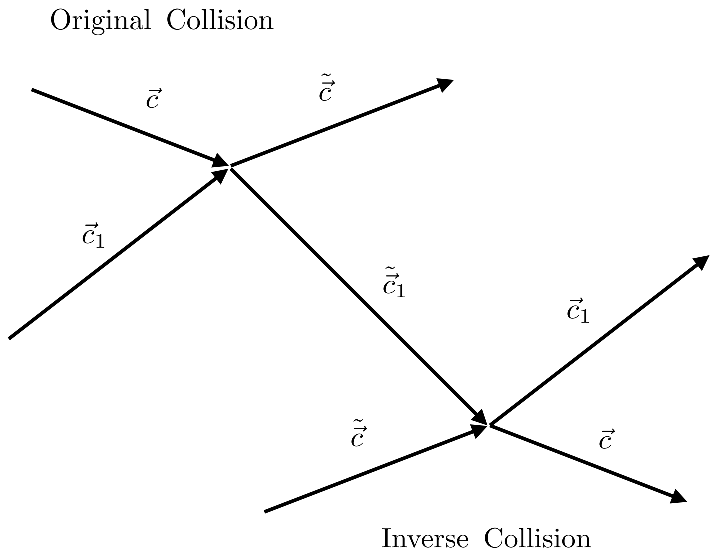


Figure 2.4: Original collision and its inverse.

Since the transformation of $\vec{x}$ to $\tilde{\vec{x}}$ and $\vec{c}$ to $\tilde{\vec{c}}$ is canonical, the following relationship can be proved [35]:

$$d\tilde{\vec{x}}_1 d\tilde{\vec{c}}_1 d\tilde{\vec{x}} d\tilde{\vec{c}} = d\vec{x}_1 d\vec{c}_1 d\vec{x} d\vec{c} \ . \tag{2.18}$$

With the above relationship, equation 2.17 can be written in the same form as equation 2.16. These two equations can be summed to get the net rate of particles entering the phase volume $\delta\vec{x}\delta\vec{c}$. This expression can then be substituted for δn in equation 2.13 [35]:

$$\frac{\partial F_1}{\partial t} + \vec{c} \cdot \frac{\partial F_1}{\partial \vec{x}} = N \int \left[F_2(\tilde{\vec{x}}, \tilde{\vec{c}}, \tilde{\vec{x}}_1, \tilde{\vec{c}}_1) - F_2(\vec{x}, \vec{c}, \vec{x}_1, \vec{c}_1) \right] d\vec{c}_1 \, |\vec{c}_1 - \vec{c}| \, s ds d\theta \ . \tag{2.19}$$

The resulting equation is the governing equation for the single-particle distribution. It is fundamentally equivalent to the BBGKY equation (equation 2.11), but has a simpler form for the right-hand side due to the assumption of hard sphere particles and binary collisions. Despite this simplification, this equation is still a function of the two-particle probability density distribution, and is therefore not closed. To close this equation, two assumptions must be made. The first assumption is that F_2 is homogeneous over the collision domain (in physical space). With this assumption, the two-particle distribution can be written as a function of velocity only [35]:

$$F_2(\vec{x}, \vec{c}, \vec{x}_1, \vec{c}_1) = F_2(\vec{c}, \vec{c}_1) \ , \tag{2.20}$$

$$F_2(\tilde{\vec{x}}, \tilde{\vec{c}}, \tilde{\vec{x}}_1, \tilde{\vec{c}}_1) = F_2(\tilde{\vec{c}}, \tilde{\vec{c}}_1) \ . \tag{2.21}$$

The second assumption is that the trajectories of the particles are uncorrelated before and after the collision (*molecular chaos*). If the particle trajectories are uncorrelated, the two-particle probability distribution function can be rewritten as a product of two mutually exclusive single-particle distributions:

$$F_2(\vec{c}, \vec{c}_1) = F_1(\vec{c})F_1(\vec{c}_1) \ , \tag{2.22}$$

$$F_2(\tilde{\vec{c}}, \tilde{\vec{c}}_1) = F_1(\tilde{\vec{c}})F_1(\tilde{\vec{c}}_1) \ . \tag{2.23}$$

Finally, the substitution $sdsd\theta = \sigma d\Omega$ is made, and equation 2.19 is rewritten in terms of the single-particle mass density f. This gives the celebrated Boltzmann equation [35]:

$$\frac{\partial f}{\partial t} + \vec{c} \cdot \nabla f = \frac{1}{m} \int \sigma \left|\vec{c}_1 - \vec{c}\right| \left[f(\tilde{\vec{c}}_1) f(\tilde{\vec{c}}) - f(\vec{c}_1) f(\vec{c}) \right] d\Omega d\vec{c}_1 \ , \tag{2.24}$$

where:

$$f = NmF_1 \ . \tag{2.25}$$

Thus, the Boltzmann equation is the governing equation for the single-particle mass density. As mentioned, this equation originates from the BBGKY equation, but uses a number of assumptions to remove the dependence on F_2. To summarise, the following are the major assumptions made in the Boltzmann equation [35]:

1. The particles are modelled as hard spheres. This means that the distance over which the particles interact is equal to the diameter of the particles, which is much smaller than the mean free path.

2. Only binary collisions (collisions involving two particles), are considered. This restricts the Boltzmann equation (in this form) to gases with low densities such that the probability of a multiparticle collision is extremely low.

3. The trajectories of colliding particles are assumed to be rectilinear before and after the collision (as depicted in figure 2.4). This assumption is called the *stosszahlansatz*.

4. $F_2(\vec{x}, \vec{c}, \vec{x}_1, \vec{c}_1)$ is assumed to be homogeneous over the collision domain (in physical space), and thus can be written as a function of $\vec{c}$ and $\vec{c}_1$ only.

5. Finally, the assumption of molecular chaos permits the replacement of the two-particle distribution with the product of two mutually exclusive single-particle distributions.

The Boltzmann equation solves the problem of closure of the governing equation for the single-particle distribution function. This reduces the number of degrees of freedom from $6N$ to only 6.

2.6 Moments of the Single-Particle Distribution

In the previous section, the Boltzmann equation was derived as the governing equation for the single-particle distribution. However, since the end goal is to study continuum flows, the calculation of the single-particle distribution is only a means of determining the corresponding macroscopic state. The single-particle distribution, $f(\vec{x}, \vec{c}, t)$, represents the expected mass density of particles located at position $\vec{x}$, with a velocity $\vec{c}$ at time t. This can be interpreted as an ensemble average of all possible microscopic states of the system that correspond to the same macroscopic state. Therefore, from the single-particle distribution, the macroscopic state can be computed at location $\vec{x}$. The procedure for computing a number of macroscopic quantities (moments), from f will be examined in this section.

The zeroth moment of the single-particle distribution is simply the integral of $f(\vec{x}, \vec{c}, t)$ over the full range of its velocity argument. Since f represents the expected mass density of particles at $\vec{x}$ moving with velocity $\vec{c}$, the integral of f over velocity space is equal to the total expected mass density of particles at $\vec{x}$ regardless of their velocity. This is equal to the macroscopic fluid density [22]:

$$\rho = \int f d\vec{c} \ . \tag{2.26}$$

Using this same principle, the first moment can be defined. The first moment is the integral over velocity space of the product of f and the particle velocity. This gives the macroscopic fluid momentum at $\vec{x}$ and time t [22]:

$$\rho\vec{u} = \int \vec{c} f d\vec{c} \ . \tag{2.27}$$

The second moment of the single-particle distribution is related to the total expected energy of the particles. The kinetic energy per unit mass of a particle is equal to $|\vec{c}|^2/2$, so the expression for the total energy can be written as follows [22]:

$$E = \frac{1}{\rho} \int \frac{|\vec{c}|^2}{2} f d\vec{c} \ . \tag{2.28}$$

The total energy E is composed of the internal energy of the fluid and the kinetic energy of the bulk fluid motion. To separate these two types of energy, the particle velocity with respect to the bulk velocity must be found. This is called the the peculiar velocity ($\vec{c}_0$), and is given by the following expression:

$$\vec{c}_0 = \vec{c} - \vec{u} \ . \tag{2.29}$$

The internal energy of the fluid is only due to the peculiar velocity of the particles; it can be calculated by substituting $\vec{c}_0$ for the total particle velocity in equation 2.28 [22]:

$$e = \frac{1}{\rho} \int \frac{|\vec{c}_0|^2}{2} f d\vec{c} \ . \tag{2.30}$$

The equipartition theorem states that the total internal energy of a system of particles is equally distributed over all of the degrees of freedom, and the contribution of each degree of freedom to the internal energy is equal to $k_B T/2$. Therefore, since the particles in the Boltzmann model only have translational energy, the total internal energy per unit mass is equal to [22]:

$$e = \frac{3}{2} RT \ . \tag{2.31}$$

The total stress tensor (which includes both pressure and viscous stresses) can also be expressed as a moment of the single-particle distribution by performing a momentum balance. If an infinitesimal surface is defined around the location $\vec{x}$ that moves with the fluid, the total stress is equal to the momentum flux leaving the surface. Since the surface is moving with the fluid, the momentum flux is calculated using the peculiar velocity [22]:

$$\mathsf{P} = \int \vec{c}_0 \vec{c}_0 f d\vec{c} \ . \tag{2.32}$$

Clearly, from this definition, the total stress tensor is symmetric ($\mathsf{P}_{\alpha\beta} = \mathsf{P}_{\beta\alpha}$). Additionally, the sum of the diagonal components can be used to define the pressure:

$$\begin{aligned} p &= \frac{1}{3} \int |\vec{c}_0|^2 f d\vec{c} \\ &= \rho R T \ . \end{aligned} \tag{2.33}$$

Thus, the gas modelled by the Boltzmann equation obeys the ideal gas law for a monatomic gas. This equation of state is built directly into the model, and thus does not need to be specified as a separate equation.

The same principle can be used to define the heat flux vector. The heat flux is equal to the flux of internal energy that passes through a infinitesimal surface that moves with the fluid. Therefore, the heat flux can be written as a function of the single-particle distribution and the peculiar velocity [22]:

$$\vec{Q} = \int \frac{|\vec{c}_0|^2}{2} \vec{c}_0 f d\vec{c} \ . \tag{2.34}$$

Since the density, momentum, energy, total stress, and heat flux can all be written as functions of the single-particle mass distribution, all of the quantities used in the macroscopic conservation equations can be found from the microscopic description. Naturally, the next step is to determine whether the governing equation for the microscopic description, the Boltzmann equation, recovers the proper macroscopic conservation laws. The macroscopic conservation laws can be determined by computing the moments of the Boltzmann equation that correspond to mass, momentum, and energy, respectively:

$$\int \frac{\partial f}{\partial t} d\vec{c} + \int \vec{c} \cdot \nabla f d\vec{c} = \int J d\vec{c} \ , \tag{2.35}$$

$$\int \vec{c} \frac{\partial f}{\partial t} d\vec{c} + \int \vec{c}(\vec{c} \cdot \nabla f) d\vec{c} = \int \vec{c} J d\vec{c} \ , \tag{2.36}$$

and

$$\int |\vec{c}|^2 \frac{\partial f}{\partial t} d\vec{c} + \int |\vec{c}|^2 (\vec{c} \cdot \nabla f) d\vec{c} = \int |\vec{c}|^2 J d\vec{c} \ , \tag{2.37}$$

where J is the collision integral, which is the right-hand side of equation 2.24. All of the integrals of the collision integral are zero since they are the moments of the conserved quantities (this will be demonstrated in section 2.7). The other integrals can be broken down into the moments defined above to give the macroscopic equations for conservation of mass, momentum, and energy, respectively [22]:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \ , \tag{2.38}$$

$$\frac{\partial \rho \vec{u}}{\partial t} + \nabla \cdot (\rho \vec{u} \vec{u}) = -\nabla \cdot \mathsf{P} \ , \tag{2.39}$$

and

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho E \vec{u}) = -\nabla \cdot (\vec{u} \cdot \mathsf{P}) - \nabla \cdot \vec{Q} \ . \tag{2.40}$$

In the above equations, the details of the total stress and the heat flux are not specified. This is because these quantities are dependent on the Knudsen number of the flow being studied. These terms will be studied in detail in section 2.9.

2.7 Properties of the Collision Integral and Boltzmann's $\mathcal{H}$-Theorem

Through the detailed analysis of binary collisions of hard spheres, it can be shown that the collision operator J has the following symmetry property [22]:

$$\begin{aligned}
\int \psi(\vec{c}) J d\vec{c} &= \frac{1}{m} \int \sigma \left|\vec{c}_1 - \vec{c}\right| \left[\tilde{f}_1 \tilde{f} - f_1 f\right] \psi(\vec{c}) d\Omega d\vec{c} d\vec{c}_1 \\
&= \frac{1}{m} \int \sigma \left|\vec{c}_1 - \vec{c}\right| \left[\tilde{f}_1 \tilde{f} - f_1 f\right] \psi(\vec{c}_1) d\Omega d\vec{c} d\vec{c}_1 \\
&= \frac{1}{m} \int \sigma \left|\vec{c}_1 - \vec{c}\right| \left[\tilde{f}_1 \tilde{f} - f_1 f\right] \psi(\vec{c}') d\Omega d\vec{c} d\vec{c}_1 \\
&= \frac{1}{m} \int \sigma \left|\vec{c}_1 - \vec{c}\right| \left[\tilde{f}_1 \tilde{f} - f_1 f\right] \psi(\vec{c}_1') d\Omega d\vec{c} d\vec{c}_1 \ . \qquad (2.41)
\end{aligned}$$

In the above equation, ψ is an arbitrary function of either $\vec{c}$, $\vec{c}_1$, $\tilde{\vec{c}}$, or $\tilde{\vec{c}}_1$. Using equation 2.41, the following simplification can be made for integrals of the product of the collision integral and an arbitrary function ψ [22]:

$$\int \psi(\vec{c}) J d\vec{c} = \frac{1}{4} \int \left[\psi(\vec{c}) + \psi(\vec{c}_1) - \psi(\tilde{\vec{c}}) - \psi(\tilde{\vec{c}}_1)\right] J d\vec{c} \ . \qquad (2.42)$$

This expression leads to the next property of the collision integral. If the arbitrary function ψ is equal to one of the five collision invariants (per unit mass) for a binary collision, namely mass (1), momentum ($\vec{c}$), or energy ($|\vec{c}|^2$), the integral over the collision operator vanishes. In other words [22]:

$$\int J d\vec{c} = \frac{1}{4}\int [1 + 1 - 1 - 1]\, J d\vec{c} = 0 \ , \tag{2.43}$$

$$\int \vec{c} J d\vec{c} = \frac{1}{4}\int \left[\vec{c} + \vec{c}_1 - \tilde{\vec{c}} - \tilde{\vec{c}}_1\right] J d\vec{c} = 0 \ , \tag{2.44}$$

$$\int |\vec{c}|^2 J d\vec{c} = \frac{1}{4}\int J(f)\left[|\vec{c}|^2 + |\vec{c}_1|^2 - |\tilde{\vec{c}}|^2 - |\tilde{\vec{c}}_1|^2\right] J d\vec{c} = 0 \ . \tag{2.45}$$

Next, these properties can be used to study the equilibrium behaviour of the collision integral. Boltzmann defined a function for the quantity $\mathcal{H}$, which represents the negative of the probability of the system being in the state defined by the single-particle distribution [10]:

$$\mathcal{H} = \int f \ln f d\vec{c} \ . \tag{2.46}$$

For a spatially homogeneous system ($f(\vec{x},\vec{c}) = f(\vec{c})$), this expression can be substituted into the Boltzmann equation to get:

$$\frac{\partial \mathcal{H}}{\partial t} = \int \left[1 - \ln f\right] J d\vec{c} \ . \tag{2.47}$$

Since $1 - \ln f$ is an arbitrary function of either $\vec{c}$, $\vec{c}_1$, $\tilde{\vec{c}}$, or $\tilde{\vec{c}}_1$, equation 2.47 can be simplified according to equation 2.42 [52]:

$$\frac{\partial \mathcal{H}}{\partial t} = \int \sigma \left|\vec{c}_1 - \vec{c}\right| \left[\tilde{f}_1 \tilde{f} - f_1 f\right] \left[\ln(f f_1) - \ln(\tilde{f}\tilde{f}_1)\right] d\Omega d\vec{c} d\vec{c}_1 \ . \tag{2.48}$$

This integral can never be positive due to the relation [52]:

$$(b-a)(\ln a - \ln b) > 0, \quad \text{for} \quad a \neq b > 0 \ . \tag{2.49}$$

This means that the quantity $\mathcal{H}$ always decreases, but it vanishes when

$$\left[f(\tilde{\vec{c}}_1) f(\tilde{\vec{c}}) - f(\vec{c}_1) f(\vec{c})\right] = 0 \ . \tag{2.50}$$

Therefore, at equilibrium (in other words the most probable state of the system) the single-particle distribution must satisfy the following condition:

$$\ln f(\tilde{\vec{c}}) + \ln f(\tilde{\vec{c}}_1) = \ln f(\vec{c}) + \ln f(\vec{c}_1) \ . \tag{2.51}$$

This means that $\ln(f(\vec{c}))$ must be a linear combination of the collision invariants for the above equation to hold true (see equation 2.42). Using this fact, the local equilibrium distribution for f that satisfies Boltzmann's $\mathcal{H}$-Theorem can be determined. This is the famous Maxwell-Boltzmann distribution, or the local Maxwellian [52]:

$$f^{(eq)} = \left(\frac{\rho}{(2\pi RT)^{3/2}}\right) e^{\left(\frac{-(\vec{c}-\vec{u})^2}{2RT}\right)} \ . \tag{2.52}$$

In summary, the Maxwell-Boltzmann distribution is the local equilibrium solution to the Boltzmann equation. It is the most probable state of the system; therefore successive collisions will cause the system to move towards this distribution. Once a system reaches equilibrium, the collision integral is zero, and so it tends to remain in equilibrium in the absence of external forces.

From the definition of the quantity $\mathcal{H}$, its value can be computed for the local Maxwellian distribution [22]:

$$\mathcal{H}(f^{(eq)}) = \int f^{(eq)} \ln f^{(eq)} d\vec{c} = -\rho \frac{S}{R} + C \ . \tag{2.53}$$

In this equation S is the thermodynamic entropy density of an ideal gas, and C is a constant. Therefore, this equation demonstrates that a decrease in $\mathcal{H}$ is equivalent to an increase in entropy. The implication of this statement is that the Boltzmann equation is irreversible. This point has been a source of controversy due to the fact that the Boltzmann equation is based on the equations of classical mechanics, which are reversible by definition. However, the irreversibility of the Boltzmann equation can be explained by its statistical nature (which is not mechanical), particularly due to the assumption of molecular chaos. In other words, because this assumption is a deviation from classical mechanics, it is not guaranteed to be reversible [22].

2.8 Model Boltzmann Equations

Though the Boltzmann equation reduces the degrees of freedom from $6N$ for the original Hamiltonian system to only 6, it is still very difficult to solve due to the complexity of the collision integral. In order to obtain approximate solutions to the Boltzmann equation, the collision integral is often replaced with a simpler model that possesses the same fundamental properties. The use of a model for the collision integral is justified as long as it still recovers the correct macroscopic transport equations. One of the simplest models is the BGK approximation, named for Bhatnager, Gross, and Krook [8].

For flows near equilibrium, the single-particle mass density can be approximated by a small deviation from the Maxwellian distribution:

$$f = f^{(eq)} + g \ . \tag{2.54}$$

Substituting this expression into the collision integral gives the following expression [22]:

$$\frac{\partial f}{\partial t} + \vec{c} \cdot \nabla f = \frac{1}{m} \int \sigma \left| \vec{c}_1 - \vec{c} \right| \left[\tilde{f}^{(eq)} \tilde{g}_1 + \tilde{f}_1^{(eq)} \tilde{g} - f^{(eq)} g_1 - f_1^{(eq)} g \right] d\Omega d\vec{c}_1 \ . \tag{2.55}$$

The order of magnitude of the right-hand side of this equation can be found by considering only the last term [22]:

$$\frac{\partial f}{\partial t} + \vec{c} \cdot \nabla_{\vec{x}} f \simeq \frac{1}{m} \int \sigma \left| \vec{c}_1 - \vec{c} \right| \left[-f_1^{(eq)} g \right] d\Omega d\vec{c}_1 \simeq -\omega g \tag{2.56}$$

As shown, this term can be replaced with a relaxation type operation where ω is the collision frequency, which is related to the total collision cross-sectional area and the average relative velocity between particles. This assumption leads to the single-relaxation-time BGK model [22]:

$$\frac{\partial f}{\partial t} + \vec{c} \cdot \nabla_{\vec{x}} f = \omega \left[f^{(eq)} - f \right] \quad . \tag{2.57}$$

Replacing the collision integral with the BGK assumption makes the Boltzmann equation a partial differential equation as opposed to an integral-differential equation. This makes it much easier to solve and it retains the following properties of the original collision integral:

1. The BGK model shares the same five collision invariants as the collision integral. This means that it conserves mass, momentum, and energy.

2. The $\mathcal{H}$-Theorem holds for the BGK model, which means that it also tends towards the local Maxwellian distribution at equilibrium.

3. As shown in the next section, the BGK model also recovers the correct macroscopic conservation equations.

It is important to note that, even though it appears that the BGK approximation replaces the collision integral with a linearised operator, the BGK approximation is still strongly nonlinear due to the dependence of the Maxwellian distribution on density, velocity, and temperature. However, this nonlinearity is tolerable since the calculation of $f^{(eq)}$ is local in physical space.

2.9 The Chapman-Enskog Expansion

In section 2.6 the link between the single-particle distribution f and the hydrodynamic quantities was established. It was also shown that if the moments of the Boltzmann equation (corresponding to the conserved quantities) are calculated, the macroscopic conservation equations are recovered (equations 2.38 - 2.40). However, in these conservation equations, the detailed expressions for the total stress tensor P, and the heat flux vector $\vec{Q}$ are not specified. This is because the integrals corresponding to P and $\vec{Q}$ cannot be evaluated without knowledge of the form of f. In this section, a multiscale expansion (named for Sydney Chapman and David Enskog) will be used to derive the expressions for the total stress tensor and the heat flux vector [11].

The first major assumption in this analysis is to write the single-particle distribution as an expansion (with ϵ as the expansion parameter):

$$f = \sum_{n=0}^{\infty} \epsilon^n f^{(n)} \quad . \tag{2.58}$$

As shown in figure 2.2, the Knudsen number indicates the scales that are important in any given flow. Thus, if ϵ is set to be the Knudsen number, the scales important to a continuum fluid can be separated from those only important for the microscopic description.

Since the equilibrium distribution must have the same moments for the collision invariants as f, the following moments can be defined for the various components of the single-particle distribution [26]:

$$\rho = \int f^{(0)} d\vec{c} \ , \tag{2.59}$$

$$\rho\vec{u} = \int \vec{c} f^{(0)} d\vec{c} \ , \tag{2.60}$$

$$3\rho RT = \int |\vec{c}_0|^2 f^{(0)} d\vec{c} \ , \tag{2.61}$$

and for $n > 0$:

$$0 = \int f^{(n)} d\vec{c}, \quad \text{for} \quad n > 0 \ , \tag{2.62}$$

$$0 = \int \vec{c} f^{(n)} d\vec{c}, \quad \text{for} \quad n > 0 \ , \tag{2.63}$$

$$0 = \int |\vec{c}_0|^2 f^{(n)} d\vec{c}, \quad \text{for} \quad n > 0 \ . \tag{2.64}$$

A similar multiscale expansion must be applied to the temporal and spatial derivatives [26]. The temporal derivative becomes:

$$\frac{\partial}{\partial t} = \sum_{n=0}^{\infty} \epsilon^n \frac{\partial_n}{\partial t} \ , \tag{2.65}$$

and the spatial derivative is:

$$\vec{c} \cdot \nabla = \vec{c} \cdot \nabla_0 \ . \tag{2.66}$$

As shown by equation 2.66, only one scale is considered for the spatial derivative. This is because convection and diffusion occur over different time scales, but over similar spatial scales.

The next step in the analysis is to apply the multiscale expansion to the Boltzmann equation. Substituting equations 2.58, 2.65, and 2.66 into the Boltzmann equation (with the BGK assumption for simplicity) gives [26]:

$$\frac{\partial_0 f^{(0)}}{\partial t} + \epsilon \frac{\partial_0 f^{(1)}}{\partial t} + \epsilon \frac{\partial_1 f^{(0)}}{\partial t} + \epsilon^2 \frac{\partial_0 f^{(2)}}{\partial t} + \epsilon^2 \frac{\partial_1 f^{(1)}}{\partial t} + \epsilon^2 \frac{\partial_2 f^{(0)}}{\partial t} + \cdots$$
$$\vec{c} \cdot \nabla_0 f^{(0)} + \epsilon \vec{c} \cdot \nabla_0 f^{(1)} + \epsilon^2 \vec{c} \cdot \nabla_0 f^{(2)} + \cdots$$
$$= -\frac{\omega}{\epsilon} \left(f^{(0)} + \epsilon f^{(1)} + \epsilon^2 f^{(2)} + \cdots - f^{(eq)} \right) \quad . \quad (2.67)$$

The terms of equal order in equation 2.67 can be set equal to each other to give [26]:

$$\frac{\partial_0 f^{(0)}}{\partial t} + \vec{c} \cdot \nabla_0 f^{(0)} = -\omega f^{(1)} \quad (2.68)$$

for the first-order terms ($O(\epsilon)$), and

$$\frac{\partial_1 f^{(0)}}{\partial t} + \frac{\partial_0 f^{(1)}}{\partial t} + \vec{c} \cdot \nabla_0 f^{(1)} = -\omega f^{(2)} \quad (2.69)$$

for the second-order terms ($O(\epsilon^2)$). Equations 2.68 and 2.69 can now be used to compute the macroscopic conservation equations to first- and second-order respectively. These equations will be analysed separately in the sections that follow.

2.9.1 First-Order Analysis

Equation 2.68 represents the terms of first-order in the Boltzmann equation with the BGK assumption. Using the process described in section 2.6, the moments of equation 2.68 can be computed to give the conservation equations to first-order: [26]:

$$\frac{\partial_0 \rho}{\partial t} + (\vec{u} \cdot \nabla_0)\rho = -\rho(\nabla_0 \cdot \vec{u}) \ , \tag{2.70}$$

$$\frac{\partial_0 \vec{u}}{\partial t} + (\vec{u} \cdot \nabla_0)\vec{u} = -\frac{1}{\rho}\nabla_0 \cdot \mathsf{P}^{(0)} \ , \tag{2.71}$$

and

$$\frac{\partial_0 T}{\partial t} + (\vec{u} \cdot \nabla_0)T = -\frac{2}{3\rho R}\left[\nabla_0 \cdot (\vec{u} \cdot \mathsf{P}^{(0)}) + \nabla_0 \cdot \vec{Q}^{(0)}\right] \ . \tag{2.72}$$

where:

$$\mathsf{P}^{(0)} = \int \vec{c}_0 \vec{c}_0 f^{(0)} d\vec{c} \ , \tag{2.73}$$

and

$$\vec{Q}^{(0)} = \int \frac{|\vec{c}_0|^2}{2}\vec{c}_0 f^{(0)} d\vec{c} \ . \tag{2.74}$$

Since $f^{(0)}$ is equal to the equilibrium distribution ($f^{(eq)}$) the first-order components of P and $\vec{Q}$ can be calculated using the integrals above [26]. Thus,

$$\begin{aligned} \mathsf{P}^{(0)} &= \int \vec{c}_0\vec{c}_0 f^{(eq)} d\vec{c} \\ &= \rho RT\mathsf{I} \\ &= p\mathsf{I} \ , \end{aligned} \tag{2.75}$$

and

$$\begin{aligned} \vec{Q}^{(0)} &= \int \frac{|\vec{c}_0|^2}{2}\vec{c}_0 f^{(eq)} d\vec{c} \\ &= 0 \ . \end{aligned} \tag{2.76}$$

Substituting equations 2.75 and 2.76 into the first-order conservation equations (2.70 - 2.72) gives [26]:

$$\frac{\partial_0 \rho}{\partial t} + (\vec{u} \cdot \nabla_0)\rho = -\rho(\nabla_0 \cdot \vec{u}) \ , \tag{2.77}$$

$$\frac{\partial_0 \vec{u}}{\partial t} + (\vec{u} \cdot \nabla_0)\vec{u} = -\frac{1}{\rho}\nabla_0 p \ , \tag{2.78}$$

and

$$\frac{\partial_0 T}{\partial t} + (\vec{u} \cdot \nabla_0)T = -\frac{2}{3\rho R}(\nabla_0 \cdot \vec{u}p) \ , \tag{2.79}$$

which are the Euler equations. Thus, the first-order components of the Boltzmann equation give the governing equations for inviscid flow. This result is intuitive since the characteristic length scales are infinitely large for flows with extremely small Knudsen numbers. This means that the gradients in the flow are very small. Consequently, the effect of diffusion is not important in these flows, so there are no heat conduction or viscous stress terms in these equations.

