

Imperial College London
Department of Mechanical Engineering

Development of a Three-Dimensional Grid Refinement Method for the Application of the Lattice Boltzmann Method to High Reynolds Flows

Zhishang Xu

January 2022

Supervised by Dr. Sina C. Stapelfeldt, Dr. Ricardo Puente,
and Prof. Ricardo Martinez-Botas

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Mechanical Engineering of Imperial College London
and the Diploma of Imperial College London

Declaration of Originality

I herewith certify that the content of this thesis is my own work and any additional material has been properly acknowledged. This thesis is original and no part of this work has been submitted to any other University or Institution for any other degree or qualification.

Zhishang Xu
January 2022

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Dr. Sina Stapelfeldt for providing me the opportunity to carry out this PhD project. I am grateful for her invaluable guidance and patient support during the past four years, which led my way from apprentice to academic researcher. I would like to acknowledge her help in my academic writing and efforts in organising various activities.

I would like to gratefully offer my thanks to my co-supervisor Dr. Ricardo Puente for his great knowledge and advice, which benefit me a lot in this research. I am grateful to his help in my writing and thesis revision.

I am thankful to Professor Mehdi Vahdati for his patient instructions in my first two years. I also appreciate his concern to me during the pandemic.

I would like to thank Mr. Sam Mitchell for sharing his thoughts and ideas of his research. I am encouraged from discussing with him.

I would like to offer my special thanks to Professor Ferdinando Rodriguez y Baena for his kind support and comfort in my most difficult time.

Many thanks to people at the Vibration UTC at Imperial College for organising seminars and socials. I appreciate Dr. Prathiban Sureshkumar for helping me with HPC setup, Dr. Wenqiang Zhang for his tips on living abroad, and Mr. Peter Higgs for his help in research equipment.

I am grateful to China Scholarship Council (CSC) for financially supporting my study and research. I am honoured for being offered this opportunity.

Last but not least, I would like to express my greatest gratitude to my parents for supporting me and believing me during this long lonely but beautiful journey.

Abstract

The lattice Boltzmann method (LBM) models fluid dynamics based on kinetic theory. It discretises the continuous Boltzmann equation in velocity, space and time and solves a transport equation for particle populations. When solved on uniform Cartesian grids, this results in a highly scalable algorithm which can accurately and efficiently model unsteady and turbulent flows. However, the reliance on uniform grids makes the LBM prohibitively expensive for the simulation of high Reynolds number flows. To reduce computational costs, grid refinement is required.

A hierarchical grid refinement method with regularised coupling is introduced in this thesis, and its implementation in the open-source LBM solver OpenLB is explained. The regularised coupling restricts the information exchange at the refinement interface to the leading order terms of the particle populations. This ensures that local viscous stresses are conserved across the interface and prevents numerical instabilities. The method is validated against two benchmarks: the vortex shedding flow around a circular cylinder and the lid-driven flow inside a cubic cavity. It is demonstrated that the grid-refined methodology accurately captures the flow features at different Reynolds numbers, including two instability modes during the cylinder wake transition. Refinement reduces the number of active cells by a factor of 28 for the cylinder at Reynolds number $Re = 300$. For the lid-driven cavity case at $Re = 3200$, the performance efficiency and continuity across the refinement interface are assessed. The regularised coupling diminishes the transmission of spurious waves and improves numerical stability.

Finally, this method is applied with a Smagorinsky subgrid model to an open shallow cavity at $Re = 50,000$. Good agreement between the LBM solution and reference experimental and numerical data in terms of time-average velocities and turbulence statistics exists. This successful application demonstrates that the newly implemented methods achieve accurate and stable LBM simulations at high Reynolds number with efficient use of computational resources.

Contents

1. Introduction	17
1.1. Background and Motivation	17
1.2. Objectives and Thesis Outline	19
2. Fundamentals	22
2.1. Introduction to Fluid Dynamics	22
2.1.1. Navier-Stokes Equations	22
2.1.2. Dimensional Scaling	24
2.2. Computational Fluid Dynamics	29
2.2.1. NS Based Methods	30
2.2.2. Particle-Based Methods	31
2.3. The Lattice Boltzmann Method	32
2.3.1. Kinetic Theory	32
2.3.2. Discretisation of the Boltzmann Equation	35
2.3.3. The Basic Collision and Streaming Algorithm	39
2.4. Chapman-Enskog Analysis	40
2.5. Non-Dimensionalisation and Parameter Selection	44
2.5.1. Lattice Units	44
2.5.2. Stability and Accuracy Analysis	46
2.6. Boundary Conditions in the LBM	49
2.6.1. Simple Bounce-Back Method	49
2.6.2. Non-Equilibrium Bounce-Back Method	50
2.6.3. Interpolated Bounce-Back Method	52
2.6.4. Periodicity Condition	52
2.6.5. Free-Slip Condition	53
2.7. Summary	53
3. State of the Art of High Reynolds LBM	54
3.1. The Multiple-Relaxation-Time Operator	55
3.2. The Regularised BGK Operator	56
3.3. Subgrid Models in LBM	57
3.4. Entropic Lattice Boltzmann Method	60

3.5.	Grid Refinement	62
3.5.1.	Non-Uniform Grid Layout Refinement	62
3.5.2.	Vertex-Centred Hierarchical Refinement	63
3.5.3.	Cell-Centred Hierarchical Refinement	64
3.5.4.	Grid Refinement with Advanced Collision Operators . . .	65
3.6.	Summary	66
4.	Grid Refinement Method with Regularised Coupling	68
4.1.	Methodology	68
4.1.1.	Grid Layout and Parameter Scaling	68
4.1.2.	Direct Coupling	70
4.1.3.	Interpolation and Restriction	71
4.1.4.	Regularised Coupling	73
4.2.	Treatment on Periodic Boundary Condition	76
4.3.	Overall Collision and Streaming Process	76
4.4.	Summary	77
5.	OpenLB Implementation	78
5.1.	The Open-Source Library OpenLB	78
5.1.1.	Main Features	78
5.1.2.	Data Structure	81
5.1.3.	Description of the Simulation Set Up Process	83
5.2.	Implementation of Grid Refinement Method	84
5.2.1.	Overall Data Structure	86
5.2.2.	The <i>Grid3D</i> Class	86
5.2.3.	The <i>Coupler3D</i> Class	89
5.2.4.	Collision and Streaming Algorithm	90
5.3.	Parallel Performance Testing	90
5.3.1.	Evaluation of OpenLB	92
5.3.2.	Evaluation of the Grid Refinement Implementation	95
5.4.	Summary	96
6.	Validation on Benchmark Cases	97
6.1.	Vortex Shedding Flow around a Circular Cylinder	97
6.1.1.	Case Description	97
6.1.2.	Domain Configuration and Parameter Selection	99
6.1.3.	Results	101
6.2.	Lid-driven Flow inside a Cubic Cavity	108
6.2.1.	Case Description	108

6.2.2. Domain Configuration and Parameter Selection	109
6.2.3. Convergence and Computational Costs	110
6.2.4. Results	113
6.3. Summary	119
7. Application on High Reynolds Number Simulations	121
7.1. Problem Specification	121
7.2. Simulation Configuration	122
7.3. Results	124
7.3.1. Comparison of Time-Averaged Velocity Profiles	125
7.3.2. Comparison of Second-Order Statistics	128
7.3.3. Influence of the C_s Coefficient	130
7.4. Summary	132
8. Conclusions and Future Work	133
8.1. Conclusions	133
8.2. Future Perspectives	134
Appendices	146
A. Appendix	147
A.1. Data Structure of Some Member Class of <i>Grid3D</i>	147
A.2. Data Structure of <i>FineCoupler3D</i> and <i>CoarseCoupler3D</i>	148
A.2.1. The <i>FineCoupler3D</i> Class	148
A.2.2. The <i>CoarseCoupler3D</i> Class	149

List of Figures

1.1. Sketch of a uniform Cartesian grid of a fluid domain. The solid line denotes the numerical boundary with boundary nodes marked in black circles. The inner fluid nodes are represented by white circles.	18
2.1. Sketch of a fluid element	22
2.2. Turbulent energy cascade	26
2.3. Turbulent kinetic energy spectrum	27
2.4. Isentropic variation of density with Mach number	28
2.5. A sketch of the distribution function against the velocity. The distribution function and its equilibrium are denoted in black and blue colour respectively.	33
2.6. D2Q9 and D3Q19 velocity sets	37
2.7. Illustration of the collision and streaming process on a lattice . .	39
2.8. Sketch of a computational grid around a 2D cylinder	45
2.9. Stability $u - \tau$ curve of BGK collision operator	47
2.10. Illustrations of particles before and after bounce back to a solid wall	49
2.11. Illustrations of wet-node method for open boundaries. The missing populations are marked in dotted lines.	51
2.12. Illustrations of the interpolated bounce-back method	52
3.1. LES length scales in the turbulent energy spectrum	58
3.2. Grid layout of the vertex-centred structure. Coarse nodes are marked as $\circ$ and fine nodes are $\times$	63
3.3. Grid layout of the cell-centred structure. Coarse nodes are marked as $\circ$ and fine nodes are $\times$	65
4.1. General layout of a coarse and a fine grid. Coarse nodes are marked as $\circ$ and fine nodes are $\times$. Nodes marked in blue in the left figure are boundary nodes.	69
4.2. Interpolation on 2D centre (a), edge (b), and corner (c) nodes . .	72

4.3. Interpolation on 1D centre (a) and edge (b) nodes	73
4.4. Problematic (a) and corrected (b) grid layout for periodic boundary	75
4.5. Collision and streaming in the overlapped region. Solid arrows de- note the population streaming from inner fluid nodes to boundary nodes, while dashed arrows represent the population streaming from boundary nodes to inner fluid nodes.	76
5.1. Scalability performance of OpenLB [92]	80
5.2. Sketch of the data structure in OpenLB.	82
5.3. Class diagram of the grid refinement data structure.	85
5.4. Scalability performance of OpenLB on HPC facilities at Imperial College	93
5.5. Strong scaling performance of OpenMP with total number of ac- tive cells 0.14 million, 8 million, and 66 million.	94
5.6. Weak scaling of the <i>Grid3D</i> class versus original OpenLB (olb) class	94
5.7. Scalability of the grid refinement code versus original OpenLB .	95
6.1. Sketch of the turbulent transition in the cylinder wake	98
6.2. Illustrations of domain layout with boundary conditions, and the refinement layout	99
6.3. Velocity contours of the magnitude ($ \mathbf{u} $), y-component (u_y), and z-component (u_z) at Re=100, 200 and 300 with refined grid . . .	102
6.3. Velocity contours of its magnitude ($ \mathbf{u} $), y-component (u_y), and z-component (u_z) at Re=100, 200 and 300 with refined grid (cont.)	103
6.4. Isosurface plot of vorticity magnitude at Re=100, 200 and 300 .	104
6.5. Isosurface plot of streamwise (x-) component of vorticity at Re=100, 200 and 300	105
6.6. Pressure coefficients around the cylinder at Re=100	106
6.7. Time-averaged velocity profiles of LBM results with refined grid compared with simulation results from Jiang et al. [99] in the cylinder wake of Re=300 at $x/D = 1$ and $x/D = 2$	107
6.8. Domain geometry and refinement layout of the 3D lid-driven cavity	109
6.9. Measured standard deviation of flow velocity versus computa- tional steps after the initial transient	110
6.10. Time-averaged contours of velocity (u_x and u_z) on the plane of symmetry ($y=0.5$) of uniform N=100, Case A with 1 level refine- ment, Case B with 2 level refinement and reference [105]	112

6.10. Time-averaged contours of velocity (u_x and u_z) on the plane of symmetry ($y=0.5$) of uniform $N=100$, Case A with 1 level refinement, Case B with 2 level refinement and reference [105] (cont.) .	113
6.11. Instantaneous streamline contours of Uniform $N = 100$, Case A and Case B on the plane of symmetry ($y=L/2$)	115
6.12. Time-averaged U_x along z -axis at $x = L/2$ and U_z along x -axis at $z = L/2$ on ($y = L/2$) plane	116
6.13. Time-averaged U_x and U_z profiles of uniform grid with resolution $N = 200$ of different lattice velocity	117
6.14. Convergence comparison of simulations with and without regularised coupling	118
7.1. Domain geometry and refinement layout of an open shallow cavity	121
7.2. Measured standard deviation of flow velocity versus computational steps after convergence in the open cavity	124
7.3. Snapshot of the instantaneous vorticity contours around the cavity on X-Z plane. The cavity locates in $x \in [0.1, 0.2]$ and $z \in [0, 0.3]$.	125
7.4. Comparison of time-averaged x- (left) and y-component (right) velocity profiles at three different streamwise locations in the cavity	126
7.4. Comparison of time-averaged x- (left) and y-component (right) velocity profiles at three different streamwise locations in the cavity (cont.)	127
7.5. Comparison of the time-averaged streamline contours from experimental and LBM-LES results	128
7.6. Comparison of time-averaged velocity fluctuation (left) and Reynolds shear stress (right) at different streamwise locations in the cavity	129
7.6. Comparison of time-averaged velocity fluctuation (left) and Reynolds shear stress (right) at different streamwise locations in the cavity (cont.)	130
7.7. Comparison of flow statistics with Smagorinsky constants $C_s = 0.1$ and $C_s = 0.25$	131

List of Tables

2.1. Velocity vectors and their weights of D2Q9	37
2.2. Conversion factors for common physical variables in a LBM simulation	44
6.1. Summary of simulation parameters of the vortex shedding case at Re=100, 200, and 300	100
6.2. Simulation parameters of different refinement levels at Re=300 .	101
6.3. Computational costs for the lid-driven cavity case, scaled to the “Uniform $N = 100$ ” case, which has $\Delta t = 2 \times 10^{-4}$, 1 million total active cells and a total convergence time of 3.7 hours	111
6.4. Grid Reynolds number at each resolution	119
7.1. Summary of simulation parameters of the high Reynolds case at different refinement levels	123

Nomenclature

Acronyms

ADM	Approximate Deconvolution Method
BGK	Bhatnagar–Gross–Krook
CFD	Computational Fluid Dynamics
DNS	Direct Numerical Simulation
ELBM	Entropic Lattice Boltzmann Method
FD	Finite Difference
FE	Finite Element
FV	Finite Volume
FVLBE	Finite Volume Lattice Boltzmann Equation
HRR	Hybrid Recursive Regularised
ISLBE	Interpolation-Supplemented Lattice Boltzmann Equation
KBC	Karlin-Bösch-Chikatamarla
LB(M)	Lattice Boltzmann (Method)
LBE	Lattice Boltzmann Equation
LES	Large Eddy Simulation
LGA	Lattice Gas Automata
MPI	Message Passing Interface
MRT	Multiple Relaxation Time
NS	Navier-Stokes
RLB	Regularised Lattice Boltzmann

RR Recursive Regularised

VTK Visualization Toolkit

WALE Wall Adapting Local Eddy-viscosity

Greek Symbols

σ' Viscous Stress Tensor

σ Total Stress Tensor

ξ Particle Velocity

Δ Filtering Length

ϵ Perturbation Expansion Parameter

η Kolmogorov Scale

γ Heat Capacity Ratio

κ Wave Number

μ Dynamic Viscosity

μ_B Bulk Viscosity

ν Kinematic Viscosity

Ω Collision Operator

ω Weight Function

ρ Fluid Density

τ Relaxation Time

θ Dimensionless Temperature

ε Energy Dissipation Rate

Roman Symbols

$\mathbf{b}$ External Body Force Density

$\mathbf{c}$ Discrete Particle Velocity

$\mathbf{H}$ Hermite Polynomials

$\mathbf{M}$	Transformation Matrix
$\mathbf{Q}_i$	Velocity Tensor of i -th Discrete Velocity
$\mathbf{S}$	Strain Rate Tensor
$\mathbf{u}, \mathbf{U}$	Fluid Velocity
$\mathbf{u}', \mathbf{U}'$	Fluctuation Velocity
$\mathbf{x}$	Physical Space
Δt	Time Spacing
Δx	Grid Spacing
C	Conversion Factor
C_s	Smagorinsky Coefficient
c_s	Speed of Sound
E	Kinetic Energy Density
e	Internal Energy Density
F	Force
f	Particle Distribution Function
f^*	Post-Collision Distribution Function
f^{eq}	Equilibrium Distribution Function
f^{neq}	Non-Equilibrium Distribution Function
k	Turbulent Kinetic Energy
l	Length
l_{mfp}	Molecular Mean Free Path Length
N	Grid Resolution
p	Pressure
q	Number of Discrete Velocities
R	Specific Gas Constant

S	Surface
s	Entropy
T	Temperature
t	Time
V	Volume
w	Discrete Weight Function

Subscripts

0	Characteristic Property
b	Boundary Node
c	Coarse Grid
f	Fine Grid
lb	Variable in Lattice Units
st	Stagnation Variable
w	Wall Node

Superscripts

$*$	Dimensionless Variable
-----	------------------------

1. Introduction

1.1. Background and Motivation

Computational fluid dynamics (CFD) is an analysis technique in which flows are modelled by numerically solving the equations governing fluid mechanics, discretised in space and time. On a macroscopic level the flow behaviour is governed by the Navier-Stokes (NS) equations, which describe the conservation of mass, momentum and energy of a continuous fluid. CFD methods based on the solution of the NS equations are currently prevalent in scientific and industrial applications. However, over the last two decades an alternative method based on the solution of the Boltzmann equations has gained popularity.

The lattice Boltzmann method (LBM) models the mesoscopic behaviour of fluid particles as described by statistical kinetic theory, representing the transport and collision of particle populations on a discrete lattice. When compared to CFD methods based on the solution of the NS equations, which typically use finite-volume or finite-element discretisation, the LBM method has a number of advantages: Firstly, the advection of particle populations is linear and non-linearity is restricted to the collision process, which takes place on local nodes, making the LBM straightforward to implement. This is achieved by the particle description and the discretisation of the particle velocity and collision process. Secondly, the transport and collision of populations is modelled in separate collision and streaming processes, making LBM very efficient to parallelise [1]. This feature significantly improves the computational speed, for example, a 40x speedup was observed using the LBM over a spectral element solver with similar number of grid cells [2].

In recent years, there has been considerable research on scale-resolving simulations for aeronautical applications. With regard to turbomachinery, the costs of large eddy simulations (LES) for full scale fans or compressors at Reynolds numbers around 2.5 million is prohibitively high [3]. Yet, especially when it comes to correctly predicting aeroelastic phenomena in turbomachinery, the accurate prediction of unsteady aerodynamics is crucial. Since most aeroelastic phenomena occur at off-design conditions, where the flow is unsteady, separated

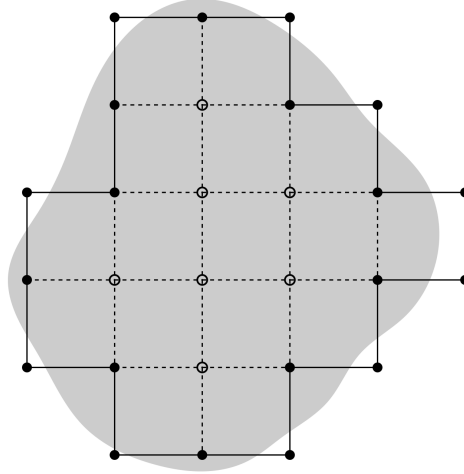


Figure 1.1.: Sketch of a uniform Cartesian grid of a fluid domain. The solid line denotes the numerical boundary with boundary nodes marked in black circles. The inner fluid nodes are represented by white circles.

and highly three-dimensional, this is extremely challenging [4] and an accurate but computationally efficient alternative to finite-volume Navier-Stokes solvers would bring huge advantages. LBM is a promising candidate because of its high locality and amenability to parallelisation [1], as most of the computations take place on local nodes. In addition, the use of Cartesian grids simplifies meshing and the use of immersed boundary methods simplifies the modelling of fluid-structure interaction, since no mesh morphing is necessary to model structural deformations.

A challenge in LBM and any CFD method is the accurate simulation of turbulence. To fully resolve small-scale eddies, very fine discretisations (grid cells/lattices) are required. As sketched in Figure 1.1, the most common form of computational grid (or lattice) used in LBM is uniform and structured. Although this grid layout simplifies meshing, computation and communication, the computational costs grow considerably as grid resolution (N) requirements increase. This poses a particular problem for high Reynolds number (Re) simulations, as the size of the smallest turbulent eddies and hence grid size (N^3) scales with $N^3 \geq Re^{9/4}$ (see Section 2.1.2 for more exposition). However, many CFD problems do not require a dense grid everywhere in the computational domain and it is sensible to refine the grid only in certain regions. When applying grid refinement to the LBM, issues arise due to the tight coupling between the discretisation and the physical model. This will be explained in Section 4.1.

Grid refinement methods are an ongoing topic of research in the LBM com-

munity. Although extensive research into grid refinement methods for the LBM has been carried out, stability problems on the refinement interface are known to cause issues in high speed turbulent flow simulations [5]. Moreover, most implementations of grid refinement are proprietary [6, 7, 8, 9], rendering their evaluation and widespread use by the wider community difficult.

1.2. Objectives and Thesis Outline

In recent years, one of the most challenging area in CFD is the simulation of unsteady turbulent flows including transition and separation [10]. Accurately predicting the turbulent flow phenomena in aerodynamics such as boundary layer separation calls for robust and reliable physical modelling, effective and qualified mesh generation, and efficient parallelisation in High Performance Computing (HPC). Despite the advantages of the LBM in mesh generation and parallelisation, its performance is inherently limited in low speed and incompressible flows.

To enable the use of the LBM for problems such as boundary layer separation over aerofoils, simulations in regimes with $Re \geq 10^5$ must be achieved. To this end, this research implements a novel grid refinement method into the open-source solver OpenLB [11], which is a widely-used solver in the LBM community. After investigating other open-source LBM codes such as Palabos, Musubi and Sailfish and commercial solvers like XFlow, and ultraFluidX, OpenLB is chosen in this research for its wide range of implemented turbulence models. It has shown good scalability on parallel clusters [12, 13], and is well documented [14] and maintained by an active community [15]. However, it currently does not have grid refinement capabilities, which limits its application to low Reynolds regimes or small computational domains.

Most existing grid refinement methods are based on reconstructing particle populations directly from one grid to the other, leading to numerical errors in aerodynamics during particle coupling and instabilities on the refinement interface especially at high Reynolds numbers. This thesis therefore aims to contribute to the LBM community by developing a novel grid refinement method and make the first step in OpenLB towards its robust use on aerospace applications, where high Reynolds and compressible flows dominate.

The objectives are to:

1. Develop and implement a novel three-dimensional grid refinement method with regularised coupling instead of conventional direct coupling.

2. Validate the method on academic benchmarks and assess the performance of LBM in comparison to Navier-Stokes based solvers.
3. Apply the grid refinement for the simulation of high Reynolds flows.

To place the research into context, this thesis will first explain the fundamentals of fluid mechanics, CFD methods and the lattice Boltzmann method. It then reviews literature on adapting the LBM towards high Reynolds simulations. The methodology of the grid refinement method with regularised coupling will be detailed afterwards. Particular treatment of periodic boundary condition is also discussed. The validation of the grid refinement method is performed on two benchmark cases, a circular cylinder and lid driven cavity, and the computational performance on parallel architectures is evaluated. A high Reynolds number simulation on an open shallow cavity is successfully carried out with the help of the implemented grid refinement method.

The chapters of this thesis are summarised below:

- Chapter 2 first briefly introduces fluid dynamics and kinetic theory, which is the theoretical basis of the LBM. Chapman-Enskog analysis is also explained, as it is a vital theoretical framework from which different variants of the LBM can be formulated and evaluated.
- Chapter 3 discusses the state of the art of adapting the LBM to high Reynolds number simulations. While most of these methods focus on advanced collision models, they must be accompanied by grid refinement to reduce the total computational costs. Various existing grid refinement approaches are discussed.
- Chapter 4 covers the regularised coupling grid refinement methodology, including detailed coupling method and interpolation algorithm of one particular refinement structure. Specific treatments on the interaction between the refinement interface and periodic boundary nodes are also discussed.
- In Chapter 5, the detailed implementation is described after introducing the data structures and algorithms of OpenLB. Its parallel performance on facilities at Imperial College is assessed afterwards.
- Chapter 6 validates the implemented grid refinement method. The results for two canonical test cases are presented, vortex shedding behind a cylinder and a lid driven cavity, assessing the accuracy of the method as well as its computational efficiency. The refinement interface with regularised coupling is demonstrated to reduce the spurious fluctuations.

- A high Reynolds number case is detailed in Chapter 7. Grid refinement is coupled with the Smagorinsky subgrid model in order to successfully achieve accurate results at a manageable computational cost. The effect of the Smagorinsky coefficient is also discussed.
- Finally, conclusions are extracted from the evaluation of the current implementation and simulation results, and some perspectives are proposed for further research.

2. Fundamentals

2.1. Introduction to Fluid Dynamics

Although composed of discrete molecular particles, fluids on a macroscopic level can be considered as a continuum which obeys the conservation of mass, momentum and energy. The continuum assumption is valid as long as the molecular mean free path is significantly smaller than the characteristic length scale. In the following sections the model of continuum fluid mechanics and the challenges associated with the simulation of engineering flows are presented.

2.1.1. Navier-Stokes Equations

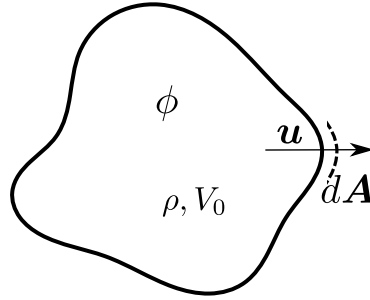


Figure 2.1.: Sketch of a fluid element

The basic model of continuum fluid mechanics is the set of Navier-Stokes equations, which arise from the application of Newton's laws to fluid elements. Considering a small fluid element in Figure 2.1 with density ρ and volume V_0 , the change rate of a conserved property ϕ can be expressed as the change rate in the control volume plus the net flux rate of ϕ across the boundary surface S with velocity $\mathbf{u}$ according to the Reynolds transport theorem, i.e.,

$$\frac{d}{dt} \int_{V_0} \rho \phi dV = \int_{V_0} \frac{\partial(\rho \phi)}{\partial t} dV + \oint_S \rho \phi \mathbf{u} d\mathbf{A} . \quad (2.1)$$

The surface integral on the right-hand side of Eq. 2.1 can be transformed into a

volume integral using the divergence theorem as

$$\oint_S \rho \phi \mathbf{u} d\mathbf{A} = \int_{V_0} \nabla \cdot (\rho \phi \mathbf{u}) dV . \quad (2.2)$$

Applying $\phi = 1$ to Eq. 2.1 and Eq.2.2, yields the conservation of mass equation in integral form. Grouping every term inside a single integral via application of the fundamental theorem of calculus, yields the continuity equation in differential form:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 . \quad (2.3)$$

Similarly, the conservation of momentum can be expressed by replacing ϕ with the fluid velocity $\mathbf{u}$, and balancing it with internal and external forces, which gives

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{b} , \quad (2.4)$$

where $\boldsymbol{\sigma}$ is the stress tensor, whose divergence gives the internal forces, and $\rho \mathbf{b}$ denotes the external body forces.

The total stress tensor $\boldsymbol{\sigma}$ is given by the sum of two components: a hydrostatic term representing the compression of the fluid element due to pressure forces, and another viscous stress tensor $\boldsymbol{\sigma}'$ dependent on deformation rates, scaled with corresponding viscosity coefficients. This deformation term can be further divided into a hydrostatic term named the bulk viscous stress proportional to the velocity divergence, representing the forces resisting an increase of volume, and another symmetric tensor called the shear viscous stress which is proportional to the strain rate tensor as shown in Eq. 2.6, representing the forces opposing a shear deformation:

$$\sigma_{\alpha\beta} = -p\delta_{\alpha\beta} + \sigma'_{\alpha\beta} = -p\delta_{\alpha\beta} + \lambda\delta_{\alpha\beta} \frac{\partial u_\gamma}{\partial x_\gamma} + 2\mu S_{\alpha\beta} , \quad (2.5)$$

where p denotes the pressure, $\delta_{\alpha\beta}$ is the Kronecker delta, μ is the dynamic viscosity, $\lambda = \mu_B - 2\mu/3$ is a coefficient of viscosity, μ_B is called the bulk viscosity, and where $S_{\alpha\beta}$ represents the strain rate tensor as

$$S_{\alpha\beta} = \frac{1}{2} \left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha} \right) . \quad (2.6)$$

In this thesis, incompressible Newtonian fluids are considered, for which $\partial\rho/\partial t = \partial\rho/\partial x_\alpha = 0$. Thus Eq. 2.3 and Eq. 2.4 become

$$\nabla \cdot \mathbf{u} = 0 , \quad (2.7)$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} , \quad (2.8)$$

where $\nu = \mu/\rho$ is the kinematic viscosity. These equations form the so-called incompressible Navier-Stokes (NS) equations.

2.1.2. Dimensional Scaling

Fluid systems vary in their magnitude of physical quantities, for example length and free stream velocity. However, the law of similarity shows that different cases present the same physics provided that they share some dimensionless numbers. According to the Buckingham π theorem, physical equations involving n variables can be equivalently rewritten in terms of $(n - k)$ dimensionless parameters, where k is the rank of the dimensional matrix. This procedure, called non-dimensionalisation, reduces the number of free parameters and gives insights of fundamental properties of the system.

The Reynolds Number

Non-dimensionalising the NS equation in Eq. 2.8 with dimensionless variables of length l , velocity $\mathbf{u}$, time t , divergence operator ∇ , pressure p :

$$\begin{aligned} l^* &= \frac{l}{l_0} , \\ \mathbf{u}^* &= \frac{\mathbf{u}}{|\mathbf{u}_0|} , \\ t^* &= \frac{t}{l_0/|\mathbf{u}_0|} , \\ \nabla^* &= l_0 \nabla , \\ p^* &= \frac{p}{\rho |\mathbf{u}_0|^2} , \end{aligned}$$

where L_0 and U_0 are the characteristic length and velocity, it becomes

$$\frac{\partial \mathbf{u}^*}{\partial t^*} + (\mathbf{u}^* \cdot \nabla^*) \mathbf{u}^* = -\nabla^* p^* + \frac{1}{Re} \nabla^{*2} \mathbf{u}^* . \quad (2.9)$$

where Re is a dimensionless number called the Reynolds number Re

$$Re = \frac{|\mathbf{u}_0| l_0}{\nu} . \quad (2.10)$$

Re represents the ratio of the inertial forces that drive the fluid advection, to the viscous forces that induce momentum diffusion. For the incompressible Newtonian fluids, the non-dimensionalised Navier-Stokes equation depend only on Re .

A small Re indicates that the fluid is dominated by shear forces and is characterised by smooth flows. Flows at this regime are called laminar, as the fluid moves following parallel layers with little or no mixing.

Once the Re goes beyond a critical value, the advective forces become dominant and the viscous forces are insufficient to damp out excessive kinetic energy. This flow, which is characterised by chaotic eddies and other flow instabilities, is called turbulent flow. Due to its irregularity and complexity, deterministic prediction of turbulence is extremely costly, as will be shown in the following. Insight on the nature of turbulence can be gained using statistical approaches under the assumptions of ergodic theory [16, 17, 18]. First, the instantaneous variable, such as velocity $\mathbf{u}$, is decomposed into a mean and a fluctuating component:

$$\mathbf{u} = \langle \mathbf{u} \rangle + \mathbf{u}' , \quad (2.11)$$

where $\langle \mathbf{u} \rangle$ denotes the ensemble average of $\mathbf{u}$, and the fluctuation component is defined such that

$$\langle \mathbf{u}' \rangle = 0 . \quad (2.12)$$

Note that the ensemble average, representing the mean value over multiple realisations, is not necessary the temporal average. The relation

$$\langle \mathbf{u} \rangle = \frac{1}{T} \int_{-T/2}^{T/2} \mathbf{u} \, dt \quad (2.13)$$

only applies when the filtering timescale T is large compared to the fluctuation timescale so that $\mathbf{u}(t)$ and $\mathbf{u}(t+T)$ are uncorrelated, but is small not to smoothen the mean components.

This decomposition provides a way to describe turbulence and understand the energy transfer in turbulent flows. The turbulent kinetic energy k is defined as the mean kinetic energy related to the turbulent eddies and is characterised by the velocity fluctuations as

$$k = \frac{1}{2} \langle \mathbf{u}' \mathbf{u}' \rangle = \frac{1}{2} \left(\langle u_x'^2 \rangle + \langle u_y'^2 \rangle + \langle u_z'^2 \rangle \right) . \quad (2.14)$$

This energy needs to be transferred into small scales and dissipated into heat by viscous stress at the dissipation scale. This is known as the energy cascade shown in Figure 2.2. It illustrates that kinetic energy is extracted from the mean flow due to instability forces at high Re , and transferred to smaller and smaller scales by the breaking down of unstable large eddies into smaller eddies. The smaller eddies keep breaking down until they reach a size where the energy is

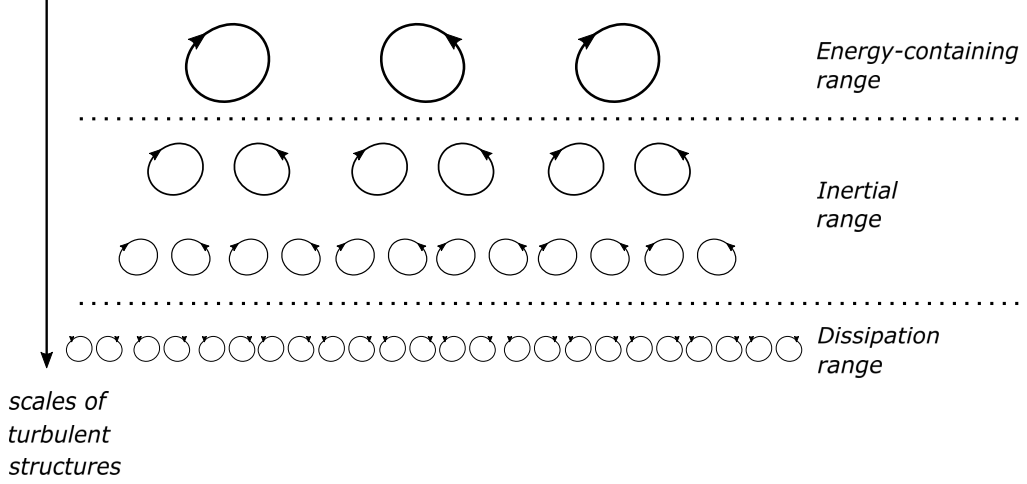


Figure 2.2.: Turbulent energy cascade

dissipated into heat due to molecular viscosity (dissipation range). This length scale characterising the size of the smallest turbulent eddies, is called the Kolmogorov scale η . On dimensional grounds, this scale is related to the energy dissipation rate ε and the viscosity ν as

$$\eta = \left(\frac{\nu^3}{\varepsilon} \right)^{1/4}. \quad (2.15)$$

In comparison, the large length scale that measures the largest turbulent eddies, is given by

$$l_0 = \frac{k^{3/2}}{\varepsilon}. \quad (2.16)$$

Fully resolving the turbulence with discrete numerical methods requires a grid spacing no larger than the Kolmogorov scale η :

$$\Delta x = \frac{l_0}{N} \leq \eta, \quad (2.17)$$

where N is the grid resolution representing the number of grid points of the characteristic length l_0 . According to Eq. 2.16, the energy dissipation rate ε scales like

$$\varepsilon \approx \frac{u_0^3}{l_0}, \quad (2.18)$$

where u_0 and l_0 are the characteristic velocity and length. Combine Eq. 2.15, Eq. 2.17 and Eq. 2.18, one can find out the minimum number of grid cells to

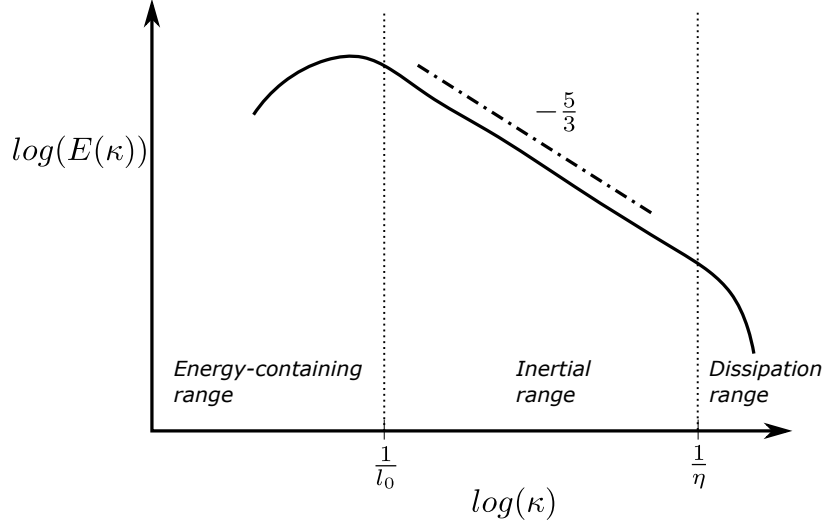


Figure 2.3.: Turbulent kinetic energy spectrum

fully resolve three-dimensional turbulence up to the Kolmogorov scale is

$$N^3 \geq Re^{9/4} . \quad (2.19)$$

This implies that the required computational costs grow rapidly when the Reynolds number increases, if neither grid refinement nor turbulence model is applied.

To better understand the turbulent energy distribution over the length scale, the turbulent kinetic energy spectrum is plotted in Figure 2.3. The spectrum characterises the distribution of the kinetic energy E with respect to the wave number κ , which is a measure of the eddy radius r , i.e. $\kappa \propto 1/r$. The total turbulent kinetic energy k can be obtained by integrating the kinetic energy of eddies over the wave number space

$$k = \int_0^\infty E(\kappa) d\kappa . \quad (2.20)$$

Spectral analysis of the turbulence scales reveals the relationship between the distribution of $E(\kappa)$ and κ in the inertial range:

$$E(\kappa) = C_\kappa \varepsilon^{2/3} \kappa^{-5/3} , \quad \frac{1}{l_0} \ll \kappa \ll \frac{1}{\eta} , \quad (2.21)$$

where C_κ is a constant. This equation is the well-known Kolmogorov spectrum law or the $-5/3$ law. It states that the energy spectra of a fully turbulent flow exhibit a $-5/3$ decay and is often used in experiments and simulations to verify

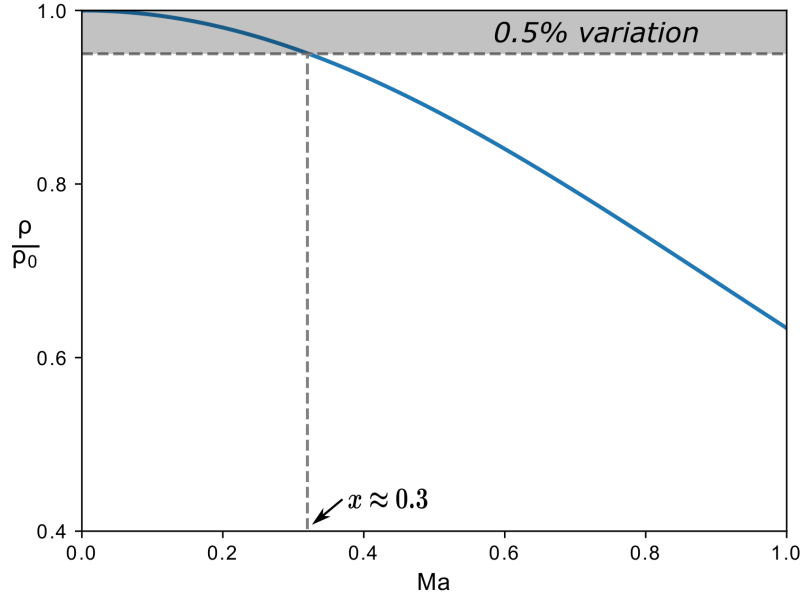


Figure 2.4.: Isentropic variation of density with Mach number

the turbulent statistics.

The Mach Number

Another dimensionless number that plays an important role in fluid mechanics is the Mach number:

$$Ma = \frac{u}{c_s}, \quad (2.22)$$

where c_s denotes the speed of sound in the fluid. The Mach number measures the flow speed in relation to the speed of acoustic wave propagation and determines the compressibility condition of the fluid. For $Ma \ll 1$, the speed of transmission of a small disturbance through the fluid, i.e. the speed of sound, is much larger than the characteristic flow speed, therefore the fluid density can be treated as constant.

In the region $0 < Ma < 1$, the compressibility of the fluid can be evaluated by the ratio between static density ρ and stagnation density ρ_{st} according to the isentropic flow relation

$$\frac{\rho}{\rho_{st}} = \left(1 + \frac{\gamma-1}{2} Ma^2\right)^{-\gamma/(\gamma-1)}, \quad (2.23)$$

where γ is the heat capacity ratio. For air at standard conditions, $\gamma = 1.4$ and

the static to total density ratio can be plotted against Ma as shown in Figure 2.4. Flows at $Ma < 0.3$, of which the variation between static density to total density is less than 5 percent, can be treated as incompressible.

