UNIVERSITÉ DE LILLE 1 - SCIENCES ET TECHNOLOGIES

DOCTORAL THESIS

Simulation of wall-bounded turbulent convective flows by Finite Volume Lattice Boltzmann Method

Presented by

Kalyan SHRESTHA

Under supervision of:

G. MOMPEAN, Professeur, Université Lille 1

E. CALZAVARINI, Maître de Conférence, Université Lille 1

A thesis submitted in fulfilment of the requirements for the degree of ***Doctor of Philosophy*** *in* ***Mechanical Engineering*** *in the*

ER1 Research group, Laboratoire de Mécanique de Lille

Ecole Doctorale Sciences pour l'Ingénieur EDSPI 072

PRES Université Lille Nord-de-France

Laboratoire Mécanique Lille

Université de Lille 1 SCIENCES ET TECHNOLOGIES

30 November 2015

Résumé

La méthode Lattice Boltzmann (LBM) est une alternative viable *à* la simulation directe (DNS) des équations de Navier et Stokes, particulièrement en Mécanique des Fluides. La clé de son succès se situe dans l'exactitude et la simplicité de l'algorithme stream-collision. De plus, La propriété conforme de parallélisation rend la LBM efficace. L'inconvénient majeur de cette méthode provient de la limitation aux mailles cubiques spatialement uniformes qui ne sont pas assez fines pour résoudre la turbulence près de la paroi. Pour y remédier, plusieurs extensions de la LBM aux mailles non homogènes ont été proposées. Ces techniques ont été revisitées dans la thèse. La revue des différentes techniques de raffinement de maillage [1–5] montre que la meilleure technique de raffinement remplit certains critères. D'une part, elle doit satisfaire aux lois de conservation et d'autre part elle doit être stable. Elle suggère l'adoption des approches de type Volumes Finis (FV LBM). Une revue de ces techniques [6–12] a permis de conclure que bien qu'intéressantes, elles présentent de nombreux inconvénients et ne possèdent pas le niveau de maturité envisagé. A ce jour, la question de savoir si une version optimisée de la FV LBM produit un meilleur résultat que l'algorithme LBM basé sur le *streaming* pour les simulations en Mécanique des Fluides reste ouverte. Cette étude présente une méthode de discrétisation de type Volumes Finis (FV) pour Lattice Boltzmann (LB) de haute précision et avec un faible co*û*t de calcul. Afin d'évaluer la performance de la méthode FV nous effectuons une comparaison systématique axée sur la précision et les performances de calcul avec la méthode de Lattice Boltzmann *streaming* standard (ST). A notre connaissance une telle comparaison n'a jamais été réalisée. En particulier, nous cherchons *à* clarifier si et dans quelles conditions l'algorithme proposé et plus généralement tout algorithme FV peut *ê*tre considéré comme la méthode de choix pour les simulations en Mécanique des Fluides. Pour cette raison l'analyse comparative est en outre étendue aux écoulements réels, en particulier les écoulements thermiques dans des conditions turbulentes. Nous présentons la première simulation des écoulements convectifs *à* haut nombre de Rayleigh réalisée avec une méthode Lattice Boltzmann de type FV avec des mailles réduites près de la paroi.

Mots clés : Méthode Lattice Boltzmann, Volumes Finis (FV), système de Rayleigh-Bénard

Abstract

Lattice Boltzmann Method (LBM) has become a viable alternative to Navier-Stokes Direct Numerical Simulations (DNS) in fluid dynamics research. The key of this success is firstly the accuracy/simplicity of the stream-collision algorithm. Next is the parallelization compliant property that makes LBM computationally efficient. One shortcoming however, comes from the limitation to spatially uniform cubic grids, which becomes particularly critical in the simulation of realistic turbulent flows where near-to-walls grid refinements are needed. To overcome this, several LBM extension to non-homogeneous grids have been proposed. These techniques have been reviewed in this thesis. Such review on different mesh refinement techniques [1–5] suggests that a better refinement technique should fulfil some properties. On one hand it should obey conservation laws and on the other hand, it has to be stable. This suggests a pathway to adopt Finite Volume approaches (FV LBM). A review on such volumetric approach to LBM [6–12] conclude that although interesting, at present such methods suffer from several drawbacks, and they lack a desired level of maturity. The question whether an optimized version of FV LBM can outperform *streaming* based LBM algorithm for fluid-dynamics simulations is still open. In this study, a new Finite Volume (FV) discretization method for the Lattice Boltzmann (LB) equation that combines high accuracy with limited computational cost is presented. In order to assess the performance of the FV method we carry out a systematic comparison, focused on accuracy and computational performances, with the standard *streaming* (ST) Lattice Boltzmann equation algorithm. To our knowledge such a systematic comparison has never been previously reported. In particular we aim at clarifying whether and in which conditions the proposed algorithm, and more generally any FV algorithm, can be taken as the method of choice in fluid-dynamics LB simulations. For this reason the comparative analysis is further extended to the case of realistic flows, in particular thermally driven flows in turbulent conditions. We report the first successful simulation of high-Rayleigh number convective flow performed by a Lattice Boltzmann FV based algorithm with wall grid refinement.

Keywords : Lattice Boltzmann method, Finite Volume (FV), Rayleigh-Bénard system

Acknowledgements

First of all, I would like to express my gratitude to my co-supervisor Dr. Enrico CALZAVARINI, who motivated me and provided guidance at all times throughout the project. He has not only been my co-advisor but also a friend, a brother, a mentor and a guardian. I am always indebted to him for imparting to me the knowledge on Lattice Boltzmann Method. I would also like to thank my supervisor Prof. Gilmar MOMPEAN, who provided advice and guidance constantly. He has always been supportive as a fatherly figure to me. This thesis would not have been successful without both of them.
I am also grateful to all the researchers in the Laboratoire de Mécanique de Lille, who have directly or indirectly supported me during my PhD. To all my friends around the globe - thank you all for being there and understanding and supporting me at all times. No one can imagine a life without friends. There is not enough space to list all the names of my friends, but I thank each and everyone of you.
Finally, I would like to dedicate this thesis to my parents for their unconditional love and support throughout my life. I owe them every bit of success.

This research has been supported by a collaboration between academic and industrial partners promoted by the organization INNOCOLD (www.innocold.org) and in part by the French National Agency for Research (ANR) by the grant (SEAS: ANR-13-JS09-0010).

Contents

List of Figures

List of Tables

Abbreviations

CFD	**C**omputational **F**luid **D**ynamics
LBM	**L**attice **B**oltzmann **M**ethod
FHP	**F**risch **H**asslacher **P**omeau
BGK	**B**hatnagar **G**ross **K**rook
MRT	**M**ulti- **R**elaxation **T**ime
NS	**N**avier- **S**tokes
CFL	**C**ourant- **F**riedrichs- **L**ewy
FD	**F**inite **D**ifference
FV	**F**inite **V**olume
FE	**F**inite **E**lement
ST	**S**treaming
DVBE	**D**iscrete- **V**elocity **B**oltzmann **E**quation
LB	**L**attice **B**oltzmann
BFECC	**B**ack-and-**F**orth **E**rror **C**ompensation and **C**orrection
LGA	**L**attice **G**as **A**utomaton
DNS	**D**irect **N**umerical **S**imulation
TVD	**T**otal **V**ariation **D**iminishing
CV	**C**ontrol **V**olume
QUICK	**Q**uadratic **U**pstream **I**nterpolation for **C**onvective **K**inematics
TANH	**H**yperbolic **T**angent
SINH	**H**yperbolic **S**ine
RB	**R**ayleigh **B**énard
2D	**T**wo- **D**imensional
3D	**T**hree- **D**imensional
BC	**B**oundary **C**ondition

Symbols

Kn	Knudsen number
m	mass
x	position
c	particle velocity
c_0	ideal speed of sound
u	bulk fluid velocity
$\mathbf{p}$	momentum
V	volume
N	number of particles
t	time
$f\ or\ g$	distribution functions
f^0	equilibrium distribution function
f^{neq}	non-equilibrium distribution function
f_i	discrete- velocity distribution function
w_i	weighting coefficients
$\tilde{f}$	redefined distribution function
$\tilde{f}^*$	distribution function representing intermediate value
$\mathbf{F}$	body force
$\mathbf{a}$	acceleration
p	pressure
$\bar{v}$	relative speed of molecules
k_B	Boltzmann constant
T	temperature
$\Delta x_f, \Delta x_c$	grid spacing for fine and coarse grid
$\Delta t_f, \Delta t_c$	time spacing for fine and coarse grid

n	ratio of lattice space between the two-grid system
A	negative viscosity
H^x	interpolation function
S_e	Finite Element domain
S_j	boundary surfaces
$\mathbf{n}$	normal vector to boundary
Nu	Nusselt number
Ar	Aspect ratio
Ra	Rayleigh number
Pr	Prandtl number
L_B	Bolgiano length
ρ	density
ρu	momentum density
ρE	energy density
ρe	moment of energy
ν	viscosity
β	thermal expansion coefficient
Ω	solid angle
σ	differential cross-section
$\Omega(f)$	collision operator
τ	relaxation time
$\tilde{\tau}$	redefined relaxation time
Π_0^0	zeroth moment of f^0
Π_α^0	first moment of f^0
$\Pi_{\alpha\beta}^0$	second moment of f^0
$\Pi_{\alpha\beta\gamma}^0$	third moment of f^0
ϵ	expansion parameter
$\mathcal{T}$	large eddy turnover time

Dedicated to my parents

Motivation and Objectives

Motivation

This PhD project has been funded by INNOCOLD (www.innocold.org) a public-private consortium for the promotion of the research on low-temperature industry in relation with Liquified Natural Gas (LNG) manipulation and its distribution. My thesis work is part of a wider research proposal which involves three PhD students and several faculty researchers from Lille 1 and Côte d'Opale Universities. The aim of the project is to conceive a computational fluid dynamic model for the description of the accidental LNG leakage and dispersion in the environment, with a special focus on the risk assessment at the LNG harbour terminal under construction near the city of Dunkerque, in the North of France. In such context, to have a real-time tool for the prediction of the spilling area, of the vaporisation process and of the concomitant wind dispersion of natural gas, would be of considerable social and economical value.

Probably the most challenging aspect of the project is that the numerical model should be able to account for processes which ranges over several orders of magnitude of spatial and temporal scales. If one considers space for instance, one shall be able to describe the liquid-gas phase transition of LNG which happens at the molecular scale up to the fluid-dynamics scale of the the gas clouds eddies around the industrial installations which are of the order of hundreds of meters.

Which is the best computational approach to tackle such a problem? Common micro-scale meteorology simulations are based on discretized forms of the macroscopic fluid-dynamics equations, i.e., Navier-Stokes equations. However, when multi physics phenomena are involved, as for instance phase-change, or thermal-mechanical coupling with complex boundary conditions, this type of approaches might not be the most convenient. The Lattice Boltzmann (LB) method has emerged in recent years as a viable alternative to traditional CFD methods, for instance it has become the privileged numerical tool in car industry. The above mentioned project granted by INNOCOLD aims at developing a LB tool for the LNG industry. In order to reach such a goal the proposal has been organised in different tasks, one of which is covered by this thesis. My thesis work deals with *the simulation of three-dimensional, buoyancy-driven, turbulent flows over a complex domain by means of the Lattice Boltzmann method.*

Objectives

A developed turbulent flow in the presence of a bounding geometry or local forcing term (for instance buoyancy) develops space inhomogeneities and as such in numerical simulations a grid refinement approach becomes necessary. For the simulation of realistic turbulent flows, grid refinement is unavoidable not just to increase accuracy but also to save memory usage and computational power. However, it is known that the standard LB method is restricted to regular cubic grids. There have been many efforts to tackle this restriction: researchers have proposed different mesh refinement techniques based on the introduction of locally nested monospaced grids, or based on interpolation schemes, or on Finite Difference or Finite Element or Finite Volume (FV) discretization approaches. However, many of the suggested extensions have important drawbacks. In general all such reformulation are computationally more expensive, or introduce extra stability limitations enforced by space/time discretization which were not present in the original LB implementation.

In this thesis my efforts puts under scrutiny the Finite-Volume discretization of the Lattice Boltzmann equation, investigating if it can be a practical approach for the simulation of turbulent flows, particularly in situation where boundaries and at the same time buoyancy forces are present. The first part of the thesis aims at developing a novel FV formulation for the Lattice Boltzmann equation which besides a high level of accuracy also displays a contained computational cost. The developed FV LB method is methodically and carefully compared with the standard *streaming* LB method. In order to do so a robust state-of-the-art parallel *C* language code is developed where both the above mentioned LB algorithms coexists simultaneously. The second part of my thesis addresses all the topics (boundary conditions, buoyancy, multi-component, large eddy simulation, complex geometry) that may be required to simulate heavy/light gas dispersion in the atmosphere.

The ***outline of this thesis*** is as follows: **Chapter 1** will introduce the Boltzmann equation, derive the classical LB BGK equations from the kinetic theory and it will show its the connections with the equation of Fluid Mechanics. After this introductory chapter on LBM, the work is divided into two parts. **Part I** will include **Chapters 2 - 5** which will discuss the research on FV approach to LB equation. **Part II** will include **Chapter 6** which will describe the structure of the code and show validations for the standard LBM.

Classical LBM is limited to uniform grid therefore there have been various efforts to impose mesh refinements. **Chapter 2** will give an overview of mesh refinement techniques tried over years in the development of LBM . In **Chapter 3**, we reason why FV approach is a good choice out of all other mesh refinement techniques and also present a review on all the ideas involved in the development of FV LB approach. In **Chapter 4**, we will propose a novel FV LB approach that improves on the shortcomings of the previous works. Also, a systematic comparison with standard LB in terms of accuracy and computational performances is carried out. **Chapter 5** will present such comparisons for a realistic flow (high Rayleigh number, 3D Rayleigh Bénard system). **Chapter 6** is the next part of the thesis where the code is briefly described with a flowchart and various validations of the standard LB algorithm are presented. Finally in **Chapter 7**, we will discuss the conclusions of the research and perspectives on how it can be further continued.

Related articles

K Shrestha, G Mompean, and E Calzavarini. **Finite Volume *vs.* Streaming based Lattice Boltzmann algorithm for fluid-dynamics simulations: a one-to-one accuracy and performance study.** *Submitted to Physical Review E*, 2015 http://arxiv.org/abs/1505.03271v2

23rd International Conference on Discrete Simulation of Fluid Dynamics, ***Oral presentation:*** **Can Finite-Volume Lattice Boltzmann outperform Streaming-based algorithm in fluid-dynamics simulations? A one-to-one accuracy and performance study.** 28 July - 1 August 2014, Paris (France)

Chapter 1

Lattice Boltzmann Method

Conventional Computational Fluid Dynamics (CFD) methods have been within the framework of Navier-Stokes equations for very long. In this thesis we are not going to talk about the conventional CFD which used Navier-Stokes equations. We are rather interested in a recently advancing method called the Lattice Boltzmann Method (LBM). Of course, its increasing popularity relies in its merits. As the thesis advances we will talk in detail about the equations used, theoretical aspects, merits and demerits, areas of applications etc. For now, this chapter tries to give some preliminary insight into its level of description of the flow, history and the basics of this technique before going to the technical explanation.

1.1 Levels of description of fluid

Now, lets see why LBM is a method different than the conventional one, why are we interested in this and why is it getting popular. Short description of the method at this introductory chapter should give a good general physical insight to the method for describing fluids. Let us take any fluid *eg.* water. Anyone depending on human eye would describe (in a fluid dynamics sense) as a continuum medium and uniformly distributed. This scale of description (called the macroscopic scale), looks for macroscopic fluid properties like viscosity, density, velocity. *Fluid mechanics equations describes this system.* These equations are enough to describe many flows and are widely popular since long. However, we already understand that we have assumed the continuum criteria *i.e.* Kn < 0.01. Although it is enough to describe most of the flows, we know that it is not descriptive to the molecular level.
Now we want to use a microscopic sight and see the detailed description of the system. The fluid consists of group of atoms and molecules separated by intermolecular distance.

A picture of fluid at microscopic level should look like molecules moving around and colliding with other molecules thus exchanging momentum and velocity. At this level, there are no viscosity, thermal conductivity etc. So at this level of description, each molecule can be best tracked with mass m_i, position $\boldsymbol{x}_i = (x_i,y_i,z_i)$ and velocity c_i. *These can be described by Newtonian mechanics equations.* Of course we are looking at the same system described in the previous paragraph, therefore there must be a link between these descriptions. Considering a small volume V, the macroscopic properties like density, momentum and energy can be described by microscopic quantities like mass, position and velocity as follows:

$$\rho(\mathbf{x},t) = \lim_{V\to 0}\left(\frac{1}{V}\sum_{x_i\in V} m_i\right) \tag{1.1}$$

$$\rho\mathbf{u}(\mathbf{x},t) = \lim_{V\to 0}\left(\frac{1}{V}\sum_{x_i\in V} m_i c_i\right) \tag{1.2}$$

$$\rho E(\mathbf{x},t) = \lim_{V\to 0}\left(\frac{1}{2V}\sum_{x_i\in V} m_i |c_i|^2\right) \tag{1.3}$$

At this level, there are no fluid properties like viscosity, density etc. This higher level of description should give a detailed information of the system but a simple calculation on the need of computing power for simulating most practical fluid flows makes it an impossible task. The number of molecules in an infinitesimal volume V should be in the order of Avogadro's number (10^{23}). Each molecule atleast requires information of 3 position vectors and 3 velocity vectors. So, N molecules would require 6N information. That means: given the initial and boundary conditions, Newton equations can be fully solved for each state of the system yielding a set of 6N functions of time. The data storage would overflow very soon. [22] gives an interesting example to illustrate this. For a centimeter cube of argon at temperature = 300K, pressure = 1atm, it is estimated that it would require around 10^{29} digits for book-keeping the state of the system over time = 1s! Also from the modelling and simulation perspective, the molecules keep colliding and changing mass and momentum even at equilibrium whose byproduct is statistical noise. These statistical noise are not desired in simulating smooth fluid flows.
There is a middle-way between the macroscopic and microscopic scales called the mesoscopic scale. *This level of description make use of the kinetic theory of gases and statistical mechanics.* They use a probabilistic approach to bridge the atomistic dynamics and continuum fluid dynamics. Considering few assumptions (this will be discussed in detail in the next chapter) kinetic theory suggests that any system can be described by a distribution function $f^N(x^N, p^N, t)$ where x, p, t, N refer to position, momentum, time

and number of particles respectively. The theory argues that instead of tracking each molecules, it is enough to track the probability of finding the particle in that state, to describe the system. The changes in this distribution function with time are given by Liouville equation but the description becomes very complex. As low order distribution function is enough to describe the physics of the system, it can be further simplified. So, taking the first order distribution function (sufficient approximation) it is feasible to describe the system. Thus, mesoscopic scale is a more detailed picture than the macroscopic scale but avoids the details of the microscopic scale. The connection of these distribution functions to the macroscopic quantities will be talked in length, later in the chapter. This level of description is the scale of LBM.
Lets take a practical example to have a clear picture of how these three scales describe. Suppose a law has to be passed from the parliament. The voice of each member of a certain party is synonymous to the ***microscopic scale****. However, the idea of each member of the party is less important than the final statement of the party which corresponds to the invisible resultant effect of the chaotic motion of particles. The final statement of the party is synonymous to the* ***mesoscopic scale*** *which contributes to the law being passed. However, the tangible impact in the society comes from the final statement from the parliament. This is synonymous to the* ***macroscopic scale.***

1.2 Birth of LBM

The development of LBM originates from the lattice gas cellular automaton models first described by Hardy *et al.* [23]. Cellular Automaton required Boolean mathematics in a triangular lattice. The basic idea was that the entity occupied positions on the grid in space and interacted with its specific neighbours to evolve in time by updating its own state. One of the simplest type of its kind is an one-dimensional, 2 state, 2 neighbour model. A detailed classification and analysis of 265 rules for such type of automaton was given by Wolfram [24].
Frish *et al.* [25] extended the lattice gas model to commonly called "FHP model" which solved 2-dimensional Navier-Stokes equation. The basic idea of the model was to satisfy the conservation laws at the microscopic level to be able to describe the real gas. The automaton could be pictured as a lattice site surrounded by hexagonally symmetric six neighbours identified by six connecting vectors $\vec{c_i}$ where $i = 1, ...6$ as shown in figure 1.1.

The properties of the particles placed at the lattice sites are as follows:

- all particles have the same speed and mass.

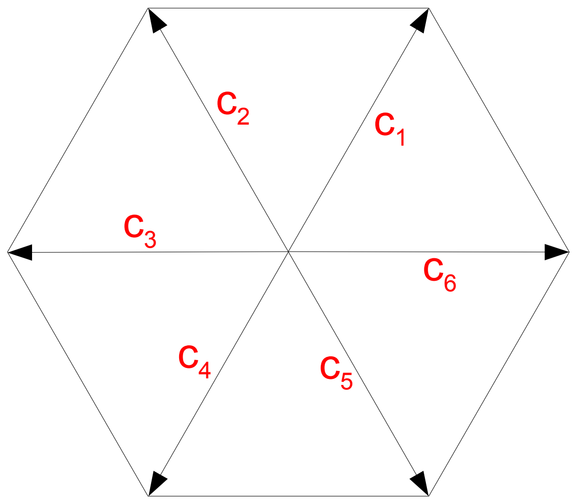


FIGURE 1.1: FHP unit velocity vectors [13]

- no two particles sitting on the same site can move along the same direction (exclusion principle).
- all particles have same energy which means that in a time-cycle, the particles hop strictly to the nearest neighbour only guided by the velocity vector.

Since the FHP model used particles, it inherited the statistical noise. McNamara and Zanetti [26] suggested that statistical noise can be avoided if the particles were replaced by velocity distribution function. Thus after some modifications to the existing problems, this technique of solving Fluid dynamics problem moved to next level with the introduction of Lattice Boltzmann models. The first paper on Lattice Boltzmann method can be credited to McNamara and Zanetti [26].

1.3 Introduction to LBM

It is understood that each system can be described with equations corresponding to either microscopic, mesoscopic or macroscopic scales, they are linked and can be derived one from the other. The framework equation for the LBM is Boltzmann equation based on the kinetic theory. So, the equations of fluid mechanics can be solved by the Boltzmann equation in a different way than the conventional CFD. He and Luo [27] proposed that the continuous BGK (Bhatnagar Gross Kook) [28] Boltzmann equation can be discretized in velocity space to achieve a numerical scheme. This used a first order discretization in time. The method was based on the idea that the velocity distribution function approaches to a equilibrium state using a single relaxation time because of collision of particles. He *et al.* [29] proposed second order discretization

in space and time confirming that the developed LB method is second order accurate. Further developments followed with the models that discretized velocity space using Hermite polynomials and the Gauss-Hermite quadrature to simulate isothermal flows (Shan *et al.* [30]) and also anisothermal flows (Nie *et al.* [31]). These topics will be dealt in detail in the coming chapters. Developments of the method followed in leaps and bounds. Researchers looked into different ways of improving the collision model (like Multiple Relaxation Time (MRT) model [32, 33], entropic model [34, 35] etc.); modifying advection method by incorporating Finite Difference [36–40], Finite Element [41–44], Finite Volume [6–12] techniques; increasing areas of application (*e.g.* multi-phase and multicomponent flows [45–50], flows with suspensions [51–55], emulsions [56], porous media [57–60], natural convection [61], reactive transport [62, 63]; combustion [64, 65], magneto-hydrodynamics [66]); pushing the boundaries to compressible flows [67, 68], high Mach flows [69]; and developing mesh refinement techniques [1–5] (which are inherently uncomfortable with the LB method) etc. In gist, discretization of the Boltzmann equation adopted for lattices called the LB method solved the fluid problem with a much simpler yet powerful approach (but with areas of limitations).
In this introduction, it would be a good preparation to get familiar with the subject by listing some important advantages and disadvantages of LBM compared to conventional CFD which have been repeated by almost all, over time and again.

1.3.1 Advantages

- Navier-Stokes (NS) equations are second-order partial differential equations. Discrete velocity Boltzmann equations are a set of first order partial differential equations.

- The major problem in NS solvers is the existence of the non-linear convective term $(u \cdot \nabla u)$. LB method doesn't have a non-linear convective term and the advection is performed by just shifting the information to the next lattice grid. The non-linearity is hidden in the equilibrium distribution function.

- One of the major advantage of LB method over NS is that it has great advantage in parallelisation. NS equations need to solve Poisson equation for pressure and while doing so has to communicate globally. However, data communication is local for LB method as pressure is found locally and obtained from Equation of State (EOS) equation.

- Although incorporating complex geometry requires careful treatment, it can be proved to be competitive or even faster than NS solver [70, 71]. Normal and shear stress components in NS solvers are non-local so they require proper handling of the geometric estimates and extrapolations. This is not necessary in LB method as normal and shear stress components are local. However, LB method doesn't have a counterpart in the no-slip boundary condition and requires to be estimated.

- LB method can easily deal with multi-component, multi-phase flows without special technique for tracking the interface. It is easy to notice this advantage because LB method is based on kinetic theory accessing to the molecular level and can deal easily with the molecular interactions. So, it is easy to distinguish different phases and different components without requiring any interface tracking technique. For the same reasons, it can be well applied for micro, nano-scale flows.

1.3.2 Disadvantages

- Commonly used LB method is limited to second order accuracy. Higher order accuracy is possible but the method becomes highly complicated.
- LB method is weakly compressible. Incompressible flows are simulated by using LB method within incompressible limit. Development of incompressible LB method is only applicable to simple flows [72].
- Traditionally and popularly, δx and δt are taken equal to 1. This suggested the Courant-Friedrichs-Lewy (CFL) number to be unity. This slows down the speed of convergence of time-dependent LB method for solving steady-state problems. Here speed of convergence is ruled by the acoustic propagation.
- In LB method, there is a link between spatial and velocity discretization. This coupling restricts the traditional LB method to have uniform square grids.

1.4 Theory of LBM

It is also important to look into the mathematical treatment of the equations to understand the underlying details of the method. This section focuses on the derivation of the Lattice Boltzmann equations. It starts from the basics of defining the behaviour of the particles to finally obtain the popularly used LB equation.

1.4.1 Boltzmann equation

As we discussed earlier, the mesoscopic description of fluid is based on the kinetic theory and statistical mechanics. To define the system under consideration, we need to make some assumptions: (*i*) The classical kinetic theory of gases assumes dilute gas with N molecules in a box of volume V. (*ii*) The *average de Broglie wavelength* of the molecule is considered fairly smaller than the average intermolecular distance. This allows to neglect the quantum behaviour of the system and follow the classical behaviour: Each molecule can be considered a particle which has defined position and momentum. (*iii*) The molecules are assumed to be of one kind and mono-atomic. (*iv*) The walls of the box containing the gas under consideration act as idealised surfaces which reflect elastically.

Now in such a system we look for a descriptive object of kinetic theory, the distribution function $f(\mathbf{x}, \mathbf{p}, t)$. This represents the density of particles with position vector $\mathbf{x}$ and momentum vector $\mathbf{p}$ at time t in a phase space. This phase space (also called μ space) is a 6 dimensional space with coordinate axes $\mathbf{x}$ (or d^3x) and $\mathbf{p}$ (or d^3p) (figure 1.2).

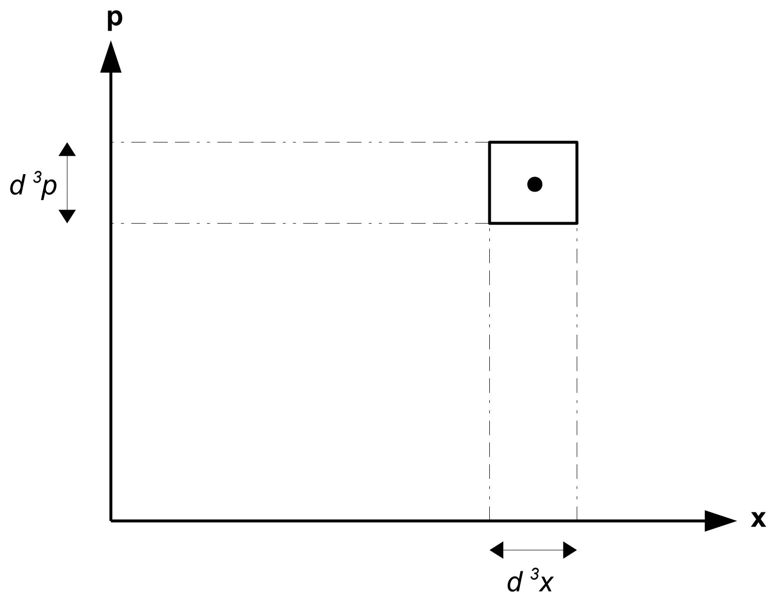


FIGURE 1.2: The 6-dimensional phase space of particles with coordinate axes $\mathbf{x}$ and $\mathbf{p}$ [14].

A point in this space would mean a state of a particle. Suppose we take a finite cubelet $\partial^3 x \partial^3 p$ about position vector $\mathbf{x}$ and momentum vector $\mathbf{p}$. The number of points in the cubelet (also the number of particles) is given by the distribution function $f(\mathbf{x}, \mathbf{p}, t)$. Now, if we consider large cubelet in μ phase space which contains large number of particles (say N) and the number of the particles doesn't vary rapidly within the cubelet then we can assume the distribution function $f(\mathbf{x}, \mathbf{p}, t)$ to be continuous. This implies,

$$\sum f(\mathbf{x}, \mathbf{p}, t) \partial^3 x \partial^3 p \approx \int f(\mathbf{x}, \mathbf{p}, t) \partial^3 x \partial^3 p = N \tag{1.4}$$

Moreover, if the molecules are uniformly distributed over volume V the distribution function f can be considered independent of $\mathbf{x}$. So,

$$\int f(\mathbf{x},\mathbf{p},t)\partial^3 p = \frac{N}{V}$$

or,

$$\rho(x,t) = \int f(\mathbf{x},\mathbf{p},t)\; d\mathbf{p} = m\int f(\mathbf{x},\mathbf{c},t)\; d\mathbf{c} \tag{1.5}$$

where, m = mass of the particle; $\mathbf{c}$ = velocity of the particle and momentum $\mathbf{p} = m\mathbf{c}$ Equation 1.5 gives the link between the mesoscopic quantity f with the macroscopic quantity- fluid density(ρ). Other macroscopic quantities like fluid velocity and internal energy can also be found as moments by weighting the distribution function f with some function of $\mathbf{c}$ and integrated over entire momentum space.
By weighting with $\mathbf{c}$ and integrating we obtain the momentum density of the fluid ($\rho\mathbf{u}(x,t)$ where $\mathbf{u}$ stands for fluid velocity),

$$\rho\mathbf{u}(x,t) = \int \mathbf{c}\; f(\mathbf{x},\mathbf{p},t)\; d\mathbf{p} = m\int \mathbf{c}\; f(\mathbf{x},\mathbf{c},t)\; d\mathbf{c} \tag{1.6}$$

By weighting with $\frac{1}{2}|\mathbf{c}|^2$ and integrating, we achieve energy density ($\rho E(x,t)$ where E stands for specific energy),

$$\rho E(x,t) = \frac{1}{2}\int |\mathbf{c}|^2\; f(\mathbf{x},\mathbf{p},t)\; d\mathbf{p} = m\,\frac{1}{2}\int |\mathbf{c}|^2\; f(\mathbf{x},\mathbf{c},t)\; d\mathbf{c} \tag{1.7}$$

It is important to note that we had assumed that the gas is monoatomic and the kinetic energy doesn't possess rotational and vibrational movements of the particles. Thus the total kinetic energy (ρE) is due to two things: kinetic energy density due to bulk movement of fluid ($\frac{1}{2}\rho|\boldsymbol{u}|^2$) + internal energy density due to random thermal movement of particles (ρe). Also, another important parameter is the peculiar velocity ($\boldsymbol{v}$) = particle velocity ($\boldsymbol{c}$) - bulk fluid velocity ($\boldsymbol{u}$). Therefore it can also be shown that,

$$\rho e(x,t) = m\,\frac{1}{2}\int |\boldsymbol{v}|^2\; f(\boldsymbol{x},\mathbf{c},t)\; d\boldsymbol{c} \tag{1.8}$$

Now, we are interested in finding the *equation of motion* of the distribution function f. It can be understood that the number of molecules will keep varying in the μ space with increase in time. Some particles would enter while some would leave the space. Suppose the particles are exposed to an external force F and there are no intermolecular collisions. Then a molecule with coordinates $(\mathbf{x},\mathbf{p})$ at time t would transform to new coordinates $(\mathbf{x}+\mathbf{c}\delta t, \mathbf{p}+\mathbf{F}\delta t)$ after very small time interval δt. This means all the particles that were in μ-space element $\partial^3 x\partial^3 p$ at coordinates $(\mathbf{x},\mathbf{p})$ at time t will exactly be found in μ-space element $\partial^3 x^{'}\partial^3 p^{'}$ at coordinates $(\mathbf{x}+\mathbf{c}\delta t, \mathbf{p}+\mathbf{xF}\delta t)$ at time $t+\delta t$ when there are no intermolecular collisions.

$$f(\mathbf{x}+\mathbf{c}\delta t, \mathbf{p}+\mathbf{F}\delta t, t+\delta t)\partial^3 x^{'}\partial^3 c^{'} = f(\mathbf{x},\mathbf{p},t)\partial^3 x\partial^3 c$$

Here, since the μ-space element $\partial^3 x^{'}\partial^3 p^{'}$ is equal to $\partial^3 x\partial^3 p$,

$$f(\mathbf{x}+\mathbf{c}\delta t, \mathbf{p}+\mathbf{F}\delta t, t+\delta t) = f(\mathbf{x},\mathbf{p},t) \tag{1.9}$$

Now if we consider intermolecular collisions, equation 1.9 needs to be complemented by a collision term,

$$f(\mathbf{x}+\mathbf{c}\delta t, \mathbf{p}+\mathbf{F}\delta t, t+\delta t) = f(\mathbf{x},\mathbf{p},t) + \delta t\left(\frac{\partial f}{\partial t}\right)_{collision} \tag{1.10}$$

Expanding the L.H.S. to first order terms in δt,

$$f(\mathbf{x}+\mathbf{c}\delta t, \mathbf{p}+\mathbf{F}\delta t, t+\delta t) = f(\mathbf{x},\mathbf{p},t) + \mathbf{c}\delta t\cdot\nabla_x f + \mathbf{F}\delta t\cdot\nabla_p f + \left(\frac{\partial f}{\partial t}\right)\delta t \tag{1.11}$$

Using equation 1.11 in equation 1.10 we get,

$$\left(\frac{\partial}{\partial t} + \mathbf{c}\cdot\nabla_x + \mathbf{F}\cdot\nabla_p\right)f(\mathbf{x},\mathbf{p},t) = \left(\frac{\partial f}{\partial t}\right)_{collision} \tag{1.12}$$

This is the *kinetic equation* for the one-body distribution function. However, the collision term involves two-body distribution function f_{12} which means finding molecule 1 around position vector x_1, momentum vector p_1 and molecule 2 around position vector x_2, momentum vector p_2 at same time t. The problem here is that to write the dynamics equation of f_{12}, we need to call a function f_{123} and so on. This is known as *BBGKY hierarchy*. We see that the equation is not closed. To close the equation, Boltzmann made following important assumptions:

- Gas is assumed very dilute with molecules interacting via elastic binary collisions. This allows a simple collision process. The elementary μ space consists of only two processes: gain of molecules(inverse collision) or loss of molecules(direct collision). This can be represented by (refer to [73]),

$$\left(\frac{\partial f}{\partial t}\right)_{collision} = gain - loss = \int (f_{1'2'} - f_{12})\,\bar{v}\,\sigma\, d\Omega\, d\vec{p_2} \tag{1.13}$$

  where, Ω = solid angle
  σ = differential cross section or the number of molecules scattered per second into the solid angle element dΩ
  $\bar{v}$ = relative speed of the molecules

$f_{1'2'} - f_{12}$ = difference in distribution function after collision. Apostrophe represents the state after collision.

- The next assumption is that the molecules only have correlations before the collision and not while at collision. This means two molecules have a very low chance of meeting with each other in their life span. This is also called *molecular chaos*.

$$f_{12} = f_1 \, f_2 \tag{1.14}$$

After closing the equation 1.12 with these assumptions, we arrive to the continuous Boltzmann equation:

$$\left(\frac{\partial}{\partial t} + \mathbf{c} \cdot \nabla_x + \mathbf{F} \cdot \nabla_p\right) f(\mathbf{x}, \mathbf{p}, t) = \int (f_{1'} \, f_{2'} - f_1 \, f_2) \, \bar{v} \, \sigma \, d\Omega \, d\vec{p_2} = \Omega(f) \tag{1.15}$$

The Boltzmann equation says some interesting things. In the equation, velocity of molecules **c** doesn't depend on f so the advection is linear. However the collision term is non-linear in regard to f but it is local. Apart from this, the Boltzmann equation looks similar to an advection equation with a source term Ω. It would have been easy to solve Ω if it was explicitly known but it is a function of f. Fortunately, two-body collisions doesn't affect significantly the measurement of many measured quantities [74]. This allows to simply approximate the collision operator without affecting much the solution as long as it captures the macroscopic behaviour and conserves the mass, momentum and energy.

$$\int \Omega(f) d\mathbf{c} = 0 \tag{1.16}$$

$$\int \mathbf{c} \, \Omega(f) d\mathbf{c} = 0 \tag{1.17}$$

$$\int |\mathbf{c}|^2 \, \Omega(f) d\mathbf{c} = 0 \tag{1.18}$$

The simplest and most popular model is a Bhatnagar-Gross-Krook (BGK) model [28] which uses single relaxation time (τ) to model the relaxation process instead of following the details of the collisions. This simplification replaces the collision operator with

$$\Omega(f) = -\frac{1}{\tau}(f - f^0) \tag{1.19}$$

where τ = single relaxation time. This is the time taken by the system to relax towards an equilibrium. Also, f^0 = equilibrium distribution function.
Finally, the continuous BGK Boltzmann equation is

$$\boxed{\left(\frac{\partial}{\partial t} + \mathbf{c} \cdot \nabla_x + \mathbf{F} \cdot \nabla_p\right) f = -\frac{1}{\tau}(f - f^0)} \tag{1.20}$$

1.4.2 Equilibrium

Boltzmann BGK equation can't be analytically solved. However we can make further assumptions to the system so that the equation can be reduced enough to be solved. One of such achievable solution is equilibrium distribution function. The assumptions made are as follows:

- There exists no external force. This allows distribution function f to be independent of $\mathbf{x}$. The distribution function reduces its dependency to $f(\mathbf{p}, t)$.
- After the initial conditions, the system is left for very long (limit of time tends to infinity). Then the system tends towards equilibrium. This means the differentiation of distribution function with respect to time is 0. *i.e.*

  $$\frac{\partial f(\mathbf{p}, t)}{\partial t} = 0 \tag{1.21}$$

  This immediately gives a physical picture of the system. At equilibrium, the collision term is null ($\Omega(f) = 0$). This means the direct and inverse collisions are dynamically balanced to have gain and loss of the molecules in exact balance. Equilibrium arranges all the directions of velocities of particles evenly around the fluid velocity.
  The solution of equation 1.21 is termed as equilibrium distribution function $f^0(\mathbf{p})$. *(At equilibrium, the velocities of particles tend to arrange evenly in all directions around fluid velocity* $\boldsymbol{u}$ *such that all directions of peculiar velocity (*$\boldsymbol{v}$ *= particle velocity - fluid velocity) are probable. Thus, distribution function is only dependent on* $\boldsymbol{v}$ *so* $f^0(\boldsymbol{v})$ *is more relevant.* $\boldsymbol{v}$ *can be geometrically visualized as a radius of a sphere around fluid velocity* $\boldsymbol{u}$ *as seen in figure 1.3.)*

As we have already assumed distribution function f to be independent of $\mathbf{x}$ we can write,

$$f^0(|\mathbf{v}|) = \phi(\mathbf{v}^2) = \phi(v_x^2 + v_y^2 + v_z^2)$$

Since distribution function needs to be positive, the velocity is squared to have positive magnitude. From the above equation, ϕ is an unknown function that needs to be determined and the most suitable function for this is an exponential function. Therefore,

$$f^0(|\mathbf{v}|) = A\, e^{-B|\mathbf{v}|^2} \tag{1.22}$$

The variables A and B can be determined by using moments of f^0.
Taking moment of density,

$$\rho = \int f^0(|\mathbf{v}|)\, d\mathbf{c} \tag{1.23}$$

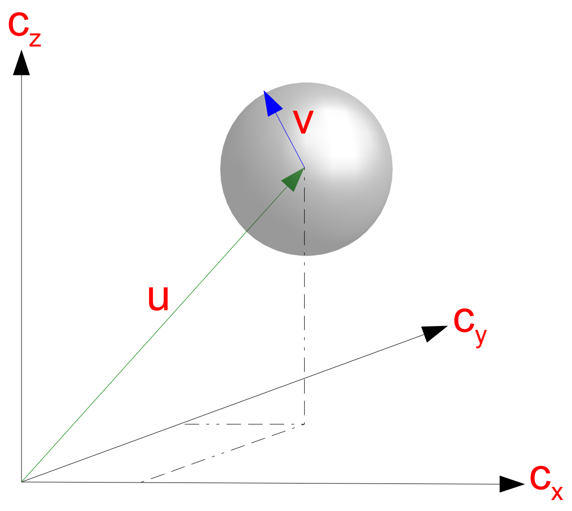


FIGURE 1.3: Spherically iso-surface of a distribution function f which is spherically symmetric around $\mathbf{c} = \mathbf{u}$ [15]

Since $f^0(|\mathbf{v}|)$ is spherically symmetric around $\mathbf{v} = 0$ (refer figure 1.3, the volume of the spherical shell of distance d$|\mathbf{v}|$ is $4\pi|\mathbf{v}|^2 d|\mathbf{v}|$. Thus,

$$f^0(|\mathbf{v}|)\, d\mathbf{c} = A\, e^{-B|\mathbf{v}|^2} 4\pi|\mathbf{v}|^2 d|\mathbf{v}| \tag{1.24}$$

Using equation 1.23 in equation 1.24 we get,

$$\rho = \int_0^\infty A\, e^{-B|\mathbf{v}|^2} 4\pi|\mathbf{v}|^2 d|\mathbf{v}|$$

$$\rho = 4\,\pi\,A \int_0^\infty |\mathbf{v}|^2 e^{-B|\mathbf{v}|^2} d|\mathbf{v}|$$

Since $\int_{-\infty}^{\infty} x^2 e^{-a\,x^2} dx = \frac{1}{2}\sqrt{\frac{\pi}{a^3}}$,

$$\rho = A \left(\frac{\pi}{B}\right)^{3/2} \tag{1.25}$$

and

$$A = \rho \left(\frac{B}{\pi}\right)^{3/2}$$

Using moment of energy,

$$\rho e = \frac{1}{2} \int |\mathbf{v}|^2 f^0(|\mathbf{v}|)\, d\mathbf{c} \tag{1.26}$$

$$\rho e = \frac{1}{2} \int_0^\infty |\mathbf{v}|^2\, A\, e^{-B|\mathbf{v}|^2} 4\pi|\mathbf{v}|^2 d|\mathbf{v}|$$

$$\rho e = \frac{1}{2}\, 4\,\pi\,A \int_0^\infty |\mathbf{v}|^4\, e^{-B|\mathbf{v}|^2} d|\mathbf{v}|$$

Since $\int_0^\infty x^n e^{-a\,x^b} dx = \frac{1}{b}\, a^{\frac{-n+1}{b}}\, \Gamma\left(\frac{n+1}{b}\right), \qquad [\Gamma(n) = (n-1)!]$

$$\rho e = \frac{1}{2}\, 4\,\pi\, A\, \frac{1}{2}\, B^{(-\frac{3}{2}-1)}\, 3\, \frac{\sqrt{\pi}}{4}$$

$$\rho e = A \left(\frac{\pi}{B}\right)^{3/2} \frac{1}{2}\,\frac{1}{2}\, 4\, B^{-1}\, \frac{3}{4}$$

Using equation 1.25 we arrive at,

$$\rho e = \frac{3}{4}\,\frac{\rho}{B} \tag{1.27}$$

From equation of state relating pressure to internal energy for monatomic gas, $p = \frac{2}{3}\rho e$. *From ideal gas law,* $p = \rho\, R\, T = \rho \frac{k_B T}{m}$ *where* k_B *and* T *are Boltzmann's constant and temperature respectively. Equating these two expression, we get* $e = \frac{3}{2}\frac{p}{\rho} = \frac{3}{2}\frac{k_B T}{m}$

Now we can identify B,

$$B = \frac{3}{4e} = \frac{m}{2\, k_B\, T}$$

Applying A and B in equation 1.22,

$$\boxed{f^0(|\mathbf{v}|) = \rho \left(\frac{m}{2\,\pi\, k_B\, T}\right)^{3/2} e^{-\frac{m|\mathbf{v}|^2}{2\,k_B\,T}}} \tag{1.28}$$

This is the Maxwell-Boltzmann distribution function. This state has high significance in the kinetic theory. The universality of this equilibrium distribution function can be explained by saying that if a dilute gas with an arbitrary initial condition is allowed with molecular interactions to move from one state to another state, the gas in course of time will finally approach to equilibrium distribution function while satisfying macroscopic conditions.

1.4.3 Towards discrete-velocity Boltzmann equation

We have already said that the analytical solution of Boltzmann equation is very complicated. So to solve fluid dynamics problems, full Boltzmann equation can only be used if it is discretized and solved numerically. At this point it is important to note that conventional CFD usually requires only discretization of space and time. In Boltzmann equation we have seen that the distribution function $f(\mathbf{x}, \mathbf{c}, t)$ is not only dependent on space and time but also velocity. So discretization of Boltzmann equation follows two steps. First, the equation is discretized in velocity space, then it is discretized in physical space and time.

Physically, discretization in velocity space would mean that the continuous particle velocity space is replaced by a discrete set of velocity vectors, fulfiling certain rules. Also, the aim is to keep the set of the discrete velocities as small as possible.

To start the velocity discretization step, we assume the simplest condition of the system: (i) ideal gas (ii) isothermal. The other conditions of the system can be further added to the discretized model to represent corresponding complex fluid flow.
The general method for velocity discretization is using Hermite polynomials and Gauss-Hermite quadrature [75, 76]. The order of quadrature determines the number of velocity vectors required in the velocity set. It should be a good compromise for the choice of order of quadrature. Sufficiently higher order is required to express the behaviour of Boltzmann equation but it should be taken care that higher order demands higher number of velocity set which further demands high computational resources.
Here we follow a simple but old method for mathematical simplicity. The discretization starts with approximating Maxwell-Boltzmann distribution by expanding up to $O(\mathbf{u}^2)$. It can be proved that expanding only up to second order is enough to behave similarly like the Navier-Stokes equations.

$$f^0 = \rho \left(\frac{m}{2\,\pi\, k_B\, T} \right)^{3/2} e^{-\frac{m|\mathbf{v}|^2}{2\, k_B\, T}}$$

$$f^0 = \frac{\rho}{(2\,\pi c_0^2)^{3/2}}\ e^{\frac{-(|\mathbf{c}|-|\mathbf{u}|)^2}{2c_0^2}}$$

(From equation of state $c_0^2 = \left(\frac{\partial p}{\partial \rho}\right) = \frac{k_B T}{m}$ and $\mathbf{c} = \mathbf{v} + \mathbf{u}$)

$$f^0 = \frac{\rho}{(2\,\pi c_0^2)^{3/2}}\ e^{\frac{-(c_\alpha c_\alpha - 2c_\alpha u_\alpha + u_\alpha u_\alpha)}{2c_0^2}}$$

After using the exponential series expanded up to $O(\mathbf{x}^2)$, $e^x = 1 + x + \frac{x^2}{2!}$ we get

$$f^0 = \frac{\rho}{(2\,\pi c_0^2)^{3/2}}\ e^{\frac{-c_\alpha c_\alpha}{2c_0^2}} \left(1 + \frac{c_\alpha u_\alpha}{c_0^2} + \frac{(c_\alpha u_\alpha)^2}{2c_0^4} - \frac{u_\alpha u_\alpha}{2c_0^2} \right)$$

Now, the random particle velocity set of the above equilibrium distribution function which have been reduced to second order of fluid velocity $\mathbf{u}$ will be converted to a discrete set of velocities c_i (i represent the number of populations describing the velocity set). Also, the coefficient $e^{-c_\alpha c_\alpha / 2c_o^2}/(2\,\pi c_o^2)^{3/2}$ is replaced by a single weighting coefficient w_i which defines the weights of each discrete velocity in the chosen velocity set. Finally we arrive to the discrete equilibrium distribution function [77],

$$\boxed{f_i^0 = \rho w_i \left(1 + \frac{c_{i\alpha} u_\alpha}{c_0^2} + \frac{c_{i\alpha} u_\alpha c_{i\beta} u_\beta}{2c_0^4} - \frac{u_\alpha u_\alpha}{2c_0^2} \right)} \tag{1.29}$$

Similarly, the distribution function $f(\mathbf{x}, \mathbf{c}, t)$ is also discretized in velocity space. The dependency of f with velocity $\mathbf{c}$ is removed and $f(\mathbf{x}, \mathbf{c}, t)$ becomes $f_i(\mathbf{x}, t)$ where the density is represented at $(\mathbf{x}, t)$ with set of velocities c_i. Using this velocity discretization, equation 1.20 turns into discrete-velocity Boltzmann equation (DVBE) with all external forcing grouped into $F_{i\alpha}$ term ,

$$\boxed{\frac{\partial f_i}{\partial t} + c_{i\alpha} \cdot \frac{\partial f_i}{\partial x_\alpha} = -\frac{1}{\tau}\left(f_i - f_i^0\right) + F_{i\alpha}} \tag{1.30}$$

1.4.3.1 Moments and constraints for the discrete velocity set

As we said earlier that continuous particle velocity set must fulfil certain conditions to convert to finite discrete velocities set . The choice of the velocity set must be done in such a way that it conserves mass and momentum. It is known that moments f^0 are equal to that of f satisfying conservation laws. *We assume that the zeroth to third moments of f_i^0 and f^0 are equal (Note: the equality should be only up to third moment because we have truncated f_i^0 to $O(\boldsymbol{u}^2)$).* These equalities will provide constraints on the velocity set defining c_i and w_i. The equalities are as follows:

Zeroth moment

Zeroth moment of f^0 (Π_0^0) is equal to zeroth moment of f_i^0,

$$\Pi_0^0 = \rho(\mathbf{x}, t) = \sum_i f_i^0(\mathbf{x}, t) \tag{1.31}$$

Expanding f_i^0 from equation 1.29 we get,

$$\rho = \rho\left[\sum_i w_i + \frac{u_\alpha}{c_0^2}\sum_i w_i c_{i\alpha} + \frac{u_\alpha u_\beta}{2c_0^2}\left(\frac{1}{c_0^2}\sum_i w_i c_{i\alpha} c_{i\beta} - \delta_{\alpha\beta}\sum_i w_i\right)\right]$$

For both sides to be equal, everything inside square brackets should be equal to 1. This can be achieved only if,

$$\boxed{\sum_i w_i = 1} \tag{1.32}$$

$$\boxed{\sum_i w_i c_{i\alpha} = 0} \tag{1.33}$$

$$\boxed{\sum_i w_i c_{i\alpha} c_{i\beta} = c_0^2 \delta_{\alpha\beta}} \tag{1.34}$$

Here we established 3 constraints of c_i and w_i. We need some more and are further given by next equality,

First moment

First moment of f^0 (Π^0_α) is equal to first moment of f_i^0,

$$\Pi^0_\alpha = \rho \mathbf{u}(\mathbf{x},t) = \sum_i c_{i\alpha} f_i^0(\mathbf{x},t) \tag{1.35}$$

$$\rho u_\alpha = \rho \left[\sum_i w_i c_{i\alpha} + \frac{u_\alpha}{c_0^2} \sum_i w_i c_{i\alpha} c_{i\alpha} + \frac{u_\alpha u_\beta}{2c_0^4} \sum_i w_i c_{i\alpha} c_{i\beta} c_{i\alpha} - \frac{u_\alpha u_\alpha}{2c_0^2} \sum_i w_i c_{i\alpha} \right]$$

$$\rho u_\alpha = \rho \left[\sum_i w_i c_{i\alpha} + \frac{\delta_{\alpha\beta}\, u_\alpha}{c_0^2} \sum_i w_i c_{i\alpha} c_{i\beta} + \frac{\delta_{\alpha\gamma}\, u_\alpha u_\beta}{2c_0^4} \sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} - \frac{u_\alpha u_\alpha}{2c_0^2} \sum_i w_i c_{i\alpha} \right]$$

Since $\sum_i w_i c_{i\alpha} = 0$ and $\sum_i w_i c_{i\alpha} c_{i\beta} = c_0^2 \delta_{\alpha\beta}$, the equation reduces to,

$$\rho u_\alpha = \rho \left[\frac{\delta_{\alpha\beta}\, u_\alpha}{c_0^2} c_0^2 \delta_{\alpha\beta} + \frac{\delta_{\alpha\gamma}\, u_\alpha u_\beta}{2c_0^4} \sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} \right]$$

$$\rho u_\alpha = \rho \left[u_\alpha + \frac{\delta_{\alpha\gamma}\, u_\alpha u_\beta}{2c_0^4} \sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} \right]$$

To hold the equality true, it must be

$$\boxed{\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} = 0} \tag{1.36}$$

Second moment

Second moment of f^0 ($\Pi^0_{\alpha\beta}$) is equalled to second moment of f_i^0,

$$\Pi^0_{\alpha\beta}(\mathbf{x},t) = \sum_i c_{i\alpha} c_{i\beta} f_i^0(\mathbf{x},t) \tag{1.37}$$

Before finding the next constraint, it is important to identify $\Pi^0_{\alpha\beta}$ in terms of macroscopic quantities. So, taking second moment of continuous function f^0,

$$\Pi^0_{\alpha\beta} = \int c_\alpha c_\beta f^0 \, d\mathbf{c}$$

Dividing c into its fluid velocity (u) and peculiar velocity (v) components,

$$\Pi^0_{\alpha\beta} = \int (u_\alpha + v_\alpha)(u_\beta + v_\beta) f^0 \, d\mathbf{c} = \int (u_\alpha u_\beta + u_\alpha v_\beta + v_\alpha u_\beta + v_\alpha v_\beta) f^0 \, d\mathbf{c}$$

It is a fact that odd-order peculiar velocity moments are zero (refer[15]). Therefore,

$$\Pi^0_{\alpha\beta} = \int u_\alpha u_\beta \, f^0 \, d\mathbf{c} + \int v_\alpha v_\beta \, f^0 \, d\mathbf{c} = u_\alpha u_\beta \int \, f^0 \, d\mathbf{c} + \delta_{\alpha\beta} \int v_\alpha^2 \, f^0 \, d\mathbf{c}$$

Taking zeroth order of $f^0 = \rho$ and using equations 1.26 and 1.27 and $c_0^2 = \frac{k_B T}{m}$ we arrive at,

$$\Pi^0_{\alpha\beta} = \rho u_\alpha u_\beta + \rho c_0^2 \delta_{\alpha\beta}$$

This gives,

$$\Pi^0_{\alpha\beta} = \rho u_\alpha u_\beta + \rho c_0^2 \delta_{\alpha\beta} = \sum_i c_{i\alpha} c_{i\beta} f_i^0 \tag{1.38}$$

$$\rho u_\alpha u_\beta + \rho c_0^2 \delta_{\alpha\beta} = \rho\left[\left(1 - \frac{u_\gamma u_\gamma}{2c_0^2}\right)\sum_i w_i c_{i\alpha} c_{i\beta} + \frac{u_\gamma}{c_0^2}\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} + \frac{u_\gamma u_\delta}{2c_0^4}\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta}\right]$$

Using equations 1.34 and 1.36, the above equation reduces to

$$\rho u_\alpha u_\beta + \rho c_0^2 \delta_{\alpha\beta} = \rho\left[\left(1 - \frac{u_\gamma u_\gamma}{2c_0^2}\right) c_0^2 \delta_{\alpha\beta} + \frac{u_\gamma u_\delta}{2c_0^4}\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta}\right]$$

$$\rho u_\alpha u_\beta + \rho c_0^2 \delta_{\alpha\beta} = \rho\left[c_0^2 \delta_{\alpha\beta} - \frac{u_\gamma u_\gamma}{2} c_0^2 \delta_{\alpha\beta} + \frac{u_\gamma u_\delta}{2c_0^4}\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta}\right]$$

For this equality to hold true, it must satisfy

$$u_\alpha u_\beta = \frac{u_\gamma u_\delta}{2c_0^4}\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} - \frac{u_\gamma u_\gamma}{2} c_0^2 \delta_{\alpha\beta}$$

$$\frac{u_\gamma u_\delta}{2c_0^4}\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} = u_\alpha u_\beta + \frac{u_\gamma u_\gamma}{2} c_0^2 \delta_{\alpha\beta}$$

$$\frac{u_\gamma u_\gamma}{2c_0^4 u_{\gamma\delta}}\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} = u_\gamma u_\gamma\left(\frac{1}{\delta_{\gamma\alpha}\delta_{\gamma\beta}} + \frac{c_0^2 \delta_{\alpha\beta}}{2}\right)$$

After applying properties of Kronecker delta and some simple algebra we arrive at,

$$\boxed{\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} = c_0^4(\delta_{\alpha\beta}\delta_{\gamma\delta} + \delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma})} \tag{1.39}$$

Third moment

Third moment of f^0 ($\Pi^0_{\alpha\beta\gamma}$) is equalled to third moment of f_i^0,

$$\Pi^0_{\alpha\beta\gamma}(\mathbf{x}, t) = \sum_i c_{i\alpha} c_{i\beta} c_{i\gamma} f_i^0(\mathbf{x}, t) \tag{1.40}$$

Again it is necessary to know the macroscopic equivalent of $\Pi^0_{\alpha\beta\gamma}$. So, to find it we take third moment of continuous function f^0,

$$\Pi^0_{\alpha\beta\gamma} = \int c_{i\alpha} c_{i\beta} c_{i\gamma} f_i^0$$

$$\Pi^0_{\alpha\beta\gamma} = \int (u_\alpha + v_\alpha)(u_\beta + v_\beta)(u_\gamma + v_\gamma) f^0 \, d\mathbf{c}$$

The odd-order peculiar velocity moments are zero therefore the equation reduces to,

$$\Pi^0_{\alpha\beta\gamma} = \int (u_\alpha u_\beta u_\gamma + u_\alpha v_\beta v_\gamma + u_\beta v_\alpha v_\gamma + u_\gamma v_\alpha v_\beta) f^0 \; d\mathbf{c}$$

Taking zeroth order of $f^0 = \rho$ and using equations 1.26 and 1.27 and $c_0^2 = \frac{k_B T}{m}$ we arrive at,

$$\Pi^0_{\alpha\beta\gamma} = \rho u_\alpha u_\beta u_\gamma + \rho c_0^2 (u_\alpha \delta_{\beta\gamma} + u_\beta \delta_{\alpha\gamma} + u_\gamma \delta_{\alpha\beta})$$

If we look back to equation 1.29, we said that we truncated f_i^0 up to $O(\mathbf{u}^2)$. However, in the above equation the first term on the RHS $u_\alpha u_\beta u_\gamma$ is of $O(\mathbf{u}^3)$ and definitely can't be reproduced in the discrete case. Therefor this term is dropped in the derivation. The new equality after dropping the $O(\mathbf{u}^3)$ term is

$$\Pi^0_{\alpha\beta\gamma} = \rho c_0^2 (u_\alpha \delta_{\beta\gamma} + u_\beta \delta_{\alpha\gamma} + u_\gamma \delta_{\alpha\beta}) \tag{1.41}$$

Now, equation 1.40 can be re-written as

$$\rho c_0^2 (u_\alpha \delta_{\beta\gamma} + u_\beta \delta_{\alpha\gamma} + u_\gamma \delta_{\alpha\beta}) = \sum_i c_{i\alpha} c_{i\beta} c_{i\gamma} f_i^0 = \rho \sum_i c_{i\alpha} c_{i\beta} c_{i\gamma} \left(1 + \frac{c_{i\alpha} u_\alpha}{c_0^2} + \frac{c_{i\alpha} u_\alpha c_{i\beta} u_\beta}{2 c_0^4} - \frac{u_\alpha u_\alpha}{2 c_0^2} \right)$$

Using constraints given by equations 1.36 and 1.39, the next constraint is found

$$\boxed{\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} c_{i\epsilon} = 0} \tag{1.42}$$

Finally, we have achieved all the required constraints of c_i and w_i for the DVBE that satisfies mass and momentum conservation laws when the polynomial in f_i is truncated up to $O(\mathbf{u}^2)$. It can be seen from the constraints that the odd moments of f_i^0 vanish and the even moments are isotropic tensors. We emphasize that the system was considered isothermal and ideal gas. If we need to look into more complex system, the velocity set gets bigger and eventually the constraints increase.