2.9.2 Second-Order Analysis

As with the first-order analysis, the first step is to calculate the second-order conservation equations. This is done by calculating the moments of equation 2.69 [26]:

$$\frac{\partial_1 \rho}{\partial t} = 0 \ , \tag{2.80}$$

$$\frac{\partial_1 \vec{u}}{\partial t} = -\frac{1}{\rho}\nabla_0 \cdot \mathsf{P}^{(1)} \ , \tag{2.81}$$

and

$$\frac{\partial_1 T}{\partial t} = -\frac{2}{3\rho R}\left[\nabla_0 \cdot (\vec{u} \cdot \mathsf{P}^{(1)}) + \nabla_0 \cdot \vec{Q}^{(1)}\right] \ . \tag{2.82}$$

where:

$$\mathsf{P}^{(1)} = \int \vec{c}_0 \vec{c}_0 f^{(1)} d\vec{c} \ , \tag{2.83}$$

and

$$\vec{Q}^{(1)} = \int \frac{|\vec{c}_0|^2}{2} \vec{c}_0 f^{(1)} d\vec{c} \ . \tag{2.84}$$

At this stage in the first-order analysis, the integrals for P and $\vec{Q}$ were simply evaluated because they were a function of $f^{(eq)}$. Unfortunately, the second-order terms are functions of $f^{(1)}$, which is currently unknown. However, equation 2.68 can be rearranged to find an expression for $f^{(1)}$ in terms of $f^{(eq)}$ [26]:

$$f^{(1)} = -\frac{1}{\omega}\left(\frac{\partial_0 f^{(eq)}}{\partial t} + \vec{c}\cdot\nabla_0 f^{(eq)}\right) = -\frac{1}{\omega} D_0 f^{(eq)} \quad . \tag{2.85}$$

where $D_0 f^{(eq)}$ is the first-order component of the total derivative of the local Maxwellian distribution. In the above equation, the temporal and spatial derivatives of $f^{(eq)}$ must be evaluated; however, the local Maxwellian is not directly dependent on position or time. To fix this problem, $f^{(eq)}$ is assumed to implicitly depend on time and position through its dependence on density, velocity and temperature. Therefore, the temporal and spatial derivatives can be written using the chain rule [26]:

$$\frac{\partial_0 f^{(eq)}}{\partial t} = \frac{\partial_0 f^{(eq)}}{\partial \rho}\frac{\partial_0 \rho}{\partial t} + \frac{\partial_0 f^{(eq)}}{\partial \vec{u}}\cdot\frac{\partial_0 \vec{u}}{\partial t} + \frac{\partial_0 f^{(eq)}}{\partial T}\frac{\partial_0 T}{\partial t} \quad , \tag{2.86}$$

and

$$\vec{c}\cdot\nabla_0 f^{(eq)} = \frac{\partial_0 f^{(eq)}}{\partial \rho}(\vec{c}\cdot\nabla_0)\rho + \frac{\partial_0 f^{(eq)}}{\partial \vec{u}}\cdot(\vec{c}\cdot\nabla_0)\vec{u} + \frac{\partial_0 f^{(eq)}}{\partial T}(\vec{c}\cdot\nabla_0)T \quad . \tag{2.87}$$

Applying this assumption, called the *ansatz*, gives the following expression for $f^{(1)}$ [26]:

$$f^{(1)} = -\frac{1}{\omega}\left[\frac{\partial_0 f^{(eq)}}{\partial \rho}\left(\frac{\partial_0 \rho}{\partial t} + (\vec{c}\cdot\nabla_0)\rho\right) + \frac{\partial_0 f^{(eq)}}{\partial \vec{u}}\cdot\left(\frac{\partial_0 \vec{u}}{\partial t} + (\vec{c}\cdot\nabla_0)\vec{u}\right)\right.$$
$$\left. + \frac{\partial_0 f^{(eq)}}{\partial T}\left(\frac{\partial_0 T}{\partial t} + (\vec{c}\cdot\nabla_0)T\right)\right] \quad . \tag{2.88}$$

The expressions for the temporal and spatial derivatives can be found from the first-order conservation equations (equations 2.70 - 2.72). The derivatives of the equilibrium distribution with respect to ρ, $\vec{u}$, and T are:

$$\frac{\partial_0 f^{(eq)}}{\partial \rho} = \frac{1}{\rho} f^{(eq)} \quad , \tag{2.89}$$

$$\frac{\partial_0 f^{(eq)}}{\partial \vec{u}} = \frac{\vec{c}_0}{RT} f^{(eq)} \quad , \tag{2.90}$$

and

$$\frac{\partial_0 f^{(eq)}}{\partial T} = \left(\frac{|\vec{c}_0|^2}{2RT^2} - \frac{3}{2T}\right) f^{(eq)} \quad . \tag{2.91}$$

Substituting these expressions into the equation for $f^{(1)}$ gives [26]:

$$f^{(1)} = -\frac{f^{(eq)}}{\omega}\left[\frac{1}{\rho}\left((\vec{c}_0 \cdot \nabla_0)\rho - \rho(\nabla_0 \cdot \vec{u})\right) + \frac{\vec{c}_0}{RT} \cdot \left((\vec{c}_0 \cdot \nabla_0)\vec{u} - \frac{1}{\rho}\nabla_0 p\right)\right.$$
$$\left. -\frac{3}{2T}\left((\vec{c}_0 \cdot \nabla_0)T - \frac{2T}{3}(\nabla_0 \cdot \vec{u})\right) + \frac{|\vec{c}_0|^2}{2RT^2}\left((\vec{c}_0 \cdot \nabla_0)T - \frac{2T}{3}(\nabla_0 \cdot \vec{u})\right)\right] \quad . \tag{2.92}$$

Grouping like terms, the expression above reduces to:

$$f^{(1)} = -\frac{f^{(eq)}}{\omega}\left[\frac{1}{RT}\left(\vec{c}_0 \cdot (\vec{c}_0 \cdot \nabla_0)\vec{u} - \frac{|\vec{c}_0|^2}{3}(\nabla_0 \cdot \vec{u})\right)\right.$$
$$\left. + \frac{1}{T}\left(\frac{|\vec{c}_0|^2}{2RT} - \frac{5}{2}\right)(\vec{c}_0 \cdot \nabla_0)T\right] \quad . \tag{2.93}$$

As with the first-order case, the expressions for the second-order components of P can be found using equation 2.96. In the expression for $f^{(1)}$, only the first term contributes to the total stress, so the second term can be ignored [26]:

$$\mathsf{P}^{(1)} = -\frac{1}{\omega RT}\int \vec{c}_0\vec{c}_0\left(\vec{c}_0\cdot(\vec{c}_0\cdot\nabla_0)\vec{u} - \frac{|\vec{c}_0|^2}{3}(\nabla_0\cdot\vec{u})\right) f^{(eq)}d\vec{c}\ . \tag{2.94}$$

Using the following definition:

$$\int c_{0i}c_{0j}c_{0k}c_{0l}f^{(eq)}d\vec{c} = \rho(RT)^2(\delta_{ij}\delta_{kl}+\delta_{ik}\delta_{jl}+\delta_{il}\delta_{jk})\ , \tag{2.95}$$

The second-order stress tensor can be evaluated to be:

$$\mathsf{P}^{(1)} = -\mu\left(\nabla_0\vec{u}+(\nabla_0\vec{u})^{\mathsf{T}}\right) - \lambda(\nabla_0\cdot\vec{u})\mathsf{I} \tag{2.96}$$

where:

$$\mu = \frac{\rho RT}{\omega}\ , \tag{2.97}$$

and

$$\lambda = -\frac{2}{3}\mu\ . \tag{2.98}$$

It is important to note that the bulk viscosity above is the same as that predicted by Stokes' theorem [51]. The second-order heat flux can be found by using equation 2.84. Only the second term in the expression for $f^{(1)}$ contributes to the heat flux [26]:

$$\vec{Q}^{(1)} = -\frac{1}{\omega T}\int \frac{|\vec{c}_0|^2}{2}\vec{c}_0\left(\frac{|\vec{c}_0|^2}{2RT} - \frac{5}{2}\right)(\vec{c}\cdot\nabla_0)Tf^{(eq)}d\vec{c} \ . \tag{2.99}$$

Using equation 2.95 and the following expression:

$$\begin{aligned}\int c_{0i}c_{0j}c_{0k}c_{0l}c_{0m}c_{0n}f^{(eq)}d\vec{c} = \rho(RT)^3\,(&\delta_{ij}\delta_{kl}\delta_{mn} + \delta_{ij}\delta_{km}\delta_{ln} + \delta_{ij}\delta_{kn}\delta_{lm}\\ &+\delta_{ik}\delta_{jl}\delta_{mn} + \delta_{ik}\delta_{jm}\delta_{ln} + \delta_{ik}\delta_{jn}\delta_{lm}\\ &+\delta_{il}\delta_{jk}\delta_{mn} + \delta_{il}\delta_{jm}\delta_{kn} + \delta_{il}\delta_{jn}\delta_{km}\\ &+\delta_{im}\delta_{jk}\delta_{ln} + \delta_{im}\delta_{jl}\delta_{kn} + \delta_{im}\delta_{jn}\delta_{kl}\\ &+\delta_{in}\delta_{jk}\delta_{lm} + \delta_{in}\delta_{jl}\delta_{km} + \delta_{in}\delta_{jm}\delta_{kl})\end{aligned} \tag{2.100}$$

The heat flux vector can be rewitten as [26]:

$$\vec{Q}^{(1)} = -k\nabla_0 T \ , \tag{2.101}$$

where:

$$k = \frac{15\rho R^2 T}{4\omega} \quad . \tag{2.102}$$

Using the expressions for the total stress and the heat flux, the second-order conservation equations can be written as [26]:

$$\frac{\partial_1 \rho}{\partial t} = 0 \quad , \tag{2.103}$$

$$\frac{\partial_1 \vec{u}}{\partial t} = \frac{1}{\rho}\nabla_0 \cdot \left[\mu\left(\nabla_0\vec{u} + (\nabla_0\vec{u})^{\mathsf{T}}\right) + \lambda(\nabla_0 \cdot \vec{u})\mathsf{I}\right] \quad , \tag{2.104}$$

and

$$\frac{\partial^{(1)} T}{\partial t} = \frac{2}{3\rho R}\left[\nabla_0 \cdot \left(\vec{u} \cdot \left(\mu\left(\nabla_0\vec{u} + (\nabla_0\vec{u})^{\mathsf{T}}\right) + \lambda(\nabla_0 \cdot \vec{u})\mathsf{I}\right)\right) - \nabla_0 \cdot (k\nabla_0 T)\right] \quad . \tag{2.105}$$

Combining the second-order conservation equations with the first-order equations gives the total conservation equations to second-order [26]:

$$\frac{\partial \rho}{\partial t} + (\vec{u} \cdot \nabla)\rho = -\rho(\nabla \cdot \vec{u}) \ , \tag{2.106}$$

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = \frac{1}{\rho}\left[-\nabla p + \nabla \cdot \mu\left(\nabla \vec{u} + (\nabla \vec{u})^{\mathsf{T}}\right) + \nabla \cdot \lambda(\nabla \cdot \vec{u})\mathsf{I}\right] \ , \tag{2.107}$$

and

$$\frac{\partial T}{\partial t} + (\vec{u} \cdot \nabla)T = \frac{2}{3\rho R}\left[\nabla \cdot \left(\vec{u} \cdot \left(\mu\left(\nabla \vec{u} + (\nabla \vec{u})^{\mathsf{T}}\right) + \lambda(\nabla \cdot \vec{u})\mathsf{I}\right)\right) - \nabla \cdot (k \nabla T)\right] \ , \tag{2.108}$$

which are the Navier-Stokes equations. Thus, when the Knudsen number is small but finite, the Boltzmann equation is equivalent to the macroscopic transport equations for viscous fluid flow.

Chapter 3

Implementation and Verification

In the previous chapter, the Boltzmann equation was derived and the Chapman-Enskog expansion was used to prove that the Navier-Stokes equations are recovered in the macroscopic limit. This demonstrates that the Boltzmann equation is a viable option for describing the motion of turbulent flow. However, since there are very few known analytical solutions to the Boltzmann equation, the only practical way to find solutions is to solve it numerically.

In the first section of this chapter, the lattice Boltzmann equation is derived as a numerical approximation to the Boltzmann equation. This involves constraining the particles in the model to travel with a discrete set of lattice velocities and discretizing the resulting set of partial differential equations in space and time. In the last section, a number of simple problems are solved with the LBM and the results are compared to analytical solutions in order to verify that the LBM algorithm has been implemented correctly.

3.1 Implementation

While the lattice Boltzmann method can be derived as a finite difference discretization of the Boltzmann equation [23], it originated from a classification of models called Lattice Gas Cellular Automata (LGCA). These models are similar to the molecular dynamics approach in that they contain discrete particles, but they have a smaller computational cost because the particles are only permitted to travel at fixed velocities along specified lattice directions. The macroscopic state at each lattice site can be calculated from the configuration of the particles at each site. The evolution is controlled by two steps: collision and streaming. During the collision step, the pre-collision states at each lattice site are converted to post-collision states according to collision rules that conserve mass, momentum, and energy. Next, during the streaming step, the collided particles are moved to their neighbouring lattice sites according to their velocities. Though these types of models are fairly *ad hoc*, as long as the lattice used is sufficiently symmetric and contains enough discrete velocities they can be shown to recover the proper macroscopic conservation equations [20].

The main advantage of using an LGCA model is the simplicity of the implementation. Instead of discretizing partial differential equations, the LGCA models produce hydrodynamic behaviour using only simple streaming and collision rules. Furthermore, LGCA uses only Boolean computations. This is because the state of a lattice node can be specified by which particle sites are occupied and which are not (the occupation numbers are therefore equal to 1 or 0). This increases computation speed, reduces memory requirements, and allows for exact conservation. However, these models suffer from a number of problems, namely: non-isotropic advection, violation of Galilean invariance, spurious invariants, noise, and a dependence of the pressure on velocity [52].

To address these problems, LGCA models were replaced by the lattice Boltzmann method [38, 25]. In the LBM, the lattice states are no longer specified with Boolean numbers; instead they are specified by the single-particle distribution f and the collision matrix is replaced with a relaxation type process.

Despite the fact that the actual evolution of the LBM was through LGCA, it can also be derived as a finite difference discretization of the Boltzmann equation [23]. Deriving the LBM as a numerical procedure to solve the Boltzmann equation emphasises its link with kinetic theory, and therefore this will be the approach discussed in the following sections.

3.1.1 The Discrete Boltzmann Equation

Even with the BGK approximation for the collision integral, it is very difficult to find analytical solutions to the Boltzmann equation. For this reason, approximate solutions are usually found by solving a discretized version of the Boltzmann equation. However, the task of finding an appropriate discretization scheme for the Boltzmann equation is more difficult than it is for the macroscopic conservation equations. This is because the working variable, f, is a function of position, velocity, and time. The dependence of f on velocity is awkward since all of the macroscopic quantities are only dependent on space and time.

If the Boltzmann equation is only discretized in space and the entire velocity space is retained, the computational resources required to compute the integrals for the moments of f would be too great. So, a method of discretizing velocity space must be found to reduce the computational complexity. To do this, the particles in the Boltzmann equation are forced to travel with a discrete set of lattice velocities.

Work in LGCA showed that the computational complexity could be drastically reduced by limiting the number of particle velocities allowed in the model and that the correct transport equations could be recovered providing the lattice is symmetric and contains a sufficient number of velocities [20].

By choosing a finite set of velocities, the single-particle distribution is transformed from a continuous distribution over velocity space to a vector of single-particle distributions corresponding to each lattice velocity. In this work, the D3Q19 lattice was used, and is displayed in figure 3.1. This lattice is called the D3Q19 because it has 19 discrete particle velocities in three dimensions. The 19 discrete particle velocities in the D3Q19 lattice in vector form are [52]:

$$\vec{c}_i = \begin{cases} (0,0,0), & i = 0 \ ; \\ (\pm 1,0,0),(0,\pm 1,0),(0,0,\pm 1), & i = 1,2,\ldots,6 \ ; \\ (\pm 1,\pm 1,0),(\pm 1,0,\pm 1),(0,\pm 1,\pm 1), & i = 7,8,\ldots,18 \ . \end{cases} \tag{3.1}$$

When the particles are constrained to travel on the D3Q19 lattice, the single-particle distribution can be split into 19 separate distributions that no longer have a velocity argument. The relationship between the continuous and discrete single-particle distribution is:

$$f_i(\vec{x},t) = w_i f(\vec{x},\vec{c}_i,t) \ . \tag{3.2}$$

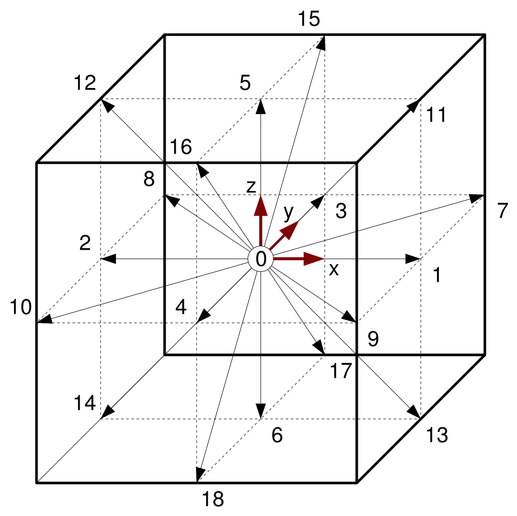


Figure 3.1: The D3Q19 lattice. The particles are constrained to travel with 19 discrete velocities.

The weighting factors, w_i, in equation 3.2 are quadrature coefficients that are determined such that the velocity moments in equations 2.26 - 2.28 can be replaced by equivalent sums. The discrete moments for mass, momentum, and energy, respectively, become:

$$\rho = \sum_{i=0}^{18} f_i \quad , \tag{3.3}$$

$$\rho\vec{u} = \sum_{i=0}^{18} \vec{c}_i f_i \quad , \tag{3.4}$$

and

$$\rho E = \sum_{i=0}^{18} \frac{|\vec{c}_i|^2}{2} f_i \ . \tag{3.5}$$

The equilibrium distribution must also be discretized. This is done by taking the values of the Maxwell-Boltzmann distribution for each of the discrete velocities and applying the same quadrature coefficients found in equation 3.2. This results in the discretized Maxwell-Boltzmann distribution:

$$f_i^{(eq)} = w_i \left(\frac{\rho}{(2\pi RT)^{3/2}} \right) e^{\left(\frac{-(\vec{c}_i - \vec{u})^2}{2RT} \right)} \ . \tag{3.6}$$

This expression contains an exponential function that is very expensive to evaluate numerically. To simplify it, the argument of the exponential function is expanded and the term that does not contain the macroscopic velocity is removed. The resulting expression is:

$$f_i^{(eq)} = w_i \left(\frac{1}{(2\pi RT)^{3/2}} \right) \rho e^{\left(\frac{-|\vec{c}_i|^2}{2RT} \right)} e^{\left(\frac{(\vec{c}_i \cdot \vec{u})}{RT} - \frac{|\vec{u}|^2}{2RT} \right)} \ . \tag{3.7}$$

Next, the second exponential term is replaced by a Taylor-series expansion about $\vec{u} = 0$, and the terms not containing $\vec{u}$ are lumped into a weighting function. This gives the final form of the discrete equilibrium distribution:

$$f_i^{eq} = W_i \rho \left[1 + \frac{(\vec{c}_i \cdot \vec{u})}{RT} + \frac{(\vec{c}_i \cdot \vec{u})^2}{2(RT)^2} - \frac{|\vec{u}|^2}{2RT} \right] \quad , \tag{3.8}$$

where the weighting functions, W_i, are:

$$W_i = w_i \left(\frac{1}{(2\pi RT)^{3/2}} \right) e^{\left(\frac{-|\vec{c}_i|^2}{2RT} \right)} \quad . \tag{3.9}$$

The next step is to determine the values of the weighting functions by ensuring that the velocity moments of the discrete equilibrium distribution are equal to those of the original equilibrium distribution. This gives the following set of equations:

$$\sum_{i=0}^{18} f_i^{(eq)} = \rho \quad , \tag{3.10}$$

$$\sum_{i=0}^{18} \vec{c}_i f_i^{(eq)} = \rho \vec{u} \quad , \tag{3.11}$$

and

$$\sum_{i=0}^{18} \vec{c}_i \vec{c}_i f_i^{(eq)} = \rho \left(RT\mathsf{I} + \vec{u}\vec{u} \right) \ . \tag{3.12}$$

Solving equations 3.10 - 3.12 for the weight coefficients yields the following results for W_i:

$$W_i = \begin{cases} 1/3, & i = 0 \ ; \\ 1/18, & i = 1, 2, \ldots, 6 \ ; \\ 1/36, & i = 7, 8, \ldots, 18 \ . \end{cases} \tag{3.13}$$

Additionally, due to the limited number of lattice velocities and the truncation of terms in the Taylor-series expansion of the discrete equilibrium distribution, the following constraint is also enforced by equations 3.10 - 3.12:

$$RT = \frac{1}{3} \ . \tag{3.14}$$

Since the specific gas constant, R, is a constant, this implies that the temperature is also fixed for the D3Q19 LBM. This has another physical implication. Since the temperature is constant, p/ρ is also constant (from the ideal-gas law). Thus,

$$\frac{\partial p}{\partial \rho} = \frac{1}{3} \ . \tag{3.15}$$

This relationship can be substituted into the definition of the speed of sound in an ideal gas,

$$c_s = \sqrt{\frac{\partial p}{\partial \rho}} \ , \tag{3.16}$$

to determine that the speed of sound for the D3Q19 LBM is:

$$c_s = \sqrt{\frac{1}{3}} \ . \tag{3.17}$$

Thus, the speed of sound is constant in the D3Q19 LBM. This means that the speed of the sound waves in the D3Q19 LBM is not dependent on the local temperature as in a conventional compressible simulation of an ideal gas. For this reason, the D3Q19 LBM is not recommended for simulating flows in which the behaviour of the sound waves is thought to be important (such as aeroacoustic simulations).

The specification of the D3Q19 lattice, along with the discrete form of the equilibrium distribution with the values for the weighting functions and the sound speed, complete the discretization of velocity space for the Boltzmann equation. The result is the discrete Boltzmann equation (DBE):

$$\frac{\partial f_i}{\partial t} + \vec{c}_i \cdot \nabla f_i = -\omega(f_i - f_i^{(eq)}) \ , \tag{3.18}$$

which is a set of first-order partial differential equations. Each of these equations can be discretized in physical space and solved simultaneously to obtain approximate solutions to the Boltzmann equation.

3.1.2 The Lattice Boltzmann Equation

The DBE, equation 3.18, is obtained by discretizing the Boltzmann equation in velocity space. The lattice Boltzmann equation (LBE) is derived by further discretizing the DBE in space and time. First, the left-hand side of the DBE can be replaced by the substantial derivative of the single-particle distribution to give:

$$\frac{Df_i}{Dt} = -\omega(f_i - f_i^{(eq)}) \ . \tag{3.19}$$

The particle velocities are constant and thus represent characteristics along which the equations can be discretized. The substantial derivative is replaced with a first-order finite difference approximation, and the method is made explicit in time by evaluating the right-hand side of the equation at time t. This gives:

$$\frac{f_i(\vec{x}+\vec{c}_i\Delta t, t+\Delta t) - f_i(\vec{x}, t)}{\Delta t} = -\omega(f_i(\vec{x}, t) - f_i^{eq}(\vec{x}, t)) \ . \tag{3.20}$$

Finally, this equation can be rearranged to isolate $f_i(\vec{x}+\vec{c}_i\Delta t, t+\Delta t)$. This gives the lattice Boltzmann equation:

$$f_i(\vec{x}+\vec{c}_i\Delta t, t+\Delta t) = (1-\hat{\omega})f_i(\vec{x}, t) + \hat{\omega} f_i^{eq}(\vec{x}, t) \ , \tag{3.21}$$

where $\hat{\omega}$ is equal to $\omega\Delta t$.

The lattice Boltzmann equation (LBE) is fully explicit since the terms on the right-hand side are all evaluated at time t. If the LBE is implemented on a uniform cubic lattice, such that $\Delta\vec{x}_i = \vec{c}_i\Delta t$ for each particle velocity, no interpolation is required and the method is exactly conservative.

3.1.3 Initial and Boundary Conditions

One of the main difficulties in using the LBM is the implementation of boundary conditions. In conventional CFD simulations, the boundary conditions correspond to the working variables of the simulation, such as density, velocity, and temperature. In the LBM, the boundary conditions must be applied in terms of the single-particle distribution, as it is the working variable. Therefore, for even the simplest boundary treatments, procedures must be developed to calculate the correct boundary distributions.

There are many different boundary conditions that can be applied in the LBM; however, only those boundaries used in this work will be described in this report. For more information on the various boundary treatments possible, see [48].

Initial Conditions

If the LBM is used to solve the incompressible Navier-Stokes equations, the initial conditions consist of the density and the three components of velocity at each lattice site. For a conventional NS solver, these variables are the working variables for the method, so there is generally no question as to how to apply the initial state. However, in the LBM the initial state is specified by the initial single-particle distribution at each lattice site. This works out to be nineteen unknowns at each lattice site for the D3Q19 LBM.

If only the density and velocities are known, the initial conditions cannot be prescribed uniquely. The reason for this problem is that the specification of f_i not only fixes the density and velocity, but it also fixes the higher moments. So, in order to specify the exact initial conditions, the higher moments must be known as well. However, the specification of these moments is often not feasible because the values of the higher moments are usually unknown.

One method of specifying the initial state is to set the initial values of f_i equal to the local equilibrium distribution for the given density and velocity. This method is convenient because it requires no extra information, but it does not accurately represent the initial state. A simple example to demonstrate this point is the solution of Poiseuille flow. The steady solution for the single-particle distribution for Poiseuille flow is not equal to the local equilibrium distribution. So, if the flow is initialized to the equilibrium state, there will be a short period during the initial time steps in which the particle distributions must converge to their proper values.

An alternative method to specifying the initial conditions is proposed by [39]. In this method, the domain is first initialized to the equilibrium distribution. Next, the LBM is run for a number of initial time steps in which the equilibrium distribution is always calculated using the initial values of density and velocity. The iterations are run this way until the solution converges. The resulting particle distributions still represent the proper initial field, but have the correct higher-order moments.

Periodic and Symmetry Boundaries

Periodic boundaries are the easiest boundary conditions to implement in the LBM. To create periodic boundaries, the domain is simply repeated in the periodic direction of the model. This is accomplished by setting the neighbours of the lattice sites on the outlet plane to their equivalent lattice sites on the inlet plane and vice versa. This method ensures exact mass conservation on the boundaries because no single-particle distributions leave the computational domain [48].

Symmetry conditions are also easily implemented in the LBM. The unknown particle distributions are set equal to their mirror image distributions that were streamed to the boundary from within the simulation domain. This means that if a particle distribution streams out of the simulation domain, it is replaced by its mirror image.

This leads to exact mass conservation at the boundaries as with the periodic conditions [48].

Body Forces

For a code that solves the incompressible NS equations, a pressure gradient can be applied to drive a flow with no change in density. However, since the LBM is a compressible method, a pressure gradient can only be established by setting up a density gradient. For example, for Poiseuille flow, a pressure gradient can be established in the channel by simply adding mass at the inlet, and removing it at the outlet. However, if a periodic boundary is desired in the streamwise direction a density difference from the inlet to the outlet can become problematic. In this case it is preferable to use a body force to drive the flow.

In the LBM, a body force is implemented by biasing the collision operator to increase the number of particles moving in the direction of the body force, and reducing the number of particles streaming in the opposite direction. This can be done by adding a term to the collision operator [48].

$$f_i(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) = (1 - \hat{\omega}) f_i(\vec{x}, t) + \hat{\omega} \Delta t f_i^{eq}(\vec{x}, t) + g_i \ , \tag{3.22}$$

where:

$$g_i = B_i \rho (\vec{e}_i \cdot \vec{g}) \ . \tag{3.23}$$

In this equation, $\vec{g}$ is the body force vector applied to the flow and the weighting factors B_i are calculated to ensure conservation of mass and momentum. For the D3Q19 lattice their values are:

$$B_i = \begin{cases} 0, & i = 0 \;; \\ 1/6, & i = 1, 2, \ldots, 6 \;; \\ 1/12, & i = 7, 8, \ldots, 18 \;. \end{cases} \tag{3.24}$$

Wall Boundaries

In conventional CFD methods, a wall boundary can be specified by either enforcing a zero velocity condition at a boundary node, or by specifying the shear stress at the edge of a boundary cell. In the LBM, specifying either the shear stress or the velocity is not enough to completely determine the single-particle distributions at a boundary node. So, for a lattice with a moderate number of lattice directions, it is not possible to uniquely determine the boundary state from the macroscopic boundary conditions.

In LGCA, wall boundary conditions are implemented using collision rules. For a no-slip boundary condition, the particles that stream toward a boundary site during the propagation step have their directions completely reversed by the collision (diffuse reflection). For a free-slip wall, the incoming particles are specularly reflected such that they maintain their momentum in the direction parallel to the wall. This type of boundary condition is commonly referred to as the bounce-back scheme [48].

While the collision step in the LBM is not dictated by collision rules like LGCA, this method offers a heuristic procedure to obtain a no-slip boundary. When the bounce-back scheme is applied to the LBM, the unknown distributions at a boundary site are set equal to the distributions moving in the opposite direction. The boundary node in this case does not undergo a collision step, and this creates a wall located halfway between the boundary node and its nearest neighbour in the simulation domain. In general, this method is only first-order accurate, so it will degrade the accuracy of the LBM. However, the bounce-back method has been shown to be second-order accurate for cases in which the flow is aligned parallel to the wall [24].

The main advantage of the bounce-back method is its simple implementation. This makes the LBM, with the bounce-back scheme, ideal for computations with complex geometry. Also, the specific implementation of the bounce-back described above has proven to be stable for turbulent flows. It should be noted that the variant of this method in which the boundary node is collided leads to instabilities when the flow is turbulent. This is because this method alters the velocities parallel to the wall. If these values do not offset each other at a boundary node, a slip velocity will be created that will cause the simulation to become unstable.

There are a number of other methods that have been proposed for wall boundaries. In one method, the hydrodynamic boundary conditions (zero velocity) are used to solve a system of equations to determine the unknown distributions at the boundary [42]. Ideally, a wall should be specified by only the wall velocity. If the D2Q6 or D2Q7 lattices are used, the number of unknown particle distributions at the boundary is equal to the number of boundary conditions. However, if a lattice is used with any more lattice velocities, there are not enough hydrodynamic conditions to solve for the unknowns. Others have proposed schemes to eliminate the slip velocity that occurs with high viscosities; however, these methods substantially increase the complexity, especially for three-dimensional models [27].

Another method exploits the fact that the LBM is a finite difference discretization of the discrete Boltzmann equation [14]. In this method, the distribution values are extrapolated for a node outside the boundary. Then, during the streaming step, these extrapolated values move to the unknown distributions on the boundary node. When the boundary node is collided, the equilibrium distribution is calculated using the hydrodynamic boundary conditions. The advantage to this method is that it can be generalised to other types of boundaries such as inlets, outlets, and zero gradient conditions. However, it was found that this method was unstable when applied to a turbulent flow. The sharp gradients near the wall made it difficult for the collision process to maintain a zero velocity and eventually led to an instability.

3.1.4 Algorithm

The ingredients necessary for a complete LBM algorithm are: an initialization procedure, the lattice Boltzmann equation (equation 3.21), the velocity moment definitions (equations 3.3 and 3.4), and appropriate boundary conditions. All of these ingredients have been specified, so the complete algorithm used in this work can now be discussed. The flow chart for the LBM algorithm used in this work is displayed in figure 3.2.

As discussed in section 1.2, the goal of this work is to simulate fully-developed turbulent channel flow. Since the flow is fully developed, the turbulence statistics are not sensitive to the initial conditions of the simulation. Thus, despite the discussion in section 3.1.3 regarding initialization, the particle distributions were set equal to

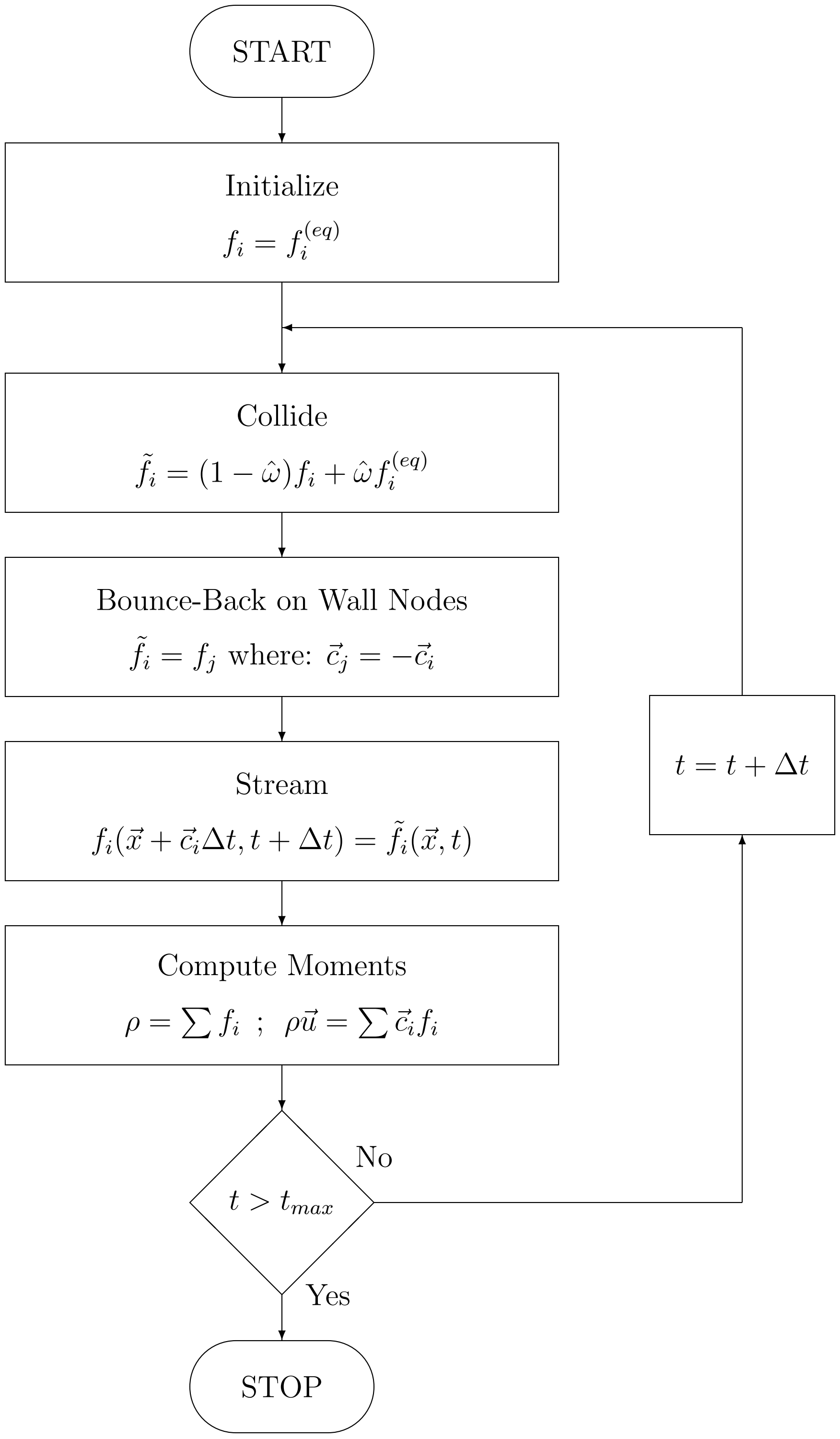

START
Initialize
$f_i = f_i^{(eq)}$
Collide
$\tilde{f}_i = (1-\hat{\omega})f_i + \hat{\omega} f_i^{(eq)}$
Bounce-Back on Wall Nodes
$\tilde{f}_i = f_j$ where: $\vec{c}_j = -\vec{c}_i$
Stream
$f_i(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) = \tilde{f}_i(\vec{x}, t)$
Compute Moments
$\rho = \sum f_i$; $\rho\vec{u} = \sum \vec{c}_i f_i$
$t > t_{max}$
No
Yes
$t = t + \Delta t$
STOP

Figure 3.2: The lattice Boltzmann method algorithm.

their equilibrium values at the beginning of the simulation. Even though this is not an accurate method of specifying the initial macroscopic state, it does not affect the results of the simulation.