The Knudsen Number

The Knudsen number is an important dimensionless number indicating whether the continuum fluid mechanics or the particle collision dynamics assumptions should be applied. It is the ratio of the molecular mean free path length l_{mfp} to the macroscopic characteristic length l_0 by definition, and is also related to the ratio of the Mach and Reynolds number:

$$Kn = \frac{l_{mfp}}{l_0} = \alpha \frac{Ma}{Re} , \quad (2.24)$$

where α is a constant. If $Kn \ll 1$, which means the mean free path is much smaller than the length scale of interest, it can be assumed that the fluid behaves as a continuous medium. This is known as the continuum hypothesis, and as a consequence, the NS equations apply. On the other hand, if $Kn \geq 1$, a statistical mechanics model that describes the particle collision dynamics should be used. Although all the problems in this thesis fall within the former case, it will be shown later in Section 2.4 that the Knudsen number plays an important role in the Chapman-Enskog analysis [19], which is used to derive the NS equations from the Boltzmann equation.

2.2. Computational Fluid Dynamics

In general, an exact solution of a partial differential equation system coupled with suitable boundary conditions, such as the NS equations, will not be available. In their stead, approximate solutions can be obtained by numerical modelling, leading to computational fluid dynamics (CFD). CFD methods can be categorised according to the modified equations they are solving, and the discretisation schemes they use. Methods based on the NS equations model the macroscopic aerodynamics, while particle-based methods model the kinetic behaviour of fluid particles.

Historically, Navier-Stokes based methods have been dominant for the vast majority of applications and will hence be reviewed here briefly to put the lattice Boltzmann method in a wider context and highlight the advantages and disadvantages of various methods.

2.2.1. NS Based Methods

There are several discretisation methods of the NS equations. The simplest one is the finite difference (FD) method, where the derivatives in the governing equations, i.e. the NS equations, are replaced by their finite difference approximations, evaluated at the nodes resulting from discretising a domain using structured grids. A structured grid is such that there is a regular connectivity between elements, that is, every inner domain element is topologically equivalent with the same number of vertices, faces and edges. Despite of the simple algorithm and straightforward implementation, it is not often used in practice due to conservation issue and simple geometry restriction [20].

In order to address these problems inherent in FD methods, a different class of methods was proposed based on discretisation of the equations using the weak form, the finite element (FE) methods, of which the finite volume (FV) method can be considered a particular case. In spite of this, historically FV methods applied to fluid dynamics have matured earlier than FE ones because of the intuitive interpretation that it has as a discretisation of the integral form equations.

Going into further detail, in order to write a FV scheme, any divergence terms are converted to surface fluxes via the divergence theorem as shown in Eq. 2.2. Then, a general form of such a scheme can be written as

$$\oint_S \rho \phi \mathbf{u} d\mathbf{A} = \int_{V_0} Q dV + \oint_S D \nabla \phi d\mathbf{A} , \quad (2.25)$$

where Q is a source term and D is a diffusion coefficient for a general quantity ϕ . The control volume formulation allows for a better capture of complex geometries and local refinement, since this scheme is suitable for structured but irregular and unstructured grids. Optimal grid generation and optimisation is itself still an intensive area of research, with recent developments including adaptive grid refinement [21, 22, 23], by which resolution is automatically increased where accuracy or stability criteria dictate.

FE methods, as mentioned, rely on the weak form of the governing partial differential equations as

$$R_i = \int_{V_0} W_i Q dV , \quad (2.26)$$

where R_i is the equation residual, W_i is the weighting function and Q is the conservation equation. The discretisation is completed by an approximation scheme, by which variable $\phi(x)$ at any arbitrary location can be interpolated

from nodal values ϕ_i using a space of basis functions $N_i(x)$, such that

$$\phi(x) \approx \sum_i \phi_i N_i(x) , \text{ for } \{N_i\} \text{ such that } \sum_i N_i(x) = 1 . \quad (2.27)$$

The FE discretisation is reported to be more stable compared to the FV approach [24]. One disadvantage of this method is that it is not designed to be conservative. It is also more complex and computationally expensive than the FD and FV methods [25]. Implementation in unstructured grids is possible, but challenging, as the weighting and basis functions need to be defined for every possible element topology.

2.2.2. Particle-Based Methods

In addition to conventional numerical methods such as FV, FD and FE that aim at solving the macroscopic equations of fluid mechanics, there exists another class of models that describe particle kinematics and interactions. The definition of particle is open enough that it allows the fluid to be represented by atoms, molecules, or statistical distributions of molecules. These particle dynamics are not directly representing the macroscopic aerodynamics, so extra procedure is required to apply these methods to CFD problems.

The lattice gas automata (LGA) is one such method that is able to recover the macroscopic NS equations [1, 26]. Gas molecules are expressed by fictitious particles on a square lattice or hexagonal lattice. Depending on the lattice layout, each particle has four or six unit velocities $\mathbf{c}_i$ pointing to its neighbouring nodes. The state of the particle at velocity $\mathbf{c}_i$ on lattice node $\mathbf{x}$ and time t is represented by a Boolean value $n_i(\mathbf{x}, t)$, with 1 and 0 denoting that such particle exist or not respectively.

Molecular collision and streaming of real gases are modelled in a similar way on these particles. The collision process updates the particle status to n_i^* with a collision operator $\Omega_i(\mathbf{x}, t)$ as

$$n_i^*(\mathbf{x}, t) = n_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t) . \quad (2.28)$$

The collision operator has a value of $\Omega_i \in \{-1, 0, 1\}$ and meets the conservation of mass $\sum_i \Omega_i(\mathbf{x}, t) = 0$ and the conservation of momentum $\sum_i \mathbf{c}_i \Omega_i(\mathbf{x}, t) = 0$. The collision process can be seen as a redistribution of particle velocities of a local node, while the total number and overall momentum of all particles are not changed. The streaming process is performed after the collision, propagating the

updated status of a node to its neighbouring ones, and reads

$$n_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = n_i^*(\mathbf{x}, t) . \quad (2.29)$$

Although the LGA method was originally intended for fluid simulations, its results face statistical noises like in real gases even when a lattice gas reaches equilibrium [26]. Another problem with the LGA is its insufficiency in dealing with three-dimensional problems [27]. The lattice Boltzmann method (LBM) was proposed to overcome these disadvantages by using the mesoscopic distribution function of particles instead of the Boolean lattice variables, as will be investigated in the following section.

2.3. The Lattice Boltzmann Method

This section introduces the theory of the lattice Boltzmann method. First, kinetic theory of gases and the Boltzmann equation will be reviewed as it forms the basis of LBM. The discretisation of the Boltzmann equation in both spatial and velocity space will be covered next.

2.3.1. Kinetic Theory

Kinetic theory links the molecular model to macroscopic fluid properties such as density, velocity, and temperature. In this section, the particle distribution function is introduced first, which represents the density of mass in both physical space and velocity space. The concept of velocity space will be shed light upon. Then, the continuous Boltzmann equation will be derived. Finally, the discretisation method for the Boltzmann equation, also both in coordinate and velocity space will be detailed.

The Particle Distribution Function

On the microscopic level, a small region in space contains numerous fluid particles, moving at various velocities. This can be statistically described using a probability distribution function as sketched in Figure 2.5, where $f(\mathbf{x}, \boldsymbol{\xi}, t)$ counts the number of particles with velocity $\boldsymbol{\xi}$ at a physical region denoted by coordinate $\mathbf{x}$ at a given time t .

Similar to the three-dimensional physical space $\mathbf{x} \in \mathbb{R}^3$, the velocity space can be constructed in three dimensions as $\boldsymbol{\xi} \in \mathbb{R}^3$. The macroscopic fluid properties can then be represented from the moments of the distribution function taken in

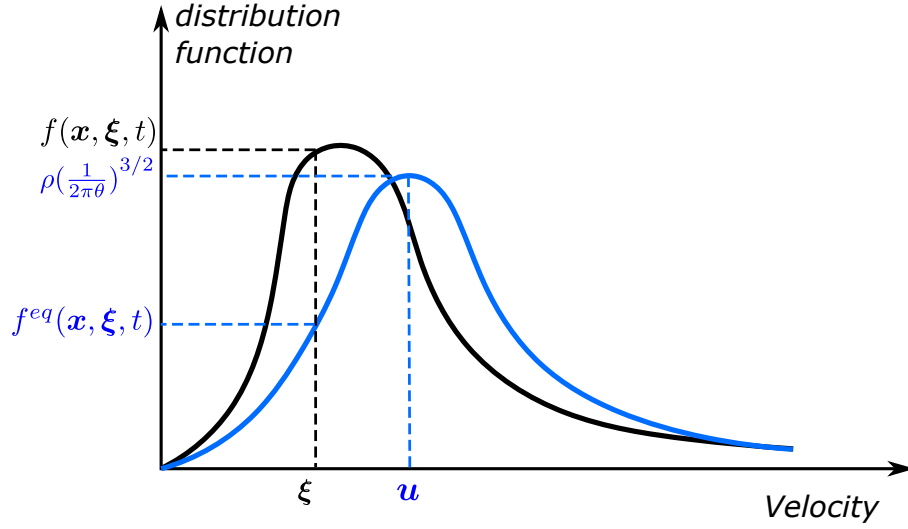


Figure 2.5.: A sketch of the distribution function against the velocity. The distribution function and its equilibrium are denoted in black and blue colour respectively.

velocity space. Density is the first moment:

$$\rho(\mathbf{x}, t) = \int f(\mathbf{x}, \boldsymbol{\xi}, t) d^3 \boldsymbol{\xi} . \quad (2.30)$$

The momentum is similarly given by the second moment:

$$\rho(\mathbf{x}, t) \mathbf{u}(\mathbf{x}, t) = \int \boldsymbol{\xi} f(\mathbf{x}, \boldsymbol{\xi}, t) d^3 \boldsymbol{\xi} , \quad (2.31)$$

where $\mathbf{u}$ is the macroscopic fluid velocity. The kinetic energy density is the third moment

$$\rho(\mathbf{x}, t) E(\mathbf{x}, t) = \frac{1}{2} \int |\boldsymbol{\xi}|^2 f(\mathbf{x}, \boldsymbol{\xi}, t) d^3 \boldsymbol{\xi} , \quad (2.32)$$

where E denotes the kinetic energy of macroscopic fluid. According to the equipartition theorem of classical statistical mechanics, the internal energy density e equals to $RT/2$ for each degree of freedom, i.e., for a 3D problem it reads $e = 3RT/2$, where R is the specific gas constant and T represents the temperature. Therefore the pressure can also be expressed as a moment of the distribution function using the ideal gas law $p = \rho RT$:

$$p = \frac{2}{3} \rho e = \frac{1}{3} \int |\mathbf{u} - \boldsymbol{\xi}|^2 f(\mathbf{x}, \boldsymbol{\xi}, t) d^3 \boldsymbol{\xi} . \quad (2.33)$$

When the fluid is at thermal equilibrium, the particle velocity distribution is given by the equilibrium distribution function $f^{eq}(\mathbf{x}, \boldsymbol{\xi}, t)$ as sketched in Fig-

ure 2.5. Non-dimensionalising $\mathbf{x}$, $\mathbf{u}$, $\boldsymbol{\xi}$ and ρ using the characteristic values, as we did in Section 2.1.2, one explicit form of the equilibrium distribution function is the Maxwell-Boltzmann distribution [28], which is written as

$$f^{eq}(\mathbf{x}^*, \mathbf{u}^*, \theta, \boldsymbol{\xi}^*) = \rho^* \left(\frac{1}{2\pi\theta} \right)^{3/2} \exp\left(-\frac{|\mathbf{u}^* - \boldsymbol{\xi}^*|^2}{2\theta}\right), \quad (2.34)$$

where $\theta = RT/V^2$ is the non-dimensional temperature. Hereafter we will omit the $*$ symbol for compact notations. It should be stated that the moments of f^{eq} are the same as those of f , e.g.,

$$\rho = \int f d^3\boldsymbol{\xi} = \int f^{eq} d^3\boldsymbol{\xi}, \quad (2.35)$$

$$\rho \mathbf{u} = \int \boldsymbol{\xi} f d^3\boldsymbol{\xi} = \int \boldsymbol{\xi} f^{eq} d^3\boldsymbol{\xi}. \quad (2.36)$$

The Boltzmann Equation

The Boltzmann equation describes the evolution of the distribution function f by taking its total derivative with respect to time t , which is

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \xi_\alpha \frac{\partial f}{\partial x_\alpha} + \frac{F_\alpha}{\rho} \frac{\partial f}{\partial \xi_\alpha} = \Omega(f), \quad (2.37)$$

where F_α denotes α component of the body force and $\Omega(f)$ is called the collision operator, which represents the redistribution of particles due to particle collisions. Instead of the original complicated two-body collision term by Boltzmann [29], a simpler form widely used in the LBM is the Bhatnagar–Gross–Krook (BGK) collision operator [30]:

$$\Omega(f) = -\frac{1}{\tau}(f - f^{eq}). \quad (2.38)$$

The BGK operator directly expresses the collision process as a relaxation towards equilibrium state. The speed of this process is controlled by the relaxation time τ , which is related to the fluid viscosity and will be discussed in the next section. The conservation of mass, momentum, and energy are automatically satisfied for the BGK operator since f and f^{eq} share equal moments as shown in Eq. 2.35 and Eq. 2.36.

2.3.2. Discretisation of the Boltzmann Equation

Discretisation in Velocity Space

Numerically solving the Boltzmann equation requires discretisation of not only time and space, but also the velocity. Applying the N -th-order Hermite series expansion, the d -dimensional continuous distribution function can be approximated by a series of Hermite polynomials $\mathbf{H}^{(n)}(\boldsymbol{\xi})$ with respect to velocity $\boldsymbol{\xi}$:

$$f(\mathbf{x}, \boldsymbol{\xi}, t) \approx \omega(\boldsymbol{\xi}) \sum_{n=0}^N \frac{1}{n!} \mathbf{a}^{(n)}(\mathbf{x}, t) \cdot \mathbf{H}^{(n)}(\boldsymbol{\xi}) , \quad (2.39)$$

where $\omega(x)$ is the weight function

$$\omega(x) = \frac{1}{(2\pi)^{d/2}} e^{-x^2/2} , \quad (2.40)$$

$\mathbf{a}^{(n)}$ is the expansion coefficient

$$\mathbf{a}^{(n)}(\mathbf{x}, t) = \int f(\mathbf{x}, \boldsymbol{\xi}, t) \mathbf{H}^{(n)}(\boldsymbol{\xi}) d^d \boldsymbol{\xi} . \quad (2.41)$$

Applying the Gauss-Hermite quadrature rule to Eq. 2.41, the continuous velocity space can be discretised into q discrete velocities $\boldsymbol{\xi}_i$

$$\mathbf{a}^{(n)}(\mathbf{x}, t) = \int f(\mathbf{x}, \boldsymbol{\xi}, t) \mathbf{H}^{(n)}(\boldsymbol{\xi}) d^d \boldsymbol{\xi} \approx \sum_{i=1}^q \frac{w_i}{\omega(\boldsymbol{\xi}_i)} f(\mathbf{x}, \boldsymbol{\xi}_i, t) \mathbf{H}^{(n)}(\boldsymbol{\xi}_i) , \quad (2.42)$$

where w_i denotes the discrete weight. Introducing a new particle velocity $\mathbf{c}_i$

$$\mathbf{c}_i = \frac{\boldsymbol{\xi}_i}{\sqrt{3}} , \quad (2.43)$$

for normalisation, the discrete distribution function can be obtained from Eq. 2.42 as

$$f_i(\mathbf{x}, t) = \frac{w_i}{\omega(\mathbf{c}_i)} f(\mathbf{x}, \mathbf{c}_i, t) . \quad (2.44)$$

The resulting force-free discrete-velocity Boltzmann equation is

$$\frac{\partial f_i}{\partial t} + c_{i\alpha} \frac{\partial f_i}{\partial x_\alpha} = \Omega(f_i) , \quad i = 1, 2, \dots, q . \quad (2.45)$$

The equilibrium distribution function can be discretised in the same way as in

Eq. 2.39. Applying the second-order Hermite series expansion, Eq. 2.34 becomes

$$f^{eq}(\mathbf{x}, \mathbf{u}, \theta, \boldsymbol{\xi}) \approx \omega(\boldsymbol{\xi})\rho [1 + \xi_\alpha u_\alpha + (u_\alpha u_\beta + (\theta - 1)\delta_{\alpha\beta})(\xi_\alpha \xi_\beta - \delta_{\alpha\beta})] . \quad (2.46)$$

Assuming isothermal flow for which $\theta = 1$, the temperature term can be removed. Replacing $\boldsymbol{\xi}$ by $\mathbf{c}$, the discrete equilibrium distribution can be written as the truncated form of the Maxwell-Boltzmann distribution

$$f_i^{eq} = w_i \rho \left(1 + \frac{c_{i\alpha} u_\alpha}{c_s^2} + \frac{Q_{i\alpha\beta} u_\alpha u_\beta}{2c_s^4} \right) , \quad (2.47)$$

where c_s is the speed of sound and $Q_{i\alpha\beta} = c_{i\alpha} c_{i\beta} - c_s^2 \delta_{\alpha\beta}$ is a symmetric velocity tensor. The speed of sound at constant entropy s is given by

$$c_s^2 = \left(\frac{\partial p}{\partial \rho} \right)_s = \gamma RT . \quad (2.48)$$

In the isothermal 3D LBM with constant temperature T_0 , the heat capacity ratio at isentropic condition becomes unity $\gamma = 1$, and the value of c_s is a constant given by

$$c_s = \sqrt{RT_0} . \quad (2.49)$$

The ideal gas law for isothermal condition then gives

$$p = \rho RT_0 = c_s^2 \rho . \quad (2.50)$$

The macroscopic fluid properties can now be calculated from the sum of the discrete distribution functions, i.e. the integrals in Eq. 2.35 and Eq. 2.36 become

$$\rho = \sum_i f_i = \sum_i f_i^{eq} , \quad (2.51)$$

$$\rho \mathbf{u} = \sum_i f_i \mathbf{c}_i = \sum_i f_i^{eq} \mathbf{c}_i . \quad (2.52)$$

The BGK operator in Eq. 2.38 can be written in a discrete form as

$$\Omega_i(f) = -\frac{1}{\tau} (f_i - f_i^{eq}) . \quad (2.53)$$

Here we introduce the non-equilibrium distribution function f_i^{neq}

$$f_i^{neq} = f_i - f_i^{eq} , \quad (2.54)$$

which represents the deviation of current distribution state to the equilibrium.

Table 2.1.: Velocity vectors and their weights of D2Q9

i	0	1	2	3	4	5	6	7	8
$\mathbf{c}_i$	(0,0)	(1,0)	(0,1)	(-1,0)	(0,-1)	(1,1)	(-1,1)	(-1,-1)	(1,-1)
w_i	$\frac{4}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$

In later discussions on the Chapman-Enskog analysis [19] in Section 2.4 we will see f_i^{neq} is linearly related to the local viscous stress state. The analysis will reveal the link between the LBE and NS equations.

According to Eq. 2.44, a velocity set can be defined based on discrete velocities $\mathbf{c}_i$ and corresponding weights w_i . There exists various velocity sets and they are usually named by $DdQq$, where d is the number of dimension of the physical space and q is the number of the discrete velocities in that set. In the construction of all these velocity sets mentioned below, the value of the speed of sound c_s is fixed as

$$c_s^2 = \frac{1}{3} \frac{\Delta x^2}{\Delta t^2}, \quad (2.55)$$

where Δx and Δt are discrete grid spacing and time step respectively. The discretisation of the space and time will be discussed in the following part of this section.

The most commonly used velocity sets are D2Q9 in 2D and D3Q19 in 3D, as illustrated in Figure 2.6, where each arrow represents a discrete velocity vector. Each discrete velocities is labelled in D2Q9 as an example, and these vectors $\mathbf{c}_i$ and their weights w_i are summarised in Table 2.1. Other popular choices include D3Q15 and D3Q27. Higher accuracy is achieved the more lattice velocities are used, at the cost of increased computational demands. Lack of sufficient discrete velocities, for example the D3Q13 lattice, may result in stability issues such as the well-known checkerboard problem [31].

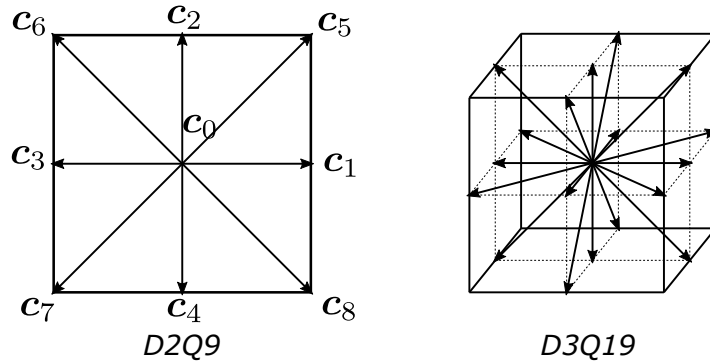


Figure 2.6.: D2Q9 and D3Q19 velocity sets

D3Q15, D3Q19, and D3Q27 are all able to recover aerodynamics to leading order as shown in Eq. 2.30 and Eq. 2.31, but their truncation errors take different functional forms, resulting in different performance when subject to numerical instabilities. Among them, D3Q15 is most efficient but may suffer from instabilities due to truncation errors [32]. D3Q27 is more expensive than the other two. For the remainder of this thesis, the D3Q19 velocity set will be assumed, and has been chosen as a compromise between accuracy and efficiency.

Discretisation in Space and Time

To discretise the Boltzmann equation in absence of body forces, Eq. 2.45 with discrete BGK operator in Eq. 2.53 is integrated over the time interval $[0, \Delta t]$, which returns

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) - f_i(\mathbf{x}, t) = \int_0^{\Delta t} \Omega_i(\mathbf{x} + \mathbf{c}_i \zeta, t + \zeta) d\zeta = -\frac{1}{\tau} \int_0^{\Delta t} [f_i(\mathbf{x} + \mathbf{c}_i \zeta, t + \zeta) - f_i^{eq}(\mathbf{x} + \mathbf{c}_i \zeta, t + \zeta)] d\zeta . \quad (2.56)$$

Applying the trapezoidal rule on the right-hand side of Eq. 2.56, it becomes

$$\begin{aligned} f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) - f_i(\mathbf{x}, t) = & \\ & -\frac{\Delta t}{2\tau} [f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) + f_i(\mathbf{x}, t)] \\ & + \frac{\Delta t}{2\tau} [f_i^{eq}(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) + f_i^{eq}(\mathbf{x}, t)] . \end{aligned} \quad (2.57)$$

Using the transformation

$$f_i + \frac{\Delta t}{2\tau} (f_i - f_i^{eq}) \rightarrow \bar{f}_i , \quad (2.58)$$

Eq. 2.57 can be rewritten as

$$\bar{f}_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) - \bar{f}_i(\mathbf{x}, t) = -\frac{1}{\bar{\tau}} (\bar{f}_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)) , \quad (2.59)$$

where $\bar{\tau} = \tau + \Delta t/2$ is the modified relaxation time [33]. Hereinafter f_i will be used in place of $\bar{f}_i$, as this transformation does not change the physical interpretation. The new variable returns the same moments as the original f_i , i.e. $\sum_i \bar{f}_i = \sum_i f_i = \rho$ and $\sum_i \mathbf{c}_i \bar{f}_i = \sum_i \mathbf{c}_i f_i = \rho \mathbf{u}$.

It can be seen that the second-order discretisation of the collision integral term is of the same form as the first-order discretisation in Eq. 2.59, which is written

as

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \Delta t \Omega_i(\mathbf{x}, t) . \quad (2.60)$$

Therefore this discretisation is second-order accurate although in a first-order form.

Eq. 2.60 is the final form of the discrete-velocity force-free Boltzmann equation, namely the lattice Boltzmann equation (LBE). From this equation it is clear that most computations take place in the collision operators, while the population propagation is simply an information exchange.

2.3.3. The Basic Collision and Streaming Algorithm

The computational algorithm of the temporal evolution is straightforward.

Substituting the discrete BGK operator in Eq. 2.53 into the lattice Boltzmann equation in Eq. 2.60 gives:

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) f_i(\mathbf{x}, t) + \frac{\Delta t}{\tau} f_i^{eq}(\mathbf{x}, t) . \quad (2.61)$$

The right hand side here is evaluated at the current time step, so it is natural to segregate Eq. 2.61 into two steps. The first one is known as collision step,

$$f_i^*(\mathbf{x}, t) = \left(1 - \frac{\Delta t}{\tau}\right) f_i(\mathbf{x}, t) + \frac{\Delta t}{\tau} f_i^{eq}(\mathbf{x}, t) , \quad (2.62)$$

where $f_i^*(\mathbf{x}, t)$ represents the post-collision distribution function on the local lattice node. Then the advection, referred to as streaming in the usual LBM

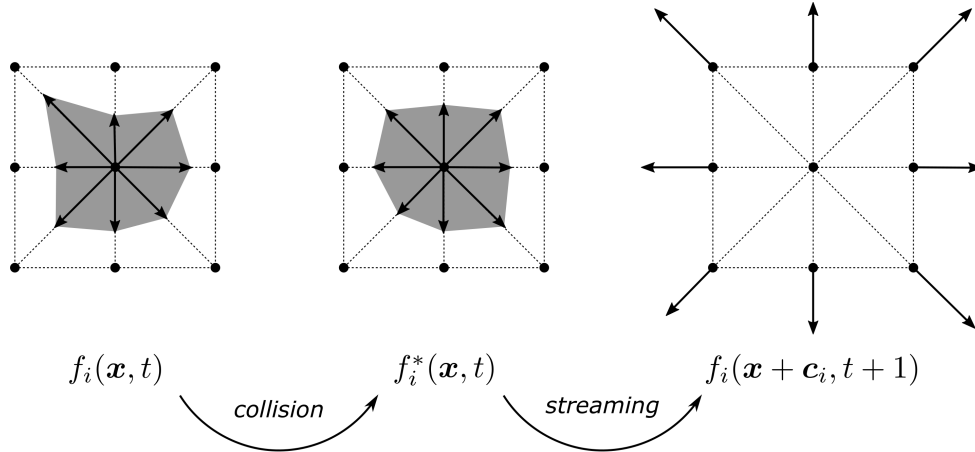


Figure 2.7.: Illustration of the collision and streaming process on a lattice

nomenclature, is performed as

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = f_i^*(\mathbf{x}, t) . \quad (2.63)$$

The schematic representation of the collision and streaming process is illustrated in Figure 2.7.

The overall computational algorithm of each time step can be summarised as in Algorithm 1.

Algorithm 1 General collision and streaming algorithm

```

while !converged do
   $\rho(\mathbf{x}, t), \mathbf{u}(\mathbf{x}, t) \leftarrow f_i(\mathbf{x}, t)$ 
   $f_i^{eq}(\mathbf{x}, t) \leftarrow \rho(\mathbf{x}, t), \mathbf{u}(\mathbf{x}, t)$ 
   $f_i^*(\mathbf{x}, t) \leftarrow f_i(\mathbf{x}, t), f_i^{eq}(\mathbf{x}, t)$ 
  if isBulkFluidNode then
     $f_i(\mathbf{x} + \mathbf{c}_i, t + 1) \leftarrow f_i^*(\mathbf{x}, t)$ 
  else if isBoundaryNode then
     $f_i(\mathbf{x} + \mathbf{c}_i, t + 1) \leftarrow f_i^*(\mathbf{x}, t)$ 
     $f_i^{missing}(\mathbf{x} + \mathbf{c}_i, t + 1) \leftarrow \text{boundary conditions}$ 
  end if
  Convergence criteria  $\leftarrow \rho(\mathbf{x}, t), \mathbf{u}(\mathbf{x}, t)$ 
  if Convergence criteria fulfilled then
    break
  end if
end while

```

2.4. Chapman-Enskog Analysis

To establish the link between the LBE in Eq. 2.60 and the NS equations in Eq. 2.8, the Chapman-Enskog analysis [19] is discussed in this section.

As mentioned in Section 2.1.2, Kn is the ratio of particle mean free path length l_{mfp} to the characteristic length l of the numerical model. For $Kn \rightarrow 0$, which means l_{mfp} is very small compared to l , the effect of molecular diffusion is negligible. The particle distribution function will therefore depend implicitly on the value of Knudsen number. Chapman-Enskog analysis is essentially a perturbation analysis of the particle distribution function in terms of the small parameter Knudsen number Kn .

A label ϵ is used to indicate the term's order in terms of Kn . The perturbation expansion therefore reads

$$f_i = f_i^{eq} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)} + \mathcal{O}(\epsilon^3) , \quad (2.64)$$

where the label ϵ^n only indicates the order Kn^n so that $\epsilon^n = \mathcal{O}(1)$ and the perturbation value exists in the high order terms as $f_i^{(n)}/f_i^{eq} = \mathcal{O}(Kn^n)$.

Under this circumstance, we can multiply Eq. 2.37 (in absence of body forces) by ξ_β , use the zeroth-order approximation $f_i = f_i^{eq}$, and then integrate over velocity space as

$$\frac{\partial}{\partial t} \int \xi_\beta f d^3\xi + \frac{\partial}{\partial x_\alpha} \int \xi_\alpha \xi_\beta f d^3\xi = 0 , \quad (2.65)$$

which returns the Euler momentum equation

$$\frac{\partial(\rho u_\beta)}{\partial t} + \frac{\partial(\rho u_\alpha u_\beta)}{\partial x_\alpha} = -\frac{\partial p}{\partial x_\alpha} . \quad (2.66)$$

We will see now retaining viscous effects arising from contribution of the perturbation terms leads to the NS equations. We start by using the first-order approximation $f_i = f_i^{eq} + \epsilon f_i^{(1)}$, and performing a Taylor expansion of the LBE in Eq. 2.60 in terms of Δt :

$$\Delta t(\partial_t + c_{i\alpha}\partial_\alpha)f_i + \frac{\Delta t^2}{2}(\partial_t + c_{i\alpha}\partial_\alpha)^2 f_i + \mathcal{O}(\Delta t^3) = -\frac{\Delta t}{\tau} f_i^{neq} . \quad (2.67)$$

Similarly, the temporal and spatial derivatives can be multiple-scale expanded in terms of Kn as

$$\partial_t = \epsilon \partial_t^{(1)} + \epsilon^2 \partial_t^{(2)} , \quad \partial_x = \epsilon \partial_x^{(1)} . \quad (2.68)$$

Substituting Eq. 2.64 and Eq. 2.68 into Eq. 2.67 and collecting the terms at two lowest order of Kn after some algebra, we get

$$\mathcal{O}(\epsilon^1) : \quad (\partial_t^{(1)} + c_{i\alpha}\partial_\alpha^{(1)})f_i^{eq} = -\frac{1}{\tau}f_i^{(1)} , \quad (2.69)$$

$$\mathcal{O}(\epsilon^2) : \quad \partial_t^{(2)}f_i^{eq} + (\partial_t^{(1)} + c_{i\alpha}\partial_\alpha^{(1)})\left(1 - \frac{1}{2\tau}\right)f_i^{(1)} = -\frac{1}{\tau}f_i^{(2)} . \quad (2.70)$$

Multiplying Eq. 2.69 with 1 and $c_{i\beta}$ respectively and summing over i give

$$\partial_t^{(1)}\left(\sum_i f_i^{eq}\right) + \partial_\alpha^{(1)}\left(\sum_i c_{i\alpha}f_i^{eq}\right) = -\frac{1}{\tau}\sum_i f_i^{(1)} , \quad (2.71)$$

$$\partial_t^{(1)}\left(\sum_i c_{i\beta}f_i^{eq}\right) + \partial_\alpha^{(1)}\left(\sum_i c_{i\alpha}c_{i\beta}f_i^{eq}\right) = -\frac{1}{\tau}\sum_i c_{i\beta}f_i^{(1)} . \quad (2.72)$$

Using Eq. 2.51 and Eq. 2.52 the above equations are equal to

$$\partial_t^{(1)}\rho + \partial_\alpha^{(1)}(\rho u_\alpha) = 0 , \quad (2.73)$$

$$\partial_t^{(1)}(\rho u_\alpha) + \partial_\beta^{(1)}(\rho c_s^2 \delta_{\alpha\beta} + \rho u_\alpha u_\beta) = 0 . \quad (2.74)$$

Substituting the explicit discrete form of f_i^{neq} in Eq. 2.47 into Eq. 2.69 and leaving out high-order terms, the explicit form of the first-order perturbation can be derived as

$$f_i^{(1)} = \left(\partial_t^{(1)} + c_{i\alpha} \partial_\alpha^{(1)} \right) \left[w_i \rho \left(1 + \frac{c_{i\alpha} u_\alpha}{c_s^2} + \frac{Q_{i\alpha\beta} u_\alpha u_\beta}{2c_s^4} \right) \right] , \quad (2.75)$$

which can be simplified using Eq. 2.73 and Eq. 2.74 as

$$f_i^{(1)} = -\frac{\rho \tau w_i}{c_s^2} \mathbf{Q}_i : \mathbf{S} , \quad (2.76)$$

where $\mathbf{S} = (\nabla \mathbf{u} + (\nabla \mathbf{u})^T)/2$ is the strain rate tensor.

Evaluating the first three moments of Eq. 2.69 and Eq. 2.70 and assembling the $\mathcal{O}(\epsilon)$ and $\mathcal{O}(\epsilon^2)$ equations, we can find

$$\left(\epsilon \partial_t^{(1)} + \epsilon^2 \partial_t^{(2)} \right) \rho + \epsilon \partial_\gamma^{(1)}(\rho u_\gamma) = 0 , \quad (2.77)$$

$$\left(\epsilon \partial_t^{(1)} + \epsilon^2 \partial_t^{(2)} \right) (\rho u_\alpha) + \epsilon \partial_\beta^{(1)} \Pi_{\alpha\beta}^{eq} = -\epsilon^2 \partial_\beta^{(1)} \left(1 - \frac{\Delta t}{2\tau} \right) \Pi_{\alpha\beta}^{(1)} . \quad (2.78)$$

The second moment of the equilibrium distribution function $\Pi_{\alpha\beta}^{eq}$ can be calculated using the explicit form of f_i^{eq} as

$$\Pi_{\alpha\beta}^{eq} = \sum_i c_{i\alpha} c_{i\beta} f_i^{eq} = \rho Q_{i\alpha\beta} . \quad (2.79)$$

Ignoring the high order terms, the second moment of the first-order perturbation $\Pi_{\alpha\beta}^{(1)}$ is proportional to the strain rate tensor $\mathbf{S}$:

$$\Pi_{\alpha\beta}^{(1)} = \sum_i c_{i\alpha} c_{i\beta} f_i^{(1)} = -2\rho c_s^2 \tau S_{\alpha\beta} + \mathcal{O}(u^3) . \quad (2.80)$$

Revisiting the derivative expansions in Eq. 2.68 and replacing $\Pi_{\alpha\beta}^{eq}$ and $\Pi_{\alpha\beta}^{(1)}$ with their explicit form in Eq. 2.79 and Eq. 2.80, the NS equations, namely the continuity equation in Eq. 2.3 and the momentum conservation equation in Eq. 2.4, can be obtained from Eq. 2.77 and Eq. 2.78, which now become

$$\partial_t \rho + \partial_\alpha (\rho u_\alpha) = 0 , \quad (2.81)$$

$$\partial_t (\rho u_\alpha) + \partial_\beta (\rho u_\alpha u_\beta) = -\partial_\alpha p + \partial_\beta \sigma'_{\alpha\beta} . \quad (2.82)$$

The resulting viscous stress tensor $\boldsymbol{\sigma}'$ has a form of

$$\sigma'_{\alpha\beta} = - \left(1 - \frac{\Delta t}{2\tau} \right) \Pi_{\alpha\beta}^{(1)} . \quad (2.83)$$

Substituting the explicit form of $\Pi_{\alpha\beta}^{(1)}$ into Eq. 2.78, we can find that the kinematic shear viscosity is linked with the relaxation time τ as

$$\nu = c_s^2 \left(\tau - \frac{\Delta t}{2} \right) \quad (2.84)$$

It should be noted that the above derivation ignores the high order error terms, e.g. $\mathcal{O}(u^3)$ in Eq. 2.80, which is only valid if $u^2 \ll c_s^2$ or $Ma^2 \ll 1$ [34]. This means the standard LBM is only applicable to simulations of the isothermal NS equations in the weakly compressible regime [35]. Work on adapting LBM schemes to compressible has been conducted in recent years, such as using extended velocity sets such as D2Q13 [36], D2Q37 [37], and even D3Q343 [38], or double distribution function (DDF) methods [39, 40].

The approximation $f_i^{neq} \approx \epsilon f_i^{(1)}$, which recovers the weakly compressible NS equations, results in a local Knudsen number Kn_l various on each computational cell and can be calculated from

$$Kn_l = \frac{1}{q} \sum_{i=0}^{q-1} \frac{f_i^{(1)}}{f_i} \approx \frac{1}{q} \sum_{i=0}^{q-1} \frac{f_i^{neq}}{f_i} . \quad (2.85)$$

Recalling the global Knudsen number Kn introduced in 2.1.2 which is determined by Ma and Re of the simulation, the local value Kn_l compared the local mean free path l_{mfp} to the characteristic length l_0 . Therefore the scaled Knudsen number Kn^* , defined as the ratio between local value Kn_l and the global one Kn

$$Kn^* = \frac{Kn_l}{Kn} , \quad (2.86)$$

indicates the scale of actual local grid sizing in comparison to the desired value.

It is possible to go on in this fashion, retaining further perturbation terms. For instance, the Burnett model [19] is obtained by retaining the second-order approximation $f_i = f_i^{eq} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)}$, which results in a more accurate model for predicting rarefied hypersonic flows [41].

2.5. Non-Dimensionalisation and Parameter Selection

2.5.1. Lattice Units

As explained in Section 2.1.2, a simulation does not necessary to have exactly same values of physical properties as the physical system. In LBM simulations dimensionless parameters are used to characterise the physical properties. According to the Buckingham π theorem, any kinematic properties can be represented in terms of three basis dimensions, which means its conversion factor can be determined from three basic conversion factors. In the LBM conversion factors C_l , C_t , and C_ρ of length, time, density respectively are typically chosen to non-dimensionalise the physical system. Conversion factors for other frequently used variables can therefore be represented using C_l , C_t , and C_ρ as summarised in Table 2.2.

To define an LBM simulation system, the Reynolds number, as well as the three conversion factors C_l , C_t and C_ρ , need to be determined. A computational grid for flows around a 2D cylinder is depicted in Figure 2.8. One can choose the cylinder diameter D as the characteristic length, and use the flow velocity u as the characteristic velocity u_0 , so that the conversion factors read

$$C_l = D , \quad C_u = u , \quad C_t = \frac{D}{u} . \quad (2.87)$$

Since the computational grid is uniform in the LBM, another popular strategy is to use the grid spacing Δx and the time spacing Δt as the characteristic length and time so that

$$C_l = \Delta x , \quad C_t = \Delta t . \quad (2.88)$$

The dimensionless properties using this reference are called in the *lattice units* and denoted with subscript *lb*. For example, the velocity in lattice units, named

Table 2.2.: Conversion factors for common physical variables in a LBM simulation

Physical variables	Physical units	Conversion factors
length	m	C_l
time	s	C_t
density	$kg \cdot m^{-3}$	C_ρ
velocity	$m \cdot s^{-1}$	$C_l C_t^{-1}$
pressure	$kg \cdot m^{-1} \cdot s^{-2}$	$C_\rho C_l^2 C_t^{-2}$
kinematic viscosity	$m^2 \cdot s^{-1}$	$C_l^2 C_t^{-1}$
force	$kg \cdot m \cdot s^{-2}$	$C_\rho C_l^4 C_t^{-2}$

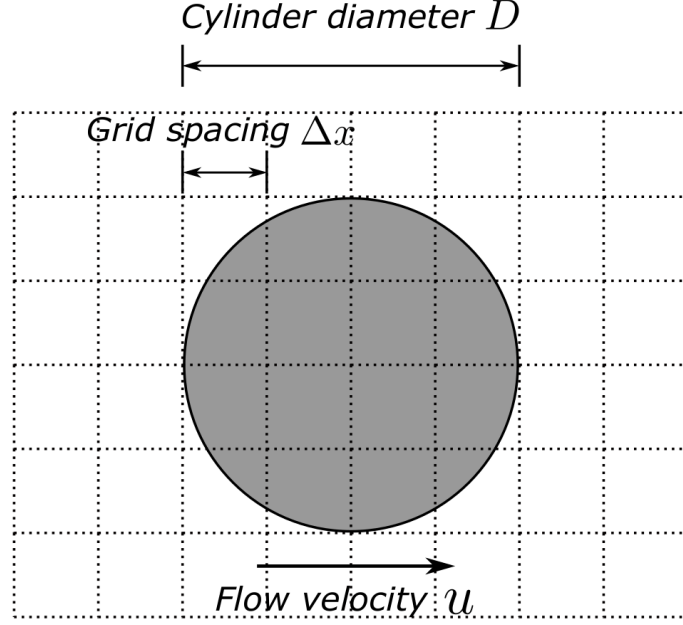


Figure 2.8.: Sketch of a computational grid around a 2D cylinder

as the lattice velocity u_{lb} , can be converted from physical velocity u as

$$u_{lb} = u \frac{C_t}{C_l} = u \frac{\Delta t}{\Delta x} . \quad (2.89)$$

The cylinder diameter is commonly described in terms of the physical characteristic length l_0 , and the grid spacing Δx is determined for a given resolution N based on l_0 and reads

$$\Delta x = \frac{l_0}{N} . \quad (2.90)$$

The lattice unit simplifies the streaming process with integer operations because the grid spacing and time spacing in lattice unit become unity:

$$\Delta x_{lb} = \frac{\Delta x}{C_l} = 1 , \quad \Delta t_{lb} = \frac{\Delta t}{C_t} = 1 . \quad (2.91)$$

Because the LBM is a weakly-compressible scheme, the density in lattice unit is also unity if using the fluid density as the characteristic value:

$$\rho_{lb} = \frac{\rho}{C_\rho} = 1 . \quad (2.92)$$

Note that the relaxation time τ has the dimension of time so its conversion factor is C_t . For simplification, τ in the following thesis are written in lattice

units if not specifically mentioned. Using the lattice unit, the discrete LBE in Eq. 2.61 then becomes

$$f_i(\mathbf{x} + \mathbf{c}_i, t + 1) = \left(1 - \frac{1}{\tau}\right) f_i(\mathbf{x}, t) + \frac{1}{\tau} f_i^{eq}(\mathbf{x}, t) . \quad (2.93)$$

Combining Eq. 2.55 and Eq. 2.84, the relation between the physical kinematic viscosity ν and the relaxation time τ is

$$\nu = \nu_{lb} \frac{C_l^2}{C_t} = \frac{1}{3} \left(\tau - \frac{1}{2} \right) \frac{\Delta x^2}{\Delta t} , \quad \tau > \frac{1}{2} , \quad (2.94)$$

while ν can be determined based on Re from Eq. 2.10.

