Computational Fluid Dynamics of Wind Power Using Lattice-Boltzmann Method on GPUs

Master's thesis in Applied Mechanics

Xinyuan Shao

Department of Mechanics and Maritime Sciences

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021:68

Computational Fluid Dynamics of Wind Power Using Lattice-Boltzmann Method on GPUs

XINYUAN SHAO

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Fluid Dynamics
Chalmers University of Technology
Gothenburg, Sweden 2021

Computational Fluid Dynamics of Wind Power
Using Lattice-Boltzmann Method on GPUs
XINYUAN SHAO

Examiner: Lars Davidson (Lada@chalmers.se)
Supervisors at Chalmers: Lars Davidson, Huadong Yao (huadong@chalmers.se)
Supervisor at company OX2: Ingemar Carlén (ingemar.carlen@teknikgruppen.se)

Master's Thesis 2021:68
Department of Mechanics and Maritime Sciences
Division of Fluid Dynamics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Computational Fluid Dynamics of Wind Power
Using Lattice-Boltzmann Method on GPUs
XINYUAN SHAO
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

Wind energy is an environmentally friendly substitute for fossil energy. In order to extract energy from the wind, wind turbine farms are being built in more and more places around the world. It is important to evaluate the wind conditions at sites before construction. However, met masts can only measure wind speed at specific locations. Usually, measurements are combined with numerical simulations to obtain more information at the location of a wind turbine. The Lattice-Boltzmann method (LBM) is a new and promising way to numerically simulate many wind energy problems with the advantage of fast speed, simple implementation and easy parallelization. In this thesis, a GPU code GANSCANS developed at the University of Manchester based on LBM is tested on wind energy scenarios, such as cases with forest cover and complex terrain. The case with forest shows good agreement with experimental data, indicating that GANSCANS is a promising simulation tool that can give reliable results with acceptable time and computational resource consumption. After validating the code, it is applied to evaluate a real-world case, Hornamossen. Several combinations of boundary conditions are verified.

Keywords: LBM, GPU, GANSCANS, wind energy, forest, complex terrain.

Acknowledgements

The two-year master's journey is about coming to an end in a way that is much faster than I thought it would be. I am so grateful to have met nice new friends and professors at Chalmers and get encouragements as well as inspirations from them.

Firstly, I want to say thank you to Prof. Lars Davidson and Prof. Huadong Yao. I can not imagine what would happen without your guidance and comfort when I was facing some tricky troubles in my project. I hope that one day I can become a dedicated professor just like you two.

I got a lot of help from the research group at the University of Manchester. Thank you Prof. Alistair Revell and Dr. Marta Camps Santasmasas.

Dr. Ingemar Carlén is a very experienced engineer and researcher from the sustainable energy company OX2. I received valuable knowledge about engineering application of wind energy from him. Thank you so much Dr. Ingemar Carlén.

The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC), partially funded by the Swedish Research Council through grant agreement no. 2018-05973. Many thanks for your support!

It is very normal to feel down and keep denying yourself sometimes. The reason may be that the weather is not good, the work is not going well, etc. There is always one way that can make me get out of this condition, that is, having fun with my friends. I really want to thank my friends Hezhe Xiao and Yao Cai. You do care about my feelings and always support me in your ways. You are always patient with me and willing to listen. Thank you.

I also want to thank my friends and professors at Nanjing University of Aeronautics and Astronautics (NUAA). NUAA is my undergraduate school and also the place where I built up my interest in mechanics. I still remember those good old days when I got up early in the morning just in order to have a good seat in the classroom. Those days are still as vivid as yesterday.

Finally, I would like to thank my father, mother, sister and brother. You always support my decisions and give me power and comfort. My luckiest thing is having a big and warm family. I am very grateful for this.

The education and experience I have has made me the person who I am now. I hope that in the near future I can dive deep into the area which I am interested in and make some progresses. Please keep going and carry on!

Xinyuan Shao, Gothenburg, August 2021

致谢

两年的硕士时光匆匆而过，它的结束比我想象中快很多。在这两年里，我认识了很多新朋友和教授。非常感谢他们给我的帮助和鼓励。

我想要感谢 Lars Davidson 教授和姚华栋教授，在你们的帮助下我才能顺利完成这次毕设。希望有一天我也能成为像你们一样的科研工作者。来自曼切斯特大学的 Alistair Revell 教授和 Marta Camps Santasmasas 博士也在这次毕设中给予我很大帮助，谢谢你们。在风能的工程应用方面，来着 OX2 公司的 Ingemar Carlén 先生传授给我许多宝贵经验，万分感谢。

在这两年中，我有时会情绪低落并且否定自己，可能是因为天气不好或者是学习遇到困难。在这种时候我会向我的朋友们求助。谢谢你们，我的朋友肖荷喆 和蔡瑶。你们时时刻刻都用你们的方式来支持我，并且总是以充足的耐心聆听我。非常感谢！

我还想谢谢我在南京航空航天大学的朋友和老师们。南航是我的母校，它给了我从建筑环境专业转到飞行器设计专业的机会还建立起了我对力学的兴趣。我还记得那些旧时光，那时我经常起得很早去教室占前排的座位。这明明已经是四五年之前的事情了，却还鲜活得像昨天一样。

最后我想谢谢我的爸爸妈妈妹妹和弟弟。你们总是默默支持我的各种决定，给我全力的协助。拥有你们是我最幸运的事。我非常感激。

我接受过的教育使我成为今天的我。希望在不远的将来，我可以在我感兴趣的领域做出一点成绩。请继续前行！

邵欣圆

2021 年 8 月

Notations

Abbreviations

CFD	Computational Fluid Dynamics
CFL	Courant–Friedrichs–Lewy number
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DNS	Direct Numerical Simulation
FVM	Finite Volume Method
GANSCANS	GPU-accelerated solver for coupled approaches to Navier-Stokes
GPU	Graphics Processing Unit
LAD	Leaf Area Density
LBE	Lattice Boltzmann Equation
LBM	Lattice-Boltzmann Method
LES	Large Eddy Simulation
LUMA	Lattice-Boltzmann at The University of Manchester
MRT	Multi-Relaxation-Time
NS	Navier-Stokes
NEWA	New European Wind Atlas
PDE	Partial Differential Equation
RANS	Reynolds-Averaged Navier-Stokes
RMS	Root Mean Square
TKE	Turbulence Kinetic Energy
TRT	Two-Relaxation-Time

Greek symbols

ρ	Density
Ω	Collision operator
ξ	Three dimensional particle velocity
$\nu_{\mathbf{0}}$	Viscosity
$\nu_{\mathbf{total}}$	Total effective viscosity
τ_0	Relaxation time
τ_{total}	Total relaxation time
τ_w	Wall shear stress
ε_{ij3}	Alternating unit tensor

ϕ	Latitude
δ	Boundary layer thickness

Roman symbols

a_f	Leaf area density
C_D	Drag coefficient
C	Smagorinsky model constant
E	Internal energy
f	Distribution function
f^{eq}	Equilibrium distribution function
F	External force
$F_{f,i}$	Drag force
$F_{c,i}$	Coriolis force
p	Pressure
Re	Reynolds number
Re_τ	Reynolds number based on the friction velocity
$\bar{\mathbf{S}}$	Strain rate tensor
t	Time
$\boldsymbol{u}$	Three dimensional mean velocity
$\boldsymbol{x}$	Three dimensional position
u_τ	Friction velocity
U	local wind speed

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

As the world's energy resources such as coal, oil and other fossil fuels are nonrenewable, mankind has been working hard at finding new and sustainable energy resources for a long time. Nuclear power was considered as a very efficient source of energy. However, after the Chernobyl disaster, people around the world are prone to resist nuclear energy and looking for eco-friendly and renewable energy sources as alternate energy supplies. Nowadays, wind energy is one of the most popular clean energy sources.

A wind farm is an area with a group of wind turbines to generate electricity. To design a new wind farm, the wind conditions at the site must be known to evaluate the power generation rate and the safety of wind turbine structures. Usually, the wind speed data is collected by setting up a met mast for 1-2 years which can measure the wind speed up to the hub height of a planned wind turbine. Then, with the help of the met mast data, the wind conditions at the wind turbine can be estimated using some simple prediction methods, for example, WAsP [6] or Reynolds-Averaged Navier-Stokes equations (RANS) [7]. However, these simulations do not consider the effects of the wakes behind wind turbines. Instead, they use simple wake models. Therefore, they are not sufficiently accurate and unable to provide the information on turbulent flow fluctuations, which introduce unsteady loads to turbine blades and consequently shorten the lifetime of wind turbines. Large Eddy Simulation (LES) [8] is more accurate but the conventional LES based on the finite volume method (FVM) needs enormous computing resources. For example, an LES case over a 10km $\times$ 10km $\times$ 1km computational domain takes 4-5 days on 400 cores, indicating this method is too expensive to be used as an industrial tool. Considering the advantages of the lattice-Boltzmann method (LBM) in parallelization, it has recently been used as a replacement for the FVM in the simulation of turbulent flows. Therefore, we are motivated to investigate this method for wind power simulations in the present study.

LBM is explicit and needs only the information of the nearest nodes, which means that LBM can be easily accelerated by parallel implementation using the Graphics Processing Unit (GPU) architecture [9]. A GPU is usually used to render pixels on the screen, but can also be applied to non-graphical computations because of its advantage in executing single-instruction multiple threads tasks [10, 11]. LUMA is an open-source code based on LBM developed at the University of Manchester [12]. It has a GPU version named GASCANS that is used in this project.

1.2 Problem statement

There are usually two choices for wind farm location, offshore and onshore. For offshore wind farms, the main disadvantage is the high maintenance costs due to corrosion of the towers caused by salt water. It is also very difficult to transmit electricity from a wind farm that is far out at sea because cables may also be corroded by the salt water. Onshore wind farms have longer history than their counterpart. However it also has disadvantages. For example, the wind farm may be located near residential areas and cause visual and noise pollution to the local residents. In order to mitigate this disadvantage, the wind farm is preferably to be located in remote areas like forests. In this thesis, the main topic is to simulate the turbulent flow field over homogeneous forest and complex terrain with forest using the fast LBM code GANSCANS. The simulation results can be used for the wind turbine structural analysis and serve as a guide for wind farm design.

1.3 Limitations

The limitation of this project is that, as seen in Figure 4.18, for the case of a complex terrain, although the forest cover is heterogeneous, the code can only simulate a forest covered with constant leaf area density (LAD) over the height, which is an unrealistic simplification. In the future, the code needs to be further developed to account for the varying LAD over the complex terrain.

1.4 Thesis aims and layout

The aim of the project is to comprehensively investigate the capability and performance of the LBM code named GANSCANS by simulating several cases: turbulent channel flow, atmospheric boundary layer with and without forest and the Hornamossen case which has complex topography and forest.
The layout of the thesis is as follows:

- The basic theoretical background is introduced in Chapter 2. The theory there is merely a compact summary of the textbook *The Lattice Boltzmann Method Principles and Practice* [1] and reorganized based on the author's understanding to give a quick and easy start-up point for the upcoming researchers who want to know a bit of LBM, as the author did.
- Chapter 3 introduces the methodology including the structure of GPUs, the features of GANSCANS and how to generate turbulence in a practical way.
- Chapter 4 contains three cases: one validation case - the turbulent channel flow, and two cases to study the influence of complex terrain and forest cover. The results of these cases are analyzed in detail and compared with existing experimental data from some articles.
- Chapter 5 gives the conclusion of this project and some perspectives for future studies.

2

Theory

This chapter is mainly focused on the theories used in this project, including LBM and basic knowledge of wind power. The aim of this chapter is to give readers a comprehensive understanding of the theories that play important role in this project. Although there are many existing textbooks that cover these theories (the one referenced here is [1]), this section will give readers an easy-to-understand way to quickly get hand of them from a beginner's perspective.

2.1 Lattice-Boltzmann Method

As no one knows the exact origin of the universe we live in, the author is always thinking about one of the philosophical questions: are those axioms describing our universe simply specific projections of the 'real' axioms, which means that other kind of projections can also be taken to simplify the expression of them? LBM is a relatively new method that is based on kinetic theory instead of conservation of mass and momentum. It has obvious advantages over the conventional computational fluid dynamics (CFD) method. For example, it is simple, scalable on parallel computers, and easy to deal with complex geometries.

2.1.1 Kinetic theory

Kinetic theory is the cornerstone of LBM. In conventional CFD, the typical scale is the scale for the gradient in some macroscopic properties, while in LBM, the typical scale is the mean free path (Figure 2.1). It can also be seen from Figure 2.1 that LBM does not invest individual molecules or fully continuum media. Instead, it tracks the distributions of collections of molecules. This trade-off makes LBM an efficient method of computing fluid dynamics with reasonable computational resources consumption.

The particle distribution function $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ is the fundamental variable in the kinetic theory. Specifically, $\boldsymbol{\xi}$ stands for three-dimensional velocity (ξ_x, ξ_y, ξ_z) and $\boldsymbol{x}$ is the three-dimensional position (x, y, z). We conceive that $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ is equivalent to the density ρ. Nonetheless, their difference lies in the fact that $\rho(\boldsymbol{x}, t)$ represents the density of mass only in physical space, while $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ represents the density of mass not only in physical space but also in velocity space.

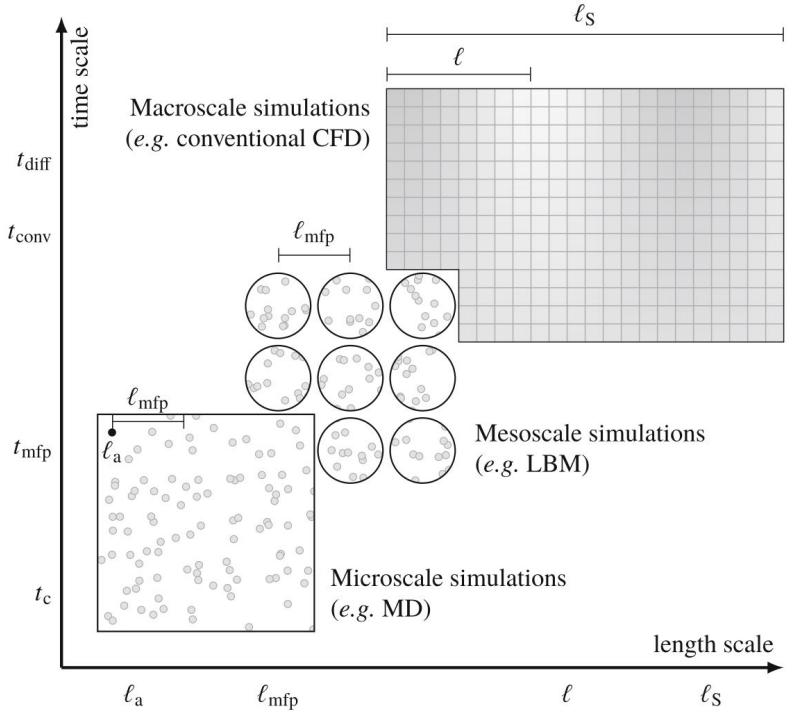


Figure 2.1: Length and time scales in different kind of simulations [1]

The moments of the distribution function f can act as a link between mesoscopic and macroscopic variables. The relations between them are as follows:

$$\rho(\boldsymbol{x}, t) = \int f(\boldsymbol{x}, \boldsymbol{\xi}, t) \mathrm{d}^3\xi \tag{2.1}$$

$$\rho(\boldsymbol{x}, t)\boldsymbol{u}(\boldsymbol{x}, t) = \int \boldsymbol{\xi} f(\boldsymbol{x}, \boldsymbol{\xi}, t) \mathrm{d}^3\xi \tag{2.2}$$

$$\rho(\boldsymbol{x}, t)E(\boldsymbol{x}, t) = \frac{1}{2}\int |\xi|^2 f(\boldsymbol{x}, \boldsymbol{\xi}, t) \mathrm{d}^3\xi \tag{2.3}$$

In order to know how the distribution function evolves in time, its total derivative with respect to time is taken:

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \left(\frac{\partial f}{\partial t}\right)\frac{\mathrm{d}t}{\mathrm{d}t} + \left(\frac{\partial f}{\partial x_\beta}\right)\frac{\mathrm{d}x_\beta}{\mathrm{d}t} + \left(\frac{\partial f}{\partial \xi_\beta}\right)\frac{\mathrm{d}\xi_\beta}{\mathrm{d}t} \tag{2.4}$$

Usually, in the literature on LBM, $\Omega(f) = \mathrm{d}f/\mathrm{d}t$ is used for the total differential. Moreover, as $\mathrm{d}t/\mathrm{d}t = 1$, $\mathrm{d}x_\beta/\mathrm{d}t = \xi_\beta$ and $\mathrm{d}\xi_\beta/\mathrm{d}t = F_\beta/\rho$, Eq.(2.4) is simplified as:

$$\frac{\partial f}{\partial t} + \xi_\beta\frac{\partial f}{\partial x_\beta} + \frac{F_\beta}{\rho}\frac{\partial f}{\partial \xi_\beta} = \Omega(f) \tag{2.5}$$

This is the Boltzmann equation in continuous space. It is worth noting that $\Omega(f)$ is usually called the collision operator and conserves the quantities of mass and momentum:

$$\int \Omega(f)\mathrm{d}^3\xi = 0 \tag{2.6}$$

$$\int \boldsymbol{\xi}\Omega(f)\mathrm{d}^3\xi = \mathbf{0} \tag{2.7}$$

In the view of fluid dynamics, calculating mesoscopic variables is not the final goal. The final goal is to get macroscale variables like density and velocities so that the flow field can be described. It is vital to prove that macroscopic fluid mechanics equations can be derived directly from the Boltzmann equation by taking the moments of it. Here only the detailed derivation procedures of the mass conservation equation is given. For the momentum and energy conservation, the procedures are similar and can be found in [1].
Integrating the Boltzmann equation over velocity space as follows:

$$\frac{\partial}{\partial t}\int f\ \mathrm{d}^3\xi + \frac{\partial}{\partial x_\beta}\int \xi_\beta f\ \mathrm{d}^3\xi + \frac{F_\beta}{\rho}\int \frac{\partial f}{\partial \xi_\beta}\mathrm{d}^3\xi = \int \Omega(f)\mathrm{d}^3\xi \tag{2.8}$$

On the basis of Eq.(2.1), (2.2), (2.6) and $\int \frac{\partial f}{\partial \xi_\beta}\mathrm{d}^3\xi = 0$, the mass conservation reads:

$$\frac{\partial \rho}{\partial t} + \frac{\partial\left(\rho u_\beta\right)}{\partial x_\beta} = 0 \tag{2.9}$$

2.1.2 The lattice Boltzmann equation

In Section 2.1.1, the continuous Boltzmann equation is obtained. However, the distribution function $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ is a seven-dimensional function ($x, y, z, \xi_x, \xi_y, \xi_z$ and t) which is very hard to solve numerically. However, for fluid dynamics problems, the exact value of $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ is not what we are looking for. We only care about whether the moments of the distribution function can recover some macroscopic variables such as density and velocities. With the proper discretization in velocity, space and time, the lattice Boltzmann equation can be obtained and it is simple to implement and parallelise.