Once the particle distributions have been initialized, the inner loop of the LBM is started. As with the LGCA, the solution of the LBE is split into two steps: collision and streaming. During the collision step the right-hand side of equation 3.21 is evaluated and the post-collision particle distributions (denoted $\tilde{f}_i$) are stored at their current lattice sites. It should be noted that the collision step is only performed on the fluid nodes, which does not include the nodes on the wall boundaries.

After the collision step, the post-collision particle distributions located on the interior nodes of the computational domain are ready to be streamed. However, before the streaming step is started, the values of the particle distributions on the boundaries must be set. Thus, the next step in the inner loop is the wall boundary subroutine. This subroutine performs a complete bounce-back on the wall boundaries. The post-collision particle distributions on the boundary nodes are set equal to the pre-collision distributions moving in the opposite direction on the lattice ($\tilde{f}_i = f_j$ where: $\vec{c}_j = -\vec{c}_i$).

The streaming step is then applied to both the fluid and wall nodes. The streaming operation simply shifts the particle distributions according to their particle velocity ($\vec{c}_i$). Provided that the LBM is implemented on a uniform cubic lattice ($\Delta x = \Delta y = \Delta z$) and the time step is set such that $\Delta t = \vec{c}_i/\vec{x}_i$, the particle distributions will travel to their neighbouring lattice sites in one time step, and no interpolation is required. This is preferable because interpolations will degrade the formal order of accuracy and increase the computational cost. It should be noted that, since the code was written for eventual use with complex geometries, the particle distributions were stored in a one-dimensional unstructured array. This makes the implementation of complex geometries much easier, but an auxiliary array is needed to store the node neighbours for the streaming routine. This reduces the computational efficiency of the code, but it gives it the geometric flexibility needed for simulating complex geometries.

The final step of the inner loop is the moment evaluation. In this step, the macroscopic density and momentum are calculated at each lattice site. These values may be written to disk at this time or simply stored in memory for use in the collision step of the next iteration. If the time variable is greater than the target value, the inner loop is exited and the LBM algorithm is stopped. If the time variable has not yet reached the target, it is incremented and the algorithm returns to the start of the inner loop for the next time iteration.

3.1.5 The Chapman-Enskog Expansion

In section 2.9 the Chapman-Enskog procedure was applied to the continuous Boltzmann equation to show that it recovers the NS equations for viscous flow. However, it is also important to perform the same analysis on the lattice Boltzmann equation to verify that it too recovers the correct macroscopic behaviour. Additionally, through this analysis it can be shown that, by adjusting the the form of the viscosity, the LBM can be modified to become second-order accurate in space and time. The first step is to write $f_i(\vec{x}+\vec{c}_i\Delta t, t+\Delta t)$ as a Taylor-series expansion about $(\vec{x}, t)$:

$$\begin{aligned} f_i(\vec{x}+\vec{c}_i\Delta t, t+\Delta t) = f_i + \Delta t &\left[\frac{\partial f_i}{\partial t} + \vec{c}_i \cdot \nabla f_i\right] \\ &+ \frac{(\Delta t)^2}{2}\left[\frac{\partial}{\partial t}\left(\frac{\partial f_i}{\partial t} + \vec{c}_i \cdot \nabla f_i\right) + \vec{c}_i \cdot \nabla\left(\frac{\partial f_i}{\partial t} + \vec{c}_i \cdot \nabla f_i\right)\right] \quad . \end{aligned} \tag{3.25}$$

Next, a multi-scale expansion is performed on f_i with ϵ as the expansion parameter:

$$f_i = \sum_{n=0}^{\infty} \epsilon^n f_i^{(n)} \quad . \tag{3.26}$$

where $f_i^{(0)} = f_i^{(eq)}$. Since the equilibrium distribution must have the same moments for the collision invariants as f_i, the following relationships must hold:

$$\rho = \sum_{i=0}^{18} f_i^{(0)} \quad , \tag{3.27}$$

$$\rho\vec{u} = \sum_{i=0}^{18} \vec{c}_i f_i^{(0)} \quad , \tag{3.28}$$

$$3\rho RT = \sum_{i=0}^{18} |\vec{c}_{0i}|^2 f_i^{(0)} \quad , \tag{3.29}$$

and for $n > 0$:

$$0 = \sum_{i=0}^{18} f_i^{(n)} \quad , \tag{3.30}$$

$$0 = \sum_{i=0}^{18} \vec{c}_i f_i^{(n)} \quad , \tag{3.31}$$

$$0 = \sum_{i=0}^{18} |\vec{c}_{0i}|^2 f_i^{(n)} \quad . \tag{3.32}$$

The multiscale expansion is also applied to the temporal and spatial derivatives. The temporal derivative is expanded as:

$$\frac{\partial}{\partial t} = \sum_{n=0}^{\infty} \epsilon^n \frac{\partial_n}{\partial t} \quad , \tag{3.33}$$

and, as before, only one spatial scale is considered since convection and diffusion occur over different time scales, but over similar spatial scales. Thus, the spatial derivative is simply:

$$\vec{c}_i \cdot \nabla = \vec{c}_i \cdot \nabla_0 \quad . \tag{3.34}$$

The next step is to substitute equations 3.25, 3.26, 3.33, and 3.34 into the LBE and also replace Δt with ϵ. These substitutions give:

$$\begin{aligned}\epsilon\frac{\partial_0 f_i^{(0)}}{\partial t} + \epsilon^2\frac{\partial_1 f_i^{(0)}}{\partial t} + \epsilon(\vec{c}_i \cdot \nabla_0 f_i^{(0)}) + \epsilon^2\frac{\partial_0 f_i^{(1)}}{\partial t} + \epsilon^2(\vec{c}_i \cdot \nabla_0 f_i^{(1)}) \\ + \frac{\epsilon^2}{2}\frac{\partial_0}{\partial t}\left[\frac{\partial_0 f_i^{(0)}}{\partial t} + \vec{c}_i \cdot \nabla_0 f_i^{(0)}\right] + \frac{\epsilon^2}{2}\vec{c}_i \cdot \nabla_0\left[\frac{\partial_0 f_i^{(0)}}{\partial t} + \vec{c}_i \cdot \nabla_0 f_i^{(0)}\right] + \cdots \\ = -\hat{\omega}\left(f_i^{(0)} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)} + \cdots - f_i^{(eq)}\right) \quad . \end{aligned} \tag{3.35}$$

Grouping terms of similar order gives the following:

$$\frac{\partial_0 f_i^{(0)}}{\partial t} + \vec{c}_i \cdot \nabla_0 f_i^{(0)} = -\hat{\omega} f_i^{(1)} \tag{3.36}$$

for the first-order terms ($O(\epsilon)$), and

$$\frac{\partial_1 f_i^{(0)}}{\partial t} + \frac{\partial_0 f_i^{(1)}}{\partial t} + \vec{c}_i \cdot \nabla_0 f_i^{(1)} + \frac{1}{2}\left[\frac{\partial_0}{\partial t}\left[\frac{\partial_0 f_i^{(0)}}{\partial t} + \vec{c}_i \cdot \nabla_0 f_i^{(0)}\right] + \vec{c}_i \cdot \nabla_0 \left[\frac{\partial_0 f_i^{(0)}}{\partial t} + \vec{c}_i \cdot \nabla_0 f_i^{(0)}\right]\right] = -\hat{\omega} f_i^{(2)} \quad (3.37)$$

for the second-order terms ($O(\epsilon^2)$). Since the left-hand side of equation 3.36 appears in equation 3.37 it can be substituted to give the following simpler expression for the terms of second-order in ϵ:

$$\frac{\partial_1 f_i^{(0)}}{\partial t} + \left(1 - \frac{\hat{\omega}}{2}\right)\left[\frac{\partial_0 f_i^{(1)}}{\partial t} + \vec{c}_i \cdot \nabla_0 f_i^{(1)}\right] = -\hat{\omega} f_i^{(2)} \quad . \quad (3.38)$$

As with the Chapman-Enskog analysis of the continuous Boltzmann equation, the first- and second-order equations will be analysed separately to compute the first- and second-order components of the conservation equations.

First-Order Analysis

The first step in the analysis of the first-order terms in the LBE is to calculate their corresponding conservation equations by taking moments of equation 3.36. This gives the following equations for the conservation of mass and momentum:

$$\frac{\partial_0 \rho}{\partial t} + (\vec{u} \cdot \nabla_0)\rho = -\rho(\nabla_0 \cdot \vec{u}) \ , \tag{3.39}$$

and

$$\frac{\partial_0 \vec{u}}{\partial t} + (\vec{u} \cdot \nabla_0)\vec{u} = -\frac{1}{\rho}\nabla_0 \cdot \mathsf{P}^{(0)} \tag{3.40}$$

where:

$$\mathsf{P}^{(0)} = \sum_{i=0}^{18} \vec{c}_{0i}\vec{c}_{0i} f_i^{(0)} \ . \tag{3.41}$$

The first-order component of the total stress ($\mathsf{P}^{(0)}$) is simply a function of the equilibrium distribution, and can be evaluated directly. The first step is to substitute the definition of the peculiar velocity:

$$\mathsf{P}^{(0)} = \sum_{i=0}^{18} \vec{c}_i\vec{c}_i f_i^{(eq)} - 2\vec{u}\sum_{i=0}^{18} \vec{c}_i f_i^{(eq)} + \vec{u}\vec{u}\sum_{i=0}^{18} f_i^{(eq)} \ . \tag{3.42}$$

Next, the resulting moments can be calculated from the definition of the equilibrium distribution to give:

$$\mathsf{P}^{(0)} = \frac{\rho}{3}\mathsf{I} + \rho\vec{u}\vec{u} - 2\rho\vec{u}\vec{u} + \rho\vec{u}\vec{u} \ . \tag{3.43}$$

Finally, grouping like terms, the following expression is obtained for the first-order component of the total stress tensor:

$$\mathsf{P}^{(0)} = \frac{\rho}{3}\mathsf{I} \ . \tag{3.44}$$

Substituting the expression for the first-order component of the total stress tensor (equation 3.44) into the first-order conservation equations gives the following macroscopic conservation equations to first-order in ϵ:

$$\frac{\partial_0 \rho}{\partial t} + (\vec{u} \cdot \nabla_0)\rho = -\rho(\nabla_0 \cdot \vec{u}) \ , \tag{3.45}$$

and

$$\frac{\partial_0 \vec{u}}{\partial t} + (\vec{u} \cdot \nabla_0)\vec{u} = -\frac{1}{\rho}\nabla_0 p \ . \tag{3.46}$$

These are the Euler equations. Thus, the LBM recovers the same conservation equations to first-order as the continuous Boltzmann equation. The energy equation is not presented because the LBM models used in this work are athermal. Thus, the temperature in these models remains constant.

Second-Order Analysis

As in the first-order analysis, the second-order conservation equations can be determined by taking moments of the second-order terms of the LBE (equation 3.37). The mass and momentum moments give the following conservation equations:

$$\frac{\partial_1 \rho}{\partial t} = 0 \ , \tag{3.47}$$

and

$$\frac{\partial_1 \vec{u}}{\partial t} = -\frac{1}{\rho} \nabla_0 \cdot \mathsf{P}^{(1)} \tag{3.48}$$

where:

$$\mathsf{P}^{(1)} = \left(1 - \frac{\hat{\omega}}{2}\right) \sum_{i=0}^{18} \vec{c}_{0i} \vec{c}_{0i} f_i^{(1)} \ . \tag{3.49}$$

The next step is to evaluate the expression of the second-order component of the total stress given in equation 3.58. The sum can be expanded using the definition of the peculiar velocity:

$$\mathsf{P}^{(1)} = \left(1 - \frac{\hat{\omega}}{2}\right)\left[\sum_{i=0}^{18} \vec{c}_i\vec{c}_i f_i^{(1)} - 2\vec{u}\sum_{i=0}^{18} \vec{c}_i f_i^{(1)} + \vec{u}\vec{u}\sum_{i=0}^{18} f_i^{(1)}\right] \quad . \tag{3.50}$$

Since the conserved moments of $f_i^{(1)}$ are zero by definition, this expression reduces to:

$$\mathsf{P}^{(1)} = \left(1 - \frac{\hat{\omega}}{2}\right)\sum_{i=0}^{18} \vec{c}_i\vec{c}_i f_i^{(1)} \quad . \tag{3.51}$$

Substituting equation 3.36 gives the following expression for $\mathsf{P}^{(1)}$ in terms of $f_i^{(eq)}$:

$$\mathsf{P}^{(1)} = -\left(\frac{1}{\hat{\omega}} - \frac{1}{2}\right)\sum_{i=0}^{18} \vec{c}_i\vec{c}_i \left(\frac{\partial_0 f_i^{(eq)}}{\partial t} + \vec{c}_i \cdot \nabla_0 f_i^{(eq)}\right) \quad . \tag{3.52}$$

Next, the moments of $f_i^{(eq)}$ can be evaluated using the following definitions:

$$\sum_{i=0}^{18} \vec{c}_{i\alpha}\vec{c}_{i\beta} f_i^{(eq)} = \frac{\rho}{3}\delta_{\alpha\beta} + \rho u_\alpha u_\beta \ , \tag{3.53}$$

and

$$\sum_{i=0}^{18} \vec{c}_{i\alpha}\vec{c}_{i\beta}\vec{c}_{i\gamma}f_i^{(eq)} = \frac{\rho}{3}(\delta_{\alpha\beta}u_\gamma + \delta_{\alpha\gamma}u_\beta + \delta_{\beta\gamma}u_\alpha) \ . \tag{3.54}$$

This results in the following expression:

$$\mathsf{P}^{(1)} = -\left(\frac{1}{\hat{\omega}} - \frac{1}{2}\right)\left[\frac{1}{3}\frac{\partial_0\rho}{\partial t}\mathsf{I} + \frac{\partial_0\rho\vec{u}\vec{u}}{\partial t} + \frac{1}{3}\left(\nabla_0\rho\vec{u} + (\nabla_0\rho\vec{u})^{\mathsf{T}} + (\nabla_0\cdot\rho\vec{u})\mathsf{I}\right)\right] \ . \tag{3.55}$$

Using the product rule, the temporal derivative of $\rho\vec{u}\vec{u}$ can be expanded. This gives:

$$\begin{aligned}\mathsf{P}^{(1)} = -\left(\frac{1}{\hat{\omega}} - \frac{1}{2}\right)\Bigg[&\frac{1}{3}\frac{\partial_0\rho}{\partial t}\mathsf{I} + \rho\vec{u}\frac{\partial_0\vec{u}}{\partial t} + \left(\vec{u}\frac{\partial_0\rho\vec{u}}{\partial t}\right)^{\mathsf{T}} \\ &+\frac{1}{3}\left(\nabla_0\rho\vec{u} + (\nabla_0\rho\vec{u})^{\mathsf{T}} + (\nabla_0\cdot\rho\vec{u})\mathsf{I}\right)\Bigg] \ . \end{aligned}\tag{3.56}$$

Next, the first-order conservation equations (3.39 and 3.40) can be substituted for the the temporal derivatives. This gives:

$$\mathsf{P}^{(1)} = -\frac{1}{3}\left(\frac{1}{\hat{\omega}} - \frac{1}{2}\right)\big[-(\nabla_0 \cdot \rho\vec{u})\mathsf{I} + \nabla_0\rho\vec{u} + (\nabla_0\rho\vec{u})^{\mathsf{T}} + (\nabla_0 \cdot \rho\vec{u})\mathsf{I}$$
$$-\vec{u}\nabla_0\rho - (\vec{u}\nabla_0\rho)^{\mathsf{T}} + O(Ma^3)\big] \quad . \tag{3.57}$$

Grouping like terms gives the final expression for the total stress tensor:

$$\mathsf{P}^{(1)} = -\mu\left(\nabla_0\vec{u} + (\nabla_0\vec{u})^{\mathsf{T}}\right) \tag{3.58}$$

where:

$$\mu = \frac{\rho}{3}\left(\frac{1}{\hat{\omega}} - \frac{1}{2}\right) \quad . \tag{3.59}$$

It should be noted that, in light of the low-Mach-number expansion performed on the equilibrium distribution, the terms of $O(Ma^3)$ can be neglected. The resulting expression is the same as the second-order stress tensor found for the Boltzmann equation (equation 2.96) minus the velocity divergence term. The expression for the viscosity is similar to the expression found from the Boltzmann equation; however, the relaxation time ($1/\hat{\omega}$ is reduced by 1/2. This modification is all that is needed to make the LBM second-order accurate.

Substituting equation 3.58 into equations 3.47 and 3.48 gives the following second-order conservation equations for mass and momentum, respectively:

$$\frac{\partial_1 \rho}{\partial t} = 0 \ , \tag{3.60}$$

and

$$\frac{\partial_1 \vec{u}}{\partial t} = \frac{1}{\rho} \nabla_0 \cdot \left[\mu \left(\nabla_0 \vec{u} + (\nabla_0 \vec{u})^{\mathsf{T}} \right) \right] \ . \tag{3.61}$$

Combining the first- and second-order conservation equations (and setting the parameter ϵ to zero) gives the conservation equations:

$$\frac{\partial \rho}{\partial t} + (\vec{u} \cdot \nabla)\rho = -\rho(\nabla \cdot \vec{u}) \ , \tag{3.62}$$

and

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = \frac{1}{\rho} \left[-\nabla p + \nabla \cdot \mu \left(\nabla_0 \vec{u} + (\nabla_0 \vec{u})^{\mathsf{T}} \right) \right] \ . \tag{3.63}$$

These are the well-known Navier-Stokes equations (minus the $(1/\rho)\nabla\cdot\lambda(\nabla\cdot\vec{u})\mathsf{I}$ term). Thus, this analysis proves that the LBM (away from boundary nodes) recovers the proper macroscopic behaviour provided that the Mach number is sufficiently low such that the missing velocity divergence term and the truncated terms in the stress tensor are negligible.

3.2 Verification

Before any CFD code can be used to solve a complex problem it must be verified to ensure it has been implemented correctly [46]. To verify a code, a problem with an exact solution must be chosen that encompasses most of the important physical phenomena that will be captured in the simulation of interest. Thus, it is important to choose a problem, or a variety of problems, that excite all of the terms in the Navier-Stokes equations to completely verify the code.

Since the eventual goal was to simulate turbulent channel flow, the verification problems were chosen to test the LBM for flows driven by a body force and with periodic and wall boundary conditions.

3.2.1 Poiseuille Flow

The first problem used to verify the LBM was fully-developed laminar channel flow. This problem is important as a verification vehicle because it has exactly the same geometry as the simulation of turbulent channel flow (see figure 3.3). The analytical solution for laminar channel flow, or Poiseuille flow, is [51]:

$$u(z) = \frac{\delta^2}{2\mu}\left(\frac{\Delta p}{L_x}\right)\left(1 - \frac{z^2}{\delta^2}\right) \quad . \tag{3.64}$$

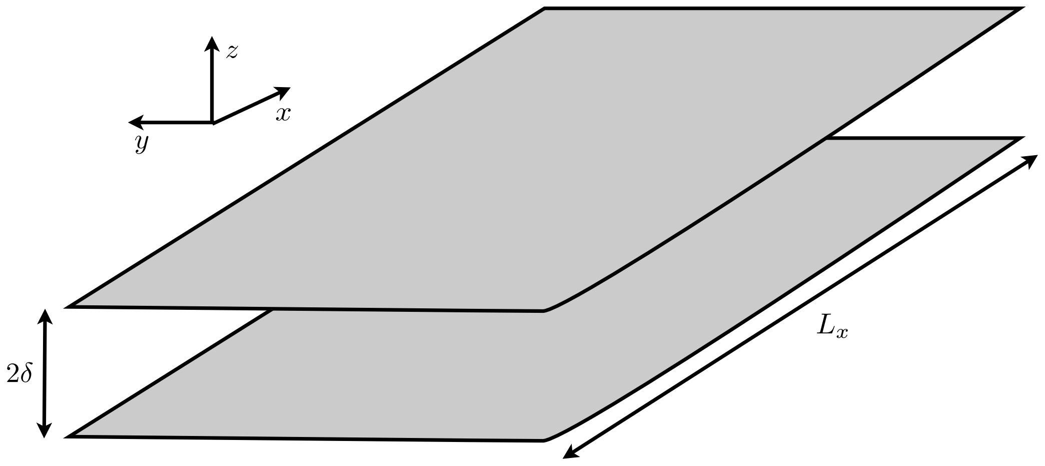


Figure 3.3: Poiseuille Flow Domain.

To examine this flow using the LBM, a three-dimensional domain was constructed with periodic boundaries in the streamwise and spanwise directions, and wall boundaries (bounce-back) in the wall-normal direction. Poiseuille flow only has gradients in the wall-normal direction, so the grid resolution in this direction is more important than in the other homogeneous directions. For the first case, 21 lattice sites were used to resolve the gradients in the wall-normal direction.

The simulation was conducted for $Re = 500$, and $Ma = 0.1$. The flow was driven by a body force, and thus the streamwise pressure gradient was zero. The required body force was calculated by rearranging the analytical solution to give an expression for the body force in terms of the Reynolds and Mach numbers:

$$g_x = \frac{2}{3\delta}\frac{Ma^2}{Re} \quad . \tag{3.65}$$

The particle distributions at each lattice site were initialized to their equilibrium values for zero velocity and unit density. There was no reason to perform the initialization procedure described in section 3.1.3 because this case does not require correct time-dependent results. The body force was applied to drive the flow, and the simulation was run until the velocity profile converged to a steady solution.

To make sure that the code gives the same result for flow oriented in all three different directions, the problems was run three times with the streamwise direction aligned with each of the three coordinate directions. The results for these simulations are compared with the exact solution in figure 3.4. The average error, which is defined as:

$$\overline{e} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{u^*(z_i^*)_{LBM} - u^*(z_i^*)_{exact}}{u^*(z_i^*)_{exact}}\right| \quad , \tag{3.66}$$

was equal to 7.6×10^{-4} for the case with 21 nodes in the wall-normal direction. This good agreement shows that the LBM has been properly implemented. The quantities in equation 3.66 with asterisks have been normalised by the largest scales in the flow, which are the centreline velocity and the channel half-width.

In order to determine the order of accuracy of the LBM, the number of nodes

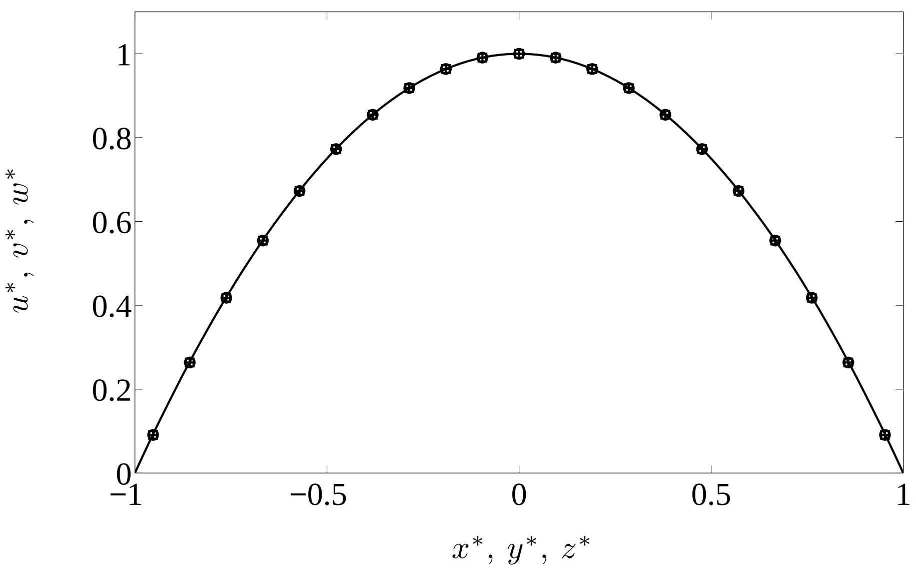


Figure 3.4: Verification of LBM for Poiseuille flow: ○ walls in the x-direction; + walls in the y-direction; □ walls in the z-direction; — exact solution [51]. The asterisks denote normalisation by the centreline velocity and the channel half-width.

in the wall-normal direction was varied to find the effect on the average error. The relationship between the number of nodes and the average error is shown in figure 3.5.

It should be noted that, in the LBM, the grid spacing cannot be changed independently of the time step. This is because the time step is defined by the time required for a particle distribution to reach its neighbouring lattice site. Therefore, if the grid resolution is halved, the time step is halved as well. For this reason, the simulations with varying grid resolution had to be run for the same amount of physical time. This means that the simulation for the coarse grid was run for fewer time steps than the finer resolution simulations.

When the average error and the number of nodes are plotted on a log-log scale (figure 3.5) the line-of-best-fit has a slope of approximately 2. This shows that the LBM

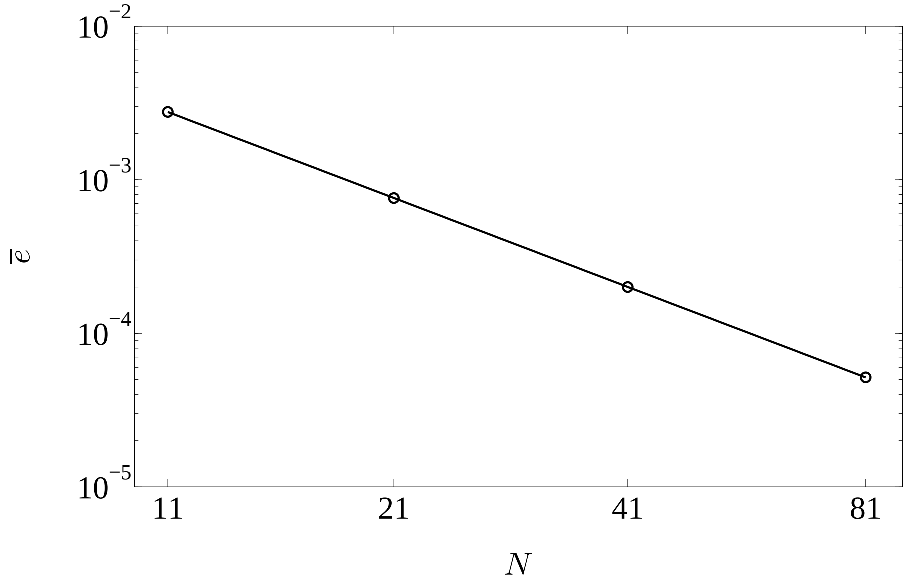


Figure 3.5: Effect of grid resolution on average error: ○ average error of LBM; — line of best fit.

is indeed second-order accurate in space as was predicted in section 3.1.5. Additionally, it demonstrates that the wall boundary conditions, while nominally first-order accurate, are in fact second-order accurate in this case since the flow is parallel to the wall nodes.

3.2.2 Duct Flow

Though the results from the verification with Poiseuille flow are encouraging, it is necessary to test a flow that has gradients in more than one direction. For this reason, fully-developed laminar flow in a rectangular duct was also solved with the LBM.

The simulation of duct flow is identical to Poiseuille flow except that wall boundaries are used in the spanwise direction as well. The geometry of the problem is shown in figure 3.6. The analytical solution for the streamwise velocity is given below [5]:

$$u(z,y) = \frac{1}{2\mu}\left(\frac{\Delta p}{L_x}\right)\left[b^2 - y^2 - \frac{4}{b}\sum_{n=0}^{\infty}\left((-1)^n \frac{1}{m^3}\frac{\cos(my)\cosh(mz)}{\cosh(ma)}\right)\right] \quad , \tag{3.67}$$

where:

$$m = \frac{(2n+1)\pi}{2b} \quad . \tag{3.68}$$

Since the analytical solution is an infinite series it cannot be evaluated exactly. Only terms larger than 10^{-13} were included in the analytical solutions that are compared with the LBM results.

For the flow inside a duct, the Reynolds number was 500, and the Mach number was 0.1. The required body force and viscosity were calculated for the desired Reynolds and Mach numbers using the exact solution.

First, a simulation was run for a square duct ($a = b$), with 41 lattice nodes used to resolve the gradients in both wall-normal directions. This grid resolution was chosen as a result of the verification work done for Poiseuille flow. The results for this simulation are plotted in figures 3.7 and 3.8.

In figure 3.7, the contours of streamwise velocity obtained from the LBM and the analytical solution are compared. The contour levels for the exact solution (left) and the LBM (right) are the same. Figure 3.8 shows the same data, but in a different

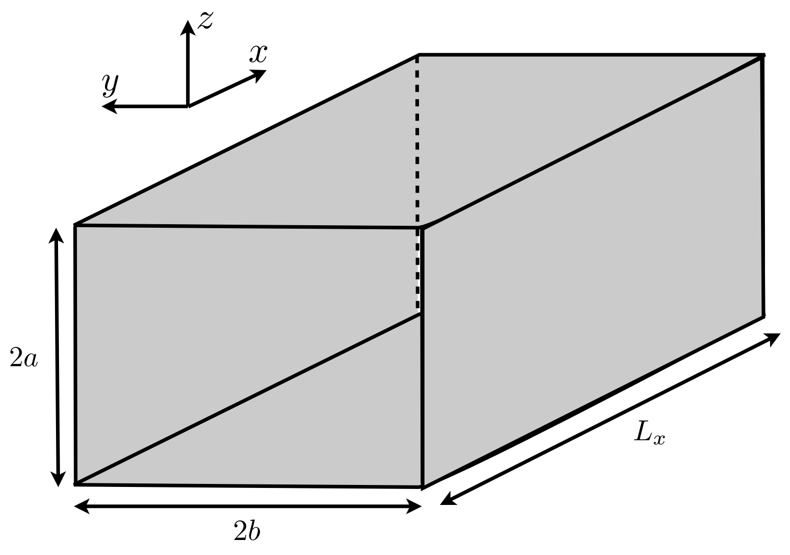


Figure 3.6: Duct Flow Domain.

format. In this figure, the velocity profile is plotted for various slices in the vertical wall-normal direction. The LBM results are shown as circles, and the exact solution is the solid line. Both of these figures demonstrate an excellent agreement between the LBM simulation and the analytical solution.

To further verify the performance of the LBM, another simulation was conducted for the flow in a duct with a rectangular cross-section. The reason for running this case was to ensure the LBM obtains the same result when $a \neq b$. This is an important test because it removes the diagonal symmetry and thus is a better example of a three-dimensional flow. Again, the Reynolds number and Mach number were kept constant, and the number of nodes used in the short direction was 31. The number of nodes used for the long side was 51, which corresponds to the same grid resolution as for the short side.

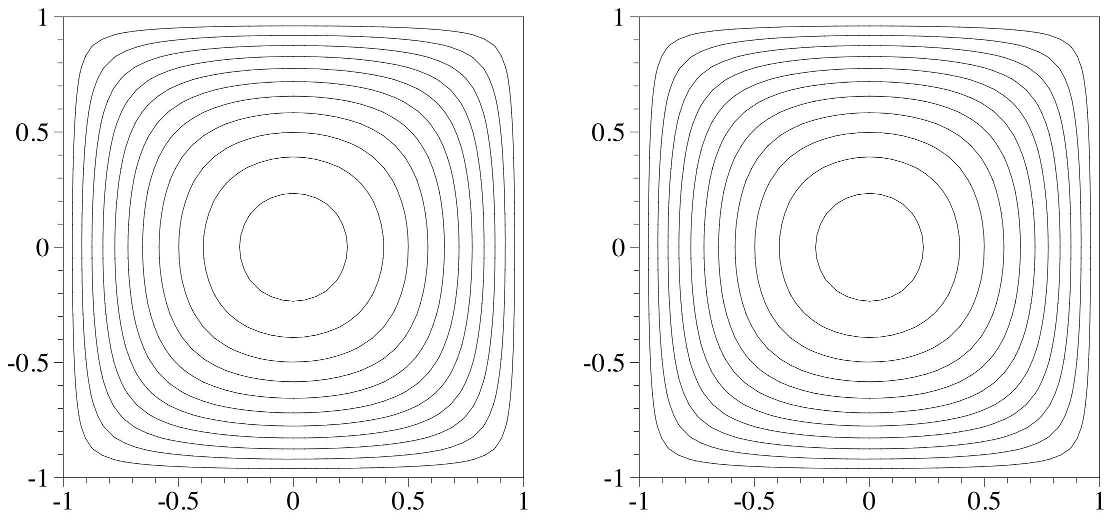


Figure 3.7: Contours of streamwise velocity (u^*) for laminar flow in a square duct.: (left) analytical solution [5]; (right) LBM. Contour levels are evenly distributed from 0 to 1.