1.4.3.2 Lattice arrangements

Now using the constraints just derived, we would like to produce specific velocity set. Although currently the more common method for this is Gauss-Hermite quadrature method [75, 76], we adopt an older but mathematically simpler approach [77] which was prevalent in the early years of LBM. The method is built on prescribed velocity vectors using which weighting coefficients are developed fulfiling the conditions that satisfy conservation laws.

In LBM, lattice arrangement is commonly described in the form of $D_n Q_p$. This system of classification of LBM was proposed by Qian *et al.* [77]. Here, n represents the dimension of the arrangement and p represents the number of populations. Lets take an example of D_2Q_9 lattice arrangement. D_2Q_9 means the model is 2 dimensional with 9 populations involved. D_2Q_9 is a square lattice as shown in figure 1.4 and most popularly used for 2D simulations. In a lattice, the populations (velocity vectors) defined

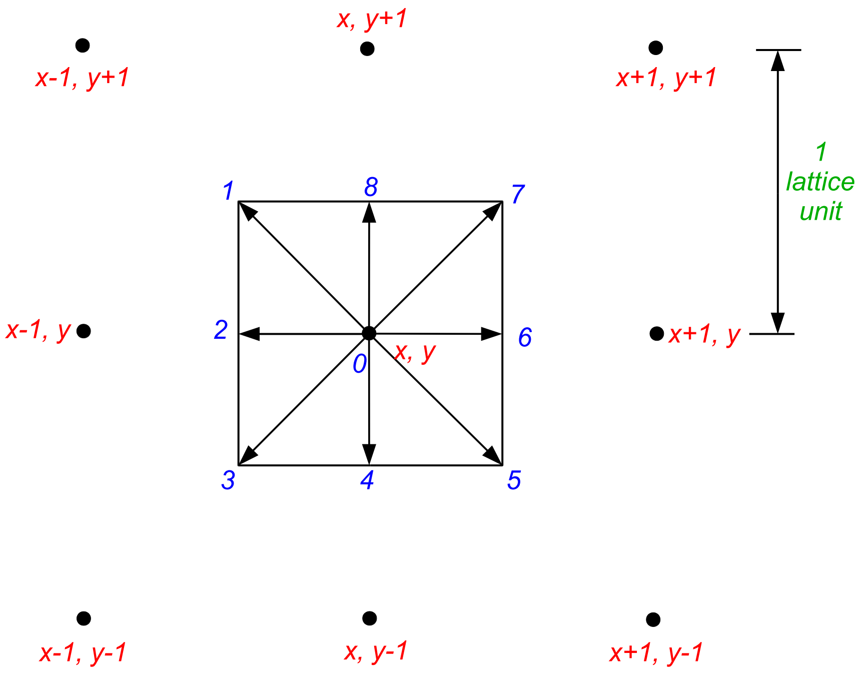


FIGURE 1.4: D_2Q_9 lattice arrangement with lattice spacing $\triangle\mathbf{x} = 1$

to connect to the neighbouring lattice points are of three types in terms of the velocity directions/magnitudes and correspondingly weighting coefficients. The nine discrete velocities are defined by

$$\mathbf{c}_i = \begin{cases} (0,0), & i=0, \\ \Big(cos(\frac{i-1}{2}\pi), sin(\frac{i-1}{2}\pi)\Big)\frac{\triangle x}{\triangle t}, & i=6,8,2,4, \\ \sqrt{2}\Big(cos(\frac{i-5}{2}+\frac{1}{4})\pi, sin(\frac{i-5}{2}+\frac{1}{4})\pi\Big)\frac{\triangle x}{\triangle t}, & i=7,1,3,5 \end{cases}$$

Directions of the velocity vectors are defined by the method of characteristics that will be explained in the next section. At the center, the velocity vector is zero (w_0). This component of the populations is assumed at rest. The velocity vectors parallel to the Cartesian x-y axes are termed *short* velocity vectors ($w_6 = w_8 = w_2 = w_4 = w_s$) and the magnitude of the velocity is $\frac{\triangle x}{\triangle t}$. $\triangle x$ and $\triangle t$ are lattice spacing for space and time. The ones directed diagonally are termed *long* velocity vectors ($w_7 = w_1 = w_3 = w_5 = w_l$). The velocity magnitude of these populations is $\sqrt{2}\,\frac{\triangle x}{\triangle t}$.
Now we can use the constraints described in the previous section to build specific velocity set for this arrangement.

Using constraint 1.32,

$$\sum_i w_i = 1 = w_0 + 4\, w_s + 4\, w_l \tag{1.43}$$

Using constraint 1.34,

$$\sum_i w_i c_{ix}^2 = \sum_i w_i c_{iy}^2 = c_0^2 = \left(\frac{\triangle x}{\triangle t}\right)^2 (2\, w_s + 4\, w_l) \tag{1.44}$$

Using constraint 1.39,

$$\sum_i w_i c_{ix}^2 c_{iy}^2 = c_0^4 = \left(\frac{\triangle x}{\triangle t}\right)^4 (4\, w_l) \tag{1.45}$$

and

$$\sum_i w_i c_{ix}^4 = \sum_i w_i c_{iy}^4 = 3\, c_0^4 = \left(\frac{\triangle x}{\triangle t}\right)^4 (2\, w_s + 4\, w_l) \tag{1.46}$$

Dividing equation 1.46 by 1.44 we get $c_0 = (\frac{\triangle x}{\triangle t})/\sqrt{3}$ where c_0 stands for the so-called lattice speed of sound, whose value depends on the specific velocity lattice topology. Using the value of c_0 in equation 1.45 and 1.44, we derive the values of w_l and w_s respectively. Finally, using the values of c_0, w_l and w_s in equation 1.43, we achieve the value for w_0. All weighting coefficients that has been developed are listed below:

$$c_0 = \left(\frac{\triangle x}{\triangle t}\right)/\sqrt{3}$$

$$w_0 = \frac{4}{9}$$

$$w_s = \frac{1}{9}$$

$$w_l = \frac{1}{36}$$

Following similar above mentioned procedures, we can get all the required weighting coefficients for the D_3Q_{19} lattice arrangement (popularly used for 3D simulations) or any 1D, 2D or 3D lattice arrangements. *Note: In LB simulations, a convenient choice is to keep $\triangle x = \triangle t = 1$ for greater simplicity.*

At this point we have successfully discretized the continuous BGK Boltzmann equation in velocity space. Based on the velocity vectors (that defines the lattice arrangement), the constraints were used to achieve the weighting coefficients, converting the continuous function to discrete one. However, the discrete-velocity Boltzmann equation further needs to be discretized in space and time. This will be explained in the next section.

1.4.4 Lattice Boltzmann equation

Discrete-velocity Boltzmann equation (1.30) is a hyperbolic partial differential equation (PDE).

$$\frac{\partial f_i}{\partial t} + c_{i\alpha} \cdot \frac{\partial f_i}{\partial x_\alpha} = -\frac{1}{\tau}\left(f_i - f_i^0\right) + F_{i\alpha}$$

One possible way of discretizing is by integrating along characteristics. The distribution function can be written as $f_i = f_i(x(s), t(s))$ where s denotes the position along the characteristics. After applying method of characteristics, we would like to have

$$\frac{\partial f_i}{\partial t} + c_{i\alpha} \cdot \frac{\partial f_i}{\partial x_\alpha} = \frac{df_i}{dt}$$

Total differentiation of $\frac{df_i}{dt}$ by chain rule gives,

$$\frac{df_i}{dt} = \left(\frac{\partial f_i}{\partial t}\right)\frac{dt}{ds} + \left(\frac{\partial f_i}{\partial x_i}\right)\frac{dx_i}{ds}$$

Now, if we set $\frac{dt}{ds} = 1$ and $\frac{dx_i}{ds} = c_i$, we reach to the L.H.S. of the DVBE equation. Hence, the linear first-order PDE is transformed to ODE (ordinary differential equation) along the characteristic line $x(s) = x(0) + c_i s$,

$$\frac{df_i}{dt} = -\frac{1}{\tau}\left(f_i - f_i^0\right) + F_i \tag{1.47}$$

Now, integrating the equation 1.47 from $s = 0$ to $s = \triangle t$ ($\triangle t$ suggesting one time-step),

$$\begin{aligned} f_i(x + c_i\triangle t, t + \triangle t) - f_i(x,t) = -\frac{1}{\tau}\int_0^{\triangle t} (f_i(x + c_i s, t + s) - f_i^0(x + c_i s, t + s))ds \\ + \int_0^{\triangle t} F_i(x + c_i s, t + s)ds \end{aligned}$$

1.4.4.1 Space-time discretization (second order)

As we can understand from the above equation that L.H.S. is exact and R.H.S. needs to be approximated. The most common approach to approximate R.H.S. is the first order discretization where the integral is approximated by rectangle method. It is fully explicit but given that it is just first order accurate approximation we choose second order discretization. The second order discretization is performed by applying semi-implicit Crank-Nicolson method. Here the integral is approximated by trapezoidal rule:

$$\begin{aligned} f_i(x + c_i\triangle t, t + \triangle t) - f_i(x,t) = -\frac{\triangle t}{2\tau}[f_i(x + c_i\triangle t, t + \triangle t) - f_i^0(x + c_i\triangle t, t + \triangle t) \\ + f_i(x,t) - f_i^0(x,t)] + \frac{\triangle t}{2}[F_i(x + c_i\triangle t, t + \triangle t) + F_i(x,t)] \end{aligned} \tag{1.48}$$

The above discretization is semi-implicit. It can be made fully explicit by introducing a redefined distribution function $\tilde{f}_i$ for the lattice populations given by

$$\boxed{\tilde{f}_i(x,t) = f_i(x,t) + \frac{\triangle t}{2\tau}[f_i(x,t) - f_i^0(x,t) - \tau F_i(x,t)]} \tag{1.49}$$

Applying equation 1.49 in equation 1.48,

$$\begin{aligned} f_i(x + c_i\triangle t, t + \triangle t) - [\tilde{f}_i(x,t) - \frac{\triangle t}{2\tau}(f_i(x,t) - f_i^0(x,t) - \tau F_i(x,t))] = \\ -\frac{\triangle t}{2\tau}[f_i(x + c_i\triangle t, t + \triangle t) - f_i^0(x + c_i\triangle t, t + \triangle t) + f_i(x,t) - f_i^0(x,t)] \\ + \frac{\triangle t}{2}[F_i(x + c_i\triangle t, t + \triangle t) + F_i(x,t)] \end{aligned}$$

$$\begin{aligned} \left(1 + \frac{\triangle t}{2\tau}\right) f_i(x + c_i\triangle t, t + \triangle t) = \tilde{f}_i(x,t) - \frac{\triangle t}{2\tau}[f_i(x,t) - f_i^0(x,t) - \tau F_i(x,t)] \\ -\frac{\triangle t}{2\tau}[f_i(x,t) - f_i^0(x,t) - \tau F_i(x,t)] + \frac{\triangle t}{2\tau}[f_i^0(x + c_i\triangle t, t + \triangle t) \\ +\tau F_i(x + c_i\triangle t, t + \triangle t)] \end{aligned} \tag{1.50}$$

Using equation 1.49 in equation 1.50 and further simplifying,

$$\tilde{f}_i(x + c_i\triangle t, t + \triangle t) = \tilde{f}_i(x,t) - \frac{\triangle t}{\tau}[f_i(x,t) - f_i^0(x,t) - \tau F_i(x,t)] \tag{1.51}$$

We still need to transform the second term of R.H.S. in the form of redefined distribution function.
It shall be noted that at equilibrium conditions, we get

$$\boxed{\tilde{f}_i^0(x,t) = f_i^0(x,t)} \tag{1.52}$$

Redefined distribution function $\tilde{f}_i(x,t)$, which is involved in the second order discretization, corrects ***only the non-equilibrium part of the distribution function*** *adding higher accuracy (to represent the non-equilibrium part better than the one described by $f_i(x,t)$). Therefore, both definitions of the distribution function should have similar equilibrium part.*

Proceeding further from equation 1.49,

$$\tilde{f}_i(x,t) = \left(1 + \frac{\triangle t}{2\tau}\right) f_i(x,t) - \frac{\triangle t}{2\tau}[f_i^0(x,t) + \tau F_i(x,t)]$$

Applying equation 1.52 in the above equation

$$\tilde{f}_i(x,t) = \left(1 + \frac{\triangle t}{2\tau}\right) f_i(x,t) - \frac{\triangle t}{2\tau}\left[\tilde{f}_i^0(x,t) + \tau F_i(x,t)\right]$$

$$f_i(x,t) = \frac{\tilde{f}_i(x,t) + \frac{\triangle t}{2\tau}\left[\tilde{f}_i^0(x,t) + \tau F_i(x,t)\right]}{\left(1 + \frac{\triangle t}{2\tau}\right)} \tag{1.53}$$

Now applying equations 1.52 and 1.53 in equation 1.51,

$$\tilde{f}_i(x + c_i\triangle t, t + \triangle t) = \tilde{f}_i(x,t) - \frac{\triangle t}{\tau}\left[\frac{\tilde{f}_i(x,t) + \frac{\triangle t}{2\tau}\left(\tilde{f}_i^0(x,t) + \tau F_i(x,t)\right)}{\left(1 + \frac{\triangle t}{2\tau}\right)} - \tilde{f}_i^0(x,t) - \tau F_i(x,t)\right] \tag{1.54}$$

$$\tilde{f}_i(x + c_i\triangle t, t + \triangle t) = \tilde{f}_i(x,t) - \frac{\triangle t}{\tau}\left[\frac{\tilde{f}_i(x,t) - \tilde{f}_i^0(x,t) + \frac{\triangle t}{2}F_i(x,t)}{1 + \frac{\triangle t}{2\tau}}\right] + \triangle t F_i(x,t)$$

$$\tilde{f}_i(x + c_i\triangle t, t + \triangle t) = \tilde{f}_i(x,t) - \frac{\triangle t}{\tau + \frac{\triangle t}{2}}\left(\tilde{f}_i(x,t) - \tilde{f}_i^0(x,t)\right) + \triangle t F_i(x,t) - \frac{\triangle t}{\tau}\frac{\triangle t}{2}\frac{1}{(1 + \frac{\triangle t}{2\tau})}F_i(x,t)$$

$$\tilde{f}_i(x + c_i\triangle t, t + \triangle t) = \tilde{f}_i(x,t) - \frac{\triangle t}{\tau + \frac{\triangle t}{2}}\left(\tilde{f}_i(x,t) - \tilde{f}_i^0(x,t)\right) + \triangle t\left(1 - \frac{\triangle t}{2(\tau + \frac{\triangle t}{2})}\right)F_i(x,t)$$

$$\boxed{\tilde{f}_i(x + c_i\triangle t, t + \triangle t) = \tilde{f}_i(x,t) - \frac{\triangle t}{\tilde{\tau}}\left(\tilde{f}_i(x,t) - \tilde{f}_i^0(x,t)\right) + \triangle t\left(1 - \frac{\triangle t}{2\tilde{\tau}}\right)F_i(x,t)} \tag{1.55}$$

Here, $\tilde{\tau} = \tau + \frac{\triangle t}{2}$ is a redefined relaxation time ($\tilde{\tau} > \frac{\triangle t}{2}$).
This is the equation for the standard LB method. It is also important to note that the equation is isothermal but with body force in the system. At this point a short comment on the forcing term is essential.

1.4.4.2 Force term

The simple way to implement it, is by the expression:

$$F_i = w_i \frac{c_i}{c_0^2} \mathbf{F}$$

where the summation over index i is not implied, w_i is a lattice dependent weight and $\mathbf{F}$ is the body force per unit volume in physical space ($\mathbf{F} = \rho\, \mathbf{a}$). The above expression satisfies the condition,

$$\sum_i F_i = 0$$

$$\sum_i c_i F_i = \mathbf{F} = \rho\, \mathbf{a}$$

These are essential to give the correct macroscopic effect of a body force term. However, when the body force is time/space dependent and discrete-velocity Boltzmann equation is discretized in space and time, the above expression needs to be refined in order to remove spurious discretization terms that would otherwise appear in the macroscopic limit. The corrected expression, first proposed by Guo *et al.* [78] is

$$F_i = w_i \left(\frac{c_i - \mathbf{u}}{c_0^2} + \frac{(c_i \cdot \mathbf{u})\, c_i}{c_0^4} \right) \mathbf{F} \tag{1.56}$$

The above correction was meant for the first order discretization of space-time. It was also observed by Guo *et al.* [78] that it was not enough to match the correct Navier-Stokes equations and verified that the expression required a multiplicative factor $1 - \frac{\triangle t}{2\tau}$. This multiplicative factor of the forcing term can be accurately obtained if we follow second order discretization in space-time as seen in equation 1.55. This proves that it is important to do second order discretization (involving redefined distribution function, $\tilde{f}$) to achieve Navier-Stokes level. Thus the correct expression for the force term is,

$$\boxed{F_i = \left(1 - \frac{\triangle t}{2\tilde{\tau}}\right) w_i \left(\frac{c_i - \mathbf{u}}{c_0^2} + \frac{(c_i \cdot \mathbf{u})\, c_i}{c_0^4} \right) \mathbf{F}} \tag{1.57}$$

1.4.5 Connection to Navier-Stokes equation : Chapman-Enskog expansion

The macroscopic behaviour displayed by the LB equation can be derived by means of a multi-scaling analysis called the Chapman Enskog expansion. This is done by introducing an expansion parameter ϵ which is proportional to the Knudsen number (Kn). Knudsen number is defined as the ratio of the mean free path to the characteristic length. As Kn approaches zero, the mean free path also approaches zero, suggesting that the number of collisions also approach zero and the system approaches to equilibrium state (as $Kn \to 0$, $f \simeq f^0$). This implies that Euler equations are the approximate macroscopic description of the fluid flow. We know that the more detailed description of the fluid flow is given by the Navier-Stokes equations. So, the more detailed description of the flow can be best represented by the addition of small perturbations to an equilibrium state, which increases with Kn. Thus, expansion parameter ϵ serves as this small perturbation depending on the increase of Kn.

$$Kn \to \epsilon Kn$$

While introducing multi-scale expansion, we expand the distribution function around equilibrium with terms in the increasing order of Kn,

$$\tilde{f}_i = \tilde{f}_i^0 + \epsilon \tilde{f}_i^1 + \epsilon^2 \tilde{f}_i^2 + \cdots\cdots$$

Expansions for time, space and force term are also introduced,

$$\frac{\partial}{\partial t} \to \epsilon \frac{\partial}{\partial t_1} + \epsilon^2 \frac{\partial}{\partial t_2} + \cdots\cdots$$

$$\frac{\partial}{\partial x_\alpha} \to \epsilon \frac{\partial}{\partial x_\alpha}$$

$$F_i \to \epsilon F_i$$

To make the derivation more convenient later, we also define the moments of body force term as follows:

$$\sum_i F_i = 0 \qquad \sum_i c_i F_i = \mathbf{F} \qquad \sum_i c_i c_i F_i = \mathbf{FF}$$

Now proceeding with the multi-scaling analysis of standard LB equation (1.55), firstly the L.H.S. needs to be expanded by Taylor expansion truncating up to second order,

$$\tilde{f}_i(x + c_i\triangle t, t + \triangle t) - \tilde{f}_i(x,t) \approx \left(c_{i\alpha}\, \triangle t \frac{\partial}{\partial x_\alpha} + \triangle t \frac{\partial}{\partial t} \right) \tilde{f}_i + \frac{1}{2} \left(c_{i\alpha} c_{i\beta} (\triangle t)^2 \frac{\partial^2}{\partial x_\alpha^2} + 2\, c_{i\alpha}\, \triangle t\, \triangle t\, \frac{\partial}{\partial x_\alpha} \frac{\partial}{\partial t} + (\triangle t)^2 \frac{\partial}{\partial t^2} \right) \tilde{f}_i$$

Then, all the multi-scale expansions are introduced into the expanded version of equation 1.55

$$\triangle t\left[c_{i\alpha}\left(\epsilon\frac{\partial}{\partial x_\alpha}\right)+\left(\epsilon\frac{\partial}{\partial t_1}+\epsilon^2\frac{\partial}{\partial t_2}\right)\right](\tilde{f}_i^0+\epsilon\tilde{f}_i^1+\epsilon^2\tilde{f}_i^2)+\frac{\triangle t^2}{2}\left[c_{i\alpha}c_{i\beta}\left(\epsilon^2\frac{\partial^2}{\partial x_\alpha^2}\right)\right.$$
$$\left.+2\,c_{i\alpha}\,\epsilon^2\frac{\partial}{\partial x_\alpha}\frac{\partial}{\partial t_1}+\epsilon^2\frac{\partial}{\partial t_1^2}\right](\tilde{f}_i^0+\epsilon\tilde{f}_i^1+\epsilon^2\tilde{f}_i^2)=-\frac{\triangle t}{\tilde{\tau}}(\epsilon\tilde{f}_i^1+\epsilon^2\tilde{f}_i^2)+\triangle t\left(1-\frac{\triangle t}{2\tilde{\tau}}\right)(\epsilon F_i)$$

The next step follows the regrouping of terms with the same order of ϵ upto $O(\epsilon^2)$ because $O(\epsilon^2)$ sufficiently contributes to the Navier-Stokes level.

$O(\epsilon)$:

$$\left(c_{i\alpha}\frac{\partial}{\partial x_\alpha}+\frac{\partial}{\partial_{t_1}}\right)\tilde{f}_i^0=-\frac{1}{\tilde{\tau}}\tilde{f}_i^1+\left(1-\frac{\triangle t}{2\tilde{\tau}}\right)F_i \tag{1.58}$$

$O(\epsilon^2)$:

$$\frac{\partial}{\partial t_2}\tilde{f}_i^0+\left(c_{i\alpha}\frac{\partial}{\partial x_\alpha}+\frac{\partial}{\partial_{t_1}}\right)\tilde{f}_i^1+\frac{\triangle t}{2}\left(c_{i\alpha}c_{i\beta}\frac{\partial^2}{\partial x_\alpha^2}+2\,c_{i\alpha}\frac{\partial}{\partial x_\alpha}\frac{\partial}{\partial t_1}+\frac{\partial}{\partial t_1^2}\right)\tilde{f}_i^0=-\frac{1}{\tilde{\tau}}\tilde{f}_i^2$$

$$\frac{\partial}{\partial t_2}\tilde{f}_i^0+\left(c_{i\alpha}\frac{\partial}{\partial x_\alpha}+\frac{\partial}{\partial_{t_1}}\right)\tilde{f}_i^1+\frac{\triangle t}{2}\left(c_{i\alpha}\frac{\partial}{\partial x}+\frac{\partial}{\partial t_1}\right)^2\tilde{f}_i^0=-\frac{1}{\tilde{\tau}}\tilde{f}_i^2 \tag{1.59}$$

Using equation 1.58 to the above equation,

$$\frac{\partial}{\partial t_2}\tilde{f}_i^0+\left(c_{i\alpha}\frac{\partial}{\partial x_\alpha}+\frac{\partial}{\partial_{t_1}}\right)\left(1-\frac{\triangle t}{2\tilde{\tau}}\right)\tilde{f}_i^1=-\frac{1}{\tilde{\tau}}\tilde{f}_i^2-\frac{\triangle t}{2}\left(1-\frac{\triangle t}{2\tilde{\tau}}\right)\left(c_{i\alpha}\frac{\partial}{\partial x_\alpha}+\frac{\partial}{\partial_{t_1}}\right)F_i \tag{1.60}$$

We have seen from the previous sections that the moments of f, f^0 and f_i^0 are same. This means the contribution of the higher order terms f^n or f_i^n (non-equilibrium part of the distribution function) to these moments are null. This implies,

$$\int f_i^n d\mathbf{c}=\int \mathbf{c}_i f_i^n d\mathbf{c}=0 \qquad for\;\; n>0 \tag{1.61}$$

Now, we write the zeroth to second moments of equation 1.58, which represent the macroscopic equations on time scale t_1 and space scale $\mathbf{x}_1$.

$$\frac{\partial\rho}{\partial t_1}+\frac{\partial\rho u_\alpha}{\partial x_\alpha}=0 \tag{1.62}$$

$$\frac{\partial\rho u_\alpha}{\partial t_1}+\frac{\Pi^0_{\alpha\beta}}{\partial x_\beta}=\mathbf{F}_{(1)} \tag{1.63}$$

$$\frac{\Pi^0_{\alpha\beta}}{\partial t_1}+\frac{\Pi^0_{\alpha\beta\gamma}}{\partial x_\gamma}=-\frac{1}{\tilde{\tau}}\,\Pi^1_{\alpha\beta}+\mathbf{FF}_{(1)} \tag{1.64}$$

Similarly, we write zeroth and first order moments of equation 1.60, which represents the macroscopic equations on time scale t_2.

$$\frac{\partial \rho}{\partial t_2} = 0 \tag{1.65}$$

$$\frac{\partial \rho u_\alpha}{\partial t_2} + \left(1 - \frac{\triangle t}{2\tilde{\tau}}\right)\frac{\Pi^1_{\alpha\beta}}{\partial x_\beta} = \mathbf{F}_{(2)} \tag{1.66}$$

Now, we recombine the moment equations of different time scales. Different orders of ϵ are added and the expansion is reversed to get the conservation equations.
Following the same procedure, equations 1.62 and 1.65 are taken to give the continuity equation.

$$\boxed{\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_\alpha}{\partial x_\alpha} = 0} \tag{1.67}$$

Similarly, momentum conservation equations are derived by taking equations 1.63 and 1.66.

$$\frac{\partial \rho u_\alpha}{\partial t} + \frac{\partial}{\partial x_\alpha}\left[\Pi^0_{\alpha\beta} + \left(1 - \frac{\triangle t}{2\tilde{\tau}}\right)\Pi^1_{\alpha\beta}\right] = \mathbf{F}$$

As seen from the above equation, we need to find $\Pi^0_{\alpha\beta}$ and $\Pi^1_{\alpha\beta}$.
$\Pi^0_{\alpha\beta} = \rho u_\alpha u_\beta + \rho c_0^2 \delta_{\alpha\beta}$ is found from equation 1.38 leading to

$$\frac{\partial \rho u_\alpha}{\partial t} + \frac{\partial}{\partial x_\alpha}\left[\rho u_\alpha u_\beta + \rho c_0^2 \delta_{\alpha\beta} + \left(1 - \frac{\triangle t}{2\tilde{\tau}}\right)\Pi^1_{\alpha\beta}\right] = \mathbf{F} \tag{1.68}$$

$\Pi^1_{\alpha\beta}$ can be found by equation 1.64 which requires $\frac{\Pi^0_{\alpha\beta}}{\partial t_1}$ and $\frac{\Pi^0_{\alpha\beta\gamma}}{\partial x_\gamma}$ to be known but before that we need to resolve some terms that are useful later.

$$\frac{\partial \rho u_\alpha u_\beta}{\partial t_1} = u_\alpha \frac{\partial u_\beta}{\partial t_1} + u_\beta \frac{\partial \rho u_\alpha}{\partial t_1} - u_\alpha u_\beta \frac{\partial \rho}{\partial t_1} \tag{1.69}$$

$$\frac{\partial \rho u_\alpha u_\beta u_\gamma}{\partial x_\alpha} = u_\alpha \frac{\partial \rho u_\beta u_\gamma}{\partial x_\gamma} + u_\beta \frac{\partial \rho u_\alpha u_\gamma}{\partial x_\gamma} - u_\alpha u_\beta \frac{\partial \rho u_\gamma}{\partial x_\gamma} \tag{1.70}$$

Resolving $\frac{\Pi^0_{\alpha\beta}}{\partial t_1}$:

$$\frac{\partial \Pi^0_{\alpha\beta}}{\partial t_1} = \frac{\partial \rho u_\alpha u_\beta}{\partial t_1} + \frac{\partial \rho c_0^2 \delta_{\alpha\beta}}{\partial t_1}$$

$$\frac{\partial \Pi^0_{\alpha\beta}}{\partial t_1} = u_\alpha \frac{\partial \rho u_\beta}{\partial t_1} + u_\beta \frac{\partial \rho u_\alpha}{\partial t_1} - u_\alpha u_\beta \frac{\partial \rho}{\partial t_1} + c_0^2 \delta_{\alpha\beta} \frac{\partial \rho}{\partial t_1} \qquad (using\ 1.69)$$

$$\frac{\partial \Pi^0_{\alpha\beta}}{\partial t_1} = u_\alpha \left(-\frac{\partial \Pi^0_{\beta\gamma}}{\partial x_\gamma} + \mathbf{F}_{(1)}\right) + u_\beta \left(-\frac{\partial \Pi^0_{\alpha\gamma}}{\partial x_\gamma} + \mathbf{F}_{(1)}\right) - u_\alpha u_\beta \frac{\partial \rho}{\partial t_1} + c_0^2 \delta_{\alpha\beta} \frac{\partial \rho}{\partial t_1} \qquad (using\ 1.63)$$

$$\frac{\partial \Pi^0_{\alpha\beta}}{\partial t_1} = -u_\alpha \frac{\partial}{\partial x_\gamma}(\rho u_\beta u_\gamma + \rho c_0^2 \delta_{\beta\gamma}) + u_\alpha \mathbf{F}_{(1)} - u_\beta \frac{\partial}{\partial x_\gamma}(\rho u_\alpha u_\gamma + \rho c_0^2 \delta_{\alpha\gamma}) + u_\beta \mathbf{F}_{(1)} + u_\alpha u_\beta \frac{\rho u_\gamma}{\partial x_\gamma} - c_0^2 \delta_{\alpha\beta} \frac{\partial \rho u_\gamma}{x_\gamma} \qquad (using\ 1.38)$$

$$\frac{\partial \Pi^0_{\alpha\beta}}{\partial t_1} = -\frac{\partial \rho u_\alpha u_\beta u_\gamma}{\partial x_\gamma} - c_0^2\left(u_\alpha \frac{\partial \rho}{\partial x_\beta} + u_\beta \frac{\partial \rho}{\partial x_\alpha}\right) + (u_\alpha \mathbf{F}_{(1)} + u_\beta \mathbf{F}_{(1)}) - c_0^2 \delta_{\alpha\beta} \frac{\partial \rho u_\gamma}{\partial x_\gamma} \tag{1.71}$$

Resolving $\frac{\Pi^0_{\alpha\beta\gamma}}{\partial x_\gamma}$:

$$\frac{\partial \Pi^0_{\alpha\beta\gamma}}{\partial x_\gamma} = \frac{\partial}{\partial x_\gamma} \rho c_0^2 (u_\alpha \delta_{\beta\gamma} + u_\beta \delta_{\alpha\gamma} + u_\gamma \delta_{\alpha\beta}) \qquad (using\ 1.41)$$

$$\frac{\partial \Pi^0_{\alpha\beta\gamma}}{\partial x_\gamma} = c_0^2 \left(\frac{\partial \rho u_\alpha}{\partial x_\beta} + \frac{\partial \rho u_\beta}{\partial x_\alpha}\right) + c_0^2 \delta_{\alpha\beta} \frac{\partial \rho u_\gamma}{\partial x_\gamma} \tag{1.72}$$

Using these two terms (1.71 and 1.72), equation 1.64 becomes,

$$\Pi^1_{\alpha\beta} = -\rho c_0^2 \tilde{\tau} \left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right) + \tilde{\tau} \frac{\partial \rho u_\alpha u_\beta u_\gamma}{\partial x_\gamma} - \tilde{\tau}(u_\alpha \mathbf{F}_{(1)} + u_\beta \mathbf{F}_{(1)}) + \mathbf{F}\mathbf{F}_{(1)} \tag{1.73}$$

There are important simplifications to do at this stage. $-\tilde{\tau}(u_\alpha \mathbf{F}_{(1)} + u_\beta \mathbf{F}_{(1)})$ and $\mathbf{F}\mathbf{F}_{(1)}$ cancel each other to correct the discrete lattice effects on stress and viscosity [78]. The other term $\tilde{\tau} \frac{\partial \rho u_\alpha u_\beta u_\gamma}{\partial x_\gamma}$ is an error term. If we recall equation 1.41, $\rho u_\alpha u_\beta u_\gamma$ was dropped while truncating upto second order. This error term is a by-product of that dropped term. We can see from the expression that to minimise the error, u^2 should be very less than c_0^2 ($i.e.$ $u^2 << c_0^2$ or $M << 1$. For incompressible flows, M is usually chosen to be around 0.1 to be within safe limits). Applying these corrections, equation 1.73 reduces to something similar to stress tensor.

$$\Pi^1_{\alpha\beta} = -\rho c_0^2 \tilde{\tau} \left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right) \tag{1.74}$$

Now applying equation 1.74 to equation 1.68 we get,

$$\frac{\partial \rho u_\alpha}{\partial t} + \frac{\partial}{\partial x_\alpha}\left[\rho u_\alpha u_\beta + \rho c_0^2 \delta_{\alpha\beta} - \left(1 - \frac{\triangle t}{2\tilde{\tau}}\right) \rho c_0^2 \tilde{\tau} \left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right)\right] = \mathbf{F} \tag{1.75}$$

Rearranging,

$$\boxed{\frac{\partial \rho u_\alpha}{\partial t} + \frac{\partial \rho u_\alpha u_\beta}{\partial x_\alpha} = -\frac{\partial p}{\partial x_\alpha} + \rho c_0^2 \tilde{\tau} \left(1 - \frac{\triangle t}{2\tilde{\tau}}\right) \frac{\partial}{\partial x_\alpha} \left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right) + \mathbf{F}} \tag{1.76}$$

This is the Navier-Stokes momentum equation with the body force term. *It is important to note that ρc_0^2 has been replaced with pressure (p). Also, it can be seen that $\rho c_0^2 \tilde{\tau}(1 - \frac{\triangle t}{2\tilde{\tau}}) = \rho c_0^2(\tilde{\tau} - \frac{\triangle t}{2})$ corresponds to the viscosity and when $\rho = 1$ the viscosity is given by*

$c_0^2(\tilde{\tau} - \frac{\triangle t}{2})$.
Finally, we have seen that discrete-velocity Boltzmann equation with a given velocity set and second order space-time discretization reproduces exactly the Navier-Stokes equations under the condition $M << 1$.

1.4.5.1 Link to macroscopic quantities

We know that the contribution of the higher order terms f^n or f_i^n (non-equilibrium part of the distribution function) to these moments are null. For all $n \geqslant 1$,

$$\int f^n d\mathbf{c} = \int f_i^n d\mathbf{c} = \int \mathbf{c} f^n d\mathbf{c} = \int \mathbf{c}_i f_i^n d\mathbf{c} = 0$$

Now if we again take equation 1.49

$$\tilde{f}_i(x,t) = f_i(x,t) + \frac{\triangle t}{2\tau}[f_i(x,t) - f_i^0(x,t)] - \frac{\triangle t}{2} F_i(x,t)$$

It can be observed that the second term in R.H.S. (non-equilibrium part) will not contribute to the moments of f_i. This will help to deduce relations between the macroscopic variables and the tilded ($\sim$) quantities.

$$\textbf{Density}, \; \rho = \sum_i f_i = \sum_i \left(\tilde{f}_i - \frac{\triangle t}{2\tau}(f_i - f_i^0) + \frac{\triangle t}{2} F_i \right) = \sum_i \tilde{f}_i \tag{1.77}$$

(non-equilibrium part doesn't contribute at all and the forcing term doesn't contribute to the zeroth moment)

$$\rho\mathbf{u} = \sum_i c_i f_i = \sum_i c_i \left(\tilde{f}_i - \frac{\triangle t}{2\tau}(f_i - f_i^0) + \frac{\triangle t}{2} F_i \right) = \sum_i c_i \left(\tilde{f}_i + \frac{\triangle t}{2} F_i \right) = \sum_i c_i \tilde{f}_i + \frac{\triangle t}{2} \mathbf{F}$$

(non-equilibrium part doesn't contribute at all)

$$\textbf{Fluid velocity}, \; \mathbf{u} = \frac{1}{\rho} \left(\sum_i c_i \tilde{f}_i + \frac{\triangle t}{2} \mathbf{F} \right) \tag{1.78}$$

Other important relations with the macroscopic quantities are:

$$\textbf{Viscosity}, \; \nu = \rho\, c_0^2 \left(\tilde{\tau} - \frac{\triangle t}{2} \right) = \rho\, c_0^2\, \tau \tag{1.79}$$

The above relation is given by equation 1.76, from the previous section.

$$\textbf{Pressure}, \; p = \rho\, c_0^2 \tag{1.80}$$

1.5 Summary

This chapter started with discussion of the behaviour of the particles in a ideal gas system consequently deriving Boltzmann equation. It followed with defining the Maxwell-Boltzmann distribution function to arrive at discrete-velocity Boltzmann equation then to discretized Lattice Boltzmann equation and finally showed its equivalence to Navier-Stokes equations.
The discrete-velocity Lattice Boltzmann equation with BGK approximation for the collision term, read as (equation 1.30)

$$\frac{\partial f_i}{\partial t} + c_{i\alpha} \cdot \frac{\partial f_i}{\partial x_\alpha} = -\frac{1}{\tau}\left(f_i - f_i^0\right) + F_{i\alpha}$$

and the discrete equilibrium distribution function was given by (equation 1.29)

$$f_i^0 = \rho w_i\left(1 + \frac{c_{i\alpha}u_\alpha}{c_0^2} + \frac{c_{i\alpha}u_\alpha c_{i\beta}u_\beta}{2c_0^4} - \frac{u_\alpha u_\alpha}{2c_0^2}\right)$$

Velocity discretization constrained the velocity vector vectors and their weighting coefficient. Taking D_2Q_9 as an example, the discrete velocities and their weighting coefficients were defined by

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

and

$$c_0 = \left(\frac{\triangle x}{\triangle t}\right)/\sqrt{3}$$

$$w_0 = \frac{4}{9}$$

$$w_s = \frac{1}{9}$$

$$w_l = \frac{1}{36}$$

Further discretization on space-time using second order discretization resulted the Lattice Boltzmann equation (1.55).

$$\tilde{f}_i(x + c_i\triangle t, t + \triangle t) = \tilde{f}_i(x,t) - \frac{\triangle t}{\tilde{\tau}}\left(\tilde{f}_i(x,t) - \tilde{f}_i^0(x,t)\right) + \triangle t\left(1 - \frac{\triangle t}{2\tilde{\tau}}\right)F_i(x,t)$$

where $\tilde{f}$ is a redefined distribution function (equation 1.49) and $\tilde{\tau}$ is a redefined relaxation time.

$$\tilde{f}_i(x,t) = f_i(x,t) + \frac{\triangle t}{2\tau}[f_i(x,t) - f_i^0(x,t) - \tau F_i(x,t)]$$

$$\tilde{\tau} = \tau + \frac{\triangle t}{2}$$

The distribution function can be used to describe macroscopic quantities as follows:

$$\textbf{Density},\ \rho = \sum_i \tilde{f}_i$$

$$\textbf{Fluid velocity},\ \mathbf{u} = \frac{1}{\rho}\left(\sum_i c_i\tilde{f}_i + \frac{\triangle t}{2}\mathbf{F}\right)$$

$$\textbf{Viscosity},\ \nu = \rho\, c_0^2\, \tau$$

Implementation:

Implementation of the standard Lattice Boltzmann equation in the code for fluid flow simulations involves two major steps:

- collision: $-\frac{\triangle t}{\tilde{\tau}}\left(\tilde{f}_i(x,t) - \tilde{f}_i^0(x,t)\right)$
  Collision process is described locally and the distribution function is updated by trying to reduce the discrepancy between actual and equilibrium one.

- streaming: We move the direction specific densities $\tilde{f}_i$ to the neighbouring lattice nodes (refer figure 1.4). This doesn't require computation but just the rearrangement of the allocated memory. *For example:* Population $\tilde{f}_7$ moves from site (x,y) to $(x+1,y+1)$ and $\tilde{f}_8$ moves from site (x,y) to $(x,y+1)$ etc. Similarly population $\tilde{f}_2$ comes from site $(x+1,y)$ to (x,y) etc.

Including these two major steps, here we present a typical algorithm for standard LBM code:

(1) Initialize with appropriate initial conditions.

(2) Compute the macroscopic quantities like density and velocity.

(3) Add collision:

$$\tilde{f}_i(x,t) = \tilde{f}_i(x,t) - \frac{\triangle t}{\tilde{\tau}}\left(\tilde{f}_i(x,t) - \tilde{f}_i^0(x,t)\right)$$

(4) Add forcing:

$$\tilde{f}_i(x,t) = \tilde{f}_i(x,t) - \frac{\triangle t}{\tilde{\tau}}\left(\tilde{f}_i(x,t) - \tilde{f}_i^0(x,t)\right) + \triangle t\big(1 - \frac{\triangle t}{2\tilde{\tau}}\big)F_i(x,t)$$

(5) Apply boundary conditions.

(6) Perform streaming.

(7) Update the macroscopic quantities.

(8) If time left for simulation, goto (3) else end.

Part I

Chapter 2

Overview of mesh refinement approaches for the LBM

2.1 Introduction

The traditional LB (Lattice Boltzmann) method has a major drawback that it is restricted to regular square grids. For many flows of practical applications, there are regions in the domain with high gradients and/or curved geometry where the flow demands high grid resolution; or pressure far-field where high resolution is not required *e.g.* simulating turbulent flows over a wing profile. We recall here that a developed turbulent flow in the presence of any sort of bounding geometry (or any local forcing term) develops space inhomogeneities and as such grid refinement in numerics becomes necessary. It is clear that in such a context, grid-refinement is not an additional requirement in order to increase the accuracy of a simulation but is rather an unavoidable need in order to save memory usage and computational power and being able to access higher - read more realistic - turbulent flows regimes. In such cases, traditional LB quickly shows its computational inefficiency.
Improvements can be done by introducing non-uniform grids. We discussed in the previous chapter that in a uniform grid, we applied method of characteristics because it drastically simplified the DVBE (Discrete-Velocity Boltzmann Equation). After applying method of characteristics to the DVBE, the distribution function of the particles varied only along the characteristic lines to reach to the neighbouring nodes giving a tight link between numerical mesh and lattice structure. So, the obvious thing to do is to **modify** or **replace** the *streaming* process. This can be done by treating advection term in various ways described below, which will introduce non-uniformity in the mesh design.

2.2 Types of mesh refinement

2.2.1 Interpolation-based mesh refinement

In 1996, He *et al.* [1] proposed an algorithm based on the interpolation of distribution function. The basic assumption is that the distribution function is sufficiently smooth in the non-uniform mesh. This algorithm involves three steps: *collision*; *streaming* and a new step *interpolation*. Collision still takes place at grid points of the mesh. After collision, the distribution functions move according to their velocity along characteristic lines but may not fall exactly on the neighbouring grid point. So it is interpolated to fall exactly at the next grid point by a quadratic interpolation scheme to maintain second order accuracy. After this, collision and advection are again repeated. The time step for the whole domain was fixed.
He et al. claimed that this algorithm retains all the advantages of the standard LBM. However the disadvantage of this method is that it can't guarantee mass and momentum conservation. Here, accuracy not only relies on the order of interpolation but it also adds numerical diffusivity and other artifacts (see discussions in [79]).

2.2.2 Rectangular LBM

To avoid interpolations, Bouzidi *et al.* [2] proposed a rectangular LB method. The idea was inspired from the work of Koelman [80] who suggested spatial non-uniformity in the lattice structure. He presented LB scheme that could work with face-centered rectangular lattice.
The basic idea of rectangular LB technique is to obtain the constraints on the isotropy of the transport coefficients and Galilean invariance by analysing the linearised dispersion equation. These constraints replace the constraints we developed in section 1.4.3.1. The velocity set developed from these constraints has spatial dependence ($\nabla c_i f \neq c_i \nabla f$) and requires large velocity set to satisfy the constraints, already making the method bulky. Although it is more general than the previous technique, the method is very complicated compared to the square grid LBM. It is severely disadvantageous. This method is very prone to instability and is stable only with multi-relaxation model for collision integral. The acceptable maximum local velocity magnitude is less than the square grid LBM. The authors themselves suggested that these issues must be improved to have a robust non-uniform grid LB model.

2.2.3 Locally embedded mesh refinement

Filippova *et al.* [3] proposed a different mesh refinement method with locally embedded grids. Also called multi-scale method, this type of mesh refinement locally refines or coarsens the lattice and time-step with identical grid structure. This means, main coarse grid and hierarchically refined grid lives on different space and time. A cartoon sketch clarifying the idea of arrangement of grids at different refinement levels is shown in figure 2.1.

To make the explanation more clear, let us suppose $\triangle x_f$, $\triangle t_f$ represent space and time spacing for fine grid respectively and $\triangle x_c$, $\triangle t_c$ for coarse grid. If n is the level of refinement, spatially the coarse and fine grid is linked by $\triangle x_c = 2^n \triangle x_f$ and temporally by $\triangle t_c = 2^n \triangle t_f$. Here it is easy to notice that the grid velocities are equal for all grids because grid spacing and time stepping are simultaneously adjusted according to the refinement (It can be justified from section 1.4.3.2 with the definition of speed of sound given by $c_0 = (\frac{\triangle x}{\triangle t})/\sqrt{3}$. Since, $\triangle x$ and $\triangle t$ simultaneously adjust, c_0 is always constant).

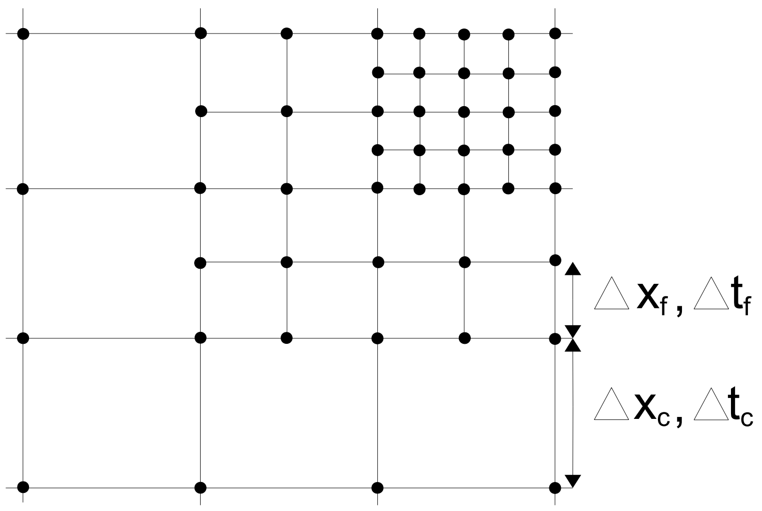

$\triangle x_f, \triangle t_f$
$\triangle x_c, \triangle t_c$

FIGURE 2.1: A typical representation of locally embedded grids [16]

The algorithm is as follows: LB equation is solved at the coarse grid level over whole domain then the boundary conditions on the fine grid level are interpolated from the rescaled distribution function of the coarse grid using second order accurate interpolation. Some of the consequences of this method are the following:

- Such refinement allows lesser time steps on coarse grid than on fine grid thus reducing computational effort.

- The approach demands adjustment of relaxation times for both levels of refinement to have equal viscosity over all domain.

$$\nu_c = \left(\frac{1}{2^n}\right)\nu_f$$

$$\tau_c = \frac{\left(\tau_f - \frac{1}{2}\right)}{2^n} + \frac{1}{2}$$

- Collision and advection take place for both coarse and fine grid. The major issue is the definition of the distribution function at the common grid nodes of the coarse and fine grid.

Filippova *et al.* [3] corrected the non-equilibrium part of the distribution function for different time steps and relaxation parameters to adjust the collision step on the common grid nodes. Lin and Lai [81] suggested that rescaling of the non-equilibrium part can be avoided by communicating the levels of grid refinement after the advection step rather than after collision step. This introduced errors at the local components of stress tensor. Dupuis and Chopard [82] proposed a different rescaling after the propagation step that was simpler to use than the method by Filippova *et al.* [3]. Recently Lagrava *et al.* [83] further improved it by applying a local average to non-equilibrium part at the first neighbouring nodes of fine grid before performing the mapping to coarse grid. This highly improved the stability.
It is again seen that the method uses interpolation so the mass conservation can't be guaranteed easily. Also, the method requires rescaling of non-equilibrium part of the distribution function so is confined to specific "single relaxation time" collision operator deleting the flexibility of incorporating "multi relaxation time" collision operator.