2.1.2.1 Discretization in velocity space

The moments are the weighted integrals of f in velocity space (see Eq.(2.1)-(2.3)). Instead of finding the exact value of f, it is a worthy compromise to only find one function whose integral is identical to that of f. Because f is a function of velocity ξ, it is natural to start from the velocity discretization.
Firstly, the equilibrium distribution function should be introduced. A pre-knowledge is that collisions tend to even out the angular distribution of particle velocities. The result is that if we let a gas develop for sufficiently long time, the distribution function f will finally reach an equilibrium distribution f^{eq}, which is a function of the mean velocity $\boldsymbol{u}$, the position $\boldsymbol{x}$ and time t. The detailed derivation can be found at [1], and here only the final form of f^{eq} is given below:

$$\begin{aligned} f^{\mathrm{eq}}(\boldsymbol{x}, |\boldsymbol{v}|, t) &= \rho\left(\frac{3}{4\pi e}\right)^{3/2} \mathrm{e}^{-3|\boldsymbol{v}|^2/(4e)} = \rho\left(\frac{\rho}{2\pi p}\right)^{3/2} \mathrm{e}^{-p|\boldsymbol{v}|^2/(2\rho)} \\ &= \rho\left(\frac{1}{2\pi RT}\right)^{3/2} \mathrm{e}^{-|\boldsymbol{v}|^2/(2RT)} \end{aligned} \tag{2.10}$$

Hermite polynomials are usually used as base functions to approach a certain well-behaved continuous function by summing the combination of Hermite series and expansion coefficients:

$$f(\boldsymbol{x}) = \omega(\boldsymbol{x}) \sum_{n=0}^{\infty} \frac{1}{n!} \boldsymbol{a}^{(n)} \cdot \boldsymbol{H}^{(n)}(\boldsymbol{x}), \quad \boldsymbol{a}^{(n)} = \int f(\boldsymbol{x}) \boldsymbol{H}^{(n)}(\boldsymbol{x}) \mathrm{d}^d x \tag{2.11}$$

where:

$$\omega(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}} \mathrm{e}^{-x^2/2} \tag{2.12}$$

Apply the Hermite series expansion like in Eq.(2.11) to f^{eq}:

$$f^{\mathrm{eq}}(\rho, \boldsymbol{u}, \theta, \boldsymbol{\xi}) = \omega(\boldsymbol{\xi}) \sum_{n=0}^{\infty} \frac{1}{n!} \boldsymbol{a}^{(n),\mathrm{eq}}(\rho, \boldsymbol{u}, \theta) \cdot \boldsymbol{H}^{(n)}(\boldsymbol{\xi}) \tag{2.13}$$

$$\boldsymbol{a}^{(n),\mathrm{eq}}(\rho, \boldsymbol{u}, \theta) = \int f^{\mathrm{eq}}(\rho, \boldsymbol{u}, \theta, \boldsymbol{\xi}) \boldsymbol{H}^{(n)}(\boldsymbol{\xi}) \mathrm{d}^d \xi \tag{2.14}$$

Substitute Eq.(2.10) into Eq.(2.14) and calculate the integration:

$$a^{(0),\mathrm{eq}} = \rho = \int f^{\mathrm{eq}} \mathrm{d}^d \xi \tag{2.15}$$

$$a_{\alpha}^{(1),\mathrm{eq}} = \rho u_{\alpha} = \int f^{\mathrm{eq}} \xi_{\alpha} \mathrm{d}^d \xi \tag{2.16}$$

$$a_{\alpha\beta}^{(2),\mathrm{eq}} = \rho \left(u_{\alpha} u_{\beta} + (\theta - 1) \delta_{\alpha\beta} \right) \tag{2.17}$$

$$a_{\alpha\beta\gamma}^{(3),\mathrm{eq}} = \rho \left[u_{\alpha} u_{\beta} u_{\gamma} + (\theta - 1) \left(\delta_{\alpha\beta} u_{\gamma} + \delta_{\beta\gamma} u_{\alpha} + \delta_{\gamma\alpha} u_{\beta} \right) \right]. \tag{2.18}$$

Eq(2.15) - (2.17) relate to the density, momentum and energy, respectively. These are the only three variables required to fulfil the conservation laws and represent the macroscopic equations. So it is sufficient to only keep the first three terms in Eq.(2.13) to recover the Navier-Stokes (NS) equations. Then, f^{eq} will be:

$$f^{\mathrm{eq}}(\rho, \boldsymbol{u}, \theta, \boldsymbol{\xi}) \approx \omega(\boldsymbol{\xi}) \rho \left[1 + \xi_{\alpha} u_{\alpha} + \left(u_{\alpha} u_{\beta} + (\theta - 1) \delta_{\alpha\beta} \right) \left(\xi_{\alpha} \xi_{\beta} - \delta_{\alpha\beta} \right) \right] \tag{2.19}$$

Moreover, another characteristics of Hermite polynomials is that the integrals of certain functions can be calculated by summing up integral function values at a few discrete points. Without giving complicated mathematical details, the velocity discretization is simply given here. The details can be found at [1].

$$f_i^{\mathrm{eq}} = w_i \rho \left[1 + \xi_{i\alpha} u_{\alpha} + \frac{1}{2} \left(u_{\alpha} u_{\beta} + (\theta - 1) \delta_{\alpha\beta} \right) \left(\xi_{i\alpha} \xi_{i\beta} - \delta_{\alpha\beta} \right) \right] \tag{2.20}$$

Unlike in Eq.(2.19), Eq.(2.20) shows that continuous velocity set is replaced with discrete velocity set and consequently, the distribution function becomes a set of distribution functions where each corresponding to one discrete velocity.
Eq.(2.20) can be further simplified by introducing the isothermal assumption and a new particle velocity $c_i = \frac{\xi_i}{\sqrt{3}}$. The final discrete equilibrium distribution function is:

$$f_i^{\mathrm{eq}} = w_i \rho \left(1 + \frac{c_{i\alpha} u_{\alpha}}{c_{\mathrm{s}}^2} + \frac{u_{\alpha} u_{\beta} \left(c_{i\alpha} c_{i\beta} - c_{\mathrm{s}}^2 \delta_{\alpha\beta} \right)}{2 c_{\mathrm{s}}^4} \right) \tag{2.21}$$

The mathematical derivations seem to be complex in this section, but the basic idea behind them is very straightforward, that is, firstly simplifying an exponential function (Eq.(2.10)) into a polynomial function with a finite number of terms and, then, discretizing the polynomial function in velocity space.

2.1.2.2 Discretization in space and time

In conventional CFD methods of FVM, the space discretization is flexible since a mesh cell or element can have an arbitrary shape. By contrast, LBM commonly adopts a uniform and structured grid. Let's start from the non-dimensional force-free Boltzmann equation with discrete velocities:

$$\frac{\partial f_i}{\partial t} + c_{i\alpha}\frac{\partial f_i}{\partial x_\alpha} = \Omega_i \tag{2.22}$$

It can be rearranged as an ordinary differential equation like Eq.(2.4):

$$\frac{\mathrm{d}f_i}{\mathrm{d}\zeta} = \left(\frac{\partial f_i}{\partial t}\right)\frac{\mathrm{d}t}{\mathrm{d}\zeta} + \left(\frac{\partial f_i}{\partial x_\alpha}\right)\frac{\mathrm{d}x_\alpha}{\mathrm{d}\zeta} = \Omega_i(\boldsymbol{x}(\zeta), t(\zeta)) \tag{2.23}$$

where:

$$\frac{\mathrm{d}t}{\mathrm{d}\zeta} = 1, \quad \frac{\mathrm{d}x_\alpha}{\mathrm{d}\zeta} = c_{i\alpha} \tag{2.24}$$

Integrate from $\zeta = 0$ to $\zeta = \Delta t$:

$$f_i\left(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t\right) - f_i\left(\boldsymbol{x}, t\right) = \int_0^{\Delta t} \Omega_i\left(\boldsymbol{x} + \boldsymbol{c}_i\zeta, t + \zeta\right)\mathrm{d}\zeta \tag{2.25}$$

The left-hand side of Eq.(2.25) is exact. The right-hand side cannot be evaluated exactly because the collision operator is not yet known. However, it is possible to approximate the right-hand side integration by first- or second-order discretization with unknown collision operator. It is proved that both first and second-order discretization can provide second-order accuracy in time [13, 14]. Taking the first-order discretization, the lattice Boltzmann equation (LBE) reads:

$$f_i\left(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t\right) = f_i(\boldsymbol{x}, t) + \Delta t\Omega_i(\boldsymbol{x}, t) \tag{2.26}$$

2.1.2.3 BGK collision operator

Until now, the fully discretized LBE of Eq.(2.26) is obtained. The only missing part of this jigsaw is the collision operator $\Omega_i(\boldsymbol{x}, t)$. The exact form of the collision operator is really complicated as it stands for all the possible outcomes of collisions. But as mentioned previously, the underlying microscopic information is not what we are mainly focusing on. The collision operator can accomplish its mission successfully as long as it can recover the macroscopic equations.
The most commonly used collision operator is the Bhatnagar-Gross-Krook (BGK) collision operator:

$$\Omega_i = -\frac{f_i - f_i^{\mathrm{eq}}}{\tau} \tag{2.27}$$

It is easy to interpret its physical meaning, that is, after a time τ, the population f_i will approach its equilibrium state f_i^{eq}. Combining Eq.(2.26) with Eq.(2.27) and making some rearrangements, it becomes:

$$f_i\left(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t\right) = \left(1 - \frac{\Delta t}{\tau}\right) f_i(\boldsymbol{x}, t) + \frac{\Delta t}{\tau} f_i^{\mathrm{eq}}(\boldsymbol{x}, t) \tag{2.28}$$

It can be noticed that f_i can be evolved to f_i^{eq} immediately or even beyond f_i^{eq} causing instability. It is found that $\tau/\Delta t \geq 1/2$ is a necessary condition for a stable LBM simulation.

2.1.3 Numerical implementation procedure of LBM

In Section 2.1.2, the basic derivations of LBM are introduced in a very concise way. For engineers, it is of high interest to apply LBM to real engineering problems. In this section, let us forget about these trivial formulas for a while and devote ourselves to summarising a whole process for implementing LBM numerically.

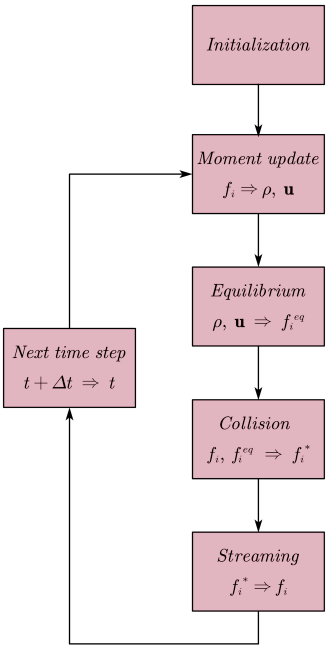


Figure 2.2: LBM algorithm

The LBM algorithm is divided into several steps:

- **Initialization**
  Usually, f_i is initialized by the initial value of $f_i^{\mathrm{eq}}(\rho(\boldsymbol{x},\ t=0),\ \boldsymbol{u}(\boldsymbol{x},\ t=0))$.
- **Moment update**
  The density and velocities can be updated in a similar way as Eq.(2.1) and Eq.(2.2) but replace the integration with summation over all populations.
- **Equilibrium**
  With the updated density and velocities, the equilibrium distribution can also be updated as Eq.(2.21).
- **Collision**
  Perform collision as the terms on the right hand side of Eq.(2.28).
- **Streaming**
  Perform streaming to obtain $f_i\left(\boldsymbol{x} + \boldsymbol{c}_i \Delta t,\ t + \Delta t\right)$ based on the terms on the right hand side of Eq.(2.28).
- **Go to the next time step**
  Go to the next time step and repeat the previous steps until it fulfils certain requirements.

2.1.4 Stability

We all want to perform a simulation without crashing caused by instability. In conventional CFD, the Courant–Friedrichs–Lewy number (CFL number) is a commonly used standard to predict whether a simulation will be stable or not. Usually, it should not be larger than 1. But for LBM, stability analysis is a more complicated task than that of conventional CFD. So here no mathematical description of stability analysis will be presented. Interested readers can refer to [15, 16] for more information. However, it is important for engineers to know some simple guidelines for improving the stability of simulations. This is exactly what we will look deeply in this section.

The Reynolds number is written as:

$$\mathrm{Re} = \frac{|\boldsymbol{u}| N \Delta x}{c_{\mathrm{s}}^2 \left(\tau - \frac{\Delta t}{2}\right)} \tag{2.29}$$

where $|\boldsymbol{u}|$ is the velocity magnitude, N is the number of lattice nodes along a characteristic length scale $N\Delta x$, Δx is the length of one node, and c_{s} is the sound speed of the isothermal model which can be calculated as $c_s^2 = (1/3)\,\Delta x^2/\Delta t^2$. In order to control the computational resource consumption, N is usually fixed to a proper number at the beginning of the simulation and then it remains unchanged. τ is adjusted to keep Re unchanged and $|\boldsymbol{u}| \ll c_s$. The fundamental procedure is shown in Figure 2.3. But remember that in the real situation when boundary conditions are applied, it is not so easy to guarantee stability even if a relaxation time τ fulfils all requirements in Figure 2.3. A possible way to increase stability is to use other advanced collision operators such as two-relaxation-time (TRT) [17] and multi-relaxation-time (MRT) [18].

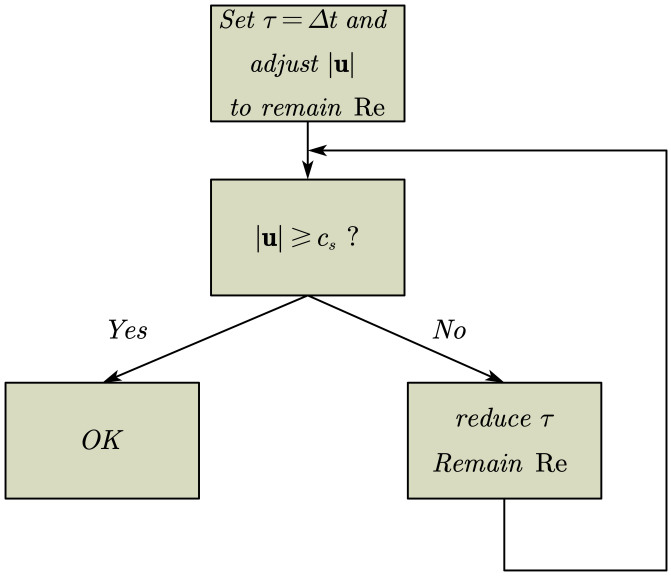


Figure 2.3: Stability

2.1.5 Accuracy

Many numerical errors may occur in LBM as in conventional CFD, namely, round-off error, iterative error, modeling error and discretization error.