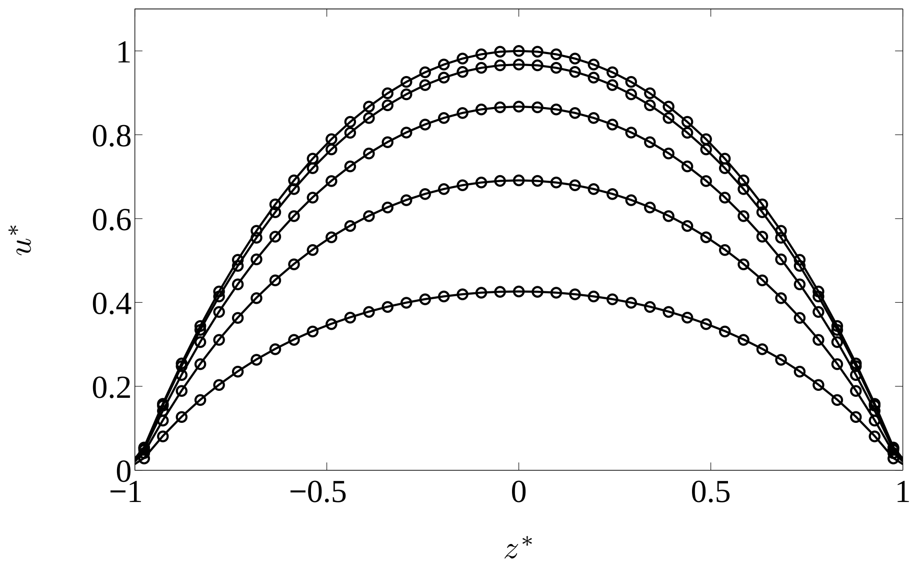


Figure 3.8: Streamwise velocity (u^*) profiles for laminar flow in a square duct.: ○ LBM; — analytical solution [5]. The asterisks denote normalisation by the centreline velocity and the channel half-width.

Figures 3.9 and 3.10 compare the results of the LBM simulation with the analytical solution. Figure 3.9 compares the contours of streamwise velocity for both the LBM and the analytical solution. The contour levels used to create these plots are the same. Figure 3.10 shows the velocity profiles in the “short” wall-normal direction for a number of locations along the “long” side of the duct. Both of these figures show excellent agreement between the LBM and the analytical solution.

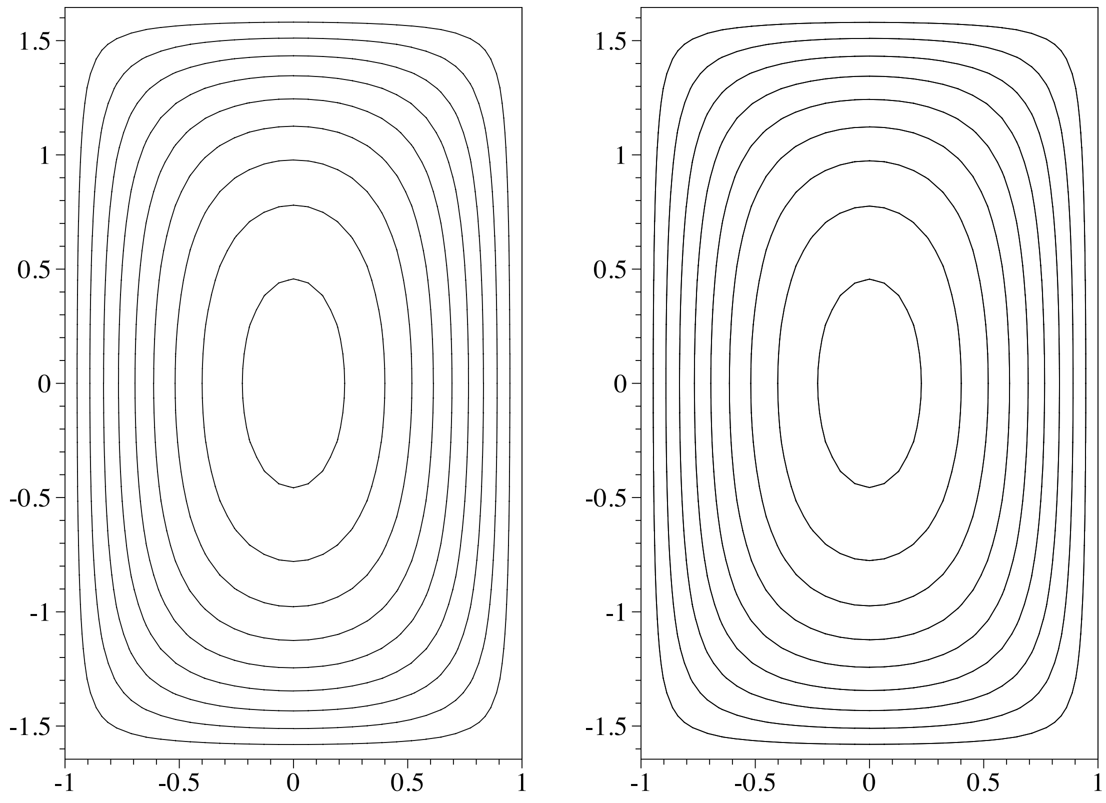


Figure 3.9: Contours of streamwise velocity for laminar flow in a rectangular duct.: (left) analytical solution [5]; (right) LBM. Contour levels are evenly distributed from 0 to 1.

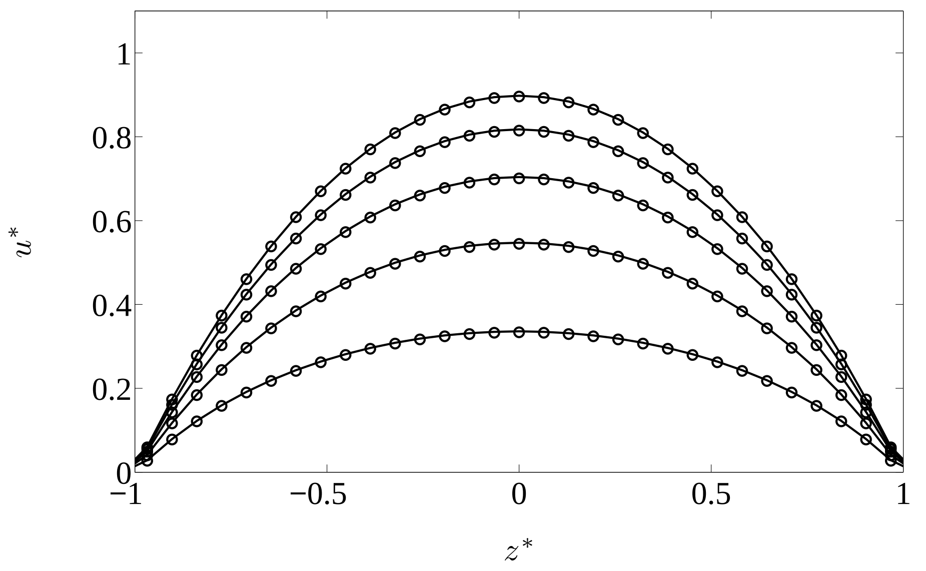


Figure 3.10: Streamwise velocity (u^*) profiles for laminar flow in a rectangular duct: ∘ LBM; — analytical solution [5]. The asterisks denote normalisation by the centreline velocity and the channel half-width.

Chapter 4

Parallelization

In chapter 3 the lattice Boltzmann method was introduced as an efficient numerical method that can be used to obtain approximate solutions to the Boltzmann equation. Additionally, the Chapman-Enskog multiscale expansion was performed on the LBM to show that it indeed recovers the hydrodynamic behaviour of the Navier-Stokes equations in the macroscopic limit. Finally, the implementation of the LBM was verified by solving a number of canonical problems and comparing the results to their corresponding analytical solutions. The next step in the development of an LBM code for direct numerical simulation of turbulent flows is parallelization, which will be discussed in this chapter.

As stated in chapter 1, direct numerical simulation of turbulent flows is very computationally intensive due to the wide range of scales that must be resolved. The serial computers currently available are not fast enough to practically compute even a small DNS. This is because there are a number of factors that place physical limitations on the performance of single processors. The speed at which data can be transmitted is limited by the speed of light, and microchips cannot be made any smaller due to limitations in the effectiveness of heat dissipation. Additionally, the cost of approaching any of these physical limitations is very expensive. For this reason, it is much more economical to perform direct numerical simulations on parallel computers.

The first section in this chapter discusses a number of concepts related to parallelization. These concepts are helpful in understanding the implementation details that are discussed in the second section. Finally, the last section discusses the parallel performance of the LBM.

4.1 Background

There are three main things that need to be considered when creating a parallel computer code: the amenability of the algorithm to parallelization, the architecture of the parallel computer, and the application programming interface that will be used to implement the parallelization. This section provides a brief introduction into each of these topics so that the reader may understand the basis for some of the implementation decisions made in this work.

4.1.1 Amdahl's Law

The total speed-up of a simulation is defined as the ratio of the wall-clock time required to run the serial version of the code (t_1) to the ratio of the wall-clock time required to run the code with n processors (t_n). The theoretical maximum speed-up is dictated by Amdahl's Law:

$$\left(\frac{t_1}{t_n}\right)_{max} = \frac{t_s + t_p}{t_s + t_p/n} = \frac{n}{(n-1)s_f + 1} \tag{4.1}$$

where: t_s is the time required to run the portion of the code that must run in serial, t_p is the time required to run the portion of the code that can be run in parallel, and s_f is the fraction of the code that must be run in serial ($s_f = t_s/(t_s + t_p)$). Amdahl's law describes the theoretical maximum speed-up that can be achieved with a particular algorithm; it does not consider any details of the parallel implementation. The speed-up predicted by Amdahl's law is never achieved in practice, but Amdahl's law is still useful as the first step in determining whether or not an algorithm is suitable for parallelization.

For algorithms with a serial fraction equal to zero, the speed-up scales linearly with the number of processors. In other words, if the number of processors is doubled, the speed-up increases by a factor of two. These algorithms are considered "embarrassingly parallel" and are the most amenable to parallelization. As the serial fraction increases, however, the scalability of the algorithm is drastically reduced. Figure 4.1 shows the scaling curves predicted by Amdahl's law for three different algorithms with serial fractions of 0.1%, 1%, and 10%. The algorithm with $s_f = 0.1\%$ scales almost linearly up to 256 processors, while the algorithm with $s_f = 1\%$ shows a significant departure from linear scaling at 32 processors, and the algorithm with $s_f = 10\%$ barely scales past 4 processors.

The reason that certain portions of an algorithm must be run in serial is because of data dependencies in the algorithm. For example, if the iterations of a loop must be run in a particular order, the loop has a data dependency and cannot be parallelized because, if the work is divided among several processors, the order of the original loop will not be preserved. Thus, this section of the code will have to be run in serial in order to obtain the correct result.

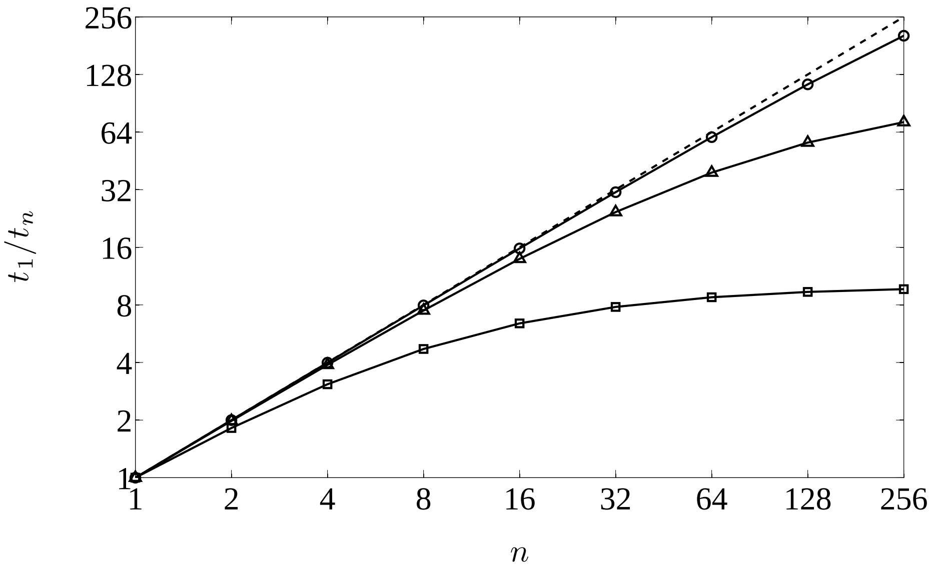


Figure 4.1: Amdahl's law. Predicted speed-up for algorithms with various serial fractions: - - - linear speed up; —○— $s_f = 0.1\%$; —△— $s_f = 1\%$; —□— $s_f = 10\%$.

Amdahl's law shows that some algorithms are more amenable to parallelization than others. If an algorithm has a high serial fraction, it will not scale to a large number of processors. If this is the case, there are two options: either parallelization should not be pursued, or the algorithm must be modified in order to reduce the serial fraction.

4.1.2 Parallel Computer Architectures

There are many types of parallel computers, but most modern machines fall into one of two categories: shared memory or distributed memory. Both of these architectures have various advantages and disadvantages that will be discussed in this section.

The key feature of a shared memory machine is that each of the multiple processors has access to a large common memory. Each processor can execute its own individual instructions but shares its memory with all of the other processors. A schematic of a typical shared memory architecture is shown in figure 4.2.

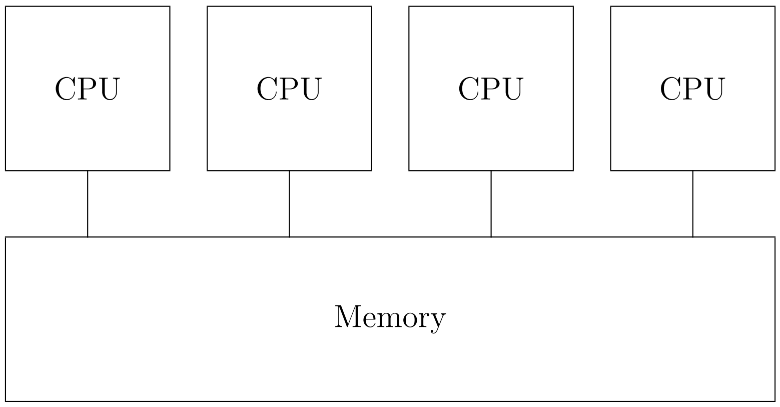


Figure 4.2: Shared memory architecture.

The main advantage of the shared memory architecture is the speed at which data can be shared among processors. Since each processor has access to the global memory there is no need to transfer data between processors. This significantly reduces the parallel overhead of the computation since message passing on a network can be time consuming. The shared memory also makes application programming interface (API) development very user-friendly because explicit instructions for sharing data between processors are not needed. The memory access, from the point of view of the programmer, is the same as for a serial machine. The only added complication is that the user needs to explicitly handle the synchronization between processors in order to ensure that the data in the shared memory gets updated in the proper order.

The shared memory architecture makes implementing parallel code relatively simple, but this comes at the cost of more complicated hardware. Since the memory is shared, special hardware is required to make sure that each processor knows when another processor has read data into its cache. If this were not implemented, it would be possible for multiple processors to be working on the same data at the same time. Because of the sophisticated hardware required for a shared memory machine, their cost increases rapidly as the number of processors is increased. This leads to a high cost per floating point operation (a standard measure of computational power).

Distributed memory machines are created by connecting serial computers in a network as shown in figure 4.3. Each processor only has direct access to its own memory and if any data is needed from another processor's memory it must be transferred over the network.

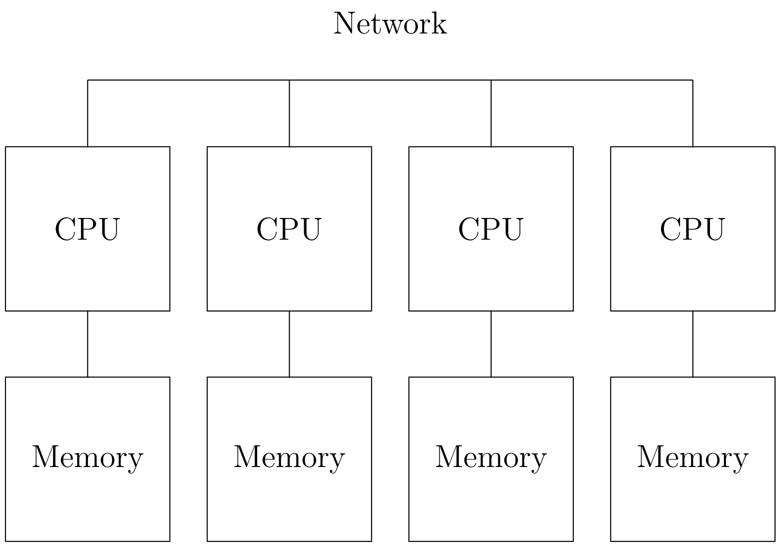


Figure 4.3: Distributed memory architecture.

The most attractive feature of distributed memory machines is their simplicity and cost effectiveness. Distributed memory machines can typically be made with standard "off-the-shelf" components, which makes them very inexpensive. A simple distributed memory machine can be constructed by simply connecting a number of personal computers together using Ethernet. The cost effectiveness of distributed memory machines allows for the construction of very large parallel machines with many processors, and, since the memory scales with the number of processors, lots of memory. This is why most of the largest high performance computers in existence are distributed memory machines.

The disadvantage of distributed memory machines is the cost associated with sharing data on a network. There are two criteria that dictate the speed at which data can be transferred between CPUs: latency and bandwidth. The latency is the time it takes for the first byte of data to be sent between processors and the bandwidth is the rate a which the following bytes arrive. If a lot of data is being transferred, the bandwidth is the most important parameter since the time it takes to send the message will be large in comparison to the time it takes to send the first byte. However, the latency becomes significant when sending a large number of small messages. In order for a distributed memory machine to have a good parallel performance, the network must have a low latency and a high bandwidth. This is achieved by using complex network topologies and interconnects, which add to the cost of the machine.

Another disadvantage of a distributed memory machine is that the transfer of data between processors, or “message passing”, must be handled explicitly by the programmer. This can become very complicated depending on the network topology. For this reason, developing parallel APIs can be difficult for distributed memory machines.

It is important to consider the computer architecture when designing a parallel code. Some algorithms are better suited to one architecture over another. Thus, the parallel computer chosen to run the code might be chosen for its architecture. For example, if a code requires a lot of memory and floating point operations, but does not need much synchronization, a distributed memory machine will be preferable. If the code is to be run on an existing parallel computer, the parallelization can be implemented in a way that maximizes its performance for that architecture.

4.1.3 Application Programming Interfaces

In order to run a code on a parallel computer, the code must be parallelized using a parallel API. The two most common APIs available today are OpenMP, and MPI.

OpenMP, which stands for Open Multiprocessing, is an API that provides extensions to C/C++ and Fortran to create programs that can run in parallel on shared memory machines. OpenMP uses a multithreading approach to parallelization by which tasks are split up and worked on in parallel by a number of threads. The master thread executes the algorithm in serial until it reaches a “fork”, which is specified using preprocessor directives, where it spawns a set of slave threads that share the workload. Once the task has been completed, the slave threads “join” together again and the master thread continues to execute the program until the next fork is reached.

The most common approach to parallelization in OpenMP is called loop level parallelism. In this approach, the iterations in a loop are distributed over multiple threads and executed in parallel. This is the simplest way to parallelize a code with OpenMP as it often requires very little modification of the serial code and the distribution of the work among threads is generally handled automatically. Multiple sections of code can executed in parallel by using parallel sections. This approach is often more efficient, but it usually requires significant modification of the serial code and the distribution of the work must be specified by the programmer.

When writing parallel code with OpenMP, the programmer is responsible for explicitly controlling the synchronization of the threads. This involves rewriting code to remove any data dependencies or using special OpenMP directives to handle them efficiently. Since OpenMP is implemented on shared memory machines, special care must also be taken to avoid race conditions. Race conditions occur when multiple threads must compete to update the same location in memory. If a race condition cannot be removed by rewriting the code, there are special directives available within OpenMP to deal with them; however, the parallel efficiency usually suffers.

The main advantage of OpenMP is its simplicity. Serial codes can be modified to run in parallel with very few extra commands and sometimes with no modification of the original serial code. This means that the time required for developing parallel code is minimal and one version of the code can be developed for use on serial and parallel computers. The disadvantage of OpenMP is its portability. Since OpenMP codes can only be run on shared memory machines, they cannot be run on a large majority of parallel computers. This means that, if the code is to be run on a distributed memory machine, a separate version must be developed.

The Message Passing Interface (MPI) is the most commonly used API for parallel programming. Like OpenMP, MPI can be used to parallelize C/C++ and Fortran programs, but MPI codes can be run on both shared memory and distributed memory machines. MPI provides a set of library functions that allow the programmer to control the communication between individual processors. The programmer is responsible for explicitly specifying the transfer of data between processors and the synchronization.

MPI allows the user to create a virtual network topology that is used to transfer data between processors. The specific MPI implementation used on a particular parallel computer is then responsible for mapping this virtual topology onto the actual network. This feature makes MPI programs extremely portable and makes programming with MPI much simpler. The MPI library includes functions to implement simple point-to-point communication between individual processors, functions that can "gather" or "scatter" information to and from a master processor, and other functions that can be used to implement synchronization.

The main advantages of using MPI are its portability and flexibility. Since code written with MPI can be run on both shared memory and distributed memory machines, the same version of the code can be run on almost all modern parallel computers. Additionally, MPI codes are often very efficient since MPI allows the programmer to control almost every detail of the communication between processors. Thus, the programmer has the flexibility required to create highly-optimized parallel codes. The disadvantage of MPI is that a large number of calls to library functions are required to create a parallel code, and a serial code often must be completely rewritten in order for it to be parallelized. This leads to a large increase in the complexity of the code from the serial version, and a very long development time.

4.2 Implementation

The first task in creating a parallel LBM code is analysing the algorithm to determine if it is suitable for parallelization. This involves searching for data dependencies and trying to determine if they can be removed by altering the algorithm.

Since the LBM is a time marching algorithm, each time step must be done in order. This is because the solution at $t = n\Delta t$ is the initial condition for the solution at $t = (n + 1)\Delta t$. Clearly the solution will be different if the order of the time steps is altered. This means that the outer loop shown in figure 3.2 is not parallelizable.

Each iteration of the outer loop involves the collision of the particle distributions, the bounce-back of the wall nodes, the streaming of particle distributions to their neighbouring lattice sites, and the computation of the new moments at each lattice node. The collision step is clearly parallelizable since the post-collision distribution ($\tilde{f}_i$) can be calculated at each node independently. Thus, the iterations in the collision loop can be run in any order and the same result will be obtained. The bounce-back and moment loops are also parallelizable for the same reason. The streaming step, however, does have a data dependency. The streaming of the particle distributions must be done in a specific order to make sure that the upstream particle distributions are not overwritten before they are streamed to their neighbouring nodes. Fortunately, this data dependency can be easily removed with auxiliary storage. As long as the post-streaming particle distributions are written to a separate array, the iterations of the streaming step can be done in any order.

Even though the outer loop of the LBM is not parallelizable, each of the inner loops can be parallelized. Thus, the serial fraction for the iterative portion of the LBM is theoretically zero. And, since the number of lattice sites is typically quite large, there are lots of iterations in the inner loops so the work can be split over many processors. For this reason, the LBM algorithm, according to Amdahl's law, is an ideal candidate for parallelization.

The computational resources for this work were provided by the High Performance Computing Virtual Laboratory (HPCVL). There are a number of high performance computers at HPCVL, all of which are shared memory machines. The Sun SPARC Enterprise M9000 Servers were chosen due to their ability to perform a large number of floating point operations. Each of the M9000 servers has 64 quad-core 2.52 GHz

SPARC64 VII processors in a shared memory arrangement with 2 TB of main memory. Each of the processing cores of the SPARC64 VII processors is capable of running 2 hardware threads. This means that each of the M9000 servers can run 512 threads simultaneously. It should be noted that each hardware thread shares the computing resources on a single core with another thread. The advantage of this design is that the computational resources on each core are used more efficiently. The disadvantage is that the hardware threads are not as powerful as individual CPU cores. However, in order to improve the performance of codes that require a lot of floating point operations, such as the LBM, each core was designed with two floating point units to ensure that the hardware threads do not have to wait to compute floating point operations.

Since the Sun SPARC M9000 servers have a shared memory architecture, it is possible to parallelize the code using either OpenMP or MPI. When deciding between these two APIs, the main factors considered were portability and simplicity. The advantage of using MPI is that the code could later be used on almost any parallel computer since MPI code can be compiled for both shared and distributed memory machines. Since shared memory machines are relatively rare in comparison to distributed memory machines, using MPI would significantly improve the portability of the code. However, parallelizing an algorithm with MPI is significantly more complex than with OpenMP.

Given that there were no immediate plans to run the code on a distributed memory machine, it was determined that simplicity of parallelization was much more important than portability. For this reason, the decision was made to use OpenMP instead of MPI. The development of a version of the code that can be run on distributed memory machines is left for future work.

As mentioned earlier, the simplest way to parallelize an existing serial code with OpenMP is to parallelize the individual loops. This approach, which is referred to as "loop level parallelism", divides up the iterations in a loop and assigns them to different processors. When the master thread reaches a parallel loop, a number of slave threads are created to execute the iterations. When all of the work is completed, the slave threads are destroyed and the master thread continues executing the program.

The drawback of loop level parallelism is the cost of creating and destroying threads. Loop level parallelism is only efficient when the time spent in the loop greatly outweighs the time spent managing threads. For large simulations that are run with a small number of threads the time required for thread management is usually insignificant. However, since the time required to create and destroy threads does not scale with the number of threads used, the time required for fork and join operations becomes more and more significant as the number of threads is increased. Thus, the speed-up of a code parallelized in this way is poor for simulations with a large number of threads.

As mentioned earlier, the outer loop of the LBM (see figure 3.2) has a data dependency and cannot be parallelized. This means that each of the four inner loops: collision, bounce-back, streaming, and moment evaluation must be parallelized. This is very inefficient because it requires four fork and join operations for every time step. When the number of processors is increased, the overhead of thread management will dominate the execution time.

The other option for parallelization in OpenMP is parallel sections. The advantage of this approach is that whole sections of code can be run in parallel without the need for constantly creating and destroying threads. For this reason, parallel sections are much more efficient for parallelizing the LBM. In this work, the entire outer loop of the LBM algorithm (figure 3.2) was placed in a parallel section, and thus all of the code inside the loop was run in parallel. In order for this to work, the code inside the outer loop had to be slightly changed so that it could be divided amongst the threads. Additionally, barriers had to be placed between the subroutines to make sure that the threads were properly synchronized. When a barrier is encountered by one of the threads it halts the execution of the code until all of the threads reach the barrier. This ensures, for example, that the bounce-back boundary condition is completed for all wall nodes before the streaming operation is started. If the streaming step is started before the bounce-back is finished, it is possible that a particle distribution could be streamed from the wall before it has been bounced back.

A domain decomposition approach was used in which the computational mesh was split into sections and one section was assigned to each processor. Given that the particle distributions are stored in a vector, the easiest way to decompose the domain is to evenly divide the nodes amongst the processors. This is called a one-dimensional domain decomposition. There are a number of advantages to using this method. The first is that all of the nodes given to each processor are contiguous in memory. Since data is read from memory in cache lines, this reduces the number of times data must be read from main memory, which is important because reading data from the cache is an order of magnitude faster than from main memory. The second benefit of using a one-dimensional decomposition is load balancing. The speed-up is dictated by the processor with the most work, so it is important to distribute the nodes as evenly as possible. The one-dimensional decomposition guarantees that the number of nodes assigned to each processor will only be different by at most one. Considering that the number of nodes in a DNS is usually greater than $O(10^6)$, the imbalance between processors is typically far less than 1%.

The disadvantage of the one-dimensional linear decomposition approach is that it usually results in domains that have large surface areas. This is an important issue for codes parallelized with MPI that run on distributed memory machines because the surface area between domains is proportional to the interprocessor communication required. However, since this code is designed for shared memory machines, communication between processors is not important. The only possible downside to large surface areas in an OpenMP code is the possibility of cache thrashing. This occurs when two threads are competing to update a memory address on the same cache line. In order to determine whether or not the surface area of the domains affects the scaling of an LBM code parallelized with OpenMP, Bespalko *et al.* [7] compared the performance of an LBM code with a linear decomposition to the performance of the same code with a domain decomposed with the graph partitioning software METIS [28]. METIS is an algorithm that finds a compromise between minimizing the surface area and balancing the number of nodes assigned to each domain. Bespalko *et al.* found that reducing the surface area of the domains made no significant difference in the parallel performance of the LBM. For this reason, the one-dimensional decomposition is used in this work for simplicity.

4.3 Performance

In order to determine the parallel performance of the LBM code, a performance test was carried out on the Sun SPARC M9000 Servers at HPCVL. The same simulation was run with an increasing number of threads and the execution time was recorded in order to compute the speed-up.

It is important to note that the scalability of the a code depends on the type of problem it is used to solve. One of the most important factors that contributes to the scalability of an LBM code is the number of computational nodes. This is because the synchronization does not scale with the number of nodes. Thus, as the number of computational nodes is increased, the significance of the synchronization is reduced. For this reason, the simulation parameters for the scaling test were chosen to give the scaling performance of a typical DNS.

The problem chosen for the scaling test was plane channel flow. This type of flow will be described in more detail in the next section; for the purposes of the scaling test, the only important feature is the computational grid. The simulation domain had the dimensions $12\delta \times \delta \times 2\delta$ in the x, y, and z directions, respectively, where δ is the half-width of the channel. This domain was meshed with a uniform cubic grid with a grid resolution of $\Delta x^+ = \Delta y^+ = \Delta z^+ = 2$. This results in a computational grid with $1080 \times 90 \times 182$ nodes in the x, y, and z directions, respectively, for a total of 17690400 computational nodes.

Figure 4.4 shows the wall-clock time required per time step for the LBM code with 2, 4, 8, 16, 32, 64, 128, and 256 threads. The code was run for 1000 time steps in each case to find the average wall-clock time needed per time step. The wall-clock time for 2 threads is approximately 20 seconds. When 256 threads are used to execute the code, a time step is completed in just under half a second, which corresponds to a 40 times speed-up. This is significant, particularly if the code is run for a large number of time steps. Direct numerical simulations typically need to be run for many time steps in order to accumulate enough samples to converge the turbulence statistics. The turbulent channel flow simulation used in this test would typically require roughly 750000 time steps to properly converge the turbulence statistics. If this code were run with 2 processors, the simulation would take just under 6 months to complete. However, the same simulation can be completed in just under 4 days if it is run with 256 threads.