2.2.3.1 Volumetric formulation of locally embedded grid

One of the better solution of conserving mass is by introducing volumetric formulation. Originally, LB method is a finite difference scheme with grid nodes (will be seen in detail in Finite Difference section). Rohde *et al.* [16] saw the possibility of reformulating to grid cells. Thus, they introduced volumetric formulation to the locally embedded grid refinement (as shown in figure 2.2) avoiding interpolation or rescaling of non-equilibrium distribution. The steps of the method can be summarised as follows:

- Collision step on coarse and fine grid cells.
- Each coarse grid cell can be considered to be a number of fine grid cells. This step consists of homogeneous redistribution of particle densities in each coarse cell to its fine grid cells.

- Propagation step on the coarse and fine grid.
- Collision step on fine grid only.
- Again perform propagation step on the coarse and fine grid after the collision to fine grid only.
- Repeat previous two steps for $n-1$ times where n is the level of refinement.
- At the final step, perform homogeneous redistribution of particle densities from fine to coarse grid.

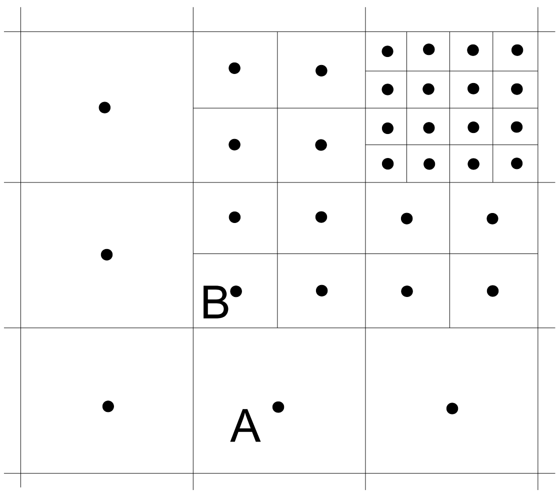
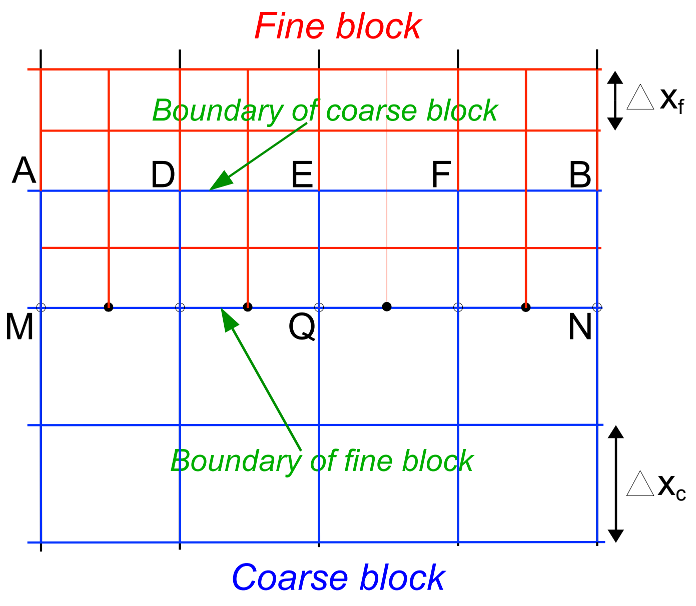


FIGURE 2.2: Cell-centered volumetric formulation to the locally embedded grid refinement. **A** stands for coarse cell and **B** for fine cell with a level of refinement = 2 [16].

This method can be considered quite robust in the sense that it conserves mass, avoids interpolation and rescaling of non-equilibrium part and shows good accuracy. However, it faces difficulty at the interface. Communication in the interface and the refinement of fine grid (also for time-step) doesn't allow it to enjoy good computational efficiency (although it is better than the regular square grid). The accuracy is dependent on the orientation of the flow with respect to the grid orientation. For flows perpendicular to the interface results in inaccurate solution. Also, it is to be noted that such method only allows cubic grids. It can not be taken as a method of choice in flows like turbulent channel flows where span-wise directions can be coarse than the normal direction. A better method should allow stretched grids or even unstructured grids.

2.2.3.2 Multi-block method

To further improve the method, Yu *et al.* [4] proposed a multi-block method. In this method, the domain is divided into blocks. Each block has a desired uniform grid



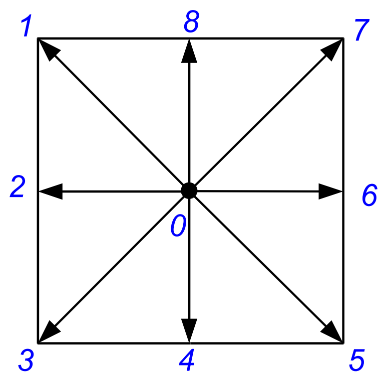


FIGURE 2.3: Representation of multi-block method focussing the interface between two blocks of different lattice spacing [4] (left); D_2Q_9 lattice arrangement supporting the explanation in this section about directions of populations' stream(right)

spacing and doesn't overlap with other blocks as shown in figure 2.3. So, the blocks communicate only through the interfaces between the blocks. The method argues that the conservation of mass and momentum is ensured with some special treatment at the interface.

$$\tilde{f}_i(c) = \tilde{f}_i^0(c) + \frac{\tau(f)-1}{n[\tau(c)-1]}\left(\tilde{f}_i(c) - \tilde{f}_i^0(c)\right)$$

$$\tilde{f}_i(f) = \tilde{f}_i^0(f) + n\frac{\tau(c)-1}{\tau(f)-1}\left(\tilde{f}_i(f) - \tilde{f}_i^0(f)\right)$$

where n is the ratio of lattice space between two-grid system. In order to highlight its advantage, it requires good communication at the interface. Referring to figure 2.3 , MN is the boundary for fine block *(f)* and AB is for coarse block *(c)*. It is understood that the boundary of coarse block is within the fine grid area and the boundary of fine block is within the area of coarse grid. Let us take a node of coarse grid (Q, which is also a boundary node of fine block) which needs to be updated smoothly to ensure proper interface communication. After the collision step at all lattice nodes (fine and coarse lattice nodes) with there corresponding relaxation time, it follows streaming step. At the streaming step, node Q obtains $\tilde{f}_i$ $(i = 6, 7, 8, 1, 2)$ components from neighbouring coarse nodes. It lacks $\tilde{f}_i$ $(i = 5, 4, 3)$ components (refer figure 2.3), which are now achieved from nodes D, E, F.

$$\tilde{f}_5(t + \triangle t, x_D) \to \tilde{f}_5(t + \triangle t, x_Q)$$

$$\tilde{f}_4(t + \triangle t, x_E) \to \tilde{f}_4(t + \triangle t, x_Q)$$

$$\tilde{f}_3(t + \triangle t, x_F) \to \tilde{f}_3(t + \triangle t, x_Q)$$

Similar process can be shown for the fine grid. There are some missing informations on the nodes on the boundary of the fine block (MN) represented by • (black dot). These missing nodes must be interpolated from the known ones at MN. To maintain the spatial symmetry at-least cubic spline interpolation along the interface is necessary. Apart from spatial interpolation, the method also requires temporal interpolation at all nodes in the boundary of fine block MN,

$$\tilde{f}_i(t + \frac{\triangle t}{2}, MN)$$

Yu *et al.* suggested three-point Lagrangian interpolation formula for temporal interpolation.

This method greatly improved on accuracy and computational efficiency and stands today as one of the most efficient refinement techniques in the LBM scenario. However, it still possesses interface (interface conserving mass and momentum) that can restrict the full potential in terms of computational efficiency. Also, it still can't be applied to stretched grids.

2.2.3.3 Quadtree grid

The other branch of development of such method is a LB method using quadtree grids (2D), octree grids (3D) proposed by Crouse *et al.* [5]. The idea was to automatically and dynamically generate grid based on the principle of *recursive decomposition* fulfiling certain criteria.

Basic unit of quadtree grids are called quadtree cells. There are two types of cells: parent cell and leaf cell. Parent cell can be subdivided into subcells, contrary to the leaf cell. Inside these subcells, scale functions are assigned at characteristic points that will control the grid density. These scale functions are prefixed or updated dynamically.

Let us take a 1 X 1 square domain with grid nodes 1, 2, 3, 4. Scale functions are prefixed inside the cell as shown in the figure 2.4. For example, we have taken scale functions as follows: scale for subcell 1 = 0.124, for subcell 2 = 0.24, for subcell 3 = 0.4 and for subcell 4 = 0.9. The scale function at center point or any characteristic point within the cell can be named $S1$. The length of the current cell can be named $S2$. If $S1 < S2$ then the cell is equally divided into subcells. The systematic divisions are as follows (refer figure 2.4):

Divide 1: All the predefined scales ($S1$) for square domain 1234 are less than $S2 = 1$. Therefore the cell is divided equally into four subcells.

Divide 2: Now, the new value of $S2$ is 0.5 as the square cell is divided. All the subcells except subcell-4 has scales $S1$ less than 0.5. Therefore subcell 1, 2 and 3 will be further subdivided.

Divide 3: The new value of $S2$ is 0.25. As scales of subcells 1 and 2 are only less than

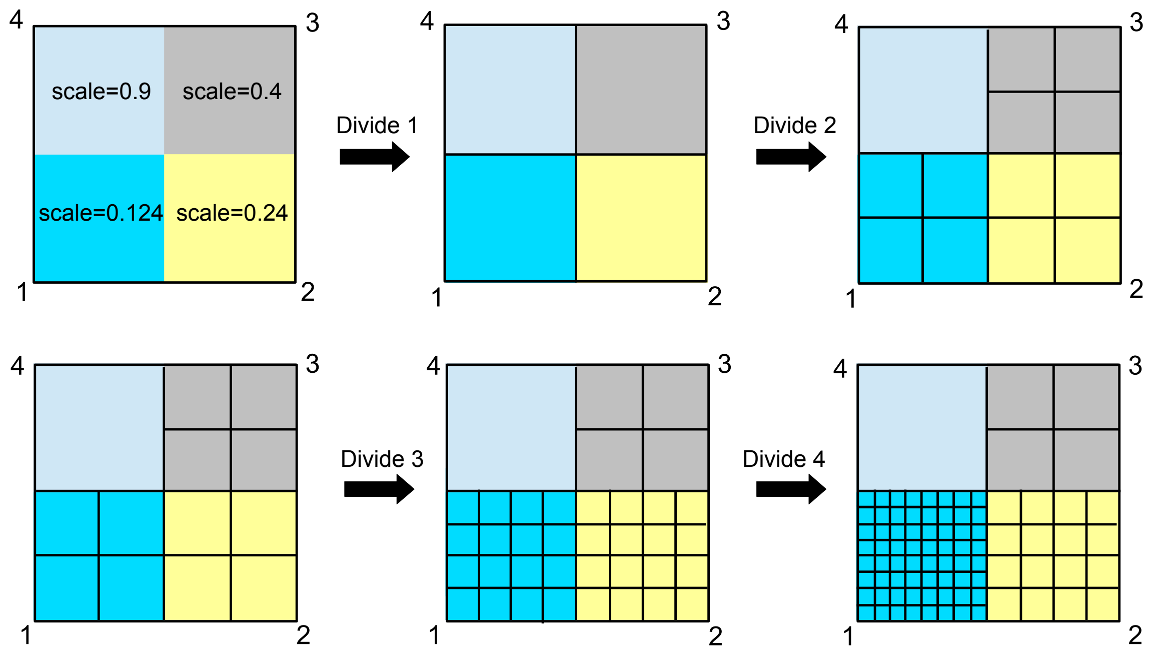


FIGURE 2.4: Procedure of generating a quadtree grid with predefined scale functions[17]

$S2$, only these subcells subdivide.

Divide 4: Finally, the only subcell with scale less than new $S2 = 0.125$ is subcell 1. Thus the subdivision takes place at subcell 1.

Inherently, this method also needs to deal with the communication at the interface between coarse and fine grids. Crouse *et al.* [5] used linear interpolation at the interface. This certainly is not enough to reach second order accuracy thus Geier *et al.* [84], Tölke *et al.* [85] suggested linear interpolation of second order moments to retain second order accuracy of LB scheme.

Recently Chen *et al.* [17] combined the benefits of He's [1] interpolation-supplemented LB model and quadtree grid and suggested linear interpolation with back-and-forth error compensation and correction (BFECC) method to achieve second order accuracy. This avoided interface treatment. The method implemented dynamic adaptive grids, without interface treatment and could be extended to unstructured grids. This helped to simulate flows with complex geometry. However, this idea using BFECC method suffered numerical diffusion (characteristic of He's method [1]). Also, the method demanded larger number of grid nodes for flows with asymmetrically placed geometry compared to multi-block method [4].

2.2.4 Off-lattice schemes

All the above methods tried *modifying* the streaming process. Off-lattice scheme tries to *replace* the streaming process.

Cao *et al.* [86] distinguished the physical symmetry and lattice symmetry in LBM. Physical symmetry is recognised as the symmetry involving the equilibrium distribution of velocities, discrete velocity space and moments and constraints of the velocity set that gives constraints for weighting coefficients and velocity vectors. Lattice symmetry is recognised as the specific directions of the distribution function hopping along characteristic lines to reach only to the nearest neighbouring node at one time-step. This constraint was due to the method of characteristics. Cao *et al.* [86] argued that physical symmetry is necessary to arrive at the correct macroscopic Navier-Stokes equation whereas lattice symmetry is not necessary. Therefore when we apply method of characteristics and follow *streaming* process, strict rule implies on the direction of velocity vectors and introduces a tight link between the physical symmetry and lattice symmetry (in a simple phrase: between mesh and lattice structure). However, these symmetries can be separated and still obtain correct macroscopic Navier-Stokes equation.
It is also important to note at this point that LBM follows the "molecular chaos" assumption such that particle-particle correlations are void, unlike LGA (Lattice Gas Automaton). Simply, this means the collision is local. This frees LBM from necessarily adopting the *streaming* process. This argument is also supported by the previous paragraph that the *streaming* process associated with lattice symmetry is not crucial to approach towards Navier-Stokes equations. So the coupling of lattice and mesh can be broken if we avoid method of characteristics and follow a different path. This means the distribution function can deviate from the characteristic line pathway. *Since the lattice is decoupled from the mesh, thus the name "off-lattice".*
This implies that the advection part should also include velocity space defined by $c+a_i\triangle t$ where a_i can be termed as the acceleration imparted on the i^{th} speed at a node. This velocity space was simplified due to the introduction of characteristic lines in equation 1.55 (equation of regular square grid).
After introducing the velocity space $(c + a_i\triangle t)$, equation 1.55 transforms to

$$\tilde{f}_i(x + c_i\triangle t, c + a_i\triangle t, t + \triangle t) - \tilde{f}_i(x, t) = \Psi_i$$

where Ψ_i represents the collision and the force term.

Now, the procedures adopted to address this acceleration term or synonymously to deviate from the characteristic lines or in a bigger view redefine the advection part, define off-lattice mesh refinement.
This was performed by burrowing the conventional CFD techniques. Mesh refinement efforts were done by incorporating *finite difference*, *finite element* and *finite volume* methods.

2.2.4.1 Finite Difference(FD) method

LBM has been considered as a special finite difference discretization case of the DVBE. It can be proved as follows:
Let us take DVBE (1.30) without force term:

$$\frac{\partial f_i}{\partial t} + c_{i\alpha} \cdot \frac{\partial f_i}{\partial x_\alpha} = -\frac{1}{\tau}\left(f_i - f_i^0\right)$$

Applying first order time discretization, first order upwind space discretization and downwind for collision term, the finite difference equivalent of the above equation follows [86]:

$$f_i(\mathbf{x}, t+\triangle t) = f_i(\mathbf{x}, t) - \frac{\triangle t}{\triangle x}[f_i(\mathbf{x}, t) - f_i(\mathbf{x} - c_i \triangle x, t)] - \frac{\triangle t}{\tau}[f_i(\mathbf{x} - c_i \triangle x, t) - f_i^0(\mathbf{x} - c_i \triangle x, t)]$$

If $\triangle t$ and $\triangle x$ are considered equal to 1, the equation becomes,

$$f_i(\mathbf{x}, t+1) = f_i(\mathbf{x} - c_i, t)] - \frac{1}{\tau}[f_i(\mathbf{x} - c_i, t) - f_i^0(\mathbf{x} - c_i, t)]$$

Shifting the reference frame by "$+c_i$" in space,

$$f_i(\mathbf{x} + \mathbf{c}_i, t+1) = f_i(\mathbf{x}, t) - \frac{f_i(\mathbf{x}, t) - f_i^0(\mathbf{x}, t)}{\tau}$$

This is the celebrated standard LBM equation without force term.
The above mentioned LB equation is due to a simple first order discretization. It can easily be extended to higher order discretization along with non-uniform meshes like in finite difference methods. Usually, different FD schemes can be applied to the gradient operator $\left(\frac{\partial f_i}{\partial x}\right)$ according to the area of application. For the time discretization, researchers usually choose from Euler scheme to different stages of Runge-Kutta scheme. Introduction of FD in LBM dates back to 1995 and before (as early as first efforts were done towards mesh refinement in LBM). Since then, numerous suggestions have been advanced in order to improve the accuracy and computational efficiency of LBM using FD method. Reider and Sterling [87] suggested fourth-order central difference scheme for space discretization and fourth-order Runge-Kutta method for temporal discretization. They compared results with standard LB and FD schemes applied to incompressible Navier-Stokes equations and showed convergence. Cao *et al.* [86] argued on the theoretical aspect that physical symmetry is only important to recover macroscopic Navier-Stokes and can be decoupled from lattice symmetry giving way for variety of options like non-uniform meshes and thermo-hydrodynamics. They also suggested semi-implicit collision schemes for stability. Mei and Shy [36] extended the method of Cao *et al.* [86] to curvilinear coordinates with non-uniform, body-fitted grids. Until this, works were

limited to explicit algorithms. Tölke *et al.* [37] improved the method with implicit algorithms. Attempts were also made to combine FD schemes with other mesh refinements in LBM (e.g. Kandhai *et al.* [38]). Further, to facilitate better stability criteria in low viscosity range and also perform better at discontinuities, Sofonea *et al.* [88] suggested flux limiters with FD LBM.

Similarly, suggestions and improvements can be abundantly lined up dealing with specific areas of application: Gou *et al.* [39], Xu *et al.* [40] (binary mixtures); Kim *et al.* [89] (fracture flow simulation); Watari *et al.* [90] (constructing thermal models because FD facilitates possibilities of having different sets of velocities); Kataoka *et al.* [91] (compressible flows); Sofonea *et al.* [92] (multiphase flows) etc.

By now, it is clear that FD LBM fulfils the purpose of facilitating mesh refinements by adding extra stability with the help of implicit schemes. It can also solve variety of fluid dynamics problems. However, this is achieved with a compromise in loss of simplicity of the original LBM. Also, this method is computationally inefficient because of its slow convergence. Not to forget that FD schemes do not always guarantee mass conservation and the method needs extra care at the discontinuities.

2.2.4.2 Finite Element(FE) method

Let us again recall DVBE(1.30) without force term:

$$\frac{\partial f_i}{\partial t} + c_{i\alpha} \cdot \frac{\partial f_i}{\partial x_\alpha} = -\frac{1}{\tau}\Big(f_i - f_i^0\Big)$$

To derive finite-element LB equation (refer [93]) from the DVBE, first the domain needs to be discretized into finite elements. These elements can be regular or irregular and wear different shapes like triangular or quadrilateral (2D) and tetrahedral, triangular prism (3D) etc. These finite elements are made of nodes. Variables associated with the element domain needs to be interpolated at its nodes. Thus each node is best described by the interpolation function and the nodal variable. Similarly for FE LBM, distribution function f_i can be distributed over x nodes as follows:

$$f_i = \sum_{x=1}^{n} H^x f_i^x = [H]\{f_i\}$$

where H^x is the interpolation function and f_i^x is interpolated variable at the x^{th} node. Using the above two equations, for each element the equation derives to:

$$[H]\left\{\frac{\partial f_i}{\partial t}\right\} + c_{i\alpha} \cdot \left[\frac{\partial H}{\partial x_\alpha}\right]\{f_i\} = -\frac{1}{\tau}[H]\Big(\{f_i\} - \{f_i^0\}\Big)$$

After applying general weighted residual formulation, the equation becomes

$$\sum \int_{S_e} \{w\}\Big([H]\Big\{\frac{\partial f_i}{\partial t}\Big\} + c_{i\alpha}\cdot\Big[\frac{\partial H}{\partial x_\alpha}\Big]\{f_i\} + \frac{1}{\tau}[H]\Big(\{f_i\} - \{f_i^0\}\Big)\Big)dS_e = 0$$

Here, the function is integrated over each finite element domain (S_e) and summed over all elements and "w" indicates the weighting function. Now the choice of the weighting function defines the type of method like Galerkin or collocation or least-squared or sub-domain or method of moments.
This method incorporated into LBM inherently enjoys great geometrical flexibility by the use of structured and unstructured mesh with refinements. However, the method is not very popular due to its difficulty in forming stable explicit schemes. Tricks can be applied to form stable schemes by transforming the first order hyperbolic DVBE (1.30) to second order parabolic equations but this will add unavoidable errors in the solution (refer [94]).
Some notable works on finite element LB method are [41–44].

2.2.4.3 Finite Volume(FV) method

Here is a brief introduction of the method. The following chapters will describe in detail. The basic idea of finite volume method is treating the advection part of DVBE using control volumes. To do so, this method involves a two-grid procedure. In this procedure, a coarse-grained *control grid* is introduced. At the center of the control grid (the easiest possible is a centroid), the macroscopic quantities evolve as seen in figure 3.3*a*. Behind the picture, this control grid contains several *uniform fine grid* [95]. These fine grids are the lattices where the original LBE dynamics occur. However the role of updating the macroscopic quantities is taken by the node of the *control grid*.

It can be described mathematically as follows. Starting with DVBE (1.30) without force term:

$$\frac{\partial f_i}{\partial t} + c_{i\alpha}\cdot\frac{\partial f_i}{\partial x_\alpha} = -\frac{1}{\tau}\Big(f_i - f_i^0\Big)$$

The above equation is integrated over a volume V (of surface S) and after applying flux theorem we get,

$$\int_V \frac{\partial f_i}{\partial t}\,dV + \int_S \mathbf{c}_i\cdot\mathbf{n}\,f_i\,dS = \int_V -\frac{1}{\tau}(f_i - f_i^0)\,dV$$

Chapter 3

Finite Volume Approach to LB equation

3.1 Motivation for Finite Volume (FV) Approach

Although all mesh refinement techniques were developed with certain advantageous characteristics, we have seen important drawbacks of all such reformulations. Compared to the original so-called *streaming based* implementation they are characterised by,

- increased complexity
- higher computational cost
- stability limitations
- can't guarantee mass conservation
- accuracy relied on interpolation
- interface treatment

Presently the best way seems to be the locally embedded grid method, which has allowed to simulate turbulent channel flows [96] and even more complex flow geometries [97]. However in this case, the advantage is limited in terms of accuracy and efficiency compared to state-of-the-art Direct Numerical Simulation (DNS) *e.g.* spectral methods. We need something better than locally embedded grid method to compete with the state-of-the-art. If we overview the history of mesh refinements section, the notion of better mesh refinement attempts seem to converge towards a method that obeys conservation laws and is stable. This suggests us a pathway to adopt finite volume approach. Also, finite volume approach allows stretched grids and avoids interface treatment which is a big plus. It can be a competitive method if we can assure it to be accurate, computationally efficient and has good stability range.

3.2 Chronological development

Development of FV algorithm took more than two decades to reach a good maturity level as it had to rival with a very simple and efficient standard Lattice Boltzmann (LB) algorithm to stand a chance. The development of the algorithm in chronological order is as follows: The idea of using a finite volume method to decouple the spatial numerical mesh from the velocity lattice structure was first proposed by Nannelli and S. Succi, [6] (see also the 1992 review by Benzi *et al.* [98]), in this seminal paper a low-order upwind scheme was suggested for the discretization of the advection (or flux) term. The idea was further refined in Amati *et al.* [7], where piece-wise linear interpolation scheme was suggested for the treatment of the flux term. While these first works were limited to stretched Cartesian grids, Chen [8] presented a volumetric formulation, based on a cell-centered discretization scheme, which allowed for the adoption of arbitrary *structured* meshes. The formulation was further developed by Peng *et al.* [9, 10] through a cell-vertex FV scheme, which displayed enhanced stability properties. More recently Ubertini *et al.* [11] addressed the problem of *unstructured* bidimensional triangular meshes, which allow great flexibility on one hand, but also reintroduce known issues related to numerical stability. To improve on stability and accuracy, Patil *et al.* [18] suggested Total Variation Diminishing (TVD) formulations for FV algorithm on unstructured mesh but overlooking the computational cost as these schemes are heavy. This was further refined in a work by Zarghami *et al.* [12] through a cell-centered FV approach on arbitrary mesh in two dimension. Despite all these contributions, at present the situation is still far from being solved. If on one side it has been shown that a satisfactory level of precision can be reached by the FV method on the other hand this is often at the price of the high computational costs needed to obtain a stable algorithm. As an example in a recent work [12], where a series of laminar but relatively complex flows over non-homogeneous meshes were simulated, a fifth-order Runge-Kutta scheme had to be adopted for the time discretization in order to have stable results. The consequence on the computational cost is evident since in such a scheme the advection terms of DVBE need to be computed five times per time step (while in the ST method it is performed just by means of a memory shift, the streaming). As a consequence the FV LB is rarely a method of choice in fluid-dynamics simulations (see also the discussion in [99]). Systematic comparisons of two algorithms in terms of accuracy and efficiency is reported by Prestininzi *et al.* [100] but for a specific context of shallow water flow with complex geometry.

The above mentioned ideas were mainly focussed on FV approach implemented to isothermal flows. In an attempt to simulate compressible thermal flows, Sbragaglia and Sugiyama [101] used multi-speed thermal model which obtains a temperature evolution equation by introducing higher number of velocity set (*e.g.* D_2Q_{81}) in the equilibrium distribution function. Bigger velocity set compared to standard lattice is required to acquire acceptable accuracy. Clearly, large number of velocity terms reduce flexibility and also pose problems in implementing boundary conditions. Sbragaglia and Sugiyama [101] coupled this approach with Peng's scheme to remove the necessity of space-filling velocity set and also increase geometrical flexibility.

3.3 Parameters influencing the FV LB development

The above section explains the development of FV LB in chronological order. Each major step of development of the algorithm focussed on either *accuracy* or *stability* or *computational efficiency* or all. So, to better understand the current status of the FV algorithm it is useful to group the developments in terms of these three major categories. The aim of this effort is to group different proposals, to be able to understand the advantages and shortcomings of each ideas and develop guidelines for an advanced algorithm to compete with the state-of-the-art. The study can be made more clear if these categories can be further divided. The divisions are based on the parameters that play role in the outcome of the category (as seen in figure 3.1).

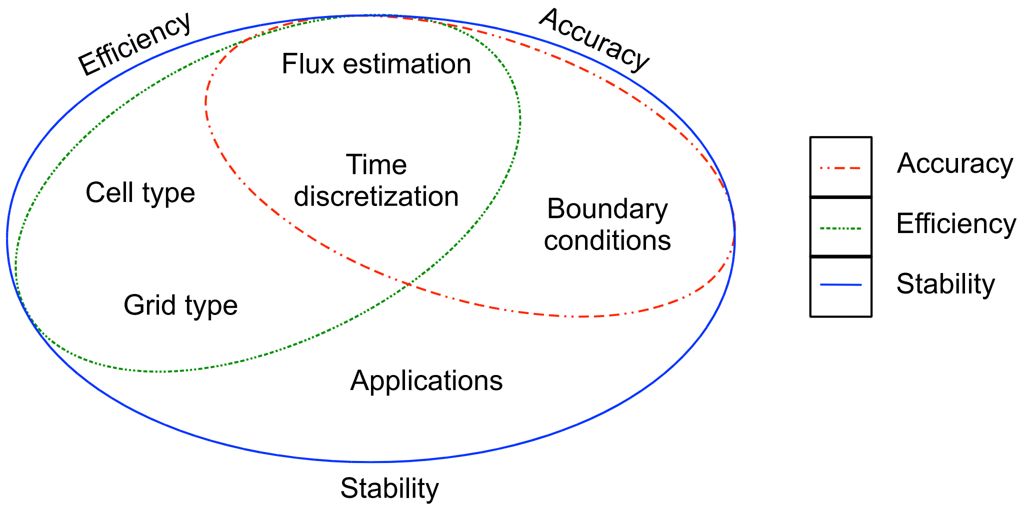
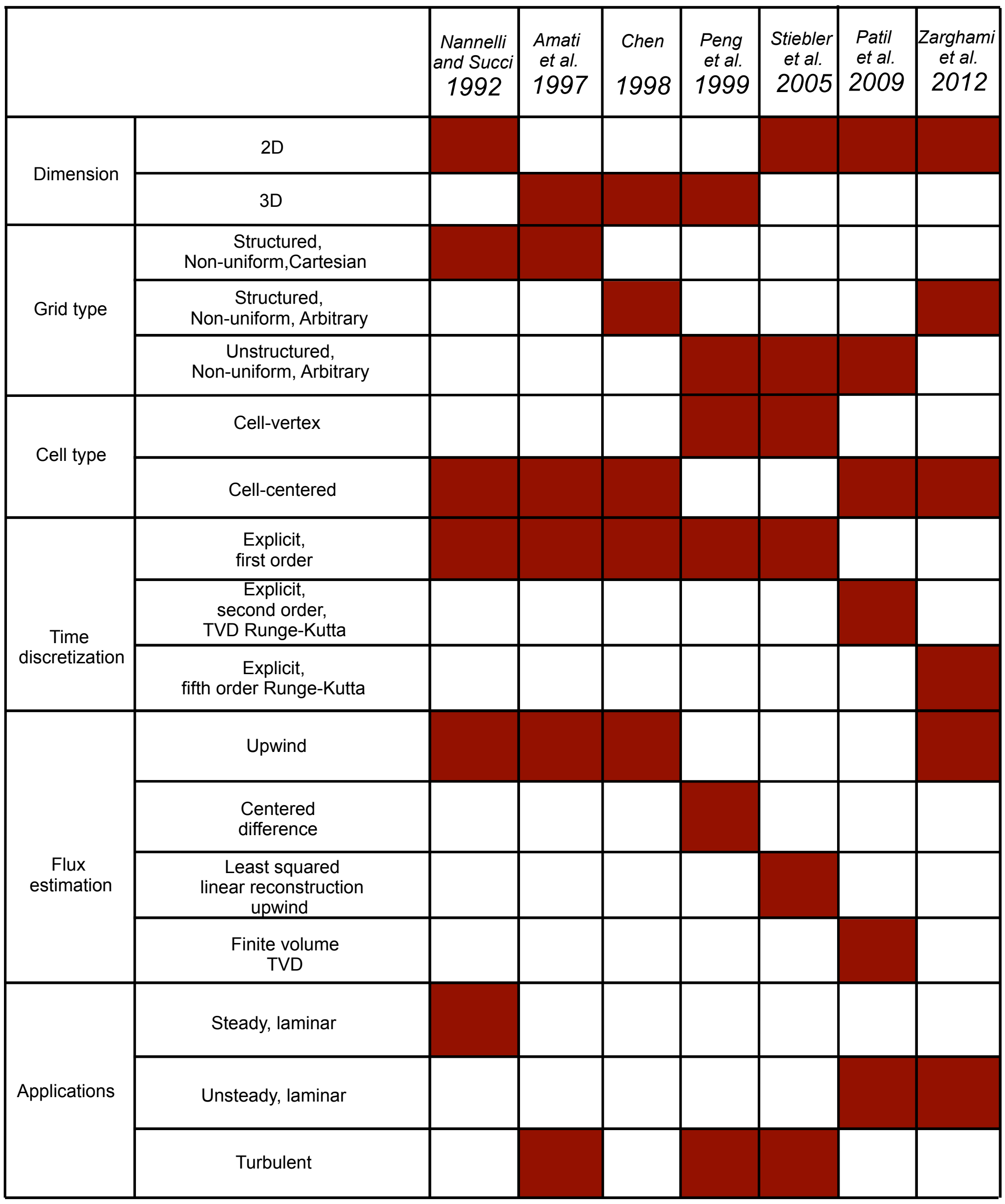


FIGURE 3.1: Venn diagram showing different parameters grouped into the main three categories (*accuracy* - red circle, *efficiency* - green circle, *stability* - blue circle). Parameters in a certain category would mean that all these parameters would influence the outcome of the category. *For example:* Parameters like flux discretization, time discretization and boundary conditions fall under the category *Accuracy*. Thus, these parameters play role in determining the accuracy of the method.

Also, a table describing the role of these parameters in the development of FV LB Approach is presented in figure 3.2. The proposals are listed column-wise in chronological order and the parameters influencing the FV LB development are listed row-wise. The red-box at the intersection of these rows and columns would mean that the certain proposal is characterised by the following parameters. *For example:* In the column of *Nanelli and Succi 1992*, there are certain number of red boxes representing the parameters which characterise their proposed method. Based on the table, it can be deduced that they proposed a 2D FV LB approach for structured, non-uniform, Cartesian grid. The type of the cell is cell-centered. The method used explicit, first order scheme for time discretization and upwind scheme for flux estimation. They tested their method with a steady, laminar flow (Poiseuille flow).



		Nannelli and Succi 1992	*Amati et al.* 1997	*Chen* 1998	*Peng et al.* 1999	*Stiebler et al.* 2005	*Patil et al.* 2009	*Zarghami et al.* 2012
Dimension	2D	🟥				🟥	🟥	🟥
	3D		🟥	🟥	🟥			
Grid type	Structured, Non-uniform,Cartesian	🟥	🟥					
	Structured, Non-uniform, Arbitrary			🟥				🟥
	Unstructured, Non-uniform, Arbitrary				🟥	🟥	🟥	
Cell type	Cell-vertex				🟥	🟥		
	Cell-centered	🟥	🟥	🟥			🟥	🟥
Time discretization	Explicit, first order	🟥	🟥	🟥	🟥	🟥		
	Explicit, second order, TVD Runge-Kutta						🟥	
	Explicit, fifth order Runge-Kutta							🟥
Flux estimation	Upwind	🟥	🟥	🟥				🟥
	Centered difference				🟥			
	Least squared linear reconstruction upwind					🟥		
	Finite volume TVD						🟥	
Applications	Steady, laminar	🟥						
	Unsteady, laminar						🟥	🟥
	Turbulent		🟥		🟥	🟥		

FIGURE 3.2: Summary of the development of FV LB Approach

Each influencing parameter has been presented in sections below.

3.3.1 Cell type

The basic element in a Finite Volume approach is a control volume (CV) or a cell. Each control volume is bounded by vertices, edges and faces. These control volumes divide the space in a structured or unstructured way. Structured mesh is identified by regular connectivity and the control volume is usually quadrilateral-shaped whereas unstructured mesh is identified by irregular connectivity and the control volume is usually triangular-shaped. All these control volumes sum-up to fill the whole domain.

For computation purposes, it is complicated to consider the values of macroscopic quantities at all locations inside the control volume. Therefore the simplest approximation is to assume the value constant inside the control volume and the constant value is assigned to a reference location. This reference location is known as *node* where the macroscopic quantities evolve in time. Now, based on the position of the node all the FV algorithms can be divided into *cell-vertex* and *cell-centered* type.

3.3.1.1 Cell-centered type

The first idea of FV approach saw cell-centered type in the work of Nanelli and Succi [6]. This paper presents the idea of two grid procedure. In this procedure, a coarse-grained *control grid* is introduced. At the center of the control grid (the easiest possible is a centroid), the macroscopic quantities evolve as seen in figure 3.3*a*. Behind the picture, this control grid contains several *uniform fine grid*. These fine grids are the lattices where the original LBE dynamics occur. However the role of updating the macroscopic quantities is taken by the node of the *control grid*.

It can be described mathematically as follows: We know that if we integrate DVBE (1.30) over a volume V (of surface S) and after applying flux theorem we get,

$$\frac{\partial f_i}{\partial t} + c_{i\alpha} \cdot \frac{\partial f_i}{\partial x_\alpha} = -\frac{1}{\tau}\left(f_i - f_i^0\right)$$

$$\int_V \frac{\partial f_i}{\partial t}\, dV + \int_S \mathbf{c}_i \cdot \mathbf{n}\; f_i\; dS = \int_V -\frac{1}{\tau}(f_i - f_i^0)\; dV$$

In a cell-centered type, we then assume that every term in the volume integrals can be considered as constant and its magnitude taken at a reference location called *node* inside V. The term in the surface integral however, carries some spatial variability. When such a surface is decomposed in M faces (as in a structured grid of nodes with connectivity index M) it is convenient to make the assumption that f_i is constant on each of the S_j surfaces perpendicular to $\mathbf{n_j}$ and denoting its value with $[f_i]_j$. This altogether leads to:

$$\frac{\partial f_i}{\partial t} + \frac{S_j}{V}\mathbf{c}_i \cdot \mathbf{n}_j \, [f_i]_j = -\frac{1}{\tau}(f_i - f_i^0) \tag{3.1}$$

where summation over the repeated index j is applied. Here it can easily be noted that only the advection term is modified to represent finite volume approach.

Referring to the above equation 3.1, f_i is the mean distribution function of the *coarse-grained cell* and not of the *lattice*. Advection of these distribution functions are performed by the inflow/outflow of flux at the boundary of CV as seen in fig 3.3*a*. These fluxes are displaced in fractional amounts according to the number of particles crossing the boundary. Types of interpolation to estimate these fluxes determine the accuracy of the FV algorithm which will be discussed in a separate section. Collision integrals are also performed at these cell-centered nodes. Similar cell-type was used by [12, 18, 100].

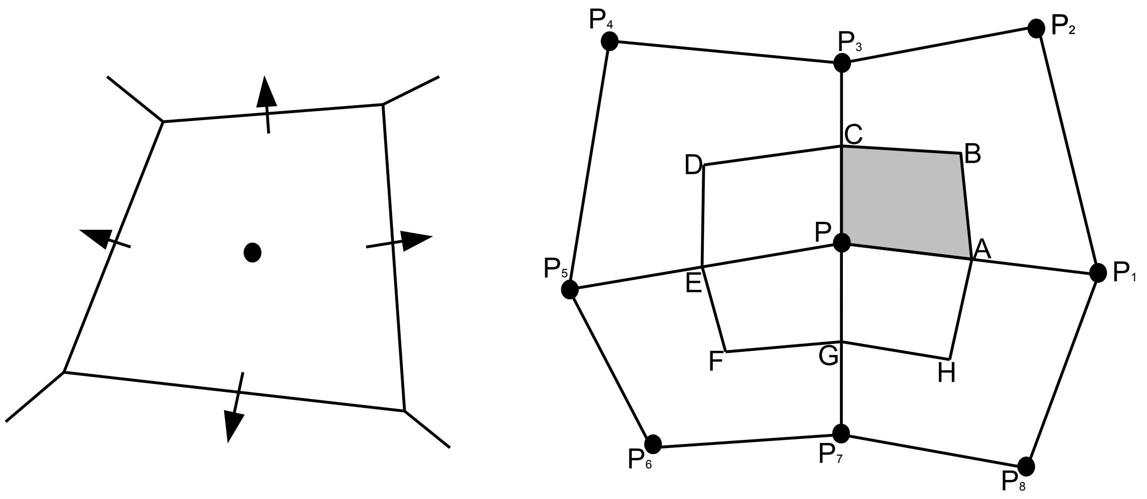


FIGURE 3.3: Typical cell-types used for a FV LB with quadrilateral elements *(a)* cell-centered type [7] *(b)* cell-vertex type [10]. Black dot represents the nodes where macroscopic quantities evolve. In cell centered type it is named control grid node whereas in cell-vertex type it is named primal grid node.

3.3.1.2 Cell-vertex type

With an idea of providing greater geometrical flexibility, Peng *et al.* [9] introduced cell-vertex type to the FV LB. This type allowed unstructured mesh with great ease. As shown in figure, the CVs were built around the grid nodes in 2D. Both triangular and quadrilateral elements are possible and have been demonstrated by [9, 10]. In this context, we use a cell-vertex type with a quadrilateral element. There are two types of grids (refer figure 3.3*b*): primal grid (consisting of grid nodes) and dual grid (consisting of nodes forming CV around a grid node). Primal nodes are the actual nodes where the macroscopic quantities evolve and dual grid nodes are used to form the control volume around a primal node (say P from the figure 3.3b). Distribution functions at primal nodes are known and using these knowns, unknown distribution functions at dual nodes are interpolated using bilinear interpolation.

We said that the control volume is formed around the grid node P. However this in not sufficient. A smart choice of dual grid nodes can simplify the problem a lot. As in figure 3.3b, the simplest choice would be $ABCDEFGH$ where A is a midpoint of edge PP_1. Similarly, points C, E, G are midpoints of PP_3, PP_5, PP_7 respectively. Points B, D, F and H are the geometric centers of elements $PP_1P_2P_3$, $PP_3P_4P_5$, $PP_5P_6P_7$ and $PP_7P_8P_1$ respectively.

$$x_A = \frac{x_P + x_{P_1}}{2}$$

$$x_B = \frac{x_P + x_{P_1} + x_{P_2} + x_{P_3}}{2}$$

$$x_C = \frac{x_P + x_{P_3}}{2}$$

It can be seen from the figure 3.3b that control volume $ABCDEFGH$ can be subdivided into 4 quadrilateral elements: $ABCP$, $PCDE$, $PEFG$, $PGHA$. For simplicity, these elements can be treated separately and later added. Let us take an element $ABCP$. Now, let us apply DVBE(1.30) (without the forcing term for simplicity) to this element. The first term of the equation takes the form,

$$\int_{ABCP} \frac{\partial f_i}{\partial t} d\sigma = \frac{\partial f_i(P)}{\partial t} S_{ABCP}$$

where $f_i(P)$ means distribution function at node P and S_{ABCP} is the surface area of element $ABCP$. To further simplify the treatment, f_i is assumed constant all over S_{ABCP} which is a general practice in finite volume methods.

The second term of DVBE represents advection. Finite volume techniques involve fluxes across edges to carry-out the process. So, the second term of 1.30 takes the form,

$$\int_{ABCP} c_{i\alpha} \frac{\partial f_i}{\partial x_\alpha} d\sigma = \mathbf{c}_i \cdot \left(\int_{AB} f_i d\mathbf{l} + \int_{BC} f_i d\mathbf{l} \right) + I$$

where I is the fluxes from the internal edges CP and PA and $\boldsymbol{l}$ represents length of edges. To note is that the internal edges CP and PA will get repeated when we treat other quadrilateral elements like $PCDE$, $PEFG$ and $PGHA$ and eventually cancel out. Now, assuming bi-linearity of the distribution functions along the edge lengths, the integrals of the *R.H.S.* of the above equation can be written as,

$$\int_{ABCP} c_{i\alpha} \frac{\partial f_i}{\partial x_\alpha} d\sigma = \mathbf{c}_i \cdot \mathbf{n}_{AB}\, l_{AB}\, \frac{f_i(A) + f_i(B)}{2} + \mathbf{c}_i \cdot \mathbf{n}_{BC}\, l_{BC}\, \frac{f_i(B) + f_i(C)}{2} + I$$

where $\mathbf{n}_{AB}$ and $\mathbf{n}_{BC}$ are the normal vectors to edges AB and BC respectively; and $f_i(X)$ represent distribution function at that point X. At this point, f_i and f_i^0 of dual grid nodes are interpolated from the primal grid nodes. For example $f_i(A)$ is interpolated from $f_i(P)$ and $f_i(P_1)$.

$$f_i(A) = \frac{f_i(P) + f_i(P_1)}{2}$$

Similarly, assuming bi-linearity of the distribution functions, collision term in DVBE takes the form,

$$\int_{ABPC} \frac{1}{\tau}(f_i^{neq}) = -\frac{S_{ABPC}}{16\tau}(9f_i^{neq}(P) + 3f_i^{neq}(P_1) + f_i^{neq}(P_2) + 3f_i^{neq}(P_3))$$

where f^{neq} represents non-equilibrium distribution function $f^{neq} = f - f^0$.
This completes the treatment of a quadrilateral element $ABPC$. It has to be similarly repeated over other quadrilateral elements of the control volume $ABCDEFGH$. Finally, summing up over the control volume, $f_i(P)$ can be updated as,

$$f_i(P, t + dt) = f_i(P, t) + \frac{dt}{S_p}\left(\sum_{around\ P} collisions - \sum_{around\ P} fluxes\right)$$

This explains all the steps required for a cell-vertex type FV algorithm. This method was followed by [9–11, 99, 101] as it is a good technique to increase geometrical flexibility. However the method suffers from numerical stability limitations and most importantly it requires large amount of book-keeping. Thus, the interest again shifted back to cell-centered type, as it is an easy technique to reduce memory storage.

3.3.2 Grid type

The main aim of FV algorithm was to incorporate non-homogeneous lattices according to the demand of the flow in the domain. It is easy to imagine that the domain, in order to improve on geometrical flexibility, demands variety of non-uniform grid types. Some examples are structured, Cartesian, non-uniform grid; structured, arbitrary, non-uniform grid; unstructured, arbitrary, non-uniform grid. The choice of the grid type is not only dependent on the required geometrical flexibility but also on a good compromise with other important features like accuracy, efficiency and stability. Most importantly, this is more related to the parallel computing efficiency of LB equation and shouldn't be compromised at all. Also, it is important to push the algorithm to three-dimensional (3D) grid to perform more realistic simulations.
Nannelli and Succi [6] proposed FV algorithm with 2D, structured, Cartesian, non-uniform grid. Amati *et al.* [7] extended the situation to 3D. Chen [50] generalised the algorithm extending its application to structured, arbitrary, 3D mesh naming it a

"supergrid" lattice. Peng *et al.* [9] proposed a cell-vertex technique that allowed 2D, unstructured, arbitrary grid. Triangular elements [9] and also quadrilateral elements [10] could be easily realized. Xi *et al.* [102] extended the technique to 3D and the same technique was followed by Ubertini *et al.* [11], Stiebler *et al.* [99], Sbragaglia and Sugiyama [101] proving their advancement in other features of the algorithm. While doing the same, Zarghami *et al.* [12] used 2D, structured, arbitrary grid emphasising the potentials of their algorithms that could still be improved in terms of geometrical flexibility.

3.3.3 Flux estimation

In the FV approach, advection transport flux estimation across the cell-surfaces takes the center stage. This is a prominent factor in deciding the order of accuracy of the method, be it a cell-vertex type or cell-centered type with triangular or quadrilateral elements. For clarity, we address a cell-centered type grid. Let us take only the advection term from equation 3.1,

$$\textit{Advection term}: \boxed{\frac{S_j}{V}\mathbf{c}_i \cdot \mathbf{n}_j \, [f_i]_j}$$

Here, $\frac{S_j}{V}\mathbf{c}_i \cdot \mathbf{n}_j$ is the projection of the boundary surface S_j along the velocity direction c_i, where j stands for the location of the surface (like east, west, north, south, top, bottom). S_j refers to surface area (3D) or edge length (2D). The other part of the above expression is $[f_i]_j$ which is associated with the reconstruction operator. Reconstruction operator expresses the f_i values at the boundary based on the neighbouring nodal values. This calls for an interpolation which determines the order of the accuracy of the method. Also it can't be forgotten that since interpolation requires nodal values from neighbouring nodes, it looses locality which was considered one of the main features of standard LBM.

In the first paper of FV LB, Nannelli and Succi [6] used a piecewise constant reconstruction operator and upwind spatial differencing scheme (refer figure 3.4a).

$$f_i(x_s) = f_i(x_\alpha)$$

x_α depending on the direction of velocity. Also to note is that x_s represents the position of the cell boundary and x_α represents position of cell node. This is the lowest order scheme which is consistent with Navier-Stokes equation in the continuum limit (see the discussions by Nannelli and Succi [6]). As upwind scheme takes only the upstream nodal value of the cell to the boundary in the direction of the movement of particles, this is

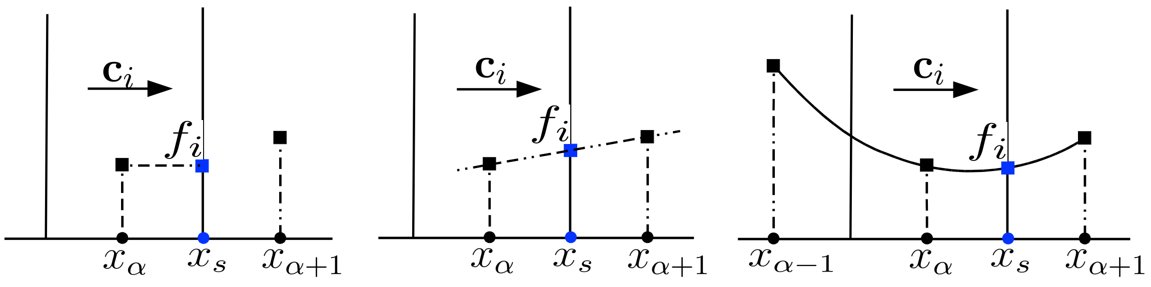


FIGURE 3.4: Sketch of different interpolation techniques for estimating f_i value at the cell boundary x_s. *(a)* piecewise constant *(b)* piecewise linear *(c)* piecewise quadratic. Note that the sketch emphasizes only the propagation in the right direction with population velocity c_i.

closest to the standard LB streaming process. However, piecewise constant reconstruction operator add significantly the numerical viscosity. Thus, they further suggested piecewise linear reconstruction operator which improved the case and was also followed by Amati *et al.* [7] (refer figure 3.4*b*).

$$f_i(x_s) = f_i(x_\alpha) + \nabla f_{i,\alpha} \cdot (x_s - x_\alpha) \tag{3.2}$$

Chen [50], while formulating a theory of FV approach for general mesh, also pointed out that to realise up-to the correct viscous order hydrodynamics, the interpolation should be at-least second order accurate. This also took account of the gradients at the cell node like in the equation 3.2.

As we saw in the cell-type section, Peng *et al.* [9] introduced cell-vertex type which involved bilinear interpolation for the flux estimation. It was a centered difference scheme ($\gamma = 2$ considering the equation below).

$$f_i(x_\alpha, x_{\alpha+1}) = \gamma f_i(x_\alpha) + (1-\gamma) f_i(x_{\alpha+1}) \tag{3.3}$$

This was also followed by Ubertini *et al.* [11] although it was observed to be numerically unstable. Stiebler *et al.* [99] pointed out that central difference schemes produced physical oscillations in the solution although it was second order accurate. Therefore they proposed least squared linear reconstruction (LSLR) upwind scheme for flux estimation. This is similar to equation 3.2 involving the computation of gradient. The gradient was computed by selecting the one that minimized the total squared error.

$$\min_{\nabla f_i(x_\alpha)} \sum_m w_m [f_i(x_m) - f_i(x_\alpha) - \nabla f_i(x_\alpha) \cdot (x_m - x_\alpha)]^2 \tag{3.4}$$

where m is the neighbouring centroids and $w_{\alpha,m}$ is a geometric weighting factor given by $\frac{1}{(x_m - x_\alpha)^2}$.

This method was computationally expensive by about 50% because of the gradient estimation step. A proof of concept was demonstrated for cell-vertex type grid in a triangular grid.

Patil *et al.* [18] recognized that the method proposed by Stiebler *et al.* [99] had difficulty in simulating flows with large local gradients as it predicted under/over-shoots apart from low computational efficiency. Thus, they proposed a better method dividing it in three layers: *(i)* Roe's splitting scheme for flux estimation at the edge/surface; *(ii)* Total Variation Diminishing (TVD) and limiter functions for solution reconstruction; and *(iii)* least squared method for gradient reconstruction (similar to Stiebler *et al.* [99]). In their method, they assumed the cell boundary to be a discontinuity thus approximating as a Reimann problem. Similar to the Riemann problem, at the cell boundary there exists two distinct left and right states of the solution, f_i^L and f_i^R as shown in figure 3.5*a*. If flux density, $F(f_i) = (\mathbf{c}_i \cdot \mathbf{n})f_i$ is considered, then flux density at the cell boundary between nodes A and B was calculated using Roe's splitting scheme,

$$F(f_i^L, f_i^R, \mathbf{n}_{AB}) = \frac{1}{2}[F(f_i^R) + F(f_i^L) - |\eta(AB)|(f_i^R - f_i^L)] \tag{3.5}$$

where $\mathbf{n}_{AB}$ is the normal vector to the boundary between the cells A and B; and $|\eta(AB)|$ is the scaled characteristic speed normal to the boundary. Fortunately for this case, these two values of F are linear along the boundary.

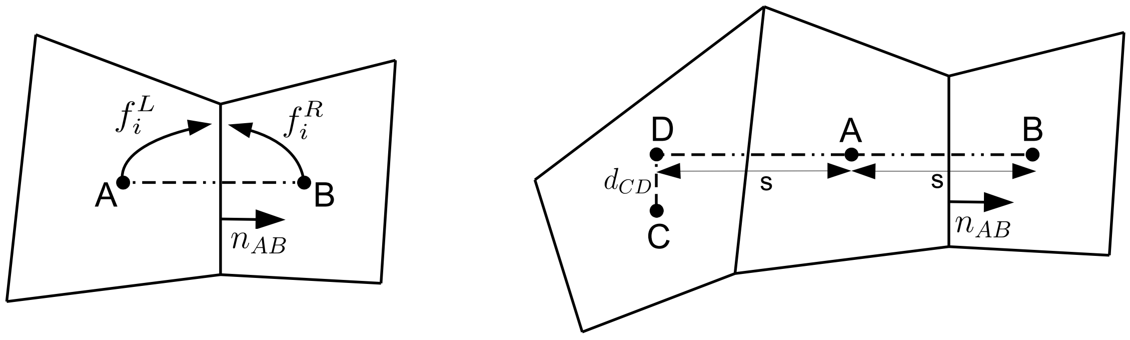


FIGURE 3.5: *(a)* Calculation of advective fluxes approximating as a Riemann problem *(b)* Finding a virtual upwind node D in a cell with centroid C for the calculation of $r - factor$. For demonstration, propagation is considered in the right direction.*(adapted from Patil et al. [18])*

Now these f_i^L and f_i^R needs to be calculated and was done using TVD and limiter functions. This means second term in the *R.H.S.* of equation 3.2 was replaced with a new term that contained a nonlinear function ϕ. $\phi(r)$ is known as flux limiter and is a function of consecutive gradient. This helped to adapt to the solution, thus increasing the stability. If we take two cells A and B, f_i^L and f_i^R are given by,

$$f_i^L = f_{i,A} + \frac{1}{2}\phi_{AB}(f_{i,B} - f_{i,A})$$

$$f_i^R = f_{i,B} + \frac{1}{2}\phi_{AB}(f_{i,A} - f_{i,B})$$

We still have to determine ϕ. There are various flux limiters which have been extensively used in the conventional CFD methods *e.g* minmod, superbee etc. Any suitable one can be used here. These flux limiters are dependent on *r-factor*. They suggested a high-order reconstruction to calculate the *r-factor* involving three consecutive nodes for an unstructured mesh (It can also be easily represented in a structured mesh). Suppose three cells A,B and C (refer figure 3.5*b*),

$$r = -\frac{\mathbf{d}_{CD} \cdot \nabla f_{i,C}}{f_{i,B} - f_{i,A}} + \frac{f_{i,A} - f_{i,C}}{f_{i,B} - f_{i,C}}$$

where $\mathbf{d}_{CD}$ is the vectorial distance between C and D. Here, gradient $\nabla f_{i,C}$ is calculated using the least squared method for gradient reconstruction suggested by Stiebler *et al.* [99]. It completes their method of estimation of flux. This improved the stability and accuracy of FV algorithm greatly but at the cost of high computation.
Few years later, Zarghami *et al.* [12] suggested upwind scheme employing second-order pressure based biasing factor in the convective fluxes with an aim of improving stability with reduced cost. Although it can reduce the computational cost compared to the previous method, in the authors' view this can't be as stable and as accurate as the method by Patil *et al.* [18].