Instead of directly specifying the value of Δt , a common approach in LBM simulations is to use the relaxation time τ or the lattice velocity u_{lb} as input parameter and implicitly calculate Δt as

$$\Delta t = C_t = \frac{\nu_{lb}}{\nu} C_l^2 = \frac{1}{3} \left(\tau - \frac{1}{2} \right) \frac{\Delta x^2}{\nu} , \quad (2.95)$$

or

$$\Delta t = C_t = \frac{u_{lb}}{u} C_l = \frac{u_{lb}}{u} \Delta x . \quad (2.96)$$

Using τ or u_{lb} is also more intuitive in evaluating the stability and accuracy conditions, as will be shown later in Section 2.5.2.

2.5.2. Stability and Accuracy Analysis

Stability

A general stability requirement in CFD is the Courant–Friedrichs–Lewy (CFL) condition [42], which states the Courant number C should satisfy:

$$C = u_{lb} \leq C_{max} , \quad (2.97)$$

where $u_{lb} = u\Delta t/\Delta x$ is the velocity in lattice units, and the value of C_{max} depends on the numerical methods used. This means that the physical convection speed u should be smaller than the information propagation speed $C_{max}\Delta x/\Delta t$ in the numerical model, so that the physical problem can be sufficiently resolved.

An additional sufficient stability condition for the BGK collision operator is that the equilibrium populations should be non-negative [43, 44]. Since f_i^{eq} is directly determined by u as shown in Eq. 2.47, an equivalent stability condition is that the lattice velocity should be no larger than the maximum achievable

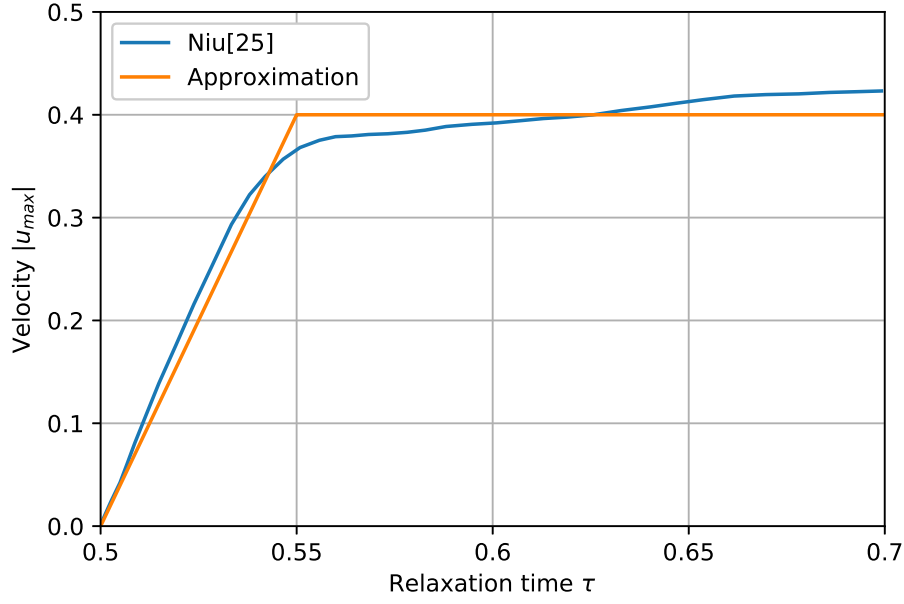


Figure 2.9.: Stability $u - \tau$ curve of BGK collision operator

velocity magnitude u_{max} [43]. Investigations revealed that the maximum lattice velocity $u_{lb,max}$ is closely related to the relaxation time τ and such a relation for D2Q9 [45] as plotted in Figure 2.9 is approximated to be

$$\begin{cases} u_{lb,max} = 8 \left(\tau - \frac{1}{2} \right) , & for \quad 0.5 < \tau < 0.55 , \\ u_{lb,max} = 0.4 , & for \quad 0.55 \leq \tau . \end{cases} \quad (2.98)$$

From Eq. 2.98 the grid Reynolds number can be defined to evaluate current grid resolution by using the properties in lattice unit:

$$Re_g = \frac{u_{lb} * 1}{\nu_{lb}} . \quad (2.99)$$

Referring to the stability condition in Eq. 2.98 and the viscosity equation in Eq. 2.94, the maximum local grid Reynolds number for BGK is suggested to be

$$Re_g \leq \frac{u_{lb,max}}{\frac{1}{3}(\tau - \frac{1}{2})} = 24 , \quad for \quad 0.5 < \tau < 0.55 . \quad (2.100)$$

Note that the stability condition listed above stands as a guideline rather than a criterion. In practical the real stability can be affected by many factors, for example the relation in Eq. 2.98 is largely dependent on the boundary conditions [46].

Accuracy

Error analysis of the isothermal LBM scheme (see [47] for a detailed exposition) reveals contributions from three sources, a spatial contributions $\mathcal{O}(\Delta x^2)$, a time contribution $\mathcal{O}(\Delta t^2)$, and a compressibility error $\mathcal{O}(Ma^2) \propto \mathcal{O}((u/c_s)^2) \propto \mathcal{O}(\Delta t^2/\Delta x^2)$. For BGK collision operator, there exists a spatial truncation error scaling as $\mathcal{O}((\tau - 1/2)^2) \propto \mathcal{O}(\Delta t^2/\Delta x^4)$ (from Eq. 2.94). Therefore the total error term for a BGK solution becomes

$$E(\Delta x, \Delta t) \approx \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta t^2/\Delta x^2) + \mathcal{O}(\Delta t^2/\Delta x^4) . \quad (2.101)$$

The goal of parameter selection is to find the balancing of these error terms so that none of them dominates. From the error analysis equation Eq. 2.101 it is known only decreasing Δx while keeping the temporal spacing unchanged, the compressibility error and the BGK truncation error will become dominant.

When changing Δx and Δt simultaneously based on an existing solution, their scaling relation need to be determined, which becomes relevant for the grid refinement as will be shown in Section 4.1. There are two common approaches: the acoustic scaling ($\Delta t_1/\Delta t_2 \propto \Delta x_1/\Delta x_2$) and the diffusive scaling ($\Delta t_1/\Delta t_2 \propto (\Delta x_1/\Delta x_2)^2$). In the acoustic scaling the change rate of Δt scales as the change of Δx , i.e.

$$\frac{\Delta t_1}{\Delta t_2} \propto \frac{\Delta x_1}{\Delta x_2} , \quad (2.102)$$

while the diffusive scaling is

$$\frac{\Delta t_1}{\Delta t_2} \propto \frac{\Delta x_1^2}{\Delta x_2^2} . \quad (2.103)$$

When refining the spatial resolution, for example, by a factor of n , the diffusive scaling requires the time step to be refined by a factor of n^2 . The diffusive scaling therefore aims to act on the compressibility error $\mathcal{O}(\Delta t^2/\Delta x^2)$, which stays unchanged in the acoustic scaling. However, the diffusive scaling requires more computational resources as the time step is finer. The acoustic scaling, on the other hand, maintains the correct scaling of the speed of sound and is suitable for potential field dominated problems.

None of the scaling methods can reduce the BGK truncation error efficiently. Instead of using an even finer time step (which further increases the computational demands), an alternative is to use other viscosity-independent collision operators such as the multiple relaxation time (MRT). Some advanced collision operators will be introduced in the following chapter.

2.6. Boundary Conditions in the LBM

Boundary conditions are essential to specifically solve the governing partial differential equations of fluid aerodynamics, in both continuum fluid dynamics and discrete numerical methods. In CFD the formulation of boundary conditions varies depending on the numerical scheme. The boundary conditions implemented in the LBM differ with standard NS based solvers, in that the particle populations are specified instead of macroscopic variables such as velocity and pressure. Here the formulation of boundary conditions used in this thesis are briefly introduced. Interested readers can refer to textbooks of standard NS solvers [48, 49] and of the LBM [35] for more explanation.

2.6.1. Simple Bounce-Back Method

The simple bounce-back condition is the standard methodology for imposing no-slip wall boundary conditions. A sketch of such a situation is depicted in Figure 2.10, where the boundary wall is placed midway between lattice nodes.

The pre-streaming populations with velocity $\mathbf{c}_6$, $\mathbf{c}_2$, $\mathbf{c}_5$ are bounced back to their opposite direction by the solid wall so that the updated populations have velocity $\mathbf{c}_8$, $\mathbf{c}_4$, $\mathbf{c}_7$ respectively. Note that the particles are reflected back rather than reflected forward so that there is no transverse movement between fluid and wall. The streaming step on bounce-back boundary node of resting walls can now be represented by

$$f_{i-opp}(\mathbf{x}_b, t + \Delta t) = f_i^*(\mathbf{x}_b, t) , \quad (2.104)$$

where the index $i - opp$ is on the opposite direction of index i so that the lattice velocities follow $\mathbf{c}_{i-opp} = -\mathbf{c}_i$.

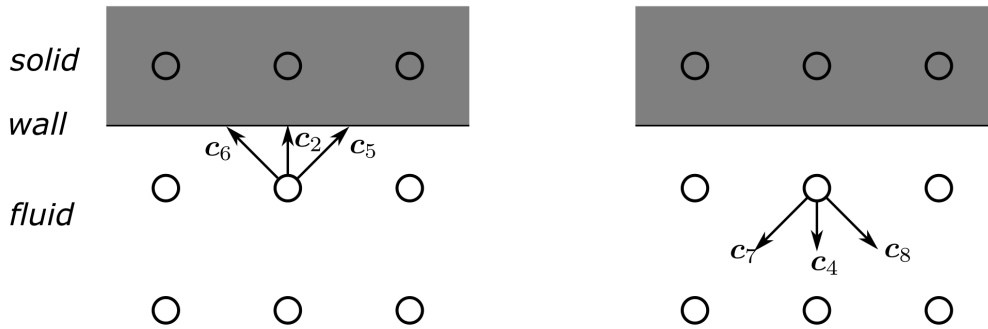


Figure 2.10.: Illustrations of particles before and after bounce back to a solid wall

For moving walls at a constant velocity, the particles bounced-back need to be corrected from the momentum change after hitting the wall to respect the Galilean invariance [35, 50], and reads

$$f_{i-opp}(\mathbf{x}_b, t + \Delta t) = f_i^*(\mathbf{x}_b, t) - 2w_i\rho_w \frac{\mathbf{c}_i \cdot \mathbf{u}_w}{c_s^2} , \quad (2.105)$$

where the $\mathbf{u}_w$ is the wall velocity, w_i and $\mathbf{c}_i$ are weight function and particle velocity of the population. The subscript w denotes the wall position $\mathbf{x}_w = \mathbf{x}_b + \mathbf{c}_i\Delta t/2$.

The bounce-back condition is a stable numerical scheme [51], with mass conservation automatically fulfilled by the reflection feature, but is only first-order accurate [51, 52] and is therefore not suitable for curved boundaries. In this work the bounce-back condition is only used on straight walls.

2.6.2. Non-Equilibrium Bounce-Back Method

Open boundaries including inlets and outlets are used to represent the boundaries where the flow enters or leaves the domain. The bounce-back condition can be adapted to open boundaries as well, for example the velocity inlet can be treated as a moving wall condition. However, the aerodynamic solution is viscosity-dependent when the bounce-back condition is used together with the BGK operator [51].

Another approach is the wet-node method, in which the boundary node is assumed to lie infinitesimally close to the actual boundary, but still in the fluid domain. Therefore the boundary condition is directly imposed on these boundary nodes, while collision and streaming in standard LB step are also performed on them.

Take D2Q9 as an example, as illustrated in Figure 2.11, some populations (i.e. f_1, f_5, f_8) are missing on the boundary nodes after the previous streaming. These unknown populations need to be re-constructed based on boundary conditions to support the next collision step. Typically only physical velocity or pressure is defined on the open boundary. Since the pressure and density are linearly related as from Eq. 2.50, the relation between boundary velocity u_b and density ρ_b need to be established via particle populations, i.e. from equations

$$\rho_b = \sum_i f_i , \quad (2.106)$$

$$\rho_b u_{b,n} = \sum_i c_{i,n} f_i = c(f_1 + f_5 + f_8) - c(f_3 + f_6 + f_7) , \quad (2.107)$$

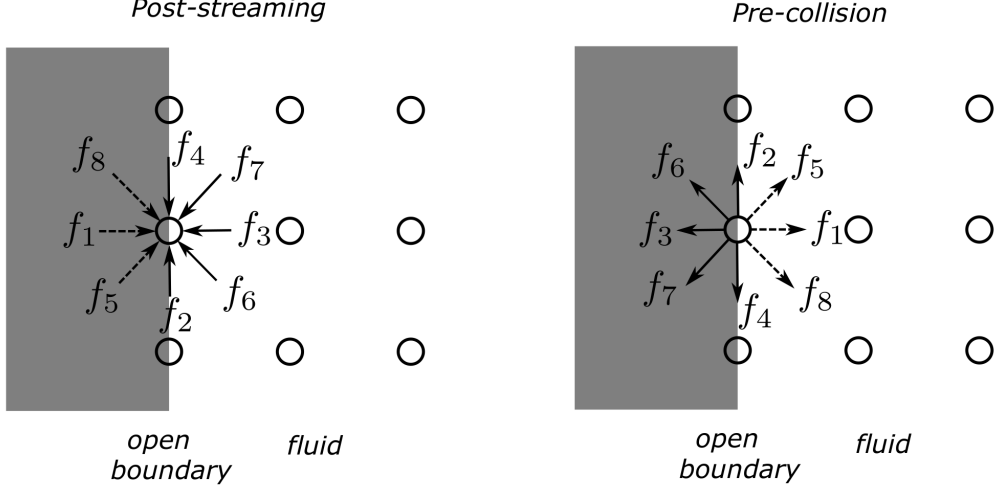


Figure 2.11.: Illustrations of wet-node method for open boundaries. The missing populations are marked in dotted lines.

where $u_{b,n}$ denotes the velocity component in the normal direction pointing into the fluid, and $c = u_{lb,n} = 1$ is the normal component of the unit lattice velocity. Combining Eq. 2.106 and Eq. 2.107 to cancel the unknown populations, the density and velocity can be related from known populations:

$$\rho_b = \frac{1}{1 - u_{lb,n}} [f_0 + f_2 + f_4 + 2(f_3 + f_6 + f_7)] . \quad (2.108)$$

One popular approach to calculate these unknown populations is the non-equilibrium bounce-back (NEBB) method (also named as Zou-He method) [53]. The NEBB method enforces the non-equilibrium term f_i^{neq} to be reflected back with a momentum correction term:

$$f_{i-opp}^{neq}(\mathbf{x}_b, t + \Delta t) = f_i^{neq}(\mathbf{x}_b, t + \Delta t) - \frac{\mathbf{n}_t \cdot \mathbf{c}_i}{|\mathbf{c}_i|} N_t , \quad (2.109)$$

where $\mathbf{n}_t$ is a unit vector tangent to the wall, N_t is the transverse momentum correction. By sorting relation between the known and unknown populations, and applying the boundary conditions, N_t can be calculated to construct the new populations. The NEBB scheme is of third-order accuracy for no-slip condition on straight boundaries, and is selected in this thesis for velocity inlet and pressure outlet.

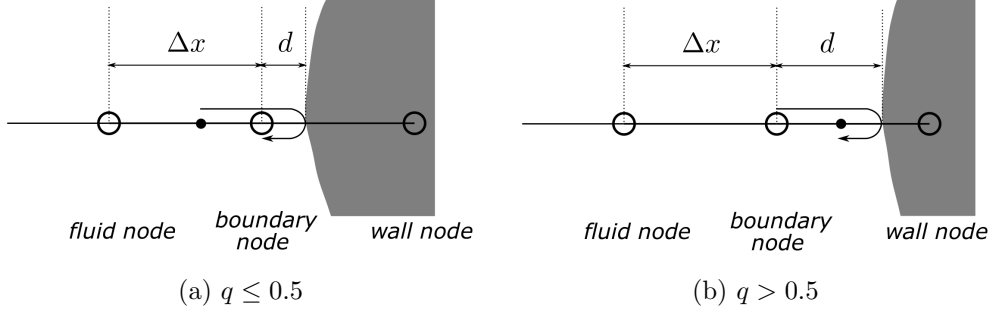


Figure 2.12.: Illustrations of the interpolated bounce-back method

2.6.3. Interpolated Bounce-Back Method

For curved boundaries, the computational node on the regular grid may not coincide with the exact boundary location. Using the simple bounce-back method on complex geometries will result in a *staircase* shape of grid will introduce geometrical discretisation errors.

The interpolated bounce-back method proposed by Bouzidi [54] improves the accuracy order on arbitrary boundary geometry to second-order by linearly interpolating the populations according to the distance to the wall.

As illustrated in Figure 2.12, two schemes can be categorised according to the distance of the wall to the boundary node. The parameter q is introduced as an indicator:

$$q = \frac{d}{\Delta x}, \quad q \in (0, 1), \quad (2.110)$$

where d denotes the distance and Δx is the grid spacing. For $q = 0.5$ this method returns to the simple bounce-back method, otherwise the bounce-back populations do not situate on grid nodes. Therefore the interpolation method is applied and for a resting wall it reads

$$f_{i-opp}(\mathbf{x}_b, t + \Delta t) = \begin{cases} 2q f_i^*(\mathbf{x}_b, t) + (1 - 2q) f_i^*(\mathbf{x}_b - \mathbf{c}_i \Delta t, t), & q \leq 0.5 \\ \frac{1}{2q} f_i^*(\mathbf{x}_b, t) + \frac{2q - 1}{2q} f_{i-opp}^*(\mathbf{x}_b, t), & q > 0.5 \end{cases}. \quad (2.111)$$

This boundary condition will be used in the simulation case with a circular cylinder geometry in Section 6.1.

2.6.4. Periodicity Condition

Periodicity conditions can be applied when the set up and solution exhibit a repeating pattern. They are also useful in the situation that physical scales of

interest are much smaller in one direction than in other. Then the large scale direction can be set as periodic, which is common in turbulence research [35]. The implementation of periodic condition is straightforward in the LBM, by letting the ghost cells at $\mathbf{x}_b$ pointing to the fluid cells on the other side at $\mathbf{x}_{b^*}$:

$$f_i(\mathbf{x}_b, t) = f_i(\mathbf{x}_{b^*}, t) . \quad (2.112)$$

2.6.5. Free-Slip Condition

Unlike the no-slip stationary wall which sets both normal and tangential component of the fluid velocity to zero, the free-slip condition enforces normal fluid velocity to zero only. In the LBM the free-slip condition can be implemented in the same way as a symmetry boundary condition, by replacing the streaming step as

$$f_{i-sym}(\mathbf{x}_b + \mathbf{c}_{i,t}\Delta t, t + \Delta t) = f_i^*(\mathbf{x}_b, t) , \quad (2.113)$$

where the subscript $i-sym$ represents the symmetric direction of index i so that the tangential velocity follows $\mathbf{c}_{i-sym,t} = \mathbf{c}_{i,t}$ and the normal velocity follows $\mathbf{c}_{i-sym,n} = -\mathbf{c}_{i,n}$.

2.7. Summary

In this chapter, the fluid dynamics models and the numerical method used in this thesis were briefly introduced, providing a theoretical background for understanding the need for grid refinement, its formulation and the implementation.

The governing equations of fluid dynamics, i.e. the Navier-Stokes equations, were introduced first, followed by some useful dimensionless numbers including Re, Ma, and Kn. Next the theory of the LBM was explained, and its stability and accuracy were discussed based on the Chapman-Enskog analysis.

One shortcoming of the LBM is its default uniform Cartesian grid structure. Although this layout simplifies meshing, computation and communication, the required computational resources grow considerably as the overall grid resolution increases for high Reynolds number simulations. Grid refinement is necessary for the LBM to take full advantage of its parallel efficiency.

3. State of the Art of High Reynolds LBM

As stated in the last chapter, resolving turbulent flows at high Reynolds number requires a grid spacing approximately equal to the Kolmogorov scale, which is expensive for the uniform grid layout adopted in the LBM. Another issue at high Reynolds number is that the BGK collision operator is known as τ -sensitive, i.e. its numerical stability and accuracy depend on the relaxation time τ [35, 55].

Considering the number of the discrete velocities used in the D3Q19 lattice, the incompressible isothermal fluid dynamics, which contain 10 variables in 3D, i.e. 1 in density, 3 in velocity, and 6 in stress tensor (as mentioned in Section 2.1.1), as pointed out by Latt et al. [56], are actually over-determined. In other words, it is possible to generate a different set of these 19 discrete populations, that leads to the same 10 aerodynamic variables. This is because of the higher order contributions from f_i in the BGK operator, which produce numerical errors in symmetrical physical properties such as the shear viscous stress tensor and harm the robustness of the method at low viscosity [57]. Specifically, the $\mathcal{O}(u^3)$ error term in Eq. 2.80 when linking the perturbation moment $\Pi_{\alpha\beta}^{(1)}$ to the strain rate tensor $\mathbf{S}$ indicates that the Galilean invariance cannot be sufficiently guaranteed by the second-order Hermite series expansion.

To improve the numerical stability and accuracy of the LBM at high Reynolds numbers, advanced collision operators have been developed. These modifications of the relaxation process can be seen as effectively introducing additional viscosity on local cells, but with more degrees of freedom since all q discrete populations can be modified. This chapter reviews some existing approaches to improve the LBM to simulate high Reynolds flows, including advanced collision operators, non-equilibrium population regularisation, subgrid modelling, and the grid refinement method.

3.1. The Multiple-Relaxation-Time Operator

The multiple-relaxation-time (MRT) collision operator is designed to overcome the accuracy and stability problems of the BGK operator. While the BGK models use a single relaxation time, the MRT operator is constructed with more free parameters to alter the relaxation process.

The significant difference between MRT and BGK is that for MRT the particle collisions are performed in moment space. This means that the moments are relaxed during the collision process instead of the populations. The q populations are linearly transformed into q moments:

$$m_k = \sum_{i=0}^{q-1} M_{ki} f_i \quad \text{for } k = 0, \dots, q-1, \quad (3.1)$$

where $\mathbf{M}$ is a 2D transformation matrix. By properly constructing $\mathbf{M}$, the moments m_k can be mapped to corresponding aerodynamics moments such as density, momentum, momentum flux tensor, etc.

This transformation provides a way to directly relax those aerodynamics terms by assigning individual relaxation parameter to each moment. Those relaxation parameters form the diagonal relaxation matrix

$$\mathbf{S} = \begin{pmatrix} \omega_0 & 0 & \cdots & 0 \\ 0 & \omega_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \omega_{q-1} \end{pmatrix}, \quad (3.2)$$

where $\omega = 1/\tau$ is the relaxation parameter.

The collision equation in Eq. 2.62 can then be written as

$$\begin{aligned} \mathbf{f}^*(\mathbf{x}, t) - \mathbf{f}(\mathbf{x}, t) &= -\mathbf{M}^{-1} \mathbf{M} \frac{1}{\tau} [\mathbf{f}(\mathbf{x}, t) - \mathbf{f}^{eq}(\mathbf{x}, t)] \\ &= -\mathbf{M}^{-1} \frac{1}{\tau} [\mathbf{M} \mathbf{f}(\mathbf{x}, t) - \mathbf{M} \mathbf{f}^{eq}(\mathbf{x}, t)] \\ &= -\mathbf{M}^{-1} \mathbf{S} [\mathbf{m}(\mathbf{x}, t) - \mathbf{m}^{eq}(\mathbf{x}, t)], \end{aligned} \quad (3.3)$$

where $\mathbf{M}^{-1}$ is the inverse matrix of $\mathbf{M}$. The streaming process is unchanged and still performed on the population space.

There are various ways to construct the transformation matrix $\mathbf{M}$. The construction is out of the scope of this thesis and here we take a brief glance at relaxation matrix structure. Interested readers are referred to [58, 35] for more information. One widely used approach is the Gram-Schmidt procedure which

results in a relaxation matrix of D2Q9, for example, in the form of

$$\mathbf{S} = \text{diag} (0, \omega_e, \omega_e, 0, \omega_q, 0, \omega_q, \omega_\nu, \omega_\nu) , \quad (3.4)$$

where the zero relaxation parameters indicates the relaxation of conserved moments of density and momentum, ω_e and ω_ν are related to bulk and shear viscosity, while ω_e and ω_q are free parameters to modify.

The MRT operator significantly improves the stability at high Reynolds number, and a number of applications have been investigated [59, 60, 61] at Reynolds number up to 1×10^6 without using turbulence models. However, a comparison study by Nathen et al. [62] showed that the MRT scheme could has convergence issue on grids of a DNS resolution, which indicates these parameters should be carefully selected.

The MRT model is more general and advanced, and on the other hand, more expensive compared to the BGK operator. The BGK operator can be seen as a specific case of the MRT model, in which all relaxation parameters are identical to each other. It should be noted that with the increased degree of freedom, these relaxation rate need to be carefully chosen empirically, as theoretical analysis is still lacking [58].

3.2. The Regularised BGK Operator

A regularised procedure on the particle populations is proposed [56] as the so-called regularised lattice Boltzmann (RLB) method to remove the higher order perturbations of f_i in Eq. 2.64. In other words, the f_i^{neq} is replaced by $f_i^{(1)}$ during this procedure, and the lattice Boltzmann equation with standard BGK operator

$$f_{i,reg}(\mathbf{x} + \mathbf{c}_i, t + 1) = f_i^{eq}(\mathbf{x}, t) + \left(1 - \frac{1}{\tau}\right) f_i^{neq}(\mathbf{x}, t) , \quad (3.5)$$

now becomes the regularised lattice Boltzmann equation

$$f_{i,reg}(\mathbf{x} + \mathbf{c}_i, t + 1) = f_i^{eq}(\mathbf{x}, t) + \left(1 - \frac{1}{\tau}\right) f_i^{(1)}(\mathbf{x}, t) . \quad (3.6)$$

Approximating $\Pi_{\alpha\beta}^{(1)} = \Pi_{\alpha\beta}^{(neq)}$, the value of $f_i^{(1)}$ can be explicitly computed from local populations by combining Eq. 2.76 and Eq. 2.80, which is

$$f_i^{(1)} = \frac{w_i}{2c_s^4} Q_{i\alpha\beta} \sum_k c_{k\alpha} c_{k\beta} f_k^{neq} . \quad (3.7)$$

The RLB scheme can also be equivalently treated as a type of MRT operator [63]. Since the MRT provides access to independent relaxation of high order moments, the regularised step can be seen as fully relaxing them to equilibrium. The RLB operator therefore improves numerical stability by eliminating the contribution of the spurious high order moments [56, 63]. However, this method is reported to introduce strong numerical dissipation compared to the BGK operator [62].

More recently a recursive regularised (RR) operator was introduced by Malaspinas [64]. This method is based on the BGK collision model but exhibited enhanced stability and accuracy compared to the MRT scheme. Compared to the standard RLB approach, which removes all higher order moments in Hermite polynomials, the RR operator preserves these effects via recursively computing the non-equilibrium Hermite coefficients. A hybrid recursive regularised (HRR) BGK was proposed by Jacob et al. [9] based on the hybridisation of the computation of the velocity gradients in the regularised non-equilibrium distribution functions. This approach provides dynamic control of the hyperviscosity term in the momentum equation at each grid point. Accurate results were observed when applying the HRR method to the flow around a circular cylinder at $Re = 3900$.

The principle of regularising the non-equilibrium distribution function instead of using its raw value can also be applied to population coupling between two computational cells. As will be show later in Section 4.1, this method is useful to increase numerical stability on refinement interfaces. It should be clarified that this regularisation procedure is independent of collision operators, such that the regularised coupling can be applied with any of them.

3.3. Subgrid Models in LBM

Large eddy simulation (LES) is a popular method for reducing the computational cost of turbulent simulations. According to the Kolmogorov theory of turbulence, eddies of small scales are universal and independent of the geometry. The LES method therefore resolves only large flow structures and models the effect of the small-scale eddies, as indicated in Figure 3.1. It filters out the small-scale eddies using the space filtering function:

$$\bar{\phi}(\mathbf{x}) = \int \phi(\mathbf{x})G(\mathbf{x} - \mathbf{x}')d\mathbf{x}' , \quad (3.8)$$

where ϕ represents a physical quantity, $\mathbf{x}'$ denotes the neighbouring position around $\mathbf{x}$, and G is a spatial filtering function, for example the rectangular

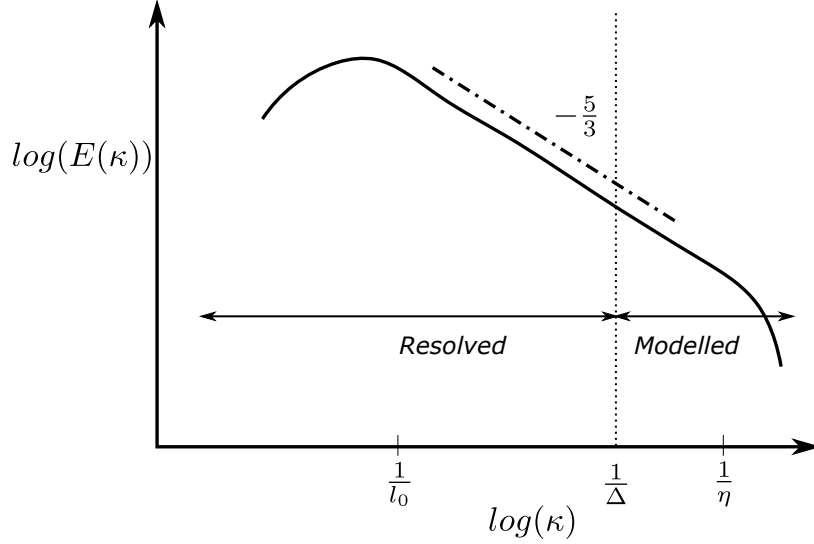


Figure 3.1.: LES length scales in the turbulent energy spectrum

window filter. This filtering operation provides a length scale $\Delta = 2|\mathbf{x} - \mathbf{x}'|$ greater than the Kolmogorov scale η as stated in Section 2.1.2, so that only eddies greater than specific scales are resolved and the small eddies are modelled as illustrated in Figure 3.1.

By applying the filtering function to the density, pressure and velocity, the resulting incompressible NS equations in 2.7 and 2.8 in Einstein notation become

$$\frac{\partial \bar{u}_\alpha}{\partial x_\alpha} = 0, \quad (3.9)$$

$$\frac{\partial \bar{u}_\alpha}{\partial t} + \bar{u}_\beta \frac{\partial \bar{u}_\alpha}{\partial x_\beta} = -\frac{1}{\bar{\rho}} \frac{\partial \bar{p}}{\partial x_\alpha} - \frac{\partial \tau_{\alpha\beta}}{\partial x_\beta} + 2 \frac{\partial \nu \bar{S}_{\alpha\beta}}{\partial x_\beta}, \quad (3.10)$$

where $\tau_{\alpha\beta} = \overline{u_\alpha u_\beta} - \bar{u}_\alpha \bar{u}_\beta$ is the Reynolds stress, $\bar{S}_{\alpha\beta}$ denotes the large scale strain rate tensor computed from the filtered velocity:

$$\bar{S}_{\alpha\beta} = \frac{1}{2} \left(\frac{\partial \bar{u}_\alpha}{\partial x_\beta} + \frac{\partial \bar{u}_\beta}{\partial x_\alpha} \right). \quad (3.11)$$

At high Reynolds number, the effect of the small eddies plays an important role on turbulent flows and the Reynolds stress must be modelled. The eddy-viscosity model is such an approach, which models the effect of the dissipation due to the exchange of momentum between small scale eddies via an artificial viscosity, i.e., the so called eddy viscosity ν_t , as

$$\tau_{\alpha\beta} - \frac{1}{3} \tau_{\gamma\gamma} \delta_{\alpha\beta} = -2\nu_t \bar{S}_{\alpha\beta}. \quad (3.12)$$

The well-known Smagorinsky-Lily model [65] expresses the eddy viscosity as

$$\nu_t = C_s \Delta^2 |\bar{\mathbf{S}}| , \quad (3.13)$$

where C_s denotes the Smagorinsky coefficient, Δ is the filter length equals to the grid spacing, and $|\bar{\mathbf{S}}| = \sqrt{2\bar{S}_{\alpha\beta}\bar{S}_{\alpha\beta}}$ is the magnitude of the strain rate tensor.

For the LBM implementation, the filtering operation in Eq. 3.8 is applied to the populations and the corresponding total viscosity becomes

$$\nu_{total} = \nu_0 + C_s \Delta^2 |\bar{\mathbf{S}}| , \quad (3.14)$$

which results in a new relaxation time as

$$\tau_{total} = \tau_0 + \frac{C_s \Delta^2}{c_s^2} |\bar{\mathbf{S}}| . \quad (3.15)$$

Note that the strain rate tensor can be locally calculated from f_i^{neq} via Eq. 2.80. This is an advantage of the LBM because these gradients are usually expensive to compute.

The accuracy of the LBE-LES formulation was assessed by Yu et al. [66] in evaluating the evolution and energy spectra of the decaying homogeneous isotropic turbulence at Taylor-microscale Reynolds number Re_λ of 53 and 119 against NS-LES results. It was observed that the LBE-LES captured instantaneous flow fields more precisely than the NS-LES of equal grid size with second-order accurate FV discretisation scheme. Comparing under-resolved simulations to resolved ones, large-scale flow behaviours were captured accurately in the LBE-LES in comparison with the LBE-DNS simulations.

There are other approaches to subgrid modeling. For example, instead of closing the LBE-LES equations from the eddy-viscosity model, Sagaut [67] extended the approximate deconvolution method (ADM), which was proposed by Stolz et al. [68] in an NS framework, into the LBM scope. This approach does not rely on the eddy-viscosity concept and is more general.

Because turbulence becomes anisotropic in the near-wall region, the Smagorinsky-Lily model is too dissipative to predict the turbulent flow near solid walls [69]. Premnath et al. [70] introduced a dynamic procedure in determining the local Smagorinsky constant. With the help of a refined grid, the coefficient is obtained from sampling of turbulence dynamics on the finest grid, assuming scale-invariance at two levels. The turbulent channel flow at friction Reynolds number Re_τ of 180 and 395, were investigated with finest dimensionless grid spacing of $\Delta^+ = 1.98$. They argued that this resolution can satisfy the near-

wall requirements of $< 1.5\eta^+ \sim 2\eta^+$, where $\eta^+ \approx 1.5 \sim 2.0$ is the Kolmogorov length scale in wall units. Weickert et al. [71] adjusted the wall adapting local eddy-viscosity (WALE) model [69] into the lattice Boltzmann framework. With a resolved grid spacing ($y^+ = 0.51$ for the finest grid resolution) in the near wall region, this method was shown to be promising in predicting the transition from laminar to turbulent regimes in turbulent channel flow at $Re_\tau = 395$.

A shear-improved Smagorinsky model for the LBM was proposed by L  v  que et al. [72] to account for the shear effects of the near wall turbulence without using a wall function or dynamic adjustment. Their approach modified the Smagorinsky eddy viscosity by subtracting the magnitude of the mean shear $|\langle \bar{\mathbf{S}} \rangle|$ from the magnitude of the strain-rate tensor $|\bar{\mathbf{S}}|$. This model was reported to reduce computational costs around the wall boundary, while it preserves a satisfactory accuracy in comparison with the dynamic Smagorinsky model and direct numerical simulation.

In addition to the wall-resolved LES methods, progress has been made with respect to wall-modelled LBM-LES methods in recent years. Extending the solution of the turbulence boundary layer equations to the LBM framework was pioneered by Malaspinas et al. [73]. Based on the MRT collision operator, satisfactory results can be obtained in plane channel flow with friction Reynolds number Re_τ up to 20,000 on a coarse uniform grid with $\Delta x^+ = \Delta y^+ = \Delta z^+ = 1000$.

While the MRT operator can be selected for its stability at high Reynolds number in studies of near wall turbulence, it is computationally expensive compared to the simple BGK operator. Developed from Malaspinas’s method, Haussmann et al. [74] therefore proposed a wall function for the BGK operator. In a fully developed turbulent channel flow with friction Reynolds number Re_τ of 1000, 2000, and 5200, they demonstrated that first and second-order turbulence statistics can be accurately predicted with under-resolved grids with dimensionless wall distance $y^+ > 12.5$.

3.4. Entropic Lattice Boltzmann Method

To improve the stability of LBM at high Reynolds number, the entropic lattice Boltzmann method (ELBM) has been proposed by Karlin [75]. Compared to the standard LBM, the ELBM restores the Boltzmann H theorem (i.e. the second law of thermodynamics), which is not satisfied during the discretisation step in

the standard LBM. A discrete entropy function is constructed as

$$H(f) = \sum_i f_i \ln\left(\frac{f_i}{w_i}\right) , \quad (3.16)$$

where $w_i > 0$ is the weight coefficient of the i -velocity. Instead of approximating from the Gauss-Hermite quadrature as in Eq. 2.47, the equilibrium distributions are obtained from minimisation of the entropy function.

The next step is to determine the time discretisation so that the total entropy does not increase with time steps. The discrete LBE in Eq. 2.93 is rewritten as

$$f_i(\mathbf{x} + \mathbf{c}_i, t + 1) = (1 - \beta) f_i(\mathbf{x}, t) + \beta f_i^{mirr}(\mathbf{x}, t) , \quad (3.17)$$

where $\alpha\beta = 1/\tau$ and α is the over-relaxation parameter, $f_i^{mirr}(\mathbf{x}, t)$ is the maximally over-relaxed mirror state defined as

$$f_i^{mirr}(\mathbf{x}, t) = f_i + \alpha(f_i^{eq} - f_i) . \quad (3.18)$$

The BGK operator can be recovered by setting $\alpha = 2$:

$$f_i^{mirr}(\mathbf{x}, t) = 2f_i^{eq} - f_i . \quad (3.19)$$

In the ELBM, α is determined by the positive root of the entropic condition

$$H(f + \alpha(f^{eq} - f)) = H(f) . \quad (3.20)$$

The ELBM is demonstrated to be unconditionally stable [76], at the price of increased computational cost due to the non-linear entropy equations. A 3D simulation using the ELBM on a turbulent vortex flow was conducted [2] and good agreement to results from a spectral element solver was observed, with a 40x speedup in the computational time using similar number of grid cells. This approach was also assessed on wall-bounded turbulence [77] and the mean velocity profile in the turbulent boundary layer was accurately captured on an under-resolved grid.

The entropic scheme can be adapted to the MRT collision operator and the so-called Karlin-Bösch-Chikatamarla (KBC) operator was proposed by Karlin et al. [78]. The KBC model further improves the numerical stability as demonstrated on a turbulent flow around a circular cylinder at $Re = 1.4 \times 10^5$. This approach was applied to the transitional flow over a low-Reynolds number airfoil at $Re = 60000$ and good agreements to NS based LES results were captured with the

help of a grid refinement method and an improved stability formulation for wall boundary conditions [79].

The ELBM is an effective but complicated method to simulate high Reynolds flow in the LBM scheme. However, even though this method reduces the minimum requirement for grid resolution, the automatic generated grid is still uniform. In consideration of numerical accuracy, a local fine grid resolution is often desired, which calls for the technique of local grid refinement.

3.5. Grid Refinement

Grid refinement is not new to the LBM. Early on, much effort has been devoted to adapting the LBE to a generalised form for arbitrary non-uniform grids. Another popular type of grid refinement method nowadays for the LBM is based on hierarchical grid layout, which is composed of a number of uniform grids at different grid resolutions. The simplicity and locality of the LBM are mostly retained on these uniform Cartesian grids. In the hierarchical layout, the coarse grid and the fine grid are overlapped to communicate with each other. Based on the relative overlapping position of each grid, the hierarchical grid refinement can be divided into two types, the vertex-centred grid refinement and the cell-centred one.

3.5.1. Non-Uniform Grid Layout Refinement

A finite volume LBE (FVLBE) was proposed by Nannelli [80] and Succi [81]. This method offers geometric flexibility but increases total computational costs because of the loss of locality. Another method using non-uniform grids is the interpolation-supplemented LBE (ISLBE) proposed by He et al. [82]. This method uses curvilinear coordinate non-uniform grids and interpolates the unknown post-collision populations from surrounding uniform grid nodes, thus still using the original discrete velocity sets in LBM. The interpolation, however, introduces extra numerical viscosity which is also observed in the FVLBE. The appearance of this artificial viscosity on the non-uniform grid can be explained by the fact that the lattice relaxation time is dependent on the grid resolution, as shown in Eq. 2.84, but a uniform relaxation time was used while no rescaling procedure is applied to the local relaxation time. However, on these irregular grid, the scaling algorithm can be quite complicated depending on different grid structures.