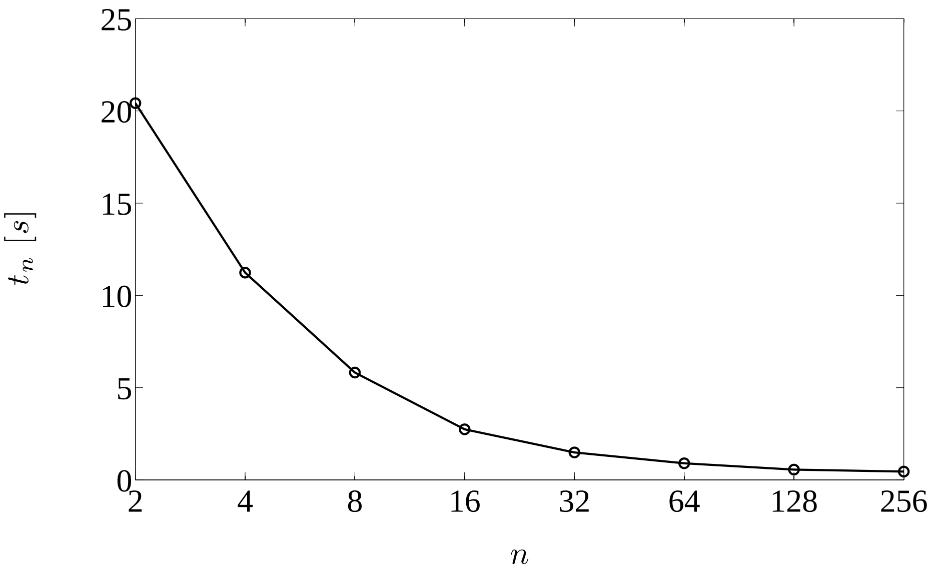


Figure 4.4: Average computational time for a single time step of the LBM.

For the cases tested, the execution time decreases monotonically with the number of threads used. This means that there was always a benefit to using more threads (in the range of threads tested) since adding threads always resulted in more speed-up. However, the efficiency of the code decreased significantly as the number of threads increased. The wall-clock time required per node is plotted in figure 4.5. This was computed by dividing the wall-clock time in figure 4.4 by the number of nodes in a single subdomain. The execution time per node is approximately $2.5 \times 10^{-}6s$ if the number of threads is between 2 and 32. Once more than 32 threads are used, the computational time required per node increases sharply. For the case with 256 threads, the per-node computational time required is $6.5 \times 10^{-}6s$, which is more than double the time required for 32 threads. The number of nodes in each subdomain for the 32 thread case is 552825. These results suggest that domains of at least this size are needed before the synchronization time is on the same order as the time needed for the computations.

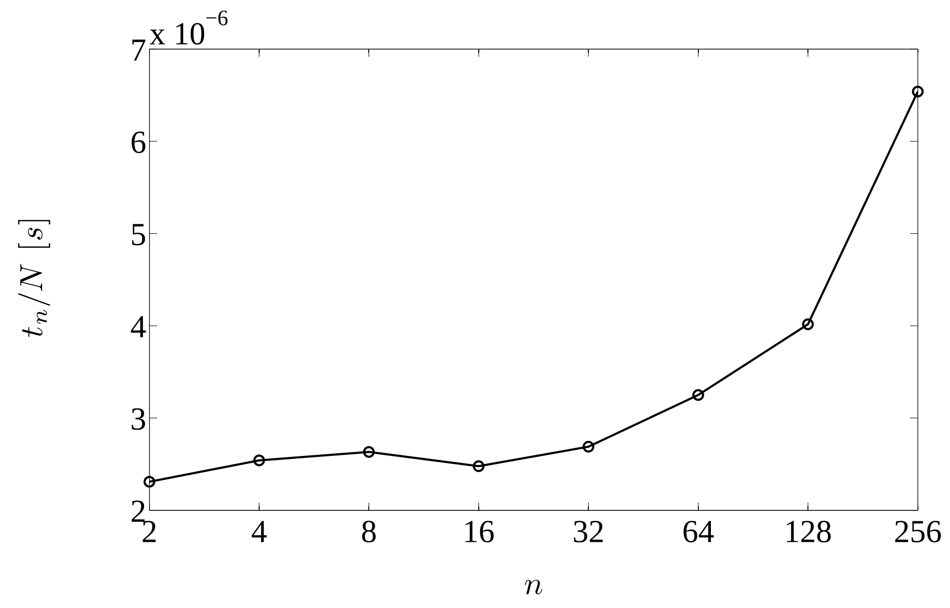


Figure 4.5: Average computational time per node for the LBM.

The decrease in efficiency of the code can also be shown by plotting the speed-up as a function of the number of threads (figure 4.6). When plotted on a logarithmic scale, the speed-up for the LBM increases linearly up to 32 threads. After this point the slope of the speed-up curve starts to decrease. Eventually, the speed-up will reach a maximum and, after this point, running the code with more threads will actually increase the execution time.

As mentioned earlier, the performance results presented are specific to the problem solved. The channel flow simulation tested scaled linearly up to 32 threads. If the size of the problem is increased the scalability is expected to improve. As long as the problem is large enough such that each subdomain contains more than 5×10^5 nodes, linear scaling is expected.

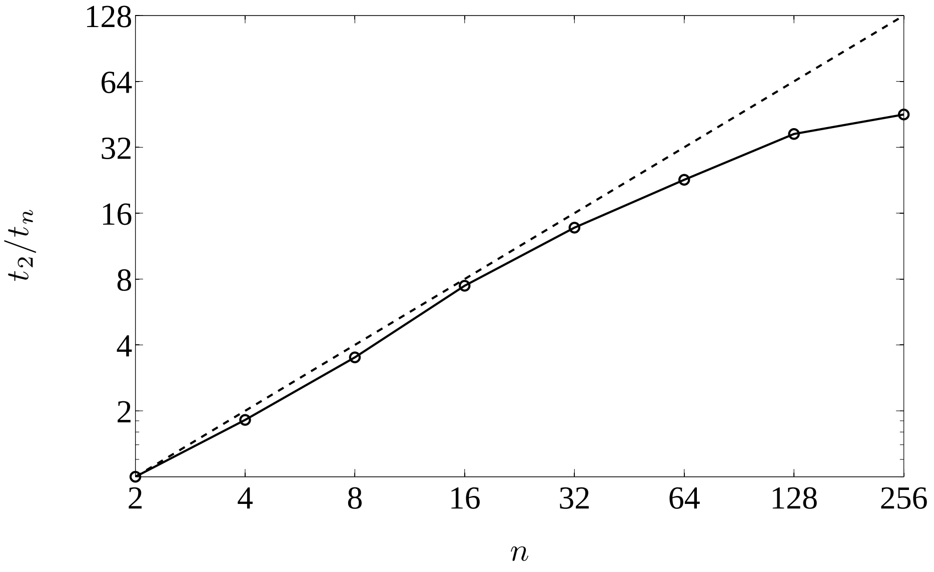


Figure 4.6: Strong scaling plot for the LBM.

Chapter 5

Methodology

In the previous chapters, the theory of the Boltzmann equation has been presented and a numerical method, the LBM, has been derived to find approximate solutions. This method was implemented in Fortran and was parallelized for use on high-performance computers with shared memory architectures.

This chapter outlines the methodology employed to validate the LBM. The first section contains a discussion of turbulent channel flow. This includes an analysis of the governing equations, important scaling parameters, and a derivation of the form of the mean velocity profile. The second section outlines the parameters used in the LBM simulation of channel flow and discusses the factors that lead to the chosen values. The parameters for the FD simulation are discussed in the final section.

5.1 Turbulent Channel Flow

As mentioned in section 1.2, fully-developed turbulent channel flow was chosen as the validation problem for this study. Turbulent channel flow has a simple geometry, boundary conditions that are simple to apply with the LBM, and a large amount of experimental and DNS data in the literature with which to compare the results.

The geometry of fully-developed channel flow, or flow between two parallel plates, is sketched in figure 5.1. If the length and width of the plates is very large in comparison to their separation distance ($a \gg \delta$ and $b \gg \delta$) the end effects can be ignored and the flow can be considered homogeneous in the x- and y-directions.

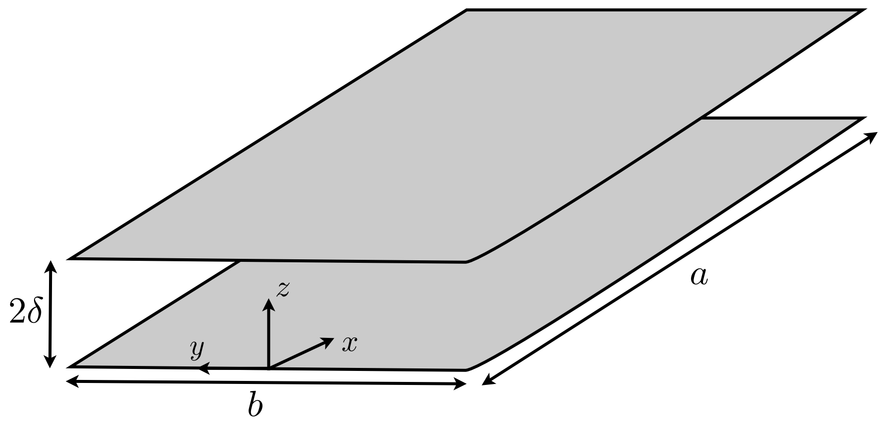


Figure 5.1: Geometry for fully-developed turbulent channel flow.

The flow is driven by a mean pressure gradient in the streamwise (x) direction. The motion caused by the pressure gradient is resisted by the shear stress caused by the no-slip condition on the surface of the two plates. When these two forces equilibrate, the turbulence statistics are no longer time-dependent and the flow is said to be statistically stationary.

5.1.1 Governing Equations

The motion of the fluid between the two parallel plates is governed by the incompressible Navier-Stokes equations:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \ , \tag{5.1}$$

and,

$$\frac{\partial}{\partial t}(\rho \vec{u}) + \nabla \cdot (\rho \vec{u}\vec{u}) = -\nabla p + \nabla \cdot \tau \ , \tag{5.2}$$

where τ is the viscous stress tensor:

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \ . \tag{5.3}$$

These equations will yield laminar solutions if the Reynolds number, based on the bulk velocity and the channel width, is less than approximately 1350. If the Reynolds number is greater than 1800, small disturbances in the flow will amplify and cause the flow to transition to turbulence [43].

Turbulent flow is characterized by unsteady, chaotic motions superimposed on a mean flow. Thus, the flow field can be broken down into its mean and fluctuating components:

$$\vec{\phi} = \left\langle \vec{\phi} \right\rangle + \vec{\phi}' \ , \tag{5.4}$$

where ϕ is a property of the flow field, such as velocity or pressure. The bracketed term is the ensemble average, and the primed term is the fluctuating component. Separating the mean and fluctuation component in this way is called a Reynolds decomposition. Replacing the velocity and pressure in equations 5.1 and 5.2 with their Reynolds decompositions and ensemble averaging the equations leads to the Reynolds equations [43]:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \langle \vec{u} \rangle) = 0 \ , \tag{5.5}$$

and

$$\frac{\partial}{\partial t}(\rho \langle \vec{u} \rangle) + \nabla \cdot (\rho \langle \vec{u} \rangle \langle \vec{u} \rangle) = -\nabla \langle p \rangle + \nabla \cdot (\langle \tau \rangle - \rho \langle \vec{u}' \vec{u}' \rangle) \ . \tag{5.6}$$

The Reynolds equations are very similar to the NS equations except for the addition of the Reynolds stress term ($\rho \langle \vec{u}' \vec{u}' \rangle$). The Reynolds stresses are not actually stresses;

they are mean momentum flux terms that arise due to the fluctuating component of the velocity field. However, in the Reynolds equations, it is convenient to refer to these terms as stresses because they affect the mean flow in the same way as the viscous stresses do.

A number of important features of turbulent channel flow can be determined by analysing the Reynolds equations, starting with equation 5.5. Since the flow is fully-developed, all spatial derivatives in the streamwise direction are zero with the exception of the pressure. The derivatives in the spanwise direction also vanish because there is no mean flow in this direction. Applying these simplifications gives the following continuity equation for channel flow [43]:

$$\frac{\partial \langle w \rangle}{\partial z} = 0 \ . \tag{5.7}$$

Integrating this equation and applying the no-slip wall boundary condition ($\langle w \rangle|_{z=0} = 0$) demonstrates that, in order to conserve mass, the mean wall-normal velocity must be zero everywhere.

The same simplifications can be made to the mean wall-normal momentum equation to give [43]:

$$0 = -\frac{\partial \langle p \rangle}{\partial z} - \frac{\partial}{\partial z} \left(\rho \langle w'w' \rangle \right) \ . \tag{5.8}$$

This equation states that the wall-normal mean pressure gradient must be equal to the gradient of the wall-normal Reynolds stress in order for the wall-normal momentum equation to be balanced. If this equation is integrated with respect to z and the no-slip boundary condition is applied, the following expression is obtained:

$$\rho \langle w'w' \rangle + \langle p \rangle = \langle p_w \rangle \quad . \tag{5.9}$$

Taking the derivative of this equation with respect to x gives the following result:

$$\frac{\partial \langle p \rangle}{\partial x} = \frac{\langle p_w \rangle}{\partial x} \quad . \tag{5.10}$$

This demonstrates that the streamwise pressure gradient does not vary in the wall-normal direction. If the simulation is driven by a body force (as an alternative to the pressure-gradient) it can be applied uniformly over the simulation domain.

Next, the streamwise momentum equation can be simplified for fully-developed turbulent channel flow [43]:

$$0 = -\frac{\partial \langle p \rangle}{\partial x} + \frac{\partial \langle \tau_{xz} \rangle}{\partial z} + \frac{\partial}{\partial z} \left(\rho \langle u'w' \rangle \right) \quad . \tag{5.11}$$

This equation can be rewritten as follows:

$$\frac{\partial \langle p \rangle}{\partial x} = \frac{\partial}{\partial z} \left(\langle \tau_{xz} \rangle - \rho \langle u'w' \rangle \right) \quad . \tag{5.12}$$

This expression shows that the pressure gradient in the streamwise direction is equal to the derivative of the total shear stress, which includes both the viscous and Reynolds stresses. If this result is combined with equation 5.10, then it can also be proved that the total shear stress is linear in the wall-normal direction. It is also known that the mean total shear stress must be zero at the centreline (due to symmetry). Thus [43]:

$$\tau + \rho \langle u'w' \rangle = -\delta \left(\frac{\partial \langle p \rangle}{\partial x} \right) \quad . \tag{5.13}$$

Since the Reynolds shear stresses on the walls must be zero due to the no-slip condition, only the viscous stress contributes to the wall shear stress. This means that:

$$\tau_w = \mu \left. \frac{\partial \langle u \rangle}{\partial z} \right|_{z=0} \quad , \tag{5.14}$$

where τ_w is the viscous shear stress at the wall. Thus, if the average wall shear stress is known for the target Reynolds number, the mean pressure gradient or body force required to drive the flow can be calculated. This principle will be used in sections 5.2.1 and **??** to derive the parameters for the LBM and FD simulations.

5.1.2 Length and Velocity Scales

One of the difficulties in working with wall-bounded turbulent flows is the two distinct regions in which different length and velocity scales are used to scale the flow. In the region very close to the wall the dominant mode of momentum transfer is through molecular transport, or viscosity. However, farther from the wall inertial transfer is much more important.

Near the wall, the important parameters are ρ, ν, and τ_w. Using these quantities, velocity and length scales can be defined [43]:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \quad , \tag{5.15}$$

and

$$\delta_\nu = \nu\sqrt{\frac{\rho}{\tau_w}} = \frac{\nu}{u_\tau} \quad . \tag{5.16}$$

These are the viscous scales of the flow. The velocity scale is called the friction velocity and the length scale is called the viscous length scale. The friction velocity can be used to define the friction Reynolds number [43]:

$$Re_\tau = \frac{u_\tau \delta}{\nu} \quad . \tag{5.17}$$

The outer region, where inertial forces are more dominant, can be scaled with the mean centreline velocity (U_0) and the channel half-width (δ) [43].

5.1.3 Mean Velocity Profile

With the definition of the velocity and length scales, a dimensional analysis can be performed to determine the form of the mean velocity profile in the inner and outer regions. The important parameters for turbulent channel flow are the density, kinematic viscosity, channel half-width, and the friction velocity. Using these variables, the mean velocity gradient can be written as a function of a universal non-dimensional function (Φ) that depends on two dimensionless groups:

$$\frac{\partial \langle u \rangle}{\partial z} = \frac{u_\tau}{z} \Phi \left(\frac{z}{\delta_\nu}, \frac{z}{\delta} \right) \quad . \tag{5.18}$$

In the region very close to the wall ($z^+ < 5$), called the viscous sublayer, the outer scales are not important, so equation 5.19 can be written in terms of the inner scales only:

$$\frac{\partial \langle u \rangle}{\partial z} = \frac{u_\tau}{z} \Phi_i \left(\frac{z}{\delta_\nu} \right) \quad . \tag{5.19}$$

Manipulating equation 5.19 for the viscous sublayer gives the following velocity profile [43]:

$$\langle u \rangle^+ \approx z^+ + O(z^{+2}) \ , \tag{5.20}$$

for $z/\delta \ll 1$. Ignoring the second-order terms, this equation shows that the velocity profile, when scaled by the viscous scales, is linear very close to the wall. Moving farther from the wall ($z^+ > 30$ to $z/\delta < 0.3$), the outer scales are still unimportant but the flow is no longer dependent on the viscosity. Thus, Φ in equation 5.18 is neither a function of z/δ_ν or z/δ and becomes a constant (κ^{-1}). Thus, in this region, the velocity profile is [43]:

$$\langle u \rangle^+ = \frac{1}{\kappa} \ln z^+ + A \ , \tag{5.21}$$

for $z/\delta \ll 1$ and $z^+ \gg 1$. This region is called the log layer. The constants are determined empirically to be $\kappa = 0.4$ and $A = 5.5$ [32].

A similar dimensional analysis can be performed to determine the form of the outer layer ($z^+ > 50$). In this region, the outer velocity and length scales are used and viscosity is not important. Thus, Φ_o is only a function of the outer variables:

$$\frac{\partial \langle u \rangle}{\partial z} = \frac{u_\tau}{z}\Phi_o\left(\frac{z}{\delta}\right) \quad . \tag{5.22}$$

This equation is valid only in the outer layer; however, for flows with sufficiently high Reynolds numbers, there is an overlap region ($z^+ > 50$ to $z/\delta < 0.1$) where both this equation and equation 5.19 are valid. In this region, Φ_o must also be constant, so equation 5.22 can be integrated to give [43]:

$$U_0^+ - \left\langle u^+ \right\rangle = -\frac{1}{\kappa}\ln\left(\frac{z}{\delta}\right) + B \quad . \tag{5.23}$$

Since this equation is integrated from the centreline, it depends on the centreline velocity. This equation is called the velocity defect law and, in the form above, is only valid in the overlap region. However, the velocity defect law can be used in the outer layer if a wake function is included. The wake function simply accounts for the deviation of the actual velocity profile from the log-law.

The main goal of calculating the velocity profile is to find a way to link the mean centreline velocity with the wall shear stress. Equation 5.23 contains U_0^+, but requires an expression for $\langle u \rangle^+$. In order to remove the dependence on $\langle u \rangle^+$, equations 5.21

and 5.23 can be summed [43]:

$$U_0^+ = \frac{1}{\kappa}\ln\left(z^+\right) - \frac{1}{\kappa}\ln\left(\frac{z}{\delta}\right) + A + B \ . \tag{5.24}$$

Assuming that the deviation from the log-law in the outer layer is small, this expression can be evaluated at the centreline of the channel. However, to do this, the constant B must be assumed to be zero in order for the velocity defect to be equal to zero at the centreline. The resulting expression is called the friction law [43]:

$$U_0^+ = \frac{1}{\kappa}\ln\left(Re_\tau\right) + A \ . \tag{5.25}$$

Thus, if the desired Reynolds number and mean centreline velocity are known, this equation can be solved iteratively for the corresponding friction velocity. The friction velocity can then be used to find the wall shear stress from equation 5.15, and the pressure gradient (or body force) can be calculated from equation 5.13.

5.2 Simulation Parameters

In this section, the parameters chosen for the LBM and FD simulations of channel flow will be justified and discussed. Both simulations were based on the $Re_\tau = 180$ spectral simulation of Moser *et al.* [41]. This Reynolds number was chosen because it is nearly the lowest possible Reynolds number at which turbulence can be sustained indefinitely. The computational cost of directly simulating a turbulent flow increases with $Re^{9/4}$, so choosing the lowest possible Reynolds number minimizes the computational cost of the simulation.

The computational domain for turbulent channel flow is sketched in figure 5.2. No-slip boundary conditions are applied at the boundaries in the wall-normal (z) direction. These boundaries represent the surface of the two plates in figure 5.1 and thus the distance between them (L_z) is set to 2δ. Cyclic, or periodic, boundaries are implemented in both the streamwise (x) and spanwise (y) directions. This is because the turbulence is homogeneous in these directions. Cyclic boundary conditions use the state of the flow at the outlet as an inlet condition. This has the effect of mimicking an infinitely large domain.

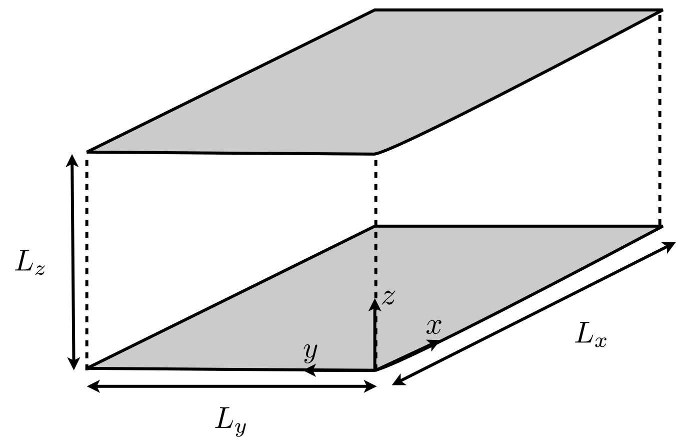


Figure 5.2: Fully-developed turbulent channel flow simulation domain.

The x- and y-dimensions of the computational domain must be chosen such that the domain is large enough to contain the biggest turbulent structures in the flow field. Moser *et al.* confirmed that their domain was large enough by ensuring that two-point correlations for the velocity and pressure fluctuations decayed to zero within the simulation domain.

As mentioned in section 1.1, there are several LBM simulations of channel flow in the literature (for example [19], [34], and [44]) that use computational domains with smaller dimensions in order to save computational time. However, reducing the size of the computational domain effectively removes the large scale structures and therefore alters the turbulence statistics. Thus, two simulations whose statistics are being compared for the purpose of validation must have the same computational domain. Otherwise, the effect of the domain size will confound any error inherent to the algorithm being validated. For this reason, the dimensions of the computational domains for the LBM and FD simulations were chosen to match the dimensions used by Moser *et al.* [41]. This allows for a proper comparison of the turbulence statistics calculated from each simulation.

The other parameters necessary to reproduce the results from the LBM and FD simulations are discussed in the sections that follow.

5.2.1 LBM Simulation Parameters

Once the Reynolds number and domain size have been chosen, the next parameter that needs to be determined is the grid resolution. It should be noted that, as mentioned in section 4.2, the LBM must be implemented on a uniform cubic lattice so that the particle distributions travel from one lattice site to the next in one time step. This means that the grid resolution must be equal in each coordinate direction. For channel flow, the grid resolution requirements are much more strict in the wall-normal direction than in the streamwise and spanwise directions. Thus, the grid resolution for the entire domain is based on the resolution requirements in the wall-normal direction. This means that the flow will be over-resolved in the streamwise and spanwise directions.

In order for a simulation to be considered a DNS, the grid resolution must be fine enough to resolve the Kolmogorov scales, which are the smallest structures in the flow. The actual size of the Kolmogorov scales can be computed from the dissipation rate (ϵ) [43]:

$$\eta \equiv \left(\frac{\nu^3}{\epsilon}\right)^{1/4} . \tag{5.26}$$

Using this relationship, the Kolmogorov length scale at the wall (where the dissipation rate is the highest) was calculated from the spectral results of Moser *et al.* [41] to be $\eta^+ \approx 2$.

In order to capture the energy content of the smallest scales and thus accurately predict the dissipation rate, the grid spacing must be on the same order as the Kolmogorov length scale. Thus, the grid resolution for the LBM simulation was chosen to be $\Delta x^+ = \Delta y^+ = \Delta z^+ = 2$. A grid resolution study was performed by Bespalko [6] on a channel simulation with a smaller computational domain in order to reduce the computational cost. It was concluded that a resolution of 2 wall units in the wall-normal direction was sufficient.

Since the channel half-width is equal to Re_τ when scaled by the viscous length scale, this grid resolution results in a grid with $1080 \times 360 \times 182$ nodes in the streamwise, spanwise, and wall-normal directions, respectively. Since the bounce-back boundary condition results in a no-slip boundary halfway between the boundary node and the first interior node, the first node away from the wall is located at $z^+ = 1$.

Though the LBM can only be used to simulate incompressible flows ($Ma < 0.3$) it is a compressible method and therefore the Mach number must be specified. Ideally, since the LBM simulation will be compared to an incompressible ($Ma \equiv 0$) simulation, the Mach number would be chosen to be vanishingly small. However, if the Mach number is very small ($Ma \ll 1$), the time scale of the flow will be much larger than the time scale of the sound waves in the simulation. Since the smallest time scale dictates the required temporal resolution, the time step would have to be made very small to ensure that the relatively fast-moving sound waves would not cause the simulation to become unstable. Thus, if the flow develops at a much slower time scale, the flow will be over-resolved temporally which makes the simulation very computationally expensive. For this reason, the Mach number of the LBM simulation was chosen to be 0.2, which is as close as possible to the maximum allowable value.

The Mach number of the simulation was based on the mean centreline velocity since it is the maximum mean velocity in the simulation. The speed of sound, according to equation 3.17, is $c_s \approx 0.577m/s$. Thus, the mean centreline velocity was chosen to be $U_0 \approx 0.115m/s$ to give a mean Mach number of 0.2.

Once the mean velocity has been chosen, the friction law (equation 5.25) can be used to calculate the friction velocity. The friction velocity obtained from this equation is $u_\tau \approx 6.25 \times 10^{-3}m/s$. The viscosity of the simulation can then be computed from the definition of the viscous length scale (equation 5.16). By definition, the viscous length scale has been set to be $\delta_\nu = 0.5m$ since a grid resolution of $\Delta x = 1m$

in lattice units is equivalent to a grid resolution of $\Delta x^+ = 2$ when normalised by the viscous length scale. Thus, the viscosity of the simulation is $\nu \approx 3.12 \times 10^{-3} m^2/s$. The relaxation time, which is related to the viscosity through equation 3.59, is set to $\hat{\omega} \approx 1.96$.

The mean wall shear stress can be computed from the definition of the friction velocity in equation 5.15. Assuming that the density does not vary significantly from its nominal value of 1, the wall shear stress is approximately $\tau_w \approx 3.90 \times 10^{-5} N/m^2$. Finally, the mean pressure gradient required to achieve this wall shear stress can be computed from equation 5.13, which was derived from the analysis of the mean streamwise momentum equation. This gives a pressure gradient approximately equal to $\partial p/\partial x = -4.34 \times 10^{-7} Pa/m$. The pressure gradient is actually implemented as a body force in the LBM ($-\partial p/\partial x = \rho \vec{g}$). This is done because implementing a true pressure gradient would require a mean density gradient.

The time step in the LBM is determined by the grid spacing since the particle distributions must travel between neighbouring lattice sites in a single time step. The lattice velocities are fixed, so once the grid spacing is chosen the time step is fixed as well. As mentioned earlier, the only control the user has over the time step is through the specification of the Mach number. The size of the time step, relative to a representative time scale in the flow, is linearly related to the Mach number. Thus, it can be made as large as possible by choosing a Mach number close to the maximum allowable value of 0.3. The time step in the LBM simulation is equal to $\Delta t^+ \approx 1.25 \times 10^{-2}$ when normalized by the viscous scales. To put this time step in perspective, a large-eddy turn-over-time (LETOT), which is the time required for a large eddy to breakdown and transfer its energy to smaller scales, is equal to $t_L^+ = 180$. Thus, approximately 1.44×10^4 time steps are required to complete one LETOT. The turbulence statistics from the LBM simulation were averaged over 60 LETOTs for a total of approximately 9×10^5 time steps after the simulation reached a statistically stationary state. The statistics were computed from snapshots of the flow field recorded approximately every 0.1 LETOTs.

The LBM simulation was run for approximately 6 weeks on the Sun M9000 servers at HPCVL using 128 threads. Each snapshot of the flow field required approximately $2.2GB$ of memory, so the total disk space used to store the results was $1.3TB$.

5.2.2 Finite Difference Simulation Parameters

The code used to run the finite difference simulation was provided by Ugo Piomelli. The details of its implementation can be found in [31] and examples of results obtained with the code can be found in [30] and [29].

The FD method solves the incompressible form of the NS equation directly. The spatial derivatives are approximated by second-order finite differences on a staggered grid. All of the terms in the NS equations, with the exception of the wall-normal convective and diffusive terms, were advanced explicitly in time using a third-order Runga-Kutta scheme. The wall-normal terms were advanced semi-implicitly with a Crank-Nicolson scheme. These terms were handled differently so that the fine grid spacing in the wall-normal direction would not limit the size of the time step that could be used. The pressure field was calculated using a fractional step method to ensure that the velocity field at each time step satisfied the continuity equation [31].

Unlike the LBM, the FD method does not require the grid resolution to be the same in each coordinate direction and it can also handle non-uniform grids. Thus, lower grid resolutions were used in the streamwise and spanwise directions to save computational time. Also, a non-uniform grid was used in the wall-normal direction to concentrate grid points near the wall where the resolution requirements are the most strict.

The computational grid chosen had $384 \times 384 \times 256$ nodes in the streamwise, spanwise, and wall-normal directions, respectively. This grid was selected by successively increasing the resolution until the first- and second-order turbulence statistics matched the results of Moser *et al.* [41]. The resolution was $\Delta x^+ = 5.6$ and $\Delta y^+ = 1.9$ in the streamwise and spanwise directions. A hyperbolic tangent function was used to create a non-uniform grid in the wall-normal direction to cluster grid nodes close to the wall. This resulted in a maximum resolution of $\Delta z^+ = 0.06$ close to the wall and a minimum resolution of $\Delta z^+ = 3.9$ near the centreline.

Instead of specifying the mean pressure gradient needed to drive the flow, the mass flux was fixed and the pressure gradient was controlled to produce the target mass flux. The advantage of this method is that the simulation reaches a statistically stationary state much faster than if the pressure gradient is fixed. Once the simulation is stationary, the pressure gradient is constant and there is no difference between a simulation run with a constant mass flux or a constant pressure gradient.

In order to run the simulation in constant mass flux mode, the governing equations were normalized by the Reynolds number based on the bulk velocity and the channel half-width. Since there is no known expression that links the bulk Reynolds number to Re_τ, the bulk Reynolds number was adjusted by trial and error until the target friction Reynolds number of 180 was reached. The friction Reynolds number was closest to the target for a bulk Reynolds number of $Re_B = 2800$.

The simulation was run with a constant CFL number of 0.8, which lead to an average time step of $\Delta t^{+} = 7.62 \times 10^{-2}$. This time step is roughly six times larger than the time step in the LBM simulation. The turbulence statistics were also averaged over approximately 1.4×10^{5} time steps, which corresponds to 60 LETOTs. As with the LBM simulation, the statistics were computed from snapshots of the flow field saved every 0.1 LETOTs.

The FD simulation was run for approximately 1 week using 32 threads on the Sun M9000 servers at HPCVL. The disk space used for each snapshot was $1.2GB$ for a total of $720GB$. Thus, for this problem, the LBM used approximately 24 times more computational resources than the FD simulation. The LBM had a much higher computational cost primarily due to the fact that its time step was roughly 6 times smaller than the time step of the FD simulation. The LBM also needed approximately twice as many computational nodes because it is not capable of stretching the grid in the streamwise direction.

Chapter 6

Results and Discussion

In the last chapter, the computational methodologies for the LBM and FD simulations of turbulent channel flow were outlined. In this chapter, the results from these simulations are presented and discussed. In the first section, the turbulence statistics, momentum balances, and one-dimensional energy spectra are compared for the two simulations. In the second section the results are discussed and explanations for the discrepancies between the two simulations are given.

6.1 Results

6.1.1 Turbulence Statistics

As mentioned in section 5.1.1, a turbulent flow can be broken down into its mean and fluctuating components with the Reynolds decomposition:

$$\phi = \langle\phi\rangle + \phi' \ , \tag{6.1}$$

where ϕ is a property of the flow such as velocity or pressure. The term in brackets ($\langle\phi\rangle$) is the mean component and is defined as an ensemble average over multiple instances of the flow. The primed term (ϕ') is the fluctuating component of the flow field and can be calculated from equation 6.1 once the mean has been calculated.