- **Round-off error**
  This kind of error arises because of finite precision of computers. It can be mitigated by using more decimal digits.
- **Iterative error**
  Our strategy for solving steady state problems is to run the simulation for sufficient long time and take that solution as a converged solution. But LBM adopts an explicit time-marching procedure, which may cause the solution differ from the real steady state after a long running time.
- **Modeling error**
  The main sources of LB modeling error come from the chosen lattice type and the chosen form of the equilibrium distributions.
- **Discretization error**
  Approximating the partial differential equations (PDEs) by algebraic equations definitely causes discretization errors. The spatial discretization error is related to Δx^2, while the time discretization error is related to Δt^2.

2.1.6 Boundary condition

Like in conventional CFD, boundary conditions are also necessary for LBM to solve certain cases. Another requirement to solve LB equations is the initial condition. However, because in this project, only the long-time behaviour of the unsteady flow, which is not relevant to the exact choice of the initial condition is concerned, the initial condition will not be discussed. Three generally used boundary condition will be introduced in detail: periodic boundary, solid boundary using the bounce-back approach and symmetry (free-slip) boundary.

2.1.6.1 Two groups of LB boundary conditions

For beginners even with a fundamental knowledge of conventional CFD, it can be confusing to understand how boundary conditions are formulated in LBM. Like data can be cell- or node-centered in the conventional CFD, data in LBM can also be defined at lattice links (link-wise) or at lattice nodes (wet-node).

- **Link-wise boundary condition**
  It can be seen that in Figure 2.4, the physical boundary and computational domain do not coincide with each other. Instead, the boundary nodes that are the solid circles shift half a cell length inside. The lattice nodes can be seen as located at the center of each computational cell.
- **Wet-node boundary condition**
  The boundary nodes lie exactly on the physical boundary. The lattice nodes are at the vertices of the computational cells.

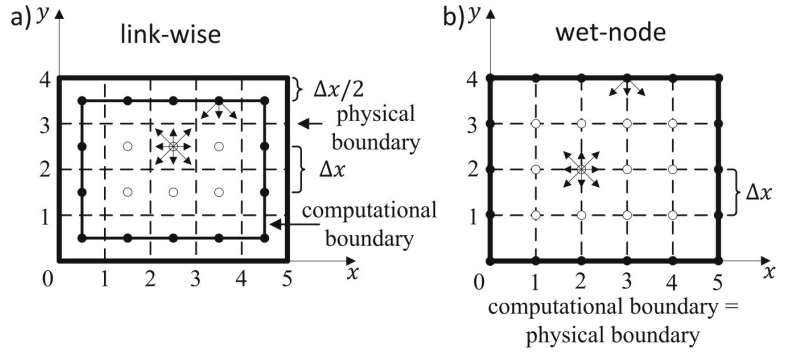


Figure 2.4: Two kinds of boundary conditions implementations in LB from [1]

2.1.6.2 Periodic boundary

The periodic boundary is widely used to simulate channel flow reaching fully developed state within a finite computational domain. By enforcing the unknown incoming populations f_i^* at the inlet to be identical to those at the outlet, the computational domain can be pretended to be infinitely long.
It can be seen that in Figure 2.5, there are virtual nodes at the inlet and outlet (at $x_0 = x_1 - \Delta x$ and $x_{N+1} = x_N + \Delta x$). The relations between the populations of the virtual nodes and those inside the domain are:

$$\begin{array}{ll} f_1^*(x_0, y_2, t) = f_1^*(x_N, y_2, t) & f_3^*(x_{N+1}, y_2, t) = f_3^*(x_1, y_2, t) \\ f_5^*(x_0, y_2, t) = f_5^*(x_N, y_2, t) \ , & f_6^*(x_{N+1}, y_2, t) = f_6^*(x_1, y_2, t) \ . \\ f_8^*(x_0, y_2, t) = f_8^*(x_N, y_2, t) & f_7^*(x_{N+1}, y_2, t) = f_7^*(x_1, y_2, t) \end{array} \tag{2.30}$$

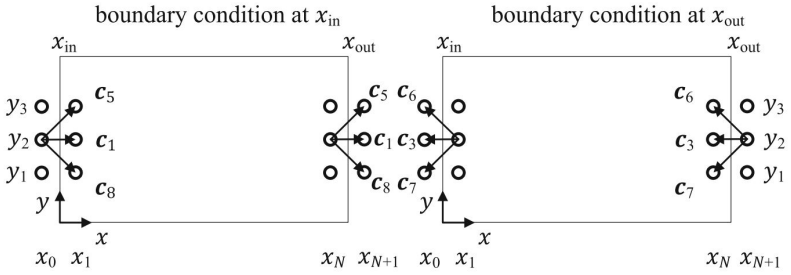


Figure 2.5: Periodic boundary condition from [1]. The computation domain starts from x_1 and ends at x_N. The virtual node exists at x_0 and x_{N+1}.

2.1.6.3 Solid boundary using bounce-back approach

The bounce-back method is an effective and simple way to deal with no-slip fluid-solid interface. The fundamental idea is quite intuitive: when the population hits the solid wall, it will be reflected back to the opposite direction as to their incoming one. As shown in Figure 2.6, the specific implementation for a bottom wall is:

$$\begin{aligned} f_2(\boldsymbol{x}_\mathrm{b}, t + \Delta t) &= f_4^*(\boldsymbol{x}_\mathrm{b}, t)\,, \\ f_5(\boldsymbol{x}_\mathrm{b}, t + \Delta t) &= f_7^*(\boldsymbol{x}_\mathrm{b}, t)\,, \\ f_6(\boldsymbol{x}_\mathrm{b}, t + \Delta t) &= f_8^*(\boldsymbol{x}_\mathrm{b}, t)\,. \end{aligned} \tag{2.31}$$

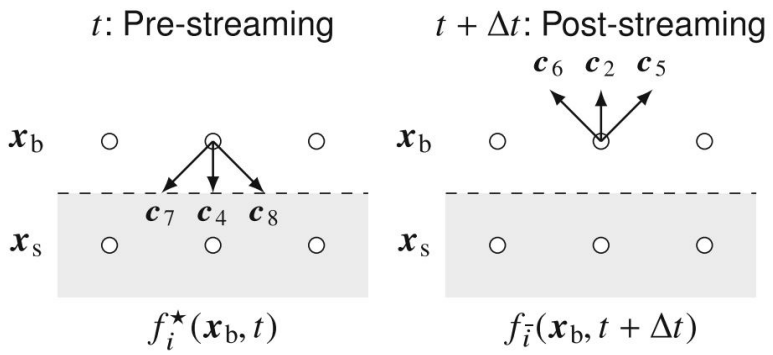


Figure 2.6: Bounce-back method at a bottom wall in [1]

2.1.6.4 Symmetry (free-slip) boundary

There is always a contradiction between the increasing requirement for accuracy and the limited computational resources. It will be beneficial to find a way to reduce the total number of nodes without affecting the accuracy. Symmetry is an inherent property of many fluid fields like Poiseuille flow. The symmetric flow can be halved by their symmetry axis. As shown in Figure 2.7, the implementation is:

$$\begin{aligned} f_7\left(\boldsymbol{x}_\mathrm{b}+\boldsymbol{c}_{7,\mathrm{t}}\Delta t, t+\Delta t\right) &= f_6^\star\left(\boldsymbol{x}_\mathrm{b}, t\right), \\ f_4\left(\boldsymbol{x}_\mathrm{b}+\boldsymbol{c}_{4,\mathrm{t}}\Delta t, t+\Delta t\right) &= f_2^\star\left(\boldsymbol{x}_\mathrm{b}, t\right), \\ f_8\left(\boldsymbol{x}_\mathrm{b}+\boldsymbol{c}_{8,\mathrm{t}}\Delta t, t+\Delta t\right) &= f_5^\star\left(\boldsymbol{x}_\mathrm{b}, t\right), \end{aligned} \tag{2.32}$$

where $c_{i,t}$ stands for the tangential velocity of the populations.

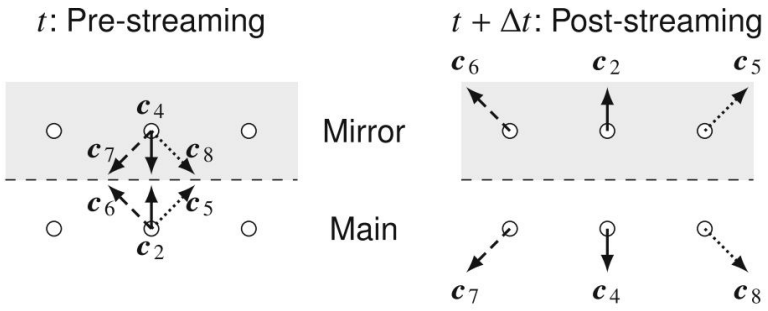


Figure 2.7: Symmetry (free-slip) boundary condition in [1]

2.1.7 Force

The flow cannot move without any energy input (no pressure, no force, no potential energy difference, etc). Force plays an important role in many hydrodynamic problems. For some incompressible flows whose driven mechanism is the pressure gradient, they can also be simulated equivalently by applying adequate body force. Body forces are easy to include in LBM. The only step is to add a procedure between

equilibrium and collision as in Figure 2.2, which calculates the source term:

$$S_i = \left(1 - \frac{\Delta t}{2\tau}\right) w_i \left(\frac{c_{i\alpha}}{c_\mathrm{s}^2} + \frac{\left(c_{i\alpha}c_{i\beta} - c_\mathrm{s}^2\delta_{\alpha\beta}\right) u_\beta}{c_\mathrm{s}^4}\right) F_\alpha \tag{2.33}$$

Then, the post-collision populations are calculated as:

$$f_i^\star = f_i + \left(\Omega_i + S_i\right) \Delta t \tag{2.34}$$

2.1.8 Smagorinsky subgrid model for LBM

The basic idea of LES is not only to resolve relatively large scales, but also to use a subgrid model to estimate the unresolved small scales. The spatial filtering is the step before applying the subgrid model. After spatial filtering, the LBE becomes:

$$\bar{f}_i\left(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t\right) = \bar{f}_i\left(\boldsymbol{x}, t\right) + \Delta t\bar{\Omega}_i\left(\boldsymbol{x}, t\right) \tag{2.35}$$

The equilibrium distribution function becomes:

$$\bar{f}_i^\mathrm{eq} = w_i\rho\left(1 + \frac{c_{i\alpha}\bar{u}_\alpha}{c_\mathrm{s}^2} + \frac{\bar{u}_\alpha\bar{u}_\beta\left(c_{i\alpha}c_{i\beta} - c_\mathrm{s}^2\delta_{\alpha\beta}\right)}{2c_\mathrm{s}^4}\right) \tag{2.36}$$

The Smagorinsky model relates the eddy viscosity to the local strain rate tensor $\bar{\mathbf{S}}$. The total effective viscosity is calculated as:

$$\nu_\mathrm{total} = \nu_0 + C\Delta^2|\bar{\mathbf{S}}| \tag{2.37}$$

with:

$$|\bar{\mathbf{S}}| = \sqrt{\bar{\mathbf{S}} : \bar{\mathbf{S}}} \tag{2.38}$$

$$\bar{\mathbf{S}} = \frac{1}{2}\left(\nabla\bar{\boldsymbol{u}} + \left(\nabla\bar{\boldsymbol{u}}\right)^T\right) \tag{2.39}$$

The relaxation time then becomes:

$$\tau_{total} = \tau_0 + \frac{3\Delta t}{\Delta x^2} C\Delta^2|\bar{\mathbf{S}}| \tag{2.40}$$

2.2 Wind energy

2.2.1 Forest modelling

The drag force caused by the existence of the forest is modeled with the help of the forest drag coefficient C_D, the leaf area density (LAD) a_f and the local wind speed $U = \sqrt{\bar{u}_i\bar{u}_i}$ [19].

$$F_{\mathrm{f},i} = -C_\mathrm{D}a_\mathrm{f}U\bar{u}_i \tag{2.41}$$

The forest drag coefficient C_D is usually taken between 0.15 and 0.2 [20, 21]. According to [22], the LAD distribution $a_f(z)$ can be described by the function:

$$a_f(z) = a_{f_m}\left(\frac{h - z_m}{h - z}\right)^n \exp\left[n\left(1 - \frac{h - z_m}{h - z}\right)\right] \tag{2.42}$$

where

$$n = \begin{cases} 6 & 0 \leq z < z_m \\ \frac{1}{2} & z_m \leq z \leq h \end{cases} \tag{2.43}$$

and z_m is the location where $a_f(z)$ gets its maximum value a_{f_m}. Taking $z_m = 0.7h$ and $a_{f_m} = 2.18$, the curve of $a_f(z)$ is shown in Figure 2.8.

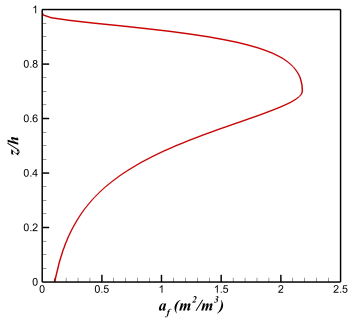


Figure 2.8: Leaf-area density

2.2.2 Coriolis

Another force that usually needs to be taken into account is the Coriolis force due to the rotation of the earth. It can be formulated as Eq.(2.44) [4].

$$F_{c,i} = 2\Omega \sin(\phi) \left(\bar{u}_j - u_{j,g}\right) \varepsilon_{ij3} \tag{2.44}$$

where Ω, $u_{j,g}$, ε_{ij3} and ϕ are the rotation rate, the geostrophic wind component, the alternating unit tensor and the latitude respectively. As the average velocity $\bar{u}_i$ varies along the height, the Coriolis force acting as a driven force also varies along the height. However, in this project, the driven force is simplified as a constant force. In the future, it is valuable to include the Coriolis force in the code to see how large effect it can bring to the simulation results.

3

Methodology

In this chapter, the methods used to carry out this project are presented, including the architecture of GPU, the features of the computational code GANSCANS and a practical way of generating turbulent flow.

3.1 GPU

The biggest difference between central processing unit (CPU) and GPU is that GPU contains more cores than CPU, but the clock speed of each core is significantly slower than that of CPU. For example, in Figure 3.1, one GPU comprises n multiprocessors and each multiprocessor has 12 cores. GPU is suitable to operate on large chunks of date in parallel as it is designed for data intensive applications. As for LBM, the collision step is local and the streaming step is merely a data shifting operation that does not need any computation at all. Because of these characteristics, LBM can straightforwardly take advantage of GPU's parallel computation features.

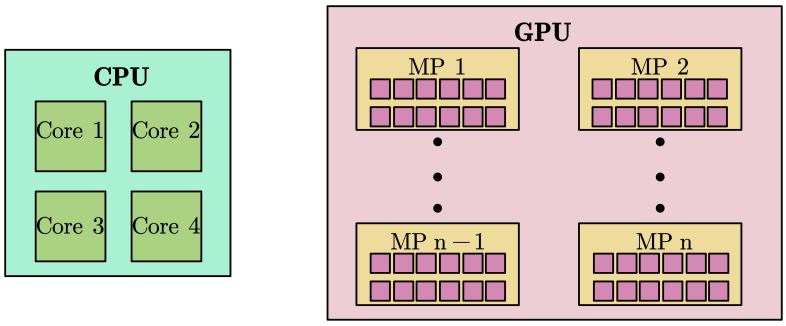


Figure 3.1: CPU and GPU architecture comparison (MP stands for multiprocessor)

3.1.1 CUDA

Compute Unified Device Architecture (CUDA) is a programming platform serving as a bridge between the hardware and the programming language. It was developed by Nvidia in order to allow easy realization of general computations on Nvidia GPUs [23]. Here, we will not give details about how it works as the goal of this thesis is not to code but to apply the code to solve real engineering problems.

3.2 Features of GANSCANS

The full name of GANSCANS is GPU-accelerated solver for coupled approaches to Navier-Stokes using C++ and CUDA [24]. It is a computationally efficient and user friendly LB solver that is mainly focused on turbulent flows [25]. It can make use of multiple GPUs but can only run on a single node [25]. All features listed in Table 3.1 [25] are available on a single GPU using double precision, but some features may not be able to run on multiple GPUs or using single precision right now. It needs to point out that GASCANS is highly capable of adding new features. For example, the forest model was implemented into GANSCANS in a short time for the requirement of this project. It is believed that in the future, GANSCANS can be developed to cope with more engineering application scenarios.