3.3.4 Time discretization

Few possibilities have been tried lately in the temporal discretization of FV algorithm only after the suggestions by Patil *et al.* [18]. Until then, time marching step was carried out in a simple first order, forward-difference, explicit Euler scheme. In such case equation 3.1 can be re-written as,

$$f_i^{(t+\Delta t)} = f_i^{(t)} - \Delta t \frac{S_j}{V} \mathbf{c}_i \cdot \mathbf{n}_j \left[f_i^{(t)} \right]_j + \frac{\Delta t}{\tau} (f_\alpha^{0\,(t)} - f_i^{(t)}) \qquad (3.6)$$

Other suggestions made by Patil *et al.* were second-order, explicit Adams-Bashforth scheme; second-order, central-difference, explicit leapfrog scheme and second-order TVD Runge-Kutta scheme. Zarghami *et al.* [12] adopted fifth-order Runge-Kutta scheme to make his algorithm stable but loosing on computational efficiency.

3.3.5 Stability issues

Stability is an important aspect of an algorithm. Many factors can influence the stability of the algorithm like discretization methods, boundary conditions and CFL number.

One of the stability criteria due to the explicit Euler time-stepping can be expressed by,

$$0 < \triangle t \leqslant 2\tau$$

Proof: *Let us suppose the LB equation without the advection and the forcing terms and assume f^0 to be constant. Then the reduced LB equation with time dependency only to the non-equilibrium distribution function writes as,*

$$\frac{df^{neq}}{dt} = -\frac{1}{\tau} f^{neq}$$

After time step $\triangle t$,

$$f^{neq\ (t+\triangle t)} = f^{neq\ (t)} - \frac{\triangle t}{\tau} f^{neq\ (t)}$$

$$f^{neq\ (t+\triangle t)} = \left(1 - \frac{\triangle t}{\tau}\right) f^{neq\ (t)}$$

And after N times $\triangle t$ time-steps we get,

$$f^{neq\ (t+N\triangle t)} = \left(1 - \frac{\triangle t}{\tau}\right)^N f^{neq\ (t)}$$

We have discussed in section 1.4.2, if a dilute gas with arbitrary initial conditions is allowed with molecular interactions, the gas in course of time will definitely reach equilibrium state. It can also be re-stated in this context that the non-equilibrium distribution function should reach 0. This is true only if the following conditions are satisfied:

- *If N is sufficiently long enough.*
- $|1 - \frac{\triangle t}{\tau}| < 1$

Therefore for the equation ($\lim_{N->\infty} (1 - \frac{\triangle t}{\tau})^N f^{neq\ (t)} = 0$) to be true, there exists a strict constraint:

$$-1 < 1 - \frac{\triangle t}{\tau} < 1$$

$$-2 < -\frac{\triangle t}{\tau} < 0$$

$$2 > \frac{\triangle t}{\tau} > 0$$

It has been shown that if we discard the advection and the forcing terms and assume f^0 to be constant, the stability region of the method is $0 < \Delta t \leq 2\tau$.

This influence of collision term on the stability criteria was a serious constraint in efficiently simulating flows with low viscosity as it puts tight bound on the maximum allowed time-step $\triangle t_{max}$. Zarghami *et al.* [12] tried fifth-order Runge-Kutta time-stepping scheme but didn't prove significant improvement on this stability region. Patil

et al. [18] suggested second-order TVD Runge-Kutta scheme and claimed to improve the time-step by 2.1 times greater than that of Euler explicit scheme.
It is also important to note that the approximation of flux for the advection term plays a part in the stability of the scheme. In the process of improving flux estimation by first-order upwind schemes, central difference schemes were proposed by Peng *et al.* [10] This was pointed out by Stiebler *et al.* [99] to be numerically unstable and proposed a LSLR upwind scheme. Similarly, Patil *et al.* [18] proposed finite volume TVD scheme with the purpose of improving stability and accuracy.
Other criteria is put by CFL number which should strictly be less than or equal to 1.

$$CFL = c_i \frac{\Delta t}{\Delta x} \leqslant 1$$

Patil *et al.* presented a CFL criteria for a general 2D unstructured grid as,

$$CFL = \frac{\triangle t \; (|c_i| + c_0)\left(l^x_{min} + l^y_{min}\right)}{A_{min}}$$

where A_{min} is the minimal area of the cell and l^x_{min} and l^y_{min} are the projected lengths of the cell-area on x and y directions respectively. This can be easily extended for a general 3D unstructured grid as,

$$CFL = \frac{\triangle t \; (|c_i| + c_0)\left(A^x_{min} + A^y_{min} + A^z_{min}\right)}{V_{min}}$$

where V_{min} is the minimal volume of the cell and A^x_{min}, A^y_{min} and A^z_{min} are the projected surface areas of the cell-volume on x, y and z directions respectively.

Regarding the cell type, it is also important to note that the cell-centered type was more stable than the cell-vertex type. Thus, the research shifted back to cell-centered type.

3.3.6 Efficiency

It was evident since the first work on finite volume by Nannelli and Succi [6] that this method would increase the computational cost per node due to computationally demanding (expensive) advection term. Advection term now involves estimation of flux which was otherwise just shift in memory allocation in the standard LBM. Also, the maximum time-step restriction given by $0 < \triangle t \leqslant 2\tau$ for Euler time-stepping scheme can add the extra cost.
However, the advantage of FV LB is that it allows grid refinement for wall bounded flows thus requiring fewer grids than the standard LB for the whole domain; or permits

larger domain size for the same number of grid points. This comparatively reduces the number of computational grids, gaining up-to around two orders of magnitude in the computational cost as reported by Amati *et al.* [7].
Systematic comparisons were performed by Prestininzi *et al.* [100] considering shallow water flows in complex domains. They concluded that if the accuracy is not to be matched between two algorithms, FV LB (with acceptable accuracy) can be 10 times less expensive than the standard LBM.

3.3.7 Boundary Conditions

In the FV approach, there are mainly three types of wall boundary conditions used to estimate proper fluxes as classified by Ubertini *et al.* [11] : *(i)* equilibrium method *(ii)* mirror method and *(iii)* covolume method.
Equilibrium method is the simplest one. The specified density and velocity at the boundary are used to form the equilibrium distribution function. Then, this equilibrium distribution sets the populations. Therefore, the populations assigned at the boundary node corresponds to the specified density and velocity. It has been seen to be used by Prestizini *et al.* [100]. Obvious drawback of the method is its inability to incorporate gradients at the boundary.

$$f_{(boundary)} = f^0(\rho, \mathbf{v})$$

The second type is the mirror type. Ghost nodes are created as a mirror to the internal fluid node, about the boundary. This can accommodate boundary gradients easily. Populations at these mirrored ghost nodes can be defined from populations at internal nodes. One usual practice is a simple extrapolation.

$$f_{(ghost\ node)} = 2 * \ f_{boundary} - f_{internal\ node}$$

This method is quite popular and have been used by Patil *et al.* [18]. Here, it is easy to realise no-slip physical condition by taking $f_{boundary} = 0$ and free-slip physical condition by taking $f_{boundary} = f_{internal\ node}$.
The above two methods can be applied to both cell-vertex and cell-centered type. However, the third type is more specifically used for cell-vertex type. The flux at the edge of boundary is simply estimated by interpolating between two cell-vertex nodes of a cell boundary. Rossi *et al.* [103] have been noticed to use this method which is comparatively difficult than the previous two in terms of implementation.
The above mentioned boundary conditions were seen not to be satisfactory for open flows. Ubertini *et al.* [11] proposed a simple recipe to introduce buffer zone at the edge

of each boundary cell. In these buffers, the same velocity field of the fluid node nearest to the boundary is imposed. Now, the boundary nodes are treated as internal nodes. This ensured zero-gradient at the boundary. This method was followed by Patil *et al.* [18]. Zarghami *et al.* [12] also followed the same idea of defining buffer zone. However the definition of populations at the inlet took inspirations from Zou and He method [104] (popular method used in standard LBM). Also, the populations at outlet buffer zone were extrapolated from the internal nodes.
Concerning the boundary conditions in the volumetric approach for thermo-hydrodynamic problems (refer [101]), diffuse reflection concept inspired from rarefied theory of gases have been successfully implemented [105]. This method ensured the net mass gain due to the boundaries is null. However, a small local density variation came as a side-effect which introduced velocity slip and temperature jump at the boundaries as reported by Sbragaglia and Sugiyama [101]. To correct this, they proposed to impose the exact local velocity and temperature by correcting the equilibrium distribution functions in the above mentioned diffuse reflection method. The corrections are small variations of velocity and temperature which are obtained with an iterative Newton-Raphson method.

3.3.8 Applications

FV LB has been validated upon variety of benchmark fluid dynamics flows. In the early years of FV LB, researchers validated their algorithm with 2D flows at low Reynolds numbers. Choice of the simplest flow for the validation purpose was Poiseuille flow as chosen by Nannelli and Succi [6] in the first proposition of FV LB. Peng *et al.* [9] applied their cell-vertex type, unstructured FV LB algorithm with 2D Taylor-Vortex flows and two-coaxial cylinders; Xi *et al.* [102] with 3D Taylor-Couette Flow and Rossi *et al.* [103] with flow past sphere at $Re = 100$. Efforts were also concentrated to improve the stability criteria. Stiebler *et al.* [99] showed the improvement in stability range with their LSLR upwind scheme for advection by simulating Poiseuille flow at $Re = 460000$. Other researchers Patil *et al.* [18] and Zarghami *et al.* [12] aimed to simulate unsteady flow (2D flow past cylinder case) to test the improvement in stability with their improved techniques. The later could simulate the case with $Re = 100$. Lid driven cavity problem has also been a flow of interest with Ubertini *et al.* [11] and Patil *et al.* [18] trying their hand. Ubertini *et al.* managed to simulate $Re = 1000$ which was later extended by Patil *et al.* to $Re = 3200$. FV LB has also been demonstrated for 3D Turbulent Flows. Amati *et al.* [7] simulated turbulent channel flow at $Re = 6000$ ($Re_\tau \sim 354$). Another significant attempt to simulate a Rayleigh-Bénard convective system was demonstrated by Sbragaglia and Sugiyama [101] by coupling multi-speed thermal model with Peng's volumetric formulation. It was a 2D system with Prandtl number, $Pr = 1$ and Rayleigh number, $Ra = 8224$.

3.4 Conclusion

Until the latest developments, FV LB approach is still not able to simulate complex, three-dimensional, fully developed turbulent flows. One of such exception is the model proposed by Amati *et al.* [7], which was probed in a three-dimensional plane turbulent channel flow. In such case however, the grid wall refinement was based on a very simple structure of halved-grid spacing near the walls and the accuracy of the method turned out to be not satisfactory. Therefore it is still far from being competitive to the most popular locally embedded mesh refinement technique for the LBM.

From the study, it can be understood that the inability of the current FV LB approach to simulate fully developed turbulent flows is due to the lack of required accuracy, good stability range and computational efficiency. The important parameters majorly influencing these characteristics are *time discretization* and *flux estimation* which are still underdeveloped for LBM. Regarding the time discretization, the best scheme applied to FV LB approach is explicit, fifth order Runge-Kutta scheme by Zarghami *et al.* [12]. This scheme was necessary to have stable results for unsteady laminar flow (2D flow past cylinder case at Re = 100). The consequence on the computational cost is evident since in such a scheme the advection term need to be computed five times per time step (while in the ST method it is performed just by means of a memory shift, the streaming). Similarly for the flux estimation, the best scheme used until now is finite volume TVD by Patil *et al.* [18]. The method relies on second-order TVD scheme with flux limiters that could obtain accurate results but the method is tedious and complex. Since the method is bulky, it influences the computational efficiency.

It can be easily understood from the above examples that the effect of a scheme is interconnected. While trying to improve the stability or the accuracy, the method looses computational efficiency or vice-versa. Therefore, to have a better FV LB approach that can stand competitive to the existing popular mesh refinement techniques in LBM, time discretization and flux estimation must be handled in a more advanced way which will not only ensure accuracy but also improve the stability range and computational efficiency.

Chapter 4

A Novel Finite Volume Lattice Boltzmann Method

4.1 Lattice Boltzmann Finite Volume Formulation

The method of characteristics is very convenient from a computational point of view because it reduces the complexity of the integration of a PDE to a simple ODE, however at the same time it introduces *a tight link between the shape of the velocity lattice and the spatial discretization mesh.* Such a constraint can be removed if one takes the more usual numerical approach based on *(i)* a direct spatial discretization of equation 1.30 combined with *(ii)* an independent time discretization phase. For the first step, several standard options are available, such as finite elements, finite differences or finite volumes methods.

The idea of using a finite volume method to decouple the spatial numerical mesh from the velocity lattice structure have been dealt in large in the previous chapters. The present chapter further explains in detail a novel finite volume Lattice Boltzmann method in order to simulate fluid flow problems with higher accuracy, greater stability properties and comparable performance as the ST method. The FV method that we propose is of the type denoted as cell-centered (as opposed to vertex-centered, see figure 4.1). Its most original features concern the semi-implicit approach taken for the time discretization (section 4.1.2) and the method of fluxes computation which adopt, for the first time in this context, a quadratic upwind QUICK scheme (section 4.1.3). In the following, we detail the steps taken in developing it.

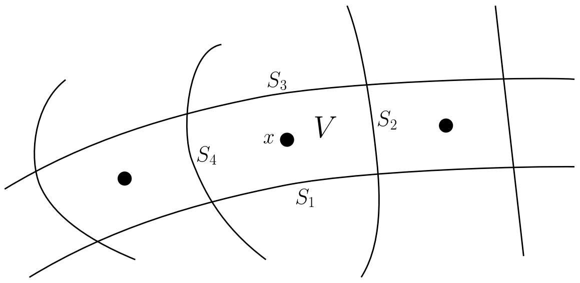


FIGURE 4.1: Cartoon of the finite volume space discretization: the dot denotes the position of the cell center (where the value of $f_\alpha(x)$ is defined), while the lines marks the cell boundaries. The cell has volume V and each boundary surface is denoted with S_j with $j = 0, \ldots, 3$ in two dimensional space.

4.1.1 Space discretization

Upon integration of equation 1.30 over a volume V (of surface S) and by applying the flux theorem we get,

$$\frac{\partial f_i}{\partial t} + \mathbf{c}_i \cdot \frac{\partial f_i}{\partial x} = -\frac{1}{\tau}\left(f_i - f_i^0\right) + F_i$$

$$\int_V \frac{\partial f_i}{\partial t}\, dV + \int_S \mathbf{c}_i \cdot \mathbf{n}\; f_i\; dS = \int_V \frac{1}{\tau}(f_i^0 - f_i)\; dV + \int_V F_i\; dV \tag{4.1}$$

We then assume that every term in the volume integrals can be considered as constant and its magnitude taken at a reference location $\mathbf{x}$ (also called *node*) inside V.
The term in the surface integral however, carries some spatial variability. When such a surface is decomposed in M faces (as in a structured grid of nodes with connectivity index M) it is convenient to make the assumption that f_i is constant on each of the S_j surfaces perpendicular to $\mathbf{n}_j$ and denoting its value with $[f_i]_j$. This altogether leads to:

$$\frac{\partial f_i}{\partial t} + \frac{S_j}{V}\mathbf{c}_i \cdot \mathbf{n}_j\, [f_i]_j = \frac{1}{\tau}(f_i^0 - f_i) + F_i \tag{4.2}$$

Where summation over the repeated index j is applied.

4.1.2 Time discretization

If the time derivative is discretized by the explicit Euler scheme, we get:

$$f_i^{(t+\Delta t)} = f_i^{(t)} - \Delta t \frac{S_j}{V}\mathbf{c}_i \cdot \mathbf{n}_j \left[f_i^{(t)}\right]_j + \frac{\Delta t}{\tau}(f_i^{0\,(t)} - f_i^{(t)}) + \Delta t\; F_i, \tag{4.3}$$

where the superscript indexes (t) and $(t + \Delta t)$ denote respectively the current and the next discrete time instant. Such an approach however, puts tight bounds on the maximum allowed Δt.

Empirically it is possible to show that this range becomes even narrower when the non-local advection term, the forcing and the time dependency in f^0 are taken into account. The fact that Δt_{max} depends on and is bounded by the value of τ is a known problem in FV LB implementations. It poses, among others, a severe limitation for the simulations of turbulent flows (*i.e.* low viscosity flows). On the opposite, such a constraint does not exist in the ST approach (where Δt is independent of τ). Different solutions have been proposed in the literature, often resorting explicit time discretization schemes of higher order, for example multi-stages Runge-Kutta schemes. However, as we mentioned above such schemes only produce marginal improvements at the expenses of considerably increasing the computations. The Runge-Kutta schemes for example requires multiple evaluations of the full right-hand-side terms on equation 4.2. We opt for a different approach, with a better trade-off between the enhancement of the stability limit for Δt and the growth in computational cost.

Similarly to what is done for the classic LB streaming method, in the steps from equation 1.47 to equation 1.55, a possible improvement consists in taking also for the FV algorithm a semi-implicit integration scheme. However, this is not directly possible for equation 4.2 because of the presence of the advection term. We propose to limit such an approach only to the collision and forcing terms. Therefore the discretization in time is applied by adopting a mixed approach: while for the collision and forcing terms a semi-implicit method is used, for the advection a simple explicit Euler is implemented. The derivation starting from equation 4.2 is as follows:

$$\frac{\partial f_i}{\partial t} + \frac{S_j}{V}\mathbf{c}_i \cdot \mathbf{n}_j \left[f_i\right]_j = \frac{1}{\tau}(f_i^0 - f_i) + F_i$$

$$\begin{aligned} f_i^{(t+\Delta t)} &= f_i^{(t)} - \Delta t \frac{S_j}{V}\mathbf{c}_i \cdot \mathbf{n}_j \left[f_i^{(t)}\right]_j \\ &- \frac{\Delta t}{2}\left(\frac{1}{\tau}(f_i^{(t)} - f_i^{0\,(t)}) - F_i^{(t)} + \frac{1}{\tau}(f_i^{(t+\Delta t)} - f_i^{eq\,(t+\Delta t)}) - F_i^{(t+\Delta t)}\right) \end{aligned} \tag{4.4}$$

Note that by bringing on the left-hand-side all the terms to be evaluated at $(t+\Delta t)$:

$$\begin{aligned} f_i^{(t+\Delta t)} + \frac{\Delta t}{2\tau}\left(f_i^{(t+\Delta t)} - f_i^{0\,(t+\Delta t)} - \tau F_i^{(t+\Delta t)}\right) &= f_i^{(t)} - \Delta t \frac{S_j}{V}\mathbf{c}_i \cdot \mathbf{n}_j \left[f_i^{(t)}\right]_j \\ &- \frac{\Delta t}{2\tau}\left(f_i^{(t)} - f_i^{0\,(t)} - \tau F_i^{(t)}\right) \end{aligned} \tag{4.5}$$

One now introduces the redefined distribution function,

$$\tilde{f}_i = f_i + \frac{\Delta t}{2\tau}(f_i - f_i^0 - \tau F_i) \tag{4.6}$$

It should be noted that $\tilde{f}_i$ is to correct ***only the non-equilibrium part of the distribution function*** *such that the LB equation is more consistent with the Navier-Stokes equations (more consistent than the one described by f_i). Therefore, both definitions of the distribution function should have similar equilibrium part.*

$$\tilde{f}_i^0 = f_i^0 \tag{4.7}$$

An alternative form of the relation 4.6 can be derived as follows:

$$\tilde{f}_i = \left(1 + \frac{\triangle t}{2\tau}\right) f_i - \frac{\triangle t}{2\tau}[f_i^0 + \tau F_i]$$

Applying equation 4.7 in the above equation

$$\tilde{f}_i = \left(1 + \frac{\triangle t}{2\tau}\right) f_i - \frac{\triangle t}{2\tau}\left[\tilde{f}_i^0 + \tau F_i\right]$$

$$f_i = \frac{\tilde{f}_i + \frac{\triangle t}{2\tau}\left[\tilde{f}_i^0 + \tau F_i\right]}{\left(1 + \frac{\triangle t}{2\tau}\right)} \tag{4.8}$$

Proceeding further, we use equation 4.6 in equation 4.5,

$$\tilde{f}_i^{(t+\Delta t)} = \left(\tilde{f}_i^{(t)} - \frac{\Delta t}{2\tau}(f_i^{(t)} - f_i^{0\,(t)} - \tau F_i^{(t)})\right) - \Delta t \frac{S_j}{V} \mathbf{c}_i \cdot \mathbf{n}_j \left[f_i^{(t)}\right]_j - \frac{\Delta t}{2\tau}(f_i^{(t)} - f_i^{0\,(t)} - \tau F_i^{(t)}) \tag{4.9}$$

$$\tilde{f}_i^{(t+\Delta t)} = \tilde{f}_i^{(t)} - \frac{\Delta t}{\tau}(f_i^{(t)} - f_i^{0\,(t)} - \tau F_i^{(t)}) - \Delta t \frac{S_j}{V} \mathbf{c}_i \cdot \mathbf{n}_j \left[f_i^{(t)}\right]_j \tag{4.10}$$

Applying equations 4.7 and 4.8 to the above equation, we arrive at

$$\tilde{f}_i^{(t+\Delta t)} = \tilde{f}_i^{(t)} - \frac{\Delta t}{\tau}\left[\frac{\tilde{f}_i^{(t)} + \frac{\Delta t}{2\tau}\left(\tilde{f}_i^{0\,(t)} + \tau F_i^{(t)}\right)}{\left(1 + \frac{\Delta t}{2\tau}\right)} - \tilde{f}_i^{0\,(t)} - \tau F_i^{(t)}\right] - \Delta t \frac{S_j}{V} \mathbf{c}_i \cdot \mathbf{n}_j \left[f_i^{(t)}\right]_j \tag{4.11}$$

$$\tilde{f}_i^{(t+\Delta t)} = \tilde{f}_i^{(t)} - \frac{\Delta t}{\tau}\left[\frac{\tilde{f}_i^{(t)} - \tilde{f}_i^{0\,(t)} + \frac{\Delta t}{2} F_i^{(t)}}{\left(1 + \frac{\Delta t}{2\tau}\right)}\right] + \Delta t F_i^{(t)} - \Delta t \frac{S_j}{V} \mathbf{c}_i \cdot \mathbf{n}_j \left[f_i^{(t)}\right]_j \tag{4.12}$$

Further simplifying,

$$\tilde{f}_i^{(t+\Delta t)} = \tilde{f}_i^{(t)} - \frac{\Delta t}{\tau + \frac{\Delta t}{2}}\left(\tilde{f}_i^{(t)} - \tilde{f}_i^{0\,(t)}\right) - \frac{\triangle t}{\tau}\frac{\triangle t}{2}\frac{1}{\left(1+\frac{\triangle t}{2\tau}\right)}F_i^{(t)} + \Delta t F_i^{(t)} - \Delta t \frac{S_j}{V}\mathbf{c}_i \cdot \mathbf{n}_j \left[f_i^{(t)}\right]_j \tag{4.13}$$

$$\tilde{f}_i^{(t+\Delta t)} = \tilde{f}_i^{(t)} - \frac{\Delta t}{\tilde{\tau}}\left(\tilde{f}_i^{(t)} - \tilde{f}_i^{0\,(t)}\right) + \Delta t\left(1 - \frac{\Delta t}{2\tilde{\tau}}\right)F_i^{(t)} - \Delta t \frac{S_j}{V}\mathbf{c}_i \cdot \mathbf{n}_j \left[f_i^{(t)}\right]_j \tag{4.14}$$

At this point the new relaxation time is introduced $\tilde{\tau} = \tau + \frac{\triangle t}{2}$.

Finally, expanding $f_i^{(t)}$ in the form of $\tilde{f}_i^{(t)}$ (in the advection term) and rearranging the above equation, we arrive to the final form of the equation:

$$\begin{aligned}\tilde{f}_i^{(t+\Delta t)} = \tilde{f}_i^{(t)} - \Delta t \frac{S_j}{V}\mathbf{c}_i \cdot \mathbf{n}_j &\left[\tilde{f}_i^{(t)} + \frac{\Delta t}{2\tilde{\tau}}(\tilde{f}_i^{0\,(t)} - \tilde{f}_i^{(t)}) + \frac{\Delta t}{2}F_i^{(t)}\right]_j \\ &+ \frac{\Delta t}{\tilde{\tau}}(\tilde{f}_i^{0\,(t)} - \tilde{f}_i^{(t)}) + \Delta t\left(1 - \frac{\Delta t}{2\tilde{\tau}}\right)F_i^{(t)}\end{aligned} \tag{4.15}$$

The above equation shares the same definitions of equation 1.55 for the tilded distribution function, $\tilde{f}_i$ and the relaxation time ($\tilde{\tau}$). The rule of computing the macroscopic fields and the viscosity $\nu = \tau\ c_0^2 = (\tilde{\tau} - \Delta t/2)\, c_0^2$ are exactly the same as for the ST algorithm. However, one can immediately note the advected field in the equation is not simply a distribution function but rather a complex term involving also the equilibrium distribution and the forcing. The main advantage of this approach is that a stability analysis under the same hypothesis mentioned above (neglecting advection, forcing and time dependences in the equilibrium function) shows now that every time step length Δt is stable.

Proof: *Here we make similar assumptions as the previous case of explicit Euler scheme. Also, we consider reduced LB equation with time dependency only to the non-equilibrium distribution function,*

$$\frac{df^{neq}}{dt} = -\frac{1}{\tau}f^{neq}$$

After time step $\triangle t$,

$$f^{neq\,(t+\triangle t)} = f^{neq\,(t)} - \frac{\triangle t}{2}\left(\frac{1}{\tau}f^{neq\,(t)} + \frac{1}{\tau}f^{neq\,(t+\Delta t)}\right)$$

Grouping the like terms,

$$\left(1 + \frac{\Delta t}{2\,\tau}\right)f^{neq\,(t+\triangle t)} = \left(1 - \frac{\Delta t}{2\,\tau}\right)f^{neq\,(t)}$$

And after N times $\triangle t$ time-steps we get,

$$f^{neq\,(t+N\triangle t)} = \left(\frac{1-\frac{\triangle t}{2\tau}}{1+\frac{\triangle t}{2\tau}}\right)^N f^{neq\,(t)}$$

We know that the for the solution to converge, the non-equilibrium distribution function should reach 0. It is evident from the above expression that if N is sufficiently long enough, the non-equilibrium distribution function will definitely reach 0 without any extra condition. Therefore, using semi-implicit method every time step length Δt is stable.

However, we find that the situation reached so far is not yet satisfactory. From simple numerical tests we observe that even with this discretization scheme the time-step size is still restricted by the relaxation time, particularly for small relaxation time values. The origin of this still limited stability of the scheme lies now in the advection term. For this reason a further refinement is proposed. We set it into place by applying the so-called Heun predictor-corrector scheme to the advection term. In other words we use the calculation of the population based on equation 4.15, now called $\tilde{f}^*$, as an intermediate value for constructing an explicit trapezoidal integration rule applied to the advection:

$$\begin{aligned}\tilde{f}_i^{(t+\Delta t)} &= \tilde{f}_i - \Delta t \frac{S_j}{V}\mathbf{c}_i \cdot \mathbf{n}_j \frac{\left[\tilde{f}_i^* + \frac{\Delta t}{2\tilde{\tau}}(\tilde{f}_i^{0*} - \tilde{f}_i^*) + \frac{\Delta t}{2}F_i^*\right]_j + \left[\tilde{f}_i + \frac{\Delta t}{2\tilde{\tau}}(\tilde{f}_i^0 - \tilde{f}_i) + \frac{\Delta t}{2}F_i\right]_j}{2} \\ &+ \frac{\Delta t}{\tilde{\tau}}(\tilde{f}_i^0 - \tilde{f}_i) + \Delta t\left(1 - \frac{\Delta t}{2\tilde{\tau}}\right)F_i \qquad (4.16)\end{aligned}$$

where F_i^* indicates the LB forcing term computed from $\tilde{f}^*$. This scheme enjoys greater stability at the additional computational price of a second evaluation of the advection term. In order to make this observations more quantitative we should first specify the way in which the flux terms $[\ldots]_j$ are computed. Indeed, the exact stability properties of the method depends upon the implementation of the advection term, that we discuss in the following section.

4.1.3 Approximation of the advection term

There exist several ways to estimate the non-local term $[f_i]_j$ and each one can be characterised by a spatial order of accuracy. The complexity of such an estimation also depends on the grid characteristics. Even for structured but irregular grids an high-order estimation of $[f_i]_j$ becomes expensive in computation terms. In order to simplify such a problem, we limit the following discussion to the case of structured regular grids, that is to say to the case where the nodes lie on lines. This is the case for instance of a non-uniform Cartesian grid (the typical case of wall-refinement), but it also apply to a uniformly skewed non-orthogonal grids.

It has been long known that fluxes in advection equations are better approximated by upwind schemes, which are interpolation schemes biased in the direction determined by the sign of the characteristic speeds (the set of $\mathbf{c}_i$ in our case). At the lowest order of accuracy, and easiest level of implementation, there exist the first order upwind scheme, increasing the refinement leads to linear interpolation schemes or even to more refined quadratic schemes (which are of 3^{rd} order of spatial accuracy). While low-order schemes introduce artificial numerical dissipations, higher-order ones lead to spurious oscillations, especially evident near the boundaries. This is also true in the present cell-centered FV implementation, in particular zero-order or linear up-wind interpolation schemes leads to inaccurate results. Even a cell-centered symmetric schemes, which here does not display extra dissipation, produces inaccurate results in the presence of boundaries. Empirically, we find that the quadratic upstream interpolation, known as QUICK method [106], is the simplest one to give accurate results both in open (*i.e.* periodic) and bounded domains.

According to this QUICK approach, on each surface S_j at position say $\mathbf{x}_{S_j}$, $[f_i]_j$ is approximated via a combination of the value of f_i in the two nodes bracketing the surface (denoted with $\mathbf{x}$ and $\mathbf{x}^+$) and a third node that is located upstream respect to direction of the projection of $\hat{\mathbf{c}_i}$ on $\hat{\mathbf{n}_j}$ (denoted either $\mathbf{x}^{++}$ or $\mathbf{x}^-$). The interpolant function is a parabola $a + b\;\xi + c\;\xi^2$, with ξ the linear coordinate spanning on the line connecting the nodes (see sketch in figure 4.2). This leads to:

$$\begin{aligned} \left[f_i(\mathbf{x}_{S_j})\right]_j &= (1-\gamma_1+\gamma_2)f_i(\mathbf{x}) + \gamma_1\; f_i(\mathbf{x}^+) - \gamma_2\; f_i(\mathbf{x}^-)\;|_{i:\;\hat{\mathbf{c}_i}\cdot\hat{\mathbf{n}_j}>0} \\ &+ \;(1-\gamma_3+\gamma_4)f_i(\mathbf{x}^+) + \gamma_3\; f_i(\mathbf{x}) - \gamma_4\; f_i(\mathbf{x}^{++})\;|_{i:\;\hat{\mathbf{c}_i}\cdot\hat{\mathbf{n}_j}<0} \end{aligned} \tag{4.17}$$

where the $\gamma_{1,2,3,4}$ are 4 coefficients, that shall be evaluated and/or stored for each surface of the control volumes.

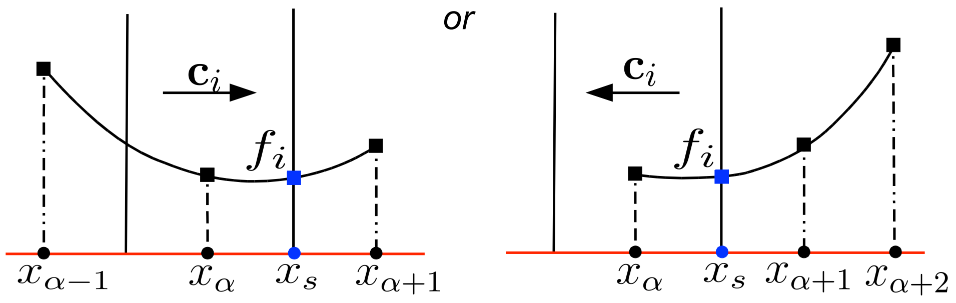


FIGURE 4.2: Sketch of the quadratic upwind interpolation scheme (QUICK) for estimating the value of f_i at the cell boundary position x_S. Note that the interpolation method make use of different nodes according to the direction of the population velocity $\vec{c}_i$.

The behaviour of these schemes can also be understood by simple tests. The *dissipative rates* of these schemes can be tested using a decaying laminar Kolmogorov flow. This is a type of flow where it is initialised with a one dimensional sinusoidal velocity amplitude profile and it is left to decay in time. For this test with laminar Kolmogorov flow, let us take an one-dimensional domain, $(L_x, L_y, L_z) = (1, 64, 1)$, where $L_{x,y,z}$ denote the spatial dimensions, which is here also the same as the number of mesh nodes (indicated with $N_{x,y,z}$). It can be seen from figure 4.3 that upwind schemes have higher dissipative rates compared to center-difference and QUICK schemes. This is due to unphysical, increased numerical viscosity in low order schemes that produce inaccurate results which can also be proved from figure 4.4. Therefore upwind scheme is not a scheme of choice.

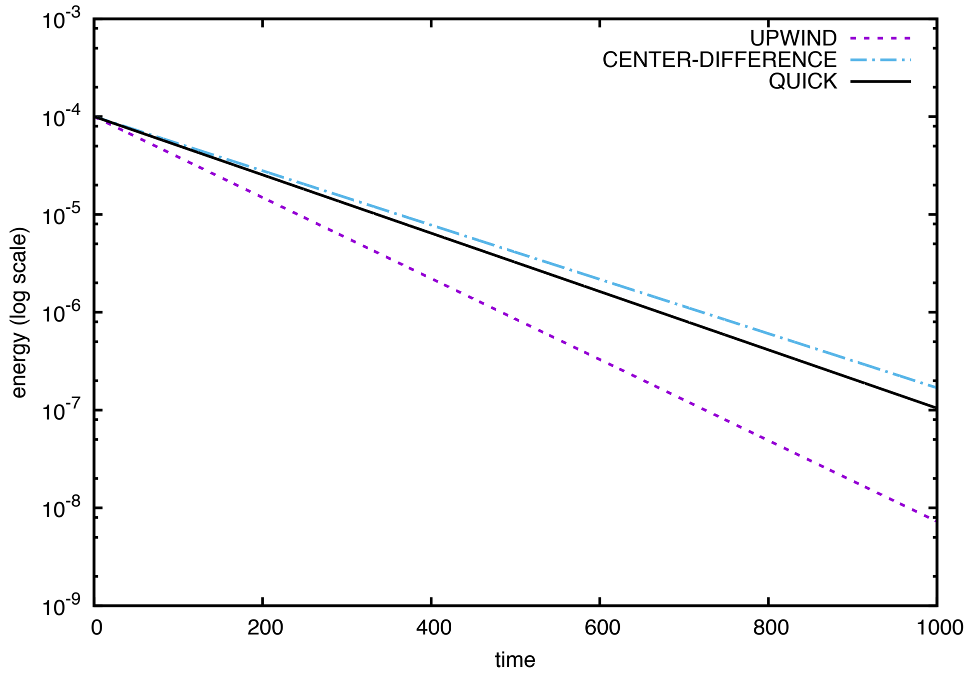


FIGURE 4.3: Study of the dissipative rates of different schemes using energy *vs* time graph

Another important test is on the *accuracy* of these schemes. For this, let us take a simple bounded flow that is initiated with a parabolic Poiseuille velocity profile in the same spatial dimensions as above. No-slip boundary conditions are applied on top and bottom boundaries and periodicity on x-direction. Uniform volume forcing is applied along x-direction.

$$F_x = \frac{4}{L_y^2}\,\nu\,U$$

where ν is viscosity. This forces a parabolic velocity profile in the fluid defined by,

$$u_x = -\frac{F_x}{2}\frac{(y^2 - L_y\,y)}{\nu}, \qquad u_y = 0$$

Figure 4.4 compares the accuracy of upwind, center-difference and QUICK methods with the help of the standard Poiseuille flow. Upwind is a first order method and clearly shows high dissipation as also seen in the previous test. Although it can't predict the correct maximum velocity, it however maintains the correct shape of the parabolic velocity profile. The high dissipation of this method may be due to the diagonal lattice speed directions introduced in LBM. Center-difference method is a second order method with natural setback of instability at higher flux and possesses non-directional nature. In this test, it is observed that although center-difference scheme could correctly predict the amplitude of the parabolic velocity profile, it faces the staircase problem. The other higher order scheme which uses quadratic upstream interpolation named QUICK is sufficiently accurate as it matches very well with the analytical solution.

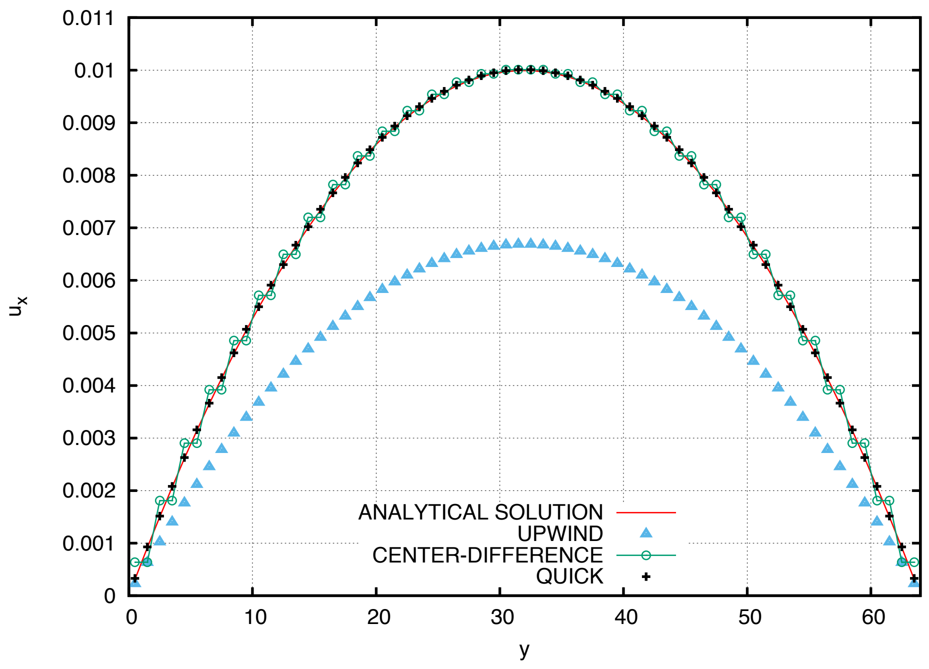


FIGURE 4.4: Test of accuracy of different advection schemes with a standard Poiseuille flow.

4.1.4 Force term

Finally a brief remark on the forcing term F_i in the FV LB equation. The FV LB equation has great similarity with the ST LB equation. Therefore all the explanations on force term in section 1.4.4.2 holds equally true for the FV LB case.
Revising briefly - The simplest way to implement force term, is by the expression:

$$F_i = w_i \, \frac{\mathbf{c}_i \cdot \rho \, \mathbf{a}}{c_0^2} \tag{4.18}$$

where the summation over index i is not implied, w_i is a lattice dependent weight and the product $\rho\,\mathbf{a}$ represents the force per unit volume in physical space (for example in case of a gravitational external field $\mathbf{a} = \mathbf{g}$). The above expression satisfies the conditions

$\Sigma_i F_i = 0$ and $\Sigma_i \mathbf{c}_i F_i = \rho\, \mathbf{a}$, which are required for equation 1.30 to give the correct macroscopic effect of a body force term. However, when the body force is time/space dependent and equation 1.30 is discretized in space and time, such as in equations (1.55 for ST LB and 4.16 for FV LB), the above expression needs to be refined in order to remove spurious discretization terms that would otherwise appear in the macroscopic limit. The corrected expression, first proposed by Guo *et al.* [107], is

$$F_i = w_i \left(\frac{\mathbf{c}_i - \mathbf{u}}{c_0^2} + \frac{(\mathbf{c}_i \cdot \mathbf{u})\ \mathbf{c}_i}{c_0^4} \right) \rho\ \mathbf{a}. \tag{4.19}$$

The above correction was not enough to match the correct Navier- Stokes equations and verified that the expression required a multiplicative factor $1 - \Delta t/(2\tau)$ (refer equations 1.55 for ST LB and 4.16 for FV LB). Finally, the expression becomes

$$F_i = \left(1 - \frac{\Delta t}{2\tilde{\tau}}\right) w_i \left(\frac{\mathbf{c}_i - \mathbf{u}}{c_0^2} + \frac{(\mathbf{c}_i \cdot \mathbf{u})\ \mathbf{c}_i}{c_0^4} \right) \rho\ \mathbf{a}. \tag{4.20}$$

Note that accordingly (by employing the relation between $\tilde{f}_i$ and $\rho\mathbf{u}$ and ρ given in section 1.4.5.1) one gets the fluid velocity as $\mathbf{u} = \Sigma_i \mathbf{c}_i \tilde{f}_i / \Sigma_i \tilde{f}_i + \frac{\Delta t}{2}\mathbf{a}$.

4.1.5 Boundary conditions

In the following we consider the implementation two types of boundary conditions (BC): *(i)* no-slip walls and *(ii)* fixed density (or equivalently pressure) boundaries. The physical domain boundaries lie on the faces of the external control volumes. Similarly to the bounce-back approach for the streaming LB algorithm, we introduce in-wall ghost cells. However, in the QUICK treatment of the advection two ghost cells are needed instead of one. The ghosts cells are located in-wall and have centers at position mirroring the first and second nodes in the fluid domain. Let's suppose that the quantity to be advected is f_i and that the boundary condition is to be imposed on the S cell surface, whose center is at $\mathbf{x}_S$. For simplicity we assume that S lies along the plane (y, z) perpendicular to x, with the x axis pointing inward (*i.e.* in the fluid bulk direction). Consequently, the first two nodes in the fluid domain are located at position $\mathbf{x}_1 = \mathbf{x}_S + \Delta x_1/2$ and $\mathbf{x}_2 = \mathbf{x}_S + \Delta x_1 + \Delta x_2/2$, where Δx_1 and Δx_2 represents the linear size of the two first discretization volumes (refer figure 4.5). Accordingly, the ghosts cells are at positions $\mathbf{x}_{-1} = \mathbf{x}_S - \Delta x_1/2$ and $\mathbf{x}_{-2} = \mathbf{x}_S - \Delta x_1 - \Delta x_2/2$. A no-slip boundary condition requires $\mathbf{u}(\mathbf{x}_S) = 0$, while the density at $\rho(\mathbf{x}_S)$ is free to take any arbitrary value. This corresponds to the constraint $\Sigma_i \mathbf{c}_\alpha f_i(\mathbf{x}_s) = 0$. The simplest way (but not the only one) to enforce it, is to set the in-wall nodes as the following:

$$f_i(\mathbf{x}_{-1}) = f_{inv(i)}(\mathbf{x}_1) \quad \text{and} \quad f_i(\mathbf{x}_{-2}) = f_{inv(i)}(\mathbf{x}_2), \tag{4.21}$$

where $inv(i)$ is an integer valued function that selects the population moving along the opposite direction with respect to $\mathbf{c}_i$. We have verified that such a choice does not introduce artificial fluctuations at the boundary, that would quickly generate instabilities. This implementation of BC, that we dub *double reflection*, has a first-order of accuracy in space (it does not implement the quadratic interpolation) and therefore it leaves room for further improvements.

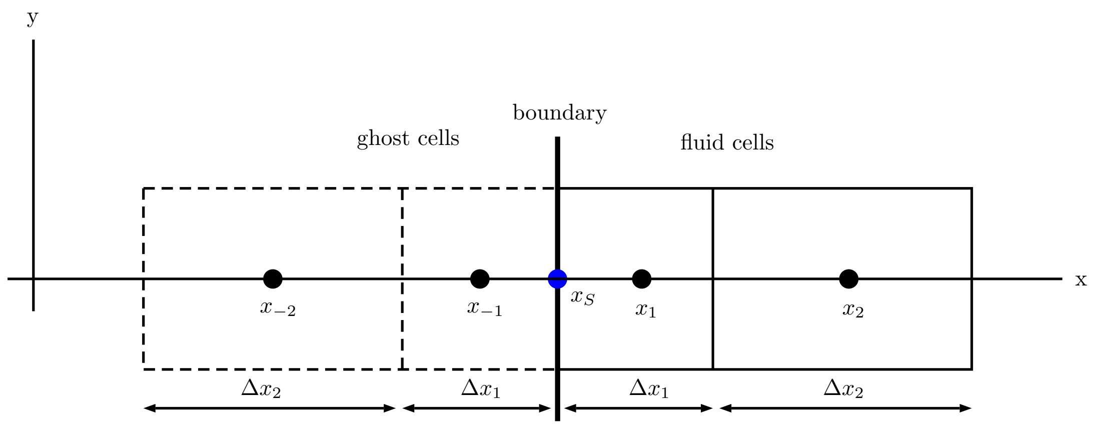


FIGURE 4.5: Illustration of the finite-volume arrangement for the implementation of the *double-reflection* boundary conditions.

If instead we are interested to impose a density value at the border, say $\rho_S = \Sigma_i f_i(\mathbf{x}_S)$, we need to resort an extrapolation strategy. We proceed as follow, first the density value $\rho(\mathbf{x}_{-1})$ is linearly extrapolated from the values ρ_S and $\rho(\mathbf{x}_1)$, similarly $\rho(\mathbf{x}_{-2})$ is derived from ρ_S and $\rho(\mathbf{x}_2)$. Second, we assign the in-wall the distribution functions as follows

$$f_i(\mathbf{x}_{-1}) = \frac{\rho(\mathbf{x}_{-1})}{\rho(\mathbf{x}_1)} f_i(\mathbf{x}_1) \quad \text{and} \quad f_i(\mathbf{x}_{-2}) = \frac{\rho(\mathbf{x}_{-2})}{\rho(\mathbf{x}_2)} f_i(\mathbf{x}_2). \tag{4.22}$$

Also the above choice, a rescaling of the bulk distribution functions, is not the only viable way for the implementation of fixed density BC, however it is one that has revealed to not to introduce wall disturbances. As a final remark, we shall note that in the implementation of equation 4.16, the boundary conditions need not to be implemented on the redefined distribution function $\tilde{f}_i$ but rather on the original $f_i = \tilde{f}_i + \frac{\Delta t}{2\tau}(\tilde{f}_i^0 - \tilde{f}_i) + \frac{\Delta t}{2} F_i$.

The algorithm presentation given so far is independent of the particular microscopic velocity lattice topology. In the present work and for the accuracy study presented in the reminder of this thesis we make the choice to always use the so called $D3Q19$ lattice, which is a standard option for three-dimensional LB simulations and reduces to the $D2Q9$ lattice for two-dimensional flow problems [108]. Generally for simple flows, it can also be shown that the lattice arrangements do not affect much the accuracy of the system (refer figure 4.6).

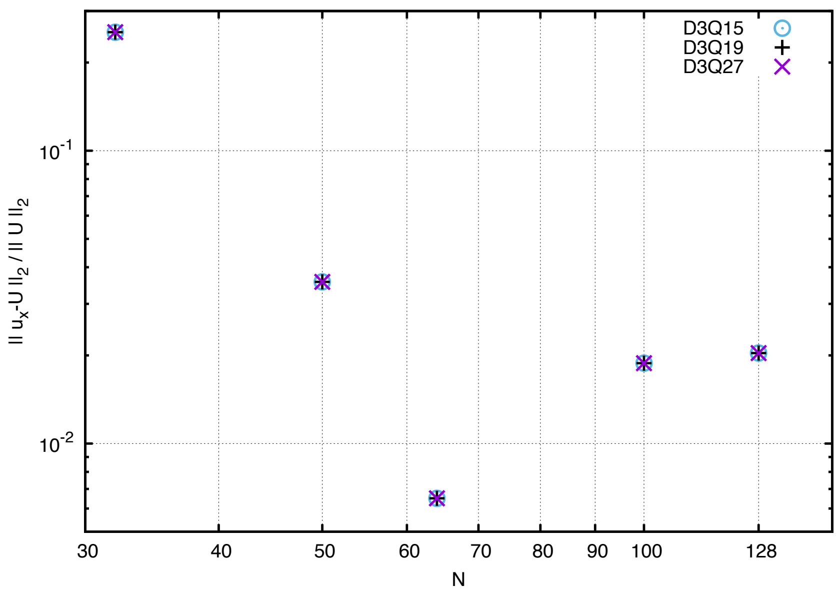


FIGURE 4.6: L_2 norm *vs.* grid size (N) in *log-log scale.* Comparison of different lattice arrangements with a Poiseuille flow of domain size $L_y = 64$. *Note: L_2 norm is defined in the later section.*

4.2 Accuracy tests

In this section we address the accuracy of the present LB FV algorithm and we compare it with the ST algorithm. In particular we approach the following questions: *(i)* to which degree the FV algorithm correctly describes the dynamics of a low Reynolds number viscous flow? Which is its order of spatial accuracy and how does it compare with ST? *(ii)* Is there any optimal usage of the FV algorithm in order to take advantage of the grid refinement and obtaining highly accurate solutions?

4.2.1 Viscosity evaluation

A simulation is performed on a one-dimensional domain, $(L_x, L_y, L_z) = (1, 64, 1)$, where $L_{x,y,z}$ denote the spatial dimensions, which is here also the same as the number of mesh nodes (indicated with $N_{x,y,z}$). The flow is initialized with a one dimensional sinusoidal velocity amplitude profile of the form

$$\mathbf{u}(x, y, z) = (u_x(y), u_y, u_z) = \left(A \sin\left(\frac{2\pi\, y}{L_y}\right), 0, 0 \right) \tag{4.23}$$

and it is left to decay in time. We monitor the behaviour of the total kinetic energy in time, $k_{tot}(t)$, which is expected to decrease exponentially as $k_{tot}(t) = \frac{1}{4} A^2 L_y^2\, e^{-2(2\pi/L_y)^2 \nu\, t}$, with ν representing the fluid kinematic viscosity. The reproduced value of ν can be deduced from a least-square fit of $k_{tot}(t)$ and then compared to the theoretically expected value $\nu = \tau\ c_0^2 = (\tilde{\tau} - \Delta t/2)\, c_0^2$. The degree of accuracy of the FV method measured in such a way is compared with the ST method and reported in figure 4.7. While it is known and expected that accuracy carries some form of dependency with the relaxation time τ, we observe that both FV and ST methods reach the maximal accuracy around $\tau = 0.5$, however ST in that very same case performs better by a factor 10. Moreover, in general the ST error grows less than the FV one for all $\tau < 0.5$.

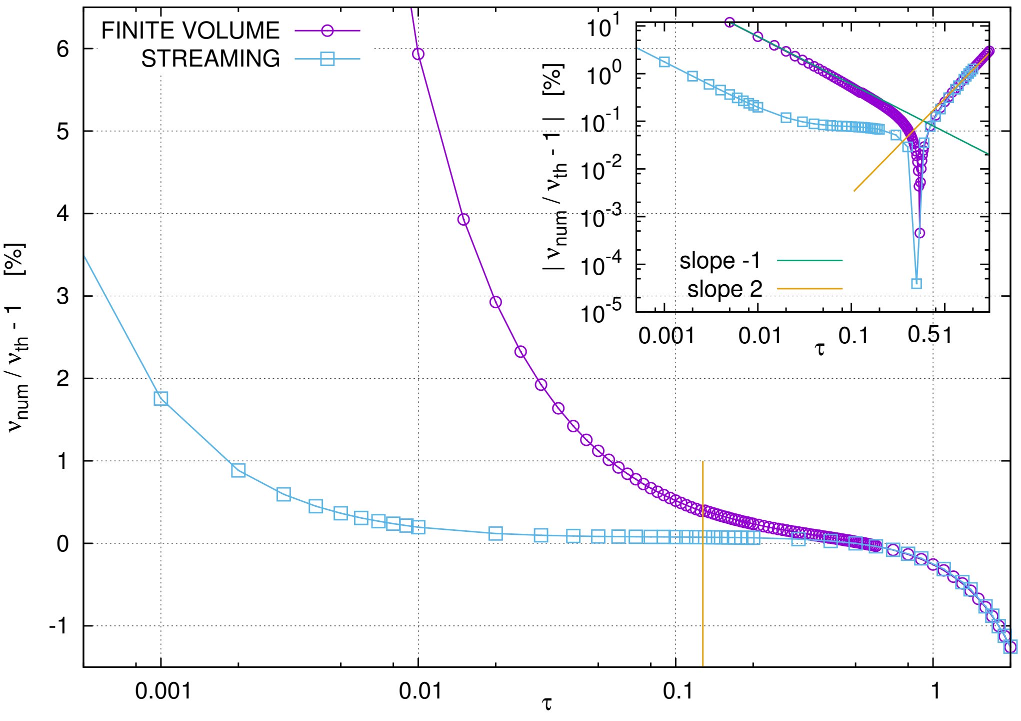


FIGURE 4.7: Relative error of measured kinematic viscosity ν_{num} respect to the expected one $\nu_{th} = \tau\ c_s^2$ as a function of the relaxation time τ. In the finite volume case $\Delta t = 1$ for $\tau \geq 0.13$ (marked with a vertical line) and $\Delta t = 0.1$ for $\tau < 0.13$, while in the Streaming case $\Delta t = 1$ always. In the inset, the absolute value of the same error in log-log scale.