In order to calculate an ensemble average, an experiment or simulation must be run many times (with slightly different, but statistically equivalent initializations) and averaged together. However, if a flow reaches a point where the turbulence statistics are not a function of time (statistically stationary), time averaging is equivalent to ensemble averaging. Similarly, if the turbulence statistics are homogeneous in a co-ordinate direction, spatial averaging in this direction is also equivalent to ensemble averaging.

Fully-developed turbulent channel flow converges to a statistically stationary state, and is also homogeneous in the streamwise and spanwise directions. Thus, the mean flow field is only a function of the wall-normal coordinate, and can be calculated by averaging the flow field in time and in the streamwise and spanwise directions.

The mean properties in this work were calculated from snapshots of the flow field that were saved from each simulation every 0.1 LETOTs. These snapshots were averaged in space and time according to the following expression:

$$\langle\phi\rangle(z) = \frac{1}{N_t}\sum_t \frac{1}{N_x}\sum_x \frac{1}{N_y}\sum_y \phi(x,y,z,t) \ . \tag{6.2}$$

where N_x and N_y are the number of nodes in the streamwise and spanwise directions, respectively and N_t is the number of snapshots. The statistics for both simulations were averaged over 600 snapshots, which is equivalent to a time duration of approximately 60 LETOTs.

The mean streamwise velocity profiles (normalized by the friction velocity) for the LBM and FD simulations are plotted in figure 6.1. The mean spanwise and wall-normal velocity profiles are zero everywhere (as derived in section 5.1.1) and thus are not plotted. The profile from the spectral simulation of Moser *et al.* [41] is also plotted to show its close agreement with the FD results. In general, the LBM and FD results agree qualitatively, though the LBM under-predicts the mean velocity outside the viscous sublayer.

The viscous sublayer is clearly visible for $z^+ < 5$. In this region the velocity increases linearly away from the wall, but it appears curved because it is plotted on a logarithmic scale. There is a very small log-law region between $z^+ \approx 30$ and $z^+ \approx 50$. The velocity profile increases logarithmically in this region and thus appears as a straight line in this plot. Finally, the outer layer is visible as a deviation from the log-law for $z^+ > 50$.

Figure 6.2 shows the difference between the LBM and FD mean velocity profiles. The difference (e) is expressed as a percentage normalized by the maximum value of the mean streamwise velocity from the FD simulation. The difference starts small near the wall since both simulations enforce a zero velocity there. The difference between the two simulations increases through the inner layer and reaches approximately 2% in the outer layer. The error improves slightly at approximately $z^+ = 50$, which is the border of the inner and outer regions of the flow.

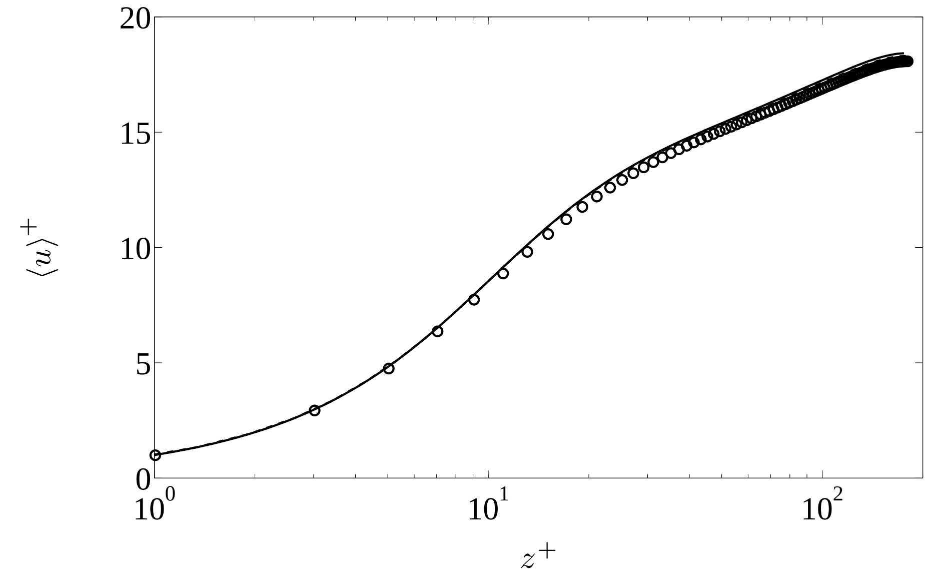


Figure 6.1: Mean streamwise velocity profile for the LBM and FD simulations: ○ LBM; —— FD; - - - Moser *et al.* [41]

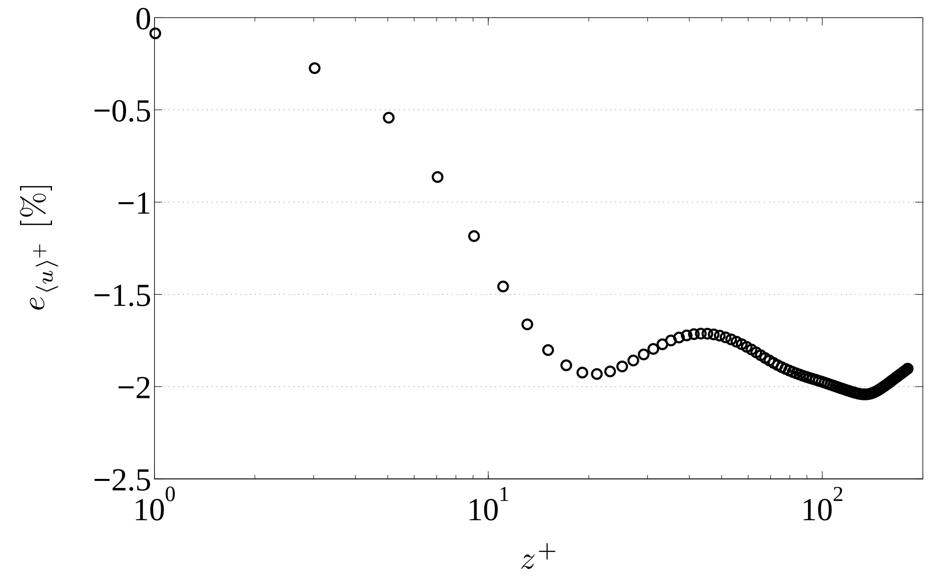


Figure 6.2: Percent difference between the mean streamwise velocity for the LBM and FD simulations.

The mean pressure profiles for the LBM and FD simulations are plotted in figure

6.3. Once again, the profile from Moser *et al.* is plotted to show its excellent agreement with the FD results. The mean pressure profiles shown are zeroed at the wall and normalized by the wall shear stress, τ_w.

There is no mean wall-normal pressure gradient in laminar channel flow. However, in a turbulent flow a mean pressure gradient arises in order to offset the wall-normal Reynolds stress. Otherwise, there would be an imbalance in the wall-normal momentum equation.

The pressure in the LBM is calculated from the ideal gas law, as described in section 4.2. Since the temperature is fixed, the pressure can be directly calculated from the density ($p = \rho/3$). In the FD method, the density does not vary since the incompressible form of the equations is used. Instead, the pressure is calculated through the solution of a Poisson equation that ensures that the velocity field satisfies the continuity equation [31].

Despite the very different approaches to calculating the pressure, the results from the LBM and FD simulations are in good qualitative agreement. However, the LBM results deviate from the curve predicted by the FD simulation near the location of the minimum pressure at $z^+ \approx 60$.

The percent difference between the mean pressure calculated from the LBM and FD simulations is plotted in figure 6.4. The LBM results agree very well up to approximately $z^+ = 10$, then the error increases sharply to almost 4% at $z^+ \approx 60$. The error improves steadily in the outer layer where the LBM over-predicts the pressure at the centre of the channel by approximately 1%.

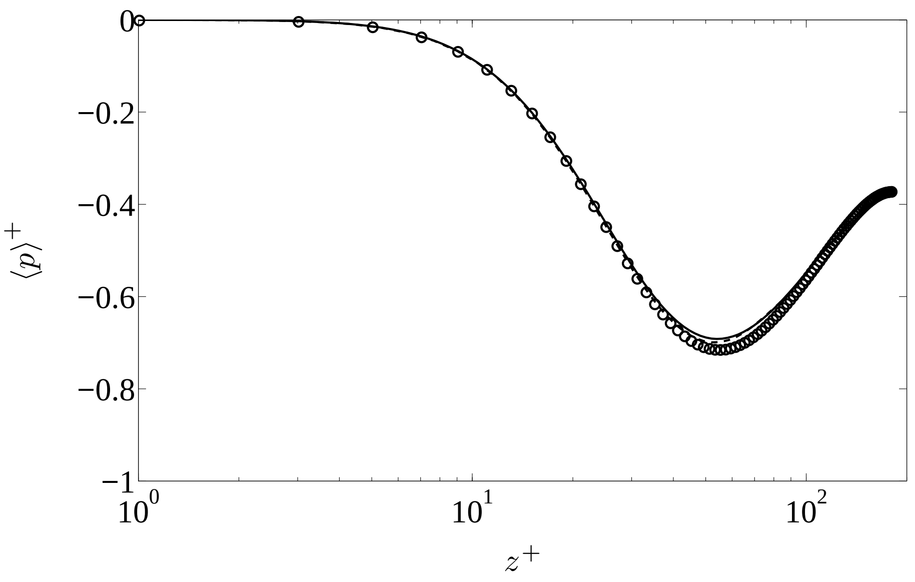


Figure 6.3: Mean pressure profile for the LBM and FD simulations: ○ LBM; —FD; --- Moser *et al.* [41]

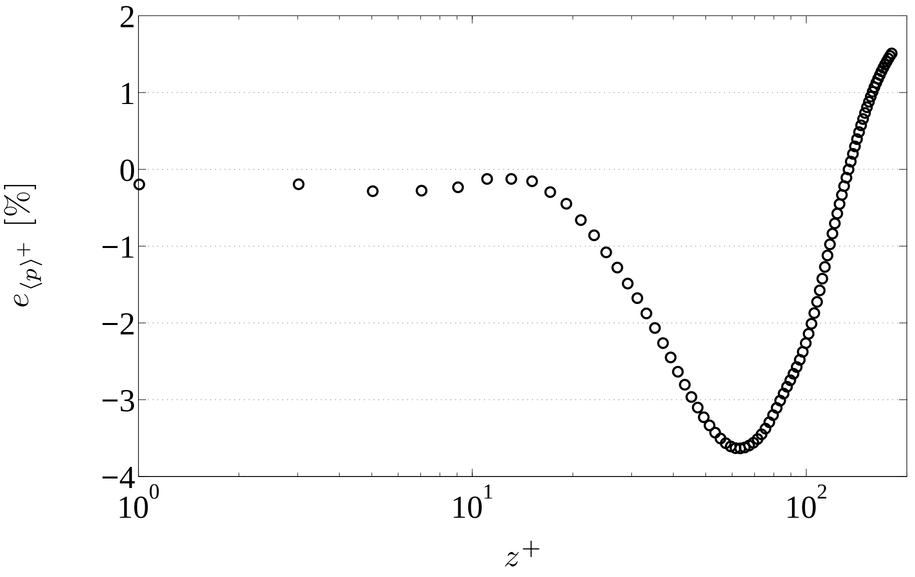


Figure 6.4: Percent difference between the mean pressure for the LBM and FD simulations.

The viscous shear stress profiles for both simulations are plotted in figure 6.5. Since there is no mean velocity in the spanwise and wall-normal directions, τ_{xz} is the only nonzero shear stress. The shear stress in this plot has been normalized by τ_w.

The mean viscous shear is at its maximum value at $z = 0$. Further away from the wall the effect of viscosity decreases and the Reynolds stress takes over as the mechanism for momentum transfer in the channel. Since the centre of the channel is a symmetry plane, the mean velocity gradient there is zero, and both the viscous and Reynolds stresses vanish.

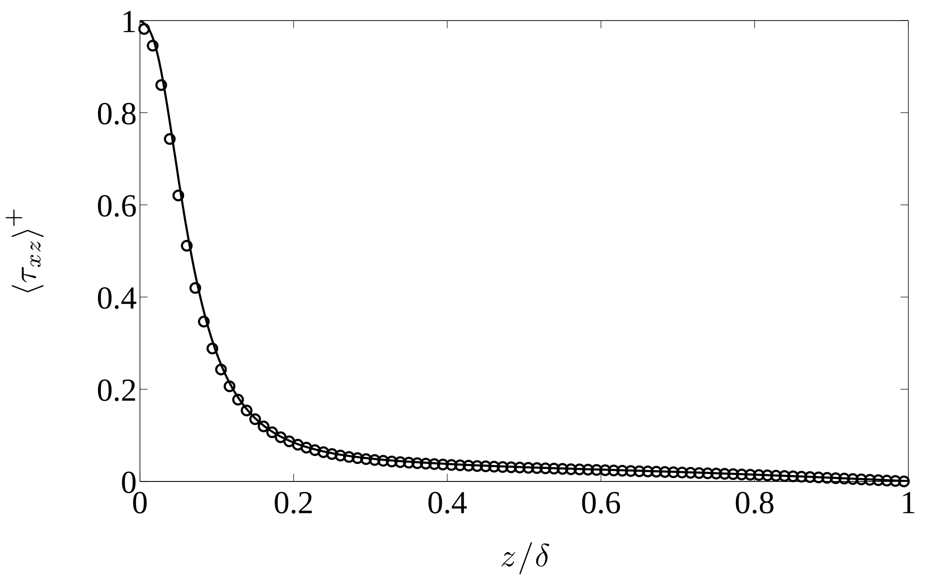


Figure 6.5: Mean viscous shear stress profile for the LBM and FD simulations: ○ LBM; —FD.

As figure 6.6 shows, the LBM under-predicts the viscous shear stress by as much as 4% in the inner layer. The match improves significantly further away from the wall, but this is expected because the viscous stress must decay to zero at the centreline by definition.

In general, the mean profiles from the LBM show good qualitative agreement with those computed by the FD method. The maximum difference between the mean velocities predicted by the two methods is approximately 2%, while the difference in

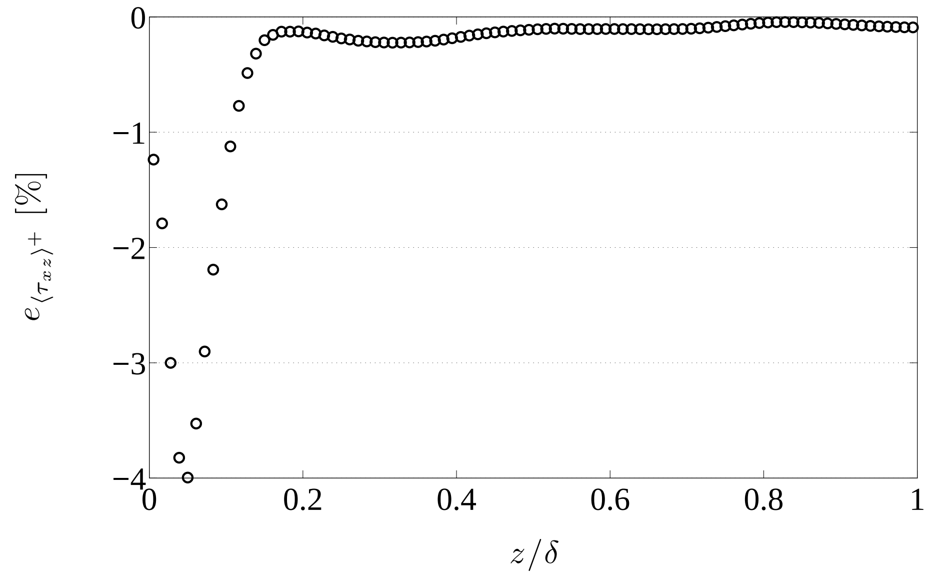


Figure 6.6: Percent difference between the mean viscous shear stress for the LBM and FD simulations.

the mean pressure is as high as 4%. It is clear from the results that, while the difference in the LBM results are not overly large, the spectral results of Moser *et al.* [41] match better with the FD statistics. This is concerning since the results from a DNS of a turbulent flow should be independent of the numerical method used to calculate them. This is because the results from a well-resolved DNS should approach the exact solution provided that the numerical errors have been carefully controlled. An explanation of the difference between the LBM and FD results is given in section 6.2.

The variance profiles for the velocity and pressure fields are examined next. The variance is defined as the square of the standard deviation, which is the average amplitude of the fluctuating component of the flow field. The variances of the components of the fluctuating velocity are of particular importance because they are related to the normal Reynolds stresses. These stresses, as discussed in section 5.1.1, are the primary mechanism in which the turbulent fluctuations interact with the mean flow field.

The variance was calculated from the snapshots saved from the LBM and FD simulations using the following expression:

$$\left\langle \phi' \phi' \right\rangle (z) = \frac{1}{N_t} \sum_t \frac{1}{N_x} \sum_x \frac{1}{N_y} \sum_y \left[\phi(x, y, z, t) - \left\langle \phi \right\rangle (z) \right]^2 \quad . \tag{6.3}$$

The variance profiles for the streamwise velocity (normalized by the square of the friction velocity) are plotted for the LBM and FD simulations in figure 6.7. The variance profile from Moser *et al.* [41] is also plotted to show how closely it matches the curve predicted by the FD method.

Due to the no-slip condition at $z = 0$, the variance is zero at the wall and it increases with z^2 in the near-wall region [43]. The peak variance occurs at $z^+ \approx 10$, which is in the buffer region between the viscous sublayer and the log-law region. In this area the total shear rate is relatively high, and the effect of viscosity is small enough that the turbulent fluctuations are not immediately damped out. The amplitude of the streamwise velocity fluctuations is much smaller in the outer layer ($z^+ > 50$) and continues to decrease towards the centre of the channel where the mean shear approaches zero.

Figure 6.2 shows the percent difference between the LBM and FD results for the streamwise velocity variance. The LBM and FD results agree well next to the wall due to the implementation of the no-slip constraint. However, the LBM predicts a sharper increase in the velocity variance than the FD simulation in the viscous sublayer. The largest difference between the two simulations occurs just past the peak

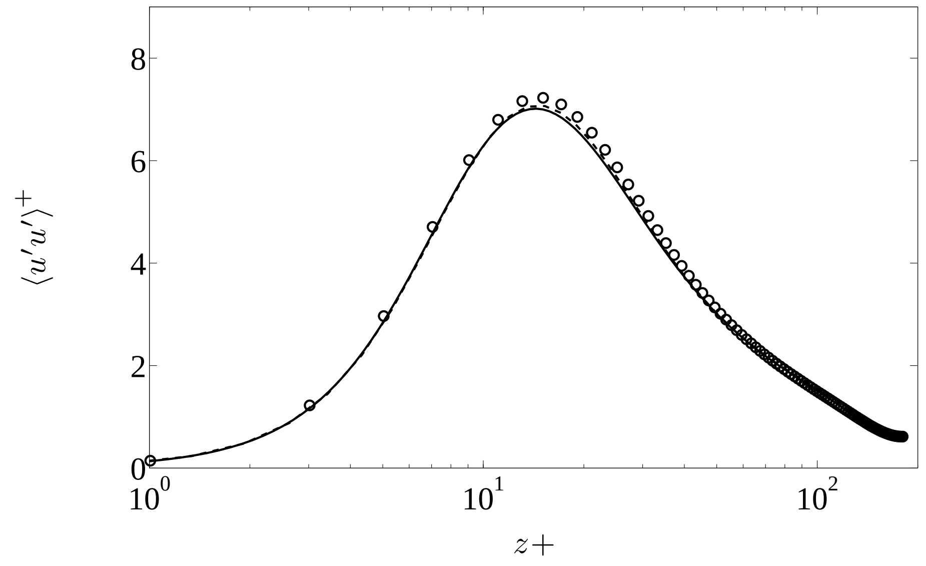


Figure 6.7: Streamwise velocity variance profile for the LBM and FD simulations: ○ LBM; —— FD; - - - Moser *et al.* [41]

variance at $z^+ \approx 20$ where the LBM is 4% higher than the FD simulation. The error decreases significantly in the outer layer and is negligible near the centreline.

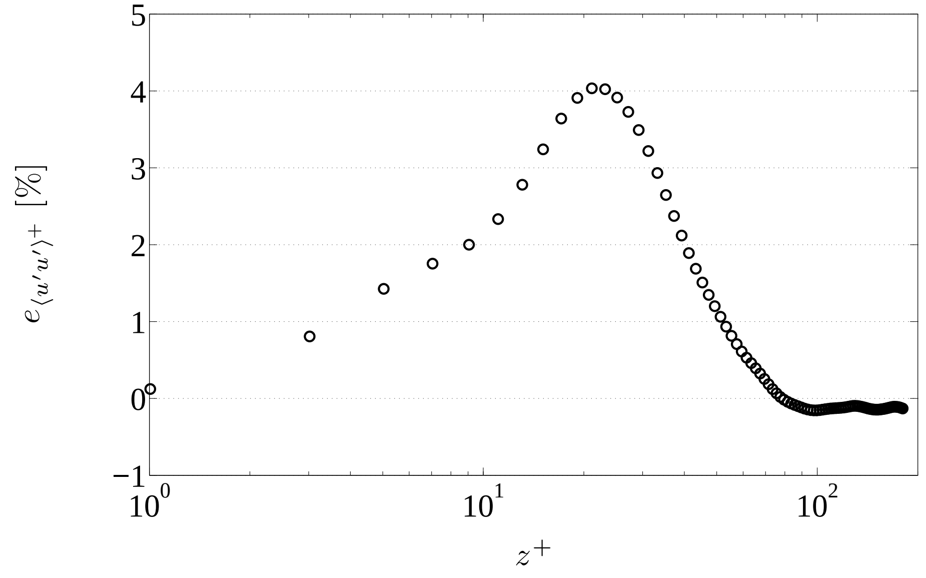


Figure 6.8: Percent difference between the streamwise velocity variance for the LBM and FD simulations.

The variance profile for the spanwise velocity fluctuations is plotted in figure 6.9. As with the streamwise velocity fluctuations, the variance is zero at the wall due to the no-slip condition. The spanwise velocity variance also increases with z^2 in the near-wall region [43]. However, its peak value occurs slightly further away from the wall at approximately $z^+ \approx 50$, which is on the border between the inner and outer layers.

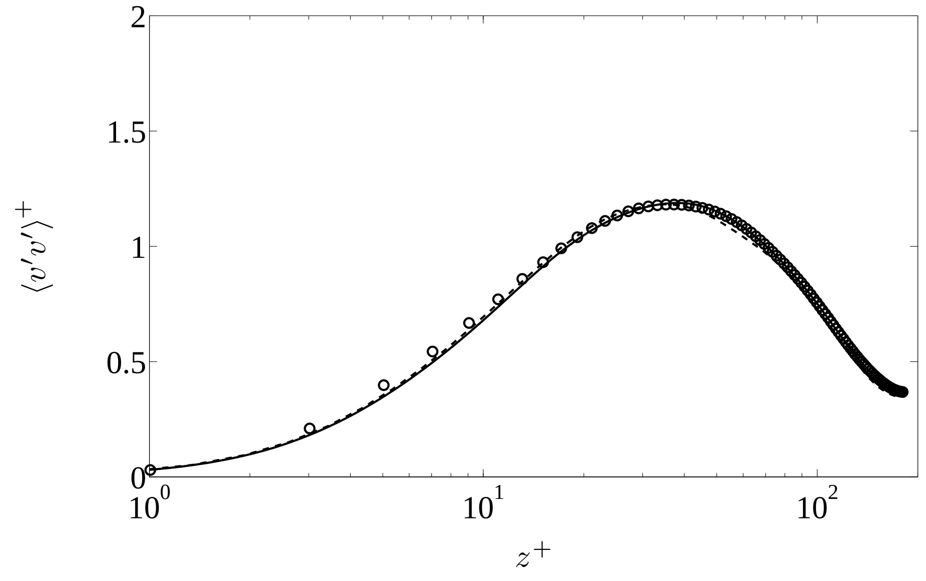


Figure 6.9: Spanwise velocity variance profile for the LBM and FD simulations: ○ LBM; —— FD; - - - Moser *et al.* [41]

The percent difference in the spanwise variance predicted by the LBM and the FD method is plotted in figure 6.10. The LBM overpredicts the rate at which the spanwise variance increases away from the wall, and the peak error is approximately 4% at $z^+ \approx 5$, which corresponds to the edge of the viscous sublayer. The difference between the two simulations decreases to only 1% in the outer region.

The profile of the variance of the wall-normal velocity fluctuations is plotted in figure 6.11. Unlike the streamwise and spanwise variance profiles, the wall-normal

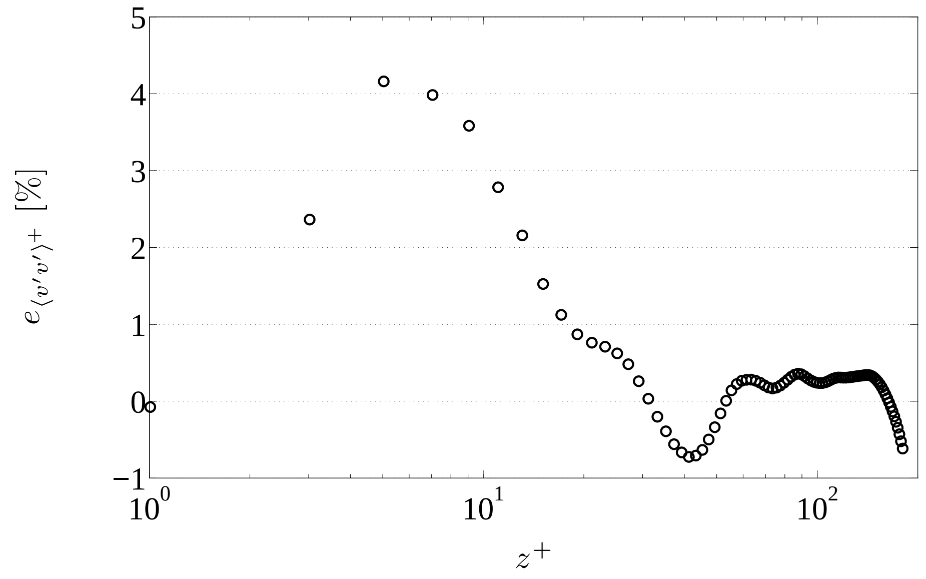


Figure 6.10: Percent difference between the spanwise velocity variance for the LBM and FD simulations.

variance increases with z^4 in the near-wall region because $\partial w/\partial z$ must be equal to zero at the wall in order to satisfy continuity [43]. Thus, the wall-normal velocity variance grows slower away from the wall than the streamwise and spanwise components. This means that the turbulent motion in the near-wall region is predominantly organized in planes parallel to the wall. The peak wall-normal variance occurs at $z^+ \approx 50$ and decreases throughout the outer layer.

The error profile for the wall-normal velocity variance (plotted in figure 6.12) is very similar to the error profiles for the other two components. The LBM again over-predicts the rate of increase of the variance in the near-wall region, with the maximum error of 3% occurring at $z^+ \approx 60$. The error decreases slightly in the outer layer, though the LBM under-predicts the variance at the centreline by nearly 2%.

The variance profiles for the pressure fluctuations from the LBM and FD simulations are plotted in figure 6.13. The results from the LBM simulation performed

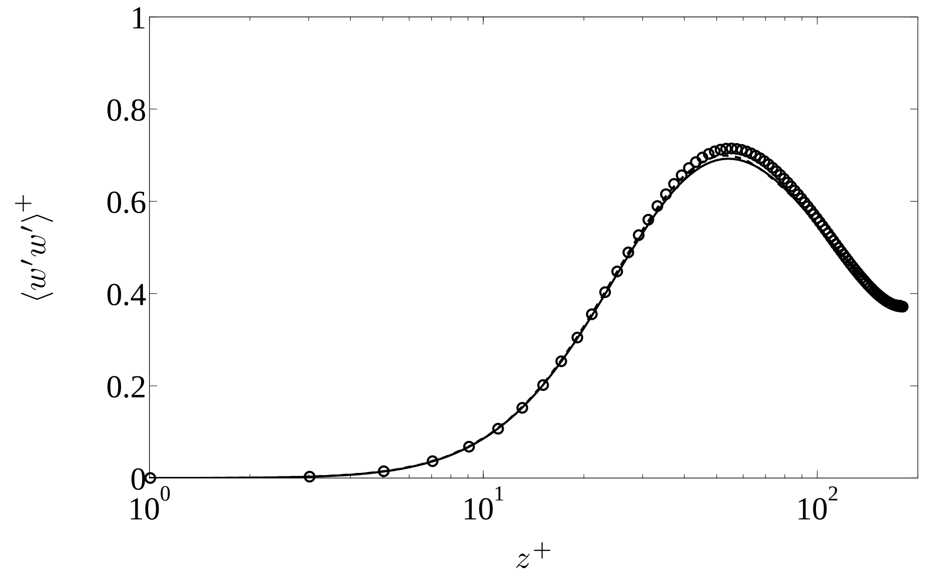


Figure 6.11: Wall-normal velocity variance profile for the LBM and FD simulations: ○ LBM; — FD; --- Moser *et al.* [41]

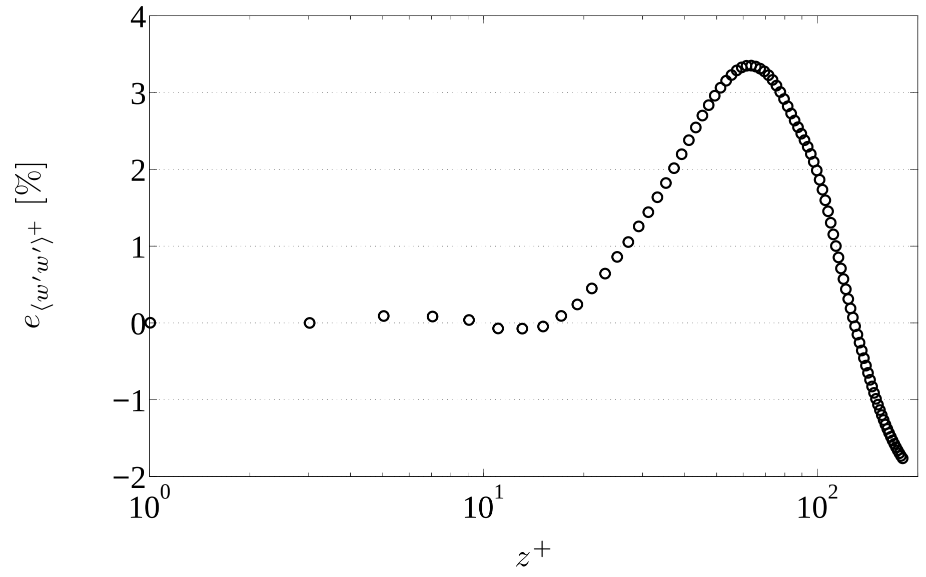


Figure 6.12: Percent difference between the wall-normal velocity variance for the LBM and FD simulations.

by Lammers *et al.* [34] are also plotted to show the effect the size of the computational domain has on the turbulence statistics. As mentioned in chapter 1 and section 5.2, the simulation of Lammers *et al.* had a much narrower computational domain

($L_y = \delta$) than the domains used in this work. As figure 6.13 shows, increasing the width of the computational domain (as done in the current LBM simulation) significantly improves the match with the FD simulation. This demonstrates the sensitivity of the turbulence statistics to the domain size.

Despite the marked improvement caused by widening the computational domain, the LBM still over-predicts the amplitude of the pressure variance in the near-wall region. The results from the simulation of Moser *et al.* [41] agree much better with the FD results than the results from the LBM simulation.

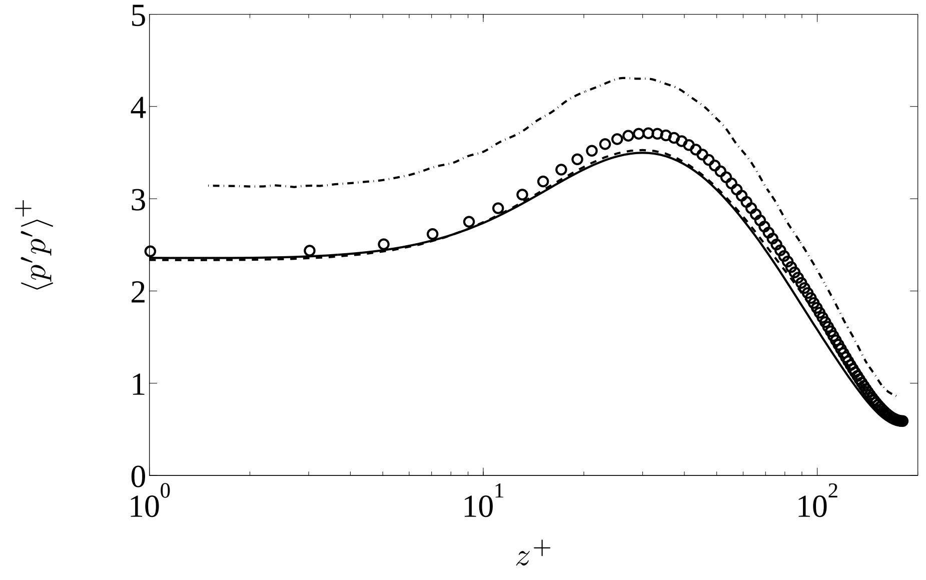


Figure 6.13: Pressure variance profile for the LBM and FD simulations: ○ LBM; —— FD; - - - Moser *et al.* [41]; -·-·- Lammers *et al.* [34].