Table 3.1: GASCANS features

Feature	Multi GPU	Single precision
Core LB solver	Yes	Yes
Point cloud reader	Yes	Yes
Body force	Yes	No
Momentum exchange	No	Yes
Immersed boundary method	No	No
LES Smagorinsky model	Yes	Yes
SEM boundary condition	Yes	Yes
Time averaging	Yes	Yes
Sponge layer	Yes	Yes
Coupling with another code	Yes	No
Restart a simulation	Yes	Yes

3.3 Turbulence generation

It may seem to be out of the scope of this thesis to investigate how to generate turbulence. But turbulence generation is definitely an important part in order to carry out the simulation successfully. Otherwise, the flow field could remain laminar or take a long time to become fully turbulent, which is infeasible given the limited simulation time. A common way of turbulence generation is to use synthetic turbulence as the inlet boundary condition [26], but it is difficult to apply to our present case. Inspirations were got from Anika et al. [2]. In their work, they put consecutive and staggered half cubes inside a channel to trigger turbulence at a relatively low Reynolds number (see Figure 3.2). Actually, from all cases that have been tested (Figure 3.3), only the case with the cube throughout the entire spanwise length (Figure 3.3a) failed to generate turbulence. Because of the limited time and resources of this project, the influence of the height of the cube on turbulence generation has not been studied yet, but it is an enlightening research topic for the future.

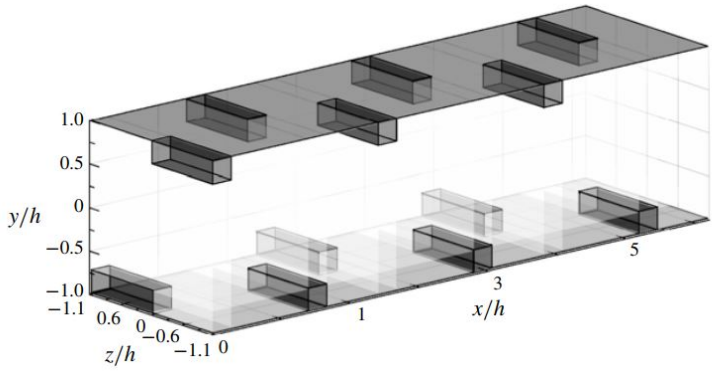


Figure 3.2: The channel with consecutive and staggered half cubes from [2]

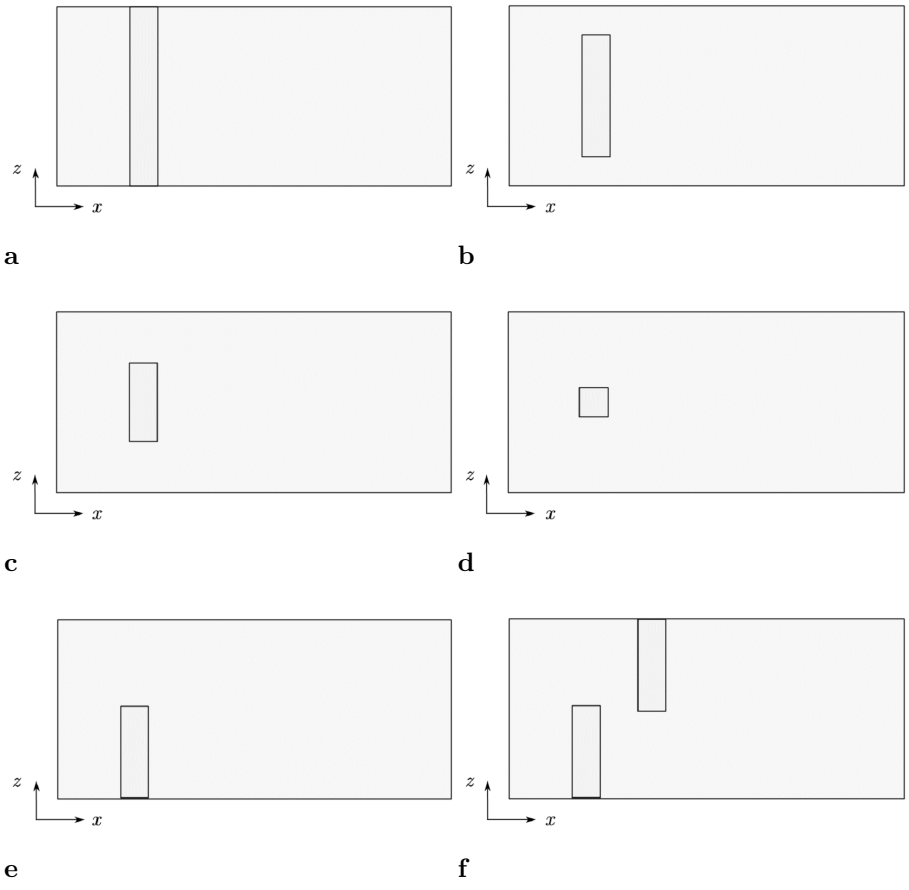


Figure 3.3: Cubes with different spanwise length tested, top view

4

Results and Discussion

4.1 Turbulent channel flow - validation case

Fully developed turbulent channel flow is a commonly used benchmark problem to test new algorithms and validate codes. Direct numerical simulation (DNS) data that are available for comparison can be traced back to early studies, for example, in [27]. This turbulent channel flow case acts as an important role to make sure that GANSCANS can give reasonable simulation results.

4.1.1 Simulation setup

The configuration of the computational domain is shown in Figure 4.1. The domain has $12H$ in the streamwise direction, $2H$ in the vertical direction and $4H$ in the spanwise direction. The upper and lower wall are set to the no-slip boundary condition, and periodic boundary condition for the other boundaries. The grid resolution is 31, which means that over the length of H, there will be 31 nodes.

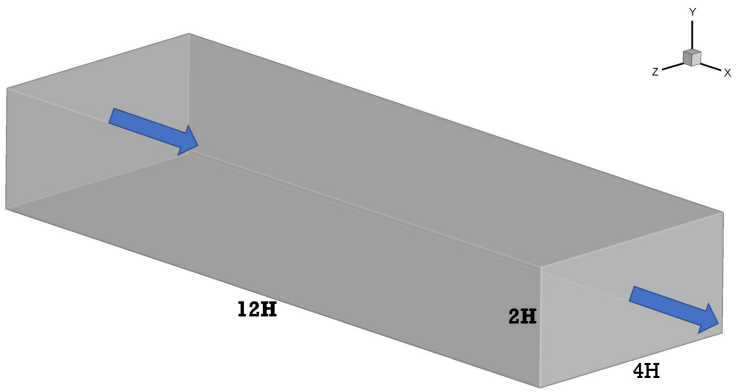


Figure 4.1: Computational domain configuration for turbulent channel flow

The flow is driven by a pressure gradient. Since the flow is fully developed, the pressure gradient should be balanced with the wall shear stress according to the following equation:

$$-\frac{\partial p}{\partial x}H = \tau_w \ , \tag{4.1}$$

where H stands for the half height of the channel. Moreover, the friction velocity

u_τ is defined as:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \ . \tag{4.2}$$

Substitute Eq.(4.2) into Eq.(4.1),

$$-\frac{\partial p}{\partial x} = \frac{\rho u_\tau^2}{H} \tag{4.3}$$

As mentioned in Section 2.1.7, the pressure gradient is usually replaced by a equivalent body force:

$$G = \frac{\rho u_\tau^2}{H} \ . \tag{4.4}$$

In order to compare with the DNS data from Moser et al. [27], the Reynolds number Re_τ as in Eq.(4.5) is set to around 180.

$$\mathrm{Re}_\tau = \frac{u_\tau H}{\nu} \tag{4.5}$$

The parameters set for the simulation are listed in Table 4.1. It needs to be noticed that C is the Smagorinsky model constant as in Eq.(2.37). Because the the flow is incompressible, the density is set to a default value which is equal to 1. As a result, the corresponding Re_τ is 177.6 theoretically.

Table 4.1: Simulation parameters for turbulent channel flow

Parameter	value	Parameter	value
Resolution	31	size X [m]	12
Time step [s]	0.001	size Y [m]	2
Kinematic viscosity ν $[m^2/\mathrm{s}]$	1/3250	size Z [m]	4
body force [N]	0.002986	C	0.01

Although the configuration of the channel flow is very simple, it is still hard to simulate. One big issue is how to trigger turbulence. Here the inspiration is got from Anika at al. [2] and more details are given in Section 3.3. A cuboid with half of the width of the channel, as shown in Figure 4.2, is put near the inlet to initialize turbulence. After 1 million time steps, the cuboid was removed and the simulation ran for another 3 million time steps before starting the time average.

4.1.2 Results

The first thing before diving into the simulation results is to make sure that the flow is fully developed. Figure 4.3 shows the velocity fluctuations in the last 50000 time steps of the simulation. The values of the velocity oscillate around the mean value which is about $0.03dx/dt = 0.968$m/s. It is safe to say that the flow is fully developed.
Figure 4.4a shows the instantaneous velocity distribution. Although the grid is relatively coarse, many turbulent features can still be observed, such as streaks in the near-wall plane. The Q-criterion is a vortex identification method [28]. The

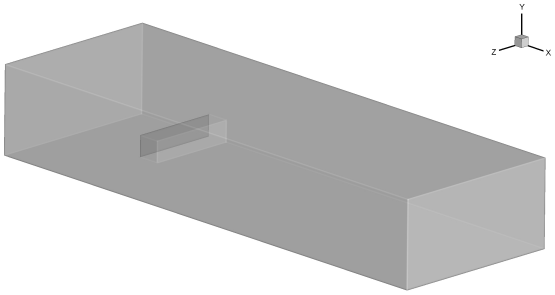


Figure 4.2: To trigger turbulence

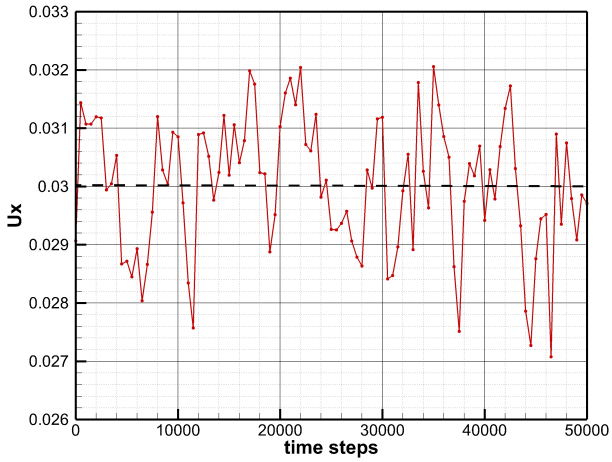
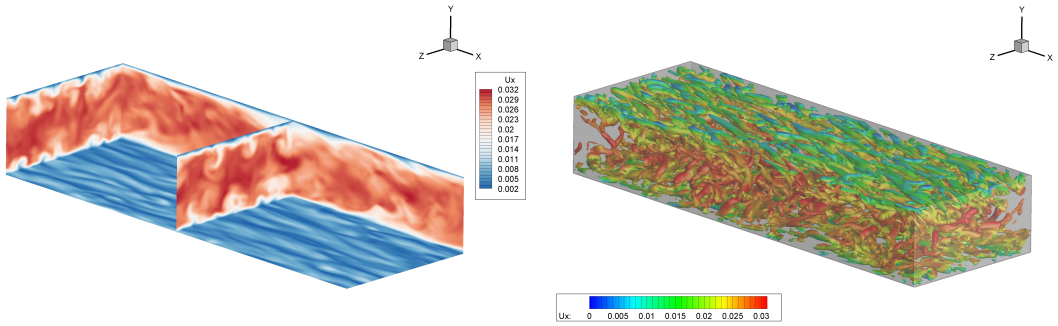


Figure 4.3: Fluctuations of velocity in the x direction in the last 50000 time steps



a **b**

Figure 4.4: (a) Visualization of instantaneous turbulent channel flow field. The bottom plane is at $y+ = 9$. (b) The isosurface of Q = 0.0002

isosurface of $Q = 0.0002$ is shown in Figure 4.4b. It can be seen that the code is able to resolve small turbulent structures.
The time-averaged results are compared with the DNS data of Moser et al. [27] and the LBM data from Koda et al. [29] quantitatively (see Figure 4.5a, 4.5b and 4.5c). Generally, the results of our present work show good agreement with the LBM data obtained by Koda et al. [29], especially in the near wall region. The discrepancy tends to be larger as y^+ increases from 10 to 100, but the largest mean velocity error is less than 5% and the largest root mean square (RMS) velocity error is less than 10%. The difference between DNS and LBM data is relatively large. The LBM simulation result significantly underestimates the mean velocity. According to Koda et al. [29], this underestimation of the mean velocity is due to the underestimation of the mass flow rate that is caused by the coarse mesh and the lack of the wall-damping function in the Smagorinsky subgrid-scale model.
All in all, GASCANS is validated through the case of turbulent channel flow. Although the results from GANSGANS show difference from DNS data, this is actually acceptable considering that GANSGANS used fewer nodes compared to DNS. It can be said with confidence that the results will be better if the mesh resolution is increased.

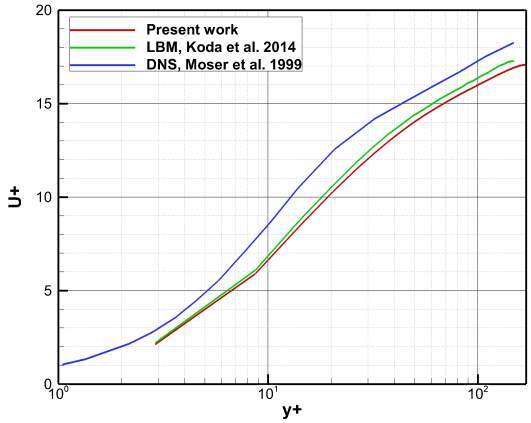
4.2 Atmospheric boundary layer

In the previous chapter, the turbulent channel flow was studied in order to verify the capability of GANSCANS. However, for real wind energy application scenarios, channel flow conditions may not be fully applicable. In this section, one of the external flow systems, the atmospheric boundary layer (ABL) is considered. Moreover, when deciding the location of a wind farm, a flat terrain without forest is definitely the first choice. However, this kind of ideal terrain is hard to find, especially in the current situation that wind energy is under high speed expansion worldwide. A compromise must be made, that is, installing wind turbines in complex terrain with hills or forest or both. This is why the influences of hills and forests on ABL are the main focus of this section.