3.5.2. Vertex-Centred Hierarchical Refinement

As illustrated in Figure 3.2, the coarse grid and the fine grid are overlapped by one coarse grid spacing Δx_c , and grid nodes belonging to the coarse and fine grid coincide in this region. These two blocks compose the basic structure of a block cascade, which can be further extended by adding more refinement levels. The alignment simplifies the information exchange between the coarse and fine grid, which will also be referred to as “coupling”. Coupling can be categorised into two types: the coupling of coinciding nodes and the coupling between free fine nodes and neighbouring coarse nodes.

Coinciding nodes share the same physical properties and hence the coupling method simply involves scaling the relaxation time, the equilibrium, and non-equilibrium part of the distribution function between the two grids. For free fine nodes, which lie in between coarse nodes as illustrated in Figure 3.2, the macroscopic quantities such as density, velocity and shear stress must be interpolated from neighbouring coarse nodes and the associated new populations must be constructed. The scaling relation of quantities between the two grids will be described in detail in Section 4.1.

One of the first LBM grid refinement methods was proposed by Filippova and Hänel [52]. They established a scaling relation between the fine and coarse grid populations based on the Chapman-Enskog analysis [19]. For this, the distribution function f_i is split into two parts: the equilibrium part f_i^{eq} which is determined by the macroscopic fluid density and velocity, and the non-equilibrium distribution function f_i^{neq} which represents local fluctuations from equilibrium

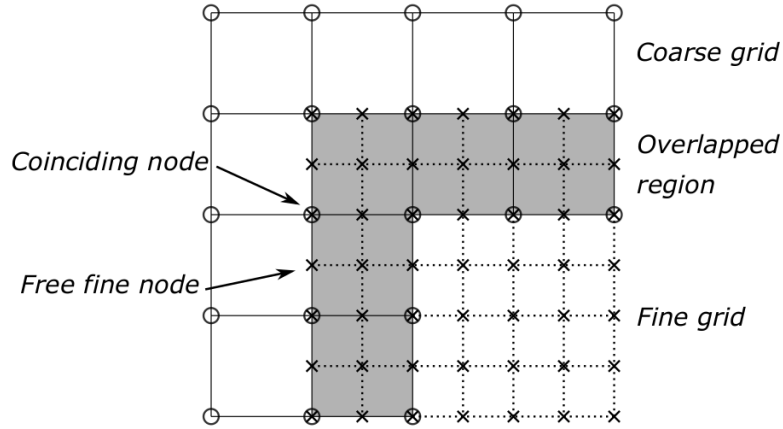


Figure 3.2.: Grid layout of the vertex-centred structure. Coarse nodes are marked as $\circ$ and fine nodes are $\times$

state. Since the particle populations can be recovered from physical properties, i.e., f_i^{eq} from density and velocity, f_i^{neq} from stress tensor, the coupling between the coarse and fine nodes can be established.

For constructing the unknown populations at free fine nodes, a second-order accurate interpolation approach was adopted. Their refinement scheme was validated on the 2D benchmark problem [83] of the steady and unsteady flow around a circular cylinder in a channel. The results were in good agreement with reference values and the CPU time was found to be comparable to that of incompressible finite-volume or finite-difference Navier-Stokes solvers.

Following this refinement scheme, an improved spatial interpolation method using a symmetric, cubic spline fitting was proposed by Yu et al. [84]. This was combined with a three-point Lagrange interpolation for temporal scaling. When compared to a uniform grid, they achieved a 39x speed up with a 5 block cascade domain.

Dupuis et al. [85] improved the stability of the refined grids by rescaling the post-streaming distribution functions instead of the post-collision populations. They demonstrated a significant speed up with a factor between 200 to 2000 of a refined grid over the coarse grid, in accelerating the flow settlement for a steady 2D Poiseuille flow.

Lagrava et al. [86] introduced a non-weighted low-pass filter aiming at removing abundant information on the fine grid that was not resolved by the coarse grid, in order to avoid numerical instability at relative high Reynolds numbers. This so-called restriction procedure was applied only to the non-equilibrium distribution functions to avoid an artificial increase of viscosity. They also demonstrated that a second-order interpolation was not able to conserve the mass in the reconstruction process and a local cubic interpolation scheme was mandatory to stabilise the scheme. This method was validated on two 2D benchmark cases: flow past a circular cylinder at $Re = 100$ and the dipole-wall collision process. In the latter case, perturbations were observed when Re reached 5000 when no filter technology was applied. They showed that introducing the filtering algorithm helped eliminate these perturbations.

3.5.3. Cell-Centred Hierarchical Refinement

A different grid layout as shown in Figure 3.3, namely the cell-centred, was used by Chen et al. [87] and Rohde et al. [88] in the volumetric scheme of grid refinement. The post-collision coarse node in the overlapped region are exploded equally onto 4 “ghost” fine nodes to supplement the collision and streaming on the rest of the fine grid. After two time steps on the fine grid, the 4 ghost nodes

are coalesced to form the coarse node, with updated incoming populations. The conservation of mass and momentum across the refinement interface are strictly satisfied in this volumetric description.

However, as the 4 ghost fine nodes all receive the same information from the coarse node, the non-equilibrium populations on these fine nodes are forced to have the same values and the local stress state is not correctly captured on the fine grid boundary. This results in a staggered solution as reported by Rohde [88] since no interpolation or rescaling methods were applied. Amor et al. [89] therefore proposed a second-order accurate interpolation method to calculate the f_i^{neq} of each ghost node from 4 surrounding coarse nodes, which was shown to smooth out the errors found in Rohde's approach.

3.5.4. Grid Refinement with Advanced Collision Operators

In recent years, more grid refinement approaches for the LBM emerged. Touil et al. [90] applied a grid refinement method to LBM-LES based on the D3Q19 lattice set. Both direct numerical simulation (DNS) and LES were carried out on a turbulent channel flow and validated against reference data from a high-resolution pseudo-spectral simulation. When comparing with a Navier-Stokes LES solver based on finite-volume discretisation, an improved accuracy in boundary layers was shown in the LBM-LES model.

Dorschner et al. [8] extended the vertex-centred grid refinement approach to entropic incompressible, thermal, and compressible 3D lattice Boltzmann

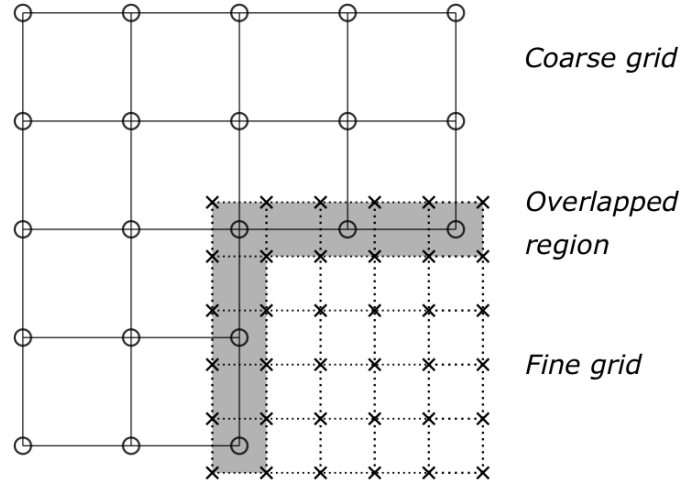


Figure 3.3.: Grid layout of the cell-centred structure. Coarse nodes are marked as $\circ$ and fine nodes are $\times$

models. Their approach was successfully validated on several turbulent benchmarks, including supersonic flow around NACA0012 airfoil with $Ma_\infty = 1.5$ and $Re = 10000$.

The RLB scheme [56], when applied together with refined grids, showed an improved performance in reducing spurious waves intrinsically generated from the refinement interface due to the change in grid resolution [5]. The regularised procedure increases the stability of the LBM by filtering out high order perturbations ($O(f_i^{(2)})$) in the non-equilibrium distribution function f_i^{neq} . The HRR method with a refined grid and a subgrid scale model yielded accurate results when applying to the flow around a circular cylinder at $Re = 3900$ [9], while further research [5] displayed its ability in dissipating the spurious waves in non-uniform grids.

3.6. Summary

The state of the art for high Reynolds number LB simulations was reviewed in this chapter. In these regimes, improvements in stability and accuracy can be achieved through advanced collision operators, non-equilibrium population regularisation and subgrid modeling. It is concluded that the Smagorinsky eddy viscosity subgrid scale model promises enhanced stability with the BGK collision operator, which is favourable because of the ease of implementation, and hence this will be used for the results reported in Chapter 7. The RLB collision operator applies a regularisation procedure on f_i^{neq} and is found to have an improved stability compared to the original BGK operator. The MRT and the ELBM are two powerful collision models. Although not used in this thesis, they provide perspectives for further research.

Although the methods above improve the stability at high Reynolds numbers, grid refinement is still necessary [9, 61, 70, 71, 79] to reduce total computational costs. Therefore this research will focus on the development and implementation of a grid refinement method which can be easily adapted to work with other advanced collision operators in future research.

Most of the existing grid refinement methods directly couple f_i^{neq} , which may create stability issues on the refinement interface at high speed turbulent flows [5]. Considering that the regularised procedure helps to reduce the spurious effects of high order moments in the non-equilibrium distribution functions, it is natural to use that method to couple populations at the refinement interface. Instead of the HRR operator [9], the standard regularised method is applied to the grid refinement method described in this thesis. This is because the

recursive regularisation procedure involves more computations for seeking higher numerical accuracy, which is expensive and not necessary on the refinement interface.

Based on this review, the vertex-centred scheme with regularised coupling was chosen and implemented in the widely-used open-source LBM code OpenLB [11]. Since most implementations of grid refinement methods are proprietary, it is hoped that this work will better serve the wider community providing with a sound base from which future research can be built. The objective is to extend the use of the solver, pushing current capability towards the high Reynolds number regime.

4. Grid Refinement Method with Regularised Coupling

This chapter describes the grid refinement methodology implemented and applied in the open-source solver OpenLB as part of this thesis. Starting by introducing the refinement grid layout, three components of the interface treatment are explained: the direct coupling, the interpolation and the restriction. Regularised coupling, which enhances the numerical stability of the refinement interface at high Reynolds number, is introduced afterwards. A corrected grid layout is proposed next, which aims to solve the dislocation issue of the periodic boundary condition when applied to a refined grid. The overall practical collision and streaming process, including the coarse coupling and the fine coupling, of a refined grid is summarised in the end.

4.1. Methodology

4.1.1. Grid Layout and Parameter Scaling

A vertex-centred refinement layout is chosen in this thesis. The interface between a coarse and fine grid is illustrated in Figure 4.1. The two grids overlap by one coarse grid spacing such that the boundary nodes of one grid coincide with interior fluid nodes of the other grid. This results in two interfaces, which are treated separately: a *fine coupling interface* at the boundary of the fine grid and a *coarse coupling interface* at the boundary of the coarse grid. On the fine coupling interface the coarse populations pass information to the fine ones, and vice versa on the coarse coupling interface.

The properties to be coupled on two grids include the spatial and temporal step sizing, the relaxation time and the distribution functions. Setting the grid spacing ratio n to 2, i.e.

$$n = \frac{\Delta x_c}{\Delta x_f} = 2 , \quad (4.1)$$

where the subscript c and f denote the properties of coarse grid and fine grid respectively, the ratio of the time step size need to be determined accordingly.

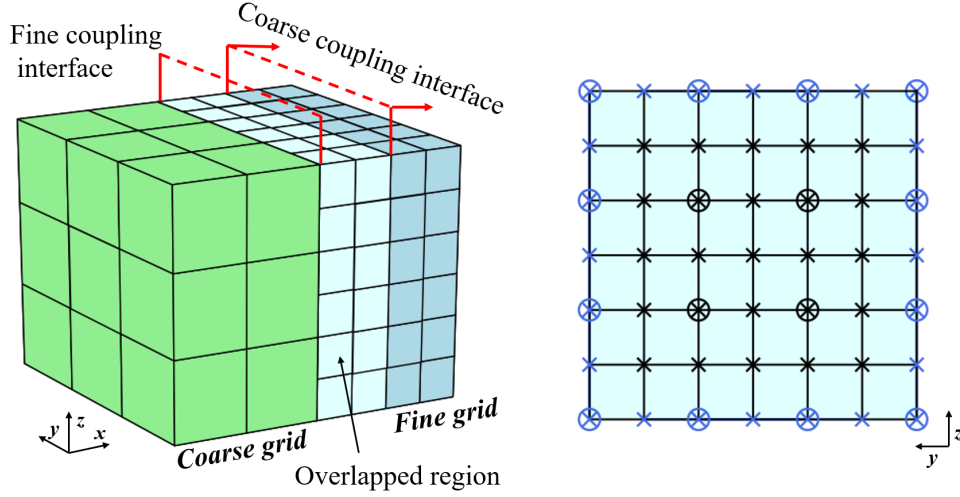


Figure 4.1.: General layout of a coarse and a fine grid. Coarse nodes are marked as $\circ$ and fine nodes are $\times$. Nodes marked in blue in the left figure are boundary nodes.

The acoustic scaling ($\Delta t \propto \Delta x$) introduced in Section 2.5.2 is applied to keep the same propagation speed of particles on the refinement interface. Hence the time step used on the fine grid is half that on the coarse grid, which reads

$$\frac{\Delta t_c}{\Delta t_f} = \frac{\Delta x_c}{\Delta x_f} = 2. \quad (4.2)$$

This means that for each coarse time step, two collision and streaming steps are performed on the fine grid nodes. A temporal interpolation on the coarse properties at time t and $(t + \Delta t_c)$ are thus used to calculate fluid properties of the intermediate state $(t + \Delta t_c/2)$.

Three different treatments are necessary to exchange population information at the interface, depending on the direction of information exchange (coarse to fine or fine to coarse) and relative node location. As illustrated in Figure 4.1, there are fine nodes which share a location with a coarse node and fine nodes which are offset, lying mid-way between coarse nodes (hereinafter named “free nodes”). The three treatments are summarised below and will be explained in the following sections:

1. **Direct coupling from coarse to fine:** For fine nodes coincident with coarse nodes, the coarse populations are rescaled and directly assigned to the fine nodes. To improve stability, a regularised coupling is applied.
2. **Interpolation from coarse to fine:** For free fine nodes, the physical

properties (density, velocity and viscous stress) are first interpolated from the surrounding coarse nodes and based on these, new local populations are calculated.

3. **Coupling from fine to coarse:** For coarse nodes coincident with fine nodes, the fine populations could be simply rescaled and directly transferred to the corresponding coarse nodes. However, to improve stability in the turbulent regime, a restriction process is needed to take advantage of information on neighbouring fine nodes. The regularised coupling can be used in place of conventional restriction method using simple averaging, since regularisation is also a correction to the non-equilibrium populations. This part will be explained in Section 4.1.4.

4.1.2. Direct Coupling

Transforming Eq. 2.62, the post-collision population f_i^* of the BGK collision operator for both refinement levels can be written in terms of the equilibrium distribution and the non-equilibrium part as

$$f_i^*(x, t) = f_i^{eq}(x, t) + \frac{\tau - 1}{\tau} f_i^{neq}(x, t) . \quad (4.3)$$

Since the equilibrium f_i^{eq} is determined only by local fluid density and velocity, the population scaling can be simplified into mapping of the relaxation time τ and the non-equilibrium distribution functions f_i^{neq} .

From Eq. 2.94, which is

$$\nu = \nu_{lb} \frac{C_l^2}{C_t} = \frac{1}{3} \left(\tau - \frac{1}{2} \right) \frac{\Delta x^2}{\Delta t} , \quad \tau > \frac{1}{2} , \quad (4.4)$$

the relation between the coarse relaxation time τ_c and the fine relaxation time τ_f can be written as

$$\nu = \frac{1}{3} \left(\tau_c - \frac{1}{2} \right) \frac{\Delta x_c^2}{\Delta t_c} = \frac{1}{3} \left(\tau_f - \frac{1}{2} \right) \frac{\Delta x_f^2}{\Delta t_f} . \quad (4.5)$$

Applying the coarse-to-fine ratio $n = \Delta x_c / \Delta x_f = \Delta t_c / \Delta t_f = 2$, the relationship between the relaxation time in each grid is

$$\tau_f = 2\tau_c - \frac{1}{2} . \quad (4.6)$$

The Chapman-Enskog analysis in Section 2.4 reveals that f_i^{neq} can be approximated by the first-order perturbation of f_i , which can be calculated from f_i^{eq}

as shown in Eq. 2.69, and reads

$$f_i^{neq} \approx f_i^{(1)} = -\tau(\partial_t^{(1)} + c_{i\alpha}\partial_\alpha^{(1)})f_i^{eq} , \quad (4.7)$$

Note that the operator $(\partial_t^{(1)} + c_{i\alpha}\partial_\alpha^{(1)})$ is based on the fine and coarse grid spacings respectively, and need to be rescaled based on the coarse-to-fine ratio as

$$\Delta t_c \partial_{t_c}^{(1)} + \Delta x_c c_{i\alpha} \partial_{\alpha_c}^{(1)} = \Delta t_f \partial_{t_f}^{(1)} + \Delta x_f c_{i\alpha} \partial_{\alpha_f}^{(1)} . \quad (4.8)$$

The scaling of $f_{i,c}^{neq}$ to $f_{i,f}^{neq}$ can therefore be written as

$$f_{i,c}^{neq} = \frac{\tau_c}{\tau_f} \frac{\Delta t_c}{\Delta t_f} f_{i,f}^{neq} = \frac{2\tau_c}{\tau_f} f_{i,f}^{neq} . \quad (4.9)$$

4.1.3. Interpolation and Restriction

Populations on free fine nodes cannot be directly coupled from coarse nodes but need to be re-constructed based on the local physical state. A fourth-order accurate interpolation method is adopted in this thesis instead of a second-order interpolation scheme. This is because the linear interpolation will result in first-order errors when summed up over the whole interface [86], but the LBM scheme is of second-order accurate globally as shown in Section 2.3.2. For the fourth-order bicubic interpolation, at least 4×4 neighbouring coarse grid nodes are needed for each fine cell.

The weight coefficients of interpolation at three different circumstances are illustrated in Figure 4.2, where the red cross marks the interpolation location and the blue circle denotes the boundary node. For fine nodes in line with the coarse grid, i.e. whose x- or y-location is equal to that of a coarse node, the bi-cubic interpolation is simplified to a 1D cubic interpolation because the weight function in the other dimension are always 1 locally and 0 elsewhere. The 1D interpolation for a fine node in a row of coarse nodes is illustrated in Figure 4.3.

When coupling a fine node to its coincident coarse node, the macroscopic physical properties, i.e. the fluid density and velocity, can be directly mapped. The coupling of the non-equilibrium distribution function f_i^{neq} is slightly different. Physically, f_i^{neq} represents the difference between the current state and its equilibrium state and is dependent on local viscous stress. Since stresses are due to variable gradients, to obtain an accurate coupling of f_i^{neq} from a fine grid to a coarse grid, not only the coincident fine node but also the neighbouring fine nodes should be taken into consideration. However, there may be high frequency waves in the fine grid that cannot be captured by the coarse grid. If these are

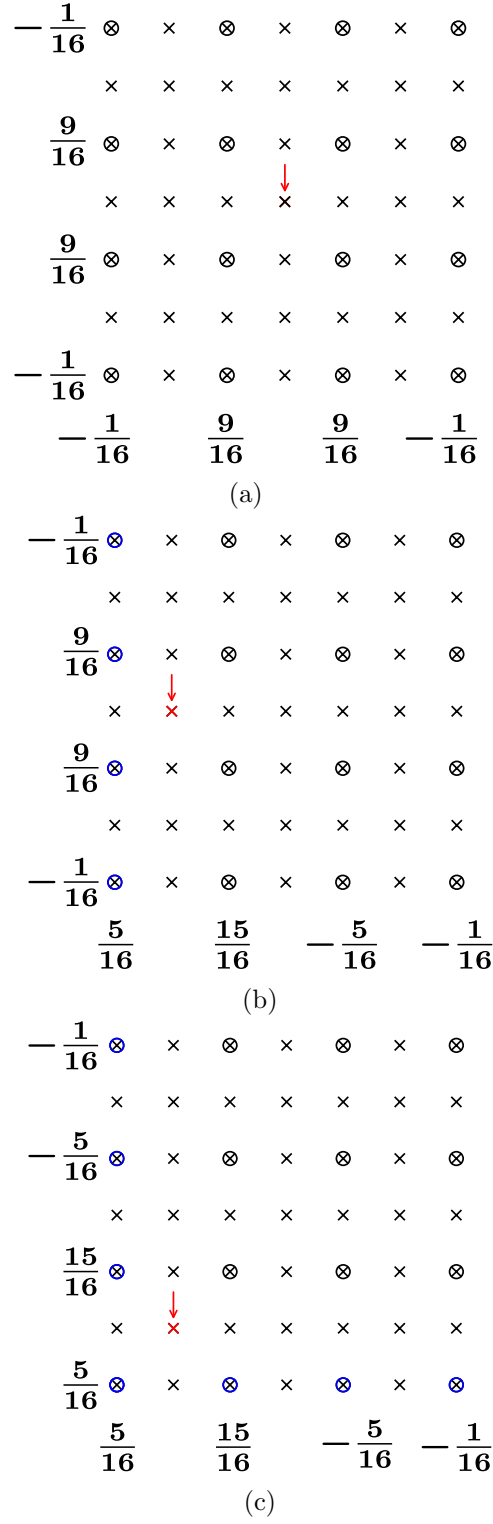


Figure 4.2.: Interpolation on 2D centre (a), edge (b), and corner (c) nodes

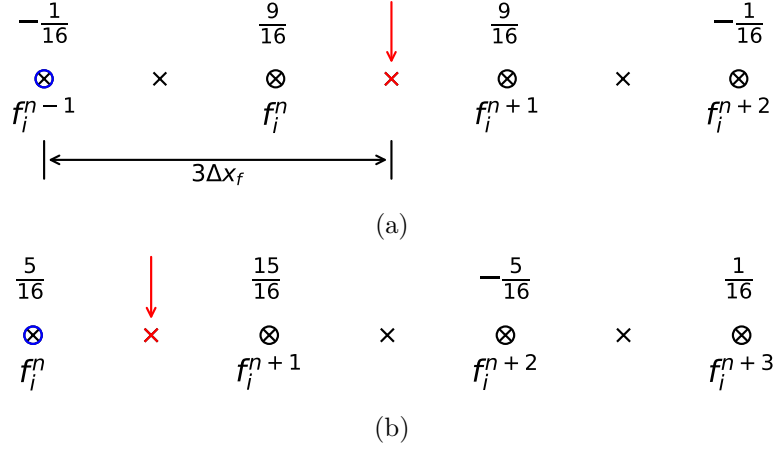


Figure 4.3.: Interpolation on 1D centre (a) and edge (b) nodes

not filtered, a drop in accuracy and stability may occur in the turbulent regime. A representative value of these fine nodes is used instead to calculate the new f_i^{neq} and this filtering procedure is called the restriction process. One simple restriction method proposed by Lagrava [86] is to take the average of all f_i^{neq} of surrounding fine nodes over each lattice direction as

$$f_{i,c}^{neq}(x, t) = \frac{1}{q} \sum_{i=0}^{q-1} f_{i,f}^{neq}(x + c_i \Delta x_f, t) . \quad (4.10)$$

An alternative to this simple averaging method is to regularise the f_i^{neq} , as will be explained in Section 4.1.4.

4.1.4. Regularised Coupling

Although the coupling algorithm above restores identical density and velocity on both grids, the stress state is not exactly preserved during this transformation. One reason is that the f_i^{neq} contains not only the stress information, but also high order terms $\mathcal{O}(\epsilon^2)$ (see Section 2.4 for more details), and the rescaling in Eq. 4.9 is only valid when the high order terms are neglected. Therefore the rescaling in Eq. 4.9 introduces numerical errors to local viscous stress. Spurious waves may generated from the refinement interface as discussed in Section 3.5.4, which will result in numerical instabilities and background noises [5].

Instead of directly constructing the unknown non-equilibrium populations from Eq. 4.9, a regularisation procedure on f_i^{neq} can be used before coupling, which uses the first-order perturbation $f_i^{(1)}$ in place of f_i^{neq} . To calculate $f_i^{(1)}$ explicitly, we first revisit Eq. 2.76, which showed the relation between $f_i^{(1)}$ and

the strain rate tensor and reads

$$f_i^{neq} \approx f_i^{(1)} = -\frac{\tau \omega_i}{c_s^2} Q_{i\alpha\beta} \partial_\alpha \rho u_\beta . \quad (4.11)$$

Substituting Eq. 4.11 into the perturbation moment $\Pi_{\alpha\beta}^{(1)} = \sum_i f_i^{(1)} c_{i\alpha} c_{i\beta}$, the non-equilibrium momentum can be expressed as

$$\Pi_{\alpha\beta}^{neq} \approx \Pi_{\alpha\beta}^{(1)} = -\tau c_s^2 (\partial_\alpha \rho u_\beta + \partial_\beta \rho u_\alpha) . \quad (4.12)$$

Comparing the right-hand side of Eq. 4.11 and Eq. 4.12, we can write

$$f_i^{(1)} = \frac{\omega_i}{2c_s^4} Q_{i\alpha\beta} \Pi_{\alpha\beta}^{neq} . \quad (4.13)$$

Using $\Pi_{\alpha\beta}^{neq} = \Pi_{\alpha\beta} - \Pi_{\alpha\beta}^{eq}$ by definition, the first-order perturbation can be explicitly calculated to be

$$f_i^{(1)} = \frac{\omega_i}{2c_s^4} Q_{i\alpha\beta} \left(\sum_i f_i c_{i\alpha} c_{i\beta} - \rho u_\alpha u_\beta - \rho c_s^2 \delta_{\alpha\beta} \right) . \quad (4.14)$$

In the regularised coupling, the unknown non-equilibrium population is replaced by the first-order perturbation, and the scaling of $f_i^{(1)}$ in Eq. 4.9 still applies, i.e.,

$$f_{i,c}^{neq} \approx f_{i,c}^{(1)} = \frac{2\tau_c}{\tau_f} f_{i,f}^{(1)} . \quad (4.15)$$

The higher order perturbations in the BGK dynamics are removed in this procedure to ensure the symmetry of the distribution functions. Considering that the aerodynamics are actually over-determined in LBM's velocity sets, this restriction provides more information to the populations on the refinement interface and better stability can thus be achieved [56].

On the other hand, the non-equilibrium momentum flux tensor is related to the viscous stress tensor as shown in Eq. 2.83, so the regularised coupling can be interpreted as a mapping between the local stress on both grids. It also ensures that physical properties are exactly preserved during coupling. For this reason, the restriction procedure using simple averaging required when coupling the coarse non-equilibrium population $f_{i,c}^{neq}$ from the coarse ones (see Section 4.1.3 for more details), can be replaced because the regularised coupling already filters the local stress state.

Compared to conventional coupling methods discussed in Section 3.5 that directly couples f_i^{neq} , the regularised coupling accurately preserves physical properties across the refinement interface, and therefore significantly improves the

numerical stability in high Reynolds simulations, which will be shown in Chapter 6. It is also worthwhile to point out that this regularised coupling method, unlike the RLB scheme mentioned in Section 3.5.4, does not depend on any particular collision operator since it modifies the post-collision populations instead of the collision process, and therefore can be applied to most schemes.

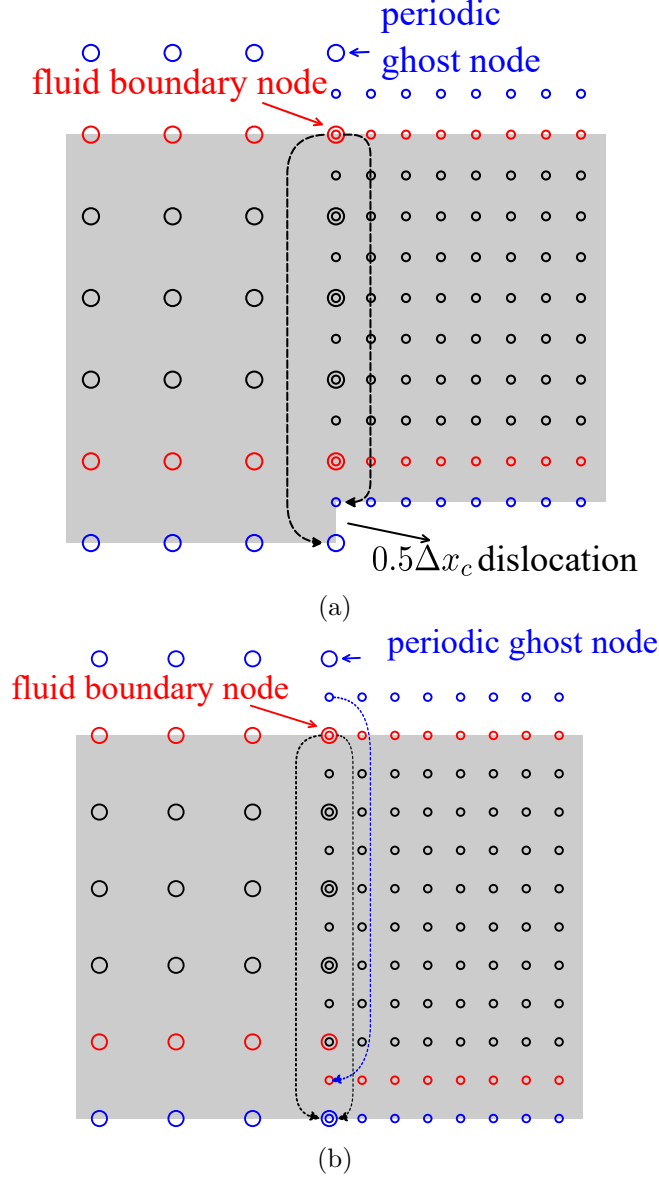


Figure 4.4.: Problematic (a) and corrected (b) grid layout for periodic boundary

4.2. Treatment on Periodic Boundary Condition

The periodic boundary condition with the original grid layout is problematic when applied to a refinement interface. As shown in Figure 4.4a, the periodic boundary conditions are implemented using ghost nodes, such that the fluid boundary node marked in red is mapped to the periodic ghost node on the other side for communication. However, different grid spacings in a refined grid will result in the $0.5\Delta x_c$ dislocation on periodic boundaries shown in Figure. 4.4a. This is because the periodic ghost nodes on the two sides contribute to an extra Δx fluid domain as shadowed in the figure, but the periodic grid contains $2\Delta x$ grid spacings, which creates a mismatch in the refined grid. This dislocation will produce a discontinuity of flow variables near the interface and a gap in the cross-sectional area.

By re-organising the grid structures as illustrated in Figure 4.4b, one more fluid layer is added to the fine grid to align same mapping locations of periodic ghost nodes and fluid nodes. Homologous fluid nodes are now mapped to one physical position and the mass flow is correctly captured with an equal cross-sectional area of both grids.

4.3. Overall Collision and Streaming Process

The overall collision and streaming process in the overlapped region, where the fine and coarse couplings take place, is illustrated in Figure 4.5. A whole computational step with time interval Δt_c contains one collision and streaming process in the coarse grid and two in the fine grid. The time step length is proportional to grid spacing in order to maintain a constant lattice velocity.

Normal collision and streaming take place on both grids first, resulting in unknown populations in the boundary nodes on the refinement interface. This

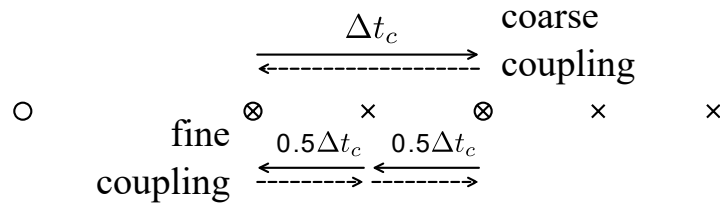


Figure 4.5.: Collision and streaming in the overlapped region. Solid arrows denote the population streaming from inner fluid nodes to boundary nodes, while dashed arrows represent the population streaming from boundary nodes to inner fluid nodes.

overlapped layout provides a coupling between the interface boundary node to the inner fluid node, e.g. the coarse boundary node coincides with the fine fluid node. These unknown populations are therefore re-calculated from corresponding fluid nodes of the other grid.

Note that the collision and streaming on fine nodes occur twice in a coarse time step, and after each collision and streaming, the fine coupling is required to calculate the unknown boundary populations. At time step $t + \Delta t_c/2$, the fine coupling involves the coarse populations at the intermediate state, which is linearly interpolated from coarse populations before and after the coarse collision and streaming, i.e. at time step t and $t + \Delta t_c$.

4.4. Summary

A vertex-centred hierarchical refinement method with regularised coupling was explained in this chapter. This coupling methodology consists of two elements: the direct coupling between coincident coarse nodes and fine nodes, the interpolation from surrounding coarse nodes to inner fine nodes. The fourth-order interpolation and the regularised coupling method were detailed. The regularised coupling method is a novel method to improve the stability around the refinement interface, and can be widely applied to popular collision operators. To solve the dislocation problem between the refinement interface and the periodic boundary, a new grid layout was proposed and explained. Finally, the overall collision and streaming process of this grid refinement strategy were summarised.

5. OpenLB Implementation

This chapter introduces the main features and basic data structure of the OpenLB software, which is necessary to describe the implementation of the grid refinement method. The implementation itself is covered next.

5.1. The Open-Source Library OpenLB

OpenLB is a widely used open-source code released under a GPLv2 license designed for LBM simulations. It is implemented in C++ with encapsulated libraries suitable for both application programmers and developers. The implementation of the grid refinement method in this work is performed on version 1.3.

5.1.1. Main Features

Velocity Sets

A wide variety of lattice types have been implemented in OpenLB, including D2Q5, D2Q9 for 2D simulations and D3Q7, D3Q13, D3Q15, D3Q19, D3Q27 for 3D simulations.

The velocity sets, named as *descriptor* in OpenLB, describes the structure of populations on each computational cell. Each descriptor containing essential parameters including d and q denoting the number of dimensions and number of discrete velocities respectively, c_i indicating the discrete velocity vectors, and w_i representing the weight of each discrete velocities. The descriptor must be chosen in advance of the simulation.

Collision Operators

Various collision operators including BGK, MRT, RLB, and entropic LB [91] are available in OpenLB. The object-oriented feature of C++ makes the implementation of new collision operators straightforward. New collision operators can be inherited from the *Dynamics* class, which provides a basic structure of

the collision functions. A typical *Dynamics* template contains a *collide()* function which performs the collision step as in Eq. 2.62 and several supplementary functions to calculate local variables such as density, velocities and momenta.

Subgrid models

Several subgrid models have been implemented in OpenLB, in a similar way to the collision operators. These include the basic Smagorinsky model and its several variants like shear-improved Smagorinsky model and dynamics Smagorinsky model, WALE and ADM model.

The objective of this work is to extend the use of LBM to high Reynolds number flows with a contained computational cost, thus subgrid scale models have been used as an essential tool for achieving this objective.

Boundary Conditions

Although locality is an important feature of the LBM, the boundary conditions sometimes require non-local computations to account for flow features in the vicinity. For example, the outlet pressure boundary may face stability issues at high Reynolds number when the unknown populations are recovered only from local populations. An alternative is to use the interpolated pressure boundary condition, which computes the stress field from neighbouring cells.

A number of local and non-local boundary conditions such as simple bounce-back, periodic boundary condition, and Bouzidi's interpolated bounce-back boundary conditions have been implemented in OpenLB. Both local and non-local boundary conditions require a new *Dynamics* object to perform special boundary dynamics and calculate missing populations on boundary cells. The non-local boundary condition need extra post-processing step called *Postprocessor* to operate the finite difference calculations. The *Postprocessor* is usually called after performing collision and streaming to correct the populations on boundaries. Two different boundary structures, i.e. the wet-node and bounce-back as stated in Section 2.6, are also available.

Hybrid Parallelisation

There are two classical parallel programming frameworks: distributed memory and shared memory. Using a distributed memory approach, the memory space that each parallel process uses is isolated and the communication between parallel processes is performed after all threads have finished a particular task. On the

other hand, using shared memory all threads share the same memory space to work on one task.

These two approaches are not mutually exclusive, and OpenLB uses a hybrid parallelisation approach employing the well known Message Passing Interface (MPI) library [93] for distributed memory parallelisation based on domain decomposition, and the OpenMP multithreading library for shared memory parallelisation of internal loops. This hybrid implementation allows to decide which approach to use depending on the problem size and available computational resources. For a small problem size with sufficient available memory and processor cores, shared memory will be more efficient than domain decomposition [94]. Domain decomposition will be then necessary if the size of the calculation exceeds the capabilities of a single processor. But even within a multiple domain calculation, distributed and shared memory techniques can be used simultaneously in order to make efficient use of all the available resources.

The scalability performance of OpenLB has been checked recently [92] on up to 32,784 cores using the common lid-driven cavity example at various grid resolution. As illustrated in Figure 5.1, the computation speed, denoted by million lattice updated per processing unit and second (MLUPps), remains at similar scale when parallel threads increases. This shows that OpenLB is well-parallelised and is suitable for simulating large scale problems.

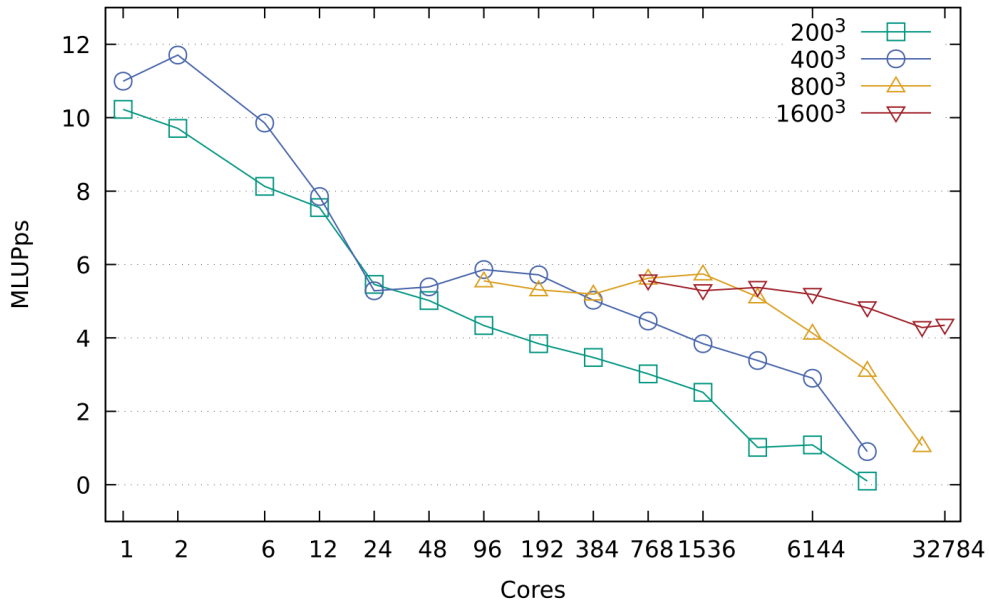


Figure 5.1.: Scalability performance of OpenLB [92]

Convergence Monitor

OpenLB provides a class *ValueTracer* to compute and record the evolution of some parameter ϕ . This is useful for computing residuals or statistics of unsteady behaviour, which is necessary to check the convergence.

In this work, a convergence criteria based on fluctuation statistics is used, by monitoring the standard deviation σ of a parameter ϕ . The parameter ϕ can be selected as local properties or their mean value over the domain. Once the standard deviation drops below the target value $\epsilon\bar{\phi}$, time-convergence is achieved

$$\sigma(\phi) = \sqrt{\frac{1}{N+1} \sum_{i=0}^N (\phi_i - \bar{\phi})^2} < \epsilon\bar{\phi}, \quad (5.1)$$

where N is the windowed time step number, $\bar{\phi}$ is the averaged ϕ_i over the last N time steps as

$$\bar{\phi} = \frac{1}{N+1} \sum_{i=0}^N \phi_i. \quad (5.2)$$

The choice of N is important and should be large enough to capture the frequencies of interest, for example the vortex shedding behind a bluff body. In these simulations, steady state solution is not achievable as the vortex shedding never ends, but the flow can be considered as converged if the fluctuation statistics approach a constant value.

Post-Processing

During each iteration of collision and streaming, macroscopic flow variables (density and velocities) are calculated. OpenLB supports writing flow state quantities into Visualization Toolkit (VTK) files for visualisation and post-processing using ParaView.

Checkpointing functions for saving and loading data are also available. Since all simulation data are stored in local cells, current simulation state can be recorded as long as all essential data is preserved. The checkpoint file can be loaded when initiating a new simulation, which is useful when performing long-time simulations.

5.1.2. Data Structure

The OpenLB data structure is sketched in Figure 5.2. The computational domain is discretised by a regular Cartesian grid, which is composed of a number of basic grid cells. The *Cell* structure represents a computational cell and contains

the populations of each discrete velocities. The structure of these populations are determined by the *descriptor* as introduced before in Section 5.1.1. Another important variable of the *Cell* class is the *Dynamics* object which represents the collision model. *Dynamics* of each *Cell* are assigned at initialisation of the program and can differ from others. This implementation provides high flexibility of OpenLB in simulating inhomogeneous fluids. The *Cell* class additionally offers interfaces to directly define the populations and macroscopic quantities during simulation.

An array of *Cell* objects compose a *BlockLattice3D*, which represents a 3D cuboid set of the computational grid. Irregular boundaries are modelled by flagging certain cells as unused. To implement the hybrid parallelisation, a higher level class called *SuperLattice3D* is defined. The difference between *BlockLattice3D* and *SuperLattice3D* is that the *BlockLattice3D* represents a partition of the whole grid, while the *SuperLattice3D* can be seen as the set of all *BlockLattice3D* objects.