The difference in the pressure variance for the two simulations is plotted in figure 6.4. The two simulations are within 2% of each other near the wall, but the error grows to approximately 7% near the location of the peak pressure variance. The error drops steadily through the outer layer and is under 2% at the centreline of the channel.

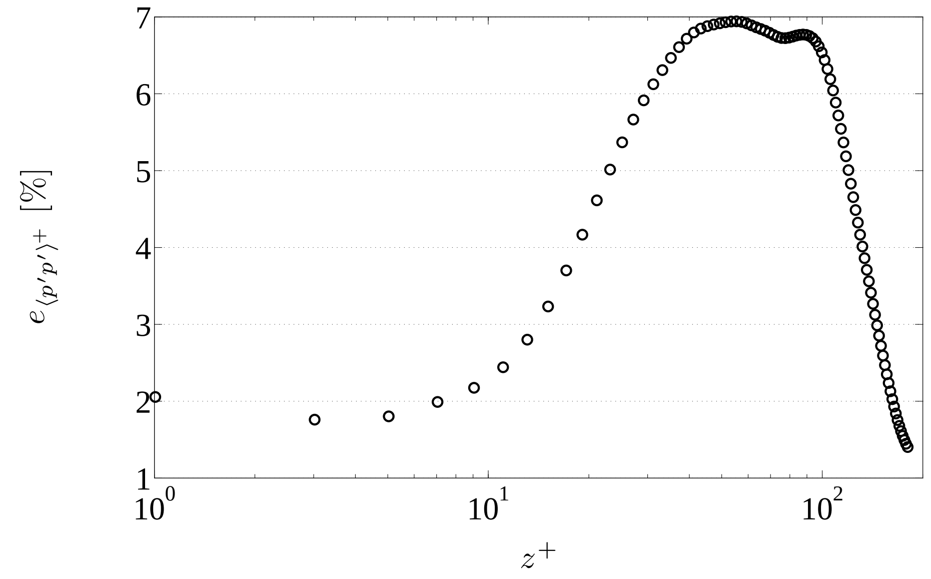


Figure 6.14: Percent difference between the pressure variance for the LBM and FD simulations.

Figure 6.15 shows the covariance of the streamwise and wall-normal velocities for both simulations. The covariance of these velocities is directly related to the Reynolds shear stress, which, along with the viscous shear, transports momentum in the channel to offset the action of the mean pressure gradient. Since there are no mean velocities in the spanwise or wall-normal directions, this is the only nonzero Reynolds shear stress.

The covariance is zero at the wall due to the no-slip condition at $z = 0$. The covariance then increases with z^3 away from the wall and reaches its peak value at approximately $z/\delta \approx 0.18$. The covariance decays to zero at the centre of the channel since it is a symmetry plane and thus there is no wall-normal velocity gradient.

The error profile for the covariance of the streamwise and wall-normal velocities is plotted in figure 6.16. Again, because of the no-slip condition, the error starts at zero near the wall. The error then increases to approximately 4% and then drops to approximately 1% beyond $z/\delta \approx 0.2$. The error profile is very similar to the profile for

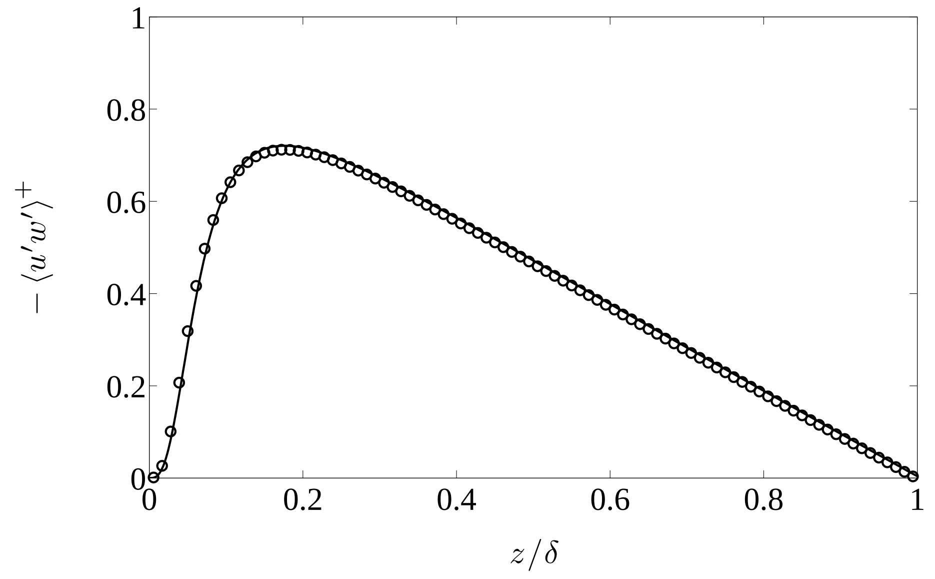


Figure 6.15: Streamwise-wall-normal velocity covariance profile for the LBM and FD simulations: ○ LBM; —— FD.

the viscous shear stress (figure 6.6) which is not surprising since these two quantities are closely related since they combine to offset the mean pressure gradient.

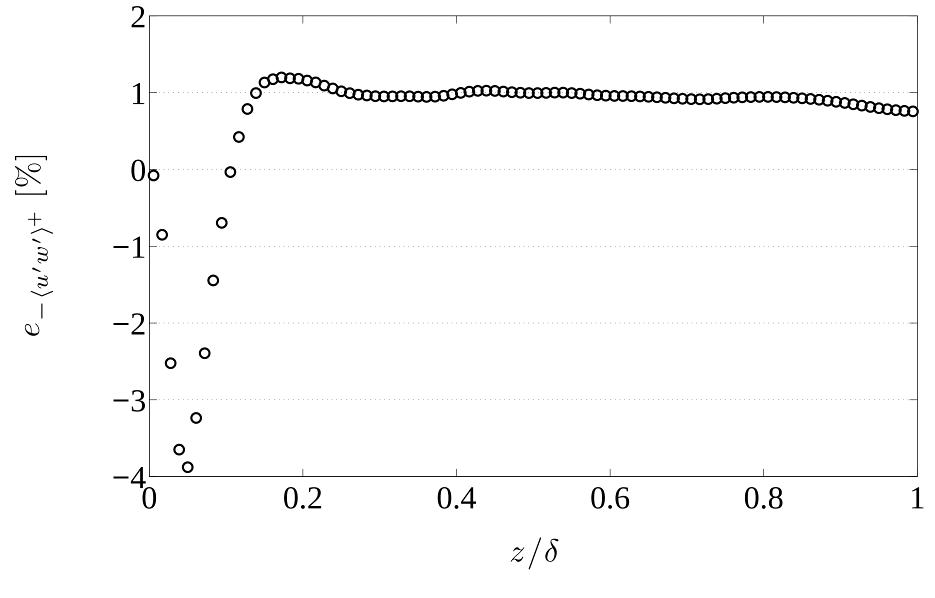


Figure 6.16: Percent difference between the streamwise-wall-normal velocity covariance for the LBM and FD simulations.

In general, the difference in the variance profiles for the LBM and the FD simulaitons is larger than for the mean profiles. The error in the variance of the velocity fluctuations is typically around 4%, while the error in the pressure variance was found to be as high as 7%. These discrepancies, particularly in the variance of the pressure fluctuations, are larger than what would normally be expected when comparing two direct numerical simulations and therefore must be explained. Causes for the over-prediction of the pressure variance are examined in section 6.2.

6.1.2 Momentum Balances

In section 3.1.5, a multiscale expansion was applied to the LBM to examine its behaviour in the macroscopic limit. This analysis showed that the LBM recovers the compressible form of the NS equations, though without the velocity divergence term and with a number of additional higher-order terms in the stress tensor.

In this section, the mean momentum balances are presented to compare the individual terms in the momentum equation computed from the LBM and FD simulations. The FD simulation works directly with the NS equations, so the individual terms were easily recovered. The LBM simulation, however, balances the momentum on the level of the particle distributions. Thus, the terms in the momentum equation had to be calculated from the particle distributions of the LBM from various velocity moments. It should be noted that Favre averaging was used for the LBM results in order to account for the variation in the density. Favre averaging, like Reynolds averaging, breaks the flow field into a mean and fluctuating component:

$$\vec{\phi} = \{\vec{\phi}\} + \vec{\phi}'' \ , \tag{6.4}$$

where $\{\vec{\phi}\} = \left\langle \rho\vec{\phi} \right\rangle / \langle \rho \rangle$. Favre averages are equivalent to Reynolds averages when the density fluctuations are very small ($Ma \ll 1$).

Figure 6.17 plots the individual terms in the streamwise mean momentum balance equation (equation 5.11). The markers are the results from the LBM simulation and the solid lines represent the FD results. All of the terms are normalized by the viscous scales. The only nonzero terms are the viscous diffusion, turbulent diffusion, and the body force. The body force terms (actually a mean pressure gradient in the FD simulation) for the two simulations are identical since they were essentially input parameters that were specified to be equal.

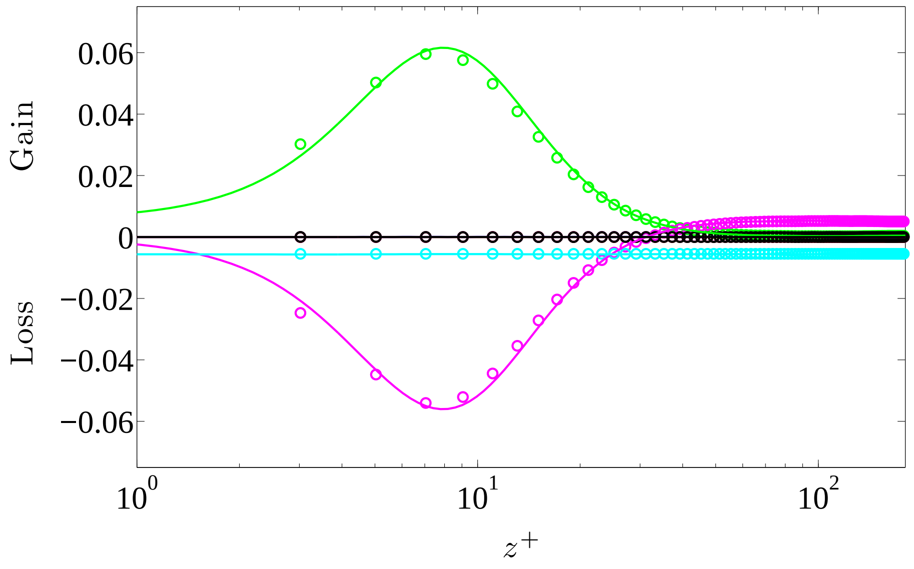


Figure 6.17: Mean streamwise momentum balance for the LBM and FD simulations: ∘ LBM; — FD. The colour denotes the term in the balance equaton: — convection; — pressure gradient; — viscous diffusion; — turbulent diffusion; — body force; — balance.

In a laminar channel flow, the viscous diffusion term would be constant and would balance the body force term. In a turbulent flow, however, the viscous diffusion works together with the turbulent diffusion to offset the action of the body force. As figure 6.17 shows, the viscous and turbulent diffusion terms calculated from the LBM and FD simulations agree qualitatively. However, there are significant errors in both terms in the near-wall region.

The black markers and solid line represent the balance for the LBM and FD simulations, respectively. These terms are calculated by summing all of the individual terms in the mean streamwise momentum equation. If there were any extra terms in the momentum equation recovered from the LBM, the balance term would be nonzero. However, since the balance term is zero for all values of z, this analysis demonstrates that the LBM balances the streamwise momentum equation accurately.

The terms in the mean wall-normal momentum balance calculated from both simulations are shown in figure 6.18. There are only two nonzero terms in the wall-normal momentum equation: the turbulent diffusion and the wall-normal pressure gradient. All of these terms would be zero for a laminar flow. However, in a turbulent flow the Reynolds stress term is nonzero, so a mean wall-normal pressure gradient arises to balance the momentum equation.

Again, if the error terms in the momentum equation recovered by the LBM were significant, the balance of the pressure gradient and the turbulent diffusion would be nonzero. However, since the balance term for the LBM simulation (plotted with black markers) is zero everywhere in the channel, the momentum equation is being balanced accurately.

It should be noted that the terms in the spanwise momentum equation were also

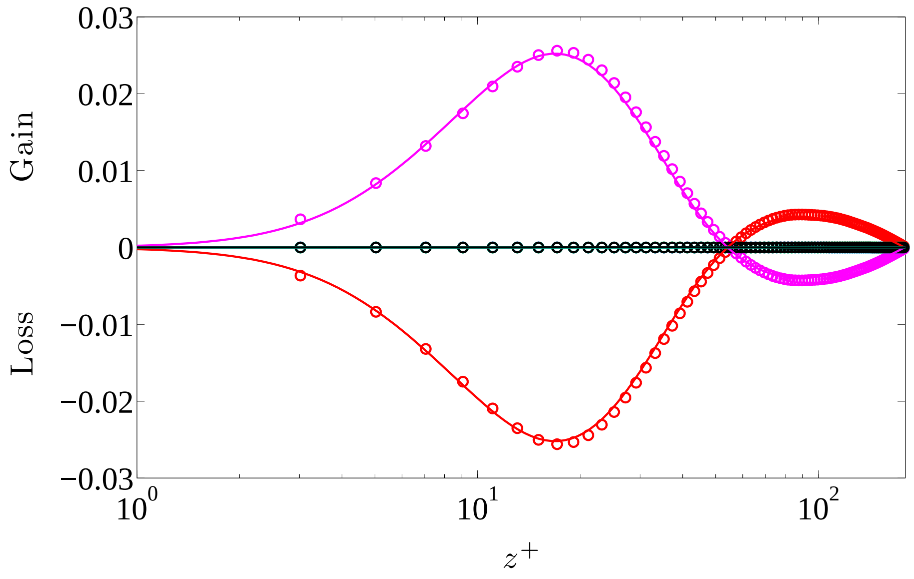


Figure 6.18: Mean spanwise momentum balance for the LBM and FD simulations: ∘ LBM; — FD. The colour denotes the term in the balance equaton: — convection; — pressure gradient; — viscous diffusion; — turbulent diffusion; — body force; — balance.

examined, but, since there are no nonzero terms, they were not plotted. However, the balance term was examined and it was found to be zero for all values of z. Thus, the LBM also balances the spanwise momentum equation.

The fact that the mean momentum equations balance accurately shows that the LBM recovers the NS momentum equations. This provides a verification that the truncated terms in the multiscale expansion of the LBM are indeed insignificant.

6.1.3 One-Dimensional Energy Spectra

In this section, the one-dimensional energy spectra for the velocity and pressure fields are examined. The energy spectrum breaks down the fluctuating component of the flow field into a series of waves with a range of different wavenumbers (equal to 2π divided by the wavelength). Plotting the energy spectrum makes it is possible to study the relative contribution that each wave makes to the total variance.

Consider the turbulent fluctuations (ϕ_i) as a function of x along a line in the streamwise direction of the channel. Since the fluctuations are periodic in this direction they can be written as a Fourier series:

$$\phi'(x,y,z,t) = \sum_{p=1}^{(N_x/2)} \left[a_p(y,z,t)\cos(2\pi px/L_x) + b_p(y,z,t)\sin(2\pi px/L_x)\right] \ , \tag{6.5}$$

where:

$$a_p(y,z,t) = \frac{1}{N_x}\sum_x \phi'(x,y,z,t)\cos(2\pi px/L_x) \ , \tag{6.6}$$

$$b_p(y,z,t) = \frac{1}{N_x}\sum_x \phi'(x,y,z,t)\sin(2\pi px/L_x) \ , \tag{6.7}$$

The coefficients a_p and b_p are called the Fourier coefficients and they are the magnitudes of the waves with wavenumbers $\kappa_x = 2\pi p/L_x$. The contribution that each wave makes to the total variance is equal to

$$2\left(a_p^2 + b_p^2\right) \quad , \tag{6.8}$$

for $p = 1, \ldots, N_x/2 - 1$, and

$$a_{N_x/2}^2 \quad , \tag{6.9}$$

for $p = N_x/2$. Using these expressions, the energy spectrum can be defined so that its integral over the range $0 < \kappa_x < \pi N_x/L_x$ is equal to the total variance. This gives:

$$E_{\phi\phi}(y,z,t) = \left(a_p^2 + b_p^2\right)\frac{L_x}{\pi} \quad , \tag{6.10}$$

which is the definition of the streamwise one-dimensional energy spectrum of ϕ along a single line in the streamwise direction of the channel. Like the turbulence statistics, the energy spectrum should only be a function of the wall-normal coordinate. It can be averaged both in time and in the spanwise direction. Thus, the ensemble averaged streamwise one-dimensional energy spectra is:

$$\left\langle E_{\phi\phi}\right\rangle(z) = \frac{1}{N_t}\sum_t \frac{1}{N_y}\sum_y E_{\phi\phi}(y,z,t) \quad . \tag{6.11}$$

The same analysis can be followed to derive an expression for the spanwise one-dimensional energy spectrum. The wall-normal spectrum is not considered because the flow field is not periodic in that direction.

The streamwise and spanwise one-dimensional energy spectra were calculated for the LBM and FD simulations at the wall-normal locations that correspond to the greatest discrepancy in the variance. The spectra for the streamwise velocity fluctuations are plotted in figure 6.19. Both the x- and y-spectra show that the results from the LBM and FD simulations agree well in the high-wavenumber range, but differ significantly for the lowest-wavenumbers. Thus, the difference between the variance of the streamwise velocity at $z/\delta = 0.12$ is primarily due to differences in the low-wavenumber range.

It should be noted that, since the energy content at the smallest wavenumber is nonzero for both the x- and y-spectra, the domain size in these directions (particularly the streamwise direction) should be larger. However, since the same domain size has been used for both simulations, this result does not affect the validity of this study.

The one-dimensional energy spectra for the spanwise and wall-normal velocities are plotted in figures 6.20 and 6.21. Once again, for both the spanwise and wall-normal velocities, the disagreement between the spectra occurs in the low-wavenumber range. The spectra agree very well for the highest wavenumbers in the simulation.

Figures 6.20 and 6.21 also show that the length of the computational domain in the streamwise direction is too small. For both the spanwise and wall-normal spectra, the lowest wavenumber has the highest energy. This suggests that the domain is too short to contain the turbulent structures with lower wavenumbers than those that

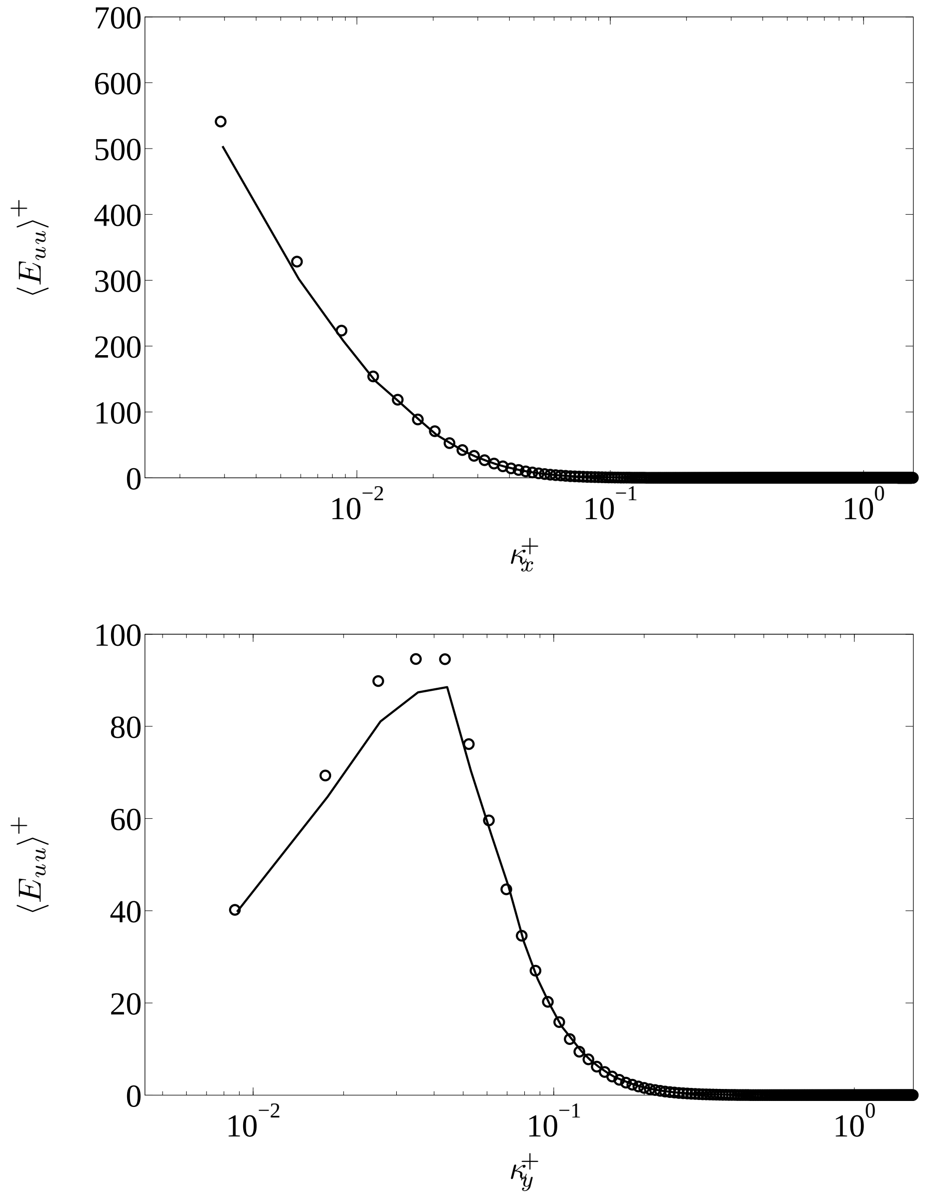
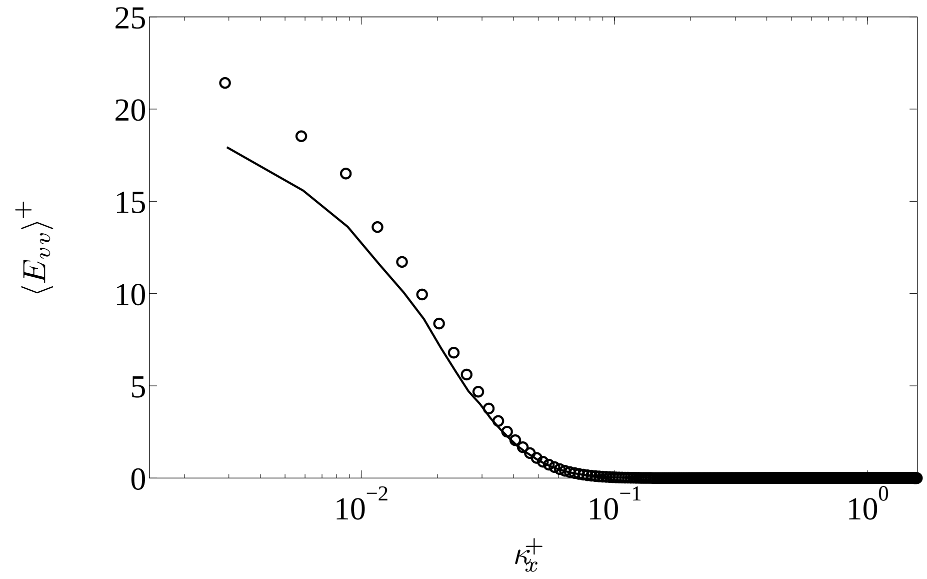


Figure 6.19: One-dimensional streamwise (top) and spanwise (bottom) energy spectra of the streamwise velocity fluctuations at $z/\delta = 0.12$: ○ LBM; — FD.

can be contained within the domain. Once again, this does not affect the validity of this study since the domain sizes for both simulations were the same.

Figure 6.22 shows the pressure spectra for the LBM and FD simulations. Once again, both the streamwise and spanwise spectra differ in the low-wavenumber range.



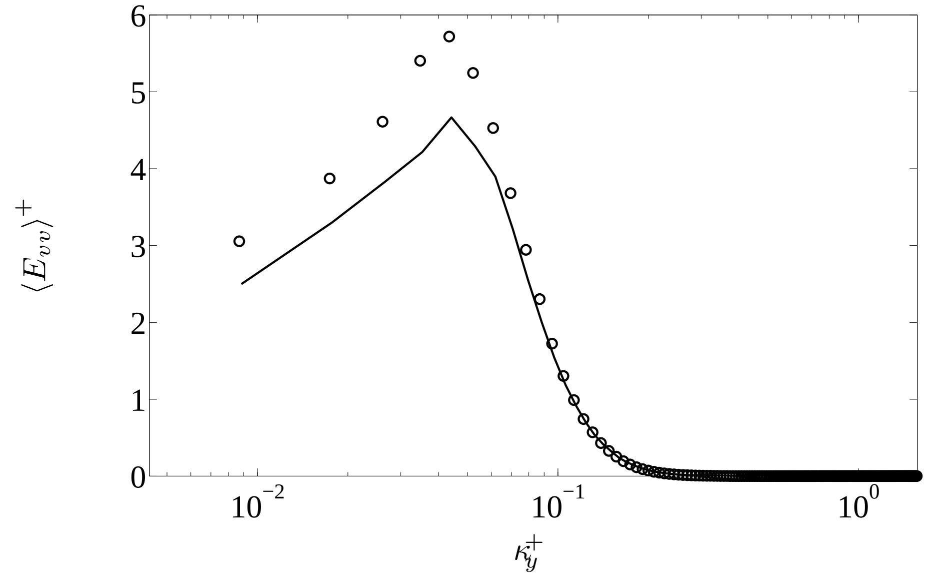


Figure 6.20: One-dimensional streamwise (top) and spanwise (bottom) energy spectra of the spanwise velocity fluctuations at $z/\delta = 0.03$: ∘ LBM; — FD.

Thus, the large discrepancy in the variance of the pressure at $z/\delta = 0.35$ is caused by a difference in the low-frequency pressure fluctuations. Another interesting observation is that the energy in the spanwise spectrum of the pressure field seems to be skewed more toward the small wavenumbers than for the velocity fluctuations. This demonstrates the importance of using a wide computational domain to obtain the

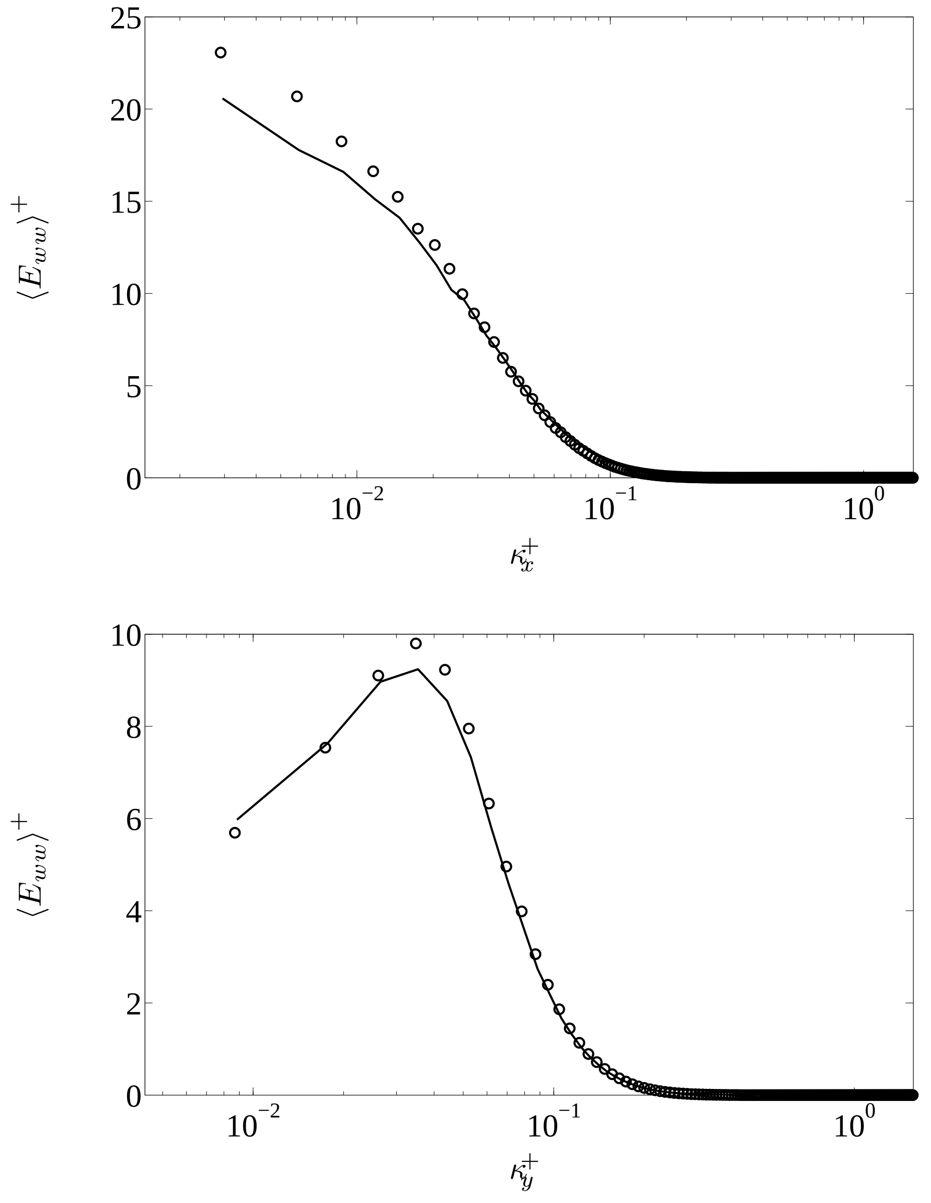


Figure 6.21: One-dimensional streamwise (top) and spanwise (bottom) energy spectra of the wall-normal velocity fluctuations at $z/\delta = 0.35$: ○ LBM; — FD.

correct pressure fluctuations.