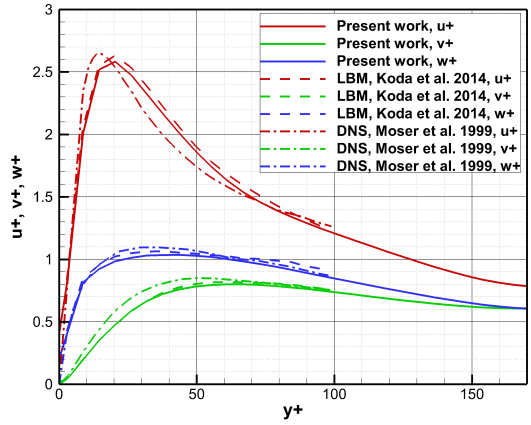
4.2.1 Influence of hills

4.2.1.1 Simulation setup

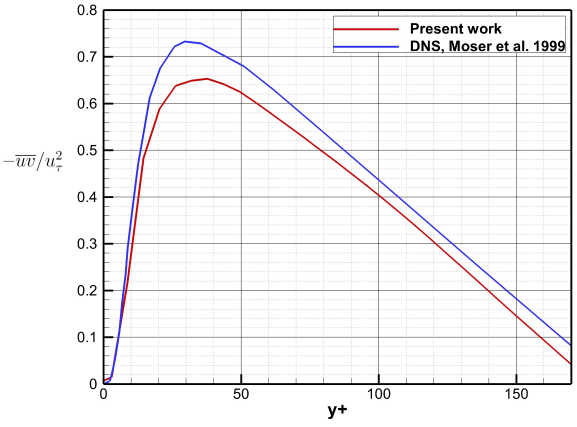
The turbulent flow over a circular hill has been thoroughly studied by Ishihara et al. [30, 31] both experimentally and numerically, so detailed data are available for comparison. The hill is a three-dimensional hill with a cosine-square cross section that can be described in Eq.(4.6). The profile of the cross section of the hill can be seen in Figure 4.6a. The height of the hill is $H = 40$ mm and the base radius L is 100 mm. The dimensions of the computational domain are 1000 mm, 800 mm and 600 mm for length, width and height respectively. The hill is located in the middle of the bottom (see Figure 4.6b). The periodic boundary condition is applied to the inlet and outlet as well as the lateral walls. The upper wall is assumed to be a free-slip wall while the lower wall and the surface of the hill are using the



a



b

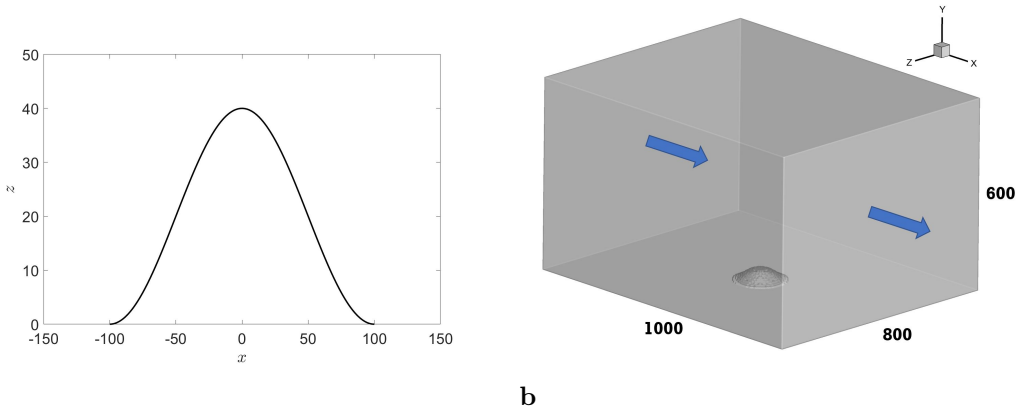


c

Figure 4.5: (a)Mean velocity profile, $U^+ = U/u_\tau$, $y^+ = yu_\tau/\nu$. (b) RMS velocity profiles of $\overline{u'u'}/u_\tau^2$, $\overline{v'v'}/u_\tau^2$ and $\overline{w'w'}/u_\tau^2$(c) Profile of $-\overline{u'v'}/u_\tau^2$

bounce-back boundary condition. The size of the cell is kept constant everywhere with $\Delta_x = \Delta_y = \Delta_z = 4$mm. The flow is driven by a constant body force in the x-direction. The parameters and their values for this simulation are tabulated in Table 4.2 for clarity.
The most important thing is that the Reynolds number of the simulation and the experiment must be kept similar to make their results comparable. According to the experiment of [30], the Reynolds number $U_\infty \delta/\nu$ of the simulated boundary is equal to 1.4×10^5. It needs to be mentioned that δ here is the boundary layer thickness at the location where the hill is mounted. For the numerical simulation, the Reynolds number with the same definition is about 3.6×10^5. Although the Reynolds numbers are not exactly the same, they are of the same order of magnitude. Therefore, it is quite reasonable to compare their results.

$$z = \begin{cases} H \cos^2\left(\pi \frac{\sqrt{x^2+y^2}}{2L}\right) & , \sqrt{x^2+y^2} < L \\ 0 & , \sqrt{x^2+y^2} \geq L \end{cases} \tag{4.6}$$



a **b**

Figure 4.6: (a) The profile of the cross section of the hill. (b) The computational domain of the simulation.

Table 4.2: Simulation parameters for the atmospheric boundary layer over a hill

Parameter	value	Parameter	value
Resolution [mm^{-1}]	0.25	size X [mm]	1000
Time step [s]	0.1	size Y [mm]	600
ν [$\text{mm}^2 \cdot \text{s}^{-1}$]	2×10^{-4}	size Z [mm]	800
body force [$\text{kg} \cdot \text{mm} \cdot \text{s}^{-2}$]	2×10^{-6}	C	0.01

4.2.1.2 Mean flow field

The effect of the hill on the flow field is investigated by examining the mean flow velocity normalized by the free-stream velocity in the middle plane of the z-direction.

In Figure 4.7, the velocity field is shown by a colored map, and the streamlines are plotted to visualize the recirculation region after the hill. It can be seen that the flow velocity is almost uniform from $y/H = 5$ to 15. In the region near the hill, the velocity decreases and a recirculation region is formed after the hill. It is worth noting that the profile of the hill is not perfectly smooth. Instead, the hill surface looks like stair steps, which is due to the low resolution of the mesh. It

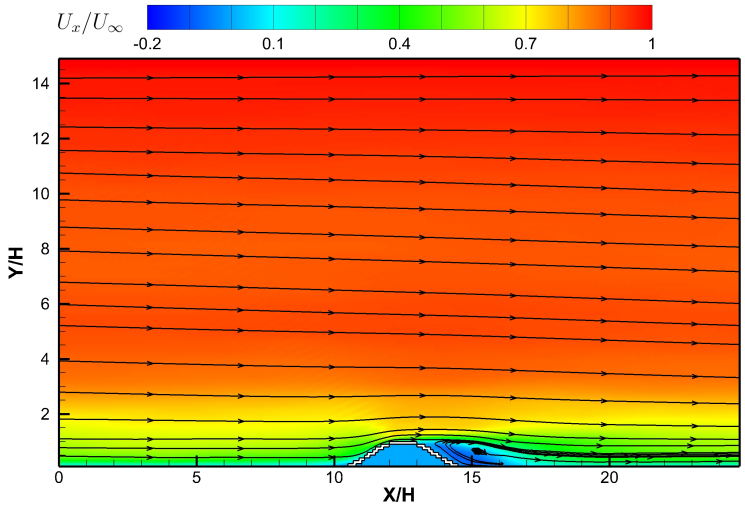
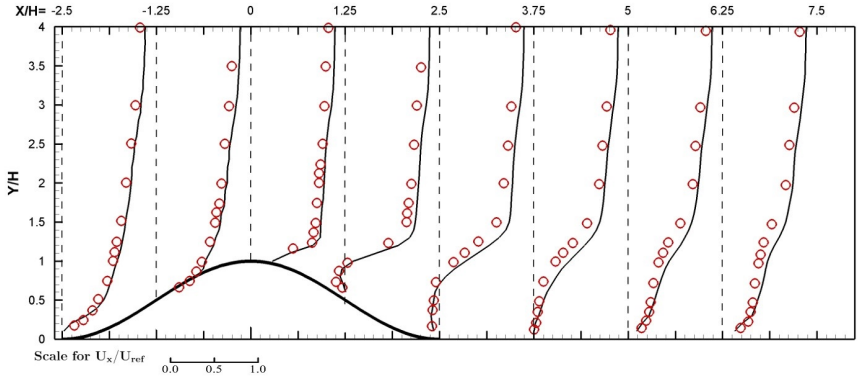


Figure 4.7: Streamlines in the middle plane of z direction, colored by normalised mean streamwise velocity

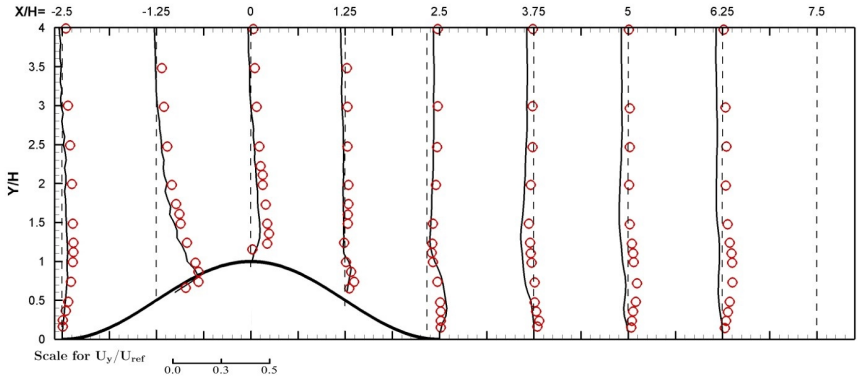
is also essential to evaluate the results quantitatively by comparing them with the experimental data which come from [30]. Figure 4.8 and 4.9 show the mean velocity profiles in the middle plane of the z-direction and the plane of $x/H = 0$,respectively. For the mean velocity in the x-direction (Figure 4.8a), the simulation results seem to match the experimental data very well at first glance. However, the normalized U_x of the simulation shows an overall overestimation compared to the experimental data. For the normalized vertical velocity, the discrepancy between the simulation and the experiment is much larger. On the windward side of the hill, the simulation slightly underestimates the normalized U_y. As it goes to the downstream of the hill, the mismatch becomes more and more significant. At the locations of $x/H = 5$ and 6.25, the simulation results are even unable to predict the sign of the normalized U_y correctly. At the $x/H = 0$ plane, the flow does not separate. In Figure 4.9a, it is noticed that at $z/H = -2.5$, the simulation significantly undervalues the normalized U_x near the hill. The situation is similar for the normalized U_z.

4.2.1.3 Turbulence field

Besides the mean velocity, another interesting topic is to investigate the turbulence field. The experimental data of the normal stresses σ_u, σ_v and σ_w are available in [30]. The normalized normal stress components in the central plane of the domain as well as in the $x/H = 0$ plane are plotted in Figure 4.10 and Figure 4.11 respectively. There are considerable differences between the simulated and experimental σ_x, as

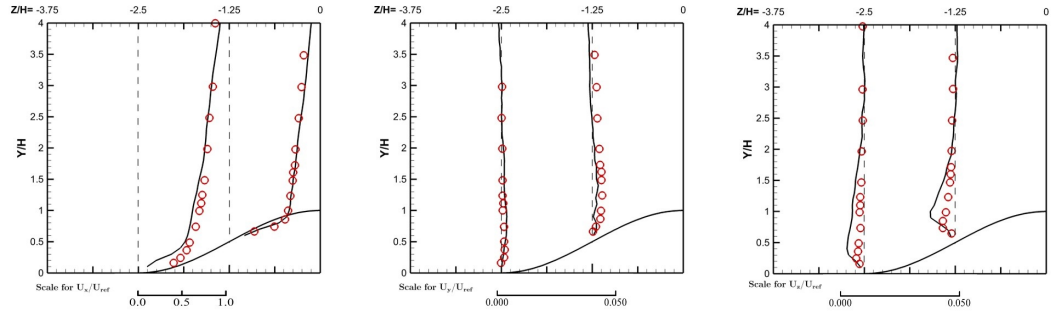


a



b

Figure 4.8: Comparison of the simulated and measured mean velocity profiles in the central plane of the domain: (a) the streamwise velocity; (b) the vertical velocity. The red spots stand for the experimental data, and the black lines are the simulation results.



a **b** **c**

Figure 4.9: Comparison of simulated and measured mean velocity profiles in the $x/H = 0$ plane: (a) streamwise velocity; (b) vertical velocity; (c) spanwise velocity. The red spots stand for the experimental data and the black lines are the simulation results.

shown in Figure 4.10a. On the windward side of the hill, namely, at $x/H = -2.5$ and $x/H = -1.25$, the σ_x is underestimated by the simulation not only in the near hill region but also in the region far away from the hill. At the summit of the hill, the simulation correctly predicts the maximum value of σ_x at the height of $y/H = 1$. However, the simulation undervalues σ_x when it goes to the region where $y/H > 1$. As it goes to the downstream region of the hill, the simulation still fails to predict the maximum value of σ_x. The simulation performs better for σ_y (Figure 4.10b). The only significant discrepancy is at $x/H = 1.25$ where the simulation is unable to give the correct peak value of σ_y. At other locations, the simulation and the experiment agree well with each other with small underestimation of the simulation. The simulation also gives satisfactory results for σ_z (Figure 4.10c) at most of the locations expect in the near-hill region at $x/H = 1.25$, where σ_y is underrated remarkably. The simulation shows an overall underestimation of the normal stress components in the $x/H = 0$ plane (Figure 4.11). The underestimation is generally lager in the near-hill region than in the far-hill region.

4.2.2 Influence of forest

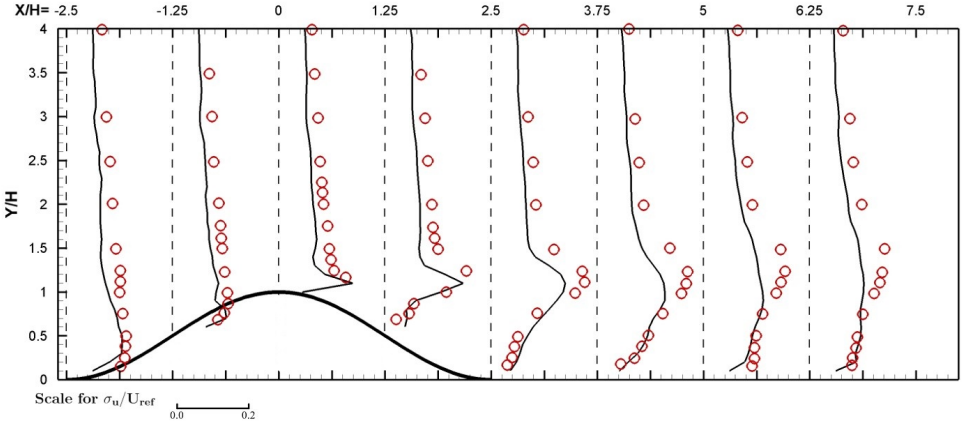
In this section, a domain with homogeneous forest is investigated. The results of the simulation are compared with the experimental data. The effect of the forest is also discussed.

4.2.2.1 Simulation setup

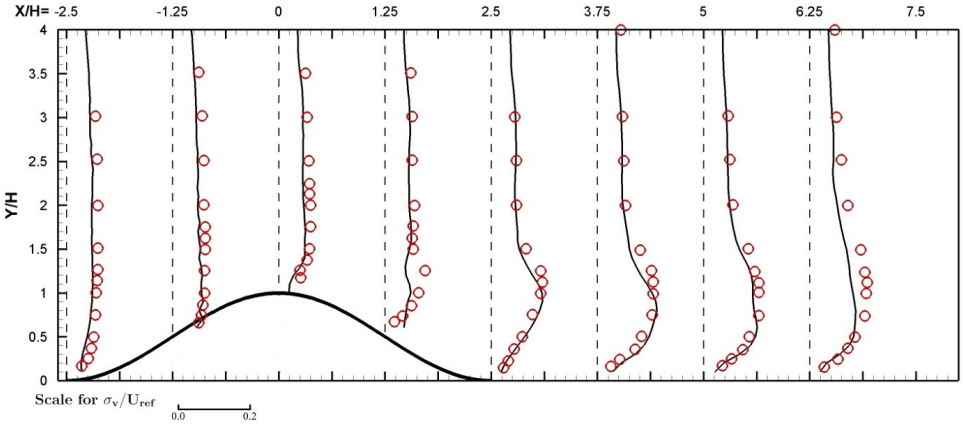
The computational domain is shown in Figure 4.12. The dimensions of the domain are 1000m, 800m and 400m for the length, width and height respectively. The homogeneous forest covers the whole ground of the domain with a height h of 20m. The flow is driven by a constant body force 2×10^{-5} N. Both of the inlet and outlet are using the periodic boundary condition while the upper wall is using the forced equilibrium boundary condition, which is equivalent to fixing the velocity of the upper wall to be 1m/s. Because the upper wall is sufficiently far away from the bottom and only the positions near to the wind turbines are of great concern, forced equilibrium boundary condition works well for this case. In reality, the LAD varies with the height as the crown region has dense leaves and the trunk region is less obstructing. But in this case, the simplified constant LAD is also tested together with the varying LAD. The distribution of the constant LAD is assumed to be a straight line which can cover the same amount of area as the varying LAD curve (see Figure 4.13). The value of the constant LAD is then 0.215.

4.2.2.2 Resolution and time consumption study

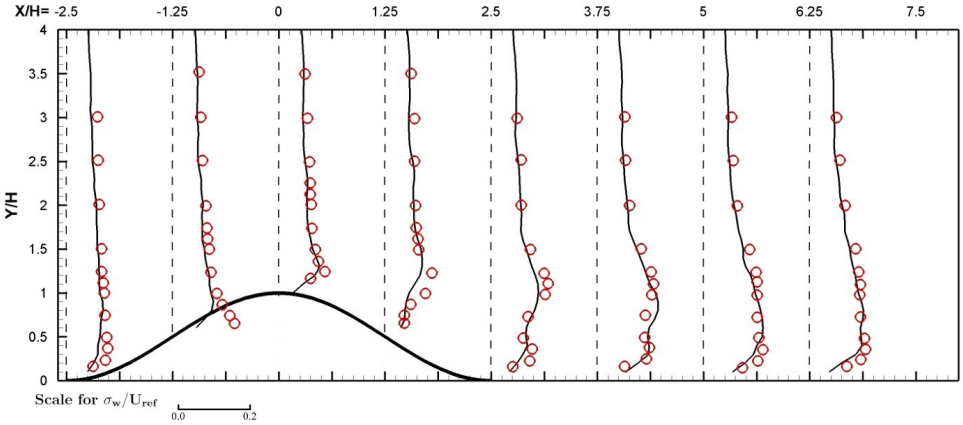
A rapid design process to optimize a prototype in a short period is the pursuit of industries, and the wind industry is not an exception. Due to the limitation of the computational time and hardware requirements, it is of utmost importance that we have an overall estimation of the computational resource consumption before we start the simulation. Table 4.3 shows all the resolutions tested in the simulation and the corresponding time and memory consumption. The GPU type used here



a

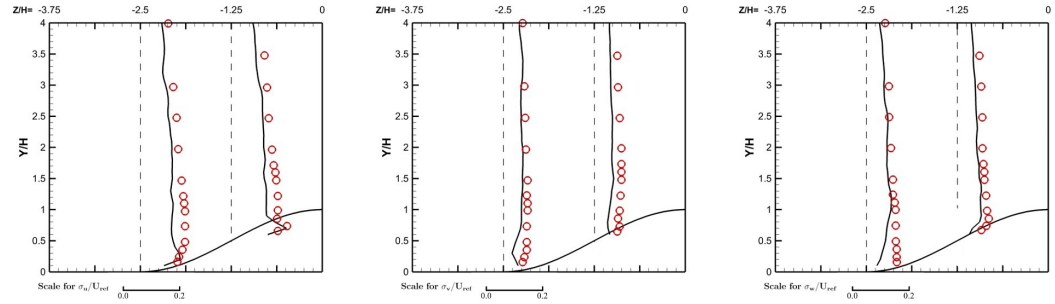


b



c

Figure 4.10: Comparison of simulated and measured normal stress profiles in the central plane of the domain: (a) the streamwise velocity; (b) the vertical velocity; (c) the spanwise velocity. The red spots stand for the experimental data and the black lines are the simulation results.



a **b** **c**

Figure 4.11: Comparison of simulated and measured normal stress profiles in the $x/H = 0$ plane: (a) the streamwise velocity; (b) the vertical velocity; (c) the spanwise velocity. The red spots stand for the experimental data and the black lines are the simulation results.