During parallel computing, the computational domain is divided into cuboids of similar volume and each of the cuboids are discretised into a *BlockLattice3D* structure. The *SuperLattice3D* provides an overall user interface and controls the communication between each *BlockLattice3D*. This data structure enables efficient parallelisation on distributed and shared platforms.

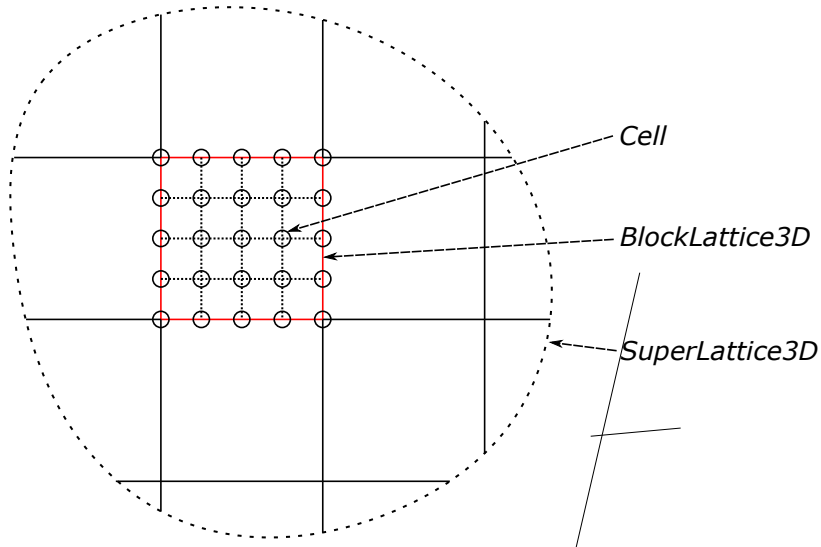


Figure 5.2.: Sketch of the data structure in OpenLB.

5.1.3. Description of the Simulation Set Up Process

In order to carry out a simulation, the following steps need to be performed:

1. Initialise
 - Define global parameters such as Reynolds number and domain length in each direction;
 - Create a *UnitConverter* object which links the dimensionless and lattice units through defining grid resolution, relaxation time, characteristic length, characteristic velocity, characteristic viscosity and density. The conversion parameter of each macroscopic property is calculated and stored to map the simulation variables to physical units.
 - Construct a *timer* to record the computational statistics such as the computational speed (measured in million lattice updated per second (MLUPs)) and lapsed time information.
 - Initiate a *ValueTracer* as convergence monitor. The monitoring parameter and target residual should be set in advance.
2. Prepare Geometry
 - Create a geometrical *indicator* of the computational domain. The *indicator* class supports importing geometry from an STL file and geometrical operations such as combination and subtraction.
 - Construct a *CuboidGeometry3D* from the *indicator*. The mesh is automatically generated based on the grid resolution defined in *UnitConverter*. If parallel computation of n MPI threads is required, the *CuboidGeometry3D* is divided into n small cuboids optimally.
 - Create a *LoadBalancer* that assigns each cuboid to a different MPI instance.
 - Construct a *SuperGeometry3D* which is a higher level structure of *CuboidGeometry3D* and *LoadBalancer*. Every voxel in the computational mesh is assigned a material number, with 0 indicating null cell, 1 representing fluid cells by default, and other numbers specifying additional collision dynamics.

3. Prepare Lattice

- Create a *SuperLattice3D* from *SuperGeometry3D*. Every voxel of the *SuperGeometry3D* is substituted by a *Cell* object representing a computational cell. A *BlockLattice3D* is initialised on each cuboid.
- Assign different *Dynamics* according to the material number of each cell. The boundary cell dynamics are also defined during this step.
- Define essential boundary conditions. Note that the boundary dynamics and boundary conditions are different. Boundary dynamics determine how the missing populations are constructed, while the boundary conditions restrict the populations based on macroscopic properties. For example, the bounce-back boundary can be fulfilled using a simple bounce-back collision dynamics, while the velocity boundary requires a constant macroscopic velocity and the unknown populations are calculated from flow properties instead of collision dynamics. The *Postprocessor* object, which is required for the non-local boundaries, are also initialised during this step.
- Define initial fluid state and boundary conditions. The initial populations are set as equilibrium according to desired macroscopic properties. For certain boundary condition such as the velocity boundary, its velocity is set to the initial value if no further operation is taken.

4. Collide and Stream

- Perform collision and streaming on *SuperLattice3D*. The boundary nodes are communicated with other lattices after inner nodes have been updated.
- Modify boundary values to achieve flexible control if desired, e.g. by gradually increasing the inlet velocity to avoid abrupt incoming pressure waves.
- Calculate the residuals and execute *ValueTracer*.
- Write output fluid results when necessary.

5.2. Implementation of Grid Refinement Method

The present implementation is based on Kummerländer’s 2D framework [95], but is adapted to 3D with the regularised coupling method. The implemented code is open-access and can be downloaded via the Github page [96].

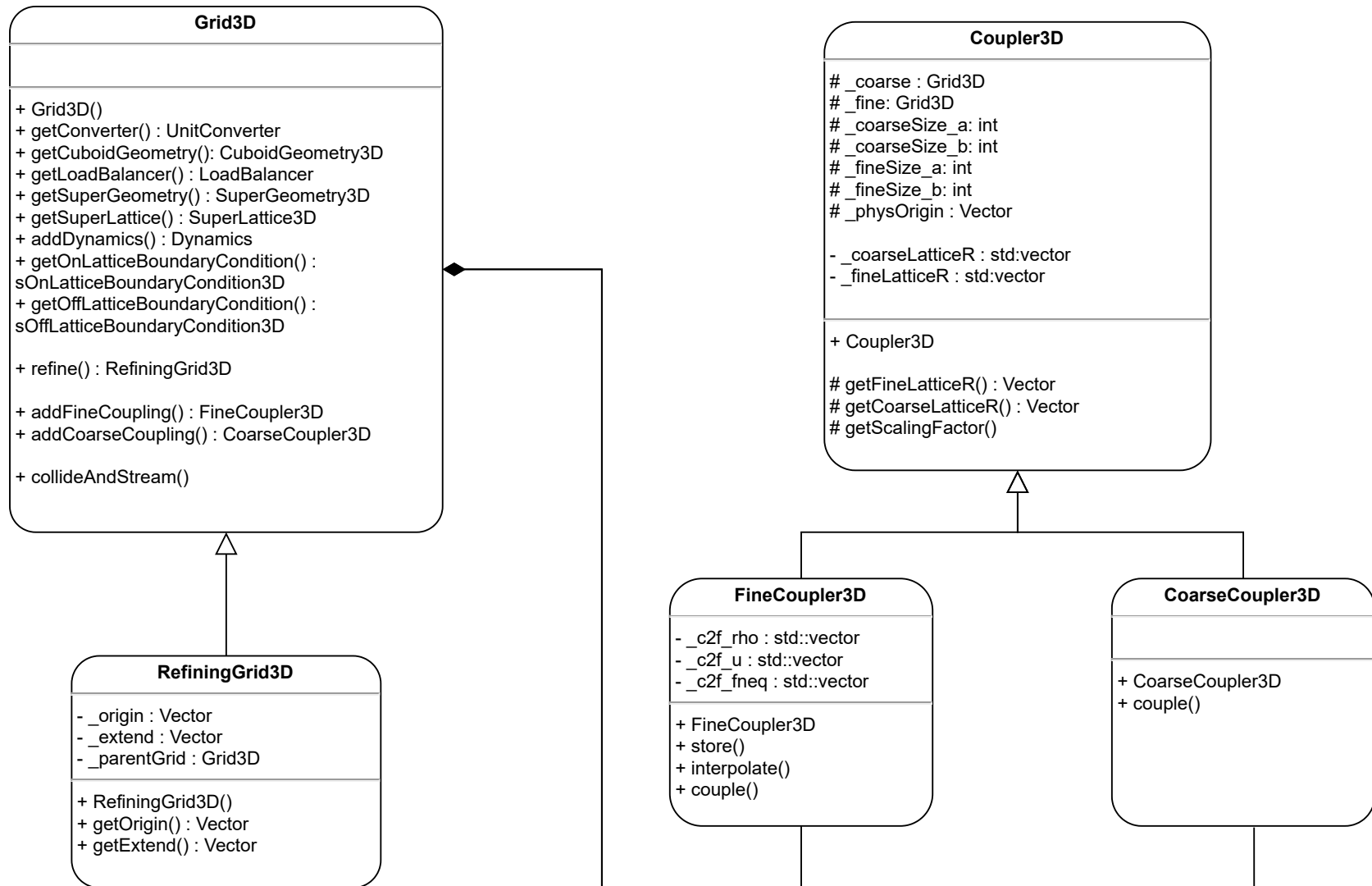


Figure 5.3.: Class diagram of the grid refinement data structure.

5.2.1. Overall Data Structure

The grid refinement method requires coupling between grids of different resolutions. As explained in the previous section, all essential classes such as *UnitConverter* and *SuperGeometry3D* are specific to a grid, which means different instances need to be created for each grid. An all-in-one class named *Grid3D* has been constructed to include all grid-dependent parameters, and inherit properties from the coarse grid to the fine one when refining the resolution. The fine grid is created based on a coarse grid via its member function *refine()*. Assigning the desired position and size of the region to be refined, the *refine()* function instantiates a fine grid object called *RefiningGrid3D*. The *RefiningGrid3D* is an inheritance of the object *Grid3D* so that they possess same data structure. The *Dynamics* and *BoundaryCondition* which is specific to a grid is also included in the *Grid3D* class. The detailed *Grid3D* structure will be explained later in Section 5.2.2.

Another important class is *Coupler3D*, which determines the mapping between the coarse grid and the fine grid. The absolute and relative positions of each interface cell are recorded. Two *Coupler3D* objects will be instantiated when a refined grid is defined: one *FineCoupler3D* object for coupling from coarse nodes to corresponding fine nodes, one *CoarseCoupler3D* object operating vice versa. When performing collision and streaming on the *Grid3D*, the *couple()* function will be called to perform coupling on each *Coupler3D* and update populations on each grid. The *Coupler3D* structure and its interpolation methods are covered in Section 5.2.3.

A simplified class diagram of these structures is shown in Figure 5.3. The overall collision and streaming step are performed using the member function *collideAndStream()* in the *Grid* class. Since all refined grids are child classes inherited from the coarsest grid, the *collideAndStream()* of all grids can be called iteratively from the base level. Section 5.2.4 will address the whole algorithm.

5.2.2. The *Grid3D* Class

A constructor of the *Grid3D* class is shown in Code 5.1. A *Grid* object contains all necessary elements to construct a computational unit of a uniform grid. These member classes such as *UnitConverter*, *CuboidGeometry3D*, *LoadBalancer*, *SuperGeometry3D*, and *SuperLattice3D* are identical to those used in general OpenLB programs. The periodicity condition in each direction can also be set and assigned to *CuboidGeometry3D* in this step. The program also provides

Listing 5.1: Constructor for class *Grid*

```

1  template <typename T, typename DESCRIPTOR>
   Grid3D<T,DESCRIPTOR>::Grid3D(FunctorPtr<IndicatorF3D<T>>&& domainF,
                                RelaxationTime<T> tau, int resolution,
                                Characteristics<T> characteristics,
5     bool periodicityX, bool periodicityY,
                                bool periodicityZ,):
   // domain indicator functor
   _domainF(std::move(domainF)),
   // Characteristic properties of the fluid
   _characteristics(characteristics),
10  // Periodicity in each direction
   _periodicityOn(periodicityX, periodicityY, periodicityZ),
   // Construct UnitConverter for Grid3D
   _converter(new UnitConverterFromResolutionAndRelaxationTime
15     <T,DESCRIPTOR>(
        resolution, tau,
        characteristics.length,
        characteristics.velocity,
        characteristics.viscosity,
20     characteristics.density)),
   // Construct CuboidGeometry for Grid3D
   _cuboids(new CuboidGeometry3D<T>(
        *domainF,
        _converter->getConversionFactorLength(),
25     periodicityX, periodicityY, periodicityZ,
#ifdef PARALLEL_MODE_MPI
        singleton::mpi().getSize()
   #else
        1
30  #endif)),
   // Construct LoadBalancer for Grid3D
   _balancer(new HeuristicLoadBalancer<T>(
        *_cuboids)),
   // Construct SuperGeometry for Grid3D
35  _geometry(new SuperGeometry3D<T>(
        *_cuboids,
        *_balancer,
        2)),
   // Construct SuperLattice for Grid3D
40  _lattice(new SuperLattice3D<T,DESCRIPTOR>(
        *_geometry))
{
   _converter->print();
}

```

Listing 5.2: Interface functions of *Grid*

```

1 Characteristics<T>                getCharacteristics() const;
  UnitConverter<T,DESCRIPTOR>&      getConverter();
  CuboidGeometry3D<T>&              getCuboidGeometry();
  LoadBalancer<T>&                  getLoadBalancer();
5 SuperGeometry3D<T>&               getSuperGeometry();
  SuperLattice3D<T,DESCRIPTOR>&     getSuperLattice();

  Dynamics<T,DESCRIPTOR>&
    addDynamics(std::unique_ptr<Dynamics<T,DESCRIPTOR>>&& dynamics);
10 sOnLatticeBoundaryCondition3D<T,DESCRIPTOR>&
    getOnLatticeBoundaryCondition();
  sOffLatticeBoundaryCondition3D<T,DESCRIPTOR>&
    getOffLatticeBoundaryCondition();

```

functions as listed in Code 5.2 to read and modify these parameters from outside of this class scope. Other members declared but not initialised such as *Dynamics* and *BoundaryCondition*, are instantiated using these functions. This structure simplifies the construction of a refined grid by inheriting data and methods from the coarse grid.

The interface of *RefiningGrid3D* is illustrated in Code A.1, which is deposited in Appendix A.1 for conciseness. To define a refined grid, the cuboid block is created first indicating the region to be refined. The block can be determined by selecting an origin position and extension lengths in three dimensions. The periodicity in each direction is same to the coarse grid by default, but is free to adjust according to physical positions. Note that the subdomain of the coarse grid (excluding the overlap layer) which is now refined should be cleared. This can be achieved by setting the material number of those cells to 0 during *prepareGeometry*.

The *Coupler3D* objects are included in a *Grid3D* class as a member to describe the coupling relation between a coarse grid and its refined grid. As listed in Code A.2 in Appendix A.1, two types of *Coupler3D* are defined: One is the *FineCoupler3D* mapping from the coarse grid to the fine grid, and the *CoarseCoupler3D* coupling vice versa. Both *Couplers3D* are declared as a member of the coarse grid, and take the fine grid as its parameter. The position of the refinement interface is calculated from the origin position and extension lengths of the *RefiningGrid3D*. It should be clarified that these two *Coupler3D* objects correspond to different refinement interface, i.e. the interface from fine to coarse and the interface from coarse to fine are not identical. Actually the two interfaces are separated by a coarse grid spacing.

Listing 5.3: General Structure of *Coupler3D*

```

1  template <typename T, typename DESCRIPTOR>
   class Coupler3D {
   protected:
       Grid3D<T,DESCRIPTOR>& _coarse; // reference of the coarse grid
5      Grid3D<T,DESCRIPTOR>& _fine;   // reference of the fine grid
       const int _coarseSize_a;      // array size of interface
       const int _coarseSize_b;      //   on the coarse grid
       const int _fineSize_a;        // array size of interface
       const int _fineSize_b;        //   on the fine grid
10
       // physical position of the interface origin
       const Vector<T,3> _physOrigin;
       // functions to map lattice position of both grids
       // from the physical position
15      const Vector<int,4>& getFineLatticeR(int x1, int x2) const;
       const Vector<int,4>& getCoarseLatticeR(int x1, int x2) const;

       // scaling factor to map coarse and fine populations
       T getScalingFactor() const;
20      T getInvScalingFactor() const;

   private:
       // vectors recording the lattice positions of every node
       std::vector<std::vector<Vector<int,4>>> _coarseLatticeR;
25      std::vector<std::vector<Vector<int,4>>> _fineLatticeR;

   public:
       // constructor of a $Coupler$ class
       Coupler3D(Grid3D<T,DESCRIPTOR>& coarse,
30                  Grid3D<T,DESCRIPTOR>& fine,
                   Vector<T,3> origin, Vector<T,3> extend);
   };

```

It is important to sort out the inheritance of coarse grid and its refined grid since the refined grid should be able to accept further refinement. Vectors storing these refined grids and *Coupler3D* objects are created as shown in Code A.3 in Appendix A.1. Interface to these refined grids from the parent coarse grid is also provided using these vectors, which is helpful when iterating the collision and streaming function.

5.2.3. The *Coupler3D* Class

The *Coupler3D* class and its derived ones function as a bridge between coarse grid and fine grid populations. Either the coarse one or the fine one contains missing populations because the node is on the lattice boundary. The *Coupler3D*

class therefore provides a way to reconstruct these nodes from the corresponding nodes of the other grid.

The base *Coupler3D* class interface is given in Code 5.3. The nodes on the refinement interface are interpreted into 2D arrays recording its absolute position in the original grid. As such, the corresponding node on the original grid can be mapped to the interface node by simply calling the new functions *getFineLatticeR()* or *getCoarseLatticeR()*. Note that if periodic boundary is defined, this mapping needs to consider the extra “ghost” nodes in the periodic direction as addressed in Section 4.2.

The *FineCouplers3D* and the *CoarseCouplers3D* are constructed to implement the coupling algorithm separately, and the *FineCoupler3D* requires extra functions for the interpolation process. Since the *FineCouplers3D* and the *CoarseCouplers3D* have similar data structure, they can inherit from the general *Coupler3D* class and implement separate member functions. The description of data structure of *FineCoupler3D* and *CoarseCoupler3D* are detailed in Appendix A.2.

5.2.4. Collision and Streaming Algorithm

As shown in Code 5.4, the collision and streaming on all grids can be performed by recursively calling itself from the coarsest level. Before calling it, the flow state at current time step of the coarse grid is recorded for future interpolation. Then the collision and streaming are performed for both coarse and fine grids respectively. The first coupling is then performed after calculating the intermediate state of the coarse properties, to prepare for the second collision and streaming on the fine grids. The second coupling of fine nodes are based on the updated coarse populations. In the end the coarse coupling is performed to update the remaining coarse nodes.

5.3. Parallel Performance Testing

It is important to test the scalability of the code in the hardware that it is going to be used in. Therefore, the software has been compiled for running in the HPC facilities at Imperial College, which comprises of machines of dual socket with Intel Xeon CPU E5-2660 V2 at clock speed of 2.20 *GHz*.

There are two approaches to scale the parallel performance in respect of total computation cores: the weak scaling and the strong scaling. In terms of weak scaling, the problem size is increased in proportion to the number of parallel threads so that the workload on each core is constant. The efficiency is calculated

Listing 5.4: Overall collision and streaming algorithm

```
1  template <typename T, typename DESCRIPTOR>
   void Grid3D<T,DESCRIPTOR>::collideAndStream()
   {
       // storing coarse properties of last time step
5      for ( auto& fineCoupler : _fineCouplers ) {
           fineCoupler->store();
       }

       // collision and streaming on coarse grid
10     this->getSuperLattice().collideAndStream();

       // first collision and streaming on fine grids iteratively
       for ( auto& fineGrid : _fineGrids ) {
           fineGrid->collideAndStream();
15     }

       // temporal interpolation and first coupling on fine grids
       for ( auto& fineCoupler : _fineCouplers ) {
           fineCoupler->interpolate();
20         fineCoupler->couple();
       }

       // second collision and streaming on fine grids iteratively
       for ( auto& fineGrid : _fineGrids ) {
25         fineGrid->collideAndStream();
       }

       // updating coarse data and second coupling on fine grids
       for ( auto& fineCoupler : _fineCouplers ) {
30         fineCoupler->store();
           fineCoupler->couple();
       }

       // coupling on coarse grid
35     for ( auto& coarseCoupler : _coarseCouplers ) {
           coarseCoupler->couple();
       }
   }
```

by comparing the real computational time $t(N)$ of N threads with respect to one thread time $t(1)$, i.e.

$$Efficiency = \frac{t(1)}{t(N)} . \quad (5.3)$$

The ideal efficiency stays as 1 but is not realistic because the parallel threads need to communicate with each other, and the communication speed is related to problem size and network latency. The strong scaling, in contrast, fixes the problem size when increasing the number of threads. The speedup can be calculated in the same way as the weak scaling:

$$Speedup = \frac{t(1)}{t(N)} . \quad (5.4)$$

Ideal speedup is linear while Amdahl's law points out the effect of the serial part of the software in real cases and the speedup profile yields

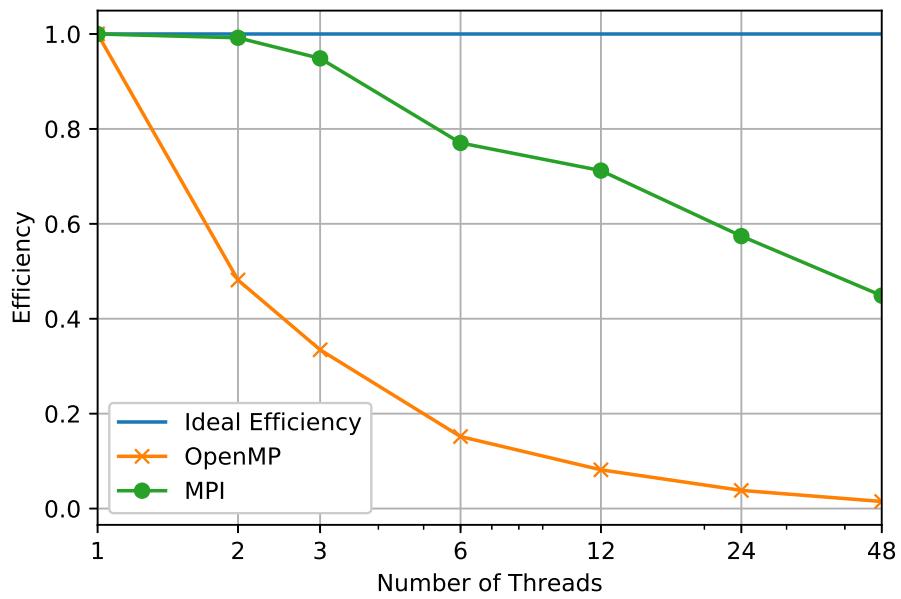
$$Speedup = \frac{1}{s + p/N} , \quad (5.5)$$

where s and p denote the proportion of the time costs of the serial part and parallel part respectively.

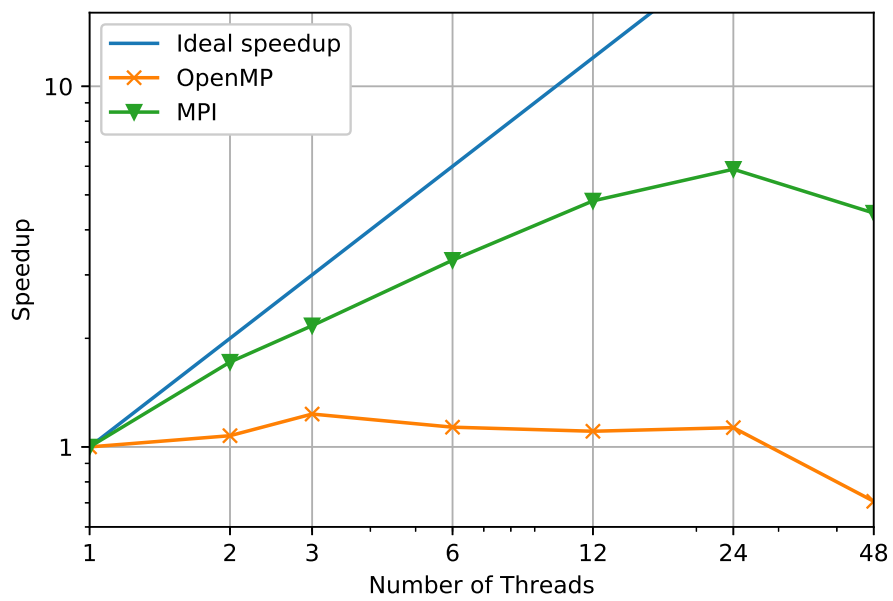
5.3.1. Evaluation of OpenLB

The weak and strong scaling have been tested for both the distributed and shared memory architecture, i.e. using MPI and OpenMP respectively, and is illustrated in Figure 5.4. The OpenLB example code "laminar/cavity3d/parallel" was chosen as the test case, which does not involve the grid refinement code implemented in this research, in order to evaluate the scalability performance of OpenLB on HPC clusters at Imperial college. For weak scaling, the number of active cells per processor was kept as 0.14 million; while the strong scaling preserved the total number of active cells over all threads to be 0.14 million. Both scaling profiles show a increased global speedup with more parallel threads used. The drop in the strong scaling speedup from $N = 24$ to $N = 48$ can be explained by the hyper-threading technology adopted in the Intel architecture, which address two virtual cores on one physical processor core to maximise total available parallel threads.

The strong scaling performance of OpenMP is largely related to the total problem size. The speedup profile can be affected if the computational load is too small to efficiently use all memory resources. As shown in Figure 5.5, test cases with different problem sizes (evaluated via total number of active cells)



(a) Weak scaling



(b) Strong scaling

Figure 5.4.: Scalability performance of OpenLB on HPC facilities at Imperial College

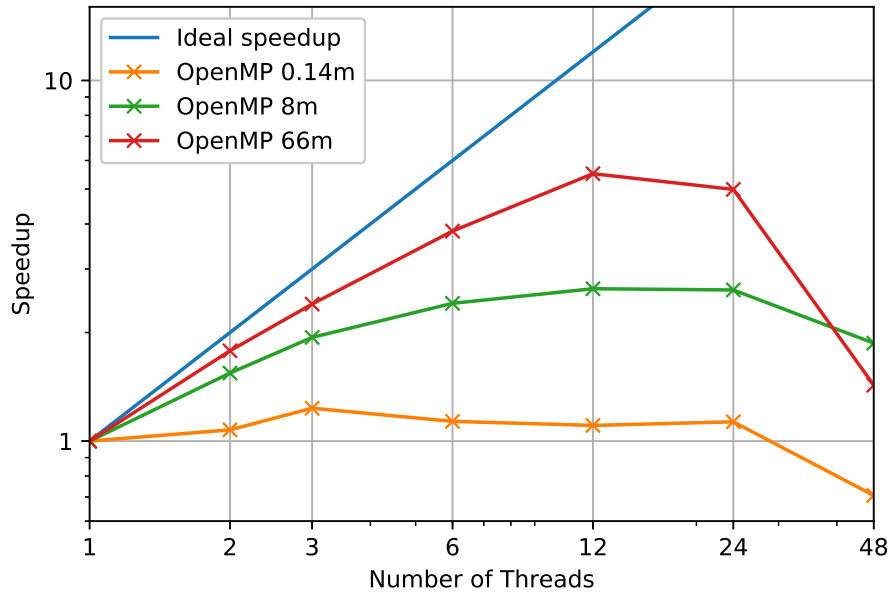


Figure 5.5.: Strong scaling performance of OpenMP with total number of active cells 0.14 million, 8 million, and 66 million.

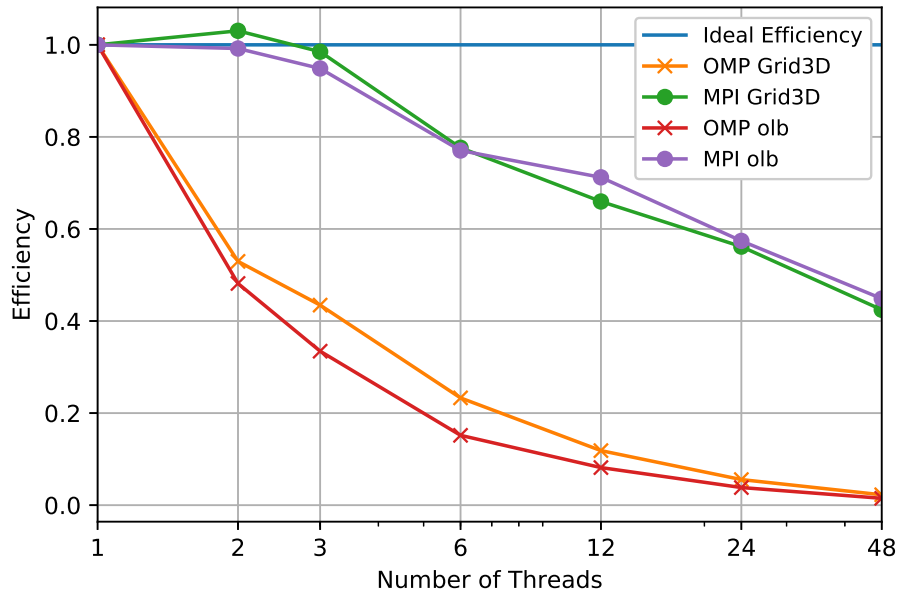


Figure 5.6.: Weak scaling of the *Grid3D* class versus original OpenLB (olb) class

show various speedup profiles. With a total number of active cells at around 66 million, an optimum speedup can be observed at 12 parallel threads. However, the optimum number of active cells per processor may vary depending on the domain partition layouts in specific problems.

The scalability performance of OpenLB at HPC clusters at Imperial college exhibits obvious difference depending on the parallelisation method. Since the emphasis of this research is on the grid refinement method and its implementation, the scalability performance of the grid refinement code will be compared with the original OpenLB to evaluate the implementation.

5.3.2. Evaluation of the Grid Refinement Implementation

Scalability tests have been performed to evaluate the performance of the grid refinement code, using the shared memory parallelism framework OpenMP. Domain decomposition required by MPI has not yet been implemented for the grid refinement interface.

The parallel performance of *Grid3D* structure, which is implemented in place of the original *SuperLattice3D*, needs to be evaluated first. Their weak scaling results of test cases with same problem size are plotted in Figure 5.6. It is shown the scalability of *Grid3D* is similar to the original OpenLB data structure.

Figure 5.7 illustrates a strong scaling test of the refined grid and the uniform

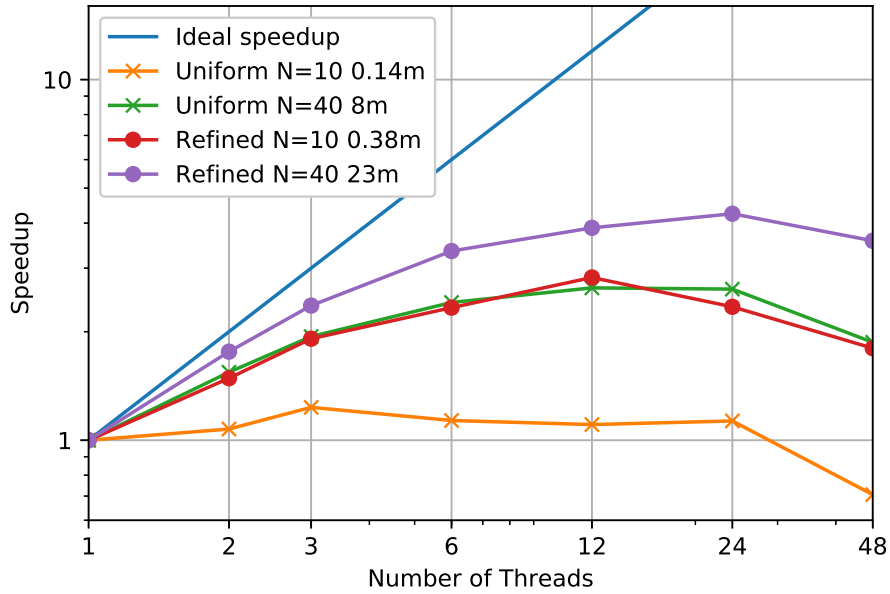


Figure 5.7.: Scalability of the grid refinement code versus original OpenLB

grid. The total number of active cells of the two cases vary, even the coarsest grid resolution of the refined grid matches the resolution of the uniform grid. Note that the collision and streaming calculation on refined cells need to be performed twice during one operation on coarse cells, so the comparison of total problem size is difficult. Nevertheless, the overall speedup trend of the refined grid is similar to the uniform one. This similar profile with and without refined grid validates that, the scalability of the grid refinement implementation is on the same level as the original OpenLB code, and is suitable for running on high performance computers.

5.4. Summary

This chapter described the implementation of the grid refinement method in the OpenLB open-source code.

The main features and the data structure of OpenLB were introduced. OpenLB, as a widely used open-source software, contains complete modules to perform and post-process an LBM simulation. Those include STL geometry input, convergence monitor, post-processing functors, various collision operators, turbulence models and boundary conditions. Efficient parallelism is achieved for MPI and OpenMP, and their hybrid application. The collision and streaming algorithm was summarised in several practical steps.

After introducing the OpenLB data structure, the main three features of the grid refinement implementation were explained: the *Grid* class, the *Coupler* class, and the overall collision and streaming function. The *Grid* class is a new data structure to group all essential components to form a computational grid. The refined grid can be constructed via inheriting from the coarse *Grid* class since they share same data structure. The *Coupler* class describes the coupling method between the coarse grid and the fine grid. The derived classes, *FineCoupler* and *CoarseCoupler* implement the particular coupling and interpolation steps. A new collision and streaming function compatible with the grid refinement architecture was implemented.

The refinement architecture was shown to have a similar scalability performance on OpenMP as the uniform OpenLB grid.

6. Validation on Benchmark Cases

The simulations and results of two benchmark cases will be shown in this section to validate the 3D grid refinement method and implementation. First, the flow around a circular cylinder in the transitional regime is studied. This test case will demonstrate the capability to efficiently calculate separated flows. The second test is the lid-driven cavity case, which involves a moving wall boundary and a complex flow with multiple interacting vortical structures with varying bulk flow velocities ranging from almost stagnant to relatively high speed ones. This test case will assess the efficiency and continuity performance of the grid refinement in comparison to the uniform grid.

6.1. Vortex Shedding Flow around a Circular Cylinder

6.1.1. Case Description

Steady incoming flow past a bluff body and the vortical instabilities in its wake have been comprehensively studied. As such extensive databases recording pressure coefficient and wake velocity profiles have been compiled, for example in [97, 98, 99], which will be used as reference data. Besides these data, the 3D vortex dynamics have also been described. Different instability modes have been reported, which differ from the regular 2D vortex shedding pattern known as Kármán vortex street. These instability modes become dominant as the Reynolds number increases. Investigations by Williamson [100] conclude that in the case of a circular cylinder, the flow structure in the wake undergo transitions between four regimes at low Re , which are sketched in Figure 6.1 and explained as:

1. The laminar steady regime for $5 < Re \leq 49$, in which the recirculation region comprises two symmetrical steady vortices and the flow structure is purely 2D. As Re increases, the length of the steady bubble increases due to the viscous stresses in the wake;

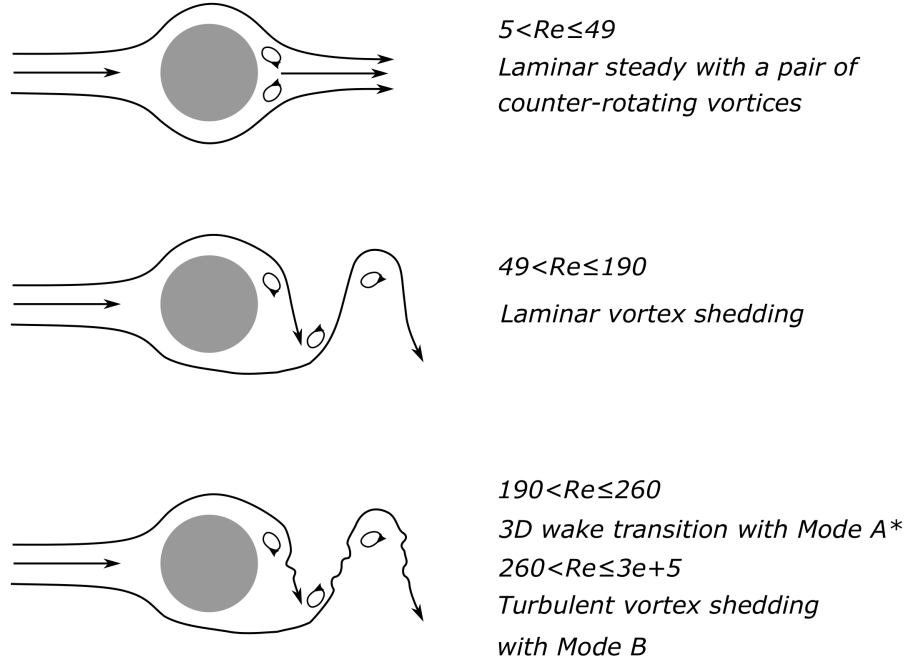
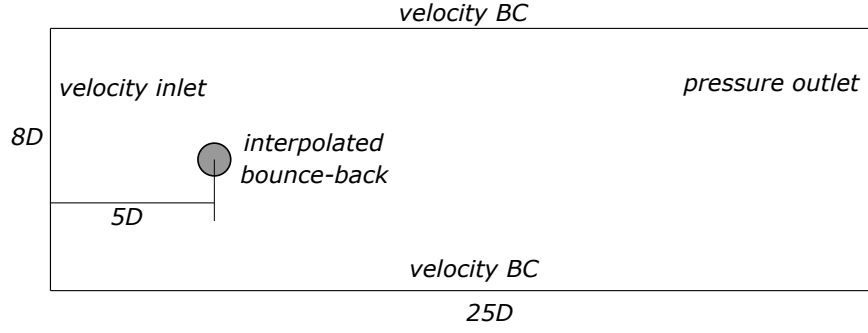


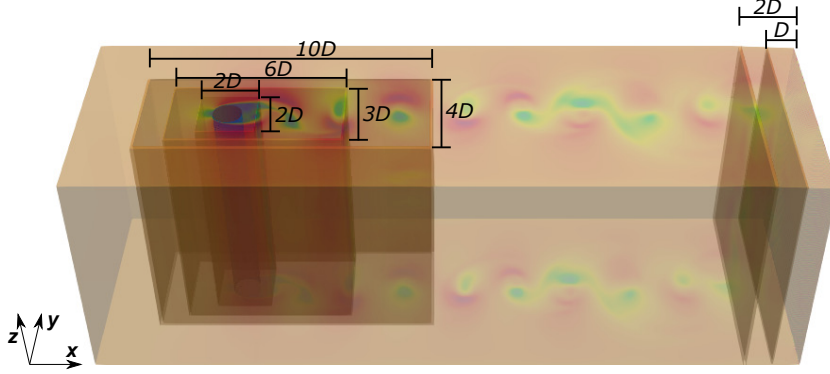
Figure 6.1.: Sketch of the turbulent transition in the cylinder wake

2. The laminar vortex shedding regime for $49 < Re \leq 190$, in which the instabilities in the recirculation region develop from the downstream end of the bubble, and the vortices detach from the cylinder periodically. The wake structure is still 2D and the vortex shedding is parallel to the cylinder axis;
3. The 3D wake transition regime for $190 < Re \leq 260$, in which two main instability modes can be identified: Mode A* instability (combination of small-scale instability called Mode A and large-scale vortex dislocation structures) with onset at $Re \approx 190$ and Mode B instability with onset at $Re \approx 260$. As will be discussed later, these modes are clearly distinct, as evidenced by their difference in wavelengths.
4. An irregular regime for $260 < Re \leq 3 \times 10^5$ with increasingly disordered Mode B structure.

This case has been chosen to validate the implementation of 3D grid refinement method and evaluate the capability of the method in capturing different 3D instability modes, ensuring that refinement interfaces do not introduce additional perturbations that interact with physical instabilities. Three cases at $Re = 100, 200$ and 300 will be run, according to the presence of laminar vortex shedding, Mode A instability, and Mode B instability respectively.



(a) Domain sketch and boundary conditions



(b) Refinement layout

Figure 6.2.: Illustrations of domain layout with boundary conditions, and the refinement layout

6.1.2. Domain Configuration and Parameter Selection

The simulation domain and layout of refinement levels are illustrated in Figure 6.2. The domain size is $25D \times 8D \times 8D$ with D being the cylinder diameter. The cylinder is placed slightly off-centre to accelerate the triggering of vortex shedding, as the symmetric geometry may take longer to generate the instability.

A constant velocity inlet and a pressure outlet are defined as the open boundary condition. A velocity boundary condition is defined on the side boundaries with same velocity as the inlet to approximate a free-stream environment. The interpolated bounce-back boundary condition as introduced in Section 2.6.3 is applied on the cylinder surface, which ensures a second-order accuracy on the curved boundary. Periodic boundary conditions are applied in the spanwise direction to approximate an infinitely long cylinder. Three levels of refinement around the cylinder are applied to improve local accuracy.