In general, the one-dimensional energy spectra predicted by the LBM agree qualitatively with the results from the FD method. However, in each case presented, the LBM over-predicts the energy content of the low-wavenumber component of the

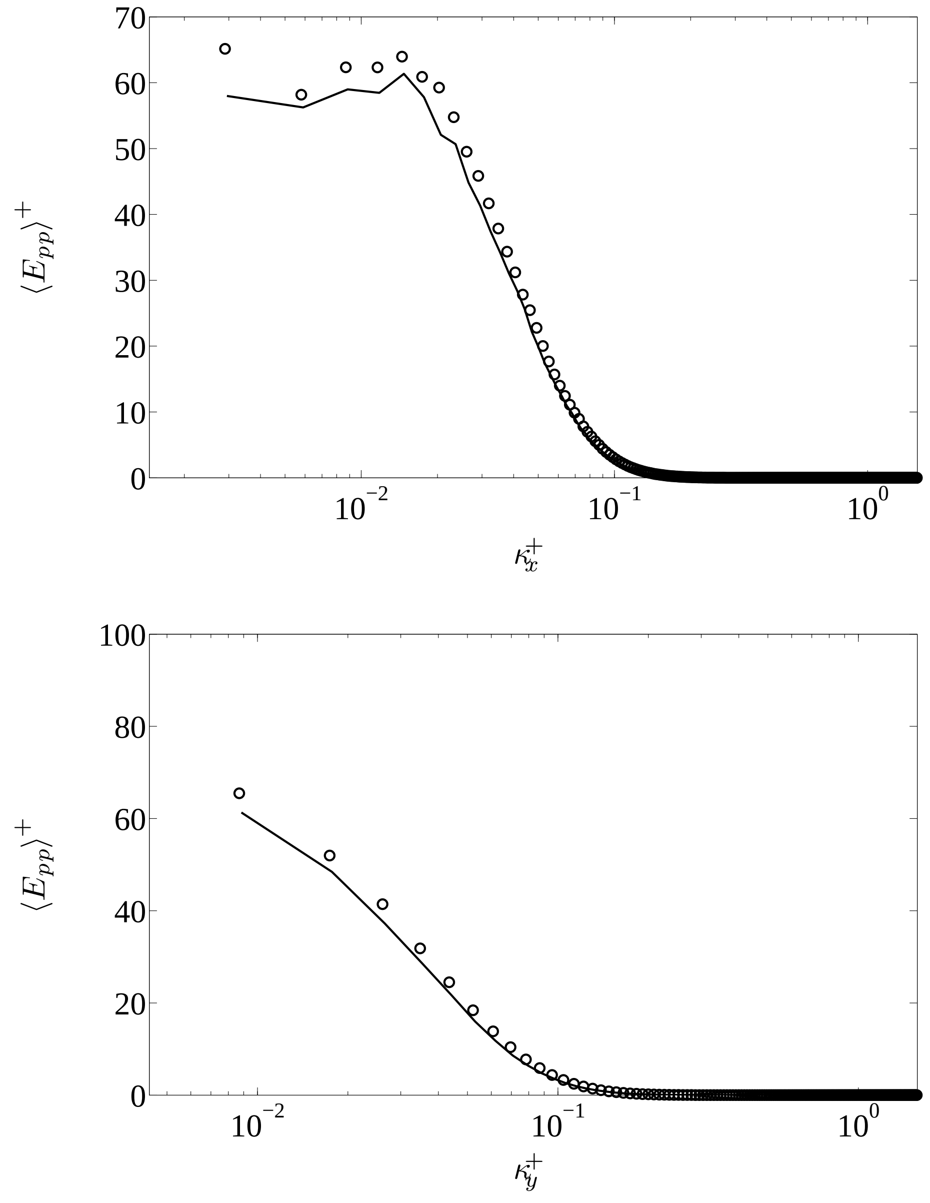


Figure 6.22: One-dimensional streamwise (top) and spanwise (bottom) energy spectra of the pressure fluctuations at $z/\delta = 0.35$: ○ LBM; — FD.

turbulent field. The cause of this over-prediction will be examined in the next section.

6.2 Discussion

In the last section, the turbulence statistics were presented for both the LBM and FD simulations of fully-developed turbulent channel flow. It was found that, while the statistics matched qualitatively, the difference between the two simulations was significant. The largest discrepancy was in the variance of the pressure, which differed by more than 7% in the near-wall region.

Since no turbulence models are used in a DNS, the results obtained should be independent of the numerical method used for the calculations. If all of the numerical errors are made vanishingly small, the difference between the results from two direct numerical simulations should theoretically approach zero. Given that the LBM and FD simulations are both well-resolved direct numerical simulations with the same computational domains, the fact that the pressure variance differs by 7% is a cause for concern and needs to be explained.

In this section, a number of hypotheses are tested to determine the cause of the differences in the turbulence statistics predicted by the LBM and FD simulations.

The first hypothesis tested was that the discrepancy in the turbulence statistics is caused by error in the momentum equation enforced by the LBM. The FD method solves the NS momentum equations directly, while the LBM balances momentum at the level of the particle distributions. Both methods guarantee that the momentum is balanced, but it is possible that these methods could lead to different behaviour. In section 3.1.5, a multiscale analysis was performed on the LBM that showed that the LBM should recover a the same momentum equation as the NS equations, but with some additional terms that are $O(Ma^2)$.

In order to verify that the LBM accurately solves the NS momentum equations, each individual term from the NS momentum equations was calculated directly from the particle distributions from the LBM simulation. As shown in the previous section, the magnitude of the discrepancies in the terms in the momentum balance were of the same order as the discrepancies in the turbulence statistics. However, when all of the terms were summed, both simulations were found to balance to within $O(10^{-5})$. This demonstrates that the NS momentum equations are being accurately balanced by the LBM and that the truncated terms in the multiscale expansion are insignificant. Therefore, the difference in the turbulence statistics must come from another source.

The second hypothesis tested was put forward by Lammers *et al.* [34]. They suggested that the over-prediction of the pressure variance by the LBM was caused by "spurious pressure fluctuations". Spurious pressure fluctuations are caused by numerical instability in the LBM [16]. They occur in simulations of high-Reynolds-number flows in which the grid resolution is too coarse to adequately resolve the smallest scales responsible for energy dissipation. They have a wavelength that is of the order of the grid spacing ($O(\Delta x^+)$).

Since spurious pressure fluctuations are high-frequency oscillations, they should appear in the energy spectra of the pressure (figure 6.22) as a spike in energy in the high-wavenumber range. However, as discussed in the previous section, the disagreement in the pressure spectra occurs in the low-wavenumber range instead. This demonstrates that there are no spurious pressure fluctuations in the current LBM simulation, and therefore they are not the cause of the over-prediction of the pressure variance.

The final hypothesis was that the discrepancy in the turbulence statistics is caused by the compressibility of the LBM. As discussed in section 4.2, the LBM is a compressible method, which means that it allows the density to vary. The finite difference simulation does not allow density fluctuations as it solves the incompressible form of the NS equations.

In order to investigate the effect of compressibility on the variance of the pressure, the results from the LBM and FD simulations were compared to a compressible channel simulation computed with a discontinuous Galerkin (DG) finite element method [50]. The DG method solves the fully-compressible form of the NS equations and allows the density to vary like the LBM. The parameters for the DG simulation were the same as for the LBM, including the Mach number. Figure 6.23 demonstrates that the DG simulation also predicts a larger pressure variance than the FD simulation near the wall. Past $z^+ \approx 30$, the LBM matches extremely well with the DG results. This suggests that the compressibility of the LBM is the cause of the error in the turbulence statistics. Thus, it is differences in the energy equation and the equation of state that cause the error in the pressure fluctuations.

Another observation that strengthens the argument that the compressibility is the cause of the discrepancy between the LBM and FD is the excellent agreement between the LBM results and the analytical solution for laminar channel flow presented in section 3.2. In laminar channel flow, there is no wall-normal pressure gradient, and no pressure fluctuations. Therefore, the Mach number has no effect on the solution. The Mach number in the LBM simulation was set to 0.1, which is significantly different from zero. Despite this difference, the velocity profile obtained by the LBM agreed with the analytical solution to within an error of $O(10^{-4})$. This demonstrates that the LBM agrees well with incompressible solutions for problems where the effect of compressibility is negligible.

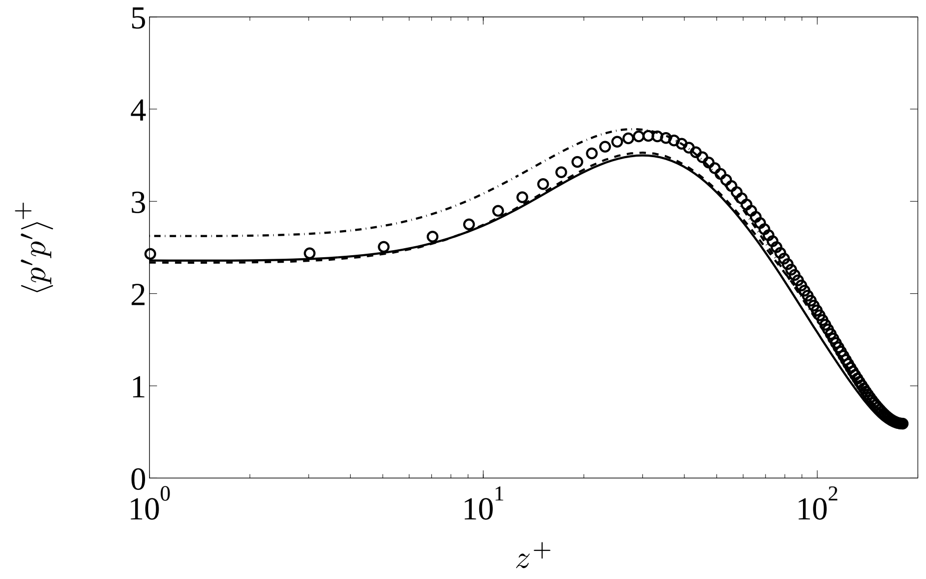


Figure 6.23: Effect of compressibility on the variance of the pressure: ○ LBM; — FD; - - - Moser *et al.* [41]; -·-· Wei and Pollard [50].

There were two reasons that the LBM was compared primarily to incompressible results instead of those from a fully-compressible simulation. First, the LBM is only capable of simulating flows up to approximately $Ma = 0.3$. As such, in the literature (see [19], [34], and [44]) the LBM is considered to be an incompressible method and is used primarily to solve incompressible problems. However, the Mach number is chosen to be as large as possible to maximize the computational efficiency. Second, as outlined in section 4.2, the LBM is not a truly compressible method because it does not have enough degrees of freedom to allow for the temperature to vary. In a fully compressible method, the pressure fluctuations are influenced by both the density and the temperature field. This is why the DG results do not match better with the LBM even though they are both compressible simulations with the same Mach number.

To demonstrate the significance of the temperature variations in a compressible flow, a Reynolds decomposition was applied to the pressure fluctuations from the DG simulation:

$$\langle p'p'\rangle = R^2[\langle \rho'\rho'T'T'\rangle + 2\langle T\rangle\langle \rho'\rho'T'\rangle + 2\langle \rho\rangle\langle \rho'T'T'\rangle + 2\langle \rho\rangle\langle T\rangle\langle \rho'T'\rangle - \langle \rho'T'\rangle^2 + \langle T\rangle^2\langle \rho'\rho'\rangle + \langle \rho\rangle^2\langle T'T'\rangle] \ . \quad (6.12)$$

Each of the above terms in the Reynolds decomposition were computed from the results of the DG simulation. The only terms that were found to be nonzero were: the density variance ($R^2\langle T\rangle^2\langle \rho'\rho'\rangle$), the temperature variance ($\langle \rho\rangle^2 R^2\langle T'T'\rangle$), and the covariance of the density and the temperature ($2\langle \rho\rangle R^2\langle T\rangle\langle \rho'T'\rangle$). These terms are plotted in figure 6.24. This plot shows that magnitude of the density and temperature fluctuations is larger than the pressure variance. However, the density and temperature work against each other to produce a smaller pressure variance. In other words, when the density increases, the temperature, on average, decreases. This is demonstrated by the fact that the covariance term is negative.

Figure 6.25 demonstrates the difference between the density and temperature fluctuations in the LBM and the DG simulations. The density fluctuations in the DG simulation near the wall are over five times larger than in the LBM simulation. The temperature variance for the LBM is by definition zero, while the temperature variations are significant in the DG simulation. Despite these differences both simulations predict similar pressure fluctuations. If the density fluctuations in the LBM were as large as those in the DG simulation, the pressure fluctuations would have to be much

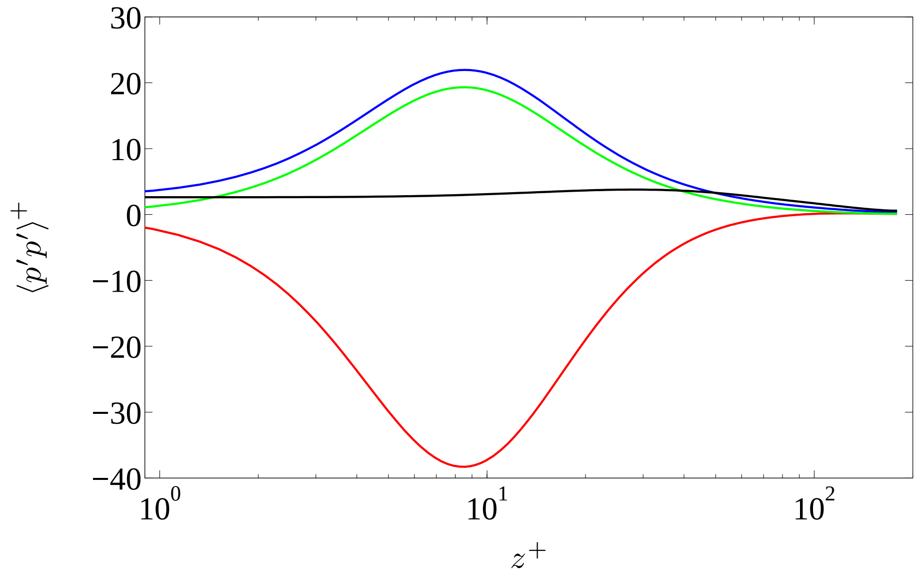


Figure 6.24: Reynolds decomposition of the pressure fluctuations from the DG simulation of Wei and Pollard [50]: 🟥 $(2\langle\rho\rangle R^2 \langle T\rangle \langle\rho' T'\rangle)^+$; 🟦 $(R^2 \langle T\rangle^2 \langle\rho'\rho'\rangle)^+$; 🟩 $(\langle\rho\rangle^2 R^2 \langle T'T'\rangle)^+$; ⬛ $\langle p'p'\rangle^+$

larger. This is because the temperature in the LBM is fixed, and therefore is not capable of counteracting the density fluctuations.

It's important to note that, while the LBM is a compressible method, it is expected to give incompressible solutions if the Mach number is made vanishingly small. However, as discussed earlier, reducing the Mach number of the simulation comes at the cost of increasing the computational time. Since the Mach number and the time step are linearly related, if the Mach number were to be reduced by ten times, the time step would have to be reduced by 10 times as well. This would lead to a 10 times increase in the computational time of the simulation, and is not feasible for a DNS.

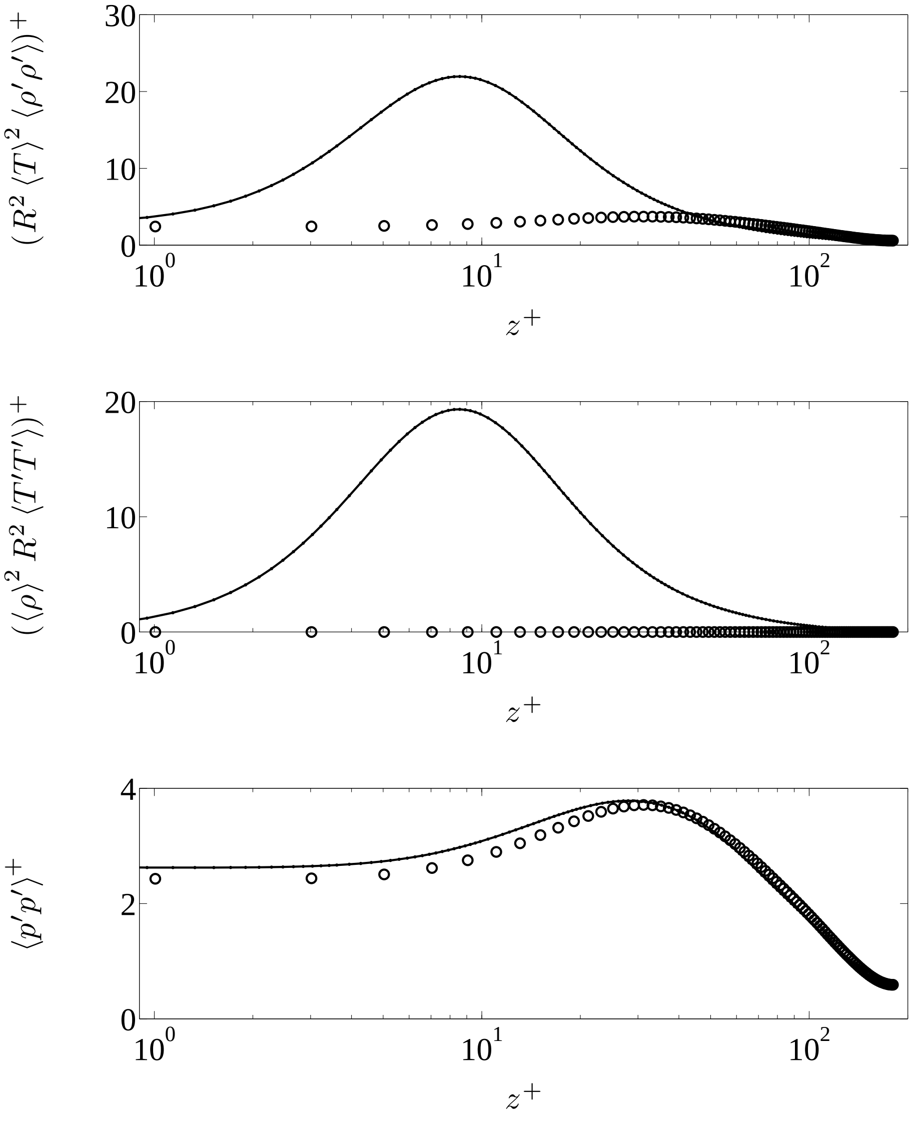


Figure 6.25: Comparison of density (top), temperature (middle), and pressure (bottom) fluctuations from the LBM and DG simulations: ○ LBM; — DG.

Chapter 7

Conclusions and Recommendations

7.0.1 Conclusions

The objective of this work was to validate the LBM for direct numerical simulation of wall-bounded turbulent flows. This was accomplished by developing an LBM code and using it to simulate turbulent channel flow. The results from this simulation were compared to those computed with a conventional Navier-Stokes-based finite difference method. The parameters for the LBM and FD simulations were based on the $Re_\tau = 180$ spectral simulation performed by Moser *et al.* [41].

Both the LBM and FD simulations were run for approximately 60 LETOTs after they became statistically stationary. The turbulence statistics were calculated from snapshots that were recorded every 0.1 LETOTs. While the statistics calculated from the two simulations were found to agree qualitatively, the mean profiles differed by as much as 4% and the variance profiles were different by as much as 7%. The largest difference was in the variance of the pressure fluctuations. Given that both the LBM and FD simulations resolved the entire spectrum of turbulence scales without using any turbulence models, this magnitude of error was deemed to be significant.

The cause of the discrepancies in the turbulence statistics was found to be the compressibility of the LBM. The LBM is a compressible method, while the FD method solves the incompressible form of the NS equations. The pressure variance profile for the LBM was found to agree better with the results from a discontinuous Galerkin simulation that solved the fully-compressible form of the equations. However, it was also found that, since the LBM does not allow the temperature to vary, the nature of the compressibility of the LBM is very different from the DG simulation. The DG simulation predicts much larger density fluctuations than the LBM, but the pressure fluctuations from the two simulations are similar because the temperature fluctuations in the DG simulation counteract the variation of the density. Therefore, the LBM is not recommended for simulating flows for which compressibility is deemed to be important.

The effect of compressibility can be reduced by lowering the Mach number; however, this reduction in error comes at the cost of increasing the simulation time. The Mach number is linearly related to the size of the time step of the simulation. Thus, as the Mach number of the simulation approaches zero, which is preferable for simulating incompressible flows, the size of the time step decreases to zero and the simulation time approaches infinity. For this reason, DNS simulations with the LBM are only feasible at Mach numbers close to the maximum allowable value of 0.3. For smaller Mach numbers, the computational cost of the simulation will greatly exceed that of a comparable NS-based method.

The conclusion of this work is that the LBM is most likely capable of achieving results similar in accuracy to conventional NS-based methods for DNS of incompressible wall-bounded turbulent flows, but at a significantly larger computational cost on conventional high-performance computers. If the computational cost is made comparable by increasing the Mach number of the LBM simulation, a significant error will be introduced due to the compressibility of the LBM. However, the findings of this work do not rule out the possibility that the LBM may be competitive with conventional methods in other situations. For problems with exceedingly complex geometry, the benefit of the LBM's simple implementation may outweigh the loss in precision in comparison to conventional methods. It is also possible that the higher computational cost of the LBM could be reduced by implementing it on emerging high-performance computing technologies such as graphical processing units (GPUs) [47]. The LBM's simple algorithm and scalability could allow it to take better advantage of these new architectures.

This study is the first of its kind in the literature to accurately quantify the difference between the turbulence statistics predicted by the LBM and a conventional Navier-Stokes solver. There have been several other validation studies in the literature ([19], [34], and [44]) but in all of these studies the computational domain of the LBM simulation was made smaller to reduce the computational cost. While these simulations were able to demonstrate a qualitative agreement with the results of Moser *et al.* [41], the error caused by the difference in the domain size confounded the error inherent to the LBM. This study demonstrated that the size of the computational domain has a significant effect on the turbulence statistics, particularly the pressure variance. The majority of the error in the variance of the pressure computed by Lammers *et al.* [34] was found to be caused by their narrow computational domain. In this work, special care was taken to ensure that the computational domains for both the LBM and FD simulations were identical to the one used by Moser *et al.*. This removes the effect of the domain size on the turbulence statistics. Therefore, the differences in the mean and variance profiles in this work are solely due to differences in the LBM and FD algorithms.

Since the parameters of the LBM and FD simulations were as similar as possible, this study provides a benchmark for the performance of the LBM in relation to a conventional solver. It was found that the LBM required roughly 24 times more computational resources than the FD method. This was primarily due to the fact that the time step in the LBM was approximately six times smaller than the time step in the FD simulation. Once again, the LBM requires a much smaller time step because it is a compressible method and therefore must resolve the relatively fast moving sound waves in the simulation. The FD method is able to use a much larger time step because it solves the incompressible form of the equations. Additionally, the LBM required roughly twice as many computational nodes than the FD simulation because it was not able to stretch the grid in the streamwise direction. These results demonstrate that, for flows with simple geometries, the computational cost of the LBM is significantly higher than for conventional NS-based solvers. However, it is possible that the LBM would compare more favourably to NS-based methods that are configured for more complex geometries. Regardless, this work provides some insight into the relative cost of DNS of turbulent flows with the LBM.

7.0.2 Recommendations

This study found that the LBM is capable of producing accurate results for simple wall-bounded incompressible turbulent flows, but its computational cost is significantly larger relative to conventional NS-based methods. However, as mentioned earlier, the LBM may be more competitive with conventional methods for problems with complex geometries. This is because some of the simplifications that increase the efficiency of conventional methods cannot be made for complex problems. In contrast, the LBM would need very little modification to simulate complex flows. Further work should be undertaken to compare the relative performance of the LBM and a conventional solver for a flow with very complex geometries.

It may also be possible to reduce the relative computational cost of the LBM by implementing it on emerging high-performance computing platforms. Due to its simple implementation, the LBM can be run on graphical processing units (GPUs) or even implemented on a field programmable gate array (FPGA), which is an integrated circuit that is designed to be reconfigurable. These new technologies have a lot of promise in the field of high-performance computing, and studies should be performed to assess the performance increase that can be achieved with these platforms.

References

[1] G. Amati, S. Succi, and R. Piva. Massively parallel lattice-Boltzmann simulation of turbulent channel flow. *International Journal of Modern Physics C*, 8(4):869–877, August 1997.

[2] G. Amati, S. Succi, and R. Piva. Preliminary analysis of the scaling exponents in channel flow turbulence. *Fluid Dynamics Research*, 24(4):201–209, April 1999.

[3] R. Benzi, M. V. Struglia, and R. Tripiccione. Extended self-similarity in numerical simulations of three-dimensional anisotropic turbulence. *Physical Review E*, 53(6):R5565–R5568, June 1996.

[4] R. Benzi and S. Succi. Two-dimensional turbulence with the lattice Boltzmann-equation. *Journal Of Physics A-Mathematical and General*, 23(1):L1–L5, January 1990.

[5] A. R. Berker. Intégration des équations du mouvement d'un fluide visqueux incompressible. In S. Függe, editor, *Encyclopedia of Physics*, volume 8, pages 1–384. Springer, 1963.

[6] D. J. Bespalko. Direct numerical simulation of turbulent channel flow using the lattice Boltzmann method. Master's thesis, Queen's University at Kingston, 2006.

[7] D. J. Bespalko, A. Pollard, and M. Uddin. Direct numerical simulation of fully-developed turbulent channel flow using the lattice Boltzmann method and analysis of OpenMP scalability. In D. J. K. Mewhort, N. M. Cann, G. W. Slater, and T. J. Naughton, editors, *High Performance Computing Systems and Applications*, pages 1–19. Springer, 2010.

[8] P. L. Bhatnagar, E. P. Gross, and M. Krook. A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. *Physical Review*, 94(3):511–525, May 1954.

[9] G. A. Bird. Monte-Carlo simulation of gas-flows. *Annual Review of Fluid Mechanics*, 10:11–31, 1978.

[10] L. Boltzmann. *Lectures on Gas Theory.* University of California Press, 1964.

[11] S. Chapman and T.G. Cowling. *The Mathematical Theory of Non-Uniform Gases: An Account of the Kinetic Theory of Viscosity, Thermal Conduction and Diffusion in Gases.* Cambridge University Press, 1970.

[12] S. Chen and G. D. Doolen. Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30:329–364, 1998.

[13] S. Y. Chen, H. D. Chen, D. Martinez, and W. Matthaeus. Lattice Boltzmann model for simulation of magnetohydrodynamics. *Physical Review Letters*, 67(27):3776–3779, December 1991.

[14] S. Y. Chen, D. Martinez, and R. W. Mei. On boundary conditions in lattice Boltzmann methods. *Physics of Fluids*, 8(9):2527–2536, September 1996.

[15] S. Y. Chen, Z. Wang, X. W. Shan, and G. D. Doolen. Lattice Boltzmann computational fluid-dynamics in 3 dimensions. *Journal of Statistical Physics*, 68(3-4):379–400, August 1992.

[16] D d'Humières, I Ginzburg, M Krafczyk, P Lallemand, and LS Luo. Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Philosophical Transactions Of The Royal Society Of London Series A-Mathematical Physical And Engineering Sciences*, 360(1792):437–451, MAR 15 2002.

[17] L. Djenidi. Lattice-Boltzmann simulation of grid-generated turbulence. *Journal of Fluid Mechanics*, 552:13–35, April 2006.

[18] J. G. M. Eggels. Direct and large-eddy simulation of turbulent fluid flow using the lattice-Boltzmann scheme. *International Journal of Heat and Fluid Flow*, 17(3):307–323, June 1996.

[19] J. G. M. Eggels, F. Unger, M. H. Weiss, J. Westerweel, R. J. Adrian, R. Friedrich, and F. T. M. Nieuwstadt. Fully-developed turbulent pipe-flow - A comparison between direct numerical-simulation and experiment. *Journal Of Fluid Mechanics*, 268:175–209, June 1994.

[20] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the Navier-Stokes equations. *Physical Review Letters*, 56(14):1505–1508, April 1986.

[21] B. J. Geurts. *Elements of Direct and Large-eddy Simulation.* Edwards, 2004.

[22] S. Harris. *An Introduction to the Theory of the Boltzmann Equation.* Dover Publications Inc., 2004.

[23] X. Y. He and L. S. Luo. Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation. *Physical Review E*, 56(6):6811–6817, December 1997.

[24] X. Y. He, Q. S. Zou, L. S. Luo, and M. Dembo. Analytic solutions of simple flows and analysis of nonslip boundary conditions for the lattice Boltzmann BGK model. *Journal of Statistical Physics*, 87(1-2):115–136, April 1997.

[25] F. J. Higuera and J. Jimenez. Boltzmann approach to lattice gas simulations. *Europhysics Letters*, 9(7):663–668, August 1989.

[26] K. Huang. *Statistical Mechanics*. John Wiley & Sons, Inc., 1963.

[27] T. Inamuro, M. Yoshino, and F. Ogino. Non-slip boundary-condition for lattice Boltzmann simulations. *Physics of Fluids*, 7(12):2928–2930, December 1995.

[28] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *Siam Journal On Scientific Computing*, 20(1):359–392, 1998.

[29] A. Keating, G. De Prisco, and U. Pionielli. Interface conditions for hybrid RANS/LES calculations. *International Journal Of Heat And Fluid Flow*, 27(5):777–788, October 2006.

[30] A. Keating, U. Piomelli, E. Balaras, and H. J. Kaltenbach. A priori and a posteriori tests of inflow conditions for large-eddy simulation. *Physics Of Fluids*, 16(12):4696–4712, December 2004.

[31] A. Keating, U. Piomelli, K. Bremhorst, and S. Nesic. Large-eddy simulation of heat transfer downstream of a backward-facing step. *Journal Of Turbulence*, 5:020, May 2004.

[32] J. Kim, P. Moin, and R. Moser. Turbulence statistics in fully-developed channel flow at low Reynolds-number. *Journal of Fluid Mechanics*, 177:133–166, April 1987.

[33] J. M. V. A. Koelman. A simple lattice Boltzmann scheme for Navier-Stokes fluid-flow. *Europhysics Letters*, 15(6):603–607, July 1991.

[34] P. Lammers, K. N. Beronov, R. Volkert, G. Brenner, and F. Durst. Lattice BGK direct numerical simulation of fully developed turbulence in incompressible plane channel flow. *Computers & Fluids*, 35(10):1137–1153, December 2006.

[35] R. L. Liboff. *Kinetic Theory: Classical, Quantum, and Relativistic Descriptions.* Prentice-Hall, 1990.

[36] L.-S. Luo, D. Qi, and L.-P. Wang. Applications of the lattice Boltzmann method to complex and turbulent flows. Technical report, ICASE, July 2002.

[37] D. O. Martinez, W. H. Matthaeus, S. Chen, and D. C. Montgomery. Comparison of spectral method and lattice Boltzmann simulations of 2-dimensional hydrodynamics. *Physics of Fluids*, 6(3):1285–1298, March 1994.

[38] G. R. Mcnamara and G. Zanetti. Use of the Boltzmann-equation to simulate lattice-gas automata. *Physical Review Letters*, 61(20):2332–2335, November 1988.

[39] R. W. Mei, L. S. Luo, P. Lallemand, and D. d'Humieres. Consistent initial conditions for lattice Boltzmann simulations. *Computers & Fluids*, 35(8-9):855–862, September 2006.

[40] P. Moin and K. Mahesh. Direct numerical simulation: A tool in turbulence research. *Annual Review of Fluid Mechanics*, 30:539–578, 1998.

[41] R. D. Moser, J. Kim, and N. N. Mansour. Direct numerical simulation of turbulent channel flow up to $Re_\tau = 590$. *Physics of Fluids*, 11(4):943–945, April 1999.

[42] D. R. Noble, S. Y. Chen, J. G. Georgiadis, and R. O. Buckius. A consistent hydrodynamic boundary-condition for the lattice Boltzmann method. *Physics of Fluids*, 7(1):203–209, January 1995.

[43] S. B. Pope. *Turbulent Flows.* Cambridge University Press, 2000.

[44] Kannan N. Premnath, Martin J. Pattison, and Sanjoy Banerjee. Generalized lattice Boltzmann equation with forcing term for computation of wall-bounded turbulent flows. *Physical Review E*, 79(2, Part 2), FEB 2009.

[45] Y. H. Qian. *Lattice gas and lattice kinetic theory applied to the Navier-Stokes equations.* PhD thesis, Université Pierre et Marie Curie, Paris, 1990.

[46] P. Roache. *Verification and Validation in Computational Science and Engineering.* Hermosa Publishers, 1998.

[47] Mohammad Amin Safi, Mahmud Ashrafizaadeh, and Amir Ali Ashrafizaadeh. Lattice Boltzmann Simulation of Binary Mixture Diffusion Using Modern Graphics Processors. In *International Conference on Fluid Mechanics, Heat Transfer, and Thermodynamics*, 2011.

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

[49] H. Tennekes and J.L. Lumley. *A First Course in Turbulence.* MIT Press, Cambridge, Massachusetts, 1972.

[50] Liang Wei and Andrew Pollard. Effect of Mach number on correlations in compressible turbulent channel flows. In K. Hanjalić, Y. Nagano, and S. Jakirlić, editors, *Turbulence, Heat and Mass Transfer 6*, pages 587–590. Begell House Inc., 2009.

[51] F.M. White. *Viscous Fluid Flow.* McGraw-Hill, 1991.

[52] D. A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models.* Lecture Notes in Mathematics. Springer-Verlag, 2000.

[53] D. Yu and S. S. Girimaji. DNS of homogenous shear turbulence revisited with the lattice Boltzmann method. *Journal of Turbulence*, 6(6), 2005.

[54] H. D. Yu, S. S. Girimaji, and L. S. Luo. DNS and LES of decaying isotropic turbulence with and without frame rotation using lattice Boltzmann method. *Journal of Computational Physics*, 209(2):599–616, November 2005.