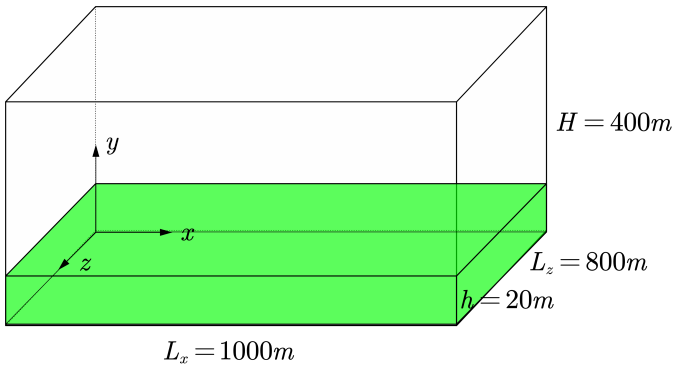


Figure 4.12: The computational domain with homogeneous forest in green color.

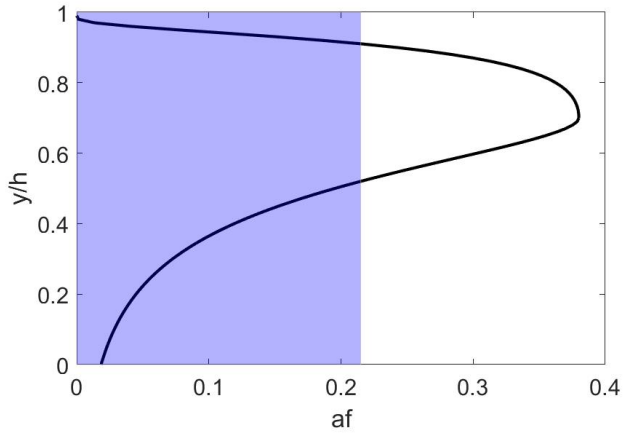
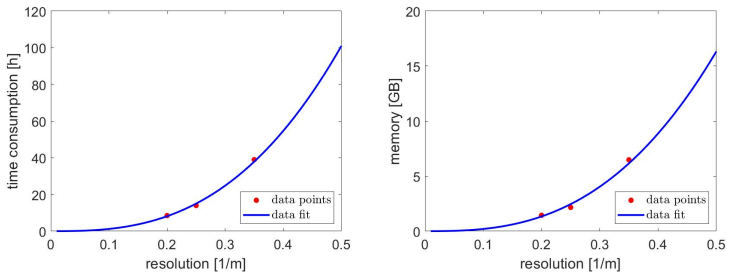


Figure 4.13: Leaf area density. The area under the blue shade is the same as the area under the black curve. The black curve follows Eq.(2.42) with $z_m = 0.7h$ and $a_{f_m} = 0.38$.

is GeForce RTX 2070 SUPER with 8GB memory. As can be seen from Figure 4.14b, the maximum resolution that this GPU can handle for this case is about 0.38. Both time and memory consumption increase exponentially with the increase of resolution. It is noticed that the time step in Table 4.3 also varies for different cases. This is because whenever the spatial resolution is refined or coarsened, the diffusive scaling $\Delta t \propto \Delta x^2$ should usually be obeyed. The time step length of the highest resolution 0.35 here is used as a standard. The time steps of the other two resolutions are chosen accordingly.

Table 4.3: The relation between resolution, time and memory consumption

Resolution	Grid size	Time step	Time per 1M steps	Memory
0.2	200x80x160=2.6M	0.3s	8.5h	1.44GB
0.25	250x100x200=5M	0.2s	14h	2.16GB
0.35	350x140x280=13.7M	0.1s	39h	6.48GB



a **b**

Figure 4.14: (a) The relation between resolution and time consumption. (b) The relation between resolution and memory consumption. The red points stand for three simulated cases and the blue curve stands for the result of data fitting.

The accuracy of the simulation results is another important factor in the choice of resolution, besides time and memory requirements. Figure 4.15 shows the results of three different resolutions. These results are compared with the experimental data from Bergström at al.[3]. The mean horizontal velocity $M = \sqrt{U^2 + W^2}$, normal stresses as well as shear stresses are all non-dimensionalized by the friction velocity $u_* = \left(\overline{u'v'}^2 + \overline{v'w'}^2\right)^{1/4}$ at location $y/h = 2$. Generally, the higher the resolution, the better the results, especially for the streamwise normal stress $\overline{u'u'}$. Therefore, in Section 4.2.2.3 the resolution of 0.35 is selected and its results are analyzed in detail.

4.2.2.3 Results

Figure 4.16 shows a detailed comparison between LBM results, conventional CFD results (Nebenfuhr et al., 2014 [4]) and experimental data (Bergström et al., 2013 [3]). From the perspective of mean velocity, as shown in Figure 4.16a, the results of

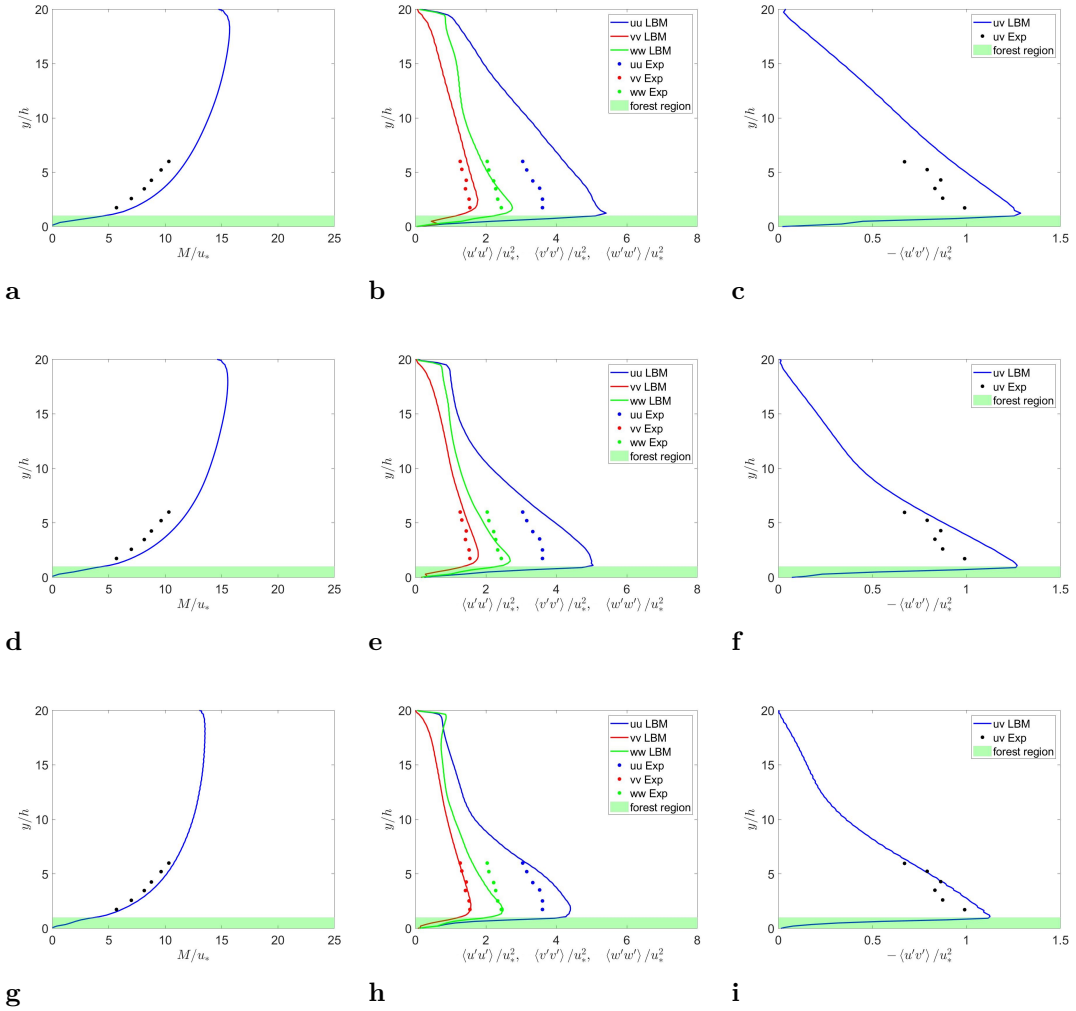
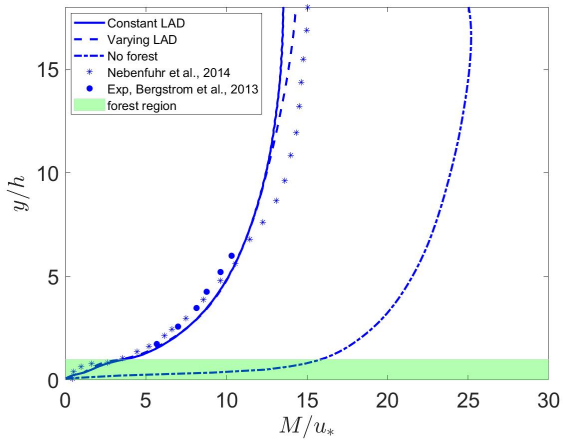


Figure 4.15: Comparison between the different resolutions. From top to bottom, the rows are the results for resolution 0.2, 0.25 and 0.35 respectively. From the left to right, the columns are the time- and space-averaged mean horizontal velocity profile, the normalized normal stresses and the normalized shear stress, respectively. The experimental data come from Bergström et al.[3]

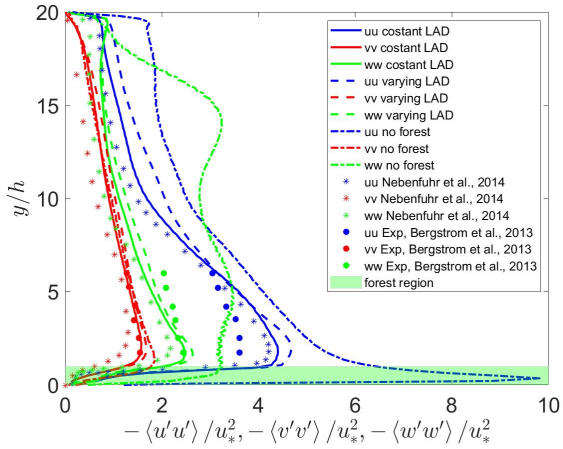
LBM simulation show good agreement with that of conventional CFD and experimental data. The influence the forest on the mean velocity is quite obvious: in the forest area where y/h is below 1 (green area in the plot), the velocity gradient of the case without forest is much higher than that of the case with forest, which means that the forest can hinder the flow in the forest region. And the influence of the forest also continues above the forest region. There is almost no difference between the mean velocities of the constant LAD and varying LAD, except in the region near the upper boundary. Figure 4.16b shows the normal stresses in the streamwise, spanwise and vertical direction. The vertical normal stress does not change significantly with or without the forest. However, this is not the case for normal stress in the spanwise direction. It is very strange that in the case without forest, the spanwise normal stress has an abnormal bulge near the area $y/h = 15$. The reason could be that the time of the data used in the time-averaged computation is not long enough. In the streamwise direction, the effect of the forest is most significant. The largest non-dimensional streamwise normal stress in the case without the forest is almost 10, whereas the case with the forest shows the stress lower than 5. Figure 4.16c shows the shear stress. The shear stresses in all those cases show similar trend and maximum value. The curves of the constant LAD and varying LAD almost completely overlap with each other in the forest region.

4.3 Hornamossen

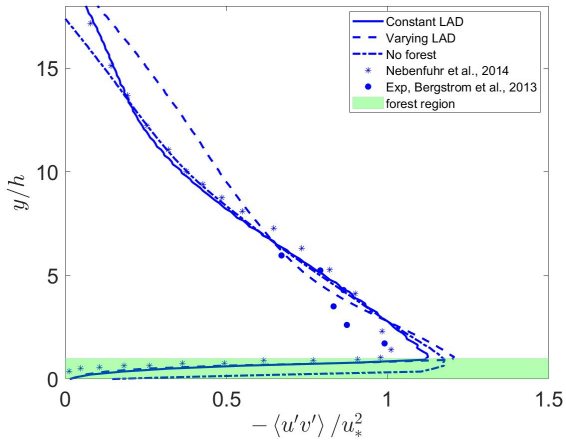
As the capability of GANSCANS to handle forest and complex terrain has been tested in the previous two cases, now it is the time to apply GANSCANS to the Hornamossen case that was initiated within the New European Wind Atlas (NEWA). The task is to simulate the flow in a complex forested terrain and to see if the model can provide acceptable wind profiles at several validation locations. The contour of the landscape and the distribution of the canopy height and LAD are shown in Figure 4.17. Different from the previous homogeneous forest, the forest cover of Hornamossen is highly heterogeneous with the canopy height and LAD varying in different locations and heights. One similar study to the Hornamossen case is the study carried out by Schubiger et al. [32].
For easier understanding of the readers, it is better to illustrate different levels of simplifications that can be done to simulate the heterogeneous forest case, as shown in Figure 4.18. It is worth noting that the simplest situation, i.e., the homogeneous forest cover and constant LAD over the forest height, is not displayed in this figure. This simplest situation excludes complex terrain as well as highly heterogeneous forest covers in real cases. A relatively complex situation is to adopt a homogeneous forest but with a varying LAD over the forest height, as shown in Figure 4.18a. But it still does not consider the geometry of the complex terrain. Sometimes it is possible to get satisfied results even if the forest cover is not considered at all, as in Figure 4.18b. There are several kinds of simplifications when the forest cover is taken into account. Figure 4.18c shows a mountain with a forest cover, but the forest cover is quite unusual since the height of it keeps the same everywhere on the mountain and LAD is a constant over height. Figure 4.18e has the same forest configuration as Figure 4.18c. The only difference is that in Figure 4.18e, the LAD changes over



a



b



c

Figure 4.16: Comparison of the mean velocity, normal stresses and shear stress with the conventional CFD results (Nebenfuhr et al., 2014 [4]) and the experimental results (Bergström et al., 2013 [3])

height. Figure 4.18f gives the most complex situation, that is, the vegetation types are different at different locations on the mountain. In some places, there is even no vegetation cover at all. Consequently, LAD changes not only over the height but also across different locations.

4.3.1 Simulation setup

Hornamossen covers a large area of 40 by 40 kilometres. It would be certainly better to take the whole topography into consideration, but because of limited time and computer resources, it is decided to simulate a 12 by 5 kilometres terrain, as shown in Figure 4.17b, 4.17d and 4.17e. The small domain can cover most of the validation points (listed in Table 4.4) as well as the turbine locations (listed in Table 4.5).

Figure 4.19 shows the 3D representation of the computational domain and the boundary conditions. The height of the domain is 1000 m. The upper boundary uses symmetry boundary condition (see Section 2.1.6.4), and the bottom boundary with the no-slip wall boundary condition (see Section 2.1.6.3). The front wall and the back wall are periodic (see Section 2.1.6.2). The inlet is imposed with the velocity inlet boundary condition, which is set with a velocity of 1m/s for the initial condition. However, it will change generally as time goes by because a constant force is exerted on the whole domain. At the outlet, the gradients of velocity and density are zero. In order to generate turbulent flow near the inlet, three cuboids are put near the inlet. It is noted that the cuboids should not be put too close to the inlet (as shown in Figure 4.20b). Otherwise, the cuboids may act as obstacles that force the flow to return at the inlet. Also, the cuboid should not be too large or too close to the region with the forest and the mountain. Otherwise, the recirculation regions of the cuboids may extend into the forest and mountain region. According to the numerical tests conducted in this thesis, a proper, but definitely not unique, combination of the cuboid size and position is proposed: the size of the cuboid is $150 \times 150 \times 150$ m and the cuboids are 500 m from the inlet. For the sake of brevity, the testing cases are not presented here. To save the necessary height of the computational domain, the absolute heights of the mountain and forest are deducted together by 205 m. The resolution of the mesh is 0.06, which means the sizes of a cell in all three dimensions are about 16.67 m. The time step length is set to 0.1s.