In order to prove the spatial order of accuracy of the FV method (for a flow without boundary conditions), we perform next test in the same flow, where τ is kept constant and the number of grid points N_y is varied together with the spatial dimension L_y (again $L_y = N_y$). *This leads to figure 4.8, which reports evidence for the fact that FV has second order of accuracy, same as the ST method.*

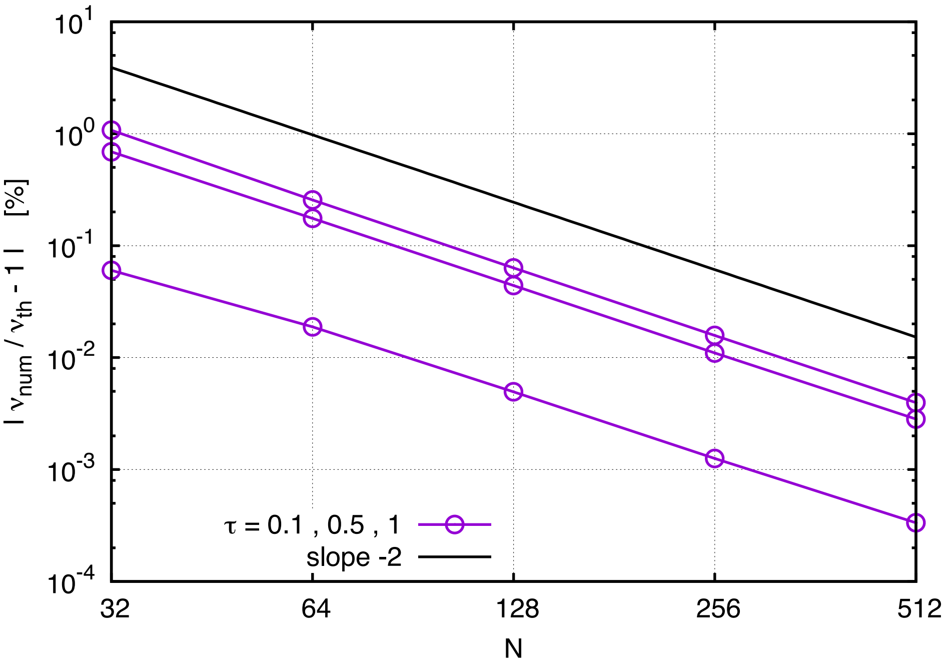


FIGURE 4.8: Finite volume: relative error of kinematic viscosity as a function of grid resolution.

4.2.2 Steady Poiseuille flow

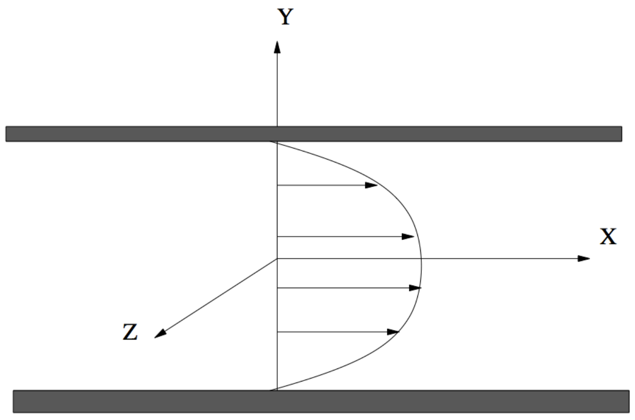


FIGURE 4.9: Coordinate system of the plane Poiseuille Flow.

Our second tests addresses the case of a simple bounded flow in the same spatial domain as above, $[L_x, L_y, L_z] = [1, 64, 1]$. The flow is initiated with a parabolic Poiseuille velocity profile $U_x(y) = 4\ U_{max} L_y^{-2}\ y\ (L_y - y)$ corresponding to a Reynolds number $Re = L_y U_{max}/\nu = 10$. A uniform volume forcing along the x direction and no-slip boundary conditions at $y = 0$ and $y = L_y$ positions are imposed, while periodicity is implemented along the horizontal direction, x. The simulated flow profile (denoted with $u_x(y)$) keeps the original theoretical shape $U_x(y)$ with tiny adjustments depending on the method. In order to compare these two functions we use the relative difference $||u_x - U||_2\ /\ ||U||_2$

where $||\dots||_2$ denotes the L_2 norm, which in its discretized form is computed as:

$$||f(x)||_2 = \left(\int_L f(x)^2 dx\right)^{1/2} = \left(\Sigma_{i=1}^N f_i^2 \Delta x_i\right)^{1/2} \tag{4.24}$$

In figure 4.10, we show the L_2 relative difference results at varying the spatial domain size in y direction, *i.e.*, changing L_y and at the same time N_y (or in other words keeping fixed the grid spacing $\Delta x \equiv L_y/N_y = 1$). The figure proves that even with boundary conditions both the FV and ST methods are of second order spatial accuracy. However, we can clearly notice that ST is still on average more accurate by a factor $\sim 8-10$ as compared to FV.

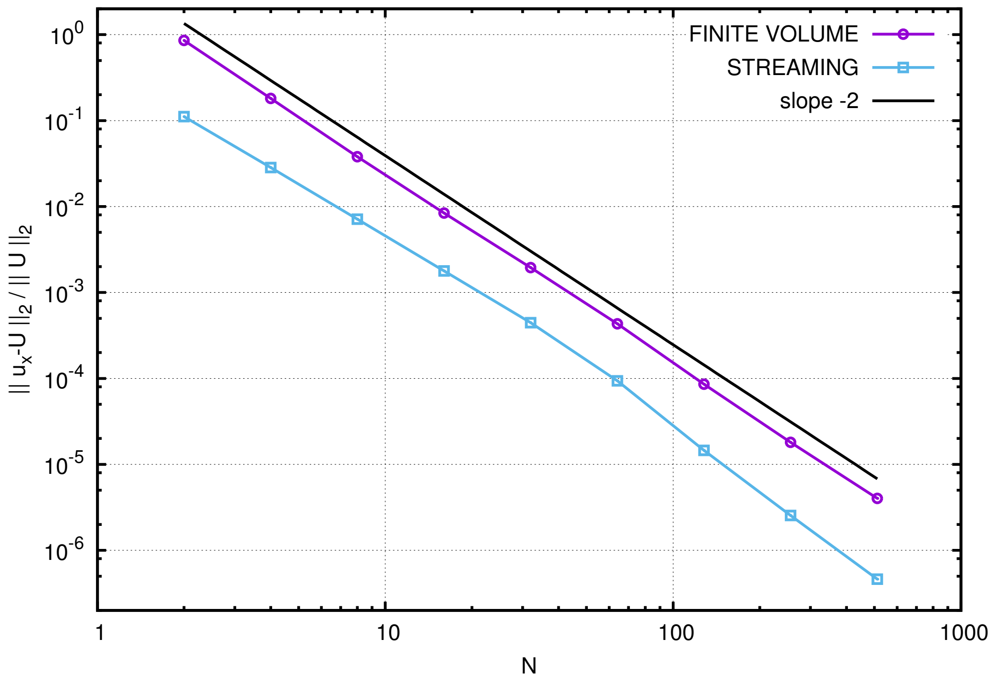


FIGURE 4.10: Relative error on the $Re = 10$ velocity Poiseuille flow profile at changing the number of grid points (N) and keeping fixed the grid spacing $\Delta x \equiv L_y/N_y = 1$. Proof that FV is same order in space of ST but less accurate of a factor 8 to 10.

As further step, we address the effect of a stretched spatial grid on the overall accuracy of the Poiseuille flow simulation. For wall boundary conditions, it is very necessary to have near wall refinements.Therefore, we implement three types of commonly used wall-normal stretched grids. Pictorial description of such stretched grids is shown in figure 4.12. In this context, grid size $N = 128$ has been used as an example.

Note: *Refined grids are subjected to extra stability constraints. It demands smaller time-step due to the decreased grid space near the walls. CFL criteria is a useful parameter that governs this stability criteria.*

$$CFL = c_i \frac{\Delta t}{\Delta x} \leqslant 1$$

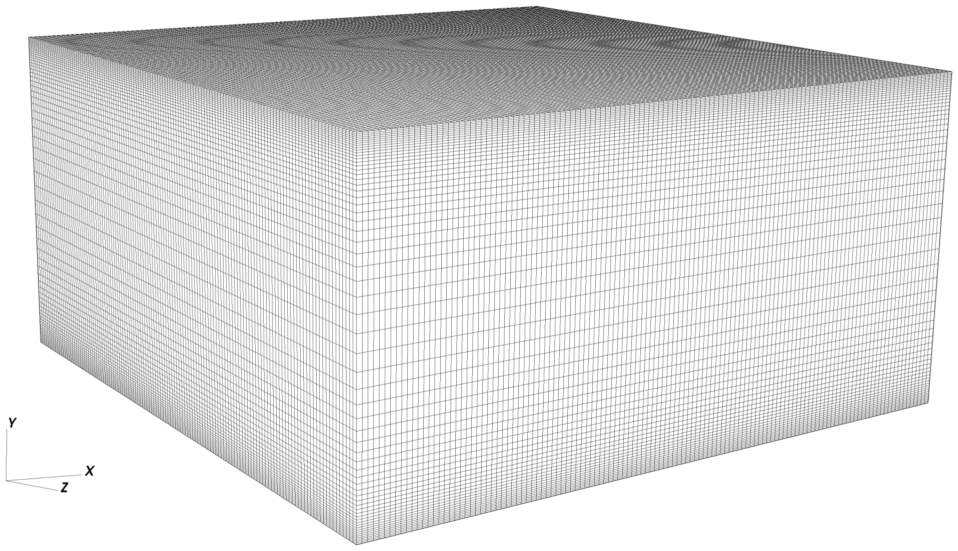


FIGURE 4.11: Pictorial representation of the grid refinement in the y-direction of a cuboid.

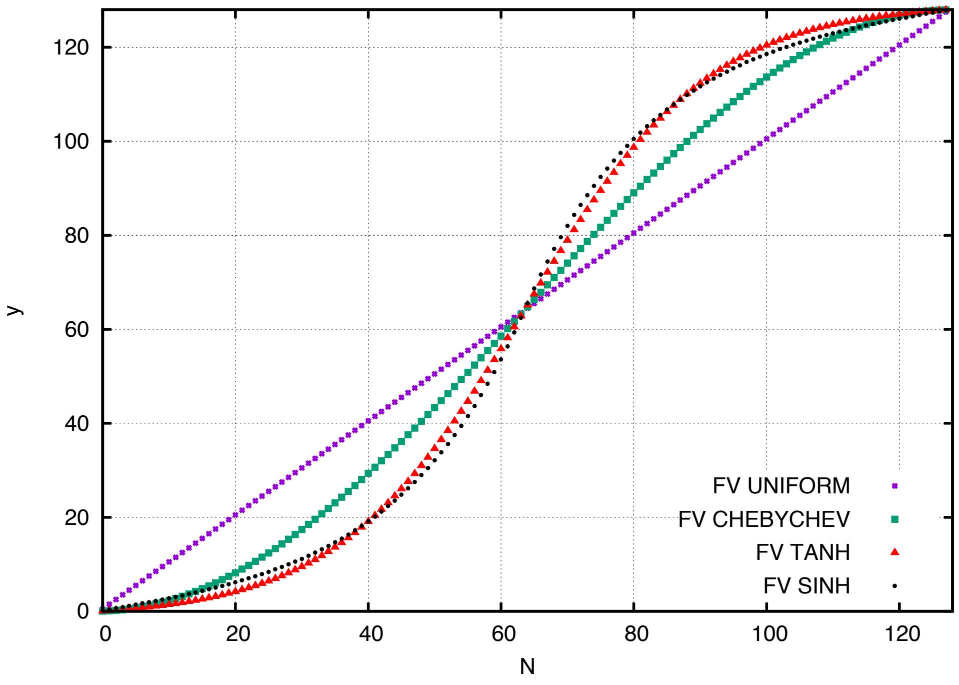


FIGURE 4.12: y-coordinate value of the cell volume centers (y) *vs.* grid size (N). Comparative representation of the position of nodes (y direction) of different grid refinements in a $[1, 128, 1]$ spatial domain for a wall bounded Poiseuille flow case.

It is understood that time-step is directly proportional to the smallest grid space obtained after refinement.

The y-coordinate value of the cell volume centers (or simply nodes) is given by

$$y_i = \frac{\xi_{i+1} + \xi_i}{2} \qquad \text{with } 0 \leqslant i < N_y$$

where the ξ_i, the coordinates at the volume boundaries, are defined as

$$\text{Chebychev nodes:} \quad \xi_i = \frac{L}{2}\left(1 - \cos\left[\frac{(i\ -\ 1/2)\pi}{N}\right]\right) \qquad \text{where } 0 \leqslant i \leqslant N \tag{4.25}$$

$$\text{hyperbolic tangent:} \quad \xi_i = \frac{L}{2}\left(1 + \frac{1}{s_1}\tanh\left[(\frac{2}{N}\ i - 1)\ \text{atanh}(s_1)\right]\right) \text{where } 0 \leqslant i \leqslant N \tag{4.26}$$

$$\text{hyperbolic sine:} \quad \xi_i = \begin{cases} \left(\frac{L/2}{\sinh(s_2/2)}\right)\sinh\left(\frac{s_2\ i}{N}\right), & \text{if } 0 \leqslant i \leqslant \frac{N}{2} \\ L - \left(\frac{L/2}{\sinh(s_2/2)}\right)\sinh\left(\frac{s_2\ (N-i)}{N}\right), & \text{if } (\frac{N}{2}+1) \leqslant i \leqslant N. \end{cases} \tag{4.27}$$

L and N denote here the total domain size and the grid size (the sub-script index y have been dropped for brevity), and s_1, s_2 are stretching factors (we have chosen $s_1 = 0.98$ and $s_2 = 6.5$). We then perform the same, $Re = 10$, Poiseuille flow simulation with the three above different grids with the FV method, and for completeness we also include the results obtained on a uniform grid by both the FV and the ST method. In order to have a better understanding on the accuracy of the methods this time we change the number of grid points (N) while keeping fixed the domain size $L = 64$. In other words what we vary here is the average grid spacing $\langle\Delta x\rangle = L/N$.

Figure 4.13 reports the results of the described test. It shows that there exist an optimum value of N for which the error is minimum, this happens both for the FV and ST algorithms, both on uniform and on stretched grids. The ST method (which can only be based on a uniform grid) performs better than the uniform-grid FV implementation for almost all the values of N, furthermore its absolute accuracy is the highest. However, the situation becomes interesting when the non-uniform grid FV method is employed. There we notice that one can get the same accuracy of the ST algorithm but with a smaller amount of grid points. For instance, in the case of the hyberbolic-tangent grid with $N = 11$ one can obtain the same accuracy as the ST algorithm with $N \simeq 46$. *This leads to a saving in memory and potentially in computational costs. In conclusion, the reduction in memory occupation at comparable accuracy seems to be the main benefit one can get from employing the wall-refined FV method rather than the standard ST.*

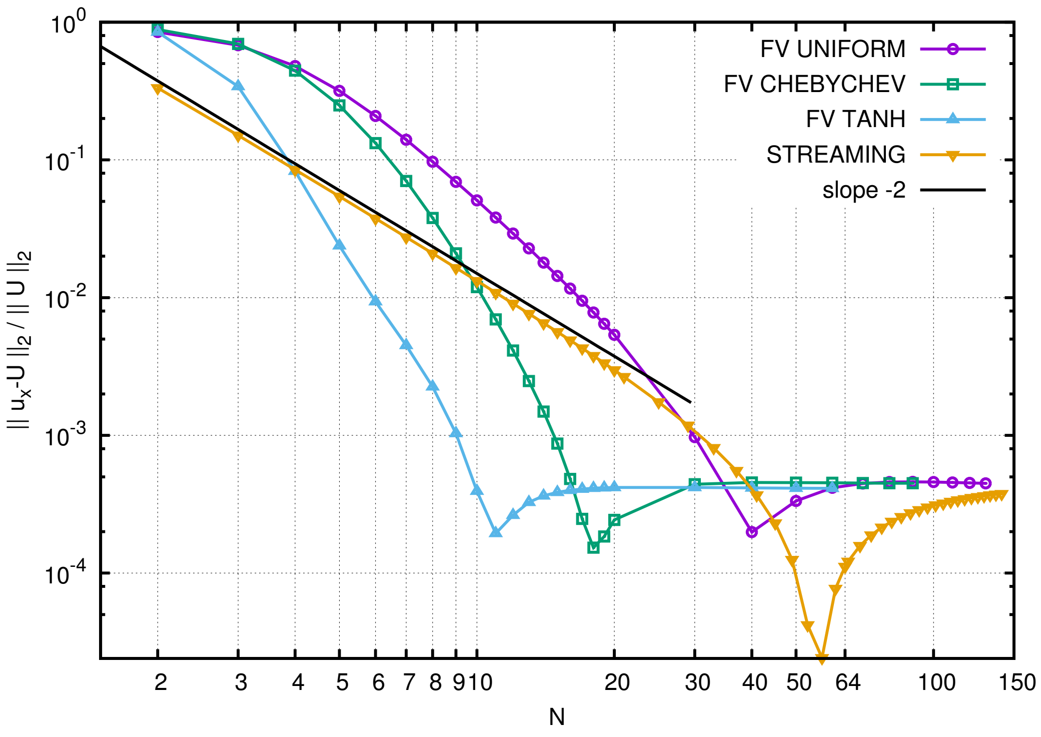


FIGURE 4.13: Relative error on the $Re = 10$ velocity Poiseuille flow profile at changing the number of grid points (N) and keeping fixed the domain size $L = 64$. Data are traced for FV with uniform grid, with Chebychev points and hyperbolic tangent grid refinement, and finally for the ST algorithm with uniform grid.

However, a situation that often occur in the simulation practice is that one wants to use all the available memory of a computer and using an algorithm with the best possible accuracy. The interesting question is then: How can we increase the accuracy at comparable memory costs? Let's imagine one wants to perform again the same Poiseuille simulation at $Re = 10$ but wants to reach a higher level of accuracy (with accuracy defined in the sense of L_2 norm). One new possibility is to adjust L and U_{max} in a way that the averaged grid spacing $\langle \Delta x \rangle = L/N$, with N left unchanged, is the one that offers the best accuracy performance for a given grid. For the above case of the hyberbolic-tangent grid this would be around $\langle \Delta x \rangle = L/N = 64/11 = 5.8$. The result of this novel Poiseuille flow test is shown in figure 4.14. We can see that when $N = 64$ the best choice is to adopt a grid with *tanh* or Chebychev spacing and with $\langle \Delta x \rangle$ much larger than unit. *Here the optimum is reached when $\langle \Delta x \rangle \simeq 20$, this produces an increase in accuracy of a factor greater than 100 compared to the case of a simulation with the ST algorithm (and this independently of the value of $\langle \Delta x \rangle$ chosen for the ST method).*

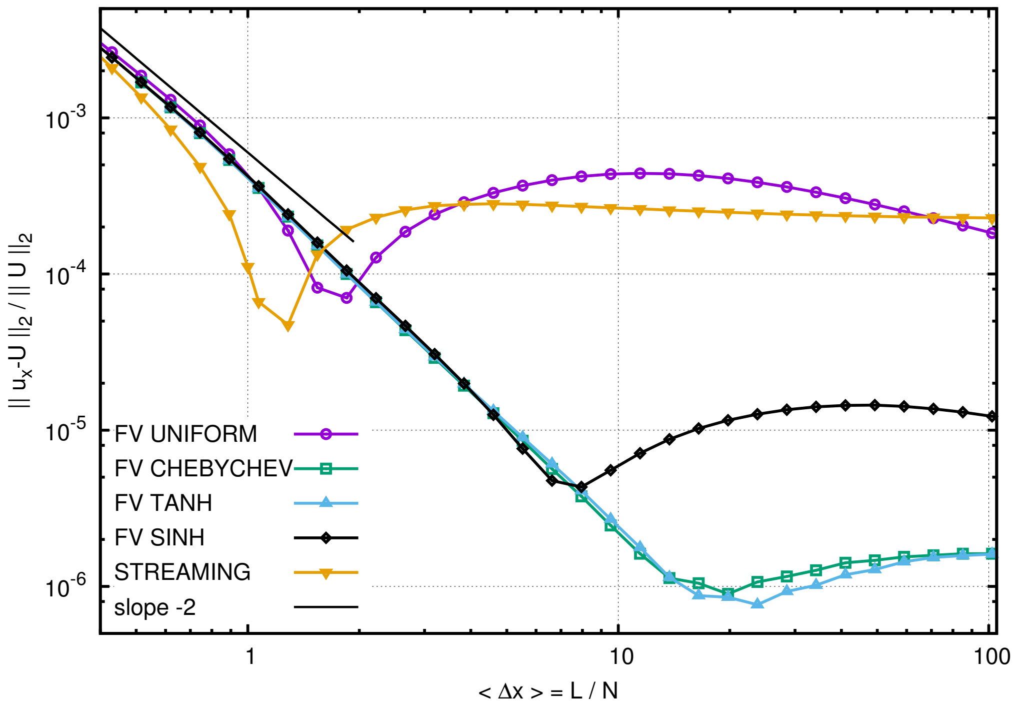


FIGURE 4.14: Relative error on the $Re = 10$ velocity Poiseuille flow profile at changing simultaneously the domain size L and the forcing amplitude, but keeping constant the number of grid points $N = 64$. Here the data are plotted versus the average grid spacing $\langle \Delta x \rangle = L/N$. Data are traced for FV with uniform grid, with Chebychev points, hyperbolic tangent and hyperbolic sine grid refinement. The corresponding relative error for the ST algorithm with uniform grid is also traced.

4.3 Performance evaluation

From a computational point of view the FV algorithm has more operations per time step than the ST algorithm. This comes from the fact that while the streaming process can be implemented simply as a shift in memory the computation of the flux term in FV involves many arithmetics operations. According to our measurements, the present FV algorithm is about 8-10 times computationally more expensive than ST algorithm per time step. However, as discussed in section 4.1.2, differently from the ST algorithm, in the FV the time-step Δt is a function of τ. The functional relation linking the maximum time-step to τ for the proposed time-discretizations can be measured and it is reported in figure 4.15. We observe that the method based on equation 4.16, semi-implicit integration in time of the collision term plus a trapezoidal correction for the advection is superior to the others. In particular, for this method $\Delta t_{max} > 1$ for $\tau > 1/8$, that is to say that *the time-step can be larger than the one used in the ST method (which is bounded to the value 1 for $\Delta x = 1$)*. The most advantageous case occurs for $\tau \sim 1/2$ - which as we have shown above is also the best condition for accuracy - in that case $\Delta t_{max} \sim 1.7$. This reduces the ratio of the computational cost FV/ST to a factor 5-6. We note that all this reasoning did not take into account the effect of non-uniform grids. As we have seen for the simple Poiseuille flow this brings further saving in terms of computational costs as compared to the ST method.

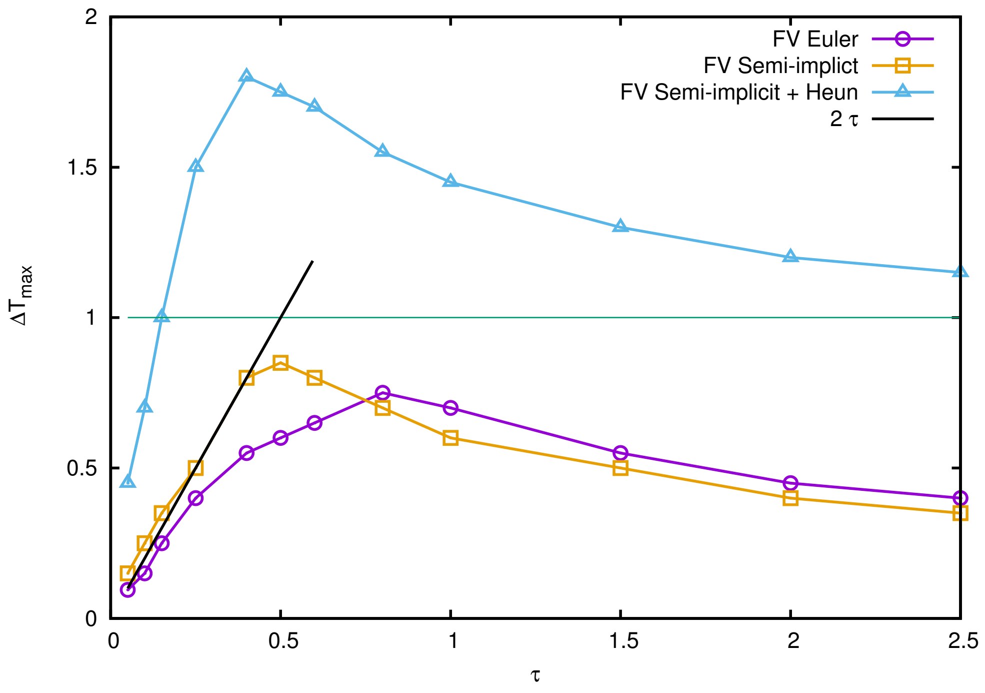


FIGURE 4.15: Maximum allowed time step in the decaying laminar Kolmogorov flow by using eq. (4.3) (FV Euler), eq. (4.15) (FV Semi-implicit) and eq.(4.16) (FV Semi-implicit + Heun). The very same result is obtained in the steady Poiseuille flow at $Re = 10$. The horizontal dashed line represents the standard choice of the time-step for the ST implementation, *i.e.*, $\Delta t = 1$ independently of τ.

Effect of changing the collision term

As we have observed so far, the stability constraint of our FV algorithm seems to be imposed essentially by the discretization of the advection term in the discrete velocity Boltzmann equation. This means that it is not dependent on the collision term. In order to better prove this observation we perform a test where the BGK collision kernel is replaced by a multiple relaxation collision term.

For simplicity we adopt the TRT(Two relaxation time) model proposed by I. Ginzburg in [109, 110]. TRT model uses just two relaxation rates w_+ and w_-. If we revise section 1.4.3.1, the constraints applied to velocity set in single relaxation time model resulted odd moments of f_i^0 to vanish. In TRT, odd moments do not vanish and are relaxed with w_- relaxation rate. The even moments are relaxed with w_+ relaxation rate. The relation of these two relaxation rates is given by a so-called magic number, Λ.

$$\Lambda = \left(\frac{1}{w_+} - \frac{1}{2}\right)\left(\frac{1}{w_-} - \frac{1}{2}\right)$$

There is a list of best magic parameters. For best stability for LB equation, the value of the magic number is taken as 1/4. Similarly, for best advection $\Lambda = 1/12$; for best diffusion $\Lambda = 1/6$; for obtaining exact location of bounce-back walls for the Poiseuille flow $\Lambda = 3/16$; *etc.*

To implement TRT, one needs to decompose populations in positive and negative parts:

$$f_i = f_i^{(+)} + f_i^{(-)} \qquad f_i^0 = f_i^{0(+)} + f_i^{0(-)}$$

Using these notations, FV LB equation 4.16 transforms to (for simplicity, consider the advection and the forcing term to be ϕ),

$$\tilde{f}_i^{(t+\Delta t)} = \tilde{f}_i + \Delta t\ w_+(\tilde{f}_i^{0(+)} - \tilde{f}_i^{(+)}) + \Delta t\ w_-(\tilde{f}_i^{0(-)} - \tilde{f}_i^{(-)}) + \phi$$

The test shows that simulations of FV LB equation with TRT model are always unstable for $\tau \leqslant 0.2$. For $\tau > 0.2$, the maximum allowed time-step is lesser than the value for *"FV Semi-implicit + Heun"* in figure 4.15. This implies that in our FV algorithm other forms of collision model are disadvantageous than BGK collision model. This proves our statement that stability constraint of the proposed FV algorithm is mainly imposed by the advection term.

Memory allocation

Finally we shall mention that memory occupation is also part of the performance of an algorithm: According to our estimate FV in the present formulation needs twice more memory allocation with the same grid size as compared to the ST. However due to grid refinement, FV requires less grid points than ST to obtain the same accuracy and thus reduces memory allocation.

Chapter 5

FV LB method at work: Application to thermal convection

5.1 Benchmark in a complex flow: high Rayleigh number thermal convection

Several LB Finite-Volume methods proposed in the past have been tested just on laminar flows as proof of principle of the proposed algorithms. Few exceptions exist in the literature in which the FV method have been benchmarked on much more complex, three-dimensional, developed turbulent flows. One of such exception is the model proposed by Amati *et al.* [7], which was probed in a three-dimensional plane turbulent channel flow. In such case however, the grid wall refinement was based on a very simple structure of halved-grid spacing near the walls and the accuracy of the method turned out to be not satisfactory (the computed mean-velocity profile could not properly reproduce the log-law of the wall).

In this section the proposed Lattice Boltzmann FV algorithm is tested to simulate a complex three-dimensional statistically steady turbulent flow. Our choice is here for the well-studied flow in the Rayleigh-Bénard (RB) cell, the prototype of thermal convection driven system [111]. The RB set-up considered in this study deals with a cuboid domain (of height H and equal lateral sizes L); it has periodic BC on the lateral walls, while on the horizontal walls no-slip and isothermal conditions are imposed. In this system, the fluid is heated from below and as such (when the heating is large enough and a small perturbation is introduced in the system) an instability arises and brings the system into convective condition. The dimensionless control parameters are the Rayleigh (Ra) and Prandtl (Pr) numbers and aspect-ratio $Ar = L/H$ [111].

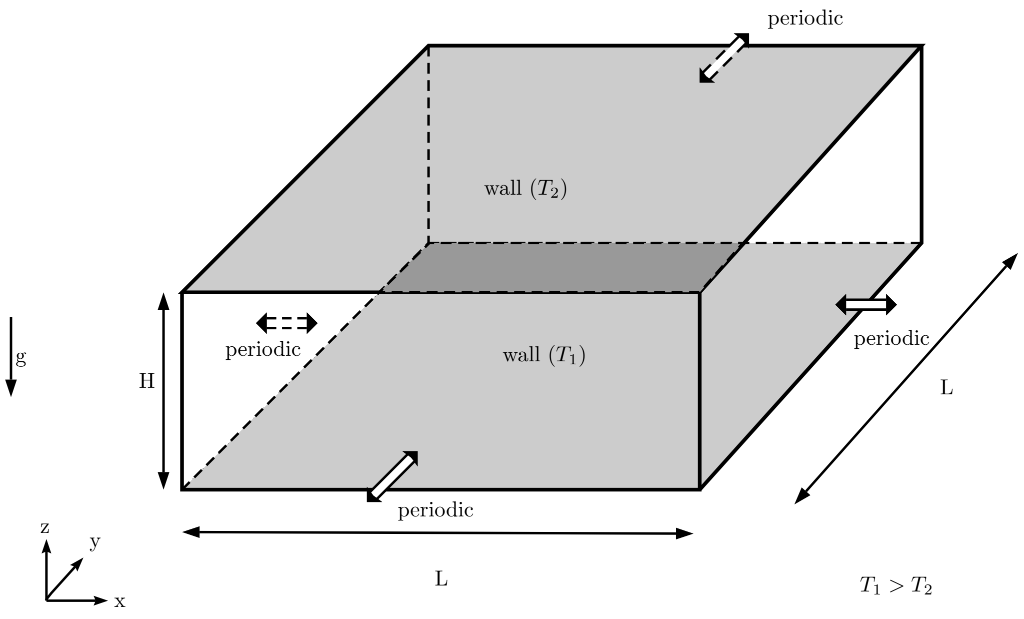


FIGURE 5.1: Cartoon of the three-dimensional Rayleigh-Bénard system.

For the LB simulation we use a double population approach [61]. This is the other type than the multi-speed thermal model used by Sbragaglia and Sugiyama [101] (mentioned in the review of FV approach (Chapter 4)). The choice of passive scalar approach (also called double population approach) is because it has better numerical stability than multi-speed thermal models. In a double population approach, beside equation 1.30 we integrate an analogous equation for the distributions g_i:

$$\frac{\partial g_i}{\partial t} + \mathbf{c}_i \cdot \nabla g_i = \frac{1}{\tau_g}(g_i^0 - g_i) \qquad \text{with } i = 0, \ldots, N_{pop} \tag{5.1}$$

with equilibrium function $g^0_\alpha = (T/\rho)\ f^0_i$ where the macroscopic temperature is computed as $T = \Sigma_i g_i$ and the thermal diffusivity corresponds to $\kappa = \tau_g\ c_0^2$. Furthermore, in the equation for f_i the forcing term F_i is assigned in order to model the buoyancy force as represented in the Boussinesq approximation. In physical space the added buoyant acceleration has the form $\mathbf{a} = -\beta(T - T_0)\mathbf{g}$ where β is the volume thermal expansion coefficient and T_0 is a reference temperature taken here as the mean temperature between the ones at the top and bottom plates.

In order to validate the double population approach also for the FV method, we first address a rather elementary simulation in steady convective laminar conditions, adapting it from a test case already conducted for the ST algorithm in Ref. [61]. The system is two-dimensional (2D) with control parameters fixed at $Ra = 10^4$, $Pr = 1$ and $Ar = 2.02$. The fluid is initially at rest ($\mathbf{u} = 0$), while the temperature field is initialised by a linear conductive profile, $T_c(z) = -\Delta T(z/H + 1/2)$, plus a small perturbation (of order $O(10^{-2})\Delta T$) breaking the left right symmetry. Given the weak, but not negligible,

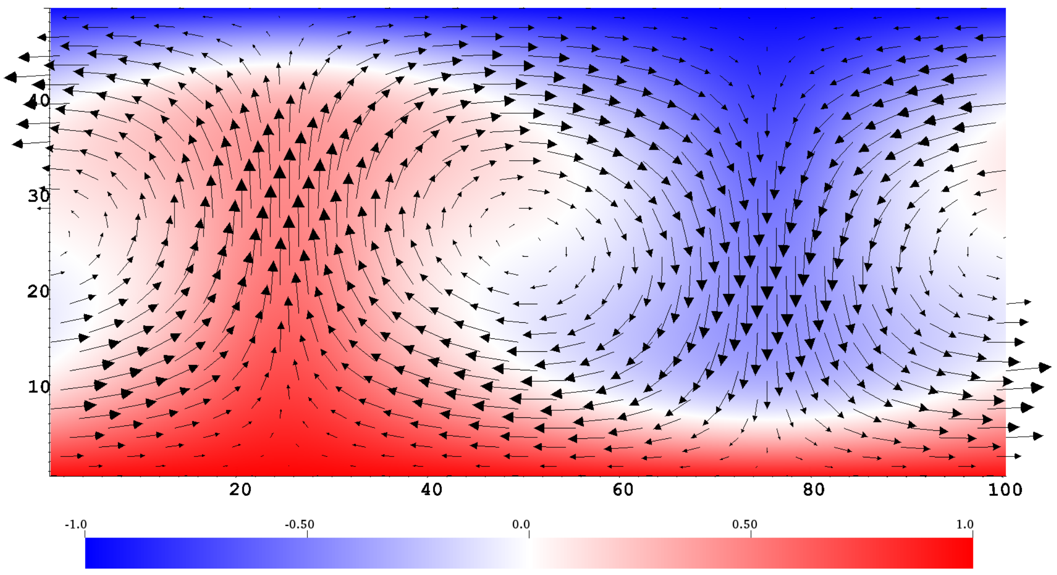


FIGURE 5.2: Finite volume LBM: Temperature field contours supported with velocity vectors for a Rayleigh-Bénard system characterised by control parameters $Ra = 10^4$, $Pr = 1$ and and $Ar = 2.02$

compressibility of the simulated flow the initial density stratification due to gravity should be also taken into account. This avoids the generation of pressure waves at the startup of the simulation. We do it via the barometric equation, this leads to $\rho(z) = \rho_0 \exp\left(-c_s^{-2}\beta g \int_0^z T_c(z')dz'\right)$, where ρ_0 is a reference density value taken at temperature T_0.

Note that in a 2D system, in order not to suppress the linear hydrodynamic instability, the cell aspect ratio (Ar) must be slightly larger than $2\pi/k_c$ (where $k_c = 3.117$ is the wave vector of the most unstable linear mode) [61]. Indeed, when $Ar = 2.02$ the initial perturbation produce an immediate kinetic energy growth and a steady convective flow pattern establishes. The dimensionless heat flux (or Nusselt number Nu) goes from the conductive unit value up to around $Nu \simeq 2.66$, see [19]. In figures 5.3 and 5.4 we report the results of simulations conducted with the two LB algorithms. We can observe (figure 5.3) that the temporal dynamics of the dimensionless global heat flux,*i.e.* the Nusselt number $Nu(t)$, is identical for the two simulations, furthermore they both agree with the analytical asymptotic value given by Clever and Busse [19]. The isocontours lines for the temperature field (figure 5.4) further display the excellent agreement between the two LB algorithms. The test exhibits not only the good quality of the present FV method but also its consistency with the standard ST method also for transient (*i.e.* time-dependent) dynamics.

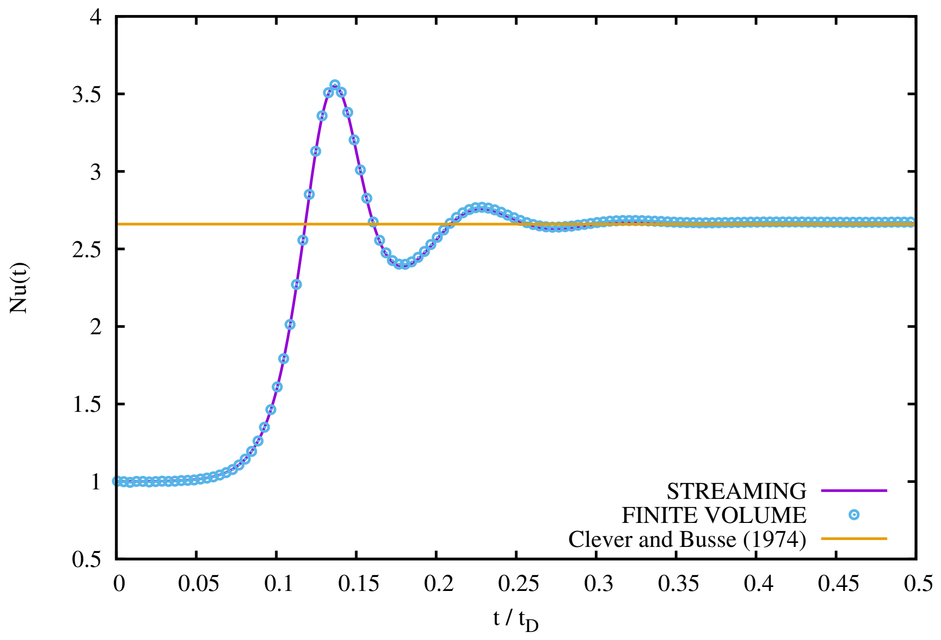


Figure 5.3: Comparison of streaming (ST) and finite-volume(FV) LB algorithms in a simulation of the Rayleigh-Bénard system in steady convective state. The system is two-dimensional, and characterised by the control parameters value $Ra = 10^4$, $Pr = 1$ and $Ar = 2.02$. Temporal dynamics of dimensionless global heat flux (Nusselt number $Nu(t)$) as a function of time, in dissipative time units $t_D = H^2/\kappa$. The steady state value is compared to Clever and Busse calculation [19].

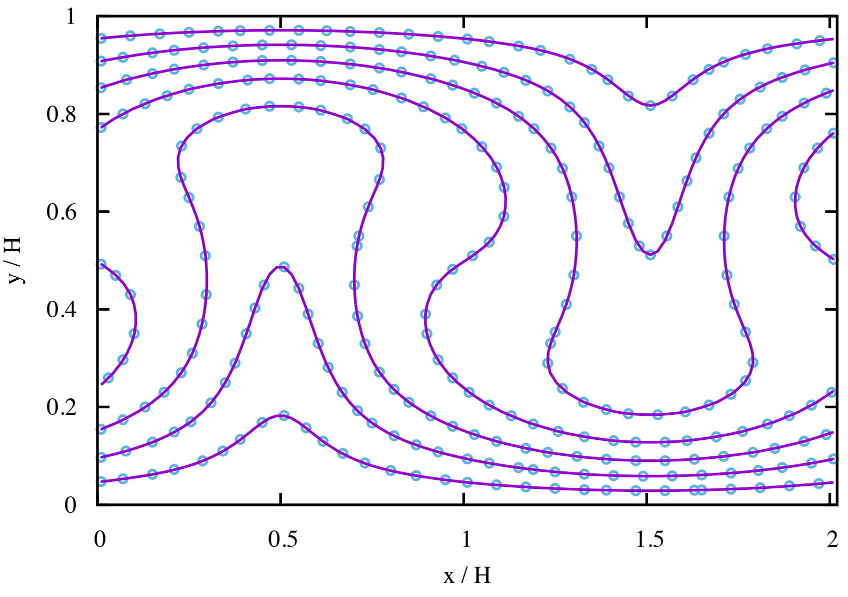


Figure 5.4: Comparison of streaming (ST) and finite-volume(FV) LB algorithms in a simulation of the Rayleigh-Bénard system in steady convective state. The system is two-dimensional, and characterised by the control parameters value $Ra = 10^4$, $Pr = 1$ and $Ar = 2.02$. Comparison of temperature isolines in the asymptotic steady state. Levels are taken at values $T_n = T_0 \pm n\ \Delta T/8$, with $n = 0, 1, 2, 3$.

	H	L	Δt	τ	τ_g	ΔT	β	g	t_{tot}
FV	640	1280	4	0.5	0.5	2	$1.325 \cdot 10^{-4}$	1	$1.28 \cdot 10^{6}$
ST	64	128	1	0.05	0.05	2	$1.325 \cdot 10^{-3}$	1	$2.48 \cdot 10^{5}$

Table 5.1: Parameter values used for the RB simulations at $Ra = 2.5 * 10^6$, $Pr = 1$ and $Ar = 2$: height (H) and width (L) of the cell, time step amplitude Δt, fluid (τ) and temperature (τ_g) relaxation times, temperature gap across the cell ΔT, thermal expansion coefficient value β and intensity of gravity g. t_{tot} is the total simulation time in numerical units.

We then move forward to a more complex case. In particular, we compare our results with the ones obtained by Kunnen *et al.* [112] for a three-dimensional (3D) simulation of a RB system (figure 5.1) characterised by: $Ra = 2.5 * 10^6$, $Pr = 1$ and $Ar = 2$ (see also [113]). In this condition the 3D system dynamics is already highly chaotic (or moderately turbulent). In [112] the authors employed a direct numerical simulation based on a staggered finite-difference discretization of the Navier-Stokes - Boussinesq equation system. The grid they adopted has size $(N_x, N_y, N_z) = (128, 128, 64)$, it is uniform in the horizontal directions and has a *sinh*-type refinement (the same as in 4.25) in the vertical direction. Our benchmark is as follow, we perform two series of simulations, one with the ST method and the other with the FV approach, the dimensionless parameters for the two cases are the same as the ones of Kunnen *et al.*, as well as the number of grid points per direction. However, while the ST uses a uniform grid in the FV case we use exactly the same grid as the one adopted in the finite-difference simulation [112]. The table 5.1 reports the numerical values of the parameters adopted for the two LB simulations. Note that the large scale velocity U which is roughly proportional to the so called free-fall velocity, *i.e.* $U \sim \sqrt{\beta g \Delta T H}$ is the same in both simulations. It is a good practice in LB simulations to always keep control of the large-scale velocity in order to prevent it to take too large values: it is worth reminding that in order to reproduce the incompressible fluid-dynamics the condition $U \ll 1$ is required (a commonly accepted rule of thumb in LB practice is $U \simeq 0.1$). In order to reach a good convergence of the statistical observables in the system the RB simulations are carried on for a total time (t_{tot}) which spans over several large eddy turnover times ($\mathcal{T}$). We estimate that $t_{tot} \simeq 12\ \mathcal{T}$ for both FV and ST simulations, with $\mathcal{T}$ computed from the zero-crossing time value of the autocorrelation function of the total kinetic energy.

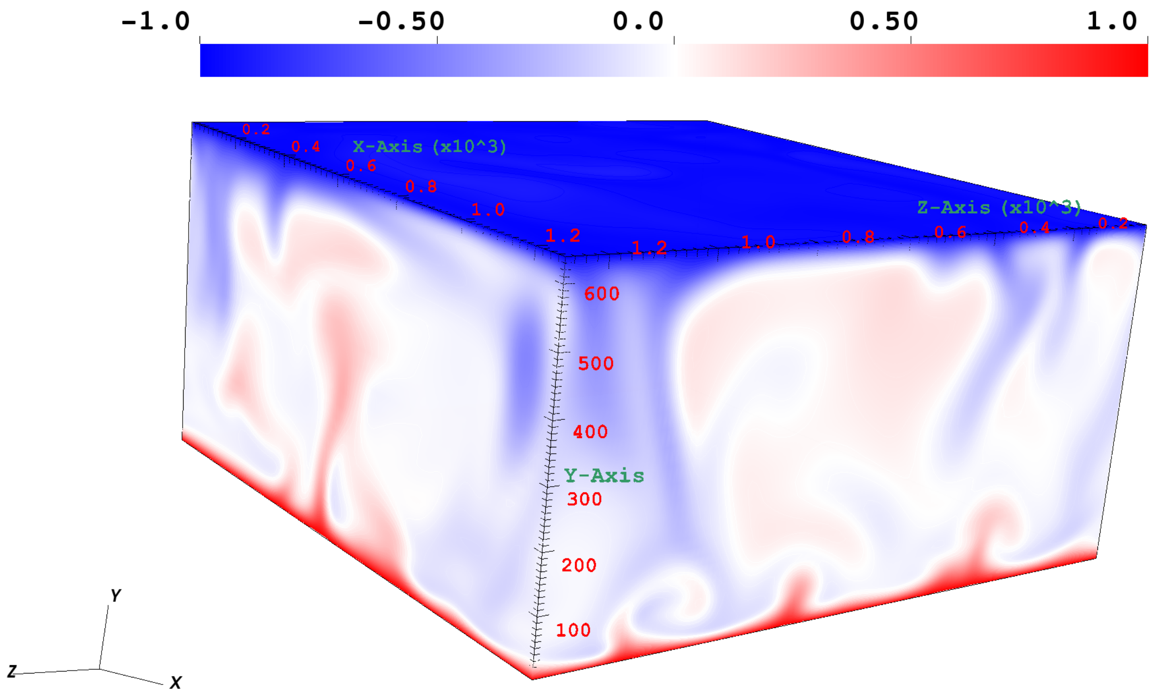


FIGURE 5.5: Finite volume LBM: Temperature field contours for a 3D Rayleigh-Bénard system characterised by control parameters $Ra = 2.5 * 10^6$, $Pr = 1$ and and $Ar = 2$

In the figures 5.6 and 5.7 we show a comparison of the vertical mean temperature profile (T_m) (averaged over horizontal planes and time) and of the vertical root-mean-square temperature (T_{rms}) profile, which are defined as follows:

$$T_m(z) = \frac{1}{t_{tot}\ L^2} \int_0^{t_{tot}} \int_0^L \int_0^L T(x,y,z,t)\ dx\ dy\ dt \tag{5.2}$$

$$T_{rms}(z) = \left(\frac{1}{t_{tot} L^2} \int_0^{t_{tot}} \int_0^L \int_0^L (T(x,y,z,t) - T_m(z))^2\ dx\ dy\ dt \right)^{1/2} \tag{5.3}$$

We find good agreement among all the three types of simulations. Furthermore, we observe that when the thickness of the boundary layer λ_T is defined by the so called slope definition, $\lambda_T \equiv \Delta T (2\ \partial_z T_m(z)|_{z=0})^{-1}$ (see figure 5.6) both the Kunnen *et al.* data and the FV ones have about 10 points in the thermal boundary layer (BL), while the ST despite its remarkable agreement with the other methods, has only 3 points in the BL. Small systematic differences can be seen on the vertical profile of the mean (turbulent) kinetic energy, $k_m(z) = t_{tot}^{-1}\ L^{-2} \int_0^{t_{tot}} \int_0^L \int_0^L \frac{1}{2}\mathbf{u}^2\ dx\ dy$, reported in figure 5.8. $k_m(z)$ has a slower rate of convergence than the temperature variance, this is the reason why small residual statistical discrepancies remain present here despite of the large number of turnover times of the simulation.

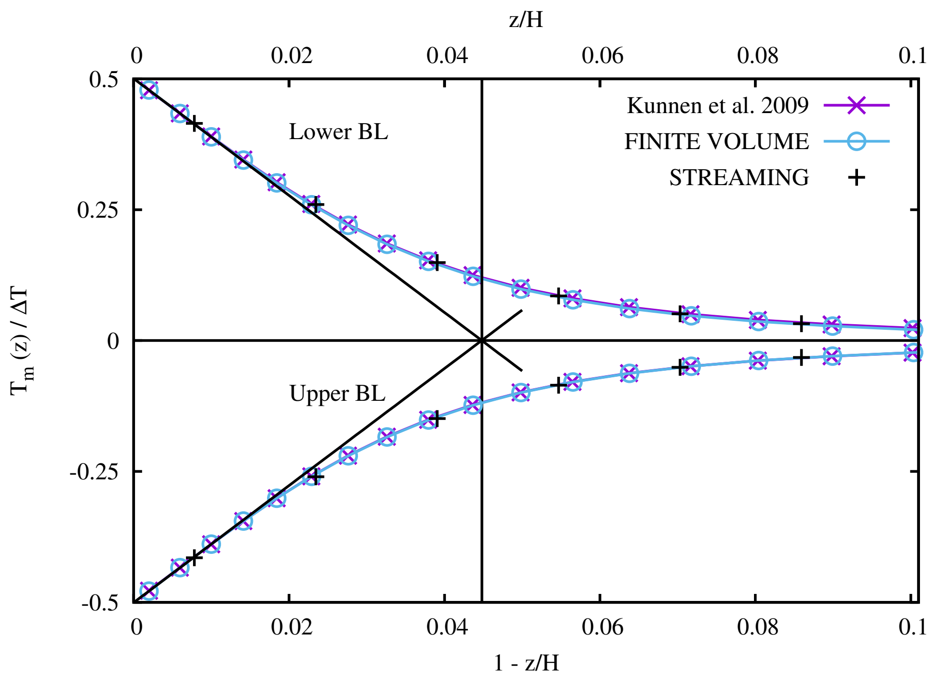


FIGURE 5.6: The mean temperature profile $T_m(z)$ - averaged over time and horizontal planes - as a function of the height z in the cell. To better appreciate the agreement between different simulation methods we show here a close-up view of the profiles in lower/upper 10% of the cell.

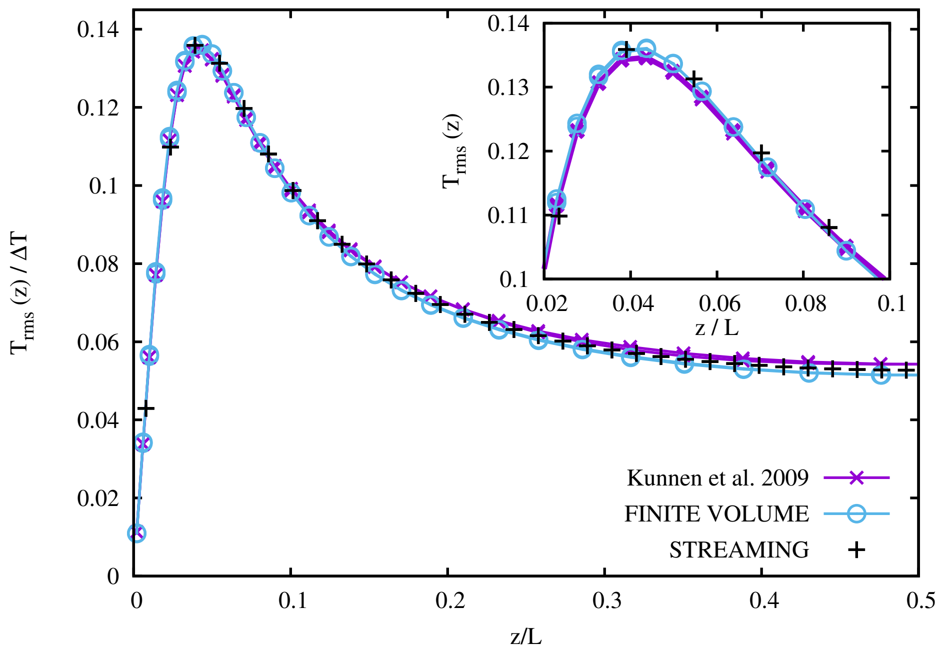


FIGURE 5.7: Root-mean-square temperature profiles T_{rms}, averaged over time and horizontal planes, as a function of the cell height z up to the cell center height. In the inset, a zoomed in view of the lower 10% of the cell.

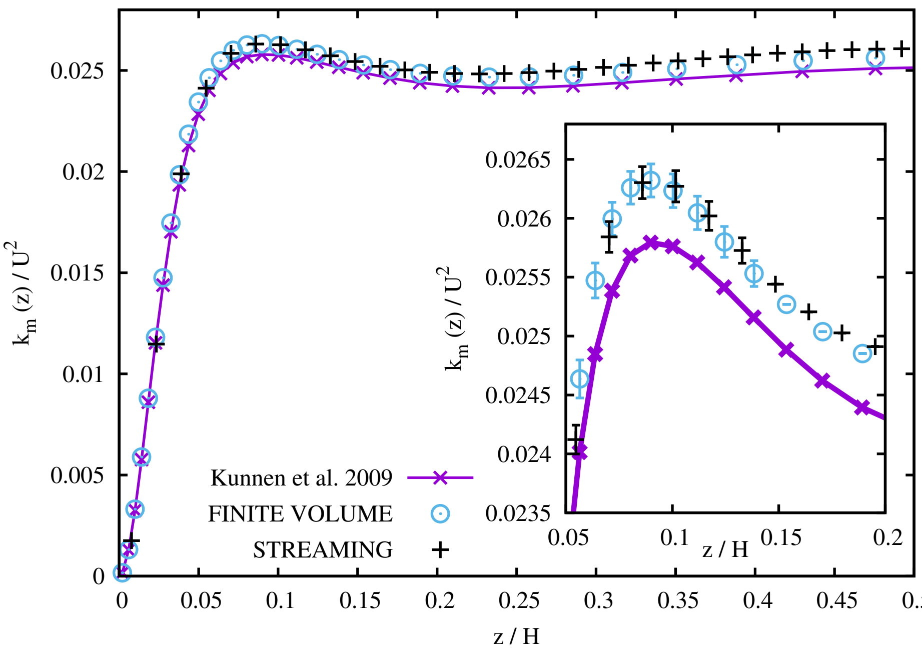


FIGURE 5.8: Mean turbulent kinetic energy $k_m(z)$, averaged over time and horizontal planes, as a function of the cell height z, and close-up view around the BL peak value (inset).