The open boundary condition in the LBM, which enforces the fluid's velocity

Table 6.1.: Summary of simulation parameters of the vortex shedding case at Re=100, 200, and 300

			Case 1	Case 2	Case 3
Reynolds number	Re	<i>input</i>	100	200	300
Characteristic length	l_0			0.1m	
Grid resolution (Coarsest)	N	<i>input</i>		10	
Length conversion factor	$C_l = \Delta x$	l_0/N		0.01m	
Characteristic velocity	u_0			1m/s	
Lattice velocity	u_{lb}	<i>input</i>		0.02	
Velocity conversion factor	C_u	u_0/u_{LB}		50m/s	
Time conversion factor	$C_t = \Delta t$	C_l/C_u		$2 \times 10^{-4}s$	
Characteristic density	ρ_0			1kg/m ³	
Lattice density	ρ_{lb}	<i>input</i>		1	
Density conversion factor	ρ_t	ρ_0/ρ_{LB}		1kg/m ³	
Characteristic kinematic viscosity	ν_0	$u_0 l_0 / Re$	$1 \times 10^{-3} \text{ m}^2/s$	$5 \times 10^{-4} \text{ m}^2/s$	$3.33 \times 10^{-4} \text{ m}^2/s$
Lattice kinematic viscosity	ν_{lb}	$\nu_0 C_t / C_l^2$	2×10^{-3}	1×10^{-3}	6.66×10^{-4}
Relaxation time	τ	$3\nu_{LB} + 0.5$	0.506	0.503	0.502
Grid Reynolds number	Re_g	Re/N	10	20	30
Total refinement levels				3	
Finest grid resolution				80	
Finest grid Reynolds number	$Re_{g,f}$		1.25	2.5	3.75

or density near boundaries, will reflect the sound waves back into the fluid [101, 102]. The reflection of these pressure waves near the outlet boundary is often a cause of simulation divergence. The regularised coupling can help reduce the strength of propagated waves, by removing the higher order errors in the non-equilibrium populations as discussed in Section 4.1.4. Therefore, in the absence of non-reflecting boundary conditions, two levels of refinement are applied near the outlet to reduce these unwanted reflections and improve stability.

As discussed in Section 2.5, each Re number run requires different scaling factors, which are summarised in Table 6.1. The grid Reynolds numbers Re_g are selected to be 10 to meet the stability criteria discussed in Section 2.5.2, while the relaxation time τ is chosen to be as small as possible to reduce the BGK truncation error. Note that if no grid refinement is applied, the grid Reynolds number around the cylinder at $Re = 300$ exceeds the recommended value $Re_g \leq 24$ and the simulation becomes unstable as tested.

The simulation parameters need to be altered on each refinement level. The parameters of the $Re = 300$ case are compiled in Table 6.2 as an example. The lattice velocity u_{lb} is kept same at all levels of grids to ensure same propagation speed and constant Mach number. The number of total active cells in the refined case is reduced to 29 million compared to the 819 million active cells which would be needed if a uniform grid with resolution $N = 80$ were used.

6.1.3. Results

Instantaneous velocity contours on the midspan section are shown in Figure 6.3. The upper plot is the contour of velocity magnitude at each Reynolds number, showing an overall development of the vortex street. The vortex pairs are still regular at $Re = 200$ but exhibit chaotic behaviour at $Re = 300$.

This disorder can be further observed in the contours of velocity components for each direction. As is seen in Figure 6.3b and Figure 6.3c, the y-component undergoes only small changes both qualitatively and quantitatively. In contrast, the magnitude of spanwise z-component rises from $O(10^{-9})$ at $Re = 100$ to $O(10^{-5})$ at $Re = 200$ and $O(10^{-1})$ at $Re = 300$. This relatively small component

Table 6.2.: Simulation parameters of different refinement levels at $Re=300$

N	u_{lb}	τ	Re_g	Ma	Δx	Δt
10	0.02	0.502	30	0.035	0.01	2×10^{-4}
20	0.02	0.504	15	0.035	0.005	1×10^{-4}
40	0.02	0.508	7.5	0.035	0.0025	5×10^{-5}
80	0.02	0.516	3.75	0.035	0.00125	2.5×10^{-5}

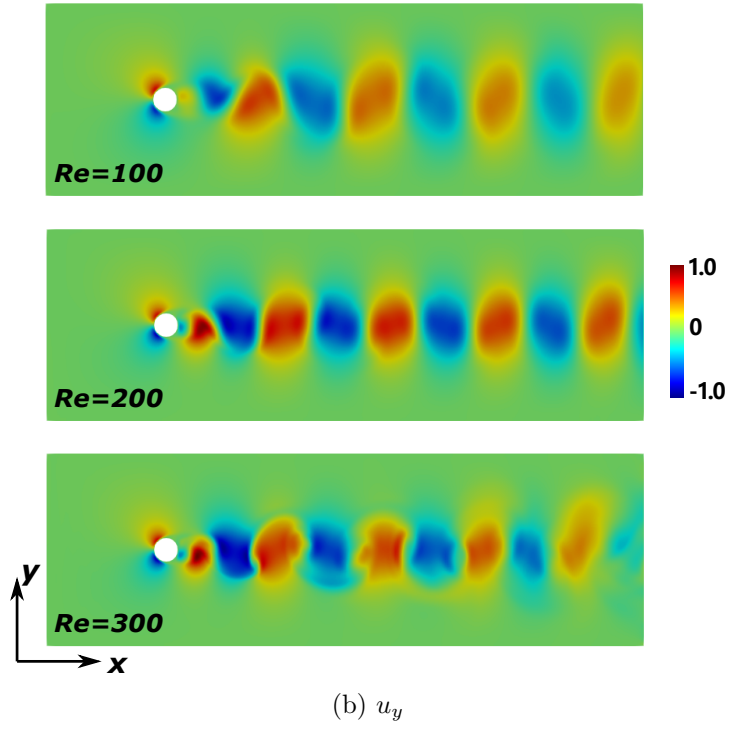
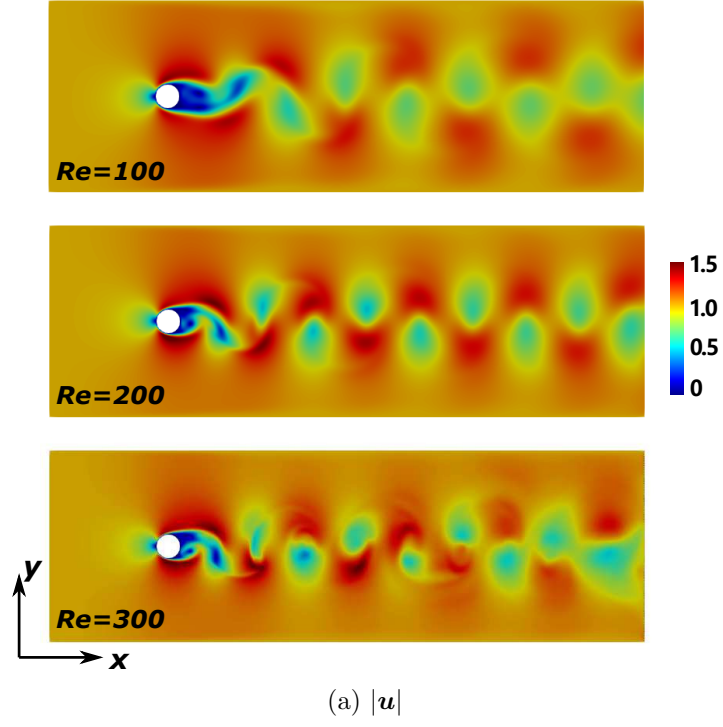


Figure 6.3.: Velocity contours of the magnitude ($|\mathbf{u}|$), y-component (u_y), and z-component (u_z) at $Re=100$, 200 and 300 with refined grid

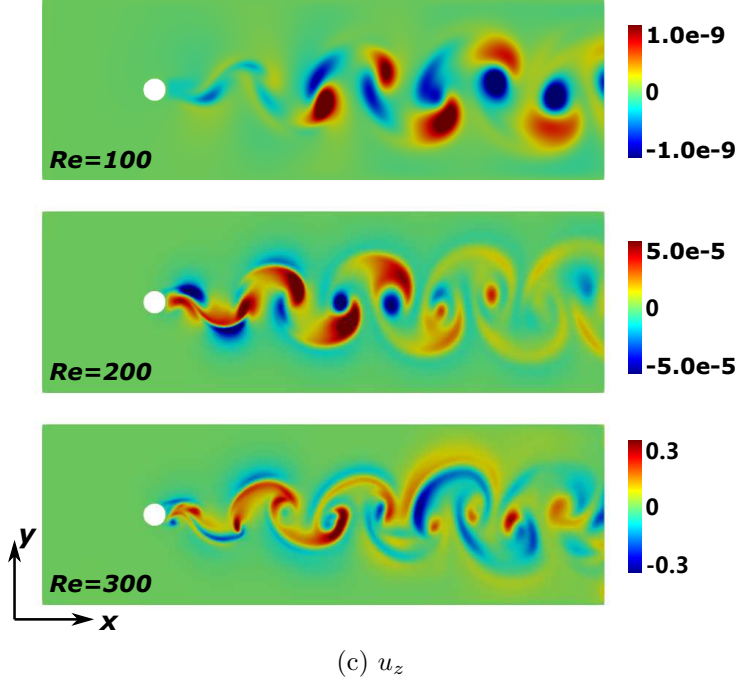


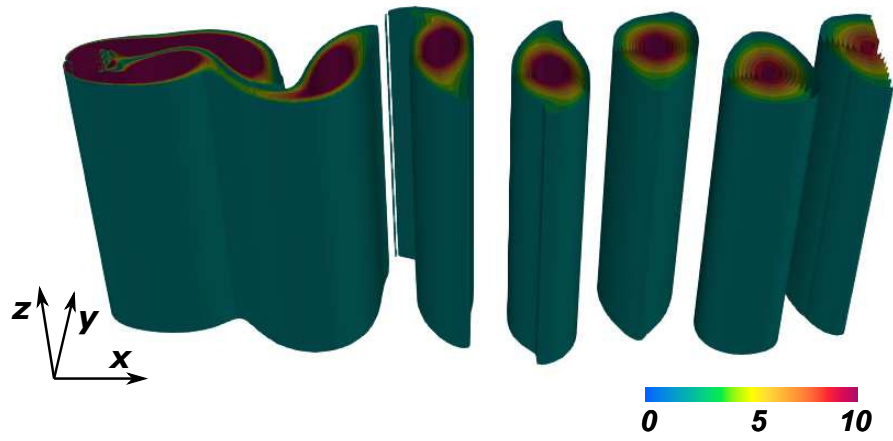
Figure 6.3.: Velocity contours of its magnitude ($|\mathbf{u}|$), y-component (u_y), and z-component (u_z) at $Re=100$, 200 and 300 with refined grid (cont.)

makes no contribution to the flow pattern when Re is low. For $Re = 300$, however, the transverse velocity has grown to about same magnitude as the other two and results in a transition in the cylinder wake.

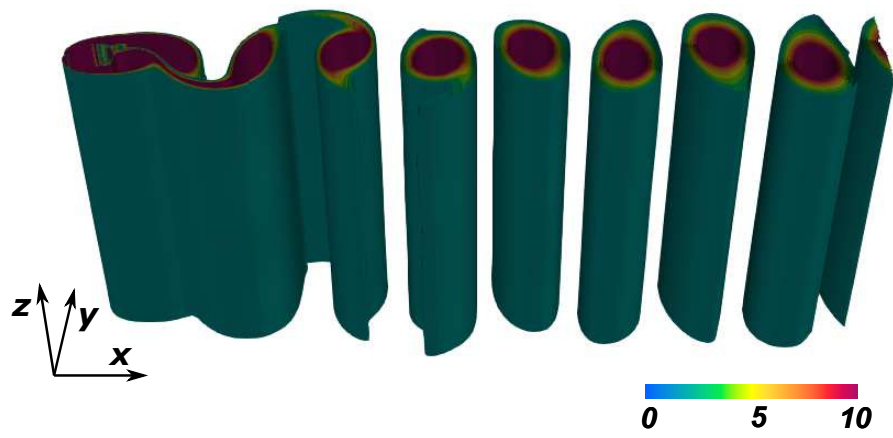
It can therefore be seen how this transition is dominated by a spanwise deformation of vortex structures. This development in the third direction that cannot be revealed by two-dimensional approaches shows the wake transition is inherently three-dimensional. From the velocity contours it can also be seen that the continuity of velocity field is preserved across the refinement interfaces.

Apart from the previously described primary instability, two small-scale three-dimensional instabilities are involved when Re increases within the studied range. The first 3D instability (named Mode A) is due to the spanwise-periodic deformation of the primary vortices at Re around 180. This small-scale instability further triggers dislocations of large-scale vortices. The combination of them, named as Mode A* by Williamson [100], induces the turbulent transition of the wake.

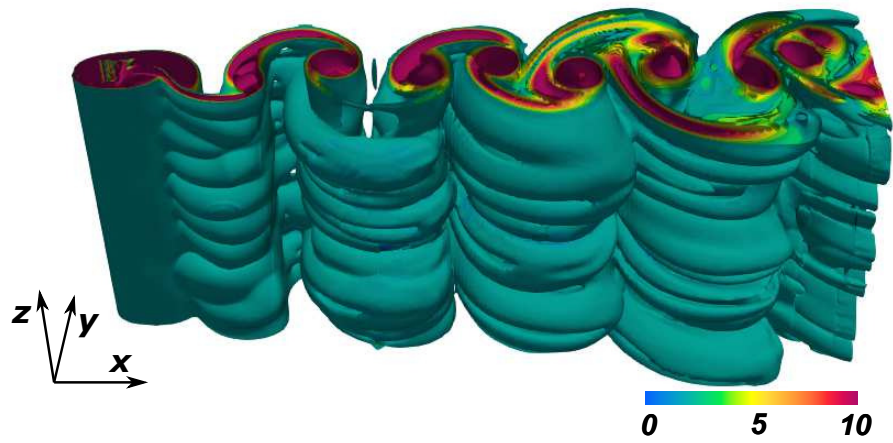
The other instability Mode B with onset at around $Re=230$ exhibits a smaller spanwise wavelength and fine-scale wave pattern, indicating a different vorticity structure from Mode A. The detailed spanwise vortex structures are further



(a) $Re=100$



(b) $Re=200$



(c) $Re=300$

Figure 6.4.: Isosurface plot of vorticity magnitude at $Re=100$, 200 and 300

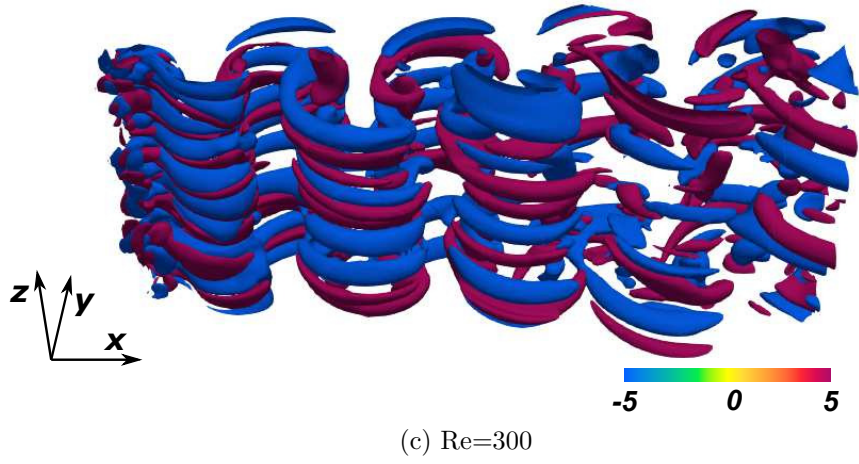
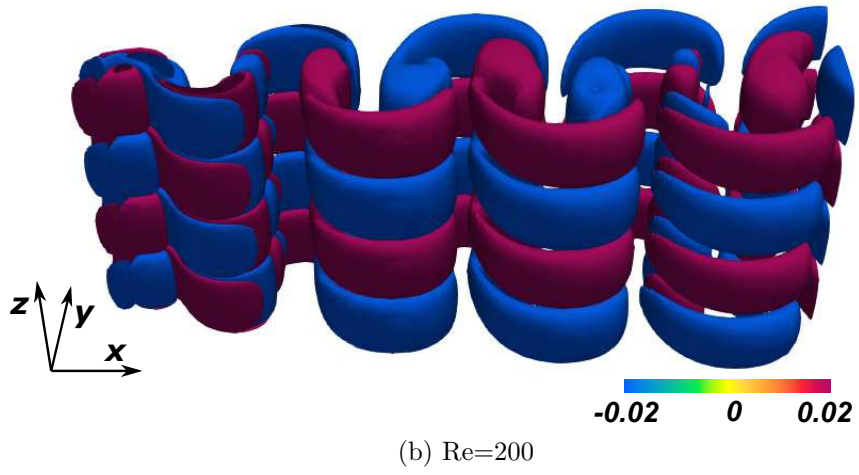
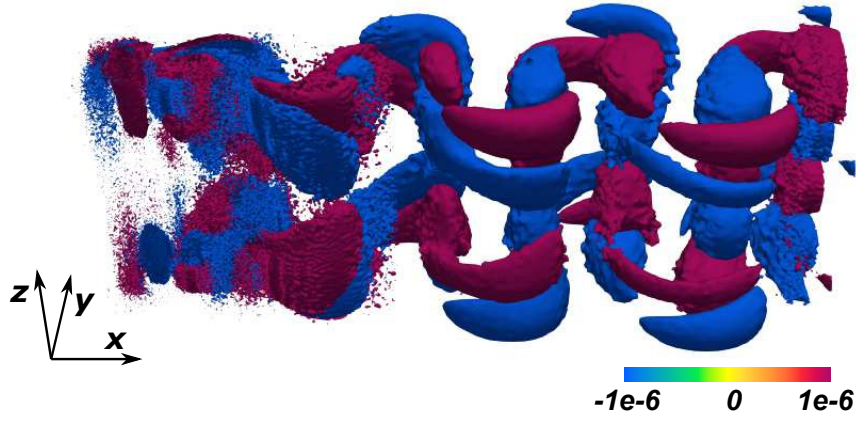


Figure 6.5.: Isosurface plot of streamwise (x -) component of vorticity at $Re=100$, 200 and 300

illustrated in Figure 6.4 and Figure 6.5 by the isosurfaces of vorticity magnitude and its streamwise component. Comparing the isosurface plots of the vorticity magnitude and the streamwise component at $Re = 100$ and $Re = 200$, it can be seen that although the main vortex structure is still of 2D shape, the Mode A instability has already appeared in the vorticity field. From $Re = 100$ to $Re = 200$, the magnitude of this instability has increased from $\mathcal{O}(10^{-6})$ to $\mathcal{O}(10^{-2})$. The streamwise vortices are out of phase from one braid to the other one in Mode A. The wavelength of this spanwise periodic pattern is around 4 diameters, which corresponds to Williamson's experimental results [100].

In contrast, the main vortex structure at $Re = 300$ is largely affected by the spanwise instability, as its magnitude rises up to same level as the main flow. The periodic pattern becomes more irregular with a wavelength around 1 diameter, and the streamwise vortices become locked in phase, which is also described by Williamson. Overall, the observation in Williamson's experiments of two instability modes and flow patterns was accurately reproduced in this study.

Further validation on the time-averaged pressure coefficient C_p around cylinder and wake velocity profiles are shown in Figure 6.6 and Figure 6.7 for quantitative validation. The pressure coefficient at $Re=100$ is in good agreement with reported values [97, 98].

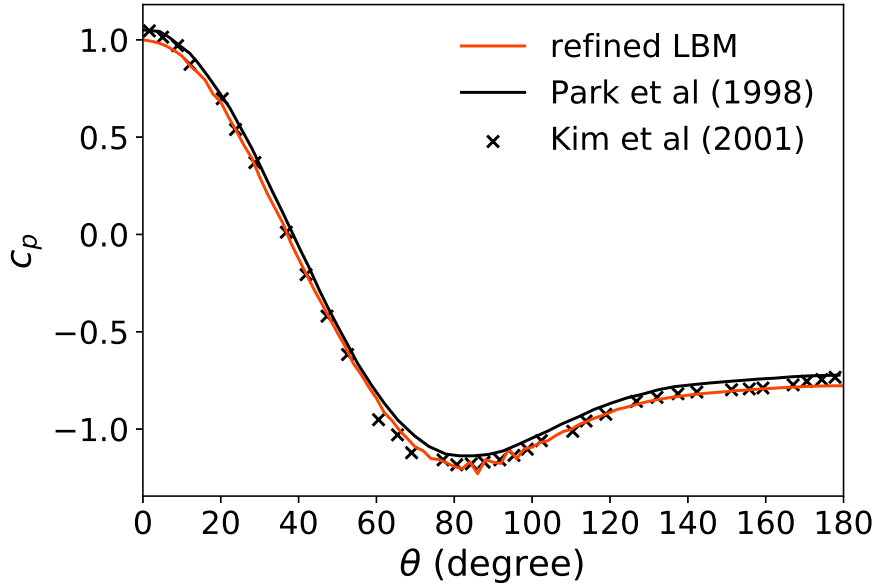
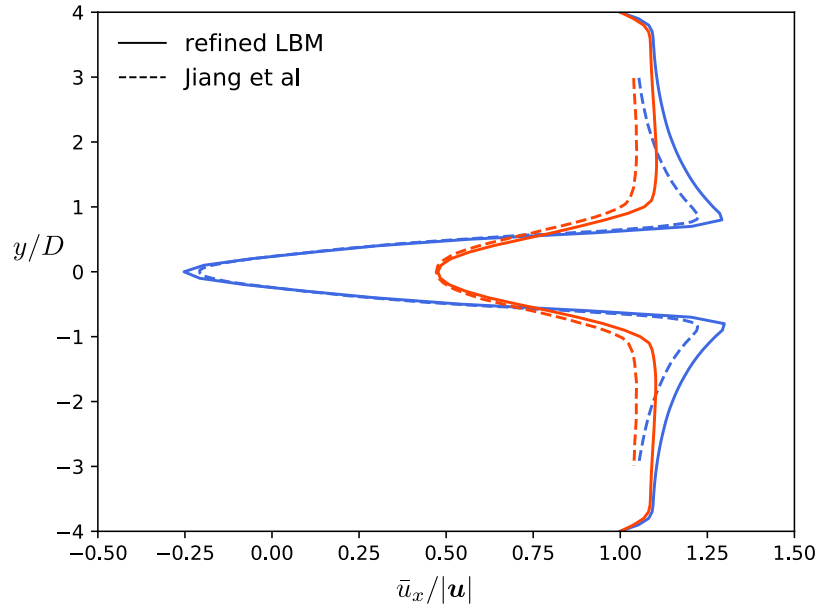
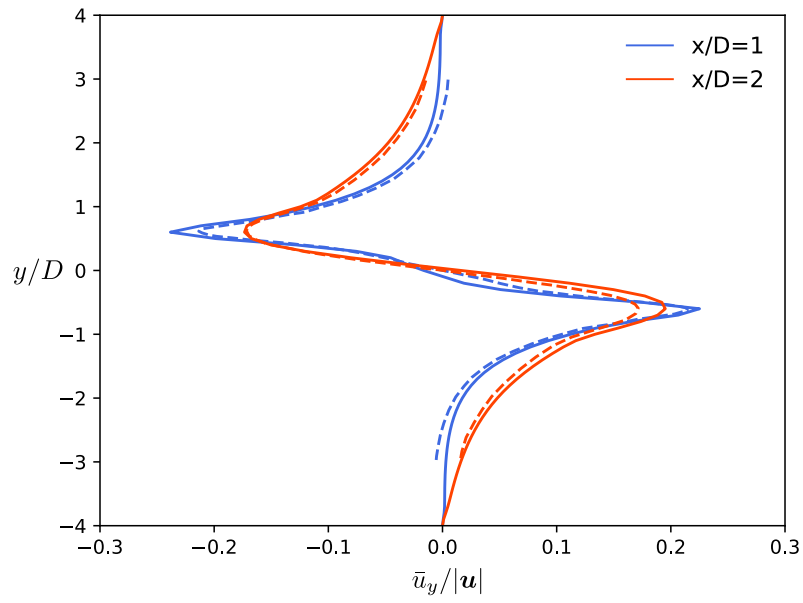


Figure 6.6.: Pressure coefficients around the cylinder at $Re=100$



(a) x-component of velocity



(b) y-component of velocity

Figure 6.7.: Time-averaged velocity profiles of LBM results with refined grid compared with simulation results from Jiang et al. [99] in the cylinder wake of $Re=300$ at $x/D = 1$ and $x/D = 2$

Wake velocity profiles at $Re = 300$ are computed by time-averaging U_x and U_y at two axial locations $x/D = 1$ and $x/D = 2$ behind the cylinder. The results are compared to the simulation results by Jiang et al. [99]. The wake deficit is accurately captured, while the external flow exhibits a bit higher U_x due to the imposed velocity in the boundaries. Overall the results are in good agreement with reference data.

This case highlights the benefits of using grid refinement. The simulations above have a resolution of $N = 10$ in the free-stream. The equivalent uniform grid case with $N = 10$ will become unstable at $Re = 300$. In that case the general grid Reynolds number is $Re_g = 30$, but the local grid Reynolds number is higher because the flow passage is accelerated across the cylinder. Grid refinement is shown to improve simulation stability by tailoring the local grid Reynolds. This is an indicator of the resolution required, and can be greater than the global value, if the local flow velocity is higher than the mean flow velocity.

The grid refinement is shown to significantly reduce computational costs. If a uniform grid with finest resolution $N = 80$ is applied, the number of total active cells is around 819 million, which is much higher than the 29 million needed by the refined grid. When accounting for the additional associated cost due to the time step sizing, the total computational costs will increase by a factor of approximately 200 using the acoustic scaling. With diffusive scaling, this would be even higher.

6.2. Lid-driven Flow inside a Cubic Cavity

6.2.1. Case Description

The lid-driven flow inside a cubic cavity is a well-studied benchmark that is of theoretical significance as an internal flow problem. The cavity is enclosed by rigid walls and driven by a plane wall with tangential movement at a constant velocity. Complex flow phenomena such as separation, recirculation and reattachment will take place with vortex formation and stretching at many scales, depending on the Reynolds number based on the lid velocity. When simulating this benchmark with the LBM method, a particular numerical issue arises as the velocities at two intersecting boundaries are not uniquely defined. This will be addressed in Section 6.2.2.

On the plane of symmetry ($y = L/2$) as shown in Figure 6.8, the vortex structure is composed of a primary vortex in the centre of the cavity and three secondary vortices generated on the corner from flow recirculation.

The lid-driven flow is a suitable benchmark to validate the implementation of

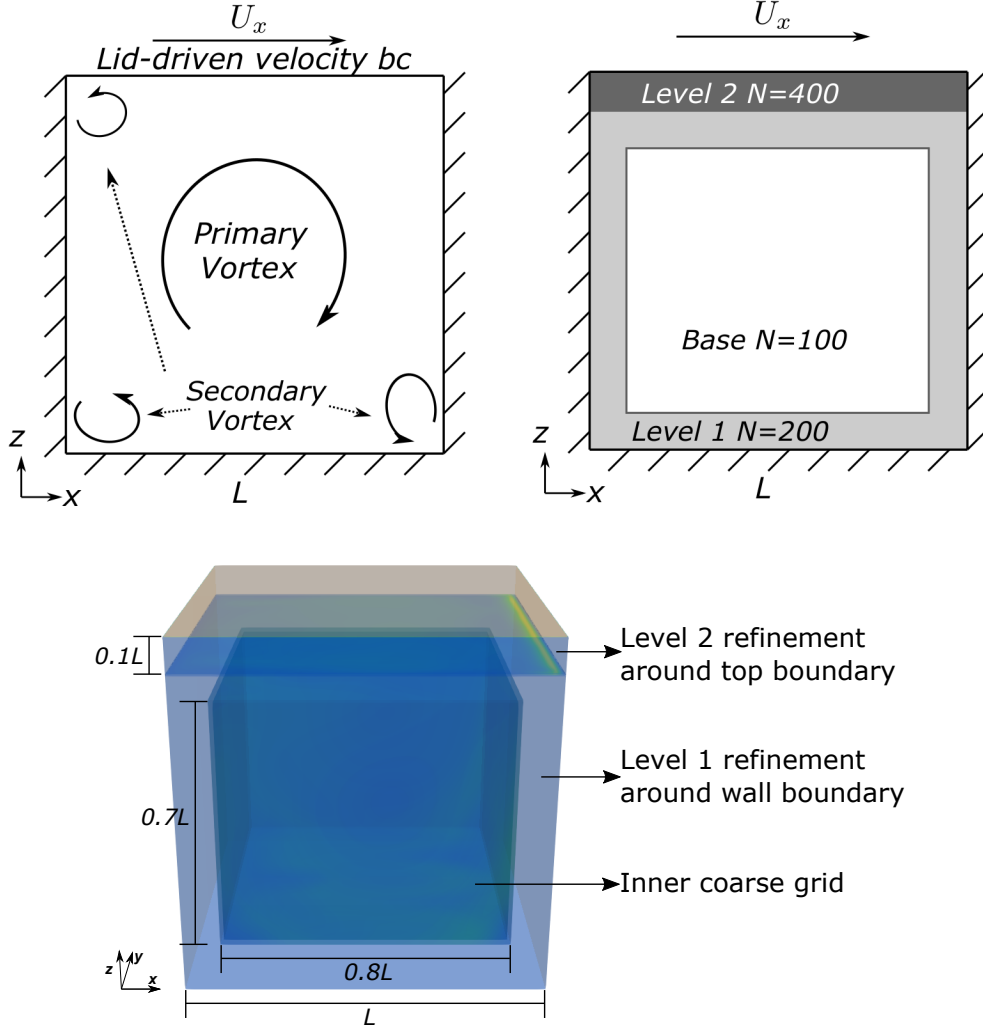


Figure 6.8.: Domain geometry and refinement layout of the 3D lid-driven cavity

the grid refinement method. The closed streamline configuration can be used to evaluate the flow continuity across the refinement interface, and the need to resolve flow structures along a range of scales demonstrates the increase in computational efficiency with respect to a uniform resolution grid.

6.2.2. Domain Configuration and Parameter Selection

The Reynolds number is set to 3200 for simulating turbulent flow and the Mach number is fixed to 0.035, within the range of validity of the incompressibility assumption [35, 103]. The geometry of the simulation domain is depicted in Figure 6.8. The cubic cavity with length L is surrounded by five stationary walls and a moving top wall with uniform velocity U_x . These resting walls are

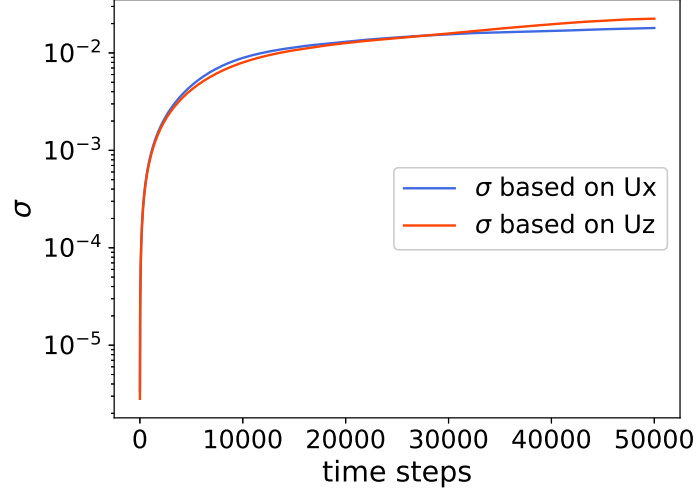


Figure 6.9.: Measured standard deviation of flow velocity versus computational steps after the initial transient

modelled using the bounce-back condition, while a Zou-He velocity condition (see Section 2.6) is applied to the top wall. The accuracy and stability of the numerical simulation may be affected by the discontinuity in velocity at the intersecting edges of rigid walls and the top moving boundary [104]. The use of a refinement interface with regularised coupling around the top layer can help reduce these errors and improve simulation stability.

Two different refinement layouts are tested and their results are compared against the result of a uniform grid. Case A involves an all-bounded refinement along all boundaries (i.e. base $N = 100$ and level 1 $N = 200$ in Figure 6.8), while Case B contains an additional refinement near the top boundary (i.e. base $N = 100$, level 1 $N = 200$, and level 2 $N = 400$ in Figure 6.8) to help stabilise the flow around the top plane.

6.2.3. Convergence and Computational Costs

The convergence monitor introduced in Section 5.1 was applied and the averaged energy across the domain was selected as the convergence indicator. After that, the convergence is double-checked via examination of continuously updated statistics, that is, mean and standard deviation σ computed in a selected set of points:

$$\sigma = \frac{1}{N_p} \sum_{p=1}^{N_p} \sqrt{\frac{1}{N_t - 1} \sum_{t=1}^{N_t} (U_{p,t} - \bar{U}_{p,t})^2}, \quad (6.1)$$

where N_p and N_t represents the total number of selected points and time steps respectively, U_{p,N_t} denotes the monitored quantity, such flow velocity, at position p and time step t , and $\bar{U}_{p,t}$ is the time-averaged value:

$$\bar{U}_{p,t} = \frac{1}{t} \sum_{tt=1}^t U_{p,tt} = \frac{t-1}{t} \bar{U}_{p,t-1} + \frac{1}{t} U_{p,t} . \quad (6.2)$$

Defining the total time variance as $Var(U_{p,t}) = \frac{1}{N_t-1} \sum_{t=1}^{N_t} (U_{p,t} - \bar{U}_{p,t})^2$, the standard deviation σ can be recursively calculated as

$$\sigma = \frac{1}{N_p} \sum_{p=1}^{N_p} \sqrt{\frac{1}{N_t-1} [(N_t-2)Var(U_{p,N_t-1}) + (U_{p,t} - \bar{U}_{p,t})^2]} , \quad (6.3)$$

which reduces the computational costs in recording the convergence data.

The symmetry plane $y = L/2$ is selected for convergence monitoring in this case. Recording of these statistics commences after the initial unphysical transient stemming from initialisation has passed. This is assumed to be the case after the residual of average energy falls below 10^{-5} . As illustrated in Figure 6.9, the standard deviations of both U_x and U_z are plotted against computational steps. Both curves start from infinite small values and reach to a constant region after sharp increases. The increase of σ is as expected because the error terms are accumulated from an initial state. Flow statistics can be therefore considered as converged.

A comparison of computational resources for uniform and refined grids are summarised in Table. (6.3). All simulations are carried out on a computational node with 48 processors, using shared memory parallelism. The number of total active cells in Case A with only 1 level refinement simulation is around 4.7 million, which is considerably less than 8.1 million lattice points of the uniform case with equal finest resolution ($N = 200$). On the other hand, Case B has a

Table 6.3.: Computational costs for the lid-driven cavity case, scaled to the “Uniform $N = 100$ ” case, which has $\Delta t = 2 \times 10^{-4}$, 1 million total active cells and a total convergence time of 3.7 hours

	physical Δt	total active cells	total time to convergence
Uniform $N = 100$	1	1	1
Uniform $N = 200$	0.5	8.1	6.9
Case A	1	4.7	3.9
Case B	1	10.9	6.9

higher finest grid resolution ($N = 400$), but takes similar time to converge when compared with the uniform grid with resolution $N = 200$. With this refinement scheme, for the same cost of a medium accuracy solution, a much better accuracy is achieved close to the walls, as will be shown in the results.

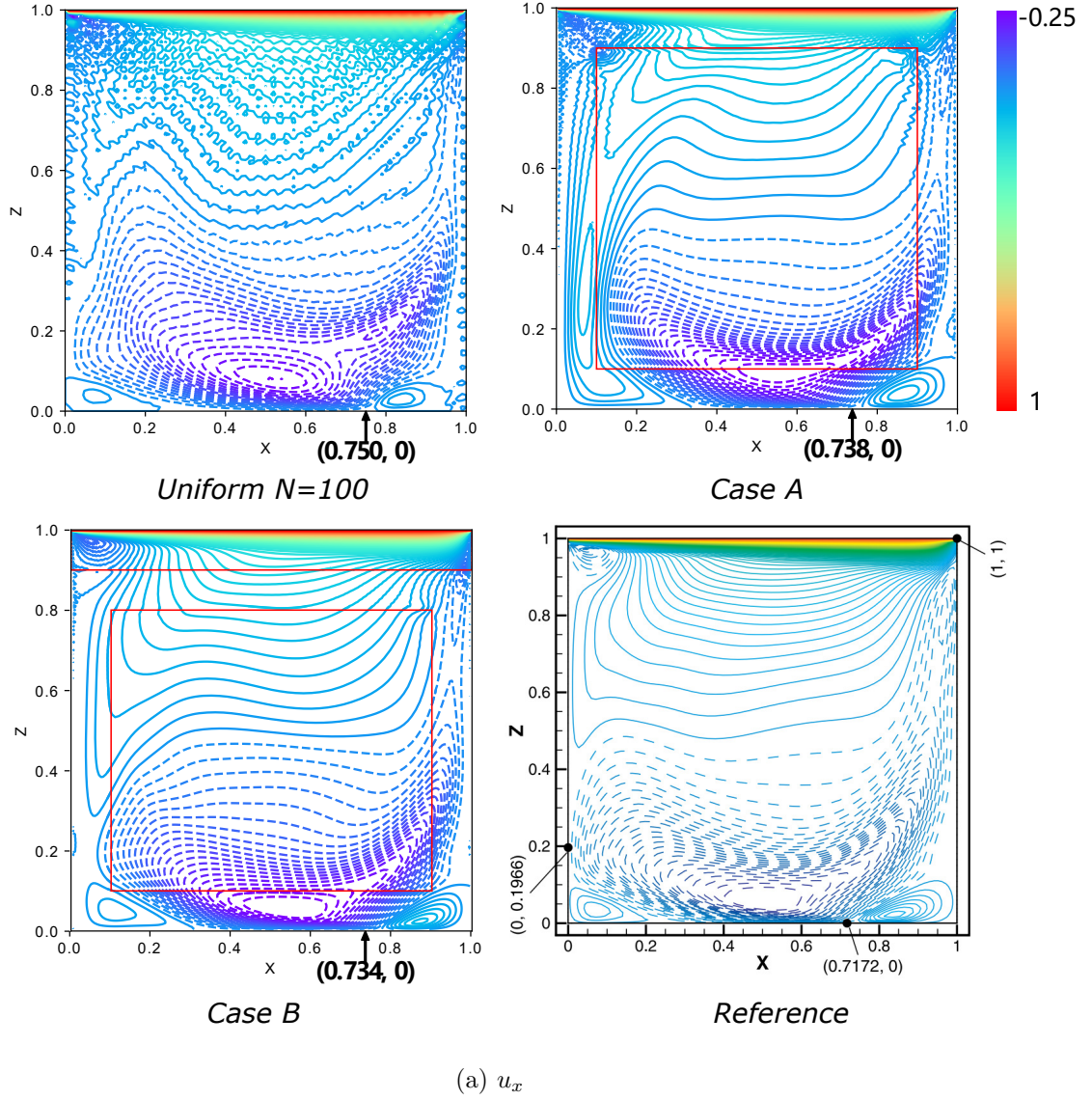


Figure 6.10.: Time-averaged contours of velocity (u_x and u_z) on the plane of symmetry ($y=0.5$) of uniform $N=100$, Case A with 1 level refinement, Case B with 2 level refinement and reference [105]

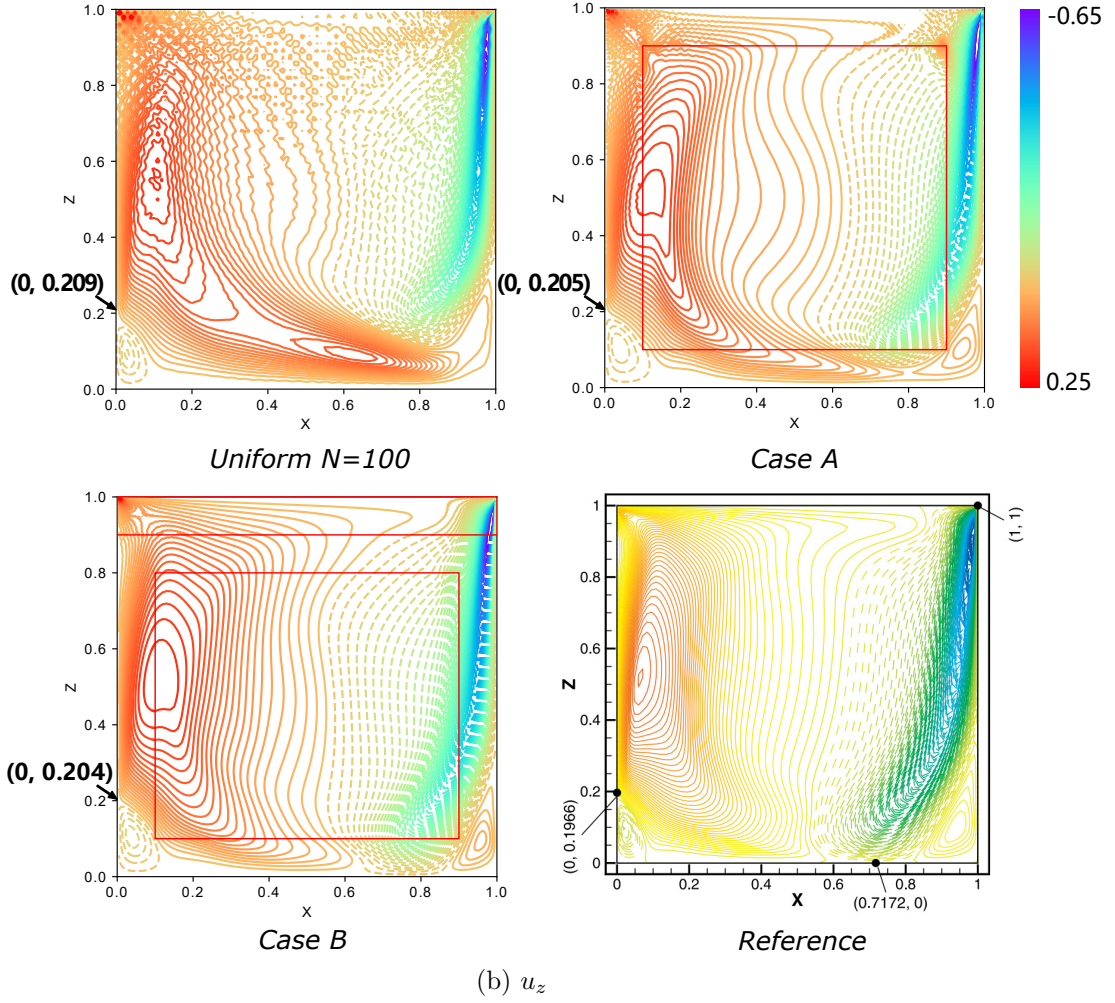


Figure 6.10.: Time-averaged contours of velocity (u_x and u_z) on the plane of symmetry ($y=0.5$) of uniform $N=100$, Case A with 1 level refinement, Case B with 2 level refinement and reference [105] (cont.)