A stationary case is simulated with the wind coming from the west and blowing towards the east. The geostrophic wind speed and direction are 11.7m/s and 264 degrees, respectively. As a simplification made here, the wind direction of 270 degrees is assumed here to facilitate the setup of boundary conditions. Because there have been no existing formulas or similar simulations in the literature that are used to predict a constant body force driving the flow, the force is determined by trial and error to be 10^{-4} N in the present study. Another simplification made is that a constant LAD of 0.215 is applied to the entire forest cover (see Figure 4.18d) because GANSCANS cannot deal with more complicated situations such as the cases shown in Figure 4.18e and 4.18f.

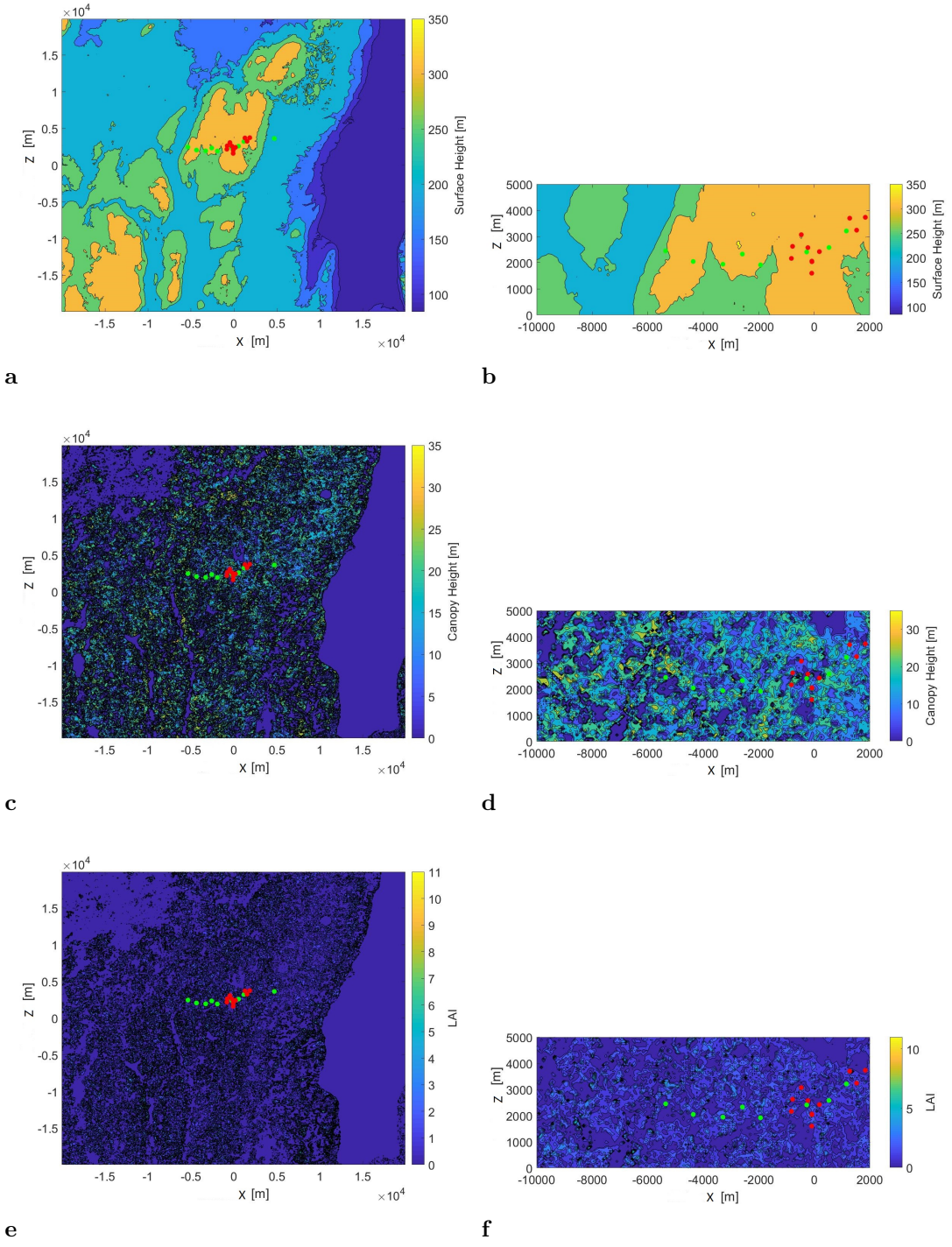


Figure 4.17: The landscape elevation of the Hornamossen site. The red spots indicate the locations of the wind turbines, and the green spots indicate the validation points where numerical data are monitored. (a) The elevation in the whole domain; (b) the elevation zoomed in to the region near the turbines and validation points; (c) the canopy height in the whole domain; (d) the canopy height zoomed in to the region near the turbines and validation points; (e) the leaf area index in the whole domain; (f) the leaf area index zoomed in to the region near the wind turbines and validation points. Data come from [5]

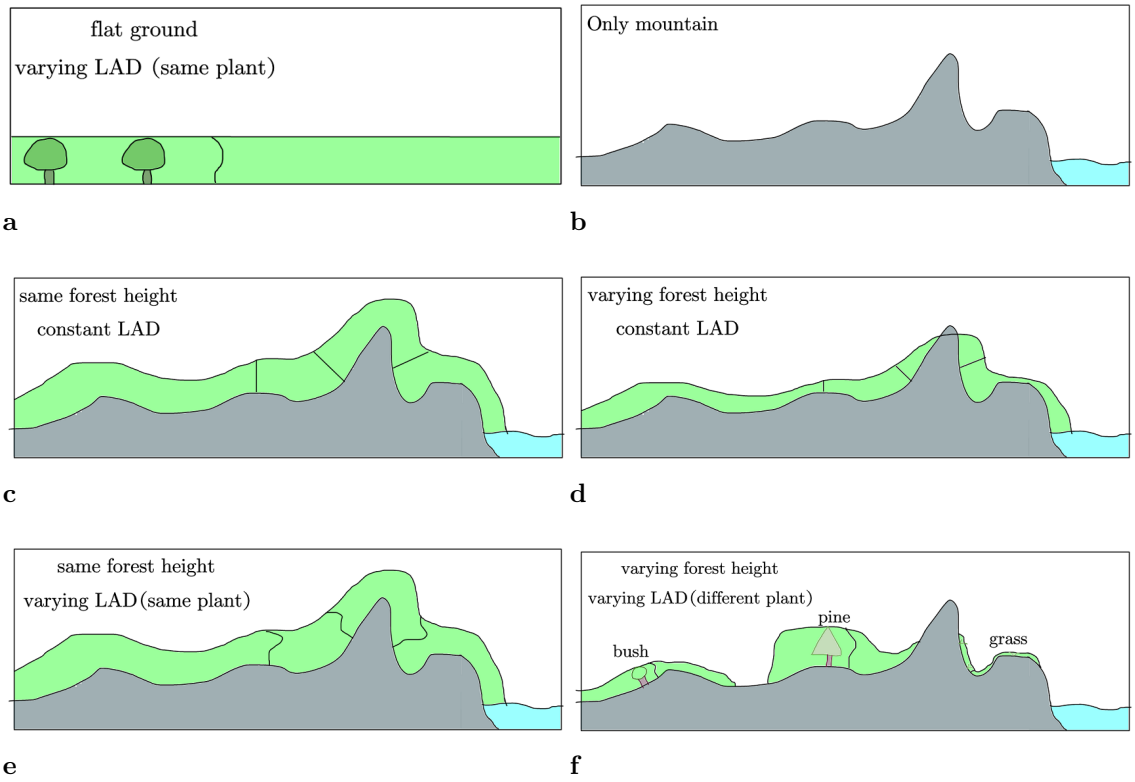


Figure 4.18: (a) The Homogeneous forest with varying LAD; (b) only mountain without forest cover; (c) the mountain with same forest height everywhere and the forest has constant LAD over height; (d) the mountain with different forest height in different locations, but the LAD of the forest is the same constant everywhere; (e) the mountain with the same forest height everywhere, but LAD is changing over the forest height; (f) the real situation in nature, where the forest height varies and even diminishes in some places, and where different types of vegetation exist with varying LAD along the heights. The green area indicates the forest cover, the grey area stands for the mountain, and blue area is the water region.

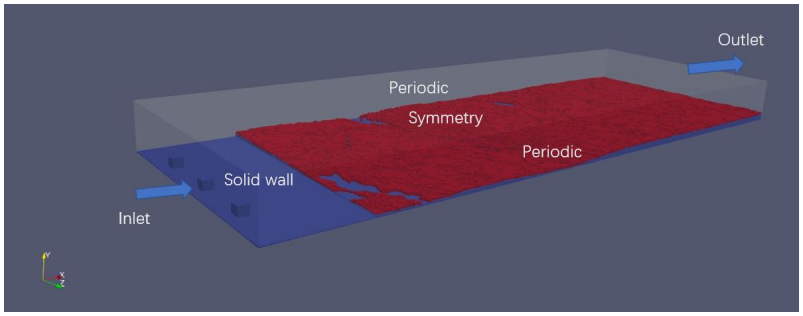
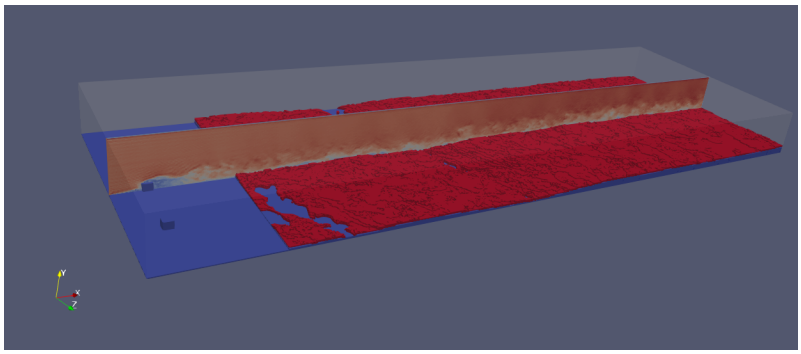
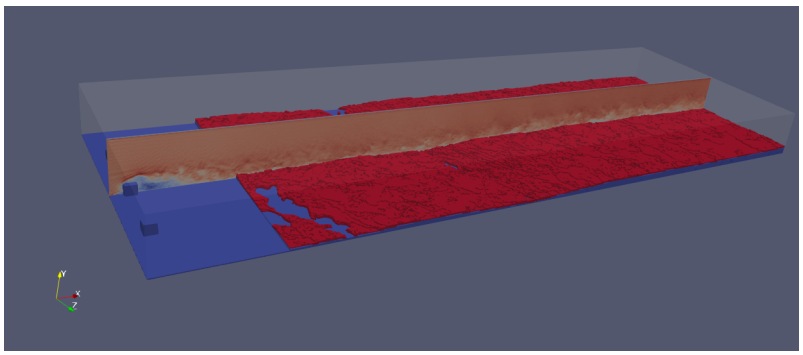


Figure 4.19: Computational domain of Hornamossen. Here the cubes are shown as an example, as their size and location are adjusted in the practice.



a



b

Figure 4.20: The effects of cubes in terms of the cube size and position.

Table 4.4: Validation points location

Site	N	E	N (sweref99TM)	E (sweref99TM)	if covered
Lidar 1	57.984	13.853	6427452	432146	Yes
Sodar 1	57.980	13.869	6427046	433138	Yes
Sodar 2	57.979	13.8884	6426942	434208	Yes
Sodar 3	57.983	13.899	6427329	434908	Yes
Lidar 2	57.979	13.910	6426917	435562	Yes
Sodar 4	57.984	13.938	6427410	437224	Yes
Sodar 5	57.986	13.952	6427581	438028	Yes
Sodar 6	57.991	13.962	6428213	438655	Yes
Sodar 7	57.995	14.022	6428626	442194	No
Tower	57.981	13.942	6427059	437410	Yes

Table 4.5: Turbines location

Site	N (sweref99TM)	E (sweref99TM)	if covered
HRM1034	6428071	437030	Yes
HRM1047	6427576	437267	Yes
HRM1083	6428243	439023	Yes
HRM1105	6428735	439337	Yes
HRM1015	6427164	436674	Yes
HRM1027	6427625	436716	Yes
HRM1057	6427423	437682	Yes
HRM1066	6427038	437413	Yes
HRM1075	6426598	437406	Yes
HRM1097	6428700	438774	Yes

4.3.2 Results

At the validation points, several sensors are installed to gather information such as wind speed and wind direction. In order to compare the simulation results with the measurement results, the data at the validation points are extracted from the computational domain. As shown in Figure 4.21, the velocity profiles at different validation points are plotted. At the region near the upper boundary, there is an abnormal increase in every profile, which may result from that the flow field has not reached a steady state yet. This assumption is confirmed by the time history of the velocity at validation points, as shown in Figure 4.22. The velocity at the validation points of two different heights (Y=200 m and Y=800 m) are extracted every 1×10^5 time steps corresponding to physical time 1×10^4 s. If the flow field has reached a steady state, the velocity should be oscillating in a small range. However, as shown in Figures 4.22a and 4.22b, the overall trend of the velocity history increases with time. It means that the flow field is still developing. It is also noticed that although the field has not achieved a steady state, the highest streamwise velocity at Y=800 m is already above 12 m/s, indicating that the constant body force exerted on the field is too large. Due to limited time and computational resources, the exact body force leading to the geostrophic wind speed of 11.7 m/s has not been found yet. But the methodology of determining this force is straightforward: first apply a constant body force (lower than 10^{-4} N from previous results), then run the simulation long enough and check the time history of the velocity at validation points to see if a generally steady mean velocity superimposed with fluctuations can be achieved.

4.3.3 Influence of boundary conditions

In the last case, it is noted that the value of the constant body force needs to be carefully determined to obtain the desired wind speed. Boundary conditions are another important factor that also influences the flow field. In this section, different boundary conditions are applied to inlet and outlet. Figure 4.23 shows the two sets of boundary conditions. The first one, as show in Figure 4.23a, uses the periodic boundary condition at the inlet and outlet, while the other boundary conditions are

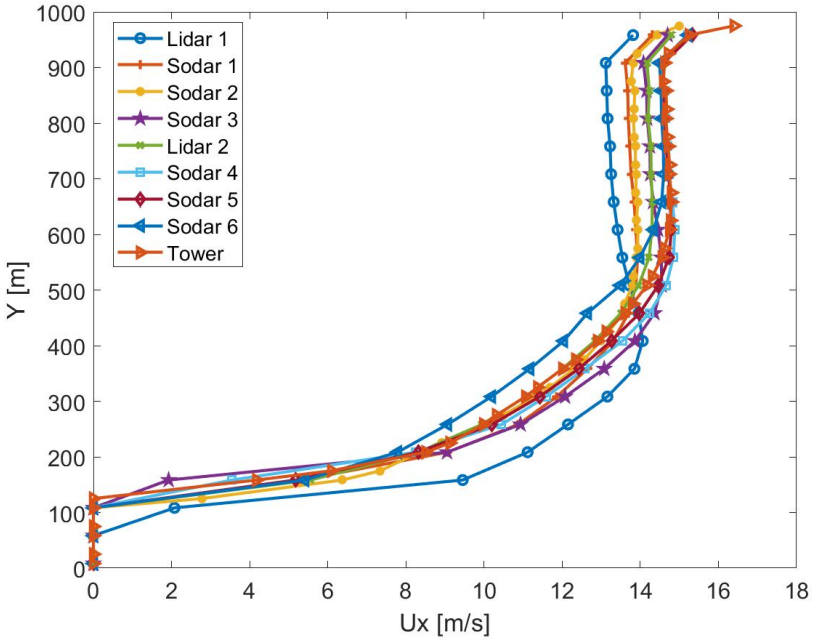
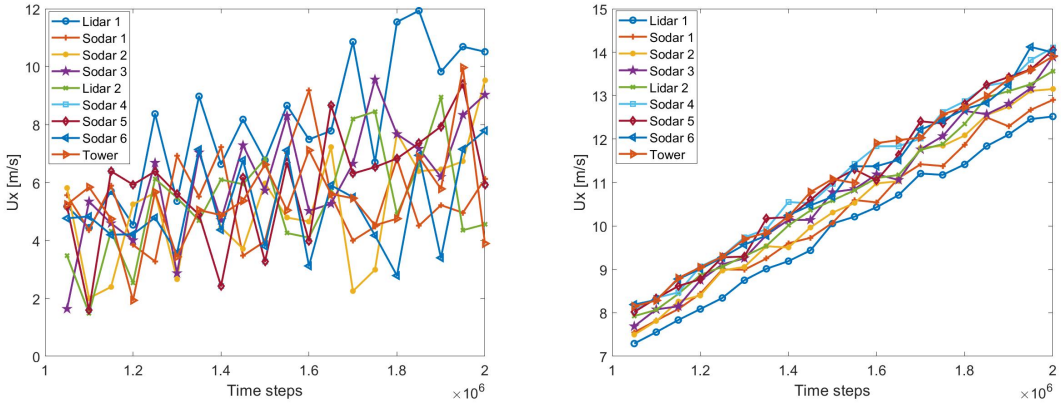