In order to appreciate more sensible differences between the FV and ST simulation one has to address either observables involving temperature-velocity correlation or small-scale quantities, which are more sensitive to the spatial resolution of the mesh, particularly close to the walls. For this reason in figures 5.9 and 5.10 we compare the time averaged quota-dependent Nusselt number $Nu(z) = \kappa\partial_z\langle T\rangle + \langle u_z T\rangle/(\kappa\Delta/H)$ (where for short $\langle\ldots\rangle$ denotes time and space horizontal averages) and the so called Bolgiano length $L_B(z) = (\beta\ g)^{-3/2}\langle\epsilon_u\rangle^{5/4}\langle\epsilon_T\rangle^{-3/4}$ (where ϵ_u and ϵ_T are respectively the velocity and temperature dissipation rates) [114]. The quantity $Nu(z)$, represents the mean heat flux across a horizontal plane in the system at height z from the bottom. Due to the energy conservation, it is expected to be constant (because heat is just transported across the cell but not created). Differently from the $Nu(z)$ the local Bolgiano scale $L_B(z)$ may vary across the cell. Such a length in fact represents a dimensional estimate of the energy injection scale in the RB system. While in non convective turbulent flows the forcing often occur at the largest scale, in a buoyancy-driven flow due to the active role of temperature, the force may act at smaller scales depending on the distance from the walls [115]. As far as Nusselt number is concerned, despite a very close mean value, we see important differences at the wall. This is due to the combined effect of the boundary conditions and the gradient computations in post-processing the data. *The FV method exhibit wall oscillations which are a factor 10 smaller than the ones seen for the ST method, making more reliable the total heat flux estimate.* Furthermore, in the $L_B(z)$ measure we observe near a 50% discrepancy at the wall and a smaller but non negligible difference in the bulk of the cell. Clearly a wall-clustered grid is needed to resolve observables built on sharp temperature and velocity gradients.

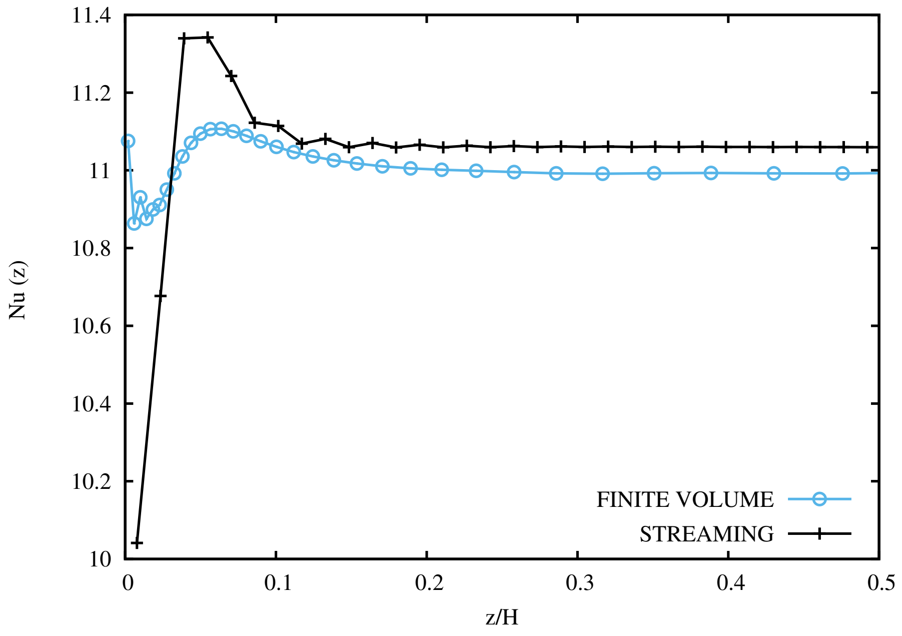


FIGURE 5.9: The time average Nusselt number $Nu(z)$ as a function of the height in the cell, up to half cell. Note that in a ideal RB system this quantity should be constant, however in numerics, often due to the effect of BC implementations, small fluctuations are observed through the cell. It is here evident the higher quality of the FV method.

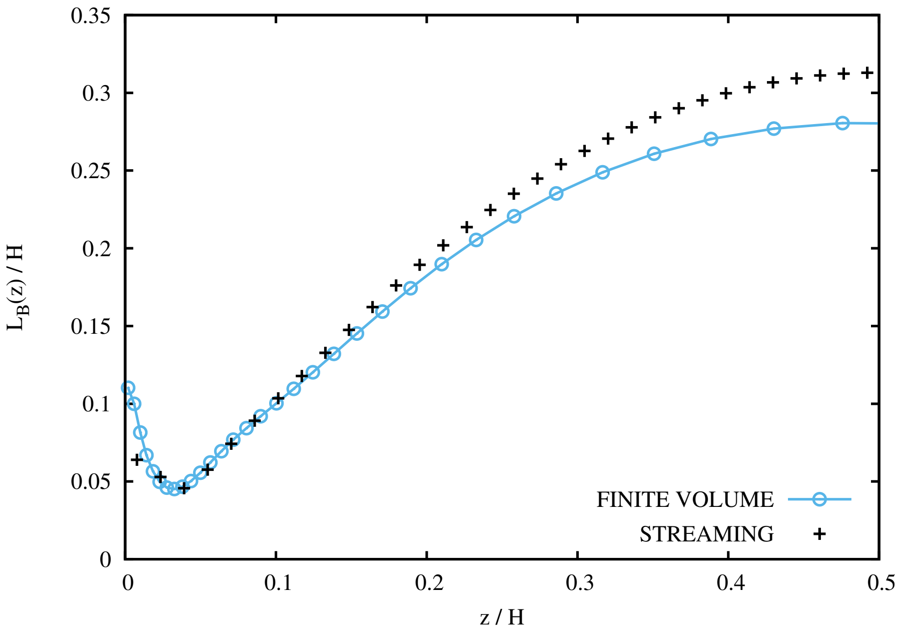


FIGURE 5.10: Bolgiano length, $L_B(z)$, averaged over time and horizontal planes, as a function of the cell height z. Note the discrepancies both at the wall and in the cell bulk.

5.1.1 Computational efficiency comparison in the high-Rayleigh number regime

We now would like to address the matter of determining which LB method is computationally more convenient. The choice to have the same U in the FV and ST simulations has an implication on the determination of the large-eddy turnover time and therefore the total number of time steps needed to perform a simulation of equivalent physical time-span. The reasoning is as follow: the large turnover time goes as $\mathcal{T} \sim H/U$ therefore on the total number of time-steps M for a simulation that should span a time $\mathcal{T}$ scale as $M \sim H/\Delta t$. It follows that the FV simulation will need in this case a number of time steps larger by a factor $10/4$ as compared to the ST one (see table 5.1). Since the FV is more expensive than ST by a factor $8-10$ per time step, we get that the added computational cost of the FV method is up to $\simeq 25$ larger than the ST method. However, such an increase in computational cost shall be properly weighted by the enhancement in the spatial resolution due to the wall stretched grid. An univocal guideline is not available in this context. A commonly employed criterion in the numerics of bounded flows is to count the number of grid nodes in the BL (another, although less restrictive, rule would be to take into account the distance of the first collocation point from the wall). Here, if we adopt such a criterion the ratio is in favour of the FV method over ST by a factor $10/3$, that means that we need approximately 3 - 4 more nodes in the ST simulation. However if we want also keep the same aspect-ratio of the simulation domain, and since ST is bounded to cubic grids, such an increase of the resolution shall be applied to every Cartesian direction, which makes the FV grid advantage greater of a factor of $(10/3)^3 \simeq 37$. In conclusion, in a ST RB simulation we need 37 times more computation nodes to perform a simulation with comparable resolution of the FV method. *By combining the above estimates, we see that a simulation of same physical time-span and same boundary layer resolution, is about* $1 - 25/37 \sim 33\%$ *less expensive for the FV method than the ST one.* In summary, even if the cost per unit physical time in a FV simulation is higher than ST, when a criterion for the minimal spatial resolution (particularly near walls or obstacles and in a three dimensional geometry) is chosen, the FV method becomes advantageous.

Finally, we have performed RB simulations at increasingly higher Ra numbers ($Ra = 10^8, 10^9$). All this simulations have around 10 grid points in the thermal BL as shown in the figure 5.11. No numerical instabilities were noticeable as Ra was increased (see figure 5.12 representing $Ra = 10^{10}$), demonstrating that the *FV algorithm can deal with turbulence at high Rayleigh number conditions.*

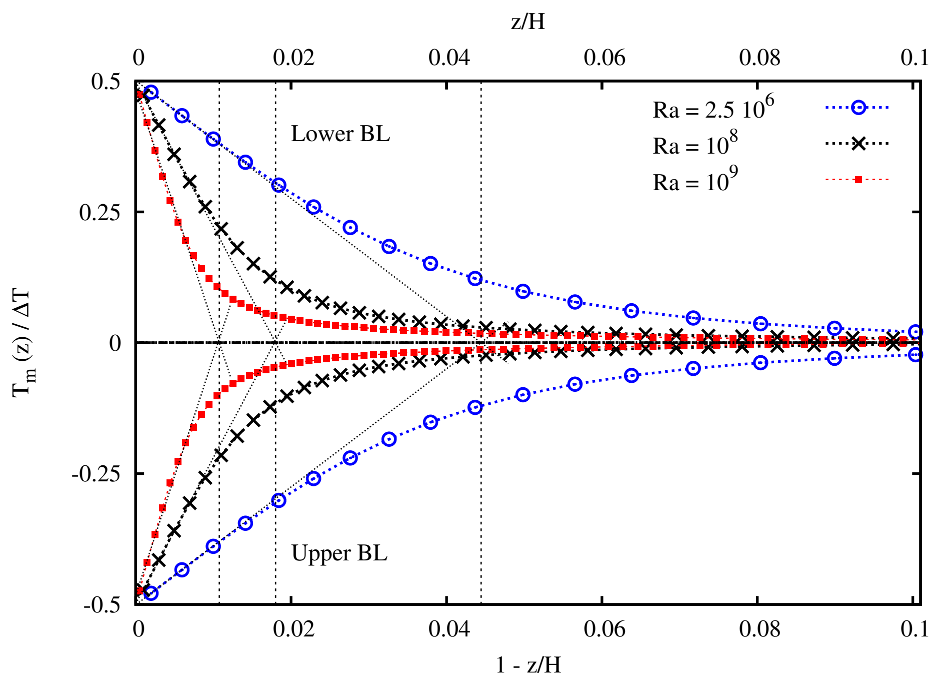


FIGURE 5.11: Mean vertical temperature profile, $T_m(z)$, close to the upper and lower plates in the Rayleigh-Bénard system at $Ra = 2.5 \cdot 10^6$ ($\circ$) , $Ra = 10^8$ ($\times$) and $Ra = 10^9$ ($\square$). The Prandl number is $Pr = 1$ and aspect ratio $Ar = 2$. The thickness of slope boundary layers is indicated by the dotted lines. Results were obtained by the FV algorithm at resolution 64×128^2, 128^3 and 256^3 respectively.

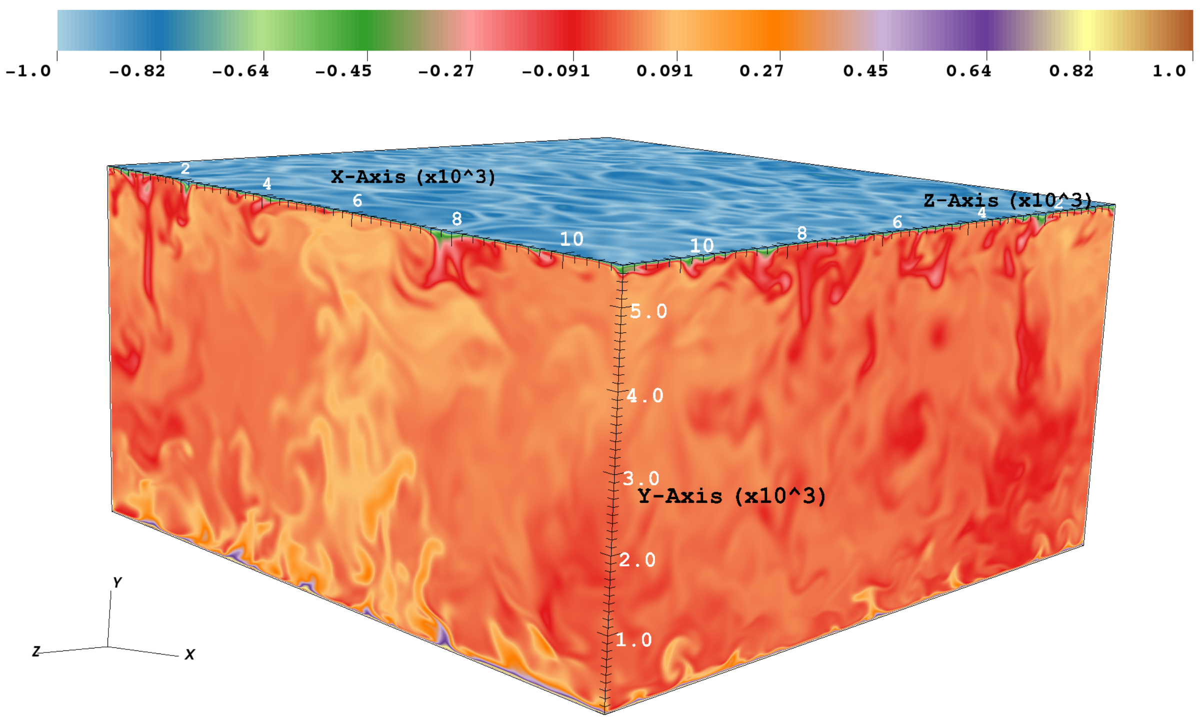


FIGURE 5.12: Finite volume LBM: Temperature field contours for a 3D Rayleigh-Bénard system characterised by control parameters $Ra = 10^{10}$, $Pr = 1$ and and $Ar = 2$.

5.2 Limitations of proposed FV LB Approach

It has been seen that the proposed FV LB Approach has great potentials in terms of increased accuracy, improved stability range and better computational efficiency than its predecessors. Successful implementation to a benchmark case of moderately turbulent, high Rayleigh number thermal convection has been demonstrated in the previous section. However, it is not free from shortcomings. In the previous chapter, we have demonstrated that we adopt a quadratic interpolation scheme for flux estimation (QUICK scheme) for good accuracy. However, figure 5.13 depicts that QUICK scheme alone could bring fluctuations at the boundaries failing to simulate highly turbulent flows (specifically channel flow turbulence). This directs for more work on this method to be able to stand competitive to the state-of-the-art CFD methods.

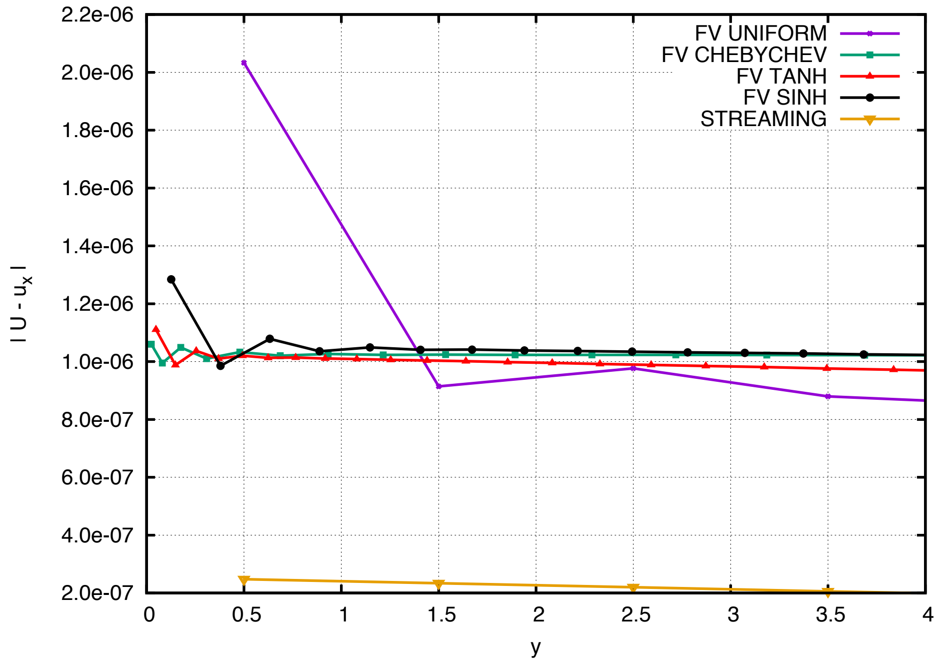


FIGURE 5.13: Relative difference ($|U - u_x|$) *vs.* y for different refinements. Zoomed near the boundary of the Poiseuille flow.

An effort to improve on this part is to use mixed schemes. Mixed scheme means that the upwind scheme is used at the boundaries since they are fluctuation-free and QUICK at the remaining domain. However, upwind scheme needs improvement because it is dissipative and inaccurate (refer figure 4.4). Therefore before using a mixed scheme, it is necessary to improve the upwind scheme to acceptable accuracy. The following section describes linearly reconstructed upwind schemes that are fairly accurate versions of upwind schemes. Linearly reconstructed upwind schemes have also been used in the past by Stiebler *et al.* [99].

5.2.1 Linearly reconstructed upwind schemes

The basic aim of advection schemes is to approximate flux at the cell boundaries. The flux could be approximated by either interpolation or extrapolation by using the information from the neighbouring cells. Taylor series expansion could be used to approximate the flux and represent some of the simple advection schemes.
Taylor series expansion about x_α (for the east boundary) gives (refer figure 5.14)

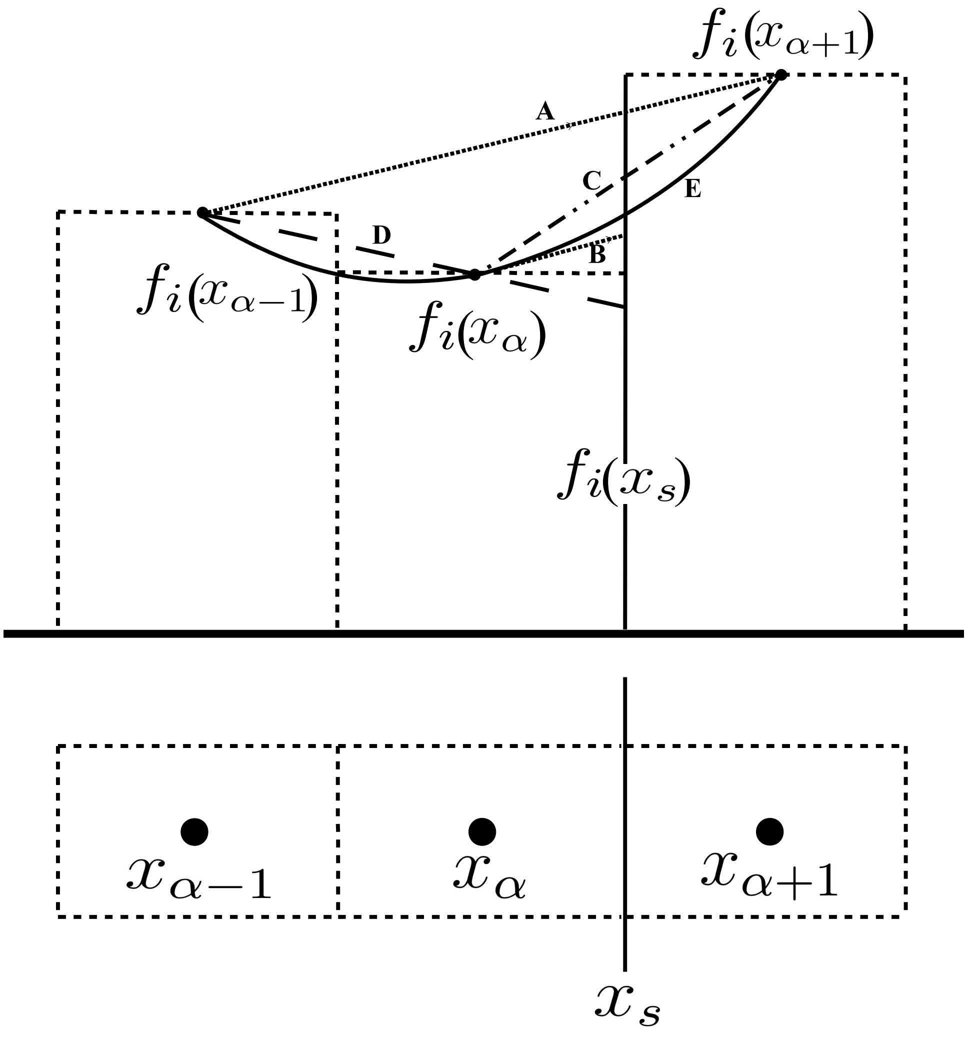


FIGURE 5.14: Schematic representation of the approximations at a cell interface using informations from the neighbouring cells

$$f_i(x_s) = f_i(x_\alpha) + (x_s - x_\alpha)\left(\frac{\partial f_i}{\partial x}\right)_\alpha + H$$

where, H means the higher order terms. Upwind differencing scheme could be achieved by considering only the first term on the right hand side. This is a first order scheme. Note that the upwind schemes are direction-dependent governed by the direction of the velocity vectors c_i. This is a piecewise constant reconstruction type which assumes the solution to be constant everywhere within the cell and represented by the value at the centroid.

$$f_i(x_s) = f_i(x_\alpha) \qquad if \quad \vec{c_i} \cdot \hat{\vec{n}} > 0$$

Second order accurate scheme could be derived by also considering the second term on the right side which represents a piecewise linear reconstruction type. This can be done by devising the gradient in the second term.

- If cells x_α and $x_{\alpha+1}$ are considered to devise the gradient, this leads to a center difference scheme. This exactly resembles the linear interpolation of x_α and $x_{\alpha+1}$ at x_s (represented by C in figure 5.14).
  If $\vec{c_i} \cdot \hat{\vec{n}} > 0$,
  $$f_i(x_s) = f_i(x_\alpha) + (x_s - x_\alpha)\left(\frac{\partial f_i}{\partial x}\right)_\alpha$$
  $$f_i(x_s) = f_i(x_\alpha) + (x_s - x_\alpha)\left(\frac{f_i(x_{\alpha+1}) - f_i(x_\alpha)}{x_{\alpha+1} - x_\alpha}\right)$$
  For uniform grid, $(x_{\alpha+1} - x_\alpha) = 2(x_s - x_\alpha)$. This implies, $f_i(x_s) = \frac{f_i(x_{\alpha+1}) + f_i(x_\alpha)}{2}$. For non-uniform grid, $f_i(x_s) = \lambda_{x_s} f_i(x_{\alpha+1}) + (1 - \lambda_{x_s}) f_i(x_\alpha)$ where λ_{x_s} is a linear interpolation factor.

- Taking cells x_α and $x_{\alpha-1}$, the gradient can be devised to resemble a linear extrapolation of $x_{\alpha-1}$ and x_α at x_s (represented by D in figure 5.14).
  If $\vec{c_i} \cdot \hat{\vec{n}} > 0$,
  $$f_i(x_s) = f_i(x_\alpha) + (x_s - x_\alpha)\left(\frac{\partial f_i}{\partial x}\right)_\alpha$$
  $$f_i(x_s) = f_i(x_\alpha) + (x_s - x_\alpha)\left(\frac{f_i(x_\alpha) - f_i(x_{\alpha-1})}{x_\alpha - x_{\alpha-1}}\right)$$

- Further, more accurate scheme can be developed by considering $x_{\alpha-1}$ and $x_{\alpha+1}$.
  If $\vec{c_i} \cdot \hat{\vec{n}} > 0$,
  $$f_i(x_s) = f_i(x_\alpha) + (x_s - x_\alpha)\left(\frac{\partial f_i}{\partial x}\right)_\alpha$$
  $$f_i(x_s) = f_i(x_\alpha) + (x_s - x_\alpha)\left(\frac{f_i(x_{\alpha+1}) - f_i(x_{\alpha-1})}{x_{\alpha+1} - x_{\alpha-1}}\right)$$
  The idea behind this type is that the slope of the line between $x_{\alpha-1}$ and $x_{\alpha+1}$ (represented by A) is equal to the slope between x_α and x_s (represented by B).
  $$\frac{f_i(x_s) - f_i(x_\alpha)}{x_s - x_\alpha} = \frac{f_i(x_{\alpha+1}) - f_i(x_{\alpha-1})}{x_{\alpha+1} - x_{\alpha-1}}$$

which simplifies to the above expression of $f_i(x_s)$. Curve E is a parabolic fit along $x_{\alpha-1}$, x_α and $x_{\alpha+1}$ which leads to very accurate, third order Quadratic Upwind Interpolation (QUICK scheme) and is marked as a reference in figure 5.14 .

To proceed with the analysis of the above mentioned linearly reconstructed upwind schemes, first let us take figure 5.15 which compares the accuracy of upwind scheme and its improved versions against QUICK scheme for a uniform grid. Also verified in figure 4.4, upwind scheme is highly dissipative and inaccurate. However, the linearly reconstructed upwind schemes show good improvement in accuracy. Moreover, the accuracy of upwind scheme of the type involving nodes $x_{\alpha-1}$ and $x_{\alpha+1}$ is very close to the reference QUICK scheme.

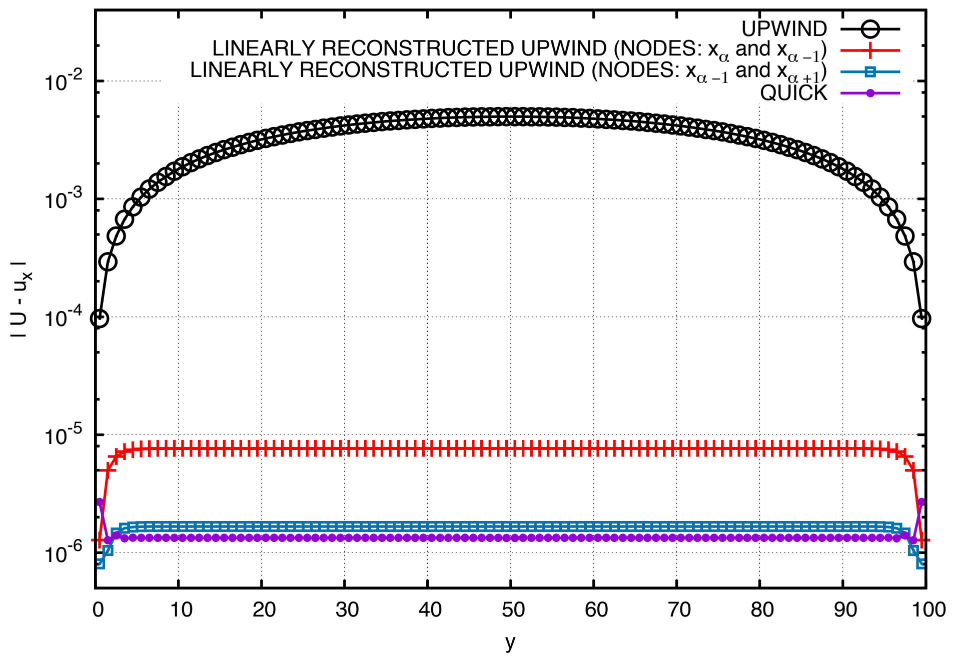


FIGURE 5.15: Poiseuille flow: Comparison of accuracy of the linearly reconstructed upwind schemes with respect to QUICK method for a uniform grid.

Proceeding further, figure 5.16 again compares the accuracy of the linearly reconstructed upwind schemes with QUICK scheme but for a refined grid with hyperbolic tangent refinement. Type of refinement is just an arbitrary choice to demonstrate the improvement in a refined grid. If we zoom in the area near the boundary (figure 5.17), it can be noticed that the improved versions of upwind schemes manage to reduce the fluctuations. High order QUICK scheme is dispersive in nature thus it introduces fluctuations at discontinuities (*e.g.* boundary wall). On the other hand, upwind scheme is dissipative in nature thus it has smooth solution at the boundary but is inaccurate. The linearly reconstructed upwind schemes improve in terms of accuracy and also maintain the property of smooth solutions near the boundary.

Expectedly, mixed scheme (linearly reconstructed upwind scheme at the boundary + QUICK scheme over remaining part of the domain) should be accurate enough and fluctuation-free near the boundary.

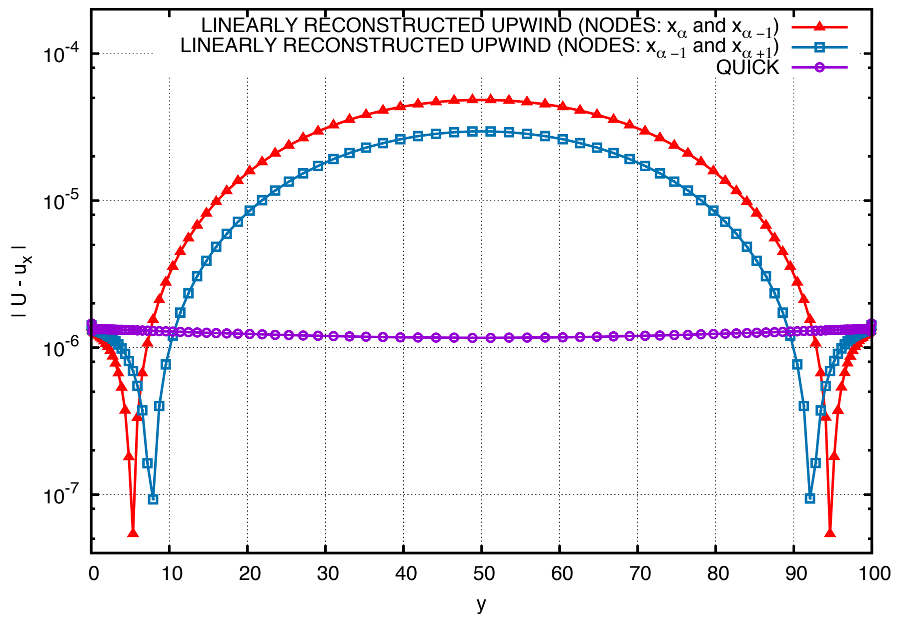


FIGURE 5.16: Poiseuille flow: Comparison of accuracy of the linearly reconstructed upwind schemes with respect to QUICK method for a refined grid with hyperbolic tangent grid refinement.

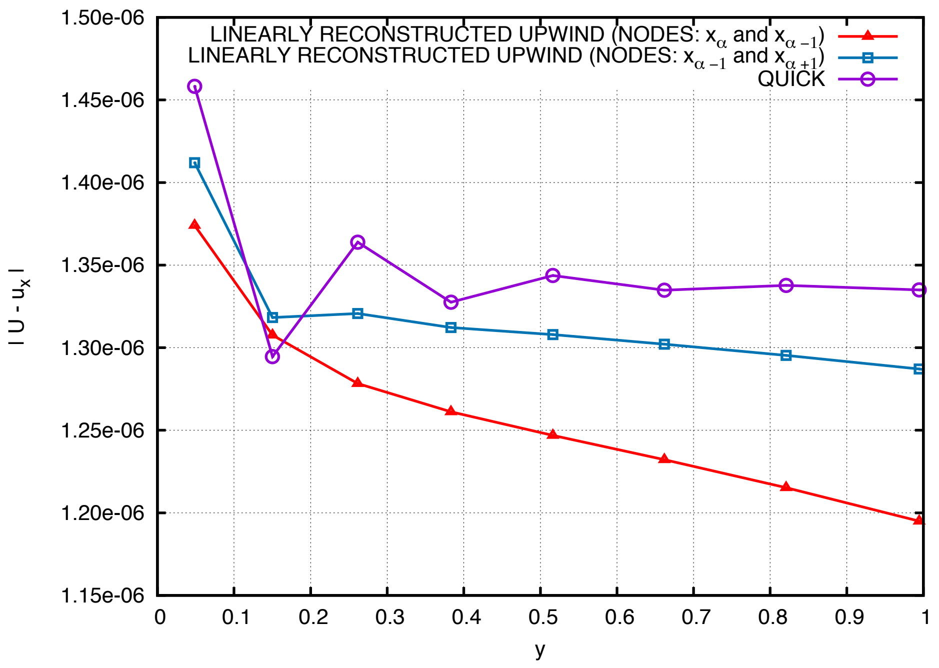


FIGURE 5.17: Poiseuille flow: Comparison of accuracy of the linearly reconstructed upwind schemes with respect to QUICK method for a refined grid with hyperbolic tangent grid refinement. *This is a zoomed view of figure 5.16 near the boundary.*

This mixed scheme was implemented for the simulation of turbulent channel flow. By the above explanations, the present situation should be able to tackle the fluctuations at the boundaries. As a result of such improvements to the advection scheme, the simulations could stand longer than previously but finally the simulation destabilised. As a concluding remark, such improvements in the advection scheme somewhat helps the simulation but is still not enough. Thus it directs us to see more areas which could cause the destabilisation of the simulation.

5.3 Conclusion

A new finite volume algorithm for the LB equation was proposed. The guidelines in the development of the new FV method were the simplicity and computational efficiency of the implementation, yet retaining a level of accuracy which takes the standard LB method as the baseline. Its most original features concern the semi-implicit approach taken for the time discretization which improved the stability range and the method of fluxes computation which adopted a QUICK scheme improving in terms of accuracy. The new method was validated through a systematic comparison with the standard streaming LB approach, by means of test case simulations in laminar as well as in unsteady and turbulent flows with heat transfer (3D Rayleigh Bénard system).
The tests showed that the FV has the same order of spatial accuracy as the ST algorithm. However for the same grid (uniform), it has much more elevated computational costs that we estimated to be around 8-10 times per time-step. This shortcoming could be addressed with the possibility to adopt stretched rectilinear grids for FV method. For the simulation of turbulent bounded flows, the number of grid nodes in the boundary layer is important to capture correct physics of the flow. In such cases, FV simulation would need approximately 3-4 less nodes than ST simulation. This leads to saving in memory and computational costs due to lesser number of grid nodes in FV simulations. It is to be noted that these results are obtained without compromising in accuracy. At the best condition of $\Delta x \simeq 20$, the accuracy of FV simulation could increase by a factor of 100 compared to the ST simulation.
Also, it was noticed that at the most advantageous case of $\tau \sim 1/2$ (which is also the best condition for accuracy), $\Delta t_{max} \sim 1.7$! Taking this into account, *i.e.* taking into consideration (for instance) the minimal number of collocation points required in a boundary layer for a proper simulation and the maximum allowable time-step, the FV algorithm surpasses the ST method. It was seen that a simulation of same physical time-span and same boundary layer resolution, was about $1 - 25/37 \sim 33\%$ less expensive for the FV method than the ST one. However, its applications are still limited and are unable to simulate benchmark problem of fully-developed turbulent channel flow.

Part II

Chapter 6

Description of the developed Lattice Boltzmann code

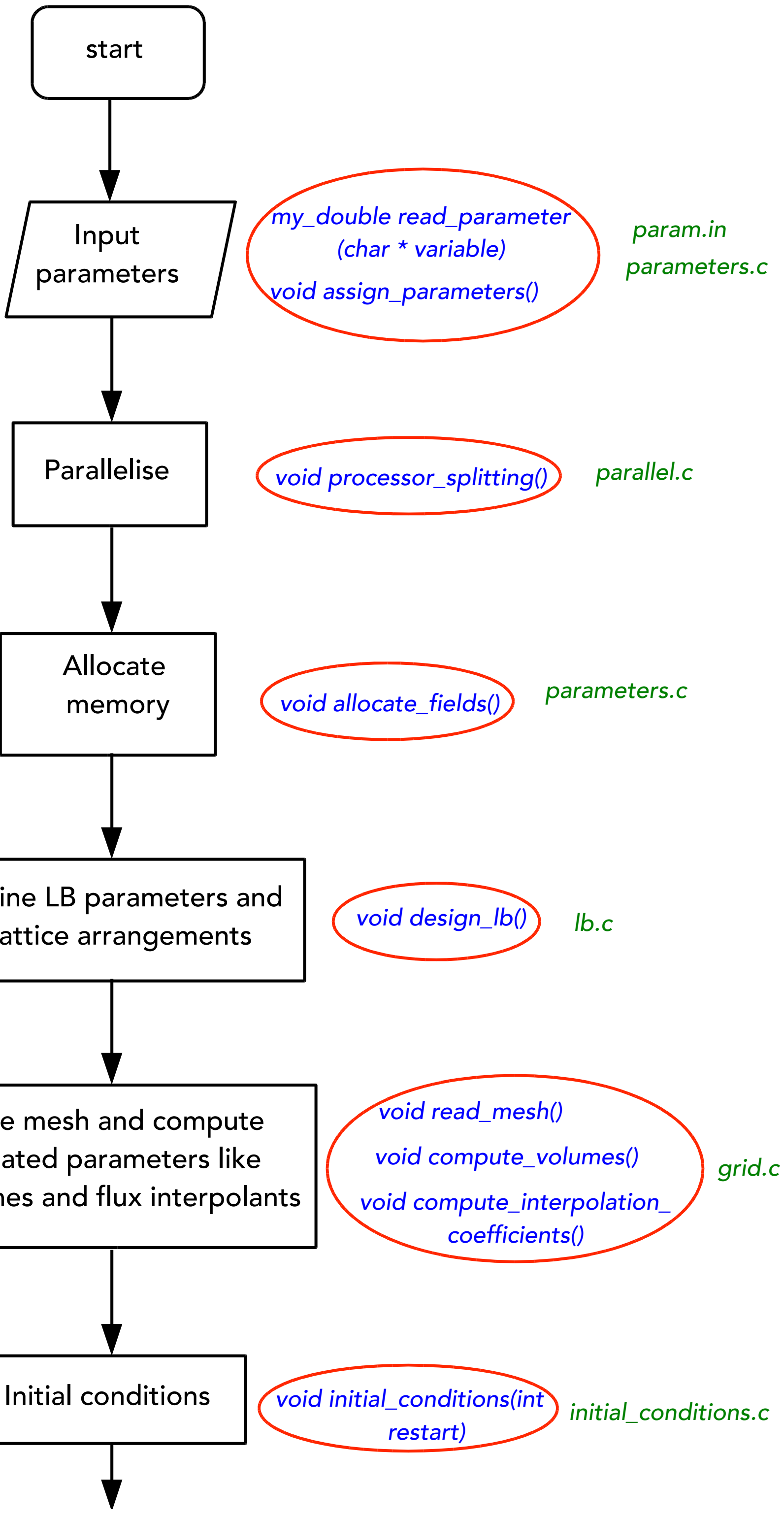
It was briefed in *Objectives* that this thesis consists of two parts. *Part I* addressed the topic of grid refinement using a FV discretization method to the LB equation. A FV LB code was developed to perform the tests and validations. Parallelly, the code is also developed in the framework of standard LB and both the LB algorithms coexists simultaneously in the code. This effort is in the direction of developing a ST LB code to be able to simulate complex flows *eg.* gas dispersion in the atmosphere, as ST LB is robust and easy to handle. This is addressed in *Part II* (this chapter). Apart from describing the structure of the code containing both LB algorithms, this part of the thesis addresses the related topics like *open boundary conditions, buoyancy, multi-component, large eddy simulation, complex geometry* required to simulate complex flows and also includes some validations.

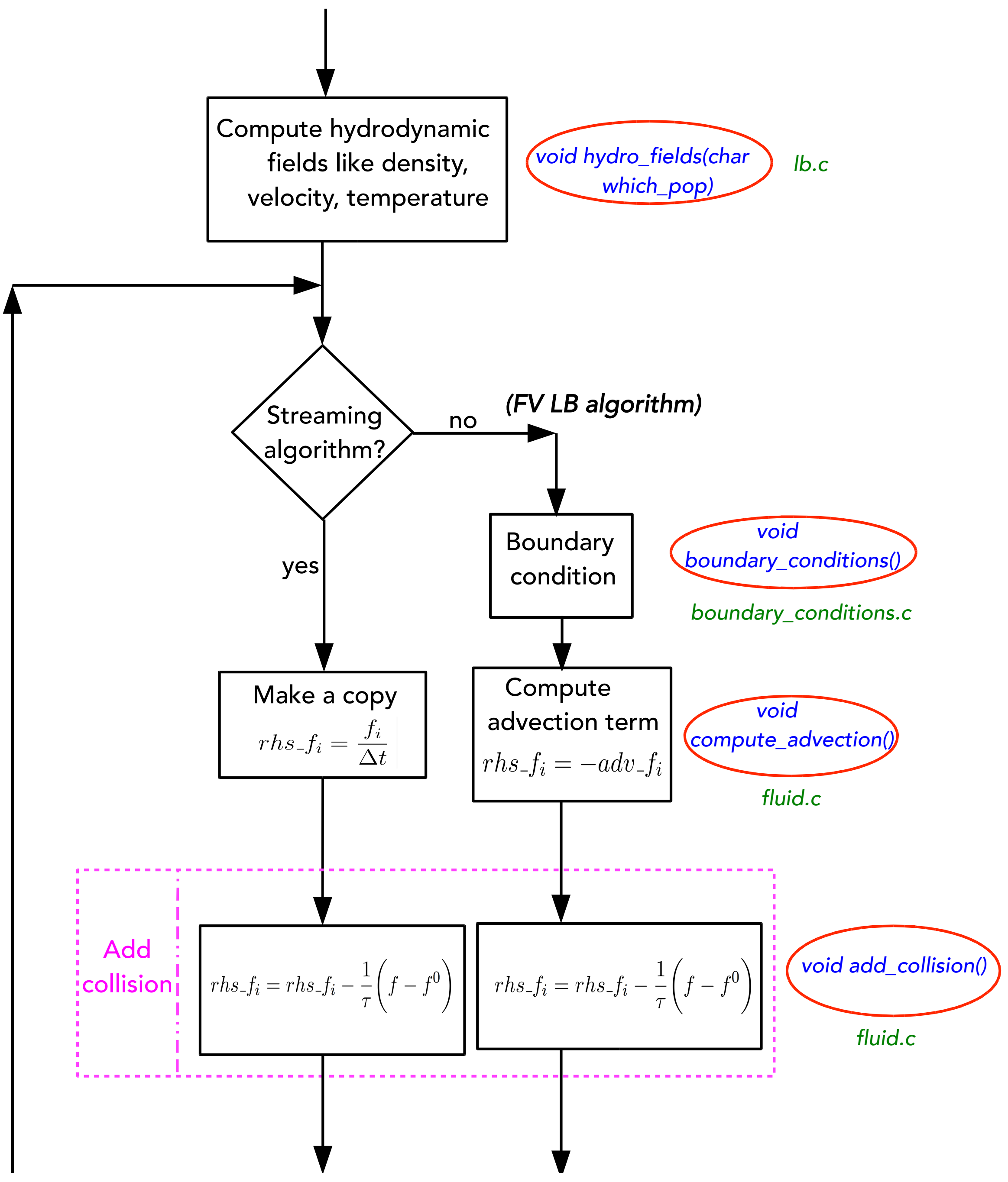
6.1 Introduction

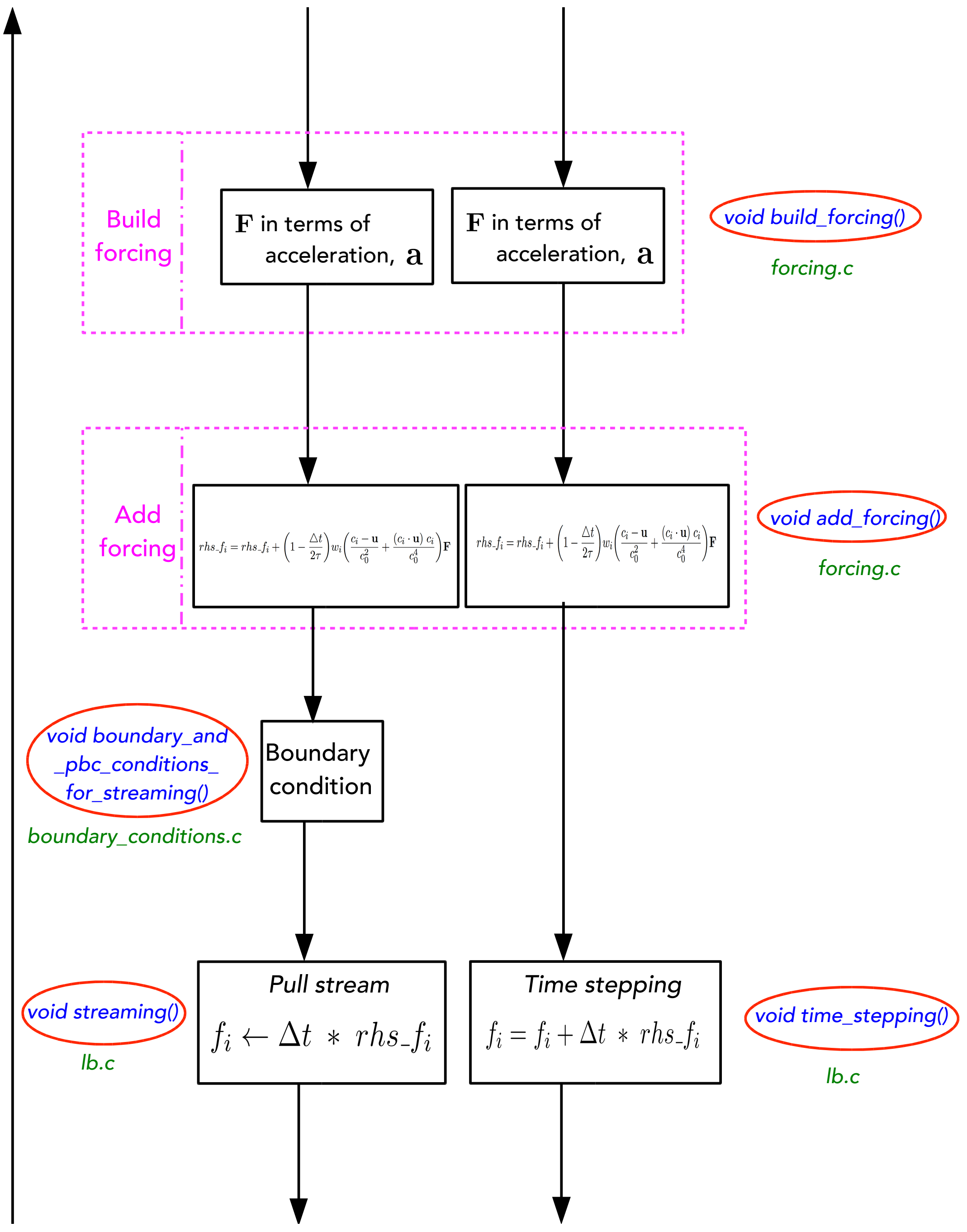
The developed code is a general purpose 3D Lattice-Boltzmann code for fluid-dynamics simulations. It is a c-language based code. The code is parallelised by means of the MPI library with efficient input/output using HDF5 library. Pre-post processing tools are written in Python.
The code is architected in such a way that various applications can be easily designed by modifying just two simple files *param.in* and $define.h$. Currently, the code is able to simulate various types of flows.

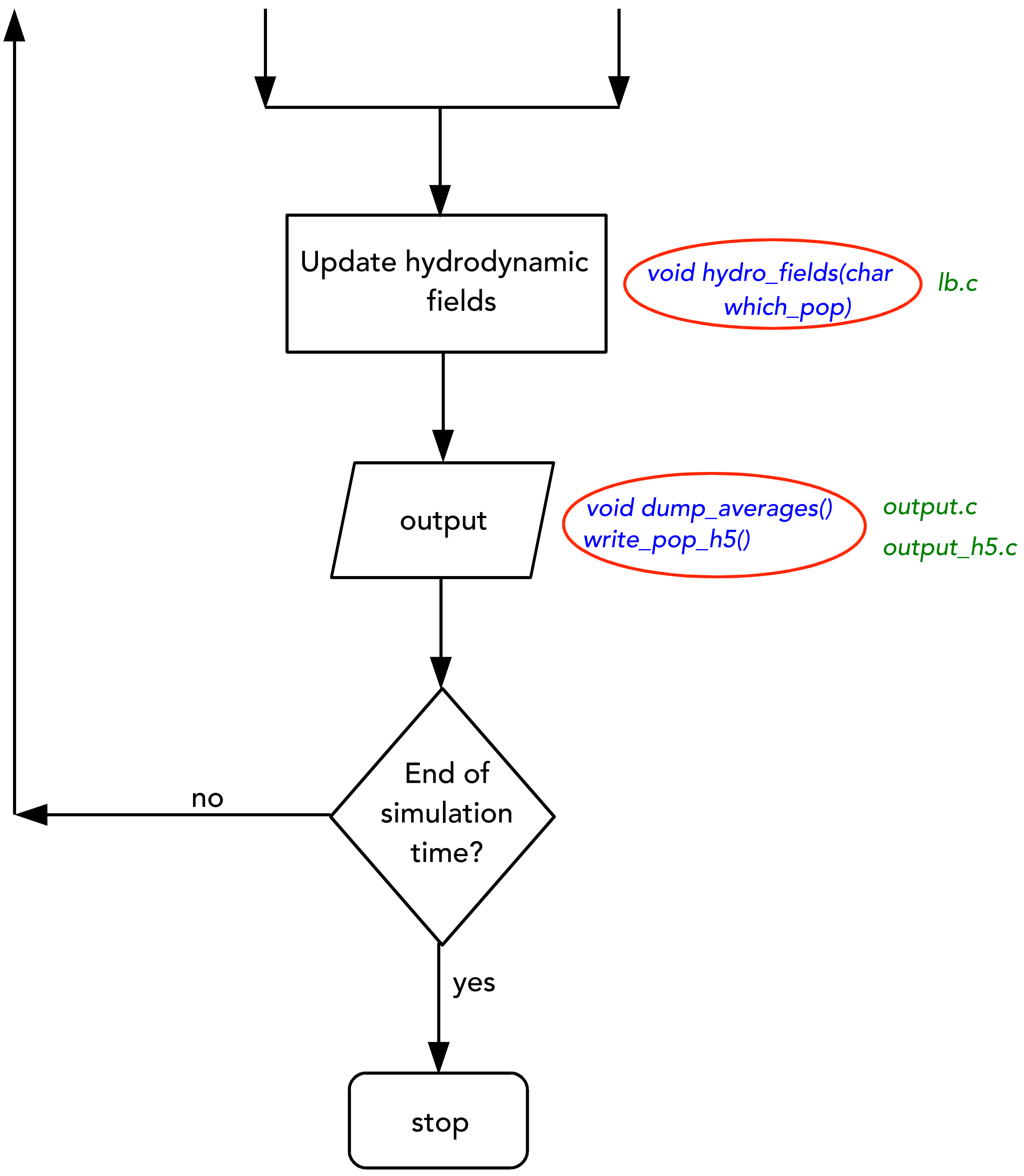
1. Fluid dynamics (with several volume forcing terms for Channel flow, Homogeneous Isotropic Turbulence, buoyancy)
2. Temperature dynamics (advection, diffusion , sink/source or reaction terms)
3. Phase change (enthalpy formulation for solid/liquid systems)
4. Scalar transport (same functionalities as temperature)
5. Lagrangian dynamics (tracers, heavy/light & active point-like particles; non-spherical Jeffrey rotation, gyrotaxis)
6. Large eddy simulation (Smagorinsky, Shear Improved Samgorinsky with Kalman Filter)

At the algorithm-level, the code consists of two LB algorithms co-existing simultaneously. One is the standard LB algorithm which is confined to regular square grids. Next one is a finite volume LB algorithm (one of the class of mesh refinement techniques in LBM) which is studied, developed and tested in this thesis. The two algorithms vary only in the definition of advection term therefore most part of the code is shared by both the algorithms and only vary in defining the advection term. The flowchart explaining the LB algorithms is presented below. At each step, the associated subroutines are shown in red-circled blue-coloured words. These subroutines are contained in *c* files whose names are attached along with the subroutines in green-coloured words.


start
Input parameters
my_double read_parameter (char * variable)
void assign_parameters()
param.in
parameters.c
Parallelise
void processor_splitting()
parallel.c
Allocate memory
void allocate_fields()
parameters.c
Define LB parameters and lattice arrangements
void design_lb()
lb.c
Define mesh and compute related parameters like volumes and flux interpolants
void read_mesh()
void compute_volumes()
void compute_interpolation_coefficients()
grid.c
Initial conditions
void initial_conditions(int restart)
initial_conditions.c


Compute hydrodynamic fields like density, velocity, temperature
void hydro_fields(char which_pop)
lb.c
Streaming algorithm?
no
(FV LB algorithm)
yes
Boundary condition
void boundary_conditions()
boundary_conditions.c
Make a copy
$rhs\_f_i = \frac{f_i}{\Delta t}$
Compute advection term
$rhs\_f_i = -adv\_f_i$
void compute_advection()
fluid.c
Add collision
$rhs\_f_i = rhs\_f_i - \frac{1}{\tau}\left(f - f^0\right)$
$rhs\_f_i = rhs\_f_i - \frac{1}{\tau}\left(f - f^0\right)$
void add_collision()
fluid.c


Build forcing
$\mathbf{F}$ in terms of acceleration, $\mathbf{a}$
$\mathbf{F}$ in terms of acceleration, $\mathbf{a}$
void build_forcing()
forcing.c
Add forcing
$rhs\_f_i = rhs\_f_i + \left(1 - \frac{\Delta t}{2\tau}\right) w_i \left(\frac{c_i - \mathbf{u}}{c_0^2} + \frac{(c_i \cdot \mathbf{u})\, c_i}{c_0^4}\right)\mathbf{F}$
$rhs\_f_i = rhs\_f_i + \left(1 - \frac{\Delta t}{2\tau}\right) w_i \left(\frac{c_i - \mathbf{u}}{c_0^2} + \frac{(c_i \cdot \mathbf{u})\, c_i}{c_0^4}\right)\mathbf{F}$
void add_forcing()
forcing.c
void boundary_and _pbc_conditions_ for_streaming()
boundary_conditions.c
Boundary condition
void streaming()
lb.c
Pull stream
$f_i \leftarrow \Delta t \, * \, rhs\_f_i$
Time stepping
$f_i = f_i + \Delta t \, * \, rhs\_f_i$
void time_stepping()
lb.c


Update hydrodynamic fields
void hydro_fields(char which_pop)
lb.c
output
void dump_averages()
write_pop_h5()
output.c
output_h5.c
End of simulation time?
no
yes
stop

Validations of the *streaming*-based algorithm

In *Part I*, while comparing the results with FV LBM, we have also seen the validation of standard LBM for laminar to turbulent flow cases. In the laminar flow category, the validations were performed for laminar Kolmogorov flow and Poiseuille flow. In the turbulent flow category, it was validated for a 3D Rayleign Bénard flow.
In this section, we further present more validations performed during the development of the code. It is important to note at this point that the motivation of *Part II* is to go in the direction of developing a LB code to be able to simulate methane gas dispersion in the atmosphere. Therefore the following developments in the code and their validations are just enough to fulfil the purpose.

6.2 Boundary conditions

Boundary conditions (B.C.) are important parameters to control the accuracy and stability of the method used. In LBM it becomes even more complex because only macroscopic quantities (*e.g.* zero velocity at the walls) are known and it has to be translated to distribution functions, f_i. The problem is complex because physical behaviour associated with mesoscopic information (distribution functions) are unknown.
Boundary conditions in LBM fall into two major groups: wet node and bounce-back approach. Wet node approach is a type where the boundary nodes are part of the fluid node. The constraints applied to the boundaries are at the end node of fluid domain. This approach allows to split the distribution function into equilibrium and non-equilibrium part and associate with the macroscopic variables of the flow. There are many implementations of wet node boundary conditions. The well-known ones are listed below :

- Inamuro boundary condition
- Zou/He boundary condition
- Regularized boundary condition
- Finite difference approximation for velocity gradients
- Guo Extrapolation method

On the other hand, bounce-back approach is a type where the boundary nodes are not the part of the fluid but are located half-way between the fluid node and boundary node.

The basic idea of bounce-back scheme is to copy the known distribution function (from within the fluid) to their unknown side but by reversing the direction. This way the required value of the macroscopic quantity is maintained at the boundary. There are two types of bounce-back schemes : full-way bounce back and half-way bounce back.
In full-way bounce back scheme, the population is replaced by inverting the velocity vectors in the collision step which is followed by the streaming / advection step. Contrary to the name, the wall is still located in between the bounce-back node and the fluid node. Half-way bounce back scheme differs to the full-way bounce back in the idea that the inversion of velocity vectors takes place during the streaming / advection step.
In the code we chose half-way bounce-back scheme and is implemented in the following way.

6.2.1 No-slip boundary condition

Physically, no-slip B.C. means zero fluid motion (velocity = 0) at the boundary walls. For simplicity, we assume here that the boundary wall is parallel to the grid. The simplest approach to have no fluid motion at the walls is by reversing the populations at the boundary node such that each population cancel each other. The implementation could be two ways. Either the physical boundary lies on the grid line (on-grid) or the physical boundary lies between two grid lines (mid-grid). Given that on-grid is just of the first order in accuracy, we chose mid-grid implementation (refer [116]).
Figure 6.1 shows the population arrangement at the buffer cell to implement no-slip boundary condition. The definition of the populations at the buffer cell which are related to the node in the fluid cell is limited to the populations that are pointing inside the domain.

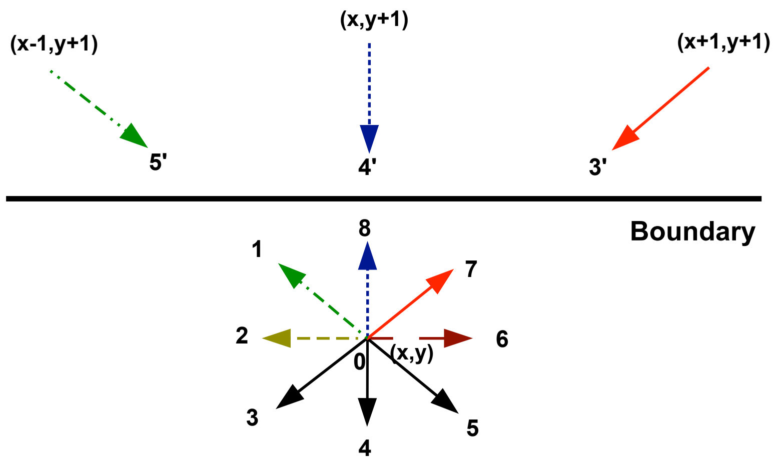


FIGURE 6.1: Schematic representation of the populations defining no-slip boundary condition at the top wall for a 2D case

In figure 6.1:

$$f^{5'}_{x-1,y+1} = f^{1}_{x,y}$$

$$f^{4'}_{x,y+1} = f^{8}_{x,y}$$

$$f^{3'}_{x+1,y+1} = f^{7}_{x,y}$$

6.2.2 Free-slip boundary condition

Free-slip boundary condition represents boundary condition where the fluid flow is prevented along the normal and allowed along the tangential direction. It occurs at the interface between two fields (*e.g.* such as air and water). This disallows any momentum exchange with the wall. Again, here we adopt mid-grid implementation because of its second order accuracy.
The implementation can be better understood from figure 6.2 which depicts the population arrangement at the buffer cell to introduce free-slip boundary condition.

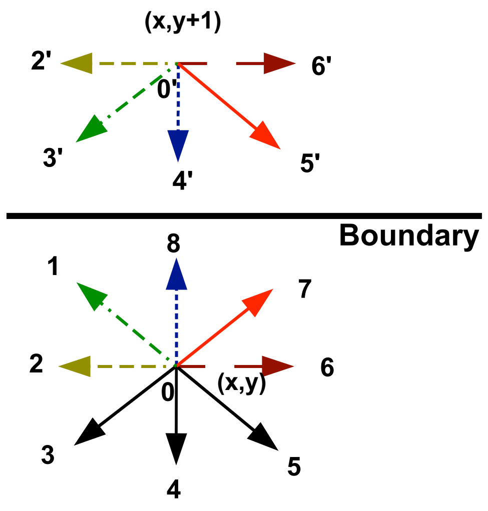


Figure 6.2: Schematic representation of the populations defining free-slip boundary condition at the top wall for a 2D case

In figure 6.2:

$$f^{2'}_{x,y+1} = f^{2}_{x,y}$$

$$f^{3'}_{x,y+1} = f^{1}_{x,y}$$

$$f^{4'}_{x,y+1} = f^{8}_{x,y}$$

$$f^{5'}_{x,y+1} = f^{7}_{x,y}$$

$$f^{6'}_{x,y+1} = f^{6}_{x,y}$$

6.2.3 Complex boundaries

One of the advantages of LB simulations over Navier-Stokes simulations is that LBM can handle complex geometries with much ease. We follow a bounce-back method to realize the treatment of fluid flows with complex geometry. This section demonstrates the easiness of the method implementation.

The complex geometry is introduced inside the homogeneous lattices. The complex geometry will occupy some lattice points inside its boundary distinguishing the solid nodes from the fluid nodes. Bounce-back scheme would mean just to apply no-slip boundary condition at the physical boundaries i.e. assigning zero velocities at the physical boundaries. This can be easily implemented by following the given steps:

- carry-on the usual collision process
- only stream at the fluid nodes
- at the nearest fluid node from the complex geometry boundary - reverse only the populations that go inside the solid node

Figure 6.3 show the schematic of how the streaming process happens in the lattices after bounce-back scheme is applied. However for complex geometry (smooth surfaces, curved geometry), this method suffers from low resolution. The boundary of the general surface geometry may not lie exactly in between two lattice points but the mid-grid bounce-back scheme assumes it to be exactly in between two lattice points. Therefore the boundary is represented by stair-wise segments. For flows where such effect doesn't affect the required details, then it is a very simple method to deal with complex boundaries. Further improvements can add to better accuracy (refer [51, 117–120]). Also, there are other techniques based on the "immersed boundary scheme" (refer [121–123]) which can implement complex boundary with better accuracy but added complexity.

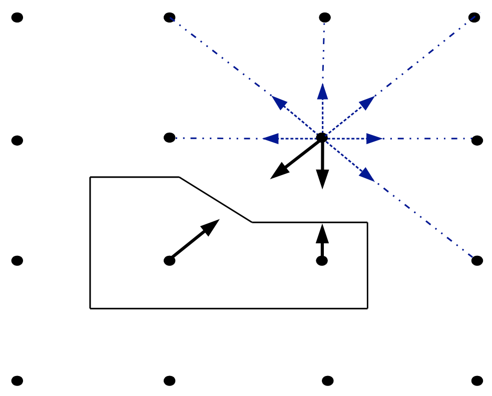


FIGURE 6.3: Schematic representation of the populations defining bounce-back scheme at the complex boundary for a 2D case

6.2.4 Open boundaries

Inlet/outlet boundary conditions are also known as the pressure (density) and velocity boundary conditions because of the nature of implementation. The most popular and relatively easy implemented inlet/outlet boundary condition credits to Zou and He [104]. Zou/He method is based on the idea of bounce-back of non-equilibrium distribution. There are two ways of implementing the method as pointed below.

- Given u_x, u_y, find ρ and unknown f_i. *(velocity boundary condition)*
- Given ρ and the velocity along the boundary, find the velocity normal to the boundary and unknown f_i. *(pressure boundary condition)*

However the drawback as mentioned by Zou and He [104] is its difficulty in implementation to general geometry because there is a need to distinguish distribution functions according to the orientation of wall and also there are different treatments at the corner nodes. This adds complexity in implementing in a computer code. Also for each type of lattice arrangement (D_2Q_9 or D_3Q_{19} *etc.*), the missing populations need to be defined uniquely because they need to be distinguished.
To further ease the implementation, we propose a new type of open B.C. based on the idea of assigning equilibrium distribution corresponding to desired density and velocity. This idea was first proposed by Grunau [20]. Although it enjoyed simplicity, it introduced significant errors. Therefore researchers [104, 124–126] proposed other accurate ways but relatively difficult implementation. Therefore we aim to extend the idea for an easy implementation with acceptable accuracy. In addition to proposing easy implemented open B.C., we add the non-reflecting pressure boundary condition proposed by Finck *et al.* [127] to further help simulate complex flows. Complex flow simulations without the non-reflecting pressure boundary condition obeys mass conservation but triggers reflecting disturbances at the outlet and will explode the simulation. So, based on the idea of assigning equilibrium distribution to the distribution function at the buffer nodes corresponding to specified density and velocity, our new boundary conditions are easy to implement (applicable to any lattice arrangements). Although it looses accuracy, it enjoys good stability and is applicable to un-straight boundaries as well.

6.2.4.1 Inlet boundary condition

Assume a 2D system (D_2Q_9) with direction nomenclature as shown in figure 6.4.

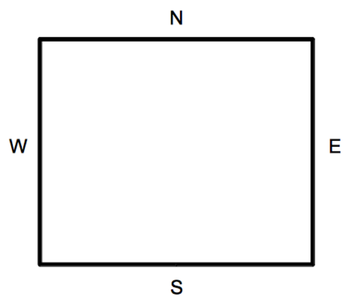


FIGURE 6.4: 2D: direction nomenclature

Let us take west boundary as inlet, as shown in figure 6.5 for clarity. Suppose we know the inlet velocity profile u_{in} (*e.g.* Poiseuille velocity profile). This can be used to calculate ρ as follows:

$$\rho_{in} = \frac{1}{1-u_{in}}[f_0 + f_4 + f_8 + 2(f_1 + f_2 + f_3)]$$

The knowledge of ρ and velocity(u_{in}) is enough. The equilibrium distribution corresponding to the estimated density and velocity is assigned at the buffer node $(x-1)$ to specify the inlet boundary condition. It should be noted that the buffer zone $(x-1)$ runs from buffer node at S to buffer node at N in the y-direction. Therefore, this type of implementation doesn't require any treatment at corner nodes which is an advantage in terms of implementation.

$$f(in)_{(x-1,y)} = f^e[\rho_{in}, u_{in}]$$

This implementation looks similar to velocity boundary condition of Zou/He method but avoids all the care to be taken to each population. Just assigning equilibrium distribution at the buffer node $(x-1,y)$ corresponding to the velocity and density is enough to specify an inlet. Also, the orientation of the wall is no issue. Any orientation of wall can be easily assigned as inlet. For example for south boundary, if we calculate ρ from the inlet velocity, the equilibrium distribution can be specified as:

$$f(in)_{(x,y-1)} = f^e[\rho_{in}, u_{in}]$$

Using this method the inlet boundary condition becomes very simple. The procedure only involves defining an appropriate density value corresponding to the desired velocity

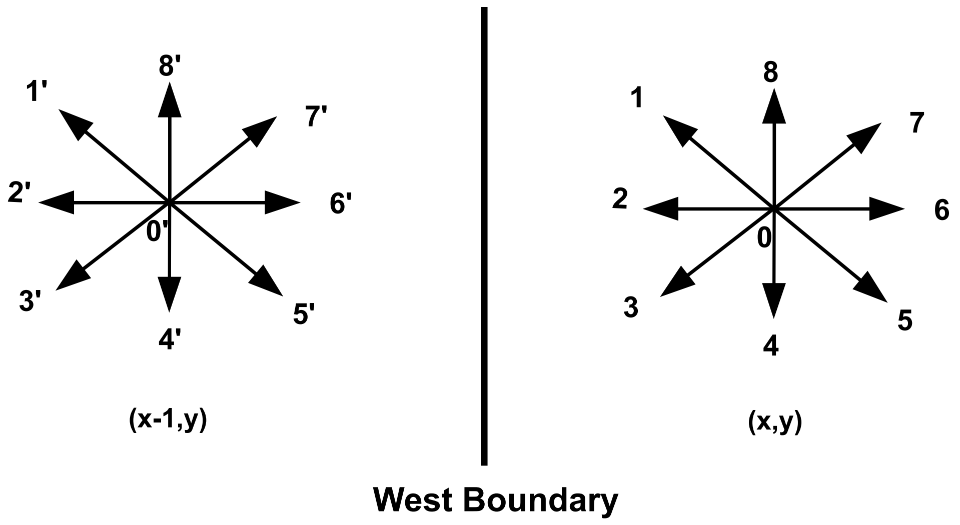


FIGURE 6.5: Schematic representation of the populations defining inlet boundary condition for a 2D case : west boundary

inlet. Then, the method automatically defines the populations corresponding to the density and velocity. This is a huge advantage in terms of implementation effort compared to the Zou/He technique where we need to take care of all the populations entering the domain. The advantage is hugely noticed for a 3D lattice configuration.

6.2.4.2 Outlet boundary condition

Let us take east boundary as outlet as shown in figure 6.6. The buffer node at the outlet needs to be defined with equilibrium distribution corresponding to appropriate density and velocity. Again, the buffer zone should run from buffer node at S to buffer node at N in y-direction.

$$f(out)_{(x+1,y)} = f^e[\rho_{out}, u_{out}]$$

Density estimation:
LBM is weakly compressible but still the density variation in the domain can be considered very negligible. Therefore the most appropriate density guess could be to borrow it from the nearest fluid node from the boundary instead of assuming 1.

$$\rho_{x+1,y} = \rho_{x,y}$$

Velocity estimation:
Using Dirichlet boundary condition, the velocity can be calculated using the approximated density and the populations at the fluid node nearest to the boundary. The populations (from the nearest fluid node) involved in estimating velocity normal to the

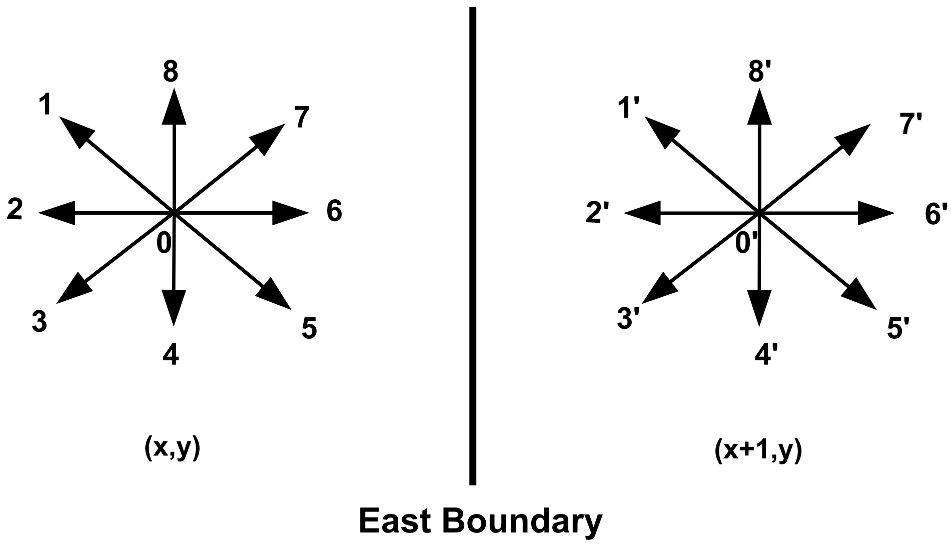


FIGURE 6.6: Schematic representation of the populations defining outlet boundary condition for a 2D case : east boundary

wall are only the ones that stream out (*e.g.* to calculate u_e, we require populations $0, 4, 8, 7, 6, 5$). The populations involved in estimating velocity along the wall uses populations $0, 2, 6, 1, 8, 7$. Since this is a tangential velocity, it is fine to use the populations from the nearest fluid node as this doesn't affect the flow inside the domain. Therefore, it can be used with accuracy. To be noted is that Zou/He method involves estimation of missing populations *i.e.* $1, 2, 3$ referring to figure 6.6. Also, it already assumes that velocity normal to wall is known apart from the density.

Now, we calculate all the velocity components at the outlet. This helps to specify outlet boundary condition even to un-straight walls which is also an advantage over Zou/He method. The procedure of calculating 2 velocity set (u, v) for a D_2Q_9 configuration to any orientation of the outlet wall is given below.

$$\rho_{out} = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8$$

$$\rho_{out} u_S = \rho_{out} u_W = \rho_{out} u_N = \rho_{out} u_E = f_7 + f_6 + f_5 - f_1 - f_2 - f_3$$

$$\rho_{out} v_S = \rho_{out} v_W = \rho_{out} v_N = \rho_{out} v_E = f_1 + f_8 + f_7 - f_3 - f_4 - f_5$$

$$\rho_{out} - \rho_{out} u_S = \rho_{out} - \rho_{out} u_W = f_0 + f_4 + f_8 + 2(f_1 + f_2 + f_3)$$

$$u_S = u_W = 1 - \frac{f_0 + f_4 + f_8 + 2(f_1 + f_2 + f_3)}{\rho_{out}}$$

$$\rho_{out} - \rho_{out} v_S = \rho_{out} - \rho_{out} v_W = f_0 + f_2 + f_6 + 2(f_3 + f_4 + f_5)$$

$$v_S = v_W = 1 - \frac{f_0 + f_2 + f_6 + 2(f_3 + f_4 + f_5)}{\rho_{out}}$$

$$\rho_{out} - \rho_{out} u_N = \rho_{out} - \rho_{out} u_E = f_0 + f_4 + f_8 + 2(f_7 + f_6 + f_5)$$
$$u_N = u_E = 1 - \frac{f_0 + f_4 + f_8 + 2(f_7 + f_6 + f_5)}{\rho_{out}}$$

$$\rho_{out} - \rho_{out} v_N = \rho_{out} - \rho_{out} v_E = f_0 + f_2 + f_6 + 2(f_1 + f_8 + f_7)$$
$$v_N = v_E = 1 - \frac{f_0 + f_2 + f_6 + 2(f_1 + f_8 + f_7)}{\rho_{out}}$$

Non-reflective boundary condition:

This definition of equilibrium boundary condition should let the flow out without triggering the reflecting waves. The above implementation is still not enough to avoid wave reflections. This need to be treated using a non-reflecting pressure boundary condition as given by Finck *et al.* [127]. This helps to simulate complex flows avoiding reflecting waves.

$$p_b^t = \frac{p_b^{t-\Delta t} + \rho c_0 (u_n^t - u_n^{t-\Delta t}) + \alpha \triangle t p_{out}}{1 + \alpha \triangle t}$$

where, c_0 is the speed of sound, u_n is the flow velocity normal to the boundary and α is the constant parameter tuned by numerical experiments (usually taken as 1.0). Superscripts t and $t - \Delta t$ represents value at current time-step and previous time-step respectively. This can be converted to the expression of density utilizing the relation $p = \rho\ c_0^2$.

$$\rho_b^t = \frac{\rho_b^{t-\Delta t} + \rho (1/c_0)(u_n^t - u_n^{t-\Delta t}) + \alpha \triangle t \rho_{out}}{1 + \alpha \triangle t}$$

Finally, $f(out)_{(x+1,y)} = f^e[\rho_{out}, u_{out}]$ can be applied which define the populations based on the estimated velocity and density.

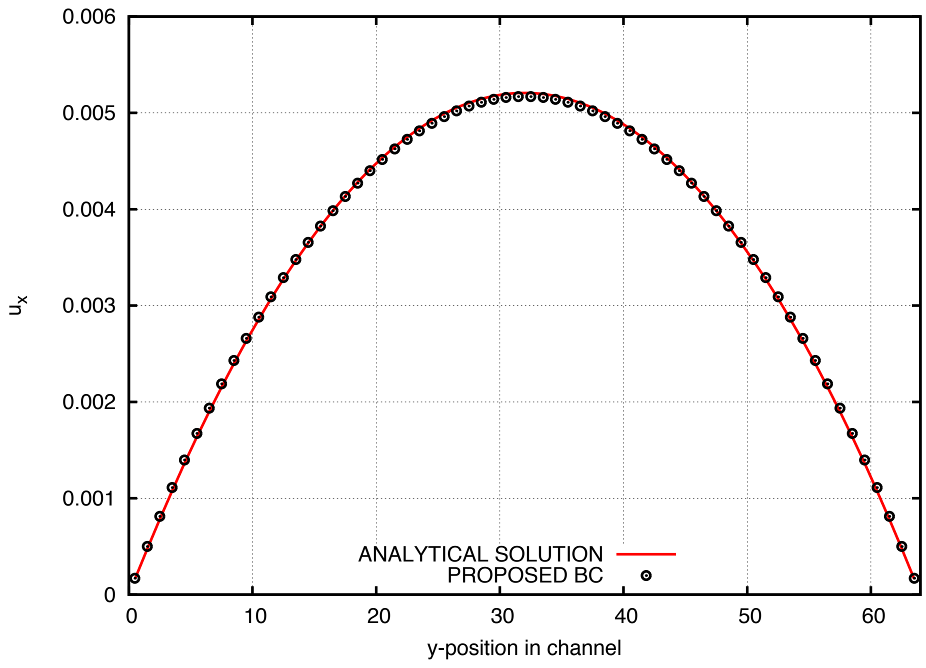


FIGURE 6.7: Test of accuracy of the proposed BC with a 2D channel flow in a $[128, 64, 1]$ spatial domain with maximum velocity = 0.005208. The proposed method shows improved accuracy, bettering the idea of Grunau [20] (see also the discussions by Latt *et al.* [21] regarding the significant error introduced by equilibrium on boundary method as proposed by Grunau). To note is that the aim of the proposed BC is only to ease the implementation effort while attaining acceptable accuracy. Also, this method efficiently couples with non-reflective BC which as a whole helps simulate complex flows. Therefore, although the proposed BC exhibits half-order accuracy, it has acceptable accuracy with great simplicity in implementation for simulating complex flows.

These boundary conditions can be easily extended to 3D case (*eg.* D_3Q_{19}) by just redefining the 3 velocity set (u, v, w). The direction nomenclature is as shown in the figure 6.8.

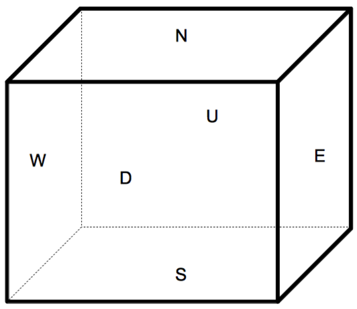


FIGURE 6.8: 3D: direction nomenclature

The density and the non-reflecting boundary condition can be estimated exactly in a similar way to define the outlet. The velocity set for D_3Q_{19} is as follows:

$$\rho_{out} = f_0+f_1+f_2+f_3+f_4+f_5+f_6+f_7+f_8+f_9+f_{10}+f_{11}+f_{12}+f_{13}+f_{14}+f_{15}+f_{16}+f_{17}+f_{18}$$

$$\rho_{out}u_S = \rho_{out}u_W = \rho_{out}u_N = \rho_{out}u_E = \rho_{out}u_D = \rho_{out}u_U =$$

$$= f_1 + f_7 + f_{11} + f_8 + f_{13} - f_2 - f_9 - f_{12} - f_{10} - f_{14}$$

$$\rho_{out}v_S = \rho_{out}v_W = \rho_{out}v_N = \rho_{out}v_E = \rho_{out}v_D = \rho_{out}v_U =$$

$$= f_3 + f_{15} + f_9 + f_{16} + f_7 - f_4 - f_{17} - f_{10} - f_{18} - f_8$$

$$\rho_{out}w_S = \rho_{out}w_W = \rho_{out}w_N = \rho_{out}w_E = \rho_{out}w_D = \rho_{out}w_U =$$

$$= f_5 + f_{15} + f_{12} + f_{17} + f_{11} - f_6 - f_{16} - f_{14} - f_{18} - f_{13}$$

$$\rho_{out} - \rho_{out}u_S = \rho_{out} - \rho_{out}u_W = \rho_{out} - \rho_{out}u_D =$$

$$= f_0 + f_3 + f_4 + f_5 + f_6 + f_{15} + f_{16} + f_{17} + f_{18} + 2(f_2 + f_9 + f_{12} + f_{10} + f_{14})$$

$$u_S = u_W = u_D =$$

$$= 1 - \frac{f_0 + f_3 + f_4 + f_5 + f_6 + f_{15} + f_{16} + f_{17} + f_{18} + 2(f_2 + f_9 + f_{12} + f_{10} + f_{14})}{\rho_{out}}$$

$$\rho_{out} - \rho_{out}v_S = \rho_{out} - \rho_{out}v_W = \rho_{out} - \rho_{out}v_D =$$

$$= f_0 + f_1 + f_2 + f_5 + f_6 + f_{11} + f_{12} + f_{13} + f_{14} + 2(f_4 + f_{17} + f_{10} + f_{18} + f_8)$$

$$v_S = v_W = v_D =$$

$$= 1 - \frac{f_0 + f_1 + f_2 + f_5 + f_6 + f_{11} + f_{12} + f_{13} + f_{14} + 2(f_4 + f_{17} + f_{10} + f_{18} + f_8)}{\rho_{out}}$$

$$\rho_{out} - \rho_{out}w_S = \rho_{out} - \rho_{out}w_W = \rho_{out} - \rho_{out}w_D =$$

$$= f_0 + f_1 + f_2 + f_3 + f_4 + f_7 + f_8 + f_9 + f_{10} + 2(f_6 + f_{16} + f_{14} + f_{18} + f_{13})$$

$$w_S = w_W = w_D =$$

$$= 1 - \frac{f_0 + f_1 + f_2 + f_3 + f_4 + f_7 + f_8 + f_9 + f_{10} + 2(f_6 + f_{16} + f_{14} + f_{18} + f_{13})}{\rho_{out}}$$

$$\rho_{out} - \rho_{out} u_N = \rho_{out} - \rho_{out} u_E = \rho_{out} - \rho_{out} u_U =$$

$$= f_0 + f_3 + f_4 + f_5 + f_6 + f_{15} + f_{16} + f_{17} + f_{18} + 2(f_1 + f_7 + f_{11} + f_8 + f_{13})$$

$$u_N = u_E = u_U =$$

$$= -1 + \frac{f_0 + f_3 + f_4 + f_5 + f_6 + f_{15} + f_{16} + f_{17} + f_{18} + 2(f_1 + f_7 + f_{11} + f_8 + f_{13})}{\rho_{out}}$$

$$\rho_{out} - \rho_{out} v_N = \rho_{out} - \rho_{out} v_E = \rho_{out} - \rho_{out} v_U =$$

$$= f_0 + f_1 + f_2 + f_5 + f_6 + f_{11} + f_{12} + f_{13} + f_{14} + 2(f_3 + f_{15} + f_9 + f_{16} + f_7)$$

$$v_N = v_E = v_U =$$

$$= -1 + \frac{f_0 + f_1 + f_2 + f_5 + f_6 + f_{11} + f_{12} + f_{13} + f_{14} + 2(f_3 + f_{15} + f_9 + f_{16} + f_7)}{\rho_{out}}$$

$$\rho_{out} - \rho_{out} w_N = \rho_{out} - \rho_{out} w_E = \rho_{out} - \rho_{out} w_U =$$

$$= f_0 + f_1 + f_2 + f_3 + f_4 + f_7 + f_8 + f_9 + f_{10} + 2(f_5 + f_{15} + f_{12} + f_{17} + f_{11})$$

$$w_N = w_E = w_U =$$

$$= -1 + \frac{f_0 + f_1 + f_2 + f_3 + f_4 + f_7 + f_8 + f_9 + f_{10} + 2(f_5 + f_{15} + f_{12} + f_{17} + f_{11})}{\rho_{out}}$$

This is advantageous because it highly reduces the implementation effort. Another big advantage is that we just need to redefine the 3 velocity set (u, v, w) without the need of corner nodes treatment. The same applies if we want to implement other lattice configurations.

6.2.5 Validation of boundary conditions

Accuracy of this novel open boundary condition is tested with a 2D laminar jet case which has an analytic solution. Schematic diagram for the jet injection set-up is shown in figure 6.9. Jet injection case is about a fluid with some momentum and/or buoyancy injected into another fluid of different speed, temperature or contamination level. The present study is about a fluid that intrudes into another medium at rest. This is named as submerged jet and the velocity contours are as shown in figure 6.10. The Reynolds number chosen for the simulation is 30.

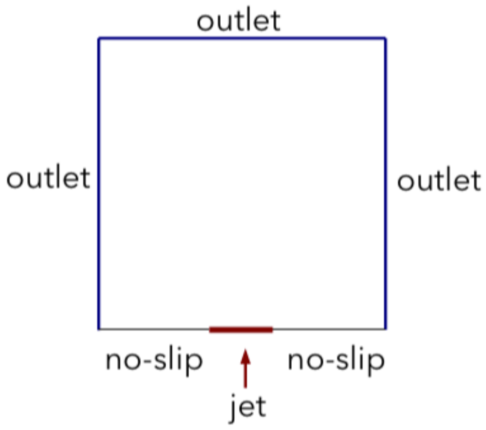


FIGURE 6.9: Schematic representation of the jet injection case setup

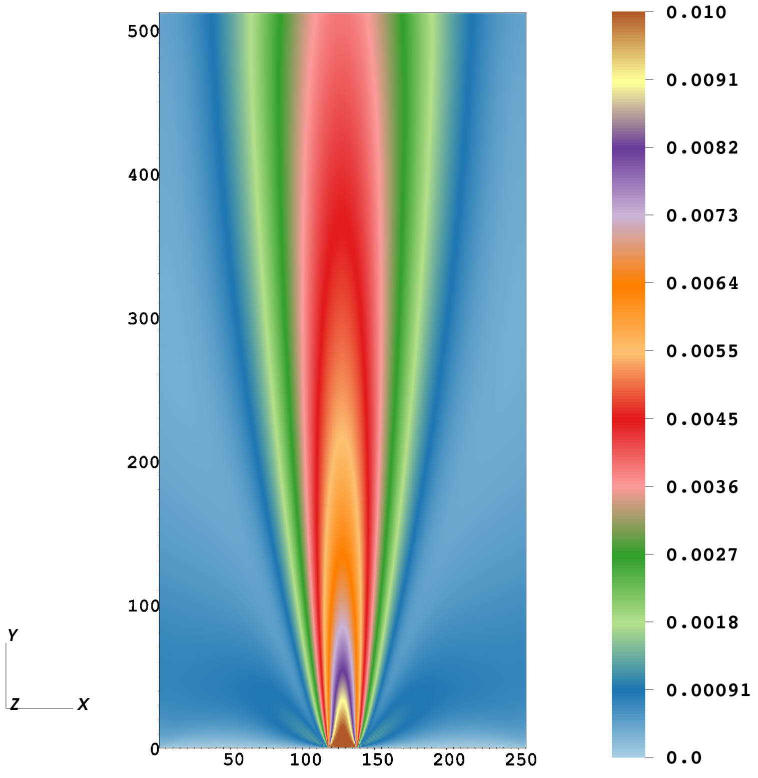


FIGURE 6.10: Velocity magnitude contours of a 2D laminar jet: Domain size = 256 X 512; Jet inlet velocity = 0.01; Reynolds number = 30.

From the analytical solution of 2D laminar jet, the local maximum velocity should decay with downstream distance as $y^{-1/3)}$ which is verified by figure 6.11. The deviation of the simulation results from the analytic solution (for $y \lesssim 50$) can be justified by the existence of potential core which is also experimentally verified. Potential core is the downstream distance of the jet where the centerline velocity is still the maximum velocity.

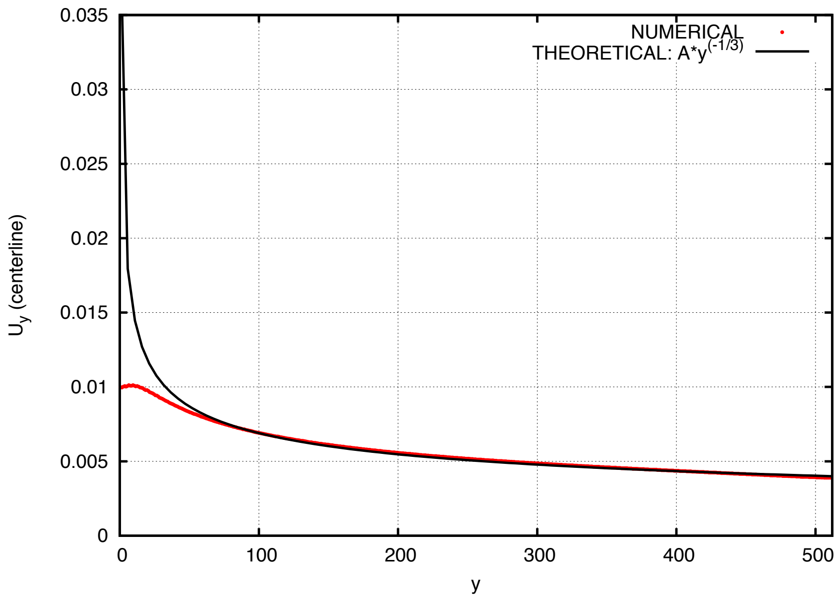


FIGURE 6.11: Decay of the centerline velocity of free jet

Also, the layer thickness should grow with downstream distance as $y^{2/3}$ which is verified by figure 6.12. This also explains the spreading rate of the jet. Jet similarity characteristics is explained by figure 6.13. Velocity profiles at different cross-sections downstream are non-dimensionalised by local maximum velocity and jet half-width thickness $(delta_{1/2})$. Jet half-width thickness is defined as the distance between the axis and the location where the local velocity equals half the local maximum velocity.

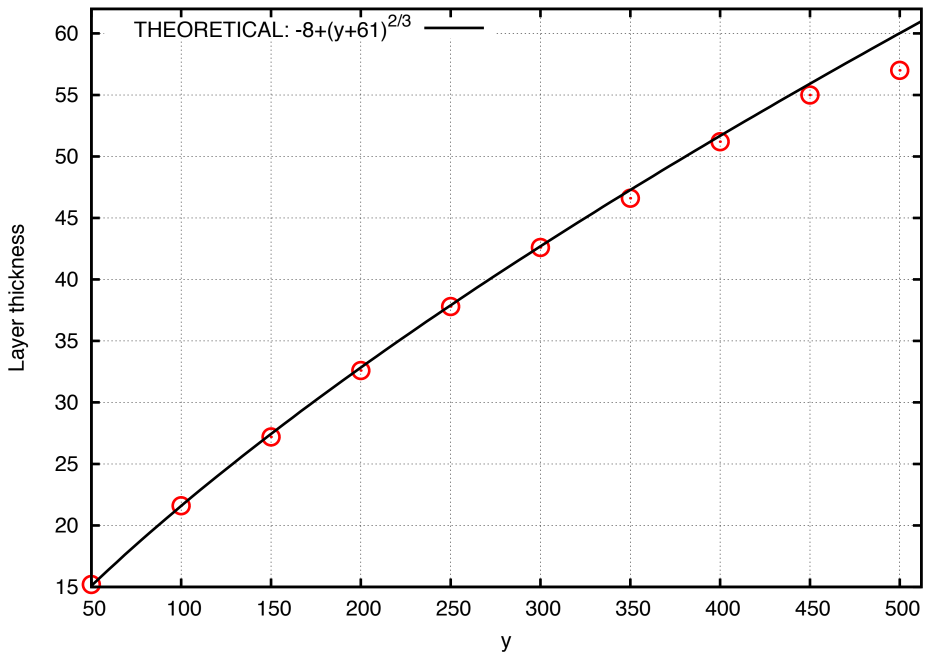


FIGURE 6.12: Growth of the jet half-width. Line $-8 + (y + 61)^{\frac{2}{3}}$ is an empirical fit to the simulation data

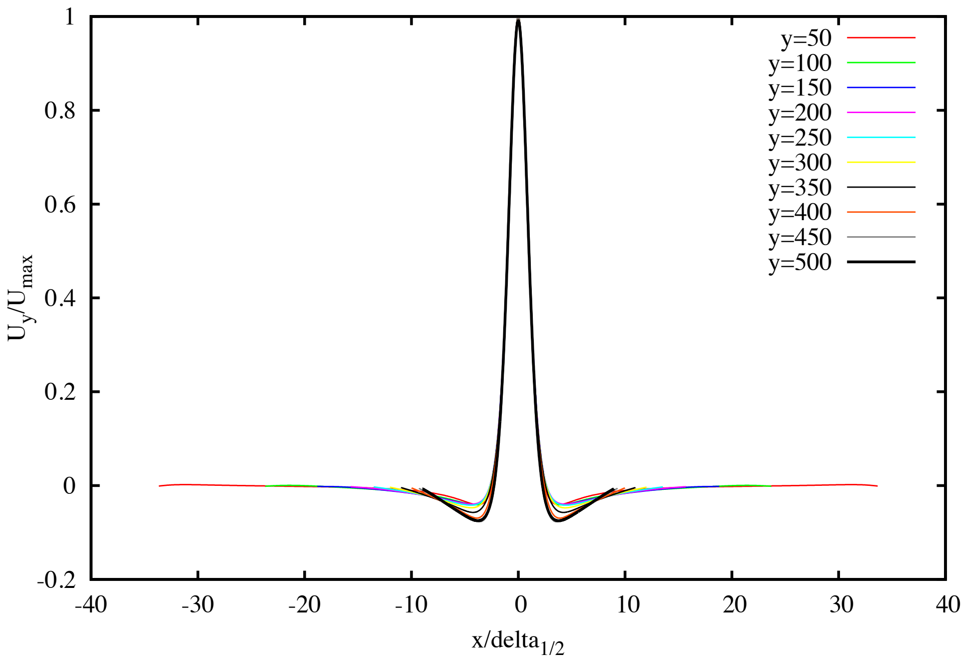


FIGURE 6.13: Cross-stream variation of the normalised v-component velocity

6.3 Validation of turbulent flows

The benchmark problem for validation of turbulent flows is a fully turbulent channel flow for which a wide dataset of experimental and DNS measurements are available. The schematic diagram for the turbulent channel flow simulation is shown in figure 6.14.

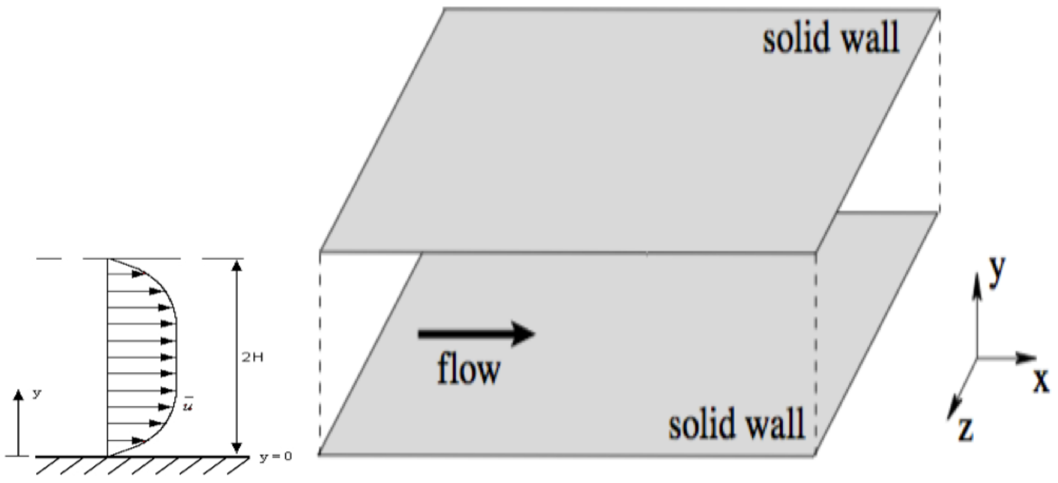


FIGURE 6.14: Schematic drawing of a fully-developed turbulent channel flow simulation.

The simulation parameters are as follows:

- Reynolds number, $Re_\tau = 180$. Re_τ is the Reynolds number based on the friction velocity. It is defined as $Re_\tau = \frac{u_\tau \delta}{\nu}$ where u_τ is the friction velocity, δ is the half-channel height and ν is the viscosity.
- grid size $(N_X \text{ X } N_Y \text{ X } N_Z) = 256 \text{ X } 128 \text{ X } 128$
- periodic boundary conditions at x-z direction and no-slip wall boundary condition at y-direction
- maximum centerline velocity = 0.004466 lattice units
- relaxation time, $\tau = 0.004764$ lattice units

Figure 6.15 shows the velocity magnitude contours of the fully turbulent channel flow. The mean velocity profile is plotted in figure 6.16 comparing LBM-DNS with the data by Moser *et al.* [128]. It is emphasized that the data by Moser *et al.* has a grid refinement at the wall, while LBM-DNS has regular grid with unresolved boundary layer. Figures 6.17, 6.18 and 6.19 are RMS fluctuations of u, v, w - velocities respectively. It can be seen from the figures that LBM-DNS compares very well with the DNS data by Moser *et al.*

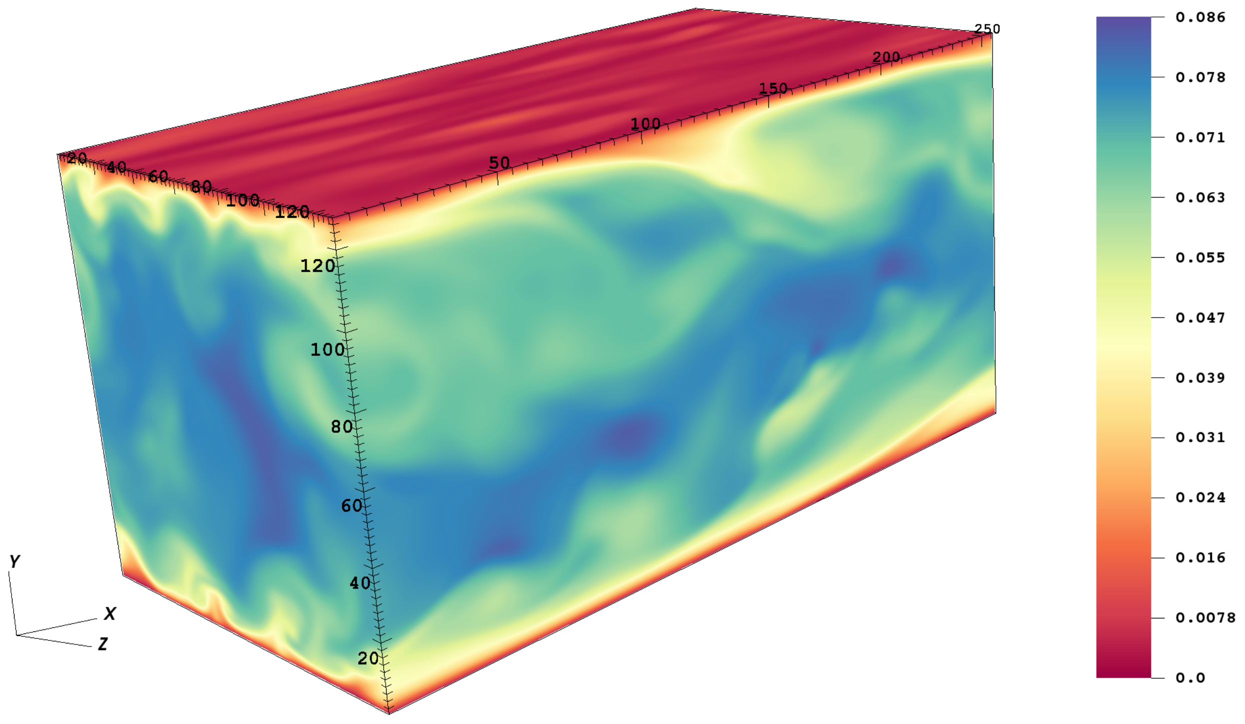


FIGURE 6.15: Contours of velocity magnitude for a fully-developed turbulent channel flow simulation.

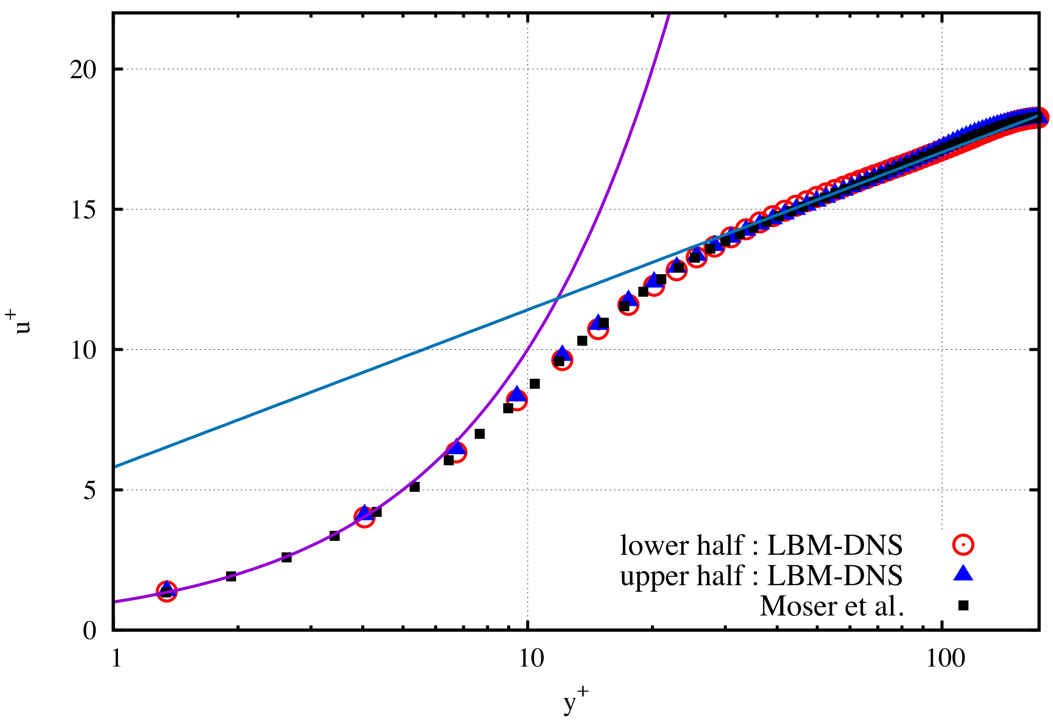


FIGURE 6.16: Mean streamwise velocity profile u^+ *vs.* y^+.

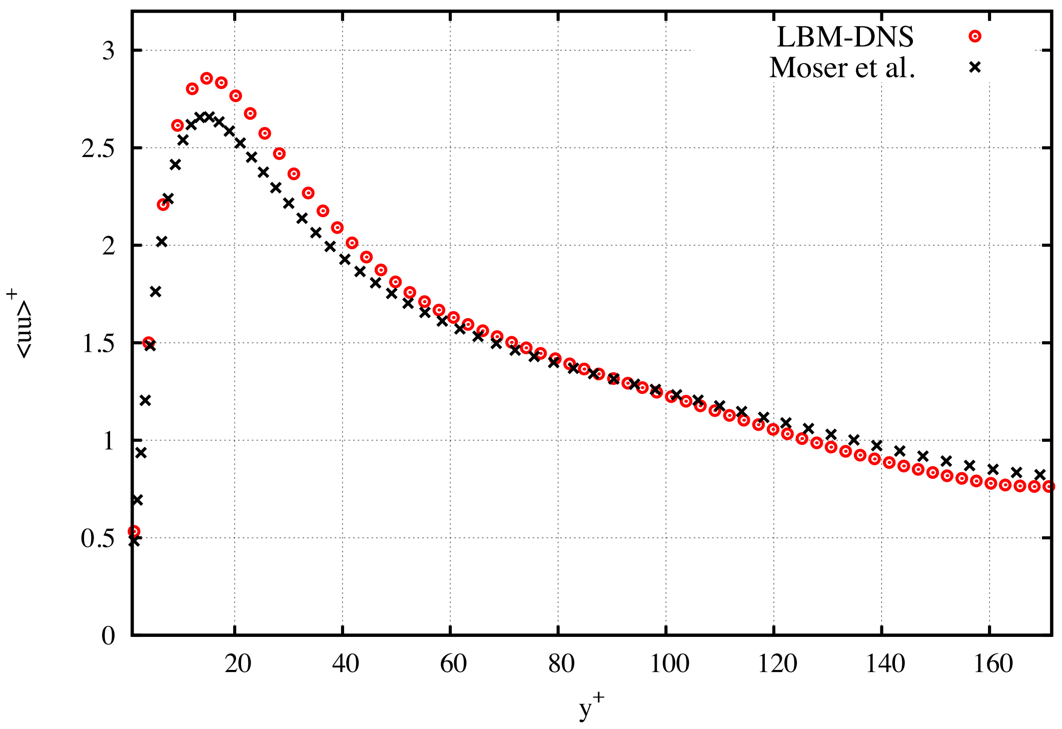


FIGURE 6.17: Root mean square of u-velocity.

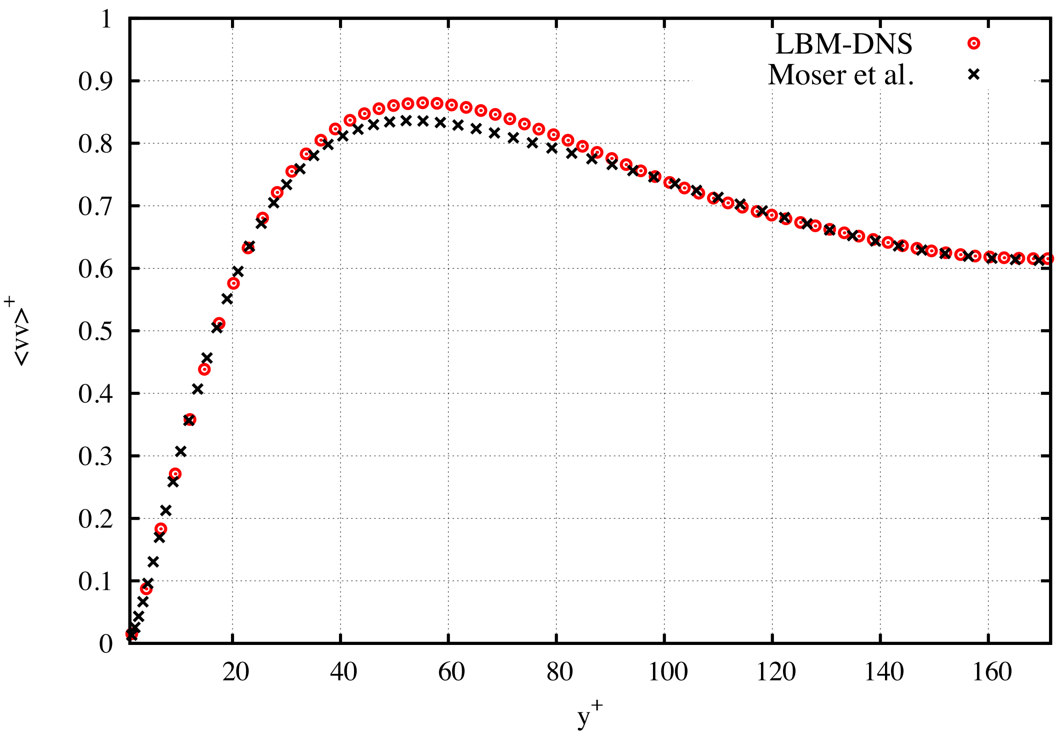


FIGURE 6.18: Root mean square of v-velocity.

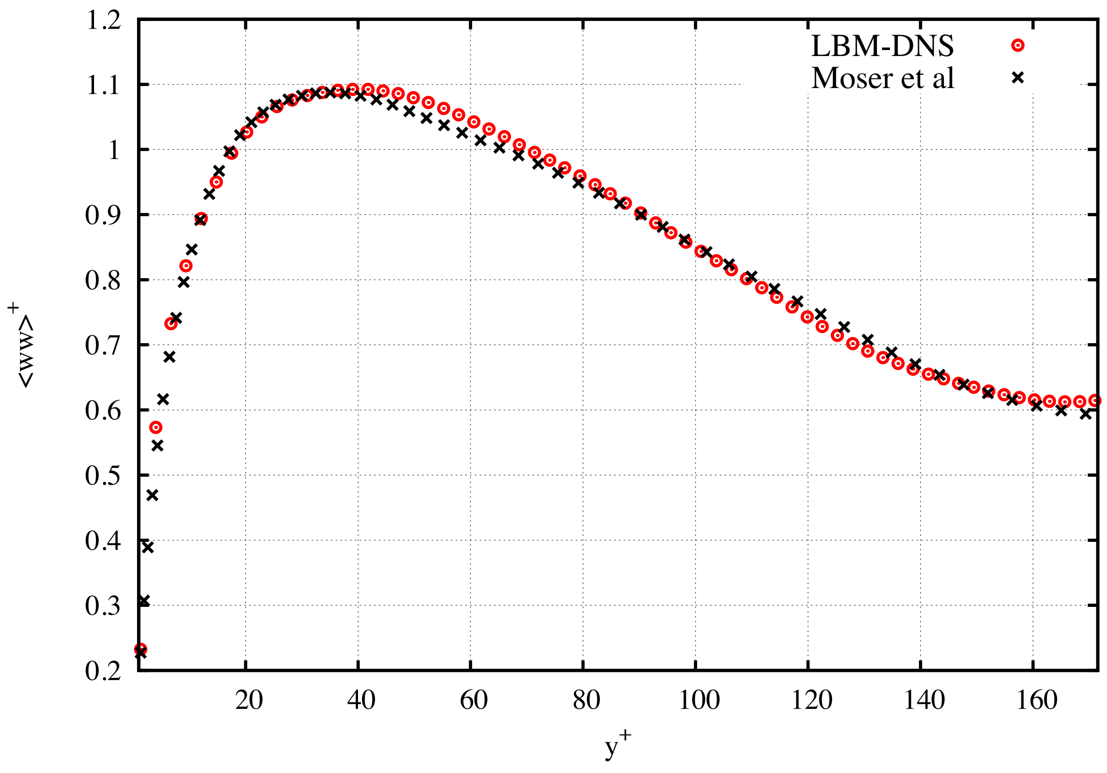


FIGURE 6.19: Root mean square of w-velocity.

6.4 Large Eddy Simulations (LES) for LBM

For engineering flows, DNS becomes a costly method. For such flows where the range of scales of turbulence is wide, LES are computationally convenient with respect to DNS. LES solves the large eddies and their dynamics and the small eddies are modelled. This is a good approximation of turbulence because large eddies are majorly responsible for the transport of momentum, heat, mass etc. while the small eddies are only responsible for the dissipation of energy.

So to achieve LES, sub-grid filter is applied to decompose the velocity into filtered component and sub-grid scale component. The advantage is - DNS requires velocity field to resolve upto the Kolmogorov scale while LES requires resolving upto the filtered velocity only. This allows for quite coarser grid. The filtered velocity component is solved numerically and evolved with the momentum equation of the Navier-Stokes. However, the filtered momentum equation also carries the residual stress tensor. LES requires modelling of this sub-grid tensor τ_{ij}^*. There have been numerous suggestions to model this residual stress tensor satisfying the required physics of the flow, out of which eddy viscosity model is one.

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j}(u_i u_j) = \frac{\partial P}{\partial x_i} + \nu \frac{\partial}{\partial x_j}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})$$

$$\frac{\partial u_i}{\partial x_i} = 0$$

$$\frac{\partial \overline{u_i}}{\partial t} + \frac{\partial}{\partial x_j}(\overline{u_i u_j}) = \frac{\partial \overline{P}}{\partial x_i} + \nu \frac{\partial}{\partial x_j}(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i})$$

$$\frac{\partial \overline{u_i}}{\partial x_i} = 0$$

Let $u_i^{'} = u_i - \overline{u_i}$. This implies, $\overline{u_i u_j} = \overline{(\overline{u_i} + u_i^{'})(\overline{u_j} + u_j^{'})}$

$$\frac{\partial \overline{u}}{\partial t} + \frac{\partial}{\partial x_j}(\overline{\overline{u_i}\,\overline{u_j}}) = \frac{\partial \overline{P}}{\partial x_i} + \nu \frac{\partial}{\partial x_j}(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}) - \frac{\partial \tau_{ij}^*}{\partial x_j}$$

where, $\tau_{ij}^* = \overline{u_i u_j} - \overline{\overline{u_i}\,\overline{u_j}}$

There are many types of LES modelling and research keeps continuing to find a good compromise between the computational cost and accuracy. Smagorinsky LES model or its variants are simple models which are easy to incorporate in LBM. Mathematically, the term τ_{ij}^* is observed to show physics similar to diffusion. So this term can be approximated as dependent on the turbulent eddy viscosity. Now, the implementation only requires to estimate turbulent eddy viscosity. Then, it is added to the molecular viscosity. Also we know that in LBM, viscosity is related to the relaxation time τ. Thus, adjusting locally the relaxation time completes the LBM-LES implementation. This greatly simplifies the implementation of LBM-LES. LES approach implemented to LBM was first proposed by Hou [129].

$$\nu_{total} = \nu_0 + \nu_t = \frac{2\tau_0 - 1}{6} + \frac{\tau_t}{3}$$

$$\tau_{total} = 3\nu_{total} + \frac{1}{2}$$

$$\tau_{total} = 3\left(\nu_0 + \nu_t + \frac{1}{2}\right)$$

Smagorinsky LES model and its variants are defined by the term ν_t. The standard Smagorinsky defines ν_t as

$$\nu_t = C_s^2 \delta x^2 \|S\|$$

where C_s is the Smagorinsky constant (0.18) , δx is the grid spacing(1 for LBM) and $\|S\|$ is the norm of the shear stress tensor, $\|S\| = \sqrt{2S_{ij}S_{ij}}$ and $S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$. Shear stress tensor, S_{ij} was calculated using central difference scheme at the bulk and first order upwind scheme at the boundaries.

Smagorinsky LES is a very simple model with good stability and easy-to-code benefits. However, the problem is that standard Smagorinsky is over dissipative near the walls. Once the flow is bounded by walls, this model fails near the wall regions because of excessive eddy-viscosity arising from the mean shear. So we need better LES model for wall-bounded flows. Another frequently used is Van Driest - Smagorinsky model [130]. Here the modification is at the Smagorinsky constant which has to be multiplied by the Van Driest damping function.

$$C_x = C_s\left(1 - exp^{-\frac{y^+}{A}}\right)$$

where y^+ is the dimensionless wall distance and A = 0.26. Now, the eddy viscosity is defined as $\nu_t = C_x^2 \delta x^2 \|S\|$.

This is still not enough for our study case where the domain expects complex geometry. Van Driest - Smagorinsky model is only best suited for plane boundaries. Also, this model lacks proper theoretical explanation and is empirical.
Another LES model that could address the described issues is Shear-Improved Smagorinsky model (SISM) which was proposed by Lévêque *et al.* [131]. The philosophy of this model is : at regions near the wall, the turbulence is highly anisotropic and experiences high shear effects. The high amount of mean shear accounts for the production

of streamwise vortices and streaks near the wall [132]. However in LES, filter is introduced dividing the scales into grid-scale and subgrid scale. These subgrid scales requires modelling. Now, a proper modelling is very important to represent correct interactions between grid scales and subgrid scales while considering high mean shear effects near the wall. Otherwise, there wont be proper transfer of energy from the grid scales to the sub-grid scales. This means ν_t should be designed in such a way that it should automatically behave like the standard Smagorinsky at the bulk and heavily diminish near the wall region (viscous sub-layer region). Lévêque *et al.* [131] studied the effects of mean shear to the fluid motions and suggested that such ν_t could be achieved by subtracting the mean shear from the instantaneous shear. This means that at regions near the wall where there is high mean shear ν_t vanishes and at the bulk where mean shear is minimum ν_t is equivalent to that of standard Smagorinsky.

$$\nu_t = C_s^2 \delta x^2 \left(\|S\| - \|\bar{S}\| \right)$$

This formulation requires the estimation of norm of mean shear, $\|\bar{S}\|$. Estimation of norm of mean shear for the Shear-Improved Smagorinsky model was suggested by Cahuzac *et al.* [133]. They suggested two time-domain smoothing algorithms named :

- Exponentially weighted moving averages
- Adaptive Kalman filtering

Main idea of these smoothing algorithms is to introduce a cut-off frequency that approximates mean flow as the low-frequency component of the velocity field and the remaining as the turbulent part. At one time, this erases the fluctuations above the cut-off frequency and extracts mean flow at each grid point and in a long run extracts only variations of the mean. Also to note is that the cut-off frequency can't be predefined because in turbulent flows there is no clear distinction of the mean part and fluctuation part over a large frequency spectra for all time-steps. However, smoothing algorithms extract mean-flow behaviour from a time series as the flow evolves.
The first type is an exponentially weighted moving average, which was introduced by Brown [134] and Holt [135]. Estimation of mean is given by:

$$[u]^{n+1} = (1 - c_{exp})\,[u]^n + c_{exp} u^{(n+1)}$$

Here, $[u]^n$ means the estimated mean of velocity component u at time n and c_{exp} is the smoothing factor defined as $c_{exp} \simeq 3.628 f_c \triangle t$. f_c is defined as $f_c = \frac{u_\tau}{\delta}$ where u_τ is the objective friction velocity (taken from DNS) and δ is the channel half width. It is

initialized with $[u]^{(0)} = u^{(0)}$.
This is the basic of the two. The main advantages are that this method is very simple and requires low memory storage but distinctly relies on a proper predefinition of cut-off frequency. This demands more advanced smoothing algorithm : adaptive Kalman filtering.
Kalman filtering is a technique borrowed from control theory. It is adaptive by nature and requires only the present measurement and previous state to update. We do not go deep into the theory but the general idea is that the flow is represented as a random walk model with noise where noise is considered as fluctuations of the instantaneous flow. The theory allows the mean velocity to fluctuate slowly (variance of the difference of mean at two consecutive time steps is fixed, $\delta[u]^n = [u]^n - [u]^{n-1}$) whereas the instantaneous velocity can vary significantly from mean (variance of the difference of instantaneous and mean velocity can update dynamically, $\delta u^n = u^n - [u]^{n-1}$). However as reported by Lévêque *et al.* [131], although the method gives a good dynamically adjusting predictions of mean, the disadvantage of this method is that the assumptions made by this theory do not agree with the physics of turbulent flows. One of the major advantage of this method is that it can be applied to domain with obstacles.

For implementation, this method involves two steps: predict and update. Step: PREDICT estimates the state at current time-step based on previous time-step and step: UPDATE adapts the estimate using an optimal Kalman gain. The algorithm is initialized with $[u]^{(0)} = u^{(0)}$. The steps are as follows:

- PREDICT
$$\widetilde{[u]}^{(n+1)} = [u]^{(n)}$$
$$\widetilde{P}^{(n+1)} = P^{(n)} + \sigma^2_{\delta[u]}$$
Here, P is error covariance and $\widetilde{[u]}$ is the estimate of mean. Also, $\sigma_{\delta[u]} = 3.628 * f_c * \triangle t * u^*$ is kept constant throughout the simulation where u^* is the objective friction velocity (taken from DNS). f_c is defined as $f_c = \frac{u_\tau}{\delta}$ where u_τ is the objective friction velocity and δ is the channel half width.

- UPDATE
This allows to estimate Kalman gain K. Kalman gain is similar to c_{exp} of exponentially weighted moving averages method.
$$K^{(n+1)} = \widetilde{P}^{(n+1)} * \frac{1}{\widetilde{P}^{(n+1)} + {\sigma^2_{\delta u}}^{(n)}}$$

and the mean is given by,

$$[u]^{(n+1)} = \widetilde{[u]}^{(n+1)} + K^{(n+1)} * (u^{(n+1)} - \widetilde{[u]}^{(n+1)})$$

Now for the next step, error covariance (P) and variance of δu are found by,

$$P^{(n+1)} = \widetilde{P}^{(n+1)} - K^{(n+1)} * \widetilde{P}^{(n+1)}$$

$$\sigma^2_{\delta u}{}^{(n+1)} = max(\widetilde{\sigma^2_{\delta u}}, 0.1 * u^{*2})$$

where, $\widetilde{\sigma^2_{\delta u}} = u^* * |[u]^{(n+1)} - u^{(n+1)}|$. $\sigma^2_{\delta u}{}^{(n)}$ takes note of the type of flow and adjusts accordingly. Suppose the flow is laminar, $\sigma_{\delta u}{}^{(n)} \simeq 0$. This leads to $K^{(n+1)} \simeq 1$ and $[u]^{(n+1)} \simeq u^{(n+1)}$.

6.4.1 Validation of LES for LBM

LES can be validated with turbulent flows. Validation can be performed again with a fully turbulent plane channel flow (figure 6.14). In this section, the variants of Smagorinsky LES models will be compared simultaneously with DNS-LBM as well as DNS data by Moser *et al.* [128].

The simulation parameters are as follows:

- Reynolds number, $Re_\tau = 180$. Re_τ is the Reynolds number based on the friction velocity. It is defined as $Re_\tau = \frac{u_\tau \delta}{\nu}$ where u_τ is the friction velocity, δ is the half-channel height and ν is the viscosity.
- grid size (N_X X N_Y X N_Z) = 128 X 64 X 64 (for LBM-LES) and (N_X X N_Y X N_Z) = 256 X 128 X 128 (for LBM-DNS)
- periodic boundary conditions at x-z direction and no-slip wall boundary condition at y-direction
- objective friction velocity, $u_\tau = 0.00427498$ lattice units (extracted from DNS)
- maximum centerline velocity = 0.004466 lattice units
- relaxation time, $\tau = 0.002382$ lattice units (for LBM-LES) and $\tau = 0.004764$ lattice units (for LBM-DNS)

Figures 6.20, 6.21, 6.22, 6.23 show comparisons on mean velocity and fluctuations. It is to be noted that the reference velocity used for scaling is the objective friction velocity as suggested by Lévêque *et al.* [131]. Objective friction velocity is the most accurate estimation of the friction velocity for the given simulation setup and Reynolds number which can also be obtained from DNS. The argument against the conventional scaling is that the present method adjusts the friction velocity error that was wrongly calculated due to less grid points at the boundary layer. This further corrects the centerline velocity error. Thus, we witness better comparison of different variants of Smagorinsky LES.

In figure 6.20, we compare standard Smagorinsky, two SISM types, Van-Driest Smagorinsky with LBM-DNS and Navier Stokes-DNS (Moser *et al.* [128]). It is emphasized that data by Moser *et al.* has a grid refinement at the wall, while LBM-DNS has regular grid with unresolved boundary layer. Looking at the viscous sublayer region, Van-Driest gives the best prediction. At the log-law region, both SISM types perform very well while standard Smagorinsky under-predicts and Van-Driest Smagorinsky over-predicts.

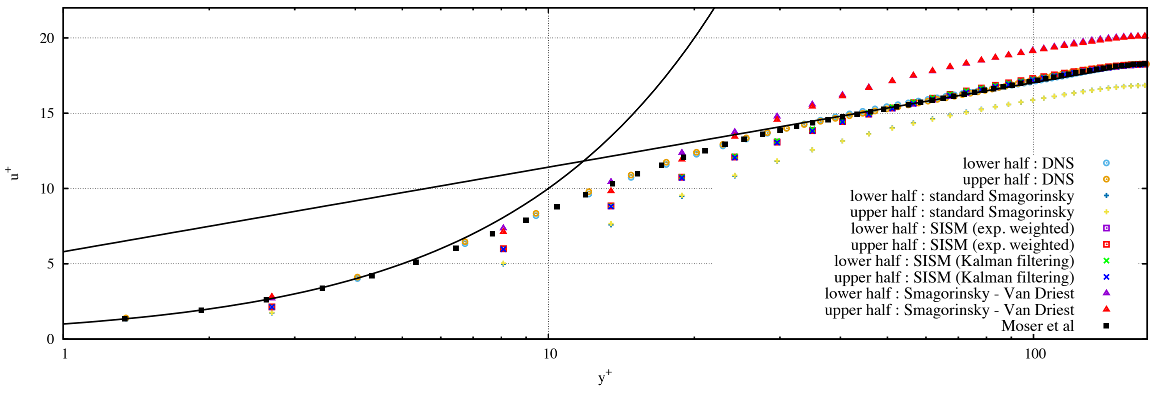


FIGURE 6.20: Comparison of different variants of Smagorinsky LES. Mean streamwise velocity profile.

Figures 6.21, 6.22, 6.23 are RMS fluctuations of u, v, w - velocities respectively. Figures depict that SISM-LES gives comparatively the best prediction.

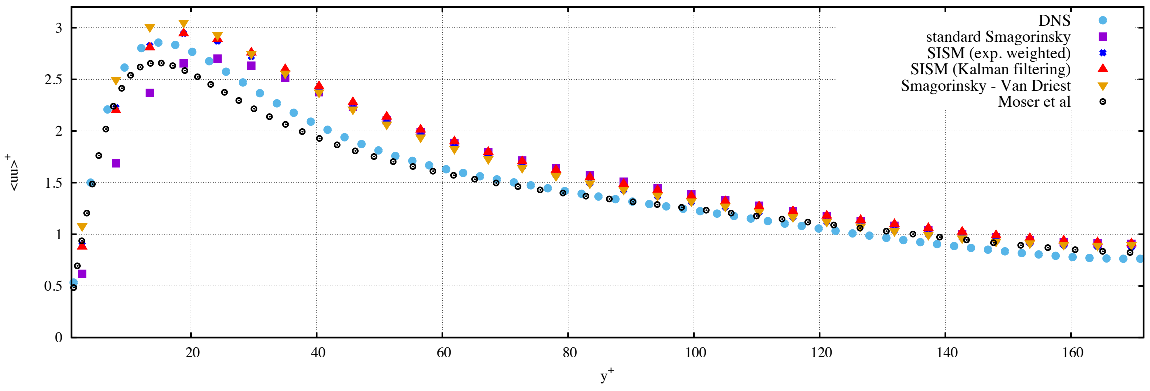


FIGURE 6.21: Comparison of different variants of Smagorinsky LES. RMS of u-velocity.

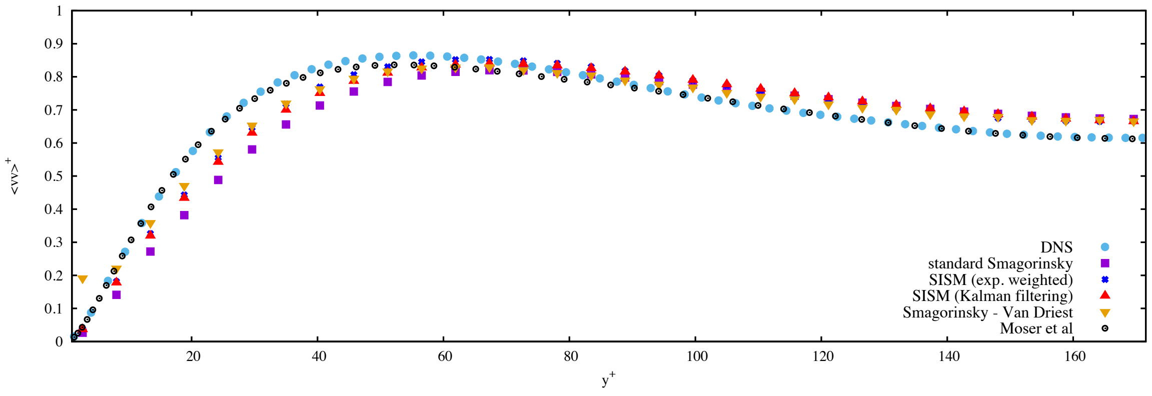


FIGURE 6.22: Comparison of different variants of Smagorinsky LES. RMS of v-velocity.

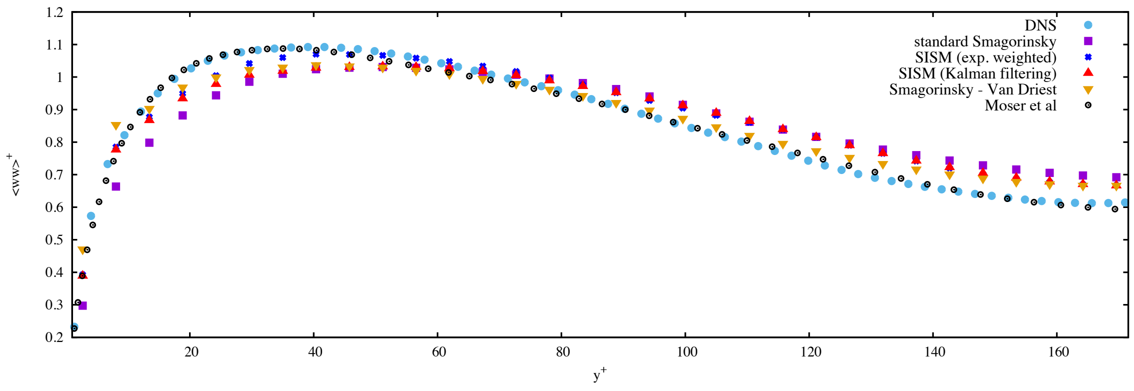


FIGURE 6.23: Comparison of different variants of Smagorinsky LES. RMS of w-velocity.

6.5 Conclusion

The developed LB code is introduced and briefly described with the help of a flow chart. FV LB approach has been described in detail in previous chapters therefore this chapter focussed on the standard LB algorithm. Different boundary conditions have been implemented in the code. Apart from the well known free-slip and no-slip bounce-back boundary conditions, a novel open boundary condition is implemented in the code. The proposed open B.C. aims to reduce the implementation effort for any lattice configurations, while maintaining acceptable accuracy. It adds the concept of non-reflecting boundary condition to further help simulate complex flows. This is validated with the help of 2D laminar jet case at Re = 30 *(laminar flow)* and fully turbulent channel flow at $Re_\tau = 180$ *(turbulent flow)*. LES is also implemented in the code. Smagorinsky LES model and its variants *(Smagorinsky-Van Driest, Shear Improved Smagorinsky model)* have been implemented. These LES models have been validated with the fully-developed turbulent channel flow at $Re_\tau = 180$.

Chapter 7

Conclusions and Perspectives

Part I

In this thesis, we started with deriving standard LB equations from the kinetic theory of gases and showing its consistency with the Navier-Stokes equations. The major drawback of the standard LB was identified to be its restriction to uniform grids. Then we presented a summary on different mesh refinement techniques implemented in the history of LBM to overcome such limitation. Presently, all these mesh refinements still lack in performance compared to the state-of-the-art CFD methods. We made a point based on the study of different mesh refinements that the notion of better mesh refinement attempts seemed to converge towards a method that obeyed conservation laws and was accurate and stable. We argued that such study suggested us a pathway to adopt finite volume approach. Subsequently, a review on different ideas proposed for the development of FV LB approach was done. This gave a general idea of the issues which needed improvement to have a competitive FV LB method which was computationally efficient, acceptably accurate and having good stability range. With this aim, a new finite volume algorithm for the LB equation was proposed. The guidelines in the development of the new FV method were the simplicity and computational efficiency of the implementation, yet retaining a level of accuracy which took the standard LB method as the baseline. Its most original features concerned the semi-implicit approach taken for the time discretisation which improved the stability range and the method of fluxes computation which adopted, for the first time in this context, a QUICK scheme improving in terms of accuracy. The new method has been validated through a systematic comparison with the standard streaming LB approach, by means of test case simulations in laminar as well as in unsteady and turbulent flows with heat transfer (3D Rayleigh Bénard system). The tests have shown that the FV has the same order of spatial accuracy as the ST algorithm, it has however a much more elevated computational costs that we estimated to be around 8-10 times per time-step. One notable advantage of the FV method was the possibility to adopt stretched rectilinear grids, which made it suitable for the simulation of turbulent bounded flows. Taking this into account, *i.e.* taking into consideration (for instance) the minimal number of collocation points required in a boundary layer for a proper simulation, the FV algorithm surpassed the ST method.

A number of improvement can still be made on the proposed algorithm. First, the boundary conditions can be improved. We have noticed that in strongly sheared flows, such as channel flow turbulence, some spurious oscillations at the boundaries can destabilise the simulations. Then we improved the situation with mixed scheme. Mixed scheme introduced linearly reconstructed upwind scheme at the boundaries and QUICK scheme at the remaining part of the domain. This helped the simulations to stand longer but finally destabilised again. Second, in stretched Cartesian grids the number of interpolant coefficients to be stored is considerably more limited than for other cases of structured grids. In the former case the treatment of advection can be further optimised compared to the one used in the present study, allowing for some extra saving in computational costs.

The FV discretization proposed in this work builds on the standard streaming algorithm, as a consequence the two algorithms do not differ much. The major difference is of course the way in which the advection is computed. However, their strong resemblance may be useful in the development of simulation codes that combine the two methods. Therefore, one possible development of the present study is to set-up simulations that use the more efficient ST method in flow domain regions where a fine grid is needed, and the FV method in regions where a more coarse grid will suffice. A schematic drawing depicting such implementation is shown in figure 7.1.

The proposed algorithm is as follows:

1. In FV nodes: compute the advection term $[f_i]$ and store it on a field adv_f_i
2. Make a copy $rhs\_f_i \leftarrow f_i$
3. Add to rhs_f_i the collision term $\frac{\Delta t}{\tau}\left(f_i - f_i^0\right)$
4. Add to rhs_f_i the forcing term $\Delta t\ F_i$
5. In ST nodes : perform a pull stream from rhs_f_i to f_i
6. In FV nodes : perform the time step $f_i \leftarrow f_i + \Delta t\ (\ rhs\_f_i - adv\_f_i)$. Here, Δt should be the same as the one used for ST nodes.

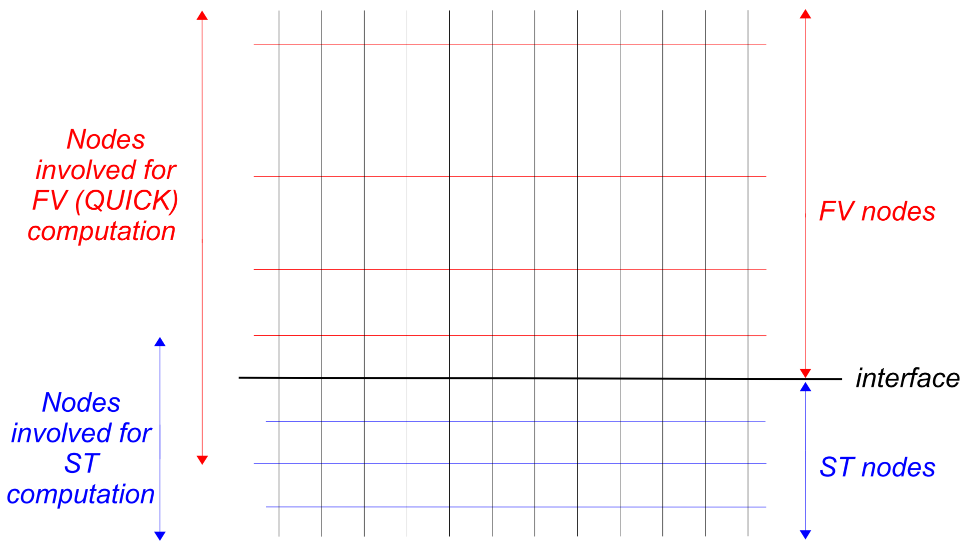

Nodes involved for FV (QUICK) computation
Nodes involved for ST computation
FV nodes
interface
ST nodes

FIGURE 7.1: Coupling of the proposed FV and ST algorithm. The interface separates the FV and ST nodes. However, the nodes actually involved in respective algorithms' computation is labelled in the left. Streaming process requires to pull information from one row of FV nodes (the first row of FV nodes should strictly have same grid size as the ST nodes). Similarly for FV, computation of flux at the interface requires information from two rows of ST nodes (due to QUICK scheme).

Among others, a typical application could be the simulation of atmospheric boundary layer with ST on a regular grid at ground level and FV with less refined grid on the upper residual layer. It is a prospect that we plan to explore in a future work.

Part II

In this part of the thesis, the developed Lattice Boltzmann code was briefly described with the help of a flowchart. The code consists of both the LB algorithms (standard LB and FV LB) co-existing simultaneously. As the FV LB approach was described in detail in Part I, different validations of standard LB is focussed in Part II of the thesis. The validations have been performed with the help of 2D laminar jet case at $Re = 30$ (laminar flow) and fully turbulent channel flow at $Re_\tau = 180$ (turbulent flow).
Different boundary conditions have been implemented in the code. Apart from the well known free-slip and no-slip bounce-back boundary conditions, a novel open boundary condition has been implemented in the code. The basic idea of the proposed open B.C. was to assign equilibrium distribution corresponding to desired density and velocity. It aimed to ease the implementation compared to the existing ones, while maintaining acceptable accuracy. Also, it added the concept of non-reflecting boundary condition to the open boundaries to further help simulate complex flows. Complex flow simulations without the non-reflecting pressure boundary condition obeys mass conservation but triggers reflecting disturbances at the outlet and will produce steep gradients which often destabilises the simulation.

For engineering flows where the range of scales of turbulence is wide, LES are computationally convenient with respect to DNS. To extend the areas of application, LES was also implemented in the present code. Smagorinsky LES is a very simple model with good stability and easy-to-code benefits. However, the problem is that standard Smagorinsky is over dissipative near the walls. Therefore to account for the anisotropy and high shear effects in the wall-bounded plane flows or even in the context of complex geometry, different variants of Smagorinsky have been implemented (Smagorinsky-Van Driest, Shear Improved Smagorinsky model). These LES models have been validated with the fully-developed turbulent channel flow at $Re_\tau = 180$.
Currently, the code (*streaming* LB algorithm) is fully equipped with all the essentials *(LES, 3D boundary conditions, can extend to temperature and concentration field, able to introduce landscape in the domain)* to simulate more realistic configurations. An example in that direction is shown by an effort to simulate a three-dimensional turbulent flow past a cylinder as shown in figure 7.2. The work will be carried on further by validating such types of flow with complex geometry in the domain.

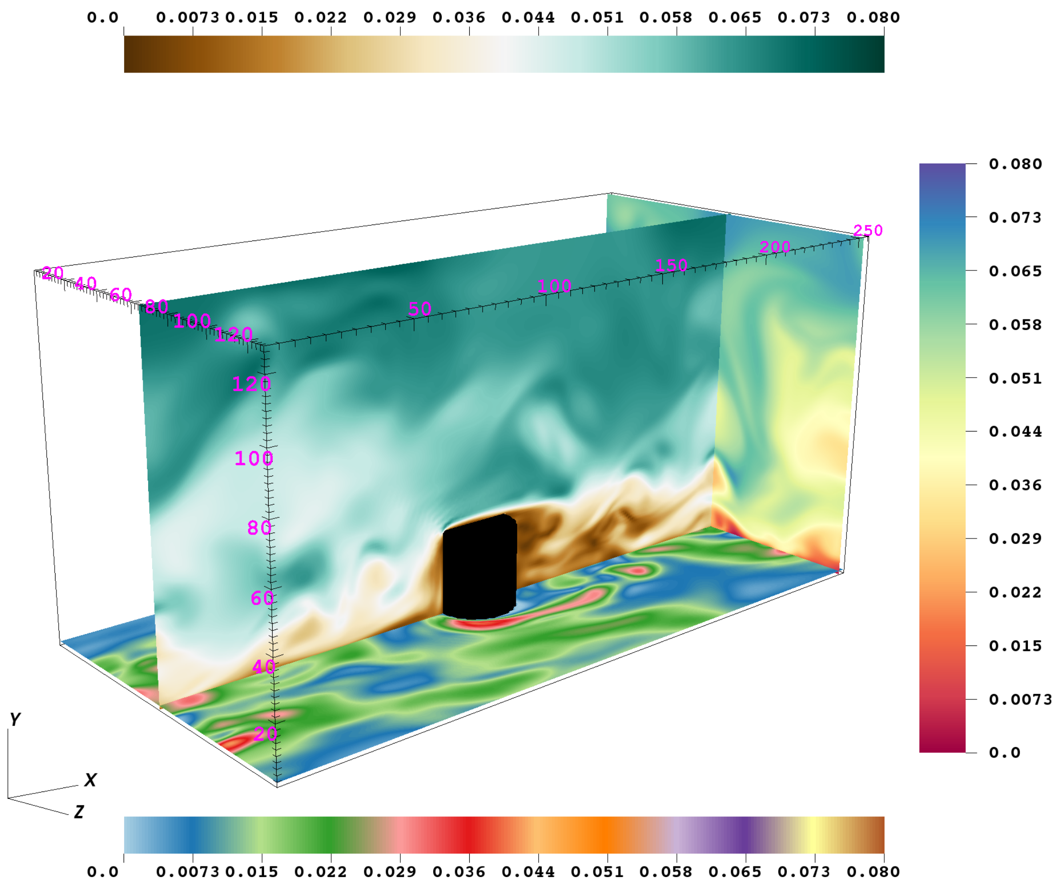


FIGURE 7.2: 3D fully turbulent flow with a cylinder at the bottom-center of the domain. The top wall is imposed with free-slip boundary condition, the bottom wall with no-slip and periodic boundary conditions at x-z directions. Three slices of velocity magnitude contours: *(i)*Slice at 50% of z-axis *(ii)*Slice at 99% of x-axis *(iii)*Slice at 1% of y-axis

Bibliography

[1] X He, L S Luo, and M Dembo. Some progress in Lattice Boltzmann Method. part I. Nonuniform mesh grids. *Journal of Computational Physics*, 129(0255), 1996.

[2] M Bouzidi, D d′Humiéres, P Lalemand, and L S Luo. Lattice Boltzmann equation on a two-dimensional rectangular grid. *J. Comput. Phys.*, 172(2), 2001.

[3] O Filippova and D Hänel. Grid refinement for lattice-BGK models. *J. Comput. Phys.*, 147(219):219–228, 1998.

[4] D Yu, R Mei, and W Shyy. A multi-block lattice Boltzmann method for viscous fluid flows. *Int. J. Numer. Methods Fluids*, 39(2):99–120, 2002.

[5] B Crouse, E Rank, M Krafczyk, and J Tölke. A LB based approach for adaptive flow simulations. *Int. J. Numer. Mod. Phys. B*, 17:109–112, 2003.

[6] F Nannelli and S Succi. The lattice Boltzmann equation on irregular lattices. *J. Stat. phys.*, 68:401, 1992.

[7] G Amati, S Succi, and R Benzi. Turbulent channel flow simulations using a coarse-grained extension of the lattice Boltzmann method. *Fluid Dyn. Res.*, 19:289, 1997.

[8] H Chen. Volumetric formulation of the lattice Boltzmann method for fluid dynamics: Basic concept. *Phys. Rev. E*, 58:3955, 1998.

[9] G Peng, H Xi, C Duncan, and S H Chou. Finite volume scheme for the lattice Boltzmann method on unstructured meshes. *Phys. Rev. E*, 59:4675, 1999.

[10] H Xi, G Peng, and S H Chou. Finite volume lattice Boltzmann method. *Phys. Rev. E*, 59:6202, 1999.

[11] S Ubertini, S Succi, and G Bella. Lattice Boltzmann method on unstructured grids: further developments. *Phys. Rev. E*, 68:016701, 2003.

[12] A Zarghami, M J Maghrebi, J Ghasemi, and S Ubertini. Lattice Boltzmann finite volume formulation with improved stability. *Commun. Comput. Phys.*, 12:42–64, 2012.

[13] M C Sukop and D T Thorne. Lattice Boltzmann modeling. *Springer*, 2007.

[14] K Huang. Statistical Mechanics. *John Wiley & Sons*, 1987.

[15] E M Viggen. The lattice Boltzmann method: Fundamentals and acoustics. *PhD thesis, Department of Electronics and Telecommunications, NTNU-Trondheim*, Februrary 2014.

[16] M Rohde, D Kandhai, J J Derksen, and H E A van den Akker. A generic, mass conservative local grid refinement technique for lattice Boltzmann schemes. *International Journal for Numerical Methods in Fluids*, 51:439–468, 2006.

[17] Y Chen and Q Cai. The lattice Boltzmann method based on quadtree mesh. *Mod. Phys. Lett. B*, 23(3):289–292, 2009.

[18] D V Patil and K N Lakshmisha. Finite volume TVD formulation of lattice Boltzmann simulation on unstructured mesh. *Journal of Computational Physics*, 228: 5262–5279, 2009.

[19] R. M. Clever and F. H. Busse. Transition to time dependent convection. *J. Fluid Mech.*, 65:625–645, 1974.

[20] D W Grunau. Lattice methods for modeling hydrodynamics. *Ph.D. thesis. Colorado State University*, 1993.

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

[22] P Gaspard. Chaos and hydrodynamics. *Physica A*, 240:54, 1997.

[23] J Hardy, Y Pomeau, and O de Pazzis. Time evolution of a two-dimensional model system. I. Invariant states and time correlation functions. *Journal of Mathematical Physics*, 14(12):1746–1759, 1973.

[24] S Wolfram. Theory and Applications of cellular automata. *World Scientific Publishing Co. Ltd.*, 1986.

[25] U Frisch, B Hasslacher, and Y Pomeau. Lattice-gas automata for the Navier-Stokes equation. *Phys. Rev. Let*, 56(14):1505–1508, 1986.

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

[27] X He and L S Luo. Theory of the lattice Boltzmann method : From the Boltzmann equation to the lattice Boltzmann equation. *Phys. Rev. E*, 56:6811–6818, 1997.

[28] 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. *Phys. Rev.*, 94:511–525, 1954.

[29] X He, X Shan, and G D Doolen. Discrete Boltzmann equation model for non-ideal gases. *Phys. Rev. E*, 57:R13–R16, Jan 1998.

[30] X Shan, X F Yuan, and H Chen. Kinetic theory representation of hydrodynamics: a way beyond the Navier-Stokes equation. *J. Fluid Mech.*, 550:413–441, March 2006.

[31] Y H Lin. Polymer Viscoelasticity: Basics, Molecular Theories and Experiments. *World Scientific Publishing, Singapore*, 2003.

[32] D d′Humiéres. Generalized lattice-Boltzmann equations. *Progress in Astronautics and Aeronautics*, 159:450–458, 1992.

[33] D d′Humiéres, I Ginzburg, M Krafczyk, P Lallemand, and L S Luo. Multiple relaxation time lattice Boltzmann models in three dimensions. *Phil. Trans. R. Soc. A*, 360:437–451, 2002.

[34] S Ansumali and I V Karlin. Single relaxation time model for entropic lattice Boltzmann methods. *Phys. Rev. E*, 65(026311), May 2002.

[35] B M Boghosian, P J Love, P V Coveney, I V Karlin, S Succi, and J Yepez. Galilean-invariant lattice-Boltzmann models with H theorem. *Phys. Rev. E*, 68(025103), August 2003.

[36] R Mei and W Shyy. On the finite difference based lattice Boltzmann method in curvilinear coordinates. *Journal of Computational Physics*, 143:426–448, 1998.

[37] J Töelke, M Krafczyk, M Schulz, E Rank, and R Berrios. Implicit discretization and nonuniform mesh refinement approaches for FD discretizations of LBGK models. *Int. J. Mod. Phys. C*, 9:1143–1158, 1998.

[38] D Kandhai, W Soll, S Chen, A Hoekstra, and P Sloot. Finite difference lattice-BGK methods on nested grids. *Computer Physics Communications*, 129:100–109, 2000.

[39] Z Guo and T S Zhao. Explicit finite difference lattice Boltzmann method for curvilinear coordinates. *Phys. Rev. E*, 67(066709), 2003.

[40] A Xu. Finite difference lattice-Boltzmann methods for binary fluids. *Phys. Rev. E*, 71(066706), 2005.

[41] T Lee and C L Lin. A characteristic Galerkin method for discrete Boltzmann equation. *Journal of Computational Physics*, 171:336–356, 2001.

[42] M Seino, T Tanahashi, and K Yasuoka. An analysis of natural convection using the thermal finite element discrete Boltzmann equation. *Computers and Fluids*, 40(1):113–117, 2010.

[43] K E Wardle and T Lee. Finite element lattice Boltzmann simulations of free surface flow in a concentric cylinder. *Computers and Mathematics with Applications*, 65: 230–238, 2013.

[44] G V Krivovichev. On the finite element based lattice Boltzmann scheme. *Applied Mathematical Sciences*, 8(33):1605–1620, 2014.

[45] A K Gunstensen, D H Rothman, S Zaleski, and G Zanetti. Lattice Boltzmann model of immiscible fluids. *Phys. Rev. A*, 43:4320, 1991.

[46] X Shan and H Chen. Lattice Boltzmann model for simulating flows with multiple phases and components. *Phys. Rev. E*, 47:1815, 1993.

[47] E G Flekkoy and H J Herrmann. Lattice Boltzmann models for complex fluids. *Physica A*, 199:1, 1993.

[48] E G Flekkoy, U Oxaal, J Feder, and T Jossang. Hydrodynamic dispersion at stagnation points: Simulations and experiments. *Phys. Rev. E*, 52:4952, 1995.

[49] M Swift, W Osborn, and J Yeomans. Lattice Boltzmann simulation of nonideal fluids. *Phys. Rev. Lett.*, 75:830, 1995.

[50] X P Chen, C W Zhong, and X L Yuan. Lattice Boltzmann simulation of cavitating bubble growth with large density ratio. *Comput. Math. Appl.*, 61:3577–3584, 2011.

[51] A J C Ladd. Numerical simulations of particulate suspensions via a discrete Boltzmann equation. Part 1. Numerical results. *J. Fluid Mech.*, 271:285–309, 1994.

[52] A J C Ladd. Numerical simulations of particulate suspensions via a discrete Boltzmann equation. Part 2. Numerical results. *J. Fluid Mech.*, 271:311–339, 1994.

[53] A J C Ladd and R Verberg. Lattice-Boltzmann simulations of particle-fluid suspensions. *J. Stat. Phys.*, 104:1191–1251, 2001.

[54] Z Xia, K Connington, S Rapaka, P Yue, J Feng, and S Chen. Flow patterns in the sedimentation of an elliptical particle. *J. Fluid Mech*, 625:249–272, 2009.

[55] K Connington, Q Kang, H Viswanathan, A Abdel-Fattah, and S Chen. Peristaltic particle transport using the lattice Boltzmann method. *Phys. Fluids*, 21:053301, 2009.

[56] B M Boghosian, P V Coveney, and A N Emerton. A lattice gas model of microemulsions. *Proc. R. Soc. Ser. A*, 452:1221–1250, 1996.

[57] A Gunstensen and D Rothman. Lattice-Boltzmann studies of immiscible two-phase flow through porous media. *J. Geophys. Res.*, 98:6431–6441, 1993.

[58] Q Kang, D Zhang, and S Chen. Unified lattice Boltzmann method for flow in multiscale porous media. *Phys. Rev. E*, 66:056307, 2002.

[59] D Zhang, R Zhang, S Chen, and W Soll. Pore scale study of flow in porous media: Scale dependency, REV, and statistical REV. *Geophys. Res. Lett.*, 27:1195–1198, 2000.

[60] Z Guo and T S Zhao. Lattice Boltzmann model for incompressible flows through porous media. *Phys. Rev. E*, 66:036304, 2002.

[61] X Shan. Simulation of Rayleigh - Bénard convection using a lattice-Boltzmann method. *Phys. Rev E*, 55:2780, 1997.

[62] Q Kang, P Lichtner, and D Zhang. Lattice Boltzmann pore-scale model for multicomponent reactive transport in porous media. *J. Geophys. Res.*, 111, 2006.

[63] Q Kang, P Lichtner, and D Zhang. An improved lattice Boltzmann model for multicomponent reactive transport in porous media at the pore scale. *Water Resour. Res.*, 43, 2007.

[64] S Chen. Analysis of entropy generation in counter-flow premixed hydrogen air combustion. *Int. J. Hydrogen Energy*, 35:1401–1411, 2010.

[65] K Yamamoto and N Takada. LB simulation on soot combustion in porous media. *Physica A*, 362:111–117, 2006.

[66] S Chen, H Chen, D Martinez, and W Matthaeus. Lattice Boltzmann model for simulation of magnetohydrodynamics. *Phys. Rev. Lett.*, 67:3776, 1991.

[67] P J Dellar. Lattice and discrete Boltzmann equations for fully compressible flow. *Computational Fluid and Solid Mechanics*, pages 632–635, 2005.

[68] B He, Y Chen, W Feng, Q Li, A Song, Y Wang, M Zhang, and W Zhang. Compressible lattice Boltzmann method and applications. *International Journal of Numerical Analysis and Modeling*, 9(2):410–418, 2012.

[69] H Yu and K Zhao. Lattice Boltzmann method for compressible flows with high Mach numbers. *Physical Review E*, 61(4), 2000.

[70] J Bernsdorf, F Durst, and M Schäfer. Comparison of cellular automata and finite volume techniques for simulation of incompressible flows in complex geometries. *Int. J. Numer. Meth. Fluids,*, 29(3):511–525, 1954.

[71] V Bhandari. Detailed investigations of transport properties in complex reactor components. *Master Thesis, Universität Erlangen-Nürnberg, Erlangen, Germany*, 2002.

[72] X He and L S Luo. Lattice Boltzmann model for the incompressible Navier-Stokes equation. *J. Stat. Phys.*, 88(927), 1997.

[73] S Succi. The lattice Boltzmann equation for fluid dynamics and beyond. *Numerical Mathematics and Scientific Computation. Clarendon Press. Oxford*, 2001.

[74] C Cercignani. Mathematical methods in kinetic theory. *Plenum Press, book*, 2nd edition:252 p., 1990.

[75] X Shan, X F Yuan, and H Chen. Kinetic theory representation of hydrodynamics: a way beyond the Navier-Stokes equation. *Journal of Fluid Mechanics*, 550:413–441, 2006.

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

[77] Y H Qian, D d′Humiéres, and P Lallemand. Lattice BGK models for Navier-Stokes equation. *Europhysics Letters*, 17(6):479–484, 1992.

[78] Z Guo, C Zheng, and B Shi. Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Phys. Rev E*, 65(046308), 2002.

[79] P Lallemand and L S Luo. Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, galilean invariance, and stability. *Phys. Rev. E*, 61(6546), 2000.

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

[81] C L Lin and Y G Lai. Lattice Boltzmann method on composite grids. *Physical Review E*, 62(2):2219–2225, 2000.

[82] A Dupuis and B Chopard. Theory and applications of an alternative lattice Boltzmann grid refinement algorithm. *Physical Review E*, 67(066707), 2003.

[83] D Lagrava, O Malaspinas, J Latt, and B Chopard. Advances in multi-domain lattice Boltzmann grid refinement. *J. Comp. Phys.*, 231(14):4808–4822, 2012.

[84] M Geier, A Greiner, and J Korvink. Bubble functions for the lattice Boltzmann method and their application to grid refinement. *Eur. Phys. J. Special Topics*, 171:173–179, 2009.

[85] J Tölke and M Krafczyk. Second order interpolation of the flow field in the lattice Boltzmann method. *Computers and Mathematics with Applications*, 58(5):898–902, 2009.

[86] N Cao, S Chen, S Jin, and D Martinez. Physical symmetry and lattice symmetry in the lattice Boltzmann method. *Phys. Rev. E*, 55:R21–R24, 1997.

[87] M B Reider and J D Sterling. Accuracy of discrete-velocity BGK models for the simulation of the incompressible Navier-Stokes equations. *Comput. Fluids*, 24: 459–467, 1995.

[88] V Sofonea, A Lamura, G Gonnella, and A Cristea. Finite-difference lattice Boltzmann model with flux limiters for liquid-vapor systems. *Phys. Rev. E*, 70(046702), 2004.

[89] I Kim, W Lindquist, and W Durham. Fracture flow simulation using a finite-difference lattice Boltzmann method. *Phys. Rev. E*, 67(046708), 2012.

[90] M Watari and M Tsutahara. Two-dimensional thermal model of the finite-difference lattice Boltzmann method with high spatial isotropy. *Phys. Rev. E*, 67(036306), 2003.

[91] T Kataoka and M Tsutahara. Lattice Boltzmann model for the compressible Navier-Stokes equations with flexible specific-heat ratio. *Phys. Rev. E*, 69(035701), 2004.

[92] V Sofonea and R F Sekerka. Viscosity of finite difference lattice Boltzmann models. *J. Comput. Phys.*, 184(422), 2003.

[93] J C Jo, K W Roh, and Y W Kwon. Viscosity of finite difference lattice Boltzmann models. *Nuclear Engineering and Technology*, 41(5), June 2009.

[94] T J R Hughes, G Scovazzi, and T E Tezduyar. Viscosity of finite difference lattice Boltzmann models. *Journal of Scientific Computing*, 43:343–368, 2010.

[95] F J Higuera. Lattice gas method based on the Chapman-Enskog expansion. *Phys. Fluids A*, 2:1049–1051, 1990.

[96] H Touil, D Ricot, and E Lévêque. Direct and large eddy simulation of turbulent flows on composite multi resolution grids by the lattice Boltzmann method. *Journal Of Computational Physics*, 256:220–233, 2014.

[97] S Geller, S Uphoff, and M Krafczyk. Turbulent jet computations based on MRT and cascaded lattice Boltzmann models. *Computers and Mathematics with Applications*, 65:1956–1966, 2013.

[98] R Benzi, S Succi, and M Vergassola. The lattice Boltzmann equation: Theory and Applications. *Phys. Rep.*, 222(145197), 1992.

[99] M Stiebler, J Toelke, and M Krafczyk. The lattice Boltzmann equation: Theory and Applications. *Computers & Fluids*, 35:814–819, 2006.

[100] P Prestininzi, M La Rocca, and R Hinkelmann. Comparative study of a Boltzmann-based Finite Volume and a lattice Boltzrnann mode1 for shallow water flows in complex domains. *International Journal of Offshore and Polar Engineering*, 24(3):161–167, September 2014.

[101] M Sbragaglia and K Sugiyama. Volumetric formulation for a class of kinetic models with energy conservation. *Phys. Rev. E*, 82:046709, 2010.

[102] H Xi, G Peng, and S C Hou. Finite-volume lattice Boltzmann schemes in two and three dimensions. *Physical Review E*, 60(3), September 1999.

[103] N Rossi, S Ubertini, G Bella, and S Succi. Unstructured lattice Boltzmann method in three dimensions. *International Journal for Numerical Methods in Fluids*, 49: 619–633, July 2005.

[104] Q Zou and X He. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Phys. Fluids*, 9(6), June 1997.

[105] S Ansumali and I V Karlin. Kinetic boundary conditions in the lattice boltzmann method. *Phys. Rev. E*, 66:026311, 2002.

[106] B P Leonard. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer Methods in Applied Mechanics and Engineering*, 19 (1):59–98, 1979.

[107] Z Guo, C Zheng, and B Shi. Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Phys. Rev. E*, 65:046308, 2002.

[108] S Succi. The lattice Boltzmann equation for fluid dynamics and beyond. *Numerical Mathematics and Scientific Computation. Clarendon Press. Oxford*, 2001.

[109] I Ginzburg. Lattice Boltzmann modeling with discontinuous collision components: Hydrodynamic and advection-diffusion equations. *Journal of Statistical Physics*, 126, 2007.

[110] I Ginzburg, F Verhaeghe, and D d′Humiéres. Two-relaxation-time lattice Boltzmann scheme: About parametrization, velocity, pressure and mixed boundary conditions. *Commun. Comput. Phys.*, 3, 2008.

[111] A V Getling. Rayleigh - bénard convection- structures and dynamics. *World Scientific, Advanced series in Nonlinear Dynamics*, 11, 1998.

[112] R P J Kunnen, B J Geurts, and H J H Clercx. Turbulence statistics and energy budget in rotating Rayleigh - bénard convection. *Eur. J. Mech. B*, 28:578–589, 2009.

[113] V Lavezzo, H. J. H. Clercx, and F. Toschi. Rayleigh-Bénard convection via lattice Boltzmann method: code validation and grid resolution effects. *J. of Physics: Conference Series*, 333:012011, 2011.

[114] E Calzavarini, F Toschi, and R Tripiccione. Evidences of Bolgiano - Obhukhov scaling in three-dimensional Rayleigh - Bénard convection. *Phys. Rev. E*, 66: 016304, 2002.

[115] D Lohse and K-Q Xia. Small-scale properties of turbulent Rayleigh-Bénard convection. *Annu. Rev. Fluid Mech.*, 42:335–364, 2010.

[116] S Succi. The lattice Boltzmann equation for fluid dynamics and beyond. *Numerical Mathematics and Scientific Computation. Clarendon Press. Oxford*, 2001.

[117] R Mei, L Luo, and W Shy. An accurate curved boundary treatment in the lattice Boltzman method. *Journal of Computational Physics*, 155:307–330, 1999.

[118] Zhaoli Guo, Chuguang Zheng, and Baochang Shi. An extrapolation method for boundary conditions in lattice Boltzmann method. *Phys. Fluids*, 14:2007–2010, 2002.

[119] M Bouzidi, M Firdaouss, and P Lallemand. Momentum transfer of a Boltzman-lattice fluid with boundaries. *Physics of Fluids*, 13:3452–3459, 2001.

[120] D Yu, R Mei, and W Shy. A unified boundary treatment in lattice Boltzman method. *AIAA*, 2003:0953, 2003.

[121] Z Feng and E Michaelides. The immersed boundary-lattice Boltzmann method for solving fluid particles interaction problems. *Journal of Computational Physics*, 195:602–628, 2004.

[122] J Wu and C Shu. An improved immersed boundary-lattice Boltzmann method for simulating three-dimensional incompressible flows. *Journal of Computational Physics*, 229:5022–5042, 2010.

[123] F-B Tian, H Luo, L Zhu, J C Liao, and X-Y Lu. An efficient immersed boundary-lattice Boltzmann method for the hydrodynamic interaction of elastic filaments. *Journal of Computational Physics*, 230:7266–7283, 2011.

[124] T Inamuro, M Yoshino, and F Ogino. A non-slip boundary condition for lattice Boltzmann simulations. *Phys. Fluids*, 7:2928–2930, 1995.

[125] P. A. Skordos. Initial and boundary conditions for the lattice Boltzmann method. *Phys. Rev. E*, 48:4823–4842, 1993.

[126] J Latt and B Chopard. Lattice Boltzmann method with regularized non-equilibrium distribution functions. *Math. Comp. Sim.*, 72:165–168, 2006.

[127] M Finck, D Hanel, and I Wlokas. Simulation of nasal flow by lattice Boltzmann methods. *Computers in Biology and Medicine*, 37:739–749, 2007.

[128] R D Moser, J Kim, and N N Mansour. Direct numerical simulation of turbulent channel flow up to Re=590. *Physics of Fluids*, 11:943–945, 1999.

[129] S Hou. A lattice Boltzmann subgrid model for high Reynolds number flows. *Fields Institute Communications*, 6:1551165, 1996.

[130] D Lilly. A proposed modification of the Germano subgrid-scale closure method. *Physics of Fluids A*, 4:633–635, 1992.

[131] E Lévêque, F Toschi, L Shao, and J P Bertoglio. Shear improved Smagorinsky model for Large Eddy Simulation of wall-bounded turbulent flows. *J. Fluid Mech.*, 570:491–502, 2007.

[132] B Perot and P Moin. Shear-free turbulent boundary layers. Part 1. physical insights into near-wall turbulence. *Journal of Fluid Mechanics*, 295:199–227, 1995.

[133] A Cahuzac, J Boudet, P Borgnat, and E Lévêque. Smoothing algorithms for mean-flow extraction in Large Eddy Simulation of complex turbulent flows. *Physics of Fluids*, 22:1–14, 2010.

[134] R G Brown. Exponential smoothing for predicting demand. *Arthur D. Little Inc., Cambridge, Massachusetts*, page 15, 1956.

[135] C C Holt. Forecasting trends and seasonal by exponentially weighted averages. *Office of Naval Research Memorandum*, 52, 1957.