6.2.4. Results

Figure 6.10 shows the time-averaged velocity contours of U_x and U_z of the uniform case with resolution $N = 100$, Case A with 1 level refinement and Case B with 2 level refinement, in comparison with reference results of a large eddy simulation based on NS equations using a dynamic Smagorinsky subgrid model [105]. The refinement interfaces are marked by red lines and the positive and negative values of velocity are denoted by solid lines and dotted lines.

The primary vortex and three secondary vortices are captured and the position of the impingement points improves with increasing levels of grid refinement.

Another remarkable difference is the smoothness and shape of velocity con-

tours, which is largely influenced by the discontinuity of boundary conditions around the top edge. The uniform grid exhibited strong fluctuations compared to the other two cases. Case A also showed fluctuations with a higher grid resolution around the corner, which indicates that a fine grid resolution can reduce the fluctuations, but not dramatically. In contrast, the refinement layer around the top boundary in Case B “blocked” the corner fluctuations from streaming into the bulk flow. This can be attributed to the application of regularised coupling on the refinement interface which helps stabilise the simulation as it removes the higher order errors in the non-equilibrium populations. The comparison of simulation with and without regularised coupling will be discussed later in this section.

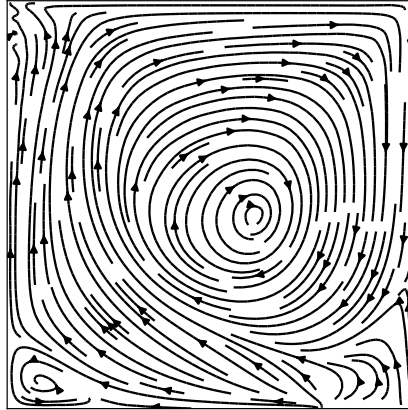
In addition, the secondary vortices on the bottom left and bottom right are not accurately captured in the u_z contour of the uniform $N=100$ case and u_x contour of Case A. In comparison, when the top refinement is employed, these vortices are well resolved, and the top left secondary vortex is only captured in Case B.

To investigate the flow continuity across the refinement interface, the transient streamline contours are illustrated in Figure 6.11. The streamline directions are not affected when passing through refinement interfaces. Neither blockage nor distortion to the velocity field is detected, indicating a continuous flow across the interface.

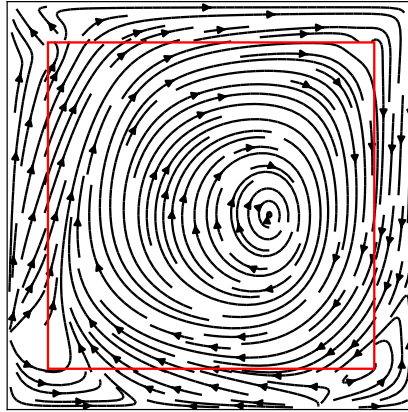
The time-averaged velocity profile of U_x along z -axis at $x = L/2$ and U_z along x -axis at $z = L/2$ on the plane of symmetry ($y = L/2$) is shown in Figure 6.12. Results of two refinement cases are compared with two uniform simulations at resolution $N = 100$ and 200 and experiments by Prasad and Koseff [106]. The deployment of first level of refinement around wall boundaries retains a similar accuracy level in the boundary layer as the $N = 200$ uniform grid. The coarse resolution $N = 100$ matches the measurements better in the bulk flow region than the fine grid ($N = 200$). This variation could be explained by the fact that the result accuracy of the BGK operator is sensitive to τ [35, 55], while according to Eq. 2.84, the value of τ depends on the grid resolution in order to preserve same physical viscosity. As analysed in Section 2.5.2, the BGK scheme produces a truncation error scaling as $(\tau - \frac{1}{2})^2$, while $(\tau - \frac{1}{2})$ is proportional to the grid resolution N with acoustic scaling ($\Delta t_1/\Delta t_2 \propto \Delta x_1/\Delta x_2$) in Eq. 2.94. Possible solutions to this problem includes using τ -independent collision operators such as MRT and switching to diffusive scaling.

This τ -dependency of the BGK operator is further investigated. As discussed in Section 2.5, with constant input Reynolds number and grid resolution, the

Uniform $N=100$



Case A



Case B

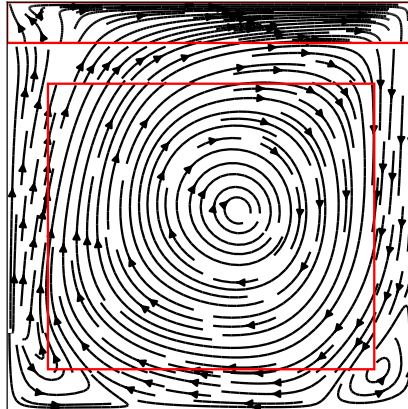


Figure 6.11.: Instantaneous streamline contours of Uniform $N = 100$, Case A and Case B on the plane of symmetry ($y=L/2$)

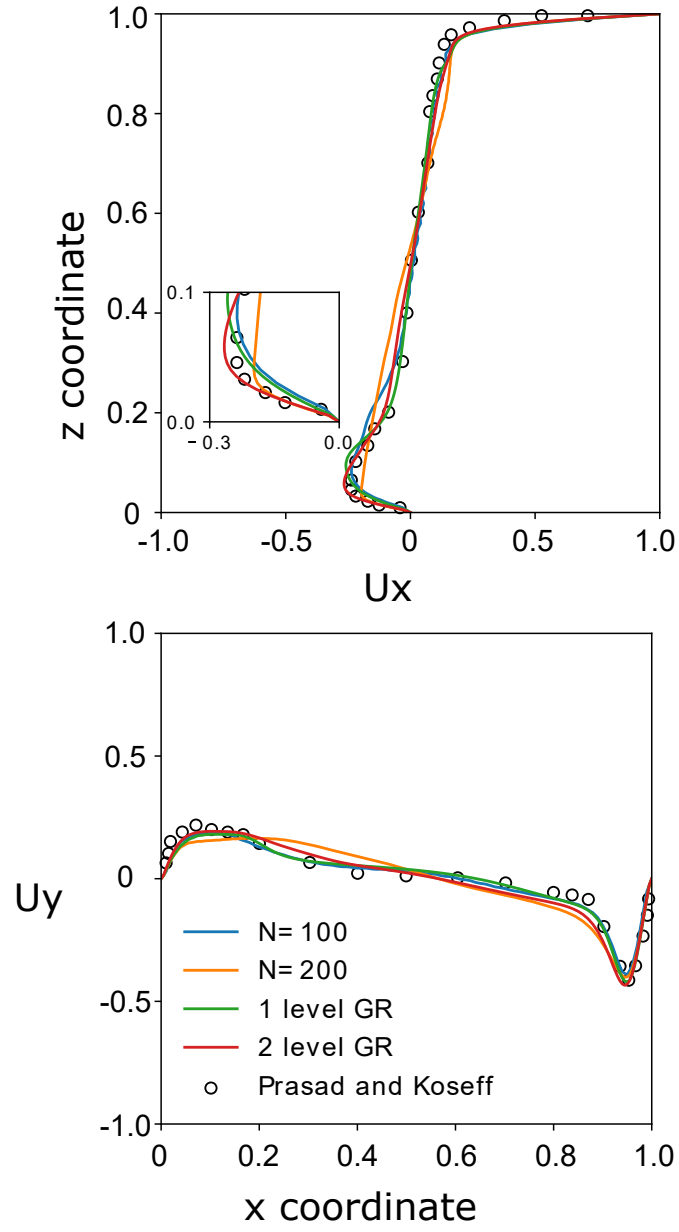
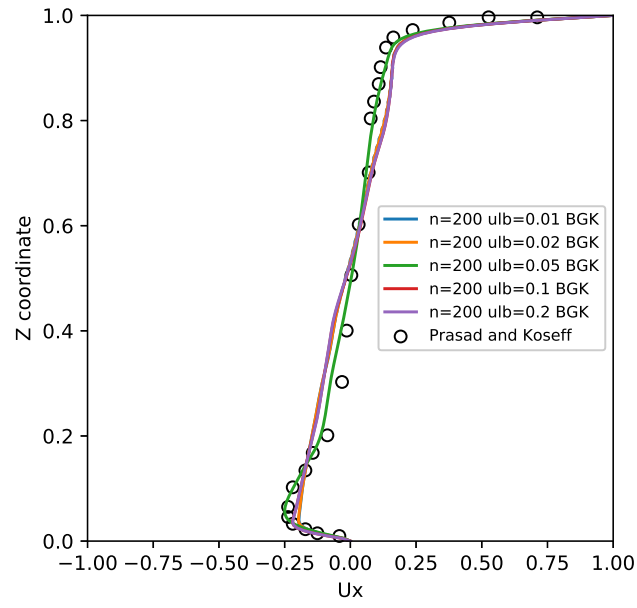
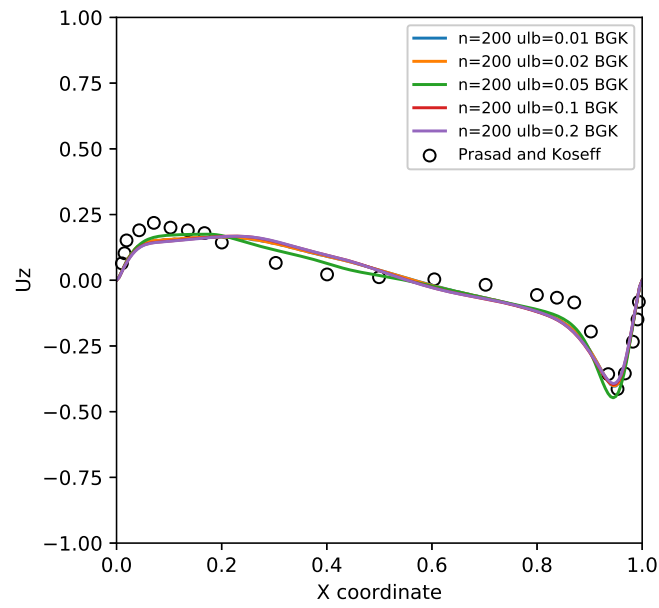


Figure 6.12.: Time-averaged U_x along z -axis at $x = L/2$ and U_z along x -axis at $z = L/2$ on $(y = L/2)$ plane



(a) u_x



(b) u_z

Figure 6.13.: Time-averaged U_x and U_z profiles of uniform grid with resolution $N = 200$ of different lattice velocity

lattice velocity u_{lb} is proportional to $(\tau - 0.5)$, which is

$$Re = Re_g N = \frac{u_{lb} N}{(\tau - 0.5)/3} . \quad (6.4)$$

Therefore varying the input τ can be seen as changing the time spacing, which will alter u_{lb} as a result. Since u_{lb} is an important parameter for evaluating the stability condition as discussed in Section 2.5.2, different simulation instances are marked with u_{lb} . In Figure 6.13 the simulation results of a uniform grid with resolution $N = 200$ with relaxation time τ are compared. Among these variants, the simulation at $u_{lb} = 0.05$ (corresponds to $\tau = 0.509375$) gives the most accurate result. It is interesting to find that the result of the lowest u_{lb} (i.e. finest time step sizing) was not the most accurate one. A possible explanation is that although the lowest u_{lb} produces smallest truncation error, it corresponds to smallest lattice viscosity (or relaxation time), which might not sufficiently damp out effects of the background fluctuations. In this case, a refinement interface with regularised coupling can be applied to help filter out these undesired errors, as indicated by above discussions and following comparison.

To this end, the performance of the regularised coupling is evaluated by comparing the simulation results with and without regularisation during refinement

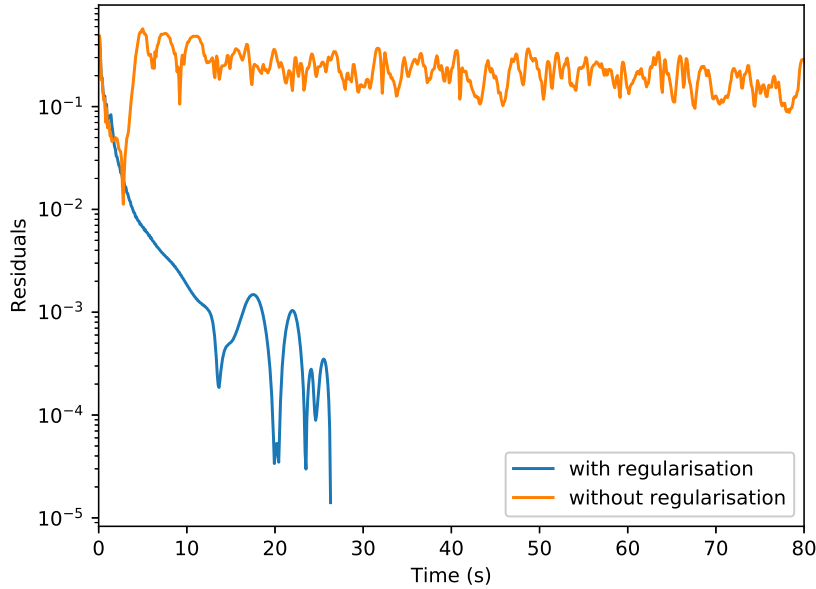


Figure 6.14.: Convergence comparison of simulations with and without regularised coupling

coupling. All simulation parameters were kept same. The simple averaging restriction method described in Section 4.1 was used in the absence of the regularised coupling. The convergence is compared, using the standard deviation of the averaged total energy throughout the domain as the residual, and is illustrated in Figure 6.14. It can be observed that without regularised coupling, the convergence takes much longer, which can be explained by the fluctuations in the domain can not be efficiently damped. This comparison demonstrated the capability of the regularised coupling in reducing the fluctuations across the refinement interface and thus improving numerical stability.

Table 6.4.: Grid Reynolds number at each resolution

	$N = 100$	$N = 200$	$N = 400$
Grid Reynolds number Re_g	32	16	8

Overall, from the comparison between refined grids and uniform grids it can be observed that with grid refinement good accuracy is obtained both in the near-boundary region and bulk flow area. Again, the grid Reynolds number was decreased by refining the grid resolution, as shown in Table 6.4. The uniform grid with $N = 100$ is close to the unstable margin. This lid-driven cavity case showed that the grid refinement with regularised coupling can not only help stabilise the simulation, but also increase numerical accuracy by filtering out the background fluctuations.

6.3. Summary

The chapter was devoted to the validation of the 3D grid refinement method in OpenLB. To progress towards the modelling of separated flows over aerofoils, which involves boundary layer transition and separation on a curved geometry, two thoroughly investigated benchmarks were selected: vortex shedding flow around and behind a circular cylinder with wake transition at a low Re regime, and lid-driven flow inside a cubic cavity with multiple interacting vortical structures. The cylinder case evaluates the performance of LBM in dealing with curved geometry and turbulence transition, while the vortical structure in the lid-driven cavity provides insight into the resolution requirements of the LBM in simulating turbulent flows.

The vortex shedding benchmark at $Re=100, 200$, and 300 reproduced the three-dimensionality of the vortex structure in the cylinder’s wake transition. Two instability modes with different spanwise wavelengths, namely Mode A and Mode B, were accurately captured with the refined grid. Both time-averaged

pressure coefficient around the cylinder and time-averaged velocity profiles in the wake were in good agreement with literature data.

The case of lid-driven flow at higher Reynolds number assessed the efficiency and continuity performance of the grid refinement in comparison to the uniform grid. The refined grid is shown to be much more efficient, as it not only decreases the number of total active cells, but also enlarges the time step size on the coarse grid. Using grid refinement, the total active cells and the total time to convergence are reduced by a half while maintaining accuracy.

The refinement applied in the lid-driven case proves its ability in improving simulation stability, thanks to the regularised coupling method, which filters out the high order errors in the non-equilibrium populations.

7. Application on High Reynolds Number Simulations

This chapter shows the application of the grid refinement method to an open shallow cavity, and evaluates the performance of the LBM with regularised coupling grid refinement at relatively high Reynolds number. The time-averaged velocity profiles, as well as the second-order statistics, are investigated and compared with results of NS based solvers. It is shown that this implementation is capable of resolving turbulent flows accurately in an efficient manner.

7.1. Problem Specification

The cavity case is based on the experiments from Grace et al. [107] with detailed LASER Doppler velocimetry (LDV) and hot-wire measurements. Scale resolving Navier-Stokes based numerical results are also available from Jefferson-Loveday [108], including the scale adaptive simulation (SAS) model and the new enhanced variant of Spalart-Allmaras delayed detached eddy simulation (SA-DDES). The comparison between the numerical results of the LBM with the Smagorinsky subgrid scale model and results from Navier-Stokes based solvers is helpful to evaluate the performance of the LBM with grid refinement at high Reynolds

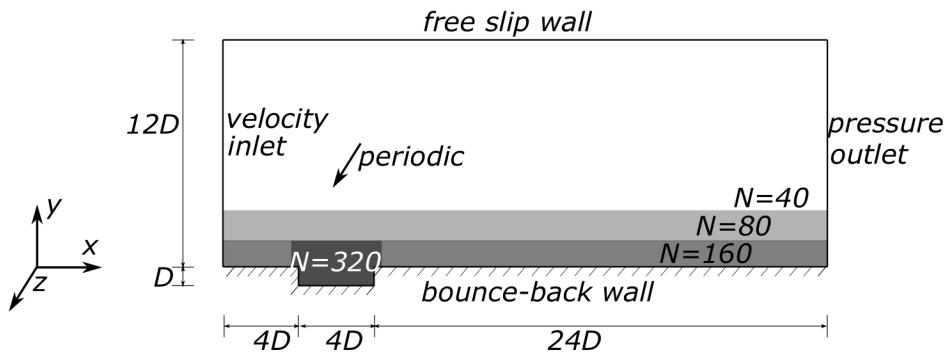


Figure 7.1.: Domain geometry and refinement layout of an open shallow cavity

numbers.

The cavity geometry is shown in Figure 7.1. The cavity has a length-to-depth ratio of 4. This geometry is defined as an open cavity, in contrast to the closed cavity where the reattachment occurs on the floor of the cavity before separating again. The spanwise length of the cavity (in the z -direction) is three times the streamwise length ($12D$). The Reynolds number is set to 50,000 based on the length of the cavity. A laminar upstream inlet boundary layer forms upstream of the cavity leading edge.

7.2. Simulation Configuration

The simulation parameters for all refinement levels are summarised in Table 7.1. The characteristic length is selected based on the cavity depth D and the cavity length L_c is $4D$.

A laminar velocity boundary, prescribed with the velocity profile of the Blasius solution in the wall boundary layer, is provided at the domain inlet. For the outlet boundary, an interpolated pressure boundary condition [109] is applied, in which the local velocity gradients are calculated from neighbouring nodes to recover the missing non-equilibrium populations on boundary nodes. The top boundary is slip-free while the bottom, including the cavity wall, is defined as simple bounce-back. Periodic boundary condition along the transverse direction is used to minimise the effect of cavity's spanwise length.

Refined grids are applied around the bottom wall and the cavity as shown in Figure 7.1. As in the previous simulations, the acoustic scaling is applied on each refinement level to ensure constant speed of sound. The coarsest resolution is $N = 40$ based on the cavity length, while the finest resolution is $N = 160$ around the bottom non-slip wall and $N = 320$ over the cavity region. The total active cell number is around 93 million, with the first off-wall grid spacing to be around $y^+ \approx 4.5$ at the finest level (Note that the physical wall position in the bounce-back boundary condition is halfway between two grid nodes). Compared to the FV simulations of which the average first off-wall grid spacing is $y_{FV}^+ < 1$, the streamwise grid spacing ($\Delta x^+ \approx 9$) and cross-stream grid spacing ($\Delta z^+ \approx 9$) are much less than the reference value ($\Delta x_{FV}^+ \approx 60$, $\Delta z_{FV}^+ \approx 90$). The limitation of the uniform grid used in LBM cannot be thoroughly avoided by the grid refinement method since the refined grid is inherently uniform. Nevertheless, the refinement position and layout can be further manipulated and optimised to improve the resolution near wall and reduce total grid size simultaneously.

A BGK collision operator coupled with a Smagorinsky subgrid model is used

Table 7.1.: Summary of simulation parameters of the high Reynolds case at different refinement levels

			Base	Refinement 1	Refinement 2	Refinement 3
Reynolds number	Re				50,000	
Characteristic length	l_0				$0.1m$	
Grid resolution	N		40	80	160	320
Length conversion factor	$C_l = \Delta x$	l_0/N	$2.5 \times 10^{-3} m$	$1.25 \times 10^{-3} m$	$6.25 \times 10^{-4} m$	$3.125 \times 10^{-4} m$
Characteristic velocity	u_0				$1m/s$	
Lattice velocity	u_{lb}				0.05	
Velocity conversion factor	C_u	u_0/u_{lb}			$20m/s$	
Time conversion factor	$C_t = \Delta t$	C_l/C_u	$1.25 \times 10^{-4} s$	$6.25 \times 10^{-5} s$	$3.125 \times 10^{-5} s$	$1.5625 \times 10^{-5} s$
Characteristic density	ρ_0				$1kg/m^3$	
Lattice density	ρ_{lb}				1	
Density conversion factor	ρ_t	ρ_0/ρ_{lb}			$1kg/m^3$	
Characteristic kinematic viscosity	ν_0	$u_0 l_0 / Re$			$2 \times 10^{-6} m^2/s$	
Lattice kinematic viscosity	ν_{lb}	$\nu_0 C_t / C_l^2$	4×10^{-5}	8×10^{-5}	1.6×10^{-4}	3.2×10^{-4}
Relaxation time	τ	$3\nu_{lb} + 0.5$	0.50012	0.50024	0.50048	0.50096
Grid Reynolds number	Re_g	Re/N	1250	625	312.5	156.25
Smagorinsky constant	C_s				0.25	

in this simulation. There is thus an additional free parameter, Smagorinsky constant C_s as shown in Eq. 3.15, whose selection should be carefully considered. The choice of C_s is independent of grid refinement, as the turbulence viscosity in Eq. 3.13 already contains a scaling on Δx . In the following, $C_s = 0.25$ is applied to all refinement levels. Note although $C_s = 0.25$ is higher than standard practise in NS based simulations [110], in LBM those conclusions may not necessarily apply, since total viscosity is determined in conjunction with the relaxation time, which is selected to be as small as possible to minimise the required grid resolution.

The grid Reynolds number, introduced in Section 2.5.2, is also calculated on each refinement level. Compared to the BGK collision operator, the Smagorinsky BGK allows for a higher stability margin for Re_g . However, if Re_g is set too high, for instance using a uniform grid with resolution $N = 40$ so that $Re_g = 1250$, the flow around the cavity will become unstable and the simulation will diverge shortly after the initialisation. This indicates that grid refinement method is necessary for applying the LBM to high Reynolds number simulations.

7.3. Results

The convergence is checked using Eq. 6.1. The instantaneous velocity is recorded on a series of spanwise nodes in the middle of the cavity. The standard deviation σ is then calculated and plotted, as shown in Figure 7.2. As explained before, σ

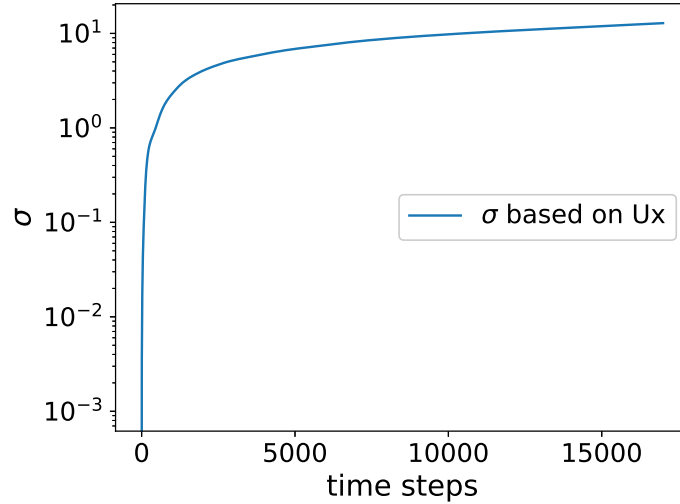


Figure 7.2.: Measured standard deviation of flow velocity versus computational steps after convergence in the open cavity

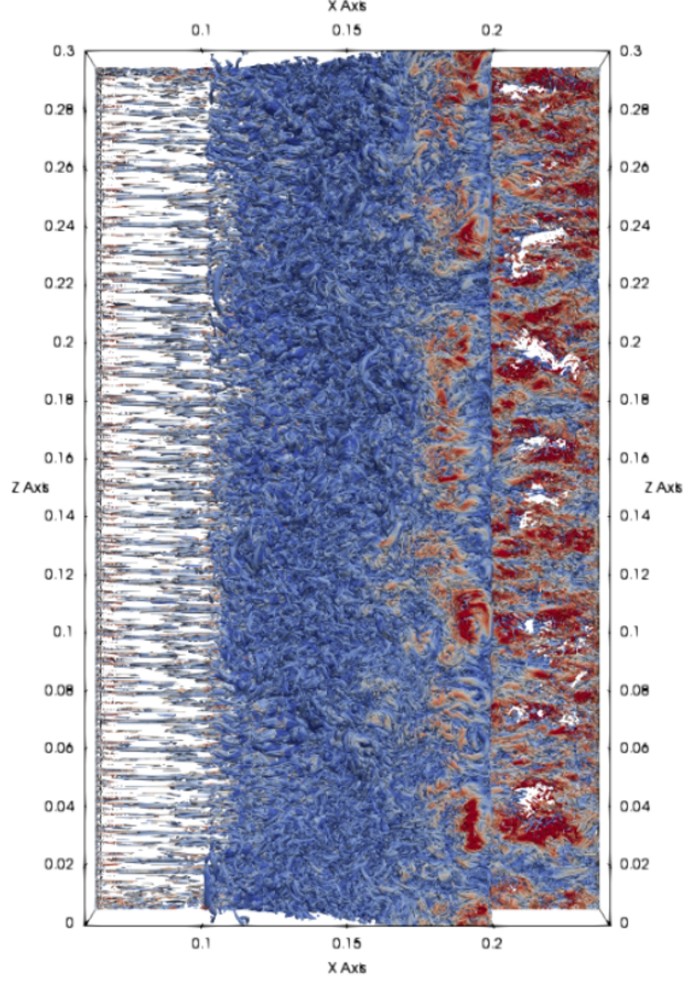


Figure 7.3.: Snapshot of the instantaneous vorticity contours around the cavity on X-Z plane. The cavity locates in $x \in [0.1, 0.2]$ and $z \in [0, 0.3]$.

starts from 0 and is expected to asymptotically approach some constant value. This indicates the flow statistics can be considered as converged.

The instantaneous flow around the cavity is illustrated by a snapshot of the vorticity contours shown in Figure 7.3. At $Re = 50,000$ the upstream boundary layer is still laminar but the cavity flow and its downstream exhibit strong turbulence. To capture and compare the turbulence statistics, time averaging of the flow in both streamwise and spanwise direction is recommended.

7.3.1. Comparison of Time-Averaged Velocity Profiles

Defining one time period $T = L_c/U_0$ as the time of the flow with mean velocity U_{inf} passing the cavity length L_c , data in sufficiently long time period $16T$ were recorded to ensure accurate time-averaging. Flow velocity profiles at x - y sections

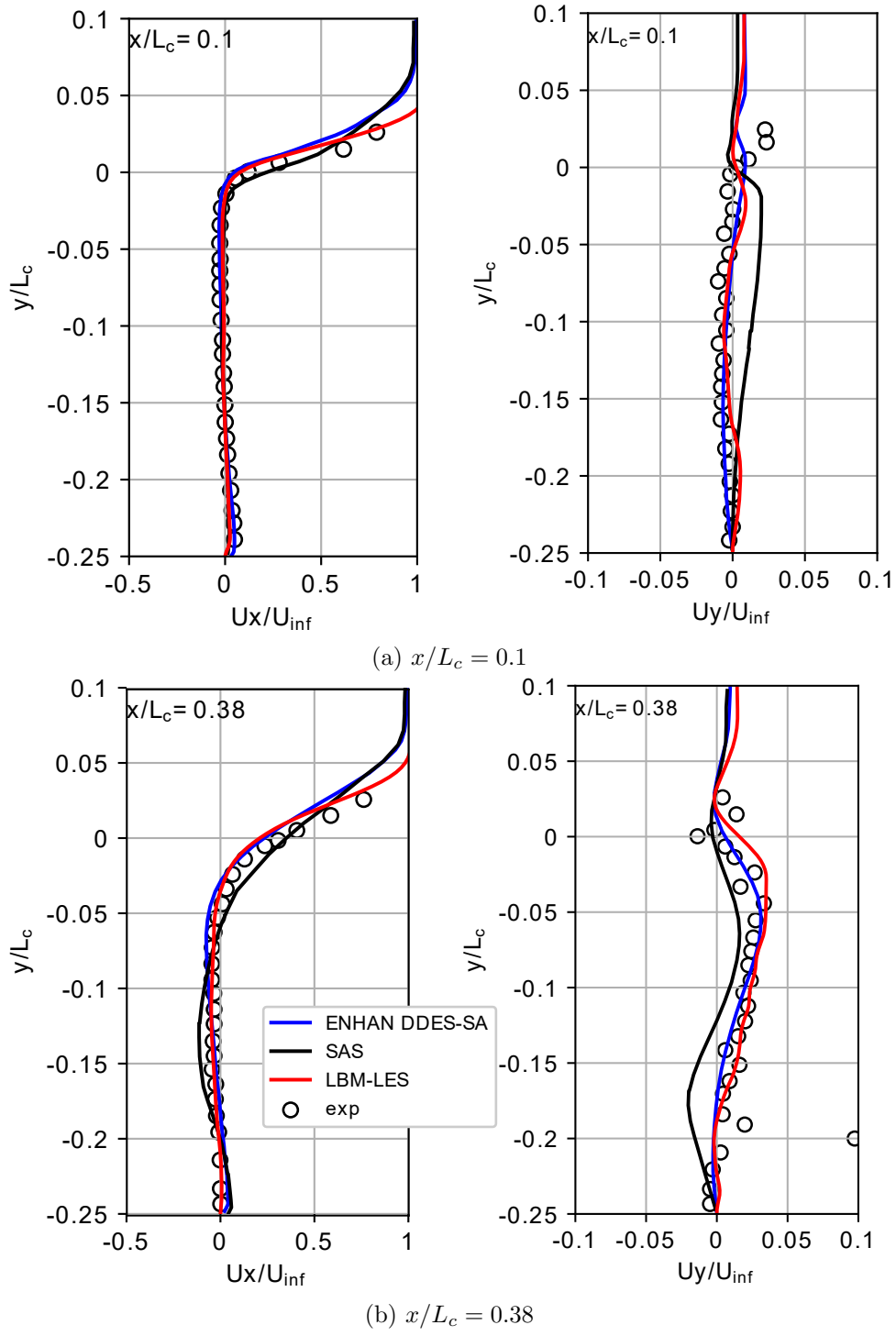


Figure 7.4.: Comparison of time-averaged x- (left) and y-component (right) velocity profiles at three different streamwise locations in the cavity

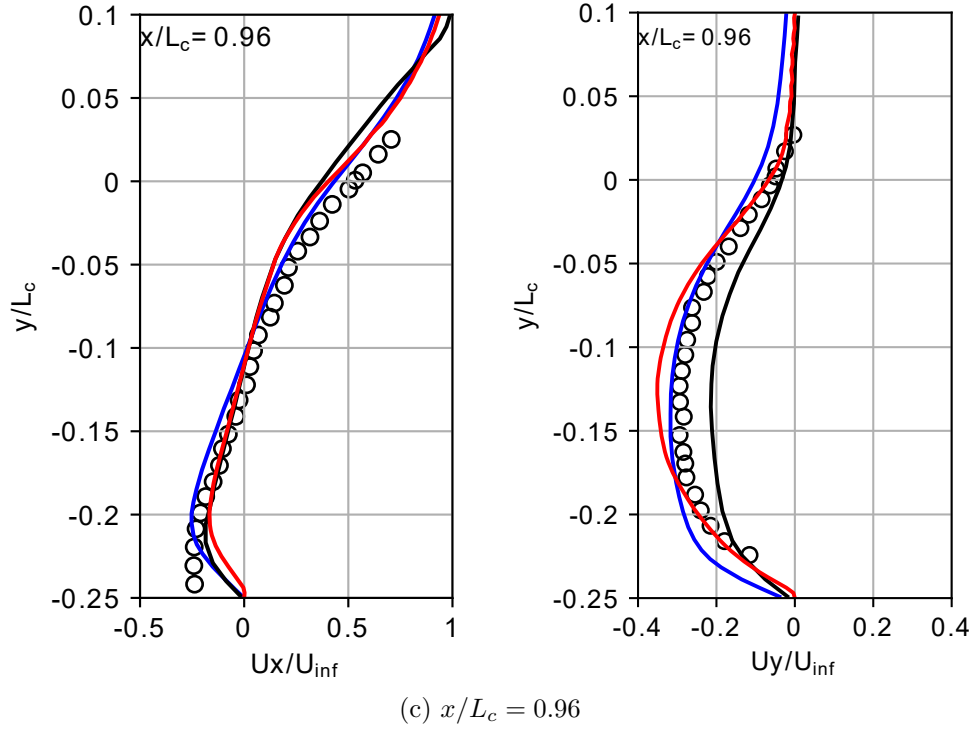


Figure 7.4.: Comparison of time-averaged x- (left) and y-component (right) velocity profiles at three different streamwise locations in the cavity (cont.)

of the cavity on $z/D = \{0, 2, 4, 6, 8, 10\}$ were recorded and averaged to account for the spanwise variance.

Time-averaged velocity profiles at three-different streamwise locations are compared with experiment and numerical data, as illustrated in Figure 7.4. The Y axis represents the cavity position in depth, with $y/L_c = -0.25$ to be the bottom and $y/L_c = 0$ to be the ceiling of the cavity. The velocity magnitude is normalised according to the far-field velocity. Both x- and y- component of the velocity profiles showed good agreement with reference values. From the U_x plots it can be observed that LBM with Smagorinsky BGK model captures the mixing flow of the incoming boundary layer flow and cavity inner flow more accurately than other results.

The time-averaged streamline contours in the cavity are plotted in Figure 7.5 for comparison. The shape and impingement position of the primary vortex are accurately captured using LBM-LES.

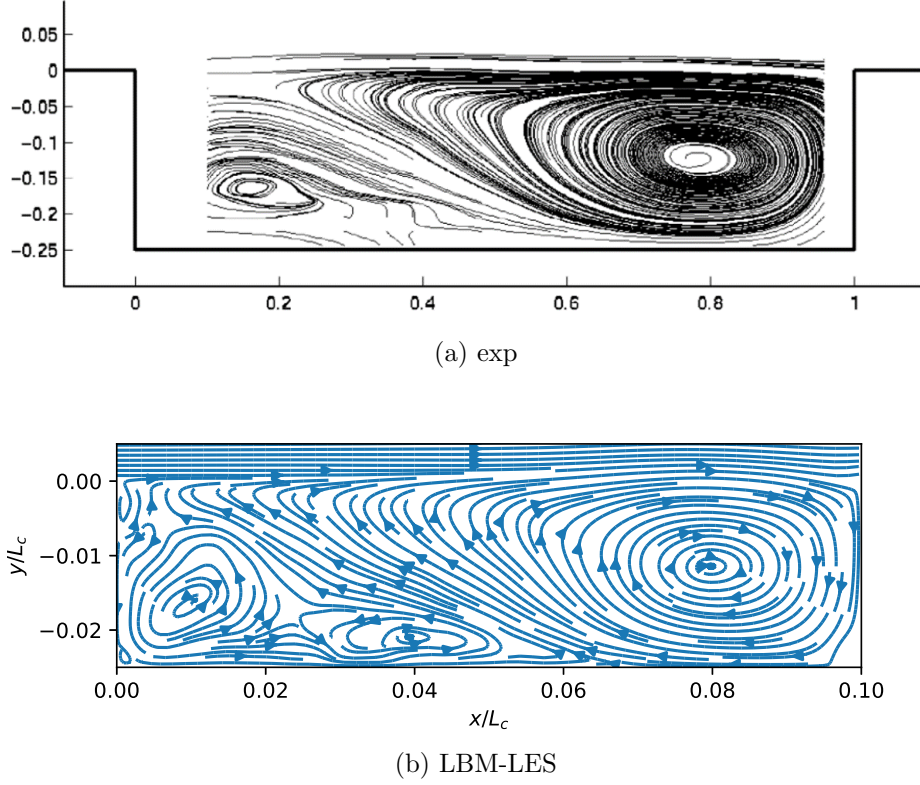


Figure 7.5.: Comparison of the time-averaged streamline contours from experimental and LBM-LES results

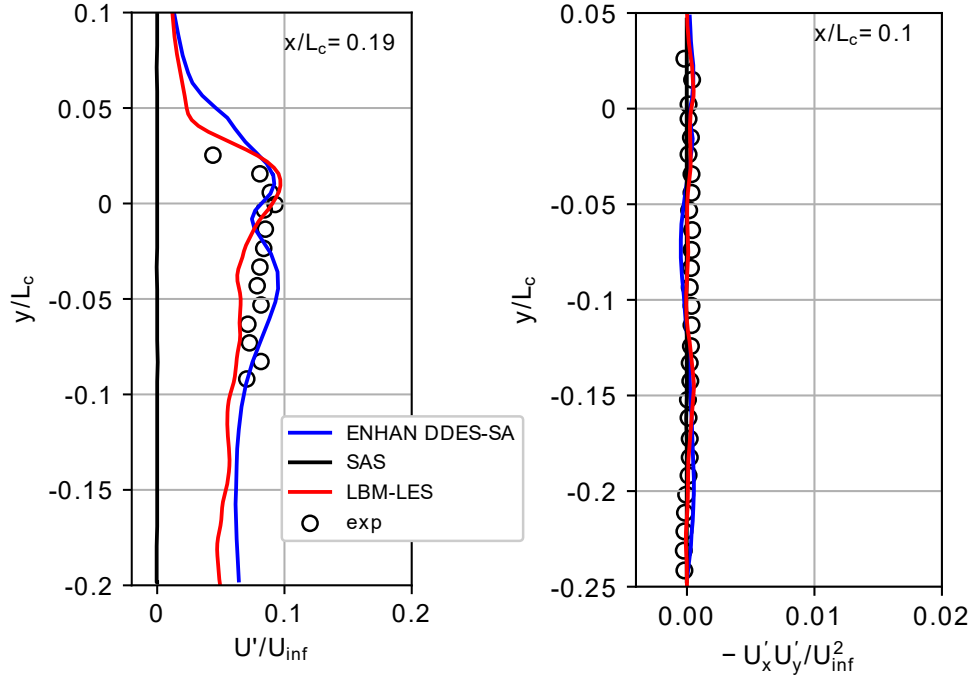
7.3.2. Comparison of Second-Order Statistics

The fluctuation velocity U' , as introduced in Section 2.1.2, describes the turbulence intensity. As explained in Section 7.3.1, flow velocity profiles of time period $16T$ were recorded and the fluctuation velocity can be computed from

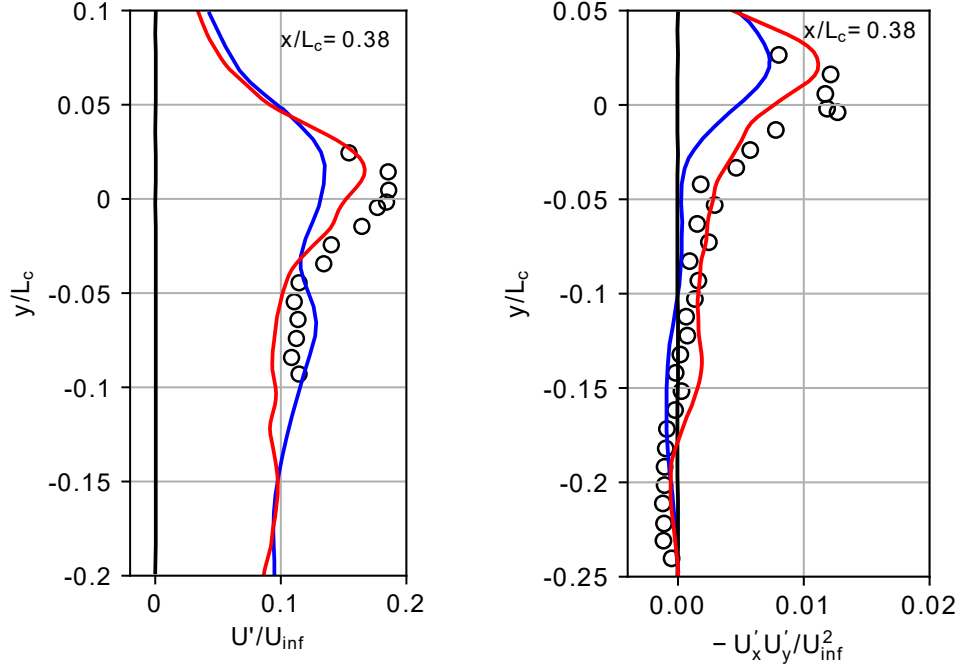
$$U'(t) = U(t) - \bar{U} , \quad (7.1)$$

where $\bar{U}$ denotes the time-averaged velocity. The magnitude of the fluctuation velocity $|U'(t)|$ is further averaged in time and spanwise direction. After non-dimensionalising using the mean flow velocity U_{inf} , the x -component of the time-averaged fluctuation velocity profiles are plotted and compared in Figure 7.6.

The downstream turbulence near the trailing edge of the cavity $x/L_c = 0.94$ is stronger than that near the leading edge at $x/L_c = 0.19$, whereas near the mid-span at $x/L_c = 0.38$, a peak around $y/L_c = 0$ was captured, indicating the effect of upstream boundary layer on the cavity flow. The LBM model follows the experimental data more closely compared to the other models, especially at



(a) $x/L_c = 0.19$ (left) and $x/L_c = 0.1$ (right)



(b) $x/L_c = 0.38$

Figure 7.6.: Comparison of time-averaged velocity fluctuation (left) and Reynolds shear stress (right) at different streamwise locations in the cavity

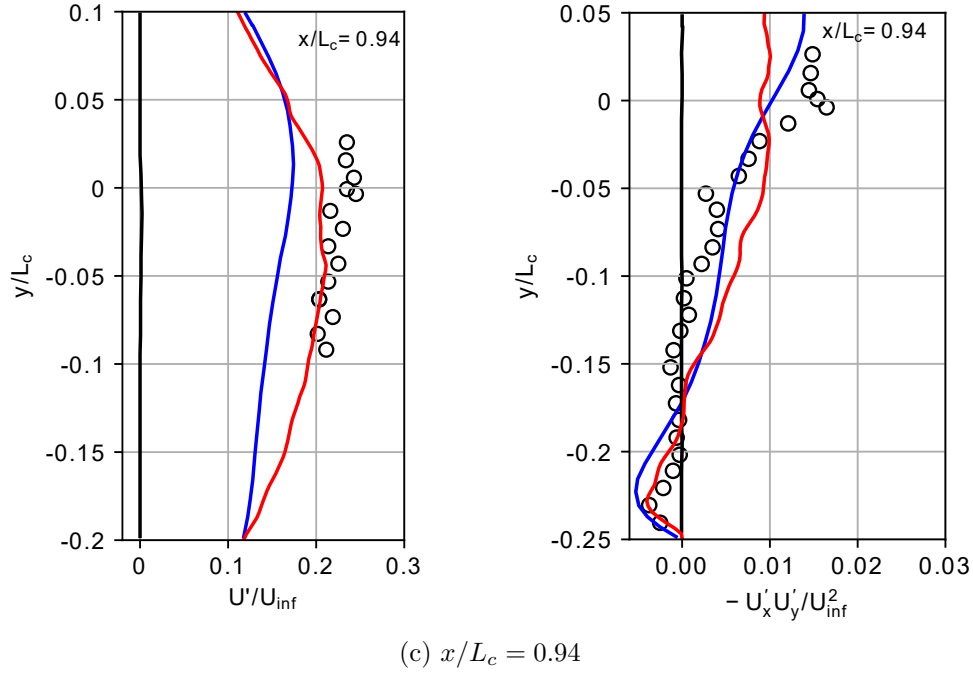


Figure 7.6.: Comparison of time-averaged velocity fluctuation (left) and Reynolds shear stress (right) at different streamwise locations in the cavity (cont.)

the peak value of the turbulent intensity.

Similar to the time-averaged fluctuation velocity, the time-averaged Reynolds stresses $-\overline{U'_x U'_y}$ is calculated and non-dimensionalised. As plotted in Figure 7.6, the profile at $x/L_c = 0.94$ exhibits a sign change in the Reynolds stress, indicating the effect of the primary vortex which was shown before in Figure 7.5. The peak in the Reynolds stress at $x/L_c = 0.38$ induced by the shear layer around $y/L_c = 0$ is also accurately captured. Among all results the LBM model shows good agreement with experimental data, especially in the shear layer near the cavity opening.

7.3.3. Influence of the C_s Coefficient

As previously mentioned, the selection of C_s has a significant impact on the simulation result. In Figure 7.7, a smaller value $C_s = 0.1$ is chosen and compared with the original $C_s = 0.25$. Results with smaller C_s value show higher overshooting, which can be explained as a smaller C_s value shortens the filter length, such that some small eddies are neither modelled correctly nor resolved because of the grid spacing limit. Since the first off-wall grid spacing y^+ in this simulation is greater than 1, a relatively large C_s value beyond the recommended

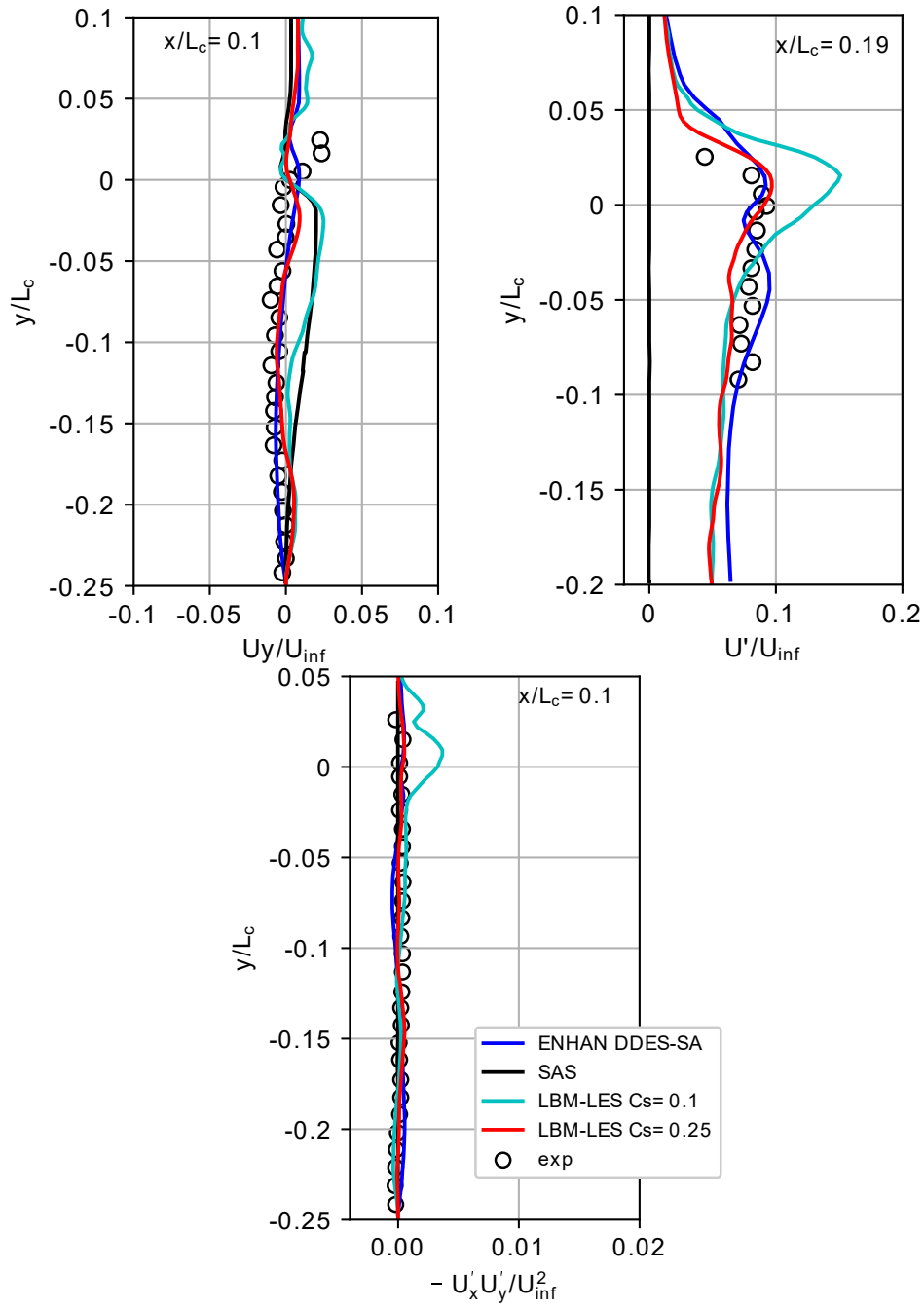


Figure 7.7.: Comparison of flow statistics with Smagorinsky constants $C_s = 0.1$ and $C_s = 0.25$

range $0.1 - 0.16$ for NS-LES [111] helps to account for the dissipation effects from the unresolved small eddies in the near-wall region.

In addition, the findings by Hou et al. [112] indicates that the Smagorinsky constant used at high Reynolds number in the LBM subgrid simulations is higher than the values in traditional NS schemes to avoid negative particle populations. The selection of proper Smagorinsky constant, which strongly depends on specific geometry [112], problem setup [67], and flows of interest [113], is important and should be carefully considered with empirical corrections [114].

7.4. Summary

This chapter applied the grid refinement method to a high turbulent flow simulations. The flow inside an open shallow cavity at $Re = 50,000$ with a laminar upstream boundary layer was investigated. Three levels of refinement were applied with the finest resolution assigned around the cavity. The Smagorinsky subgrid scale model was applied to reduce computational cost.

The results in terms of time-averaged velocity profiles and second-order statistics are compared against NS based solvers. In both cases the present method predicted the flow statistics accurately, especially the turbulent fluctuations. The selection of a proper C_s coefficient in the subgrid scale model is shown to have a large impact on the results.

This successful application indicates that the current implementation is capable of performing LBM simulations at high Reynolds number with efficient use of computational resources. The grid refinement method greatly reduces the required computational costs, while the regularised coupling method is helpful to increase the numerical stability around the refinement interface.

8. Conclusions and Future Work

8.1. Conclusions

The original LBM discretises the computational domain uniformly because of the discrete velocities structure. This layout simplifies the meshing but imposes severe demands on computational resources. For the calculation of industrial flows, the required grid resolution to directly resolve the turbulent structures and eddies grows exponentially with the Reynolds number. However, in most cases only some regions need to be refined to resolve the relevant features.

This thesis described the methodology and implementation of a three-dimensional vertex-centred hierarchical grid refinement with regularised coupling in OpenLB, in an effort to extend the range of applicability of the lattice Boltzmann method to high Reynolds numbers applications. Referring back to the objectives listed in Section 1.2, the main conclusions and achievements of the present work are summarised here.

1. *A three-dimensional grid refinement with the regularised coupling method has been developed and implemented in OpenLB. This presents an important step towards the use of the solver to aerospace applications.*

The three-dimensional grid refinement method is based on the vertex-centred hierarchical grid layout, in which the simplicity and locality of the LBM are mostly retained. The regularised procedure, which recovers the non-equilibrium populations from local stress field, provides a way to further increase the stability on the refinement interface. By exactly preserving aerodynamics across the refinement interface, this regularised coupling plays a significant role in efficiently applying the LBM to high Reynolds simulations. Implementation of the method in the open-source framework OpenLB maximises the value of the research outcomes for the wider community, as it enables the study of an extended range of flow regimes with confidence.

2. *The grid refinement method has been validated against two benchmark test cases. Its accuracy and efficiency performance have been assessed.*

Two thoroughly investigated benchmarks were selected: vortex shedding flow around and behind a circular cylinder with wake transition at a low Re regime, and lid-driven flow inside a cubic cavity with multiple interacting vortical structures. Flow characteristics were captured accurately compared with literature data, while significant computational cost reduction against uniform grid was demonstrated. The regularised coupling method was proven to reduce the propagation of fluctuations and improve the numerical stability around the refinement interface.

3. *The method was applied to high speed flows at $Re=50,000$ and delivered a good agreement with experimental data and state-of-the-art NS based solvers.*

The simulation of turbulent flow inside the shadow cavity at a high Reynolds number in Chapter 7 demonstrated this conclusion. Applying a Smagorinsky-BGK subgrid scale model, the LBM was shown to accurately capture the turbulent statistics. The grid refinement method is necessary to locally refine the grid, in order to meet the resolution requirements and reduce total computational loads.

8.2. Future Perspectives

Although this novel regularised coupling grid refinement method extends the LBM to higher Reynolds simulations, more research is necessary to extend the LBM solver capabilities to aeronautical problems. To achieve that, grid refinement needs to be automated to follow complex geometries. In addition, the implementation of wall models can reduce the resolution requirements near solid boundaries. Compressible LBM is another important and active research topic, which adopts extended velocity sets and enables LBM simulation of compressible flows. To further advance the state of the art of high Reynolds simulations and aerospace applications in OpenLB, some future perspectives beyond the present work can be recommended as:

1. *Optimise the current parallel performance and implement a distributed memory formulation that can take advantage of multiple node architectures.*

Current implementation of the grid refinement is only applicable to OpenMP, which is used on shared-memory architectures. To apply the grid refinement on distributed-memory nodes, the grid must be divided into small cuboids to be

assigned to different MPI threads. In this implementation, the coarse grid and the refined grid are of two different data types, which means they need to be partitioned separately. The partition of the refinement interface then becomes a problem because the communication between the two grids should be performed after collision and streaming on both grids have finished. A possible way to avoid that problem is to parallelise the refinement interface independently of the belonging grid, at a price of decreasing the parallel efficiency.

2. *Develop and implement an adaptive grid refinement method based on the current implementation.*

Manually defining the refinement position and layout is to some extent inefficient and cumbersome. Take the cavity case in Chapter 7 for instance, even though the total number of grid size rose up to around 100 million, the first off-wall grid spacing ($y^+ \approx 4.5$) is still relatively large when compared to the FV simulation, in which a mesh of 2.7 million nodes allows for ($y^+ < 1$) with the help of non-uniform grid.

Adaptive grid refinement method aims to automatically generate and adjust refinement regions during simulations. An adaptive structure can be implemented based on the current refinement scheme by dynamically refining and coarsening the base grid. The scaled Knudsen number, which was introduced for the Chapman-Enskog analysis in Section 2.4, indicates the scale of actual local grid sizing in comparison to the desired value and can be used as an indicator of necessary refinement levels. A high Kn^* suggests that current grid resolution is coarse with respect to complicated local fluid structures, in which case the refining step should be performed.

3. *Apply the grid refinement with compressible LBM formulations.*

As addressed in Chapter 2, the general LBM recovers weakly-compressible NS equations. To extend the use of OpenLB to aeronautical applications, compressible LBM models which exist in literature [36, 37, 38, 39, 40] ought to be implemented. The grid refinement method is compatible with compressible models, since it does not depend on specific velocity sets or collision dynamics. The grid refinement method described in this thesis will play an important role in reducing computational costs in compressible LBM.

Bibliography

- [1] S. Succi. *The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond*. Numerical Mathematics and Scientific Computation. Clarendon Press, 2001.
- [2] S. S. CHIKATAMARLA et al. “Lattice Boltzmann method for direct numerical simulation of turbulent flows”. In: *Journal of Fluid Mechanics* 656 (2010), 298–308.
- [3] “Eddy resolving simulations in aerospace – Invited paper (Numerical Fluid 2014)”. In: *Applied Mathematics and Computation* 272 (2016). The 9th International Symposium on Numerical Analysis of Fluid Flow and Heat Transfer - Numerical Fluids 2014, pp. 582–592.
- [4] M. A. Leschziner and D. Drikakis. “Turbulence modelling and turbulent-flow computation in aeronautics”. In: *The Aeronautical Journal (1968)* 106.1061 (2002), 349–384.
- [5] T. Astoul et al. “Analysis and reduction of spurious noise generated at grid refinement interfaces with the lattice Boltzmann method”. In: *Journal of Computational Physics* 418 (2020), p. 109645.
- [6] J. Qi, H. Klimach, and S. Roller. “Implementation of the compact interpolation within the octree based Lattice Boltzmann solver Musubi”. In: *Computers & Mathematics with Applications* 78.4 (2019). ICMMES-2015, pp. 1131–1141.
- [7] P. Valero-Lara and J. Jansson. “Multi-domain Grid Refinement for Lattice-Boltzmann Simulations on Heterogeneous Platforms”. In: *2015 IEEE 18th International Conference on Computational Science and Engineering*. 2015, pp. 1–8.
- [8] B. Dorschner et al. “Grid refinement for entropic lattice Boltzmann models”. In: *Phys. Rev. E* 94 (5 2016), p. 053311.
- [9] J. Jacob, O. Malaspinas, and P. Sagaut. “A new hybrid recursive regularised Bhatnagar–Gross–Krook collision model for Lattice Boltzmann method-based large eddy simulation”. In: *Journal of Turbulence* 19.11-12 (2018), pp. 1051–1076.

- [10] J. Slotnick et al. “CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences”. In: *Nnasa/Cr-2014-218178* March (2014).
- [11] M. Krause. *OpenLB – Open Source Lattice Boltzmann Code*. URL: <https://www.openlb.net>.
- [12] V Heuveline and M. J. Krause. “OpenLB: towards an efficient parallel open source library for lattice Boltzmann fluid flow simulations”. In: *International Workshop on State-of-the-Art in Scientific and Parallel Computing* January (2010), Vol. 9.
- [13] V. Heuveline, M. J. Krause, and J. Latt. “Towards a hybrid parallelization of lattice Boltzmann methods”. In: *Computers and Mathematics with Applications* 58.5 (2009), pp. 1071–1080.
- [14] *OpenLB user guide, associated to release 1.3 of the code*. 2019. URL: <https://www.openlb.net/user-guide>.
- [15] M. Krause. *OpenLB Forums*. URL: <https://www.openlb.net/forum/>.
- [16] J. v. Neumann. “Proof of the Quasi-Ergodic Hypothesis”. In: *Proceedings of the National Academy of Sciences* 18.1 (1932), pp. 70–82.
- [17] G. D. Birkhoff. “Proof of the Ergodic Theorem”. In: *Proceedings of the National Academy of Sciences* 17.12 (1931), pp. 656–660.
- [18] C. C. Moore. “Ergodic theorem, ergodic theory, and statistical mechanics”. In: *Proceedings of the National Academy of Sciences* 112.7 (2015), pp. 1907–1911.
- [19] S. Chapman et al. *The Mathematical Theory of Non-uniform Gases: An Account of the Kinetic Theory of Viscosity, Thermal Conduction and Diffusion in Gases*. Cambridge Mathematical Library. Cambridge University Press, 1990.
- [20] J. H. Ferziger, M. Perić, and R. L. Street. *Computational methods for fluid dynamics*. Vol. 3. Springer, 2002.
- [21] A. Nagarajan and S. Soghrati. “Conforming to interface structured adaptive mesh refinement: 3D algorithm and implementation”. In: *Computational Mechanics* 62.5 (2018), pp. 1213–1238.
- [22] A. B. Lambe and A. Czekanski. “Topology optimization using a continuous density field and adaptive mesh refinement”. In: *International Journal for Numerical Methods in Engineering* 113.3 (2018), pp. 357–373.

- [23] S. Zhang, A. L. Gain, and J. A. Norato. “Adaptive mesh refinement for topology optimization with discrete geometric components”. In: *Computer Methods in Applied Mechanics and Engineering* 364 (2020), p. 112930.
- [24] K. S. Surana et al. “k-version of finite element method in gas dynamics: higher-order global differentiability numerical solutions”. In: *International Journal for Numerical Methods in Engineering* 69.6 (2007), pp. 1109–1157.
- [25] K. Huebner et al. *The Finite Element Method for Engineers*. A Wiley-Interscience publication. Wiley, 2001.
- [26] B. Chopard and M. Droz. *Cellular Automata Modeling of Physical Systems*. Aléa-Saclay. Cambridge University Press, 2005.
- [27] D d'Humières, P Lallemand, and U Frisch. “Lattice Gas Models for 3D Hydrodynamics”. In: *Europhysics Letters (EPL)* 2.4 (1986), pp. 291–297.
- [28] M. Kardar. *Statistical Physics of Particles*. Cambridge University Press, 2007.
- [29] R. Lerner and G. Trigg. *Encyclopedia of Physics*. Wiley, 1990.
- [30] 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”. In: *Phys. Rev.* 94 (3 1954), pp. 511–525.
- [31] D. A. Wolf-Gladrow. “5. Lattice Boltzmann Models”. In: *Lattice Gas Cellular Automata and Lattice Boltzmann Models: An Introduction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 159–246.
- [32] G. Silva and V. Semiao. “Truncation errors and the rotational invariance of three-dimensional lattice models in the lattice Boltzmann method”. In: *Journal of Computational Physics* 269 (2014), pp. 259–279.
- [33] S. Ubertini, P. Asinari, and S. Succi. “Three ways to lattice Boltzmann: A unified time-marching picture”. In: *Phys. Rev. E* 81 (1 2010), p. 016311.
- [34] Y. H Qian and S. A Orszag. “Lattice BGK Models for the Navier-Stokes Equation: Nonlinear Deviation in Compressible Regimes”. In: *Europhysics Letters (EPL)* 21.3 (1993), pp. 255–259.
- [35] T. Krüger et al. *The Lattice Boltzmann Method*. Graduate Texts in Physics March 2015. Cham: Springer International Publishing, 2017, pp. 106–112.

- [36] F. J. Alexander et al. “Lattice Boltzmann model for compressible fluids”. In: *Phys. Rev. A* 46 (4 1992), pp. 1967–1970.
- [37] C. Coreixas et al. “Recursive regularization step for high-order lattice Boltzmann methods”. In: *Phys. Rev. E* 96 (3 2017), p. 033306.
- [38] N. Frapolli, S. S. Chikatamarla, and I. V. Karlin. “Entropic lattice Boltzmann model for compressible flows”. In: *Phys. Rev. E* 92 (6 2015), p. 061301.
- [39] Q. Li et al. “Coupling lattice Boltzmann model for simulation of thermal flows on standard lattices”. In: *Physical Review E* 85.1 (2012).
- [40] S. Guo et al. “An efficient lattice Boltzmann method for compressible aerodynamics on D3Q19 lattice”. In: *Journal of Computational Physics* 418 (2020), p. 109570.
- [41] R. K. Agarwal, K.-Y. Yun, and R. Balakrishnan. “Beyond Navier–Stokes: Burnett equations for flows in the continuum–transition regime”. In: *Physics of Fluids* 13.10 (2001), pp. 3061–3085.
- [42] R. Courant, K. Friedrichs, and H. Lewy. “On the Partial Difference Equations of Mathematical Physics”. In: *IBM Journal of Research and Development* 11.2 (1967), pp. 215–234.
- [43] I. Ginzburg, D. d’Humières, and A. Kuzmin. “Optimal Stability of Advection-Diffusion Lattice Boltzmann Models with Two Relaxation Times for Positive/Negative Equilibrium”. In: *Journal of Statistical Physics* 139.6 (2010), pp. 1090–1143.
- [44] B. Servan-Camas and F. T.-C. Tsai. “Non-negativity and stability analyses of lattice Boltzmann method for advection–diffusion equation”. In: *Journal of Computational Physics* 228.1 (2009), pp. 236–256.
- [45] X. D. Niu et al. “Investigation of Stability and Hydrodynamics of Different Lattice Boltzmann Models”. In: *Journal of Statistical Physics* 117.3 (2004), pp. 665–680.
- [46] J. Latt et al. “Straight velocity boundaries in the lattice Boltzmann method”. In: *Phys. Rev. E* 77 (5 2008), p. 056703.
- [47] P. A. Skordos. “Initial and boundary conditions for the lattice Boltzmann method”. In: *Phys. Rev. E* 48 (6 1993), pp. 4823–4842.
- [48] G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge Mathematical Library. Cambridge University Press, 2000.

- [49] J. Anderson. *Computational Fluid Dynamics*. Computational Fluid Dynamics: The Basics with Applications. McGraw-Hill Education, 1995.
- [50] A. J. C. Ladd. “Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation”. In: *Journal of Fluid Mechanics* 271 (1994), 285–309.
- [51] I. Ginzburg and D. d’Humières. “Multireflection boundary conditions for lattice Boltzmann models”. In: *Phys. Rev. E* 68 (6 2003), p. 066614.
- [52] O. Filippova and D. Hänel. “Grid Refinement for Lattice-BGK Models”. In: *Journal of Computational Physics* 147.1 (1998), pp. 219–228.
- [53] Q. Zou and X. He. “On pressure and velocity boundary conditions for the lattice Boltzmann BGK model”. In: *Physics of Fluids* 9.6 (1997), pp. 1591–1598.
- [54] M. Bouzidi, M. Firdaouss, and P. Lallemand. “Momentum transfer of a Boltzmann-lattice fluid with boundaries”. In: *Physics of Fluids* 13.11 (2001), pp. 3452–3459.
- [55] S. C. Fu, R. M. So, and R. C. Leung. “Modeled Boltzmann equation and its application to direct aeroacoustics simulation”. In: *AIAA Journal* 46.7 (2008), pp. 1651–1662.
- [56] J. Latt and B. Chopard. “Lattice Boltzmann method with regularized pre-collision distribution functions”. In: *Mathematics and Computers in Simulation* 72.2-6 (2006), pp. 165–168.
- [57] R. Zhang, X. Shan, and H. Chen. “Efficient kinetic method for fluid simulation beyond the Navier-Stokes equation”. In: *Phys. Rev. E* 74 (4 2006), p. 046703.
- [58] P. Lallemand and L.-S. Luo. “Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability”. In: *Phys. Rev. E* 61 (6 2000), pp. 6546–6562.
- [59] J. Lu et al. “Immersed boundary lattice Boltzmann model based on multiple relaxation times”. In: *Phys. Rev. E* 85 (1 2012), p. 016711.
- [60] K. Suga et al. “A D3Q27 multiple-relaxation-time lattice Boltzmann method for turbulent flows”. In: *Computers & Mathematics with Applications* 69.6 (2015), pp. 518–529.
- [61] B. An et al. “New applications of numerical simulation based on lattice Boltzmann method at high Reynolds numbers”. In: *Computers & Mathematics with Applications* 79.6 (2020), pp. 1718–1741.

- [62] M. J. K. Patrick Nathen Daniel Gaudlitz and N. A. Adams. “On the Stability and Accuracy of the BGK, MRT and RLB Boltzmann Schemes for the Simulation of Turbulent Flows”. In: *Communications in Computational Physics* 23.3 (2018), pp. 846–876.
- [63] A. Montessori et al. “Regularized lattice Bhatnagar-Gross-Krook model for two- and three-dimensional cavity flow simulations”. In: *Phys. Rev. E* 89 (5 2014), p. 053317.
- [64] O. Malaspinas. *Increasing stability and accuracy of the lattice Boltzmann scheme: recursivity and regularization*. 2015.
- [65] J. Smagorinsky. “General Circulation Experiments with the Primitive Equations”. In: *Monthly Weather Review* 91.3 (1963), pp. 99–164.
- [66] H. Yu, S. S. Girimaji, and L.-S. Luo. “DNS and LES of decaying isotropic turbulence with and without frame rotation using lattice Boltzmann method”. In: *Journal of Computational Physics* 209.2 (2005), pp. 599–616.
- [67] P. Sagaut. “Toward advanced subgrid models for Lattice-Boltzmann-based Large-eddy simulation: Theoretical formulations”. In: *Computers and Mathematics with Applications* 59.7 (2010), pp. 2194–2199.
- [68] S. Stolz and N. A. Adams. “An approximate deconvolution procedure for large-eddy simulation”. In: *Physics of Fluids* 11.7 (1999), pp. 1699–1701.
- [69] F. Nicoud and F. Ducros. “Subgrid-Scale Stress Modelling Based on the Square of the Velocity Gradient Tensor”. In: *Flow, Turbulence and Combustion* 62.3 (1999), pp. 183–200.
- [70] K. N. Premnath, M. J. Pattison, and S. Banerjee. “Dynamic subgrid scale modeling of turbulent flows using lattice-Boltzmann method”. In: *Physica A-statistical Mechanics and Its Applications* 388 (2009), pp. 2640–2658.
- [71] M. Weickert et al. “Investigation of the LES WALE turbulence model within the lattice Boltzmann framework”. In: *Computers & Mathematics with Applications* 59.7 (2010). Mesoscopic Methods in Engineering and Science, pp. 2200–2214.
- [72] E. LÉVÊQUE et al. “Shear-improved Smagorinsky model for large-eddy simulation of wall-bounded turbulent flows”. In: *Journal of Fluid Mechanics* 570 (2007), 491–502.
- [73] O. Malaspinas and P. Sagaut. “Wall model for large-eddy simulation based on the lattice Boltzmann method”. In: *Journal of Computational Physics* 275 (2014), pp. 25–40.

- [74] M. Haussmann et al. “Large-eddy simulation coupled with wall models for turbulent channel flows at high Reynolds numbers with a lattice Boltzmann method — Application to Coriolis mass flowmeter”. In: *Computers & Mathematics with Applications* 78.10 (2019), pp. 3285–3302.
- [75] I. V. Karlin, A. Ferrante, and H. C. Öttinger. “Perfect entropy functions of the Lattice Boltzmann method”. In: *Europhysics Letters (EPL)* 47.2 (1999), pp. 182–188. URL: [10.1209/epl/i1999-00370-1](https://doi.org/10.1209/epl/i1999-00370-1).
- [76] S. Ansumali and I. V. Karlin. “Stabilization of the lattice Boltzmann method by the H theorem: A numerical test”. In: *Phys. Rev. E* 62 (6 2000), pp. 7999–8003.
- [77] S. Chikatamarla and I. Karlin. “Entropic lattice Boltzmann method for turbulent flow simulations: Boundary conditions”. In: *Physica A: Statistical Mechanics and its Applications* 392.9 (2013), pp. 1925–1930.
- [78] I. V. Karlin, F. Bösch, and S. S. Chikatamarla. “Gibbs’ principle for the lattice-kinetic theory of fluid dynamics”. In: *Phys. Rev. E* 90 (3 2014), p. 031302.
- [79] B. Dorschner, S. S. Chikatamarla, and I. V. Karlin. “Transitional flows with the entropic lattice Boltzmann method”. In: *Journal of Fluid Mechanics* 824 (2017), 388–412.
- [80] F. Nannelli and S. Succi. “The lattice Boltzmann equation on irregular lattices”. In: *Journal of Statistical Physics* 68.3-4 (1992), pp. 401–407.
- [81] S. Succi, G. Amati, and R. Benzi. “Challenges in lattice Boltzmann computing”. In: *Journal of Statistical Physics* 81.1-2 (1995), pp. 5–16.
- [82] X. He, L.-S. Luo, and M. Dembo. “Some Progress in Lattice Boltzmann Method. Part I. Nonuniform Mesh Grids”. In: *Journal of Computational Physics* 129.2 (1996), pp. 357–363.
- [83] M. Schäfer et al. “Benchmark Computations of Laminar Flow Around a Cylinder”. In: *Flow Simulation with High-Performance Computers II: DFG Priority Research Programme Results 1993–1995*. Ed. by E. H. Hirschel. Wiesbaden: Vieweg+Teubner Verlag, 1996, pp. 547–566.
- [84] D. Yu, R. Mei, and W. Shyy. “A multi-block lattice Boltzmann method for viscous fluid flows”. In: *International Journal for Numerical Methods in Fluids* 39.2 (2002), pp. 99–120.

- [85] A. Dupuis and B. Chopard. “Theory and applications of an alternative lattice Boltzmann grid refinement algorithm”. In: *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics* 67.6 (2003), p. 7.
- [86] D. Lagrava et al. “Advances in multi-domain lattice Boltzmann grid refinement”. In: *Journal of Computational Physics* 231.14 (2012), pp. 4808–4822.
- [87] H. Chen et al. “Grid refinement in lattice Boltzmann methods based on volumetric formulation”. In: *Physica A: Statistical Mechanics and its Applications* 362.1 (2006), pp. 158–167.
- [88] M. Rohde et al. “A generic, mass conservative local grid refinement technique for lattice-Boltzmann schemes”. In: *International Journal for Numerical Methods in Fluids* 51.4 (2006), pp. 439–468.
- [89] G. Eitel-Amor, M. Meinke, and W. Schröder. “A lattice-Boltzmann method with hierarchically refined meshes”. In: *Computers & Fluids* 75 (2013), pp. 127–139.
- [90] H. Touil, D. Ricot, and E. L  v  que. “Direct and large-eddy simulation of turbulent flows on composite multi-resolution grids by the lattice Boltzmann method”. In: *Journal of Computational Physics* 256 (2014), pp. 220–233.
- [91] S. Ansumali. “Minimal kinetic modeling of hydrodynamics”. en. PhD thesis. Z  rich: ETH Zurich, 2004.
- [92] M. J. Krause et al. “OpenLB—Open source lattice Boltzmann code”. In: *Computers & Mathematics with Applications* 81 (2021). Development and Application of Open-source Software for Problems with Numerical PDEs, pp. 258–288.
- [93] E. Gabriel et al. “Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation”. In: *Proceedings, 11th European PVM/MPI Users’ Group Meeting*. Budapest, Hungary, 2004, pp. 97–104.
- [94] S. Y. L. Sol Ji Kang and K. M. Lee. “Performance Comparison of OpenMP, MPI, and MapReduce in Practical Problems”. In: *Advances in Multimedia* 2015 (2015), p. 9.
- [95] A. Kummerl  nder. *Cgit page of 2D grid refinement framework*. URL: https://code.kummerlaender.eu/grid_refinement_bsc_thesis/.
- [96] Z. Xu. *Github page of the grid refinement implementation*. URL: https://github.com/Windrocer/olb_gr.

- [97] J. Park and H. Choi. “Numerical solutions of flow past a circular cylinder at reynolds numbers up to 160”. In: *KSME International Journal* 12.6 (1998), pp. 1200–1205.
- [98] J. Kim, D. Kim, and H. Choi. “An Immersed-Boundary Finite-Volume Method for Simulations of Flow in Complex Geometries”. In: *Journal of Computational Physics* 171.1 (2001), pp. 132–150.
- [99] H. Jiang et al. “Three-dimensional direct numerical simulation of wake transitions of a circular cylinder”. In: *Journal of Fluid Mechanics* 801 (2016), pp. 353–391.
- [100] C. H. Williamson. “Three-dimensional wake transition”. In: *Journal of Fluid Mechanics* 328 (1996), pp. 345–407.
- [101] E. W. Kam, R. M. So, and R. C. Leung. “Lattice boltzmann method simulation of aeroacoustics and nonreflecting boundary conditions”. In: *AIAA Journal* 45.7 (2007), pp. 1703–1712.
- [102] S. Izquierdo, P. Martínez-Lera, and N. Fueyo. “Analysis of open boundary effects in unsteady lattice Boltzmann simulations”. In: *Computers & Mathematics with Applications* 58.5 (2009). Mesoscopic Methods in Engineering and Science, pp. 914–921.
- [103] S. C. Fu and R. M. So. “Modeled lattice boltzmann equation and the constant-density assumption”. In: *AIAA Journal* 47.12 (2009), pp. 3038–3042.
- [104] J. Koplik and J. R. Banavar. “Corner flow in the sliding plate problem”. In: *Physics of Fluids* 7.12 (1995), pp. 3118–3125.
- [105] D. Samantaray and M. K. Das. “Nature of turbulence inside a cubical lid-driven cavity: Effect of Reynolds number”. In: *International Journal of Heat and Fluid Flow* 80.November 2018 (2019), p. 108498.
- [106] A. K. Prasad and J. R. Koseff. “Reynolds number and end-wall effects on a lid-driven cavity flow”. In: *Physics of Fluids A* 1.2 (1989), pp. 208–218.
- [107] S. M. Grace, W. G. Dewar, and D. E. Wroblewski. “Experimental investigation of the flow characteristics within a shallow wall cavity for both laminar and turbulent upstream boundary layers”. In: *Experiments in Fluids* 36.5 (2004), pp. 791–804.
- [108] R. J. Jefferson-Loveday. “Hybrid rans-les for turbomachinery”. In: *Proceedings of the ASME Turbo Expo 2C-2018* (2018), pp. 1–12.

- [109] P. A. Skordos. “Initial and boundary conditions for the lattice Boltzmann method”. In: *Phys. Rev. E* 48 (6 1993), pp. 4823–4842.
- [110] V. M. Canuto and Y. Cheng. “Determination of the Smagorinsky–Lilly constant CS”. In: *Physics of Fluids* 9.5 (1997), pp. 1368–1378.
- [111] B. Galperin and S. A. Orszag. *Large eddy simulation of complex engineering and geophysical flows*. eng. Cambridge: Cambridge University Press, 1993.
- [112] S. Hou et al. *A Lattice Boltzmann Subgrid Model for High Reynolds Number Flows*. 1994.
- [113] V. M. Canuto and Y. Cheng. “Determination of the Smagorinsky–Lilly constant CS”. In: *Physics of Fluids* 9.5 (1997), pp. 1368–1378.
- [114] J. MEYERS and P. SAGAUT. “On the model coefficients for the standard and the variational multi-scale Smagorinsky model”. In: *Journal of Fluid Mechanics* 569 (2006), 287–319.

Appendices

A. Appendix

A.1. Data Structure of Some Member Class of *Grid3D*

To avoid redundancy, some code blocks illustrating the data structures of some member classes of *Grid3D* in Section 5.2.2 are shown here. These includes the data structures of *RefiningGrid3D* interface, the two types of *Coupler3D* object, and the vectors storing the refined grids and *Coupler3D* objects required in *Grid3D*.

Listing A.1: Declaration of *RefiningGrid3D*

```
1  template <typename T, typename DESCRIPTOR>
   class RefiningGrid3D : public Grid3D<T,DESCRIPTOR> {
   private:
       const Vector<T,3> _origin;
       const Vector<T,3> _extend;
       Vector<bool,3> _periodicityOn;
       Grid3D<T,DESCRIPTOR>& _parentGrid;

   public:
10     RefiningGrid3D(Grid3D<T,DESCRIPTOR>& parentGrid,
        Vector<T,3> origin, Vector<T,3> extend);
        RefiningGrid3D(Grid3D<T,DESCRIPTOR>& parentGrid,
        Vector<T,3> origin, Vector<T,3> extend,
        bool periodicityX, bool periodicityY,
15     bool periodicityZ);

        Vector<T,3> getOrigin() const;
        Vector<T,3> getExtend() const;

20     // Indicator of the subsection of the domain of two grids
        // which part in the coarse grid need to be set as null
        std::unique_ptr<IndicatorF3D<T>> getRefinedOverlap() const;
};
```

Listing A.2: Couplers used in *Grid3D*

```

1 FineCoupler3D<T,DESCRIPTOR>& addFineCoupling(
    Grid3D<T,DESCRIPTOR>& fineGrid,
    Vector<T,3> origin, Vector<T,3> extend);
CoarseCoupler3D<T,DESCRIPTOR>& addCoarseCoupling(
5   Grid3D<T,DESCRIPTOR>& fineGrid,
    Vector<T,3> origin, Vector<T,3> extend);

```

Listing A.3: Vectors storing child refined grids and couplers of *Grid3D*

```

1 std::vector<std::unique_ptr<RefiningGrid3D<T,DESCRIPTOR>>>
    _fineGrids;
std::vector<std::unique_ptr<FineCoupler3D<T,DESCRIPTOR>>>
    _fineCouplers;
5 std::vector<std::unique_ptr<CoarseCoupler3D<T,DESCRIPTOR>>>
    _coarseCouplers;

```

Listing A.4: The structure of *FineCoupler3D*

```

1 template <typename T, typename DESCRIPTOR>
class FineCoupler3D : public Coupler3D<T,DESCRIPTOR> {
private:
    // vectors recording the properties of original coarse nodes
5   std::vector<std::vector<Vector<T,1>>> _c2f_rho;
    std::vector<std::vector<Vector<T,DESCRIPTOR::d>>> _c2f_u;
    std::vector<std::vector<Vector<T,DESCRIPTOR::q>>> _c2f_fneq;
public:
    FineCoupler3D(Grid3D<T,DESCRIPTOR>& coarse,
10   Grid3D<T,DESCRIPTOR>& fine,
    Vector<T,3> origin, Vector<T,3> extend);
    void store();
    void interpolate();
    void couple();

```

A.2. Data Structure of *FineCoupler3D* and *CoarseCoupler3D*

A.2.1. The *FineCoupler3D* Class

The interface of *FineCoupler3D* is given in Code A.4. As mentioned before in Section 4.3, the fine grids performs collision and streaming twice during one coarse time step, and each collision and streaming requires a coupling to calculate the missing populations. The temporal interpolation of the coarse properties in the fine nodes is also done here.

The original state of the coarse nodes need to be stored in an vector using the *store()* function before the collision and streaming on the coarse grid. The intermediate temporal states of the coarse properties are computed via the *interpolate()* function, and the coupling is performed through *couple()* function. Functions to calculate fourth-order spatial interpolations are included as a local function instead of a member function. The interpolation and regularisation procedure are performed in the *couple()* function.

A.2.2. The *CoarseCoupler3D* Class

The interface of the *CoarseCoupler3D* class is shown in Code A.5. It is relatively simple since it involves no temporal interpolations. The restriction procedure is also implemented in the *couple()* function to increase flow stability at high Reynolds number. If the regularised coupling is used, the restriction procedure can be skipped as addressed in Section 4.1.

Listing A.5: The structure of *CoarseCoupler3D*

```

1  template <typename T, typename DESCRIPTOR>
   template <typename T, typename DESCRIPTOR>
   class CoarseCoupler3D : public Coupler3D<T,DESCRIPTOR> {
   public:
5     CoarseCoupler3D(Grid3D<T,DESCRIPTOR>& coarse,
                     Grid3D<T,DESCRIPTOR>& fine,
                     Vector<T,3> origin, Vector<T,3> extend);
       void couple();
   };

```