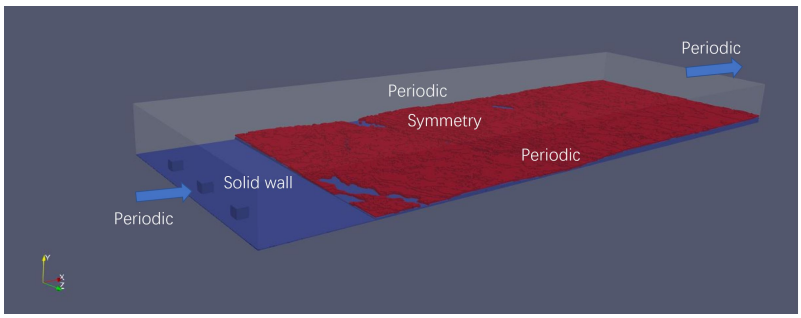
Figure 4.21: Velocity profiles of U_x at validation points



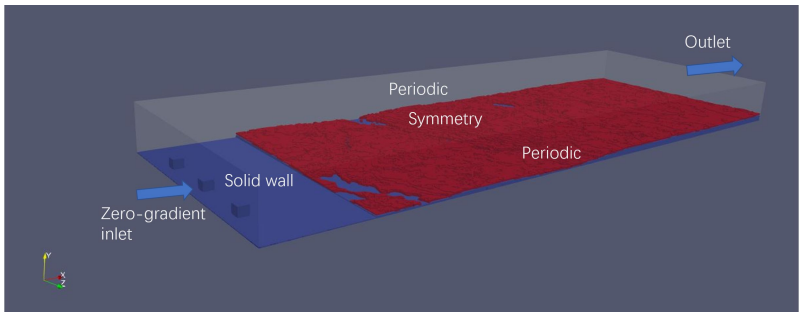
a **b**

Figure 4.22: The time history of the velocity at the validation points (a) inside boundary layer at Y=200 m, and (b) outside boundary layer at Y=800 m

the same as in Figure 4.19. Similarly, the second simulation employs a boundary condition of zero velocity and density gradient at inlet and outlet. The initial flow field for the two simulations with the new boundary conditions comes from the previous simulation shown in Figure 4.19. Therefore, the only difference among the simulations is in the boundary conditions.
Figures 4.24 and 4.25 are the preliminary results of the whole flow field. It needs to be mentioned that the velocity shown in these figures is in lattice unit. In order to transfer the lattice velocity to the physical velocity, a transfer factor of $dx/dt = 166.7$ should be multiplied. For the case with periodic inlet and outlet, it appears that after 1 million more time steps, the maximum value of the velocity still keeps changing. For the case with the zero-gradient inlet and outlet, the inlet velocity is fixed at about 10 m/s. However, after 6×10^5 time steps, the simulation diverges.

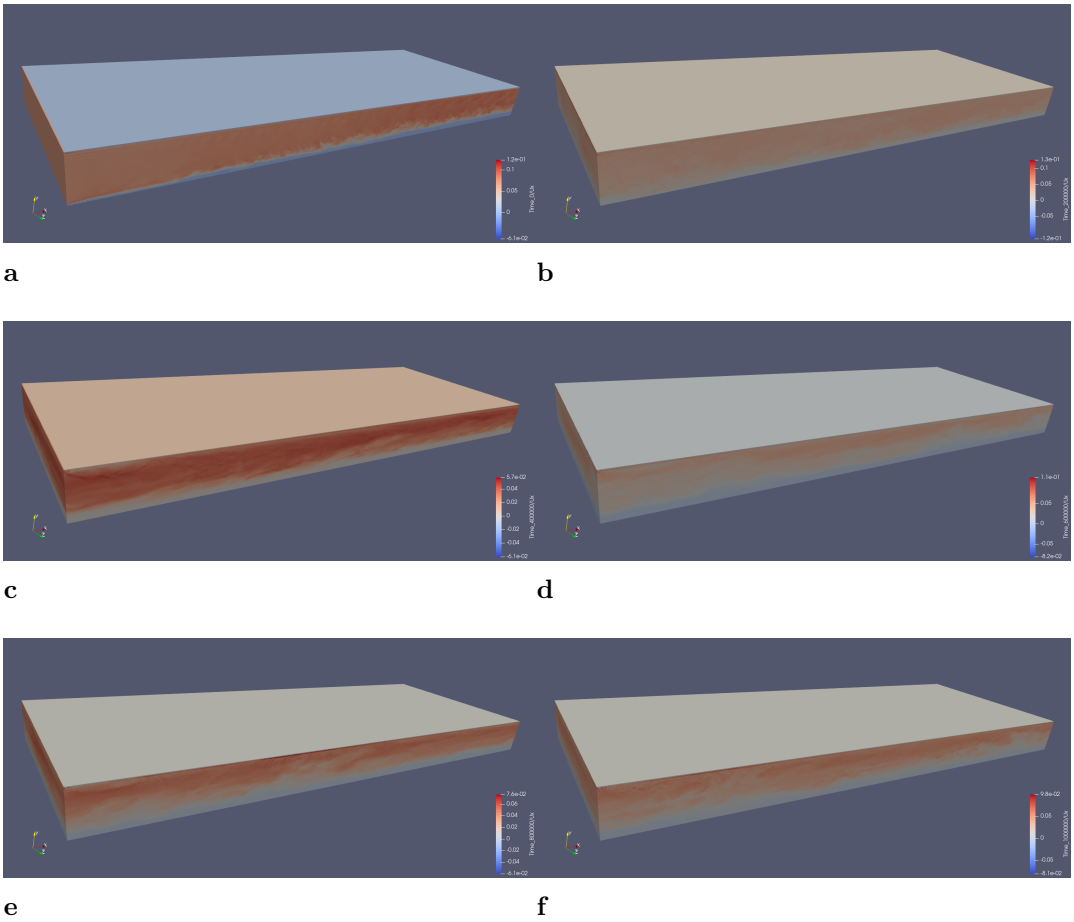


a



b

Figure 4.23: Other boundary conditions tested for the Hornamossen case (the size and location of the cubes may change): (a) both inlet and outlet are periodic; (b) only inlet is changed from the velocity inlet to the zero-gradient boundary condition, and other boundary conditions are the same as those in Figure 4.19.

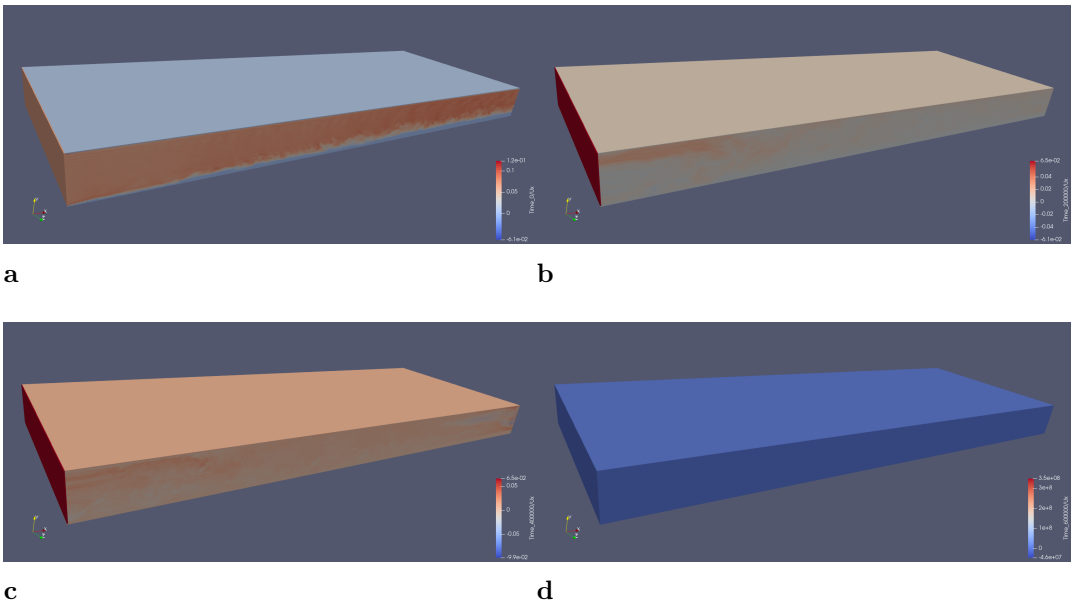


a **b**

c **d**

e **f**

Figure 4.24: Periodic inlet and outlet



a **b**

c **d**

Figure 4.25: Zero-gradient inlet and outlet

5
Conclusions

In this thesis work, the performance of GASCANS, a lattice Boltzmann computational fluid dynamics solver programmed for GPU architectures using C++ and CUDA, has been tested in three cases: turbulent channel flow, an atmospheric boundary layer with a hill, and an atmospheric boundary layer with forest cover. The simulation results of these cases have been compared with the experimental data and show good agreement. These simulations confirm that GASCANS can be applied to a wide range of wind-power engineering problems. In addition, a complicated case named Hornamossen with complex terrain and forest cover was also tested. However, some difficulties were encountered in the determination of numerical parameters such as a constant body force driving the flow, boundary conditions, and methods of generating or triggering turbulence at the inlet.
There are many open questions that have been noted in the course of this thesis work. They can be explored in the future.

- The effect of the Coriolis force. In this project, the Coriolis force is ignored, and the flow is driven by a constant body force. This makes it very difficult to control the flow velocity. If the Coriolis force is taken into account, a certain value of the geostrophic wind speed can be set, then the Coriolis force can drive the flow and finally reach a steady state.
- The varying LAD over the forest height. GASCANS can now only deal with varying LAD in flat terrain. In order to make the simulation setup better represent the real situation, it is worthwhile to develop a code module that accounts for the effect of varying LAD.
- Multiple GPUs. In this thesis work, only one GPU is employed. For large scale simulations, however, parallel computing with multiple GPUs is necessary. A future work will be to testify the performance of GASCANS using multiple GPUs.

Overall, the present study provides a comprehensive evaluation of GASCANS and paves the way for the future development of this LBM code.

Bibliography

[1] Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlend Magnus Viggen. The lattice boltzmann method. *Springer International Publishing*, 10(978-3):4–15, 2017.

[2] N. N. Anika, L. Djenidi, and S. Tardu. Roughness effect in an initially laminar channel flow. *Journal of Fluid Mechanics*, 892, 2020.

[3] Hans Bergström, Henrik Alfredsson, Johan Arnqvist, Ingemar Carlén, Ebba Dellwik, Jens Fransson, Hans Ganander, Matthias Mohr, Antonio Segalini, and Stefan Söderberg. Wind power in forests: wind and effects on loads, 2013.

[4] Bastian Nebenführ and Lars Davidson. Influence of a forest canopy on the neutral atmospheric boudary layer-a les study. In *ETMM10: 10th International ERCOFTAC Symposium on Turbulence Modelling and Measurements*, 2014.

[5] Johan Arnqvist. The hornamossen diurnal cycle benchmark for abl models in forested and moderately complex terrain. `https://thewindvaneblog.com/the-hornamossen-diurnal-cycle-benchmark-for-abl-models-in-forested-and-moderat`

[6] DTU Wind Energy. Wasp. `https://www.wasp.dk/`.

[7] O Reynolds. On the dynamical theory of incompressible viscous fluids and the determination of the criterion, cambridge phil. *Trans*, pages 123–164, 1894.

[8] Joseph Smagorinsky. General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather review*, 91(3):99–164, 1963.

[9] Wei Li, Xiaoming Wei, and Arie Kaufman. Implementing lattice boltzmann computation on graphics hardware. *The Visual Computer*, 19(7):444–456, 2003.

[10] Pablo R Rinaldi, EA Dari, Marcelo J Vénere, and Alejandro Clausse. A lattice-boltzmann solver for 3d fluid simulation on gpu. *Simulation Modelling Practice and Theory*, 25:163–171, 2012.

[11] Joshua A Anderson, Chris D Lorenz, and Alex Travesset. General purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of computational physics*, 227(10):5342–5359, 2008.

[12] Adrian RG Harwood, Joseph O'Connor, Jonathan Sanchez Munoz, Marta Camps Santasmasas, and Alistair J Revell. Luma: A many-core, fluid–structure interaction solver based on the lattice-boltzmann method. *SoftwareX*, 7:88–94, 2018.

[13] Xiaoyi He, Shiyi Chen, and Gary D Doolen. A novel thermal model for the lattice boltzmann method in incompressible limit. *Journal of computational physics*, 146(1):282–300, 1998.

[14] James D Sterling and Shiyi Chen. Stability analysis of lattice boltzmann methods. *Journal of Computational Physics*, 123(1):196–206, 1996.

[15] DN Siebert, LA Hegele Jr, and PC Philippi. Lattice boltzmann equation linear stability analysis: Thermal and athermal models. *Physical Review E*, 77(2):026707, 2008.

[16] XD Niu, C Shu, YT Chew, and TG Wang. Investigation of stability and hydrodynamics of different lattice boltzmann models. *Journal of statistical physics*, 117(3):665–680, 2004.

[17] Irina Ginzburg. Advantages and disadvantages of the two-relaxation-times lattice boltzmann scheme. *Fluid-Structure Interactions in Soft-Matter Systems: From the Mesoscale to the Macroscale*, page 43, 2012.

[18] Dominique d'Humieres. Multiple–relaxation–time lattice boltzmann models in three dimensions. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 360(1792):437–451, 2002.

[19] Roger H Shaw and Ulrich Schumann. Large-eddy simulation of turbulent flow above and within a forest. *Boundary-Layer Meteorology*, 61(1):47–64, 1992.

[20] Bastian Nebenführ and Lars Davidson. Large-eddy simulation study of thermally stratified canopy flow. *Boundary-layer meteorology*, 156(2):253–276, 2015.

[21] Sylvain Dupont and Yves Brunet. Influence of foliar density profile on canopy flow: a large-eddy simulation study. *Agricultural and forest meteorology*, 148(6-7):976–990, 2008.

[22] Branislava Lalic and Dragutin T Mihailovic. An empirical relation describing leaf-area density inside the forest for environmental modeling. *Journal of Applied Meteorology*, 43(4):641–645, 2004.

[23] Nvidia. Cuda zone. *https://developer.nvidia.com/cuda-zone*, 2020.

[24] A. R. G. Harwood M. Camps Santasmasas. Gascans: Gpu-accelerated solver for coupled approaches to navier-stokes (in progress). *Software X*, 2021.

[25] M. Camps Santasmasas. About ganscans. *https://github.com/cfdemons/GASCANS/wiki/About*, 2020.

[26] L Di Mare, M Klein, WP Jones, and J Janicka. Synthetic turbulence inflow conditions for large-eddy simulation. *Physics of Fluids*, 18(2):025107, 2006.

[27] Robert D. Moser, John Kim, and Nagi N. Mansour. Direct numerical simulation of turbulent channel flow up to $re_\tau = 590$. *Physics of Fluids*, 11(4):943–945, 1999.

[28] Václav Kolář. Vortex identification: New requirements and limitations. *International Journal of Heat and Fluid Flow*, 28(4):638–652, 2007.

[29] Yusuke Koda and Fue-Sang Lien. The lattice boltzmann method implemented on the gpu to simulate the turbulent flow over a square cylinder confined in a channel. *Flow, Turbulence and Combustion*, 94(3):495–512, 2014.

[30] Susumu Oikawa Takeshi Ishihara, Kazuki Hibi. A wind tunnel study of turbulent flow over a three-dimensional steep hill. *Journal of Wind Engineering and Industrial Aerodynamics*, 83:95–107, 1999.

[31] Kazuki Hibi Takeshi Ishihara. Numerical study of turbulent wake flow behind a three dimensional steep hill. *Wind and Structures*, 5:317–328, 2002.

[32] Alain Schubiger, Sarah Barber, and Henrik Nordborg. Evaluation of the lattice boltzmann method for wind modelling in complex terrain. *Wind Energy Science*, 5(4):1507–1519, 2020.

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY